文章编号:1001 - 9081(2009)10 - 2820 - 03

基于 STM32F103VB 的应用编程技术的实现

张舞杰[1],南亦民[2]
(1. 华南理工大学 自动化科学与工程学院,广州 510640; 2. 广州民航职业技术学院 通讯系,广州 510403)
(zhang_liu73@163.com)

摘 要:针对嵌入式应用中更新升级固件的需求,在阐述应用编程(IAP)技术原理的基础上,以具有 Cortex-M3 内核的微控制器 STM32F103VB 为平台,给出了基于 STM32F103VB IAP 技术的实现方案,并对方案的可靠性进行了探讨。最后讨论了 IAP 技术的具体实现方式。该方案实现了以具有 STM32F103VB 微控制器的嵌入式系统终端软件的在线升级,提高了软件维护的方便性,缩短了终端软件系统的开发周期。

关键词:固件升级;应用编程;Cortex-M3;STM32F103VB
中图分类号: TP311 **文献标志码**:A

Design and implementation of IAP techniques based on STM32F103VB

ZHANG Wu-jie[1], NAN Yi-min[2]
(1. *College of Automation Science and Engineering, South China University of Technology, Guangzhou Guangdong* 510640, *China*;
2. *Department of Correspondence, Guangzhou Civil Aviation College, Guangzhou Guangdong* 510403, *China*)

Abstract: According to the requirements of firmware upgrade in embedded applications, the on-line software update solution based on the ARM Cortex-M3 32-bit RISC processor—STM32F103VB was given after introducing the feature of the In Application Programming (IAP) technique, and then the reliability of the solution was discussed. Finally, realization of on-line software update was given in detail. The solution realized on-line software upgrading based on STM32F103VB microprocessors in embedded applications, and shortened the development cycle and decreased the difficulty of maintenance for system software.

Key words: firmware upgrade; In Application Programming (IAP); Cortex-M3; STM32F103VB

0 引言

ARM 是目前嵌入式领域应用最广泛的 RISC 微处理器结构,以其低成本、低功耗、高性能等优点占据了嵌入式系统应用领域的领先地位。2007 年 6 月意法半导体(ST)公司发布了基于 ARM Cortex-M3 的 STM32 系列产品,这些产品结合了高性能、实时、低功耗、低电压、高集成度和易于开发等特性,给 MCU 用户带来了前所未有的自由空间[1]。

STM32 系列产品按性能分成两个不同的系列:STM32F103 是增强型系列,工作在 72MHz,带有片内 RAM 和丰富的外设。STM32F101 是基本型系列,工作在 36 MHz。两个系列的产品拥有相同的片内闪存选项,在软件和引脚封装方面兼容。

对于多数基于闪存的系统而言,当固件已装入终端产品后,更新升级固件是一个重要的需求,这是有关在应用中编程(In Application Programming,IAP)[2,3]的能力。IAP 是用户自己的程序在运行过程中对 User Flash 的部分区域进行烧写,目的是为了在产品发布后可以方便地通过预留的通信口对产品中的固件程序进行更新升级。例如用户在开发完一个系统后要增加新的软件功能,可以使用 IAP 技术在线升级程序,避免重新拆装设备。下面以意法半导体(ST)公司的 STM32F103 为例,讨论 IAP 技术的实现。

1 IAP 技术的原理

通常在用户需要实现 IAP 功能时,即用户程序运行中作自身的更新操作,需要在设计固件程序时编写两个项目代码,第一个项目程序不执行正常的功能操作,而只是通过某种通信管道(如 CAN,USART,USB 等)接收程序或数据,执行对第二部分代码的更新;第二个项目代码才是真正的功能代码。这两部分项目代码都同时烧录在 User Flash 中,当芯片上电后,首先执行第一个项目代码,它做如下操作:1) 检查是否需要对第二部分代码进行更新;2) 如果不需要更新则转到 4);3)执行更新操作;4)跳转到第二部分代码执行。

第一部分代码必须通过其他手段,如 JTAG 或 ISP 烧入;第二部分代码可以使用第一部分代码 IAP 功能烧入,也可以和第一部分代码一道烧入,以后需要程序更新时再通过第一部分 IAP 代码更新。

2 STM32F103 的存储器组织

STM32F103 的程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4 GB 的线性地址空间内。可访问的存储器被分为 8 个 512 MB 的块。数据字节以小端格式存放在存储器中,也即一个字的最低有效字节被存放在该字的最低地址字节中。

片内集成的 Flash、SRAM 被映射到如图 1 所示的地址空间中。

SRAM:20 KB,地址范围 0x2000 0000 ~ 0x2000 4FFF。

FLASH:由主存储块 Main Block,信息块 Information Block 组成。

收稿日期:2009 - 04 - 08。 **基金项目**:广东省博士启动基金资助项目(8451064101000594)。
作者简介:张舞杰(1970 -),男,湖南长沙人,博士,主要研究方向:图像处理、模式匹配、嵌入式系统、智能控制; 南亦民(1971 -),男,河南驻马店人,讲师,主要研究方向:计算机应用、嵌入式系统。

其中,主存储块用于存放用户程序,128 KB,地址范围 0x0800 0000 ~ 0x0801 FFFF。信息块又包括系统存储区 System Memory 和用户选择字节 Option Bytes 两个部分。System Memory 地址范围 0x1FFFF F000 ~ 0X1FFFF F7FF 共计 2 KB,用于存放通过 UART1 进行 ICP 编程的 BOOTLOADER;Option Bytes 包含 16 个字节。

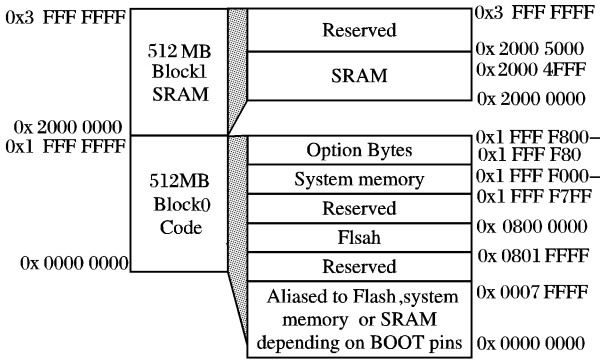

图 1　Flash、SRAM 映射

3　启动配置

在 STM32F103 中,可以通过 BOOT[1:0]引脚选择三种不同启动模式。

表 1　启动模式

启动模式	说明	启动模式选择管脚	
		BOOT1	BOOT0
用户闪存存储器	用户闪存存储器被选为启动区域	X	0
系统存储器	用户闪存存储器被选为启动区域	0	1
内嵌 SRAM	内嵌 SRAM 被选为启动区域	1	1

通过设置选择管脚,对应到各种启动模式的不同物理地址将被映像到第 0 块(启动存储区)。

用户可以通过设置 BOOT1 和 BOOT0 引脚的状态,来选择在复位后的启动模式。即使被映像到启动存储区,仍然可以在它原先的存储器空间内访问相关的存储器。在经过启动延迟后,CPU 从位于 0x0 开始的启动存储区执行代码。

4　IAP 技术的实现方案

STM32F103 微控制器可以运行用户自己编写的引导加载程序 Bootloader 来实现 IAP。引导加载程序必须通过一个通讯接口(如 CAN,USART,USB 等)接收系统程序代码,校验后将正确完整的代码写入指定的 Flash 区域中[4]。

本文采用 Usart1 作为通讯接口,系统上电复位后,首先执行 Bootloader 程序,Bootloader 检测 PB9 引脚的电平(PB9 接按钮),如电平为低,则更新系统程序,否则执行原有的系统程序(当用户按下按钮,PB9 脚为低电平)。

4.1　Flash 分区存储

IAP 技术是从结构上将 Flash 存储器映射为两个存储体,当运行一个存储体上的用户程序时,可对另一个存储体重新编程,之后将控制从一个存储体转向另一个[5]。本文根据 Bootloader 程序的大小把 128 KB 的 Flash 分为两个区域,0x0800 0000 ~ 0x0800 1FFF 8KB 的空间存放 Bootloader,0x0800 2000 ~ 0x0801 FFFF 120KB 的空间存放系统程序。

4.2　IAP 实现流程

Bootloader 存放在 0x0800 0000 起始的 8 KB 空间中,当从用户闪存存储器中启动时,0x0800 0000 映射到 0x0 地址处。系统复位后,用户没有按下按钮则直接转向 0x0800 2000 处执行原有的系统程序,如果用户按下了按钮则初始化串口后,显示 IAP 菜单。用户选择 1 则下载更新系统程序;选择 2 则执行原有的系统程序;选择 3 则去除 0x0800 2000 ~ 0x0801 FFFF Flash 区域的写保护,然后系统复位。引导过程流程如图 2 所示。

在下载更新系统程序的过程中,必须保证传送数据的准确性。为了解决在文件传输过程中的误码问题,本文采用了是 YMODEM 协议[6]。使用 1 024 字节的数据包以提高传输效率,YMODEM 使用 CRC 检测数据传输中的错误。

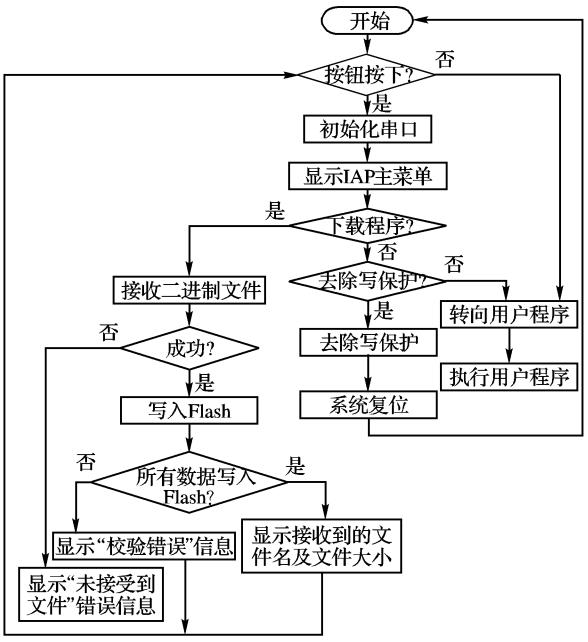

图 2　引导过程流程

4.3　可靠性探讨

程序升级过程难免会出现突然掉电及异常复位、传输误码、Bootloader 区误擦除等各种情况,导致升级失败甚至系统瘫痪。因此必须提供一套可靠的软硬件机制来保证终端 IAP 的正常工作。以下是采取的措施:

1)掉电及异常复位处理。由于本文采用 Bootloader 和系统程序分区存放,用户闪存存储器被选为启动区域,而 Bootloader 区被映射到启动存储区,掉电及异常复位后首先执行 Bootloader,Bootloader 根据系统程序区的起始地址处存放的代码来识别上次升级失败,并能够引导用户再次升级。直到升级成功,Bootloader 才会引导系统程序实现升级后的用户功能。

2)通信的误码处理。程序每一次升级要传输的系统程序代码的内容和文件大小都不一样,但都要保证数据传输的准确性。为防范这些错误,在进行通信过程中使用“文件传输协议”来检测错误或纠错,直到数据正确无误地到达。本文采用普遍使用的计算机远程文件传输协议 YMODEM。

3)Bootloader 区误擦除处理。Bootloader 区的代码一旦被误擦除,将无法实现 IAP 功能,因此在 Bootloader 调试成功后应予以写保护。所有 STM32 的芯片都提供对 Flash 的保护,防止对 Flash 的非法访问——写保护和读保护。写保护是以

四页(1KB/页)Flash 存储区为单位提供保护,对被保护的页实施编程或擦除操作将不被执行,同时产生操作错误标志。要写保护 Bootloader 区,就不能从 Bootloader 区启动,只能从系统存储器启动。设置 BOOT[1:0]选择启动模式为系统存储器,系统存储区中放置一段程序,这段程序可以通过 STM32 的 USART1 接口接收命令,并执行对内部 Flash 的擦除、烧写和保护等操作。

4.4 IAP 技术的具体实现

1)系统程序的下载地址设置为 0x0800 2000。在 MDK 环境下,通过 project→option for target ax→target 选项卡,设置 IROM1 的起始地址为 0x0800 2000。

2)在引导加载程序中设置系统程序下载的地址

```
#define ApplicationAddress0x8002000
```

3)定义一个函数指针

```
typedefvoid ( *pFunction)(void);
pFunction Jump_To_Application;
```

4)执行系统程序

```
if (Push_Button_Read() == 0x00)
{/* 如果按钮按下 */
 …
 /* 执行引导加载程序编程 Flash */
}
else
{
 if ((( *(vu32 *) ApplicationAddress) & 0x2FFE0000 ) == 0x20000000)
   { JumpAddress = *(vu32 *) (ApplicationAddress + 4);
   Jump_To_Application = (pFunction) JumpAddress;
   __MSR_MSP( *(vu32 *) ApplicationAddress);
   Jump_To_Application();
 }
}
```

在离开复位状态后,Cortex-M3 首先读取下列两个 32 位整数的值。

1)从地址 0x00000000 处取出 MSP(主堆栈指针)的初始值。

2)从地址 0x00000004 处取出 PC 的初始值——这个值是复位向量,LSB 必须是 1。然后从这个值所对应的地址处取指。

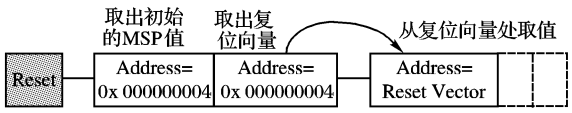

图3 复位序列

这与传统的 ARM 架构不同——其实也和绝大多数的其他单片机不同。传统的 ARM 架构总是从 0 地址开始执行第一条指令。它们的 0 地址处总是一条跳转指令。在 Cortex-M3 中,0 地址处提供 MSP 的初始值,然后就是向量表。向量表中的数值是 32 位的地址,即为该异常 handler 的入口地址,而不是跳转指令。向量表的第一个条目指向复位后应执行的第一条指令。

Cortex-M3 使用的是向下生长的满栈,MSP(主堆栈指针)要指向 SRAM 的后面。如果用户没有按下按钮,则首先判断 0x0800 2000 地址存放的是不是系统程序代码。

```
(( *(vu32 *) ApplicationAddress) & 0x2FFE0000 ) == 0x20000000)
```

因为 0x0800 2000 地址存放的是 MSP,这条语句是检测堆栈指针是否指向了一个有效的 SRAM 地址。但是它的主要作用是检测引导加载程序是否下载了一个有效的系统程序代码,或者 Flash 的内容是否因为在数据传输过程中或电源问题而被破坏了。如果是有效的系统程序代码,取出复位向量:

```
JumpAddress = *(vu32 *) (ApplicationAddress + 4);
```

函数指针指向复位向量指向的地址:

```
Jump_To_Application = (pFunction) JumpAddress;
初始化堆栈指针
__MSR_MSP( *(vu32 *) ApplicationAddress);
执行系统程序
Jump_To_Application();
```

表2 向量表结构

异常类型	表项地址偏移量	异常向量
0	0x00	MSP 的初始值
1	0x04	复位
2	0x08	NMI
3	0x0C	硬 fault
4	0x10	MemMange fault
5	0x14	总线 fault
6	0x18	用法 fault
7~10	0x1C~0x28	保留
11	0x2C	SVC
12	0x30	调试监视器
13	0x34	保留
14	0x38	PendSV
15	0x3C	SysTick
16	0x40	IRQ#0
17	0x44	IRQ#1
18~255	0x48~0x3FF	IRQ#2~#239

5 结语

STM32 系列 32 位闪存微控制器基于突破性的 ARM Cortex-M3 内核,是 ARM 公司专为嵌入式应用而开发的内核。本文针对智能终端的系统软件在线升级和维护困难的问题,以具有 Cortex-M3 内核的微处理器 STM32F103VB 为平台,usart 作为通信接口,给出了一种在线升级方案。在实际应用中,取得了良好的运行效果。通信接口如采用无线,则可实现无线升级。文中程序基于 RealView MDK 开发平台并调试通过。本方案稍作修改,即可应用于其他具有 Cortex-M3 内核的微处理器芯片。

参考文献:

[1] 张慧娟. ST 大幅扩展 STM32 微控制器产品系列[J]. 世界电子元器件,2008(7):118-118.

[2] 袁璐,宋华. 基于 Zigbee 和 IAP 的在线升级方案[J]. 测控技术,2008,27(10):79-82,85.

[3] 王伟,黄建娜. 基于 LPC2134 IAP 功能的数据存取实现[J]. 装备制造技术,2008(8):103-104,125.

[4] 彭华成,何岭松,许晓晖. 智能仪表软件远程升级的技术实现[J]. 机械与电子,2007(6):61-63.

[5] 姜晓梅,李祥和,任朝荣,等. 基于 ARM 的 IAP 在线及远程升级技术[J]. 计算机应用, 2008,28(2): 519-521.

[6] 李彩霞,潘峰. 数据传输中的 XMODEM YMODEM[J]. 武警技术学院学报, 1998,14(2):36-40.