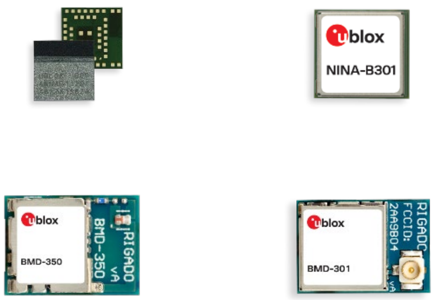
Using the public IEEE address from UICR

ANNA-B1, BMD-3, NINA-B1, NINA-B3

Application note



Abstract

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

www.u-blox.com
UBX-19055303 - R01

Document information

Title	**Using the public IEEE device address from UICR**
Subtitle	ANNA-B1, BMD-3, NINA-B1, NINA-B3
Document type	Application note
Document number	UBX-19055303
Revision and date	R01 16-Dec-2019
Disclosure restriction	

This document applies to the following products:

Product name	Type number
ANNA-B112	All
BMD-300	All
BMD-301	All
BMD-340	All
BMD-341	All
BMD-345	All
BMD-350	All
BMD-380	All
NINA-B111	All
NINA-B112	All
NINA-B301	All
NINA-B302	All
NINA-B306	All

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.

1.1 Public device address

A public device address is a unique combination of a company ID and a company-assigned ID. Public device addresses are resolvable. The company ID 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.

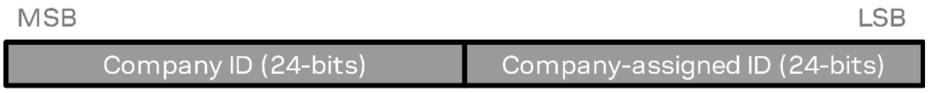


Figure 1: Public device address

u-blox products are provided with public device addresses. The u-blox company ID can be D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB, or 94:54:93 (legacy BMD and R41Z).

1.2 Random device address

Random addresses are generally not resolvable. There are two types: static and private. The static address is assigned at the factory, or can be assigned upon a power cycle.

For u-blox modules based on Nordic Semiconductor SoCs, the 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. 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 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 0:0.

Figure 3: Private random device address

2 Module device address programming

As noted above, device addresses (MAC, Bluetooth) are programmed at the factory by u-blox and have an IEEE company identifier of 94:54:93 or 6C:1D:EB as the first three bytes. The last three bytes are a unique assignment for each device. For the ANNA-B1, BMD-2, BMD-3, NINA-B1, and NINA-B3 series modules, this address is stored in the User Information Configuration Registers (UICR) beginning at 0x10001080.

UICR Register	Address	Description	Remarks
NRF_UICR + 0x80	0x10001080	Bluetooth_addr [0] (0xCC)	Example value
NRF_UICR + 0x81	0x10001081	Bluetooth_addr [1] (0xBB)	Example value
NRF_UICR + 0x82	0x10001082	Bluetooth_addr [2] (0xAA)	Example value
NRF_UICR + 0x83	0x10001083	Bluetooth_addr [3] (0xEB)	Bytes [5..3] are one of the following: D4:CA:6E, CC:F9:57, 60:09:C3, 6C:1D:EB, or 94:54:93 (legacy BMD and R41Z)
NRF_UICR + 0x84	0x10001084	Bluetooth_addr [4] (0x1D)	
NRF_UICR + 0x85	0x10001085	Bluetooth_addr [5] (0x6C)	

Table 1: Bluetooth address

The address is contained within the 2D barcode printed on the module label.

See each module data sheet for the 2D barcode and human readable text details.

A common 2D barcode scanner is the Honeywell Xenon 1900g family.

2.1 Application code

Nordic Semiconductor's examples use a static random MAC address located in the FICR. In order to use the public device address, some modifications to the source code is required.

Locate the `gap_params_init()` function and add lines noted in **bold** 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 *** */

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

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

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

//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 *** */

    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);
}
```

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

Here is nRF Connect using an nRF52840 Dongle as the scanning device. The modified Nordic_Blinky example is running on a BMD-350-Eval EVK board.

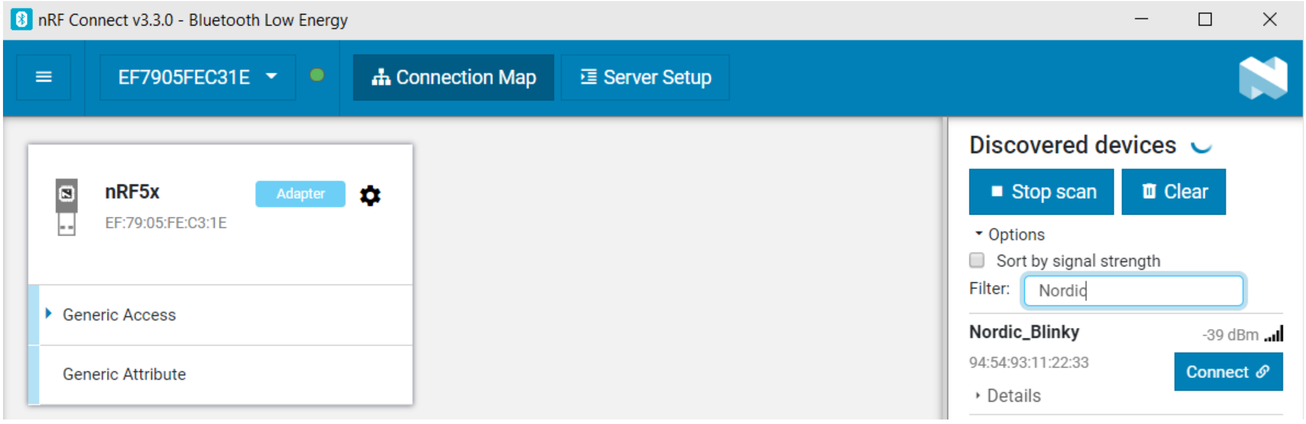


Figure 4: nRF Connect scanning for Nordic_Blinky

2.2 Device address restoration

Depending on where a project is in development, this MAC address may get erased due to a chip erase or recovery. It's easy to restore the address, though. Using a binary editing utility such as HxD, create a file that has 8 bytes in the following format:

```
ZZ YY XX 93 54 94 FF FF
```

For illustration, let's assume the address is 94:54:93:11:22:33 (since "XX:YY:ZZ" are not valid hex digits):

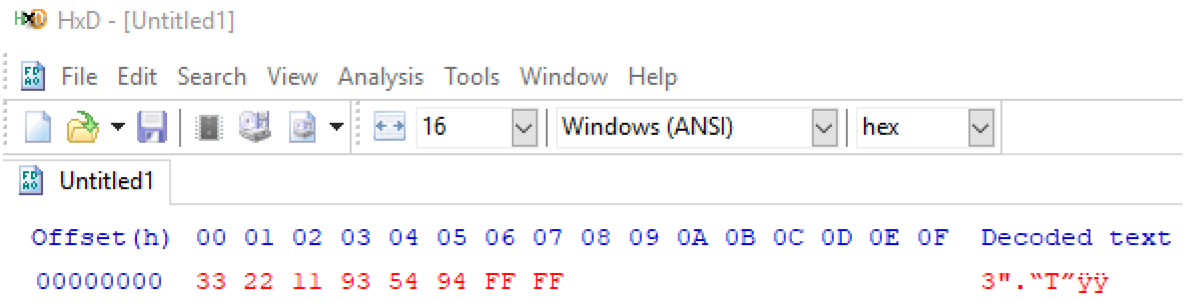


Figure 5: Editing a binary file in HxD

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

Save this file to a convenient name and location. In this case, we'll place this file on the desktop:

`bd_addr.bin`

The address can be restored from this file from within SEGGER Embedded Studio (and other IDEs) by loading the file through the target connection:

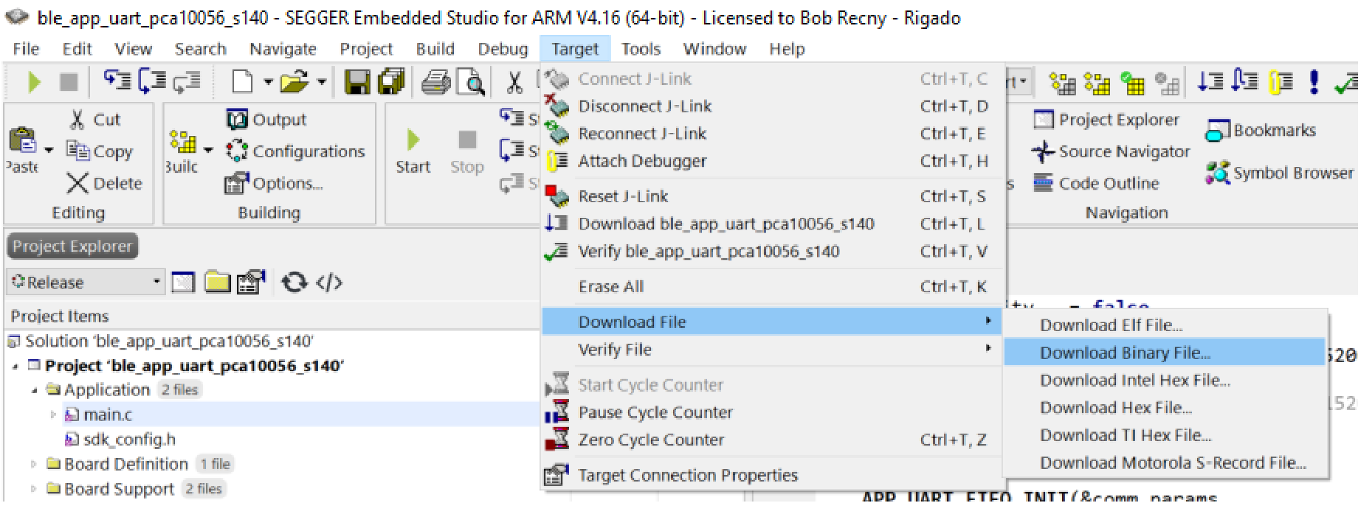


Figure 6: Download bd_addr.bin file with public device address

Select the file `bd_addr.bin`. Next enter the UICR address of `0x10001080`:

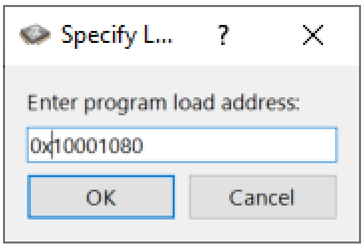


Figure 7: UICR location of public device address

The J-Link commander can also be used. 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:

```
//save the MAC from UICR + 0x80
savebin bd_addr.bin 0x10001080 8 // memory reads must be in 4-byte increments

//restore the MAC from the file and program to UICR + 0x80
loadbin bd_addr.bin 0x10001080
```

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
SDK	Software Development Kit
SES	SEGGER Embedded Studio
SoC	System on Chip: microcontroller with integrated peripherals
UICR	User Information Configuration Register

Table 2: 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

☞ 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

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@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 2 8448 2016
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