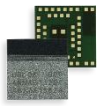
# Using the public IEEE device address from UICR

## ANNA-B1, ANNA-B40, BMD-3, NINA-B1, NINA-B30, NINA-B40, NORA-B1

Application note







### Abstract

Incorporate the unique, public IEEE device address into application code running on a Nordic Semiconductor nRF52-based open CPU module.

# Document information

| | | |
|---|---|---|
| **Title** | **Using the public IEEE device address from UICR** | |
| **Subtitle** | ANNA-B1, ANNA-B40, BMD-3, NINA-B1, NINA-B30, NINA-B40, NORA-B1 | |
| **Document type** | Application note | |
| **Document number** | UBX-19055303 | |
| **Revision and date** | R04 | 16-Nov-2021 |
| **Disclosure restriction** | C1-Public | |

This document applies to the following products:

| **Product name** | |
|---|---|
| ANNA-B112 | Open CPU types |
| ANNA-B40 | |
| BMD-3 | |
| NINA-B1 | Open CPU types |
| NINA-B30 | |
| NINA-B40 | |
| NORA-B1 | |

# Contents

# 1 Overview

There are two types of Bluetooth device addresses, public and random. The random category is further divided into static random and private random. This application note describes a method of utilizing the public device address provided with u-blox modules based on the Nordic Semiconductor nRF5 family of ICs. See the Bluetooth Core Specification [1], Vol 6, Part B.

## 1.1 Public device address

A public device address is a unique combination of a company ID called the MAC Address Block Large (MA-L) and a company-assigned ID. Public device addresses are resolvable. The MA-L is provided by the Institute of Electrical and Electronics Engineers (IEEE) and consists of the first six hex digits of the device address. The remaining six digits are a unique serial number for each device.


**Figure 1: Public device address**

u-blox products are provided with public device addresses. The u-blox MA-L can be D4:CA:6E, CC:F9:57, 60:09:C3, or 6C:1D:EB. Legacy BMD-3 modules are assigned the MA-L of 94:54:93.

## 1.2 Random device address

There are two types of random addresses: static and private. Static addresses are assigned at the factory or can be randomly assigned upon a power cycle. This value cannot change except at a power cycle.

For u-blox modules based on Nordic Semiconductor SoCs, nRF5 SDK examples default to using a static random device address that is stored in the DEVICEADDR[0..1] registers of the Factory Information Configuration Register (FICR) bank. nRF Connect SDK (NCS) addresses default to a randomly generated address at power-up. This address begins with the two most significant bits of 1:1.

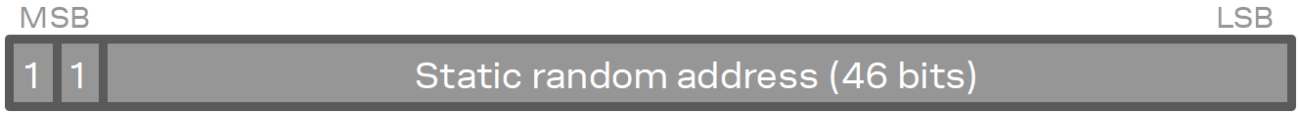



**Figure 2: Static random device address**

The private random address can be combined with the hash of an Identity Resolving Key (IRK), defined in the Bluetooth Core Specification, v4.2 and later, to create a resolvable address for the device. Private random addresses begin with the two most significant bits of 0:1.


**Figure 3: Resolvable private random address**

A non-resolvable random address has the two most significant bits of 0:0.


**Figure 4: Non-resolvable private random device address**

# 2 Device address factory programming

Device addresses are programmed at the factory by u-blox and include an IEEE MAC Address Block Large (MA-L) assignment within its first three least-significant bytes. The three most-significant bytes include a unique assignment for an individual device. This address is stored in the User Information Configuration Registers (UICR): CUSTOMER[0..1] for all modules, and additionally in OTP[0..1] for NORA-B1.

## 2.1 ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, NINA-B4

ANNA-B1, NINA-B1, NINA-B4, and NINA-B4 series modules store the device address starting with the IEEE MA-L at register CUSTOMER[0], byte 0.

| UICR register | Address | Description | Remarks |
|---|---|---|---|
| CUSTOMER[0] | 0x10001080 | Bluetooth_addr [5] (0xCC) | Address bytes [5..3] are one of the following: D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB |
| CUSTOMER[0] | 0x10001081 | Bluetooth_addr [4] (0xF9) | |
| CUSTOMER[0] | 0x10001082 | Bluetooth_addr [3] (0x57) | |
| CUSTOMER[0] | 0x10001083 | Bluetooth_addr [2] (0x11) | Address bytes [2..0] example value |
| CUSTOMER[1] | 0x10001084 | Bluetooth_addr [1] (0x22) | |
| CUSTOMER[1] | 0x10001085 | Bluetooth_addr [0] (0x33) | |
| CUSTOMER[1] | 0x10001086 | 0xFF | Unused bytes |
| CUSTOMER[1] | 0x10001087 | 0xFF | |

**Table 1: Device address assignment for ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, and NINA-B4**

## 2.2 BMD-3

BMD-3 series modules store the device address starting with the IEEE MA-L at register CUSTOMER[1], byte 2[1].

| UICR register | Address | Description | Remarks |
|---|---|---|---|
| CUSTOMER[0] | 0x10001080 | Bluetooth_addr [0] (0x33) | Address bytes [2..0] example value |
| CUSTOMER[0] | 0x10001081 | Bluetooth_addr [1] (0x22) | |
| CUSTOMER[0] | 0x10001082 | Bluetooth_addr [2] (0x11) | |
| CUSTOMER[0] | 0x10001083 | Bluetooth_addr [3] (0x57) | Address bytes [5..3] are one of the following: D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB (standard u-blox), or 94:54:93 (legacy BMD) |
| CUSTOMER[1] | 0x10001084 | Bluetooth_addr [4] (0xF9) | |
| CUSTOMER[1] | 0x10001085 | Bluetooth_addr [5] (0xCC) | |
| CUSTOMER[1] | 0x10001086 | 0xFF | Unused bytes |
| CUSTOMER[1] | 0x10001087 | 0xFF | |

**Table 2: Device address assignment for BMD-3**

## 2.3 NORA-B1

NORA-B1 series modules store the device address starting with the IEEE MA-L at two locations: application core OTP[0], byte 0 and network core CUSTOMER[0], byte 0.

| UICR Register in application core | Address | Description | Remarks |
|---|---|---|---|
| OTP[0] | 0x00FF8100 | Bluetooth_addr [5] (0xCC) | Address bytes [5..3] are one of the following: D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB |
| OTP[0] | 0x00FF8101 | Bluetooth_addr [4] (0xF9) | |

[1] The BMD-3 series legacy byte order is retained to accommodate existing designs.

| UICR Register in application core | Address | Description | Remarks |
|---|---|---|---|
| OTP[0] | 0x00FF8102 | Bluetooth_addr [3] (0x57) | |
| OTP[0] | 0x00FF8103 | Bluetooth_addr [2] (0x11) | Address bytes [2..0] example value |
| OTP[1] | 0x00FF8104 | Bluetooth_addr [1] (0x22) | |
| OTP[1] | 0x00FF8105 | Bluetooth_addr [0] (0x33) | |
| OTP[1] | 0x00FF8106 | 0xFF | Unused |
| OTP[1] | 0x00FF8107 | 0xFF | Unused |

**Table 3: Device address assignment for NORA-B1 application core**

| UICR Register in network core | Address | Description | Remarks |
|---|---|---|---|
| CUSTOMER[0] | 0x01FF8300 | Bluetooth_addr [5] (0xCC) | Address bytes [5..3] are one of the following: D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB |
| CUSTOMER[0] | 0x01FF8301 | Bluetooth_addr [4] (0xF9) | |
| CUSTOMER[0] | 0x01FF8302 | Bluetooth_addr [3] (0x57) | |
| CUSTOMER[0] | 0x01FF8303 | Bluetooth_addr [2] (0x11) | Address bytes [2..0] example value |
| CUSTOMER[1] | 0x01FF8304 | Bluetooth_addr [1] (0x22) | |
| CUSTOMER[1] | 0x01FF8305 | Bluetooth_addr [0] (0x33) | |
| CUSTOMER[1] | 0x01FF8306 | 0xFF | Unused |
| CUSTOMER[1] | 0x01FF8307 | 0xFF | Unused |

**Table 4: : Device address assignment for NORA-B1 network core**

## 2.4 Module labels

The Bluetooth device address is contained within the 2D data matrix printed on the module label.


**Figure 5: Module label with data matrix**

See each module data sheet for the data matrix printing details.

A common 2D barcode scanner with which to read the module label is the Honeywell Xenon 1900g [7].

# 3 Bluetooth device address restoration

If the public device address is erased because of a chip erase or recovery, the address can be restored using command line tools like the J-Link commander [6]or a binary editing utility such as HxD [2]. The address can also be restored in an Integrated Development Environment (IDE) using SEGGER Embedded Studio (SES) [5].

☞ Only zero bits may be written to the User Information Configuration Register (UICR). Writing a "1" to bits already set to 0 will result in a failure. Before writing to the UICR, a full chip erase or recovery is needed to restore the register to 0xFFFFFFFF, as described in the following sections.

## 3.1 File preparation

The device address for ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, NINA-B4, and NORA-B1 modules is stored in the UICR using a different notation than that used for BMD-3 modules.

### 3.1.1 ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, NINA-B4, NORA-B1 preparation

For ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, NINA-B4, and NORA-B1, the device address is stored with the most significant byte (MSB) at location zero. For example, the device address CC:F9:57:11:22:33 is given in the following byte order:

```
CC F9 57 11 22 33 FF FF
```

The last two bytes are unused and set to `0xFF`.

Figure 6 shows how a binary file containing the device for ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, and NINA-B4 is edited using a binary editing utility like HxD [2].

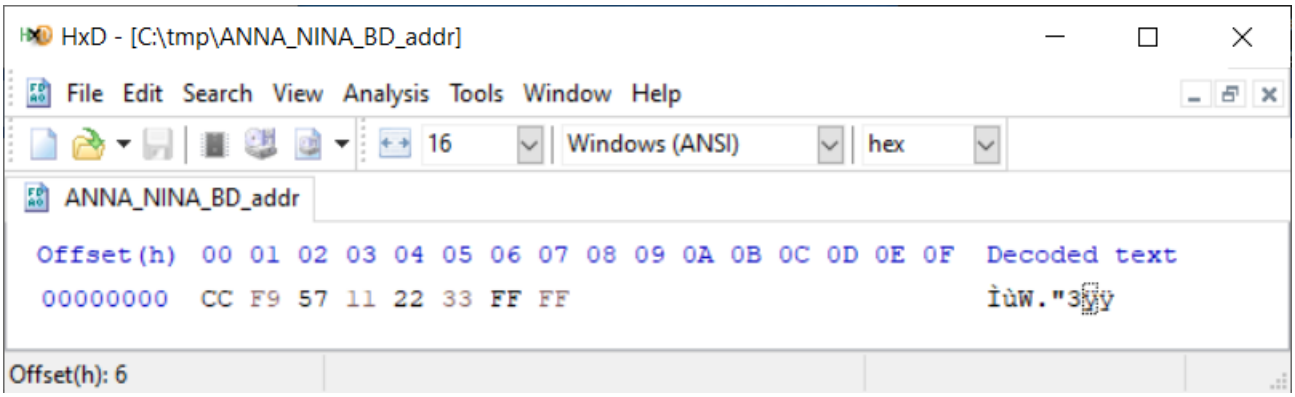



**Figure 6: Editing a binary file in HxD for ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, and NINA-B4**

Notice that the device address is in MSB (most significant byte) first order. The two bytes of 0xFF are fillers, because reads and writes to the UICR need to take place in 4-byte increments.

Save this with a convenient file name (e.g., `ANNA_NINA_BD.bin`) and location.

### 3.1.2 BMD-3 preparation

To maintain compatibility with legacy designs, the order of the address byte for BMD-3 modules is given in LSB notation, i.e., in reverse order to the MSB address byte for ANNA and NINA modules:

```
33 22 11 57 F9 CC FF FF
```

As with the ANNA and NINA modules, the last two bytes of the device address for BMD-3 modules are unused and set to `0xFF`. Figure 7 shows how a binary file containing the device address for BMD-3 is edited to the Bluetooth device address on the module label, using a binary editing utility like HxD [2].

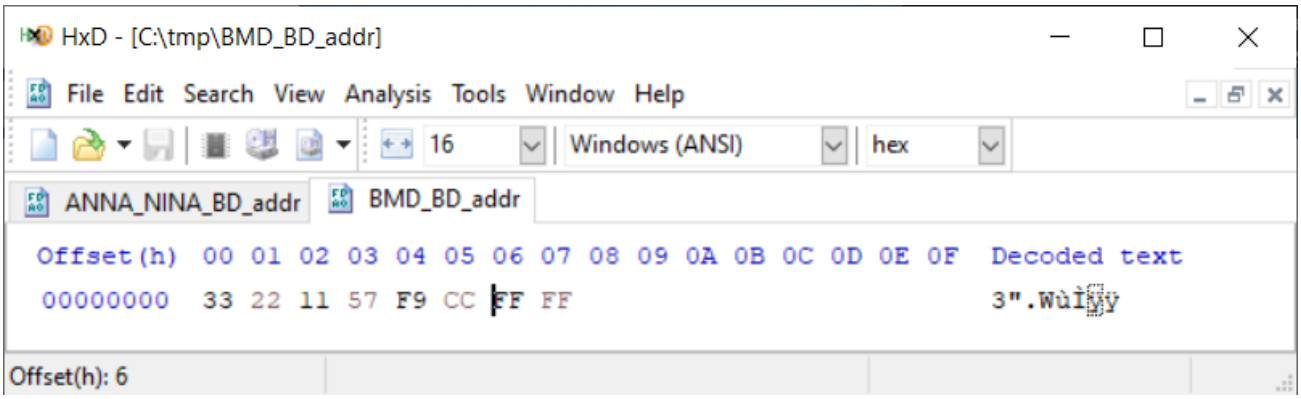



**Figure 7: Editing a binary file in HxD for BMD-3**

Save this with a convenient file name (e.g., `BMD_BD.bin`) and location.

## 3.2 Restoring the address

The device address can be restored from the `ANNA_NINI_BD.bin` or `BMD_BD.bin` file saved in one of the previous sections from within SEGGER Embedded Studio (SES) or the SEGGER J-Link Commander [6].

### 3.2.1 Restoring the address within an IDE

The device address can be restored from within SES (and other IDEs) by loading the `.bin` file through the target connection.

1. Open a suitable project in SES.
   For NORA-B1, SES connects to the core specified when a project is opened. For information about opening a project for either the application or network core, see also the EVK-NORA-B1 user guide [8]. For other modules, start SES and open a suitable project, for example:
   `<sdk_install_location>\nRF5_SDK_17.0.2_d674dde\examples\ble_peripheral\ble_app_blinky`
2. Connect to the target J-Link interface with <Ctrl+T>, C.
3. Select “Download File” then “Download Binary File…”, as shown in Figure 8.

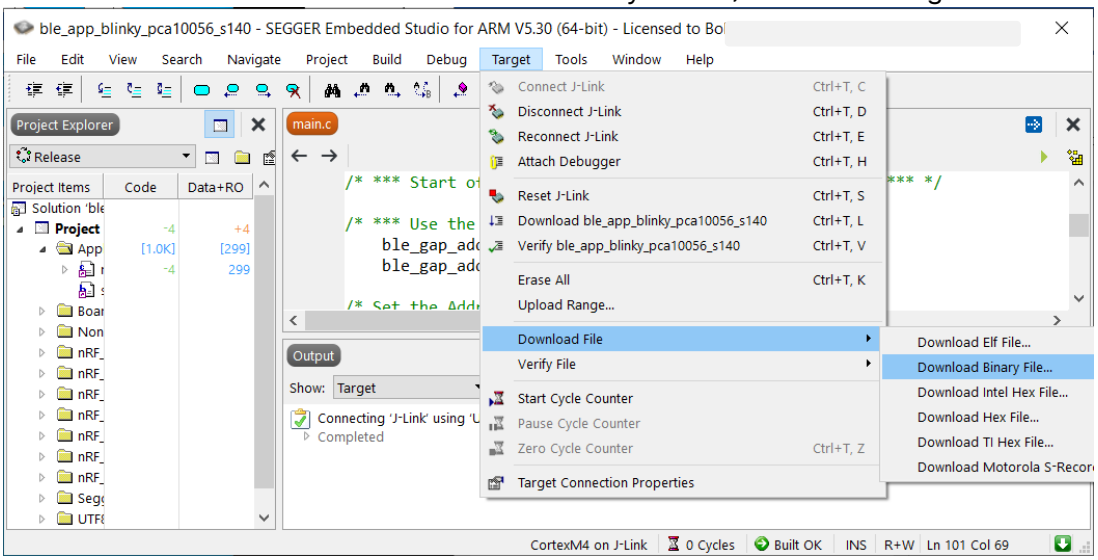



**Figure 8: Download device address file with public Bluetooth device address**

4. Select the `.bin` file saved in the previous section.
5. Enter the UICR address:
   - For ANNA-B1, ANNA-B40, BMD-3, NINA-B1, NINA-B4 or NINA-B4, enter the UICR address `0x10001080`.
   - For NORA-B1 application core, enter the UICR address `0x00FF8100`.
   - For NORA-B1 network core, enter the UICR address `0x01FF8300`.

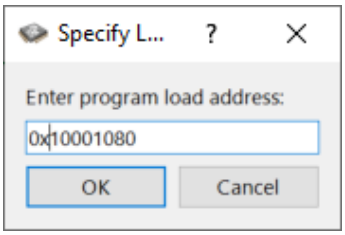



**Figure 9: UICR load address of public Bluetooth device address**

### 3.2.2 Restoring the address using command-based utilities

The device address can also be restored from the `.bin` file from within SEGGER J-Link Commander.

When programming end-product application code, the following J-Link commands may be used within a script to automate the saving and restoration of the device address.

For ANNA-B1, ANNA-B40, BMD-3, NINA-B1, NINA-B3, and NINA-B4 modules:

1. Save the Bluetooth device address from `UICR + 0x80`. Memory reads must be in 4-byte increments.
   ```
   J-Link>savebin address_file_name.bin 0x10001080 8
   ```
2. Restore the Bluetooth device address from the file and program to `UICR + 0x80`.
   ```
   J-Link>loadbin address_file_name.bin 0x10001080
   ```

For NORA-B1 modules, the address depends on the connected core.

For the application core:

1. Save the Bluetooth device address from `UICR + 0x100`. Memory reads must be in 4-byte increments.
   ```
   J-Link>savebin address_file_name.bin 0x00FF8100 8
   ```
2. Restore the Bluetooth device address from the file and program to `UICR + 0x100`.
   ```
   J-Link>loadbin address_file_name.bin 0x00FF8100
   ```

For the network core:

1. Save the Bluetooth device address from `UICR + 0x300`. Memory reads must be in 4-byte increments.
   ```
   J-Link>savebin address_file_name.bin 0x01FF8300 8
   ```
2. Restore the Bluetooth device address from the file and program to `UICR + 0x300`.
   ```
   J-Link>loadbin address_file_name.bin 0x01FF8300
   ```

# 4 Application code

## 4.1 nRF5 SDK

nRF5 SDK supports ANNA-B1, ANNA-B40, BMD-3, NINA-B1, NINA-B30, and NINA-B40.

By default, Nordic Semiconductor SDK examples use a static random MAC address located in the FICR. To use the public device address, modifications to the source code are required.

If an ANNA-B1, ANNA-B40, NINA-B1, NINA-B3, or NINA-B4 module is used, add the following directive near the beginning of `main.c`:

```
/* *** Start of added code – first block for ANNA and NINA only *** */

#define ANNANINA

/* *** End of added code – first block for ANNA and NINA only *** */
```

For all modules, locate the `gap_params_init()` function and add lines noted between the **bold** comments below.

```
/**@brief Function for the GAP initialization.
 *
 * @details This function will set up all the necessary GAP (Generic Access Profile)
 * parameters of the device. It also sets the permissions and appearance.
 */
static void gap_params_init(void)
{
    uint32_t err_code;
    ble_gap_conn_params_t gap_conn_params;
    ble_gap_conn_sec_mode_t sec_mode;

BLE_GAP_CONN_SEC_MODE_SET_OPEN(&sec_mode);

/* *** Start of added code – second block for all modules *** */

/* *** Use the u-blox IEEE public device address *** */
    ble_gap_addr_t addr;
    ble_gap_addr_t tempaddr;

/* Set the Address type to Public */
    addr.addr_type =     BLE_GAP_ADDR_TYPE_PUBLIC;
    tempaddr.addr_type = BLE_GAP_ADDR_TYPE_PUBLIC;

/* copy the u-blox address from the UICR */
    memcpy(tempaddr.addr, (uint8_t*)0x10001080UL, BLE_GAP_ADDR_LEN);

/* Reverse byte order for ANNA-B1, NINA-B1, NINA-B3, NINA-B4 */
#ifdef ANNANINA
    for (int i = 0; i <= 5; i++){
      addr.addr[i] = tempaddr.addr[5-i];
    }
#else /* for BMD-3, the bytes are in the correct order */
    for (int i = 0; i <= 5; i++){
      addr.addr[i] = tempaddr.addr[i];
    }
#endif /* ANNANINA */

/* Tell the SoftDevice to use the u-blox address */
    err_code = sd_ble_gap_addr_set(&addr);
    APP_ERROR_CHECK(err_code);}
```

```
/* *** End of added code – second block for all modules *** */

    err_code = sd_ble_gap_device_name_set(&sec_mode, (const uint8_t *) DEVICE_NAME,
                                          strlen(DEVICE_NAME));
    APP_ERROR_CHECK(err_code);

    memset(&gap_conn_params, 0, sizeof(gap_conn_params));

    gap_conn_params.min_conn_interval = MIN_CONN_INTERVAL;
    gap_conn_params.max_conn_interval = MAX_CONN_INTERVAL;
    gap_conn_params.slave_latency = SLAVE_LATENCY;
    gap_conn_params.conn_sup_timeout = CONN_SUP_TIMEOUT;

    err_code = sd_ble_gap_ppcp_set(&gap_conn_params);
    APP_ERROR_CHECK(err_code);

}
```

## 4.2 nRF Connect SDK – nRF52 single core

nRF Connect SDK (NCS) supports ANNA-B112, ANNA-B40, BMD-3, NINA-B1, NINA-B30, and NINA-B40 single-core nRF52-based modules. For NORA-B1 support, see nRF Connect SDK – nRF53 dual core. Changes to the project configuration file, `prj.conf`, are not necessary.

The function `bt_ctlr_set_public_addr()` is used to assign the Bluetooth device address. Prior to calling `bt_ctlr_set_public_addr()`, the device address must be read from the UICR. Three sections of code need added to an application.

Near the top of `main.c`, insert the following lines:

```
/* *** Start of added code – first block *** */

/* bt_ctlr_set_public_addr() is defined in controller.h */
#include <bluetooth/controller.h>

/* Uncomment the following line For ANNA and NINA modules only */
//#define ANNANINA


/* *** End of added code – first block *** */
```

Near the top of `void main(void)`, insert the following lines:

```
/* *** Start of added code – second block *** */

        uint8_t addr[6];
/* Read Bluetooth device address from UICR */
#ifdef ANNANINA /* UICR byte order is reversed for ANNA and NINA */
        addr[0] = ((NRF_UICR->CUSTOMER[1] & 0x0000ff00) >> 8);
        addr[1] = ((NRF_UICR->CUSTOMER[1] & 0x000000ff));
        addr[2] = ((NRF_UICR->CUSTOMER[0] & 0xff000000) >> 24);
        addr[3] = ((NRF_UICR->CUSTOMER[0] & 0x00ff0000) >> 16);
        addr[4] = ((NRF_UICR->CUSTOMER[0] & 0x0000ff00) >> 8);
        addr[5] = ((NRF_UICR->CUSTOMER[0] & 0x000000ff));
#else
        addr[5] = ((NRF_UICR->CUSTOMER[1] & 0x0000ff00) >> 8);
        addr[4] = ((NRF_UICR->CUSTOMER[1] & 0x000000ff));
        addr[3] = ((NRF_UICR->CUSTOMER[0] & 0xff000000) >> 24);
        addr[2] = ((NRF_UICR->CUSTOMER[0] & 0x00ff0000) >> 16);
        addr[1] = ((NRF_UICR->CUSTOMER[0] & 0x0000ff00) >> 8);
        addr[0] = ((NRF_UICR->CUSTOMER[0] & 0x000000ff));
#endif /* ANNANINA */

/* *** End of added code – second block *** */
```

Prior to enabling the Bluetooth controller with `bt_enable()`, insert the call to set the address:

```
/* *** Start of added code – third block *** */

        bt_ctlr_set_public_addr(addr);

/* *** End of added code – third block *** */
```

## 4.3 nRF Connect SDK – nRF53 dual core

nRF Connect SDK (NCS) also supports NORA-B1 dual-core nRF53-based modules. Example applications (samples) provided by NCS use the Nordic Semiconductor Bluetooth SoftDevice Controller `hci_rpmsg` running on the network core. Calls must be passed from the application core to the network core. The SoftDevice controller `hci_rpmsg` is used without modification.

Follow the procedure described in the EVK user guide [8] to prepare the EVK, compile, and load `hci_rpmsg` onto the network core.

The application code for the application core requires additional settings to enable vendor specific HCI remote procedure messages to pass the new address information. Edit the `prj.conf` project overlay file for the application core to include the following lines anywhere in the file:

```
/* *** Start of added code – prj.conf *** */

# Enable vendor specific HCI extensions
CONFIG_BT_HCI_VS=y
CONFIG_BT_HCI_VS_EXT=y

/* *** End of added code – prj.conf *** */
```

☞ Be sure to edit `prj.conf` prior to opening the application core project in SES.

Near the top of `main.c`, add the following lines:

```
/* *** Start of added code – first block *** */

#include <bluetooth/hci_vs.h>

/* *** End of added code – first block *** */
```

Add the following code to the application `main.c` file, before `void main(void)`:

```
/* *** Start of added code – second block *** */

void ble_set_bd_addr(bt_addr_t* addr) {
   struct net_buf *buf;
   int err;

   buf = bt_hci_cmd_create(BT_HCI_OP_VS_WRITE_BD_ADDR, sizeof(*addr));
   if(!buf) {
      printk("No RPC command buffers available\n");
   }

   net_buf_add_mem(buf, addr, sizeof(*addr));

   err = bt_hci_cmd_send_sync(BT_HCI_OP_VS_WRITE_BD_ADDR, buf, NULL);
   if(err) {
      printk("Device address cannot be set (err %d)\n", err);
   }
}

/* *** End of added code – second block *** */
```

To use the vendor specific HCI comands, Bluetooth needs to be enabled first. Within `void main(void)`, add the code noted below just after `bt_enable()` and following error check:

```
   err = bt_enable(NULL);
   if (err) {
      printk("Bluetooth init failed (err %d)\n", err);
      return;
   }

   printk("Bluetooth initialized\n");

/* *** Start of added code – third block *** */

        uint8_t pub_addr[6];

        pub_addr[0] = ((NRF_UICR->OTP[1] & 0x0000ff00) >> 8);
        pub_addr[1] = ((NRF_UICR->OTP[1] & 0x000000ff));
        pub_addr[2] = ((NRF_UICR->OTP[0] & 0xff000000) >> 24);
        pub_addr[3] = ((NRF_UICR->OTP[0] & 0x00ff0000) >> 16);
        pub_addr[4] = ((NRF_UICR->OTP[0] & 0x0000ff00) >> 8);
        pub_addr[5] = ((NRF_UICR->OTP[0] & 0x000000ff));

        bt_addr_t addr = {{pub_addr[0], pub_addr[1], pub_addr[2],
                           pub_addr[3], pub_addr[4], pub_addr[5]}};
        ble_set_bd_addr(&addr);

/* *** End of added code – third block *** */

   if (IS_ENABLED(CONFIG_SETTINGS)) {
      settings_load();
```

## 4.4 Check the Bluetooth device address

Once the new code is added, compile the application, and run it on an EVK.

Figure 10 shows nRF Connect for Desktop using a BMD-300-EVAL as the scanning device. The modified `Nordic_LBS`[2] example is running on an EVK-NORA-B1 evaluation board.

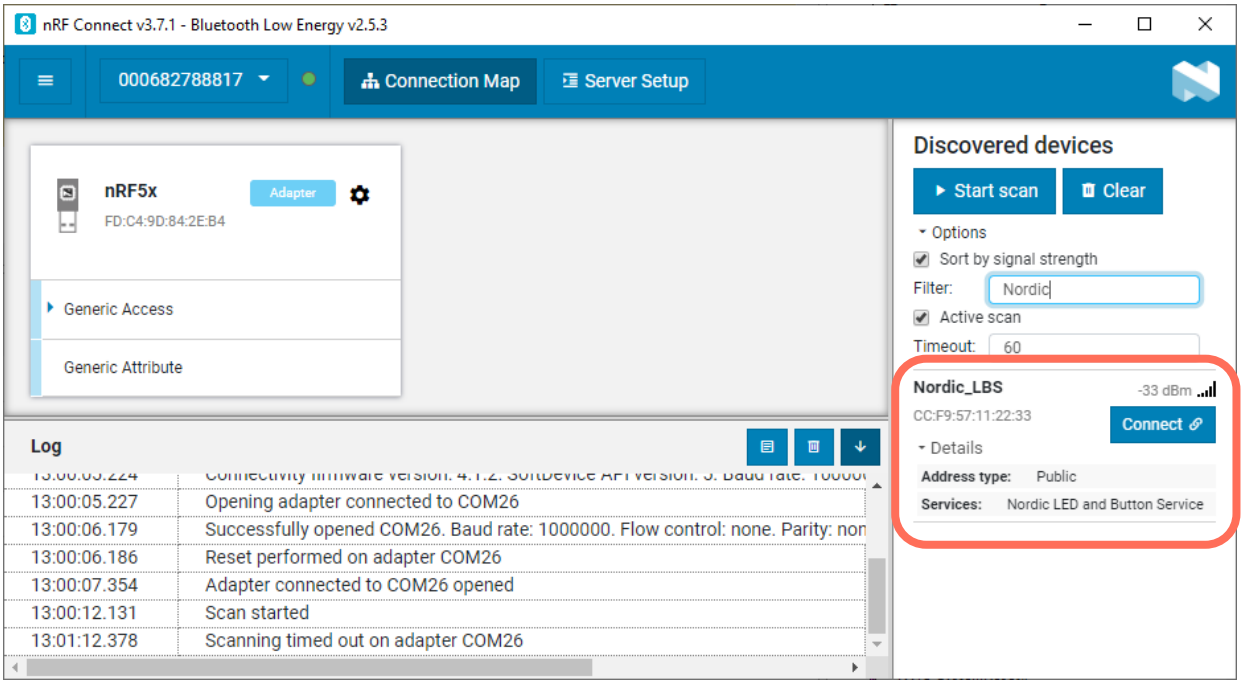



**Figure 10: nRF Connect scanning for Nordic_LBS**

[2] nRF5 SDK default advertising name is “Nordic_Blinky”. nRF Connect SDK default advertising name is “Nordic_LBS”.

# Appendix

# A Glossary

| Abbreviation | Definition |
|---|---|
| CPU | Central Processing Unit |
| FICR | Factory Information Configuration Registers |
| HxD | Hex and binary file editor |
| IDE | Integrated Development Environment |
| IEEE | Institute of Electrical and Electronics Engineers |
| LSB | Least significant bit |
| MAC | Media Access Controller |
| MCU | Microcontroller Unit |
| NCS | Nordic Connect SDK |
| SDK | Software Development Kit |
| SES | SEGGER Embedded Studio |
| SoC | System on Chip: microcontroller with integrated peripherals |
| UICR | User Information Configuration Register |

**Table 5: Explanation of the abbreviations and terms used**

# Related documents

[1] Bluetooth Core Specification
[2] HxD hex and binary file editor
[3] Nordic Semiconductor InfoCenter
[4] Nordic Semiconductor nRF5 SDK
[5] SEGGER Embedded Studio
[6] SEGGER J-Link Commander
[7] Honeywell Xenon 1900g family
[8] EVK-NORA-B1 user guide, UBX-20030319

For regular updates to u-blox documentation and to receive product change notifications, register on our homepage (www.u-blox.com).

# Revision history

| Revision | Date | Name | Comments |
|---|---|---|---|
| R01 | 16-Dec-2019 | brec | Initial release |
| R02 | 13-Jul-2020 | brec | Updated product list on page 2, enhanced information in Table 2 |
| R03 | 17-Feb-2021 | brec | BMD and ANNA / NINA store address differently, updated section 2 to reflect this, added code to section 3 to handle both cases |
| R04 | 16-Nov-2021 | brec | Revised document structure and added new section for nRF5 SDK (NCS), added ANNA-B40 to product scope |

# Contact

For complete contact information, visit us at www.u-blox.com.

**u-blox Offices**


**North, Central and South America**

**u-blox America, Inc.**

Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

**Regional Office West Coast:**

Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

**Technical Support:**

Phone: +1 703 483 3185
E-mail: support_us@u-blox.com

**Headquarters**
**Europe, Middle East, Africa**

**u-blox AG**

Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

**Asia, Australia, Pacific**

**u-blox Singapore Pte. Ltd.**

Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

**Regional Office Australia:**

Phone: +61 3 9566 7255
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

**Regional Office China (Beijing):**

Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Chongqing):**

Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Shanghai):**

Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Shenzhen):**

Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office India:**

Phone: +91 80 405 092 00
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

**Regional Office Japan (Osaka):**

Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

**Regional Office Japan (Tokyo):**

Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

**Regional Office Korea:**

Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

**Regional Office Taiwan:**

Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com