Wireless IoT Entwicklung mit Nordic Semiconductor

Kapitel 3: Nordic PMIC & Fuel Gauge Integration

February 2026

nRF54L15DK / nRF Connect SDK 3.2.1

NORDIC® SEMICONDUCTOR

Adding a Nordic PMIC Fuel Gauge

Hands-on Guidance /w nPM2100-EK

Adding the nPM2100 Fuel Gauge

Overview of necessary steps

- Adjust device tree file to add a I2C/TWI peripheral, use of overlays
- Include necessary libraries for the Nordic fuel gauge
- Make changes to our source code, and include helper functions
- Connect the nPM2100-EK with our nRF54L15-DK (verify I/O voltages)

Device Tree Configuration (HW)

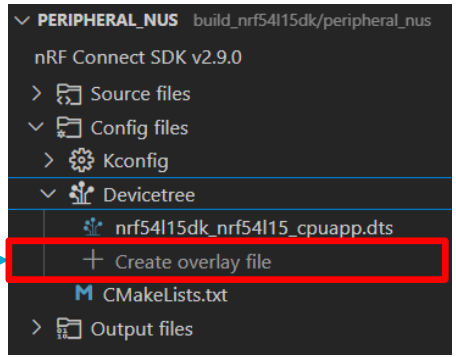
To use the PMIC/I2C peripheral we must make changes to our board device tree.

We need to do the following:

- Assign I/O pins to I2C peripheral (pinctrl)
- Enable the I2C peripheral
- Add information for the attached I2C device

We do this in the overlay file for our board

Create new overlay file



Add the pin-control

In the overlay file we add new I/O groups into the “pinctrl” segment.

We create a sleep and default state

```
&pinctrl {
    i2c21_default: i2c21_default {
        group1 {
            psels = <NRF_PSEL(TWIM_SDA, 1, 11)>,
                <NRF_PSEL(TWIM_SCL, 1, 12)>;
            bias-pull-up;
        };
    };

    i2c21_sleep: i2c21_sleep {
        group1 {
            psels = <NRF_PSEL(TWIM_SDA, 1, 11)>,
                <NRF_PSEL(TWIM_SCL, 1, 12)>;
            low-power-enable;
        };
    };
};
```

Enable I2C peripheral

In the overlay file

- add our PMIC device *(Fuel Gauge use only)*
- assign the pins to the I2C 21 peripheral *(on nRF54L15-DK I2C20 is blocked by UART20)*

Enable I2C - 21

Assign driver and reg. addr.

Select the I/O pins (pinctrl)

Define I/O pin state names

```
&i2c21 {
    status = "okay";
    npm2100ek_pmic: pmic@74 {
        compatible = "nordic,npm2100";
        reg = <0x74>;

        npm2100ek_vbat: vbat {
            compatible = "nordic,npm2100-vbat";
        };
    };
    pinctrl-0 = <&i2c21_default>;
    pinctrl-1 = <&i2c21_sleep>;
    pinctrl-names = "default", "sleep";
};
```

Project configuration (file:prj.conf, SW)

Enable the Zephyr sensor driver

```
CONFIG_SENSOR=y
```

Enable the Nordic Fuel Gauge Library
Enable the functionality for primary cells

```
CONFIG_NRF_FUEL_GAUGE=y
CONFIG_NRF_FUEL_GAUGE_VARIANT_PRIMARY_CELL=y
```

Enable floatf print for our debugging via the console

```
CONFIG_REQUIRES_FLOAT_PRINTF=y
```

Source Code Integration

Include file headers

```
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/sensor.h>
#include <nrf_fuel_gauge.h>
```

Add global variables for the fuel gauge library

```
static int64_t ref_time;
static const struct device *vbat;
static const struct battery_model_primary FG_model;
static const float battery_current = 5e-3f;
```

Add required helper functions to initialize and run the fuel gauge

```
static int read_sensors(const struct device *vbat,[…])
int fuel_gauge_init(const struct device *vbat)
int fuel_gauge_update(const struct device *vbat)
```

Device Tree Access on Source Code Level

Parameters, devices and other segments of the device tree can be accessed through device pointers and by node label names.

Example, access to the *vbat sensor* of the nPM2100.

```
static const struct device *vbat = DEVICE_DT_GET(DT_NODELABEL(npm2100ek_vbat));
```

The Vbat sensor is used to retrieve the voltage of the battery and temperature on the nPM2100 die, this data is used as input for the Nordic fuel gauge algorithm.

Select Fuel Gauge Model

The Nordic SDK comes with a set of predeveloped models for a primary cell fuel gauge.

Those can be found under:
→ <SDK-InstallationPath>\v3.0.0\nrfxlib\nrf_fuel_gauge\include\battery_models\primary_cell

Select for example the Alkaline AA model (single cell):

```
static const struct battery_model_primary FG_model = {
    #include <battery_models/primary_cell/AA_Alkaline.inc>
};
```

Main Routine Integration

In the main.c file or within the main software routine:

- Check if the (I2C) peripheral is connected
- Run the init function and continuously feed the fuel gauge driver for updates

```
if (!device_is_ready(vbat)) {
    printk("vbat device not ready.\n");
    return -1;
}
printk("PMIC device ok, init fuel gauge\n");
err = fuel_gauge_init(vbat);
if (err < 0) {
    printk("Could not initialise fuel gauge.\n");
    return -1;
}
printk("Fuel gauge initialised using model %s\n", FG_model.name);

while (1) {
    fuel_gauge_update(vbat);
    k_msleep(1000);
}
```

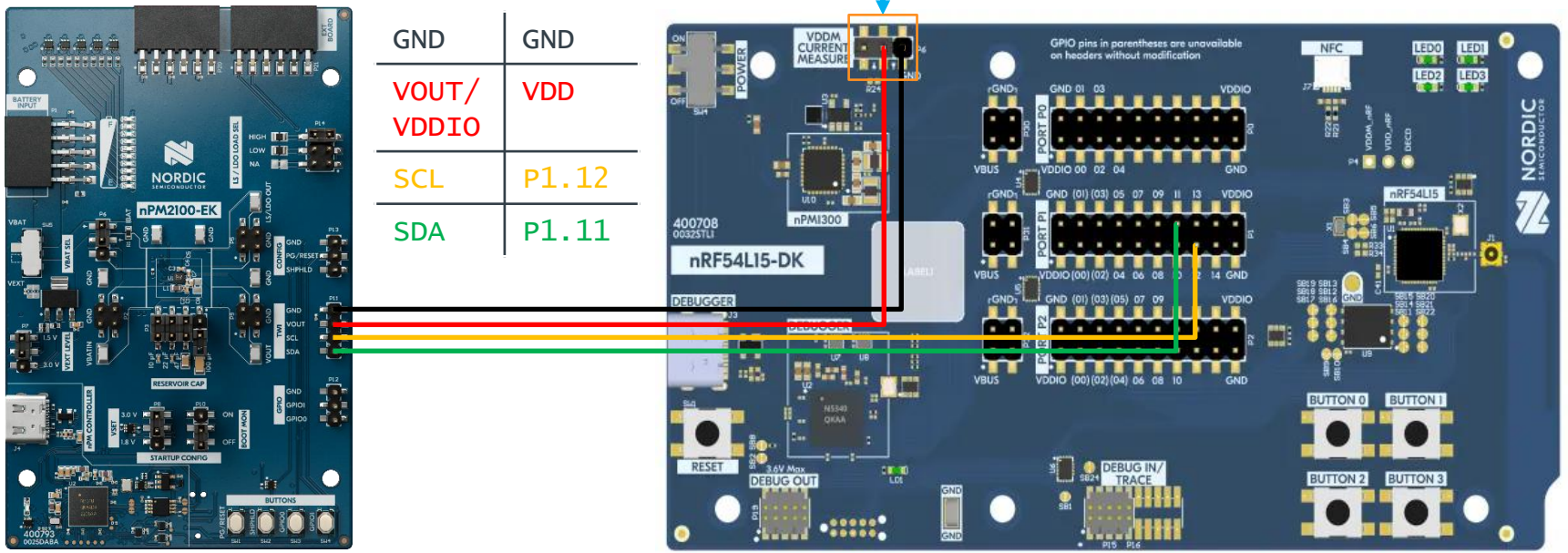
Connect nPM2100-EK via I2C (TWI)

STEP 1: Remove jumper P6 / VDDM Current Measure

STEP 2: Connect wiring as shown

STEP 3: Plug-in nPM2100 battery extension

GND	GND
VOUT/ VDDIO	VDD
SCL	P1.12
SDA	P1.11



Notes for connecting the nPM2100-EK

The default IO voltage on the nPM2100 is VOUT (def. selected with VSET = 3V).
The default IO voltage on the nRF54L15-DK is set by its VDD (def. supplied at 1.8V).
The shown configuration will also power the nRF54L15 SoC from the nPM2100 / battery.

nPM2100-EK	nRF54L15-DK
GND	GND
VOUT/VDDIO	VDD
SCL	P1.12
SDA	P1.11

To obtain equal I/O levels between both boards, nPM2100 will also power the nRF54L15 SoC.

The nPM2100-EK will by **default output 3V** at VOUT selected through Pin jumper **P8 / VSET**.

On the nRF54L15-DK, **remove jumper P6, VDDM current measure** to disconnect the DK's onboard PMIC (nPM1300).
nRF54L15 VDD is now supplied through pin 2 (center) of P6.

NORDIC®
SEMICONDUCTOR