TI Designs
Bluetooth® ***Low Energy for Energy Monitoring***

TI Designs

TI Designs provide the foundation that you need including methodology, testing, and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDC-BLE-METER-READINGS	Tool Folder Containing Design Files
CC2650	Product Folder
MSP430I2040	Product Folder
UN2003LV	Datasheet
CC-DEVPACK-DEBUG	Product Folder
CC2650STK	Tool Folder
TIDM-3OUTSMTSTRP	Tool Folder

TI E2E™ Community

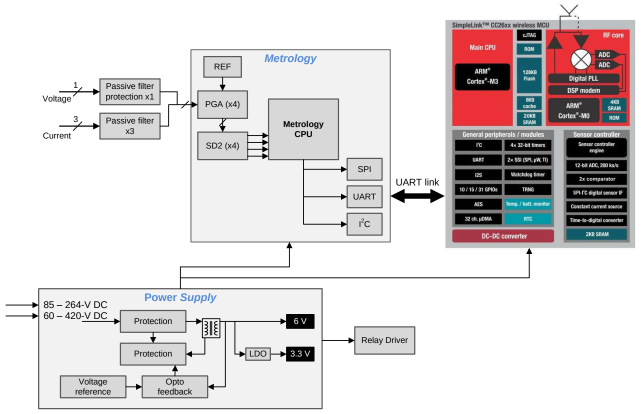
ASK Our E2E Experts
WEBENCH® Calculator Tools

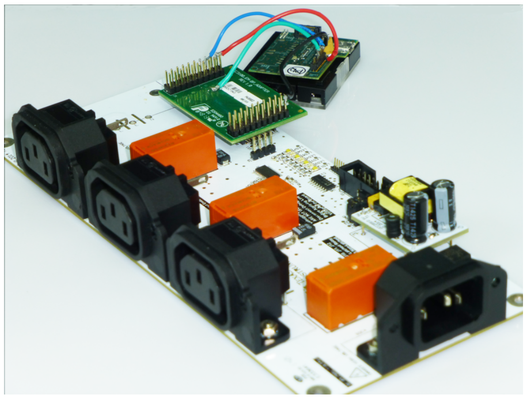
Design Features

- Reading of Energy Monitor Parameters With *Bluetooth®* Low Energy (LE) Link
- Simple Creation of Profile for *Bluetooth* Low Energy
- Interface to Energy Monitoring Device Through Universal Asynchronous Receiver and Transmitter (UART)

Featured Applications

- Utility Energy Monitoring
- Industrial Embedded Energy Monitoring
- Home Automation
- Appliances





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

1.1 Cautions and Warnings

The TIDC-BLE-METER-READING TI design is designed to operate hardware powered directly from an AC supply. TI advises that only professionals with the appropriate technical training operate this hardware.

Before operating the hardware please read the safety-related documents that accompany the user's guide.

CAUTION

Read user guide before use.

CAUTION

Do not leave the EVM powered when unattended.

CAUTION

HOT SURFACE: Contact may cause burns. Do not touch.

CAUTION

HIGH VOLTAGE: Electric shock possible when connecting board to live wire. The board should be handled with care by a professional. For safety, use isolated equipment with overvoltage and overcurrent protection.

1.2 System Description

This design is about an application for reading an energy monitor device over a *Bluetooth*® Low Energy (BLE) link using the SimpleLink™ CC2650 multi-standard wireless MCU and corresponding SensorTag module as the development platform. This guide details dedicating, defining, and implementing a profile to meter reading. The module is then connected to the hardware (with a minor modification) from the TI design TIDM-3OUTMSTSTRP as the metering data source. This TI design also includes an Android application that functions as a remote reader and control terminal.

As an example to show readers how to create custom-tailored services and applications, this document details the process of creating the aforementioned services incorporated into the profile of the SensorTag application.

1.3 CC2650

The CC2650 is a multi-standard, 2.4-GHz ultra-low-power wireless MCU. The CC2650 in this design functions as the *Bluetooth* LE controller. The CC2650 is a self-contained controller that comprises an RF controller, MCU, and a sensor controller. These components allow a very simple and self-contained implementation of a *Bluetooth* LE device. In addition to the *Bluetooth* LE low-level functionalities, and because the CC2650 contains an MCU, including the profile services is possible, as well as implementing the application and interface to monitor energy on a single chip.

1.4 MSP430i2040

The MSP430i2040 MCU functions as the metrology processor in this design. The MSP430i2040 device allows accurate energy measurements with its four 24-bit sigma-delta analog-to-digital converters (ADCs), providing voltage, current, power (active, reactive, and apparent), power factor, and frequency readings of three AC outlets. The MSP430i2040 only requires a few passive external components to directly interface to the voltage divider and current shunt for voltage and current measurements.

2 Block Diagram

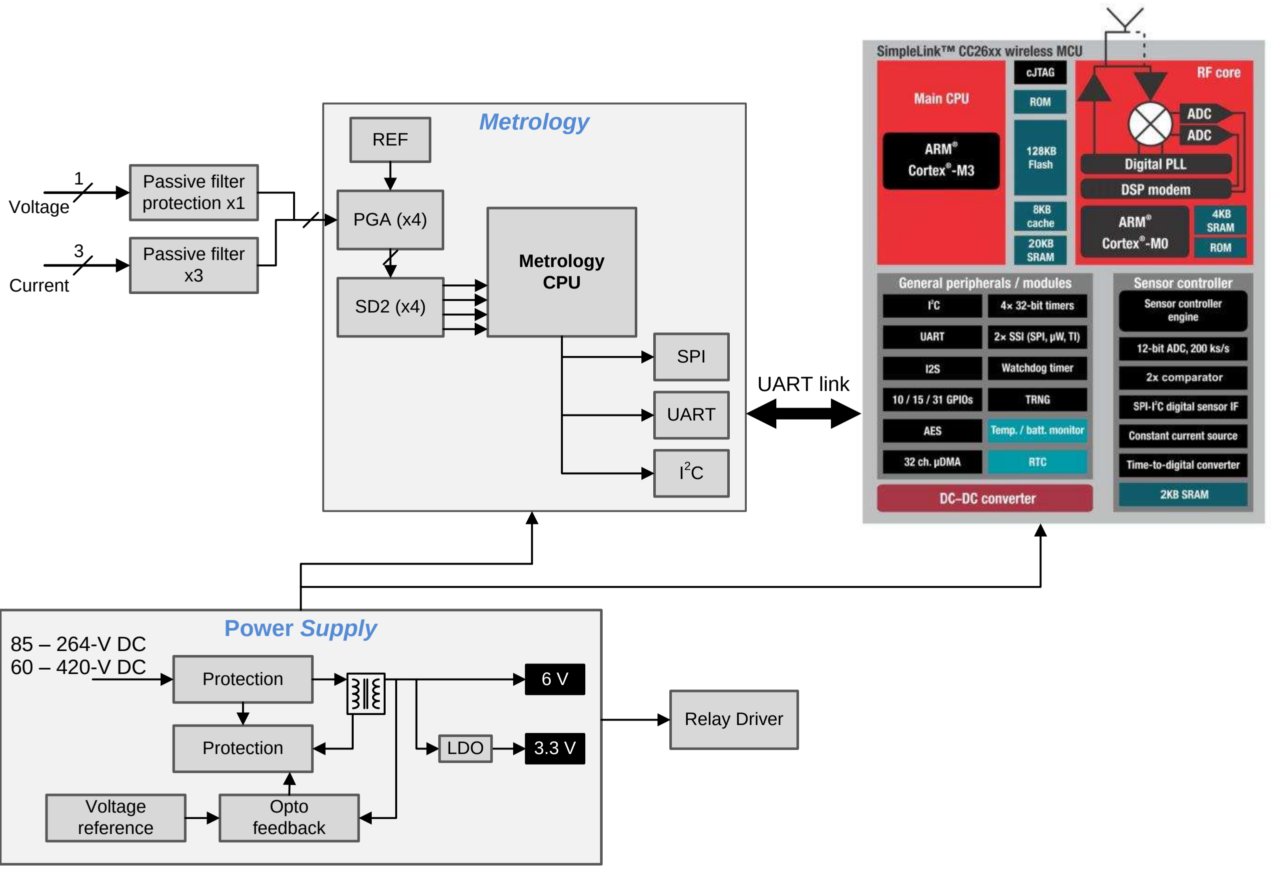


Figure 1. TIDC-BLE-METER-READINGS System Block Diagram

2.1 Highlighted Products

2.1.1 CC2650

- MCU
  - Powerful ARM® Cortex-M3
  - EEMBC CoreMark® Score: 142
  - Up to 48-MHz of clock speed
  - 128KB of in-system programmable flash
  - 8-KB SRAM for cache
  - 20-KB ultra-low leakage SRAM
  - 2-pin cJTAG and JTAG debugging
  - Supports over-the-air upgrade (OTA)
- Ultra-low power sensor controller
  - Can run autonomous from the rest of the system
  - 16-bit architecture
  - 2-KB ultra-low leakage SRAM for code and data
- Efficient code size architecture, placing drivers, *Bluetooth* LE controller, IEEE 802.15.4 MAC, and Bootloader in ROM

2.1.2 MSP430i2040

- Supply voltage range of 2.2 V to 3.6 V
- 16-bit RISC architecture, up to 16.384-MHz system clock
- Memories
  - Up to 32KB of flash main memory
  - 1KB of flash information memory
  - Up to of 2KB of RAM
- Clock system
  - 16.384-MHz internal DCO
  - DCO operation with internal or external resistor
  - External digital clock source
- Up to four 24-bit sigma-delta ADCs with differential programmable-gain amplifier (PGA) inputs
- Two 16-bit timers with three capture and compare registers each
- Enhanced universal serial communication interfaces (eUSCIs)
  - eUSCI_A0
    - Enhanced UART with automatic baud-rate detection
    - IrDA encoder and decoder
    - Synchronous SPI
  - eUSCI_B0
    - Synchronous SPI
    - I^2C
- 16-bit hardware multiplier

3 System Design Theory

Basic *Bluetooth* LE

This section is a brief introduction to *Bluetooth* LE implementation. For more information, refer to the *Texas Instruments Wiki* [4] and the *Bluetooth Developer Portal* for more details [5].

To begin working with *Bluetooth* LE, understanding the definitions of a few key terms is important, specifically the following roles and operations: Master, Slave, Client, and Server; Read, Write, Notify, and Indicate; and Profile, Service, and Characteristic.

- A Master (sometimes called “central“) is the device or devices that scan for other devices. Usually a master is implemented on a smart device such as a smartphone, tablet, or PC usually implement a master.
- A Slave (sometimes called “peripheral”) is the device that promotes its own visibility and waits for a connection. Usually a slave is implemented on a remote device.
- A Client is a device that remotely accesses data or resources through *Bluetooth* LE over a Generic Attribute Profile (GATT) protocol (the BLE protocol). A client is usually implemented on a master, although this is not a requirement. The client is not the same as a master.
- A Server is a device that sources data, receives data, or provides resources to clients. A server is usually implemented on a slave; although this is not a requirement. The server is not the same as a slave.

An important thing to note when working with *Bluetooth* LE is that a single device is not restricted to either being a master or slave (except a device cannot function as both simultaneously). A single device can also be configured to be a client of one set of data while functioning as a server of another set of data, which is why a device can function as a client and serve in parallel.

In the application level, to transfer data over *Bluetooth* LE, four operations are used:

- Read operation −The client device requesting data from a server device
- Write operation −The client device request to transmit data to a server device
- Notify −The action where a server device transmits data into the client device
- Indicate –This action is similar to Notify except that Indicate requires an acknowledgment from the client

Before using the above operations for data exchange, identifying whether the server device provides the desired resources and data that the client is seeking is necessary. *Bluetooth* LE Utilizes a data structure known as a profile for this identification.

- A characteristic uses a data structure to contain the unique identification (in the form of a 128-bit or 16-bit universally unique identifier, or UUID) for accessing the data and the permission to access the data.
- A service uses a data structure to contain the information and unique identification (in the form of a 128-bit or 16-bit UUID) of the service. A service also contains a collection of characteristics that the service provides.
- A profile uses a data structure to contain the unique identification (in the form of a 128-bit or 16-bit UUID) of the profile, then the service and data is provided. A profile contains a collection of services that it provides.

Figure 2 shows the hierarchical relationship between the profile, service, and characteristic roles or operations.

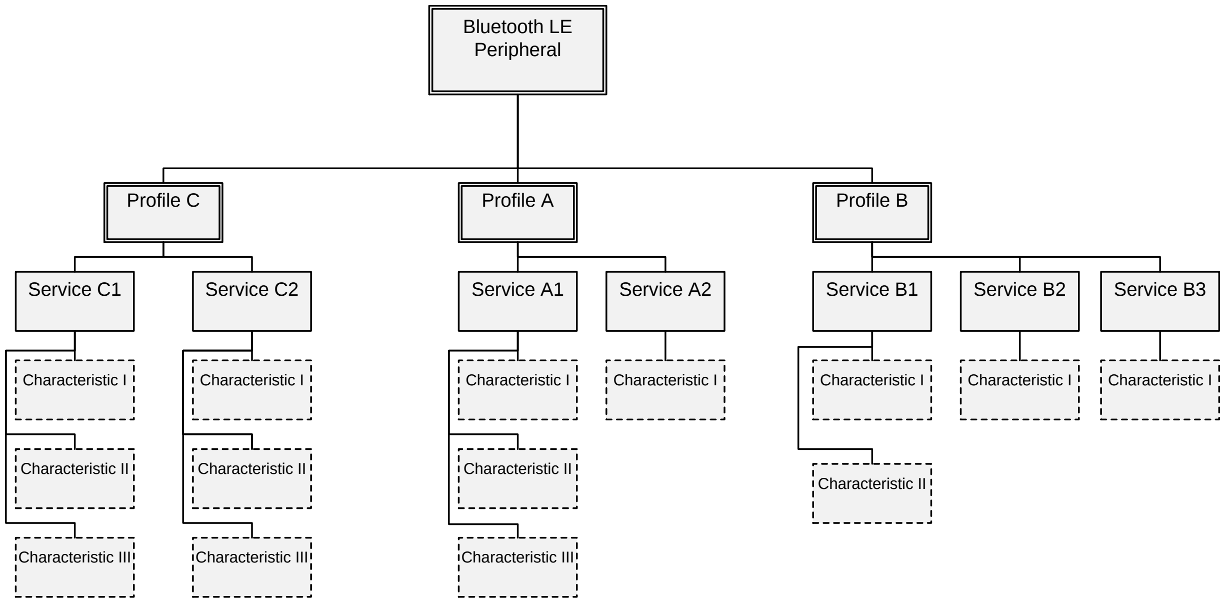


Figure 2. Profile, Service, and Characteristic Hierarchy

3.1 *Metering Service Definition*

The TIDC-BLE-METER-READINGS design designates the original CC2650 SensorTag (CC2650STK) board as a slave device running as a server. The user then adds the metering service profile to the original SensorTag profile.

The first step to adding the meter reading function with *Bluetooth* LE is to define the metering service. This service provides the characteristics that allow the client application to access the required information.

The UUID chosen for the metering services is based on the TI UUID used in the SensorTag application: F000XXXX-0451-4000-B000-000000000000 (where “XXXX” is” EE00). The UUID characteristics are defined with a sequence number starting from EE01.

The access attribute of the dynamic parameters (voltage, current, power, power factor, and frequency) is “read” and “notify”. The static characteristics are designed to be read-only and the controllable characteristics are read-and-write enabled. Figure 3 shows the definition of the metering service and its characteristics.

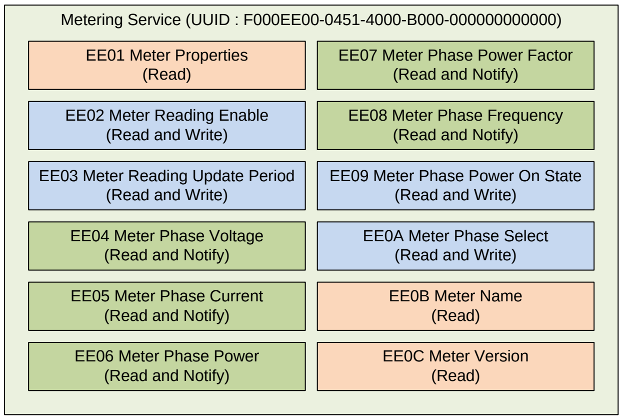
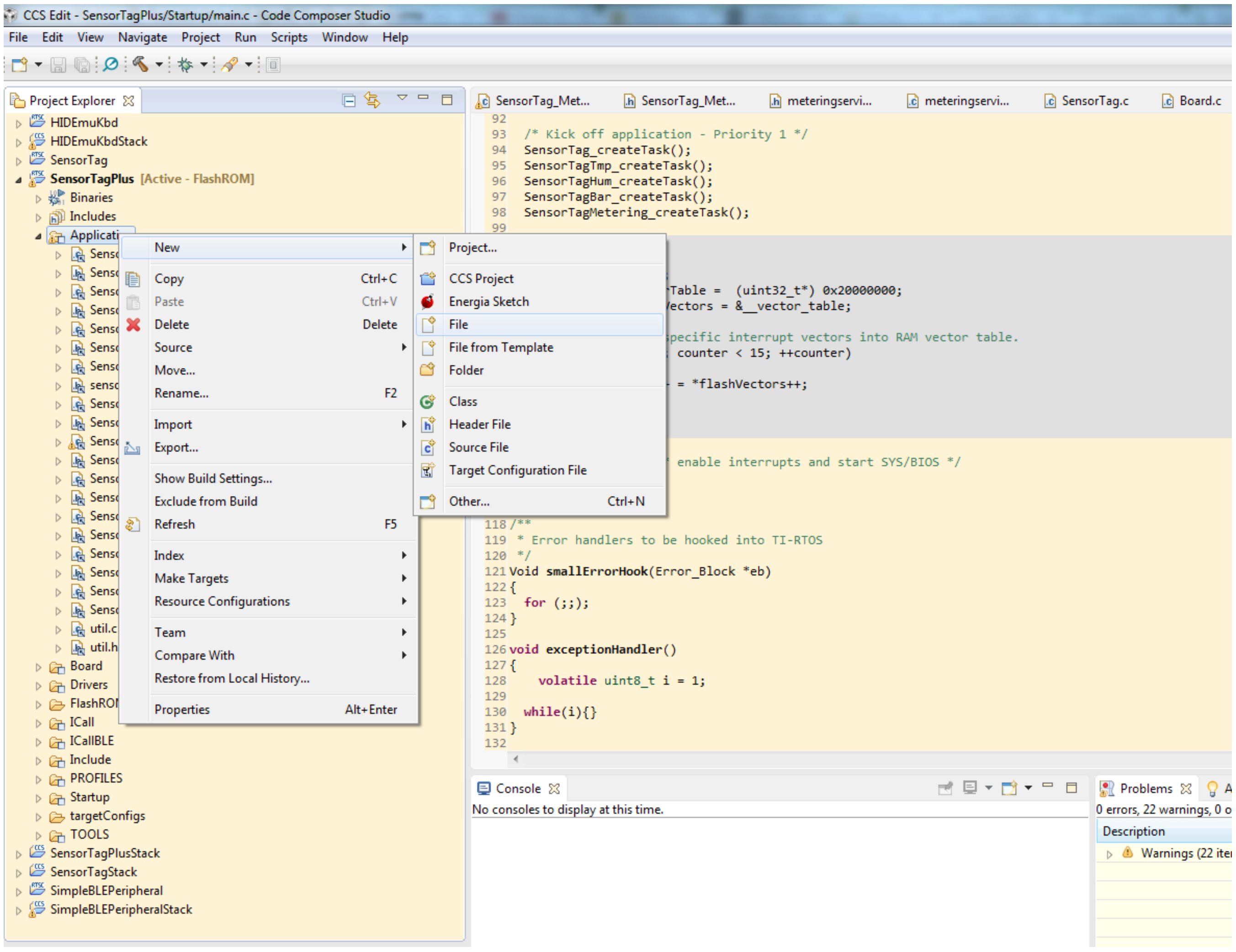

Metering Service (UUID : F000EE00-0451-4000-B000-000000000000)
EE01 Meter Properties (Read)
EE07 Meter Phase Power Factor (Read and Notify)
EE02 Meter Reading Enable (Read and Write)
EE08 Meter Phase Frequency (Read and Notify)
EE03 Meter Reading Update Period (Read and Write)
EE09 Meter Phase Power On State (Read and Write)
EE04 Meter Phase Voltage (Read and Notify)
EE0A Meter Phase Select (Read and Write)
EE05 Meter Phase Current (Read and Notify)
EE0B Meter Name (Read)
EE06 Meter Phase Power (Read and Notify)
EE0C Meter Version (Read)

Figure 3. Metering Service

3.2 Adding Files

The user must add the following files to the project:

1. In the *SensorTag→ Application* folder, add a link to the file (see Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8):
   ORG_PROJ_DIR\..\..\..\Source\Application\SensorTag_Metering.c
2. In the same folder, add a link to the file:
   ORG_PROJ_DIR\..\..\..\Source\Application\SensorTag_Metering.h
3. In the *SensorTag→ Board→ Devices* folder, add a link to the file:
   PROJECT_LOC\..\..\Source\Device\dlt645.h
4. In the same folder, add a link to the file:
   PROJECT_LOC\..\..\Source\Device\dlt645client.c

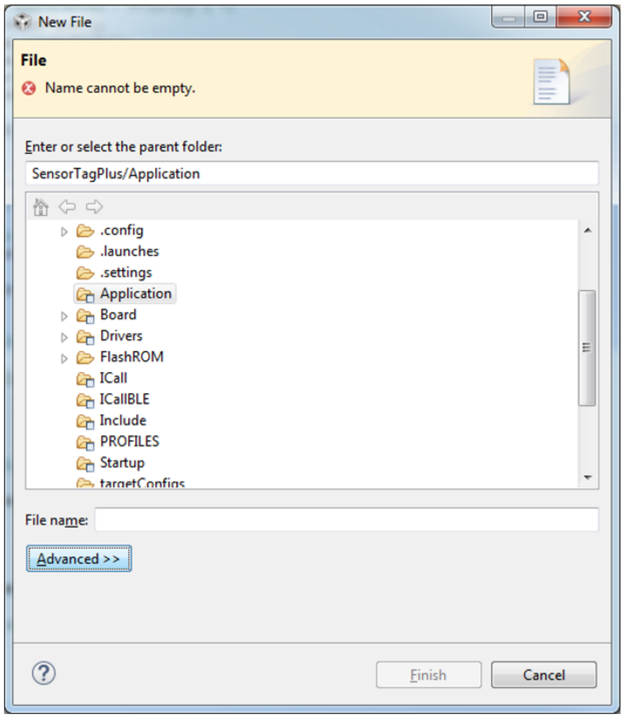


```
/* Kick off application - Priority 1 */
SensorTag_createTask();
SensorTagTmp_createTask();
SensorTagHum_createTask();
SensorTagBar_createTask();
SensorTagMetering_createTask();

Table =  (uint32_t*) 0x20000000;
ectors = &__vector_table;

pecific interrupt vectors into RAM vector table.
 counter < 15; ++counter)

 = *flashVectors++;

 enable interrupts and start SYS/BIOS */

/**
 * Error handlers to be hooked into TI-RTOS
 */
Void smallErrorHook(Error_Block *eb)
{
  for (;;);
}

void exceptionHandler()
{
    volatile uint8_t i = 1;

  while(i){}
}
```

No consoles to display at this time.

0 errors, 22 warnings, 0 o

Figure 4. Add Link to File 1: Right-Click Folder to Add − Select *New→ File*



File

Name cannot be empty.

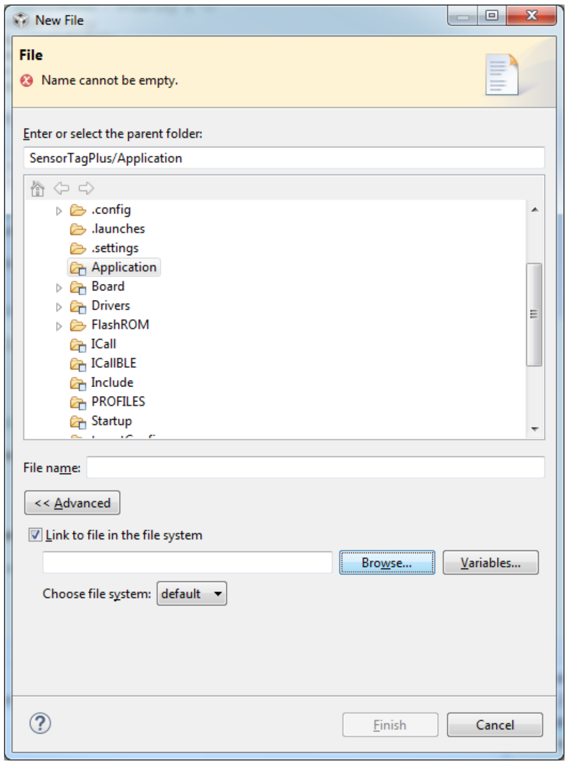
Enter or select the parent folder:

SensorTagPlus/Application

- .config
- .launches
- .settings
- Application
- Board
- Drivers
- FlashROM
- ICall
- ICallBLE
- Include
- PROFILES
- Startup
- targetConfigs

File name:

Figure 5. Add Link to File 2: Click "*Advanced >>*"



File

Name cannot be empty.

Enter or select the parent folder:

SensorTagPlus/Application

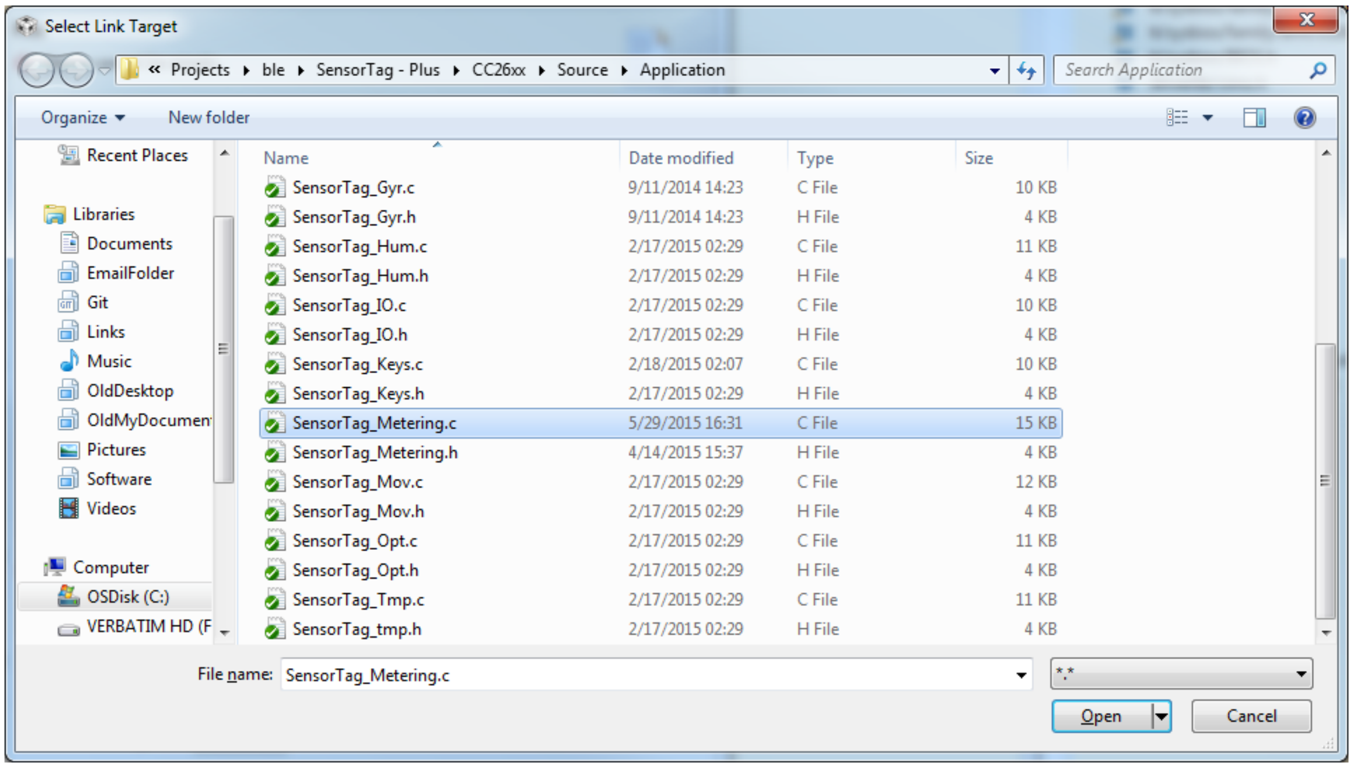
- .config
- .launches
- .settings
- Application
- Board
- Drivers
- FlashROM
- ICall
- ICallBLE
- Include
- PROFILES
- Startup

File name:

☑ Link to file in the file system

Choose file system: default

Figure 6. Add Link to File 3: Check "*Link to file in the file system*"

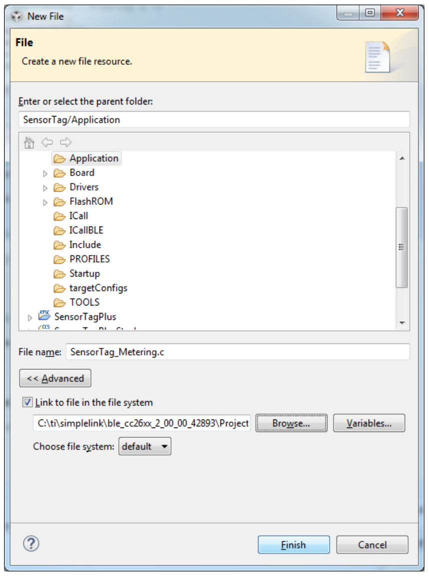


Name	Date modified	Type	Size
SensorTag_Gyr.c	9/11/2014 14:23	C File	10 KB
SensorTag_Gyr.h	9/11/2014 14:23	H File	4 KB
SensorTag_Hum.c	2/17/2015 02:29	C File	11 KB
SensorTag_Hum.h	2/17/2015 02:29	H File	4 KB
SensorTag_IO.c	2/17/2015 02:29	C File	10 KB
SensorTag_IO.h	2/17/2015 02:29	H File	4 KB
SensorTag_Keys.c	2/18/2015 02:07	C File	10 KB
SensorTag_Keys.h	2/17/2015 02:29	H File	4 KB
SensorTag_Metering.c	5/29/2015 16:31	C File	15 KB
SensorTag_Metering.h	4/14/2015 15:37	H File	4 KB
SensorTag_Mov.c	2/17/2015 02:29	C File	12 KB
SensorTag_Mov.h	2/17/2015 02:29	H File	4 KB
SensorTag_Opt.c	2/17/2015 02:29	C File	11 KB
SensorTag_Opt.h	2/17/2015 02:29	H File	4 KB
SensorTag_Tmp.c	2/17/2015 02:29	C File	11 KB
SensorTag_tmp.h	2/17/2015 02:29	H File	4 KB

File name: SensorTag_Metering.c

.

Figure 7. Add Link to File 4: Select File to Add



Enter or select the parent folder: SensorTag/Application

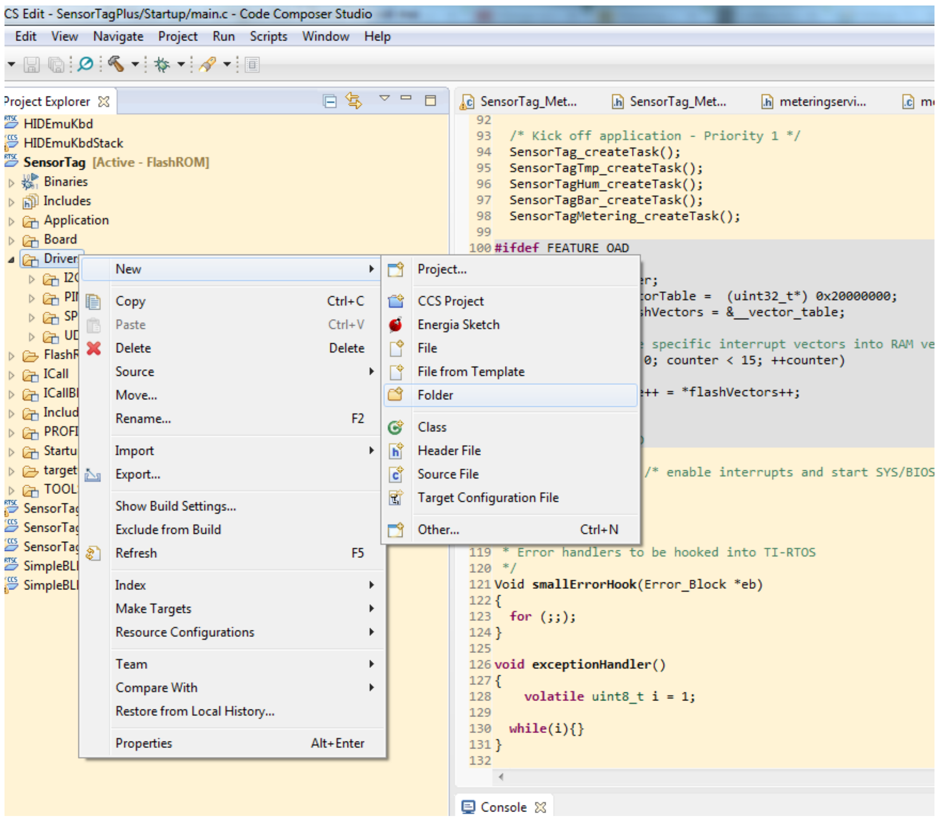
File name: SensorTag_Metering.c

[✓] Link to file in the file system: C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Project

Choose file system: default

Figure 8. Add Link to File 4: Click "*Finish*"

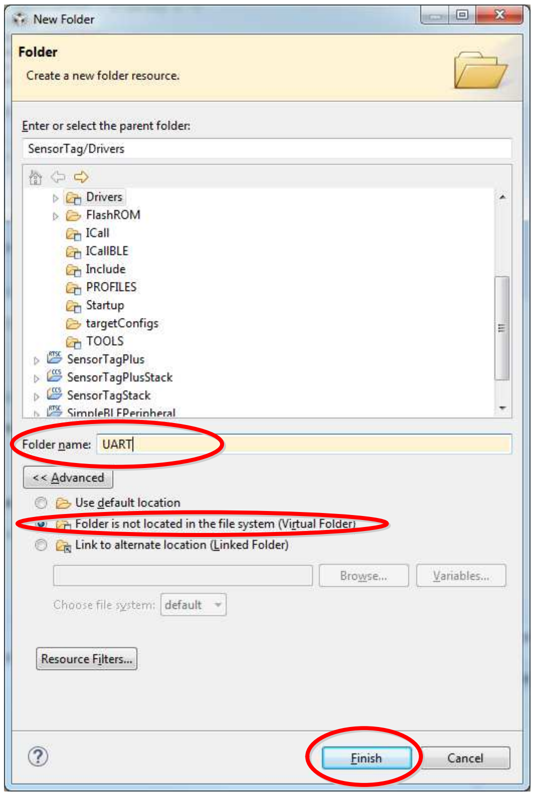
5. In the SensorTag->PROFILES add a link to the file: *ORG_PROJ_DIR\..\..\..\..\..\Profiles\SensorProfile\CC26xx\meteringservice.c*
6. In the SensorTag->PROFILES add a link to the file: *ORG_PROJ_DIR\..\..\..\..\..\Profiles\SensorProfile\CC26xx\meteringservice.h*
7. Create a virtual folder in the project under *SensorTag→ Drivers* and name it '"UART" (Figure 9)



```
  /* Kick off application - Priority 1 */
  SensorTag_createTask();
  SensorTagTmp_createTask();
  SensorTagHum_createTask();
  SensorTagBar_createTask();
  SensorTagMetering_createTask();

#ifdef FEATURE_OAD
                                    r;
                                    orTable =  (uint32_t*) 0x20000000;
                                    hVectors = &__vector_table;

                                     specific interrupt vectors into RAM ve
                                    0; counter < 15; ++counter)

                                    ++ = *flashVectors++;

                                    /* enable interrupts and start SYS/BIOS

 * Error handlers to be hooked into TI-RTOS
 */
Void smallErrorHook(Error_Block *eb)
{
  for (;;);
}

void exceptionHandler()
{
    volatile uint8_t i = 1;

  while(i){}
}
```

Figure 9. Create Virtual Folder 1: Right-Click Folder to Add—Select *New→ Folder*

8. Click the “*Advanced* >>” button, enable the radio option for “*Folder is not located in the file system (Virtual Folder)*”, type "UART" into the*Folder name* field, and then click the “*Finish*" button (see Figure 10)



Enter or select the parent folder: SensorTag/Drivers

Folder name: UART

[✓ Folder is not located in the file system (Virtual Folder)]

Choose file system: default

Figure 10. Create Virtual Folder 2

9. In this virtual folder, add link to file:
   TI_RTOS_DRIVERS_BASE\ti\drivers\UART.c
10. In the same folder add link to file:
    TI_RTOS_DRIVERS_BASE\ti\drivers\UART.h
11. In the same folder add link to file:
    TI_RTOS_DRIVERS_BASE\ti\drivers\uart\UARTCC26XX.c
12. In the same folder add link to file:
    TI_RTOS_DRIVERS_BASE\ti\drivers\uart\UARTCC26XX.h

3.3 Coding for Metering Service, Metering Application, and Changes to SensorTag Application Code

For further details, view Section 1, Section 2, and Section 3 of the coding details document (TIDCAH5).

3.4 Board File

To make the board function correctly, the final board file requires a final, minor modification that defines how the pins are connected.

Click on the link *SensorTag→ Startup→ Board.c→ CC26XXST_0120/Board.c* and change the highlighted lines:

```
PIN_Config BoardGpioInitTable[] = {
    Board_LED1       | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,     /* LED initially off
    Board_LED2       | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,     /* LED initially off
    Board_KEY_LEFT   | PIN_INPUT_EN | PIN_PULLUP | PIN_IRQ_BOTHEDGES |
PIN_HYSTERESIS,        /* Button is active low          */
    Board_KEY_RIGHT  | PIN_INPUT_EN | PIN_PULLUP | PIN_IRQ_BOTHEDGES |
PIN_HYSTERESIS,        /* Button is active low          +/
    Board_RELAY      | PIN_INPUT_EN | PIN_PULLDOWN | PIN_IRQ_BOTHEDGES |
PIN_HYSTERESIS,      /* Relay is active high          */
    Board_MPU_INT    | PIN_INPUT_EN | PIN_PULLDOWN | PIN_IRQ_NEGEDGE |
PIN_HYSTERESIS,        /* MPU_INT is active low         */
    Board_TMP_RDY    | PIN_INPUT_EN | PIN_PULLUP | PIN_HYSTERESIS,
/* TMP_RDY is active high        */
    Board_BUZZER     | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,     /* Buzzer initially off          */
    Board_MPU_POWER  | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MAX,    /* MPU initially on              */
    Board_MIC_POWER  | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MIN,     /* MIC initially off             */
    Board_SPI_FLASH_CS | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MIN,  /* External flash chip select    */
    Board_SPI_DEVPK_CS | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MIN,  /* DevPack chip select           */
   Board_AUDIO_DI | PIN_INPUT_EN | PIN_PULLDOWN,
/* Audio DI                      +/
    Board_AUDIODO | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MIN,       /* Audio data out                */
    Board_AUDIO_CLK | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
    Board_DP2 | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
    Board_DP1 | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
    Board_DP0 | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
    Board_DP3 | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
//    Board_DP4_UARTRX | PIN_INPUT_EN | PIN_PULLDOWN,
/* DevPack */
//    Board_DP5_UARTTX | PIN_INPUT_EN | PIN_PULLDOWN,
/* Devpack */
    Board_DEVPK_ID | PIN_INPUT_EN | PIN_NOPULL,
/* Device pack ID - external PU  */
    PIN_TERMINATE
};

/* UART hardware parameter structure, also used to assign UART pins */
const UARTCC26XX_HWAttrs uartCC26XXHWAttrs[CC2650_UARTCOUNT] = {
    {    /* CC2650_UART0 */
       .baseAddr = UART0_BASE,
        .intNum = INT_UART0,
        .powerMngrId = PERIPH_UART0,
//        .txPin = Board_EB_UART_TX,
//        .rxPin = Board_EB_UART_RX,
        .txPin = Board_DP4_UARTRX,
        .rxPin = Board_DP5_UARTTX,
        .ctsPin = PIN_UNASSIGNED,
        .rtsPin = PIN_UNASSIGNED
    },
};
```

NOTE: Remember to click on the save button to save this change. The CCS IDE does not automatically save upon building the project.

3.5 Hints

3.5.1 Alternative to Modifying SensorTag Code

The provided software installation file consists of a CCS project that contains all the necessary information and changes involved (except for the *board.c* file that Section 3.4 discusses). The folder "SensorTag – Plus" consists of the whole set of source code for the *SensorTag* application, including the metering application. The folder "SensorProfile – Plus" contains all of the profiles SensorTag uses in addition to the source files for the metering service profile. Utilize the following steps to add the project CCS.

1. Copy the "SensorTag – Plus" folder to the following directory path: *C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble*
2. Copy the “SensorProfile – Plus” folder into the following directory path: *C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble\Profiles*
3. Import the project “SensorTagPlus” into the CCS software
4. Import the project “SensorTagPlusStack” into the CCS software
5. Rebuild the "SensorTagPlusStack" project and download the resultant code to the SensorTag hardware
6. Modify the *Board.c* file as Section 3.4 describes
7. Rebuild the SensorTagPlus project and download the resultant code to the SensorTag Hardware
8. In the case of an error that prompts that ST_METER is undefined, open *SensorTagPlus→ Board→ Interfaces→ sensor.h* and make the changes

3.5.2 CCS Environment

After compiling the project and code modifications, the code can successfully run on the hardware using:

- TI's Code Composer Studio™ (CCS) software Versions 6.0.1 and 6.1.0
- TI-RTOS Version “tirtos_simplelink_2_11_01_09”
- TI's *Bluetooth* LE-Stack™ (BLE-Stack) Version 2.0.0.42893

The following directory paths assume that the user has the default folder settings for CCS, TI-RTOS, and BLE-STACK 2.00.00.42893.

- The CCS 6 file location is *C:\ti\ccsv6*
- The TI-RTOS folder location is *C:\ti\tirtos_simplelink_2_11_01_09*
- The BLE-Stack files and all of the example projects are in the following location: *C:\ti\simplelink\ble_cc26xx_2_00_00_42893*

Download the CCS Version 6 software here: http://www.ti.com/tool/ccstudio.

Download the BLE-Stack software here: http://www.ti.com/tool/ble-stack.

4 Getting Started Hardware

4.1 *TIDM-3OUTSMTSTRP Hardware Modification*

The TIDM-3OUTSMTSTRP device requires a minor modification to allow more current to supply to the 3.3-V rail.

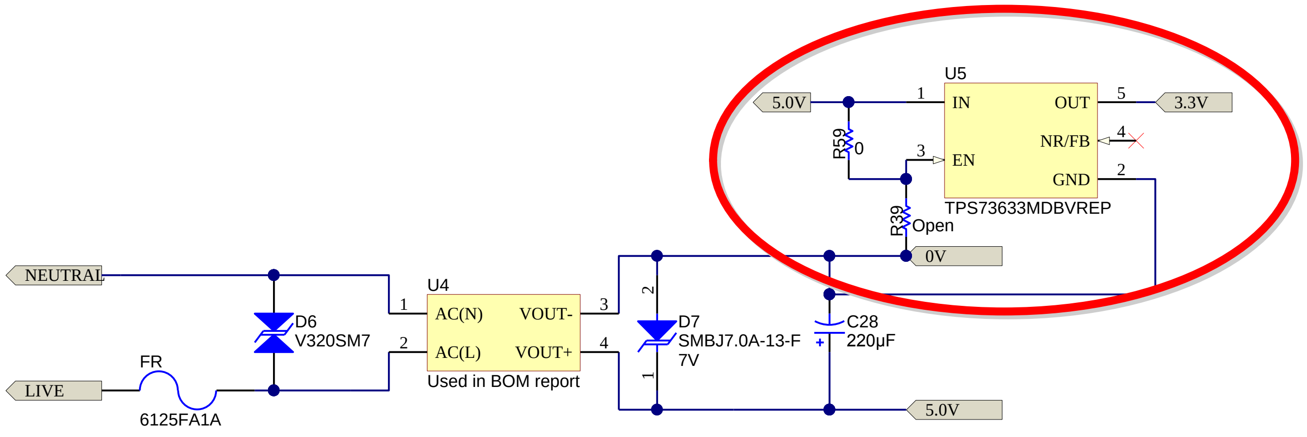


Figure 11. TIDM-3OUTSMTSTRP Hardware Modification

The LDO must be changed to TPS73633DBVREP. As this LDO is high-enabled, the 0-Ω resistor R39 must be opened and R59 must be added to connect pin 3 to 5.0 V, instead

4.2 *Connecting Sensor-Tag to TIDM-3OUTSMTSTRP*

Before connecting the units, the user must remove the battery from the SensorTag, as this application does not require a battery to operate. After removing the battery, place the SensorTag hardware back into its plastic case and then connect it to the CC-DEVPACK-DEBUG device, as the TI Debugger DevPack Quick Start Guide instructs [5].

Connecting the SensorTag hardware to the TIDM-3OUTSMTSTRP device is simple and only requires to connect the VDD, GND, TXD, and RXD pins; however, this may require some soldering skill.

The following diagram in Figure 12 shows the connection of the two boards.

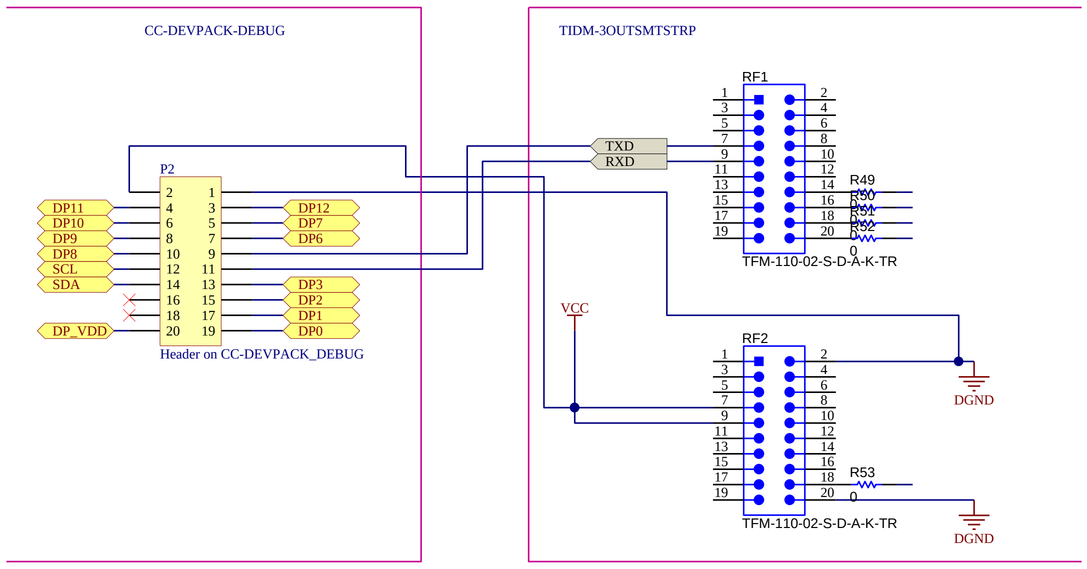


Figure 12. Connections Between SensorTag and CC-EVPACK-DEBUG to TIDM-3OUTSMTSTRP

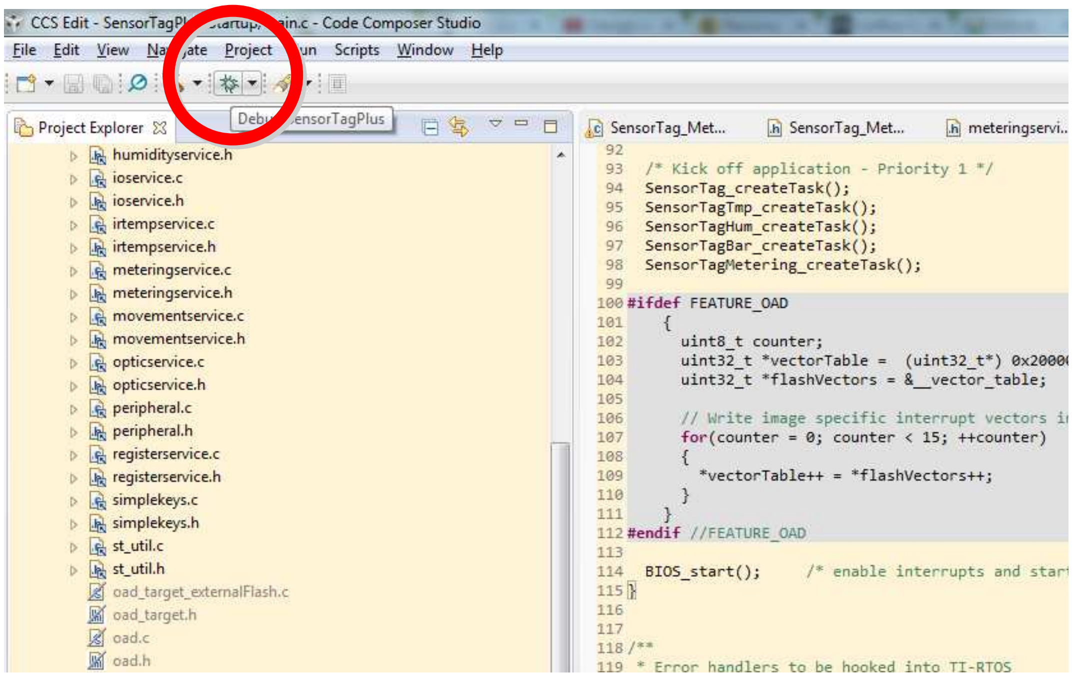
5 Getting Started Firmware

After connecting the hardware, the next step is to download the firmware to the corresponding hardware.

5.1 Loading Firmware to CC2650 Sensor-Tag

To download the firmware:

1. Connect the SensorTag hardware to the CC-DEVPACK-DEBUG, as the TI Debugger DevPack Quick Start Guide instructs [5].
2. With CCS Version 6 or above installed on the PC, connect the CC-DEVPACK-DEBUG's micro USB connector a USB port.
3. Select the "SensorTagStack" project, click on the debug button, and wait for the firmware code to download completely.
4. Select the "SensorTag" project that contains all of the additions and changes made for the *Bluetooth* LE meter reading application. Click on the debug button (see Figure 13) and wait for the firmware code to download completely.
5. Disconnect the CC-DEVPACK-DEBUG device from the USB connection.



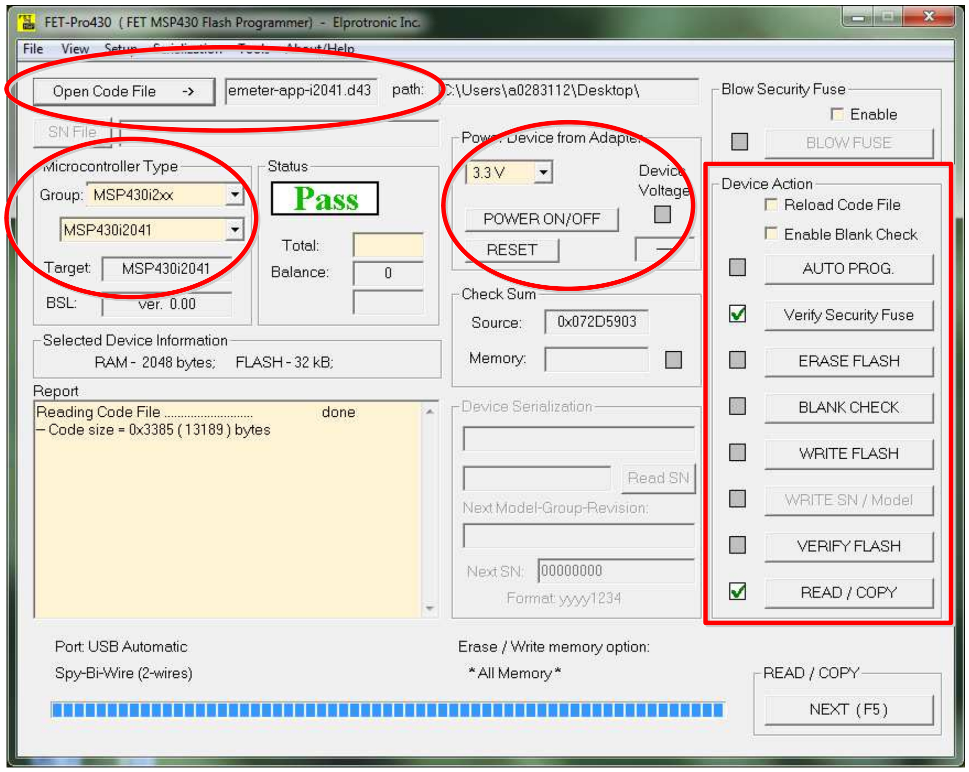
```
  /* Kick off application - Priority 1 */
  SensorTag_createTask();
  SensorTagTmp_createTask();
  SensorTagHum_createTask();
  SensorTagBar_createTask();
  SensorTagMetering_createTask();

#ifdef FEATURE_OAD
  {
    uint8_t counter;
    uint32_t *vectorTable =  (uint32_t*) 0x2000
    uint32_t *flashVectors = &__vector_table;

    // Write image specific interrupt vectors i
    for(counter = 0; counter < 15; ++counter)
    {
      *vectorTable++ = *flashVectors++;
    }
  }
#endif //FEATURE_OAD

  BIOS_start();     /* enable interrupts and star
}

/**
 * Error handlers to be hooked into TI-RTOS
```

Figure 13. Debug Button

5.2 Loading Firmware to TIDM-3OUTSMTSTRP

Perform the following steps to download the firmware to the TIDM-3OUTSMTSTRP hardware. The hardware must use the USB debugging interface MSP-FET430UIF to perform the download

1. Temporarily disconnect VDD from the SensorTag, as this is required to program the TIDM-3OUTSMTSTRP device
2. Connect the MSP-FET430UIF to P1 of the TIDM-3OUTSMTSTRP device and short positions 1–2 on J8 with a jumper (C21 may require temporary removal for the programming)
3. Skip to Step 7 if the Elprotronic FET-Pro-430 Lite software is installed on the PC
4. Visit https://www.elprotronic.com/productdata to download the FET-Pro-430 Lite software
5. Unzip the downloaded .zip file and run the *setup.exe* executable file
6. Follow the prompted instructions to complete the install
7. Launch the FET-Pro-430 Lite software
8. Proceed to update the firmware if prompted for an MSP-FET430UIF firmware update
9. Click the "*Open Code File*" button and select “*emeter-app-i2041.d43*” inside the install folder of this design
10. Check the settings in the boxed-fields named *Microcontroller Type*, *Power Device from Adapter*, and *Device Action* (see Figure 14)


FET-Pro430 (FET MSP430 Flash Programmer) - Elprotronic Inc.

Open Code File -> emeter-app-i2041.d43 path: C:\Users\a0283112\Desktop\

Microcontroller Type
Group: MSP430i2xx
MSP430i2041
Target: MSP430i2041
BSL: ver. 0.00

Status: Pass
Total:
Balance: 0

Power Device from Adapter: 3.3 V

Check Sum
Source: 0x072D5903
Memory:

Selected Device Information
RAM - 2048 bytes; FLASH - 32 kB;

Report
Reading Code File .... done
– Code size = 0x3385 (13189) bytes

Device Serialization
Next SN: 00000000
Format: yyyy1234

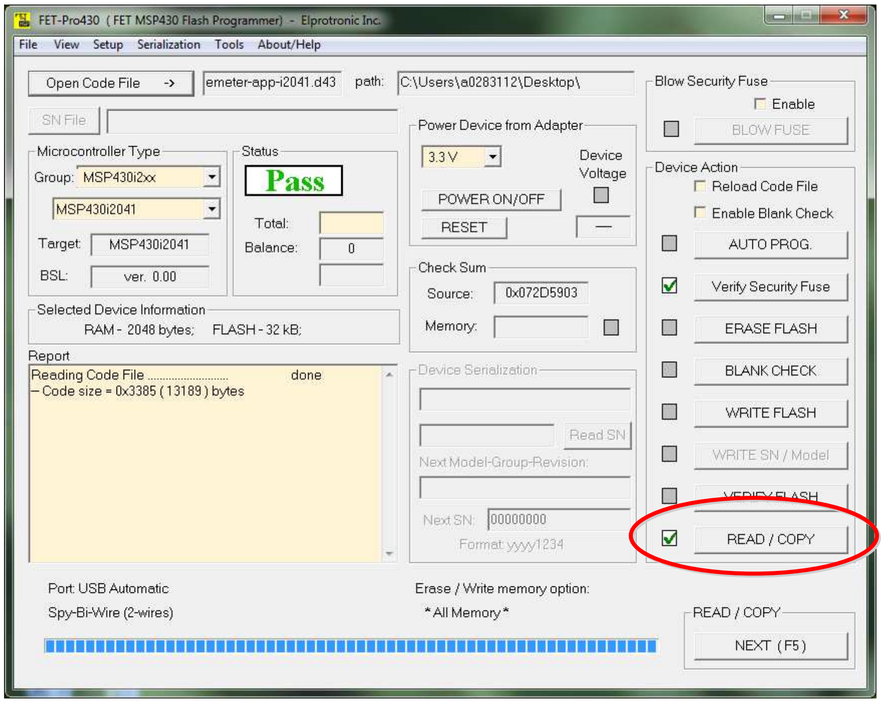
Device Action
☐ Reload Code File
☐ Enable Blank Check
☐ AUTO PROG.
☑ Verify Security Fuse
☐ ERASE FLASH
☐ BLANK CHECK
☐ WRITE FLASH
☐ WRITE SN / Model
☐ VERIFY FLASH
☑ READ / COPY

Blow Security Fuse: ☐ Enable

Port: USB Automatic
Spy-Bi-Wire (2-wires)
Erase / Write memory option: * All Memory *
READ / COPY

Figure 14. TIDM-3OUTSMTSTRP Programming Setup

11. Click the "*READ/COPY*" button to copy the original code and when *Flash Memory Data* window appears (see Figure 16), click the "*Copy*" button, paste the data to an open note pad application, and save it as a text file named “*original.txt*”. The creation of this text file is to restore to the original in the case of errors.



emeter-app-i2041.d43 path: C:\Users\a0283112\Desktop\

Microcontroller Type
Group: MSP430i2xx
MSP430i2041
Target: MSP430i2041
BSL: ver. 0.00

Status: Pass
Total:
Balance: 0

3.3 V

Check Sum
Source: 0x072D5903
Memory:

Selected Device Information
RAM - 2048 bytes; FLASH - 32 kB;

Report
```
Reading Code File ........................        done
-- Code size = 0x3385 ( 13189 ) bytes
```

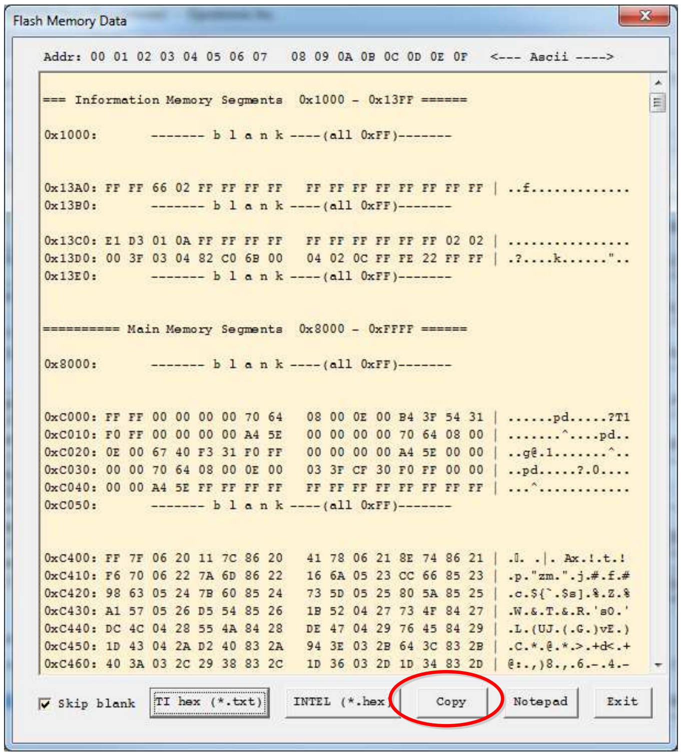
Next Model-Group-Revision:
Next SN: 00000000
Format: yyyy1234

Port: USB Automatic
Spy-Bi-Wire (2-wires)

Erase / Write memory option:
* All Memory *

READ / COPY

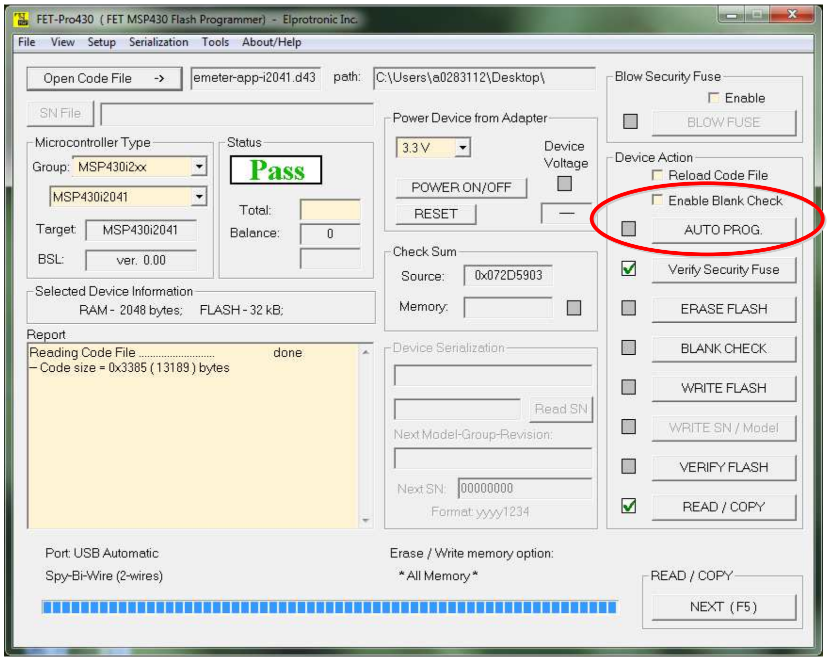
Figure 15. Read Original Firmware



```
Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F   <--- Ascii ---->

=== Information Memory Segments  0x1000 - 0x13FF ======

0x1000:       ------- b l a n k ----(all 0xFF)-------


0x13A0: FF FF 66 02 FF FF FF FF   FF FF FF FF FF FF FF FF | ..f.............
0x13B0:       ------- b l a n k ----(all 0xFF)-------

0x13C0: E1 D3 01 0A FF FF FF FF   FF FF FF FF FF FF 02 02 | ................
0x13D0: 00 3F 03 04 82 C0 6B 00   04 02 0C FF FE 22 FF FF | .?....k......"..
0x13E0:       ------- b l a n k ----(all 0xFF)-------


========== Main Memory Segments  0x8000 - 0xFFFF ======

0x8000:       ------- b l a n k ----(all 0xFF)-------


0xC000: FF FF 00 00 00 00 70 64   08 00 0E 00 B4 3F 54 31 | ......pd.....?T1
0xC010: F0 FF 00 00 00 00 A4 5E   00 00 00 00 70 64 08 00 | .......^....pd..
0xC020: 0E 00 67 40 F3 31 F0 FF   00 00 00 00 A4 5E 00 00 | ..g@.1.......^..
0xC030: 00 00 70 64 08 00 0E 00   03 3F CF 30 F0 FF 00 00 | ..pd.....?.0....
0xC040: 00 00 A4 5E FF FF FF FF   FF FF FF FF FF FF FF FF | ...^............
0xC050:       ------- b l a n k ----(all 0xFF)-------


0xC400: FF 7F 06 20 11 7C 86 20   41 78 06 21 8E 74 86 21 | .. .|. Ax.!.t.!
0xC410: F6 70 06 22 7A 6D 86 22   16 6A 05 23 CC 66 85 23 | .p."zm.".j.#.f.#
0xC420: 98 63 05 24 7B 60 85 24   73 5D 05 25 80 5A 85 25 | .c.${`.$s].%.Z.%
0xC430: A1 57 05 26 D5 54 85 26   1B 52 04 27 73 4F 84 27 | .W.&.T.&.R.'sO.'
0xC440: DC 4C 04 28 55 4A 84 28   DE 47 04 29 76 45 84 29 | .L.(UJ.(.G.)vE.)
0xC450: 1D 43 04 2A D2 40 83 2A   94 3E 03 2B 64 3C 83 2B | .C.*.@.*.>.+d<.+
0xC460: 40 3A 03 2C 29 38 83 2C   1D 36 03 2D 1D 34 83 2D | @:.,)8.,.6.-.4.-
```

Figure 16. Copy and Save Original Firmware

12. Close the *Flash Memory Data* window and click the "*AUTO PROG*" button (see Figure 17)



Code file: emeter-app-i2041.d43 path: C:\Users\a0283112\Desktop\

Microcontroller Type — Group: MSP430i2xx; MSP430i2041; Target: MSP430i2041; BSL: ver. 0.00

Status: Pass; Total: ; Balance: 0

Power Device from Adapter: 3.3 V; Device Voltage: —

Check Sum — Source: 0x072D5903; Memory:

Selected Device Information: RAM - 2048 bytes; FLASH - 32 kB;

Device Serialization — Next Model-Group-Revision: ; Next SN: 00000000; Format: yyyy1234

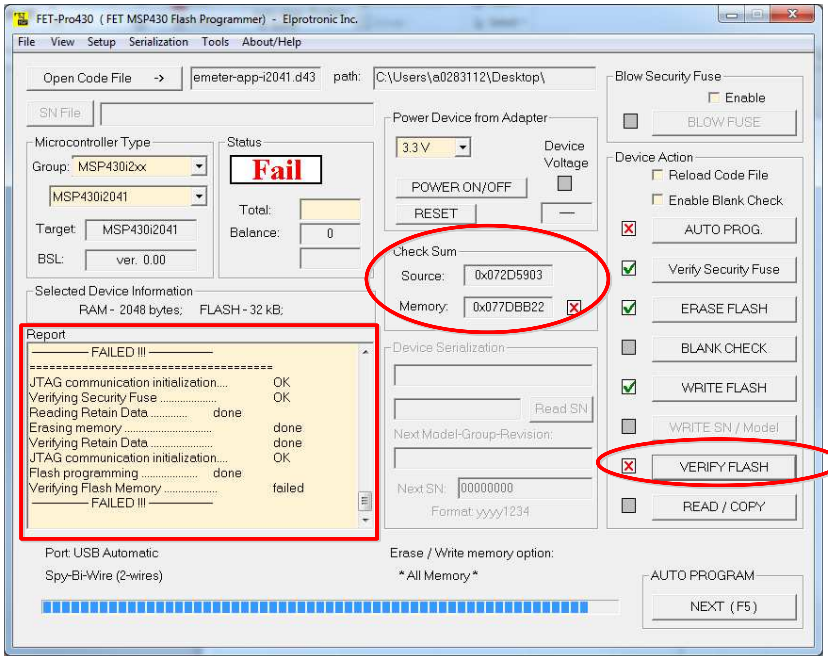
Report:

```
Reading Code File .........................        done
-- Code size = 0x3385 ( 13189 ) bytes
```

Port: USB Automatic
Spy-Bi-Wire (2-wires)
Erase / Write memory option: * All Memory *
READ / COPY

Figure 17. Start Firmware Programming

13. The software prompts a failure to verify (still troubleshooting this issue at the time of this writing); however, check that everything verifies an "OK" status except for "*Verifying Flash Memory*" (see Figure 18)



Code file: emeter-app-i2041.d43 path: C:\Users\a0283112\Desktop\

Microcontroller Type — Group: MSP430i2xx; MSP430i2041; Target: MSP430i2041; BSL: ver. 0.00

Status: Fail; Total: ; Balance: 0

Power Device from Adapter: 3.3 V; Device Voltage: —

Check Sum — Source: 0x072D5903; Memory: 0x077DBB22

Selected Device Information: RAM - 2048 bytes; FLASH - 32 kB;

Device Serialization — Next Model-Group-Revision: ; Next SN: 00000000; Format: yyyy1234

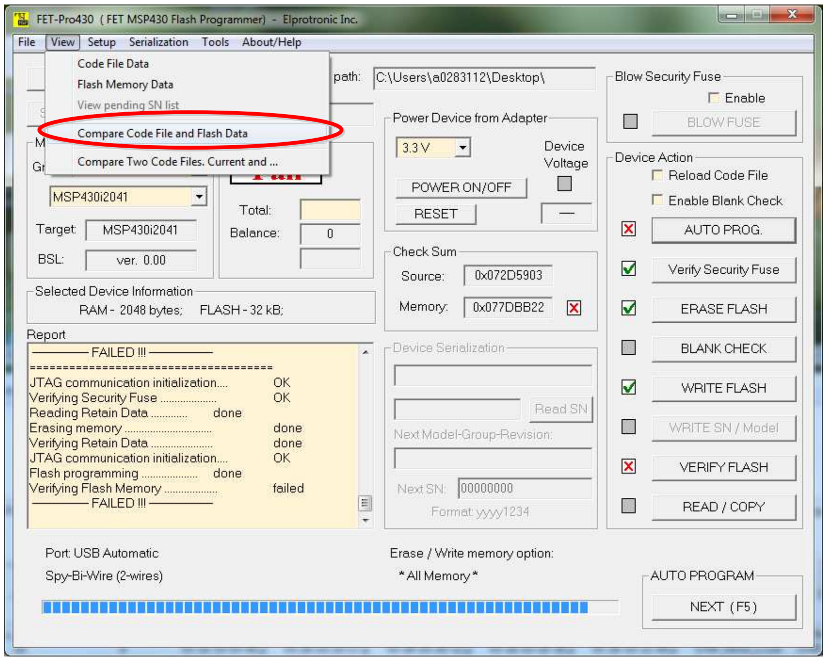
Report:

```
---------------- FAILED !!! ----------------
=======================================
JTAG communication initialization....     OK
Verifying Security Fuse ..................     OK
Reading Retain Data ............     done
Erasing memory ..............................     done
Verifying Retain Data ...................     done
JTAG communication initialization....     OK
Flash programming ..................     done
Verifying Flash Memory .................     failed
---------------- FAILED !!! ----------------
```

Port: USB Automatic
Spy-Bi-Wire (2-wires)
Erase / Write memory option: * All Memory *
AUTO PROGRAM

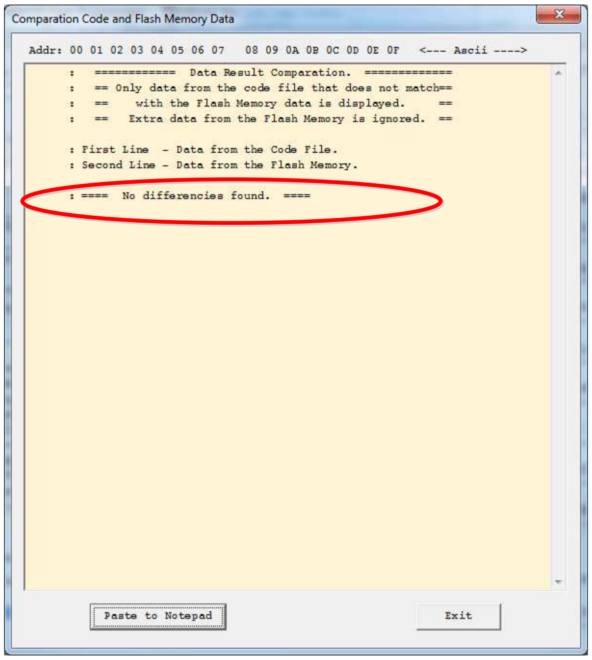
Figure 18. Note Failure on VERIFY and Confirm *Check Sum*

14. Click on *View→ Compare Code File and Flash Data*. Select "Yes" when prompted to read the data from the device first. After running the execution, **check the last line of the** "***Comparation Code and Flash Memory Data***" window (see Figure 20) for the message "==== No differencies found. ====" [sic].



```
path: C:\Users\a0283112\Desktop\
3.3 V
MSP430i2041
Target: MSP430i2041
BSL: ver. 0.00
Total:
Balance: 0
Check Sum
Source: 0x072D5903
Memory: 0x077DBB22
Selected Device Information
RAM - 2048 bytes; FLASH - 32 kB;
Next SN: 00000000
Format: yyyy1234
Report
---------- FAILED !!! ----------
========================================
JTAG communication initialization....      OK
Verifying Security Fuse ..................  OK
Reading Retain Data ...........       done
Erasing memory ...........................  done
Verifying Retain Data ....................  done
JTAG communication initialization....      OK
Flash programming ..................  done
Verifying Flash Memory ................    failed
---------- FAILED !!! ----------
Port: USB Automatic
Spy-Bi-Wire (2-wires)
Erase / Write memory option:
* All Memory *
```

Figure 19. Perform Direct Code Comparison to Verify Written Firmware



```
Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F   <--- Ascii ---->
    :   ============  Data Result Comparation.  =============
    :   == Only data from the code file that does not match==
    :   ==    with the Flash Memory data is displayed.     ==
    :   ==   Extra data from the Flash Memory is ignored.  ==

    : First Line  - Data from the Code File.
    : Second Line - Data from the Flash Memory.

    : ====  No differencies found.  ====
```

Figure 20. Code Comparison Report

15. Re-connect the VDD line to the SensorTag board, completing the firmware programming of the TIDM-3OUTSMTSTRP.

After the code is downloaded onto both boards the application is ready to run. As soon as both boards are code-enabled, then download the *BLESensorTag.apk* for Android and install it to an Android device.

6 Test Setup

To perform the test, be sure to first follow the hardware setup and firmware loading steps that Section 5.2 describes. Then apply AC voltage to the AC input of the power strip hardware. The LED's on the TIDM-3OUTSMTSTRP device light up and the LED on the SensorTag flashes.

To start the testing, launch the *BLE SensorTag* application on an Android device. Please note that due to the variety of currently-existing Android devices, this android application has not been tested on every android device. The android application may not run properly on particular devices. The following devices have successfully tested the Android application: Sony Xperia Z Ultra C6833, Samsung Galaxy TabPro 8.4 STM-325, and the Red Mi 2 HM 2LTE-CMCC).

7 Test Data

7.1 Reading Test

Upon launching the Android applications, the *Opening Screen* appears (Figure 21). Click the "*Scan*" button to start scanning for SensorTag devices. When the applications locates a valid SensorTag device, the *Device Selection* window appears (Figure 22). Click the "*Connect*" button for the desirable device and the application starts looking for the service that the SensorTag supports (Figure 23). After discovering the supported service, the application automatically runs. Select the *METERING* tab to view the parameters for the energy monitor (Figure 24).

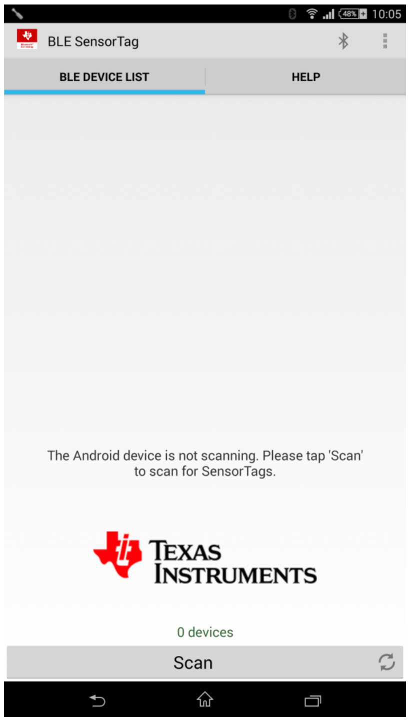

10:05
BLE SensorTag
BLE DEVICE LIST
HELP
The Android device is not scanning. Please tap 'Scan' to scan for SensorTags.
Texas Instruments
0 devices
Scan

Figure 21. Opening Screen

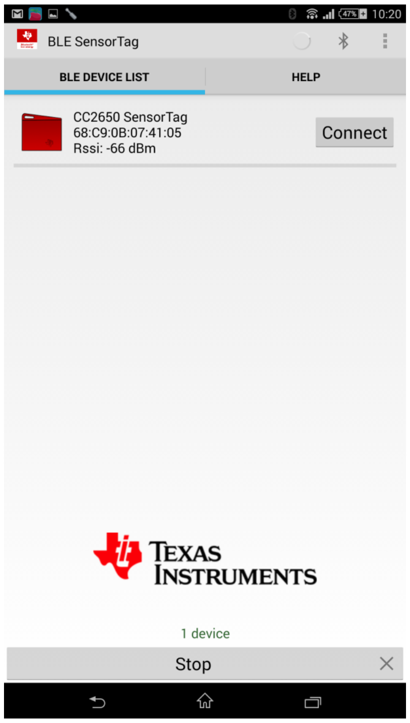
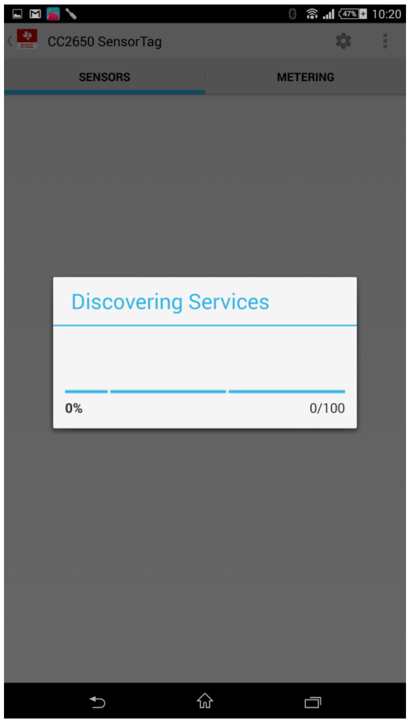


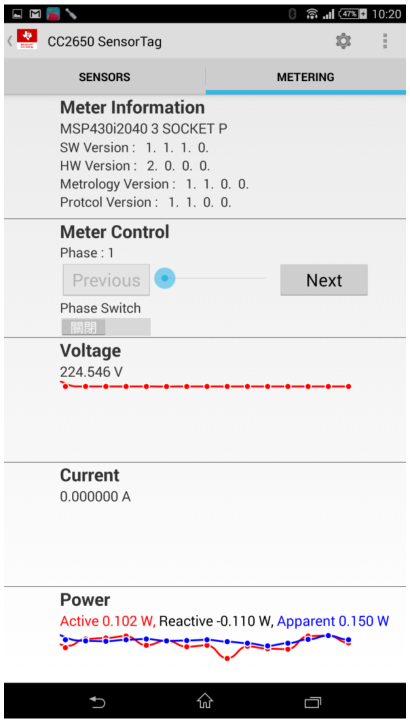
Figure 22. Device Selection



Discovering Services

0% 0/100

Figure 23. Service Discovery



Meter Information
MSP430i2040 3 SOCKET P
SW Version : 1. 1. 1. 0.
HW Version : 2. 0. 0. 0.
Metrology Version : 1. 1. 0. 0.
Protcol Version : 1. 1. 0. 0.

Meter Control
Phase : 1
Phase Switch

Voltage
224.546 V

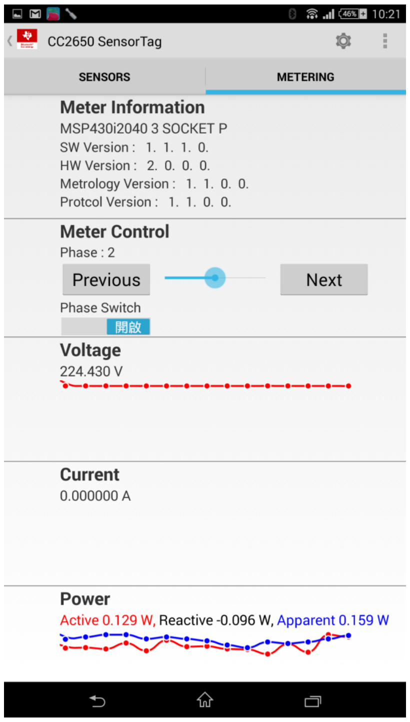
Current
0.000000 A

Power
Active 0.102 W, Reactive -0.110 W, Apparent 0.150 W

Figure 24. Running Screens

7.2 *Controlling Test*

The readings in the *METERING* tab can be set to monitor all three outlets. Click on the "*Next*" or "*Previous*" button to toggle through the readings of different outlets and to control the relays of different outlets (Figure 25).



Meter Information
MSP430i2040 3 SOCKET P
SW Version : 1. 1. 1. 0.
HW Version : 2. 0. 0. 0.
Metrology Version : 1. 1. 0. 0.
Protcol Version : 1. 1. 0. 0.

Meter Control
Phase : 2
Phase Switch

Voltage
224.430 V

Current
0.000000 A

Power
Active 0.129 W, Reactive -0.096 W, Apparent 0.159 W

Figure 25. Application Running With Socket 2 Data

8 Design Files

8.1 Schematics

To download the schematics for the metering board, see the design files at TIDM-3OUTSMTSTRP.

To download the schematics for the SensorTag board, see the design files at TIDC-CC2650STK-SENSORTAG.

8.2 Bill of Materials

To download the bill of materials (BOM) for the metering board, see the design files at TIDM-3OUTSMTSTRP.

To download the bill of materials (BOM) for the SensorTag board, see the design files at TIDC-CC2650STK-SENSORTAG.

9 Distribution Software Files

To download the software files for this reference design, please see the link at TIDC-BLE-METER-READING.

As Figure 26 shows, the software files are distributed using a self-extracting executable file that, by default, installs into the *TIDC-BLE-METER-READING-SOFTWARE* folder on the user's desktop.

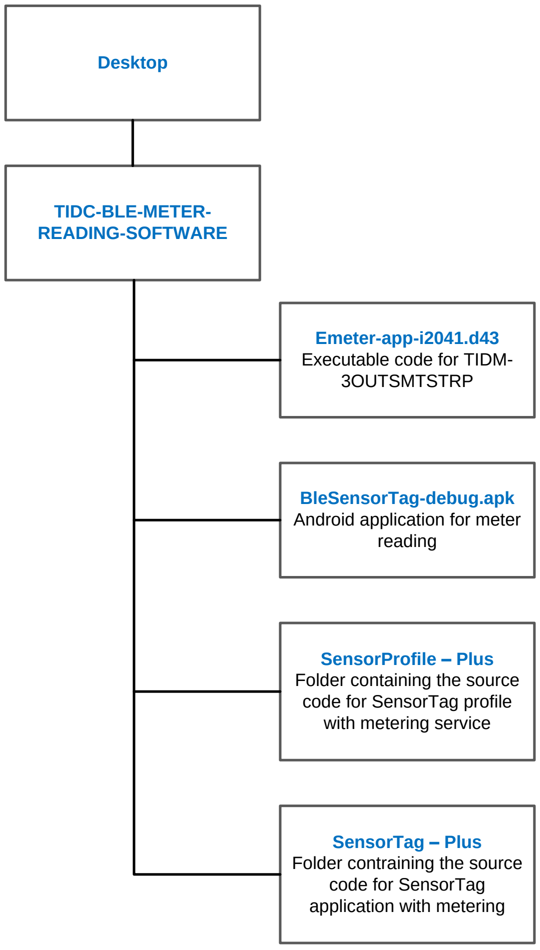

Desktop
TIDC-BLE-METER-READING-SOFTWARE
Emeter-app-i2041.d43
Executable code for TIDM-3OUTSMTSTRP
BleSensorTag-debug.apk
Android application for meter reading
SensorProfile – Plus
Folder containing the source code for SensorTag profile with metering service
SensorTag – Plus
Folder contraining the source code for SensorTag application with metering

Figure 26. Distribution Software

10 References

1. Texas Instruments, *Three-Outlet Smart Power Strip*, TIDM-3OUTSMTSTRP User's Guide, (TIDU453)
2. Texas Instruments, *Multi-Standard CC2650 SensorTag Design Guide*, C-CC2650STK-SENSORTAG User's Guide, (TIDU862)
3. Texas Instruments, *Three-Outlet Smart Power Strip*, CC-DEVPACK-DEBUG User's Guide, (TIDU453)
4. Texas Instruments, *Bringing wireless scalability to intelligent sensing applications*, CC-DEVPACK-DEBUG White Paper, (SWRY014)
5. Texas Instruments, *TI Debugger DevPack Quick Start Guide*, Getting Started Guide, (SWRT023)
6. Texas Instruments, *Category:BluetoothLE*, Texas Instruments Wikipedia Entry, http://processors.wiki.ti.com/index.php/Category:BluetoothLE
7. Bluetooth® Developer Portal, Developer Log-In and Support Forum, https://developer.bluetooth.org/Pages/default.aspx

11 About the Author

MARS LEUNG received his Bachelor of Engineering in Hong Kong Polytechnic University and Master of Science in Chinese University of Hong Kong. Mars is a field application engineer specializing in MCU application support and development; a senior smart-card application engineer specializing in smart-card payment system definitions and implementation; a staff engineer specializing in MCU and new module definitions; and a staff engineer in analog system applications specializing in the digital system and video processing of dynamic LED backlight control. Mars currently works as a staff engineer on the Texas Instruments Smart Grid Application Team, which specializes in embedded electricity metering applications.