

# AN1188: EFP01 Coulomb Counting



This application note describes the configuration, calibration, and operation of the Coulomb counter on the EFP01 Energy Friendly Power Management IC (PMIC). The integrated Coulomb counter can losslessly measure the charge drawn from each DCDC and LDO. A software example demonstrating the Coulomb counting feature is included in this application note.

## KEY POINTS

- Ability to losslessly measure charge drawn from each DCDC and LDO.
- Internally integrated calibration registers for accurate calibration.
- Example C-code
  - Multiple IDE projects

## 1. Device Compatibility

This application note supports the EFP01 Energy Friendly PMIC product family.

The EFP01 devices consist of:

- EFP0104
- EFP0108
- EFP0109
- EFP0111

## 2. System Overview

### 2.1 Introduction

The EFP01 Energy Friendly PMIC product family is designed to support a range of battery-powered applications, as well as other systems requiring high performance and low energy consumption.

A block diagram of the EFP01 family is shown in [Figure 2.1 EFP01 Block Diagram on page 3](#). The diagram shows a superset of features available on the family, which vary by part number. For more information about specific device features, consult the Ordering Information section in the datasheet.

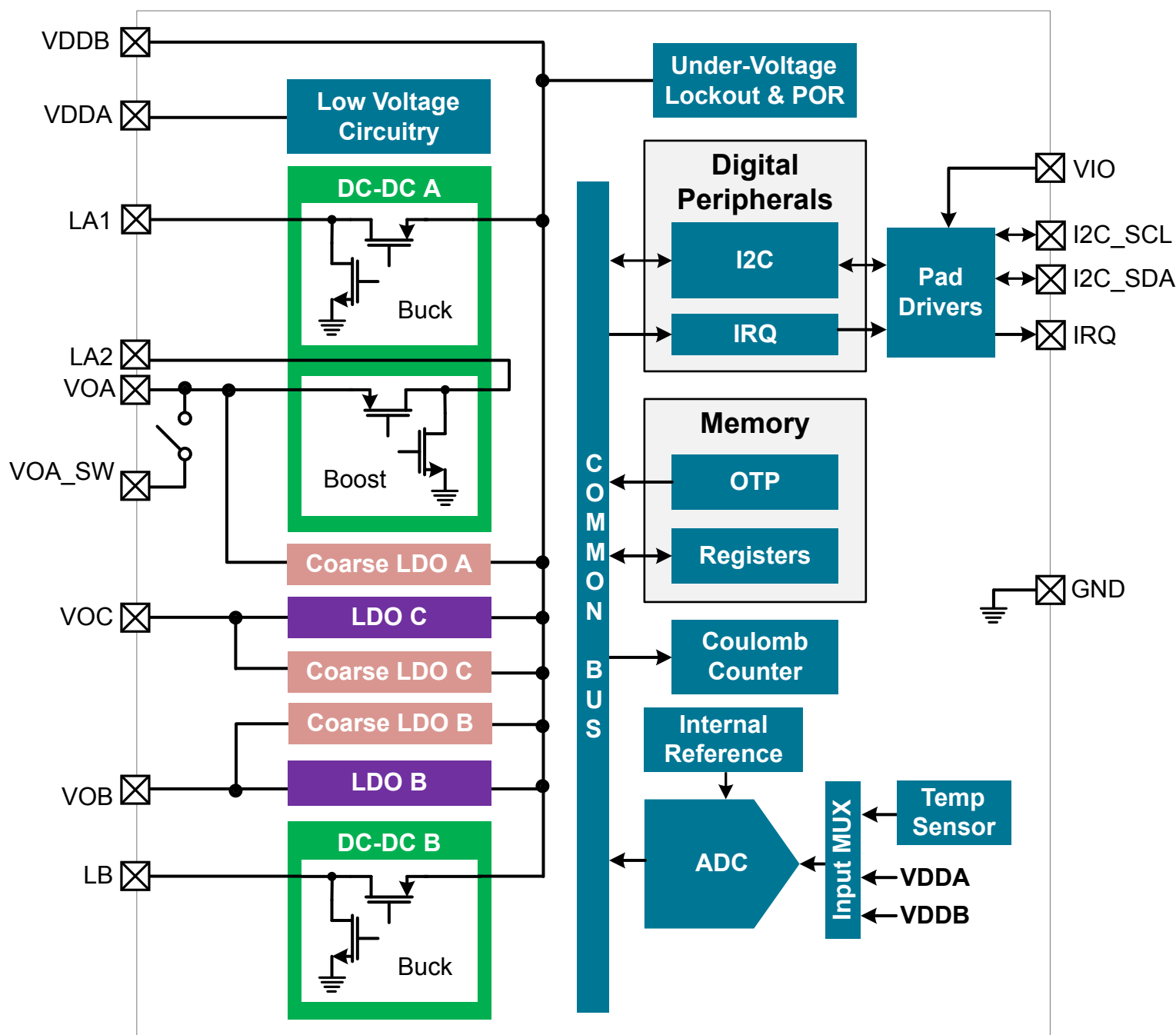


Figure 2.1. EFP01 Block Diagram

## 2.2 Power Modes

The EFP01 can provide up to three voltage rails for EFR32 and EFM32 devices from a single input supply voltage.

The EFP01 has two independent DCDC converters: DCDC A and DCDC B, each requiring an external inductor. DCDC A can use buck/boost, wired buck, or wired boost configurations, while DCDC B only supports buck configuration.

In addition, the EFP01 has two integrated LDOs supplied from the VDDB pin: LDO B and LDO C. LDO B is internally wired in parallel with DCDC B. LDO C can be used independently, or it can be externally wired in parallel with DCDC A.

Finally, each EFP01 output (VOA, VOB, VOC) has its own coarse regulator in parallel for use in EM4. The coarse regulators have very low quiescent current draw, but have poor output regulation (e.g., the output may range from ~1.7 to 3.4 V) and can only support very light loads (~100  $\mu$ A).

## 2.3 Energy Modes

The EFP01 operates in 3 different energy modes to optimize efficiency based on the expected load: EM0, EM2, and EM4. The Coulomb counter is available in EM0 and EM2 modes. In EM4 mode, all DCDCs and LDOs are disabled, and the load is powered by the coarse regulators. Since the Coulomb counter is exclusive to the DCDCs and the LDOs, it is not available in EM4 mode.

### 3. Coulomb Counter

The EFP01's DCDC converters use pulse-frequency modulation (PFM) to drive each DCDC and LDO output (details on PFM DCDC operation can be found in [AN1187: EFP01 Design Considerations](#)). The EFP01's Coulomb counter counts the number of PFM pulses delivered to each of its outputs. If the charge-per-pulse (CPP) is known, the total charge from an output can be calculated:

$$\text{Total Charge} = \text{Number of PFM pulses} \times \text{CPP}$$

#### 3.1 Calibration

Before use, the EFP01's Coulomb counter must be calibrated to determine the CPP. To aid in calibration, the EFP01 can provide 8 different current loads (4 high and 4 low), which can be enabled on each of the 3 outputs: VOA, VOB, and VOC. Once the known internal current load is applied to the desired output, the EFP01 counts the number of 10 MHz clock cycles required for a fixed number of PFM pulses to occur.

The EFP01 calibration registers/bitfields consist of the following:

- CC\_CAL.CCL\_SEL: Selects the output to calibrate (VOA, VOB, or VOC)
- CC\_CAL.CCL\_LVL: Selects the magnitude of the current load applied to the selected output
- CC\_CAL.CC\_CAL\_NREQ: Selects the number of PFM pulses to count over during calibration
- CCC\_MSBY/CCC\_LSBY: During calibration, this 16-bit register pair is retasked to store the number of 10 MHz clock cycles that were counted. The CCC\_MSBY/CCC\_LSBY registers are always used during calibration, regardless of which output is calibrated.

After the CC\_CAL register is configured, calibration can be initiated by setting the CC\_CAL\_STRT bit in the CMD register. The Coulomb counter should be disabled before initializing the calibration. If the Coulomb counter was previously enabled, it can be disabled by writing a 0 to the CC\_EN bit in the CC\_CTRL register.

Calibration status can be monitored via the CCC\_ISDONE bit in the STATUS\_G register. This flag can additionally be configured to generate an interrupt by writing a 1 to the CCC\_ISDONE\_UNMASK bit in the STATUS\_GM register. Once the CCC\_ISDONE flag is set, indicating that the EFP01's calibration process is complete, the host firmware can calculate the CPP. Given the load current, number of PFM pulses, and calibration time, the equation to calculate CPP is given by:

$$\text{CPP} = \frac{\text{Calibration Time} \times \text{EFP Load Current}}{\text{Number of PFM pulses}}$$

In reality, the system processor also draws some load current during calibration, so a more accurate equation is:

$$\text{CPP} = \frac{\text{Calibration Time} \times (\text{EFP Load Current} + \text{System Load Current})}{\text{Number of PFM pulses}}$$

Since the System Load Current is unknown, the user can't accurately calculate a CPP value with a single calibration. Two calibrations must be taken: one with a large EFP Load Current, and another with a small EFP Load Current.

The equation for the large load current is:

$$I_{\text{large load}} + I_{\text{system}} = \frac{\text{CPP} \times \text{NumPulses}_{\text{large}}}{\text{Time}_{\text{large}}}$$

The equation for the small load current is:

$$I_{\text{small load}} + I_{\text{system}} = \frac{\text{CPP} \times \text{NumPulses}_{\text{small}}}{\text{Time}_{\text{small}}}$$

Combining these two equations give:

$$\text{CPP} = \frac{\frac{I_{\text{large load}}}{\text{NumPulses}_{\text{large}}} - \frac{I_{\text{small load}}}{\text{NumPulses}_{\text{small}}}}{\frac{\text{Time}_{\text{large}}}{\text{NumPulses}_{\text{large}}} - \frac{\text{Time}_{\text{small}}}{\text{NumPulses}_{\text{small}}}}$$

Given that NumPulses =  $2^{(\text{NREQ} + 1)}$  and Time = Count /  $10^7$ , we can expand the equation to:

$$\text{CPP} = \frac{\frac{I_{\text{large load}}}{10^7 \times 2^{\text{NREQ}_{\text{large}}+1}} - \frac{I_{\text{small load}}}{10^7 \times 2^{\text{NREQ}_{\text{small}}+1}}}{\frac{\text{Count}_{\text{large}}}{10^7 \times 2^{\text{NREQ}_{\text{large}}+1}} - \frac{\text{Count}_{\text{small}}}{10^7 \times 2^{\text{NREQ}_{\text{small}}+1}}}$$

Note that this solution doesn't require any knowledge of the system load current — however, it assumes that the system load current remains relatively constant during the duration of the calibration measurements.

### 3.1.1 NREQ Calculation and Overflow Prevention

The CCC\_MSBY and CCC\_LSBY registers don't have overflow protection during calibration (i.e., the CC\_FULL status flag is never set nor is an interrupt ever requested). Therefore, the CC\_CAL\_NREQ value should be selected to ensure that the maximum number of 10 MHz clock cycles can be counted without overflow. A maximum allowed NREQ value can be safely determined using the procedure described below:

1. Set NREQ = 0 (2 PFM pulses)
2. Set CC\_LVL to the current load you plan to run the calibration with
3. Set CMD.CC\_CAL\_STRT = 1 and read the resulting COUNT from the CCC\_MSBY and CCC\_LSBY result registers to determine the count-per-pulse:

$$COUNTPERPULSE = \frac{COUNT}{2^{NREQ+1}} = \frac{COUNT}{2}$$

4. Because the combined COUNT register width is 16-bits, the maximum number of PFM pulses we can generate without overflow is given by:

$$MAX PULSES = \frac{2^{16}}{COUNTPERPULSE}$$

5. Therefore, the maximum supported NREQ without risk of overflow is given by:

$$NREQ_{max} = \text{ROUNDDOWN}\left(\log_2\left(\frac{2^{17}}{COUNT}\right) - 1\right)$$

## 3.2 Recalibration

Changes in external and internal conditions of the device can affect the CPP. Due to the following changes, a recalibration event may be necessary to maintain an accurate CPP:

- Significant change in input supply voltage
- Significant change in temperature
- Change in DCDC Operating Mode
- Change in Energy Mode

### 3.2.1 Input Supply Voltage

A significant change in input supply voltage can affect the CPP. When using a power supply that can change voltage, such as a battery, it is important to periodically check the input supply voltage and recalibrate the Coulomb counter accordingly. The supply voltage applied to either the VDDA or VDDDB pin of the EFP01 can be read using the VDD\_AVG\_MSN and VDD\_AVG\_LSBY registers.

In the EFP0111 Boost Bootstrap configuration, the supply voltage is not connected to either the VDDA or VDDDB pins, so the EFP01 cannot measure the supply voltage. Instead, the host processor's ADC should monitor the supply voltage.

### 3.2.2 Temperature

A significant change in temperature can affect the CPP. It is important to periodically check the temperature of the EFP01 and recalibrate the Coulomb counter accordingly. The EFP01's internal die temperature can be read from the TEMP\_MSN and TEMP\_LSBY registers.

### 3.2.3 DCDC Operating Mode

The CPP may vary depending on the operating mode of the converter. For example, if DCDC B is configured in buck with LDO mode (i.e., the converter automatically switches between buck and LDO modes depending on input voltage), the CPP when the converter is in buck can differ from the CPP when the converter is operating in LDO mode. For this reason, it may be necessary to calibrate a given DCDC converter in each of its expected operating modes. Thus if DCDC B is in buck with LDO mode, and the battery voltage is sufficiently close to the output voltage such that the converter may switch to LDO mode, calibration should be performed on DCDC B once in buck mode and again in LDO mode.

The current DCDC operating modes of DCDC A and DCDC B are reported by the CCA\_MODE and CCB\_MODE fields of the CC\_MODE register, respectively. For calibration purposes, the operating modes can be temporarily forced using the BB\_MODE and BK\_MODE fields in the BB\_CTRL3 and BK\_CTRL1 registers for DCDC A and DCDC B, respectively. Host firmware is expected to maintain a CPP for each operating mode of the converter.

### 3.2.4 Energy Mode

The EFP01's energy mode can affect the CPP if the peak current settings in EM0 and EM2 are different.

The VOB output has independent result registers for EM0 and EM2, and may need to be calibrated in each energy mode if the output target voltage or the peak current settings differ between EM0 and EM2.

The VOA output has a shared result register pair for EM0 and EM2. The BB\_IPK and BB\_IPK\_EM2 peak current values are expected to be the same to ensure a consistent CPP for VOA, regardless of energy mode.

## 3.3 Operation

During operation, the Coulomb counter stores the count of PFM pulses into the corresponding register pair. Each register pair is composed of a register that contains the most significant byte of the pulse count, and another register that contains the least significant byte of the pulse count. There are four result register pairs:

- CCA\_MSBY/CCA\_LSBY: Stores pulse count for VOA in EM0/2
- CCB0\_MSBY/CCB0\_LSBY: Stores pulse count for VOB in EM0
- CCB2\_MSBY/CCB2\_LSBY: Stores pulse count for VOB in EM2
- CCC\_MSBY/CCC\_LSBY: Stores pulse count for VOC in EM0/2

When DCDC A is operating with LDO C in parallel (e.g., the EFP0104), the resulting counts are split between the CCA and CCC result registers. When DCDC A is powering the load, the counts are stored in the CCA result register. When LDO C is powering the load, the counts are stored in the CCC result registers, even though LDO C is driving the VOA output.

### 3.3.1 Prescaler

The actual value stored in a given Coulomb counter result register pair is scaled according to the CC\_PRESCAL field in the CC\_CTRL register. This setting applies globally such that the count in a given result register pair represents  $2^{(16 - (2 \times \text{CC\_PRESCAL}))}$  PFM pulses. Note that the prescaler setting does not affect the CCC\_MSBY/LSBY registers during calibration.

### 3.3.2 Enabling/Disabling

In order to start Coulomb counting, write a 1 to the CC\_EN bit in the CC\_CTRL register.

In order to stop Coulomb counting, write a 0 to the CC\_EN bit in the CC\_CTRL register.

### 3.3.3 Servicing

Once enabled, the Coulomb counter result registers will eventually overflow, so some amount of firmware maintenance is required. The CC\_THRSH field in the CC\_CTRL register sets the desired overflow threshold (50%, 62.5%, 75%, or 87%) for setting the CC\_FULL flag in the STATUS\_G register. Note that the CC\_FULL\_UNMASK bit in the STATUS\_GM register must be set to 1 so that an interrupt can be requested when the CC\_FULL flag is set.

When the firmware receives a CC\_FULL interrupt, all relevant Coulomb counter result registers should be read and added to local host firmware variable counts. Additionally, each converter's operating mode should be determined (by reading the CCA\_MODE or CCB\_MODE field in the CC\_MODE register for DCDC A or DCDC B, respectively) in order to perform battery life calculations using the relevant CPP value. The Coulomb counter result registers must be cleared after reading them by writing a 1 to the CC\_CLR bit in the CMD register.

## 4. Firmware Configuration and Maintenance

### Firmware Configuration and Maintenance

The EFP01 calibration registers are configured by a host device. The host can write to the EFP01's CC\_CAL register to configure the calibration routine as guided in [3.1 Calibration](#). Once calibration is complete, the host can enable the Coulomb counter by setting the CC\_EN bit in the CC\_CTRL register. Afterwards, the host can follow the operation routine as described in [3.3 Operation](#).

The EFP01's Coulomb counter can function with minimal software intervention once it's been started. Below are the maintenance steps required by the host firmware:

- The Coulomb counter result registers will periodically overflow. Read and clear the result registers as instructed in [3.3.3 Servicing](#).
- Calculate the total charge for each output based on the equation shown in [3. Coulomb Counter](#).
- Monitor any change in conditions that could affect the CPP value as indicated in [3.2 Recalibration](#).

The [Figure 4.1 Coulomb Counter Firmware Routine Flowchart on page 9](#) shows the typical procedure of the host firmware. Note that this flowchart is a guideline rather than an absolute standard. However, disabling the Coulomb counter prior to calibration and calibrating prior to operation should be strictly followed.



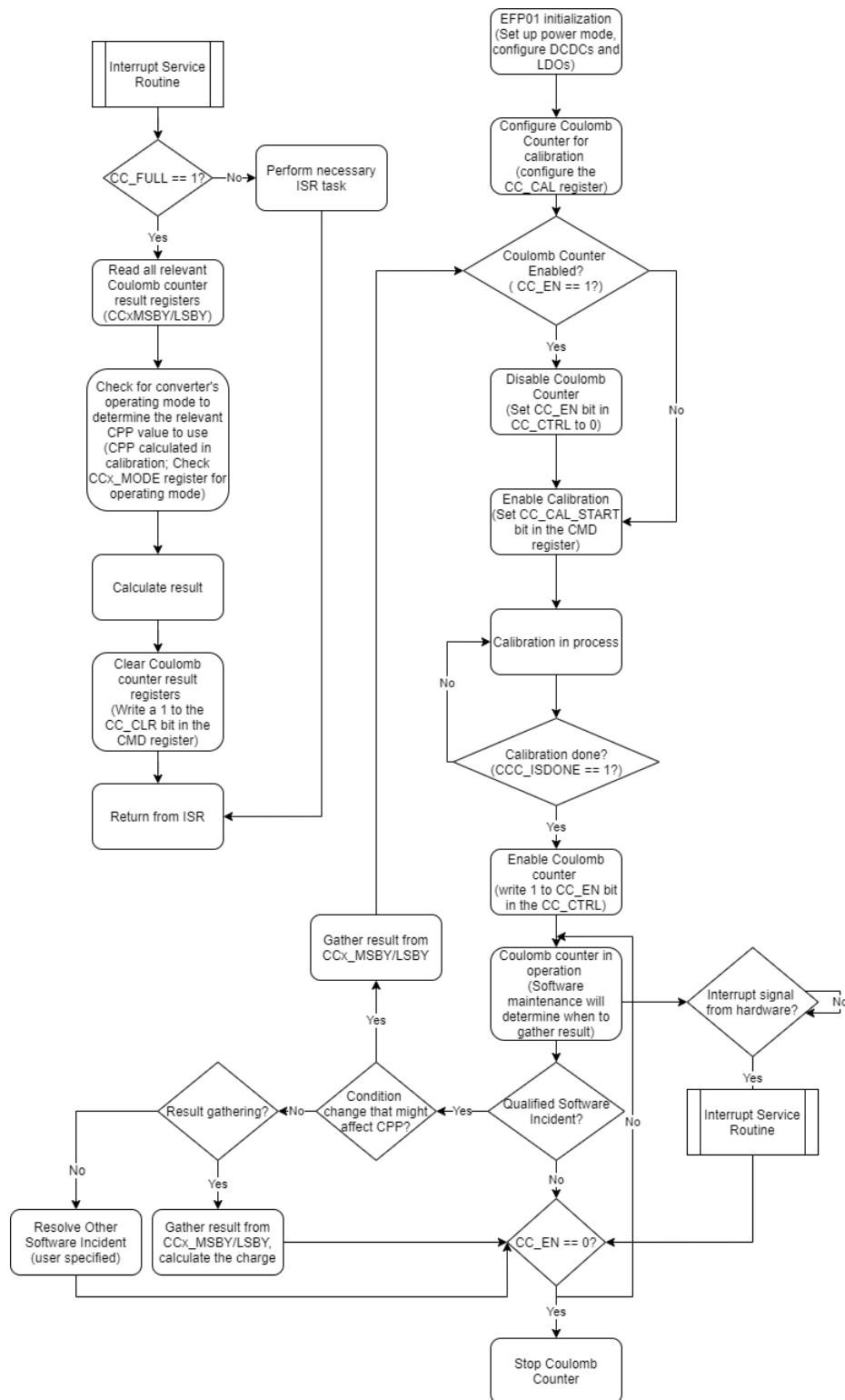


Figure 4.1. Coulomb Counter Firmware Routine Flowchart

## 5. Software Example

### Coulomb Counting with EFR32MG21 + EFP0104

This application note provides a software example that demonstrates the Coulomb counting feature on the BRD4179B SoC board. The BRD4179B is comprised of an EFR32MG21 powered by an EFP0104. This example uses the integrated Coulomb counter to monitor charge and current consumption on the DCDC A output (VOA) and displays the information via the VCOM port.

This software example undergoes three main stages:

- EFR32MG21 (host) and EFP0104 configuration and communication setup
- Coulomb counter calibration
- Coulomb counter operation and calculation of the charge consumption on VOA

[UG422: EFR32xG21 2.4 GHz 10 dBm with EFP Wireless Starter Kit User's Guide](#) contains additional information regarding the BRD4179B SoC board.

## 6. Related Documents

- [EFP01 Energy Friendly PMIC Family Data Sheet](#)
- [AN1187: EFP01 Design Considerations](#)
- [UG422: EFR32xG21 2.4 GHz 10 dBm with EFP Wireless Starter Kit User's Guide](#)

## 7. Revision History

### Revision 0.1

June 2020

- Initial Revision.

Silicon Labs

# Simplicity Studio™4



## Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



**IoT Portfolio**  
[www.silabs.com/IoT](http://www.silabs.com/IoT)



**SW/HW**  
[www.silabs.com/simplicity](http://www.silabs.com/simplicity)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support and Community**  
[community.silabs.com](http://community.silabs.com)

### Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required, or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

### Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, ClockBuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, ISOModem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.  
400 West Cesar Chavez  
Austin, TX 78701  
USA

<http://www.silabs.com>