BootLoader 程序在 MCF51AC 系列单片机上的实现①

刘 林，张晓丹，张作峰

(东方电子股份有限公司 技术中心，烟台 264000)

摘 要：BootLoader 程序几乎是所有智能设备的必备功能。当产品出厂后如果发现有重大 BUG 或需功能升级，则它可以帮助工程人员方便的更新程序。嵌入式软件一般与硬件联系比较紧密，因此 BootLoader 程序的实现在不同的硬件平台上具有不同的方法，但是其实现的基本思想是一致的。提出了在 BootLoader 程序实现过程中普遍会遇到的几个难点问题，并以飞思卡尔公司 ColdFire V1 微控制器 MCF51AC 系列单片机平台为例介绍了这几个难点问题的解决方法，对其他平台具有借鉴意义。

关键词：BootLorder；MCF51AC；引导程序；中断向量表

BootLoader Program Implementation on the MCF51AC Family of Microcontrollers

LIU Lin, HANG Xiao-Dan, ZHANG Zuo-Feng

(Technology Center, Dongfang Electronics Co. Ltd, Yantai 264000, China)

Abstract: BootLoader program is the essential function of almost all intelligent devices. If the products are found to have significant BUG or need to be upgraded after leaving the factory, it can help engineers to easily update programs. Generally speaking, embedded software is more closely linked with the hardware. So that the successfully application of BootLoader program on different hardware platforms are diversified, in which, however, the same basic idea is applied. This paper raises several problems often encountered in the operation process of the BootLoader program and introduces the solutions for them by taking ColdFire V1 microcontroller MCF51AC MCU platform of Freescale Company as an example, which will present some references for the other platforms.

Key words: BootLorder; MCF51AC; leading program; interrupt vector table

BootLoader（启动引导程序）是智能设备一般都具备的一个功能。虽然它不是用户所关心的一个功能，但是对于智能设备来说确是一个非常重要功能。大家都知道，无论水平多高的程序员，写出的程序不可避免的会存在问题。对于电力行业的智能设备来说，当设备已在现场运行，如何方便快捷的更换主程序，修改掉已知的问题就是急需解决的一个问题。这个问题正是由 BootLoader 来解决的。BootLoader 是在主程序运行之前运行的一段程序，一般比较简短，完成的功能也比较简单，最重要的功能就是完成主程序的更新[1,2]。

1 BootLoader实现过程分析

设备上电复位后，首先运行 BootLoader 程序。BootLoader 在对 CPU 进行必要的初始化后，对启动模式进行检测，一般情况下 BootLoader 会跳转到主程序运行，当检测到需要驻留运行时，BootLoader 才继续运行。这就需要一种方法通知 BootLoader 当前是需要驻留运行还是跳转到主程序中运行，常用的方法有通过跳线位置判断、通过超级终端发送固定的字符序列、通过按键等等。BootLoader 驻留运行后，会进行一些必要功能的初始化，如对 CPU 功能模块、串口等，初始化完成后进入通信任务循环，等待接收命令并执行

① 基金项目:安徽省教育厅自然科学基金(2005KJ004ZD)
收稿时间:2011-03-22;收到修改稿时间:2011-04-23

相应的命令；如果通信任务等待超时，则自动跳转到主程序运行。程序的流程图如图 1 所示。BootLoader 程序一般与硬件平台联系紧密，特别是对于嵌入式系统来说更是如此，不同的硬件平台 BootLoader 实现的方法都各不相同。但是在实现思路上基本是一致的，一般需要解决以下几个难点问题：

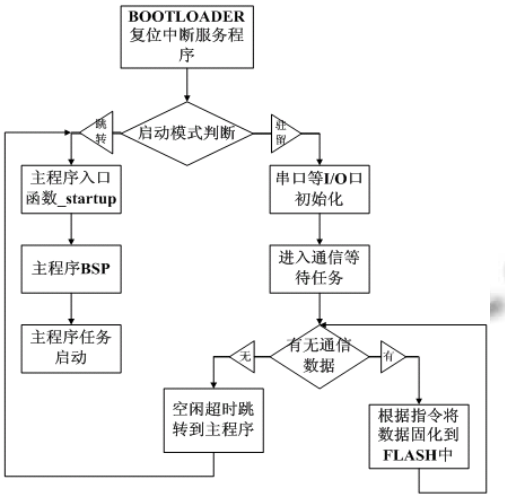

图 1 Bootloader 流程图

(1) 由于 BootLoader 与主程序是相互独立的程序，因此就需要解决两个程序的中断向量不一致的问题，通常微处理器中断向量的入口地址是固定的，并且不是可编程的，因此如何让两个程序都能正确使用自己的中断向量，是 BootLoader 实现的难点之一。

(2) 更新主程序实际上就是对存储程序的 FLASH 进行擦除和重新编程的过程。大部分的 FLASH 当程序在其中运行时，是不允许对其自身进行擦写操作的。对于支持外扩 FLASH 的微处理器，如 MC68332,解决这个问题的方法是用两片 FLASH，一片存储 BootLoader 程序，一片存储主程序，这样互相更新时就不存在冲突的问题了。有的微处理器虽然不支持外扩 FLASH，但是自身提供了某种机制，在固定的区域可以实现相似的功能，如 Atmel 公司的 Atmega 单片机。对于既不支持外扩 FLASH，自身也没有提供类似功能的微处理器，一般需要把相应的程序搬移到 RAM 中运行，如 MCF51AC 系列单片机，但是如何简单方便的把程序搬移到 RAM 中也是 BootLoader 实现的难点之一。

(3) 在程序更新的过程中，不可避免的要遇到把什么样程序文件下载到设备中问题，这就涉及到程序的存储格式问题。众所周知，计算机和单片机只能执行二进制代码，而各家编译器生成的映像文件其格式也是五花八门，如何把目标代码格式转换成计算机能认识的二进制格式，则是 BootLoader 实现的另一难点。

2 中断向量表映射

MCF51AC 系列单片机共包括 256 个中断向量，需要占用 1024 大小的空间，前 64 个中断用于内部中断使用，64~102 用于周边设备和 7 个软件中断，103~255 保留未使用。其中断向量表固定存放在 1M 地址的边缘，默认存放在地址为 0x000000 开始的 FLASH 区域且不可移动[3]。

正常情况下，要使用某个中断，只需要在中断服务函数前面加上编译指令 interrupt 和中断向量号即可，这时编译器就会把该中断服务函数的入口地址存放到相应的中断向量中。由于 BootLoader 程序和主程序分别生成不同的中断向量表且位置重叠，因此如果按正常情况使用就会造成两个程序的中断向量冲突。由于只有一张中断向量表，所以 BootLoader 程序考虑不使用中断模式，又考虑到 BootLoader 程序必须要使用复位向量（因为程序复位应首先跳转到 BootLoader 程序而不是主程序），再考虑到程序下载简单方便，决定使用中断向量映射表解决这个问题，下面详细介绍下中断向量映射表的生成及程序执行过程。

添加了中断向量映射表的 FLASH 空间划分示意图如图 2 所示。BootLoader 程序区固定从 0x410 开始，其大小可以根据需求自行设定；中断向量映射表固定从 0x5400 开始，其大小固定为 0x400；主程序代码区则从 0x5800 开始。

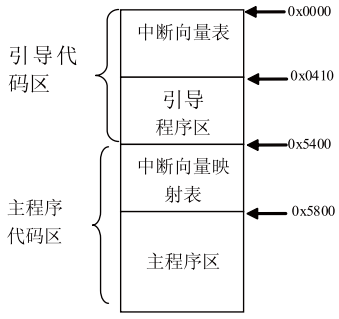

图 2 FLASH 空间划分示意图

添加中断向量映射表后程序的执行过程如图 3 所

示。首先程序复位或加电后从 0x000000 位置开始执行，这个位置存放的是复位相关的 2 个异常向量，一个是初始化 SP 堆栈指针，一个是初始化 PC 指针，这 2 个异常向量是 Bootloader 程序生成的，指向 Bootloader 程序的入口地址，Bootloader 程序运行后首先要做一个运行模式的判断，是进入 Bootloader 程序执行还是跳转到主程序执行，如果进入主程序，则通过中断向量映射表跳转到主程序的入口地址开始执行主程序。在主程序运行过程中假设发生一个 Vector N 中断，首先到中断向量表中读取指令，这个指令是指向中断向量映射表相应位置的，这时再读取指令就能跳转到正确的中断服务处理程序了。

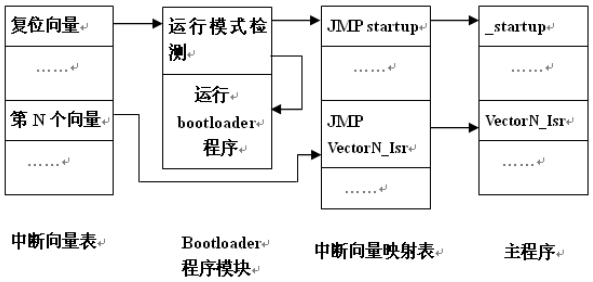

图 3　程序运行过程示意图

2.1 中断向量映射表的生成

中断向量映射表实际上是编辑一个文件，这个文件定义了 MCF51AC 系列单片机除 2 个复位相关异常向量外所有使用的 100 多个中断的中断服务程序，每个中断服务程序只有一条汇编的 jmp 语句，如果该中断没有被使用则跳转到固定的异常处理函数，如果使用了则跳转到真正的中断服务程序。

在这个映射表中没有定义中断向量表的前 2 个复位向量（堆栈和 PC 指针复位向量），只定义了一个主程序的入口函数 StartupEntry（），这个函数负责让程序跳转到真正的入口函数_startup（）。此外还定义一个异常处理函数 IllegalTrap（），这个函数内容是个死循环，程序如果跳到这里来表示有未使用的中断产生或有其他异常错误需要查找。

中断向量映射表生成后，在 Bootloader 工程和主程序工程中必须保持一致，只允许修改其中的函数内容。Bootloader 生成的程序是通过仿真器写到 FLASH 中的，会占用中断向量表、Bootloader 程序区；而主程序生成的程序通过 Bootloader 下载到 FLASH 中，占用中断向量映射表和主程序区。这样两个工程就可以随意添加文件和功能而不用考虑相互的影响了

2.2 FLASH 空间分配的实现

要实现如图 2 所示的 FLASH 空间的分配需要通过修改工程的链接命令文件 Project.lcf 来实现。这个文件是 Codewarrior IDE 工程向导在创建新工程时自动生成的，定义了 ROM 和 RAM 资源的大小及其分配空间。链接命令文件共包含两个字段，分别是 MEMEORY{} 和 SECTION{}。

首先需要修改的是 MEMORY{}字段,这个字段定义了 ROM 和 RAM 的空间分配情况，并给不同的空间定义了名称。我们把不同的空间称之为片段（segment）。缺省的 lcf 文件只定义了存放于 FLASH 的 code 段和存放于内存的 userram 段这两个片段，根据需要我们还要增加 BootLoad（存放引导程序）和 VectorRemap（存放中断向量映射表）这两个片段。增加后的 MEMORY{}字段如图 4 所示。接着需要修改的是 SECTIONS{}字段。

```
MEMORY {
    BootLoad    (RX)  : ORIGIN = 0x00000410, LENGTH = 0x00004F
    VectorReMap (RX)  : ORIGIN = 0x00005400, LENGTH = 0x000004
    code        (RX)  : ORIGIN = 0x00005800, LENGTH = 0x0003A8
    userram     (RWX) : ORIGIN = 0x00800000, LENGTH = 0x000080
}
```

图 4　lcf 文件的 MEMORY 字段定义

这个字段定义了程序的所有代码编译链接存放的规则。根据具体需求需要增加两个新的代码名称.text_bootload 和.text_vectormap。其目的是通知编译器，将源程序中标示为.text_bootload 的代码存放到 BootLoad 指定的空间，将 .text_vectormap 的代码存放到 VectorReMap 指定的空间。

源程序如果不加声明，默认都属于.text 代码，编译链接时会存放到 code 指定的空间。要把代码声明为其他代码名称就需要用预编译指令#pragma。声明的范围是从声明开始到文件结束或遇到其他的代码声明。

经过上述链接命令文件的修改和源程序的声明，就可以实现图 1 所示的 FLASH 空间划分。

3　FLASH在线更新程序的实现

MCF51AC 系列单片机不支持外扩 FLASH，因此要实现 Bootloader 的下载功能，只有把 Bootloader 相关的函数搬到 RAM 中运行才能实现。具体的实现还

是需要修改链接命令文件 Project.lcf。

在链接命令文件的 SECTIONS{}字段增加一个.ramcode 代码编译指令如图 5 所示。其意义是通知编译器，把源程序中标示为.ramcode 的代码编译到 userram（RAM 空间）中。众所周知由于 RAM 掉电丢失数据的特性，其中是不能存储代码数据的，因此这些数据其实还是存放在 ROM 中的某个位置，只是运行时从 ROM 中拷贝到 RAM 中而已。

```
.ramcode : AT(___ROM_AT1)
{
    *(.ramcode)
    . = ALIGN(0x4);

} >> userram
```

图 5 ramcode 代码编译指令

这样修改后只是通知编译器.ramcode 代码的存放位置和拷贝到 RAM 后的位置。要实现真正的拷贝工作还要修改 SECTIONGS{}字段后面.romp 代码编译指令区。修改后.romp 代码编译指令区如图 6 所示。该段代码的意义一个是通知编译器在_romp_at 地址存放.romp 代码，另一个是通知编译器需拷贝的内容和大小。

```
_romp_at = ___ROM_AT + SIZEOF(.data);
.romp : AT(_romp_at)
{
    __S_romp = _romp_at;
    WRITEW(___ROM_AT1);
    WRITEW(ADDR(.ramcode));
    WRITEW(SIZEOF(.ramcode));
    WRITEW(___ROM_AT);
    WRITEW(ADDR(.data));
    WRITEW(SIZEOF(.data));
    WRITEW(0);
    WRITEW(0);
    WRITEW(0);
}
```

图 6 romp 代码编译指令

经过上述修改后就完成了函数拷贝工作的定义。

4 创建Bootloader的复位中断服务程序

通过 Codewarrior IDE 工程向导创建一个新工程后，默认的入口函数（复位中断服务程序）是_startup（），其实现是在 startup.c 文件，把该函数定义为复位向量中断服务程序是在 exceptions.c 异常处理文件中实现的。由于该函数编译链接的存放地址是在 code 段，因此我们需要重新创建一个 Bootloader 的复位中断服务程序，并把他存放在 BootLoad 段。

Bootloader 的复位中断服务程序的创建很简单，只需要在函数名前加上预编译指令 interrupt 和 PC 指针初始化中断向量号即可。在该函数中，首先对 CPU 进行必要初始化，然后执行拷贝函数把 ROM 中的代码拷贝到 RAM 中执行。再接着是启动模式的判断，决定是跳转到主程序运行还是在 Bootloader 驻留运行。

5 程序格式转换

嵌入式系统编译器生成的映像文件格式有很多种，MCF51AC 系列单片机采用的是 S-记录格式。这种格式是 MOTOROLA 公司推荐采用的十六进制目标文件传送格式，主要优点是使传输过程可视。S-记录格式既包括通信协议，又包括计算机之间传输的程序和数据，还包括检错功能。

对于计算机来说，它只认识二进制的机器码，因此真正要把映像文件固化到 FLASH 上，还得需要对 S-记录格式的映像文件进行转化。如果把转化的工作放在嵌入式系统中，必然会增加程序对各种资源的消耗，因此最好把转换过程放在 PC 机上完成，通过 VC 等编写一个转换的小程序，把 S-记录格式的映像文件转换成 BIN 格式，再通过串口下载到设备中直接固化到 FLASH 上即可。

要对 S-记录格式进行转换就要对其编码规则有所了解。下面就介绍 S-记录格式的编码规则。S-记录是由几个字段组成的 ASCII 码字符串。这个字符串包括记录类型、记录长度、地址、代码/数据、校验和等。S-记录用两个 ASCII 码字符表示一个二进制数字节（8 位），第一个字符为高 4 位，第 2 个字符为低 4 位。组成一个 S-记录的 5 个字段及其内容如表 1 所示。

表 1 S-记录各字段内容

字段名称	字符数	内容
记录类型	2	共 8 种：S0、S1、S2、S3、S4、S5、S7、S8、S9
记录长度	2	记录中除记录类型和记录长度之外的字符对数目（十六进制）
地址	4、6 或 8	2、3 或 4 字节数据/代码存储地址，
代码/数据	0~n	0~n 个字节可执行代码
校验和	2	记录长度、地址、代码/数据字段数据之和的反码

S-记录定义了 8 种记录类型，其意义如下：

S0——每个 S-记录的记录头

S1——包括代码/数据内容和驻留代码/数据的 2 个字节首地址

S2——包括代码/数据内容和驻留代码/数据的 3 个字节首地址

S3——包括代码/数据内容和驻留代码/数据的 4 个字节首地址

S5——用一个指定模块发送的 S1、S2 和 S3 记录的数目，这个技术出现在地址字段，该记录无代码/数据字段

S7——S3 记录块的终止记录，其地址字段包括所使用的指令的 4 个字节地址，该记录无代码/数据字段

S8——S2 记录块的终止记录，其地址字段包括所使用的指令的 3 个字节地址，该记录无代码/数据字段

S9——S1 记录块的终止记录，其地址字段包括所使用的指令的 2 个字节地址，该记录无代码/数据字段

每个 S-记录一般只有一个记录头，一个终止记录，根据情况数据会使用 S1/S2/S3 记录。MCF51AC 系列单片机使用的是 S3 记录块。根据上述定义与解释很容易就可以通过 VC 编写一个格式转换程序将 S-记录格式的映像文件转换成 BIN 格式文件了。

6 结语

本论文介绍了 Bootloader 程序的功能及其执行过程，总结分析了 Bootloader 程序实现过程中可能遇到的几个难点问题，并以 MCF51AC 系列单片机平台为例，详细介绍了这几个难点的解决方案，在解决方案中详细介绍了 MCF51AC 系列单片机链接命令文件和 S-记录数据格式的内容，给想详细了解并使用 MCF51AC 的初学者提供了较好的参考案例。本论文提出的方法已经在实际产品中得到了应用，并稳定运行于多个现场。本论文提出的这几个难点具有普遍意义，其解决方案对于其他单片机平台实现 Bootloader 程序具有借鉴意义。

参考文献

1 徐宇清,黄彦平,夏耘.S3C44BOX 的 Bootloader 技术分析.上海理工大学学报,2005,27(4):369–372.

2 田冲,杨志.基于 AT91 RM9200 的引导程序分析与设计.微处理机,2008,10(5):137–140.

3 MCF51AC256 ColdFire Integrated Microcontroller Reference Manual. Freescale Semiconductor 2008.

(上接第 109 页)

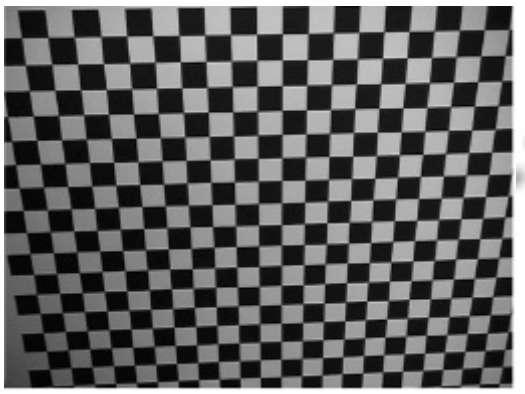

图 5 标定板校正图

4 结语

本文提出了一种基于等间距线的图像畸变校正方法，通过检测线段间距离来计算图像畸变中心和畸变参数，具有计算量小、精确度高的优点，能达到良好的图像校正效果。该算法具有良好的通用性和实用性，能广泛应用于图像畸变校正领域。

参考文献

1 Weng JY, Cohen P, Herniou M. Camera calibration with distortion models and accuracy evaluation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1992, 14(10): 965–980.

2 Zhang ZY. Flexible camera calibration by viewing a plane from unknown orientations. International Conference on Computer Vision (ICCV'99), Corfu, 1999. 666–673.

3 Lee SH, Lee SK, Choi JS. Correction of radial distortion using a planar checkerboard pattern and its Image. IEEE Trans. On Consumer Electronics, 2009,55(1):27–33.

4 Diego GA, Javier GL, Pablo RG. An Automatic Approach for Radial Lens Distortion Correction From a single Image. Sensors journal, 2011,11(4):956–965.