



ZigBee Sleepy End Device Lab Worksheet

In this worksheet we provide a step-by-step guide to create, build and run ZigBee 3.0 end device and sleepy end device applications based on EmberZNet Stack 6.6.4. If you use a later release in the future, most of the instructions should still apply, although there could be minor differences not foreseen at the time of this document.

These exercises help you get familiar with ZigBee 3.0 in the EmberZNet Stack, Simplicity Studio v4 development environment, and the Wireless Start Kit (WSTK) with EFR32MG modules. We assume that you have a WSTK and the following software requirements:

- Simplicity Studio 4
- EmberZNet 6.6.4
- GCC 7.2

KEY FEATURES

- Step-by-step guide to creating, building and running ZigBee 3.0 applications based on EmberZNet 6.6.4
- Use Simplicity Studio v4 as the development tool
- ZigBee end device polling
- Zigbee end device keepalive and aging
- Zigbee end device rejoin

1 Pre-requisites

Make sure you have installed the EmberZnet 6.6.4 SDK and GCC toolchain on your PC.

1.1 Check EmberZnet SDK

1. Launch Simplicity Studio v4.
2. “Windows”→”Preference”→”Simplicity Studio”→”SDKs”, make sure “EmberZnet 6.6.4” is installed

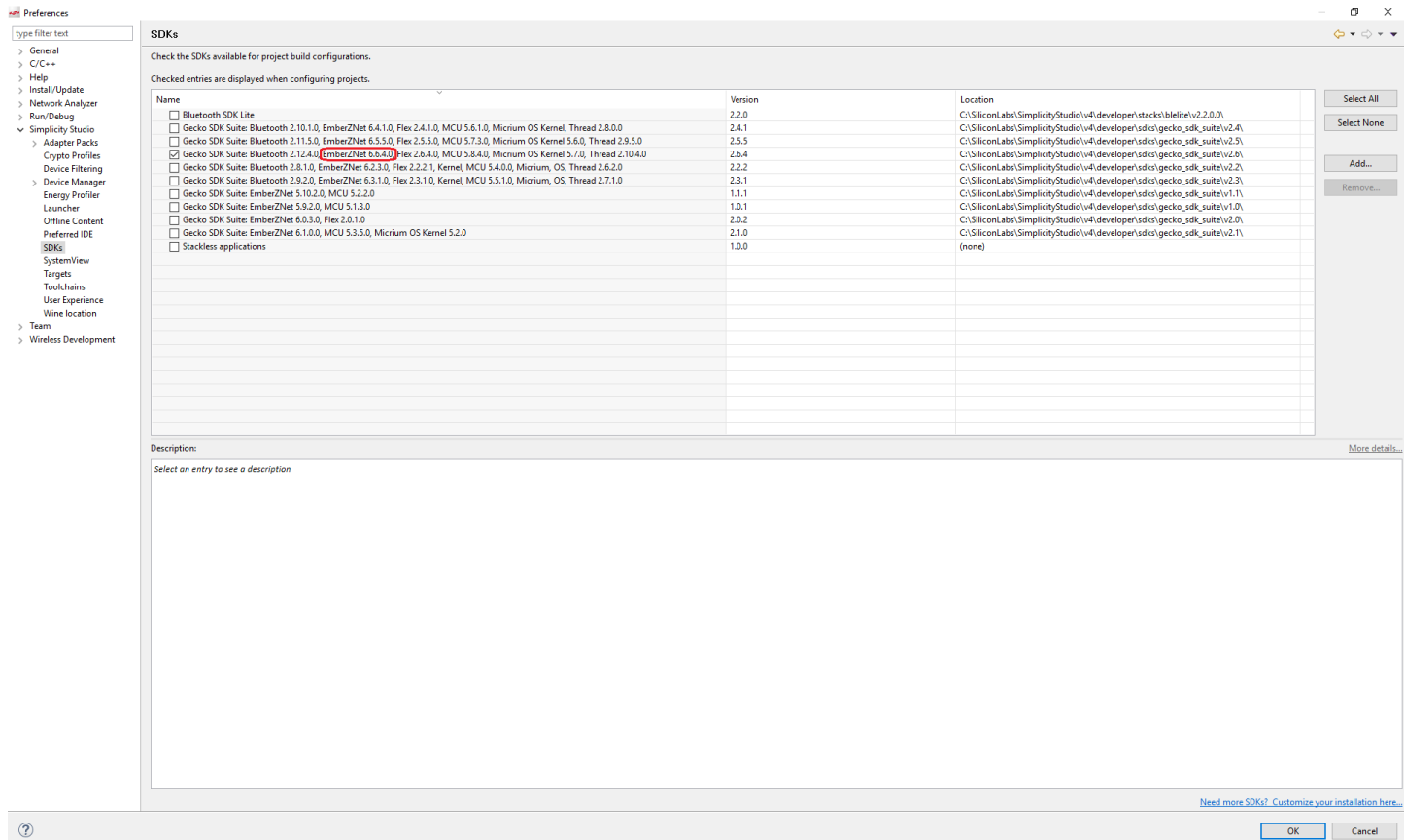


Figure 1 Check SDK in Simplicity Studio

1.2 Check Toolchains

1. Launch Simplicity Studio v4.
2. “Windows”→”Preference”→”Simplicity Studio”→”Toolchains”, make sure GCC toolchain is installed.

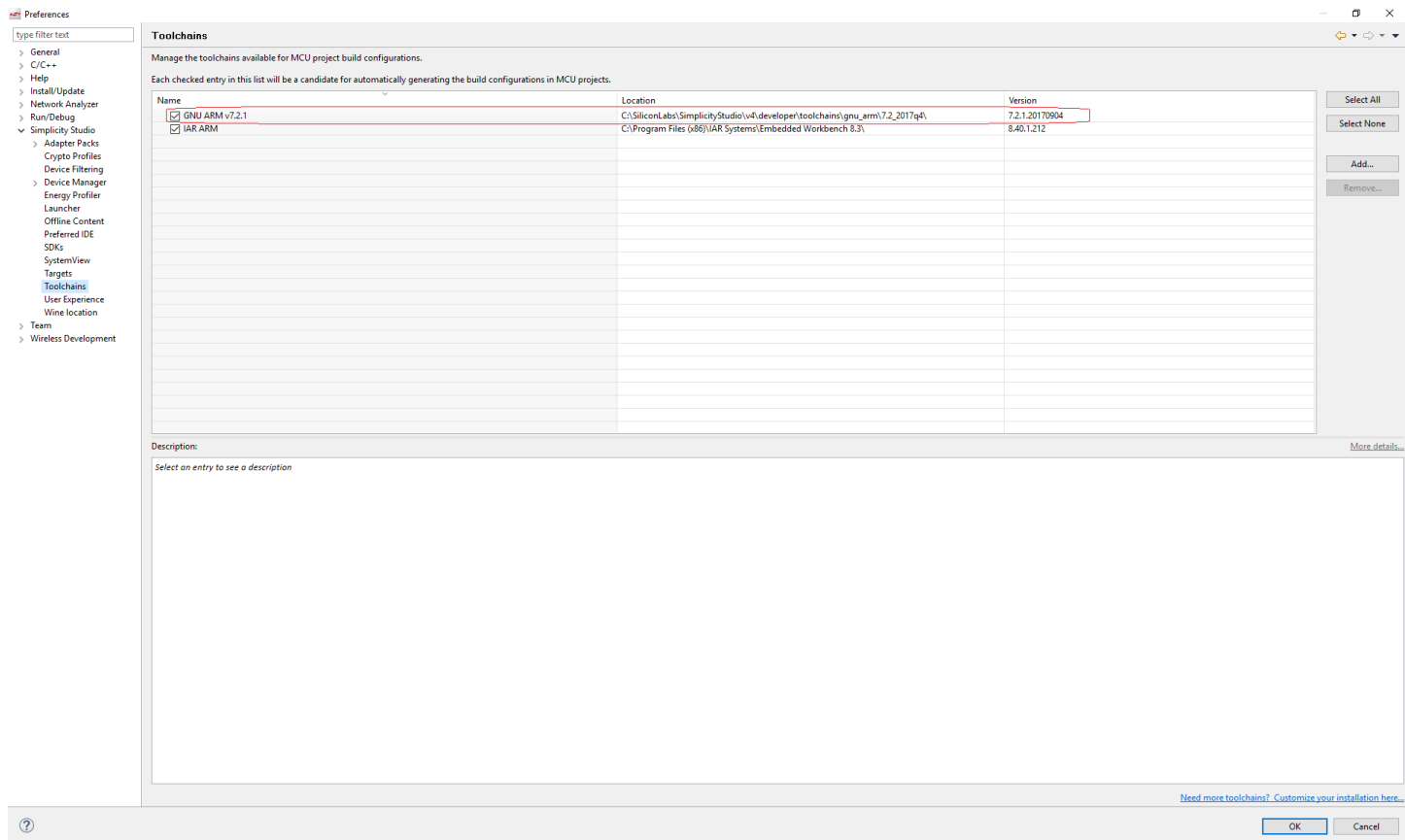



Figure 2 Check Toolchain in Simplicity Studio

1.3 Coordinator

We will provide a coordinator here and all trainee's devices can join this coordinator.

1.4 How to flash the program

1. Start Simplicity Studio, then connect your device to PC;
2. In the menu bar, find the icon  for "Flash Programmer", press it;
3. In the popup window, select the device;

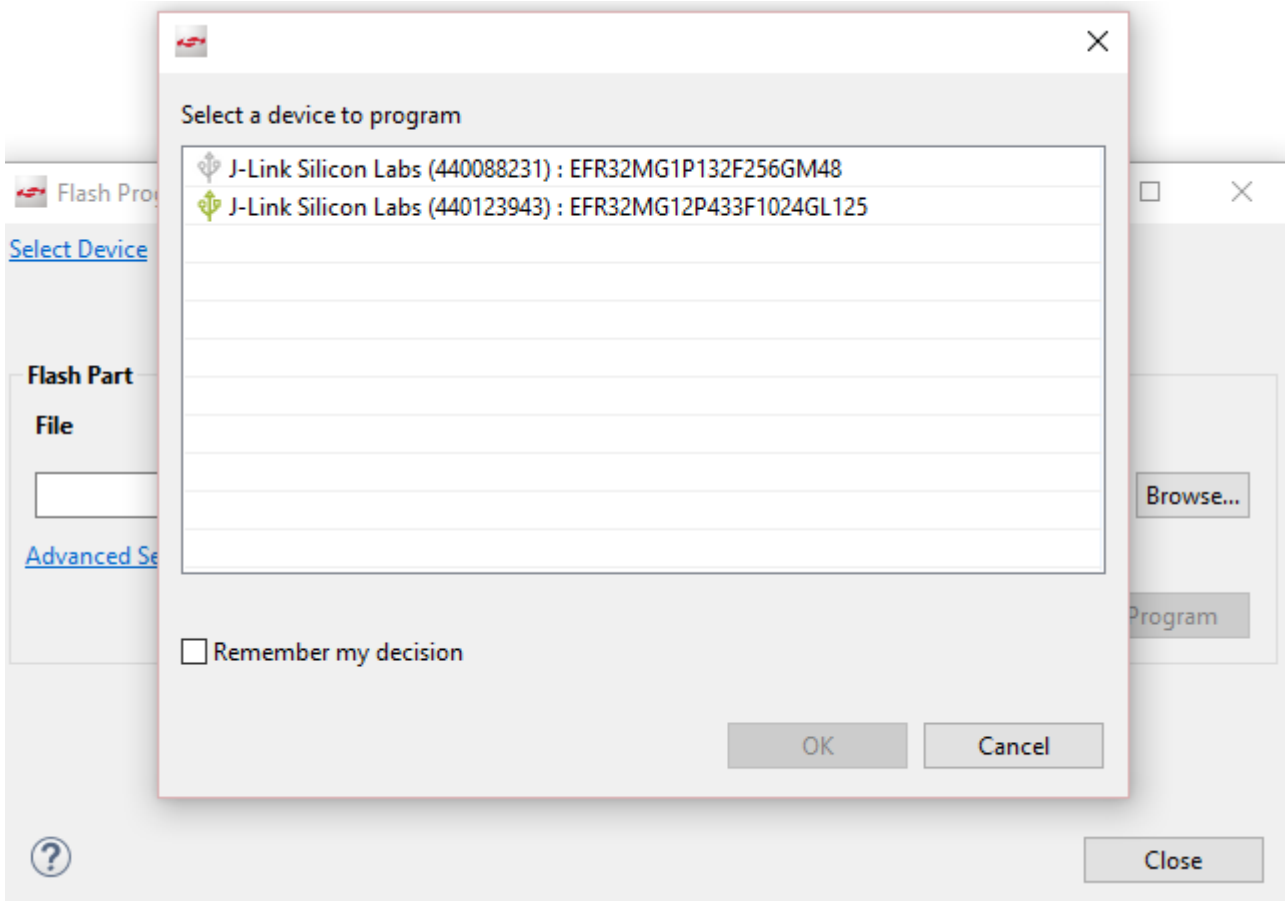


Figure 3 Select device

4. Then in the next window, click “browse” to select the generated image (.s37 or .hex) of your project, press “Program”. You can also press “erase” if needed. Normally you just need to erase the device once before when you start the hands-on. The generated image is in the binary folder of your project. You can select it in “Project Explorer”.

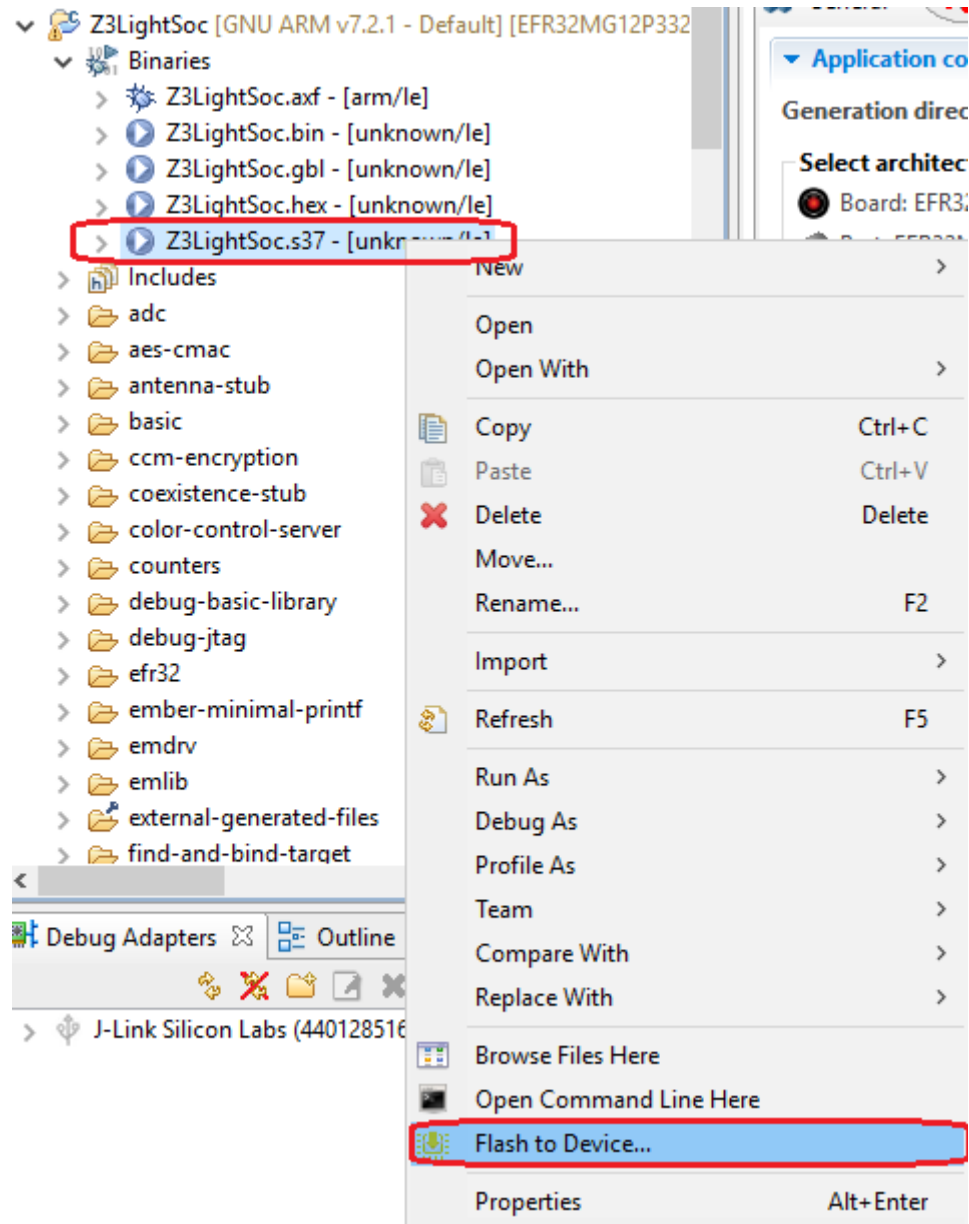


Figure 4 Select image

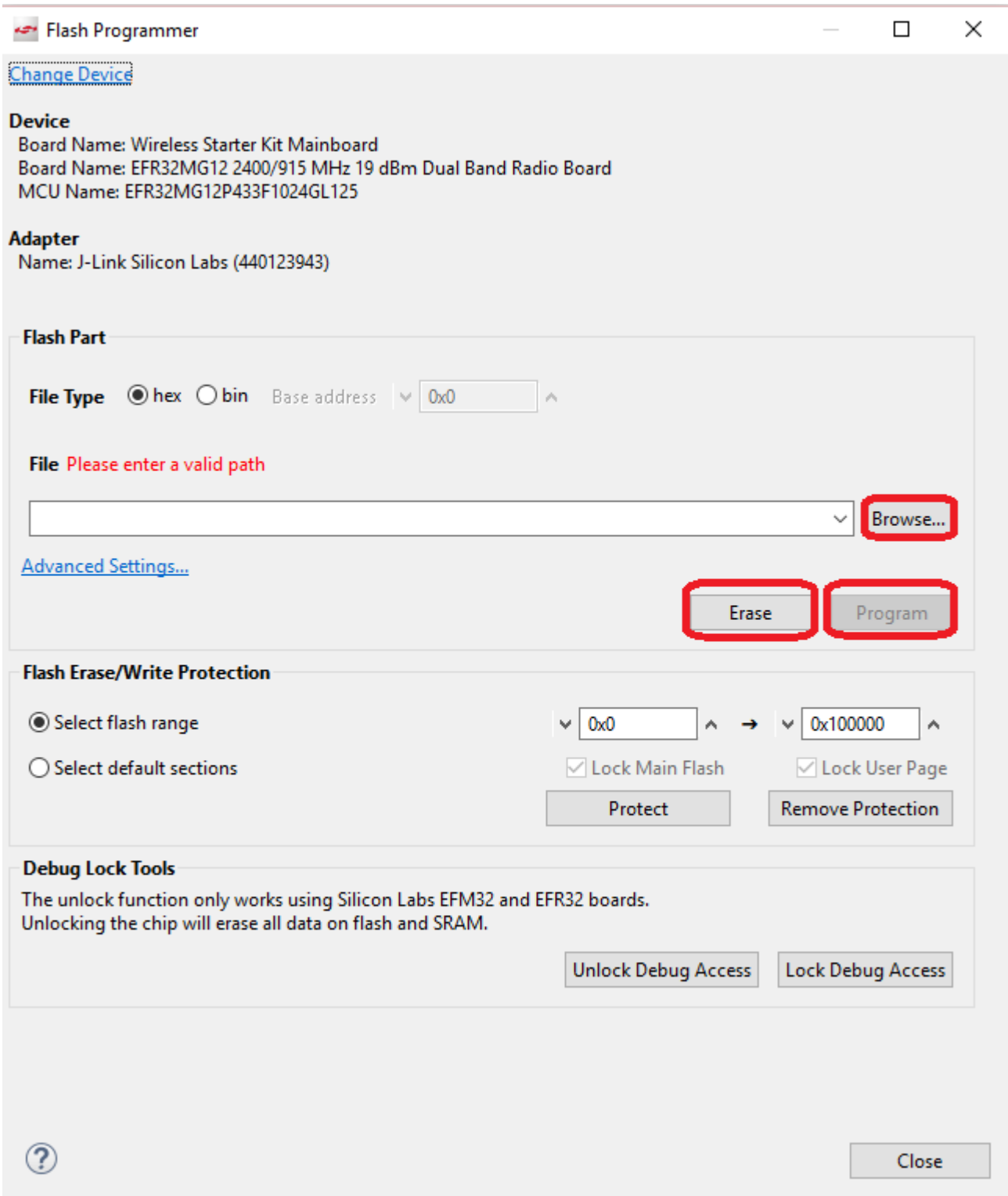
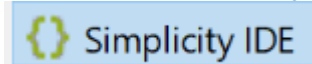


Figure 5 Flash application

1.5 How to open console

Simplicity Studio has integrated a console so that it's convenient to debug through console. To use the console, you need:

1. Change to "Simplicity IDE" perspective:
2. Select your adapter in the "Debug Adapters" window, right click and select "connect";



3. Select your adapter in the “Debug Adapters” window, right click and select “Launch console”;

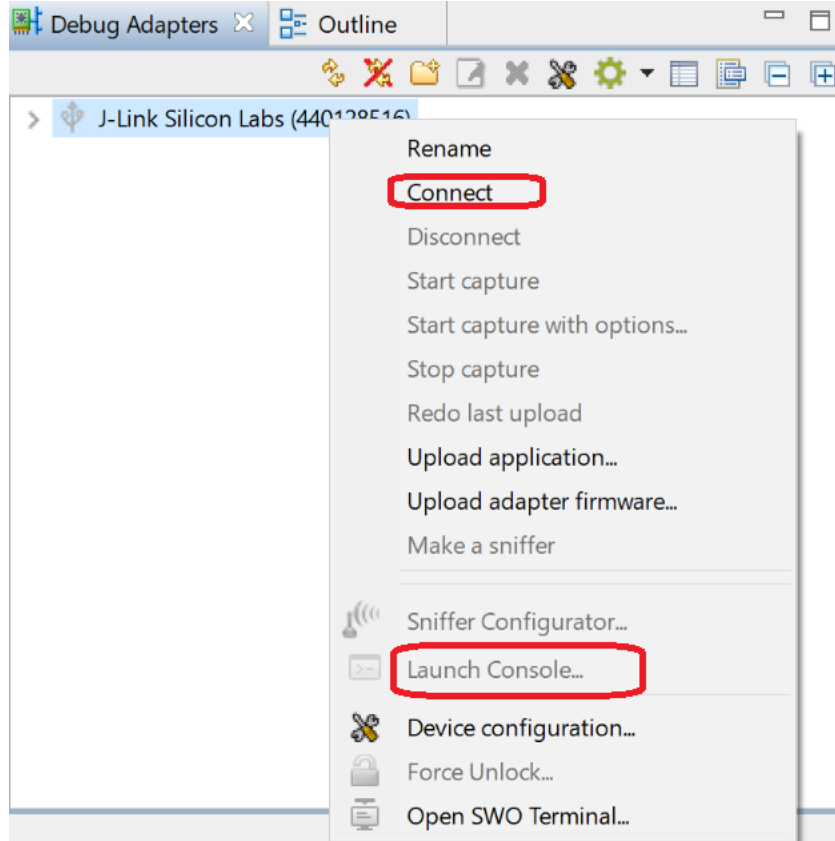
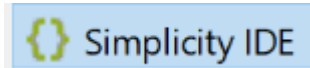


Figure 6 Launch console

1.6 How to start Capture



1. Change to “Simplicity IDE” perspective
2. Select your adapter in the “Debug Adapters” window, right click and select “connect”;
3. Select your adapter in the “Debug Adapters” window, right click and select “Start Capture”;

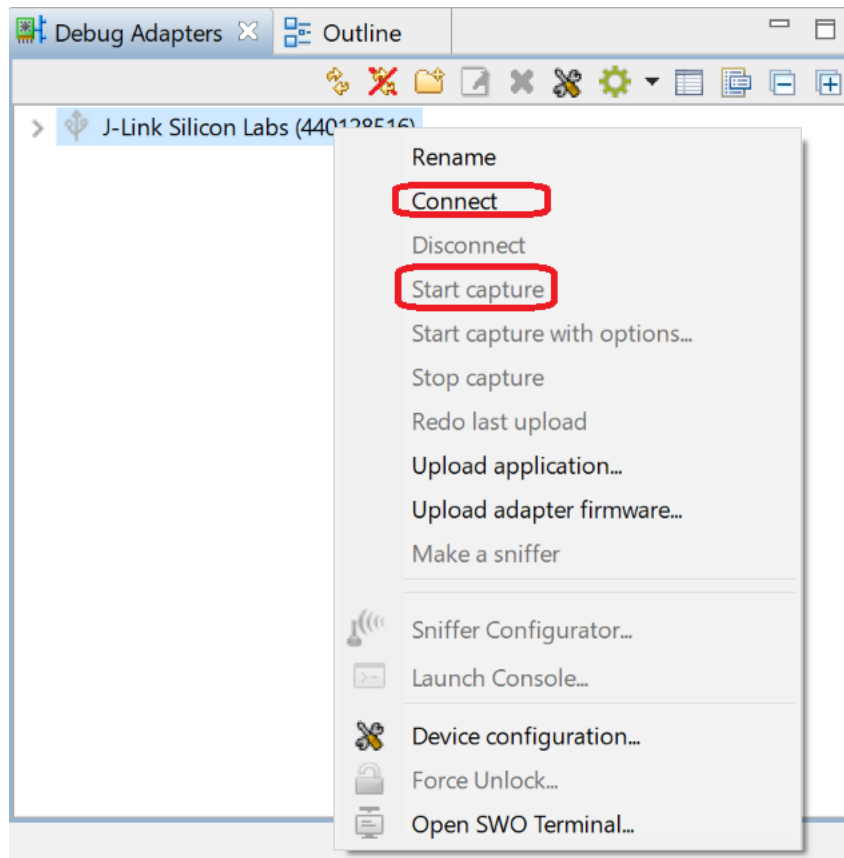


Figure 7 Start Capture

1.7 How to start Energy Profiler

1. Start Energy Profiler:

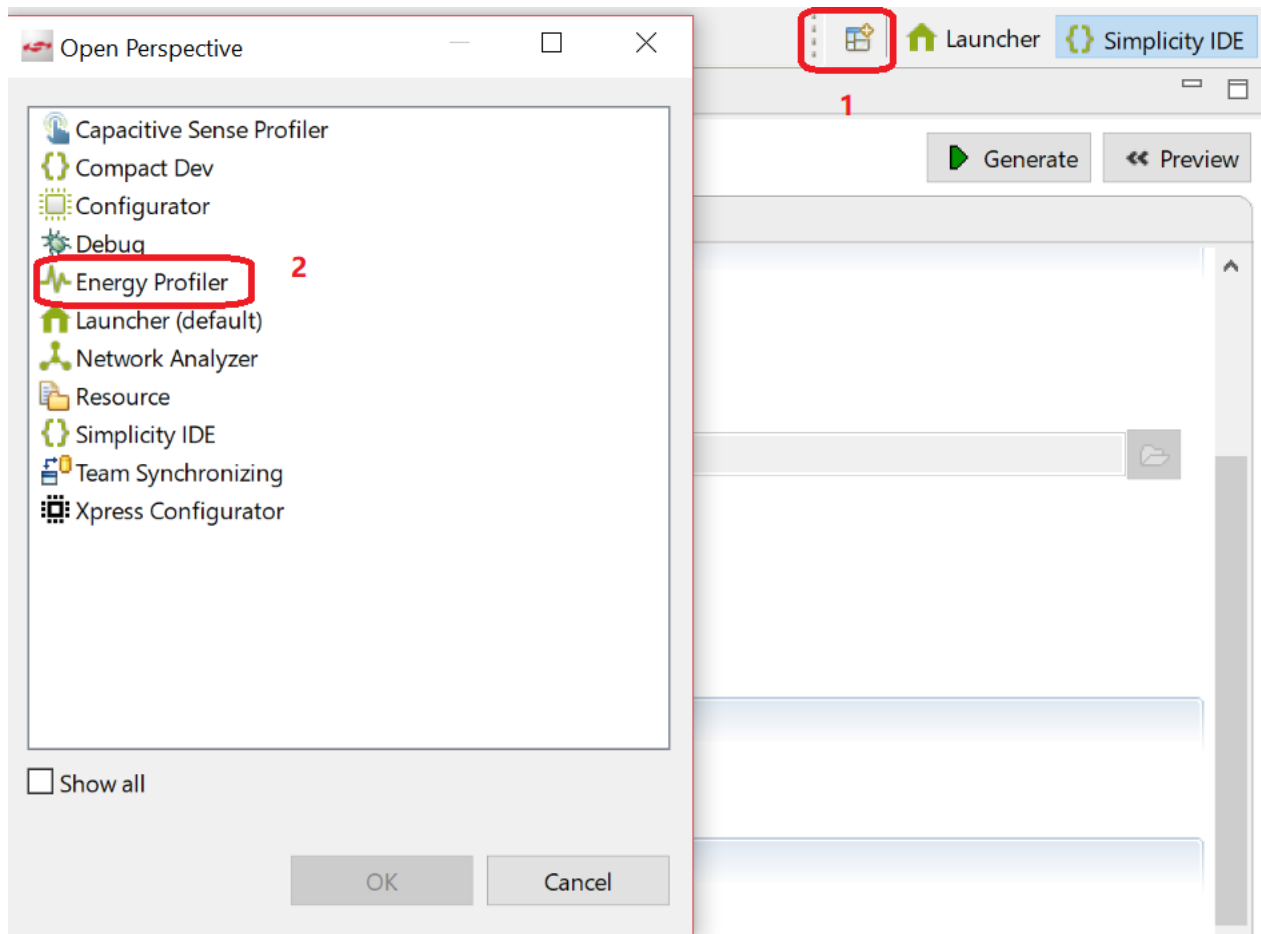


Figure 8 Start Energy Profiler

2. In the tool, on the left top, select "Quick Access", then select "Start Energy Profiler Capture";

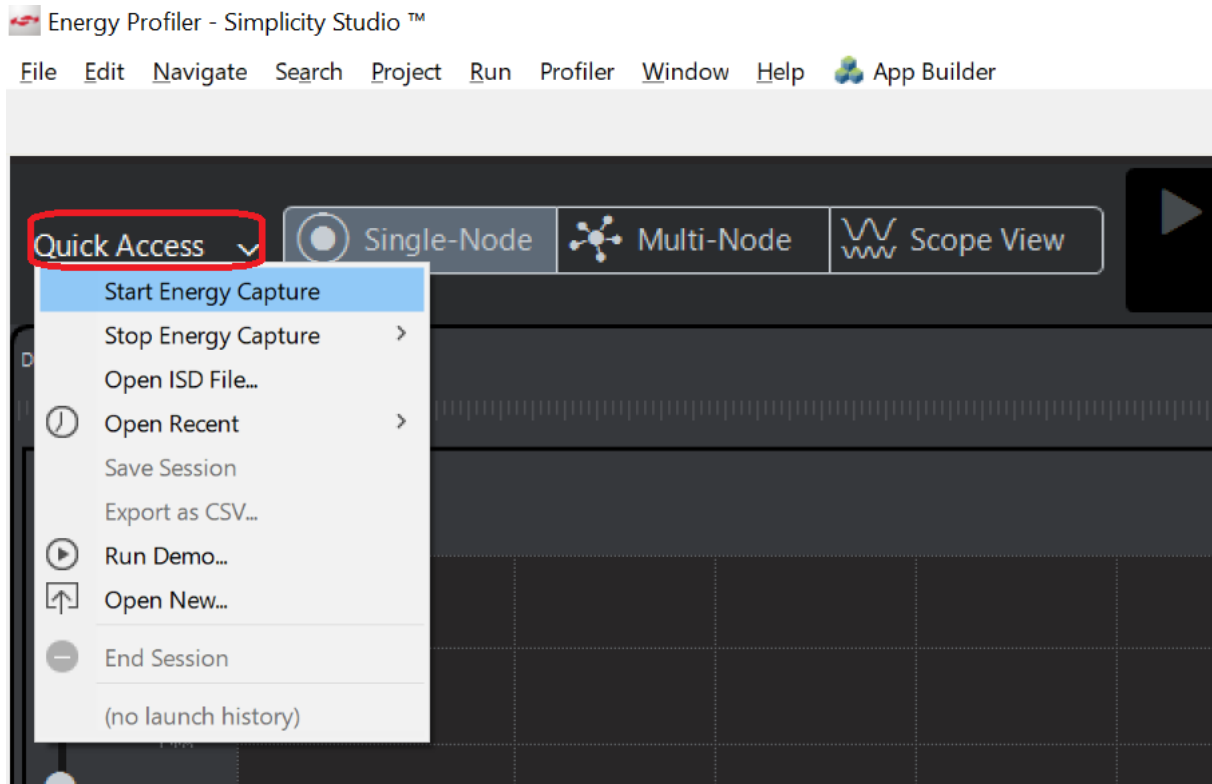


Figure 9 Start Energy Profiler Capture

3. Select your adapter;
4. When the debugger is connected (Normally if you just programmed a new image, the debugger will keep connected), the sleep current will be much higher. You need to unplug/plug the cable and then measure again.

2 Build the ZigBee sleepy end device

1. Go to File -> New -> Project. This will bring up the New Project Wizard
2. Select "Silicon Labs AppBuilder Project". Click Next.
3. Select "Silicon Labs Zigbee". Click Next.
4. Select our latest EmberZNet stack for SoC (in this case EmberZNet 6.6.4 GA SoC). Click Next.
5. On the bottom, select "ZigbeeMinimal". Click Next.
6. Name your project, such as "MyZSED". Click Next.
7. In next window (Project Setup), select board to BRD4162A, and compiler to "GCC v7.2" (If you don't have it, please install any other). Click Finish.

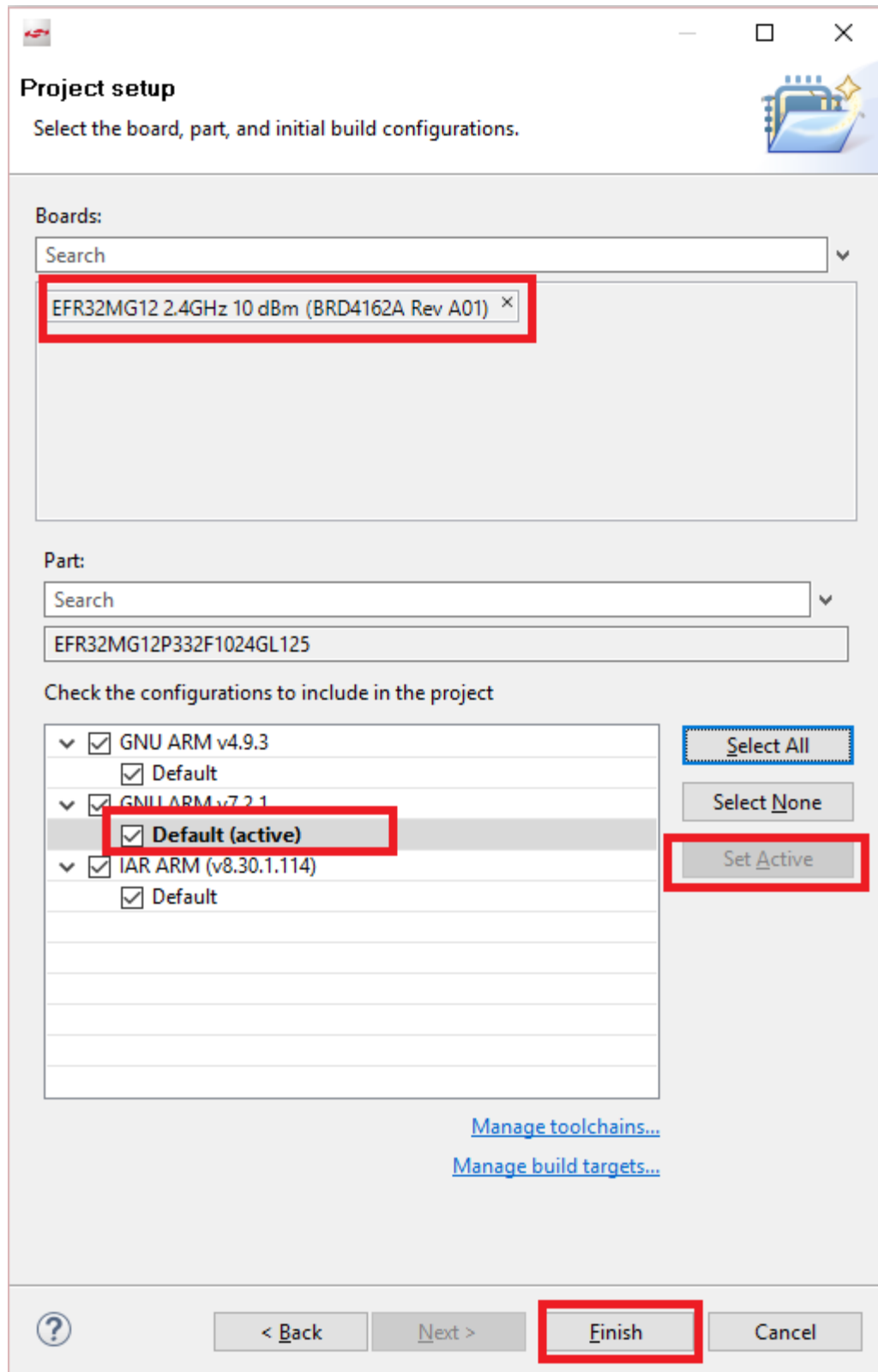


Figure 10 select board and compiler

8. The new project should have been created now, with the project configuration file (an .isc file) open.
9. Click "Zigbee Stack" tab, select "Zigbee Device Type" to "Sleepy End Device".

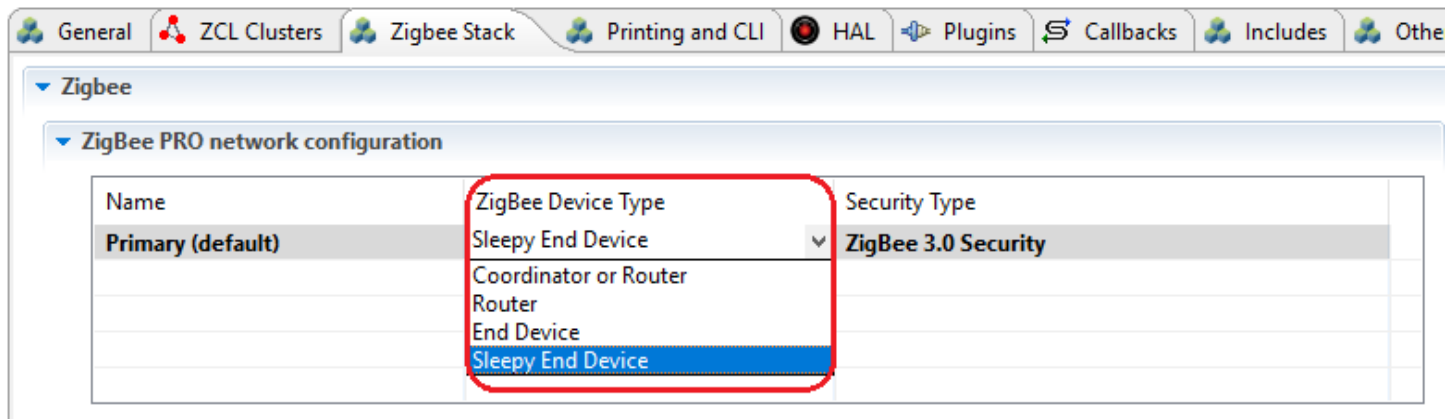


Figure 11 Select Zigbee device type

10. Click "ZCL clusters" tab,
 - a. In "ZCL device type" field, set "ZCL device type" to "HA Temperature Sensor"

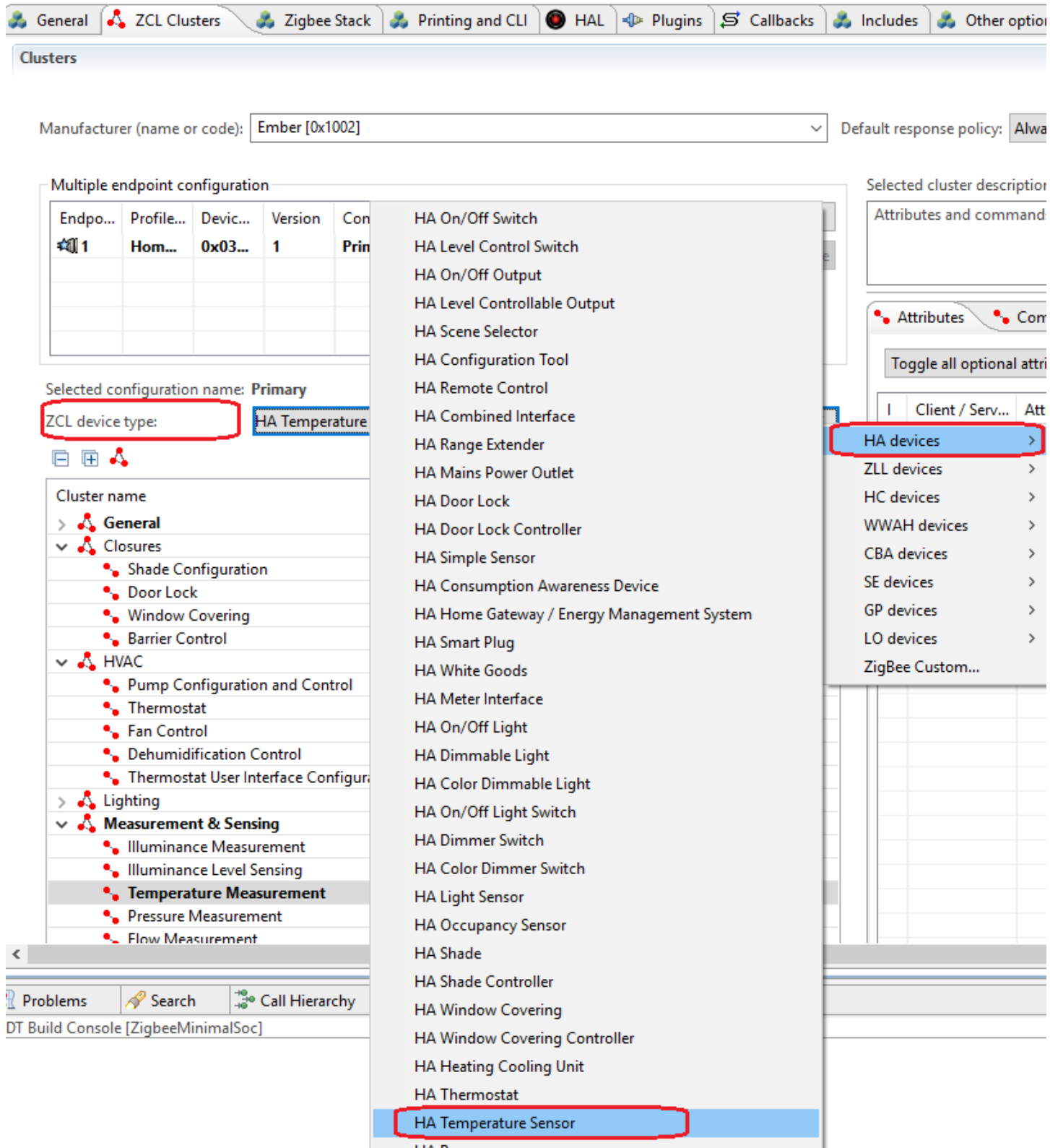


Figure 12 ZCL device type

- Make sure cluster "Temperature Measurement" server side is selected, then make sure the attribute "measured value" is selected. After that, turn to "Reporting" tab, and make sure the attribute "measured value" is selected. After this step, we can save temperature data in attribute "measured value" of cluster "Temperature Measurement", and we also can report this attribute to the coordinator.

Manufacturer (name or code): Ember [0x1002] Default response policy: Always

Multiple endpoint configuration

| Endpo... | Profile... | Devic... | Version | Configuration | Network |
|----------|------------|----------|---------|---------------|---------|
| 1 | Hom... | 0x03... | 1 | Primary | Primary |

Selected configuration name: Primary

ZCL device type: HA Temperature Sensor

Cluster name

| Cluster... | Client | Server | Mfg Id |
|---|--------|--------|--------|
| General | | | |
| Closures | | | |
| Shade Configuration | | | 0x0100 |
| Door Lock | | | 0x0101 |
| Window Covering | | | 0x0102 |
| Barrier Control | | | 0x0103 |
| HVAC | | | |
| Pump Configuration and Control | | | 0x0200 |
| Thermostat | | | 0x0201 |
| Fan Control | | | 0x0202 |
| Dehumidification Control | | | 0x0203 |
| Thermostat User Interface Configuration | | | 0x0204 |
| Lighting | | | |
| Measurement & Sensing | | | |
| Illuminance Measurement | | | 0x0400 |
| Illuminance Level Sensing | | | 0x0401 |
| Temperature Measurement | | | 0x0402 |
| Pressure Measurement | | | 0x0403 |
| Flow Measurement | | | 0x0404 |

Selected cluster description: Attributes and commands for configuring the measurement of temperature, and reporting temperature measurements.

Attributes Commands Reporting

| R | Client / Serv... | Attribute name | Min Interval (s) | Max Interval (s) | Reportable change |
|-------------------------------------|------------------|--------------------|------------------|------------------|-------------------|
| <input checked="" type="checkbox"/> | Server | measured value | 1 | 65534 | 0 |
| <input type="checkbox"/> | Server | min measured value | 1 | 65534 | 0 |
| <input type="checkbox"/> | Server | max measured value | 1 | 65534 | 0 |
| <input type="checkbox"/> | Server | cluster revision | 1 | 65534 | 0 |
| <input type="checkbox"/> | Client | cluster revision | 1 | 65534 | 0 |

Figure 13 attribute and reporting

- c. Select attribute “manufacture name” under cluster “Basic”, then set the default value to **your name**. We set this so that we can trace the reported data from the gateway side.

Manufacturer (name or code): Ember [0x1002] Default response policy: Always

Multiple endpoint configuration

| Endpo... | Profile... | Devic... | Version | Configuration | Network |
|----------|------------|----------|---------|---------------|---------|
| 1 | Hom... | 0x03... | 1 | Primary | Primary |

Selected configuration name: Primary

ZCL device type: HA Temperature Sensor

Cluster name

| Cluster... | Client | Server | Mfg Id |
|----------------------------------|--------|--------|--------|
| General | | | |
| Basic | | | 0x0000 |
| Power Configuration | | | 0x0001 |
| Device Temperature Configuration | | | 0x0002 |
| Identify | | | 0x0003 |
| Groups | | | 0x0004 |
| Scenes | | | 0x0005 |
| On/off | | | 0x0006 |

Selected cluster description: Attributes for determining basic information about a device, setting user device information such as location, and enabling a device.

Attributes Commands Reporting

Toggle all optional attributes

| I | Client / Serv... | Attribute name | Attr ID | Manufacturer co... | E | F | S | Type | B | Default |
|-------------------------------------|------------------|----------------------|---------|--------------------|--------------------------|--------------------------|-------------------------------------|----------|--------------------------|---------|
| <input checked="" type="checkbox"/> | Server | ZCL version | 0x0000 | | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | INT8U | <input type="checkbox"/> | 0x03 |
| <input type="checkbox"/> | Server | application version | 0x0001 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | INT8U | <input type="checkbox"/> | 0x00 |
| <input type="checkbox"/> | Server | stack version | 0x0002 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | INT8U | <input type="checkbox"/> | 0x00 |
| <input type="checkbox"/> | Server | hardware version | 0x0003 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | INT8U | <input type="checkbox"/> | 0x00 |
| <input checked="" type="checkbox"/> | Server | manufacturer name | 0x0004 | | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | CHAR... | <input type="checkbox"/> | Jim |
| <input type="checkbox"/> | Server | model identifier | 0x0005 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | CHAR... | <input type="checkbox"/> | |
| <input type="checkbox"/> | Server | date code | 0x0006 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | CHAR... | <input type="checkbox"/> | |
| <input checked="" type="checkbox"/> | Server | power source | 0x0007 | | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | ENUM8 | <input type="checkbox"/> | 0x00 |
| <input type="checkbox"/> | Server | generic device class | 0x0008 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ENUM8 | <input type="checkbox"/> | 0xFF |
| <input type="checkbox"/> | Server | generic device type | 0x0009 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ENUM8 | <input type="checkbox"/> | 0xFF |
| <input type="checkbox"/> | Server | product code | 0x000A | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | OCTET... | <input type="checkbox"/> | |

Figure 14 set manufacture name

- d. Under cluster “Basic”, in “Reporting” tab, enable reporting of attribute “manufacture name”, set the min interval to 15 seconds and max interval to 30 seconds.

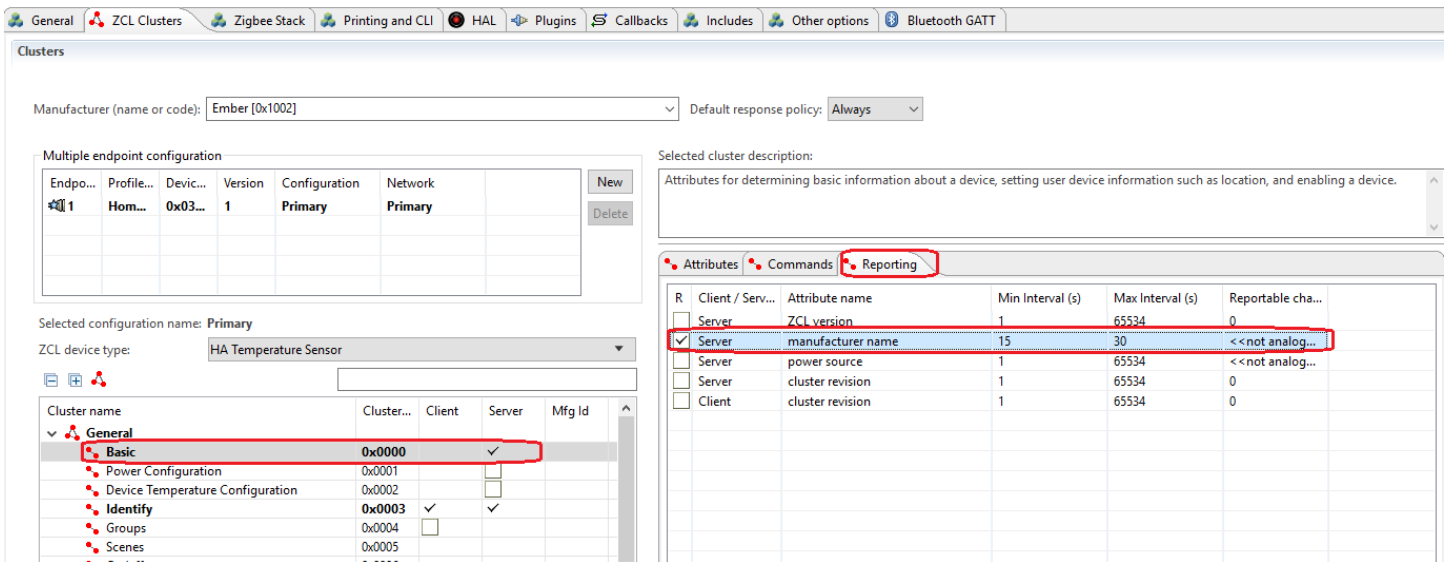


Figure 15 enable reporting of attribute “manufacturer name”

11. Click “Plugins” tab:

- Enable “Idle/Sleep” plugin, then in the properties of this plugin, enable the option “Stay awake when NOT joined”; So that when device hasn’t joined, the device can still keep awake, then you can use the command line to operate. Please also enable the option “Use button to force wakeup or allow sleep”. When the device is asleep, the command line interface won’t be available. With this interface, you can use button0 to force the device stay awake and use button1 to force the device allow sleep.

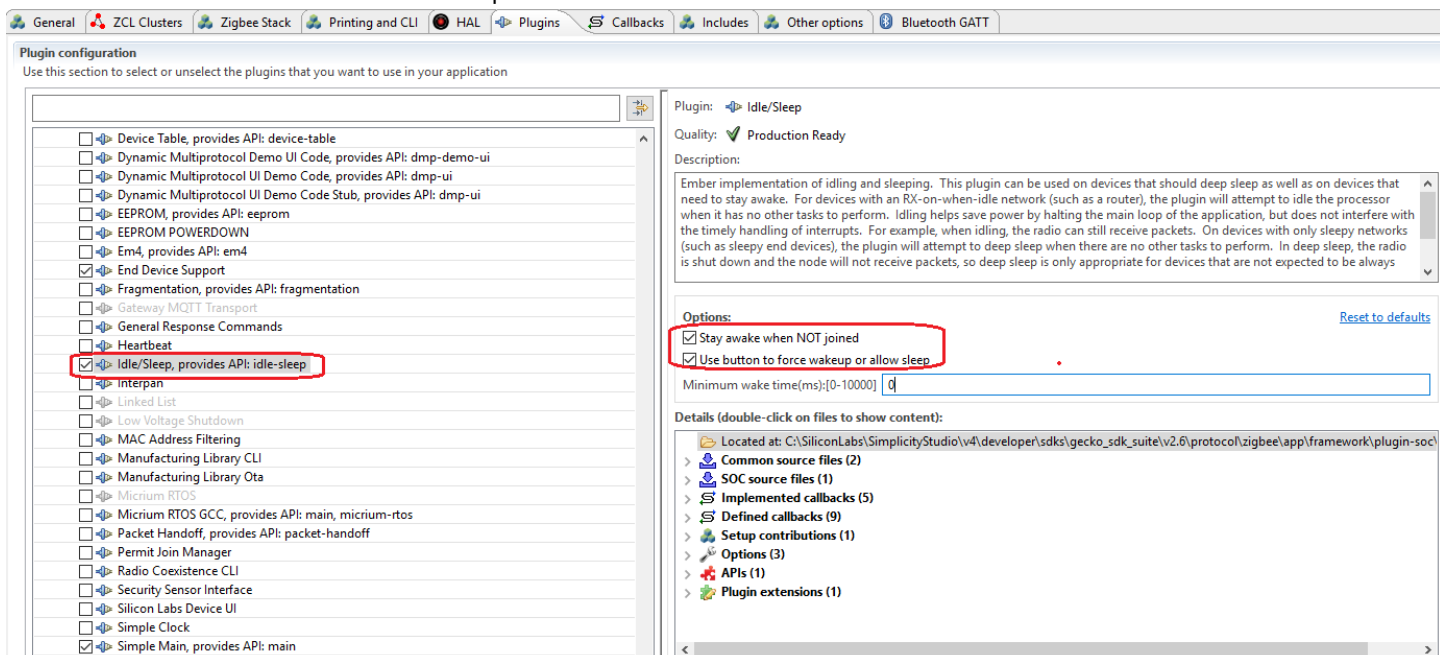


Figure 16 Plugin “Idle/Sleep”

- Select “End Device Support” plugin, in the properties, set the short poll interval to 1s and long poll interval to 10s.

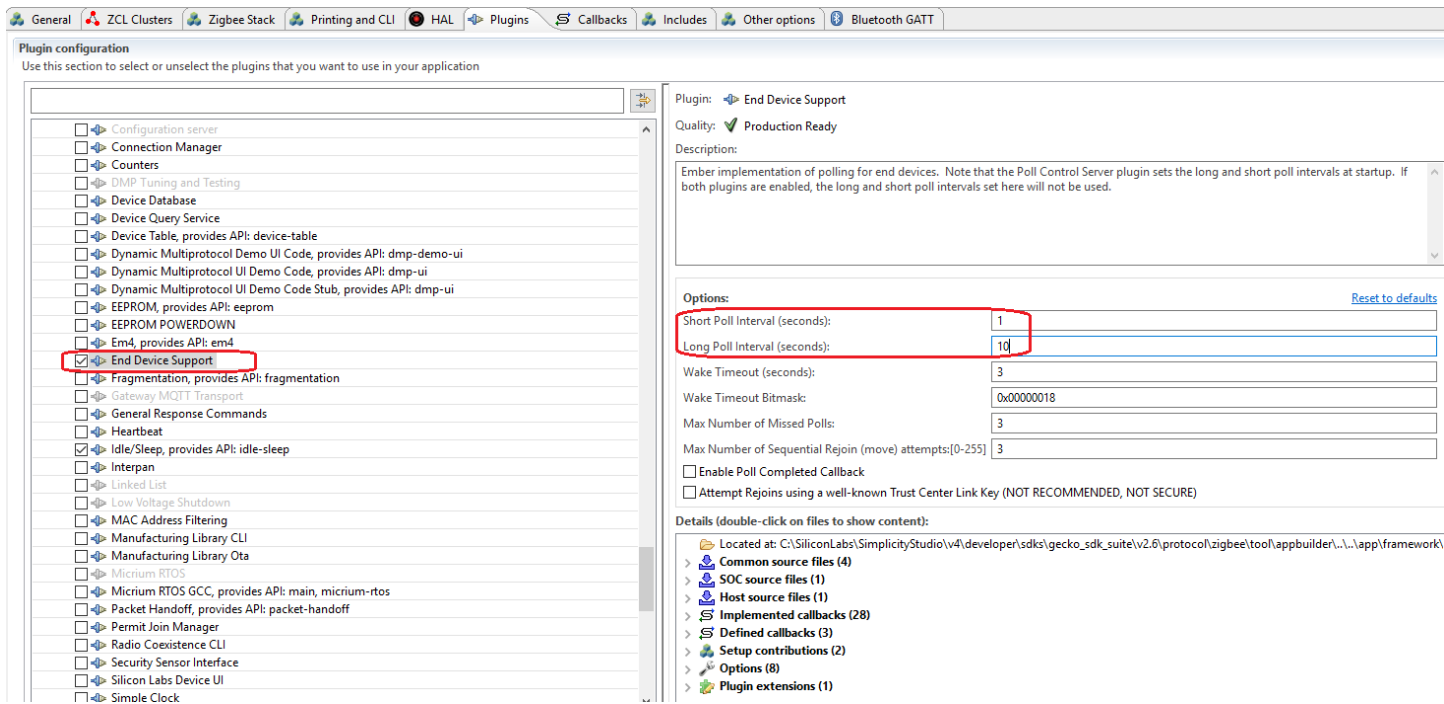


Figure 17 Plugin “End Device Support”

- c. Enable plugin “Reporting”, and in the properties, set reporting table size to 10. With this step, the attributes which will be reported will be saved in the reporting table and will be reported periodically or reported after it is changed.

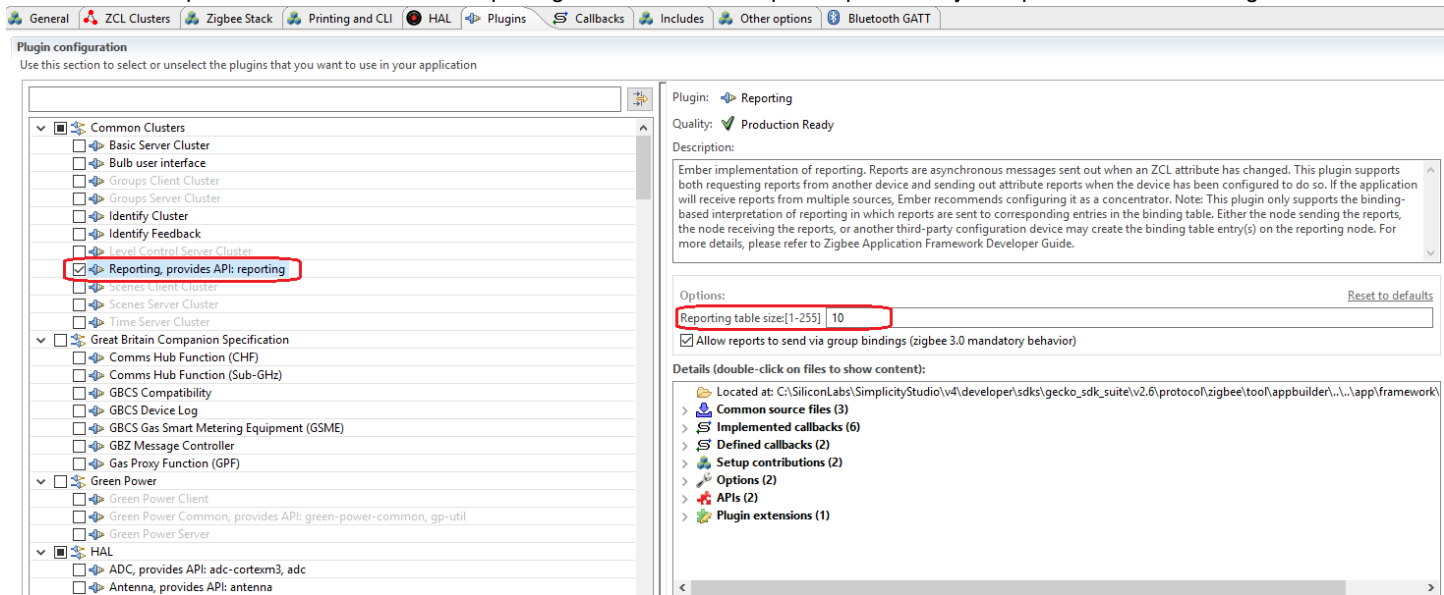


Figure 18 Plugin “Reporting”

- d. Enable plugin “Find and Bind Initiator”. With this step, the sleepy end device can start the “finding and binding” process, and with that, it can setup the binding table automatically. You can refer to Zigbee BDB spec section 8.5/8.6 to learn more about this procedure.

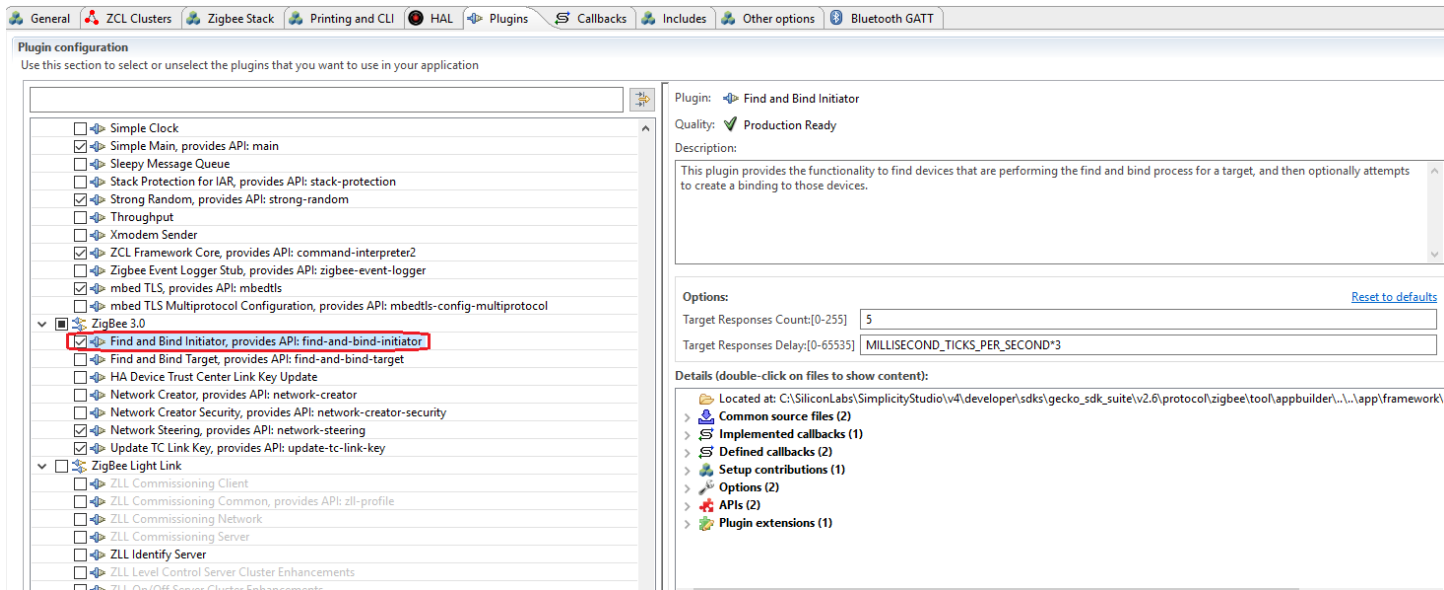


Figure 19 Plugin “Find and Bind Initiator”

12. Click “Callbacks” tab:

- Unfold “Non-cluster related”, enable “Main Init” callback emberAfMainInitCallback; You can input “Main Init” in the filter to find it quickly.

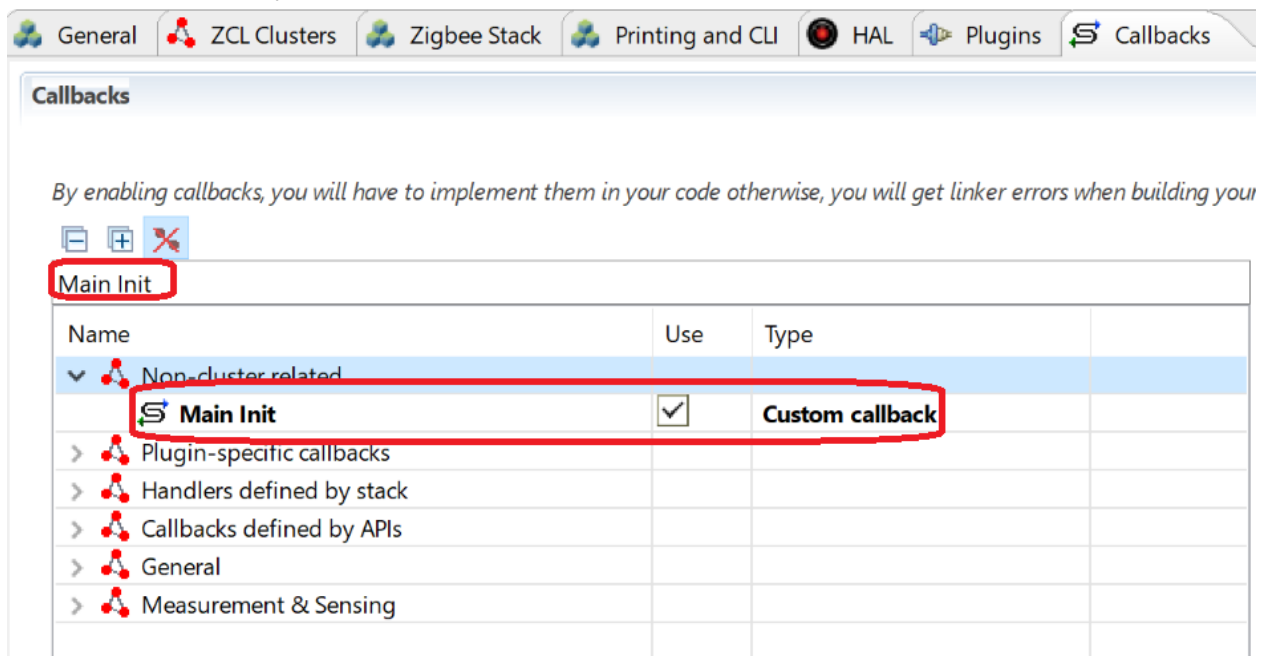


Figure 20 Callback “Main Init”

- Unfold “Non-cluster related”, enable “Stack Status” callback emberAfStackStatusCallback;
- Unfold “Plugin-specific callbacks”, enable “Complete” callback emberAfPluginFindAndBindInitiatorCompleteCallback of plugin “find and bind initiator”;

13. Click “Printing and CLI” tab, enable the debug print of “Reporting”:

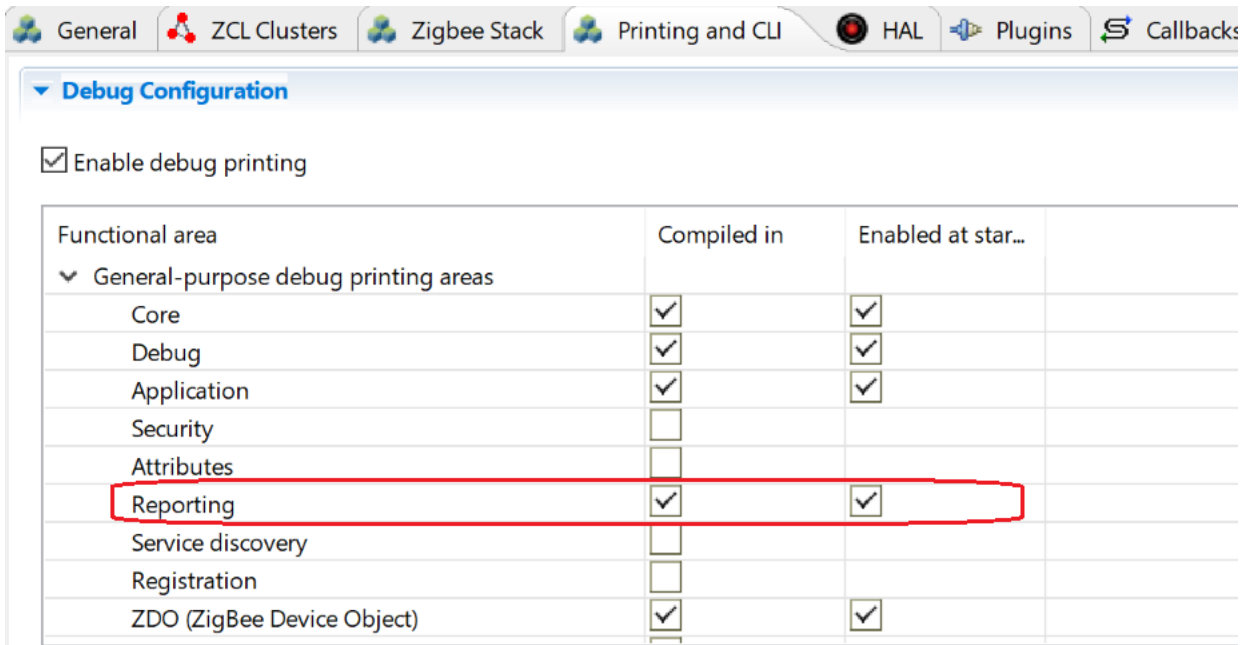


Figure 21 Printing and CLI

14. Click "Includes" tab, scroll to the bottom (You might need to scroll the bar on the very right as well), in the "Event Configuration" field, hit "New" button to add a custom event customWriteAttributeEventData and its handler customWriteAttributeEventHandler.
15. Save the modified Project .ISC file and click "Generate". Notice the project files appearing in Project Explorer. A window saying "generating successfully" will appear. Click OK.
16. Edit <projectname>_callbacks.c:
 - a. Modify function emberAfStackStatusCallback as below:

```
bool emberAfStackStatusCallback(EmberStatus status)
{
    // This value is ignored by the framework.
    if (EMBER_NETWORK_UP == status) {
        //start find and bind procedure when joins network
        EmberStatus status = emberAfPluginFindAndBindInitiatorStart(1);
        emberAfCorePrintln("Find and bind initiator %p: 0x%X", "start", status);
    }

    return false;
}
```

- b. Add function emberAfPluginFindAndBindInitiatorCompleteCallback as below:

```
void emberAfPluginFindAndBindInitiatorCompleteCallback(EmberStatus status)
{
    emberAfCorePrintln("Find and bind initiator %p: 0x%X", "complete", status);
}
```

- c. Add the following source code snippets to read temperature from adc:

```
#include "em_adc.h"

EmberEventControl customWriteAttributeEventData;

static void AdcSetup(void)
{
    /* Enable ADC clock */
    CMU_ClockEnable(cmuClock_ADC0, true);

    /* Base the ADC configuration on the default setup. */
}
```

```

ADC_Init_TypeDef      init  = ADC_INIT_DEFAULT;
ADC_InitSingle_TypeDef sInit = ADC_INITSINGLE_DEFAULT;

/* Initialize timebases */
init.timebase = ADC_TimebaseCalc(0);
init.prescale = ADC_PrescaleCalc(400000, 0);
ADC_Init(ADC0, &init);

/* Set input to temperature sensor. Reference must be 1.25V */
sInit.reference = adcRef1V25;
sInit.acqTime   = adcAcqTime8; /* Minimum time for temperature sensor */
sInit.posSel    = adcPosSelTEMP;
ADC_InitSingle(ADC0, &sInit);
}

static uint32_t AdcRead(void)
{
    ADC_Start(ADC0, adcStartSingle);
    while ( (ADC0->STATUS & ADC_STATUS_SINGLEDV) == 0 ) {
    }
    return ADC_DataSingleGet(ADC0);
}

static float ConvertToCelsius(int32_t adcSample)
{
    uint32_t calTemp0;
    uint32_t calValue0;
    int32_t readDiff;
    float temp;

    /* Factory calibration temperature from device information page. */
    calTemp0 = ((DEVINFO->CAL & _DEVINFO_CAL_TEMP_MASK)
        >> _DEVINFO_CAL_TEMP_SHIFT);

    calValue0 = ((DEVINFO->ADC0CAL3
        /* _DEVINFO_ADC0CAL3_TEMPREAD1V25_MASK is not correct in
        current CMSIS. This is a 12-bit value, not 16-bit. */
        & 0xFFFF)
        >> _DEVINFO_ADC0CAL3_TEMPREAD1V25_SHIFT);

    if ((calTemp0 == 0xFF) || (calValue0 == 0xFFF)) {
        /* The temperature sensor is not calibrated */
        return -100.0;
    }

    /* Vref = 1250mV
    TGRAD_ADCTH = 1.84 mV/degC (from datasheet)
    */
    readDiff = calValue0 - adcSample;
    temp      = ((float)readDiff * 1250);
    temp      /= (4096 * -1.84);

    /* Calculate offset from calibration temperature */
    temp      = (float)calTemp0 - temp;
    return temp * 100;
}

void customWriteAttributeEventHandler()
{
    int32_t sample = 0;
    int16_t temp = 0;

    emberEventControlSetInactive(customWriteAttributeEventData);

    sample = AdcRead();
    temp = ConvertToCelsius(sample);
    emberAfCorePrintln("sample=%d", sample);
}


```

```
emberAfCorePrintln("temp=%d", temp);

emberAfWriteServerAttribute(1,
                             ZCL_TEMP_MEASUREMENT_CLUSTER_ID,
                             ZCL_TEMP_MEASURED_VALUE_ATTRIBUTE_ID,
                             &temp,
                             ZCL_INT16S_ATTRIBUTE_TYPE);

emberEventControlSetDelayMS(customWriteAttributeEventData, 30000);
}

void emberAfMainInitCallback(void)
{
    AdcSetup();
    emberEventControlSetDelayMS(customWriteAttributeEventData, 30000);
}
```

17. Select the project in Project Explorer window and compile your project by clicking on the Build icon . Ensure that the build completes with 0 errors.

3 Test and observe the polling and the current.

1. Choose one WSTK as the sleepy end device, flash bootloader (You can use the pre-built bootloader described below) to WSTK, and then flash sleepy end device application to WSTK.

```
C:\SiliconLabs\SimplicityStudio\v4\developer\sdk\gecko_sdk_suite\v2.6\platform\bootloader\sample-apps\bootloader-storage-internal-single\efr32mg12p332f1024g1125-brd4162a
```

Please choose the pre-built bootloader `bootloader-storage-internal-single-combined.s37`

2. On the console of the switch, run following command to join the network.

```
//Find a joinable network and join it
MyZSED> plugin network-steering start 0
```

3. After the device joined a network, it will enter sleep mode. In this case, the command line is not available. If you need to run debug command, please press **button0** to **force** the device stay **awake**. After you finished debugging, press **button1** to **allow** the device to enter **sleep** mode. **Make sure you have unplug then plug the cable once after you flashed the program. (The debugger could cause a high current).**
4. Use info command to check whether you joined the right network.

```
//Check the pan id and channel of ZSED. They should be the same as we provided.
MyZSED> info
MFG String:
AppBuilder MFG Code: 0x1002
node [(>)000B57FFFE648DD8] chan [18] pwr [3]
panID [0x2019] nodeID [0x0000] xpan [0x(>)A3E54612381CBF6B]
parentID [0xFFFF] parentRssi [0]
stack ver. [6.4.1 GA build 408]
nodeType [0x01]
Security level [05]
network state [02] Buffs: 73 / 75
Ep cnt: 2
ep 1 [endpoint enabled, device enabled] nwk [0] profile [0x0104] devId [0x0100] ver [0x00]
  in (server) cluster: 0x0000 (Basic)
  in (server) cluster: 0x0003 (Identify)
  in (server) cluster: 0x0004 (Groups)
  in (server) cluster: 0x0005 (Scenes)
  in (server) cluster: 0x0006 (On/off)
ep 2 [endpoint enabled, device enabled] nwk [0] profile [0x0104] devId [0x0100] ver [0x00]
  in (server) cluster: 0x0000 (Basic)
  in (server) cluster: 0x0003 (Identify)
  in (server) cluster: 0x0004 (Groups)
  in (server) cluster: 0x0005 (Scenes)
  in (server) cluster: 0x0006 (On/off)
Nwk cnt: 1
nwk 0 [Primary (pro)]
  nodeType [0x04]
  securityProfile [0x05]
```

5. If you joined a wrong network, please leave the network and join again.

```
//Leave network
MyZSED> network leave

//join again
MyZSED> plugin network-steering start 0
```

6. Make sure you have joined the right network before you continue.

7. Get the NWK key by “keys print” command, then add it to Simplicity Studio.

```
MyZSED>keys print
EMBER_SECURITY_LEVEL: 05
NWK Key out FC: 00000010
NWK Key seq num: 0x00
NWK Key: 78 87 4D 1F DE B4 08 21 5B 83 DE 43 E9 FD B4 CC
Link Key out FC: 00000002
TC Link Key
Index IEEE Address          In FC    Type  Auth  Key
-      (>)90FD9FFFE7B81BD  0000F003 L      y    95 34 1A 83 18 22 AC A5 89 4E 45 90 60 DA D3
D7
Link Key Table
Index IEEE Address          In FC    Type  Auth  Key
0/0 entries used.
Transient Key Table
Index IEEE Address          In FC    TTL(s) Flag  Key
0 entry consuming 0 packet buffer.
```

8. Start a capture with Network Analyzer, observe the polling interval. Normally it should poll every 10 seconds.
9. Start a capture, then reset the sleepy end device. Observe the rejoin procedure in the sniffer.
10. Observe the periodically report and also the polling at this stage;

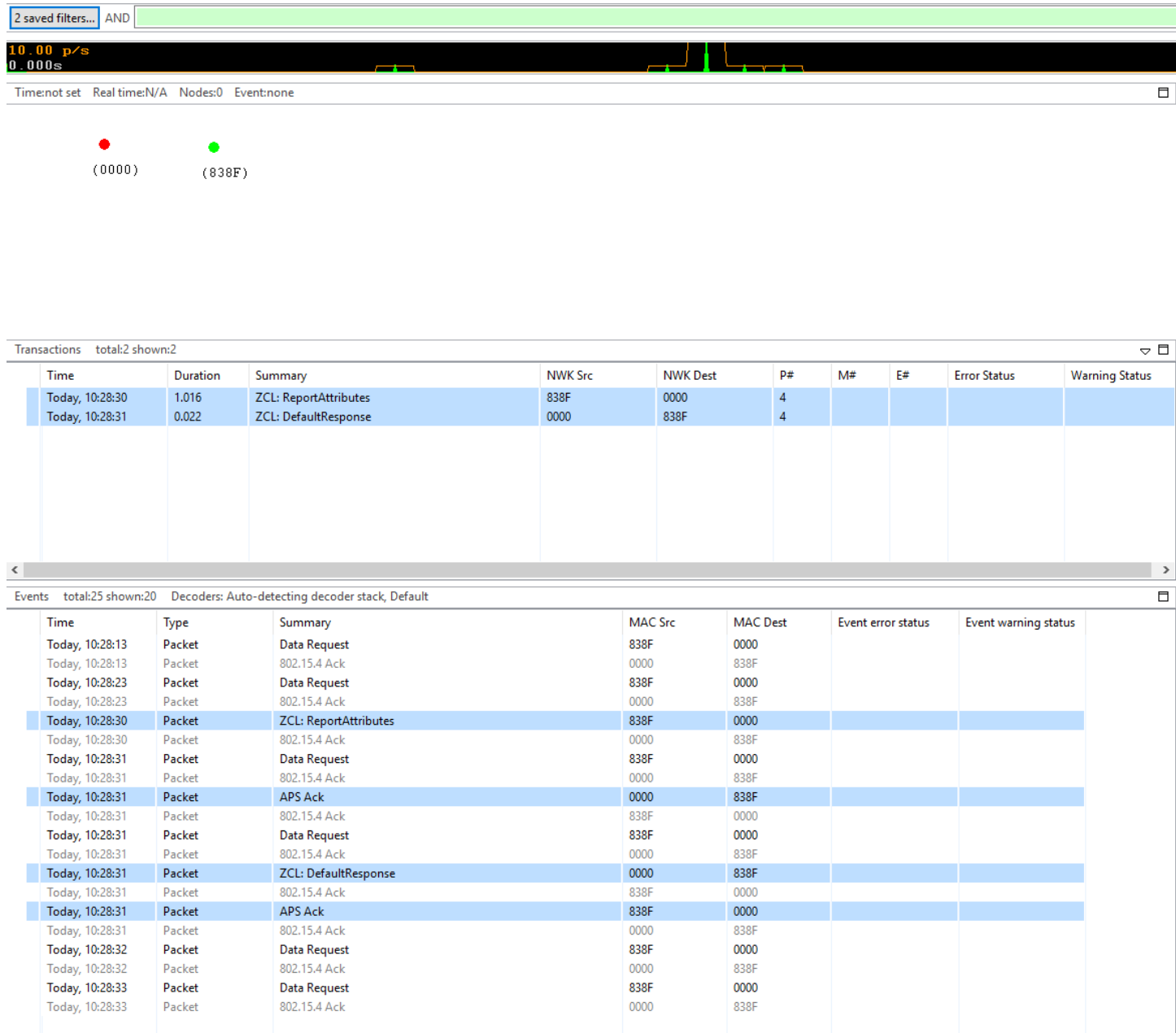


Figure 22 Observe the report and polling

11. Start Energy Profiler and measure the sleepy current. Record the average current;

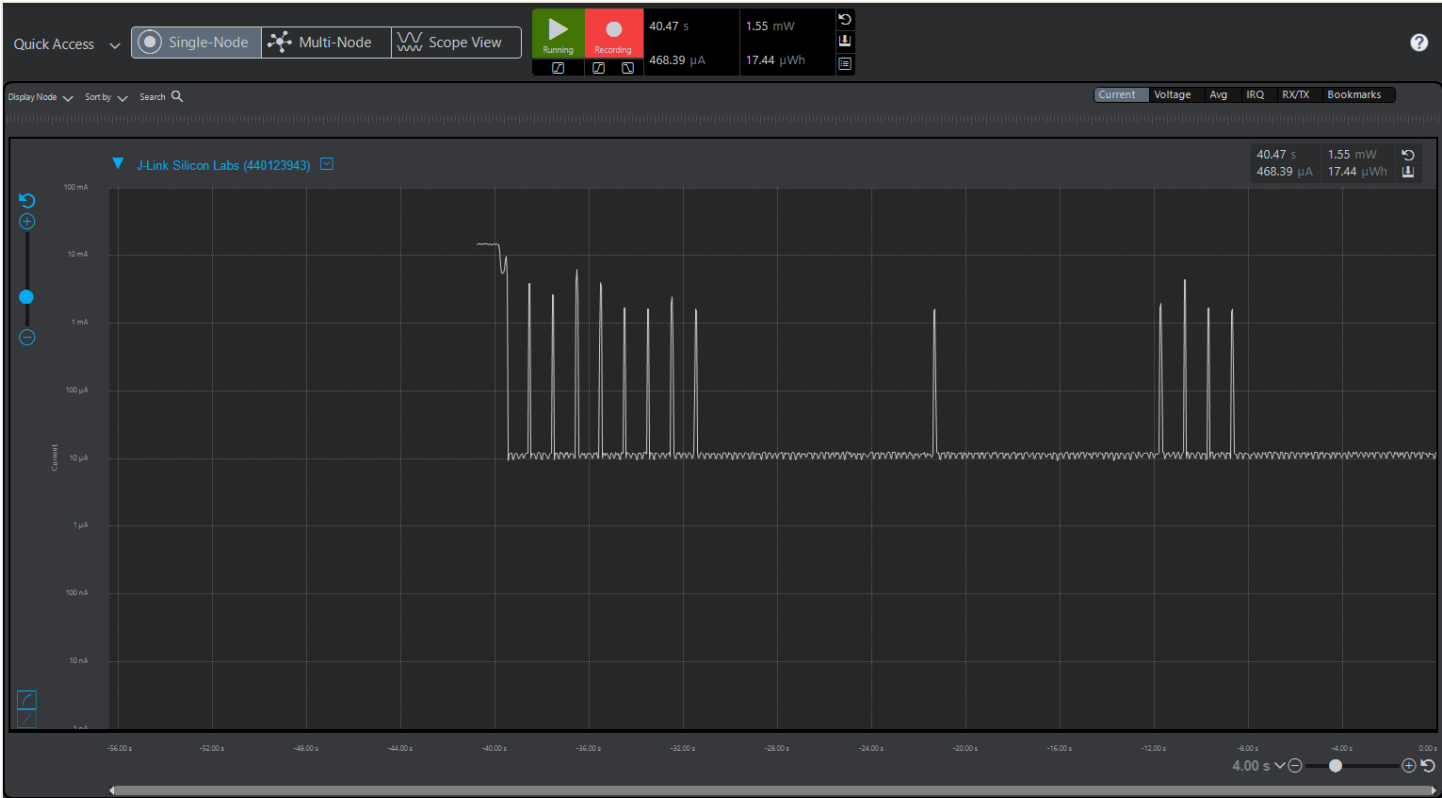


Figure 23 Observe the current