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:

- EFP0101
- EFP0102
- EFP0103
- EFP0104
- EFP0106
- EFP0107
- EFP0108
- EFP0109
- EFP0110
- 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.

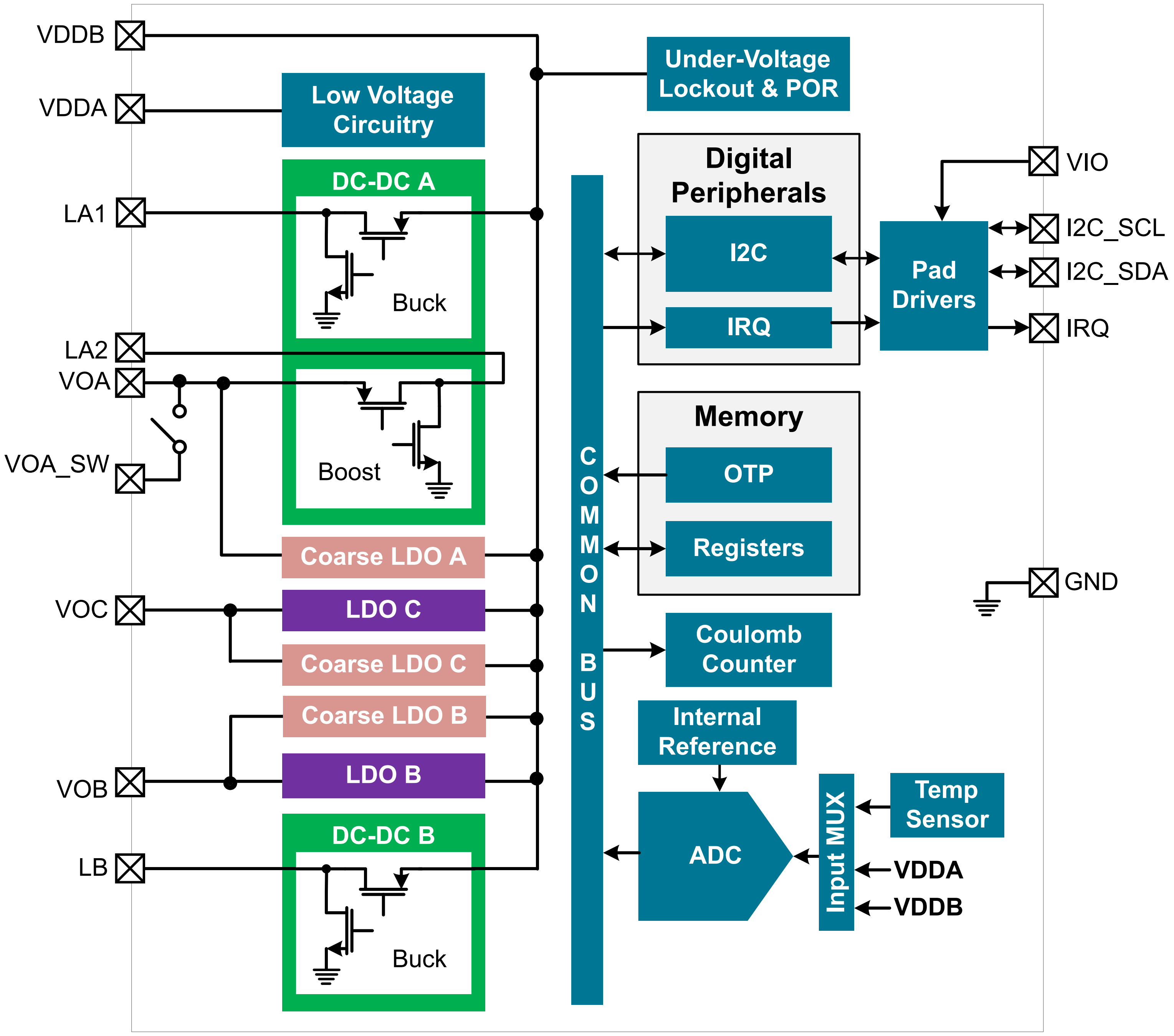


Figure 2.1. EFP01 Block Diagram

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 µA).

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 intializing 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:

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

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

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

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:

$$I_{large\ load} + I_{system} = \frac{CPP \times NumPulses_{large}}{Time_{large}}$$

The equation for the small load current is:

$$I_{small\ load} + I_{system} = \frac{CPP \times NumPulses_{small}}{Time_{small}}$$

Combining these two equations give:

$$CPP = \frac{I_{large\ load} - I_{small\ load}}{\frac{NumPulses_{large}}{Time_{large}} - \frac{NumPulses_{small}}{Time_{small}}}$$

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

$$CPP = \frac{I_{large\ load} - I_{small\ load}}{\frac{10^7 \times 2^{NREQ_{large}+1}}{Count_{large}} - \frac{10^7 \times 2^{NREQ_{small}+1}}{Count_{small}}}$$

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 do not support overflow protection during calibration (i.e., the CC_FULL status flag will never be set and no interrupt will be generated). For optimal accuracy, the CC_CAL_NREQ value should be selected to ensure that the maximum number of 10 MHz clock cycles can be counted without risk of overflow. A maximum allowed NREQ value can safely be 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:

$$NREQ_{max} = ROUNDDOWN\left(LOG_2\left(\frac{2^{17}}{COUNT}\right) - 1\right)$$

3.2 Recalibration

Changes in external and internal conditions of the device can affect the accuracy of the previously determined charge-per-pulse (CPP). For example, recalibration may be required to maintain an accurate CPP after the following events:

- Significant change in input supply voltage
- Significant change in temperature
- Change in DCDC Operating Mode (e.g., Buck mode to LDO mode)
- Change in Energy Mode (e.g. EM0 -> EM2)

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 VDDB 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 VDDB 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 correpsonding register pair. Each register pair is composed 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/CCB_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_PRESCL field in the CC_CTRL register. This setting applies globally such that the count in a given result register pair represents $2^{(16 - (2 \times \text{CC\_PRESCL}))}$ 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

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. Aftwards, 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.

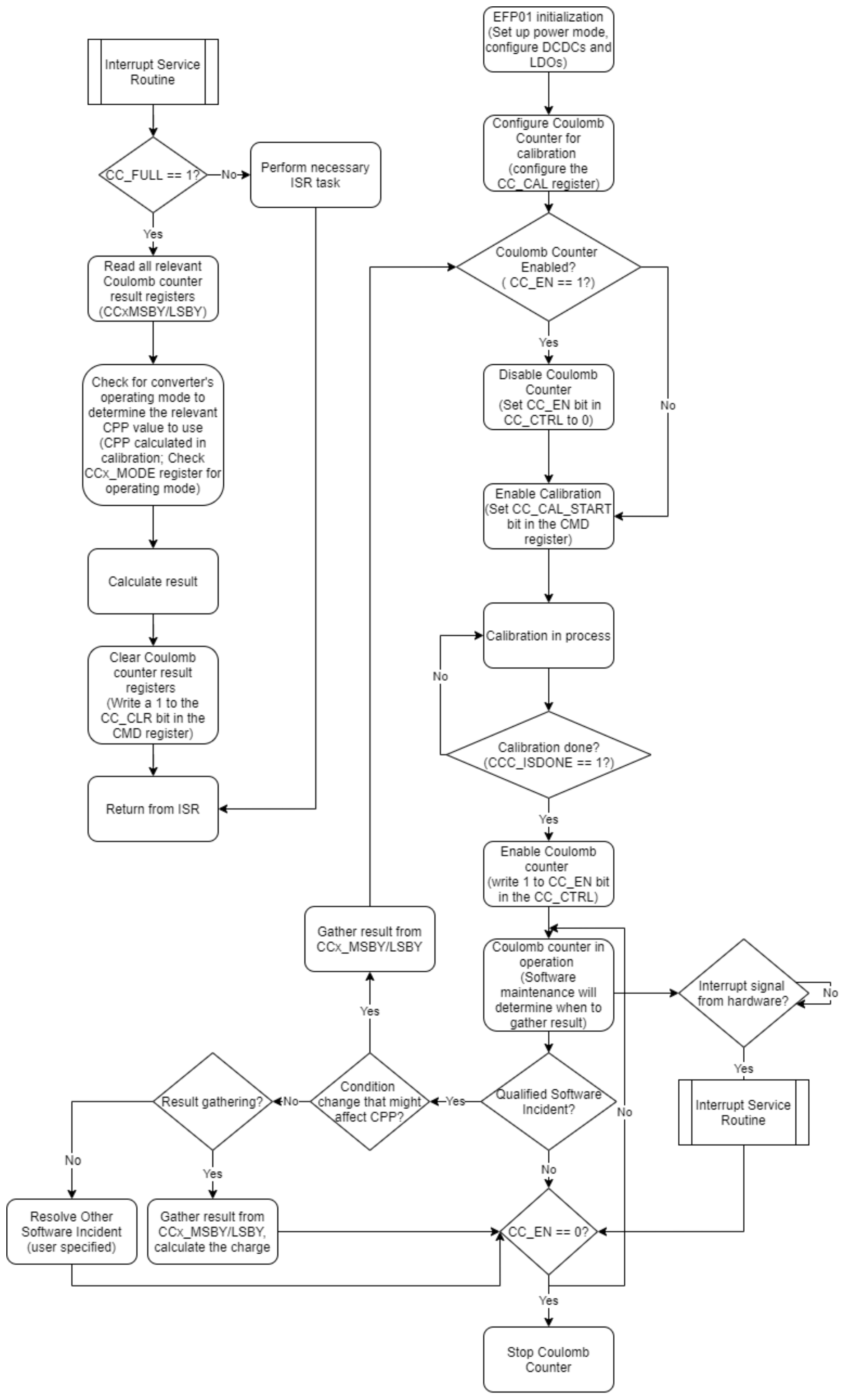

Interrupt Service Routine
CC_FULL == 1?
No
Perform necessary ISR task
Yes
Read all relevant Coulomb counter result registers (CCxMSBY/LSBY)
Check for converter's operating mode to determine the relevant CPP value to use (CPP calculated in calibration; Check CCx_MODE register for operating mode)
Calculate result
Clear Coulomb counter result registers (Write a 1 to the CC_CLR bit in the CMD register)
Return from ISR
EFP01 initialization (Set up power mode, configure DCDCs and LDOs)
Configure Coulomb Counter for calibration (configure the CC_CAL register)
Coulomb Counter Enabled? (CC_EN == 1?)
Yes
No
Disable Coulomb Counter (Set CC_EN bit in CC_CTRL to 0)
Enable Calibration (Set CC_CAL_START bit in the CMD register)
Calibration in process
No
Calibration done? (CCC_ISDONE == 1?)
Yes
Enable Coulomb counter (write 1 to CC_EN bit in the CC_CTRL)
Coulomb counter in operation (Software maintenance will determine when to gather result)
Interrupt signal from hardware?
No
Yes
Interrupt Service Routine
Gather result from CCx_MSBY/LSBY
Yes
Qualified Software Incident?
Yes
Condition change that might affect CPP?
No
Result gathering?
No
Yes
Resolve Other Software Incident (user specified)
Gather result from CCx_MSBY/LSBY, calculate the charge
No
No
CC_EN == 0?
Yes
Stop Coulomb Counter

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 0.3

July 2021

- Added EFP0101, EFP0102, EFP0103 and EFP0106 OPNs to device compatibility list.

Revision 0.2

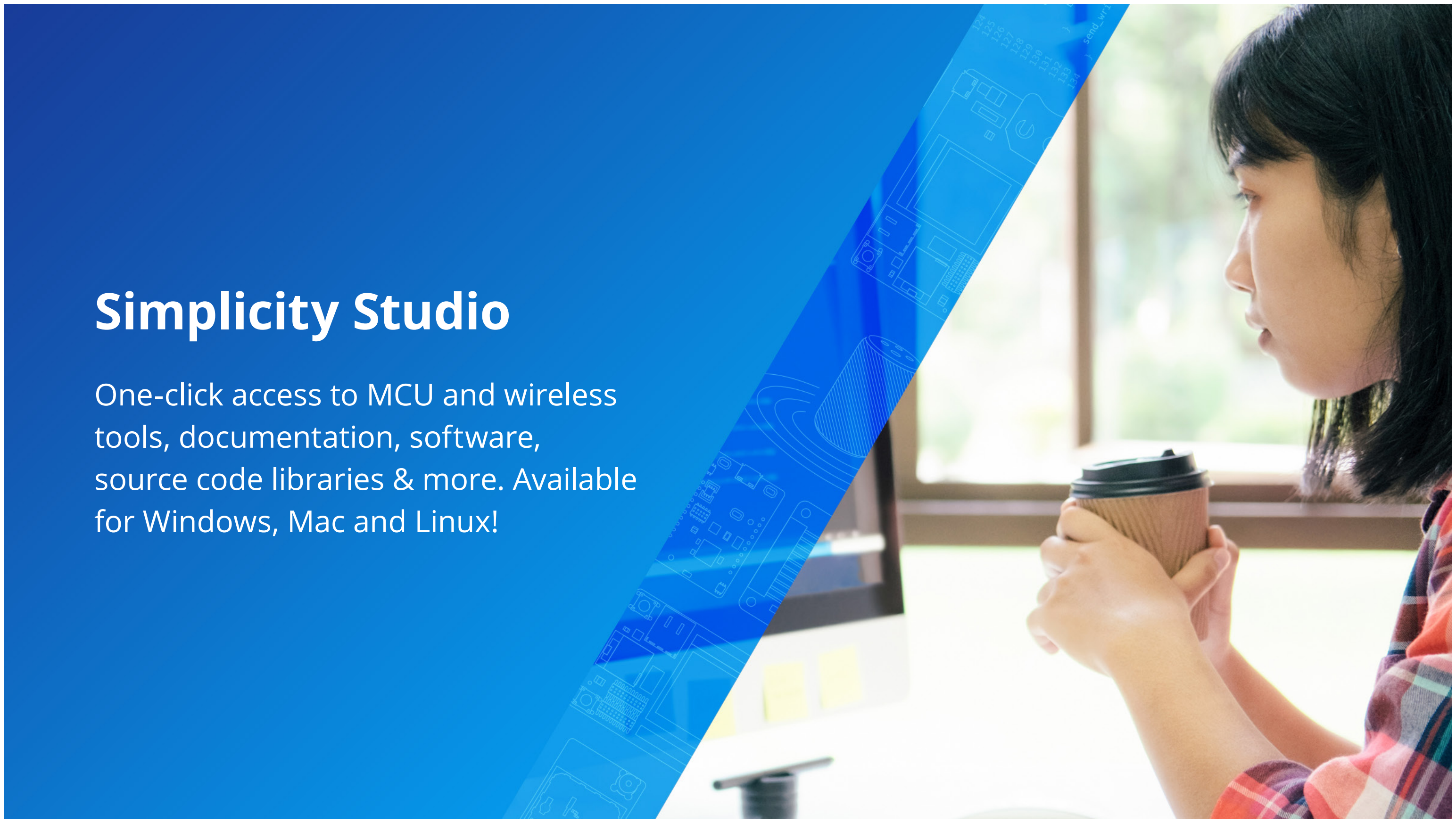
April 2021

- Added EFP0107 and EFP0110 OPNs to device compatibility list.
- Clarifications to 3.1 Calibration section.
- Clarifications to the 3.1.1 NREQ Calculation and Overflow Prevention section.

Revision 0.1

June 2020

- Initial Revision.


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

IoT Portfolio	**SW/HW**	**Quality**	**Support & Community**
www.silabs.com/IoT	www.silabs.com/simplicity	www.silabs.com/quality	www.silabs.com/community

SILICON LABS

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

www.silabs.com