

## Application note

---

### N32H47X\_48X series BOOT Jump application note

---

#### Introduction

N32H47X\_48X series MCU embedded boot program (BOOT), stored in System Memory, used to reprogram user program (Main FLASH) through USART3 or USB-FS interface (full speed USB device, DFU protocol).

NSING technologies MCU series products provide a variety of starting mode, can be selected by BOOT0 pin and option byte configuration. In practice, the MCU is usually set to Main Flash boot mode. If you want to use the embedded bootstrap program, you must change the MCU to System Memory boot mode and then power it on again. For details on the startup mode, refer to the corresponding user manual.

This document describes a BOOT jump method to enable users to use the embedded bootstrap mode without changing the BOOT mode.

This document applies to the N32H47X\_48X series of NSING Technologies.

**NSING Technologies All Rights Reserved**

# Content

<b>Content.....</b>	<b>II</b>
<b>1. Hardware requirements .....</b>	<b>1</b>
<b>2. Operation method .....</b>	<b>1</b>
2.1 Parameters definition .....	1
2.1.1 Function pointer .....	1
2.1.2 Necessary parameters.....	1
2.2 Method of use .....	1
2.2.1 System Clock Setting.....	1
2.2.2 API functions .....	3
2.3 The sample application .....	5
2.3.1 BOOT test .....	5
<b>3. Version history.....</b>	<b>8</b>
<b>4. Notice.....</b>	<b>9</b>

## 1. Hardware requirements

Currently, bootstrap programs embedded in MCU only support USART3 or USB-FS interface, and The corresponding IO ports are PA9/PA10 (USART3) and PA11/PA12 (USB). Ensure that the port connection is available before use.

## 2. Operation method

### 2.1 Parameters definition

#### 2.1.1 Function pointer

A function pointer type must be defined in advance: `typedef void (*pFunction)(void);`

#### 2.1.2 Necessary parameters

The following parameters must be defined:

```
/* Internal SRAM config fo MCU */
```

```
#define SRAM_BASE_ADDR          (0x20000000)
```

```
#define SRAM_SIZE                (0x24000)
```

```
/* Constant for BOOT */
```

```
uint32_t BootAddr = 0x1FFF01a1, SPAddr = 0x200024a8, _mainAddr = 0x1FFF0189;    // BOOT  
V1.0 Version
```

```
#define SRAM_VECTOR_WORD_SIZE   (128)
```

```
#define SRAM_VECTOR_ADDR        (SRAM_BASE_ADDR+SRAM_SIZE-0x200)
```

Note:

- 1) `SRAM_SASE_ADDR` is the starting address of the chip SRAM, `SRAM_SIZE` is the size of SRAM and needs to be modified according to the specific chip SRAM resources used. Users must reserve an area of 0x200 bytes of SRAM for BOOT jumps;
- 2) Other parameters cannot be modified
- 3) The default parameter values are applicable to most applications and do not need to be modified.

## 2.2 Method of use

### 2.2.1 System Clock Setting

Refer to the following function to set the system clock to HSI. It is not recommended to write this function yourself. Please call it directly.

`ErrorStatus SetSysClockToHSI(void)`

```
{  
    uint32_t timeout_value = 0xFFFFFFFF;  
    ErrorStatus ClockStatus;  
  
    RCC_DeInit();  
  
    RCC_EnableHsi(ENABLE);  
  
    /* Wait till HSI is ready */  
    ClockStatus = RCC_WaitHsiStable();  
  
    if (ClockStatus == SUCCESS)  
    {  
        /* Enable Prefetch Buffer */  
        FLASH_PrefetchBufSet(FLASH_PrefetchBuf_EN);  
  
        /* Flash 0 wait state */  
        FLASH_SetLatency(FLASH_LATENCY_0);  
  
        /* HCLK = SYSCLK */  
        RCC_ConfigHclk(RCC_SYSCLK_DIV1);  
  
        /* PCLK2 = HCLK */  
        RCC_ConfigPclk2(RCC_HCLK_DIV1);  
  
        /* PCLK1 = HCLK */  
        RCC_ConfigPclk1(RCC_HCLK_DIV1);  
  
        /* Restore HSI to PLL frequency division as default value */  
        RCC->PLLCTRL |= 0x00040000;  
  
        /* Select HSI as system clock source */
```

```
RCC_ConfigSysclk(RCC_SYSCLK_SRC_HSI);

/* Wait till HSI is used as system clock source */
while (RCC_GetSysclkSrc() != RCC_CFG_SCLKSTS_HSI)
{
    if ((timeout_value--) == 0)
    {
        return ERROR;
    }
}
else
{
    /* HSI fails */
    return ERROR;
}
return SUCCESS;
}
```

### 2.2.2 API functions

By calling the following API (Jump\_To\_BOOT), the MCU jumps directly to the bootstrap program (BOOT)

```
void Jump_To_BOOT(void)
{
    uint8_t value1[8] = {0x00}, value2 = 8;
    uint32_t i, *pVec;

    /* Init vector */
    pVec = (uint32_t *)SRAM_VECTOR_ADDR;
    for(i=0; i<SRAM_VECTOR_WORD_SIZE; i++)
        pVec[i] = 0;

    /* clean up ICache*/
}
```

```
FLASH_iCacheCmd(DISABLE);
FLASH_iCacheRST();

/* Get BOOT Version */
Get_NVR(0x1FFF37A8,value1,value2);

if((value1[0] & 0xFF) == 0x11)    //BOOT V1.1 Version
{
    SPAddr = 0x200024C0;
    pVec[SysTick_IRQn+16]          = 0x1FFF2F15;
    pVec[USART3_IRQn+16]           = 0x1FFF2F95;
    pVec[USB_FS_LP_IRQn+16]        = 0x1FFF3095;
}
else                               //BOOT V1.0 Version
{
    pVec[SysTick_IRQn+16]          = 0x1FFF2d31;
    pVec[USART3_IRQn+16]           = 0x1FFF2db1;
    pVec[USB_FS_LP_IRQn+16]        = 0x1FFF2eb1;
}

/* Disable all interrupt */
__disable_irq();

/* Config IWDG */
IWDG_ReloadKey();
IWDG_WriteConfig(IWDG_WRITE_DISABLE);
IWDG_SetPrescalerDiv(IWDG_PRESCALER_DIV256);

/* Config system clock with HSI */
SetSysClockToHSI();

/* Set JumpBoot addr */
pFunction JumpBoot = (pFunction)BootAddr;
```

```
/* Initialize Stack Pointer */
__set_MSP(SPAddr);

/* Enable interrupt */
__enable_irq();

/* Initialize vector table */
SCB->VTOR = SRAM_VECTOR_ADDR;

/* Jump to BOOT */
JumpBoot();
}
```

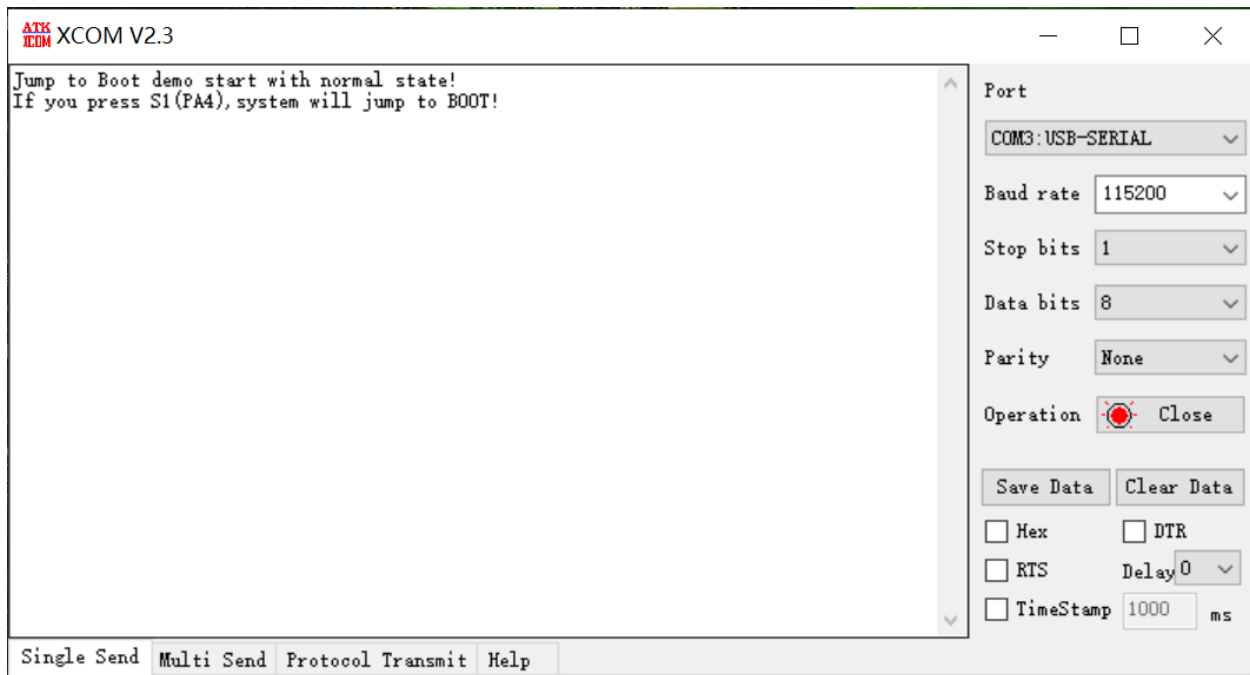
## 2.3 The sample application

With reference to the sample software package JumpToBOOT, it demonstrates how to jump to BOOT. After the jump is successful, the program can be updated through UART3 interface.

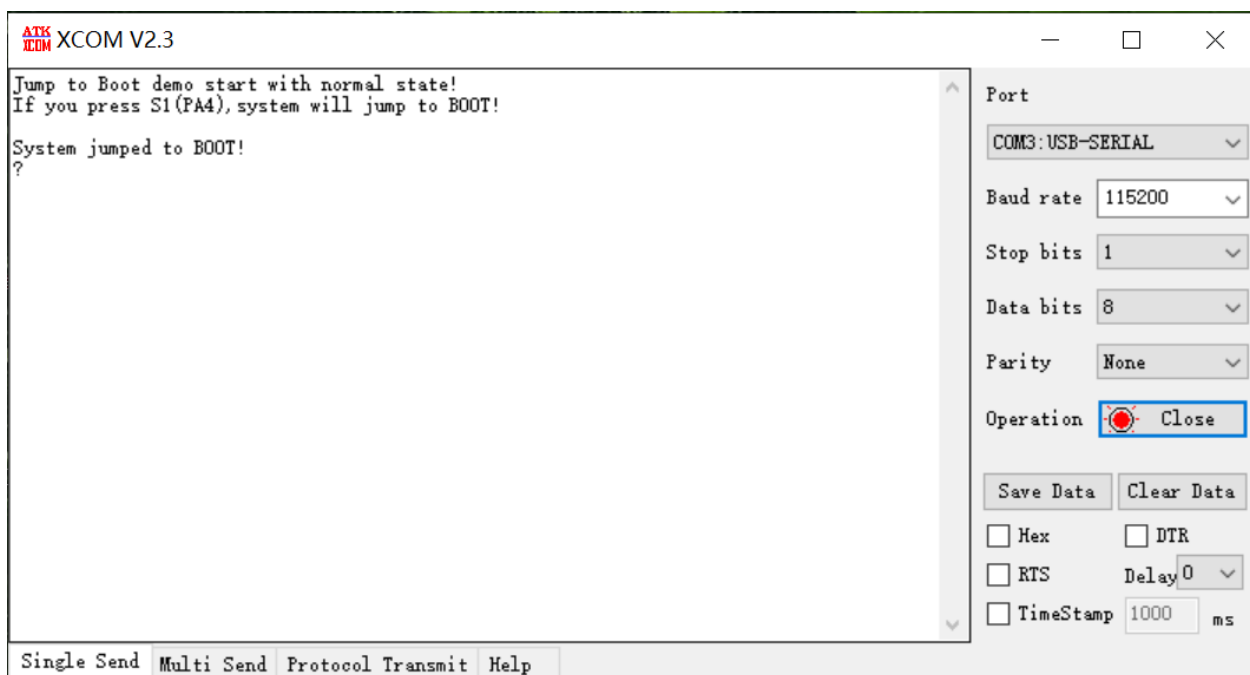
### 2.3.1 BOOT test

Based on N32H487ZEL7-STB V1.0, the test process is demonstrated.

1. Under KEIL, change the chip model to N32H487ZEL7. After compiling, burn it to the development board. Connect PC through USB cable, turn on the power supply, and check the prompt information through serial port tool on PC.



2. Open the serial port in the serial port tool and press KEY1 to switch to BOOT.



3. Close the serial port again in the serial port tool (If not closed, the download tool will show the serial port failed to open), and the connection is successful through the BOOT download tool, as shown in the following figure.

**Nations MCU Download Tool V1.3.4**

Model File Operations Offline downloader Language Help

**Status**

```

The device is connected!
Read chip information successfully!
Chip Model:N32H487 Series
Boot command set version:V1.0
Boot Subversion:V1.0
Flash size:512KB
UCID:0x36081114055033504E353130300D362B
UID:0x3608115033504E35310D362B
The current chip is in read protection L0

```

**Configuration**

Interface: USART Baud rate: 115200

Device: COM7 Disconnect

**Download**

Erase Mode: Erase by file

Start address:0x 08000000

Download File: Browse

Download

- ☐ The front cross page does not erase the current page
- ☐ The behind cross page does not erase the current page
- ☐ Unlock the read protection L1 before downloading
- ☐ Enable reading protection L1 after downloading
- ☐ Enable reading protection L2 after downloading

Clear Recount Download Hide<<

Copyright (C) 2020-2025 Nations Technologies Inc.

Total:2

Pass:2

Fail:0

2025-01-02 17:03:13

### 3. Version history

Version	Date	Modify
V1.0.0	2025-01-02	Create a document
V1.1.0	2025-09-16	Put the example project into the SDK

## 4. Notice

This document is the exclusive property of NSING TECHNOLOGIES PTE. LTD. (Hereinafter referred to as NSING). This document, and the product of NSING described herein (Hereinafter referred to as the Product) are owned by NSING under the laws and treaties of Republic of Singapore and other applicable jurisdictions worldwide. The intellectual properties of the product belong to NSING Technologies Inc. and NSING Technologies Inc. does not grant any third party any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only. NSING reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NSING and obtain the latest version of this document before placing orders. Although NSING has attempted to provide accurate and reliable information, NSING assumes no responsibility for the accuracy and reliability of this document. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NSING be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product. NSING Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, Insecure Usage'. Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to supporter sustain life. All Insecure Usage shall be made at user's risk. User shall indemnify NSING and hold NSING harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage Any express or implied warranty with regard to this document or the Product, including, but not limited to. The warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law. Unless otherwise explicitly permitted by NSING, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.