E73-TBA/E73-TBB User Manual

nRF52810/nRF52832 Test Suite

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.

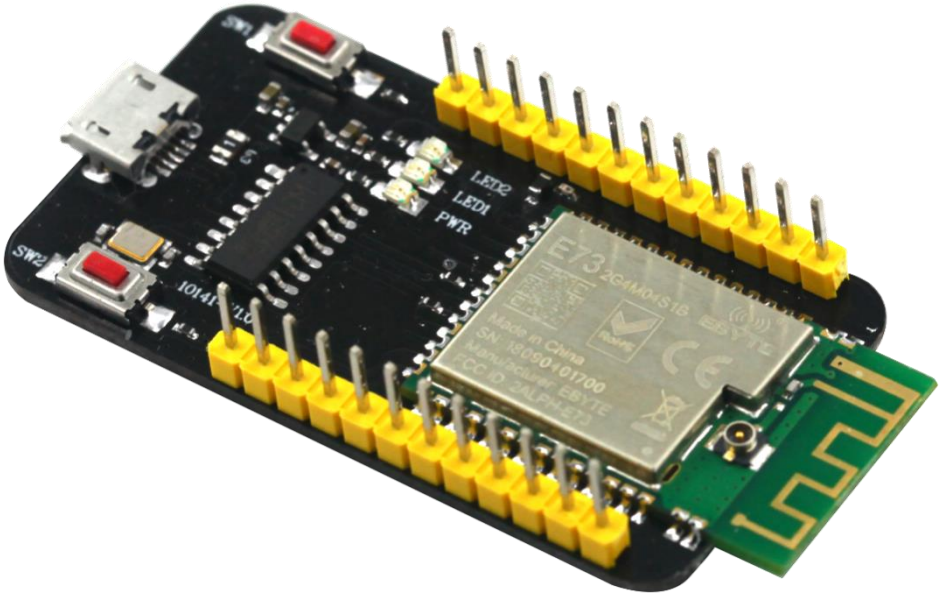
Contents

1. Development Board Hardware Introduction

1.1 Brief Introduction

With the continued popularity of smart wear and IOT Internet of Things, more and more companies and individuals have begun to develop wearable products and IoT products. In order to facilitate the development of multi-protocol SoC based on nRF52832/nRF52810 series for R&D personnel, founders and students, EBYTE developed nRF52832/nRF52810 test board, which has below features:

- NRF52832/nRF52810 chip with low power consumption, high performance and multi-protocol on board.
- Support MESH networking
- Support BLE 5.0
- Integrated high performance USB to serial chip
- Two user buttons
- Two user LEDs
- An NFC Antenna Interface
- Compact size: 28*52.5MM, easy to carry
- IPEX interface, external antenna
- Reserved SWD interface for external JLINK debugger



1.2 Principle Description

Block diagram of E73-TBX principle

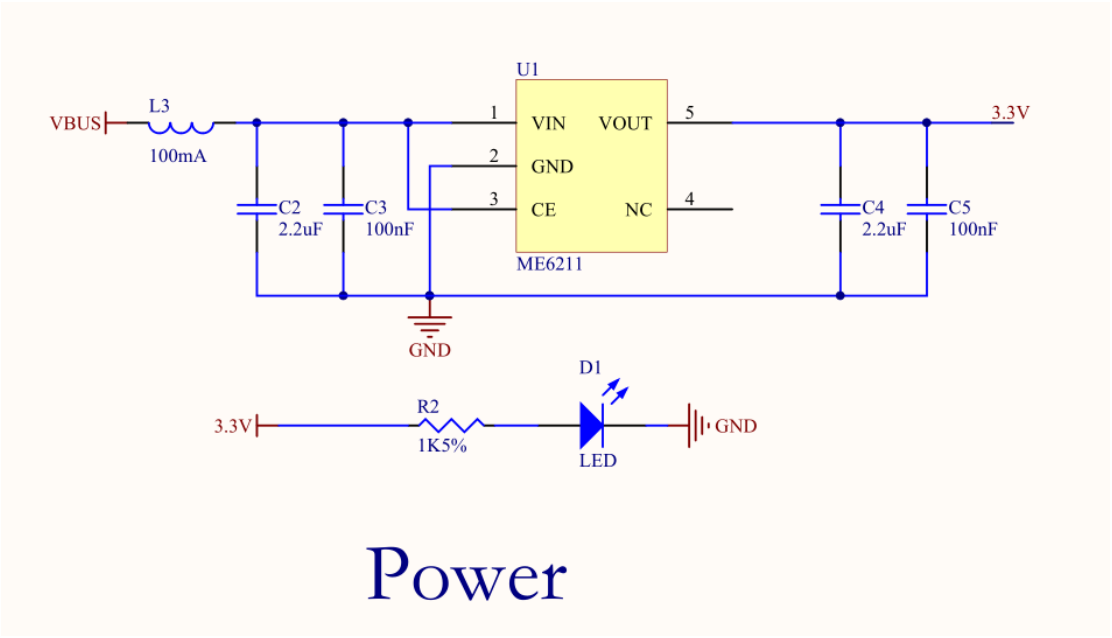
1.2.1. Chip Introduction

NRF52832 chip is a new multifunctional SoC chip with 32-bit CORTEX-M4 core, which supports Bluetooth BLE, ANT+, 2.4G private protocols, launched by NORDIC. Compared with the previous generation chip nRF51, the chip has the characteristics of faster speed, lower power consumption and stronger function. The following are the main features of nRF52832 chip:

- Integrated high performance 2.4G RF transceiver supporting multiple protocols
- 32/64MHZ external clock
- On-chip integration 512KB FLASH, 64 KB RAM
- 7.7mA TX at +4dBm
- Air compatible NRF24L series, NRF24AP series, NRF51 series SoC
- 20 PPI channels
- Settable transmit power: -20dBm-4dBm
- Ultra-low standby current: 400nA
- Receiving sensitivity - 96 dBm (BLE)
- Integrated NFC, support near-field detection and wake-up, support OOB pairing
- Integrated BALUN
- Integrated PDM, IIS interface
- 3X4 channel PWM output
- 12BIT 200KSPS ADC

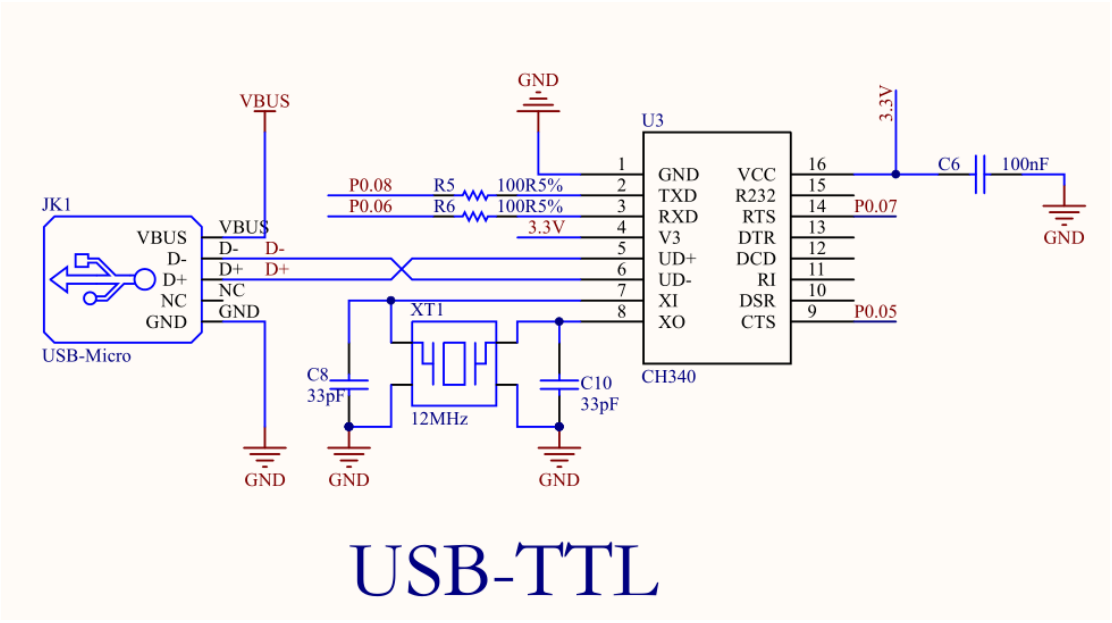
1.3 Development Board Circuit Diagram Introduction

1.3.1. Power circuit

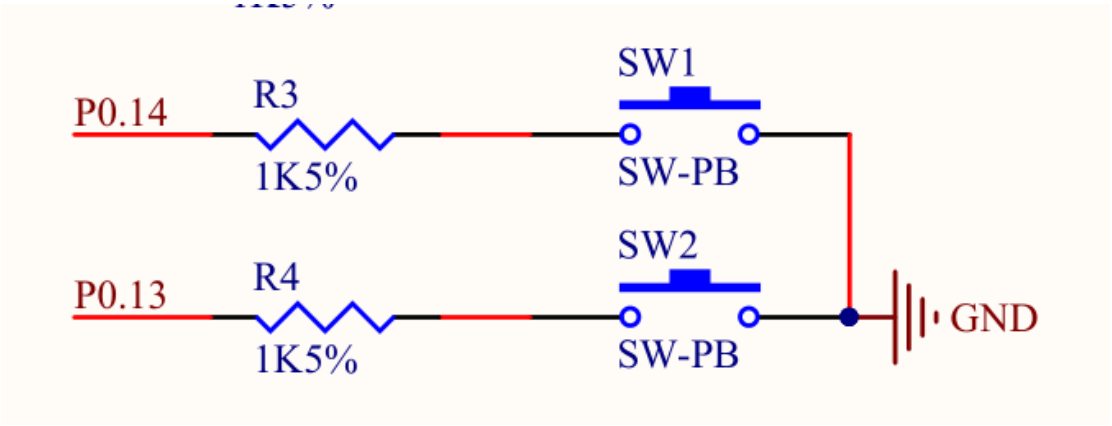


After the 5V power comes in from the USB port, it is protected by a self-recovery fuse, and a stable 3.3V voltage is output through the ME6211.

1.3.2. USB to Serial Circuit

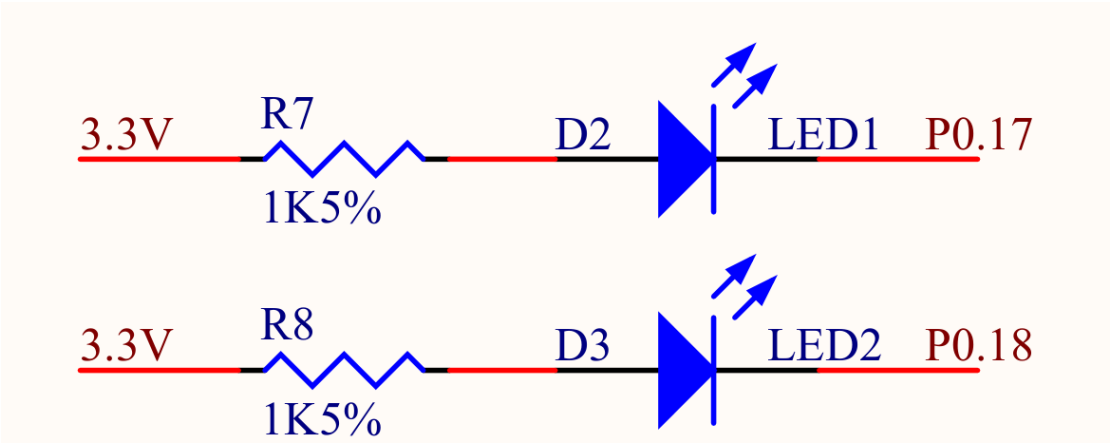


1.3.3.User button



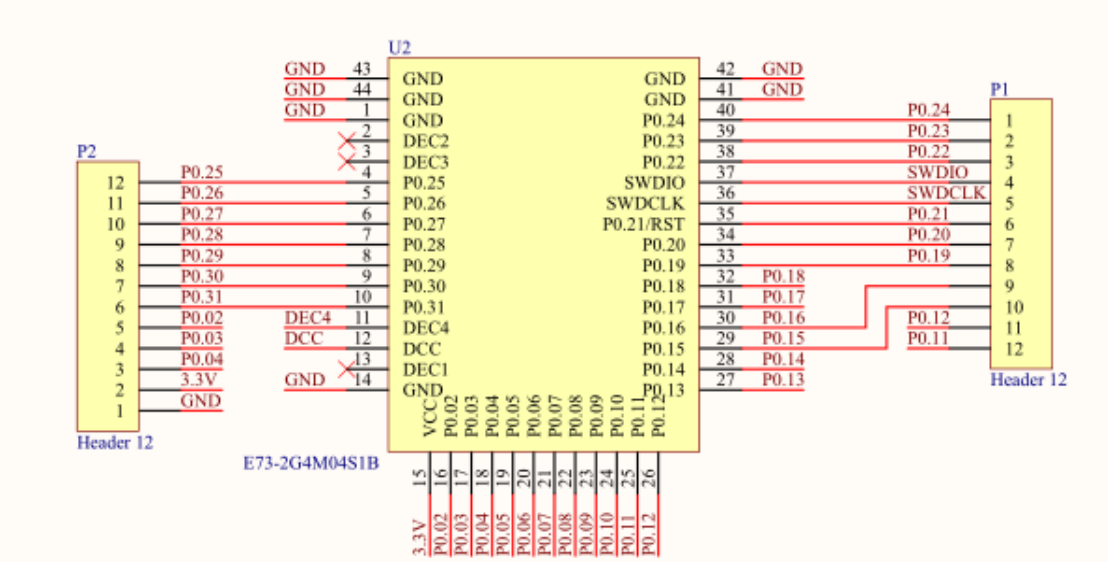
Reserve 2 user buttons.

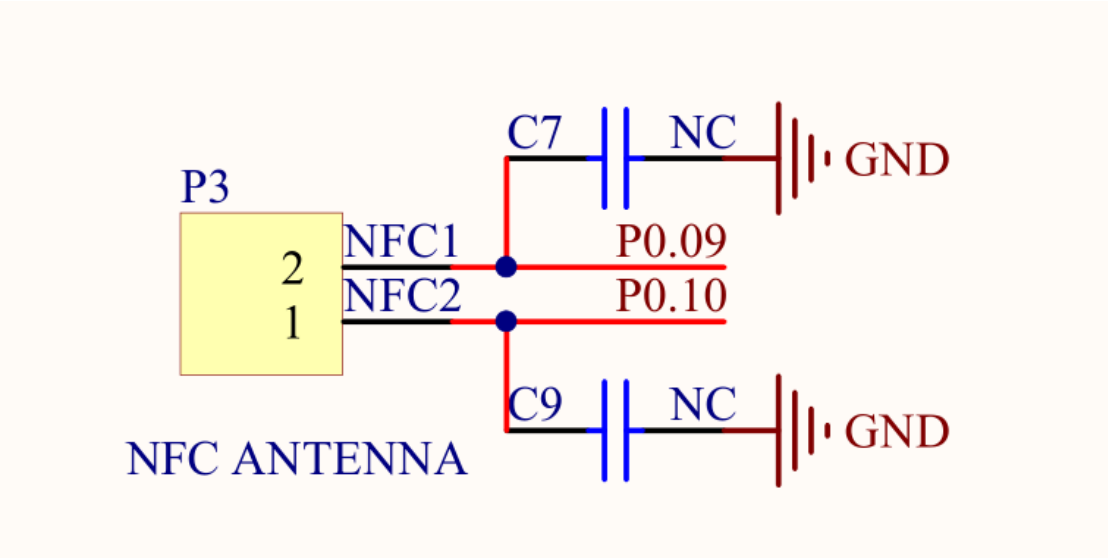
1.3.4. User LED



Reserve 2 user LED

1.3.5. Extended pin header



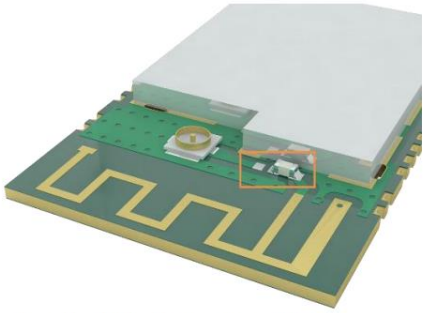


The remaining GPIO and NFC interfaces are brought out.

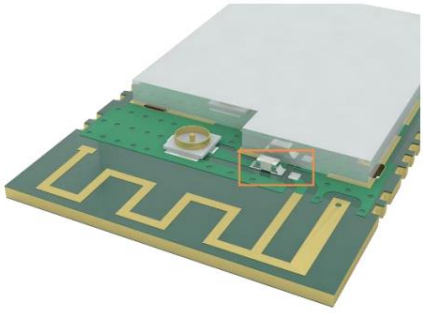
1.4. Development Board Interface Description

1.4.1. External antenna connection

The IPEX antenna mount is reserved on the E73-TBX. If users need to use the external antenna, please solder the last matching capacitor of the module to the position inside the circle, and then plug the external copper tube or rubber antenna into the IPEX seat. The antenna selection part of the corresponding module manual can be seen in detail.



Choose: Onboard PCB antenna



Choose: IPEX interface

1.4.2. Power Supply and Serial Port

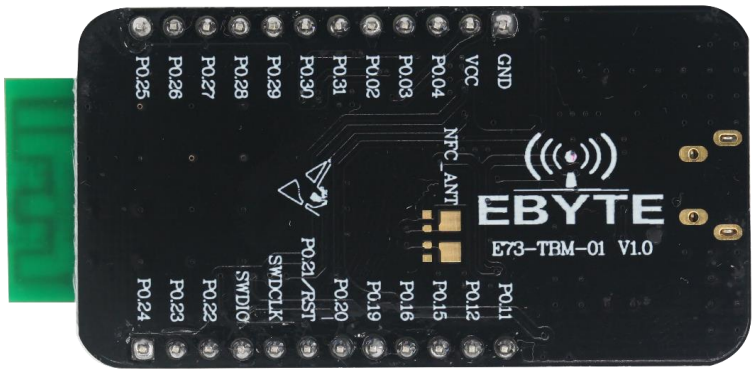
The USB port on the development board provides 5V power for the development board, and it is also the communication port of the onboard USB to serial port chip CH340G and PC. The development board supports USB port and JLINK debugger power supply. But please don't both supply power at the same time.

1.4.3. Program Download Interface

The development board has reserved 4-wire SWD interface, support common debugging tools such as JLINK V8, JLINK V9. It’s not recommend to use JLINK OB, which is bad for NRF52 and unstable.

The position of the debug port is as follows: there is a pin definition on the back of the test board, please check, pay attention: VCC=3.3V

1.4.4. Pin Definition



1.5. Development Board Open-Box Testing

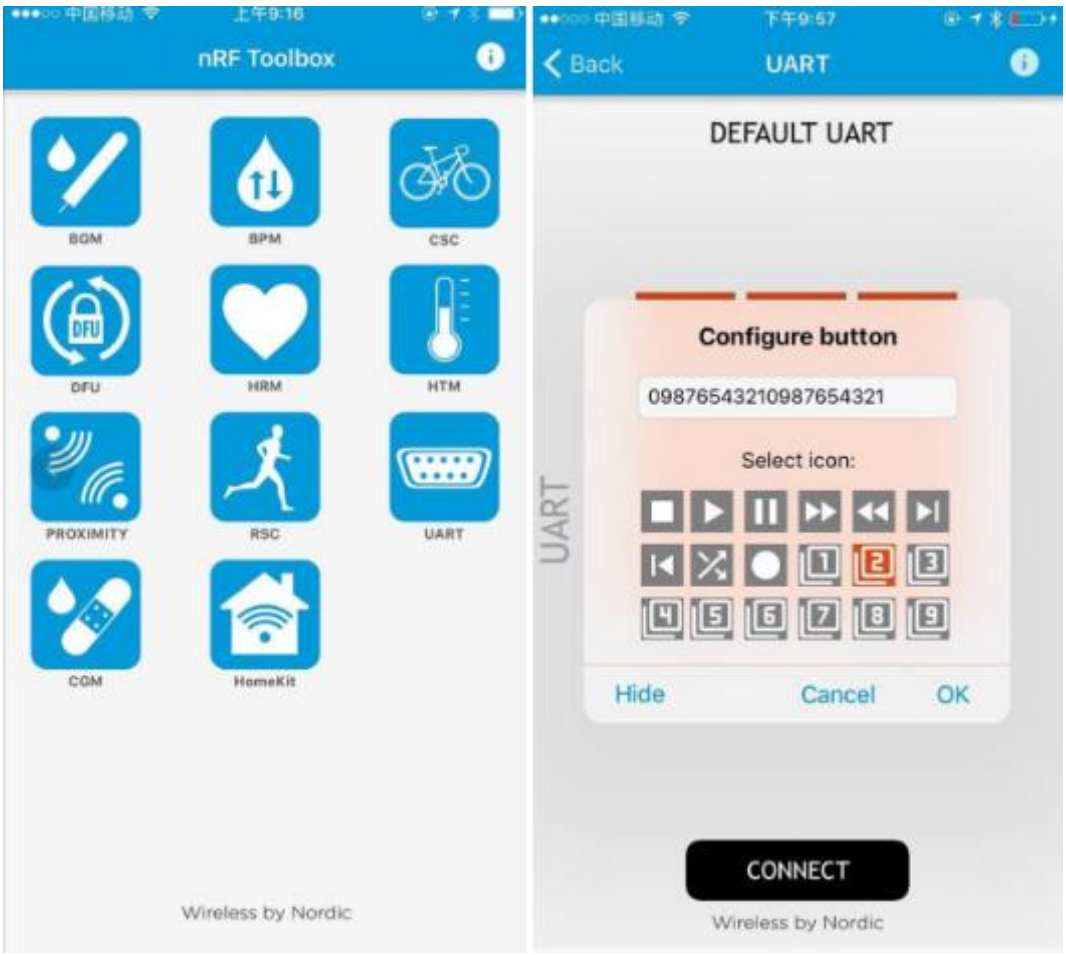
1.5.1. Packing List

E73-TBX test board x 1（E73-TBA/E73-TBB called E73-TBX）
MICRO USB cable x 1

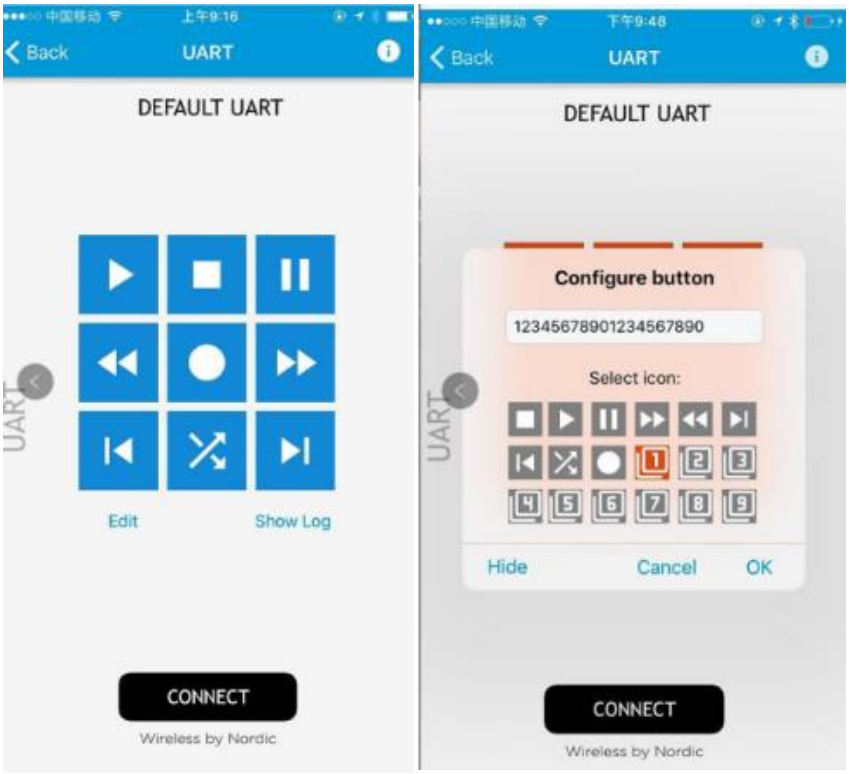
1.5.2. Open-Box Testing

When received the packet, please check whether all parts are include. When shipped, the development board burns bluetooth transparent transmission program already. The below testing method is to ensure whether the board is well or not via the data transmission function between PC serial port and APP.

1) Install the serial port assistant in the computer and install nRF Toolbox in Phone. Run the nRF Toolbox in phone, and find the icon of “UART” as below:



2) After clicking “UART”, there’s 9 buttons. We can set the icons of buttons ourselves. Also We can set the data contents that APP sends when clicking on these icons. Click “EDIT” to enter the editing of the button icon and the editing of the sent content.



3) We set the first button icon to "1"; when the button is pressed, the content sent is set to "12345678901234567890".

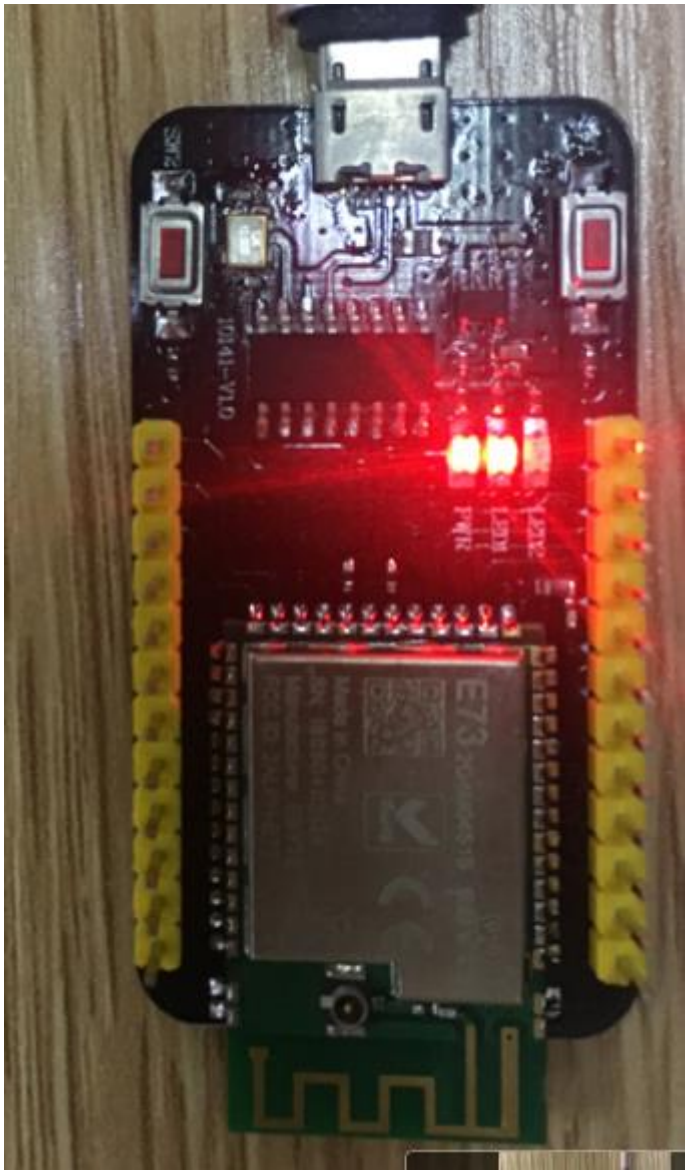
4) We set the second button icon to "2"; when the button is pressed, the content sent is set to "09876543210987654321"

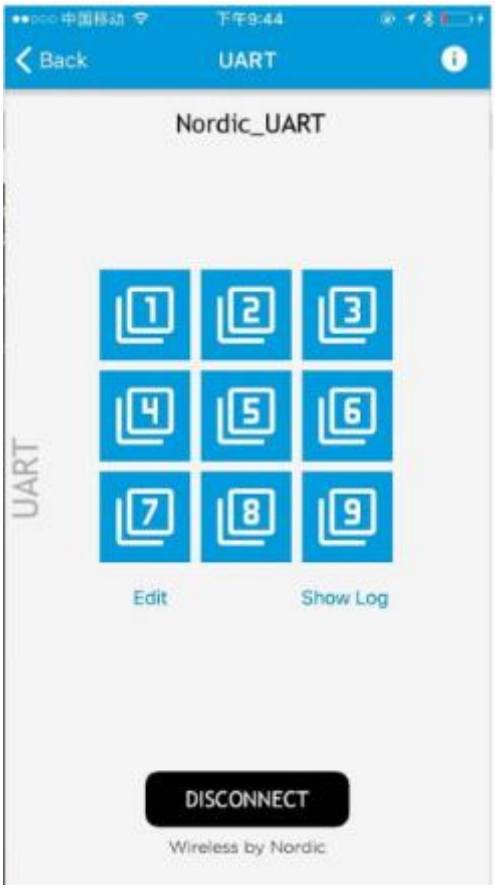
5) We set the third button icon to "3" and set the send content to "inforlink_nRF52832 E73TBM01".

6) After editing, click “DONE” to exit the setting interface.

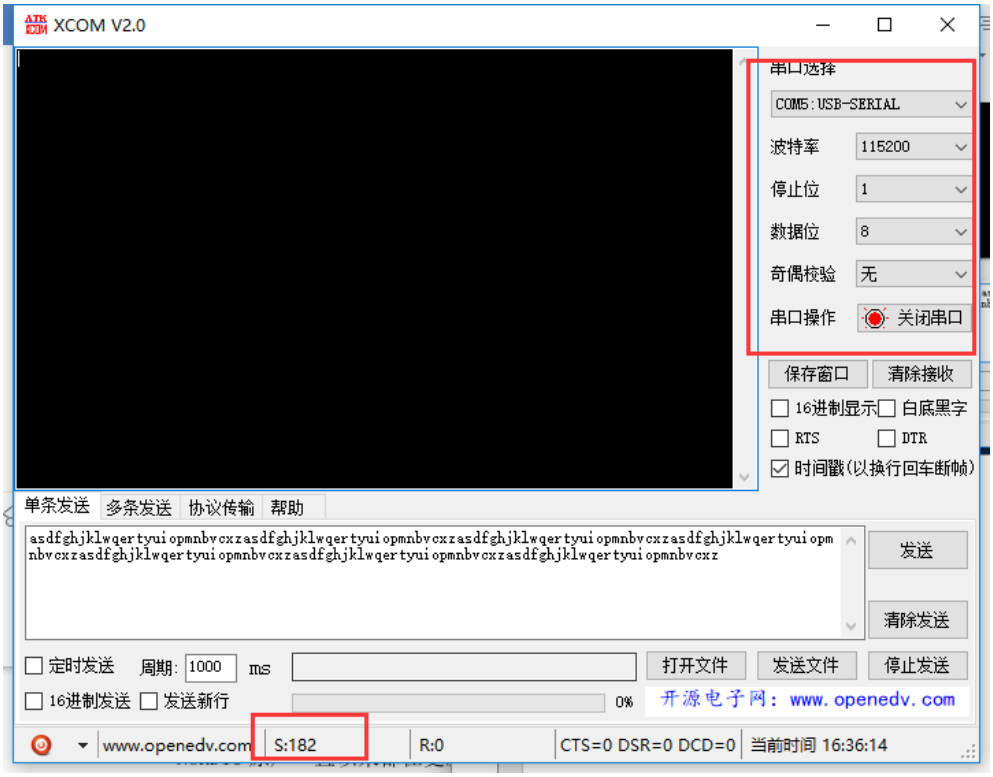
7) Connect the development board to the computer with the USB cable. The red LED is flashing, indicating that a Bluetooth broadcast is being sent.

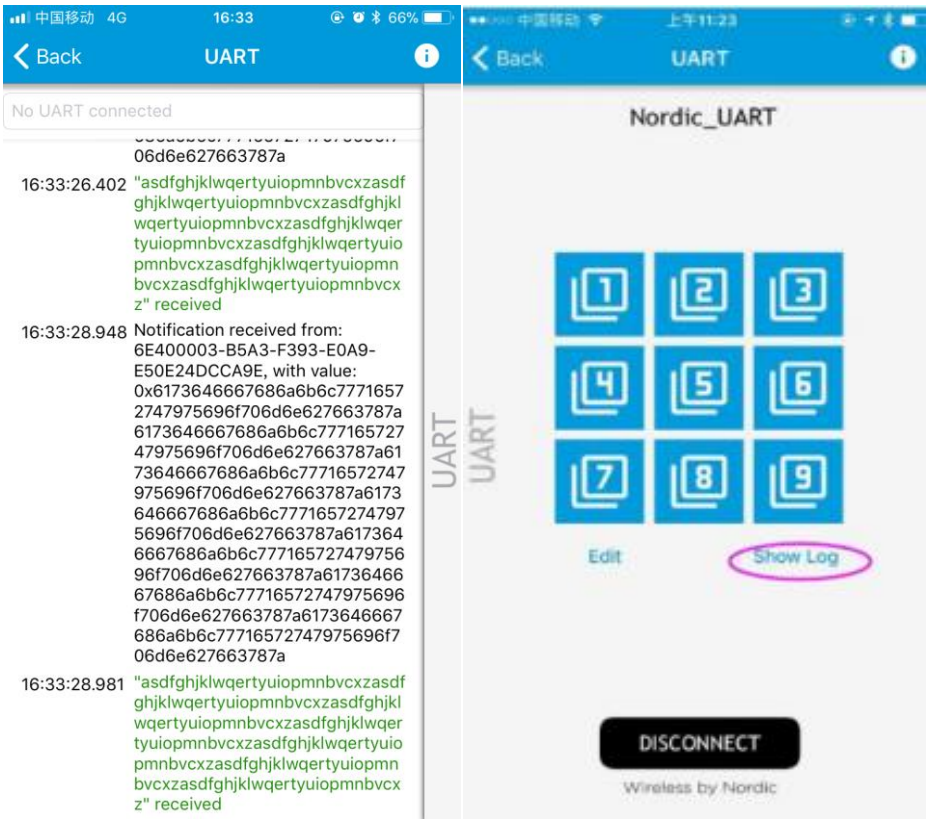
8) Click on the “CONNECT” link and a Bluetooth device named “Nordic UART” will be found. Click to establish a connection.



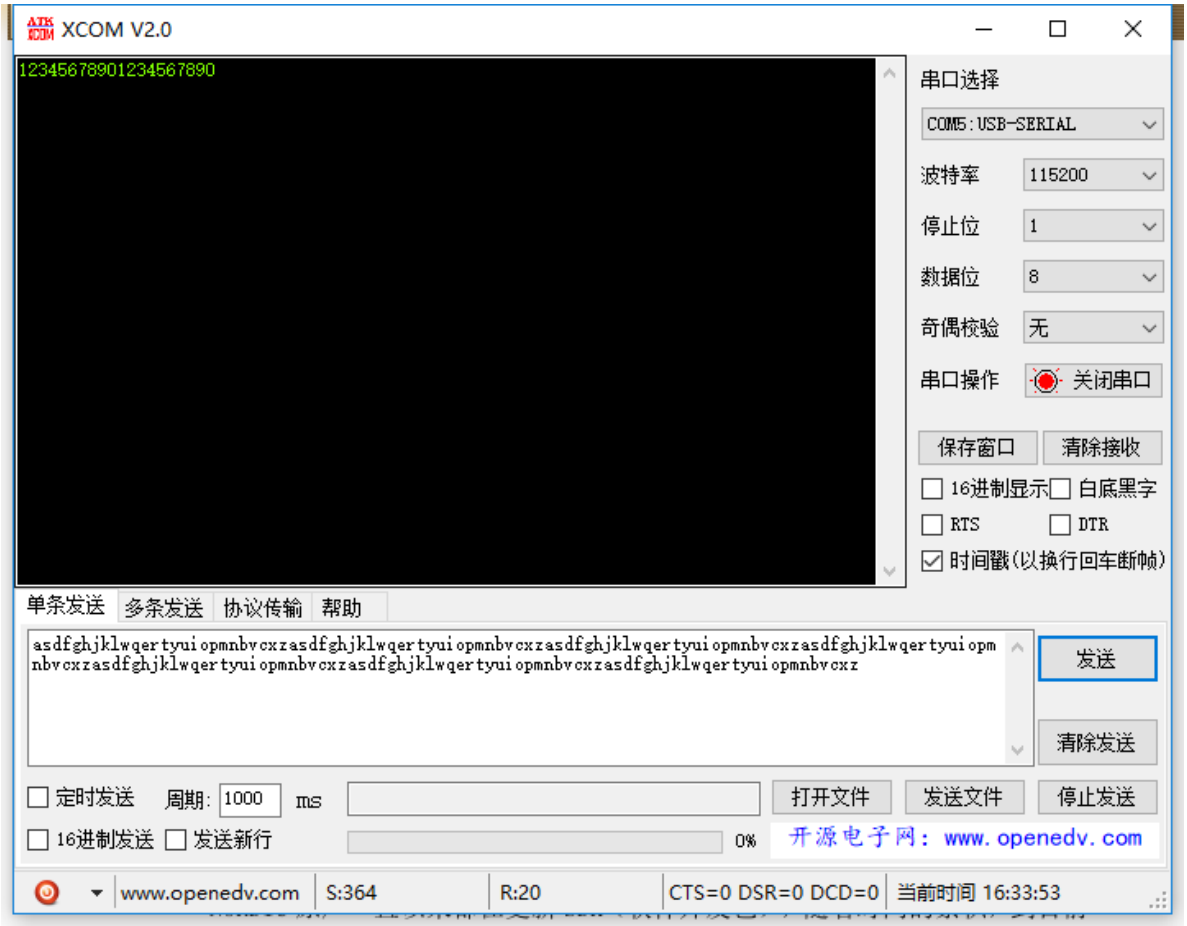


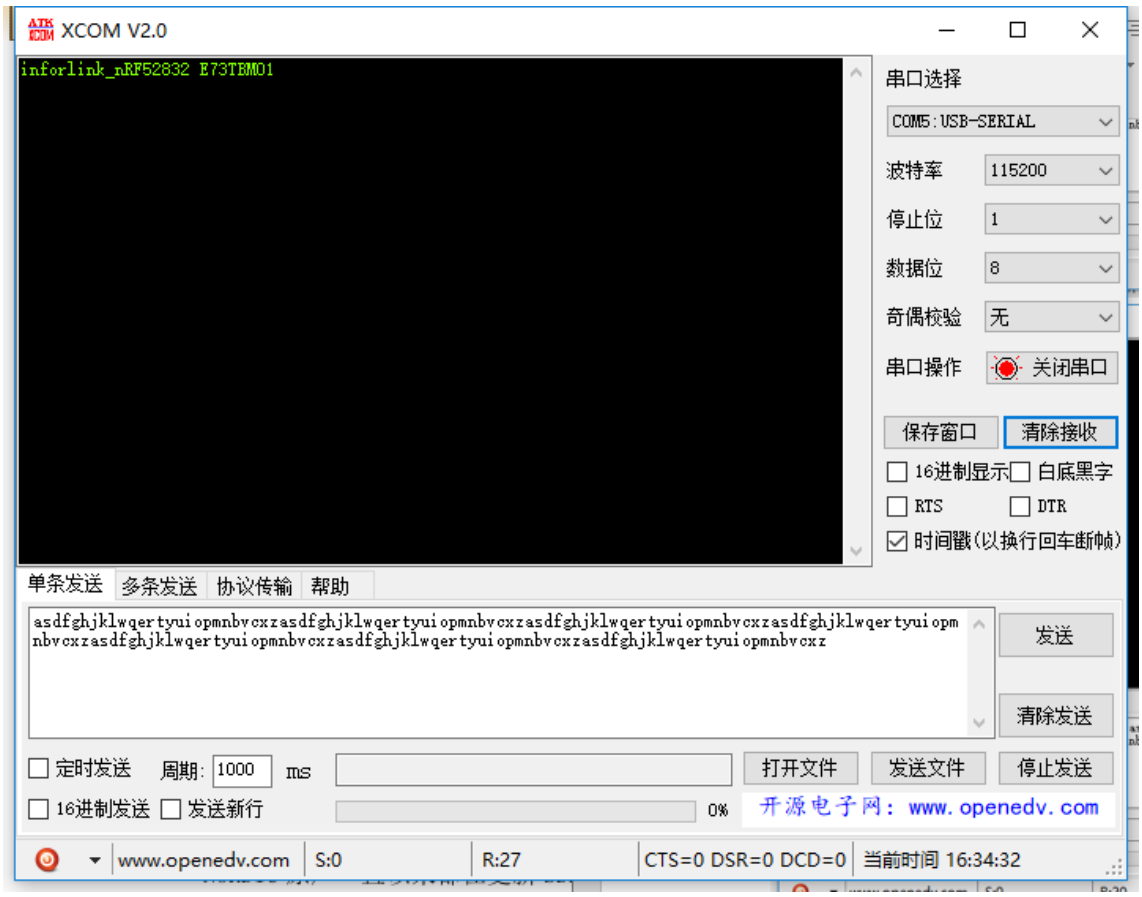
9) Run the PC serial port assistant software, set the parameters as shown below. In the sending data window, enter the data "1234567890", click Send, we can see the data sent by PC in the LOGO area of APP.

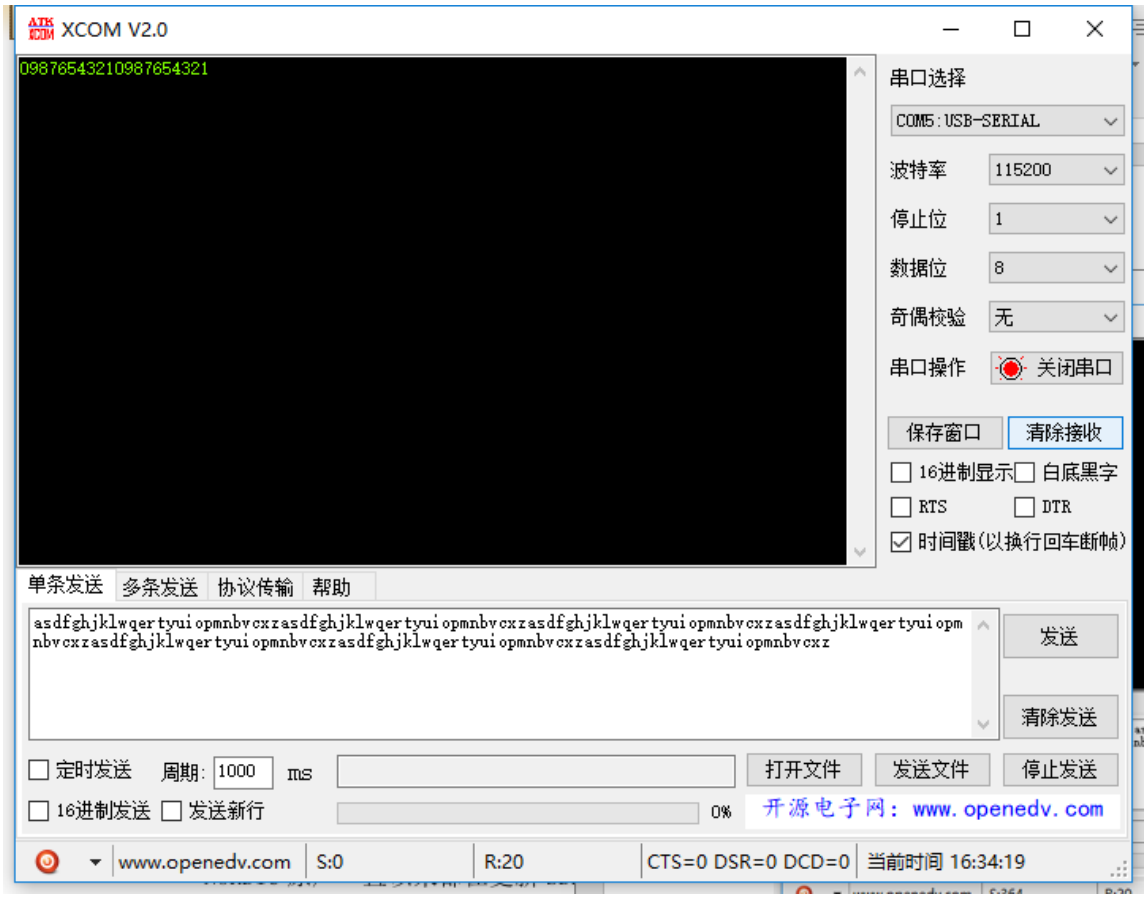




10) We click “1” button, “2” button and “3” button on the mobile APP terminal, the PC serial port will receive the corresponding data.



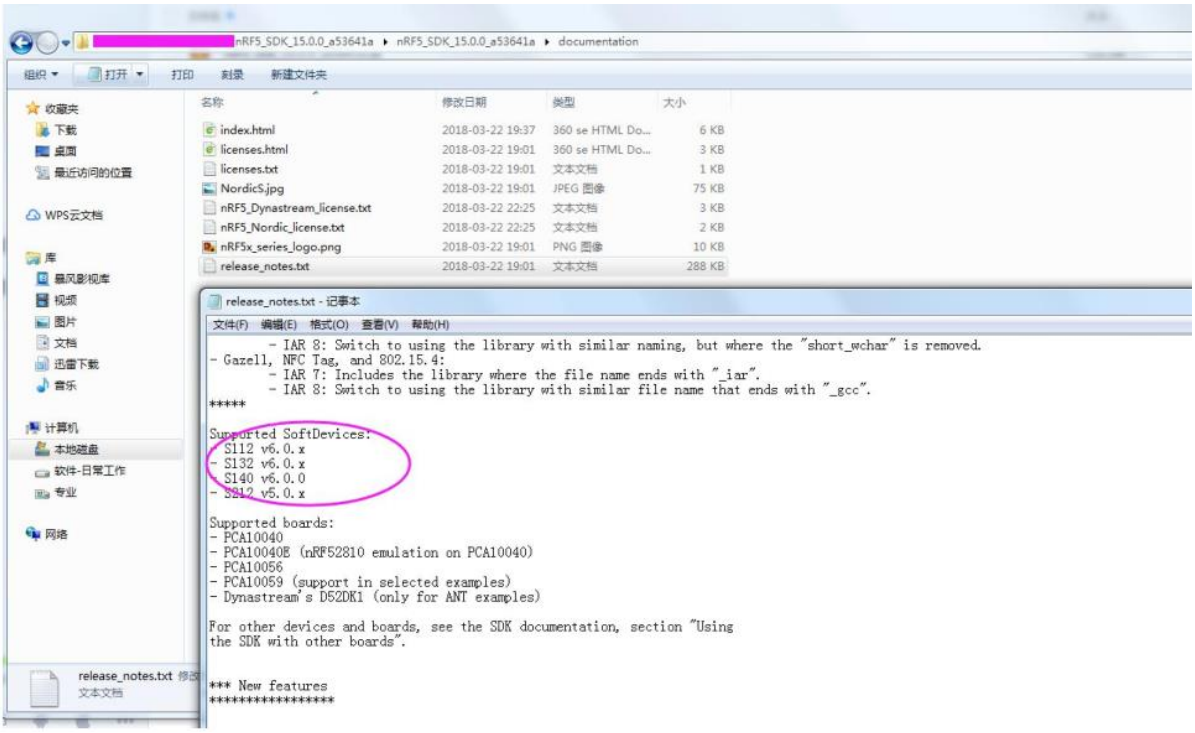




2. Development tool introduction

2.1. Description of program burning

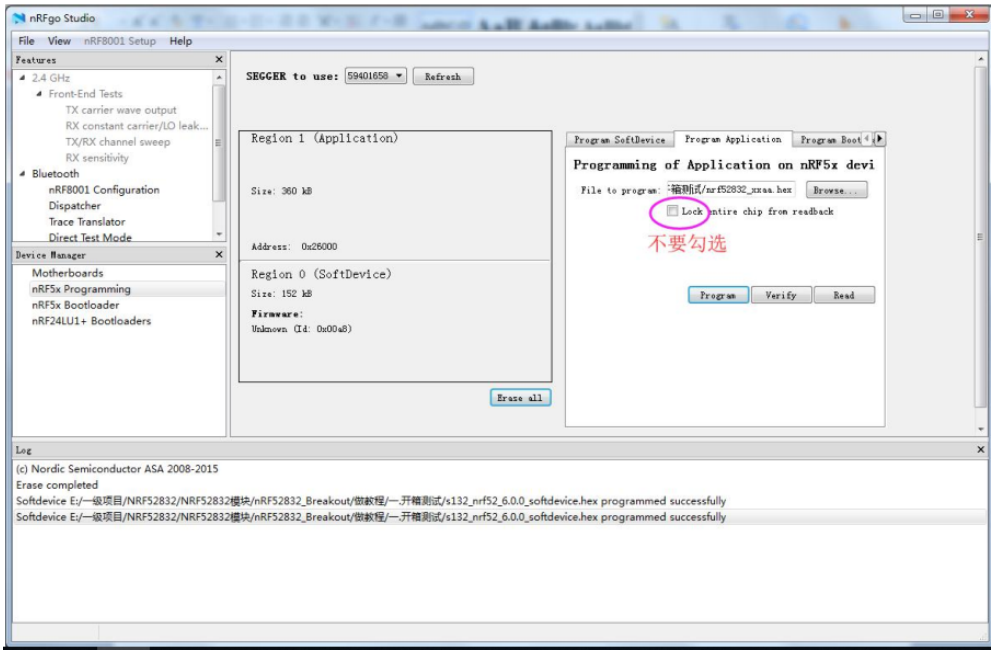
1）Different SDK versions only correspond to the specified protocol stack version. Please refer to the RELEASE NOTE attached to the SDK for the correspondence between the SDK and the protocol stack. See below:



2) Be clear when the protocal stack can be burned.

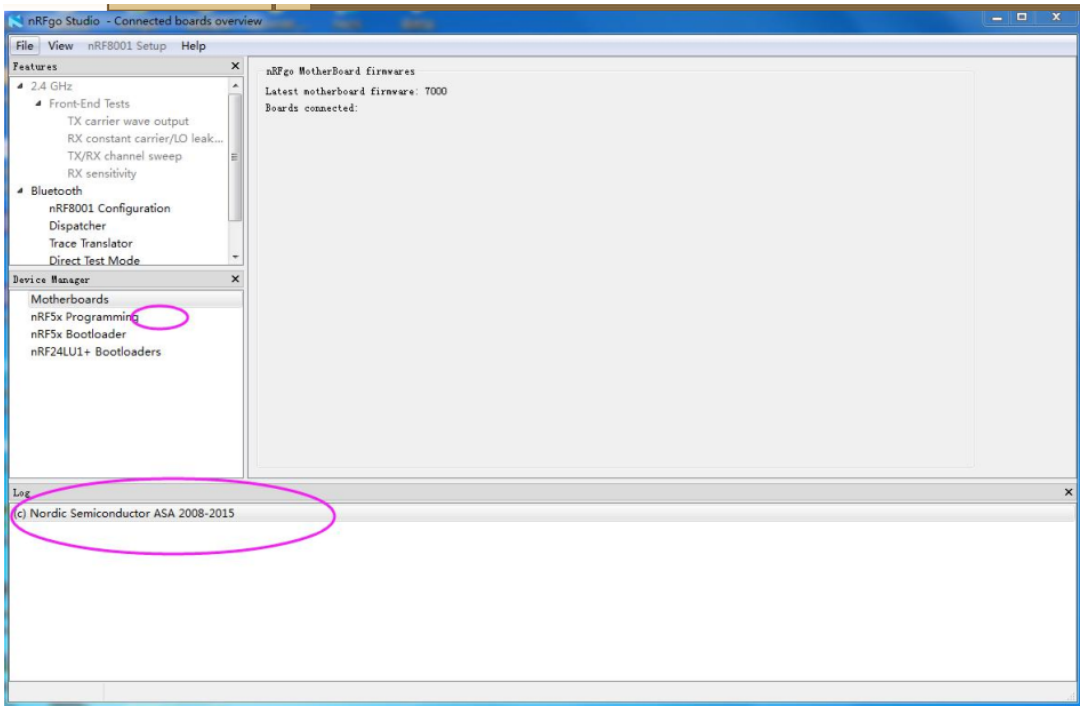
The nRF5X of NORDIC has a feature that the chip work as SoC to run Bluetooth, or work as a normal MCU.

3) Please do not select readback protection when burning the program during development. Use nRFgo Studio to burn the program, do not select the readback protection. Otherwise, It will be failed when you burn the program in MDK.



2.2. Instructions for NRFGO STUDIO

One end of JLINK V9 is plugged into the USB port of the computer, and the other end is connected to the SWD burning interface of the test board. After double-clicking the nRFgo Studio icon on the start menu or desktop, the following interface appears:



Please notice the two circles in above picture. If the font in the first small circle shows black, which means JLINK V9 burner is detected by computer, and the driver is installed already. If it's grey, which means JLINK is failed to connected to computer, then check below:

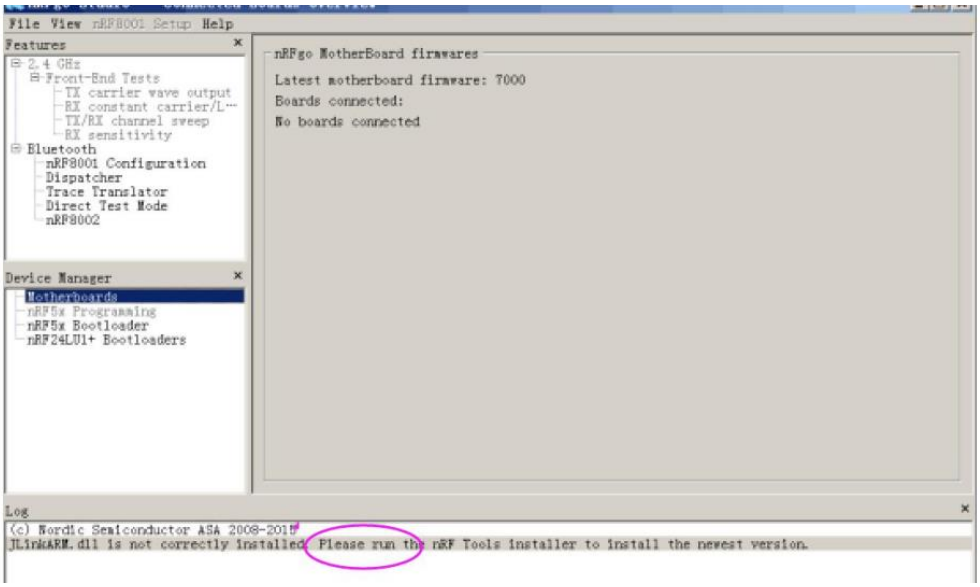
4) Whether JLINK V9 is on good condition

5)Whether USB cable is ok

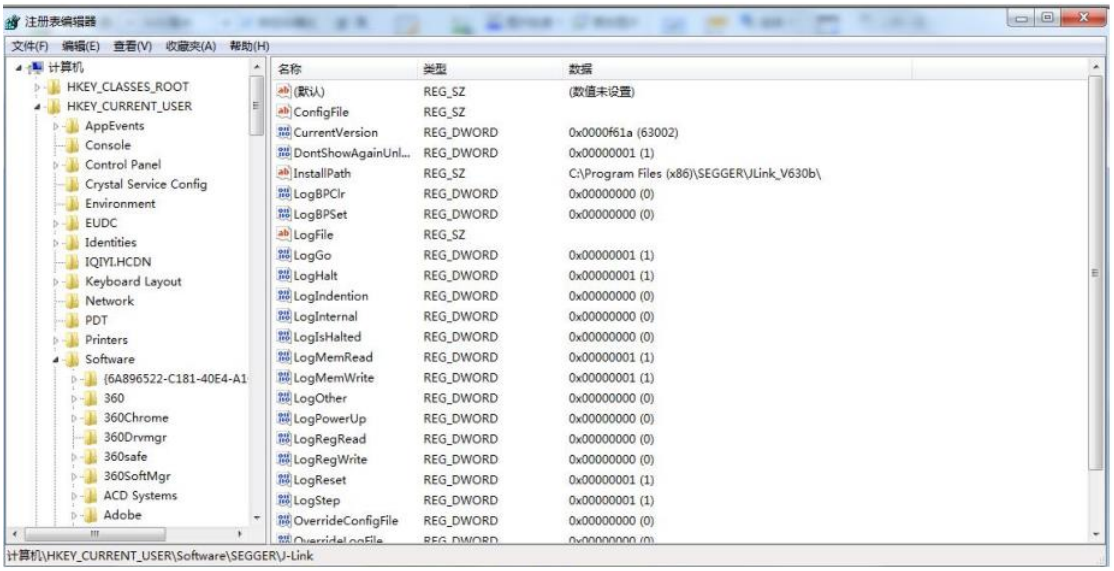
6) The VCC and GND is on the correct connection in case of overcurrent

7) Whether the USB port of PC is loose

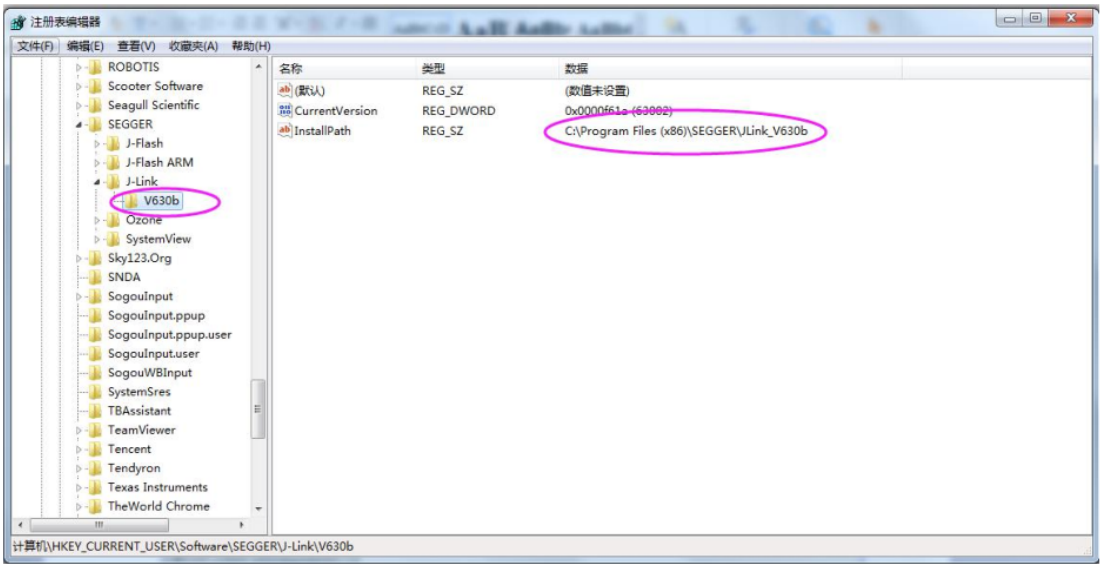
8) Whether it indicates abnormal information in the second circle in above picture as below:



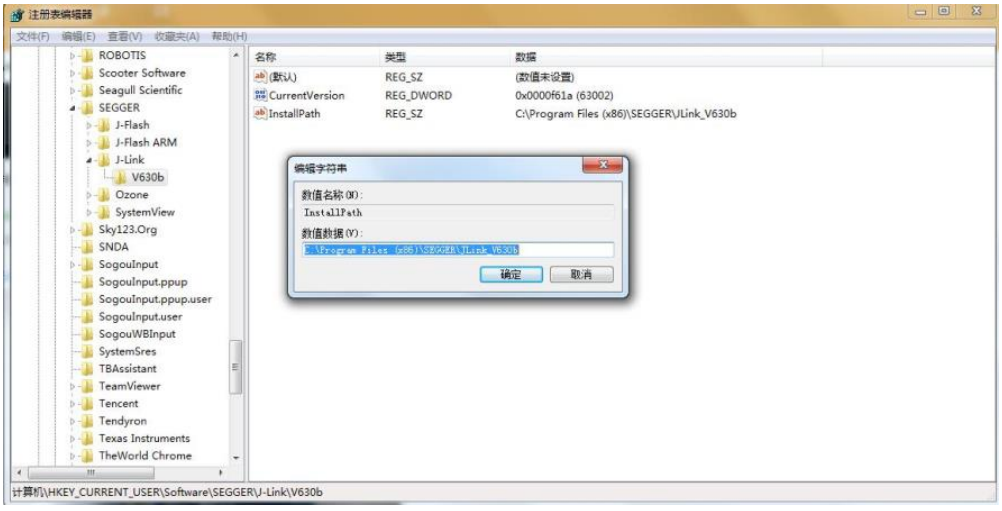
If our computer installed the JLINK aleady, the above issue will occur if we install another version JLINK. The solution is to update the registry manually.



Press the WIN + R key, enter regedit in the pop-up dialog box, and return. The following interface will appear:

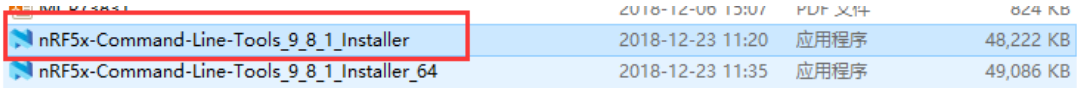


Find the HKEY_CURRENT_USER/Software/J-Link/V630b, then to revise InstallPath.



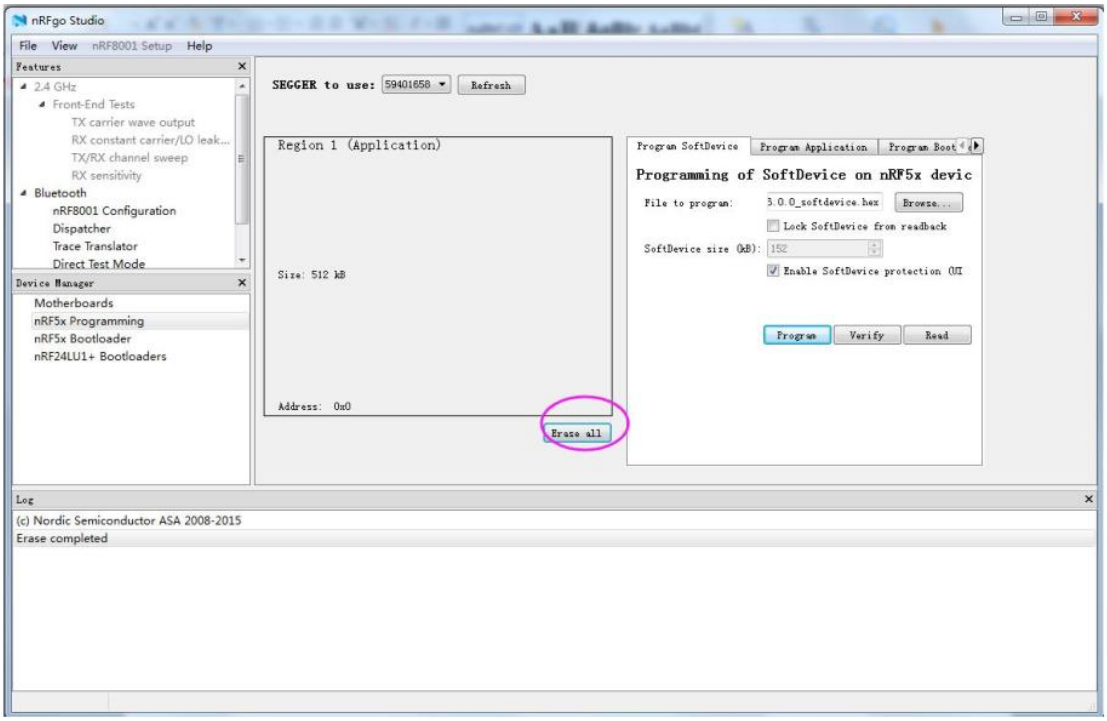
If the 64-bit system does not recognize the NRF52810 chip, it can be solved by direct installation.

Here are the steps for using nRFgo Studio to Burn Programs:



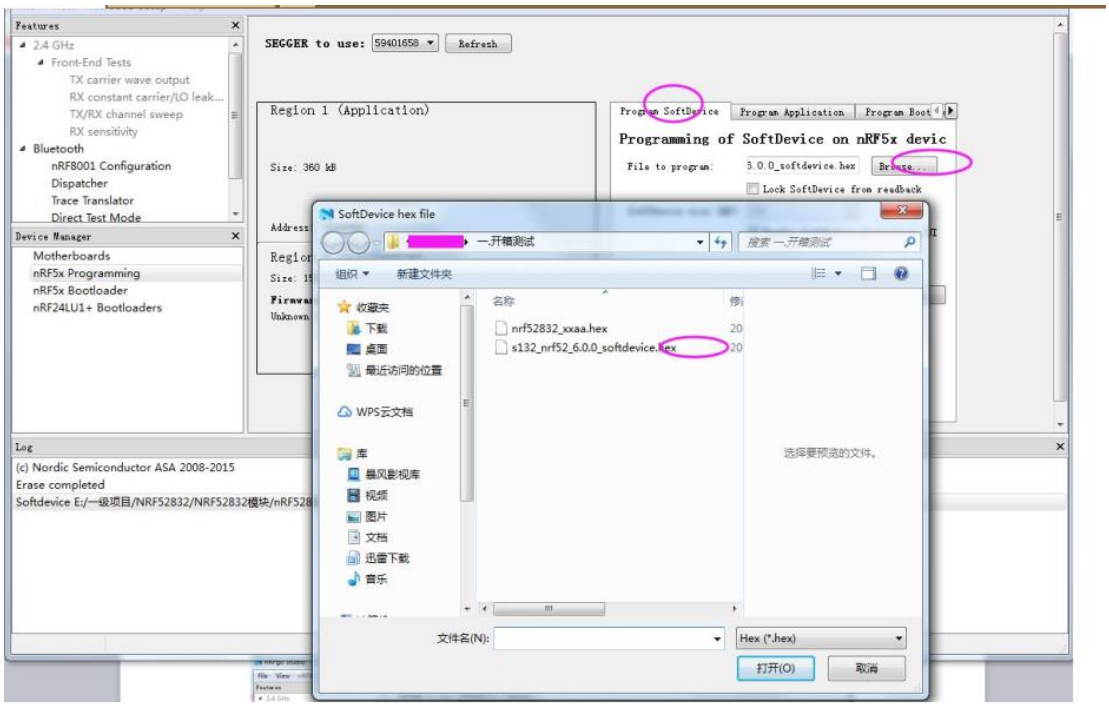
1) ERASE ALL

Erase the chip before burning it. After it succeeds, the bottom of the Logo prompt will prompt Erase complete.



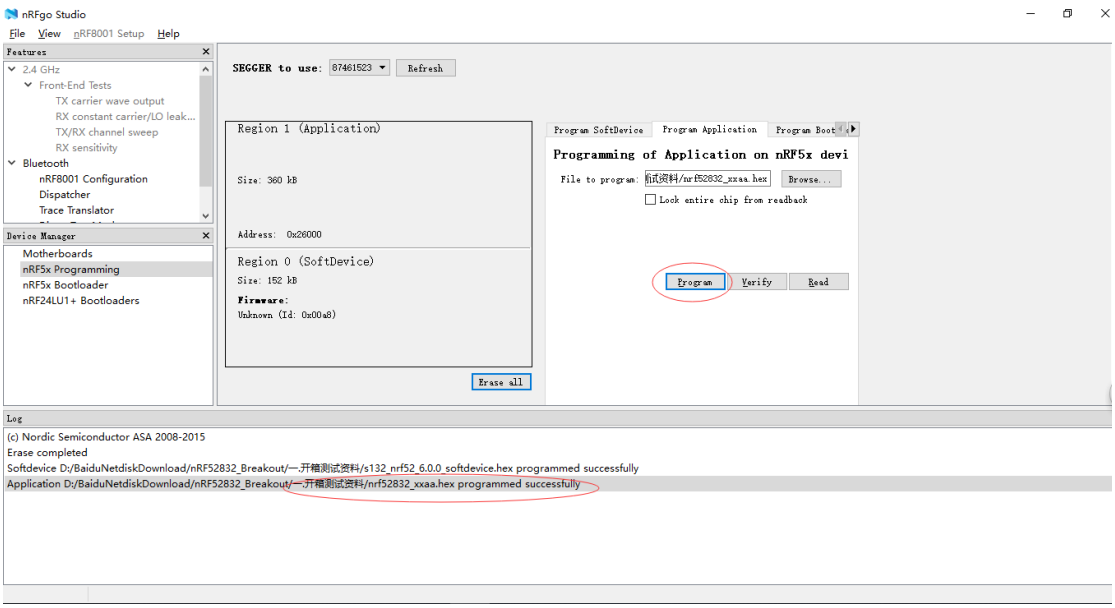
2) Program SoftDevice

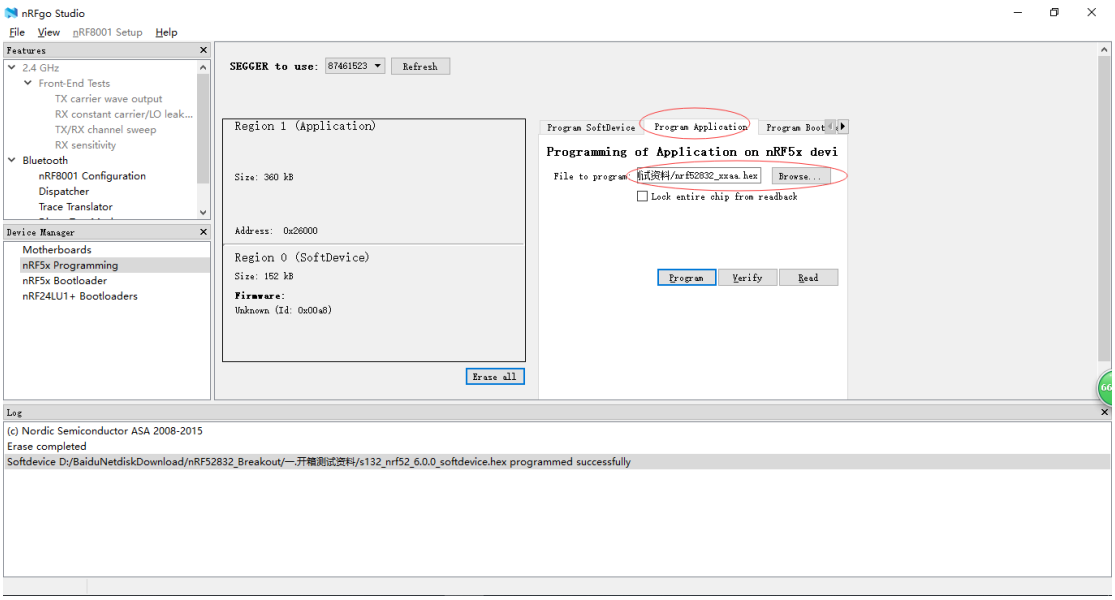
Click Browser, find the path to the HEX file you want to burn, double-click it, and then click Program.



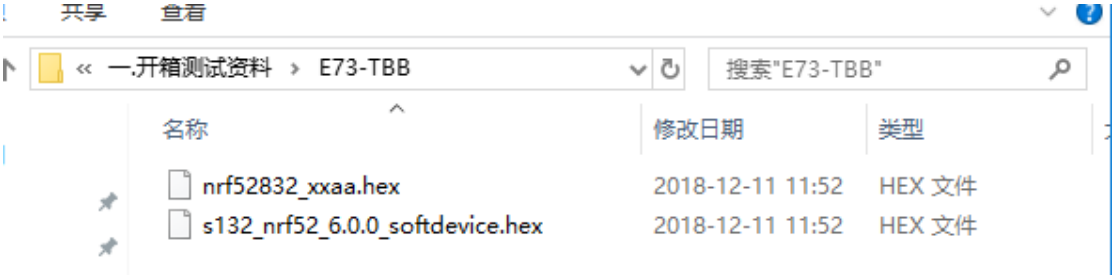
3) Program Application

It's same to program SoftDevice



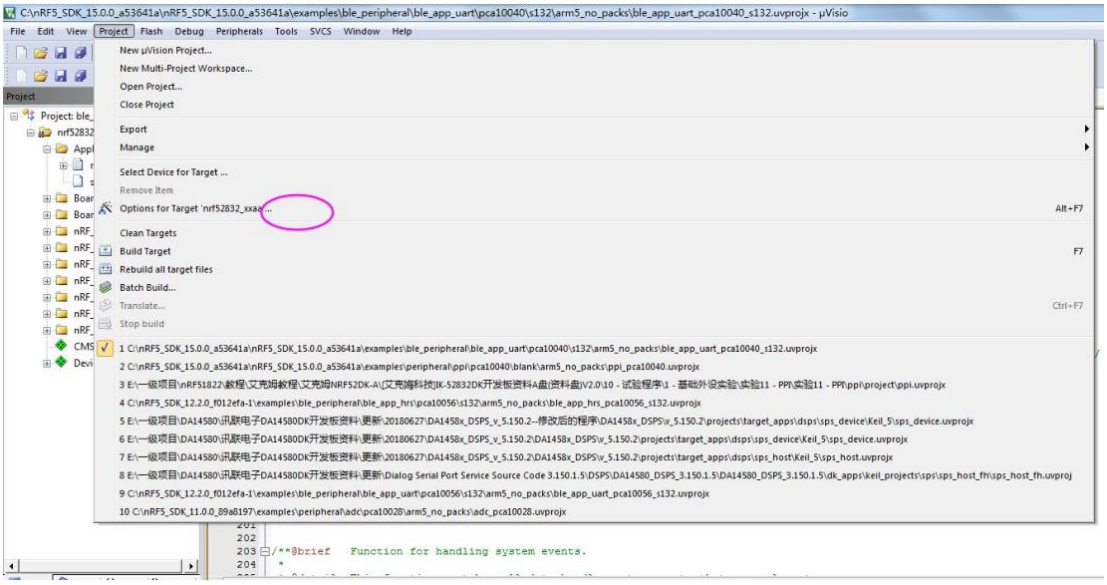


We finished all the introduction on how to use NRFgo Studio to burn the protocol stack and application. Here's another point need to be noticed that the protocol stack and application burning are in sequential order. Users can burn the HEX to the development board to be familiar with it:

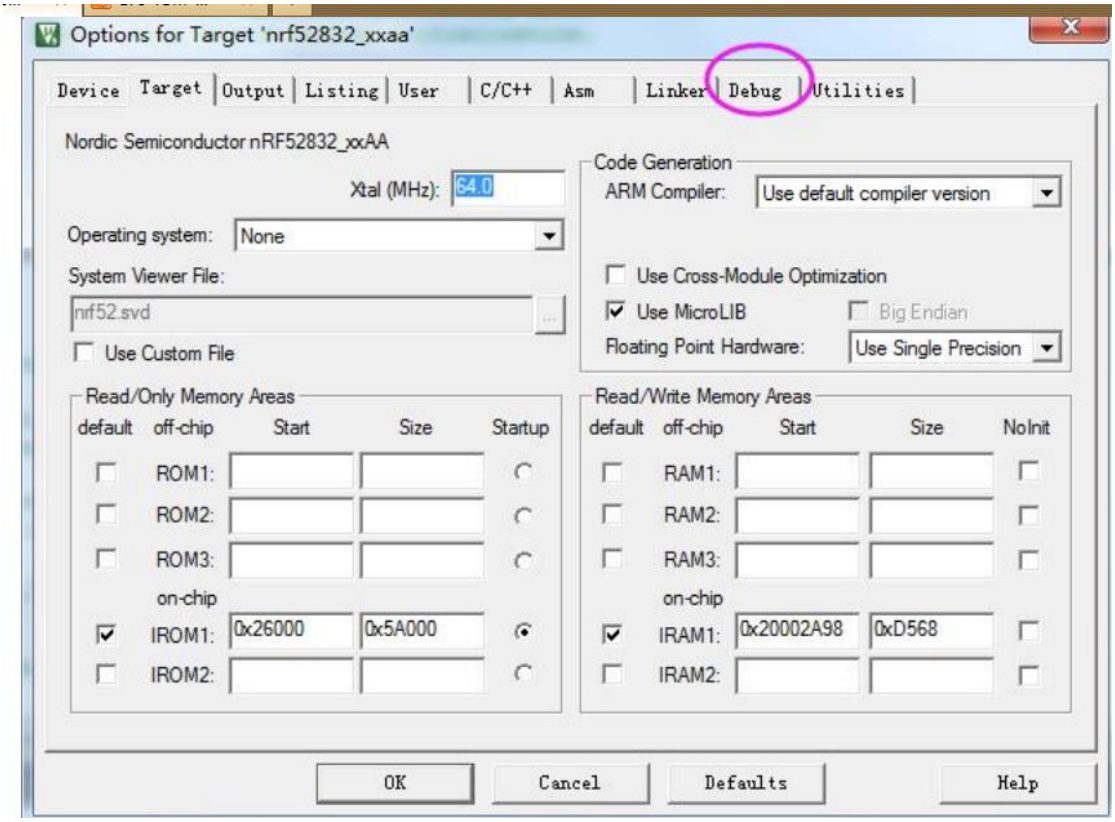


2.3. Introduction to debugging simulation burning tool

We'd advise users to choose JLINK V9 because it has better speed stability. And we need to configure some settings as below:



Select Option for Target xxxx under Project.



Note to select of SWD interface.

3. Basic knowledge of Bluetooth 4.X BLE

3.1. What is BLE

BLE is the abbreviation of Bluetooth Low Energy. Also, BLE is called as Bluetooth Smart, which is a short-distance transmission protocol with low-power and low-rate. It features low power consumption, easy to use, easy to connect etc., which has been widely used in smart wear, smart home, wireless sensor networks and other fields.

3.2. The difference between traditional Bluetooth and low-power Bluetooth

We call Bluetooth Classic as traditional Bluetooth. The main difference are below points:

1）Transmitting power

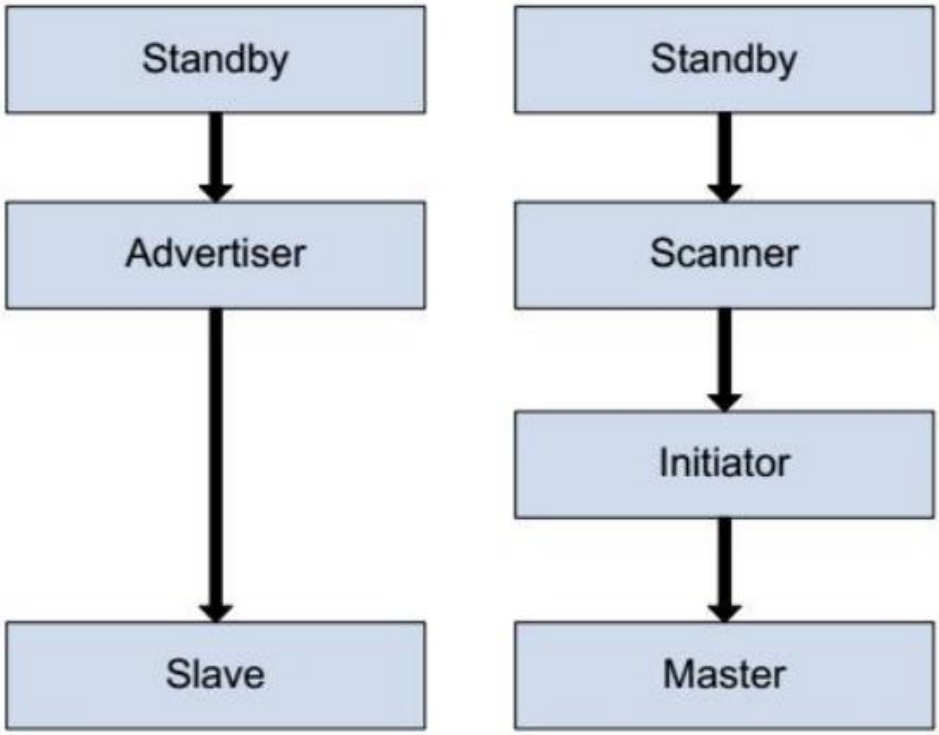
2）Communication distance

3）Data transmission rate

4）The traditional Bluetooth has CLASS1、CLASS2、CLASS3 version, which have higher transmission power, longer communication distance. Some version can reaches 100m. With higher transmission rate, It's suitable for audio transmission. While the low-power Bluetooth has totally opposite features, such as the low transmission power from -30-+4dBm, which is suitable for small data over short distances transmission.

3.3. The working status of the device and the type of Bluetooth device

BLE device has following working status:



When the BLE device does not establish a connection with any device and there is no broadcast, It's on the ready state. If the device is connected to the host by sending a Bluetooth broadcast from the ready state, we call it a slave. If the device from the ready state is connected to the slave by scanning the broadcast of the Bluetooth peripheral, we call it the host.

3.4. Analysis of Bluetooth Broadcasting

3.4.1. Preface

The message is composed of data bytes and is transmitted in bits, which inevitably involves byte order issue. For each byte transmission, it always starts at the lowest bit. For example, 0x80 is sent as 00000001,0x01 is sent as 10000000. At the same time, most byte fields are sent from low bytes. The sending sequence of 0x010203 is110000000-100000010000000. Since the packet capture software may not fully know which data is sent from the low byte, some of the captured broadcast data may need to be read backwards in bytes.

3.4.2. BLE Broadcast Data Structure

In the link layer, BLE broadcast messages are divided into the following parts:
| Leader | Access Address | Header | Length | Broadcast Data | Check|
The general package grabbing tool will display segment after grabbing broadcast data. This paper mainly analyses the content of broadcast data segment.

Leader (1 byte):
It's used to configure the receiver's automatic gain control.

Access address (4 bytes):

For broadcast messages, It's fixed 0x8e89bed6. (The access address also determines the sequence of the preamble)

Header (1 byte):

The sequence is broadcasting message type (4bit), reserved bit (2bit), sending data address class (1bit), receiving address type (1bit).

Length (1 byte):

Indicates the length of broadcast data (broadcast address AdvA + data AdvData)

Broadcast data:

The data segment we need to parse will be explained in detail later.

Check (3 bytes:):

CRC check

3.4.3. Packet capture analysis

P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
					Type	TxAdd	RxAdd	PDU-Length					
4	+27032 =94308	0x25	0x8E89BED6	ADV_IND	0	0	0	33	0x060504030201	0B 09 4E 6F 72 64 69 63 5F 48 52 4D 03 19 34 12 02 01 06 07 03 0D 18 0F 18 0A 18	0x57FE39	-52	OK

Access address in red is the fixed Access Address for Broadcasting 0x8e89bed6. (The packet capture software does not convert byte order)

In the header field:

Broadcast type is universal connectable broadcast (type is 0)

The address type is public address, and both TxAdd and RxAdd are 0.

The length field indicates that the length of ADV + AdvData and the address of the broadcasting device are set by myself.

ble_gap_addr_t add={.addr_type=BLE_GAP_ADDR_TYPE_PUBLIC,

.addr={0x01,0x02,0x03,0x04,0x05,0x06}}

Because the address is 48-bit address, LSB format. So the real address is 0x060504030201. A lot of data in AdvDara is what we need to parse. The following is the detailed information, which has the header data added by the capture software.

```
+-------------------------------------------------+---------------- - - -
|     Packet sniffer frame header                 |
+----+-------------+-------------------------+-------+
|info| Packet nbr. | Time stamp              | Length|  Packet data
+----+-------------+-------------------------+-------+---------------- - - -
| 01 | 04 00 00 00 | EB 01 B9 02 00 00 00 00 | 2D 00 | 2C D6 BE 89 8E 00 21 01 02 03 04
+----+-------------+-------------------------+-------+---------------- - - -
```

Start from packet data

2C is the total bytes of packet data

D6 BE 898E is the access address (byte order problem)

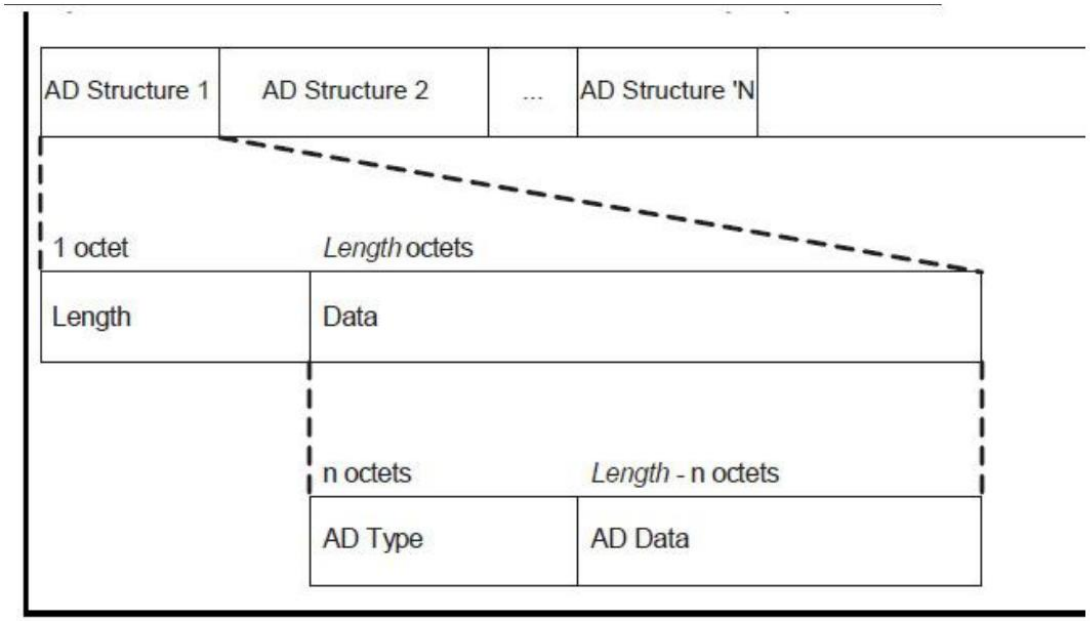
00 is the header field (generic broadcast type, public address)

21 is the value of the length field, indicating the total number of bytes of ADV + AdvData

01 02 03 04 05 06 is the address of broadcasting device

Bluetooth Instruction 4.1 states that the data format is

| length | AD type | AD data|



That is, Advdata is composed of data segments in this format.

Length means length of a small piece of data

AD type indicates the meaning of AD Data data

The definition of AD type can be one of the following figures.

Macros

```
#define BLE_GAP_AD_TYPE_FLAGS 0x01
#define BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_MORE_AVAILABLE 0x02
#define BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_COMPLETE 0x03
#define BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_MORE_AVAILABLE 0x04
#define BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_COMPLETE 0x05
#define BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_MORE_AVAILABLE 0x06
#define BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_COMPLETE 0x07
#define BLE_GAP_AD_TYPE_SHORT_LOCAL_NAME 0x08
#define BLE_GAP_AD_TYPE_COMPLETE_LOCAL_NAME 0x09
#define BLE_GAP_AD_TYPE_TX_POWER_LEVEL 0x0A
#define BLE_GAP_AD_TYPE_CLASS_OF_DEVICE 0x0D
#define BLE_GAP_AD_TYPE_SIMPLE_PAIRING_HASH_C 0x0E
#define BLE_GAP_AD_TYPE_SIMPLE_PAIRING_RANDOMIZER_R 0x0F
#define BLE_GAP_AD_TYPE_SECURITY_MANAGER_TK_VALUE 0x10
#define BLE_GAP_AD_TYPE_SECURITY_MANAGER_OOB_FLAGS 0x11
#define BLE_GAP_AD_TYPE_SLAVE_CONNECTION_INTERVAL_RANGE 0x12
#define BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_16BIT 0x14
#define BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_128BIT 0x15
#define BLE_GAP_AD_TYPE_SERVICE_DATA 0x16
#define BLE_GAP_AD_TYPE_PUBLIC_TARGET_ADDRESS 0x17
#define BLE_GAP_AD_TYPE_RANDOM_TARGET_ADDRESS 0x18
#define BLE_GAP_AD_TYPE_APPEARANCE 0x19
#define BLE_GAP_AD_TYPE_MANUFACTURER_SPECIFIC_DATA 0xFF
```

The following 0B indicates that the length of this piece of data is 11 bytes, that is 09 4E 6F 72 64 69 63 5F 48 52 4D.

According to above table,09 indicates that AD type is a complete local name. We defined "Nordic_HRM" hexadecimal as 4E 6F 72 64 69 63 5F 48 52 4D.

The next 03 represents the next small segment of data bits with three bytes, so the latter 19,34,12 belongs to this segment. The next is 02, while 01 06 belongs to this section. 01 represents FLAG, flag indicates the physical connection function, such as limited discovery mode, does not support classic Bluetooth.

Bit 0: LE limited discovery mode
Bit 1: LE normal discovery mode
Bit 2: BR/EDR not supported
Bit 3: Supports both BLE and BR/EDR for Same Device Capable(Controller)
Bit 4: Support for BLE and BR/EDR for Same Device Capable(Host)
Bit 5...7: reserved

That is, the 06 data indicates that the connection function of the device is the normal discovery mode and does not support classic Bluetooth.

07 indicates 03 0D 18 0F 18 0A 18 belongs to this section. According to above table, The 03 following data represents the complete 16bit uuid list

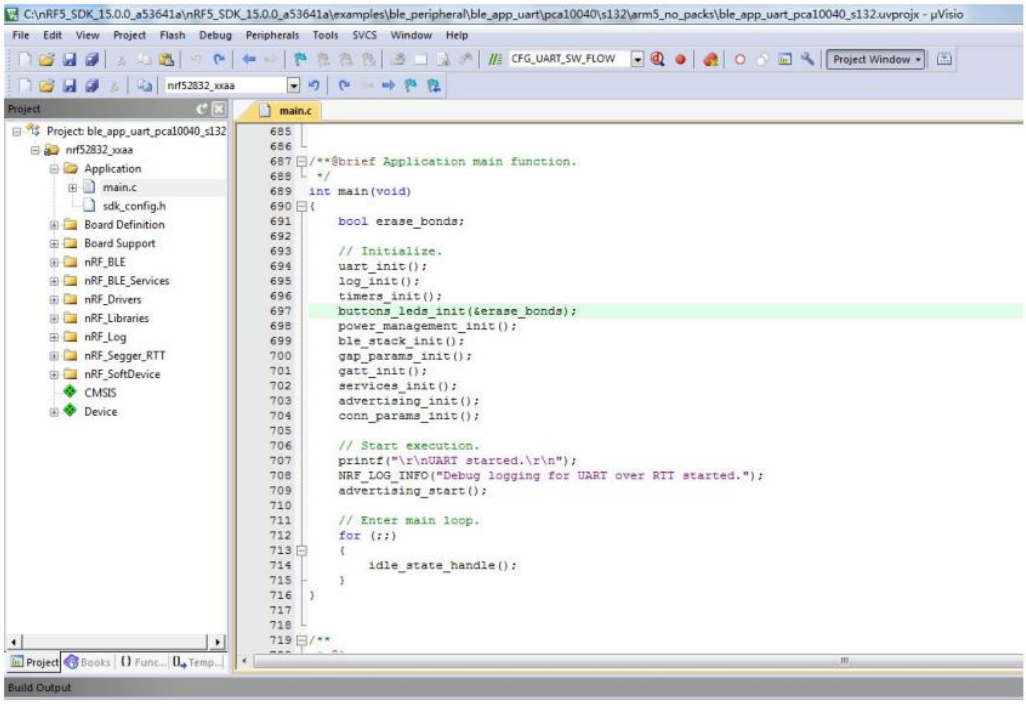
```
ble_uuid_t adv_uuids[] =
{
{0x180D, BLE_UUID_TYPE_BLE},
{0x180F, BLE_UUID_TYPE_BLE},
{0x180A, BLE_UUID_TYPE_BLE}
};
```

It is the broadcast UUID we set, 39 FE 57 represents the CRC check.

4. The Program Structure of Bluetooth in SDK

It has a fixed process to develop nRF5X series SoC. Let's take BLE_APP_UART in the official routine as an example to illustrate the program structure of SDK.

The main function is similar to the following steps:



Power_management_init(); //Power management initialization, necessary

Ble_stack_init()// protocol stack initialization, necessary

Gap_params_init(); // Initialization of the GAP parameter, necessary

Gatt_init(); // GATT initialization, necessary

Services_init(); // Establishment of Bluetooth service

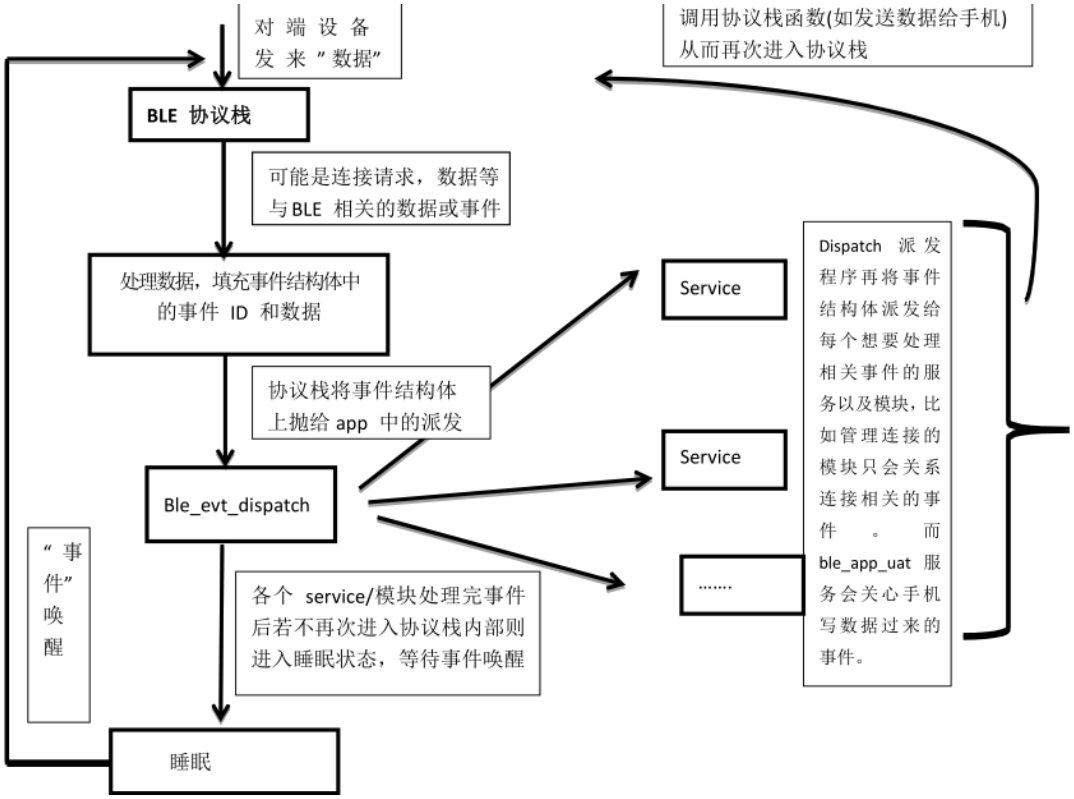
Advertising_init(); // broadcast data initialization, necessary

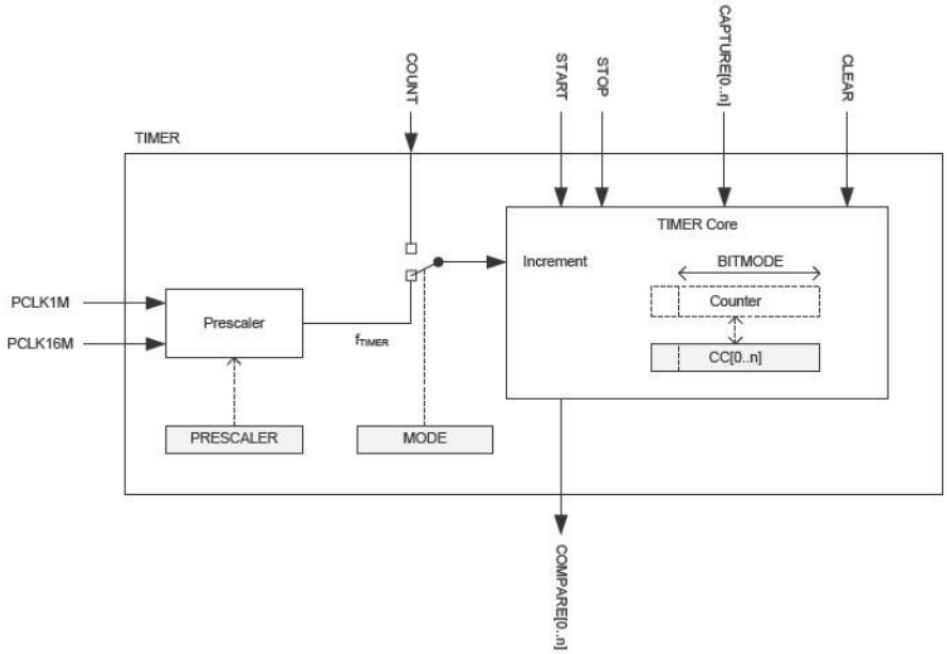
Conn_params_init(); //Connection parameter initialization, if you do not need to update the connection parameters after the connection, It can be unnecessary.

5. Peripheral TIMER

The nRF52832 series SoC has five TIMERs, the corresponding number is TIMER0-TIMER4. All of the 5 TIMERs can operate in both the timing mode and the counting mode.

5.1. Structure of TIMER





Timer block diagram

TIMER consists of the following components:

1) Counting clock source. There are 1M and 16M clock sources.

2) Clock source divider. Used to set the crossover, the range is 0-9 power of 2.

3) Timing/counting mode selection. Used to configure whether TIMER works in timer mode or counter mode.
4) Timing/counter digits. 8-bit, 16-bit, 24-bit, 32-bit selectable.
5) CC[n] register. CC is an abbreviation for capture and comparison. There's 6 CC registers. When the CAPUTURE TASK is executed, the current internal counter value is immediately copied to the CC register.
6) Tasks, events, etc.
TIMER has two modes of operation: timing mode and counting mode. When in timing mode, we call it a timer. When in the counting mode, we call it a counter. Both modes can be started with the START task and stopped with the STOP task.

The timer after stopped by the STOP task can be started by re-executing the START task. When TIMER is in timed mode, the TIMER internal counter counts once every pulse of the FTIMER clock. The frequency of the FTIMER clock can be calculated as follows:

$$f_{TIMER} = 16\ MHz\ /\ (2^{PRESCALER})$$

The corresponding COMPARE[n] event is triggered when the value of the internal counter is equal to the value set in CC[n] (n=1-5). If we enable the interrupt, an interrupt will be generated.
When TIMER is in count mode, the TIMER internal counter counts once every pulse on the COUNT pin.
tasks:
1) START: Start timing/counter
2) STOP: stop timing / counter
3) SHUTDOWN: The timer/counter is powered down, and cannot be started by START unless reset.
Notice:
1) Timers/counters can only be configured when they have been stopped, otherwise unpredictable consequences can be resulted in.
2) When in timed mode, if <=1MHZ, for power saving, the timer will automatically acquire the count pulse from the PCLK1M clock source instead of the PCLK16M clock source.

5.2. Introduction of each register

24.5.1 SHORTS

Address offset: 0x200

Shortcut register

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				L K J I H G F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0_CLEAR			Shortcut between COMPARE[0] event and CLEAR task
					See *EVENTS_COMPARE[0]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
B	RW	COMPARE1_CLEAR			Shortcut between COMPARE[1] event and CLEAR task
					See *EVENTS_COMPARE[1]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
C	RW	COMPARE2_CLEAR			Shortcut between COMPARE[2] event and CLEAR task
					See *EVENTS_COMPARE[2]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
D	RW	COMPARE3_CLEAR			Shortcut between COMPARE[3] event and CLEAR task
					See *EVENTS_COMPARE[3]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
E	RW	COMPARE4_CLEAR			Shortcut between COMPARE[4] event and CLEAR task
					See *EVENTS_COMPARE[4]* and *TASKS_CLEAR*

This register can associate TASK with EVENT. For example, COMPARE[n]_CLEAR can clear the count value of the internal counter when the internal counter is equal to CC[n].

24.5.2 INTENSET

Address offset: 0x304

Enable interrupt

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0			Write '1' to Enable interrupt for COMPARE[0] event
					See *EVENTS_COMPARE[0]*
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	COMPARE1			Write '1' to Enable interrupt for COMPARE[1] event
					See *EVENTS_COMPARE[1]*
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	COMPARE2			Write '1' to Enable interrupt for COMPARE[2] event

Interrupts allow the setting of registers. When set to "1", the corresponding COMPARE [n] event produces a COMPARE [n] interrupt.

24.5.3 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0			Write '1' to Disable interrupt for COMPARE[0] event
					See *EVENTS_COMPARE[0]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	COMPARE1			Write '1' to Disable interrupt for COMPARE[1] event
					See *EVENTS_COMPARE[1]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	COMPARE2			Write '1' to Disable interrupt for COMPARE[2] event
					See *EVENTS_COMPARE[2]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	COMPARE3			Write '1' to Disable interrupt for COMPARE[3] event
					See *EVENTS_COMPARE[3]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
E	RW	COMPARE4			Write '1' to Disable interrupt for COMPARE[4] event
					See *EVENTS_COMPARE[4]*
			Clear	1	Disable

Close COMPARE [n] interrupt.

24.5.4 MODE

Address offset: 0x504

Timer mode selection

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				A A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	MODE			Timer mode
			Timer	0	Select Timer mode
			Counter	1	Select Counter mode Deprecated
			LowPowerCounter	2	Select Low Power Counter mode

Timing/counting mode setting.

24.5.5 BITMODE

Address offset: 0x508

Configure the number of bits used by the TIMER

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				A A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	BITMODE			Timer bit width
			16Bit	0	16 bit timer bit width
			08Bit	1	8 bit timer bit width
			24Bit	2	24 bit timer bit width
			32Bit	3	32 bit timer bit width

TIMER digit setting.

24.5.6 PRESCALER

Address offset: 0x510

Timer prescaler register

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id	A A A
Reset 0x00000004	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Id	RW	Field	Value Id	Value	Description
A	RW	PRESCALER		[0..9]	Prescaler value

Pre-dividing register.

24.5.8 CC[1]

Address offset: 0x544

Capture/Compare register 1

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id	A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Id	RW	Field	Value Id	Value	Description
A	RW	CC			Capture/Compare value Only the number of bits indicated by BITMODE will be used by the TIMER.

Capture/Compare Register.

5.3. Design of the program

Here’re the steps we use TIMER:

1) Set the working mode
2) Setting pre-dividing frequency (counter is not applicable)
3) Set the value of the CC[n] register
4) Enable interruption (interrupt mode)
5) Enable the START task
6) COMPARE EVENT arrives, clears the value of the internal counter, clears the interrupt (interrupt mode).

Revision History

Version	Date	Explain	Operator
1.0	2019-3-5	Initial Version	A11

About us

Website: www.ebyte.com Sales: info@cdebyte.com Support: support@cdebyte.com
Tel: +86-28-61399028 Ext. 812 Fax: +86-28-64146160
Address: Innovation Center B333~D347, 4# XI-XIN road,High-tech district (west), Chengdu, Sichuan, China

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.