



# <docnum>: 4 Understand FLiRS devices

---

This hands-on exercise will demonstrate what a Z-Wave FLiRS device is.

The exercise will use the Doorlock sample application that ships as part of the Z-Wave Embedded SDK

This exercise is part of the series 'Z-Wave 1-Day Course'.

- 1 Include using SmartStart
- 2 Decrypt Z-Wave RF Frames using the Zniffer
- 3A Compile Switch On/Off and Enable Debug
- 3B Modify Switch On/Off
- **4 Understand FLiRS devices**

## KEY FEATURES

---

- Understand key features of a FLiRS device.
- Use the Energy Profiler to capture the power consumption.

## 1 Introduction

In this exercise we will explore a Z-Wave FLiRS device, and learn the benefits of a “*listening sleeping device*”; a battery powered device that must be communicated with at any time with short latency.

### 1.1 Hardware Requirements

- 1 WSTK Main Development Board
- 1 Z-Wave Radio Development Board: ZGM130S SiP Module
- 1 UZB Controller
- 1 USB Ziffer

### 1.2 Software Requirements

- Simplicity Studio v4
- Z-Wave 7 SDK
- Z-Wave PC Controller
- Z-Wave Ziffer



Figure 1: Main development board with Z-Wave SiP module

### 1.3 Pre-requisites

Previous Hands-On exercises has covered how to use the PC Controller and Ziffer application to build a Z-Wave network and capturing the RF communication for development purpose. This exercise assumes you are familiar with these tools.

Previous Hands-On exercises has also covered how to use the sample applications that ships with the Z-Wave SDK. This exercise assumes you are familiar with using and compiling one of the sample applications.

## 2 Compile the Doorlock Sample Application

In this section we will be compiling the Doorlock Sample Application. The steps required are the same, as for Switch On / Off, which we covered in exercise “3A Compile Switch OnOff and enable debug”. In the following, the steps are summarized, but you should refer to exercise 3A if you want instructions in how to enable and use the serial debugger.

### 2.1 Open sample project

- Connect your Z-Wave hardware to the USB port of the computer and it should show up in the ‘Debug Adapters’ section in Simplicity Studio.
- Click once on the “J-Link Silicon Labs” which instructs to studio the show relevant information about Z-Wave 700.
- Under “Software Example” click on the DoorLock sample application.

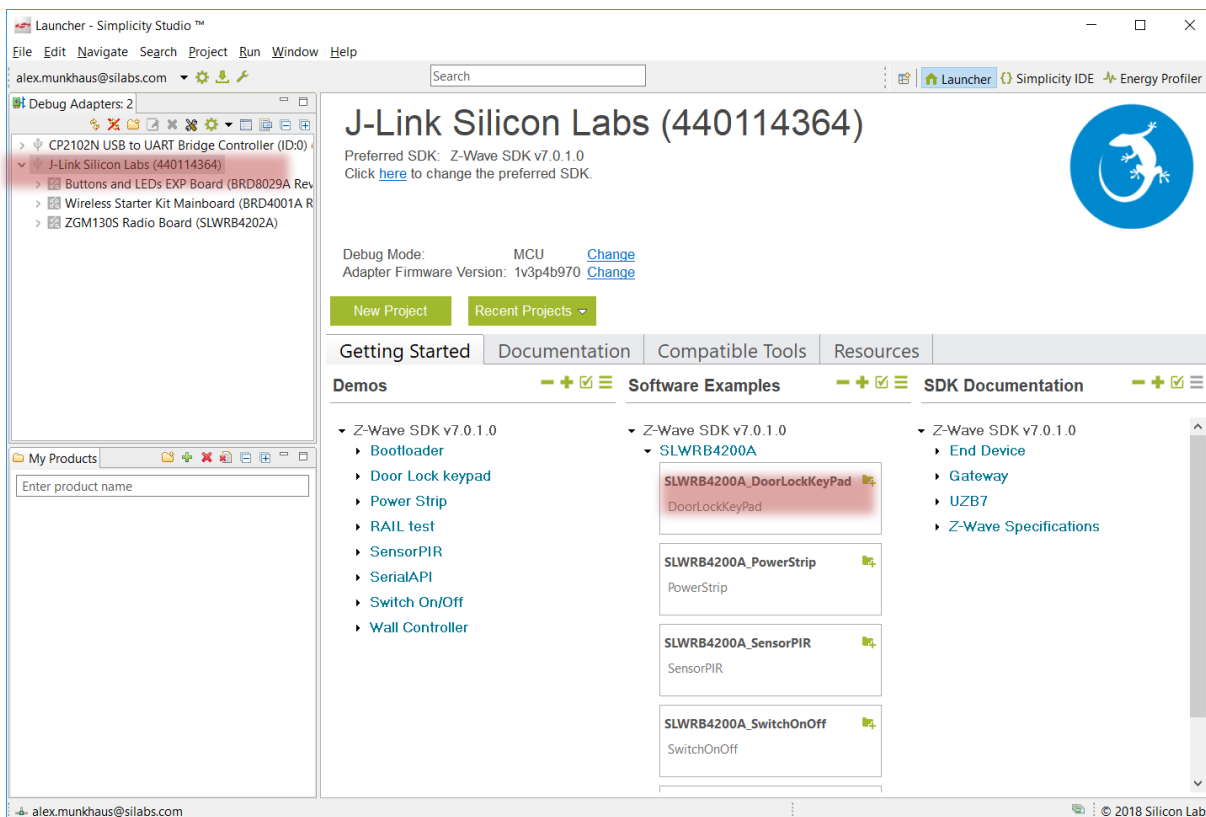


Figure 2: Open a Z-Wave sample application: DoorLock

### 2.2 Set the frequency

The sample app will not compile just yet. You need to set the frequency that matches the region you intend to use the Z-Wave Product in.

- In the main source file “DoorLockKeyPad.c”, locate the variable APP\_FREQ:

```
static const SRadioConfig_t RadioConfig =  
{  
    .iListenBeforeTalkThreshold = ELISTENBEFORETALKTRESHOLD_DEFAULT,  
    .iTxPowerLevelMax = APP_MAX_TX_POWER,  
    .iTxPowerLevelAdjust = APP_MEASURED_ODBM_TX_POWER,  
    .eRegion = APP_FREQ  
};
```

**Figure 3: APP\_FREQ needs to be set to your RF region**

Refer to Table 1 for a complete list of supported frequencies by the SDK.

**Hint** Navigate to [Silicon Labs website](https://www.silabs.com), to see which countries has been approved for the Z-Wave RF.

**Table 1: Overview of a possible frequencies**

Frequency Region	Variable to use
Europe	REGION_EU
United States of America	REGION_US
Australia/New Zealand	REGION_ANZ
Hong Kong	REGION_HK
Malaysia	REGION_MY
India	REGION_IN
Israel	REGION_IL
Russia	REGION_RU
China	REGION_CN
Japan	REGION_JP
Korea	REGION_KR


In this guide we will be using the European frequency, thus we enter 'REGION\_EU'.

```
static const SRadioConfig_t RadioConfig =  
{  
    .iListenBeforeTalkThreshold = ELISTENBEFORETALKTRESHOLD_DEFAULT,  
    .iTxPowerLevelMax = APP_MAX_TX_POWER,  
    .iTxPowerLevelAdjust = APP_MEASURED_ODBM_TX_POWER,  
    .eRegion = REGION_EU  
};
```

**Figure 4: APP\_FREQ is set to REGION\_EU**

### 2.3 Compile the DoorLock Application

You have now configured the Z-Wave sample application, and you are ready to compile.

- Click on the 'Build'  button to start building the project.
- When the build finishes after a short while, a new folder named 'Binaries' are showed in the Project Explore. Expand the folder and right click on the \*.hex file to select 'Flash to Device..'
- Select the connected hardware in the pop-up window. The 'Flash Programmer' is now prefilled with all needed data, and you are ready to click on 'Program'.
- Click 'Program'.

After a short while the programming finishes, and your end device is now flashed with a Z-Wave sample application.

### 3 Include and run the Doorlock Sample Application

In this section we will be including the Doorlock Sample Application into the Z-Wave Network. In previous exercise “2A Decrypt Z-Wave RF Frames using the Zniiffer”, we already added the DSK into the provisioning list of the PC Controller.

**Hint** The internal file system is not erased between reprogramming. This allows a node to stay in a network and keep the same network keys when you reprogram it.

If you need to change e.g. the frequency at which the module operates or the DSK, you need to “Erase” the chip before the new frequency will be written to the internal NVM.

This means, the DSK will still be valid despite we just programmed our device with a completely different sample application.

If you are using a new device or if you haven't previously added the DSK to the PC Controller, refer to exercise “2A Decrypt Z-Wave RF Frames using the Zniiffer” for instructions in how to readout the DSK from a device and add it to the PC Controller.

#### 3.1 Remove / Include the old device from / to the PC Controller

Since the DSK is the same, the PC Controller thinks that device is already included, though as a Switch On/Off. We need to remove the association to the Switch On/Off sample application to this DSK.

- In the PC Controller, click on “Remove”
- On the device, click on “BTN1” to set the device in learn mode.
- The device should now be removed from the PC Controller.

When the old association is removed, the PC Controller will automatically include the DoorLock sample application thanks to SmartStart. When successfully, the PC Controller should look like Figure 5.

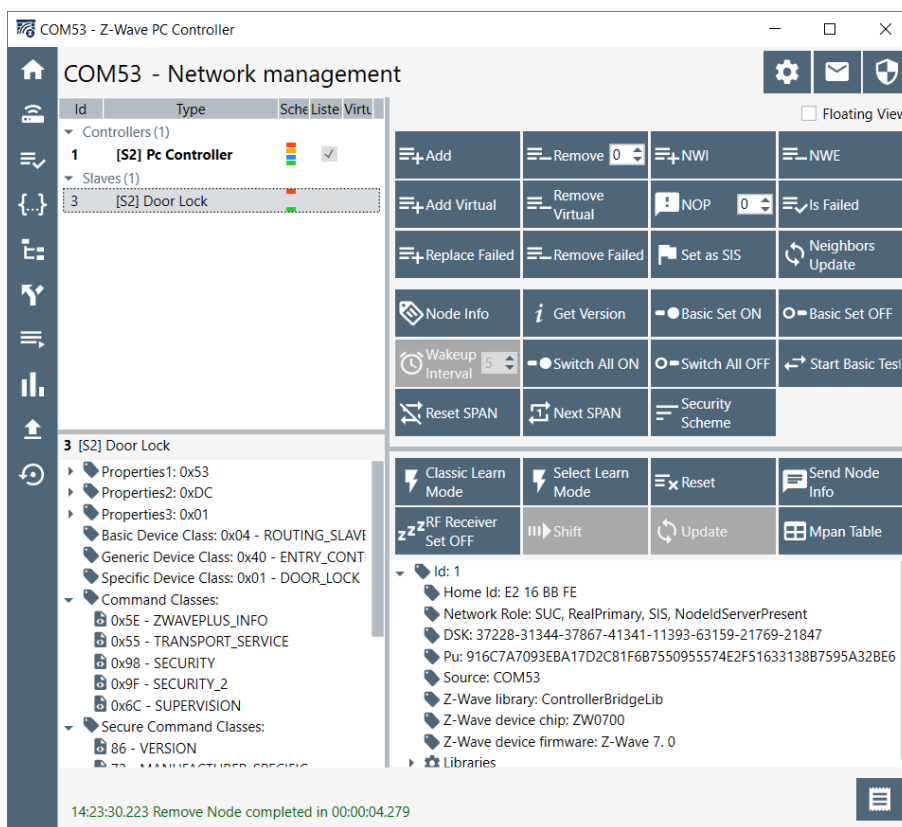


Figure 5: DoorLock added into Z-Wave network

### 3.2 Test the functionality

In this section, we will briefly test the functionality of the DoorLock Sample application.

**Hint** The functionalities of all sample applications are described in document “INS14278 – How to Use Certified Apps” found in the documentation section of Simplicity Studio.

Test the Lock and Unlock functionality. In the following steps we will be unlocking the door:

- In the PC Controller, double click on “62 – DOOR\_LOCK” under secure Command Classes in the lower left corner.
- This opens the “Command Classes” view in the PC Controller and selects the Door Lock Command class.
- Set the Command to “0x01 – DOOR\_LOCK\_OPERATION\_SET”
- Set the “Target Value” to “00-DOOR\_UNSECURED”
- Click “Send”.

Verify that LED3 is now ON.

Next, we will lock the door, and LED3 should turn OFF:

- Set the “Target Value” to “FF-DOOR\_SECURED”
- Click “Send”.

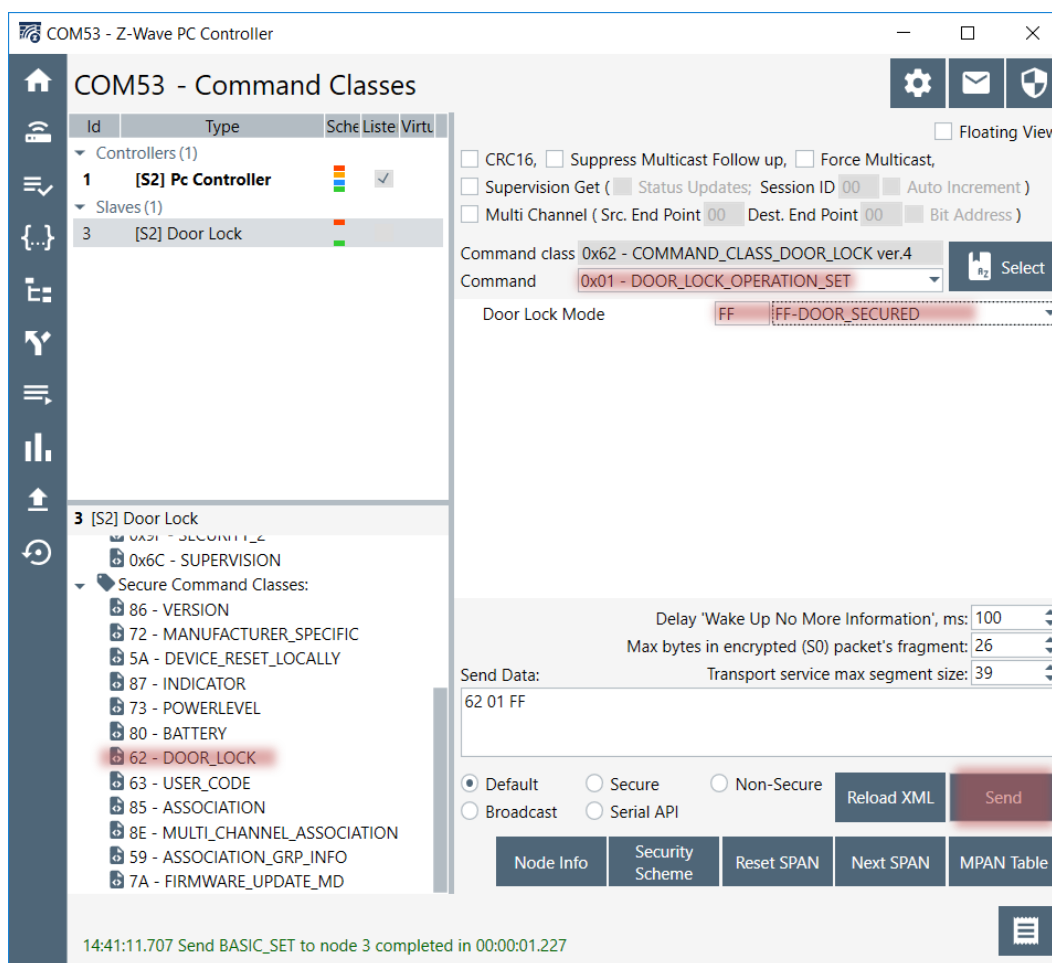


Figure 6: Command Class View in PC Controller

### 3.3 The Wake-Up Beam for a FLiRS Device


If a Z-Wave controller or other node in the network needs to communicate with a battery-powered device such as a door lock, the controller sends a special beam signal. The purpose of this beam is to wake up the FLiRS device.

The FLiRS device alternates between sleep mode and a partially awake mode in which it is listening for this beam signal at the rate ranging from once per second to four times per second (this is the designer's choice). When the FLiRS device receives this beam, it immediately fully wakes up and then communicates with the controller or other Z-Wave device utilizing standard Z-Wave protocol commands. If the device does not hear a Beam it goes back to full sleep for another period until it partially awakes again and listens for a Beam. It is this partially awake mode combined with the special Beam that provides for battery lives on par with fully sleeping devices while providing communications latencies of around one second.

**Hint** For a more in-depth description of Z-Wave FLiRS devices, refer to white paper [“Z-Wave FLiRS: Enabling Wireless Smart Door Locks and Thermostat”](#)

The WakeUp Beam can be seen in the Z-Wave Ziffer. This section will not cover how to capture a Ziffer trace – refer to exercise “2A Decrypt Z-Wave RF Frames using the Ziffer” for instructions in how to use the Ziffer.

The beam cannot be seen in the Ziffer, if the trace is filtered on HomeID.

- Click on Drop Filter  in the Ziffer to make sure the trace is not filtered on HomeID.

In Figure 7 a trace is shown for a wake-up sequence:

- The controller sends 3 requests to the FLiRS device, to ensure the device cannot be reached without beaming, which is a heavy load in the Z-Wave network.
- Since the device did not respond to the direct response, a WakeUp beam is initiated.
- When the Beam ends, the controller sends the command again,
- and the device acknowledges the message.

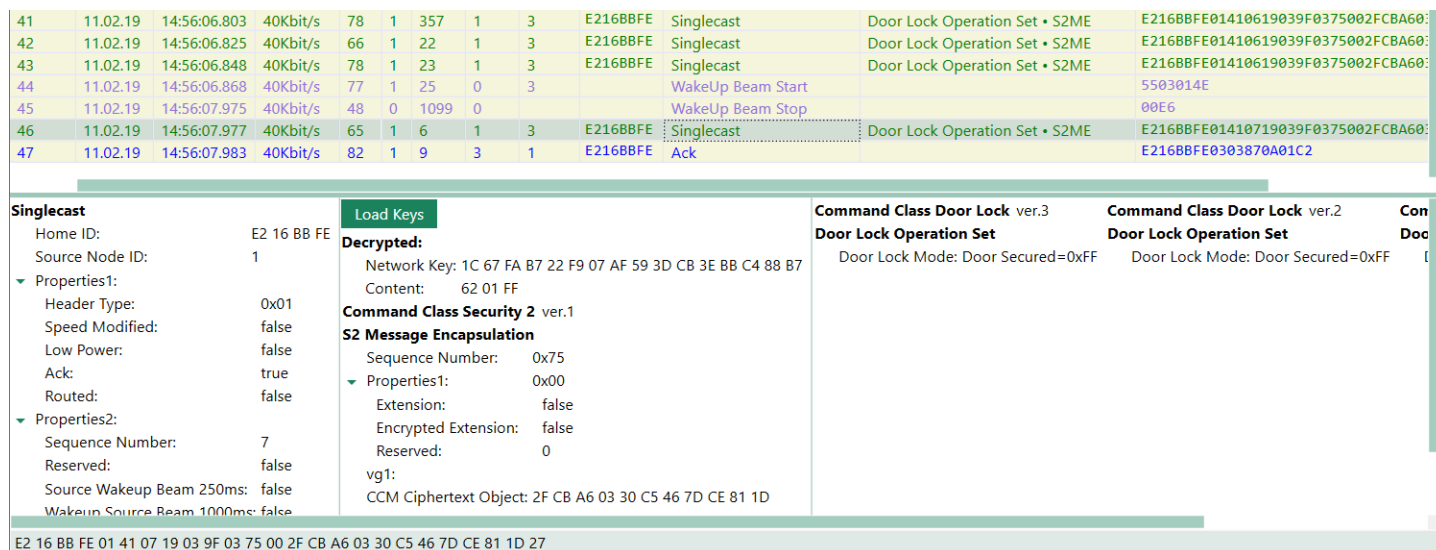



Figure 7: Ziffer trace of a WakeUp Beam

**Hint** A WakeUp Beam cannot be seen, if the trace is filtered on HomeId.



## 4 Power Consumption of DoorLock

In this section, we will be using the Energy Profiler in Simplicity Studio to monitor the energy consumption of the DoorLock FLiRS device.

- In Simplicity Studio, open the “Energy Profiler” by clicking on “Open Perspective” button  in the upper right corner.
- In the “Energy Monitor” click on “Quick Access” and click on “Start Energy Capture”.
- Select your device in the pop-up window and click OK.

The Energy Profiler now starts to capture and display the energy consumption, see Figure 8. Notice how the energy consumption raises every section, when the device must wake-up to listen for a Beam. Also notice, the fast wake-up and fall-to-sleep times, resulting in a very low average power consumption.

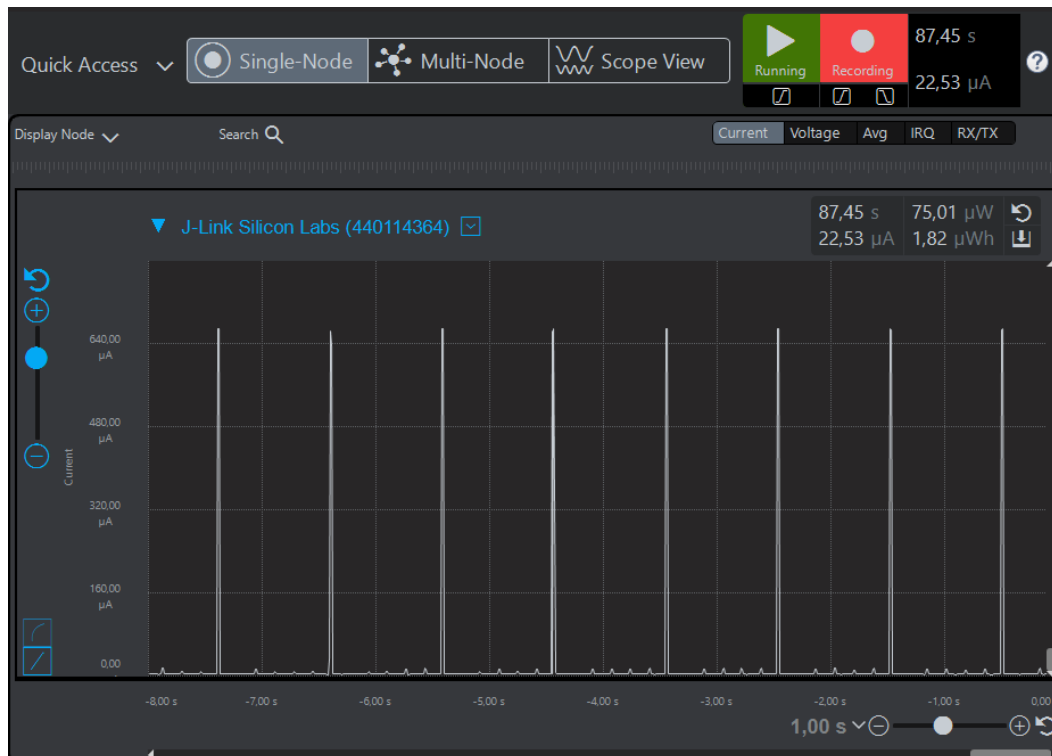


Figure 8: Energy Consumption of a FLiRS device

Let's try to wake-up the device.

- In the PC Controller, send a command to the device (refer to section “3.2 Test the functionality” for instructions)
- Notice the current consumption when the device wakes up to communicate with the controller. Refer to Figure 9.

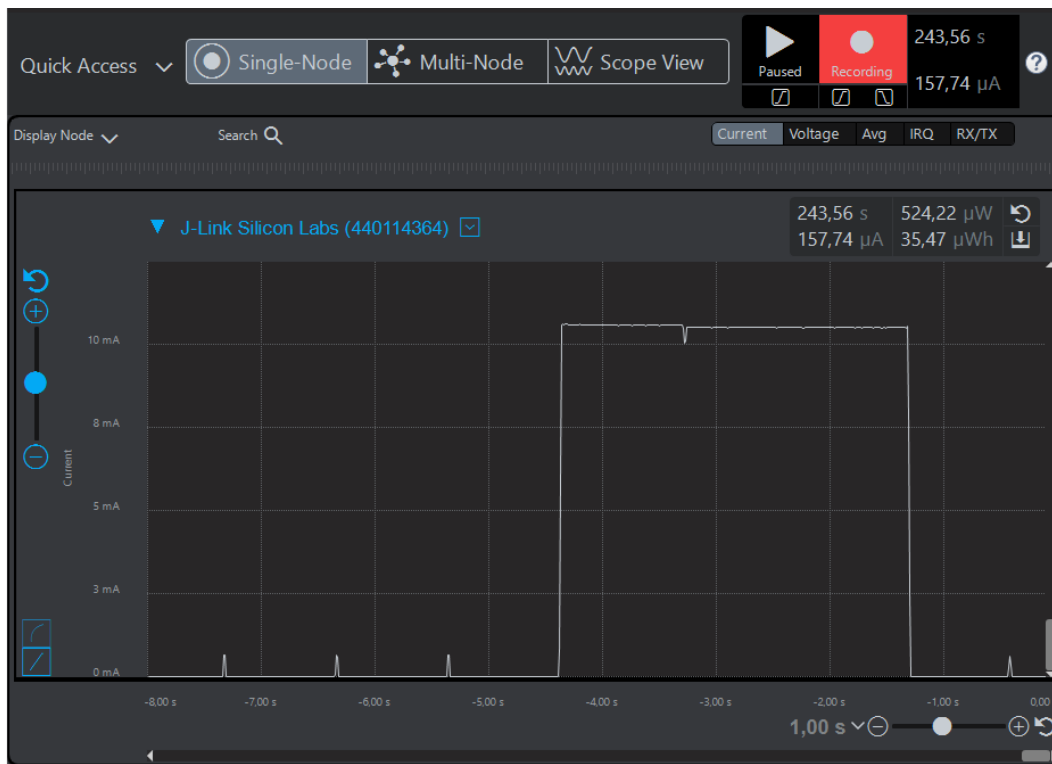


Figure 9: Energy Consumption when waking up

*This concludes the tutorial in how to use a FLiRS device.*