nRF52 DK

User Guide

v1.3.1

4397_500 v1.3.1 / 2020-04-08

NORDIC
SEMICONDUCTOR

Contents

Revision history

Date	Version	Description
April 2020	1.3.1	Editorial changes
December 2018	1.3	Updated: • Introduction on page 4 • Software tools on page 7 • Measuring current on page 21 • Preparing the development kit on page 21 • NFC antenna interface on page 27
February 2017	1.2	• Created PDF for nRF52 DK v1.1.x (valid for all DK versions) • Added MSD on page 11 • Updated: • Interface MCU firmware on page 10 • Solder bridge configuration on page 28

1 Introduction

The nRF52 DK is a Development Kit (DK) that includes hardware, firmware source code, documentation, hardware schematics, and layout files.

This kit can be used for developing for either the nRF52832 or nRF52810 SoCs.

The key features of the development kit are:

- nRF52832 flash-based ANT™/ANT+™, *Bluetooth*® Low Energy SoC solution
- Buttons and LEDs for user interaction
- I/O interface for Arduino form factor plug-in modules
- SEGGER J-Link OB debugger with debug out functionality
- Virtual COM port interface via UART
- Drag-and-drop Mass Storage Device (MSD) programming
- Supporting NFC-A listen mode
- Mbed Enabled

For access to firmware source code, hardware schematics, and layout files, see www.nordicsemi.com.

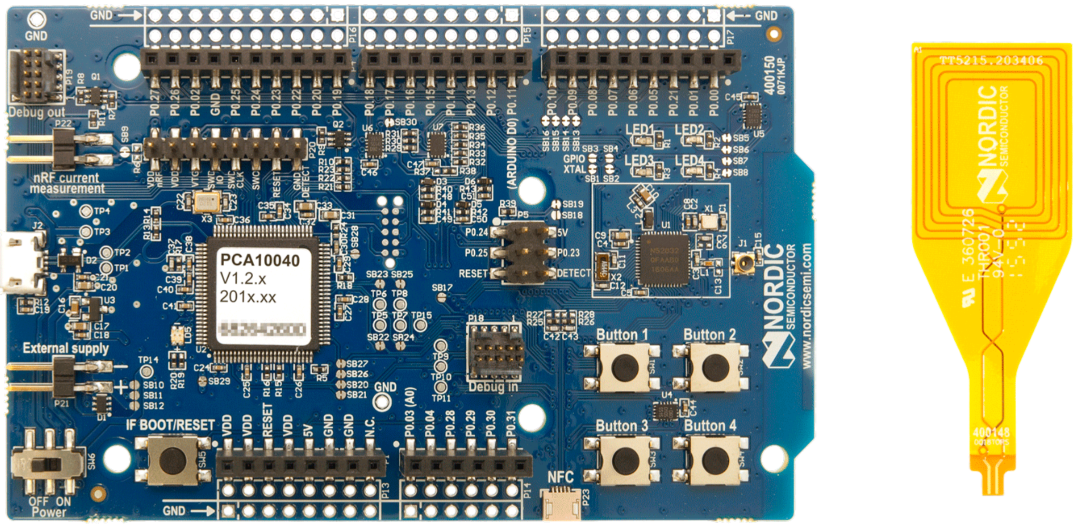


Figure 1: nRF52 DK (PCA10040) v1.2.x and NFC tag

Developing with nRF52810

In addition to developing with the nRF52832, you can also use this kit for developing with the nRF52810. This is possible because the nRF52810 is a subset of nRF52832 with smaller RAM and flash and with less peripheral resources. Starting from nRF5 SDK v14.1.0, the SDK supports developing for the nRF52810 including an example project that emulates the nRF52810. This project is located in the `pca10040e` subfolder.

For more detailed documentation regarding nRF52810 development, see Developing for nRF52810 in the SDK documentation.

Environmental Protection

Waste electrical products should not be disposed of with household waste.

Please recycle where facilities exist. Check with your local authority or retailer for recycling advice.

2 Setting up the development kit

Before you start developing, prepare your development kit hardware by completing a few easy steps and download the required software.

1. To set up the hardware, follow the instructions in Getting started with the nRF52 DK.
2. To set up the software, follow the instructions in Nordic tools and downloads in nRF5 Getting Started. Actual software required depends on your OS and Development IDE.

3 Software tools

The extensive range of supporting software tools help you with testing and programming on your chip.

S112 and S132 SoftDevices

Bluetooth Low Energy concurrent multi-link protocol stack solution supporting simultaneous Central/Peripheral/Broadcaster/Observer role connections.

nRF5 SDK

The nRF5 Software Development Kit (SDK) provides source code of examples and libraries forming the base of your application development.

nRF Command Line Tools

nRF Command Line Tools is a package that contains JLinkARM, JLink CDC, nRFjprog, and mergehex. The nRFjprog is a command line tool for programming nRF5 Series chips. It is also useful in a production setup. See also nRF Command Line Tools.

nRF5x-pynrfjprog

the nRF5x-pynrfjprog utility is a simple Python interface for the nrfjprog DLL. It is useful for scripting, especially in automated tests. See also nRF5x pynrfjprog.

nRF Connect for Desktop

nRF Connect is a desktop application for developing with Nordic devices. It consists of a core delivery in which apps can be plugged to add different functionalities. All apps are available for download from nRF Connect for Desktop and new apps will be available upon release.

The core apps supporting the nRF52 DK are Bluetooth Low Energy, Programmer, and Power Profiler.

nRF Connect for Mobile

nRF Connect for mobile is a powerful generic tool that allows you to scan and explore your Bluetooth Low Energy devices and communicate with them on a smartphone. nRF Connect for mobile supports a number of Bluetooth SIG adopted profiles together with the Device Firmware Update (DFU) profile from Nordic Semiconductor.

We also recommend some third party software tools that are useful when developing with our products:

Keil MDK-ARM Development Kit

Keil MDK-ARM Development Kit is a development environment specifically designed for microcontroller applications that lets you develop using the nRF5 SDK application and example files.

SEGGER J-Link Software

The J-Link software is required to debug using the J-Link hardware that is packaged with our development kits.

SEGGER Embedded Studio (SES)

SES is a complete cross-platform Integrated Development Environment (IDE), so you can run it on different operating systems. See also Getting started with nRF5 SDK and SES (nRF51 & nRF52 Series).

4 Start developing

After you have set up the development kit and installed the toolchain, it is time to start developing.

There are several ways to continue from here:

- Running precompiled examples

  See the step-by-step instructions on how you can quickly test a precompiled example without having to use the full toolchain, it is a matter of copying and pasting a precompiled hex file onto your development kit.
- Compiling and running a first example

  Test that you have set up your toolchain correctly by compiling, programming and running a very simple example.
- Running examples that use a SoftDevice

  Before you can run more advanced examples that use Bluetooth or ANT, you must first program the SoftDevice on the DK.

5 Interface MCU

The interface MCU on the DK is running either SEGGER J-Link OB or mbed OB interface firmware and is used to program and debug the firmware of the nRF52832 IC.

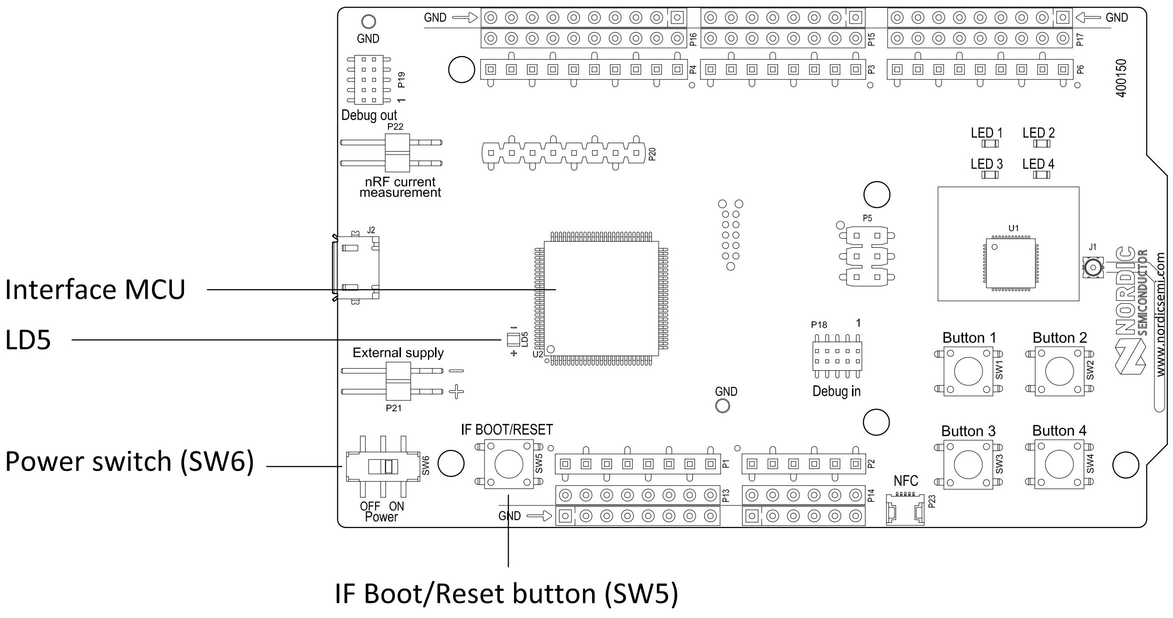


Figure 2: Interface MCU

5.1 IF Boot/Reset button

The nRF52 DK is equipped with an IF (Interface) Boot/Reset button (**SW5**).

This button is connected to the interface MCU on the DK and has two functions:

- Resetting the nRF52832 device.
- Entering bootloader mode of the interface MCU.

During normal operation the button will function as a reset button for the nRF52832 device. For this to work, pin reset on P0.21 needs to be enabled for the nRF52832 device. The button is also used to enter the bootloader mode of the interface MCU. To enter bootloader mode, keep the reset button pressed while powering up the DK until LED (**LD5**) starts to blink. You can power up the DK either by disconnecting and reconnecting the USB cable, or toggle the power switch (**SW6**).

Note: Pin reset can be enabled by defining the CONFIG_GPIO_AS_PINRESET variable in the project settings. This can be done by defining the preprocessor symbol in Keil. Go to: **Project** > **Options for Target** > **C/C++** > **Preprocessor Symbols** > **Define** and add the CONFIG_GPIO_AS_PINRESET variable after NRF52.

This functionality can be removed by doing a `nRFjprog --recover`.

5.2 Virtual COM port

The onboard interface MCU features a virtual COM port via UART.

The virtual COM port has the following features:

- Flexible baud rate setting up to 1 Mbps.[1]
- Dynamic Hardware Flow Control (HWFC) handling.
- Tri-stated UART lines when no terminal is connected.

Table 1: Relationship of UART connections on nRF52832 and the interface MCU on page 10 shows an overview of the UART connections on nRF52832 and the interface MCU.

Default GPIO nRF52832	UART nRF52832	Interface MCU UART
P0.05	RTS	CTS
P0.06	TXD	RXD
P0.07	CTS	RTS
P0.08	RXD	TXD

Table 1: Relationship of UART connections on nRF52832 and the interface MCU

The UART signals are routed directly to the interface MCU. The UART pins connected to the interface MCU are tri-stated when no terminal is connected to the virtual COM port on the computer.

Note: The terminal software used must send a DTR signal in order to configure the UART interface MCU pins.

The **P0.05** (RTS) and **P0.07** (CTS) can be used freely when HWFC is disabled on the nRF52832.

Note: The mbed OB interface does not support HWFC through the virtual com port.

5.2.1 Dynamic Hardware Flow Control (HWFC) handling

When the interface MCU receives a DTR signal from a terminal, it performs automatic HWFC detection.

Automatic HWFC detection is done by driving **P0.07** (CTS) from the interface MCU and evaluating the state of **P0.05** (RTS) when the first data is sent or received. If the state of **P0.05** (RTS) is high, HWFC is assumed not to be used. If HWFC is not detected, both CTS and RTS can be used freely by the nRF application.

After a power-on reset of the interface MCU, all UART lines are tri-stated when no terminal is connected to the virtual COM port. Due to the dynamic HWFC handling, if HWFC has been used and detected, **P0.07** (CTS) will be driven by the interface MCU until a power-on reset has been performed or until a new DTR signal is received and the detection is redone. To ensure that the UART lines are not affected by the interface MCU, the solder bridges for these signals can be cut and later resoldered if needed. This might be necessary if UART without HWFC is needed while **P0.05** (RTS) and **P0.07** (CTS) are used for other purposes.

5.3 Interface MCU firmware

The onboard interface MCU is factory programmed with an mbed-compliant bootloader. This feature makes it possible to swap interface firmware between the factory preloaded SEGGER J-Link OB and the mbed interface firmware.

See section IF Boot/Reset button on page 9 on how to enter the bootloader.

[1] Baud rate 921 600 is not supported through the Virtual COM port.

To swap interface MCU firmware, simply drag the Interface image (.bin) into the mounted bootloader drive on the connected computer and power cycle the DK.

Both the mbed interface firmware and the J-Link OB image can be downloaded from www.nordicsemi.com.

Note:

- The J-Link serial number is linked to the interface MCU and will not change even when swapping the interface MCU firmware, so it can be useful to write the serial number on a sticker on the DK.
- When in bootloader mode, do not drag and drop any file except those downloaded from www.nordicsemi.com for use with the interface MCU. If a wrong file is used, it can overwrite the bootloader and ruin the interface MCU firmware without the possibility of recovery.

5.4 MSD

The interface MCU features a Mass Storage Device (MSD). This makes the development kit appear as an external drive on your computer.

This drive can be used for drag-and-drop programming. Files cannot be stored on this drive. By copying a HEX file to the drive, the interface MCU will program the file to the device.

Note:

- Windows might try to defragment the MSD part of the interface MCU. If this happens, the interface MCU will disconnect and be unresponsive. To return to normal operation, the development kit must be power cycled.
- Your antivirus software might try to scan the MSD part of the interface MCU. Some antivirus programs trigger a false positive alert in one of the files and quarantine the unit. If this happens, the interface MCU will become unresponsive.
- If the computer is set up to boot from USB, it can try to boot from the development kit if the development kit is connected during boot. This could be avoided by unplugging the development kit before a computer restart, or changing the boot sequence of the computer.

You can also disable the MSD of the kit by using the **`msddisable`** command in J-Link Commander. To enable, use the **`msdenable`** command. These commands take effect after a power cycle of the development kit and should stay this way until changed again.

6 Hardware description

The nRF52 DK (PCA10040) can be used as a development platform for the nRF52832 and nRF52810 devices. It features an onboard programming and debugging solution.

In addition to radio communication, the nRF52832 device can communicate with a computer through a virtual COM port provided by the Interface MCU.

6.1 Hardware drawings

nRF52 DK hardware drawings show both sides of the PCA10040.

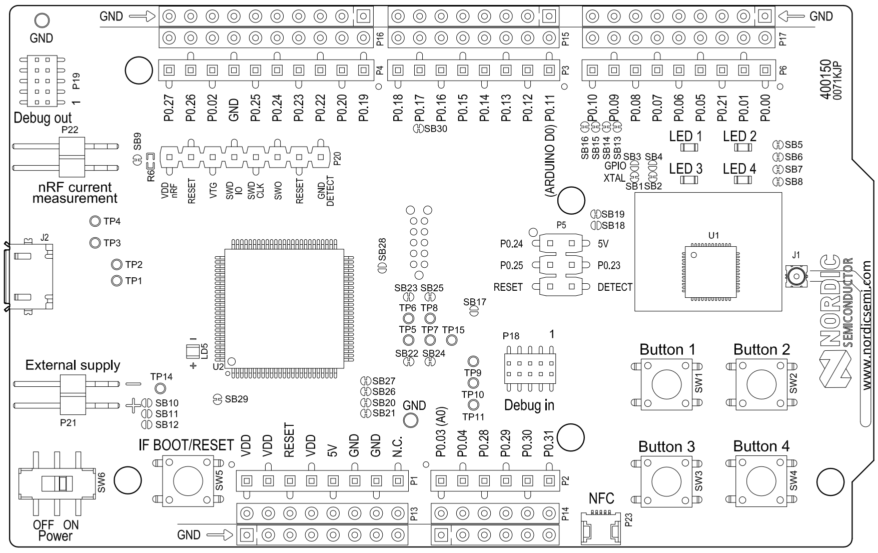
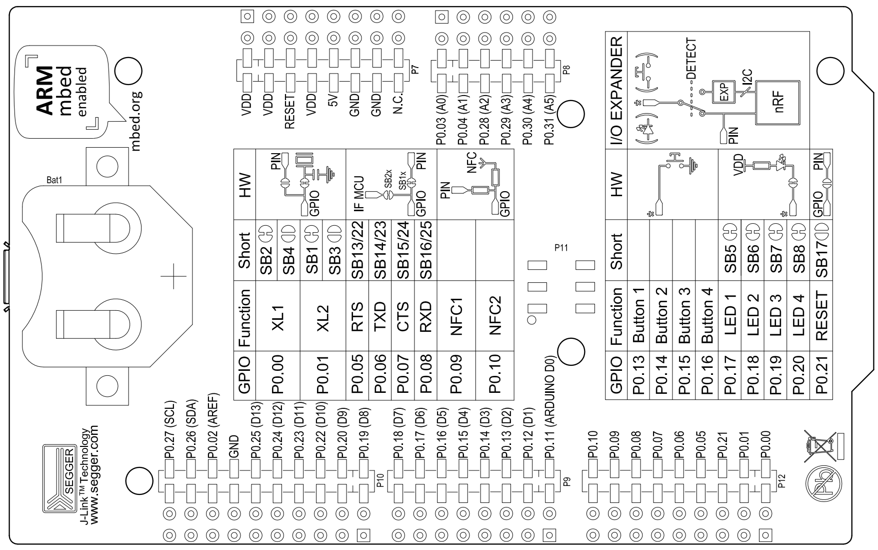


Figure 3: nRF52 DK (PCA10040) top view



ARM mbed enabled
mbed.org
Bat1
SEGGER
J-Link™ Technology
www.segger.com
VDD VDD RESET VDD 5V GND GND N.C.
P7
P0.03 (A0) P0.04 (A1) P0.28 (A2) P0.29 (A3) P0.30 (A4) P0.31 (A5)
P8
I/O EXPANDER
DETECT
EXP
I2C
nRF
PIN
HW
Short
Function
GPIO
SB2 SB4 SB1 SB3 SB13/22 SB14/23 SB15/24 SB16/25
XL1 XL2 RTS TXD CTS RXD NFC1 NFC2
P0.00 P0.01 P0.05 P0.06 P0.07 P0.08 P0.09 P0.10
IF MCU
NFC
P11
SB5 SB6 SB7 SB8 SB17
Button 1 Button 2 Button 3 Button 4 LED 1 LED 2 LED 3 LED 4 RESET
P0.13 P0.14 P0.15 P0.16 P0.17 P0.18 P0.19 P0.20 P0.21
P0.27 (SCL) P0.26 (SDA) P0.02 (AREF) GND P0.25 (D13) P0.24 (D12) P0.23 (D11) P0.22 (D10) P0.20 (D9) P0.19 (D8)
P10
P0.18 (D7) P0.17 (D6) P0.16 (D5) P0.15 (D4) P0.14 (D3) P0.13 (D2) P0.12 (D1) P0.11 (ARDUINO D0)
P9
P0.10 P0.09 P0.08 P0.07 P0.06 P0.05 P0.21 P0.01 P0.00
P12

Figure 4: nRF52 DK (PCA10040) bottom view

6.2 Block diagram

The nRF52 DK block diagram shows the connections between the different blocks.

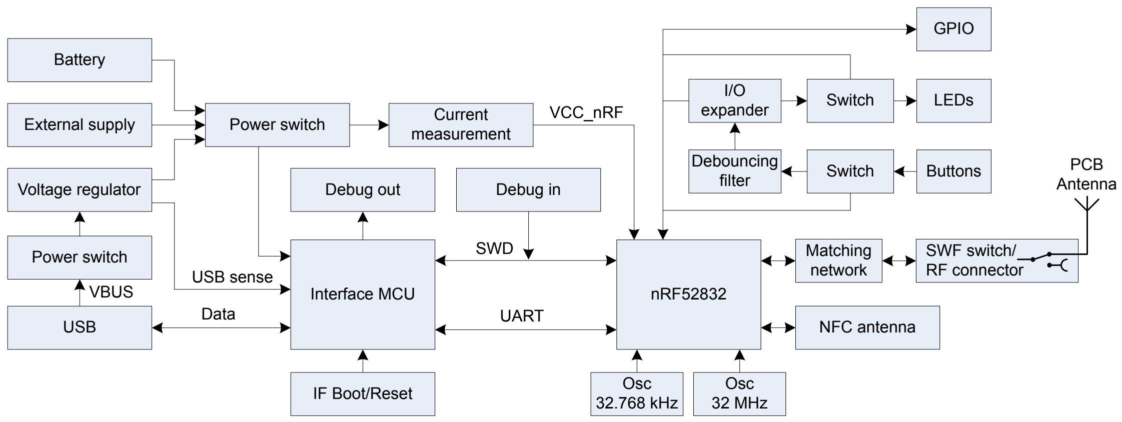


Figure 5: nRF52 DK block diagram

6.3 Power supply

The nRF52 DK has three power options: 5 V from the USB, external power supply, and coin cell battery.

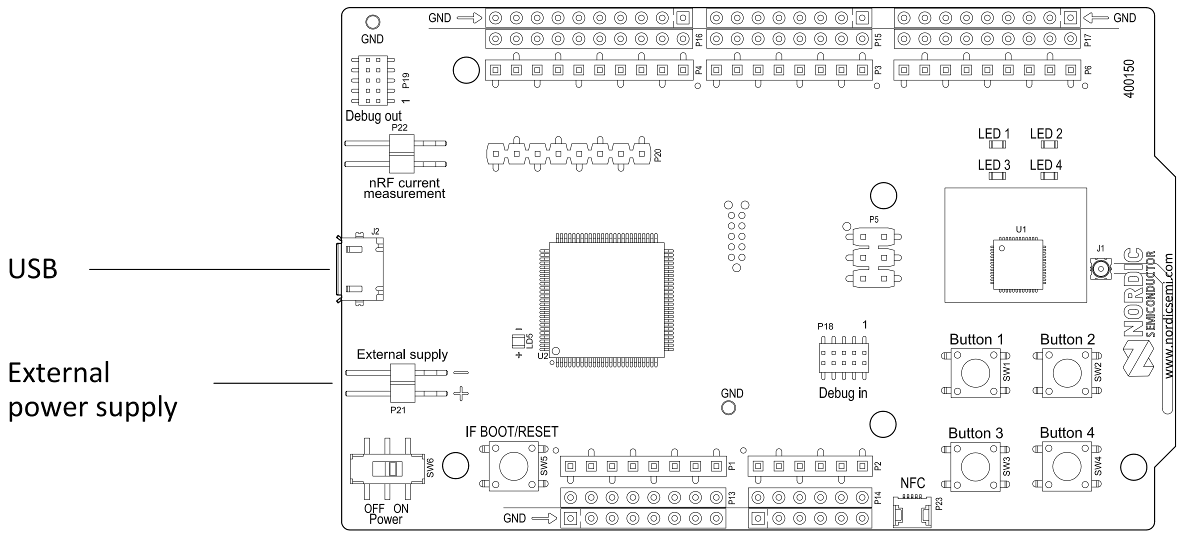


Figure 6: Power supply options, front

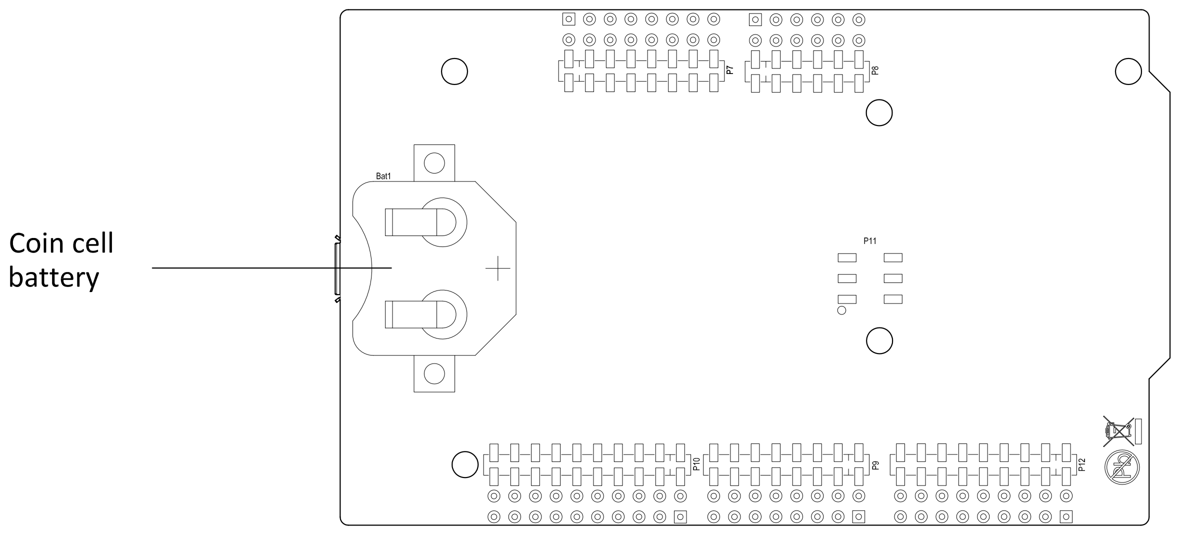


Figure 7: Power supply options, back

The 5 V from the USB is regulated down to 3.3 V through an onboard voltage regulator. The battery and external power supply are not regulated. The power sources are routed through a set of diodes (**D1A**, **D1B**, and **D1C**) for reverse voltage protection, where the circuit is supplied from the source with the highest voltage.

Note: When USB is not powered, the interface MCU is in dormant state and will draw an additional current of ~ 20 µA in order to maintain the reset button functionality. This will affect current consumption, but not the nRF52832 current measurements, as described in the Measuring current on page 21 section.

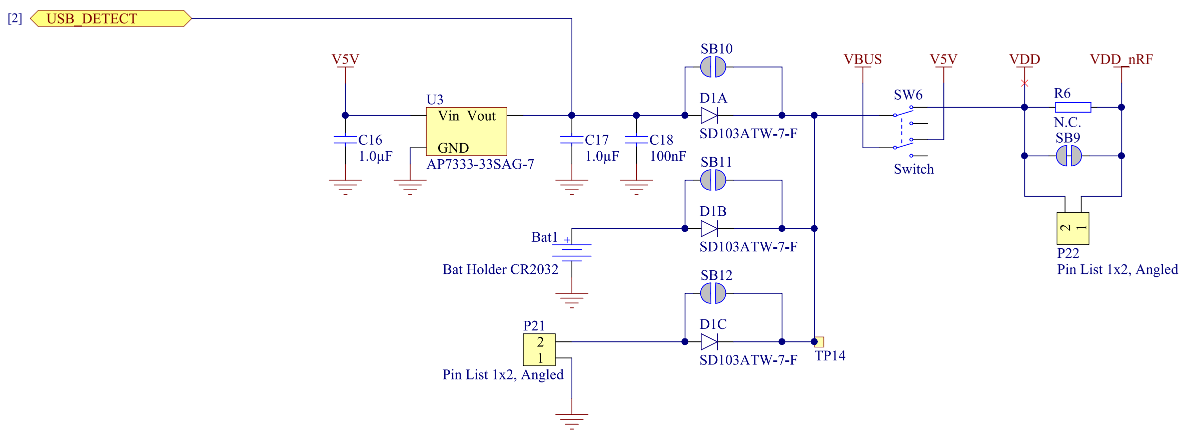


Figure 8: Power supply circuitry

The reverse voltage protection diodes will add a voltage drop to the supply voltage of the circuit. To avoid this voltage drop, the diodes can be bypassed by shorting one or more solder bridges.

Power source	Protection bypass	Voltage level
USB	SB10	3.3 V
Coin-cell battery	SB11	Battery
External supply	SB12	1.7 V - 3.6 V

Table 2: Protection diode bypass solder bridges

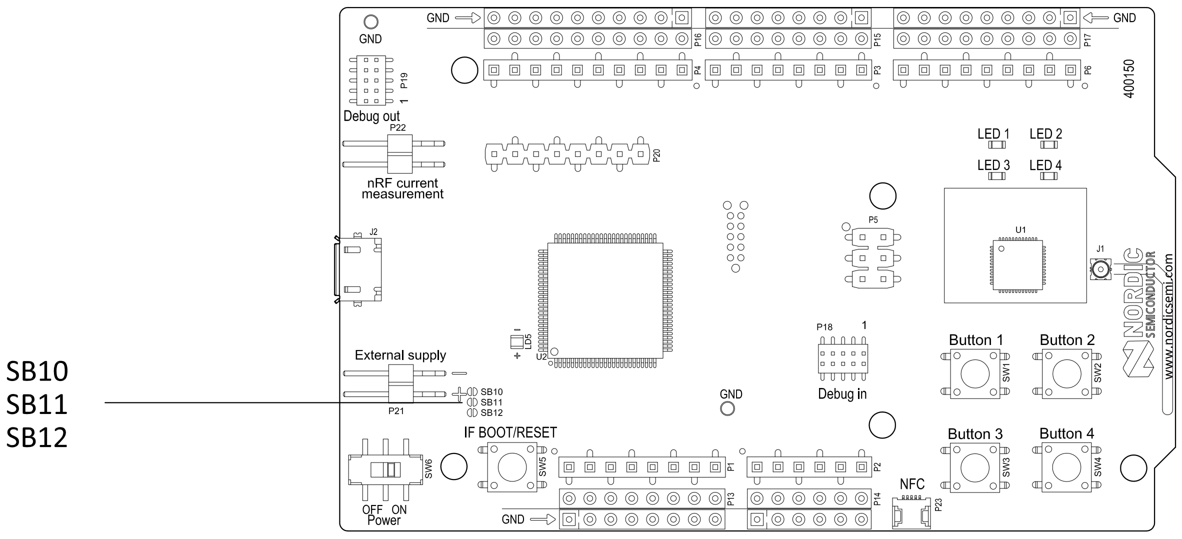


Figure 9: Protection diode bypass solder bridges

Note: Connect only one power source at the time. Shorting the solder bridges removes the reverse voltage protection.

6.4 Connector interface

Access to the nRF52832 GPIOs is available from connectors **P2**, **P3**, **P4**, **P5**, and **P6**. The **P1** connector provides access to ground and power on the nRF52 DK.

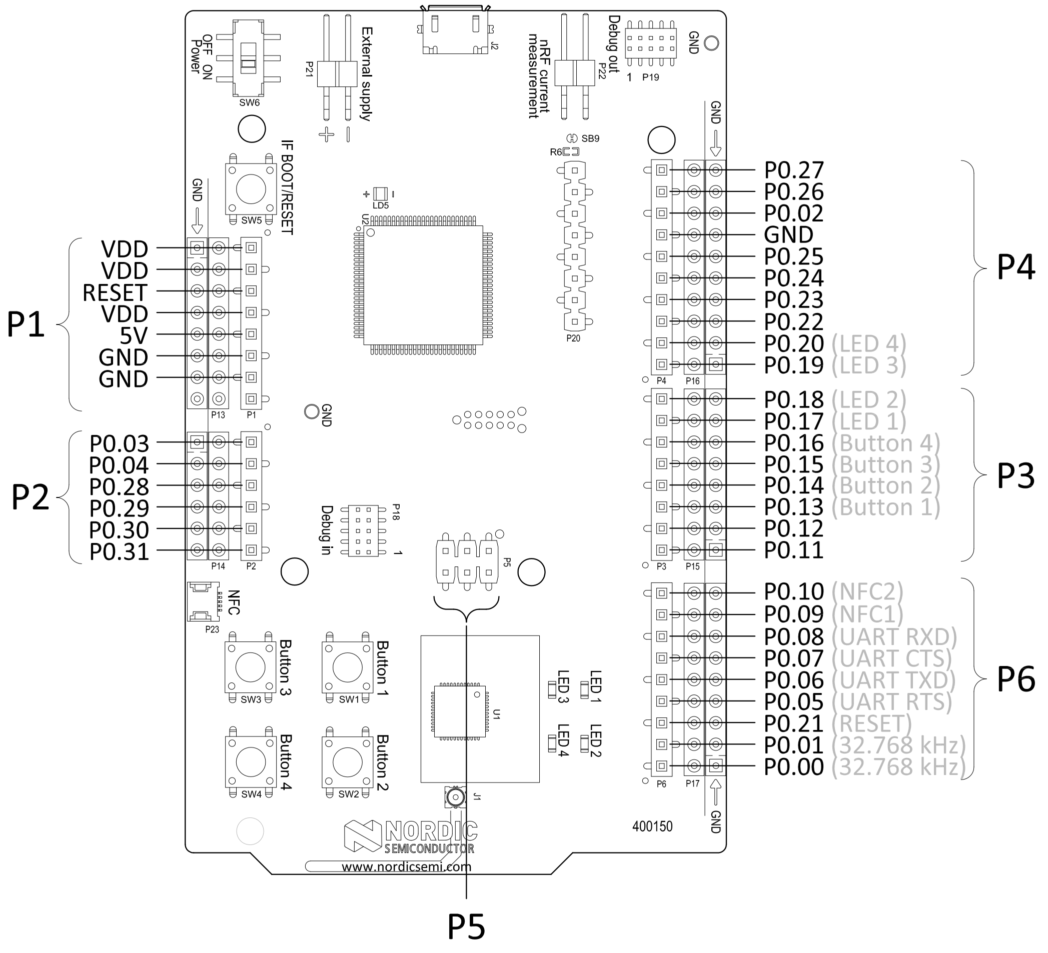


Figure 10: nRF52 DK connectors

The signals are also available on connectors **P7**, **P8**, **P9**, **P10**, **P11**, and **P12**, which are on the bottom side of the DK. By mounting pin lists on the connector footprints, the nRF52 DK can be used as a shield for Arduino motherboards[2] or other boards that follow the Arduino standard.

For easy access to GPIO, power, and ground, the signals can also be found on the through-hole connectors **P13**–**P17**.

Note: Some pins have default settings.

- **P0.00** and **P0.01** are by default used for the 32 kHz crystal and are not available on the connectors. For more information, see Section 32.768 kHz crystal on page 20.
- **P0.05**, **P0.06**, **P0.07**, and **P0.08** are by default used by the UART connected to the interface MCU. For more information, see Section Virtual COM port on page 9.
- **P0.09** and **P0.10** are by default used by NFC1 and NFC2. For more information, see Section NFC antenna interface on page 27.
- **P0.13**–**P0.20** are by default connected to the buttons and LEDs. For more information, see Section Buttons and LEDs on page 17.

When the nRF52 DK is used as a shield together with an Arduino standard motherboard, the Arduino signals are routed as shown in Figure 11: Arduino signals routing on the nRF52 DK on page 17.

[2] Only 3.3 V Arduino boards.

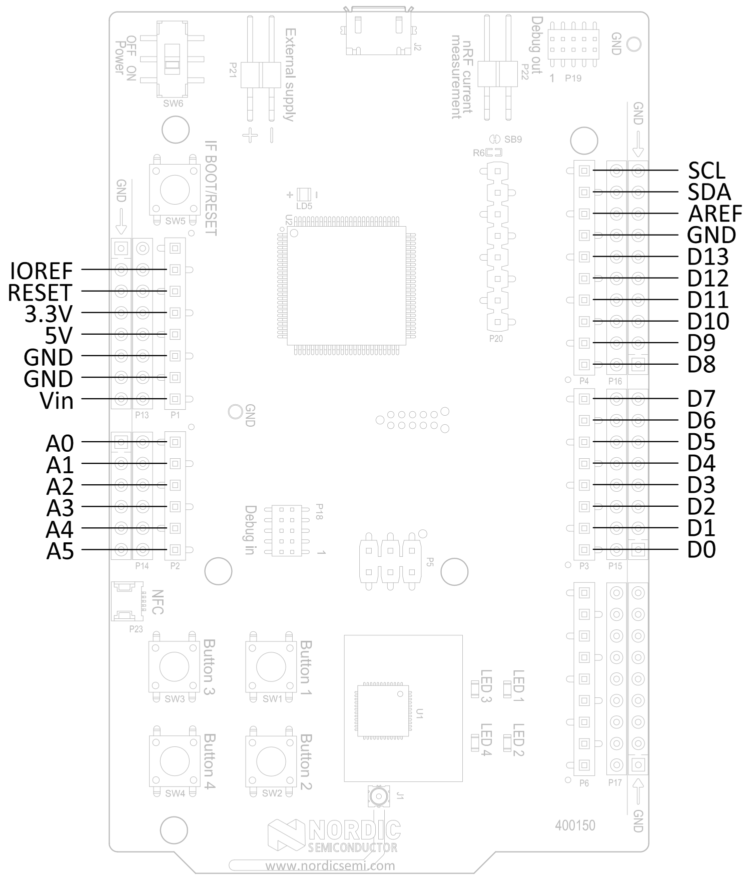


Figure 11: Arduino signals routing on the nRF52 DK

6.5 Buttons and LEDs

The four buttons and four LEDs on the nRF52 DK are connected to dedicated I/Os on the nRF52832 chip.

Part	GPIO	Short
Button 1	P0.13	-
Button 2	P0.14	-
Button 3	P0.15	-
Button 4	P0.16	-
LED 1	P0.17	SB5
LED 2	P0.18	SB6
LED 3	P0.19	SB7
LED 4	P0.20	SB8

Table 3: Button and LED connection

If GPIO **P0.17**–**P0.20** are needed elsewhere, the LEDs can be disconnected by cutting the short on **SB5**–**SB8**, see Figure 12: Disconnecting the LEDs on page 18. The LEDs and buttons can also be disconnected by using the I/O expander as described in I/O expander for buttons and LEDs on page 18.

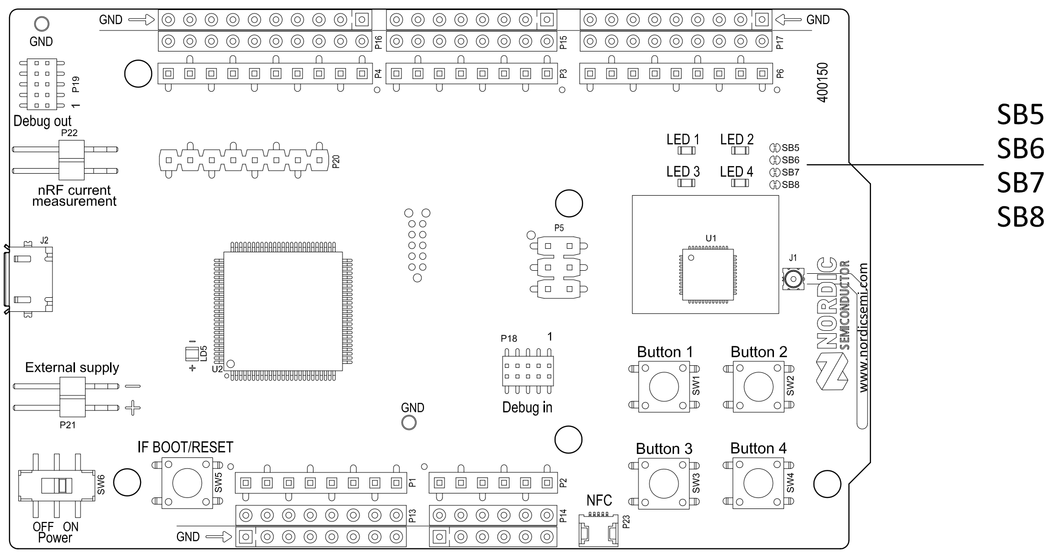


Figure 12: Disconnecting the LEDs

The buttons are active low, meaning the input will be connected to ground when the button is activated. The buttons have no external pull-up resistor, so to use the buttons the **P0.13**–**P0.16** pins must be configured as an input with an internal pull-up resistor.

The LEDs are active low, meaning that writing a logical zero ('0') to the output pin will illuminate the LED.

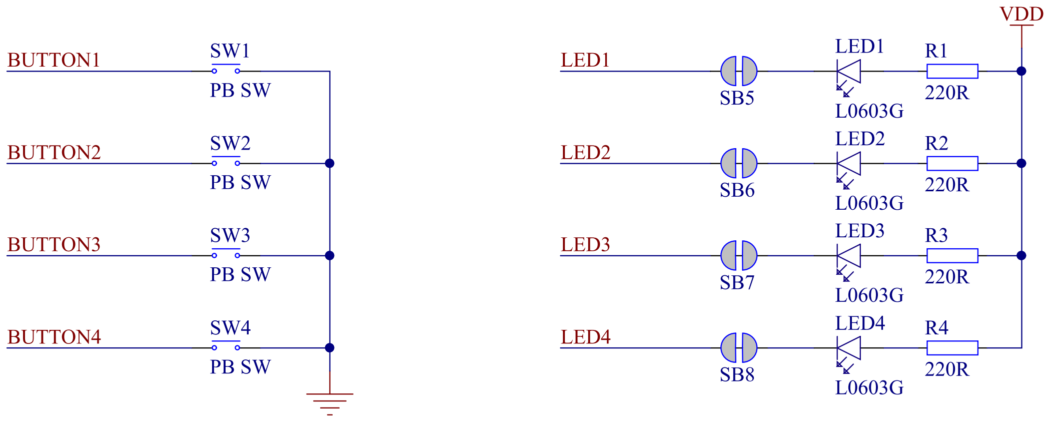


Figure 13: Button and LED configuration

6.5.1 I/O expander for buttons and LEDs

The nRF52 DK has an I/O expander to avoid conflicts with boards that follow the Arduino standard, the onboard GPIOs for the buttons and LEDs would otherwise possibly conflict with such boards.

GPIO	Part	Arduino signal
P0.13	Button 1	D2
P0.14	Button 2	D3
P0.15	Button 3	D4
P0.16	Button 4	D5
P0.17	LED 1	D6
P0.18	LED 2	D7
P0.19	LED 3	D8
P0.20	LED 4	D9

Table 4: GPIO connection

The I/O expander will release these GPIOs for general use when the nRF52 DK is used together with boards that follows the Arduino standard. The I/O expander can be permanently enabled by shorting solder bridge **SB18** or permanently disabled by cutting the shorting track on **SB19**. You must also short **SB18** when cutting **SB19** for full compatibility with the Arduino standard.

The I/O expander can be temporarily enabled by connecting SHIELD DETECT to ground.

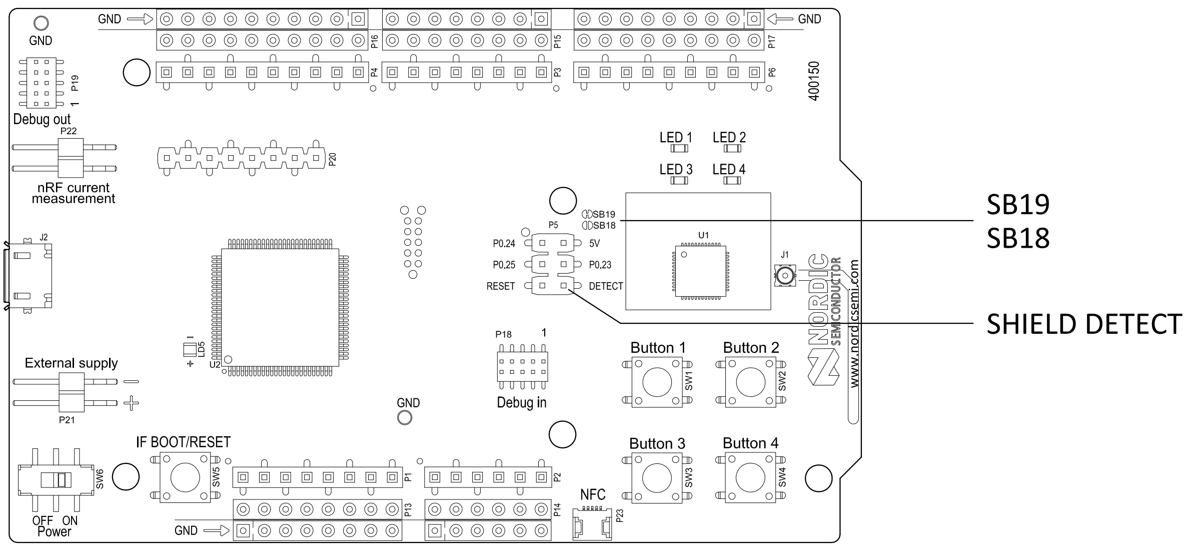


Figure 14: Enable or disable I/Os for Arduino standard

In addition to the buttons and LEDs, the following GPIOs are used for the I/O expander:

I/O expander signal	GPIO
/INT	P0.17
SDA	P0.26
SCL	P0.27

Table 5: I/O expander connection

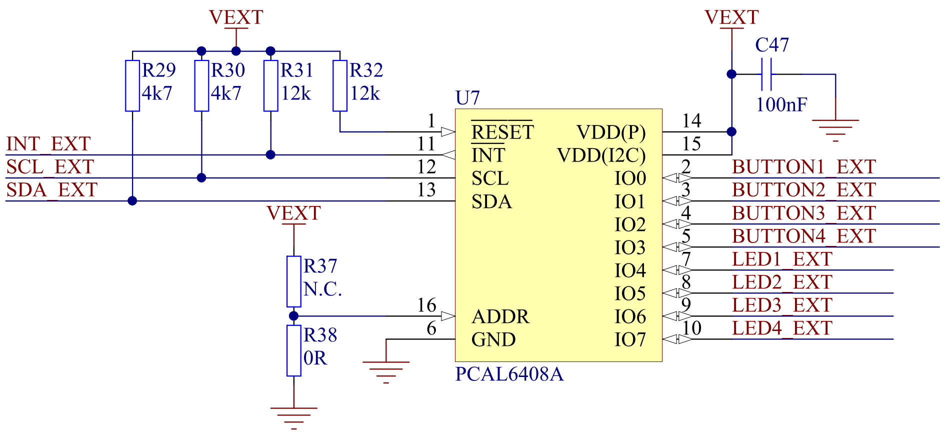


Figure 15: I/O expander schematic

Note: SW debouncing should not be needed when using the I/O expander. Each button on the nRF52 DK is equipped with a debouncing filter.

6.6 32.768 kHz crystal

nRF52832 can use an optional 32.768 kHz crystal (X2) for higher accuracy and lower average power consumption.

On the nRF52 DK, **P0.00** and **P0.01** are by default used for the 32.768 kHz crystal and are not available as a GPIO on the connectors.

Note: When using ANT/ANT+, the 32.768 kHz crystal (X2) is required for correct operation.

If **P0.00** and **P0.01** are needed as normal I/Os, the 32.768 kHz crystal can be disconnected and the GPIO routed to the connectors. Cut the shorting track on **SB1** and **SB2**, and solder **SB3** and **SB4**. See Figure 16: Configuring P0.00 and P0.01 on page 20 for reference.

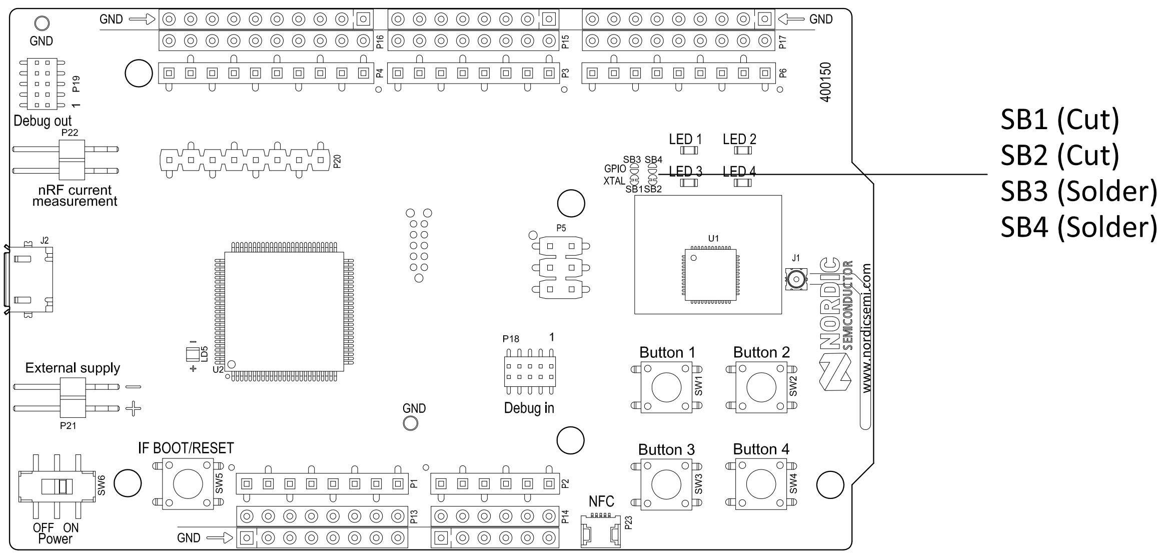


Figure 16: Configuring P0.00 and P0.01

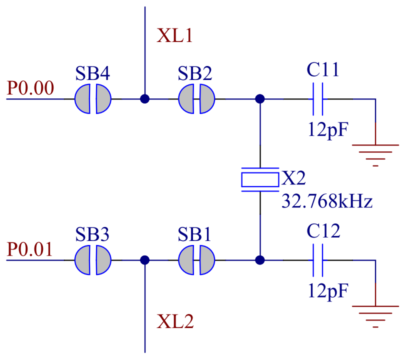


Figure 17: 32.768 kHz crystal and SB1 to SB4 schematic

6.7 Measuring current

The current drawn by the nRF52832 device can be monitored on the nRF52 DK.

There are several types of test equipment that can be used to measure current, each type has some advantages and some disadvantages. The different test equipment types are:

- Power analyzer
- Oscilloscope
- Ampere-meter
- Power Profiler Kit

Power analyzer and Power Profiler Kit measurements will not be described in the present document. For more information on Power Profiler Kit, see Power Profiler Kit User Guide.

See the section Using an oscilloscope for current profile measurement on page 22 for instructions.

See the section Using an ampere-meter for current measurement on page 23 for instructions.

Note: When measuring the current consumption:

- Do not use the USB connector to power the DK during current measurements. Power the DK from a coin cell battery, or use an external power supply on the External Supply connector **P21**.
- The current measurements will become unreliable when a serial terminal is connected to the Virtual COM port.
- After programming the nRF52832 device, the USB must be disconnected and the development kit power cycled to reset the debugger chip before current measurement.

For more information on current measurement, see the tutorial Current measurement guide: Introduction.

6.7.1 Preparing the development kit

To measure the current, you must first prepare the development kit.

The suggested configurations actually split the power domains for the SoC and the rest of the DK, and bypass protection components in the power supply chain.

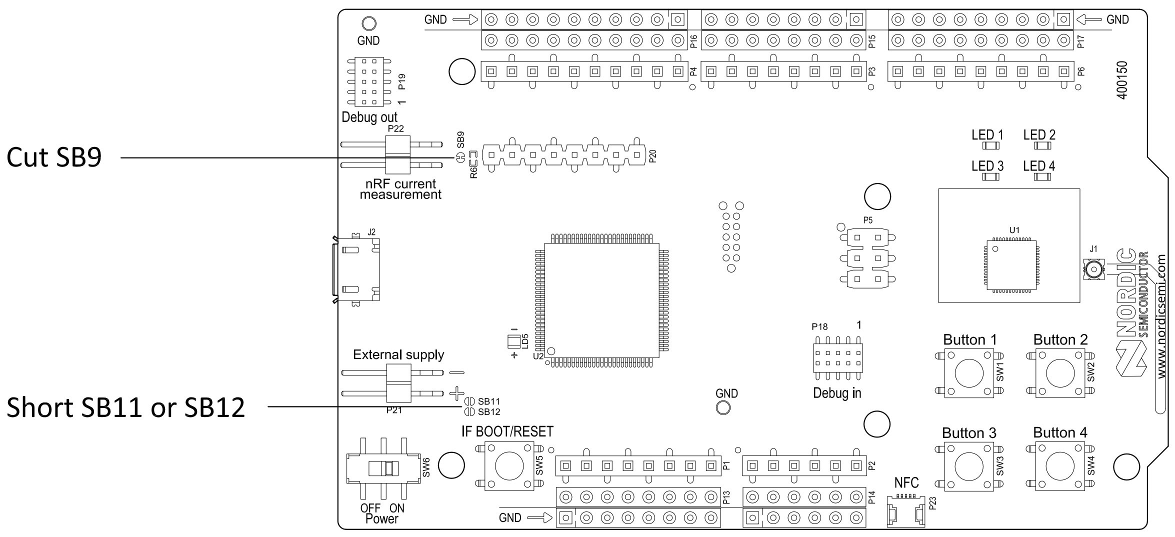


Figure 18: Prepare the development kit for current measurements

1. Cut the PCB track shorting solder bridge **SB9** to put **P22** in series with the load.
2. Unless powered by USB, short solder bridge **SB11** (if using coin cell battery) or **SB12** (if using external power supply) to bypass the protection diode which would otherwise give a voltage drop.

Note: While **SB11** or **SB12** is shorted, the development kit must not be powered from the USB if there is a battery or external supply connected because the protection diode has been bypassed.

To restore normal kit function after measurement:

- Solder **SB9** or apply a jumper on **P22**
- Cut or de-solder **SB11** or **SB12** to reconnect the protection diode

To reprogram the nRF chip while the DK is prepared for current measurement, disconnect external supply, ensure there is no battery inserted, remove measurement devices from **P22**, add a jumper to **P22**, and then connect the USB cable.

6.7.2 Using an oscilloscope for current profile measurement

Follow the step by step instructions below to measure the current using an oscilloscope.

1. Make sure you have prepared the development kit as described in Preparing the development kit on page 21.
2. Mount a 10 Ω resistor on the footprint for **R6**.
3. Connect an oscilloscope in differential mode or similar with two probes on the pins of the **P22** connector as shown in Figure 19: Current measurement with oscilloscope on page 23.
4. Calculate or plot the instantaneous current from the voltage drop across the 10 Ω resistor by taking the difference of the voltages measured on the 2 probes. The voltage drop will be proportional to the current. The 10 Ω resistor will cause a 10 mV drop for each 1 mA drawn by the circuit being measured.
5. The plotted voltage drop can be used to calculate the current at a given point in time, calculate average current over a period, or integrated to calculate the energy used over a period.

Some tips to reduce noise:

- Use probes with 1× attenuation
- Enable averaging mode to reduce random noise
- Enable high resolution function if available

Use minimum 200 kSa/s (one sample every 5 µs) to be able to get the correct average current measurement.

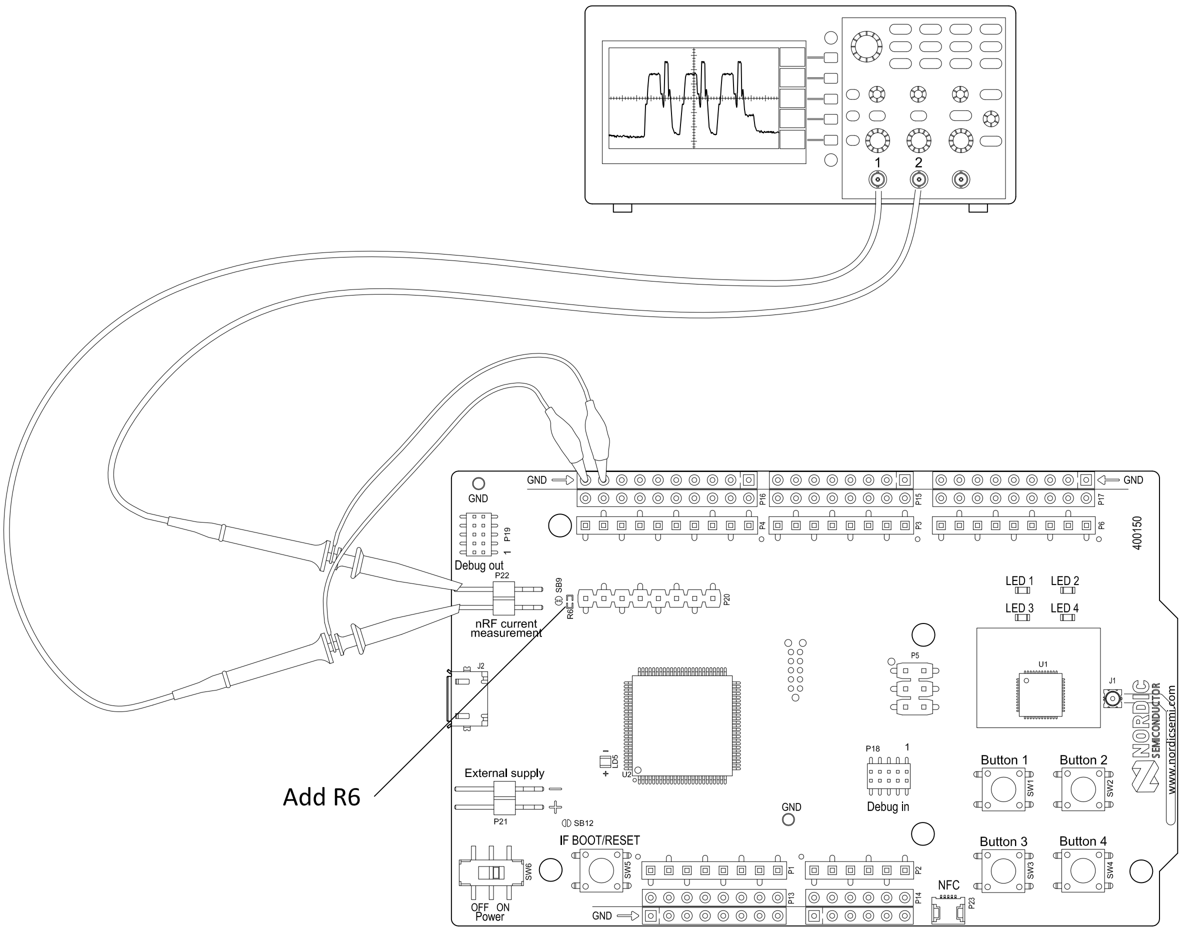


Figure 19: Current measurement with oscilloscope

6.7.3 Using an ampere-meter for current measurement

Follow the instructions below to measure the current using an ampere-meter.

This will monitor the current in series with the nRF device.

1. Make sure you have prepared the development kit as described in Preparing the development kit on page 21.
2. Connect an ampere-meter between the pins of connector **P22** as shown in Figure 20: Current measurement with an ampere-meter on page 24.

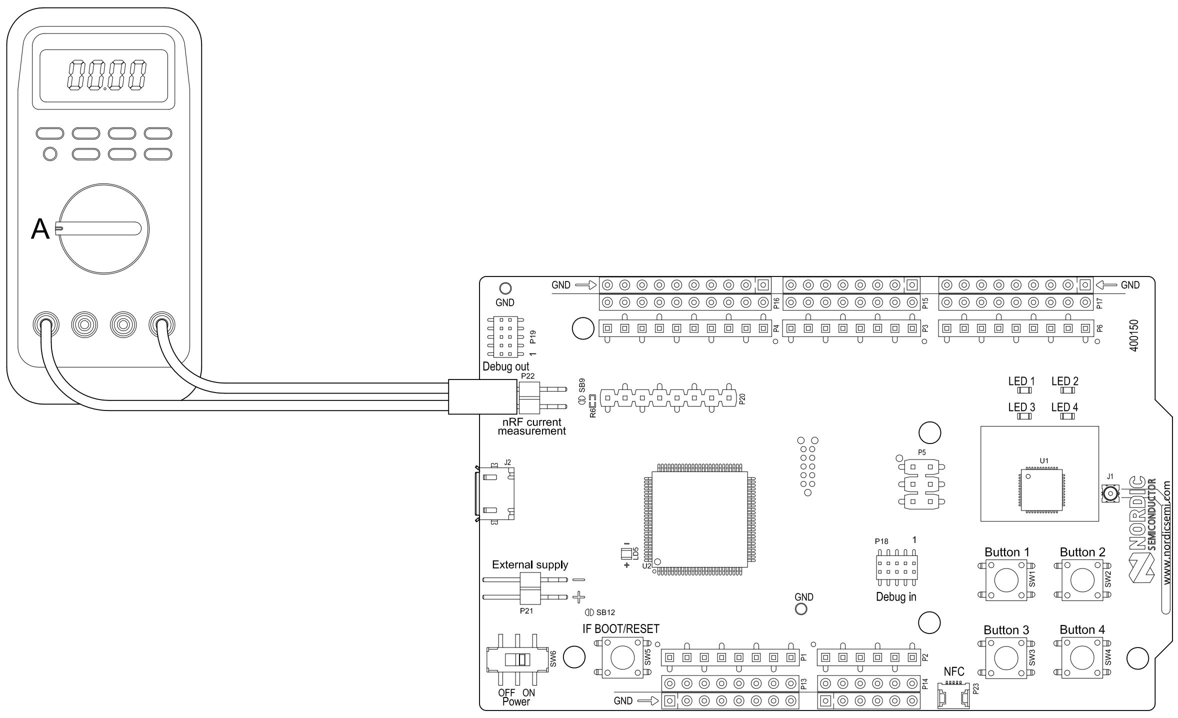


Figure 20: Current measurement with an ampere-meter

Note: An ampere-meter will measure the average current drawn by the nRF52832 if:

- The nRF52832 is in a state where it draws a constant current, or, the activity on the device changing load current, like BLE connection events, is repeated continuously and has a short cycle time (less than 100 ms) so the ampere-meter will average whole load cycles and not parts of the cycle.
- The dynamic range of the ampere-meter is wide enough to give accurate measurements from 1 µA to 15 mA.
- Recommendation: Use true RMS ampere-meter.

6.8 RF measurements

The nRF52 DK is equipped with a small size coaxial connector (**J1**) for conducted measurements of the RF signal.

The connector is of SWF type (Murata part no. MM8130–2600) with an internal switch. By default, when there is no cable attached, the RF signal is routed to the onboard PCB trace antenna.

A test probe is available (Murata part no. MXHS83QE3000) with a standard SMA connection on the other end for connecting instruments (the test probe is not included with the kit). When connecting the test probe, the internal switch in the SWF connector will disconnect the PCB antenna and connect the RF signal from the nRF52832 device to the test probe.

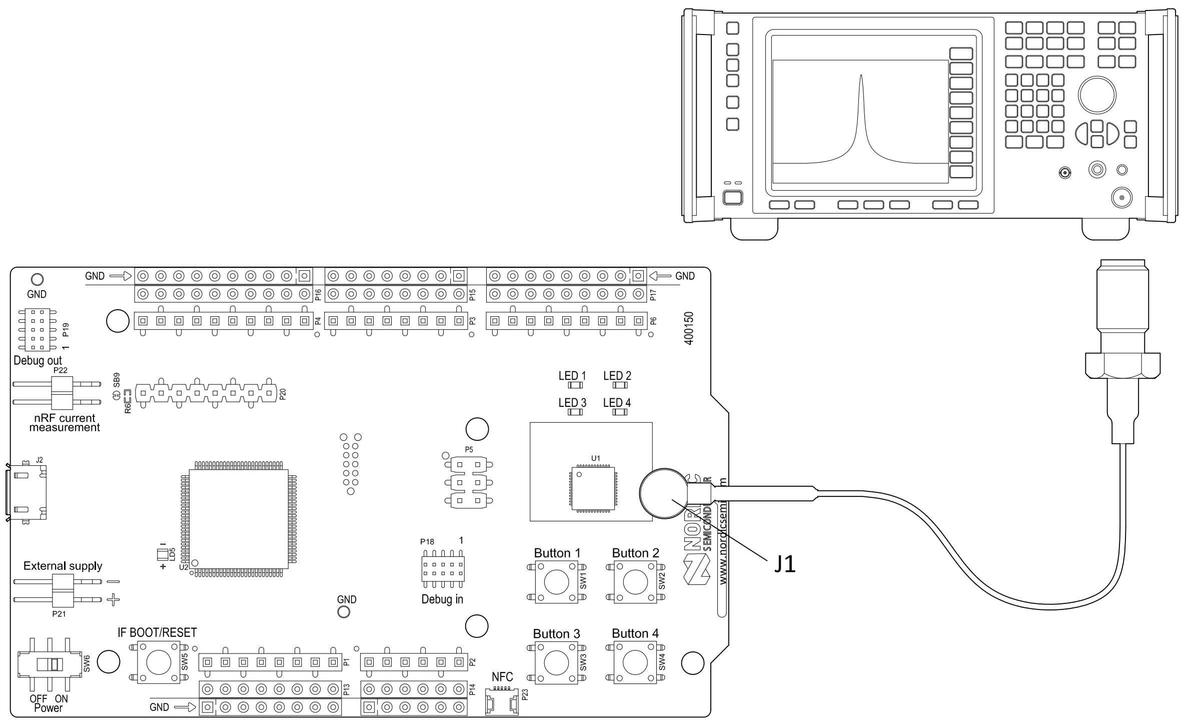


Figure 21: Connecting a spectrum analyzer

The connector and test probe will add loss to the RF signal which should be taken into account when doing measurements, see the following table.

Frequency (MHz)	Loss (dB)
2440	1.0
4880	1.7
7320	2.6

Table 6: Typical loss in connector and test probe

6.9 Debug input

The Debug in connector (**P18**) makes it possible to connect external debuggers for debugging while running on battery or external power supply.

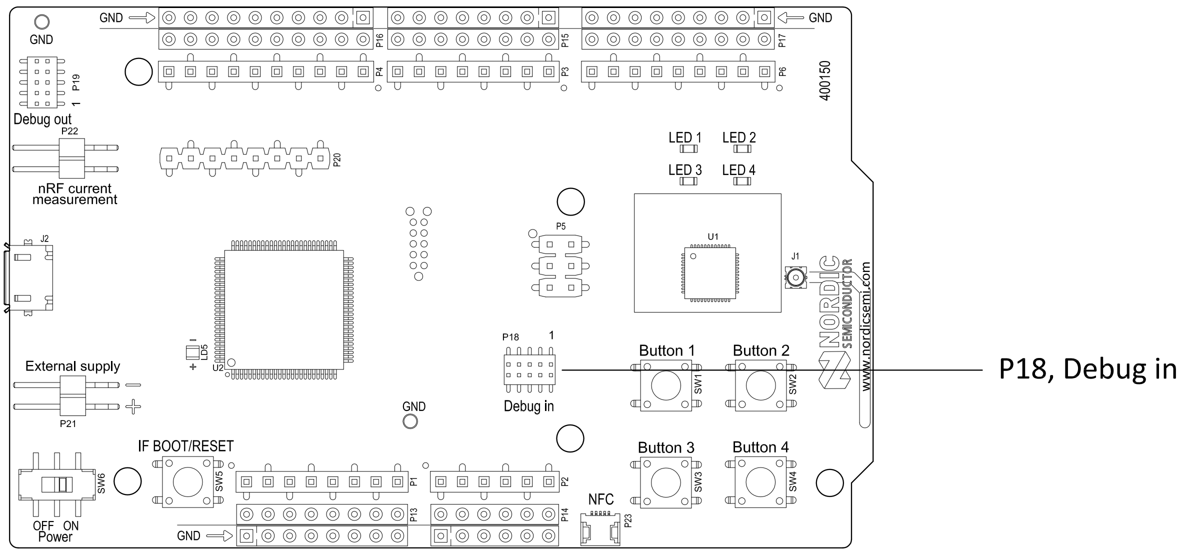


Figure 22: Debug input connector

6.10 Debug output

The nRF52 DK supports programming and debugging nRF51 and nRF52 devices mounted on external boards. To debug an external board with SEGGER J-Link OB IF, connect to the Debug out connector (**P19**) with a 10-pin cable.

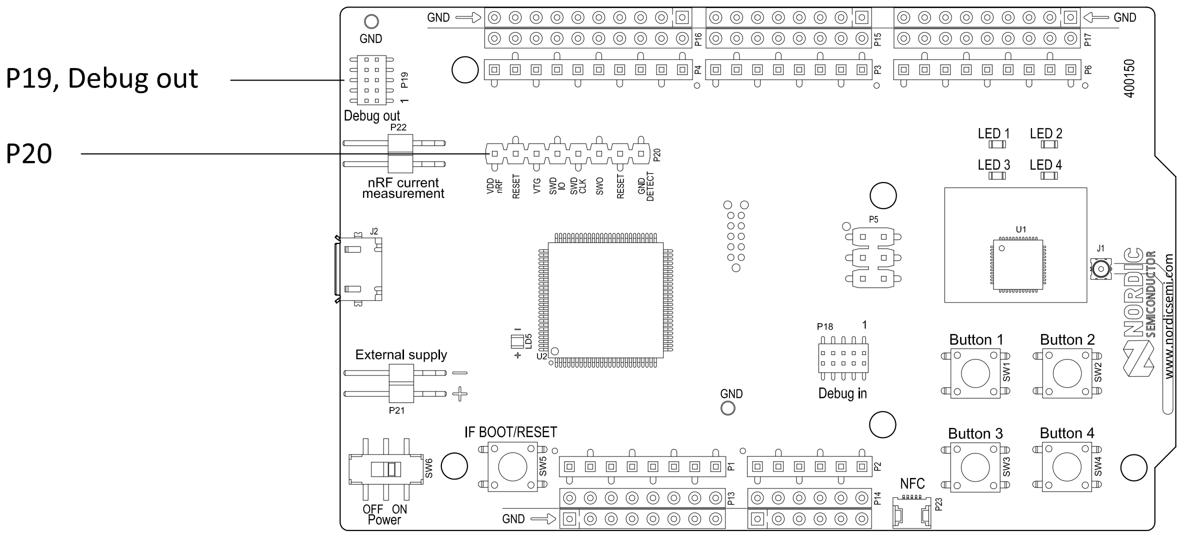


Figure 23: Debug output connector

When the external board is powered, the interface MCU will detect the supply voltage of the board and program/debug the target chip on the external board instead of the onboard nRF52832.

Note: The voltage supported by external debugging/programming is 3.0 V.

You can also use **P20** as a debug out connection to program shield mounted targets. For the Debug out header (**P19**), the interface MCU will detect the supply voltage on the mounted shield and program/debug the shield target.

If the interface MCU detects target power on both **P19** and **P20**, it will by default program/debug the target connected to **P19**.

6.11 NFC antenna interface

The nRF52 DK supports a Near Field Communication (NFC) tag.

NFC-A listen mode operation is supported on nRF52832. The NFC antenna input is available on connector **P23** on the nRF52 DK.

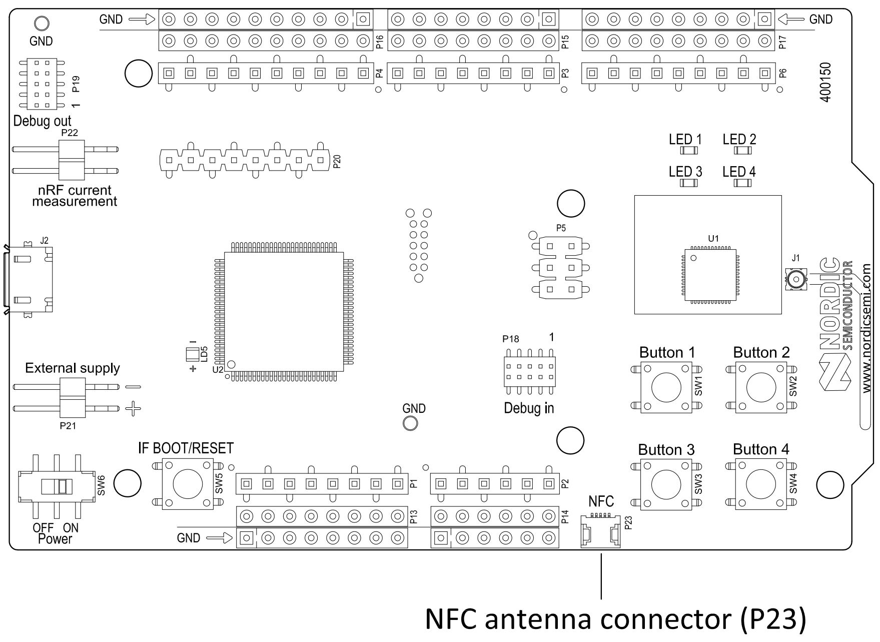


Figure 24: NFC antenna connector

NFC uses two pins, pin 11 (**NFC1**) and pin 12 (**NFC2**) to connect the antenna. These pins are shared with GPIOs (**P0.09** and **P0.10**) and the PROTECT field in the NFCPINS register in UICR defines the usage of these pins and their protection level against abnormal voltages. The content of the NFCPINS register is reloaded at every reset.

Note: The NFC pins are enabled by default.

NFC can be disabled and GPIOs enabled by defining the CONFIG_NFCT_PINS_AS_GPIOS variable in the project settings. The way of doing this depends on the IDE/toolchain in use:

- When using SEGGER Embedded Studio, go to **Project** > **Edit Options** > **Code** > **Preprocessor** > **Preprocessor Definitions** and add the CONFIG_NFCT_PINS_AS_GPIOS variable.
- When using Keil, go to **Project** > **Options for Target** > **C/C++** > **Preprocessor Symbols** > **Define** and add the CONFIG_NFCT_PINS_AS_GPIOS variable.

This functionality can be removed by doing a `nRFjprog --recover`.

Pin 11 and pin 12 are by default configured to use the NFC antenna, but if pin 11 and pin 12 are needed as normal GPIOs, **R25** and **R26** must be NC and **R27** and **R28** must be shorted by 0R.

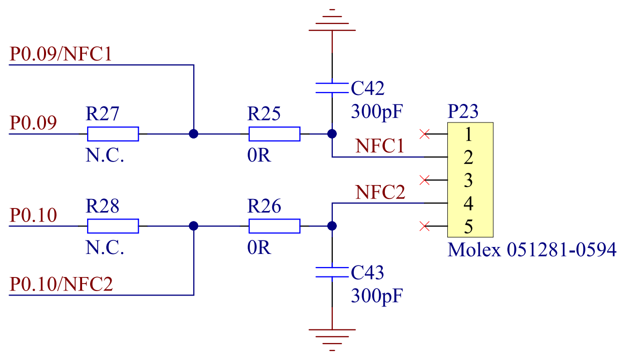


Figure 25: NFC input schematic

6.12 Solder bridge configuration

The following tables show a complete overview of the solder bridges on the nRF52 DK.

Solder bridge	Default	Function
SB1	Closed	Cut to disconnect the 32.768kHz on P0.01.
SB2	Closed	Cut to disconnect the 32.768kHz on P0.00.
SB3	Open	Short to enable P0.01 as normal GPIO.
SB4	Open	Short to enable P0.00 as normal GPIO.
SB5	Closed	Cut to disconnect LED1.
SB6	Closed	Cut to disconnect LED2.
SB7	Closed	Cut to disconnect LED3.
SB8	Closed	Cut to disconnect LED4.
SB9	Closed	Cut for current measurements.
SB10	Open	Short to bypass the reverse voltage protection diode on the USB power.
SB11	Open	Short to bypass the reverse voltage protection diode on the coin-cell battery power.
SB12	Open	Short to bypass the reverse voltage protection diode on the external supply power.
SB13	Closed	Cut to disconnect P0.05 from the connector interface.
SB14	Closed	Cut to disconnect P0.06 from the connector interface.
SB15	Closed	Cut to disconnect P0.07 from the connector interface.
SB16	Closed	Cut to disconnect P0.08 from the connector interface.
SB17	Open	Short to connect P0.21 to the connector interface RESET.
SB18	Open	Short to permanently enable the I/O expander.
SB19	Closed	Cut to permanently disable the I/O expander.
SB20	Closed	Cut to isolate SWDIO from nRF52832 to the interface MCU.
SB21	Closed	Cut to isolate SWDCLK from nRF52832 to the interface MCU.
SB22	Closed	Cut to isolate P0.05 from nRF52832 to the interface MCU.
SB23	Closed	Cut to isolate P0.06 from nRF52832 to the interface MCU.
SB24	Closed	Cut to isolate P0.07 from nRF52832 to the interface MCU.
SB25	Closed	Cut to isolate P0.08 from nRF52832 to the interface MCU.

Table 7: Solder bridge configuration for nRF52 DK (all versions)

Solder bridge	Default	Function
SB26	Closed	Cut to isolate P0.18 from nRF52832 to the interface MCU.
SB27	Closed	Cut to isolate P0.21 from nRF52832 to the interface MCU.
SB28	Open	Short to reset the interface MCU.
SB29	Closed	Cut to disable power for interface MCU.
SB30	Closed	Cut to isolate P0.17 from I/O expander interrupt line.

Table 8: Solder bridge configuration for nRF52 DK v1.0.0 and later

Legal notices