

Application note

N32G45x&N32G4FR&N32WB452 series BOOT Jump application note

Introduction

N32G45x, N32G4FR, N32WB452 series MCU embedded boot program (BOOT), stored in System Memory, used to reprogram user program (FLASH) through USART1 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, BOOT1 pin. In practice, the MCU is usually set to Flash boot mode (BOOT0=0). If you want to use the embedded bootstrap program, you must change the MCU to System Memory boot mode (BOOT0=1, BOOT1=0) 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 N32G452 series, N32G455 series, N32G457 series, N32G4FR series, and N32WB452 series of NSING Technologies.

NSING Technologies All Rights Reserved

Content

Content.....	II
1. Hardware requirements	1
2. Operation method	1
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 The BOOT V2.1 test	5
2.3.2 The BOOT V2.2 test	7
3. Version history.....	10
4. Notice.....	11

1. Hardware requirements

Currently, bootstrap programs embedded in MCU only support USART1 or USB-FS interface, and the corresponding IO ports are PA9/PA10 (USART1) and PA11/PA12 (USB) respectively. 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:

```
#define SRAM_BASE_ADDR          (0x20000000)
#define SRAM_SIZE                (0x20000)
#define SRAM_VECTOR_WORD_SIZE  (64)
#define SRAM_VECTOR_ADDR        (SRAM_BASE_ADDR+SRAM_SIZE-0x100)
#define BOOT_MARK1_ADDR         (0x1FFFF2D0)    /* BOOT NVIC */
#define BOOT_MARK2_ADDR         (0x1FFFF288)    /* BOOT Code */
#define BOOT_MARK3_ADDR         (0x40024C00)
```

Note:

- 1) `SRAM_BASE_ADDR` is the start ADDRESS of the SRAM, and `SRAM_SIZE` is the size of the SRAM, which needs to be changed based on the chip SRAM resources. The user must reserve the last 0x100 bytes of SRAM for BOOT jump.
- 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 functions to set the system clock to 72MHz 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_GetFlagStatus(RCC_FLAG_HSIRD) == RESET)
{
    /* If HSI failed to start-up,the clock configuration must be wrong.
       User can add some code here to dela with this problem */
}

/* Enable ex mode */
RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR,ENABLE);
PWR->CTRL3 |= (uint32_t)0x00000001;

/* Enable ICACHE and Prefetch Buffer */
FLASH_SetLatency(FLASH_LATENCY_2);
FLASH_PrefetchBufSet(FLASH_PrefetchBuf_EN);
FLASH_iCacheCmd(FLASH_iCache_EN);

/* AHB prescaler factor set to 1,HCLK = SYSCLK = 72M */
RCC_ConfigHclk(RCC_SYSCLK_DIV1);
/* APB2 prescaler factor set to 1,PCLK2 = HCLK/1 = 72M */
RCC_ConfigPclk2(RCC_HCLK_DIV1);
/* APB1 prescaler factor set to 2,PCLK1 = HCLK/2 = 36M */
RCC_ConfigPclk1(RCC_HCLK_DIV2);

/* Config PLL */
RCC_ConfigPll(RCC_PLL_SRC_HSI_DIV2, RCC_PLL_MUL_18);

/* Enable PLL */
RCC_EnablePll(ENABLE);
while (RCC_GetFlagStatus(RCC_FLAG_PLLRD) == RESET)
{
```

```
}

/* Switch PLL clock to SYSCLK. */
RCC_ConfigSysclk(RCC_SYSCLK_SRC_PLLCLK);
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)
{
    uint32_t i,*pVec,*pMark;
    uint32_t BootAddr,SPAddr;

    /* Disable all interrupt */
    __disable_irq();

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

    /* Config MMU */
    pMark = (uint32_t*)(BOOT_MARK3_ADDR);
    *pMark = (uint32_t)0x00000011;

    /* Config system clock as 72M with HSI and PLL */
    SetSysClock_HSI_PLL();
}
```

```
/* Reset peripheral used by boot */
USART_DeInit(USART1);
GPIO_DeInit(GPIOA);
RCC_EnableAPB1PeriphReset(RCC_APB1_PERIPH_USB, ENABLE);
RCC_EnableAPB1PeriphReset(RCC_APB1_PERIPH_USB, DISABLE);

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

/* Get SP addr */
SPAddr = *((uint32_t *)BOOT_MARK2_ADDR);

/* Get usefull fuction addr */
pMark = (uint32_t *)BOOT_MARK1_ADDR;
if(*pMark != *BOOT_V2.3 and above*)
{
    BootAddr                = pMark[0];
    pVec[SysTick_IRQn+16]    = pMark[1];
    pVec[USART1_IRQn+16]     = pMark[2];
    pVec[USB_LP_CAN1_RX0_IRQn+16] = pMark[3];
    pVec[RTC_IRQn+16]        = pMark[4];
}
else
{
    if(SPAddr != 0xFFFFFFFF) /* BOOT V2.2 */
    {
        pVec[SysTick_IRQn+16]    = 0x1FFF0A67;
        pVec[USART1_IRQn+16]     = 0x1FFF0A9F;
        pVec[USB_LP_CAN1_RX0_IRQn+16] = 0x1FFF0ACF;
        pVec[RTC_IRQn+16]        = 0x1FFF0AD3;
```

```

        BootAddr                = 0x1FFF00D9;
    }
    The else / * BOOT V2.1 * /
    {
        pVec[SysTick_IRQn+16]    = 0x1FFF10D7;
        pVec[USART1_IRQn+16]     = 0x1FFF115D;
        pVec[USB_LP_CAN1_RX0_IRQn+16] = 0x1FFF117F;
        pVec[RTC_IRQn+16]        = 0x1FFF1183;
        pVec[EXTI15_10_IRQn+16]   = 0x1FFF10ED;
        BootAddr                 = 0x1FFF0101;
        SPAddr                   = 0x20008690;
    }
}

/* Enable interrupt */
__enable_irq();

/* Jump to boot */
pFunction JumpBoot = (pFunction)BootAddr;
__set_MSP(SPAddr);
SCB->VTOR = SRAM_VECTOR_ADDR;
JumpBoot();
}

```

2.3 The sample application

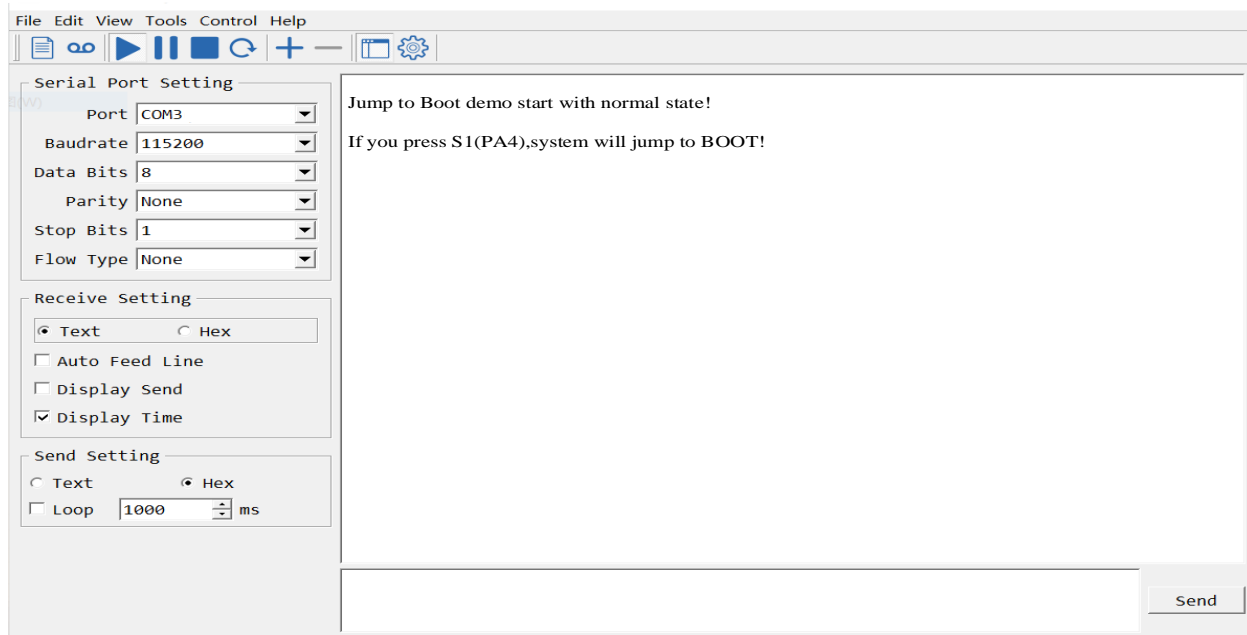
With reference to the sample software package Nations.n32G45X_Bootjump, it demonstrates how to jump to BOOT. After the jump is successful, the program can be updated through USART1 or USB interface. Pass the test on BOOT V2.1 and V2.2.

2.3.1 The BOOT V2.1 test

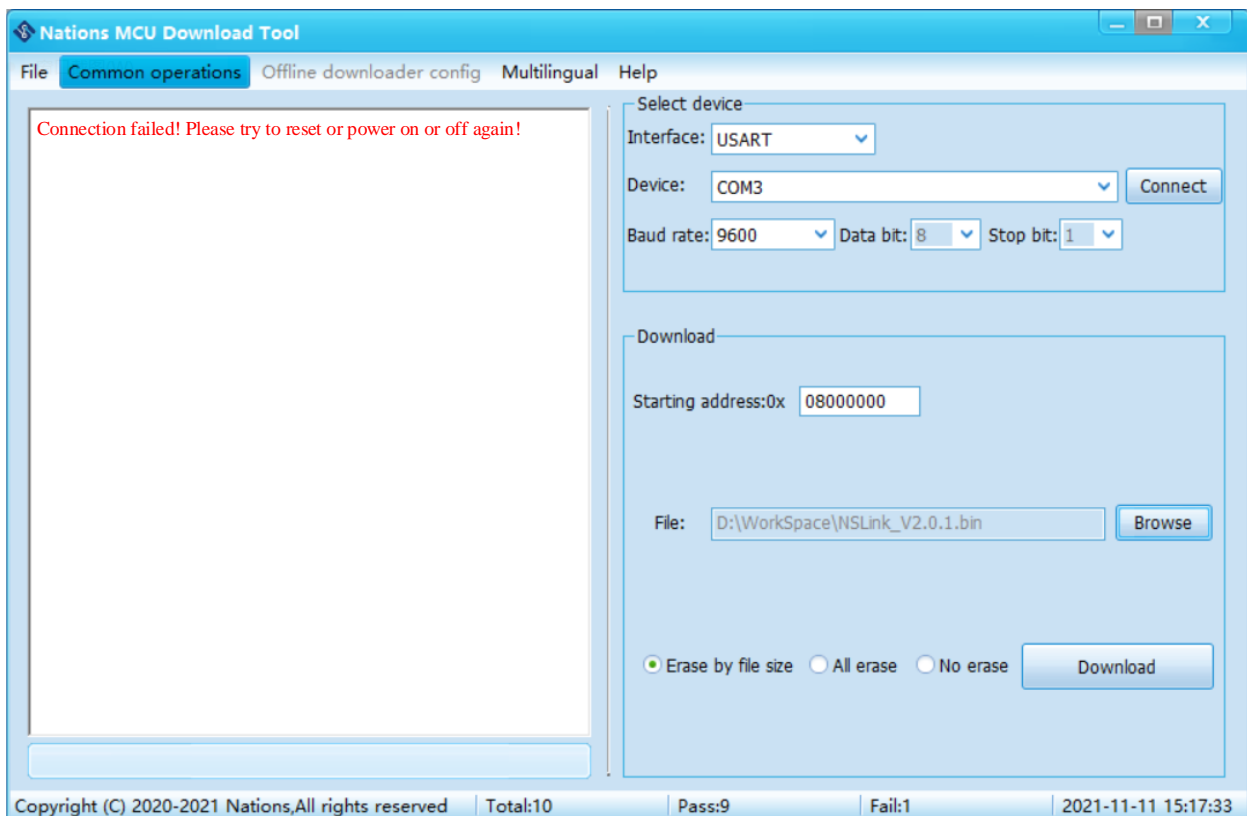
Based on N32G45XCL-STB, the test process is demonstrated.

1. Under KEIL, change the chip model to N32G455CCL7. After compiling, burn it to the development board. Connect PC and J4 through USB cable, turn on the power supply, and check

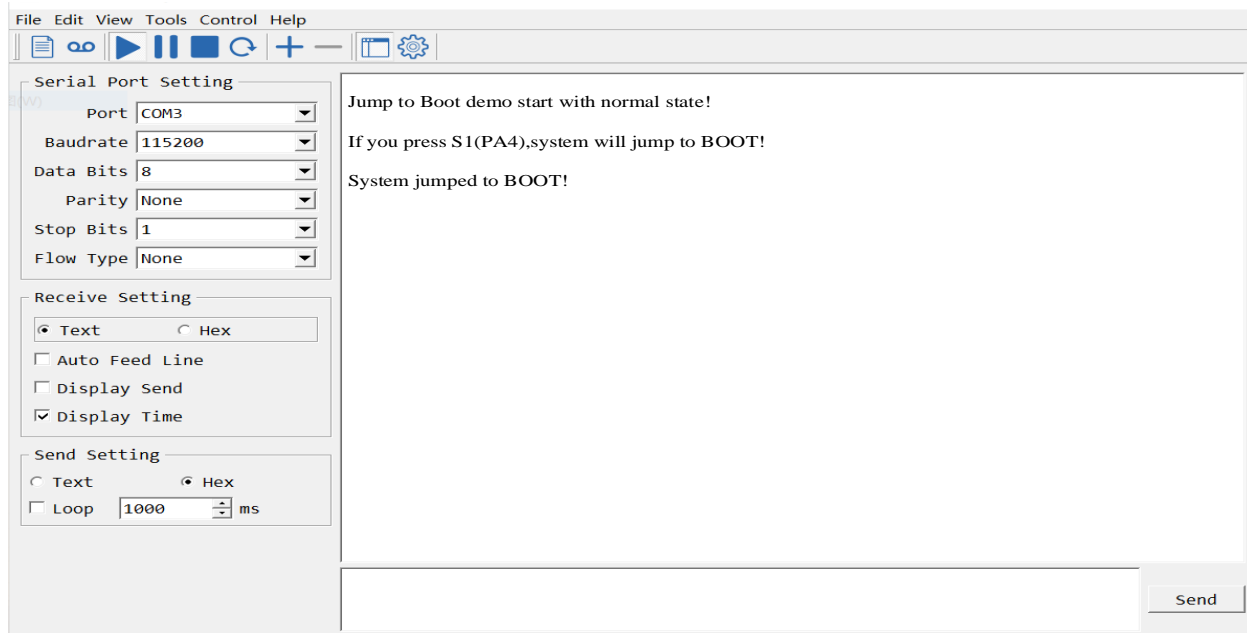
the prompt information through serial port tool on PC.



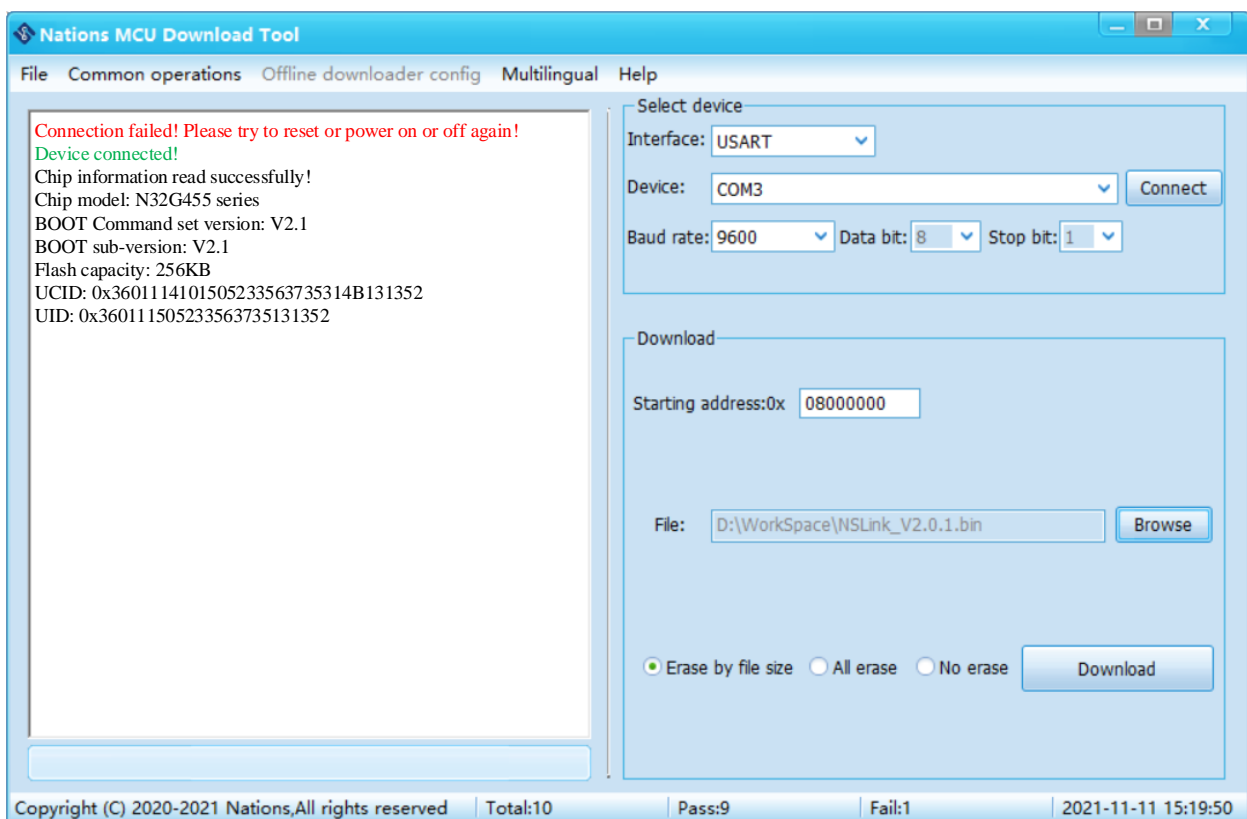
2. Close the serial port in the serial port Tool, open the BOOT download tool “Nations MCU Download Tool”, and select the corresponding serial port connection. A message is displayed indicating that the connection fails.



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



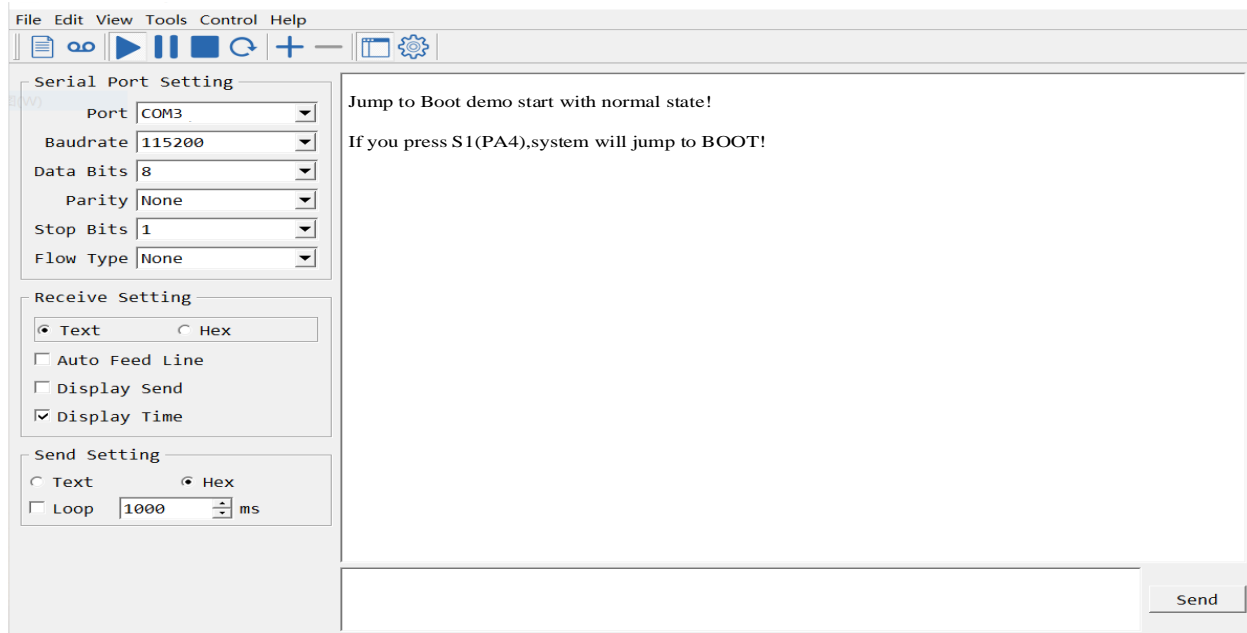
4. Close the serial port again in the serial port tool, and the connection is successful through the BOOT download tool, as shown in the following figure.



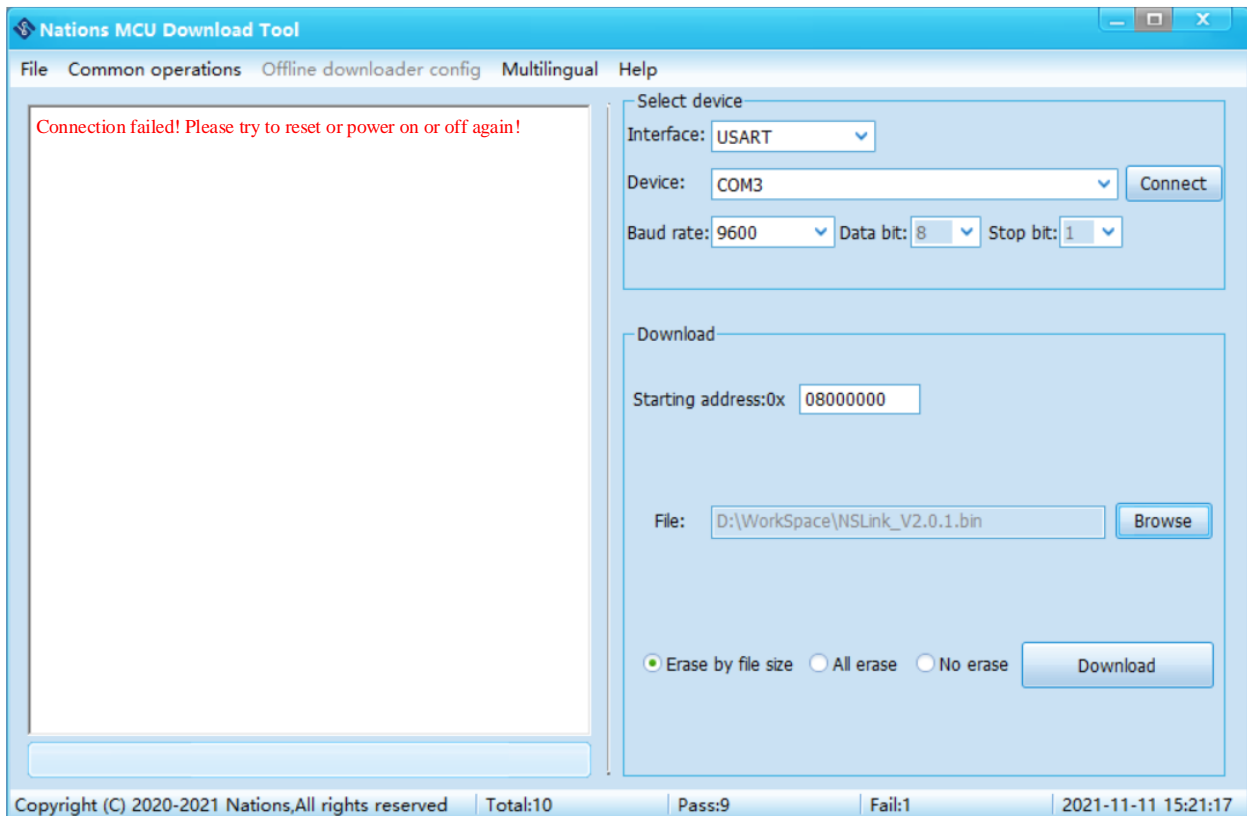
2.3.2 The BOOT V2.2 test

Based on N32G4FRKQ-STB, the test process is demonstrated.

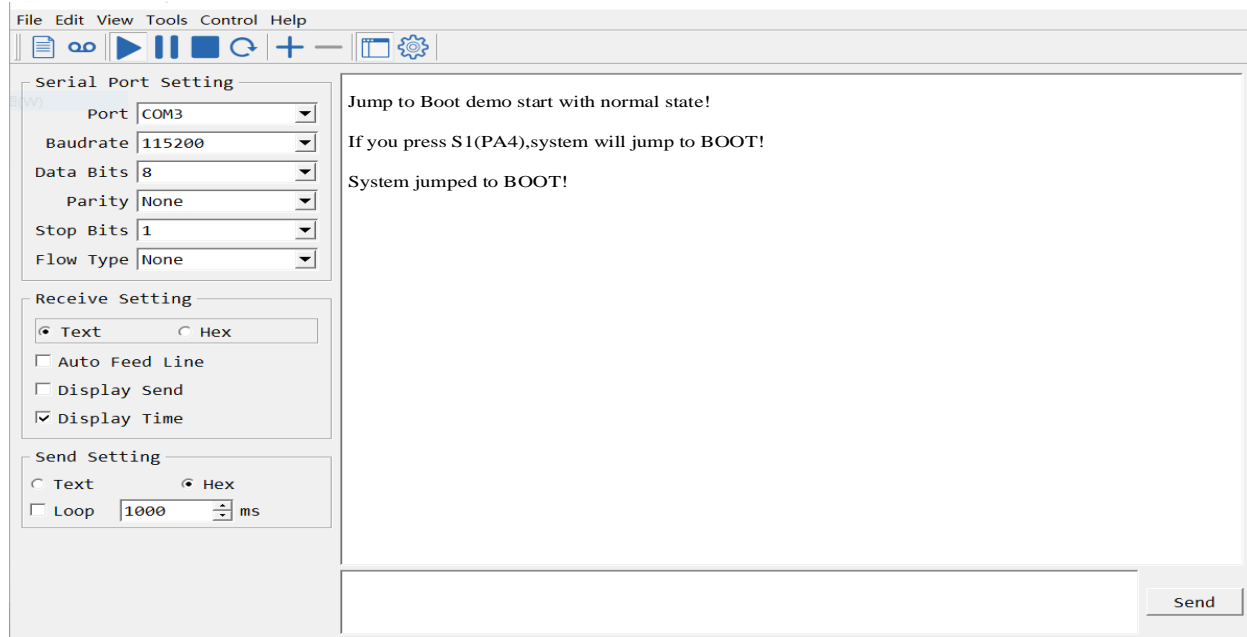
1. Under KEIL, change the chip model to N32G4FRKQL7, After compiling, burn it to the development board, connect PC and J4 through USB cable, turn on the power supply, you can view the prompt information through serial port tool on PC.



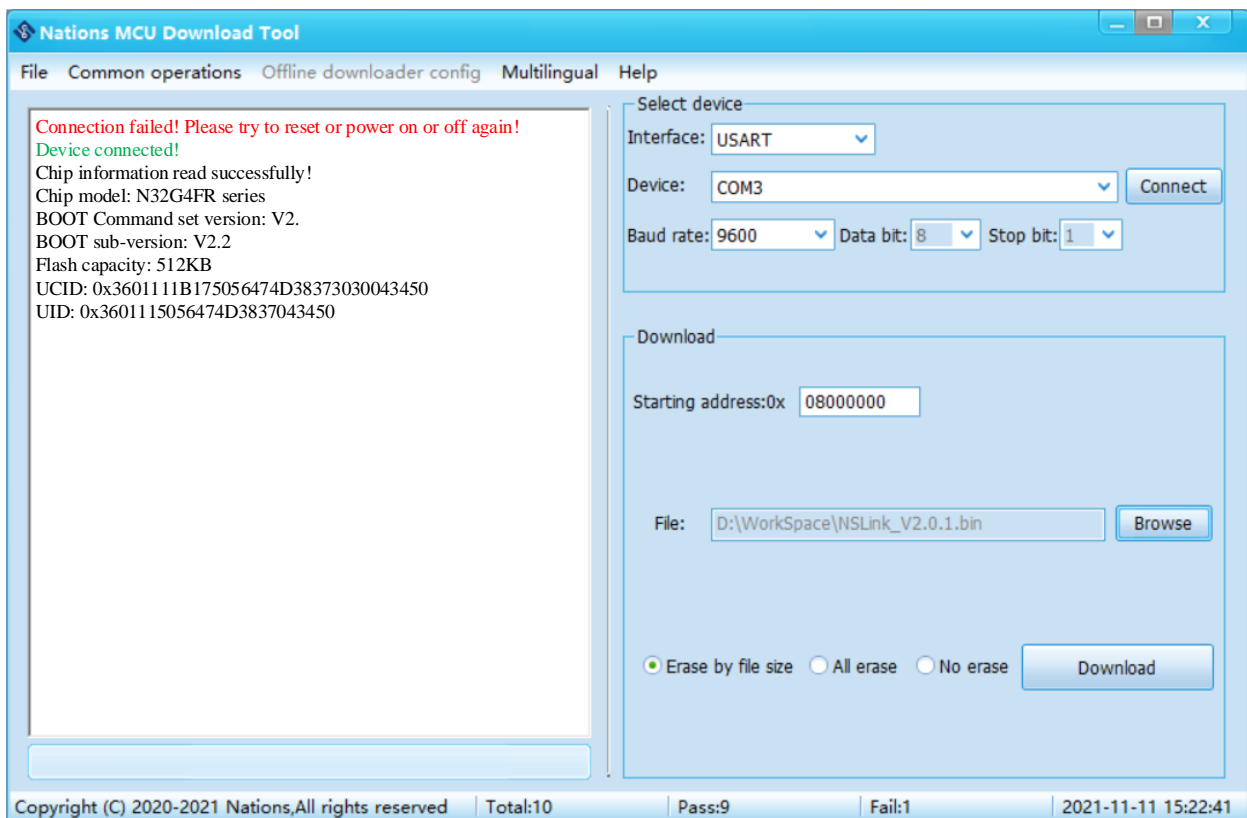
2. Close the serial port in the serial port Tool, open the BOOT download tool “Nations MCU Download Tool”, and select the corresponding serial port connection. A message is displayed indicating that the connection fails.



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



4. Close the serial port again in the serial port tool, and the connection is successful through the BOOT download tool, as shown in the following figure.



3. Version history

Version	Date	Modify
V1.0	2021-2-6	Create a document
V1.1	2021-3-4	Optimize routines and add test process demonstrations.
V1.2	2025-5-30	Introduction description correction.

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.