

## Application note

---

### N32G05x series BOOT Jump application note

---

#### Introduction

N32G05x series MCU embedded boot program (BOOT), stored in System Memory, used to reprogram user program (Main FLASH) through UART1 interface.

Nations 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 N32G05x series of Nations Technologies.

**Nations 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 .....	2
2.3 The sample application .....	3
2.3.1 BOOT test .....	4
<b>3. Version history</b> .....	<b>6</b>
<b>4. Notice</b> .....	<b>7</b>

## 1. Hardware requirements

Currently, bootstrap programs embedded in MCU only support UART1 interface, and corresponding IO ports are PA9/PA10 by default, and can be configured as PB10/PB11, PD10/PD11, PA2/PA3 by option byte USER6[1:0]. 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:

```
uint32_t BootAddr = 0x1FFF0025;
```

```
uint32_t SPAddr = 0x20000660;
```

Note:

1) 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 functions to set the system clock to 64MHz and use HSI+PLL as the clock source.

```
void SetSysClock_HSI_PLL(void)
```

```
{
```

```
    /* It is necessary to initialize the RCC peripheral to the reset state.*/
```

```
    RCC_DeInit();
```

```
    /* Enable HSI    */
```

```
    RCC_EnableHsi(ENABLE);
```

```
    while (RCC_WaitHsiStable() != SUCCESS)
```

```
    {
```

```
        /* If HSI failed to start-up,the clock configuration must be wrong.
```

```
        User can add some code here to dela with this problem    */
```

```

}

/* Sets the code latency value */
FLASH_SetLatency(FLASH_LATENCY_2);

/* AHB prescaler factor set to 1,HCLK = SYSCLK = 64M */
RCC_ConfigHclk(RCC_SYSCLK_DIV1);
/* APB2 prescaler factor set to 1,PCLK2 = HCLK = 64M */
RCC_ConfigPclk2(RCC_HCLK_DIV1);
/* APB1 prescaler factor set to 2,PCLK1 = HCLK/2 = 32M */
RCC_ConfigPclk1(RCC_HCLK_DIV2);

/* Config PLL */
RCC_ConfigPll(RCC_PLL_SRC_HSI,RCC_PLL_MUL_16,RCC_PLL_PRE_2,
RCC_PLOUT_DIV_1);

/* Enable PLL */
RCC_EnablePll(ENABLE);
/* Wait till PLL is ready */
while ((RCC->CTRL & RCC_CTRL_PLLRDF) == 0)
{
}
/* Select PLL as system clock source */
RCC_ConfigSysclk(RCC_SYSCLK_SRC_PLLCLK);

/* Wait till PLL is used as system clock source */
while (RCC_GetSysclkSrc() != RCC_CFG_SCLKSTS_PLL);
}

```

### 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)
{
    /* Disable all interrupt */
    __disable_irq();

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

    /* Config system clock as 64M with HSI and PLL */
    SetSysClock_HSI_PLL();

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

    /* Initalize Stack Pointer */
    __set_MSP(SPAddr);

    /* Enable interrupt */
    __enable_irq();

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

## 2.3 The sample application

With reference to the sample software package Nations.N32G05x\_bootjump, it demonstrates how to jump to BOOT. After the jump is successful, the program can be updated through UART1 interface.

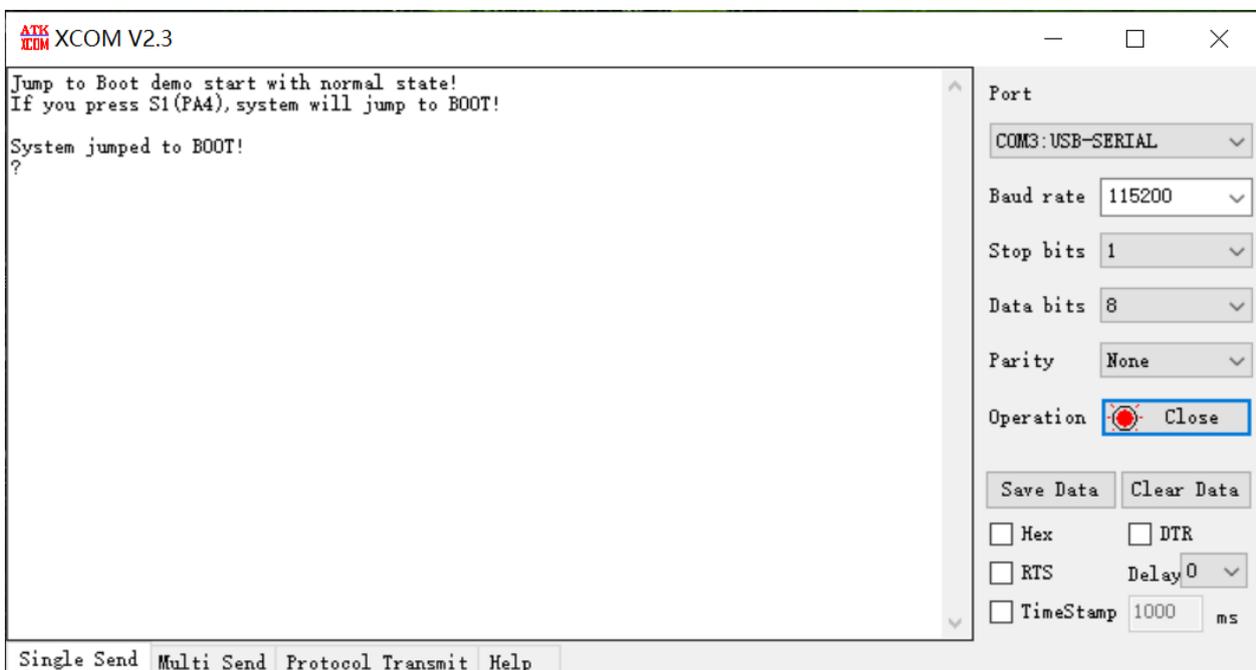
### 2.3.1 BOOT test

Based on N32G05XRBQ7-STB V1.0, the test process is demonstrated.

1. Under KEIL, change the chip model to N32G05xRB. 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.

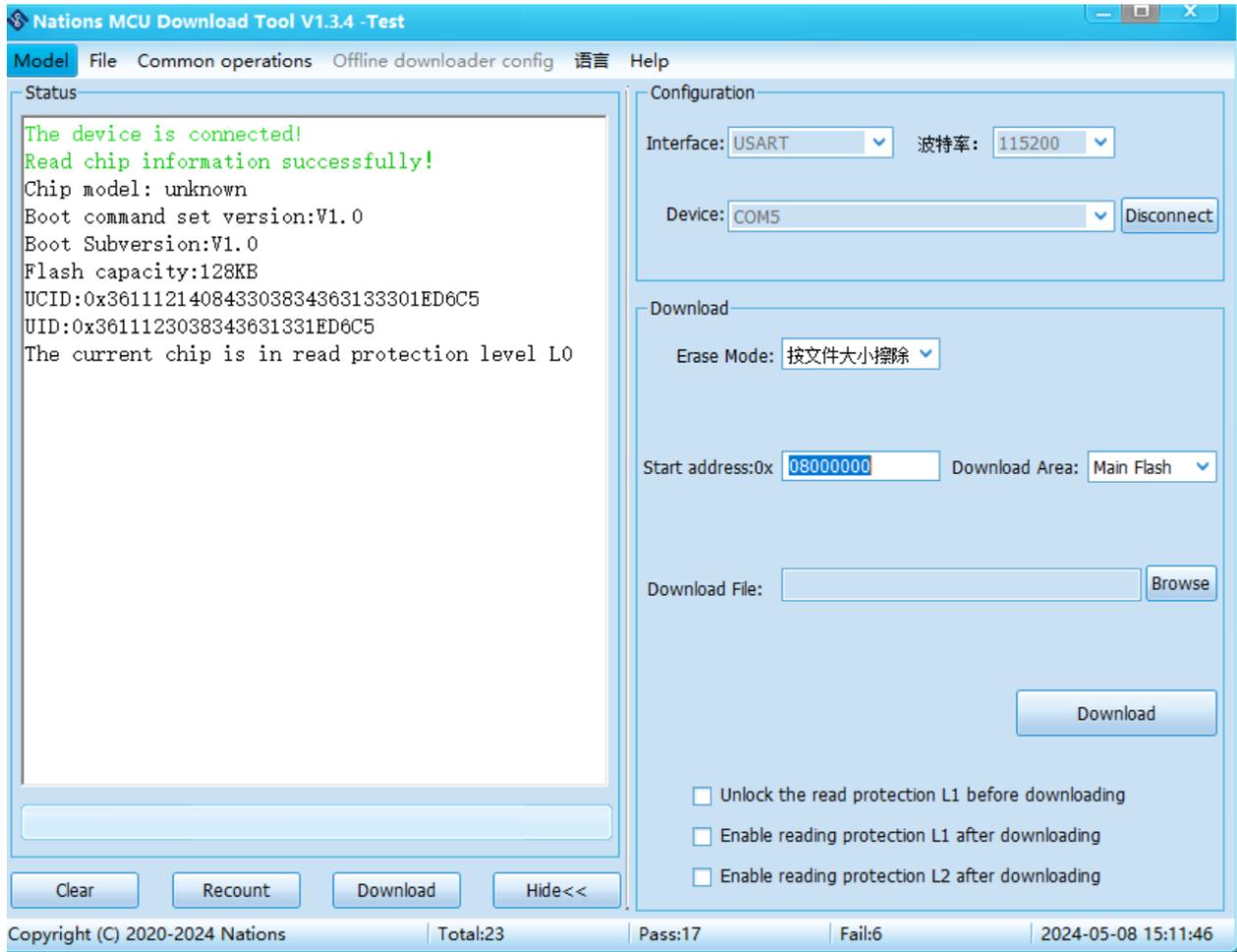


2. Open the serial port in the serial port tool and press KEY1 to switch to BOOT.



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.



### 3. Version history

<b>Version</b>	<b>Date</b>	<b>Modify</b>
<b>V1.0.0</b>	<b>2024-05-14</b>	<b>Create a document</b>

## 4. Notice

This document is the exclusive property of Nations Technologies Inc. (Hereinafter referred to as NATIONS). This document, and the product of NATIONS described herein (Hereinafter referred to as the Product) are owned by NATIONS under the laws and treaties of the People's Republic of China and other applicable jurisdictions worldwide.

NATIONS does not grant 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.

NATIONS reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NATIONS and obtain the latest version of this document before placing orders.

Although NATIONS has attempted to provide accurate and reliable information, NATIONS 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 NATIONS 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.

NATIONS 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 support or sustain life.

All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS 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 NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.