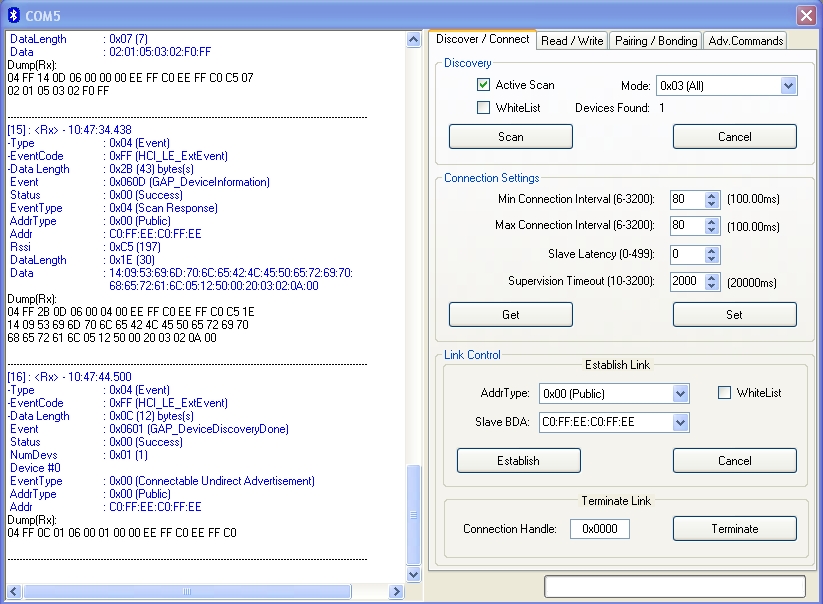
|  |
| --- |
| Wayne State University |
| CC2540 Mini Development Kit |
| Weekly Report |

|  |
| --- |
| Dr. Amar S Basu  Bomi Shim(fa5913) |

**1st week (5/28-06/02)**

Success to connect dongle to BTool. Understand how to work BTool.



First, push the right button on keyfob. And open the BTool, and click the scan -> Get -> Set -> Establish.

Get load current connection settings in dongle. Set save the change of connection settings in dongle.

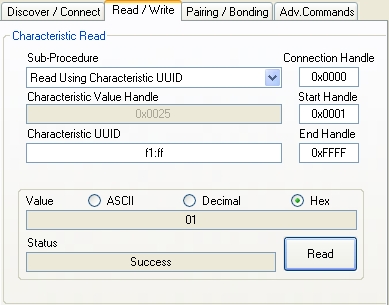
Sometimes dongle cannot read Slave BDA, but to use Read/Write fuction, this is very important. Until BTool receive the Slave BDA, keep trying to connect.

Follow CC2540 Mini Development Kit User's Guide (Rev.B).

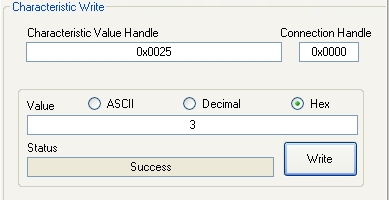
gatt.jpg

There are two way to read a value. Characteristic value handle and UUID. If you want to use value handle, just write hex number in this case 2800. But if you want to use UUID, order is changed. You should write 00:28. Both cases should have GATT read permission.

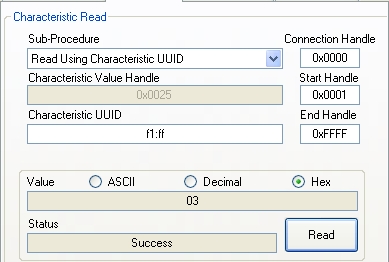
Also writing a value has a two way to do that. Using hex number and UUID. It should have GATT write permission.



If you success read 0xFFF1, then number, 0x0025, will show in characteristic value handle box. To use this number, you can write on 0x0025.



I write 3, Hex on 0x0025. You can write any number as long as maximum byte allow.



If you read f1:ff again, you can see the change of value from 01 to 03.

**2nd week (6/4-6/9)**

Download IAR Embedded Workbench. The CD from Basu is out of date. So I downloaded from website with 30-days license key. To understand simpleBLEPeripheral.c code, I read HAL\_Driver\_API.pdf.

Read BLE sample Application Guide. I read bloodpressure.c file in blood pressure sensor part.

If(events & BP\_START\_DEVICE\_EVT) *//Start the device*

{

void GAPROLE\_StartDevice (&bloodpressure\_peripheralCBS);  *//GAP function set up*

GAPBondMgr\_Register((gapbondCBS\_t\*)&timeAppBondCB);

updateUI();

return (event^BP\_START\_DEVICE\_EVT);

}

If(events & BP\_START\_DISCOVERY\_EVT) *//start discovery search for time service on collector*

{

If(timeAppPairingStarted)

{

timeAppDiscPostponed = TRUE; *//postponed discovery until pairing complete*

}

else

{

timeAppDiscState = timeAppDiscStart();

}

Return(events^BP\_START\_DISCOVERY\_EVT);

}

If(events&TIMER\_BPMEAS\_EVT) //perform final measurement

{

bpFianlMeas();

return (events^TIMER\_BPMEAS\_EVT);

}

If(events & BP\_TIMER\_CUFF\_EVT)

{

cuffMeas();

cuffCount++;

if(cuffCount < CUFF\_MAX) *//Keep checking the pressure until cuffCount<CUFF\_MAX.*

{

osal\_start\_timerEx(bloodpressureTaskId, BP\_TIMER\_CUFF\_EVT, TIMER\_CUFF\_PERIOD);

}

else

{

osal\_start\_timerEx(bloodpressureTaskID, TIMER\_BPMEAS\_EVT, TIMER\_CUFF\_PERIOD);

}

return(events^BP\_TIMER\_CUFF\_EVT);

}

**3rd week (6/11-6/16)**

Suddenly USB dongle connection has a problem. Reason was this. When I connect to dongle to BLE, Serial Port Settings is different from dongle’s property. It should be same in every detail.

BTOOL: handle usb dongle. (Slave)

IAR: handle keyfob. (Master)

|  |  |  |  |
| --- | --- | --- | --- |
| number | Byte | Read | Write |
| 1 | 1 | O | O |
| 2 | 1 | O | X |
| 3 | 1 | X | O |
| 4 | 1 | X | X |
| 5 | 5 | O | X |

<SIMPLE PROFILE CHAR>

Using SmartRF packet Sniffer program

To do so, first of all, using Flash Programmer to download sniffer\_fw\_cc2540\_usb.hex into dongle. This hexfile is located in C:/Program Files/Texas Instruments/SmartRF Tools/Packet Sniffer/bin/general/firmware. Don’t be confused between sniffer\_fw\_cc2540.hex and sniffer\_fw\_cc2540\_usb.hex

If the download is success, usb dongle’s LED change from red to green.

Open the packet sniffer program and press F5. If you push right button on keyfob, it start to work.



<Packet Sniffer>

To use this program, we can understand information transfer between dongle and keyfob. You can see RSSI value keeps changing. RSSI means received signal strength indicator, that is why it keeps changing.

To understand how keyfob works, using LED function. I modified simpleBLEPeripheral project when I push left button, the LED become green and right button, the LED become off. Using LED function is easiest way to understand keyfob. To do so, put below code HandleKey part in simpleBLEPeripheral.c file.

if ( keys & HAL\_KEY\_SW\_1 )

{

SK\_Keys |= SK\_KEY\_LEFT;

void HalLedInit (void);

HalLedSet (HAL\_LED\_1, HAL\_LED\_MODE\_ON);

}

if ( keys & HAL\_KEY\_SW\_2 )

{

SK\_Keys |= SK\_KEY\_RIGHT;

void HalLedInit (void);

HalLedSet (HAL\_LED\_ALL, HAL\_LED\_MODE\_OFF);

}

**4th week (06/18-06/23)**

Design for Op-Amp Voltage Detector

1. Whenever chip detect voltage, turn on the green LED(LED1)
2. If voltage is higher than 3V, turn on the red LED(LED2)
3. Voltage should be saved in port 0.2, GATT table.
4. When push the button on USB dongle, voltage output should be updated and LED should be blinked.

To do 3 and 4, use GATTmsg and GATTcallbacks function.

CC2540 OP-Amp datasheet (p.173~174)

<http://www.ti.com/lit/ug/swru191c/swru191c.pdf>

To detect voltage, first use the battery check function in keyfobdemo application software.

<http://processors.wiki.ti.com/index.php/Category:KeyFobDemo>

1. First of all, using flash programmer, download CC2540MiniDKDemoSlave.hex on keyfob.
2. Open and connect BTool program.
3. Write 19:2A in characteristic UUID box.
4. Choose decimal on value.
5. Read the value. The percentage of battery amount returns the value. CR2032 battery is 3V. So if the value is 100, it means that there are 3V in the battery.

Result

|  |  |
| --- | --- |
| Model | Value(Percentage) |
| Panasonic | 62 |
| Duracell | 67 |

Read battery check code in KeyFobDemo project. Battservice.h (location: APP->keyfobdemo.c)

I attached below code in hal\_adc.c and PerformPeriodicTask().

Uint8 temp;

Temp=HalAdcRead(HAL\_ADC\_CHANNEL\_7, HAL\_ADC\_RESOLUTION\_8);

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &temp);

And also, to check is it working right or not, if they detect voltage LED2 will be light. I put those code in hal\_adc.c but the LED2 keep lighting even there is no voltage.

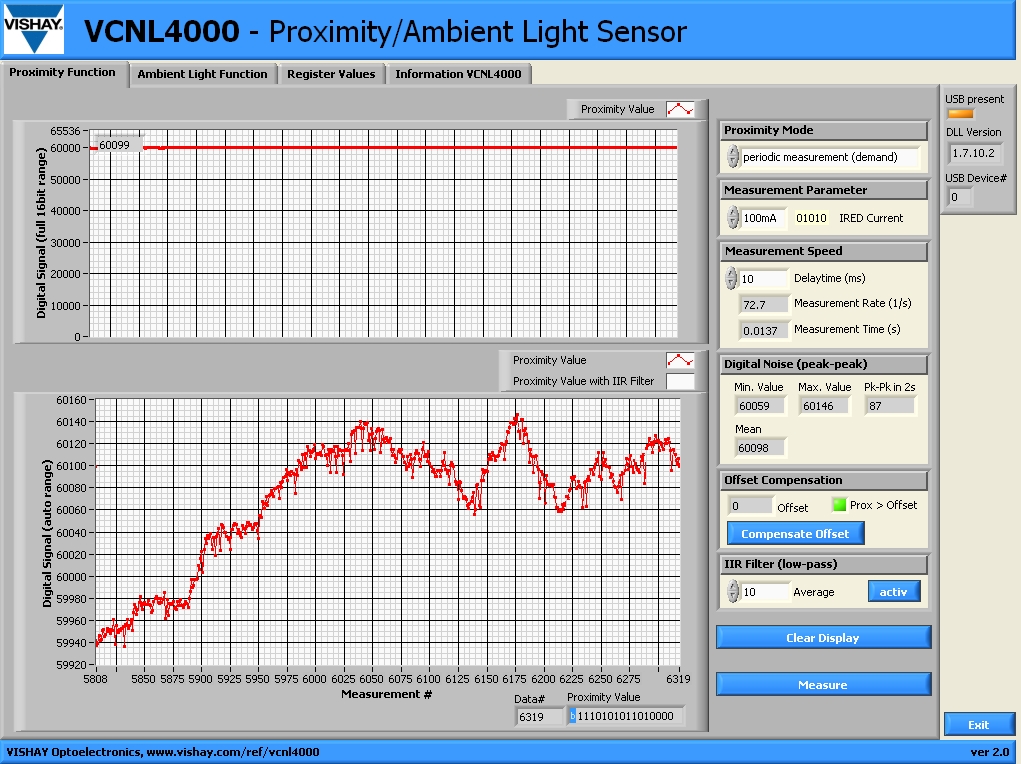


Whether keyfob detect voltage or not, BTool keep read same value 08 and 03.

**5th week (06/25-06/30)**

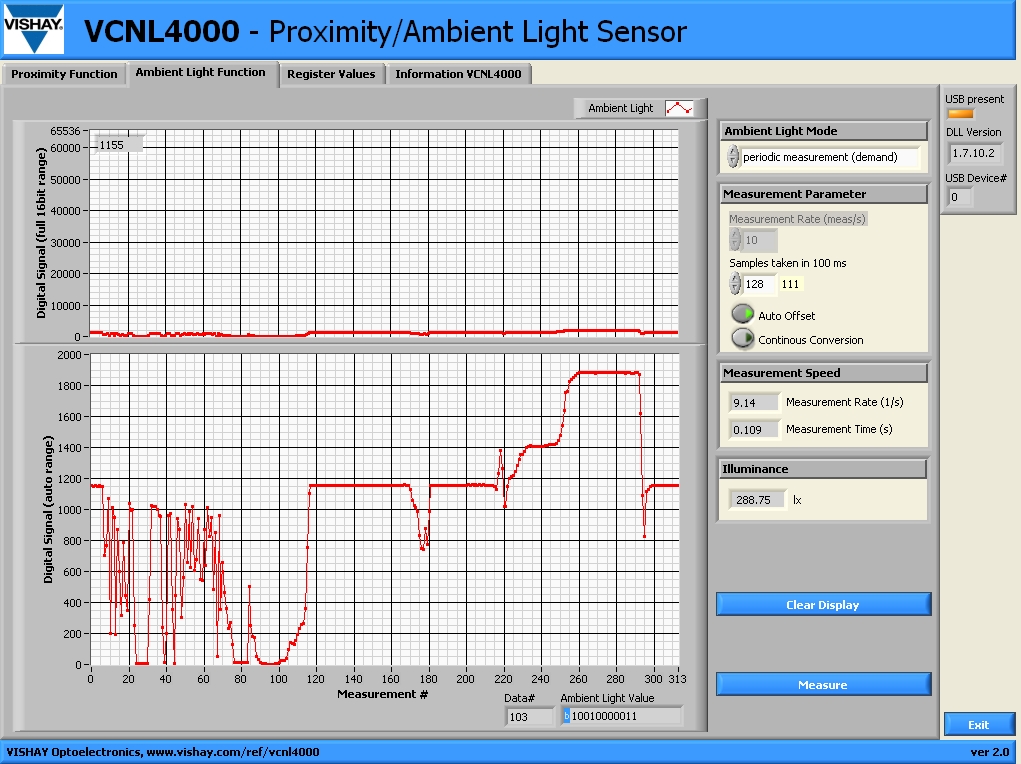
Receiving VCNL4000. USB and chip should be contacted right side(When you put VCNL4000 to CPU, then the chip should be up side.

VCNL4000 can detect two things, proximity and light.



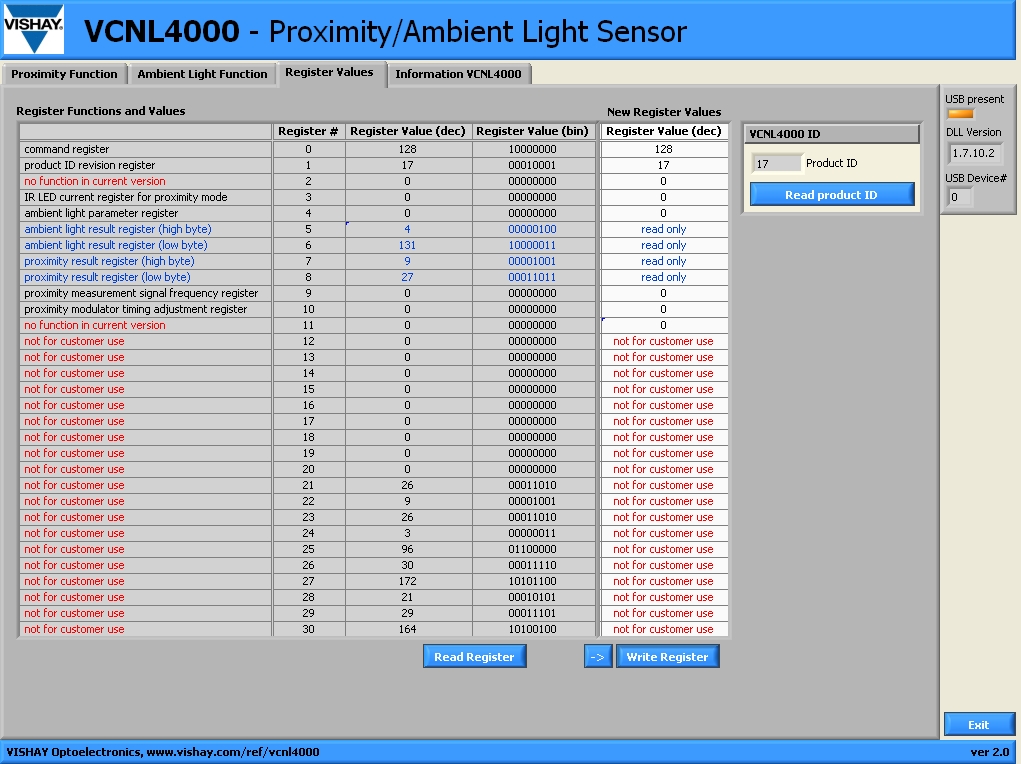
<Proximity Sensor>

Only close to 5-10mm stuff can be detected.



<Ambient Light Sensor>

The lighter, the bigger signal is detected.

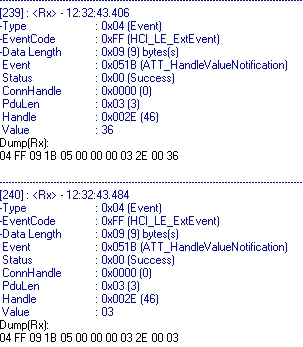
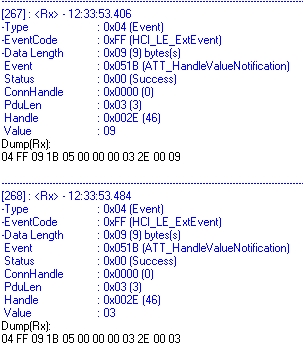


<Register Values>

It shows us address of data.

<CC2540>

I kept trying that when CC2540 detect voltage, LED2 is enlighten but didn’t success. Instead, I success detect voltage without LED2. Since CC2540’s Vdd is 3V so the minimum voltage should be more than 3V.



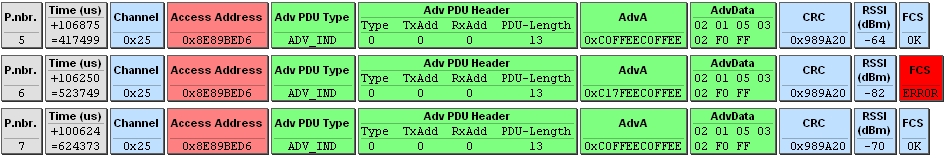
<Left : undetected, Right: when 3.65V is detected>

I experiment three cases depend on PIN number.

|  |  |  |  |
| --- | --- | --- | --- |
| V\Pin | 1, 2(GND) | (1&5), 2(GND) | 1, (2&5)(GND) |
| 3.0 | 09(9) | 09(9) | 00 |
| 3.5 | 09(9) | 09(9) | 00 |
| 3.6 | 0A(10) | 12(18) | 00 |
| 3.7 | 2A(42) | 67(103) | 00 |
| 3.8 | 7F(127) | 7F(127) | 00 |
| 3.9 | 7F(127) | 7F(127) | 00 |
| 4.0 | 7F(127) | 7F(127) | 00 |

<Value Hex numbers (Decimal) depend on Voltage/Used Pin>

The hex number of value below 3.0V is 09 and upper 3.8V is 7F.

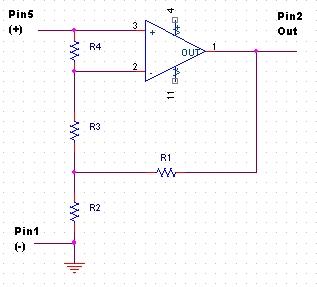
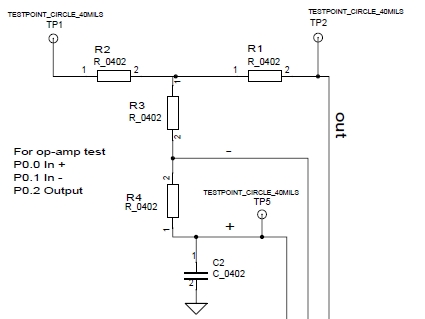


I want to observe when CC2540 detect voltage, how packet sniffer works. But there is no difference between undetected and detected. Only one time sniffer has FCS error. It returned to normal soon.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Voltage | Hex | Decimal | | 3.40 | 09 | 9 | | 3.50 | 09 | 9 | | 3.60 | 0A | 10 | | 3.63 | 0E | 14 | | 3.65 | 1A | 26 | | 3.68 | 23 | 35 | | 3.70 | 3C | 60 | | 3.73 | 53 | 83 | | 3.75 | 6E | 110 | | 3.78 | 7F | 127 | | 3.80 | 7F | 127 | |  |

<Left: Table for value/voltage, Right: Graph for decimal value/voltage (Used Pin: 1&2)>

Below 3.6V and upper 3.78V has the same value. Between 3.63-3.75V, the number of value is proportioned.

<CC2540 discrete schematic>

(+): Pin05 / (-): Pin01 / output: Pin02

From the schematic, this formula is deducted.

Before using BTool, to calculate amplitude rate(R1, R2) detect Pin02(output) by multi-meter.

|  |  |
| --- | --- |
| Input V | Output V |
| 0.0 | 0.00 |
| 0.5 | 0.52 |
| 1.0 | 1.10 |
| 1.5 | 1.52 |
| 2.0 | 2.01 |
| 2.5 | 2.52 |
| 3.0 | 3.03 |

We can draw a conclusion that amplitude rate is almost 1 therefore limR1=0.

Now I tried to detect output voltage via BTool, but the result seems like unsound. Result was before 3V, value was 09. And after 3V, value was 7F. But at 3V, it has different value, 25-29. And also, I found that when I connected CC2540 and power supply, the CC2540 didn’t turn on(I modified the code when I push left butten, LED1 is light on). So before I connected power supply, I started to open BTool program and make wireless connection. After that I attached power supply and voltage also. Maybe I used wrong pin, since if I use another pin, such as 1 for (+), 2 for (-), the LED1 is turned on.

**6th week (07/02-07/07)**

Try to use buzzer and accelerometer function in keyfob. These codes are located in keyfobdemo.c, keyfobDemo.eww.

1. Buzzer

If left button is pushed, LED1 and buzzer are working. If right button is pushed, LED1 and buzzer are stopped.

Put below code in to the if ( keys & HAL\_KEY\_SW\_1 ).

void buzzerInit(void);

buzzerStart( BUZZER\_ALERT\_LOW\_FREQ ); // *BUZZER\_ALERT\_HIGH\_FREQ is also available.*

void HalLedInit (void);

HalLedSet (HAL\_LED\_1, HAL\_LED\_MODE\_ON);

Put below code in to the if ( keys & HAL\_KEY\_SW\_2 ).

HalLedSet (HAL\_LED\_ALL, HAL\_LED\_MODE\_OFF);

buzzer\_state = BUZZER\_OFF;

buzzerStop(); //*if you write ‘****void*** *buzzerstop(****void****)’ it’s not working properly.*

You can change the frequency by modifying code.

// Buzzer beep tone frequency for "Low Alert" (in Hz)

#define BUZZER\_ALERT\_LOW\_FREQ 294

|  |  |
| --- | --- |
| Scale | Frequency |
| Do | 262 |
| Re | 294 |
| Mi | 330 |
| Fa | 349 |
| Sol | 392 |

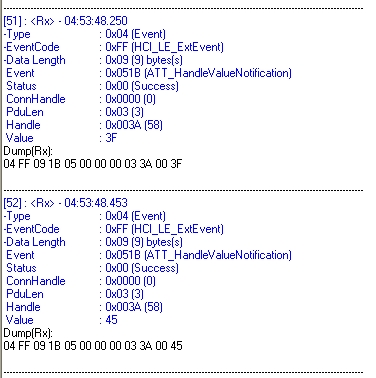
By using timer function, you can play song.

1. Accelerometer

Since we use keyfobDemo code, not simpleBLEPeripheral, we should check where the value is saved. To do so, check the keyfob Demo wiki site.

UUID address is followed.

|  |  |
| --- | --- |
| address | Description |
| 0XFFA1 | ACCEL\_ENABLER\_UUID |
| 0XFFA2 | ACCEL\_RANGE\_UUID |
| 0XFFA3 | ACCEL\_X\_UUID |
| 0XFFA4 | ACCEL\_Y\_UUID |
| 0XFFA5 | ACCEL\_Z\_UUID |



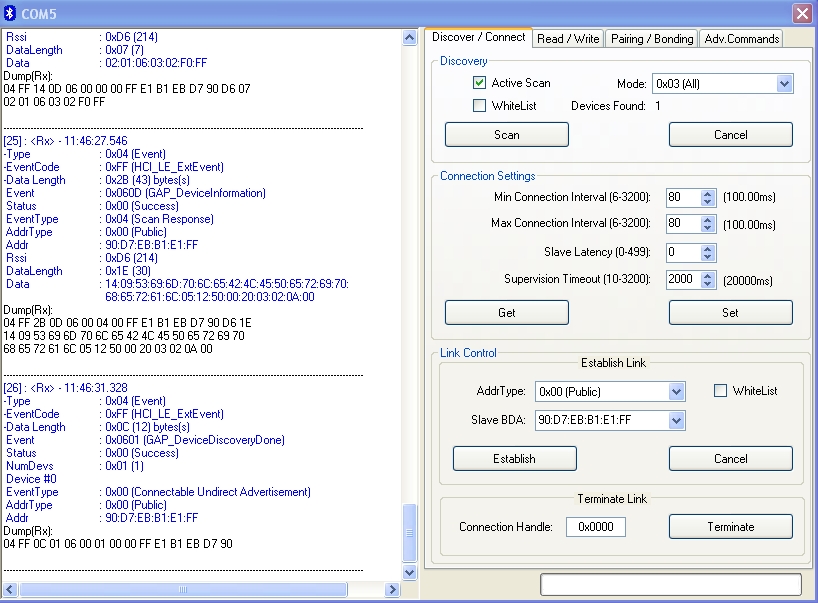
<Reading z value, 0XFFA5>

If you read z value, you can only detect z motion, not x and y.

CC2541 has been arrived. I debugged simpleBLEPeripheral code and simpleBLECentral code, but it was failed. I should figure out how to insert code from IAR workbench program.

**7th Week (07/09-07/14)**

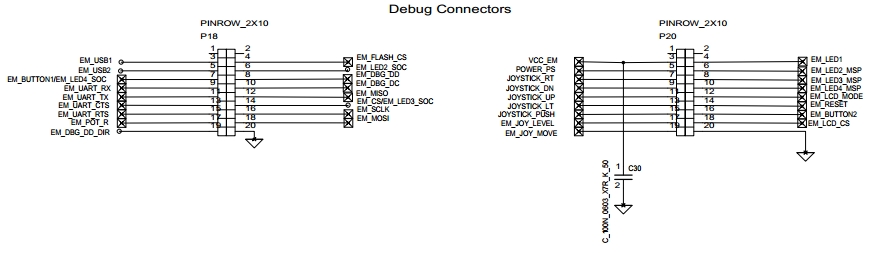
Have a problem to detect SmartRF05EB from CC Debugger. Solution was connecting USB, not debugger. Even if without debugger, USB can detect SmartRF05EB and also download code from IAR.



Connect with USB dongle in BTool program. The board which has SimpleBLEPeripheral code is detected in BLE program. Read/Write function is same as CC2540mini.



Using packet sniffer program. You can see RSSI values keep changing. This is because I change the position of antenna in CC2541 EM.

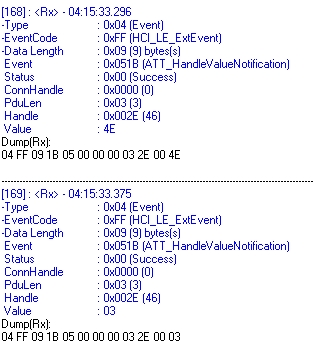
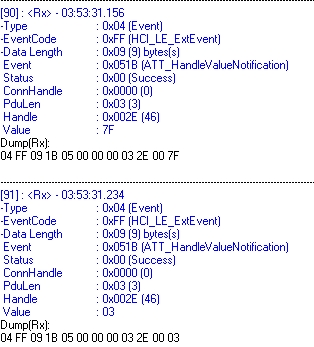


To connect VCNL4000 into SmartRF05EB, I should focus on schematic. I will connect on debug collector pin, p18 and p20.

There are good examples on youtube about SmartRF05EB pulse sensor.

<http://www.youtube.com/watch?v=RdnQpurrA-4>

<http://www.youtube.com/watch?v=5sS74MEzr2o>



<Left: normal mode / Right: pin17 in P18. Value is changed from 7F to 4E>

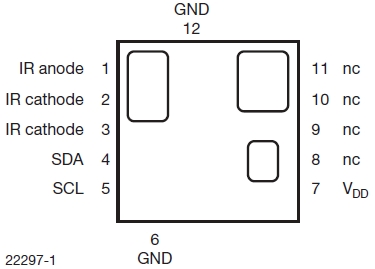
To figure out which pin is right to do a test, I put voltage from power supply. I use GND at pin20 in P20 and touch every other pin with 2.8V. However, only pin17 in P18 make value changed. Pin17 is EM\_POT\_R which is related to potmeter. So this time, I fix the (+) voltage to pin17 and turn potmeter clockwise and counter clockwise. But nothing happened.

I also do this with pin20 in P18, but the result was same.

**8th Week (07/16-07/21)**

I try to detect Vdd but it is 0V. Problem was p5&6 are not 1:1 matched to p18&20. For example, Vcc in p6 is pin7 and 9. But in p20, it is pin 3. So you should compare those two.

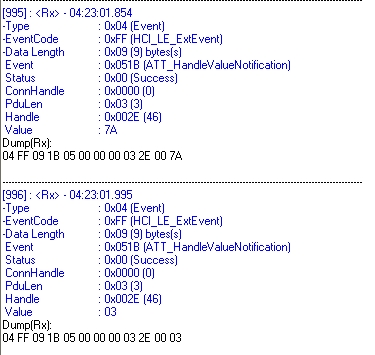
Vdd on board: 3.26V



<VCNL4000 datasheet>

|  |  |  |
| --- | --- | --- |
|  | VCNL4000 | SmartRF |
| SDA | 4 | 3(p18) |
| SCL | 5 | 5(p18) |
| GND | 6, 12 | 20(p18) |
| Vdd | 7 | 3(p20) |

<Pin match order>



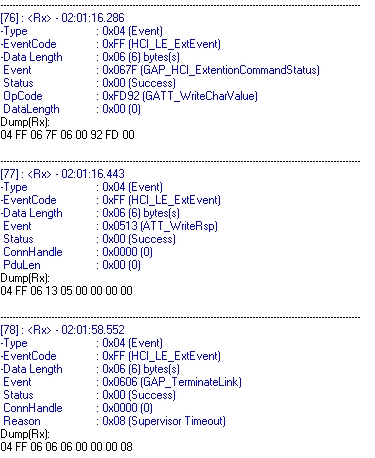
For simple test, using op-amp function in CC2541, I put voltage from power supply and detect the value of voltage by BTool software. But only at 4V, the value is changed, 7A. In other arrange of voltage, it was 7A and 03.

I connected correct pin from VCNL to board, and observed the change of value through BTool. But nothing changed. To check whether VCNL4000 is on or not, modify code when VCNL4000 is on then LED4 will on. The simpleBLEPeripheral code in IWR is basically for CC2540mini. So if you download this code into board, it’s not working. So try to make new code for CC2541.

**9th Week (07/23-07/28)**

To make code for VCNL4000, use the proximity code in CC2540 keyfob. To do so, you should check UUID number in h file. Here is the order.

1. Open BTool and establish connection with keyfob.
2. In read section, write 06:2A(alert level) in discover characteristic by UUID.
3. Check the Hex value. We will use second value. For example, if hex value is 0A 25 00 06 2A 04 28 00 06 2A, then we will use 25.
4. In write section, write 0x0025 at characteristic value handle and write 01(low level alert) or 02(high level alert).
5. Stay away from USB dongle with keyfob. If the proximity is far enough, keyfob buzzer and led start to act.



<Using proximity with keyfob. The connection is terminated.>

Use Bluetooth in iphone, try to connect to keyfob. To do so, you should download MiniDKDemoSlave.hex file to the keyfob, not simpleBLEPeripheral.hex. And also, down load TI app in iphone. This app is only able to detect keyfob. This app can detect battery level, accelerometer and make the keyfob buzzer.



<Smart RF studio, during connect with iphone>

If you start to connect with iphone, USB dongle cannot detect keyfob. The iphone’s Bluetooth is acting like simpleBLECentral.

<http://blog.bluetooth-smart.com/2012/01/16/cc2540dk-mini-received/>

<http://processors.wiki.ti.com/index.php/Category:BluetoothLE>

To use VCNL4000 in smartRF05EB, use I2C function. Unfortunately, CC2540, which is the most similar with CC2541, doesn’t have I2C function moreover, there is no I2C code in IAR workbench. So refer to the CC2533. Below webpage has the I2C code.

<http://e2e.ti.com/support/low_power_rf/f/538/t/172539.aspx>

You should add those files in IAR workbench code and write #include **“hal\_i2c.h”** at <simpleBLEPeripheral.c> .

**10th Week (07/30-08/04)**

-Vacation

**11th Week (08/06-08/11)**

Try to put CC2533 I2C header and complier file to the simpleBLEPeripheral project. But I faced linker error, [e46] external defined. It means that those files didn’t include properly. What I did was put those files into the right directories.

Header file location: <C:\Texas Instruments\BLE-CC254x-1.2.1\Components\hal\include>

Complier file location: <C:\Texas Instruments\BLE-CC254x-1.2.1\Components\hal\target\CC2540EB>

In <hal\_i2c.h> file, you can see this code.

#if HAL\_I2C\_MASTER

#define HAL\_I2C\_POLLED FALSE // Master does not use ISR and does not need periodic polling.

#else // if HAL\_I2C\_SLAVE

#if !defined HAL\_I2C\_POLLED

#define HAL\_I2C\_POLLED FALSE // Prefer the ISR as a Slave to speed reaction to Master READ req.

#endif

#endif

To make this code work, you should define two variables first.

#define HAL\_I2C TRUE

#define HAL\_I2C\_MASTER TRUE

I put those code in <simpleBLEPeripheral.c>, <hal\_i2c.h> and <ioCC2541.h>. You should make sure those codes are prior to the code, #if(define HAL\_I2C)&&(HAL\_I2C==TRUE) since C language is working sequential way. For example, <hal\_i2c.h> file has this code.

If you look at <hal\_i2c.h>, you can notice that how HAL I2C function works.

#if HAL\_I2C\_MASTER

**void HalI2CInit(uint8 address, i2cClock\_t clockRate);**

#else

void HalI2CInit(uint8 address, i2cCallback\_t i2cCallback);

#if HAL\_I2C\_POLLED

void HalI2CPoll(void);

#endif

#endif

**i2cLen\_t HalI2CRead(i2cLen\_t len, uint8 \*pBuf);**

**i2cLen\_t HalI2CWrite(i2cLen\_t len, uint8 \*pBuf);**

uint8 HalI2CReady2Sleep(void);

void HalI2CEnterSleep(void);

void HalI2CExitSleep(void);

#endif

#endif

Uint8 byte = 26x;

Uint8[] sendBuffer;

sendBuffer [0] = 26h;

sendBuffer[1] = 83h;

halI2CWrite(2,sendBuffer)

Uint8[] returnBuffer;

returnBuffer[0] = 27h;

halI2cRead(1,returnBuffer);

print (returnBuffer)

It’s very different from <HAL\_API.pdf> manual which is originally included in the file. Therefore, you should be careful and check <hal\_i2c.h> file when you use those function. In this project, we only use three functions, HalI2CInit, HalI2CRead and HalI2CWrite.

For example,

HalI2CInit(uint8 address, i2cClock\_t clockRate);

It should be

HalI2CInit(I2CADDR, i2cClock\_123KHZ);

uint8 address(I2CADDR) comes from <ioCC2541.h>, and i2cClock\_t clockRate(i2cClock\_123KHZ) comes from <hal\_i2c.h>.

**12th Week (08/13-08/18)**

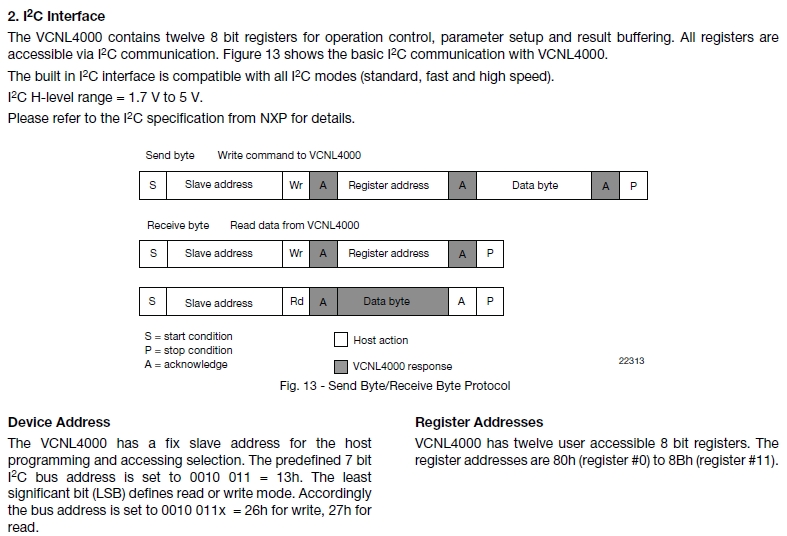
To check the code is working properly, use <printf> function with output window.

How to use <printf> function with output window.

1. Use <printf> function properly. Such as printf("%d\n", data);
2. Click the Make\_Restart\_Debugger
3. Go to View>Terminal I/O
4. Make this code Go

To make appropriate code, check <HAL Driver API.pdf> file. This document has different function from mine, but almost similar.

Here is the code in the <simpleBLEPeripheral.c> file, < performPeriodicTask> section. This code is based on it.



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

uint8 sendBuffer[8];

uint8 data;

sendBuffer[0] = 0x87; *//VCNL proximity register address.*

sendBuffer[1] = 0x80; *//Command Register.*

HalI2CWrite(1, sendBuffer);

HalI2CWrite(1, &sendBuffer[1]);

uint8 receiveBuffer[1]; *//Space to receive data from VCNL*

data=HalI2CRead(1, receiveBuffer);

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &data);

printf("%d\n", data);

printf("%d\n", SIMPLEPROFILE\_CHAR4);

printf("\*\*\*\*\*\*\n");

data=HalI2CRead(1, receiveBuffer);

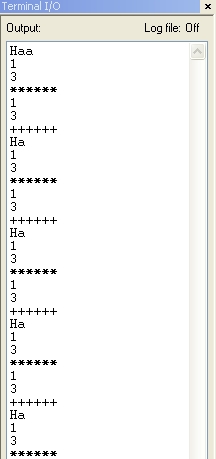
SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &data);

printf("%d\n", data);

printf("%d\n", SIMPLEPROFILE\_CHAR4);

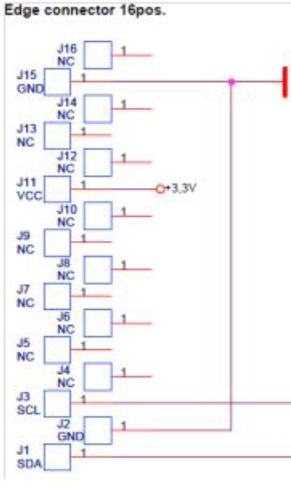
printf("++++++\n");

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



<Result>

Also you should check pin number of VCNL. Specific information is in the document, <VCNL Demokit >.



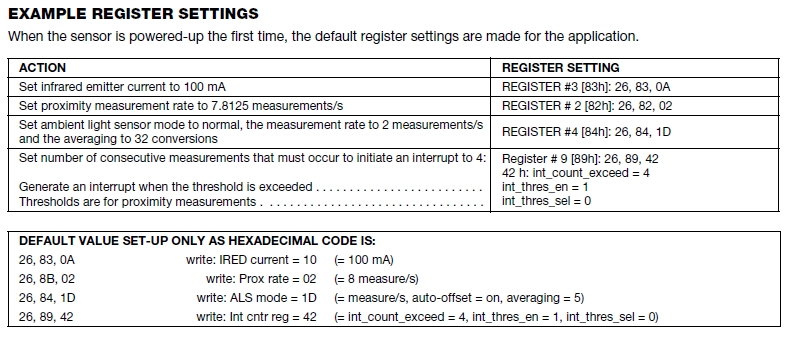
<Pin number>

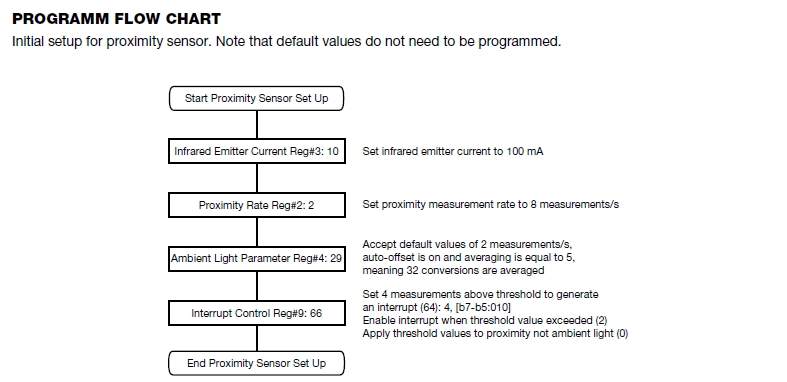
**13th Week (08/20-08/25)**

Success to get the return value 1, which means length of data byte. But still cannot read the specific data from VCNL4000.

One of the smartRF05EB has been broken. While I change the pin, LED suddenly turned off. Whenever I use USB power source(Pin11), it’s not working. However DC battery mode is working very well. The board which has simpleBLEPeripheral code has to be debugged a lot, so the broken one should have simpleCentral code since broken board cannot be connected with USB.

Check VCNL4010 application note. It has more information about I2C code. You should be careful Reg#82 in VCNL4010 is Reg#89 in VCNL4000.





<VCNL4010 application note>

**14th Week (08/27-09/01)**

Success to read the data of proximity in VCNL4000.

<Code>

uint8 sendBuffer[100];

uint8 data;

uint8 receiveBuffer[10];

receiveBuffer[0] = 0x00;

receiveBuffer[1] = 0x00;

receiveBuffer[2] = 0x00;

sendBuffer[0] = 0x83; //LED current

sendBuffer[1] |= 0x0A;

sendBuffer[2] = 0x89; //measurement frequency

sendBuffer[3] |= 0x02;

sendBuffer[4] = 0x80; //Command

sendBuffer[5] |= 0x08;

sendBuffer[6] = 0x81;

sendBuffer[7] = 0x87;

HalI2CWrite(**2**, &sendBuffer[4]); //Command

HalI2CWrite(**2**, &sendBuffer[0]); //LED current

HalI2CWrite(**2**, &sendBuffer[2]); //measurement frequency

HalI2CWrite(1, &sendBuffer[6]);

data=HalI2CRead(1, &receiveBuffer[0]);

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &data);

printf("%d\n", receiveBuffer[0]);

printf("\*\*\*\*\*\*\n");

HalI2CWrite(1, &sendBuffer[7]);

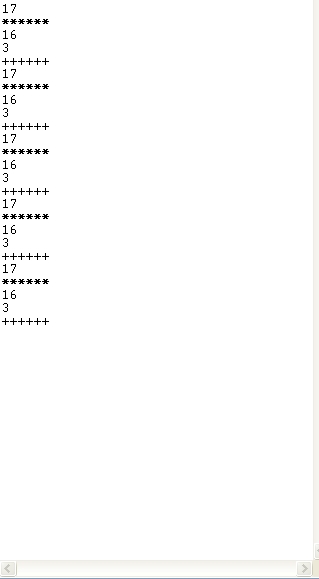
data=HalI2CRead(**2**, &receiveBuffer[1]);

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint16), &data);

printf("%d\n", receiveBuffer[1]);

printf("%d\n", receiveBuffer[2]);

printf("++++++\n");



<Result>

17: Data from register 0x81(Product ID revision).

16: Data from register 0x87(Proximity result, High byte)

3: Data from register 0x88(Proximity result, low byte)

Data from register 0x87 and 0x88 keep changing whenever I download and debug the code. Main problem is those value is not changed during debugging mode. To solve this problem, check accelerometer function in keyfob code.

**15th Week (09/03-09/08)**

Success to keep changing the proximity data from VCNL4000. In the very last of the code, I put one sentence.

HalI2CWrite(**2**, &sendBuffer[4]); //Command

I use write function again. And then, it’s working.

Here is the final code.

uint8 sendBuffer[100];

uint8 data;

uint8 receiveBuffer[10];

receiveBuffer[0] = 0x00;

receiveBuffer[1] = 0x00;

receiveBuffer[2] = 0x00;

sendBuffer[0] = 0x83; //LED current

sendBuffer[1] = 0x0A;

sendBuffer[2] = 0x89; //measurement frequency

sendBuffer[3] = 0x02;

sendBuffer[4] = 0x80; //Command

sendBuffer[5] = 0x08;

sendBuffer[6] = 0x81;

sendBuffer[7] = 0x87;

HalI2CWrite(2, &sendBuffer[4]);

HalI2CWrite(2, &sendBuffer[0]);

HalI2CWrite(2, &sendBuffer[2]);

HalI2CWrite(1, &sendBuffer[7]);

data=HalI2CRead(2, &receiveBuffer[1]);

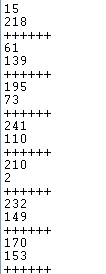
SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint16), &data);

printf("%d\n", receiveBuffer[1]);

printf("%d\n", receiveBuffer[2]);

printf("++++++\n");

HalI2CWrite(2, &sendBuffer[4]);



<Result>

To get the hearth rate graph, I start to use BLE Health Demo program.

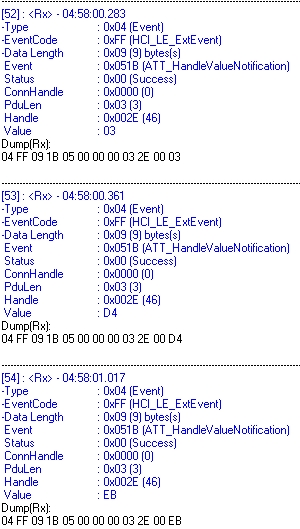
<http://processors.wiki.ti.com/index.php/Category:HealthDemo>

**16th Week (09/10-09/15)**

After I put two lines in the code, I success to receive value using BTool.

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &receiveBuffer[0]);

SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR4, sizeof(uint8), &receiveBuffer[1]);



<Result>

GATT Profile Table

**<READ>**

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: Clock**  Description: This characteristic is used to send the measurement operation time.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Mandatory | | Writing | Excluded | | Notification | Excluded | |

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: Model Number**  Description: This characteristic represents the number of the model of the device.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Mandatory | | Writing | Excluded | | Notification | Excluded | |
| **Name: Manufacturer Name**  Description: This characteristic represents the name of the manufacturer of the device.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Mandatory | | Writing | Excluded | | Notification | Excluded | |

*Follow the original GATT table of CC2541.*

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: Notification**  Description: This characteristic is used to send the notification. After detect the actual data, notification would be updated.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Excluded | | Writing | Excluded | | Notification | Mandatory | |

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: Actual Data**  Description: This characteristic is used to send the actual data. It should be 2 bytes. Whenever actual data is operated, notification value would be updated.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Mandatory | | Writing | Excluded | | Notification | Excluded | |
| **Name: Body Sensor Location**  Description: The Body Sensor Location characteristic of the device is used to describe the intended location of the heart rate measurement for the device.  Requirement: Optional | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Mandatory | | Writing | Excluded | | Notification | Excluded | |

**<WRITE>**

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: LED Current**  Description: This characteristic is used to control LED current. Default is 100mA. The current can be changed depend on the body sensor location.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Excluded | | Writing | Mandatory | | Notification | Excluded | |

|  |  |
| --- | --- |
| **Overview** | **Properties** |
| **Name: Frequency Measurement**  Description: This characteristic is used to control the frequency measurement.  Requirement: Mandatory | |  |  | | --- | --- | | **Properties** | **Requirement** | | Reading | Excluded | | Writing | Mandatory | | Notification | Excluded | |

About WRITE part, check the proximity function in keyfob demo code. In keyfob, using write function, you can change the level of the alert. Page 16 of this report has information about it.

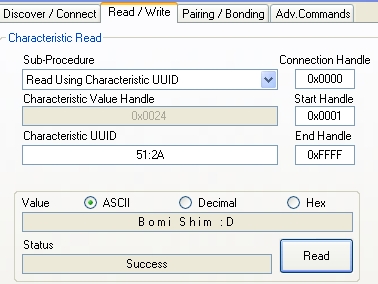
CC2540 Attribute Table

<http://e2e.ti.com/support/low_power_rf/f/538/t/134385.aspx>

**17th Week (09/17-09/22)**

I create new profile in device information service.





<Result>

The code I attached is followed.

**In <devinfo.c>**

// My Name

CONST uint8 devInfoMyNameUUID[ATT\_BT\_UUID\_SIZE] =

{

LO\_UINT16(DEVINFO\_MY\_NAME\_UUID), HI\_UINT16(DEVINFO\_MY\_NAME\_UUID)

};

// My Name

static uint8 devInfoMyNameProps = GATT\_PROP\_READ;

static const uint8 devInfoMyName[] = "Bomi Shim :D";

// My Name

{

{ ATT\_BT\_UUID\_SIZE, characterUUID },

GATT\_PERMIT\_READ,

0,

&devInfoMyNameProps

},

// Manufacturer Name Value

{

{ ATT\_BT\_UUID\_SIZE, devInfoMyNameUUID },

GATT\_PERMIT\_READ,

0,

(uint8 \*) devInfoMyName

},

case DEVINFO\_MY\_NAME:

memcpy(value, devInfoMyName, sizeof(devInfoMyName));

break;

case DEVINFO\_MY\_NAME\_UUID:

// verify offset

if (offset >= sizeof(devInfoMyName))

{

status = ATT\_ERR\_INVALID\_OFFSET;

}

else

{

// determine read length

\*pLen = MIN(maxLen, (sizeof(devInfoMyName) - offset));

// copy data

memcpy(pValue, &devInfoMyName[offset], \*pLen);

}

break;

**In <devinfo.h>**

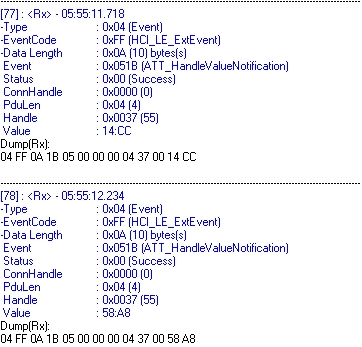
#define DEVINFO\_MY\_NAME 9

#define DEVINFO\_MY\_NAME\_UUID 0x2A51 // My Name

I success to receive 2byte data by making SIMPLE CHAR6. This variance is combined CHAR4 and CHAR5. In <simpleGATTprofile.c>file, you can find the Profile Attributes Table. The first line of the code, the variance is declared as array.

static gattAttribute\_t simpleProfileAttrTbl[SERVAPP\_NUM\_ATTR\_SUPPORTED]

To add more profiles for CHAR6, you should enlarge the array size. To do so, change the value of the SERVAPP\_NUM\_ATTR\_SUPPORTED.



<Result>

Size of data become twice.

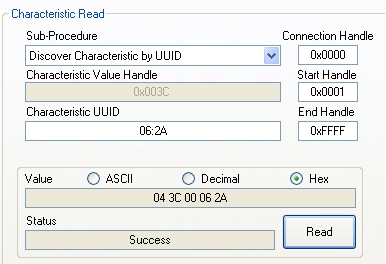
**18th Week (09/24-09/29)**

Success to create WRITE permission attribute table, frequency measurement adjustment. I copied and changed code from <proxreport> in keyfobdemo. I changed variables and set the specific options. Location of files is below.

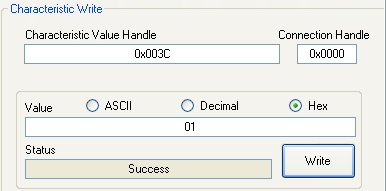
C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\Profiles\SimpleProfile

C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\Profiles\Freq

To create new code properly, not only <freq.c> and <freq.h>, but also you should care about <simpleBLEPeripheral.c> file.



To use WRITE\_PERMISSION attribute, you need those return value. Especially second value is important. If you didn’t change the code in right way, you can’t see the return value.



But when you write on SIMPLEPROFILE\_CHAR series, you should add 1 on the second value. Such as, if you get 10 37 00 F6 FF, then you should use 0x0038 when you use characteristic write function.

The GATT table I made is followed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type (hex)** | **Type (#DEFINE)** | **Hex/Text Value(default)** | **R/W** | **Notes** |
| 0x2800 | GATT\_PRIMARY\_SERVICE\_UUID | 0x180A | READ | Start device information service |
| 0x2803 | GATT\_CHARACTER\_UUID | 02 (read permission)  24 00 (handle 0x0024)  51 2A (UUID 0x2A51) | READ | My name String characteristic |
| 0x2A51 | DEVINFO\_MY\_NAME\_UUID | “Bomi Shim :D” | READ | Value |
| 0x2800 | GATT\_PRIMARY\_SERVICE\_UUID | 0XFFF0 (SIMPLEPROFILE\_SERV\_UUID) | READ | Start Simple GATT profile service |
| 0X2803 | GATT\_CHARACTER\_UUID | 10 (notify only)  37 00 (handle 0x0037)  F6 FF (UUID 0xFFF6) | READ | Characteristic 6 declaration |
| 0xFFF6 | SIMPLEPROFILE\_CHAR6\_UUID | 01:02 (2bytes) | (none) | Value |
| 0x2902 | GATT\_CLIENT\_CHAR\_CFG\_UUID | 00: 00 (2bytes) | READ | WRITE | Configuration |
| 0x2901 | GATT\_CLIENT\_USER\_DEST\_UUID | “Characteristic 6” (17bytes) | READ | User description |
| 0x2800 | GATT\_PRIMARY\_SERVICE\_UUID | 0x1803  (FREQ\_SERVICE\_UUID) | READ | Start adjustment frequency measurement |
| 0x2803 | GATT\_CHARACTER\_UUID | 04 (write permission)  3C 00 (handle 0x003C)  06 2A (UUID 0x2A06) | READ |  |
| 0x2A06 | FREQ\_LEVEL\_UUID | 0 (PP\_FREQ\_LEVEL\_00) | WRITE | Value |
| 0x2800 | GATT\_PRIMARY\_SERVICE\_UUID | 0x1802  (CURRENT\_SERVICE\_UUID) | READ | Start adjustment LED current |
| 0x2803 | GATT\_CHARACTER\_UUID | 04 (write permission)  3F 00 (handle 0x003F)  07 2A (UUID 0x2A07) | READ |  |
| 0x2A07 | CURRENT\_LEVEL\_UUID | 0 (PP\_CURRENT\_LEVEL\_00) | WRITE | Value |

Result when adjustment frequency measurement services from BTool.

|  |  |  |
| --- | --- | --- |
|  | Command | Result |
| 1  (3.125MHz) |  |  |
| 2(1.5625MHz) |  |  |
| 3(781.3kHz) |  |  |
| 4(390.6kHz) |  |  |

<Result from BTool>

Result when adjustment LED current services from BTool.

|  |  |  |
| --- | --- | --- |
|  | Command | Result |
| 0 (0mA) |  |  |
| 1 (20mA) |  |  |
| 2 (40mA) |  |  |
| 3 (60mA) |  |  |
| 4 (80mA) |  |  |
| 5 (100mA) |  |  |
| 6 (120mA) |  |  |
| 7 (140mA) |  |  |
| 8 (160mA) |  |  |
| 9 (180mA) |  |  |
| 10 (200mA) |  |  |

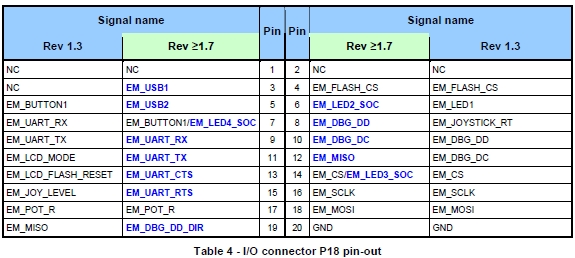
**19th Week (10/01-10/06)**

To save the proximity data, use SOC 8 memory chip. SmartRF05 board already has memory chip in it, but the space is too small and in the future we want to use just VCNL4000 chip without board, therefore use another chip, SOC 8. In order to use SOC 8, learn SPI function.

<http://www.digikey.com/product-detail/en/MX25L6445EM2I-10G/1092-1065-ND/2744748>

<Hardware aspect>

Memory chip has 4 pin with it: SCLK(serial clock), CS(chip select), MISO(SPI data master input slave output), and MOSI(SPI data master output slave input). Probe connector in smartRF05 board is followed.



We will use pin 12, 14, 16 and 18 in p18.

<Software aspect>

Even though <simpleBLEperipheral> project has UART function in HAL, but SPI function should be used. <keyfobdemo> project has SPI function itself in <cma3000d.c> and <cma3000c.h>. Copy those files and paste it into <simpleBLEperipheral>. Also put both of files in this directory.

C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\SimpleBLEPeripheral\Source

In <cma3000d.h> files, you can see the two function for SPI.

***void spiWriteByte(uint8 write)***

*{*

*U0CSR &= ~0x02; // Clear TX\_BYTE*

*U0DBUF = write;*

*while (!(U0CSR & 0x02)); // Wait for TX\_BYTE to be set*

*}*

***void spiReadByte(uint8 \*read, uint8 write)***

*{*

*U0CSR &= ~0x02; // Clear TX\_BYTE*

*U0DBUF = write;*

*while (!(U0CSR & 0x02)); // Wait for TX\_BYTE to be set*

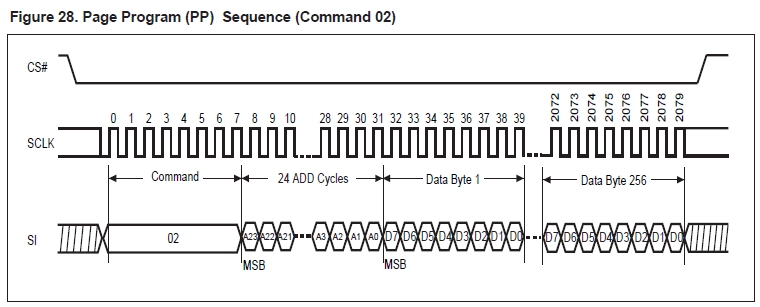
*\*read = U0DBUF;*

*}*

The way SPI works is similar with I2C. Therefore, function <spiWriteByte> is working as “Save”, and function <spiReadByte> is working as “Load”.

**20th Week (10/08-10/13)**

To save data from VCNL4000, we should consider two parts. One is address number in SOP 8, the other one is memory data byte which is stored in SOP 8. There are two types of saving data, page program(PP) and continuously program(CP). Compare to PP, CP needs to be declared more command. Since it could be regarded as CP consumes more energy, we use PP.



Create whole code in <simpleBLEPeripheral.c>. Code is followed.

*uint8 sendBuffer[100];*

*uint8 receiveBuffer[10];*

*uint8 spiBuffer[3];*

*uint16 data;*

*receiveBuffer[0] = 0x00;*

*receiveBuffer[1] = 0x00;*

*receiveBuffer[2] = 0x00;*

*sendBuffer[0] = 0x83; //LED current*

*sendBuffer[1] = 0x0A;*

*sendBuffer[2] = 0x89; //measurement frequency*

*sendBuffer[3] = 0x02;*

*sendBuffer[4] = 0x80; //Command*

*sendBuffer[5] = 0x08;*

*sendBuffer[6] = 0x81;*

*sendBuffer[7] = 0x87;*

*spiBuffer[0] = 0x00; //MSB*

*spiBuffer[1] = 0x00;*

*spiBuffer[2] = 0x00; //LSB*

*while((spiBuffer[2]=0xFF)&(spiBuffer[1]=0xFF)&(spiBuffer[0]=0x7F)){*

*for(int count=0;count<128;count++)*

*{*

*HalI2CWrite(2, &sendBuffer[4]);*

*HalI2CWrite(2, &sendBuffer[0]);*

*HalI2CWrite(2, &sendBuffer[2]);*

*HalI2CWrite(1, &sendBuffer[7]);*

*HalI2CRead(2, &receiveBuffer[0]);*

*data = receiveBuffer[1] << 8;*

*data |= receiveBuffer[0];*

*SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR6, SIMPLEPROFILE\_CHAR6\_LEN, &data);*

*spiWriteByte(0x02); //command*

*spiWriteByte(&spiBuffer[0]); //Address1 MSB*

*spiWriteByte(&spiBuffer[1]); //Address2*

*spiWriteByte(&spiBuffer[2]); //Address3 LSB*

*spiWriteByte(receiveBuffer[0]);*

*spiWriteByte(receiveBuffer[1]);*

*spiBuffer[1]++;*

*printf("%d\n", receiveBuffer[0]);*

*printf("%d\n", receiveBuffer[1]);*

*printf("++++++++\n");*

*HalI2CWrite(2, &sendBuffer[4]);*

*}*

*if(spiBuffer[1]==0xFF){*

*spiBuffer[0]+=0x01;*

*spiBuffer[1]++;*

*}*

*}*

**21th Week (10/15-10/20)**

Modify the previous code.

*while((spiBuffer[2]<0xFF)&(spiBuffer[1]<0xFF)&(spiBuffer[0]<0x7F)){*

*for(int count=0;count<128;count++)*

*{*

*HalI2CWrite(2, &sendBuffer[4]);*

*HalI2CWrite(2, &sendBuffer[0]);*

*HalI2CWrite(2, &sendBuffer[2]);*

*HalI2CWrite(1, &sendBuffer[7]);*

*HalI2CRead(2, &receiveBuffer[0]);*

*data = receiveBuffer[0] << 8;*

*data |= receiveBuffer[1];*

*SimpleProfile\_SetParameter(SIMPLEPROFILE\_CHAR6, SIMPLEPROFILE\_CHAR6\_LEN, &data);*

*spiWriteByte(0x02); //command*

*spiWriteByte(&spiBuffer[0]); //Address1 MSB*

*spiWriteByte(&spiBuffer[1]); //Address2*

*spiWriteByte(&spiBuffer[2]); //Address3 LSB*

*spiWriteByte(receiveBuffer[0]);*

*spiWriteByte(receiveBuffer[1]);*

*spiBuffer[2]++;*

*printf("%d\n", receiveBuffer[0]);*

*printf("%d\n", receiveBuffer[1]);*

*printf("%d\n", data);*

*printf("++++++++\n");*

*printf("%d\n", spiBuffer[2]);*

*printf("%d\n", spiBuffer[1]);*

*printf("%d\n", spiBuffer[0]);*

*printf("\*\*\*\*\*\*\*\*\n");*

*HalI2CWrite(2, &sendBuffer[4]);*

*}*

*if(spiBuffer[2]==0xFF){*

*spiBuffer[1]+=0x01;*

*spiBuffer[2]=0x00;*

*}*

*if(spiBuffer[1]==0xFF){*

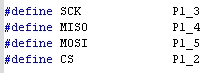
*spiBuffer[0]+=0x01;*

*spiBuffer[1]=0x00;*

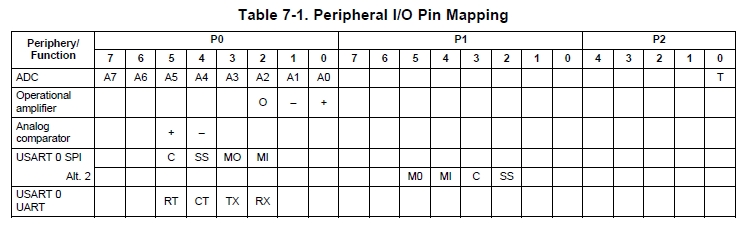
*}*

*}*

From CC2540 <cma3000d.c> code, you can see that SPI pin number is declared.



Check the pin mapping in CC2541.



<SWRU191C.pdf>

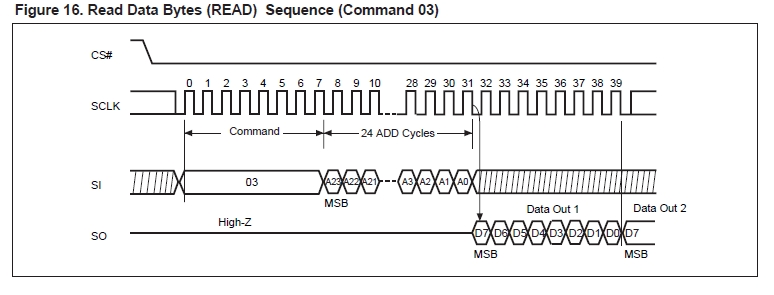
Used pin number in <cma3000d.c> file is mapping with USART 9 SPI Alt.2 in P1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | SOC | CC2541 | SmartRF05EM | Bridge color |  |
| SCK | 6 | P1\_3 | P18\_16 | Blue | P18\_4 |
| CS | 1 | P1\_2 | P18\_14 | Yellow | P20\_18 |
| MOSI | 5 | P1\_5 | P18\_18 | Gray | P18\_16 |
| MISO | 2 | P1\_4 | P18\_12 | Red | P18\_14 |
| Vcc | 8 |  | P20\_3 |  | P20\_3 |
| GND | 4 |  | P18\_20  P20\_20 | Black | P18\_20  P20\_20 |

<Pin numbers>

**22th Week (10/22-10/27)**

Success to connect memory to CC2541. And using command sets in mamory chip manual, I made code to read memory. Code is followed.



*spiWriteByte(0x06); //WREN*

*spiWriteByte(0x03); //command*

*spiWriteByte(spiBuffer[0]); //Address1 MSB*

*spiWriteByte(spiBuffer[1]); //Address2*

*spiWriteByte(spiBuffer[2]); //Address3 LSB*

*printf("%d\n", spireceiveBuffer[0]);*

*printf("%d\n", spireceiveBuffer[1]);*

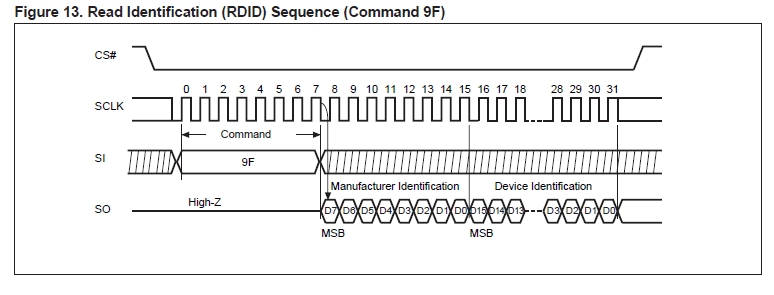
*printf("ooooooooooo\n");*

*spiReadByte(&spireceiveBuffer[0], receiveBuffer[0]);*

*spiReadByte(&spireceiveBuffer[1], receiveBuffer[1]);*

|  |  |
| --- | --- |
|  | spirecieveBuffer[0] Before get data  spireceiveBuffer[1] Before get data  spirecieveBuffer[0] After get data  spireceiveBuffer[1] After get data  MSB proximity data from VCNL4000  LSB proximity data from VCNL4000  LSB memory address  Memory address  MSB memory address |

However, even though VCNL4000 proximity data keeps changing, memory’s data is always 255. Maybe pin connection has problem. To check whether pin connected properly, I use RDID command, read identification. RDID command operates reading manufacture information.



**23th Week (10/29-11/03)**

ddd