AN1187: EFP01 Design Considerations

This application note provides an overview and the theory of operation for the EFP01 Energy Friendly Power Management IC (PMIC). Hardware design recommendations are provided along with a discussion of the software driver and how to integrate it into a EFR32 Wireless Gecko software project.

KEY POINTS

- EFP01 expands the input supply range of EFR32-based systems.
- EFP01 reduces power consumption of systems that use EFR32 devices without an integrated DCDC converter.
- EFP0111 enables EFR32's radio to operate at higher output power using coin cell batteries.

1. Pulse Frequency Modulation (PFM) Theory

1.1 Ideal Buck Converter Overview

1.1.1 Buck Converter Operation

In a simple, ideal buck converter (as shown in the figure below), the high-side (PMOS) and low-side (NMOS) switches are turned on and off complementarily to generate a lower output voltage from a higher input voltage.

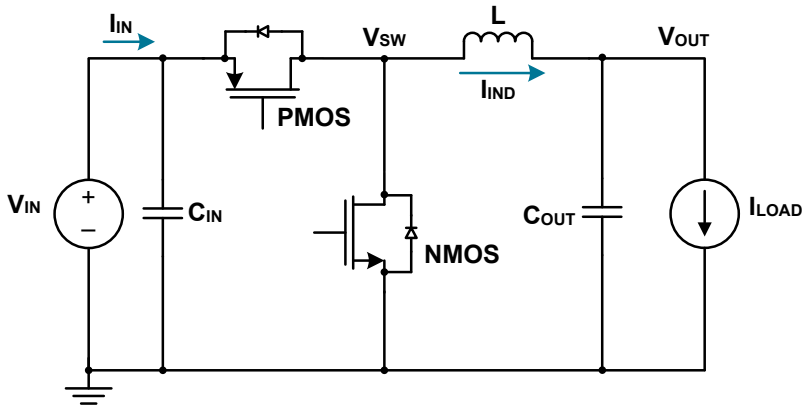


Figure 1.1. Simplified Buck Converter

The current through the inductor (I_{IND}) is typically limited by a fixed or programable peak current setting (I_{PK}). The theoretical maximum output current an ideal buck converter can deliver is $I_{PK}/2$.

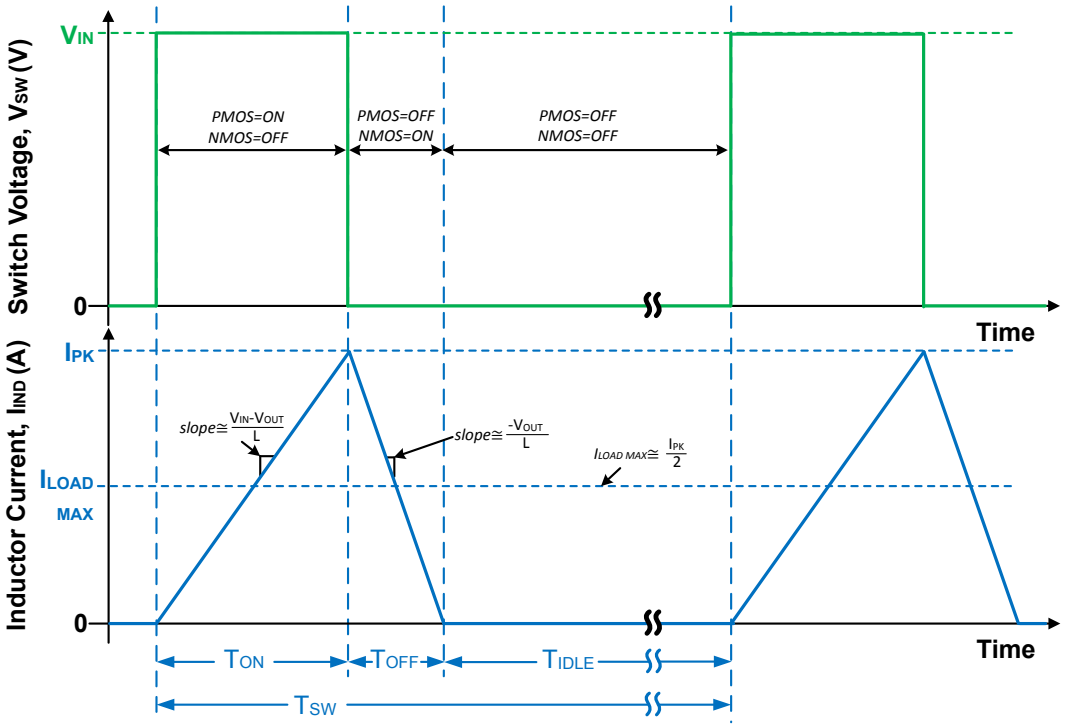


Figure 1.2. Buck PFM Switching Waveforms

The figure above shows the voltage at the switch node (V_{SW}) and the current through the inductor (I_{IND}) as the (assumed ideal) PMOS and NMOS switches are enabled and disabled. Note that this switching waveform is an example of Discontinuous Conduction Mode (DCM) operation, that is, the current through the inductor is allowed to reach zero.

The waveform for a single switching cycle can be divided into three distinct time periods:

1. On-time (T_{ON}): During this time, the PMOS switch is closed (forcing V_{SW} to be equal to V_{IN}) and the NMOS switch is opened. The current through the inductor (I_{IND}) during T_{ON} is determined by the following equation:

   $$I_{IND} = time \times \frac{V_{OUT} - V_{IN}}{L}$$

   Once I_{IND} reaches the I_{PK} current limit, the converter enters the off-time period.
2. Off-time (T_{OFF}): During this time, the PMOS switch is open and the NMOS switch is closed (forcing V_{SW} to be equal to GND). The current through the inductor (I_{IND}) is determined by the following equation:

   $$I_{IND} = time \times \frac{-V_{OUT}}{L}$$

   After I_{IND} reaches zero, the converter enters the idle-time period.
3. Idle-time (T_{IDLE}): During this time, both the PMOS and NMOS switches are open. The current through the inductor (I_{IND}) is zero. When the output voltage (V_{OUT}) droops beyond a set threshold (determined by an internal comparator), the converter will re-enter the on-time period and the cycle repeats. The converter can supply the maximum output load current ($I_{PK}/2$) when T_{IDLE} is zero.

The switching period (T_{SW}) and the switching frequency (F_{SW}) can be determined by the sum of the time periods in the switching cycle, as shown below:

$$T_{SW} = T_{ON} + T_{OFF} + T_{IDLE} \quad and \quad F_{SW} = \frac{1}{T_{SW}}$$

The converter input current can be limited by forcing a minimum switching period, T_{SW}. For a forced minimum switching period, the maximum battery current is determined by:

$$I_{BATT\_LIMIT} = \frac{L \times I_{PK}^2}{2 \times T_{SW} \times (V_{IN} - V_{OUT})}$$

1.1.2 Buck Converter Output Ripple Voltage

The figure below illustrates the contribution of the different buck mode switching periods to the output ripple voltage.

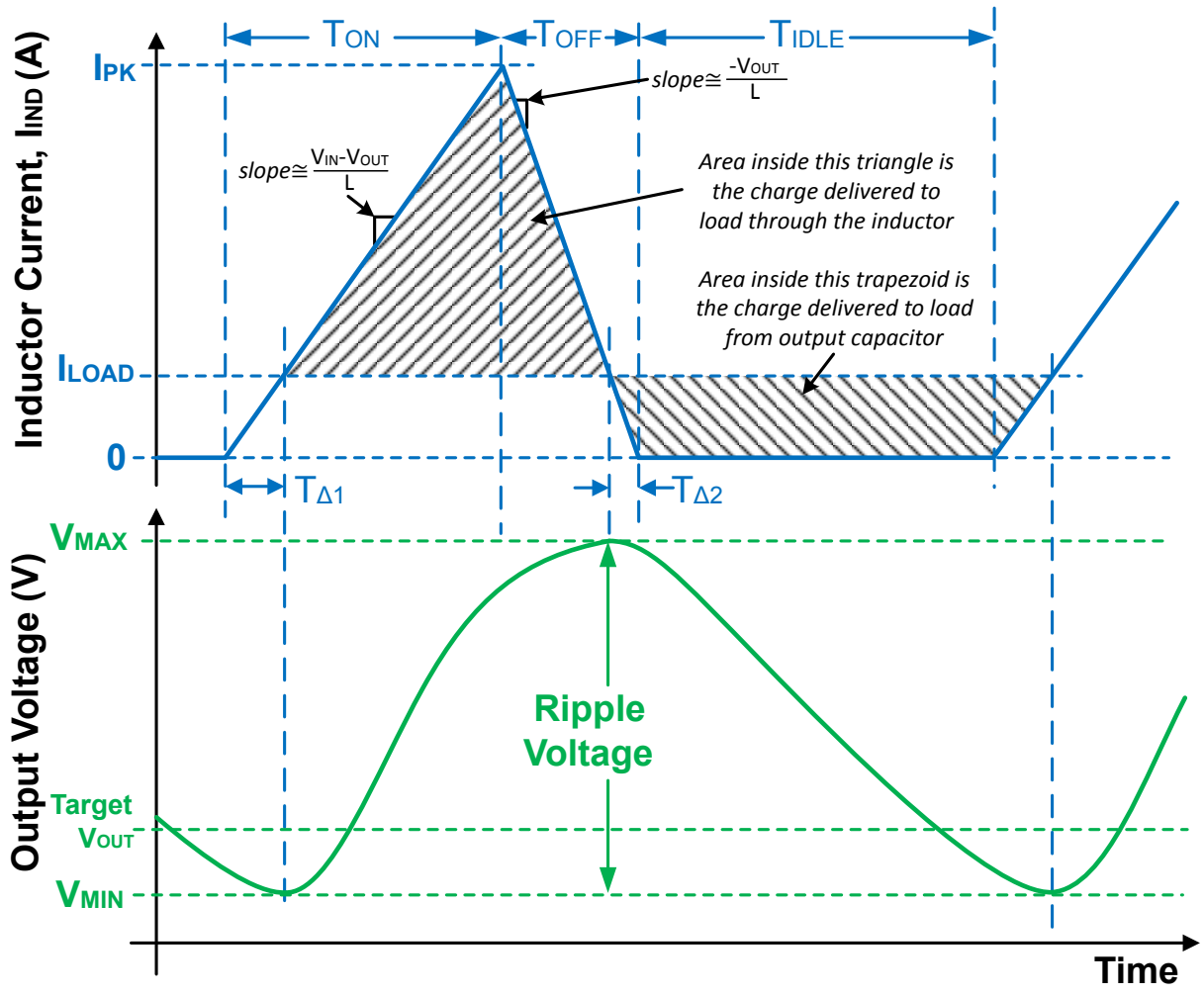


Figure 1.3. Buck Output Ripple Voltage

Ignoring the effects of capacitor ESR, the output ripple voltage (V_R) for a buck converter can be approximated by $V_R = \frac{Total\ \ Charge\ \ into\ \ C_{OUT}\ \ (in\ \ Coulombs)}{C_{OUT}\ \ (in\ \ Farads)}$, where the total charge into the output capacitor (C_{OUT}) is the area of the triangle shown in the figure above. The resulting ripple voltage equation is:

$$V_R = \frac{L \times (I_{PK} - I_{LOAD})^2}{2 \times C_{OUT}} \times \left(\frac{1}{V_{IN} - V_{OUT}} - \frac{1}{V_{OUT}}\right)$$

From the V_R equation, it can be seen that the following parameter changes increase ripple voltage (assuming all other parameters are held constant):

- Smaller output capacitance (C_{OUT})
- Smaller difference between input and output voltages (V_{IN} - V_{OUT})
- Larger inductor (L)
- Larger peak current (I_{PK})
- Smaller load current (I_{LOAD}).

1.1.3 T_{ON} Limiting

When T_{ON} limiting is enabled, the PFM controller terminates the inductor charging cycle before the programmed peak current setting (I_{PK}, shown in red in the diagram below) is reached, resulting in a reduced peak current (I'_{PK}, shown in blue below). This reduction in peak current has several effects:

1. As shown in 1.1.2 Buck Converter Output Ripple Voltage, the output ripple voltage is directly proportional to the square of the peak current (I_{PK}). Therefore, because T_{ON} reduces the maximum peak current, the output ripple voltage is also reduced.
2. Because the maximum output current in an ideal buck converter is $I_{PK}/2$, a reduced peak current results in a reduced maximum output load current.
3. For a fixed output load current, a reduced T_{ON} causes less current transferred to the load per switch event, which results in an increased switching frequency F_{SW}.

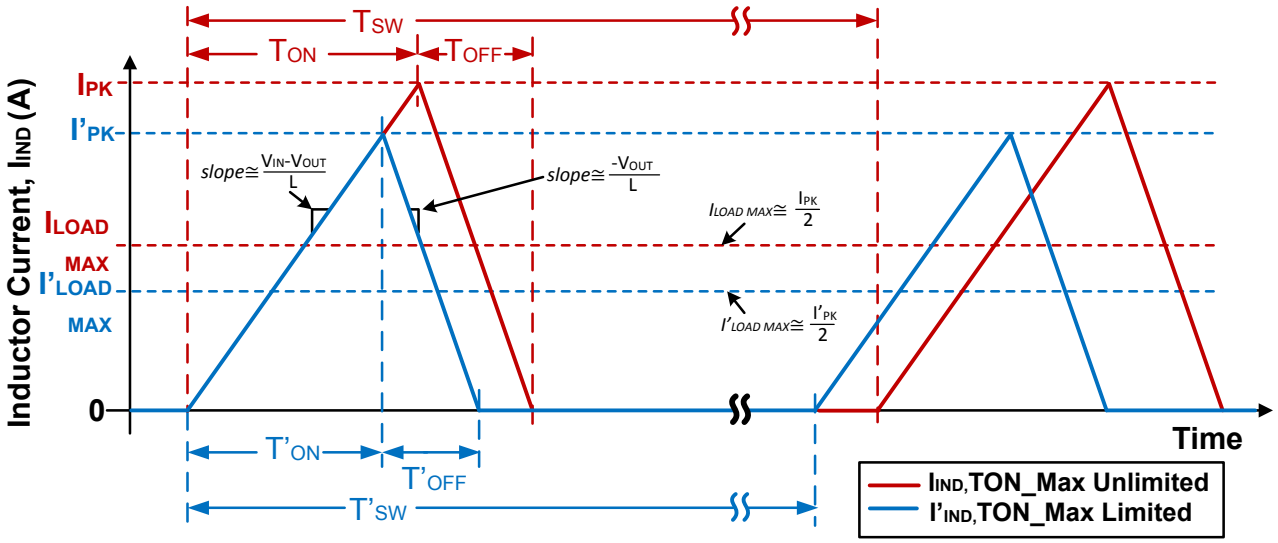


Figure 1.4. T_{ON} Limiting

1.2 Ideal Boost Converter Overview

1.2.1 Boost Converter Operation

In a simple, ideal boost converter (as shown in the figure below), the high-side (PMOS) and low-side (NMOS) switches are turned on an off complementarily to generate a higher output voltage from a lower input voltage.

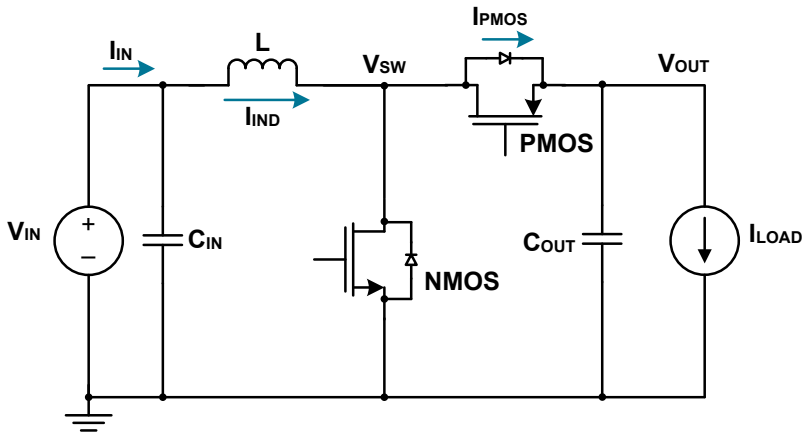


Figure 1.5. Simplified Boost Converter

The current through the inductor (I_{IND}) is typically limited by a fixed or programable peak current setting (I_{PK}). Note that the average input current (I_{IN}) is the same as the average inductor current (I_{IND}) for a boost converter. The maximum output current an ideal boost converter can deliver is .

$$I_{LOAD\_MAX} = \frac{I_{PK}}{2} \times \frac{V_{IN}}{V_{OUT}}$$

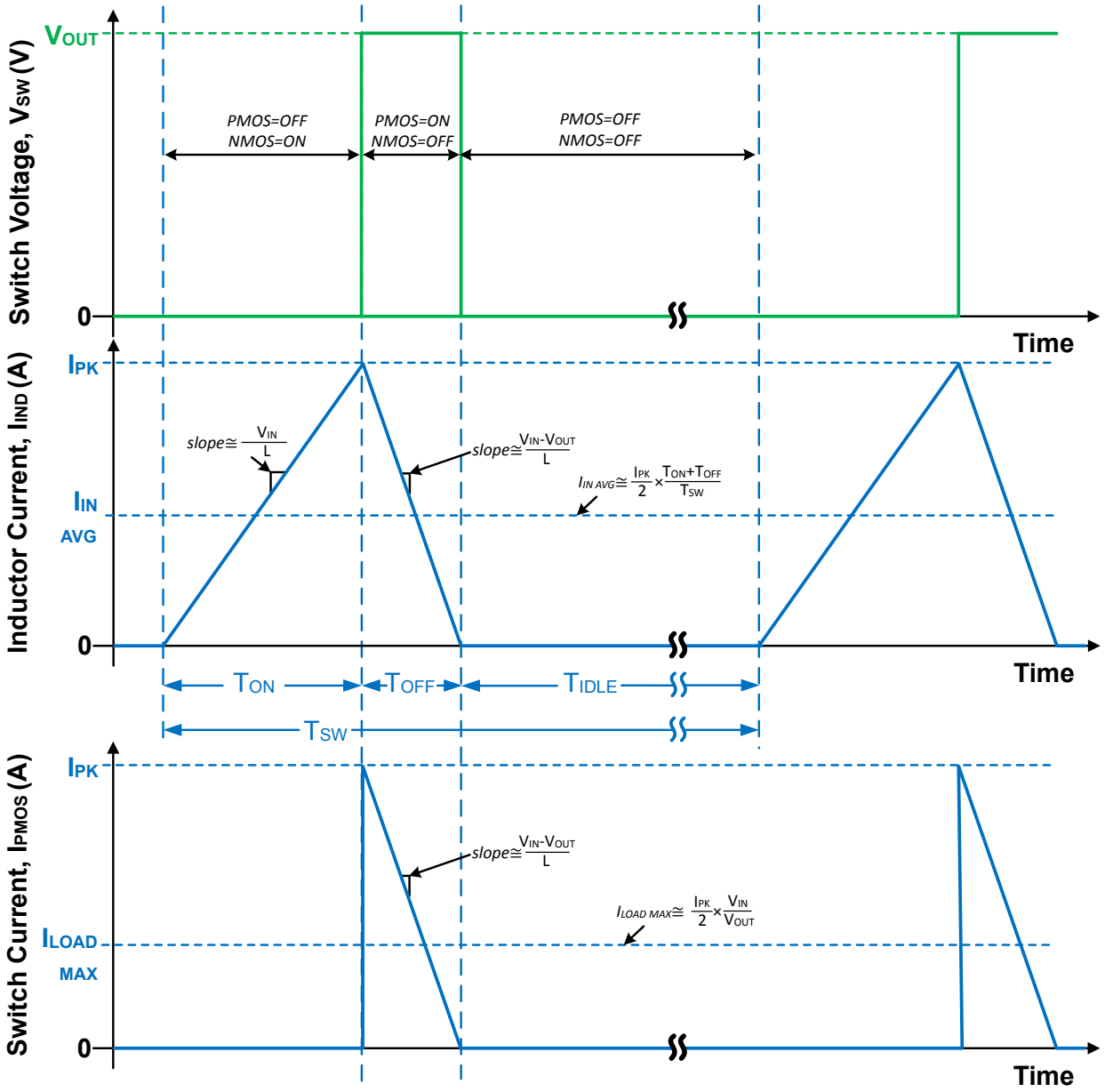


Figure 1.6. Boost PFM Switching Waveforms

The figure above shows the voltage at the switch node (V_{SW}) and the current through the inductor (I_{IND}) as the PMOS and NMOS switches are enabled and disabled. Note that this switching waveform is an example of Discontinuous Conduction Mode (DCM) operation, that is, the current through the inductor is allowed to reach zero.

The waveform for a single switching cycle can be divided into three distinct time periods:

1. On-time (T_{ON}): During this time, the PMOS switch is open and the NMOS switch is closed (forcing V_{SW} to ground), effectively forcing the voltage V_{IN} across the inductor. The current through the inductor (I_{IND}) during T_{ON} is determined by the following equation:

   $$I_{IND} = time \times \frac{V_{IN}}{L}$$

   After I_{IND} reaches the I_{PK} limit, the converter enters the off-time period.

2. Off-time (T_{OFF}): During this time, the PMOS switch is closed and the NMOS switch is open (forcing V_{SW} to be equal to V_{OUT}). The current through the inductor (I_{IND}) is determined by the following equation:

   $$I_{IND} = time \times \frac{V_{IN} - V_{OUT}}{L}$$

   After I_{IND} reaches zero, the converter enters the idle-time period.
3. Idle-time (T_{IDLE}): During this time, both the PMOS and NMOS switches are open. The current through the inductor (I_{IND}) is zero. When the output voltage (V_{OUT}) drops below a set threshold (determined by an internal comparator), the converter re-enters the on-time period, and the cycle repeats. The converter can supply the maximum output load current when T_{IDLE} is zero.

The switching period (T_{SW}) and the switching frequency (F_{SW}) can be determined by the sum of the time periods in the switching cycle, as shown below:

$$T_{SW} = T_{ON} + T_{OFF} + T_{IDLE} \quad and \quad F_{SW} = \frac{1}{T_{SW}}$$

The converter input current can be limited by forcing a minimum switching period, T_{SW}. For a forced minimum switching period, the maximum load current is determined by:

$$I_{LOAD\_LIMIT} = \frac{I_{PK}}{2} \times \frac{V_{IN}}{V_{OUT}} \times \frac{(T_{ON} + T_{OFF})}{T_{SW}}$$

Likewise, for a forced minimum switching period, the maximum battery current is determined by: $I_{BATT\_LIMIT} = \frac{I_{PK}}{2} \times \frac{(T_{ON} + T_{OFF})}{T_{SW}}$

1.2.2 Boost Converter Output Ripple Voltage

The figure below illustrates the contribution of the different boost mode switching periods to the output ripple voltage.

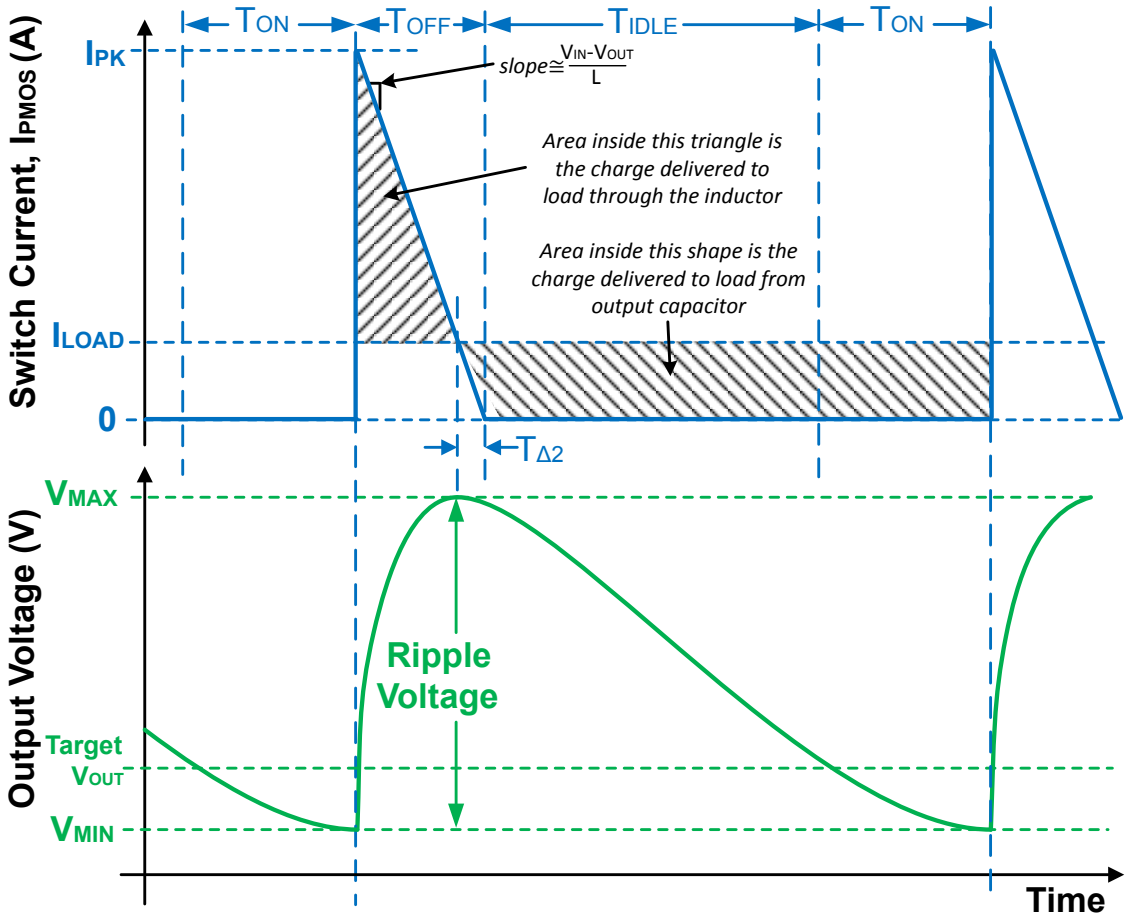


Figure 1.7. Boost Output Ripple Voltage

Ignoring the effects of capacitor ESR, the output ripple voltage (V_R) for a boost converter can be approximated by $V_R = \frac{Total\ \ Charge\ \ into\ \ C_{OUT}\ \ (in\ \ Coulombs)}{C_{OUT}\ \ (in\ \ Farads)}$ where the total charge into the output capacitor (C_{OUT}) is the area of the triangle shown in the figure above. The resulting ripple voltage equation is given by:

$$V_R = \frac{L \times (I_{PK} - I_{LOAD})^2}{2 \times C_{OUT} \times (V_{OUT} - V_{IN})}$$

From the V_R equation, it can be seen that the following parameter changes increase ripple voltage (assuming all other parameters are held constant):

- Smaller output capacitance (C_{OUT})
- Increased difference between output and input voltages (V_{OUT} - V_{IN})
- Larger inductor (L)
- Larger peak current (I_{PK})
- Smaller load current (I_{LOAD}).

1.3 Ideal Pulsed-Current Low-Dropout Regulator (LDO) Overview

1.3.1 Pulsed-Current LDO Overview

Unlike a traditional LDO, a pulsed-current LDO dynamically switches on and off.

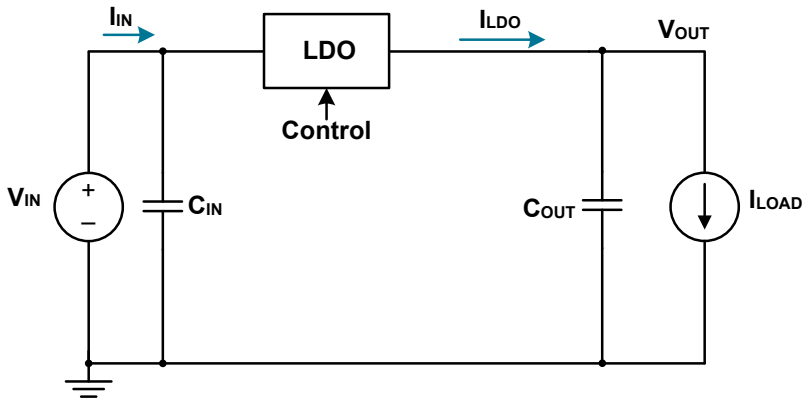


Figure 1.8. Simplified Pulsed-Current LDO

Figure 1.9 LDO Switching Waveforms on page 10 shows the current through the LDO (I_{LDO}) as the LDO is switched on and off.

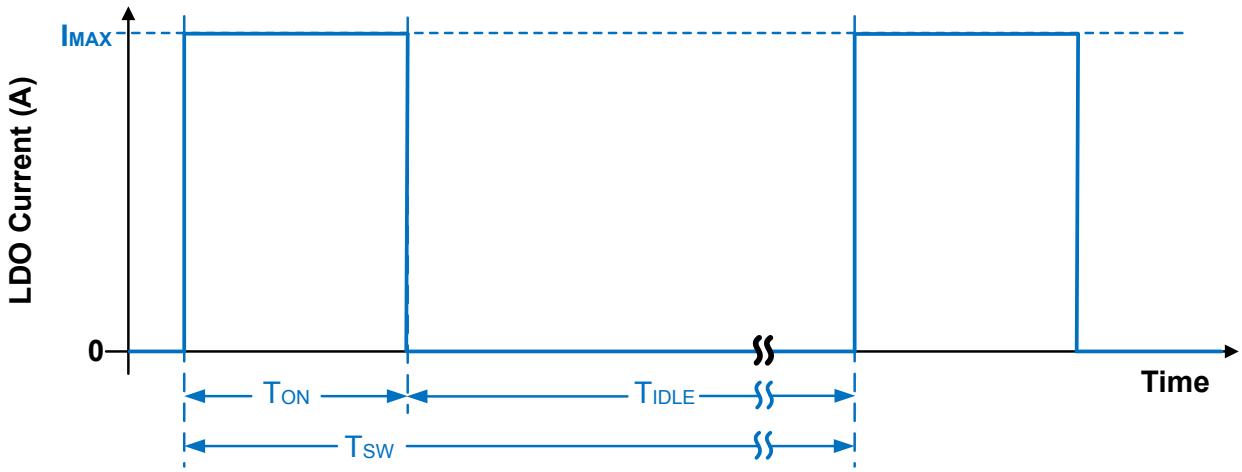


Figure 1.9. LDO Switching Waveforms

The waveform for a single switching cycle can be divided into two distinct time periods:

1. On-time (T_{ON}): During this time period, the LDO is on and delivering current I_{MAX}.
2. Idle-time (T_{IDLE}): During this time period, the LDO is off.

The switching period (T_{SW}) and the switching frequency (F_{SW}) can be determined by the sum of the time periods in the switching cycle, as shown below:

$$T_{SW} = T_{ON} + T_{IDLE} \quad and \quad F_{SW} = \frac{1}{T_{SW}}$$

In a pulsed-current LDO (as shown in Figure 1.8 Simplified Pulsed-Current LDO on page 10), the current through the LDO (I_{LDO}) is typically limited by a fixed or programmable maximum current setting (I_{MAX}). The LDO can supply the maximum output load current

(I_{MAX}) when T_{IDLE} is zero. Alternately, the input (and output) current can be limited by forcing a minimum switching period, T_{SW}. For a forced minimum switching period, the maximum output current is determined by:

$$I_{LOAD\ MAX} = I_{MAX} \times \frac{T_{ON}}{T_{SW}}$$

1.3.2 Pulsed-Current LDO Output Ripple Voltage

As shown in the figure below, the output ripple voltage for a pulsed-current LDO is a function of the maximum current, the load current, and the output capacitor.

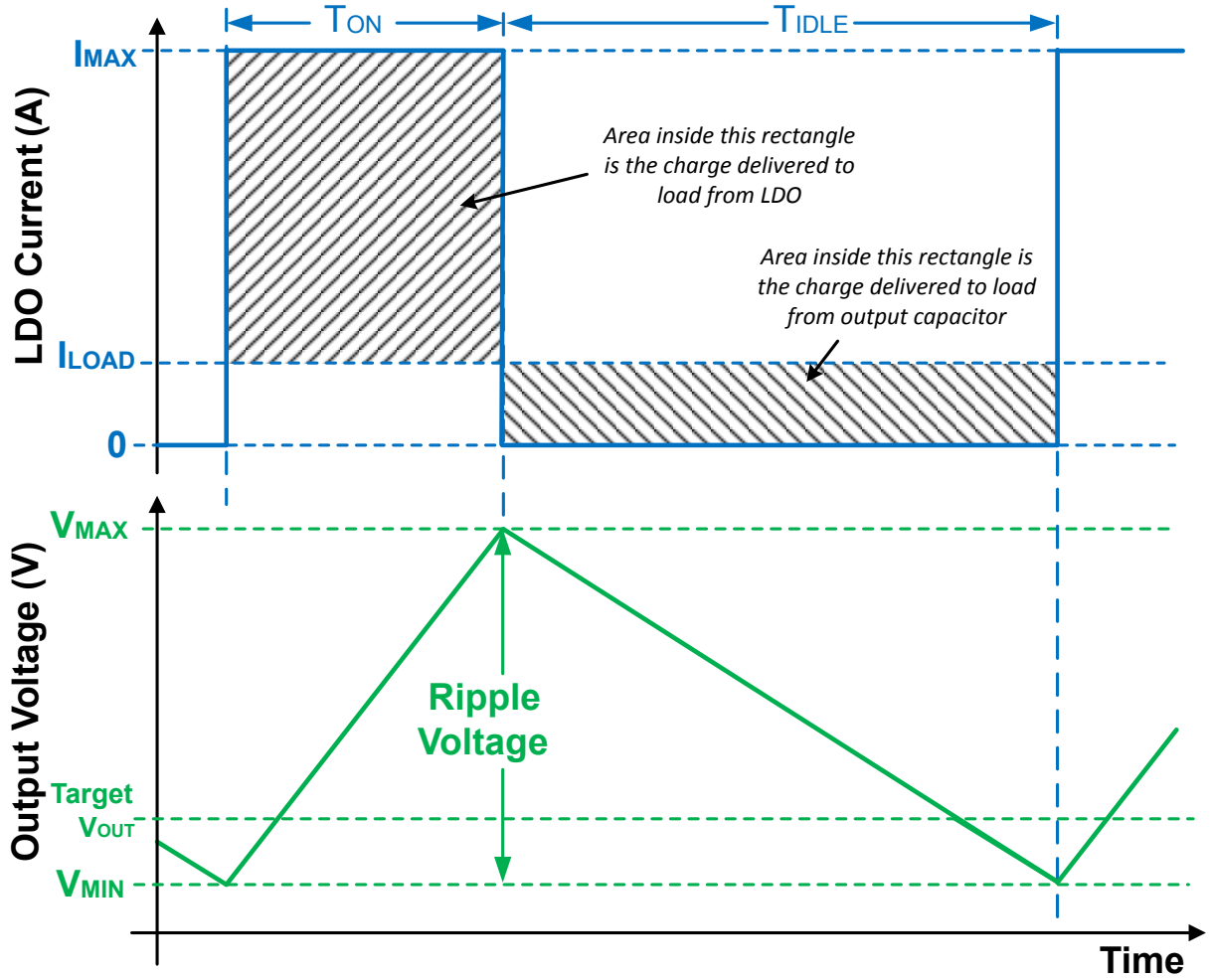


Figure 1.10. LDO Output Ripple Voltage

Ignoring the effects of capacitor ESR, the output ripple voltage (V_R) for a pulsed current LDO can be approximated by

$$V_R = T_{ON} \times \frac{(I_{MAX} - I_{LOAD})}{C_{OUT}}$$

From the V_R equation, it can be seen that the following parameter changes increase the ripple voltage (assuming all other parameters are held constant):

- Smaller output capacitance (C_{OUT})
- Larger maximum current (I_{MAX})
- Smaller load current (I_{LOAD}). Note that ripple voltage is at its maximum when I_{LOAD} is at its smallest.

2. EFP01 Components Selection Guide

2.1 DC-DC Input/Output Capacitor

The input/output capacitors should have a temperature range reflecting the environment in which the application will be used. For example, a suitable choice might be X5R ceramic capacitors with a change in capacitance of ±15% over the temperature range -55 °C to +85 °C (standard temperature range devices) or -55 °C to +125 °C (extended temperature range devices).

Careful attention should be paid to the characteristics of the input/output capacitors over temperature and bias voltage. Some capacitors (particularly those in smaller packages) can experience a dramatic reduction in capacitance value as the temperature or bias voltage increases. A change pushing the DC-DC output capacitance outside the data sheet specified limits may result in output instability.

The Murata GRM31CR71A106KA01 10 µF capacitor was used for all of the DC-DC validation and characterization testing.

Table 2.1. Recommended DC-DC Input/Output Capacitor (or equivalent)

Manufacturer	Part Number	Value (µF)	Voltage Rating (V)	Dielectric	Operating Temperature (°C)	Package
Murata	GRM31CR71A106KA01	10 ± 10%	10	X7R	-55 to +125	1206

2.2 DC-DC Inductor

The Samsung CIG22H2R2MAE inductor was used for all of the DC-DC validation and characterization testing.

Table 2.2. Example Buck Mode DC-DC Inductors

Manufacturer	Part Number	Value (µH)	$I_{saturation}$ (mA)	DCR (Ω)	Operating Temperature (°C)	Height (mm)	Package
Samsung	CIG22H2R2MAE	2.2 ± 20%	1100	0.138 ± 20%	-40 to +125		1008/2520
Murata	LQM18PZ2R2MFH	2.2 ± 20%	300	0.47 (max)	-55 to +125		0603/1608
TDK	MLZ2012N2R2LT000	2.2 ± 20%	170	0.156 (max)	-55 to +125	0.85 ± 0.20	0805/2012
Murata	LQH3NPN2R2MJRL	2.2 ± 20%	1800	0.0816 (max)	-40 to +105	1.2	1212/3030

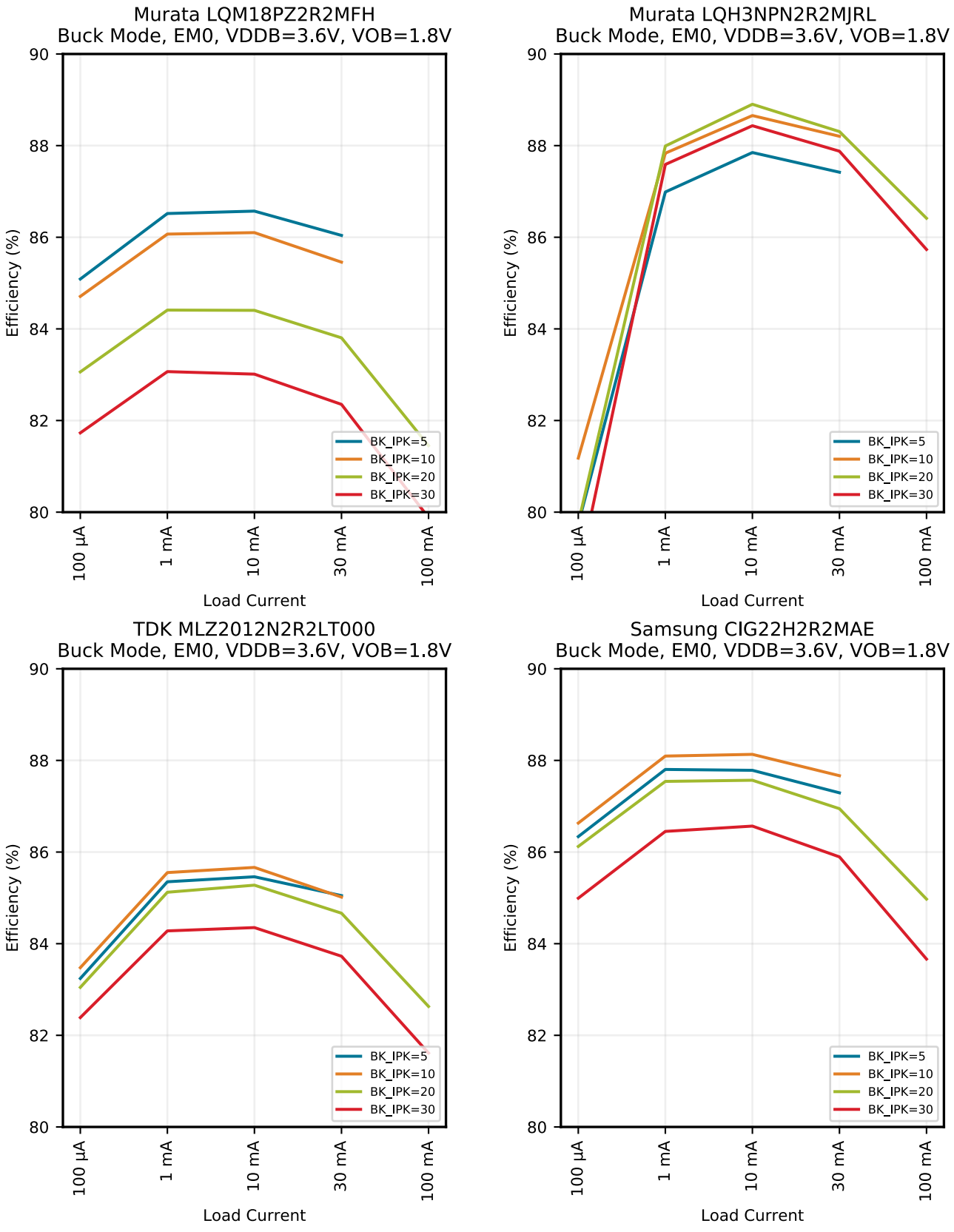
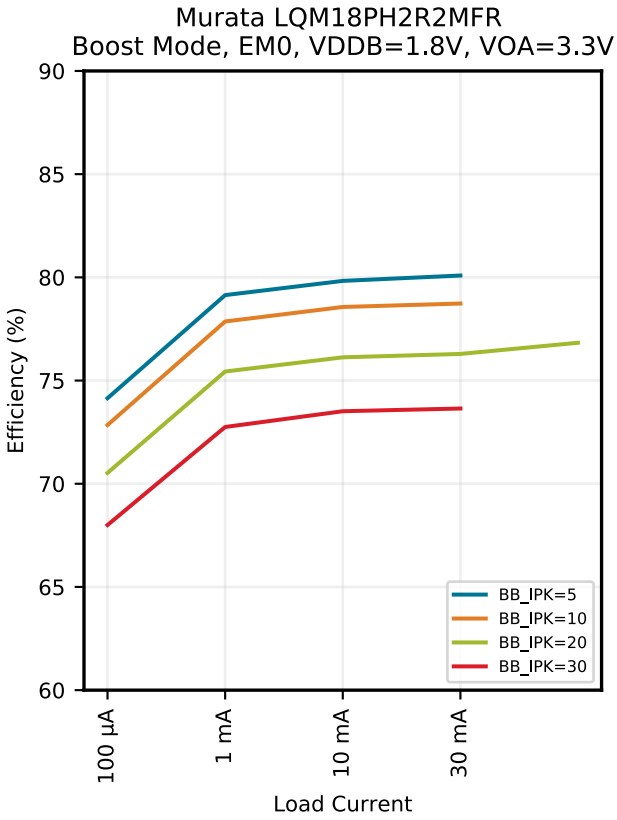
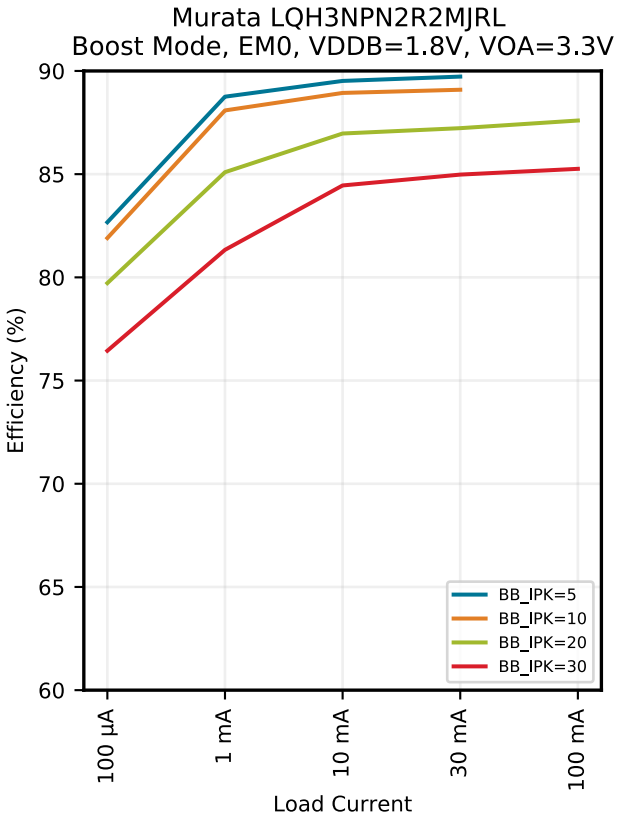


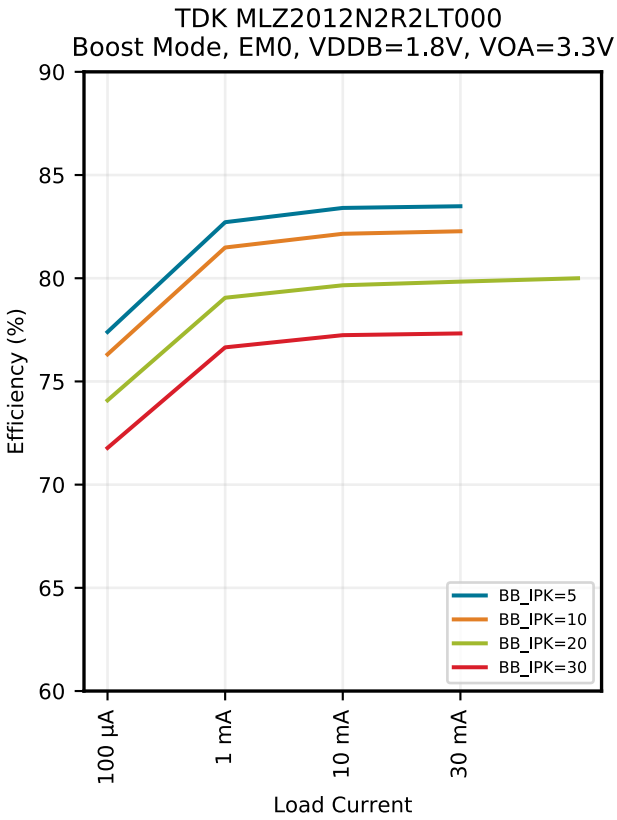
Figure 2.1. Inductor Efficiency Plots, Buck Mode

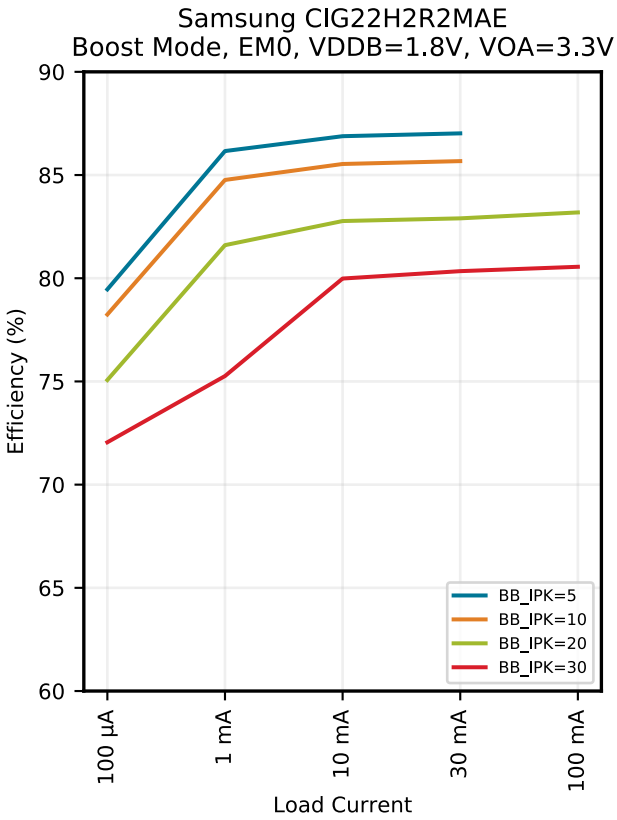
Table 2.3. Example Boost Mode DC-DC Inductors

Manufacturer	Part Number	Value (µH)	$I_{saturation}$ (mA)	DCR (Ω)	Operating Temperature (°C)	Height (mm)	Package
Samsung	CIG22H2R2MAE	2.2 ± 20%	1100	0.138 ± 20%	-40 to +125		1008/2520
Murata	LQM18PH2R2MFR	2.2 ± 20%	150	0.38 (max)	-55 to +150	0.95	0603/1608
TDK	MLZ2012N2R2LT000	2.2 ± 20%	170	0.156 (max)	-55 to +125	0.85 ± 0.20	0805/2012
Murata	LQH3NPN2R2MJRL	2.2 ± 20%	1800	0.0816 (max)	-40 to +105	1.2	1212/3030


Murata LQM18PH2R2MFR
Boost Mode, EM0, VDDB=1.8V, VOA=3.3V
Efficiency (%)
90
85
80
75
70
65
60
100 µA
1 mA
10 mA
30 mA
Load Current
BB_IPK=5
BB_IPK=10
BB_IPK=20
BB_IPK=30


Murata LQH3NPN2R2MJRL
Boost Mode, EM0, VDDB=1.8V, VOA=3.3V
Efficiency (%)
90
85
80
75
70
65
60
100 µA
1 mA
10 mA
30 mA
100 mA
Load Current
BB_IPK=5
BB_IPK=10
BB_IPK=20
BB_IPK=30


TDK MLZ2012N2R2LT000
Boost Mode, EM0, VDDB=1.8V, VOA=3.3V
Efficiency (%)
90
85
80
75
70
65
60
100 µA
1 mA
10 mA
30 mA
Load Current
BB_IPK=5
BB_IPK=10
BB_IPK=20
BB_IPK=30


Samsung CIG22H2R2MAE
Boost Mode, EM0, VDDB=1.8V, VOA=3.3V
Efficiency (%)
90
85
80
75
70
65
60
100 µA
1 mA
10 mA
30 mA
100 mA
Load Current
BB_IPK=5
BB_IPK=10
BB_IPK=20
BB_IPK=30

3. EFP01 Measurements

3.1 Taking Current & Efficiency Measurements

The pulsed-current operation of the PFM converter means that the input power supply current can have a very large dynamic range with currents in the microamp range spiking upwards of 100 mA and then returning to the microamp level. As such, taking average current measurements with a digital multi-meter (DMM) can be challenging. Below are some tips for ensuring accurate measurements.

1. For low current (nA or µA-level) measurements, the PCB should be thoroughly cleaned using alcohol and/or an ultrasonic bath. Flux and other residues left on a PCB during assembly can be responsible for several microamps of leakage current.
2. A large, very low-leakage capacitor (for example, a ceramic or several large ceramic capacitors wired in parallel) may need to be added at the EFP01 input. Using a DMM in series with the power supply creates a low-pass filter (using the sense resistor of the DMM and the large capacitor).
3. If supported, the DMM should be configured to use the largest aperture window (number of power line cycles, or NPLC) and highest average count.
4. If supported, any line synchronization or autozeroing capabilities of the DMM should be enabled.

3.2 EFP01 Impact on EFM32/EFR32 Current Draw

Use of EFP01 brings sizable improvements in current draw to applications using EFM32 and EFR32 devices due simply to the greater efficiency with which its regulators operate relative to those integrated on the microcontrollers themselves. In the case where EFP01 provides 1.8 V for all supplies except IOVDD on a Series 2 EFR32 device, current draw is reduced in all operating modes by nearly 30%. When the EFM32/EFR32 on-chip LDO is disabled and EFP01 also provides the 1.1 V core digital supply, current draw in run (EM0) and sleep (EM2) modes is reduced again by nearly 30%. Assuming 90% operation in sleep mode, this effectively doubles runtime in battery-powered applications.

Table 3.1 EFP0104 + EFR32BG21 Current Draw Comparison on page 17 illustrates the reduction in current draw seen under common operating conditions when the EFP01 provides only 1.8 V for the device power supply inputs and when it also provides the 1.1 V core digital supply.

Table 3.1. EFP0104 + EFR32BG21 Current Draw Comparison

Measurement	EFR32BG21 Alone[1]	EFR32BG21 with EFP01 Providing 1.8V on VOA[1]	EFR32BG21 with EFP01 Providing 1.8V on VOA and 1.1V on VOB[2]
EM0 (µA/MHz)	46.4	32.7	23.4
EM2 (µA)	5.04	3.75	2.6
TX, 0 dBm PA @ +0 dBm (mA)	9.3	6.645	6.104
TX, 10 dBm PA @ +10 dBm (mA)	33.1	23.2	22.7
RX Listen @ 2 Mbps BLE (mA)	9.854	7.034	6.036

Note:

1. IOVDD = VBATT = 3 V. AVDD = DVDD = PAVDD = RFVDD = 1.8 V. The EFM32/EFR32 on-chip LDO derives the core digital supply from DVDD.
2. IOVDD = VBATT = 3 V. AVDD = DVDD = PAVDD = RFVDD = 1.8 V. DECOUPLE = 1.1 V. The EFP0104 powers DECOUPLE on the EFM32/EFR32 device. Secure Element firmware must be updated to version 1.1.3 or later to permit disabling of the core digital supply LDO.

4. EFP01 Reference Schematics

4.1 RF Noise Filtering

At higher current levels and peak current settings, EFP01 can have ripple voltages as high as 70 mV. Because the switching frequency of a PFM converter is linearly proportional to load current, frequency planning is not possible.

A pi filter using a small, cheap, low DCR ferrite with high real impedance at 1 MHz and up (e.g., Murata BLM18AG601SN1) can be used as shown below to filter the switching ripple.

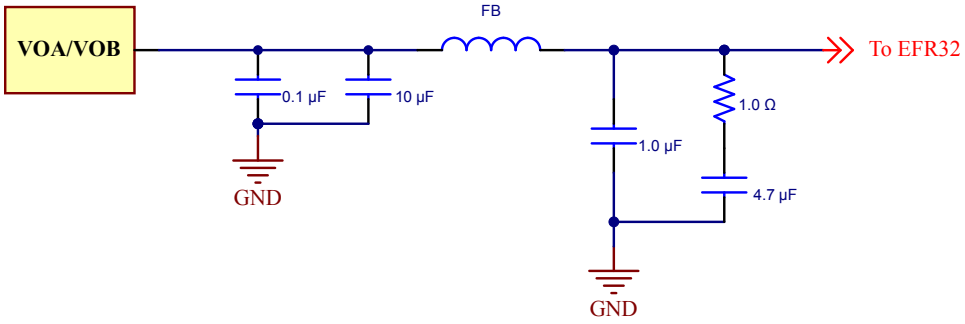


Figure 4.1. RF Filter

4.2 Reference Schematics

The following diagrams show the circuit configurations and passive component values for different EFP01.

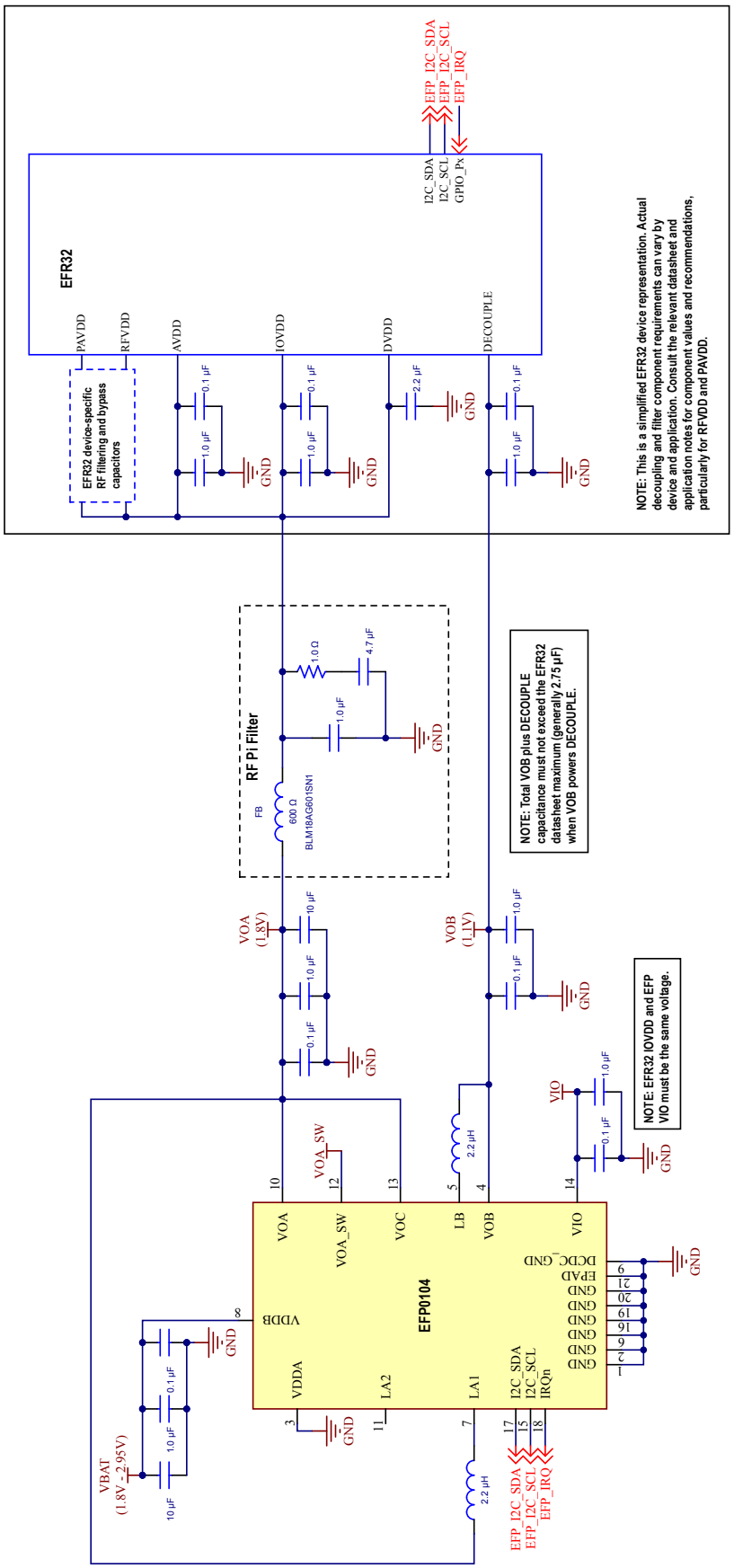


Figure 4.2. EFP0104 Reference Schematics

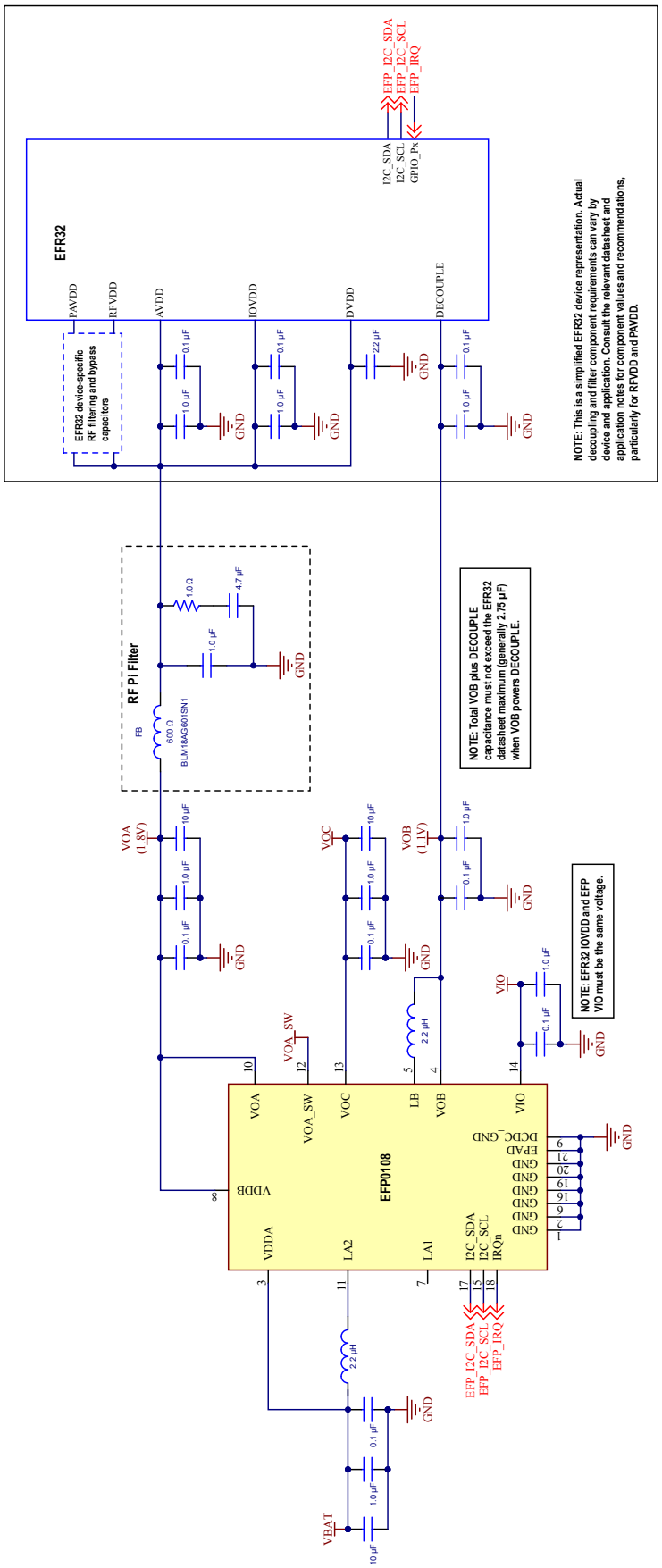


Figure 4.3. EFP0108 Reference Schematics

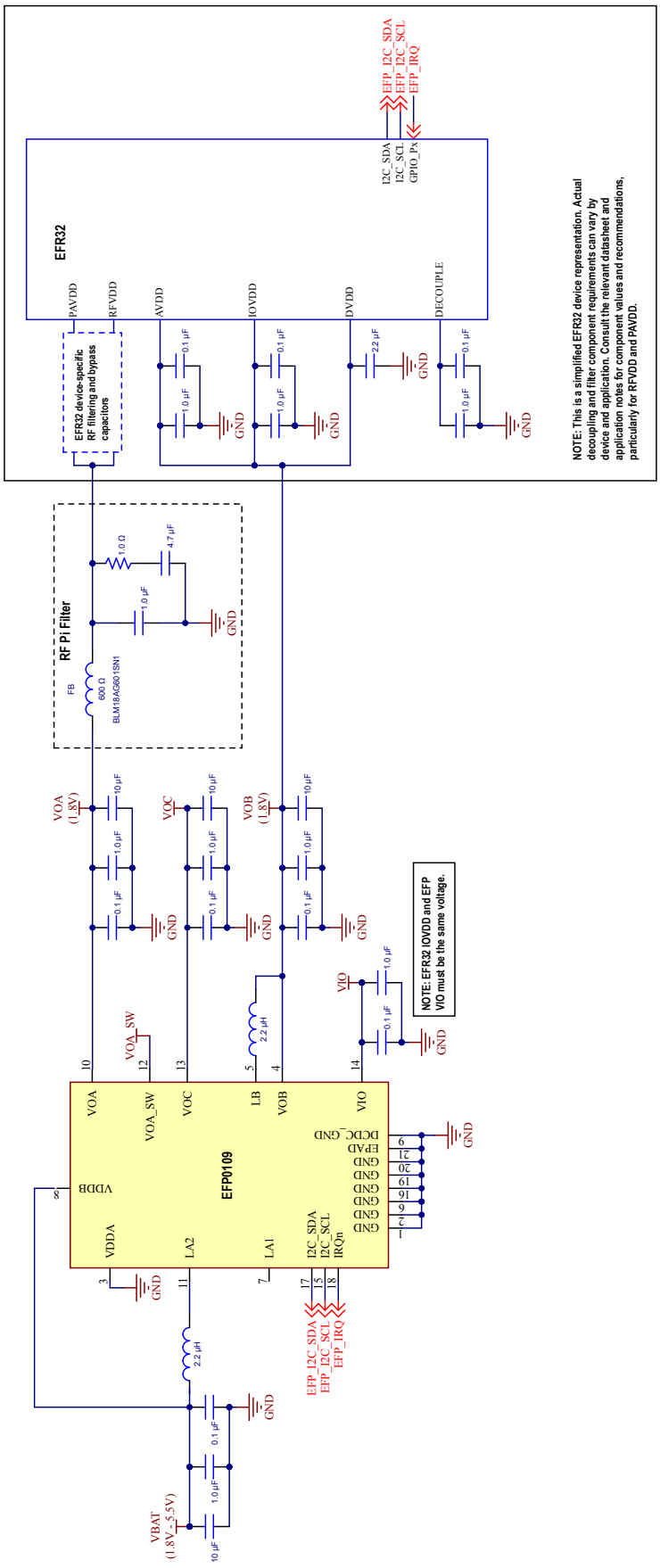


Figure 4.4. EFP0109 Reference Schematics

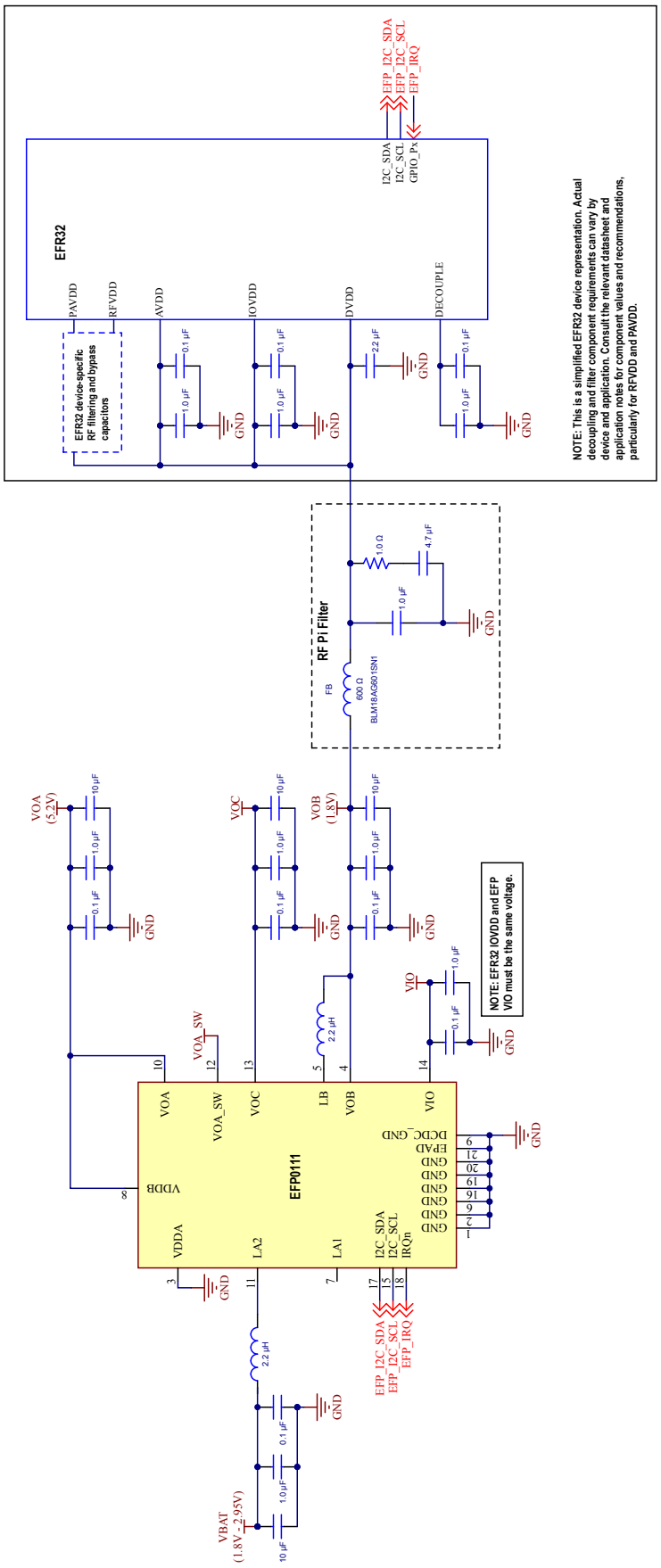


Figure 4.5. EFP0111 Reference Schematics

5. EFP01 Layout Considerations

Because the DC-DC converters on EFP01 can switch high currents at a high frequency, special layout considerations are required to optimize operation and reduce noise:

- The following connections to the EFP01 are to be made with minimum trace length and trace resistance:
- - LA1, LA2, and/or LB pins to corresponding inductors
  - VOA, VOB, and/or VOC outputs to corresponding 10 µF & 0.1 µF filter capacitors
  - Battery or main supply voltage to VDDB and/or VDDA pin(s)
  - DCDC_GROUND to PCB ground
- The EFP01 should be oriented on the PCB in such a way so that the DCDC inductors are located as far as possible from any noise-sensitive circuitry (e.g., a radio antenna). Ideally, the inductor(s) should be placed on the opposite side of the PCB such that there is a solid ground plane shielding the noisy inductor(s) from the sensitive RF circuitry.
- For more detailed radio-specific layout guidelines for EFR32, see application note, AN928: EFR32 Layout Design Guide.
- Use an array of vias under the center GND pad to achieve datasheet specified thermal and noise performance as shown in the figure below. These should be 16 mil diameter and 8 mil drill.

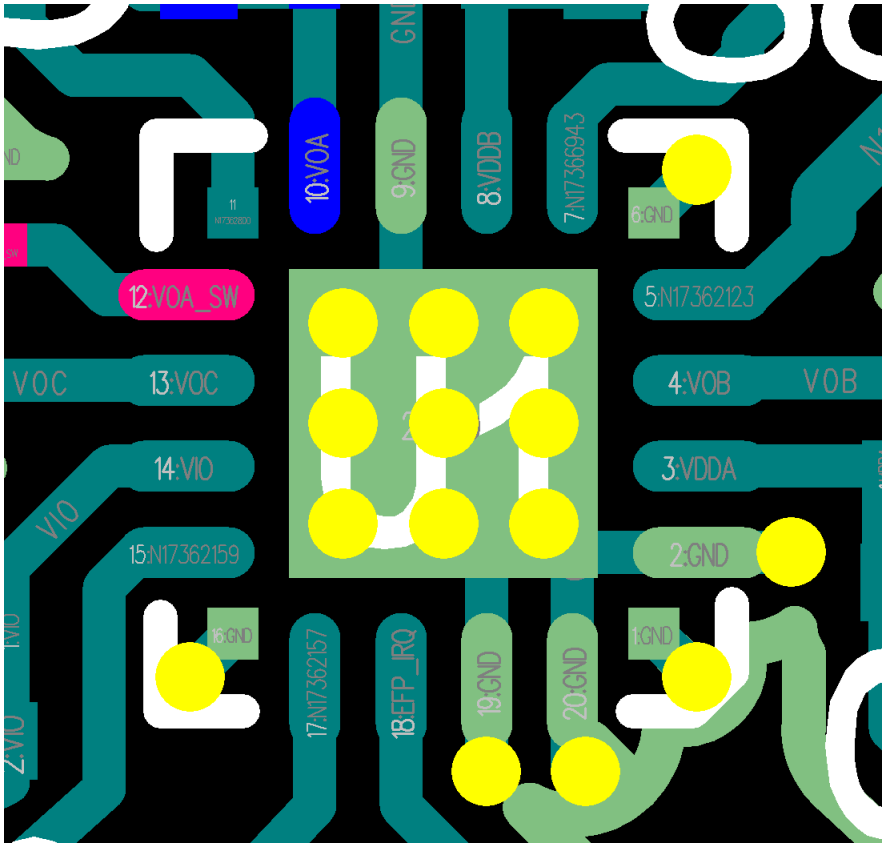


Figure 5.1. Center Ground Pad Via Arrangement

6. EFP01 SDK Support

To simplify use of the EFP01, Gecko SDK includes a driver to properly configure it for efficient operation. Use of the driver is strongly encouraged because it includes workarounds for any errata or issues affecting device performance. For information about the driver, see the [**Gecko HAL and Driver API Reference Guide**] tile in Simplicity Studio.

6.1 Driver Files and Simplicity Studio Projects

EFP01 support is found in the `hardware/driver/efp` directory with the full path being `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.7\hardware\driver\efp` for Gecko SDK version 2.7 in the default Simplicity Studio directory. As shown in Figure 6.1 EFP01 Device Driver Files on page 24, the driver consists of a main C file, two header files, and a template configuration file.

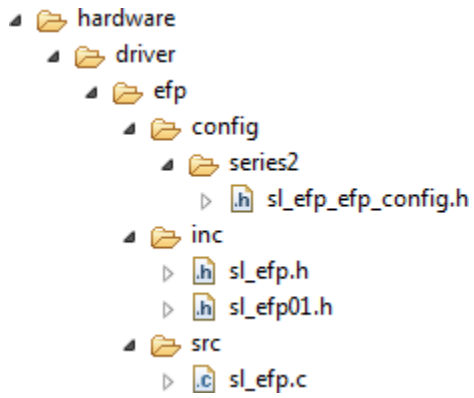


Figure 6.1. EFP01 Device Driver Files

Simplicity Studio's wireless template projects, such as *soc-empty* in the Bluetooth stack, automatically include EFP01 support when the target is a Wireless Starter Kit (WSTK) mainboard and an EFR32 radio board that includes EFP01, e.g. the EFR32MG21 2.4 GHz 10 dBm with EFP Radio Board (BRD4179B). For projects created by specifying a part number in the the Launcher's My Products panel and then clicking the New Project button, the EFP01 driver files must be manually copied into the project.

EFP01 setup is handled with other board level initialization in the project's `init-board.c` file with the addition of three segments of code. The first segment consists of the necessary header files and global variables:

```
#include "sl_efp.h"
#include "sl_efp_efp_config.h"

static sl_efp_handle_data_t efp_handle_data;
static sl_efp_handle_t      efp = &efp_handle_data;
```

Note that the `sl_efp_efp_config.h` provided by the SDK is a template configuration file that can and should be customized for the specific system at hand. Additionally, Simplicity Studio's EFP configuration tool, discussed separately in application note *AN1245: EFP Configuration Tool*, generates a suitable `sl_efp_efp_config.h` file that can be manually copied into a project.

The second segment of code consists of local variables that need to be added to the `initBoard()` function:

```
  sl_status_t status;
  uint8_t i2cCtrl, devRevId;
  uint8_t tmp;
```

The actual EFP01 initialization is appended to the `initBoard()` function:

```
// Initialize the EFP
sl_efp_init_data_t init = SL_EFP_INSTANCE_INIT;
init.enable_irq_pin = false;
status |= sl_efp_init(efp, &init);

// Set VOB EM23 targets
sl_efp_set_vob_em23_voltage(efp, 1100);
sl_efp_write_register_field(efp,
                            EFP01_BB_IPK,
                            1,
                            _EFP01_BB_IPK_BB_IPK_EM2_MASK,
                            _EFP01_BB_IPK_BB_IPK_EM2_SHIFT);

// Set VOB target to higher level to guarantee it will overdrive the EFR32 LDO output
sl_efp_set_vob_em01_voltage(efp, 1130);

// Set min peak current
sl_efp_write_register_field(efp, EFP01_BK_IPK, 0,
                            _EFP01_BK_IPK_BK_IPK_MASK,
                            _EFP01_BK_IPK_BK_IPK_SHIFT);

// Set min Ton time
sl_efp_write_register_field(efp, EFP01_BK_CTRL1,
                            1,
                            _EFP01_BK_CTRL1_BK_TON_MAX_MASK,
                            _EFP01_BK_CTRL1_BK_TON_MAX_SHIFT);

// Set max current limit
sl_efp_write_register_field(efp, EFP01_BK_CTRL2,
                            15,
                            _EFP01_BK_CTRL2_BK_IRI_CON_MASK,
                            _EFP01_BK_CTRL2_BK_IRI_CON_SHIFT);

// Enable VOB
sl_efp_set_vob_mode(efp, efp_vob_mode_buck);

// Make sure VOB is ready before turning off internal LDO regulator
do {
  status = sl_efp_read_register(efp, EFP01_STATUS_LIVE, &tmp);
} while (((tmp & _EFP01_STATUS_LIVE_VOB_INREG_LIVE_MASK) == 0)
         || (status != SL_STATUS_OK));

// Set desired peak current
sl_efp_write_register_field(efp, EFP01_BK_IPK, 10,
                            _EFP01_BK_IPK_BK_IPK_MASK,
                            _EFP01_BK_IPK_BK_IPK_SHIFT);

// Set desired TON MAX
sl_efp_write_register_field(efp, EFP01_BK_CTRL1,
                            7,
                            _EFP01_BK_CTRL1_BK_TON_MAX_MASK,
                            _EFP01_BK_CTRL1_BK_TON_MAX_SHIFT);

// Disable current limit
sl_efp_write_register_field(efp, EFP01_BK_CTRL2,
                            0,
                            _EFP01_BK_CTRL2_BK_IRI_CON_MASK,
                            _EFP01_BK_CTRL2_BK_IRI_CON_SHIFT);

// Turn off EFR32xG21 internal LDO regulator
sl_efp_emu_ldo_enable(efp, false);

// Set desired VOB voltage
sl_efp_set_vob_em01_voltage(efp, 1100);
```

Setup is specific to both the EFP01 derivative and the EFR32 device being used. For example, the code above runs on the EFR32MG21 2.4 GHz 10 dBm with EFP Radio Board (BRD4179B). Initially, the EFP0104 derivative on this particular radio board delivers 1.13V on its VOB output to overdrive DECOUPLE with 1.13V until the internal regulator on EFR32MG21 can be disabled, after which the VOB output level is reduced to the usual 1.1V core digital supply voltage.

6.2 Initialization Breakdown

1. Declare a structure of type `sl_efp_init_data_t`.

```
typedef struct {
  unsigned int        config_size;                ///< Number of register writes inside the configuration
data. Set to 0 on preprogrammed parts.
   uint8_t                *config_data;                    ///< Configuration data, pairs of (addr,data),
(addr,data),... Set to NULL on preprogrammed parts.
  bool                is_host_efp;               ///< True if this EFP powers host SoC.
   sl_efp_em_transition_mode_t  em_transition_mode;  ///< Method for controlling EFP Energy Mode (EM)
transitions.
  bool                enable_irq_pin;            ///< Initialize a GPIO pin as EFP IRQ input.
  GPIO_Port_TypeDef   irq_port;                  ///< GPIO port to use for EFP IRQ GPIO pin.
  unsigned int        irq_pin;                   ///< GPIO pin number to use for EFP IRQ GPIO pin.
  I2C_TypeDef         *i2c_peripheral;           ///< I2C peripheral instance pointer.
  GPIO_Port_TypeDef   i2c_scl_port;              ///< GPIO port to use for I2C SCL signal.
  unsigned int        i2c_scl_pin;               ///< GPIO pin number to use for I2C SCL signal.
  GPIO_Port_TypeDef   i2c_sda_port;              ///< GPIO port to use for I2C SDA signal.
  unsigned int        i2c_sda_pin;               ///< GPIO pin number to use for I2C SDA signal.
  #if defined(_SILICON_LABS_32B_SERIES_0)
    unsigned int      i2c_port_location;         ///< I2C location number to use for I2C signals.
  #elif defined(_SILICON_LABS_32B_SERIES_1)
    unsigned int      i2c_scl_port_location;     ///< I2C location number to use for I2C SCL signal.
    unsigned int      i2c_sda_port_location;     ///< I2C location number to use for I2C SDA signal.
  #endif
} sl_efp_init_data_t;
```

   Parameter details:

   - `config_size`: The output of the EFP01 Configuration Tool is a header file with address and data pairs to be written. This parameter represents the size of the address/data array. If unused, set to 0.
   - `config_data`: Contains the address and data pair data array output from the EFP01 Configuration Tool generated header file. If unused, set to NULL.
   - `is_host_efp`: Set to True when the EFP01 is powering the EFR32/EFM32 host processor (default case).
   - `em_transition_mode`: Configures the energy mode transition method
     - `efp_em_transition_mode_i2c` — (Default) Energy mode transitions are made using I2C communications.
     - `efp_em_transition_mode_gpio_bitbang` — Energy mode transitions are made using a bit-banged direct mode for host processors without hardware support for EFP01 direct mode transitions. EM4 cannot be used in this mode. See additional usage warnings in the Direct Mode Control section in the EFP01 datasheet.
     - `efp_em_transition_mode_emu` — Energy mode transitions are made using direct mode control only on host processors with hardware support for EFP01 Direct Mode Control. EM4 is supported in this configuration. Direct mode hardware support is available on EFR32xG22 and later devices.
   - `enable_irq_pin`: Use the EFP01's IRQn output.
   - `irq_port`: If `enable_irq_pin` is true, this specifies the host processor GPIO port to which the EFP01's IRQn output is connected.
   - `irq_pin`: If `enable_irq_pin` is true, this specifies the host processor GPIO pin to which the EFP01's IRQn output is connected.
   - `i2c_peripheral`: Pointer to host processor I2C peripheral that is connected to the EFP01's I2C pins.
   - `i2c_scl_port`: Specifies the host processor GPIO port to which the EFP01's I2C_SCL pin is connected
   - `i2c_scl_pin`: Specifies the host processor GPIO pin to which the EFP01's I2C_SCL pin is connected.
   - `i2c_sda_port`: Specifies the host processor GPIO port to which the EFP01's I2C_SDA pin is connected
   - `i2c_sda_pin`: Specifies the host processor GPIO pin to which the EFP01's I2C_SDA pin is connected.
   - `i2c_port_location`: For Series 1 EFR32/EFM32 devices, selects the I2C port location.
   - `i2c_scl_port_location`: For Series 2 EFR32/EFM32 devices, selects the I2C SCL port location.
   - `i2c_sda_port_location`: For Series 2 EFR32/EFM32 devices, selects the I2C SDA port location.
2. Create a driver handle for the EFP01 as shown below:

```
static sl_efp_handle_data_t efp_handle_data;
static sl_efp_handle_t      efp = &efp_handle_data;
```

3. Call `sl_efp_init(efp, &init);` passing both the EFP handle and the `sl_efp_init_data_t` structure, as arguments.

While the sequence above is sufficient to configure the EFP01 driver, additional board/processor specific initialization may be required. Such is the case for the EFR32MG21, as briefly discussed near the end of 6.1 Driver Files and Simplicity Studio Projects, where further driver calls are be necessary to adjust regulator output prior to routine operation.

7. Revision History

Revision 0.4

June, 2020

- 3.2 EFP01 Impact on EFM32/EFR32 Current Draw section added.

Revision 0.3

May, 2020

- Updated key points on front page.

Revision 0.2

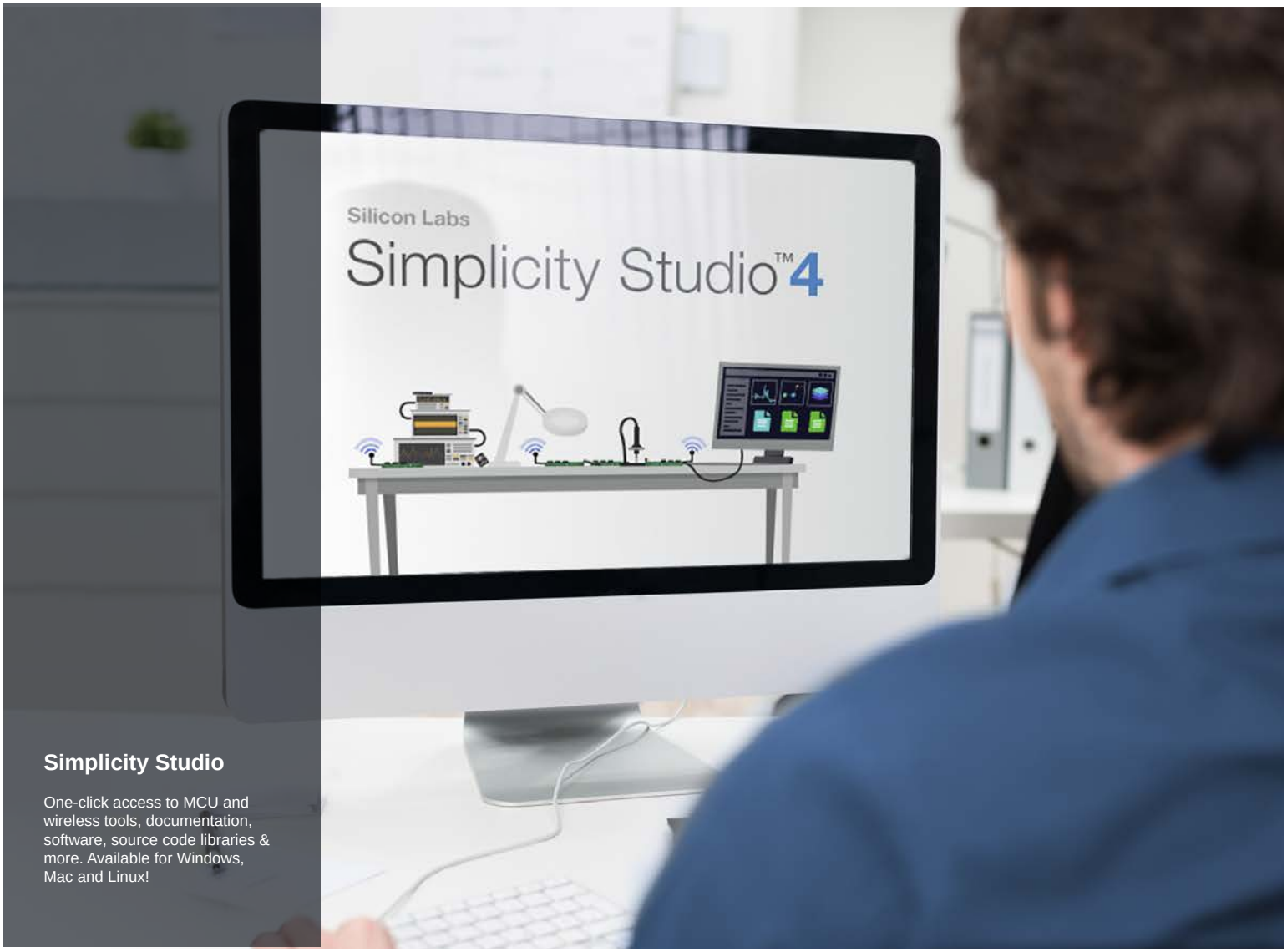
April, 2020

- Language updates made throughout.
- Added recommended boost mode inductors and efficiency graphs.
- 4.1 RF Noise Filtering updated.
- 4. EFP01 Reference Schematics improved.
- 6.1 Driver Files and Simplicity Studio Projects section added.

Revision 0.1

December, 2019

- 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
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com