Application Note
ZHCA676 – Dec. 2016

F28M35x 串口在线升级设计

马桂龙 C2000 产品技术支持

摘要

对嵌入式系统而言，主控芯片的升级越来越重要，传统的升级方式需要依赖编程器、仿真器等特定工具，并且需要保留 JTAG 接口，或采用外设引导方式（如 SCI boot）时需要配置 GPIO 引脚状态，对系统的升级带来不便。本文针对嵌入式系统传统升级方式不便之处，提出一种基于串口的在线升级方法，对 C2000 的 concerto 系列 F28M35x 进行在线升级。设计了基于上位机软件 GUI 和 F28M35x 的 bootloader 程序，通过 M3 子系统的串口实现对 F28M35x 双核的在线升级，同时具有防掉电升级失败功能。

目录

图

1. F28M35x 串口在线升级简介

嵌入式系统的在线升级是现在许多产品必带的一个功能，根据编程接口的不同，有 JTAG，UART，CAN，SPI 等多种方式，尽管编程接口有所不同，但是其在线升级的原理是类似的，都是通过外部触发条件，使微控制芯片脱离常规应用程序的执行流程，跳到程序空间某个固定位置的执行代码(bootloader)，擦除 FLASH，通过相应的通信协议接收升级数据，并将升级数据烧写到芯片的 FLASH 存储区域。串口在线升级方式是最常见的一种在线升级方式，本文以串口在线升级方式实现 C2000 一款双核产品 F28M35x 的在线升级。F28M35x 属于 C2000 Concerto 系列，它包含作为主机功能的 M3 子系统和作为控制功能的 C28 子系统。本文通过串口在线升级方式，实现对 F28M35x 两个子系统在线升级，通过上位机软件 GUI 控制 F28M35x 整个升级过程。

2. 设计原理

本文设计的在线升级方案包括以下三个核心模块：

- 上位机软件 GUI，实现与 F28M35x 命令交互和升级代码的传输
- F28M35x 的 bootloader 程序，包含通信代码和 Flash API 编程代码
- 用户升级的应用程序 AppCode

GUI 是位于 PC 端基于 LabVIEW 平台开发的上位机软件。用户通过 GUI 发送升级命令，传输升级代码给 F28M35x，并显示升级过程中的各种状态量。

Bootloader 程序是存放在某一 Flash 扇区的用于实现 F28M35x 升级功能的代码，其中，通信代码实现与上位机 GUI 的命令交互和升级代码的接收，Flash API 编程代码完成 Flash 擦除，升级代码烧写和验证。由于 F28M35x 是一颗双核芯片，包含 M3 和 C28 子系统，因此它们有各自的 bootloader，存放在各自的 Flash 扇区中。

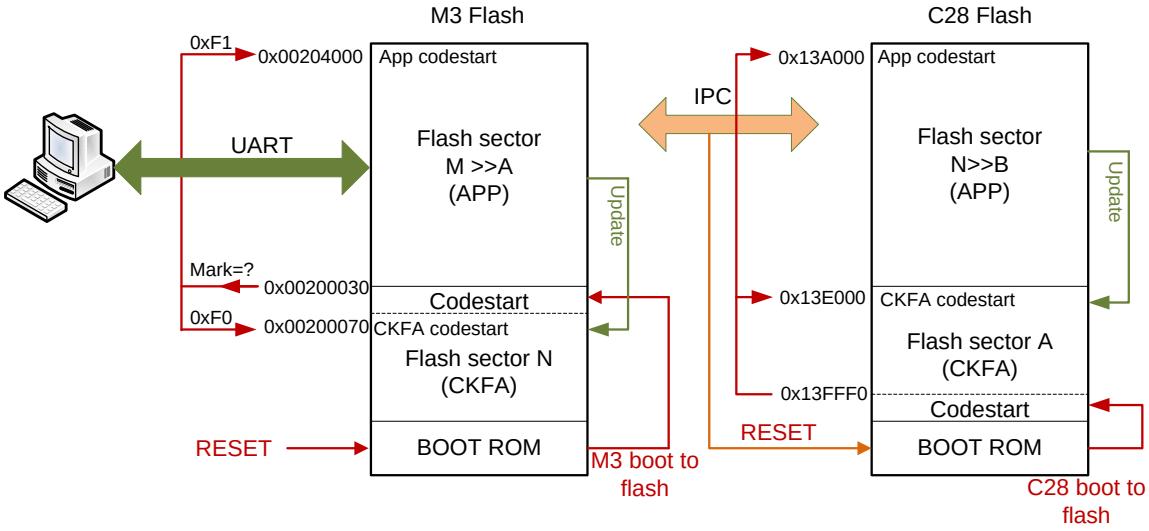
AppCode 是用户编译生成的 Hex 格式的升级代码，M3 子系统根据规定的通信协议将该升级代码分块接收到 M3 的 RAM 存储区或共享 RAM 存储区，在 bootloader 的 Flash API 编程代码的控制下将升级代码烧写到相应的子系统。

本文设计的在线升级原理框图如图一所示。

M3 子系统的 bootloader 程序放在 Flash N 扇区，其他扇区用于存放应用代码。系统上电复位后，M3 从 Flash 引导，开始执行起始代码 Codestart，这段代码主要实现 M3 子系统防升级失败功能，系统通过判断一个升级状态标志位决定程序的跳转。若没有升级失败情况，则系统会跳转到用户程序入口运行；若出现升级失败情况，则系统会跳转到 bootloader 程序入口等待升级命令。M3 子系统用户程序中应保留有一段代码用于监测串口命令，当串口接收到升级命令时，M3 子系统停止用户程序的运行，跳转到 bootloader 程序执行升级操作。M3 子系统的 bootloader 程序除了需要实现本系统的升级功能，还需要具备 C28 子系统升级时的中转功能，作为上位机软件 GUI 和 C28 子系统的媒介，传递命令和升级代码。

C28 子系统的 bootloader 程序放在它的 Flash A 扇区，其他扇区用于存放应用代码。C28 在上电复位后，M3 通过 Inter Processor Communications (IPC) 控制 C28 从 Flash 引导，与 M3 子系统类似，

C28 会判断一个防升级失败的标志位来决定从应用程序中运行或是跳到 bootloader 程序中运行。C28 的应用程序同样会有一段代码用于监视 IPC 发送过来的命令，如果收到 IPC 发送过来的升级命令，则跳到 bootloader 程序中进行应用程序的升级。C28 的在线升级需要以 M3 子系统作为媒介，当 M3 子系统收到串口命令为升级 C28 时，会通过 IPC 将升级命令转发给 C28，同时 M3 子系统接收升级代码，并把这些升级代码放在双核的共享 RAM 中，然后通过 IPC 协助 C28 完成应用程序的升级。



图一 F28M35x 在线升级原理框图

3.软件设计

3.1 F021 Flash API v1.50 介绍

F021 Flash API v1.50 是一个软件函数库，里面包含了对 F021 (65nm) 片上 Flash 存储器的擦除、编程、校验等函数。F28M35x 的 Flash 是 F021（65nm）的片上存储器， bootloader 程序中的 Flash API 将采用这个库中的相关函数，其中两个关键的函数为 Fapi_issueAsyncCommandWithAddress()，Fapi_issueProgrammingCommand()。

Fapi_issueAsyncCommandWithAddress()用来擦除规定地址的 Flash 扇区，例如擦除 A 扇区：

```
oReturnCheck = Fapi_issueAsyncCommandWithAddress(Fapi_EraseSector,
                    (uint32 *)Bzero_SectorA_start);
```

Fapi_issueProgrammingCommand()用来对一特定地址的 Flash 进行编程，由于 F28M35x 的 Flash 带 ECC 保护功能，因此在进行编程的时候，需要生成一个 ECC 校验值。对于 F28M35x，这个函数能够实现 1~16 个字节的编程。需要注意 ECC 的生成和 Flash 的编程都有地址对齐的相关限制。对于 ECC，每个 ECC 校验码的生成是每 64 位的数据，对齐 64 位的地址边界，如 M3 的 0x00~0x07 的 8 个数据会产生一个 ECC。对于 Flash 数据编程，要注意起始地址加上编程字节数不能超过一个 bank 的宽度，例如在 0x4 的起始地址实现 14 个字节数据的编程在 16 字节 bank 宽度的 Flash 上是不允许的。例如对 M3 子系统 Flash 编程：

```
oReturnCheck = Fapi_issueProgrammingCommand(
                (uint32 *) u32Index,
                au8DataBuffer,
                0x10,
```

```
                0,
                Fapi_AutoEccGeneration);
```

3.2 在线升级通信协议

本文设计的在线升级方案使用上位机 GUI 软件实现与 F28M35x 进行通信，完成命令交互与升级代码的传输。为了实现稳定可靠的通信，制定了一套在线升级通信协议。

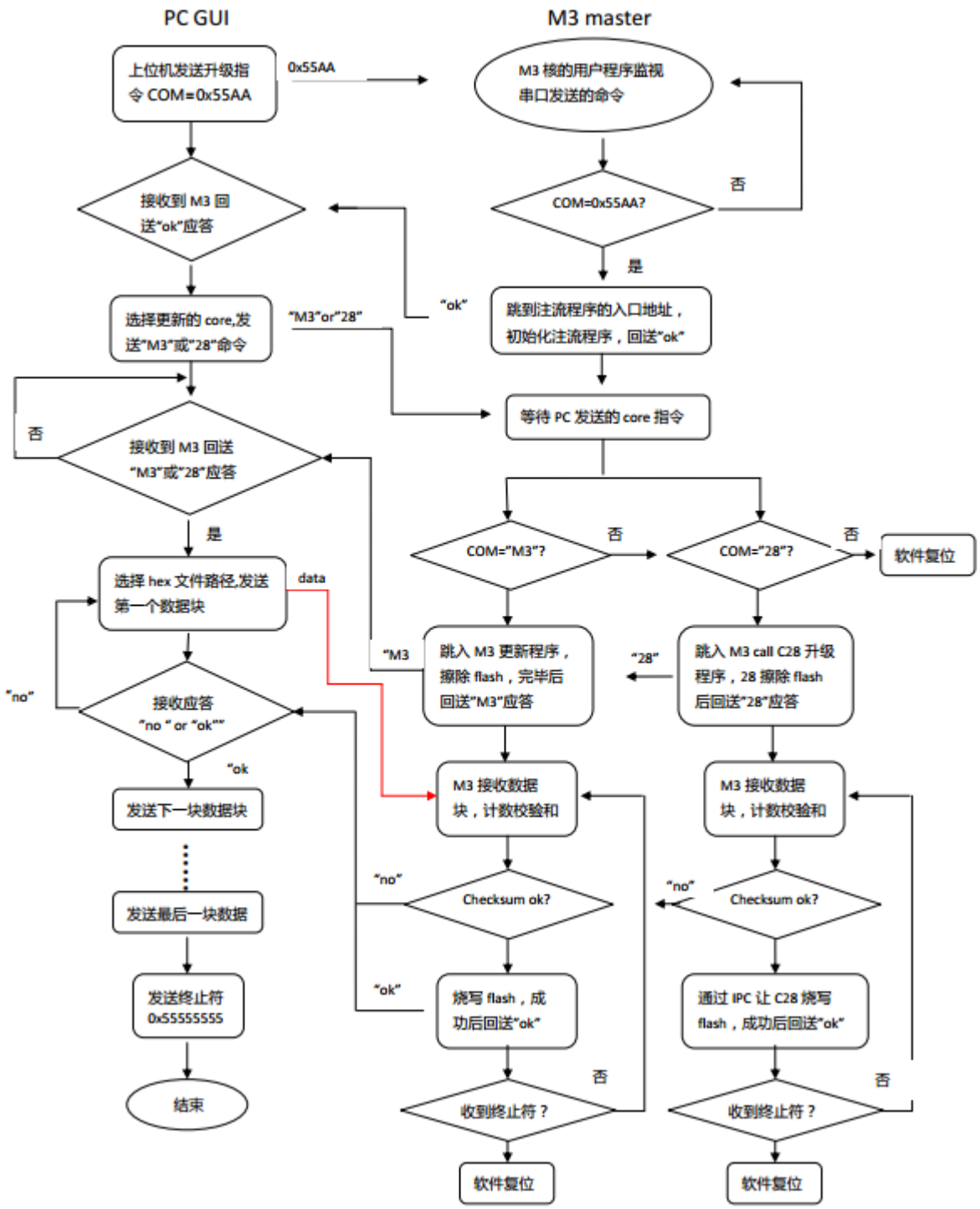
上位机软件 GUI 与 M3 子系统的通信协议如图二所示。

GUI 发送升级指令为 0x55AA，M3 接收到该指令后跳转到 M3 bootloader，并回送“ok”应答；

GUI 发送升级内核命令“M3”或“C28”，M3 接收到该命令后跳转到相应的子函数，并回送“M3”或“C28”应答；

在发送升级代码时，每发送一个数据块，都需要 M3 通过串口回送该数据块编程状态值“ok”或“no”应答；

当升级代码发送完毕时，GUI 发送终止符 0x55555555 给 M3，M3 接收到终止符后进行软件复位。



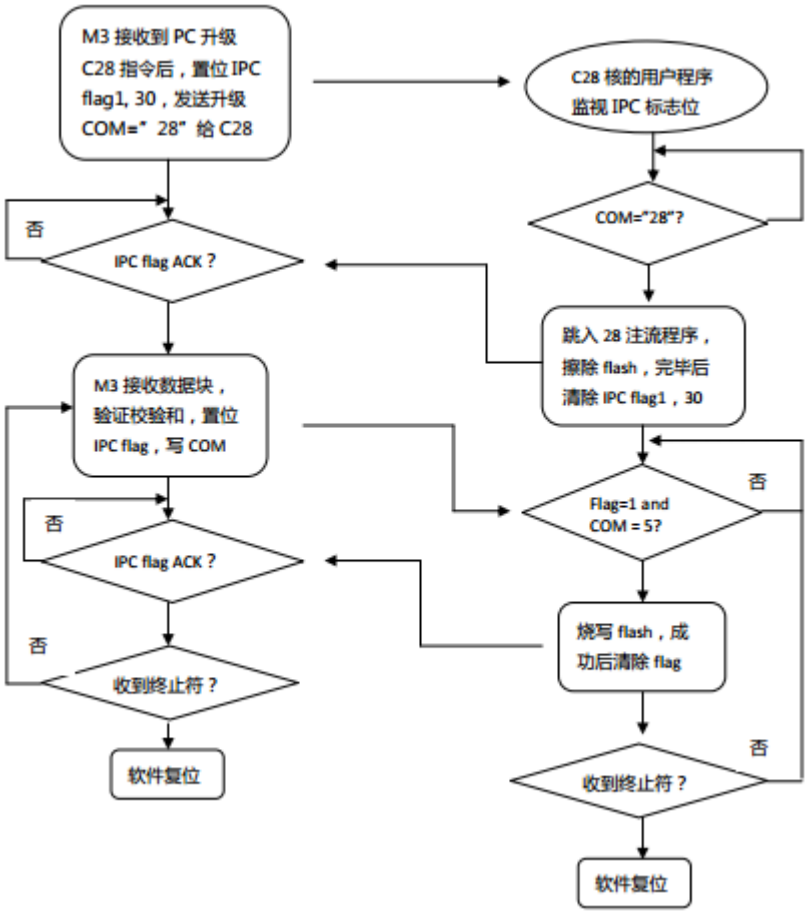
图二 GUI 与 M3 子系统通信协议

M3 子系统与 C28 子系统的通信协议如图三所示。

M3 接收到 GUI 升级 C28 子系统命令后，置位 IPC flag1 和 flag30，升级指令 COM 为“28”，C28 接收到该指令后，擦除 Flash，清除标志位；

M3 接收 GUI 发送的升级代码，完成校验后置位 IPC flag1 和 IPC30，数据指令 COM 为“0x5”，C28 接收到该指令后，执行 Flash 编程，清除标志位；

M3 收到 GUI 发送的终止符后，将该终止符传送给 C28，C28 接收到该终止符后，进行软件复位。



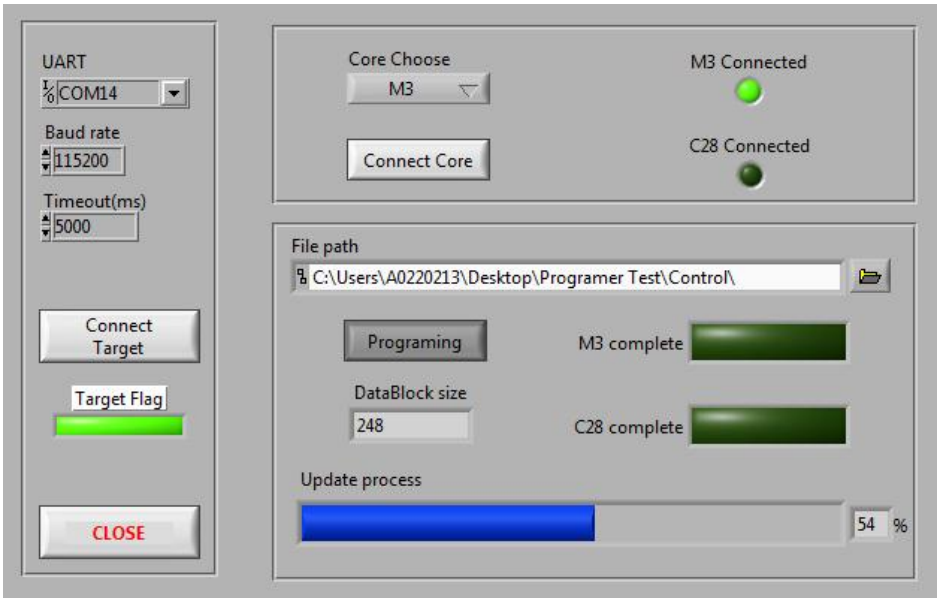
图三 M3 子系统与 C28 子系统 IPC 通信协议

3.3 GUI 设计

本文设计的上位机软件 GUI 是基于 NI 公司的 LabVIEW 软件开发的，如图四所示。该 GUI 主要包括几个部分：串口配置模块，文件的读取和解析模块，命令按钮以及相关状态指示灯。其中，文件的读取和解析模块为 GUI 设计的关键部分，该模块读取用户升级代码生成的 hex 文件，解析成一个个带固定格式的数据块，数据块格式为：

Address	Length	Data	Checksum
32 bits	16 bits	1~244 bytes	16 bits

在启动 GUI 后，将 F28M35x 的串口连接到电脑，首先配置串口，然后点击 Connect Target 按键，向 F28M35x 发出升级命令，当 M3 子系统接收到该升级命令后，会回送应答，此时 Target Flag 变绿色。接着在 Core Choose 选择升级的内核，点击 Connect Core，将升级内核命令发送给 M3，再收到应答后相应的指示灯会变绿色。下一步即可通过 File path 浏览对应的 hex 文件，点击 Programming 按钮，Update process 进度条显示升级的进度，当烧写完成后，相应的指示灯会变绿色。升级完毕，点击 CLOSE 按键，关掉 GUI 软件。



图四 GUI 显示界面

3.4 bootloader 程序设计

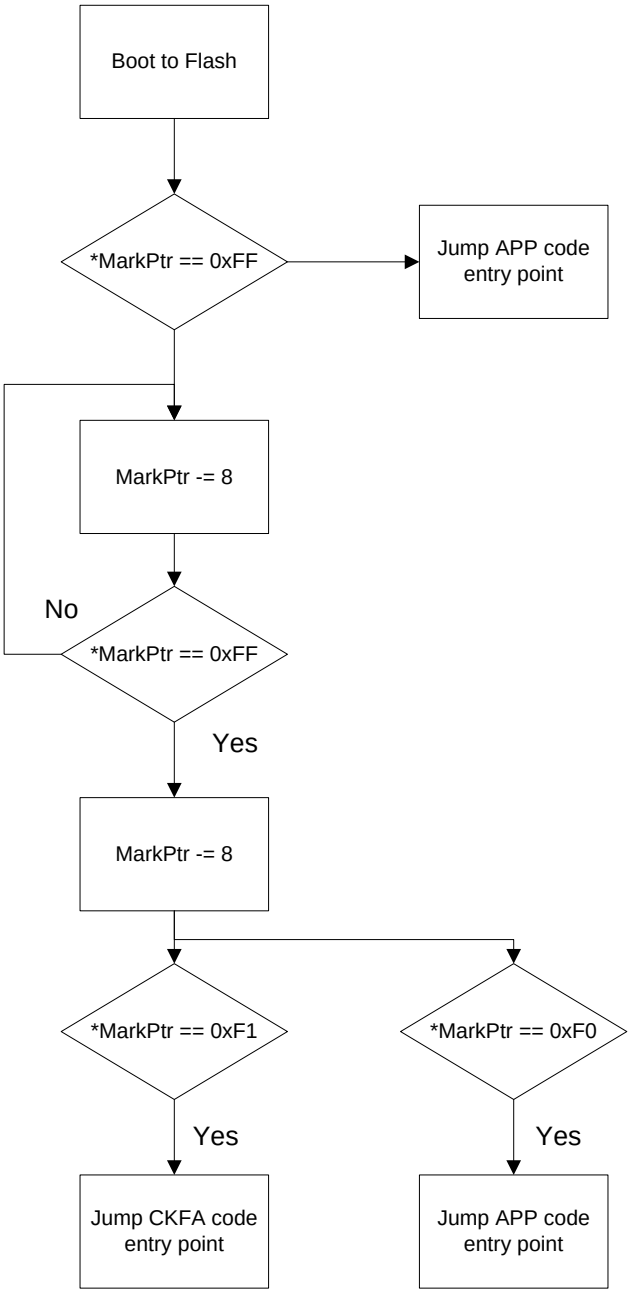
3.4.1 防升级失败功能

在系统升级过程中，会存在一些因素，如在编程环节突然掉电，导致系统升级失败。由于此时系统原有的应用程序已经被擦除掉，此时如果没有一些可靠的措施进行重新升级，则会导致系统不能继续运行，因此本文在 bootloader 代码中增加了防烧写失败功能的设计。

以 M3 子系统的为例， M3 boot to Flash 的入口地址是 0x200030，该入口地址处，本文在此处添加了一段防升级失败的判断代码，通过查询判断升级状态值 Mark 来决定系统是跳进应用程序正常运行，还是跳进 bootloader 程序进行重新升级。具体实现流程如图五所示。此处为了节省 Flash 空间，将 Mark 状态值放在 bootloader 所在扇区 N，利用 bootloader 剩余的空间记录升级状态。另外，由于 F021 片上 Flash 带 ECC 保护功能，ECC 校验值是以 64 位为单位进行计算的，在这个 ECC 所属的 64 位的区域内，如果已写入 ECC 校验值，则不能再次写入 Mark 值。因此此处的 Mark 升级状态记录必须间隔 8 个字节记录一个状态值。根据 bootloader 最终占用的 Flash N 扇区的空间，确定从 0x203900 作为记录 Mark 标志的起始地址。如果起始地址内容为 0xFF，则表示系统没升级过，否则地址增加 8，不断查询，知道出现该地址内容为 0xFF，则往前减 8 的地址内容则为上次升级状态标志值 Mark。如果 Mark 值为 0xF1，表示升级失败，程序跳到 bootloader 中运行；如果 Mark 值为 0xF0，表示升级成功，程序跳到应用程序正常运行。防升级失败功能软件流程图如图五所示。

另一方面，在系统升级时，需要在 bootloader 程序中增加 Mark 状态值的记录。在调用 Flash API 擦除应用程序的 Flash 扇区之前，必须在相应的 Mark 地址调用 Flash API 写入 0xF1，并在系统成功完成升级后，再调用 Flash API 在该地址增加 8 个单元处再写入 0xF0 表示升级成功。

C28 子系统的放升级失败功能与 M3 子系统类似，在 C28 boot to Flash 的 code_start 代码增加一段防升级失败程序，需要注意 C28 的汇编指令与 M3 的汇编指令是不同的，同时 C28 的存储单元以 word 为单位，Mark 记录的地址间隔为 4。



图五　防升级失败功能软件流程图

3.4.2 M3 bootloader 程序

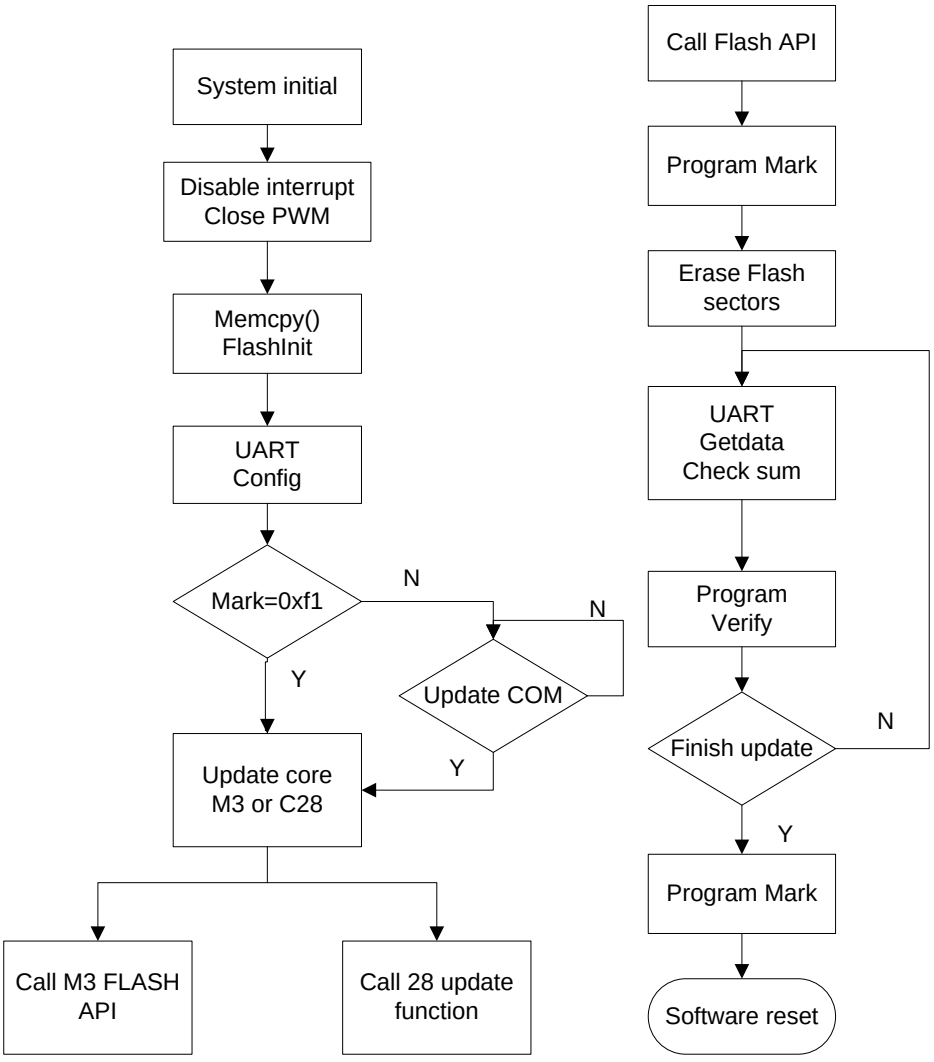
M3 的 bootloader 程序包含串口通信功能和 Flash API 功能，当升级 C28 子系统时，M3 的 bootloader 还具有媒介功能，作为 GUI 和 C28 子系统的命令和数据中转站。

当 M3 的应用程序监视到串口收到升级命令时，M3 子系统会停止应用程序的执行，跳到 M3 bootloader 的_c_int00 入口开始执行，在 bootloader 的主函数中，会重新初始化 M 子系统，包括系统时钟，Flash，shared RAM，UART 等模块。初始化完毕后会回送应答给 GUI，然后等待接收升级子系统命令，当 GUI 发送下来的命令是升级 M3 子系统时，M3 bootloader 会跳转到 M3_CallFlashAPI ()子 函数，该函数执行擦除 Flash 应用程序扇区，并回送应答给 GUI，表示 M3 子系统 Flash 擦除完毕，可以接收升级代码进行系统更新。 然后进入一个主循环，在这个循环中会根据预先设定的通信协议，数据格式，通

过串口接收并校验每一个数据块的是数据，然后再将该数据块编程到相应的地址，再回送一个应答给 GUI 表示这个数据块已经编程完毕，可以发送下一个数据块。最后 GUI 会发送一个结束符表示升级数据发送完毕，这时 M3 子系统会跳出主循环，接着软件复位子系统。

M3 bootloader 还具备媒介功能，当 GUI 发送下来的命令是升级 C28 子系统时，M3 bootloader 会转到 M3_CallC28FlashAPI () 子函数，在该函数里，M3 子系统通过 IPC 模块将实现指令信息的交互。M3 子系统通过 IPC 将升级指令传送给 C28，然后等待 C28 回送应答，并将该应答回送给 GUI，接着同样进入一个主循环接收 GUI 发送下来的数据块，M3 子系统接收并校验 C28 子系统升级代码的每一个数据块，并存放在共享 RAM 中，然后通过 IPC 命令让 C28 子系统读取每一个升级代码的数据块，C28 子系统完成每一个数据块的编程后，都会通过 IPC 回送应答给 M3 子系统，M3 子系统再回送该应答给 GUI，最终完成 C28 子系统的升级工作。

M3 bootloader 的软件流程图如图六所示。

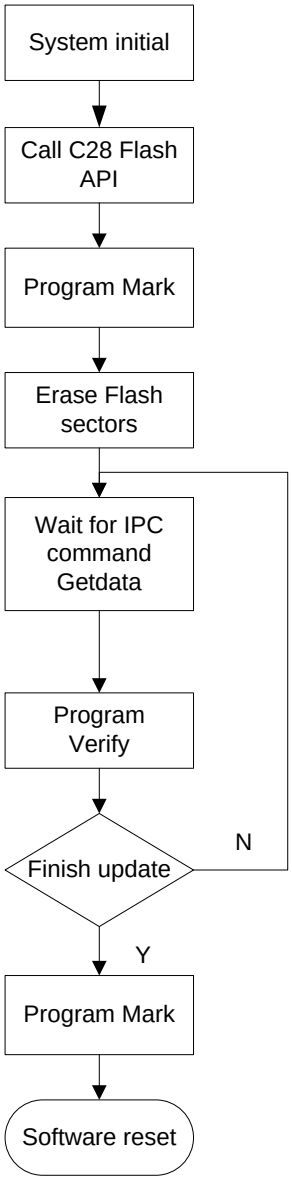


图六 M3 子系统 bootloader 软件流程图

3.4.3 C28 bootloader 程序

C28 子系统的 bootloader 程序包含 IPC 通信功能和 Flash API 功能。与 M3 子系统的 bootloader 相比，C28 子系统的 bootloader 简单许多，关于升级代码的接收和校验已经由 M3 子系统完成。当 C28 子系统的应用程序监视到 IPC 传送过来的升级命令时，C28 子系统会跳转到 C28 bootloader 的_c_int00 开始执行升

级操作。在 C28 的 bootloader 主函数中，会关 PWM 模块，关 C28 子系统中断，然后完成相关的初始化功能，接着调用 CallFlashAPI () 子函数。在该函数中，与 M3 子系统的 M3_CallFlashAPI ()功能类似，会擦除应用程序的 Flash 扇区，回送 IPC 应答给 M3 子系统，然后进入主循环，通过 M3 子系统传递的 IPC 指令，将每一个数据块编程到对应的 Flash 区域，知道接收到结束符。最后软件复位，完成 C28 子系统的在线升级。C28 子系统 bootloader 的软件流程图如图七所示。

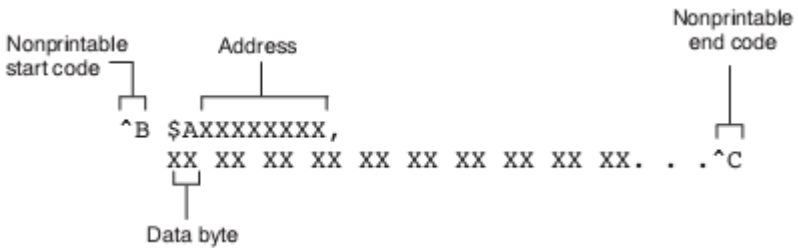


图七 C28 子系统 bootloader 软件流程图

4 F28M35x 在线升级的具体实现

4.1 升级代码的 hex 文件生成

本文采用的目标文件格式为 ASCII-Hex 格式，其格式如图 xx 所示。ASCII-Hex 目标文件结构很简单，只包含起始地址以及 Flash 数据，方便上位机软件 GUI 的解析。



当准备系统在线升级时，需要将升级代码编译生成.out 文件，然后再将.out 文件通过 hex2000.exe 工具转换成 ASCII 格式的.hex 文件。本文介绍一种简单的 ASCII-Hex 文件的生成方法：

新建一个文件夹，将 hex2000.exe、升级代码的.out 文件放在该文件夹中，同时在该文件夹中新建一个 txt 文件，并修改其后缀名为.cmd，如 config.cmd。右键选择编辑该文件，写入 hex2000.exe 转换配置信息并保存，例如

example. out
-map example.map
-o example.hex
-order MS
-romwidth 16
- memwidth 8 (C28 子系统应修改参数为 16)
-a

相关配置选项含义可以在参考文档 1 的 11.2.1 节。

接着再创建另一个 txt 文件，并修改其后缀名为.bat，如 ToHex.bat。右键选择编辑该文件，写入以下内容并保存。

hex2000.exe config.cmd

这样文件夹中将有四个文件，包括 hex2000，升级代码.out 文件，config.cmd 文件，ToHex.bat。通过双击 ToHex.bat 文件，即可在该文件夹中生成 ASCII-Hex 文件以及 memory map 文件。

4.2 在线升级方案实现

根据系统功能要求和通信协议建立两个 bootloader 工程，经过编译调试后加载到 F28M35x 的两个子系统中。然后根据用户的需求建立两个应用工程，编译调试后也加载到 F28M35x 的两个子系统中。应注意应用程序功能的加载不能擦除各自 bootloader 程序所在的扇区。为了实现系统的防升级失败功能，以及接收到升级命令时能够正确跳转，需要对 M3 子系统和 C28 子系统的 CMD 文件进行相应修改。

系统上电复位 boot to Flash 后会跳转到防升级失败功能代码，判断系统是跳转到 bootloader 升级程序区或是跳转到用户应用程序正常运行，本设计将防升级失败功能代码放在 bootloader 工程的 Flash 入口地址处，如 M3 子系统的 ResetISR()，C28 子系统的 code_start。M3 子系统的 bootloader 工程代码分配在 N 扇区，并设定其_c_int00 地址为 0x00200070；C28 子系统的 bootloader 工程代码分配在 A 扇区，并设定其_c_int00 地址为 0x13e000。当用户程序接收到升级指令时，会跳转到各自 bootloader 的_c_int00 实现升级过程。

用户程序工程与 bootloader 工程是相互独立的，本文设定 M3 子系统用户程序的入口地址为 0x00204000，C28 子系统用户程序的入口地址为 0x13A000。系统上电后，会先执行防升级失败功能代码，如果系统正常，则需要跳转到该入口地址执行用户程序。

如前所述，防升级失败功能代码需要判断升级状态值 Mark。为了节省 Flash 空间，本文利用 bootloader 所在扇区未编程区域记录 Mark 升级状态值。对 bootloader 编译后，通过查看 memory map 文件，可以获得 bootloader 所在扇区的使用空间情况，因此确定 M3 子系统 Mark 记录起始地址为 0x203800，C28 子系统 Mark 记录起始地址为 0x13F780。

在加载上述 bootloader 工程和应用程序工程后，此时 F28M35 已具备串口在线升级功能。根据用户需求，通过修改应用程序功能并编译可获得需要升级代码的.out 文件，并通过 hex2000 工具转换成

ASCII-hex 文件。打开 PC 上的 GUI 软件，将 F28M35x 的串口与 PC 串口连接起来，配置 GUI 上的串口信息，根据 3.3 节 GUI 使用步骤，即可实现 F28M35x 两个子系统的在线升级功能。本文在 F28M35x control card 上进行测试，测试结果显示可以通过 M3 子系统的串口实现 M3 和 C28 两个内核的独立升级，以及同时升级功能；在升级过程中，强制断电后，再次上电仍可以实现重新升级功能，即防升级失败功能。

5. 总结

本文开发出一种针对 Concerto F28M35x 双核芯片的串口在线升级方案。详细介绍了该系统在线升级的设计原理，并给出整个串口在线升级方案的软件设计过程，包括在线升级的通信协议、PC 端 GUI 设计以及 F28M35x 的 M3 子系统和 C28 子系统的 bootloader 的设计。最后介绍了该串口在线升级方案的具体实现过程，包括升级代码的 hex 文件的生成，应用程序和 bootloader 的地址分配、工程编译和加载、GUI 具体操作。测试结果表明，该方案能够通过 M3 子系统的串口完成 F28M35x 双核的同时升级和独立升级功能，同时还具有防升级失败的功能，当系统升级过程中出现掉电时，通过重新上电，能够再次进行系统升级。

参考文档

1. TMS320C28x Assembly Language Tools v6.1 user's guide (SPRU513E)
2. TMS320C28x Optimizing C/C++ Compiler v6.1 (SPRU514E)
3. F021 Flash API Version 1.50 Reference Guide (SPNU501A)
4. TMS320F281x Boot ROM Serial Flash Programming (SPRAAQ2)
5. Concerto F28M35x Technical Reference Manual (SPRUH22B)
6. F28M35x Concerto Microcontrollers Datasheet (SPRS742C)

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265