高密度 SPI EEPROM——SA25C020 的 DSP 引导

■ 大连海事大学 杨兵 夏志忠

摘 要 介绍 TMS320VC55XX 系列 DSP 基于 24 位高密度 SPI EEPROM——SA25C020 的引导、启动加载方法；分析整个过程，并结合实例着重研究基于 C5509A 的引导、加载方法和实现；提供具体的电路设计和编制的相应实现软件。

关键词 SPI EEPROM boot TMS320VC5509A SA25C020

引 言

DSP 的引导是涉及 DSP 独立工作的关键性问题，通常采用的方法是由 Flash 等器件引导的，但是相对 Flash 的占用空间大、扇区擦除的难度和时延来说，SPI EEPROM 不失为一个好的选择。传统 EEPROM 的容量太小，无法充分利用 DSP 的程序空间。

SA25C020 的 2Mb SPI EEPROM 是以色列的 Saifun Semiconductor 公司于 2005 年推出的高密度 EEPROM 产品，是业界首个结合小型 SO8 封装、低功耗和高性能特点的器件，专为需要高耐用性和低功耗的应用而设计和测试，针对持续可靠的非挥发性存储方案。它的价位接近闪存(Flash)，加上其节省空间的封装形式，使得它成为 DSP 引导的一个新的选择。TI 公司的 TMS320VC5509A 是一款集成了 A/D、USB 接口等的便携式 DSP。最重要的是它支持 24 位的 SPI EEPROM 引导。这两款芯片的结合，易于形成空间小、功耗低的便携式解决方案，有助于新的小型化、低功耗应用的实现，如硬盘、光盘(包括 DVD)、机顶盒、打印机、游戏卡以及无线产品。

1 引导系统硬件设计

DSP 引导系统硬件配置框图如图 1 所示。其中 TMS320VC5509A 有两种封装形式，此处采用 PGE3 形式。一个可以独立运行的 DSP 系统必须包括：

◇ DSP 芯片；

◇ 电源、时钟以及必要的初始化外围设置；

◇ 用于引导的非易失性的程序存储器，如本设计中采用的 SA25C020；

◇ JTAG 接口用于外部下载程序。

2 引导关键技术

引导主要步骤如下：

① 利用 CCStudio2.0 建立应用程序，并产生 *.out 文件；

② 建立引导表，生成相关文件；

③ 将引导表转换为 DSP 可以加载的数据格式，生成 *.dat 文件；

④ 利用 CCStudio2.0 建立 EEPROM 烧写程序，将引导表写到 EEPROM 中。

2.1 引导程序制作

制作一个被加载的工程 xf。程序的功能是使 DSP 的 XF 脚接的 LED 闪烁。主要调用程序如下：

```
void XF_TEST() {
    int i,j;
    while(1) {                          //循环输出波形
      asm(" BSET XF ");                 //XF 设置为高电平
```

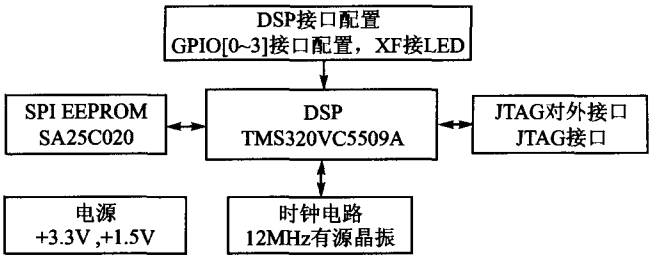

图 1 DSP 引导系统硬件配置框图

```
        for(i=0;i<5000;i++)          //时延
          for(j=0;j<100;j++){}
        asm(" BCLR XF ");            //XF 设置为低电平
        for(i=0;i<5000;i++)          //时延
          for(j=0;j<100;j++){}
    }
}
```

编译工程，在工程文件 xf/Debug 文件夹中生成 xf.out 文件。

2.2 引导表(boot table)制作

构造引导表的方法有两种：一种是使用 hex conversion utility；另外一种是根据 boot table 的结构自己在 SPI EEPROM 烧写程序中进行构造。对于第 2 种，需要构造者充分掌握引导表的构造(详情请见参考文献[1]，第 17 页)。这里仅说明使用 hex conversion utility 制作引导表的方法。下面阐述制作引导表所需要的环境和操作。

2.2.1 hex conversion utility 数据转换工具

hex conversion utility 是 TI 公司 CCS 中本身带有的一个数据转换工具。它有几个版本。对于 C54xx 使用的 hex500.exe，其绝对位置在\ti\c5400\cgtools\bin\hex500.exe；对于 C55xx 使用的 hex55.exe，其位置一般在\ti\c5500\cgtools\bin\hex55.exe。使用 hex conversion utility 工具还需要以下文件：

① *.out 文件，是 CCS 编译好的要存入 EEPROM 的 DSP 可执行文件。这个文件对于 hex conversion utility 是数据源文件。

② *.cmd 文件，用来填写 hex conversion utility 工具执行时的命令参数的文件。这些命令参数也可以在命令执行时写在命令的后面，而不采用 *.cmd 文件。

③ 输出文件是 hex conversion utility 生成的文件，可以是多种文件格式，在 *.cmd 文件的-o 参数中设置，如 ASCII - Hex、Intel、Motorola - S1/S2/S3、TI - Tagged、Tektronix 等。这里选用输出文件为 Intel 格式。

2.2.2 制作过程

在 2.1 节中已经制作好了 xf.out 文件。下面建立 xf.cmd 文件，内容如下：

```
xf.out      /* 设置输入数据源文件为 xf.out */
-boot       /* 标明本次操作是用于构造 boot table */
-o xf.io    /* 设置输出文件名称为 xf.io */
-i          /* 选择输出文件格式为 Intel 格式 */
-serial8    /* 串行 8 位数据 */
-v5510:2    /* 用于 VC5509/VC5509A 等的必要版本设置 */
```

编写好 xf.cmd 文件后，将 hex55.exe、xf.out、xf.cmd 放在同一个目录下。进入命令行并且到上述 3 个文件所在的目录下，执行命令“hex55 xf.cmd”，如图 2 所示。这样就可得到输出的引导表文件 xf.io。

```
D:\>cd xf\hex

D:\XF\HEX>hex55 xf.cmd
Translating xf.out to Intel format...
   "xf.out"    ==> .text          <BOOT LOAD>
   "xf.out"    ==> .cinit         <BOOT LOAD>

D:\XF\HEX>
```

图 2 hex55 执行效果图

2.3 引导表数据转换

引导表制作成功以后，其数据并不能被 DSP 直接读写，而需要把它转换成 DSP 可以读写的格式(即 CCS 数据文件格式)，才能把这些数据烧写到存储器中。这就需要对引导表文件进行数据转换。xf.io 的数据格式如下(其中黑体字为有效数据)：

:20000000**00002DC000000000000023E0000010046D**
3E6310000384A76138898AA31003853
:20002000**4A12A89021046444E6310000384B76006498A**
A3100384B12A89021760064A80404
…… (xf.io，具体格式请见参考文献[2]，第 14～39 页)

CCS 数据文件由 CCS 文件头和数据两部分构成。文件头指明文件类型、数据类型、起始地址和长度等信息，后为数据，每个数据占 1 行。以下即为 CCS 数据文件的文件头格式：

文件类型	数据类型	起始地址	数据页号	数据长度

采用 VC++编写程序 DSP_dataconvert 进行数据格式转换，将 Intel 格式的数据转换为 DSP 可以加载的数据格式。以下为 CCS 的数据格式，粗体字为有效数据。

1651 1 4000 1 14f ;1651 文件类型，1 表示十六进制格式
0x0000 ;4000 数据装载起始地址，1 数据页号，14f 数据
;长度
0x02DC
0x0000
0x0000
…… (xf.dat)

2.4 SPI EEPROM 烧写程序

SPI EEPROM 读过程操作一般要先执行 WREN 命令，打开写使能信号，RDSR 读取寄存器状态信号，WRITE 写 EEPROM。SA25C020 的指令结构和读写过程与普通的 16 位指令结构和读写过程是兼容的，只是在写地址时 SA25C020 的是 24 位的。

一般烧写程序中的执行顺序为：WREN→RDSR→WRITE

24 位 EEPROM 具体写程序如下：

```
void SPI_WrieSignal(Uint32 address, Uint32 * dataadd,
        Uint32 datalength) {
  Uint32 k =0;                  //计数器
  Uint16 EPROM_status = 0;//状态寄存器
  int i=0;                      //计数器
  SPI_WriteEN();                //写 EEPROM 使能 WREN
  EPROM_status = SPI_ReadStatusReg();
       //读 EEPROM 状态寄存器 RDSR
  while(! (EPROM_status & 0x2)){};
       //判断 EEPROM 状态寄存器的低二位是不是为 1
  hMcbsp = MCBSP_open(MCBSP_PORT0, MCBSP_
           OPEN_RESET);//打开 MCBSP0
  SPI_wrdatainit(hMcbsp);
                //初始化 MCBSP0 为时钟停止模式
  address = address & 0xFFFFFF;
                //获取要写的 EEPROM 的地址
  while(! MCBSP_xrdy(hMcbsp)){};
                //如果 EEPROM 已经准备好
  GPIO_RSET(IODATA,0x00);
                //复位 GPIO
  MCBSP_write32(hMcbsp,(SPI_WRITE + address));
                //写入 6 位命令和 24 位地址
  SPIWR_Delay(); //写时延
  for(i=0;i<datalength;i++) {
       while(! MCBSP_xrdy(hMcbsp)){};
                //如果 EEPROM 已经准备好
       MCBSP_write32(hMcbsp, * dataadd);
                //写入 32 位数据
       dataadd++;                //dataadd 自加 2
       SPIWR_Delay();            //读写时延
  }
  GPIO_RSET(IODATA,0x10);        //复位 GPIO
  for(k = 0; k <0x10000;k++)     //时延
  {}
  MCBSP_close(hMcbsp);           //关闭 MCBSP0
}
```

3 结 论

本设计方案大大节省了设计空间，降低了功耗，经过实际安装与调试完全可行。说明了基于高密度 SPI EEPROM 的 DSP 应用系统独立运行的过程。与同类的 SPI EEPROM 相比，具有容量大的特点；与 Flash 引导相比，具有读写、擦除简单的特点。硬件电路具有良好的可扩展性。以此电路为基础，可以适用于 C5509A 的小型化、低功耗和便携式应用的开发。

参考文献

[1] Using the TMS320VC5503/VC5507/VC5509/VC5509A Bootloader. Application Report SPRA375E, October 2004.

[2] TMS320C55X Assembly Language Tools User's Guide. Literature Number: SPRU280G March 2003.

[3] 刘益成. TMS320C54x DSP 应用程序设计与开发[M]. 北京：北京航空航天大学出版社，2002.

[4] TMS320VC5501/5502/5503/5507/5509/5510 DSP Multichannel Buffered Serial Port (McBSP) Reference Guide, Literature Number: SPRU592E April 2005.

[5] SA25C020 Advanced Information. 20, July, 2003.

(收稿日期：2006-08-11)

一棵以桌面为根节点的倒置树。对树进行“后根遍历”就能够容易地得到 Z 序。

图 7 描述了对象树的建立过程。对象树的采用极大地简化了桌面管理，能够在不增加额外工作的情况下方便地组合对象和实现 Z 序管理。

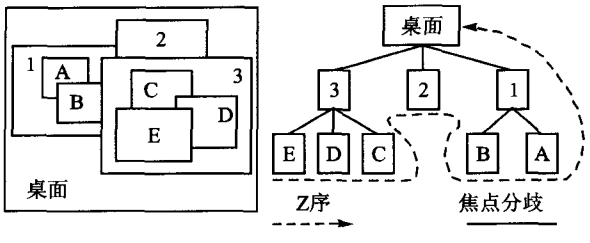

图 7 Z 序和对象树

3 小 结

将来的 GUI 系统将越来越复杂，要求 GUI 系统实现的功能也越来越丰富，这就需要一个更加开放且伸缩性好的体系结构。本文提出的嵌入式 GUI 体系结构具有很强的灵活性，且可移植性好，能够很好地应用于嵌入式领域的各种环境。

参考文献

[1] 罗蕾. 嵌入式实时操作系统及应用开发[M]. 北京：北京航空航天大学出版社，2005.

[2] Charles Petzold. Windows 程序设计[M]. 第 5 版. 北京：北京大学出版社，1999.

[3] Don Batory, Sean O'Malley. The design and implementation of hierarchical software systems with reusable components [J]. ACM TOSEM archive Volume1, ACM Press, 1992.

[4] Luo Qi, Luo Lei. A Universal Solution of an Embedded Multitasking GUI System [C]. ICESS'05 archive Volume 00, IEEE Computer Society, 2005.

[5] Richard N Taylor, et al. Chiron-1: a software architecture for user interface development, maintenance, and run-time support[J]. ACM TOCHI archive Volume2, Issue 2, ACM Press, 1995.

(收稿日期：2006-09-11)