

应用笔记

N32G45x&N32G4FR&N32WB452系列BOOT跳转应用笔记

简介

N32G45x、N32G4FR、N32WB452 系列 MCU 内嵌有自举程序 (BOOT)，存放在 System Memory 内，用于通过 USART1 或都 USB-FS 接口 (全速 USB 设备，DFU 协议) 对用户程序 (FLASH) 进行重新编程。

国民技术 MCU 系列产品提供多种启动模式，可通过 BOOT0、BOOT1 引脚来选择。在实际应用中，MCU 通常设置为 Flash 启动模式 (BOOT0=0)。如果要使用内嵌的自举程序，必须将 MCU 修改为 System Memory 启动模式 (BOOT0=1, BOOT1=0) 后重新上电。有关启动模式的详细说明请参照对应的用户手册。

本文档介绍了一种 BOOT 跳转方法，便于用户在正常使用中不修改启动模式也能使用内嵌的自举模式。

本文档适用于国民技术的 N32G451 系列、N32G452 系列、N32G455 系列、N32G457 系列、N32G4FR 系列、N32WB452 系列产品。

国民技术 版权所有

目录

目录	II
1. 硬件需求	1
2. 操作方法	1
2.1 参数定义.....	1
2.1.1 函数指针.....	1
2.1.2 必要参数.....	1
2.2 使用方法.....	1
2.2.1 系统时钟设置.....	1
2.2.2 API 函数	3
2.3 应用示例.....	5
2.3.1 BOOT V2.1 测试	5
2.3.2 BOOT V2.2 测试	8
3. 历史版本	10
4. 声 明	11

1. 硬件需求

目前 MCU 内嵌的自举程序仅支持 USART1 或 USB-FS 接口，对应的 IO 端口分别为 PA9/PA10 (USART1)、PA11/PA12 (USB)，使用前必须确保端口连接可用。

2. 操作方法

2.1 参数定义

2.1.1 函数指针

必须预先定义一个函数指针类型：typedef void (*pFunction)(void);

2.1.2 必要参数

必须预先定义以下几个参数：

```
#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)
```

注意：

- 1) SRAM_BASE_ADDR 是芯片 SRAM 的起始地址，SRAM_SIZE 为 SRAM 大小，需要根据具体使用的芯片 SRAM 资源修改。用户必须预留 SRAM 最后大小为 0x100 字节的区域用于 BOOT 跳转；
- 2) 其他参数不能修改；
- 3) 默认的参数值适用于大部分应用情况，不需要修改。

2.2 使用方法

2.2.1 系统时钟设置

参照下面的函数，将系统时钟设置为 72MHz，采用 HSI+PLL 作为时钟源。

```
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 函数

调用下面的 API (Jump_To_BOOT)，MCU 直接跳转到自举程序 (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 != 0xFFFFFFFF)    /*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;
    }
    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();
}

```

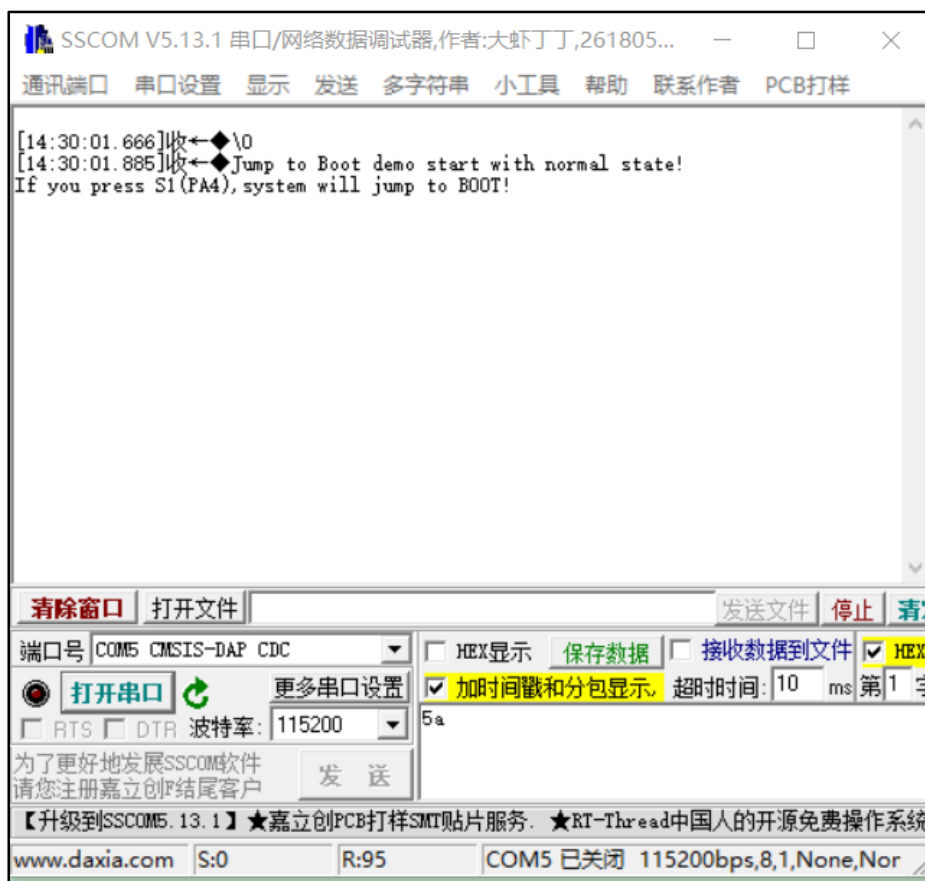
2.3 应用示例

参照示例软件包 Nations.N32G45X_bootjump，演示了如何跳转到 BOOT，跳转成功后可通过 USART1 或 USB 接口更新程序。在 BOOT V2.1 与 V2.2 上测试通过。

2.3.1 BOOT V2.1 测试

基于 N32G45XCL-STB，演示测试流程。

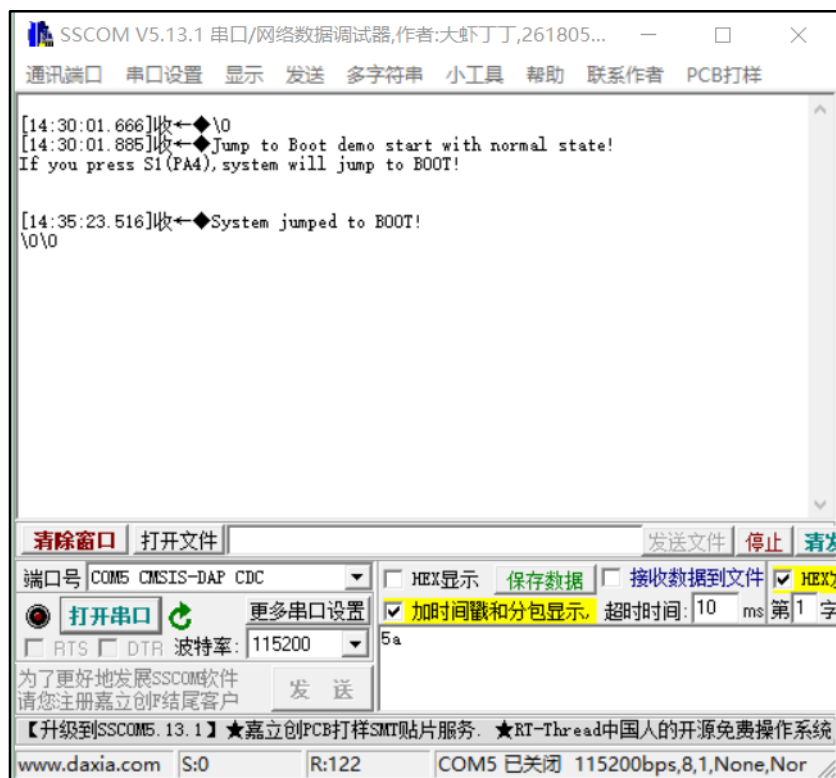
1. 在 KEIL 下将芯片型号改为 N32G455CCL7，编译后烧录到开发板，通过 USB 线连接 PC 与 J4，接通电源，可在 PC 上通过串口工具查看提示信息。



2. 在串口工具中关闭串口，打开 BOOT 下载工具“Nations MCU Download Tool”，选择对应串口连接，提示连接失败，如下图所示。



3. 在串口工具中打开串口，按下按键 KEY1，系统跳转至 BOOT。



4. 再次在串口工具中关闭串口，通过 BOOT 下载工具连接成功，如下图所示。



2.3.2 BOOT V2.2 测试

基于 N32G4FRKQ-STB，演示测试流程。

1. 在 KEIL 下将芯片型号改为 N32G4FRKQL7，编译后烧录到开发板，通过 USB 线连接 PC 与 J4，接通电源，可在 PC 上通过串口工具查看提示信息。



2. 在串口工具中关闭串口，打开 BOOT 下载工具“Nations MCU Download Tool”，选择对应串口连接，提示连接失败，如下图所示。



3. 在串口工具中打开串口，按下按键 KEY1，系统跳转至 BOOT。



4. 再次在串口工具中关闭串口，通过 BOOT 下载工具连接成功，如下图所示。



3. 历史版本

版本	日期	备注
V1.0	2021-2-6	创建文档
V1.1	2021-3-4	优化例程，增加测试过程演示。
V1.2	2023-6-20	增加 N32G451 系列

4. 声 明

国民技术股份有限公司（下称“国民技术”）对此文档拥有专属产权。依据中华人民共和国的法律、条约以及世界其他法域相适用的管辖，此文档及其中描述的国民技术产品（下称“产品”）为公司所有。

国民技术在此并未授予专利权、著作权、商标权或其他任何知识产权许可。所提到或引用的第三方名称或品牌（如有）仅用作区别之目的。

国民技术保留随时变更、订正、增强、修改和改良此文档的权利，恕不另行通知。请使用者在下单购买前联系国民技术获取此文档的最新版本。

国民技术竭力提供准确可信的资讯，但即便如此，并不推定国民技术对此文档准确性和可靠性承担责任。

使用此文档信息以及生成产品时，使用者应当进行合理的设计、编程并测试其功能性和安全性，国民技术不对任何因使用此文档或本产品而产生的任何直接、间接、意外、特殊、惩罚性或衍生性损害结果承担责任。

国民技术对于产品在系统或设备中的应用效果没有任何故意或保证，如有任何应用在其发生操作不当或故障情况下，有可能致使人员伤亡、人身伤害或严重财产损失，则此类应用被视为“不安全使用”。

不安全使用包括但不限于：外科手术设备、原子能控制仪器、飞机或宇宙飞船仪器、所有类型的安全装置以及其他旨在支持或维持生命的应用。

所有不安全使用的风险应由使用人承担，同时使用人应使国民技术免于因为这类不安全使用而导致被诉、支付费用、发生损害或承担责任时的赔偿。

对于此文档和产品的任何明示、默示之保证，包括但不限于适销性、特定用途适用性和不侵权的保证责任，国民技术可在法律允许范围内进行免责。

未经明确许可，任何人不得以任何理由对此文档的全部或部分进行使用、复制、修改、抄录和传播。