# Zephyr: VBUS

## Medical Electrical Equipment (BME590L)

Mark L. Palmeri, M.D., Ph.D.

April 03, 2023

Duke | BME

# What is VBUS

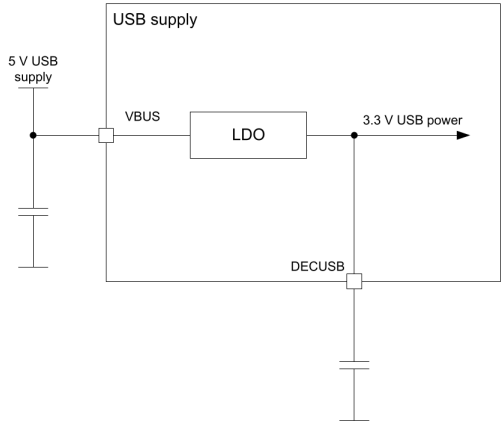


*Figure 21: USB voltage regulator*

# VBUS: Product Specification

## 5.3.2 USB supply

When using the USB peripheral, a 5 V USB supply needs to be provided to the VBUS pin.

The USB peripheral has a dedicated internal voltage regulator for converting the VBUS supply to 3.3 V used by the USB signalling interface (D+ and D- lines, and pull-up on D+). The remainder of the USB peripheral (USBD) is supplied through the main supply like other on-chip features. As a consequence, VBUS and either VDDH or VDD supplies are required for USB peripheral operation.

When VBUS rises into its valid range, the software is notified through a USBDETECTED event. A USBREMOVED event is sent when VBUS goes below its valid range. Use these events to implement the USBD start-up sequence described in the USBD chapter.

When VBUS rises into its valid range while the device is in System OFF, the device resets and transitions to System ON mode. The RESETREAS register will have the VBUS bit set to indicate the source of the wake-up.

See VBUS detection specifications on page 82 for the levels at which the events are sent ($V_{BUS,DETECT}$ and $V_{BUS,REMOVE}$) or at which the system is woken up from System OFF ($V_{BUS,DETECT}$).

When the USBD peripheral is enabled through the ENABLE register, and VBUS is detected, the regulator is turned on. A USBPWRRDY event is sent when the regulator's worst case settling time has elapsed, indicating to the software that it can enable the USB pull-up to signal a USB connection to the host.

The software can read the state of the VBUS detection and regulator output readiness at any time through the USBREGSTATUS register.

Duke | BME

# VBUS Detection

## 5.3.8.7 VBUS detection specifications

| Symbol | Description | Min. | Typ. | Max. | Units |
|---|---|---|---|---|---|
| $V_{BUS,DETECT}$ | Voltage at which rising VBUS gets reported by USBDETECTED | 3.4 | 4.0 | 4.3 | V |
| $V_{BUS,REMOVE}$ | Voltage at which decreasing VBUS gets reported by USBREMOVED | 3.0 | 3.6 | 3.9 | V |

Duke | BME

# Why is this important?

- Safety - shock risk!
- USB charging / power provides a physical connection to a power source.
- IEC60601 has very different safety testing requirements for "wall" powered vs. battery-powered devices.

# main.c

```
#include <nrfx_power.h>
```

## Warning: Non-Zephyr Library

Using the functionality of this library is non-ideal. Why?

- Board-specific (`nrfx`)
- Breaks the Zephyr ecosystem.

# main.c: create function to check VBUS status

```c
// declare function
int check_vbus(void);

// capture return code from function
err = check_vbus();
if (err) {
    break; // will cause the loop this function is called from to be exited
}
```

You may not need to "break" in response to `VBUS` detection; that is just an example of potential action to take.

Duke | BME

# check_vbus()

```
int check_vbus() {
    /* check for voltage on VBUS (USB-C charging cable attached)
    Returns:
        0 - VBUS not detected
        -1 - VBUS detected (need to kill device function)
    */
    usbregstatus = nrf_power_usbregstatus_vbusdet_get(NRF_POWER);;
    if (usbregstatus) {
        LOG_ERR("VBUS voltage detected.  Device cannot be operated while
        ↪  charging.");
        return -1;
    }
    else {
        LOG_DBG("VBUS voltage checked and not detected.");
    }
    return 0;
}
```

# Refactor Potential

- `nrf_power_usbregstatus_vbusdet_get()`: `NRFX`-specific function
- `NRF_POWER`: `NRFX`-specific macro
- These are not defined for non-`NRFX` boards!
- Would be much better to implement this using Zephyr: USB-C VBUS.

Duke | BME