doi: 10.7690/bgzdh.2014.02.028

一种基于 TMS320C 6713 串口烧写 FLASH 实现自启动的方法

官琴[1]，赖文娟[2]，张筱波[3]
(1. 中国兵器工业第五八研究所特种电子技术部，四川 绵阳 621000；2. 中国兵器工业第五八研究所军品部，四川 绵阳 621000；3. 总装重庆军代局驻绵阳地区军代室，四川 绵阳 621000)

摘要：针对 TMS320C6713 平台烧写应用程序到 FLASH 中时存在的不便，介绍一种方便简洁的代码烧写方式。详细分析了 TMS320C6713 自启动原理、过程和烧写流程。从应用程序文件转换、串口发送应用程序和 TMS320C6713 硬件平台烧写 3 个方面详细描述了通过串口发送应用程序代码烧写 FLASH 的步骤。分析结果证明：该方法使用一种 CCS 版本即可完成 FLASH 烧写和应用代码的开发，并通用于各种版本 CCS 编译/开发平台，极大方便了软件开发人员。

关键词：TMS320C6713；FLASH 烧写；自启动
中图分类号：TJ02　　**文献标志码**：A

A TMS320C6713 Serial Programming FLASH Realization Method Based on Self Starting

Guan Qin[1], Lai Wenjuan[2], Zhang Xiaobo[3]
(1. *Department of Special Electronic Technology*, *No.* 58 *Research Institute of China Ordnance Industries*, *Mianyang* 621000, *China*; 2. *Department of Military Products*, *No*. 58 *Research Institute of China Ordnance Industries*, *Mianyang* 621000, *China*; 3. *PLA Presentation Office in Mianyang District*, *PLA Military Representation Bureau of General Equipment Headquarters in Chongqing*, *Mianyang* 621000, *China*)

Abstract: According to the TMS320C6713 platform programmer application to the FLASH in trouble. This paper introduces a simple code writing style. This paper analyzes TMS320C6713 since the start of principle, process and the writing process. Conversion from the application file types, serial port to send applications and TMS320C6713 hardware platform programmer three aspects described in detail through the serial port to send the application code programmer FLASH steps. Analyzed result proved the method can achieve FLASH programmer and application code is developed using a CCS version, and apply to various version of the CCS compiler/development platform, greatly facilitates for software developers.

Keywords: TMS320C6713; FLASH programmer; self-starting

0　引言

使用数字信号处理 DSP(digital signal processing)芯片的硬件平台，最终都需要脱离仿真器独立运行。这就需要 DSP 硬件平台在断电后仍能保存应用程序代码。TMS320C67X 是 TI 公司生产的 C6000 系列产品中唯一能做浮点运算的 DSP 芯片，当其时钟速率达到 300 MHz 时指令周期时间为 3.3 ns，在家庭音频、3D 图形、雷达、语音识别和控制等领域都有广泛的应用。TMS320C6713(以下简称 6713)系列没有自带 FLASH 存储器(以下简称 FLASH)，一般应用程序要烧写到外扩 FLASH 中，由 FLASH 引导自启动。

通常使用 FLASHBURN 将代码烧写到 FLASH 存储器。FLASHBURN 为 TI 公司为 C6000 系列开发的烧写工具，运行环境为 CCS3(code composer studio)或 CCS2[1]。如使用 CCS4 以上版本，开发代码和烧写就需使用 2 种 CCS 版本，很不方便。基于此，笔者选用 TI 公司 TMS320C6713 芯片，介绍一种通过串口将应用程序代码烧写到 FLASH 中的方式，该方式只需使用一种 CCS 版本即可完成 FLASH 烧写和应用代码的开发。

1　TMS320C6713 自启动原理

C6000 系列一般将应用程序代码烧写到外扩 FLASH 存储芯片中，再由 FLASH 引导自启动。但由于 FLASH 读取速度较慢，在上电复位后，6713 通过 boot 段代码把 FLASH 中的应用程序代码引导到内部 RAM 中执行[2]。

上电复位后，6713 会自动搬移 FLASH 起始地址开始的 1 kbyte 内容到内部 RAM，由 DMA 或 EDMA 自动完成。当 6713 开始执行程序时，首先从内部 RAM 的起始地址开始执行程序，这样就等于开始执行 FLASH 起始地址开始的 1 kbyte 的内容，即存放 6713 的启动程序 Bootloader。Bootloader 是一个复制程序，主要功能是将 FLASH 中存放的应用程序代码复制到 6713 的 RAM 中去执行[1,3-4]。

6713 上电复位后会加载 FLASH 起始地址开始

收稿日期：2013-09-31；**修回日期**：2013-10-16
作者简介：官琴(1979—)，女，四川人，本科，工程师，从事嵌入式软件开发研究。

1kbyte 内容，而这 1kbyte 内容存放就是 Bootloader 程序，这样就把 Bootloader 程序加载到 6713 起始地址开始的 1kbyte 空间中去，开始执行 Bootloader 程序。而 Bootloader 的功能就是把 FLASH 中存放的应用程序代码复制到 6713 的 RAM 中，当 Bootloader 程序在 6713 的 RAM 中执行完成后，所有的应用程序也就从 FLASH 中复制到 6713 的 RAM 中，然后 6713 调到 c_int00()复位中断去执行，开始执行应用程序，实现了 6713 的 FLASH 引导自启动[3]。

2 烧写流程及工具准备

由于 6713 系列多使用黑匣子方式工作，因此工作中一般通过外设触发其工作。本平台使用串口将.hex 文件发送到 6713 硬件平台，平台收到.hex 文件后立即烧写到 FLASH 中，即完成整个烧写过程。需要准备如下工具：

1) 具有 windosXP 操作系统工控机 1 台；2) 文件转换工具软件 hex6x.exe(TI 公司提供)；3) 自制 windosXP 操作系统串口发送程序。

6713 平台烧写流程如图 1。

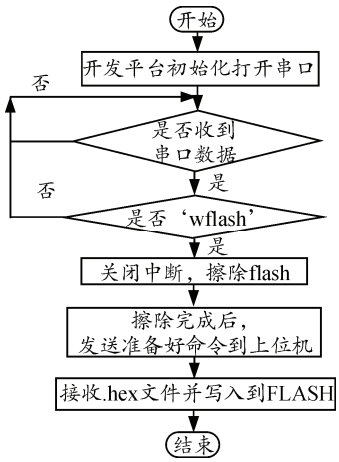

图 1 烧写流程

3 TMS320C6713 串口烧写步骤

3.1 文件转换

CCS 编译完成后生成的目标文件类型为.out 文件，是 6713 无法识别的文件类型，需要将.out 转换为 6713 可识别的十六进制文件类型.hex 文件，再将.hex 文件烧写到 FLASH 中[1,5-6]。具体操作如下。

将 hex6x.exe、.cmd 文件、.out 文件拷贝到同一目录中，执行如图 2 所示命令。

hex6x.exe 为 TI 公司提供文件转换可执行程序。

.cmd 文件负责 DSP 的内存分配。配置好 DSP 的内存才能使应用程序和 Bootloader 程序分配到 DSP 的内存空间。具体内容如下：

```
\code.out   //待转换 out 文件名
-a   //文件格式为 hex 格式
-memwidth   8//系统内存储器的每字节的位数，和 FLASH 一致，8 位
-boot //生成引导程序表
-bootorg   0x90000400 //程序代码首地址
-bootsection .boot_load   0x90000000//bootloader 首地址
ROMS
{
 FLASH:   org = 0x90000000, len = 0x80000,
 romwidth = 8, files = { code.hex}
}//org：flash 起始物理地址；
  len：flash 容量；
  romwidth：每字节位数；
  files：生成 hex 文件名
```

文件路径可以在.cmd 文件中自行设定。

转换后得到.hex 文件，该文件明确了各段的烧写地址。.hex 文件内容(其内容为十六进制)如图 3。

```
C:\>hex6x.exe fpga_loader_ahex.cmd
Translating \0228.out to ASCII-Hex format...
   "\0228.out"   ==> .boot_load
   "\0228.out"   ==> .hwi_vec    (BOOT LOAD)
   "\0228.out"   ==> .bios       (BOOT LOAD)
   "\0228.out"   ==> .pinit      (BOOT LOAD)
   "\0228.out"   ==> .text       (BOOT LOAD)
   "\0228.out"   ==> .cinit      (BOOT LOAD)
   "\0228.out"   ==> .switch     (BOOT LOAD)
   "\0228.out"   ==> .gblinit    (BOOT LOAD)
   "\0228.out"   ==> .sysinit    (BOOT LOAD)
   "\0228.out"   ==> .trcdata    (BOOT LOAD)
   "\0228.out"   ==> .rtdx_text          (BOOT LOAD)
   "\0228.out"   ==> .const      (BOOT LOAD)
   "\0228.out"   ==> .args       (BOOT LOAD)
   "\0228.out"   ==> .trace      (BOOT LOAD)
   "\0228.out"   ==> .log        (BOOT LOAD)
   "\0228.out"   ==> .sts        (BOOT LOAD)
```

图 2 hex6x.exe 执行过程

```
□
$A90000000,
C2 08 80 00 00 80 00 00 00 00 00 00 28 7C 80 01 68 00 80 01 28 00 00 02
68 C0 00 02 AA 04 00 00 6A 00 00 00 66 36 8C 02 C2 29 00 00 00 20 00 00
92 00 00 20 76 36 90 02 00 60 00 00 C2 08 80 03 C2 08 00 03 42 09 18 03
2A 00 80 03 6A 7C 80 03 2A 00 00 02 6A 04 48 02 28 00 82 01 68 00 C8 01
66 36 8C 00 E2 F7 0C 01 92 0F 00 60 2A 3C 80 01 6A 00 80 01 00 40 00 00
66 36 0C 00 E2 F7 0C 01 92 0B 00 60 2A 48 80 01 6A 00 80 01 00 40 00 00
64 36 0C 02 92 0A 00 30 00 80 00 00 E2 F7 0C 01 92 07 00 60 2A 58 80 01
6A 00 80 01 00 40 00 00 26 36 8C 02 C2 29 00 00 92 FF FF 2F 92 F8 FF 3F
C0 08 80 00 A0 67 8C 30 36 36 90 02 A0 87 8F 32 40 89 94 81 62 03 04 00
00 80 00 00 28 00 80 01 68 00 C8 01 42 29 18 03 F6 02 10 03 C2 08 00 01
62 03 0C 00 00 80 00 00 78 00 00 00 03 FF FF FF 33 BF FF FF 00 00 00 00
03 C9 75 52 22 8A A2 22 00 50 11 57 78 05 00 00 29 85 0A 00 00 00 00 00
$A90000400,
E0 B4 00 00 00 02 00 00 00 A4 00 00 2A 70 5A 00 6A 00 00 00 62 03 00 00
C2 08 00 00 A2 03 00 02 00 00 00 00 00 00 00 00 00 00 00 00 12 00 00 00
00 60 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 12 00 00 00 00 00 60 00 00 12 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 F6 54 3C 00 AA 01 00 00 F6 22 3C 00
2A 90 07 00 6A 00 00 00 62 03 00 00 E6 42 3C 00 00 60 00 00 F6 54 3C 00
2A 02 00 00 F6 22 3C 00 2A 90 07 00 6A 00 00 00 62 03 00 00 E6 42 3C 00
00 60 00 00 F6 54 3C 00 AA 02 00 00 F6 22 3C 00 2A 90 07 00 6A 00 00 00
```

图 3 .hex 文件内容

3.2 串口发送软件

串口发送软件的主要功能为读取.hex 文件，并将文件内容通过串口发送到 6713 硬件平台。

根据 6713 自启动原理，.hex 文件分为 Bootloader 和 code 段 2 个部分。图 3 中$A9000000 后的内容为 Bootloader 代码，从 FLASH 首地址 0x90000000 开始烧写；$A90000400 后为程序段代码从 FLASH 二次载入代码地址 0x90000400 中开始烧写。

(下转第 96 页)