

## Abstract

Optimizing embedded applications for overall efficiency should be an integral part of the development process as it is important to understand how peripherals, software algorithms, and power saving modes work together.



This application note describes the power consumption analysis of an [L-Tek FF1502](#) Bluetooth Low Energy (BLE) beacon with [ULINKplus](#). Beacons are typically low power devices, which sleep most of the time and wake-up briefly to broadcast a message to nearby portable electronic devices (such as mobile phones for example). We'll show how to analyze the battery lifetime of an application running on the beacon by using the ULINKplus debug adapter that enables high-precision power analysis together with [Arm® Keil® MDK](#). The findings are used to improve the hardware design and to implement software changes that lead to reduced power consumption and longer battery life.

## Contents

Abstract .....	1
Summary.....	2
FF1502 – a Bluetooth Low Energy beacon .....	3
Application .....	3
Measurement setup .....	3
Reference design .....	4
Triggering sensors .....	4
Reading sensor data.....	5
Broadcasting .....	6
Sleeping.....	7
Software investigations .....	8
Redesigned software .....	8
Triggering sensors .....	9
Reading sensors .....	10
Broadcasting .....	10
Sleeping.....	10
Hardware investigations.....	11
Battery life calculation.....	13
Conclusion .....	13

## Summary

In this application note, we show how to use ULINK $plus$  to dramatically reduce the power consumption of a battery-driven IoT application. Using this versatile debug probe together with the latest debug features of  $\mu$ Vision, it is possible to increase the battery lifetime from just under a year (358 days) to nearly two and a half years (863 days).

Figure 1 shows the power profile of the original HW and SW design. The active time of the application is 51 ms and requires a total charge of 131  $\mu$ As :



Figure 1 Power measurement for the reference design

After optimization, these values reduce dramatically. The overall active time is reduced to 38 ms and the required charge is only 31  $\mu$ As as can be seen on Figure 2:



Figure 2 Power measurement for the optimized hard- and software

The next chapters explain the details about the application and the soft- and hardware changes that made this possible.

## FF1502 – a Bluetooth Low Energy beacon

The L-Tek FF1502 is a Bluetooth Low Energy (BLE) based sensor tag which operates as a broadcaster, i.e. beacon. Beacons are typically low power devices which sleep most of the time and wake-up briefly to broadcast a message to nearby portable electronic devices (such as mobile phones for example).

The small form factor and XBee compatible pin-out make the FF1502 a perfect building block for IoT end node applications. The device can be powered by a single CR2032 coin cell battery or via USB. It is based on the Nordic nRF51822 microcontroller (Arm Cortex-M0-based) with an integrated Bluetooth radio. Various sensors are connected to the microcontroller via I2C:

- temperature & humidity (Si7020-A20)
- light (BH1750)
- accelerometer (MMA8653FC)
- gyroscope (FXAS21002C)
- magnetometer (MAG3110).

The USB connector and CR2032 battery are connected to a DC/DC switch which provides power to the complete circuit (including the capacitor network).

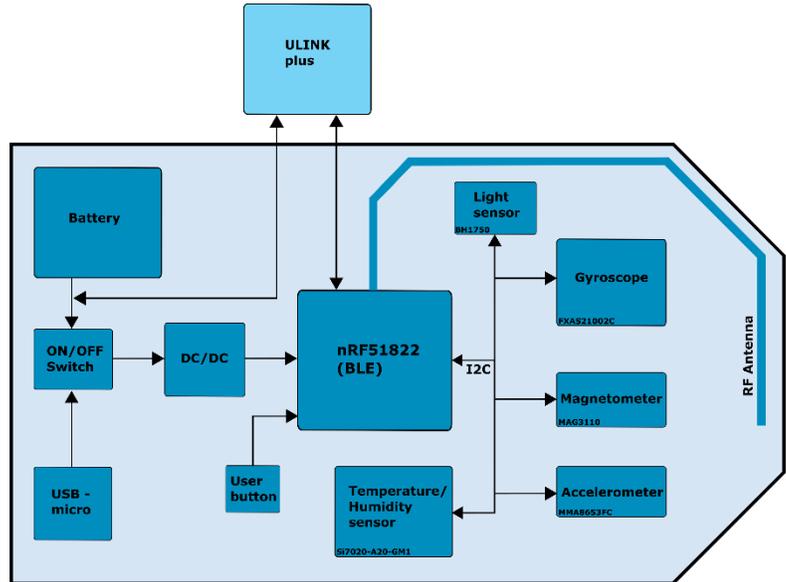


Figure 3 FF1502 block diagram

## Application

While running, the application idles most of the time with the MCU and all sensors being in low power mode. Every ten seconds, the device wakes-up and reads the sensor data (temperature/humidity/light). This data is then broadcasted via BLE. Then, the device goes back to sleep again.

## Measurement setup

The ULINKplus debug adapter supports shunt-based power measurement. In this setup, ULINKplus measures the battery voltage and total current provided to the device.

A 33.2 Ohm external shunt (10 mA range) is used for measuring during application execution, while only the ULINKplus' internal shunt (2.5 mA range) is used for measuring the MCU's low power mode. Using the internal shunt in active mode results in a voltage drop that is too large for the MCU to run.

All measurements are done in the new  **Energy Measurement without Debug** mode of  $\mu$ Vision. This mode is specifically useful for low-power targets as the debug circuitry and connection consume additional energy that is skewing the measurement results in the standard debug mode. In the Energy Measurement without Debug mode, the Arm CoreSight unit within the microcontroller is not active and thus only the energy of the application is measured.

The **System Analyzer** window in the  $\mu$ Vision IDE displays timing and energy information alongside to the markers and cursors in the current and voltage lines.

## ULINKplus offset calibration for very low current measurements

Before starting to measure the currents, you need to calibrate your set up. The ULINKplus user's guide (available at [www.keil.com/-fos](http://www.keil.com/-fos)) explains how to do this.

## Reference design

In the following, the original hardware and software design is called "reference design". The initial power measurement (see Figure 1) shows a power intensive active cycle (with a duration of 51 ms) which is repeated periodically (every ten seconds). According to  $\mu$ Vision's System Analyzer, the total electrical charge for the active cycle is:

$$Q_{\text{ref\_active}} = 131 \mu\text{As}$$

The cycle can be divided into the following phases:

- **triggering sensors** over the I2C bus to start a single measurement
- **reading sensor data** over the I2C bus
- **broadcasting** the sensor data over BLE
- **sleeping** before and after

## Triggering sensors

Sensor measurement is triggered by issuing commands over the I2C bus and waiting until the data is ready.

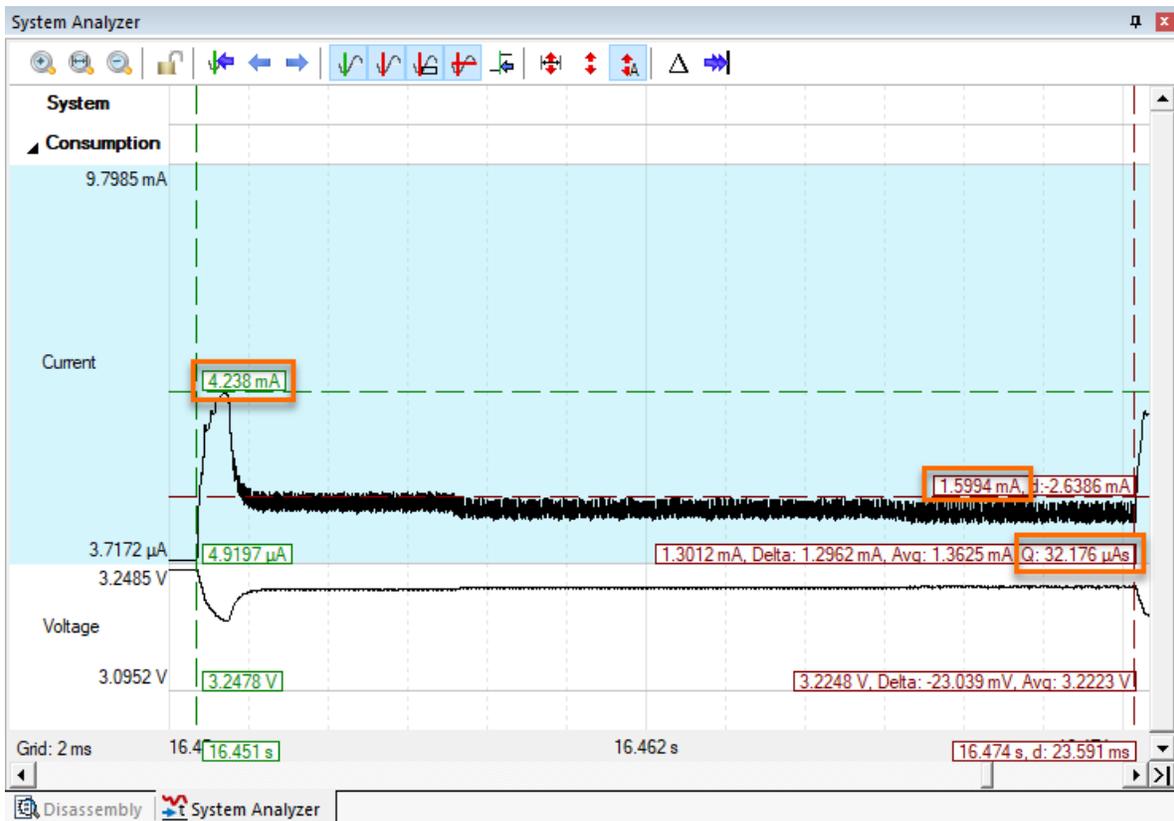


Figure 4 Power consumption profile on the reference design when triggering on-board sensors

There is a 4.2 mA current spike at the beginning when the MCU is running at full speed (48 Mhz), communicating over I2C. This is expected since 4 mA is the typical MCU run current (according to the MCU's datasheet). After that, the MCU should wait for around 20 ms in low power mode. The expected current is around 300  $\mu$ A which is the current required by the sensors during measurement (according to the sensor datasheets). Therefore, the actual current of around 1.6 mA seems to be too high and should be investigated. The total electrical charge for triggering sensors according to the reading in  $\mu$ Vision's System Analyzer is:

$$Q_{\text{ref\_triggerSensors}} = 32.2 \mu\text{As}$$

## Reading sensor data

Sensor data is read over I2C in polling mode.

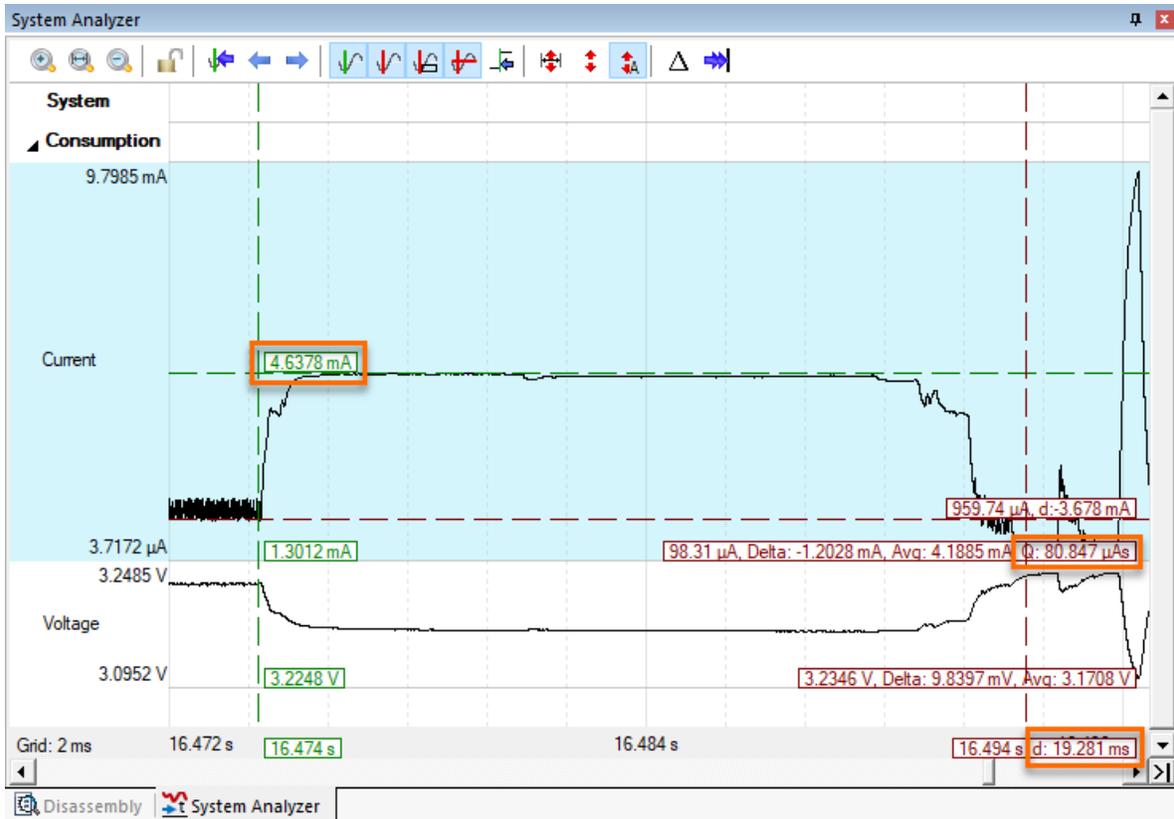


Figure 5 Power consumption profile on the reference design when reading sensor data

The MCU is active during the entire read process which matches the measured average current of 4.6 mA. However, the whole read process takes about 19 ms which seems to be long due to the small amount of executed I2C transfers. Therefore, the read time needs to be investigated and it should also be considered to use event-driven I2C communication rather than polling mode. Total electrical charge for reading sensors:

$$Q_{\text{ref\_readSensors}} = 81 \mu\text{As}$$

## Broadcasting

Sensor data is broadcasted using the Bluetooth Low Energy (BLE) protocol.

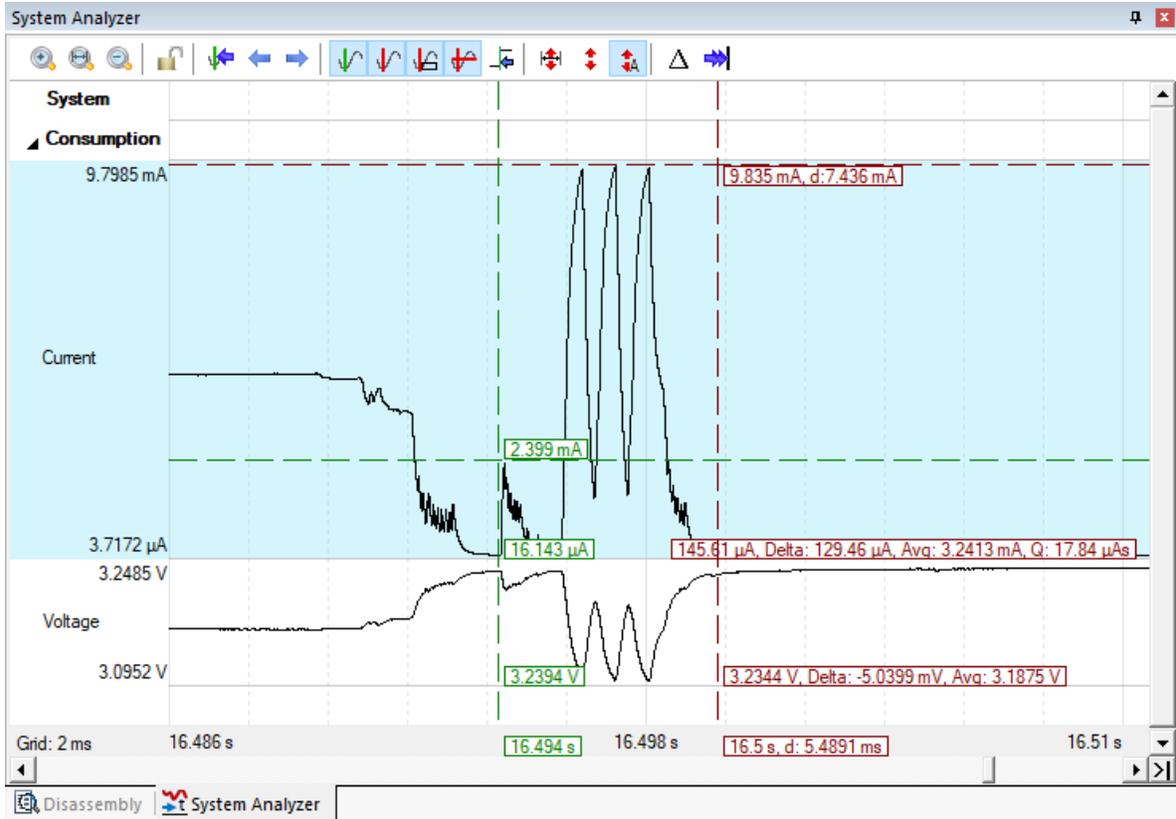


Figure 6 Power measurement profile on the reference design while broadcasting data

First, the active MCU prepares the data and configures the BLE. This explains the first smaller current spike. After that, BLE broadcasts the data over 3 channels. This explains the three similar 10 mA current spikes. Total electrical charge for broadcasting:

$$Q_{\text{ref\_broadcast}} = 17.8 \mu\text{As}$$

## Sleeping

After the data has been broadcasted, the MCU and all sensors go back to low power mode and should draw as little current as possible.

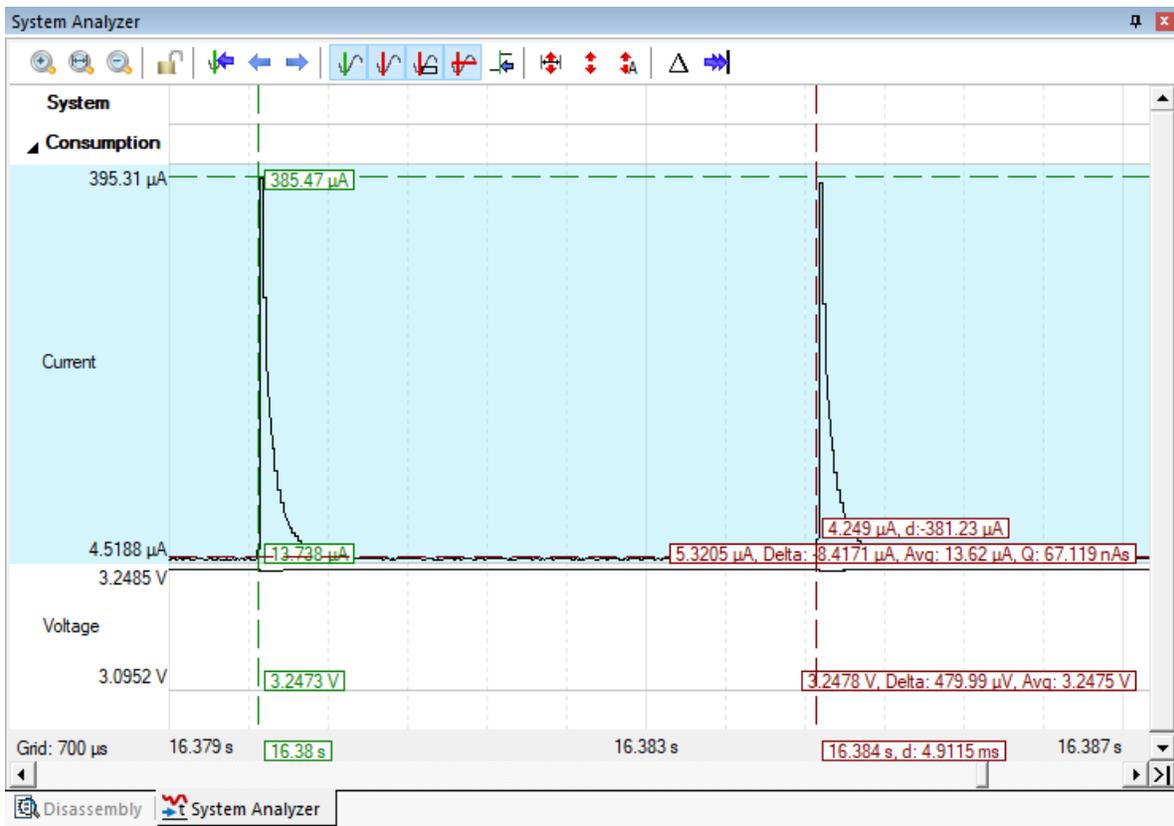


Figure 7 Power measurement profile on the reference design during sleep phase

The entire system is powered by a battery. However, it is not connected directly, but through a DC/DC switcher (which has a better efficiency than a linear regulator). The switcher introduces current pulses with an average 4.9 ms period. The residual 4.5 µA current seems strange and should be investigated.

Total electrical charge for a single pulse:

$$Q_{\text{ref\_sleepPulse}} = 67 \text{ nC}$$

Average pulse period:

$$t_{\text{ref\_sleepPulse}} = 4.9 \text{ ms}$$

Average current during sleep:

$$I_{\text{ref\_sleep}} = Q_{\text{ref\_sleepPulse}} / t_{\text{ref\_sleepPulse}} = 13.7 \text{ µA}$$

The expected sleep current is around 8 µA (total for all chips: MCU, sensors, DC/DC switcher). The actual sleep current is roughly 5 µA higher (same as the residual value) and should be investigated.

## Software investigations

### Unexpected high current during sensor trigger phase

Due to a bug in the software, an extra I2C write operation was issued which led to increased sensor current. After removing that code, the current dropped from around 1.6 mA to the expected 300  $\mu$ A during the sensor trigger phase. This reduces the electrical charge for the *Triggering Sensors* phase from 32  $\mu$ C to 6.5  $\mu$ C.

### Long I2C transfer time during sensor read phase

The long I2C transfer time was caused due to the same software bug which not just increased the sensor current but also caused I2C clock stretching by the sensor. The read phase is drastically shortened after removing the extra I2C write. This significantly reduces the electrical charge for the *Reading sensor data* phase from 81  $\mu$ C to just 7.3  $\mu$ C.

### Event-driven I2C transfers vs. polling mode

Using event-driven I2C transfers enables the MCU to go to sleep while waiting for the I2C transfer to complete (rather than running at full power while polling the I2C transfer state). This further reduces the required current during the *Reading sensor data* phase. However, the impact is small due to the already shortened I2C transfer time.

## Redesigned software

The software of the reference design was updated based on findings from the software investigations. The I2C write operation bug was fixed and event-driven communication was introduced. The newly measured power profile shows the updated power intensive cycle (which is much shorter now with 38 ms):



Figure 8 Power measurement profile after software redesign

The total electrical charge for the active cycle is now:

$$Q_{\text{active}} = 31 \mu\text{As}$$

## Triggering sensors

Triggering the sensors and waiting until data is ready takes the same times as in the reference design. However, the current during measurement is much smaller (264  $\mu\text{A}$  instead of 1.5 mA) due to a smaller sensor current:



Figure 9 Power measurement profile for triggering sensors after software redesign

Total electrical charge for triggering sensors:

$$Q_{\text{triggerSensors}} = 6.2 \mu\text{As}$$

## Reading sensors

Reading the sensor data is much faster than in the reference design due to removed clock stretching (3 ms instead of 19 ms):



Figure 10 Power measurement profile for reading sensors after software redesign

The total electrical charge for reading sensors is:

$$Q_{\text{readSensors}} = 6.6 \mu\text{As}$$

## Broadcasting

The broadcasting phase has not changed in comparison to the reference design.

## Sleeping

The sleeping phase has not changed in comparison to the reference design. Average current during sleep:

$$I_{s\text{leep}} = 13.7 \mu\text{A}$$

## Hardware investigations

The **Sleeping** phase measurements show an average current of 13.7  $\mu\text{A}$  which is higher than the expected 9  $\mu\text{A}$  (based on the datasheets). To isolate the problem, the application was modified so that the MCU immediately entered low power mode without any other functionality (no sensor read or BLE broadcast). In this mode, current pulses occur with an average 4.9 ms period (same as in the application). The residual 5  $\mu\text{A}$  current still exists.

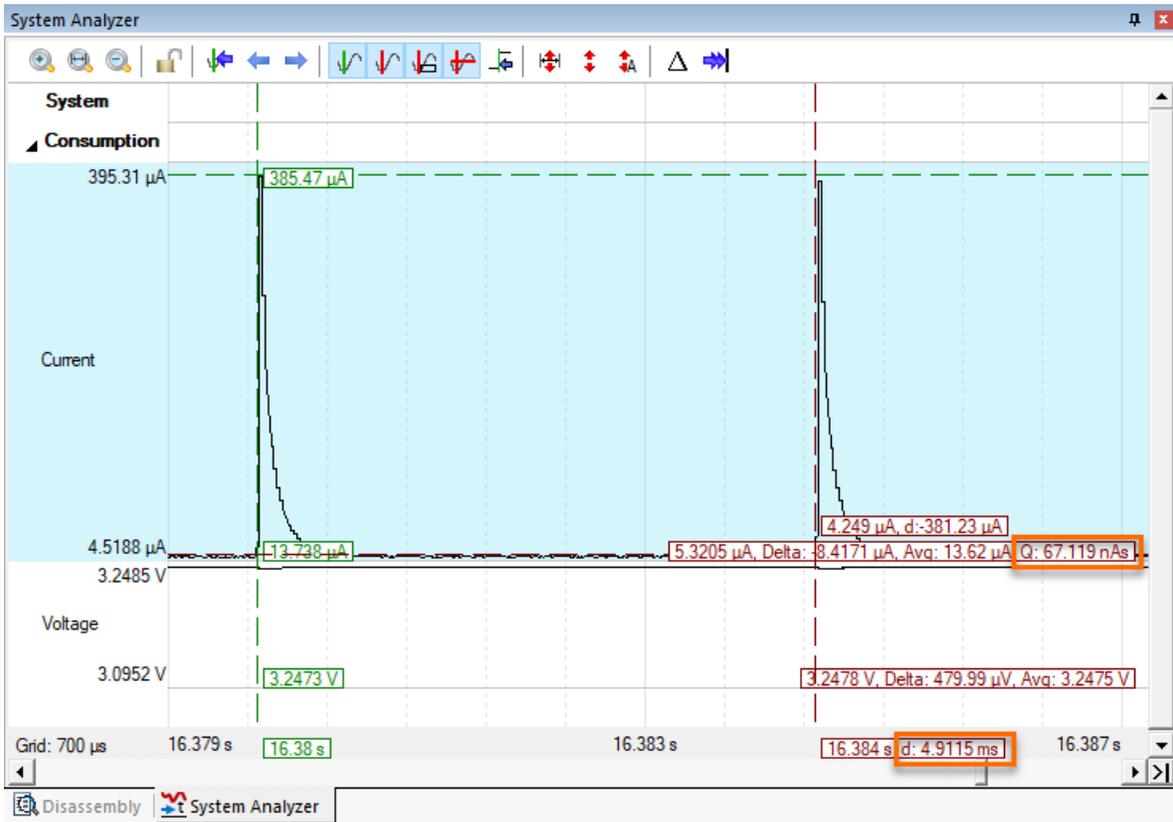


Figure 11 Power measurement profile for the sleeping phase before optimization

The total electrical charge for a single pulse is:

$$Q_{\text{test\_sleepPulse}} = 67 \text{ nAs}$$

Average pulse period:

$$t_{\text{test\_sleepPulse}} = 4.9 \text{ ms}$$

Average current during sleep:

$$I_{\text{test\_sleep}} = Q_{\text{test\_sleepPulse}} / t_{\text{test\_sleepPulse}} = 13.7 \mu\text{A}$$

Despite the slightly smaller peaks, the pulses are a bit wider and carry the same electrical charge. So, the average sleep current for the application or the low power test is the same (DC/DC switcher and the capacitor network only influence the waveforms but not the average value).

To investigate further, the off state of test application has been measured:

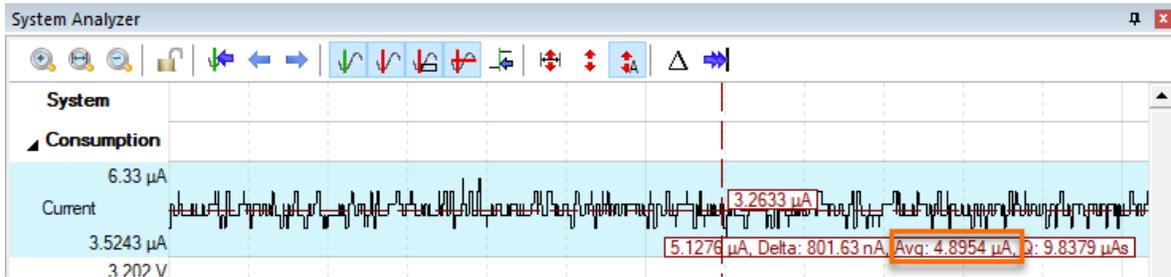


Figure 12 Power measurement profile for the "off" state

The expected current is zero. However, the actual measured current is 4.9 µA. This hints to a hardware issue which is addressed in the hardware redesign.

### Hardware redesign

The hardware design was analyzed to determine the source of the additional 5 µA current consumption. The analysis showed that this was caused by the reverse current of the protection diode D2.

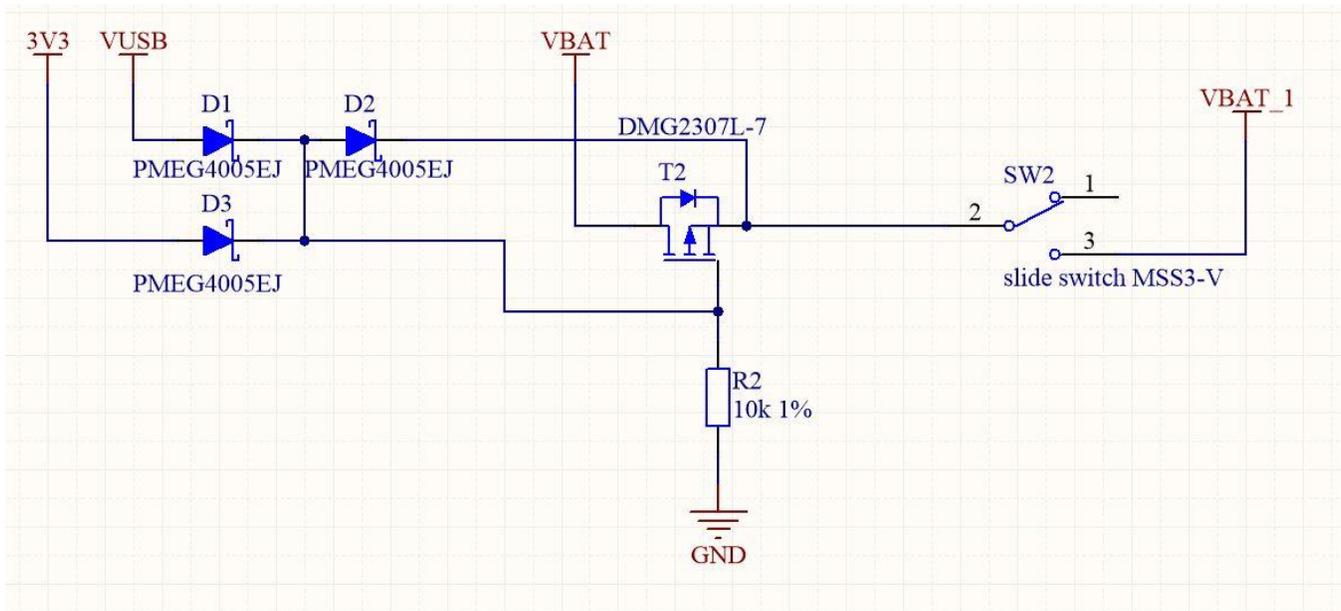


Figure 13 Original schematic

Replacing the original D2 diode PMEG4005EJ with PMEG6010CEH almost eliminates this current (< 0.2 µA). This leads to an average current during sleep:

$$I_{s\text{leep}} = 8 \mu\text{A}$$

## Battery life calculation

The FF1502 runs from a CR2032 battery with the following capacity:

$$C_{\text{battery}} = 230 \text{ mAh}$$

The FF1502 samples and broadcasts sensor values every 10 seconds:

$$t_{\text{broadcast\_interval}} = 10 \text{ s}$$

### Reference design

$$I_{\text{ref\_active}} = (Q_{\text{ref\_active}} - I_{\text{ref\_sleep}} * t_{\text{ref\_active}}) / t_{\text{broadcast\_interval}} = 13.8 \text{ }\mu\text{A}$$

$$I_{\text{ref\_total}} = I_{\text{ref\_sleep}} + (Q_{\text{ref\_active}} / t_{\text{broadcast\_interval}})$$

$$I_{\text{ref\_total}} = 13.7 \text{ }\mu\text{A} + (131 \text{ }\mu\text{As} / 10 \text{ s}) = 26.8 \text{ }\mu\text{A}$$

$$t_{\text{ref\_BatteryLife}} = C_{\text{battery}} / I_{\text{ref\_total}} = 358 \text{ days}$$

### Redesigned software and hardware

$$Q_{\text{active}} = 31 \text{ }\mu\text{As}$$

$$I_{\text{total}} = I_{\text{sleep}} + (Q_{\text{active}} / t_{\text{broadcast\_interval}})$$

$$I_{\text{total}} = 8 \text{ }\mu\text{A} + (31 \text{ }\mu\text{As} / 10\text{s}) = 11.1 \text{ }\mu\text{A}$$

$$t_{\text{BatteryLife}} = C_{\text{battery}} / I_{\text{total}} = 863 \text{ days}$$

## Conclusion

Today you can find a rich portfolio of Cortex-M based microcontroller devices. Well suited for low-power and battery operation, these devices combine fast computing and comprehensive peripherals with multiple power saving modes.

Optimizing embedded applications for overall efficiency should be an integral part of the development process as it is important to understand how peripherals, software algorithms, and power saving modes work together. To simplify this task, we developed **ULINKplus**, a universal debug and trace adapter that supports power measurement. Together with **Keil MDK**, this debug unit gives you unparalleled insight into the operation of your microcontroller applications. It shows which parts of the application code consume more power and helps you to verify the usage of low-power configurations.

**ULINKplus** is designed for software engineers and is easy to connect to the target hardware. Compared to oscilloscopes, it does not require complex configuration settings and shows a higher dynamic range allowing you to spot small current differences. It even synchronizes the power measurement data with the program execution information when using trace or event annotations.

In this example, we demonstrated that the overall energy used by an existing application can be well optimized through improvements to software and hardware. The original battery life was more than doubled as a result of optimization.