TMS320C671X 系列 DSP 的引导程序设计技术

徐达旺　赵　浩　刘元商
（中国电子科技集团公司第四十一研究所　青岛　266555）

摘　要：TMS320C6000 系列 DSP 的程序上电加载过程比较复杂，为了简化设计，介绍了该系列 DSP 的一种引导程序的设计技术，即 DSP 的运行程序存放在外接只读存储器或 FLASH 中，上电复位后，通过 FLASH 进行多级引导，DSP 程序在内部 RAM 和外部 RAM(SDRAM)中运行。以 TMS320C6713 为例，对 TMS320C671X 系列 DSP 的引导方式和 FLASH 加载过程进行了说明，然后详细说明了二级引导程序的几种实现方式，提出了一个最易设计和维护的方法："启动时拷贝表法"，实现了"全自动引导"。该技术已经成功应用于某微波测量仪器。

关键词：TMS320C671X；Bootloader；SDRAM；FLASH；拷贝表
中图分类号：TP311.11 TP368.1 TN06　　**文献标识码**：A　　**国家标准学科分类代码**：510.10

Techniques of bootloader design for TMS320C671X series DSP

Xu Dawang　Zhao Hao　Liu Yuanshang
(The 41st Research Institute of CETC, Qingdao 266555, China)

Abstract: Program loading of TMS320C6000 series DSP after power on is somewhat complex. In order to make it easier, the technique of bootloader design for this series DSP is introduced in this paper. That is, the DSP program is stored in an external ROM or FLASH. After power on reset, it is booting up from FLASH, and running in internal RAM and external RAM (SDRAM). Firstly, as an example, the booting method of TMS320C671X series DSP and the process of FLASH booting are illuminated. Secondly, some approaches of second booting program are detailed discussed, and an easiest way of design and maintenance is given, that is "the booting copy table". Thus an automatic boot loading is accomplished. This technique has been used in some microwave measurement instrument.

Keywords: TMS320C671X; Bootloader; SDRAM; FLASH; Copy table

1　引　言

TMS320C671X 系列数字信号处理器(DSP)是 TI 公司的 32 位 DSP 芯片，广泛用于嵌入式系统[1-2]中。该系列 DSP 可配置成两种引导方式[3]，主机加载或仿真器加载、ROM 或 FLASH(以下简称 FLASH)加载。

目前，关于 FLASH 加载的引导过程有很多介绍，但或多或少需要手动维护，并且需要较多专业知识。

本文以 TMS320C6713 为例，介绍 FLASH 加载的全自动引导过程。

2　TMS320C6713 DSP 外接 FLASH 加载过程

基于 FLASH 启动加载流程如图 1 所示。

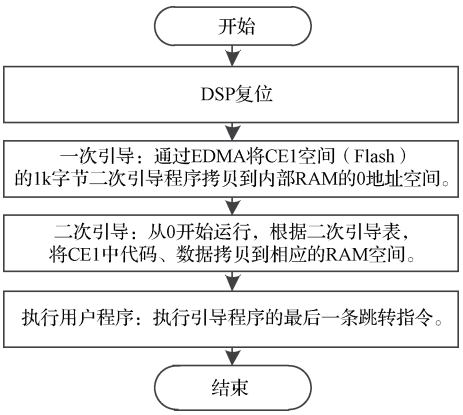

图 1　基于 FLASH 启动加载流程

收稿日期：2016-08

DSP内部复位完毕，将CE1空间(FLASH空间)开始的1k字节的ROM代码，使用缺省的ROM定时，通过EDMA方式拷贝到地址0，并从0处开始执行[3]。FLASH中的其他代码则需要用最开始的1k代码来再次引导(记为bootloader[4-5]，即二级引导)，直到FLASH存储器中的所有数据全部拷贝到DSP的RAM中。二级引导程序的最后是一条跳转指令，转到C程序入口函数c-int00，该函数启动用户的应用程序main[6-7]。

3 二级引导程序的编写

二级引导程序一般需要自己编写。如果应用程序不是很大，可以在DSP的内部RAM中运行。但是如果应用程序无法在内部RAM中运行，则二级引导程序必须将全部或部分代码或数据拷贝到SDRAM中。SDRAM一般为EMIF(external memory interface)接口的CE0空间，使用之前必须进行正确的配置才能使用。

3.1 为应用程序的节指定装入和运行地址

在链接应用程序时，用户需要考虑将应用程序的COFF(common object file format)节(SECTION)[8]放在哪里。在许多情况下，需要为该节指定两个不同的地址：装入地址和运行地址。装入地址决定该节的原始数据的装入位置。对于该节的任何引用(例如标签)指定其运行地址。这意味着当应用程序访问一个节时，将期望在其运行地址处找到该节。因此，如果为一个节指定不同的装入和运行地址，在使用前，该节必须从装入地址拷贝到运行地址。

在决定将节链接到哪里时，需要考虑以下几点。所有代码和已初始化数据在ROM中必须有一个装入地址。所有非常量数据在RAM中必须有一个运行地址。任何从ROM拷贝到RAM的代码应该给定合适的装入和运行地址。对于其他代码节，可以选择将他们保留在ROM中，即在ROM中有一个装入和运行地址。在ROM中有一个运行地址，可能效率比较低，因为ROM通常比RAM慢，然而这样将节约RAM的空间，并且不必拷贝该节。大多数情况下，速度更重要，应该在RAM中给出这些代码节的运行地址。一个值得注意的例外的节是，它包含初始化表或启动代码。这些节仅被使用一次，启动过程中，它首先将自己拷贝到RAM中。

将需要拷贝的节装入地址、运行地址等存放为一个表，称为拷贝表[8]，其数据结构为：

```
typedef struct copy_record {
    unsigned int load_addr;   //装入地址
    unsigned int run_addr;    //运行地址
    unsigned int size;        //节大小
} COPY_RECORD;
typedef struct copy_table {
  unsigned short rec_size;//单个记录大小
  unsigned short num_recs;//有多少个节
  COPY_RECORD recs[1];
} COPY_TABLE;
```

下面给出几种产生拷贝表的方式，并进行比较。

3.1.1 手工编辑和维护拷贝表

1)在编译应用程序时产生.map文件(包含装入地址、运行地址和大小)；

2)根据.map文件编辑拷贝表，修正装入地址、运行地址和大小；

3)重新编译应用程序，以合并已更新的拷贝表。

每次应用程序的一个较小改变，均需要重复上述过程，确保拷贝表中的内容是最新的。该方法的缺点是不易维护，在应用程序不大、且不太复杂时，此方法是可行的。

3.1.2 使用链接器操作产生装入地址、运行地址等

建立用户自定义的链接器(Linker)文件(.cmd)，用链接器操作来指定相应的地址和大小[8]。

链接器的操作包括：

LOAD_START(sym)或START(sym)：定义sym为相应分配单元的装入起始地址。

LOAD_END(sym)或END(sym)：定义sym为相应分配单元的装入结束地址。

LOAD_SIZE(sym)或SIZE(sym)：定义sym为相应分配单元的装入长度。

RUN_START(sym)：定义sym为相应分配单元的运行起始地址。

RUN_END(sym)：定义sym为相应分配单元的运行结束地址。

RUN_SIZE(sym)：定义sym为相应分配单元的运行长度。

例如：

```
SECTIONS {
    .text:
load = FLASH, run =SDRAM,
LOAD_START(_FLASH_code_ld_start),
RUN_START(_FLASH_code_rn_start),
SIZE(_FLASH_code_size)
}
```

_FLASH_code_ld_start等可以被拷贝表所引用。拷贝中的实际值在每次应用程序链接时自动更新。这样省略了3.1.1方法中的步骤1)。

虽然在维护拷贝表时显著减少了工作量，但仍然必须保持拷贝表的中内容与链接命令文件中的符号保持同步。理想的情况是，链接器自动产生拷贝表。这将避免两次编译应用程序，并且从管理启动拷贝表的内容中解脱出来。

3.1.3 由链接器自动产生拷贝表

利用链接器的操作table()来自动产生拷贝表。例如3.1.2中的例子可以写成如下形式：

```
SECTIONS {
.text:
```

```
load=FLASH,run=SDRAM, table(_copy_table)
}
```

自动产生拷贝表_copy_table,每个表含装入地址、运行地址和大小。

使用该方法,不用担心拷贝表的建立与维护了,可以在C/C++或汇编源代码中引用拷贝表的地址。该方法的缺点是需要对每个节、组(GROUP)或联合(UNION)[8]指定唯一的拷贝表名。

3.1.4 启动时拷贝表

链接器支持一个专用的拷贝表名,BINT或bint,可以用来建立运行时拷贝表。(该方法仅适用于Code Composer Studio 3.0及以上的版本[2])。

例如3)中的例子可以写成如下形式:

```
SECTIONS {
.text:
load = FLASH, run =SDRAM, table(BINIT)
...
}
```

链接器将建立符号为_binit_拷贝表,该表包含,在启动时,需要从装入地址拷贝到运行地址的所有的模块列表。如果未使用table(BINIT),则该值为-1,表明启动时拷贝表不存在。

不难看出,上述4种方法都可产生拷贝表,只是难易程序和可维护性不同。推荐使用第4种方法,这种方法基本不需要对拷贝表午行维护。

3.2 为DSP/BIOS的节指定装入和运行地址

在大多数情况下,用户的应用程序需要用DSP/BIOS,这样在应用程序中就包含了DSP/BIOS节。也可以为DSP/BIOS节指定不同的装入和运行地址。

DSP/BIOS的配置工具有两种:

1) 采用图形化配置工具,形成.cdb文件;

2) 采用文本配置(TextConf):保存后形成.tcf或.tci文件,运行脚本工具(conf)产生.cdb文件。

.cdb文件编译后产生包括链接文件(.cmd)在内的多个文件。链接文件不能由用户自己修改。因此不能用3.1中的方法2~4进行处理。但如果用户程序采用方法4,DSP/BIOS采用方法1,也不好,至少是一致性不好,且也增加了后级维护工作。

下面给出一个变通的方法,在DSP/BIOS的cmd文件中产生启动时拷贝表。

3.2.1 文本配置(TextConf)工具的使用

利用文本配置TextConf[9]工具产生DSP/BIOS配置文件myprg.tcf,并为有关的BIOS节(如.bios、.trcdata、.hwi_vec、.rtdx_text、.sysinit、.gblinit等)指定FLASH装入地址和内部RAM运行地址。

例如给BIOS节指定装入地址为FLASH,运行地址为内部RAM:

```
tibios.MEM.LOADBIOSSEG = FLASH;
tibios.MEM.BIOSSEG = tibios.IRAM;
```

3.2.2 编写修改tableBinitCmd.tci脚本文件

该函数的功能是为具有load>和run>的行的末尾加上“, table(BINIT)”,即建立“启动时拷贝表”。

调用函数调用前:

```
.bios:{} load > FLASH, run > IRAM
```

调用函数调用后:

```
.bios:{}load>FLASH run>IRAM, table(BINIT)
```

程序示例如下(编程语法为JavaScript[9]):

```
my_utils.tableBinitCmd=function(inFile, outFile) {
var loadMode=new RegExp("load\s*>", "i");
var runMode=new RegExp("run\s*>", "i");
var outLine = "";
var iFound = false;
var cLine;
if ((null == inFile) || (null == outFile)){
printf("参数错误。");
return(null);
}
var src = new java.io.BufferedReader(new java.io.FileReader(inFile));
while (null ! = (cLine = src.readLine())){
cLine = String(cLine);
if (null ! = cLine.match(loadMode)){
iFound = true;
while(null==cLine.match(runMode){
outLine += cLine + "\n";
cLine=String(src.readLine());
}
cLine = cLine + ", table(BINIT)";
}
outLine += cLine + "\n";
}
src.close();
if (false == iFound){
return(null);
}
var dst = new java.io.FileWriter(outFile);
dst.write(outLine);
dst.close();
return (outFile);
}
```

3.2.3 为BIOS的节建立拷贝表

在myprg.tcf中调用tableBinitCmd.tci中的tableBinitCmd函数修改.cmd文件,自动在装入地址和运行地址不同的节加上“table(BINIT)”。

示例程序如下:

```
my_utils = {};
```

```
utils.importFile("tableBinitCmd.tci");
prog.gen();
my_utils.tableBinitCmd(prog.name + "cfg.cmd",
prog.name + "cfg.cmd");
```

3.2.4 用 conf 产生.cdb 文件,并增加后编译选项

给.cdb 文件加上后编译步骤(Post-build steps),例如:

```
MYM(Install_dir)\bin\utilities\tconf\tconf -Dconfig.tiRoot="MYM(Install_dir)" -Dconfig.platform="Dsk6713" -Dconfig.importPath="tci;" myprg.tcf"
```

3.3 二级引导程序中的 EMIF 配置

由于二级启动程序需要将某些程序节拷贝到 SDRAM 中执行,因此,在拷贝程序之前必须 EMIF 接口配置。一旦 EMIF 配置完毕,CE0 空间(SDRAM)就可以由二级启动程序或用户程序所使用。特别需要注意的是,EMIF 只需要在二级启动程序中配置一次,不能在应用程序中进行二次配置,否则将重新初始化 EMIF 接口,导致在二级启动过程中拷贝到 SDRAM 中的程序和数据丢失,造成应用程序无法正常运行。下面以 90 MHz 外部时钟的 SDRAM 为例说明 EMIF 的配置。

SDRAM 配置包括全局配置(GCTL)、CE0 空间配置、SDRAM 控制配置(SDRAMCTL)、SDRAM 定时(SDRAMTIM)配置、SDRAM 扩展(SDRAMEXT)配置等。

将 GCTL 配置为 0x78,CE0 配置为 32 位的 SDRAM(0xffffbf33),SDRAMCTL 配置为 0x47115000,SDRAMTIM 配置为 0x2bf,SDRAMEXT 配置为 0xa8529。

EMIF 配置示例汇编指令[8,10]如下:

```
EMIF_GCTL   .equ   0x01800000
EMIF_CE0    .equ   0x01800008
EMIF_GCTL_V   .equ   0x00000078
EMIF_CE0_V   .equ   0xffffbf33
; *EMIF_GCTL = EMIF_GCTL_V
          mvkl   EMIF_GCTL,A4
     ||     mvkl   EMIF_GCTL_V,B4
          mvkh   EMIF_GCTL,A4
     ||     mvkh   EMIF_GCTL_V,B4
          stw    B4,*A4
; *EMIF_CE0 = EMIF_CE0_V
          mvkl   EMIF_CE0,A4
     ||     mvkl   EMIF_CE0_V,B4
          mvkh   EMIF_CE0,A4
     ||     mvkh   EMIF_CE0_V,B4
          stw    B4,*A4
```

3.4 二级引导程序中的拷贝过程

```
          mvkl   ___binit__, a3
          mvkh   ___binit__, a3
          ldw    *a3++, b0
          nop    4
          shr    b0, 16, b1
copy_section_top:
          ldw    *a3++, b4
ldw    *a3++, a4
          ldw    *a3++, b0
          nop    2
  [! b1] b copy_done
nop    5
copy_loop:
          ldb    *B4++,B5
          sub    b0,1,b0
  [b0] b      copy_loop
          nop    2
          stb    B5,*A4++
          nop    2
          b      copy_section_top
          sub    b1,1,b1
          nop    4
copy_done:
          mvkl .S2 _c_int00, B0
          mvkh .S2 _c_int00, B0
          b      .S2 B0
          nop    5
```

4 结 论

该方法与其他方法的对比如表 1 所示。

表 1 效果对比

项目	方法	
	启动时拷贝表	其他
地址维护	自动维护	手动维护
程序维护	无需修改引导程序	需要修改引导程序

介绍了 TMS320C671X 系列 DSP 的 FLASH 加载的实现过程,重点对程序中节的重定位进行说明。阐述了 TMS320C671X 系列 DSP 几种常用的 FLASH 加载方法,对其进行了比较,重点介绍了最易设计与维护的方法:“启动时拷贝表”,给出了比较完整的源代码,大大降低了用户的使用门槛,对使用 TMS320C6000 系列 DSP 的 FLASH 加载应用有很大的参考价值。

参 考 文 献

[1] 李树彪,韩敬伟. 基于多任务的智能测量仪器嵌入式软件设计[J]. 仪器仪表学报,2013,34(12):1-7.

[2] 刘明亮,朱江森. 数字信号处理对电子测量与仪器的影响研究[J]. 电子测量与仪器学报,2014,28(10):1041-1046.

[3] DANIEL K, GANGADHAR S. Creating a second-level bootloader for FLASH bootloading on TMS320C6000 platform with code composer studio 2.2[M]. Texas Instruments Literature, 2006.
[4] 邓国荣,刘厚钦. 基于 NOR Flash 的 OMAPL138 双核系统自举引导启动实现[J]. 电子技术应用, 2014, 40(2):19-22.
[5] 李飞平,卿鄢波,腾奇志,等. 基于 TMS320C6678 的多核程序加载研究与实现[J]. 电子技术应用, 2014, 41(3):31-34.
[6] 高源, 罗秋凤. 基于 DSP28335 程序移植方法的研究与实现[J]. 电子测量技术, 2013,36 (3): 84-88.
[7] 郭元兴, 朱楚为. 基于 TMS320C6713 大容量上电自举的实现[J]. 通信技术 , 2013,46 (5): 107-110.
[8] Number L. TMS320C6000 assembly language tools user's guide [M]. Texas Instruments Literature, 2011.
[9] DSP/BIOS textConf user's guide [M]. Texas Instruments Literature, 2006.
[10] 王浩. 基于 GCC 的 TMS320C67xx 汇编代码的优化[J]. 国外电子测量技术,2015,34(5):61-65.

作 者 简 介

徐达旺(通讯作者),1972 年出生,男,学士,高级工程师,主要研究方向为微波测量仪器。
E-mail: xudawang2008@163.com

是德科技推出新型 PXIe 机箱,帮助工程师高效满足 PXI 应用要求

新型 PXIe 机箱包含高中低 3 种档次,可以灵活适应各种研发和制造应用

2016 年 12 月 1 日,是德科技公司(NYSE:KEYS)日前宣布推出 3 种具有不同规格和性能的新型 PXIe 机箱。这些新型机箱包括 1)是德科技业内领先的 10 插槽第 3 代机箱,设计用于高性能、台式和研发应用;2)是德科技经济高效的 5 插槽第 1 代机箱和 3)是德科技重新设计的 18 插槽第 2 代机箱,具有改进的电源和有利于系统集成的新特性。

是德科技第 3 代 M9010A 10 插槽机箱针对模块散热和高性能而优化,可提供业界领先的声功率级与出色的插槽散热性能,满足高性能 PXIe 模块的要求。这款 10 插槽机箱为小通道数的研发应用提供了出色的平台,而此前介绍的第 3 代 M9019A 18 插槽机箱支持测试多通道、高性能制造应用,例如 MIMO 和 PA/FEM。当与是德科技 PXIe 高性能系统模块和 PC 主机适配器结合使用时,两种第 3 代 PXI 机箱均实现了与外部 PC 连接高达 16 GB/s 的系统带宽——这是业内首创。

M9010A 第 3 代 PXIe 10 插槽机箱提供的高性能特性包括:

1)全部混合接口的第 3 代背板,每个插槽都配有 ×8 PCIe 链路和 ×24(双链路)系统插槽

2)超静音、大容量散热系统

3)通过两个前面板 SMB 触发端口可访问 PXI 触发(0:7)

4)对于大型系统配置,可实现一键控制多机箱电源排序

5)高功率容量支持高性能 PXIe 模块

M9005A PXIe 5 插槽机箱是小型、低成本应用的最佳选择,例如低通道数的 VNA 系统。M9005A 可提供:

1)一个综合系统模块,具有一个×1 电缆接口,用于连接外部 PC

2)3 个混合插槽和两个 PXIe 插槽

3)第 1 代,×1 背板性能

M9018B 第 2 代 PXIe 插槽机箱是一个经济高效的平台,用于构建不需要第 3 代性能的大型系统。M9018B 是 M9018A 的更新版本并可提供:

1)全部混合接口的第 2 代背板,每个插槽配有混合的×4和×8 链路

2)高性能散热系统

3)通过两个前面板 SMB 触发端口可访问 PXI 触发(0:7)

4)对于大型系统配置,可实现一键控制多机箱电源排序

5)新电源为下一代 PXI 模块提供更大功率

是德科技公司市场总监 Neil Martin 表示"我们致力于通过提供广泛的 PXI 基础设施,覆盖从基础型到行业领先的高数据带宽各种选择,灵活满足工程师的需求。这款较小体积的机箱现在提供了新的选择,包括高带宽、第 3 代通信背板、基本性能和更低成本。"

是德科技快速的维修周转时间、业界领先的校准、内核交换策略以及标准的 3 年保修,可最大限度延长系统的正常运行时间,降低总体拥有成本。

有关是德科技 PXIe 机箱和控制器的更多信息,请访问 www.keysight.com/find/pxi-chassis。浏览产品图片,请访问 www.keysight.com/find/pxi-chassis_images。