
AN_基于CAN接口的二级BOOT方案

简介

在嵌入式产品研发过程中，用户会自己开发二级 **BOOT** 代码（后面用 **IAP** 表示）升级应用程序（后面用 **APP** 表示），而用于升级的外设接口也有很多，例如 **I2C**、**USART**、**CAN**、**SPI**、**ETH**、**USB** 等等。

本文档主要针对国民技术 **MCU** 系列产品在上述应用场景，指导用户如何使用 **IAP** 示例，通过 **CAN** 外设接口实现 **IAP** 升级 **APP** 的功能。

本文档仅适应于带 **CAN** 外设接口的国民技术 **MCU** 产品，目前支持的产品系列有 **N32G43x** 系列、**N32L43x** 系列、**N32L40x** 系列产品。

目录

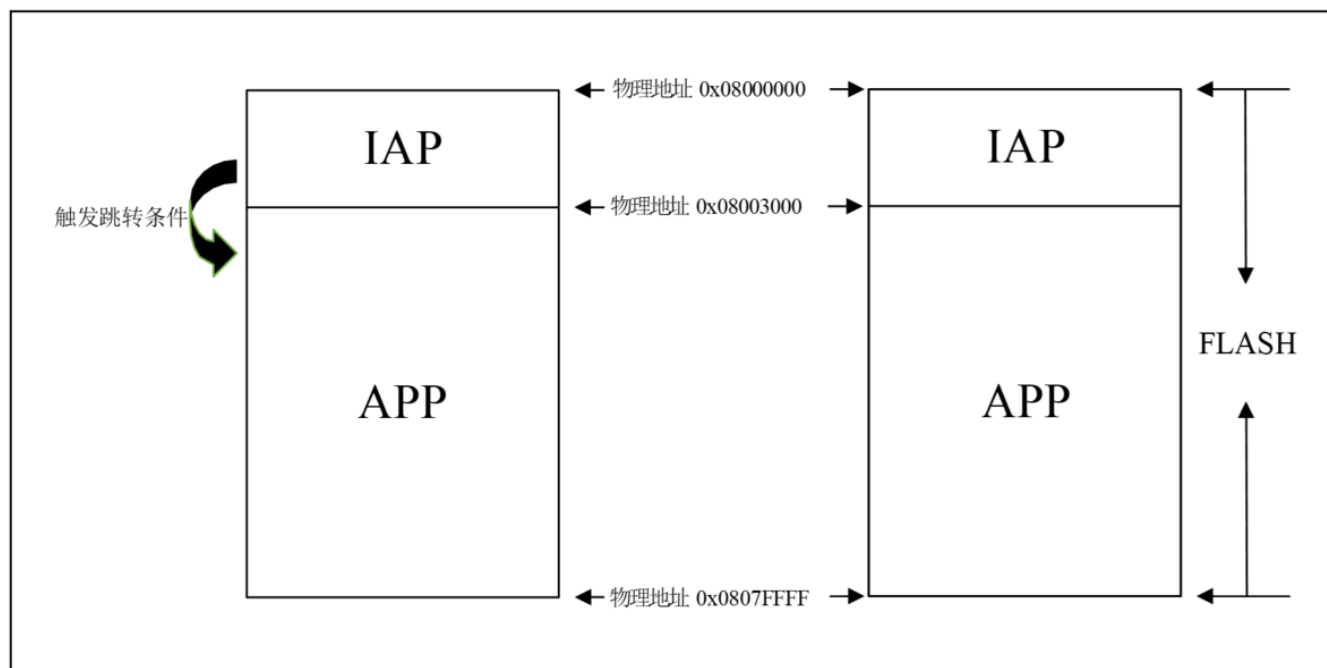
目录.....	I
1 IAP 升级 APP 实现机制.....	1
2 基于 CAN 外设接口的 IAP 例程功能说明.....	2
2.1 触发条件判断.....	2
2.2 IAP 升级模式.....	2
3 升级指令说明	4
3.1 指令及数据结构.....	4
3.2 指令说明.....	4
4 演示 IAP 工程.....	10
4.1 硬件连接.....	10
5 总结.....	12
6 历史版本.....	13
7 声明.....	14

1 IAP 升级 APP 实现机制

用户在开发一些产品时，需要对产品在后续使用时升级 APP 软件代码。这时就需要在 MCU 的 FLASH 内预先内置 IAP 代码，后续通过之前预留的外设接口下载更新 APP。

IAP 的代码一般存放在 MCU 的 FLASH 最前面的地址段，APP 的代码存放在 IAP 之后，如图 1-1 所示。

图 1-1 IAP、APP 存放地址和跳转示意图



通常会将芯片运行情况分为两种模式：

1. APP 应用模式，芯片上电后运行 IAP 代码，触发了跳转条件后跳转到 APP 执行代码；
2. IAP 升级模式，芯片上电后运行 IAP 代码，未触发跳转条件，执行 IAP 代码，用于升级 APP。

2 基于 CAN 外设接口的 IAP 例程功能说明

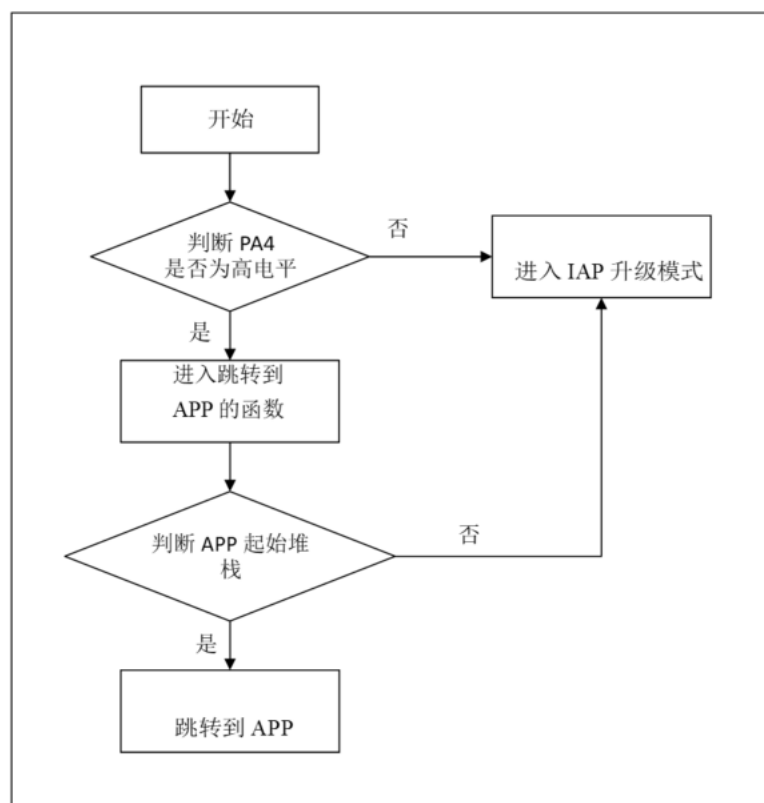
在 SDK 例程目录（Nationstech.N32G43x_Library.1.2.0\projects\n32g43x_EVAL\Applications\IAP\CAN）下打开对应的 KEIL 工程。IAP 例程功能主要分为两个部分：

- 1.判断跳转到 APP 的触发条件是否生效，并依此控制程序是否跳转到 APP；
- 2.进入 IAP 升级模式，通过 CAN 外设接口接收上位机（也可以是其他芯片）下发的指令并做出相应的回复；

2.1 触发条件判断

IAP 例程的触发条件是 GPIO PA4 的电平状态（可以通过将 N32G43x 系列最小系统开发板 N32G43XR-STB 的 PA4 接到高电平或者低电平），下图 2-1 是 PA4 不同电平状态时程序流程走向示意图。

图 2-1 PA4 不同电平状态时程序流程走向示意图

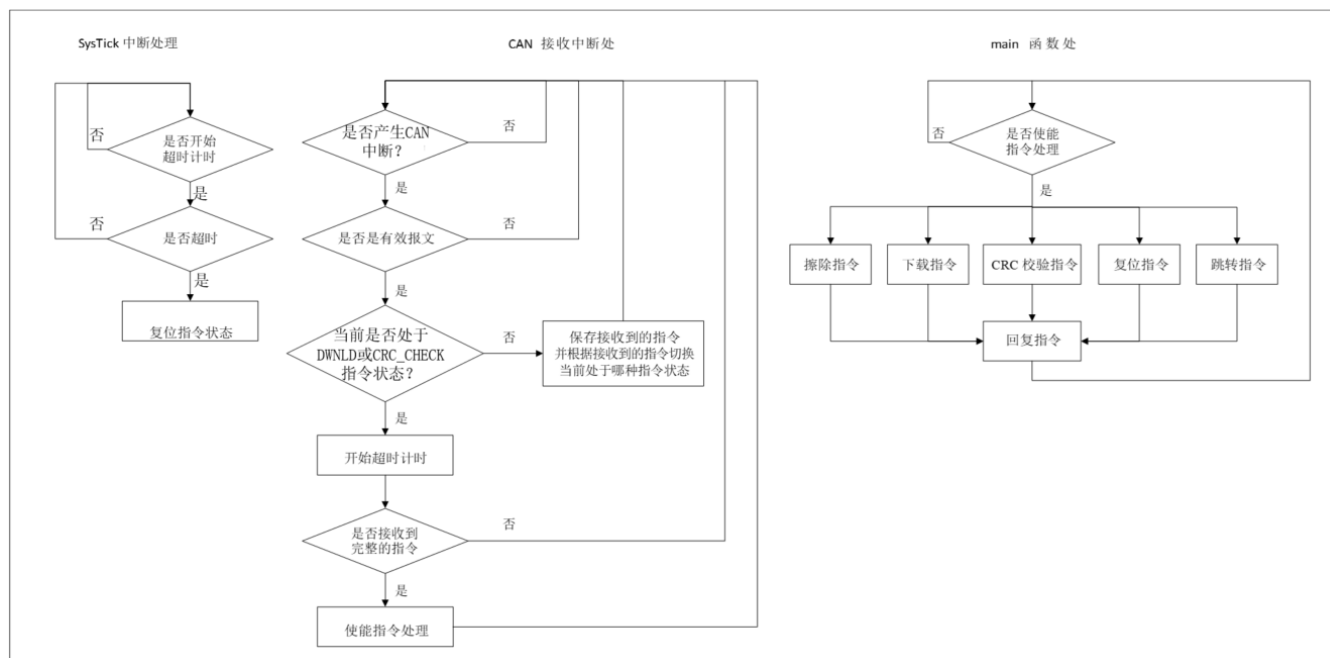


2.2 IAP 升级模式

进入 IAP 升级模式后，通过 CAN 外设接口接收升级指令和数据，升级指令详情见第 3 章节升级指令说明。由于一帧 CAN 报文最多包含 8 字节有效数据，部分指令需要接收多帧 CAN 报文才能组成完整的指令。

下图 2-2 描述了 IAP 升级模式下升级指令处理流程：

图 2-2 升级指令处理流程示意图



3 升级指令说明

下面详细说明了全部的指令数据结构和具体含义。

3.1 指令及数据结构

3.1.1 指令列表

表 3-1 指令列表

指令名称	一级指令字段	二级指令字段	指令说明
CMD_FLASH_ERASE	0x10	0x00	擦除指定范围 FLASH
CMD_FLASH_DWNLD	0x11	0x00	下载 APP 到 FLASH
CMD_DATA_CRC_CHECK	0x12	0x00	CRC 校验下载的 APP
CMD_SYS_RESET	0x13	0x00	复位系统
CMD_APP_GO	0x14	0x00	跳转到 APP 运行

3.1.2 指令数据结构

这里介绍下文阐述中的一些约定，其中，“<>”代表必须包含的字段，“()”代表根据不同指令包含的数据字段。

1、接收指令数据结构：

<CMD_H + CMD_L + LEN + Par[0] + Par[1] + Par[2] + Par[3]> + (DAT)。

CMD_H 代表一级指令字段，CMD_L 代表二级指令字段；LEN 代表接收数据长度（即 DAT 的字节长度）；

Par[0]~ Par[3]代表 4 个字节指令参数；DAT 代表接收的具体数据；

2、回复指令数据结构：

< CMD_H + CMD_L + LEN + Par[0] + Par[1] + Par[2] + Par[3]> + (DAT)。

CMD_H 代表一级指令字段，CMD_L 代表二级指令字段，回复的一、二级指令字段和对应接收的相同；LEN 代表发送数据长度（即 DAT 的字节长度）；DAT 代表回复指令的具体数据；Par[0]~ Par[1]两个字节代表向上层返回的指令执行结果，Par[2]~ Par[3]两个字节保持为保留 0；若接收到的一级指令、二级指令字段不属于指令列表中的任何指令，回复 Par[0] = 0xBB，Par[1] = 0xCC。

3.2 指令说明

3.2.1 CMD_FLASH_ERASE

以页为单位擦除 FLASH 的功能，擦除页地址编号和和页数由用户提供，擦除的 FLASH 空间不能超过整个 FLASH 空间，且至少擦除 1 个页。

接收指令：

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段：0x10							
1(CMD_L)	二级命令字段：保留							

2~3(LEN)	发送数据长度：保留
4~7(Par)	页地址编号 2 字节：0~255 页数 2 字节：1~256
(DAT)	无

- LEN 发送数据长度：0x00(LEN[0])、0x00(LEN[1])， $LEN = LEN[0] + (LEN[1] \ll 8)$ 。

- 擦除地址和范围由 Par 字段中的 4 个字节构成：

Par[0~1]：页地址编号 2 字节(0~255) 页地址编号= Par [0] + Par [1] << 8；

Par[2~3]：页数 2 字节(1~256) 页数= Par [2] + Par [3] << 8；

0 号页首地址为 0x0800_3000，以后的页地址编号加 1，首地址累加 0x800。比如：

1 号页首地址为 $0x0800_3000 + 1 * 0x800 = 0x0800_3800$

2 号页首地址为 $0x0800_3000 + 2 * 0x800 = 0x0800_4000$

整个擦除的地址范围

比如：页地址编号为 0x01，页数为 0x02

则擦除的地址范围：

$(0x0800_3000 + 1 * 0x800) \sim (0x0800_3000 + 1 * 0x800 + 2 * 0x800)$

即（页地址编号的首地址）~（页地址编号的首地址+ 页数*页的大小）

- 保留值：0x00

回复指令：

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段：0x10							
1(CMD_L)	二级命令字段：保留							
2~3(LEN)	发送数据长度：保留							
4~7(Par)	Par[0]：状态字节 1 Par[1]：状态字节 2 Par[2] ~ Par[3]：保留							
(DAT)	无							

- LEN 发送数据长度：0x00(LEN[0])、0x00(LEN[1])， $LEN = LEN[0] + (LEN[1] \ll 8)$ 。

- 状态字节(Par[0]、Par[1])根据命令执行情况分为：

1.返回成功：状态标志位(0xA0、0xB0)。

2.返回失败：状态标志位(Par[0]、Par[1])：

(1) (0xE0、0x10): 擦除 FLASH 失败;

(2) (0xE0、0x11): 擦除 FLASH 地址超出整个 FLASH 大小。

3.2.2 CMD_FLASH_DWNL

该命令提供用户下载代码到指定 FLASH 中。接收数据长度必须 4 字节对齐。

接收指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x11							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	接收数据长度: N							
4~7(Par)	下载 FLASH 的起始地址							
8~8+(N-1)(DAT)	DAT[0~N-1]: 下载的具体数据							

- LEN 接收数据长度: 0xXX(LEN[0])、0xXX(LEN[1]), $LEN = LEN[0] + (LEN[1] \ll 8)$
- Par[0~3]: 下载 FLASH 的起始地址, 合成规则为 $Address = Par[0] | (Par[1] \ll 8) | (Par[2] \ll 16) | (Par[3] \ll 24)$ 。
- DAT[0~N-1]: 下载的具体数据, 最大 256 个字节, $4 \leq N \leq 256$, N 必须为 4 的倍数。
- 保留值: 0x00;

回复指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x11							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	发送数据长度: 保留							
4~7(Par)	Par[0]: 状态字节 1 Par[1]: 状态字节 2 Par[2] ~ Par[3]: 保留							
(DAT)	无							

- LEN 发送数据长度: 0x00(LEN[0])、0x00(LEN[1]), $LEN = LEN[0] + (LEN[1] \ll 8)$ 。
- 状态字节(Par[0]、Par[1])根据命令执行情况分为:
 1. 下载成功: 状态标志位(0xA0、0xB0)。
 2. 下载失败: 状态标志位(Par[0]、Par[1])

- (1) (0xE0、0x10): 下载 FLASH 失败;
- (2) (0xE0、0x11): 下载 FLASH 地址超出整个 FLASH 大小;
- (3) (0xE0、0x12): 下载 FLASH 起始地址不是 4 字节对齐;
- (4) (0xE0、0x13): 下载 FLASH 数据长度不是 4 的倍数, 或者大于 256;

3.2.3 CMD_DATA_CRC_CHECK

该命令用于校验下载数据是否正确, 数据下载完成后进行 CRC32 校验, 接收指令需包含下载数据的 CRC 值和校验起始地址以及校验长度。

接收指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x12							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	发送数据长度: 8							
4~7(Par)	校验起始地址							
8~15(DAT)	DAT[0:3]: 32bit CRC 校验值 DAT[4:7]: 校验长度(单位: 字节, 必须 4 的倍数)							

- LEN 发送数据长度: 0x08(LEN[0])、0x00(LEN[1]), $LEN = LEN[0] + (LEN[1] \ll 8)$;
- Par[0~3]: 校验起始地址, 其合成规则为 $Address = Par[0] | (Par[1] \ll 8) | (Par[2] \ll 16) | (Par[3] \ll 24)$, Address 只能是在 FLASH 范围内;
- DAT[0~3]: 32bit CRC 校验值, 其合成规则为 $CRC32 = DAT[0] | (DAT[1] \ll 8) | (DAT[2] \ll 16) | (DAT[3] \ll 24)$;
- DAT[4~7]: 校验长度, 其合成规则为 $CRC_LEN = DAT[4] | (DAT[5] \ll 8) | (DAT[6] \ll 16) | (DAT[7] \ll 24)$, CRC_LEN 只能是在有效范围内, 必须是 4 的倍数。
- 保留值: 0x00;

回复指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x12							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	发送数据长度: 保留							

4~7(Par)	Par[0]: 状态字节 1 Par[1]: 状态字节 2 Par[2] ~ Par[3]: 保留
(DAT)	无

- LEN 发送数据长度: 0x00(LEN[0])、0x00(LEN[1]), $LEN = LEN[0] + (LEN[1] \ll 8)$;
- 状态字节(Par[0]、Par[1])根据命令执行情况分为:
 - 1.校验成功: 状态标志位(0xA0、0xB0);
 - 2.校验失败: 状态标志位(Par[0]、Par[1]):
 - (1) (0xE0、0x10): CRC 校验失败;
 - (2) (0xE0、0x11): CRC 校验地址超出整个 FLASH 大小;
 - (3) (0xE0、0x13): CRC 校验长度不是 4 的倍数;

3.2.4 CMD_SYS_RESET

该指令用于软件复位 IAP 程序。

接收指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x13							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	发送数据长度: 保留							
4~7(Par)	保留							
(DAT)	无							

- 保留值: 0x00;

回复指令:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段: 0x13							
1(CMD_L)	二级命令字段: 保留							
2~3(LEN)	发送数据长度: 保留							
4~7(Par)	Par[0]: 状态字节 1 Par[1]: 状态字节 2							
(DAT)	无							

- 状态字节(Par[0]、Par[1])根据命令执行情况分为：

- 1.返回成功：状态标志位(0xA0、0xB0)；
- 2.返回失败：状态标志位(0xE0、0x10)。

3.2.5 CMD_APP_GO

下载完应用程序到 FLASH 后跳转 APP 程序执行。

接收指令：

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段：0x14							
1(CMD_L)	二级命令字段：保留							
2~3(LEN)	发送数据长度：保留							
4~7(Par)	保留							
(DAT)	无							

- 保留值：0x00；

回复指令：

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	一级指令字段：0x14							
1(CMD_L)	二级命令字段：保留							
2~3(LEN)	发送数据长度：保留							
4~7(Par)	Par[0]：状态字节 1 Par[1]：状态字节 2							
(DAT)	无							

- 状态字节(Par[0]、Par[1])根据命令执行情况分为：

- 1.返回成功：状态标志位(0xA0、0xB0)；
- 2.返回失败：状态标志位(0xE0、0x10)。

4 演示 IAP 工程

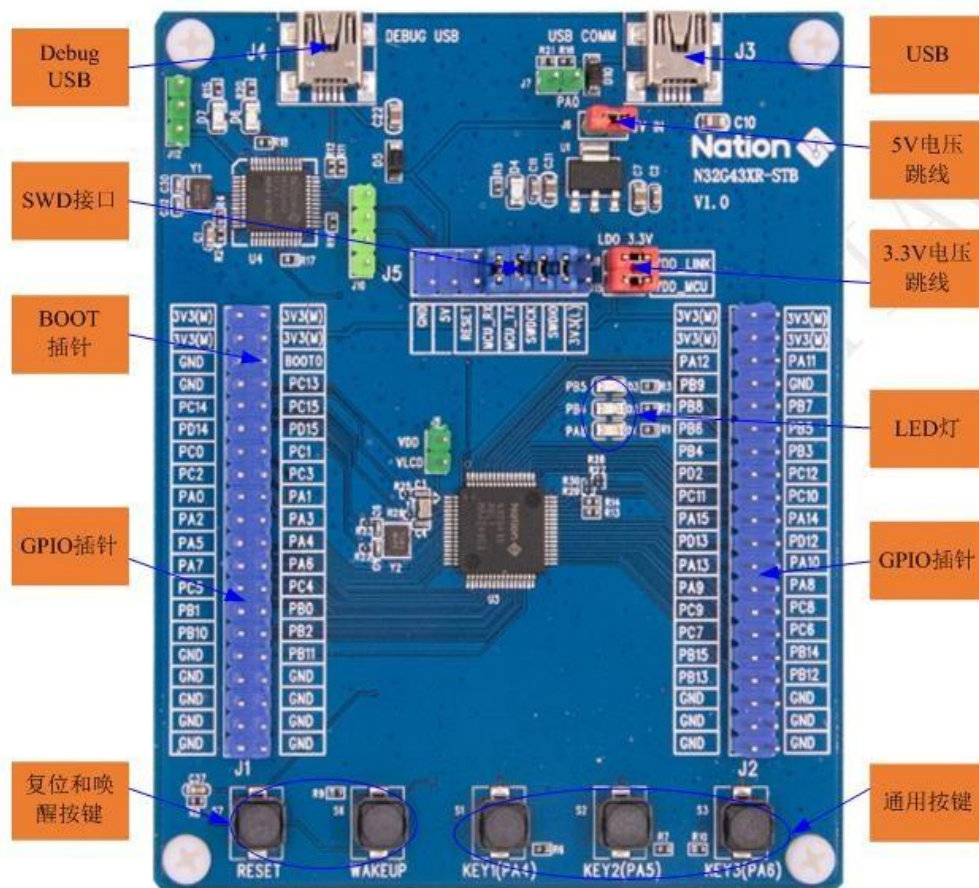
以下小节将重点介绍如何使用 IAP 工程。

4.1 硬件连接

4.1.1 开发板

选择 N32G43x 系列最小系统开发板 N32G43XR-STB V1.0，如图 4-1 N32G43XR-STB V1.0 最小系统开发板所示。开发板的使用指南请查看《UG_N32G43XR-STB 开发板硬件使用指南 V1.0》文档。

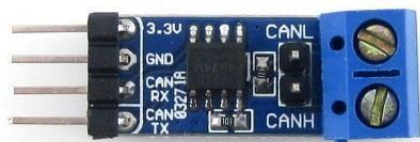
图 4-1 N32G43XR-STB V1.0 最小系统开发板



CAN 外设接口选择 CAN1 对应的 PD0、PD1，波特率选择 500Kbps。

CAN 总线发送接收器选择 SN65HVD230，如下图 4-2 CAN 总线发送接收器开发板所示。

图 4-2 CAN 总线发送接收器开发板



使用 CAN 协议分析仪 CANNalyst-II 发送 CAN 指令，如下图 4-3 CAN 协议分析仪所示。

图 4-3 CAN 协议分析仪



4.1.2 软件测试

下载 IAP 程序到 N32G43XR-STB V1.0 最小系统开发板，复位运行。打开 CAN 协议分析仪 CANNalyst-II 的上位机软件，配置波特率 500Kbps。下发指令等待回复，如图 4-4 所示。

图 4-4 CAN 总线发送接收器开发板

	序号	系统时间	时间标识	CAN通道	传输方向	ID号	帧类型	帧格式	长度	数据
不支持的指令	00000	20:14:02.190	无	ch1	发送	0x0000	数据帧	标准帧	0x08	x 00 00 00 00 00 00 00 00
	00001	20:14:02.195	0x2FAF4E	ch1	接收	0x0400	数据帧	标准帧	0x08	x 00 00 08 00 BB CC 00 00
擦除指令	00002	20:14:02.236	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 10 00 00 00 00 00 FA 00
	00003	20:14:02.465	0x2FB9C5	ch1	接收	0x0400	数据帧	标准帧	0x08	x 10 00 08 00 A0 B0 00 00
下载指令	00004	20:14:02.486	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 00 00 01 00 30 00 08
	00005	20:14:02.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00006	20:14:02.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00007	20:14:03.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00008	20:14:03.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00009	20:14:03.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00010	20:14:03.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00011	20:14:04.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00012	20:14:04.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00013	20:14:04.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00014	20:14:04.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00015	20:14:05.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00016	20:14:05.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00017	20:14:05.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00018	20:14:05.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00019	20:14:06.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00020	20:14:06.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00021	20:14:06.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00022	20:14:06.986	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00023	20:14:07.236	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00024	20:14:07.486	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00025	20:14:07.736	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00026	20:14:07.986	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00027	20:14:08.236	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00028	20:14:08.486	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00029	20:14:08.736	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00030	20:14:08.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00031	20:14:09.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00032	20:14:09.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00033	20:14:09.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00034	20:14:09.986	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00035	20:14:10.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
	00036	20:14:10.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 11 22 33 44 55 66 77 88
CRC校验指令	00037	20:14:10.505	0x30F34C	ch1	接收	0x0400	数据帧	标准帧	0x08	x 11 00 08 00 A0 B0 00 00
	00038	20:14:10.737	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 12 00 08 00 00 30 00 08
	00039	20:14:10.987	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 14 01 8D 18 00 01 00 00
复位指令	00040	20:14:11.015	0x310686	ch1	接收	0x0400	数据帧	标准帧	0x08	x 12 00 08 00 A0 B0 00 00
	00041	20:14:11.237	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 13 00 00 00 00 00 00 00
跳转指令	00042	20:14:11.255	0x311047	ch1	接收	0x0400	数据帧	标准帧	0x08	x 13 00 08 00 A0 B0 00 00
	00043	20:14:11.487	无	ch1	发送	0x0400	数据帧	标准帧	0x08	x 14 00 00 00 00 00 00 00
	00044	20:14:11.495	0x311A09	ch1	接收	0x0400	数据帧	标准帧	0x08	x 14 00 08 00 A0 B0 00 00

5 总结

IAP 例程实现了简单的基于 CAN 外设接口升级 APP 的二级 BOOT。用户可以在此基础上再添加指令或者数据加解密等功能。

除了基于 CAN 外设接口，还有基于 USART、USB、I2C、SPI 等外设接口的 IAP 示例代码，统一采用相同的指令集。

6 历史版本

版本	日期	备注
V1.0	2020.10.10	新建文档

7 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。请使用人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。