doi: 10.7690/bgzdh.2014.12.017

一种基于串口的 DSP 程序烧写的方法

卢峥[1]，程涛[2]，李坤贺[1]，郭铁轩[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. 总装重庆军代局驻重庆地区军代室，重庆 400050；3. 总装重庆军代局驻重庆北碚区军代室，重庆 400700）

摘要：针对由于弹载计算机尺寸和结构的限制而导致更新 DSP 程序非常麻烦的问题，提出一种基于串口的 DSP 程序烧写方法。以弹载计算机最为常用的 DSP 处理器 TMS320C6747 为例，分析 DSP 的 BOOT 过程，给出串口烧写的实现流程和关键步骤的具体实现方法，并进行测试校验。结果表明：该方法稳定可靠，能较好地解决弹载计算机 DSP 程序的更新和烧写问题。

关键词：DSP；弹载计算机；TMS320C6747；中断
中图分类号：TJ711.02 **文献标志码：**A

DSP Programming Method Based on Serial Port

Lu Zheng[1], Cheng Tao[2], Li Kunhe[1], Guo Tiexuan[3]
(1. *Department of Special Electronic Technology*, *No*. 58 *Research Institute of China Ordnance Industry*, *Mianyang* 621000, *China*; 2. *PLA Presentation Office in Chongqing District*, *PLA Representation Bureau of General Equipment Department in Chongqing*, *Chongqing* 400050, *China*; 3. *PLA Presentation Office in Beibei District*, *PLA Representation Bureau of General Equipment Department in Chongqing*, *Chongqing* 400700, *China*)

Abstract: Because of the limitation of missile borne computer dimension and structure, it is hard to renew DSP program. Put forwards the DSP programming method based on serial port. Takes the example of the DSP processor TMS320C6747, which is the common processor for missile borne computer, analyze BOOT of DSP, give the implementation method of serial port programming realization process and key steps, and carry out testing. The result shows that the method is stable and reliable, and can solve renew and programming problems of DSP program.

Keywords: DSP; missile borne computer; TMS320C6747; interrupt

0 引言

在弹载计算机中，为保证系统的高可靠、低功耗和低成本，通常会利用 DSP 芯片来完成整个弹载计算机的控制和处理。目前弹载计算机中较为常用的 DSP 当属 TI 公司的 TMS320C6747，这是一款高性能和低功耗的 DSP，为当前 TI 主推的一款高性价比处理器。

TMS320C6747 程序烧写到启动 Flash 的主要方式：仿真器通过 JTAG 口连接 DSP，然后利用 CCS 自带的 FlashBurn 工具或者是用户自己编写的 Flash 烧写程序将 DSP 程序写入到 Flash 中[1]，无论采用哪种方式，烧写程序时都需要通过仿真器连接 DSP 的 JTAG 口。但是在目前的一些弹载设备中，由于整个设备体积和结构的限制，当弹载计算机装入弹载设备后，DSP 的 JTAG 口往往无法引出到设备外，因此当需要进行 DSP 程序更新时就必须将弹载计算机从设备中取出，非常麻烦并且不可靠。基于上述情况，笔者提出了一种基于 TMS320C6747 的串口实现 DSP 程序烧写的方法，解决了弹载设备装配完成后需要进行 DSP 程序更新时的问题。

1 DSP 的 BOOT 过程

TMS320C6747 内部没有用于用户程序存储的 ROM，掉电后内部所有单元的数据都会丢失，因此需要在设备上电时，通过外部设备将用户程序装入 DSP 的内部存储器中，实现程序的加载和启动，此过程叫作 DSP 程序的 BOOT 过程[2]。TMS320C6747 提供了多种 BOOT 方式，这些 BOOT 程序在芯片出厂时已经固化在专用的 ROM 中，通过相应的 BOOT 引脚状态可确定采用哪种启动模式。TMS320C6747 的启动模式选择如表 1[3]所示。

从表 1 可以看出：除了 HPI 和 UART 方式外，其余方式都需要在外部挂接 Flash 芯片，当设备启动时，根据 BOOT 的配置模式，DSP 自动从相应接口的存储设备将代码读入 DSP 中，因此，将最终的程序烧写到 Flash 中就成了 DSP 自启动的关键步骤。

收稿日期：2014-07-07；**修回日期：**2014-08-20
基金项目：国家科技支撑课题“群体性事件现场态势感知与信息综合处理技术研究”（2012BAK06B01）
作者简介：卢峥（1981—），男，山西人，硕士，从事信号与信息处理研究。

表 1 Boot Mode Selection

SPI1_CLK EQEP1S	SPI0_CLK EQEP1I	SPI0_SIMO[0] EQEP0S	SPI0_SOMI EQEP0I	nSPI0_ENA nUART0_CTS EQEP0A	Multiplexed Pins	
BOOT[7]	BOOT[2]	BOOT[1]	BOOT[0]	BOOT[3]	Boot Mode	AIS
0	0	0	1	X	NOR	Yes(1)
0	0	1	0	X	HPI	No
0	1	0	1	X	SPI0 Flash	Yes
0	1	1	0	X	SPI1 Flash	Yes
0	1	1	1	X	NAND 8	Yes
1	0	0	0	0	NAND 16(2)	Yes
0	0	0	0	0	I2C0 Master	Yes
0	0	0	0	1	I2C0 Slave	Yes
0	0	1	1	0	I2C1 Master	Yes
0	0	1	1	1	I2C1 Slave	Yes
0	1	0	0	0	SPI0 EEPROM	Yes
0	1	0	0	1	SPI1 EEPROM	Yes
1	0	0	1	0	SPI0 Slave	Yes
1	0	0	1	1	SPI1 Slave	Yes
1	0	1	1	0	UART0	Yes
1	0	1	1	1	UART1	Yes
1	0	1	0	0	UART2	Yes
1	1	1	1	0	Emulation Debug	N/A

2 串口烧写的实现流程

串口是弹载计算机外部最常用的接口，弹载计算机通常要引出多个串口到弹载设备外面，他们的主要作用包括炮弹或航弹发射前各类数据的实时装订，例如气温、气压、风速、风向和弹道系数，设备的实时控制等。

因此，笔者利用 DSP 的串口完成程序的烧写，其实现的流程如图 1[4]所示。

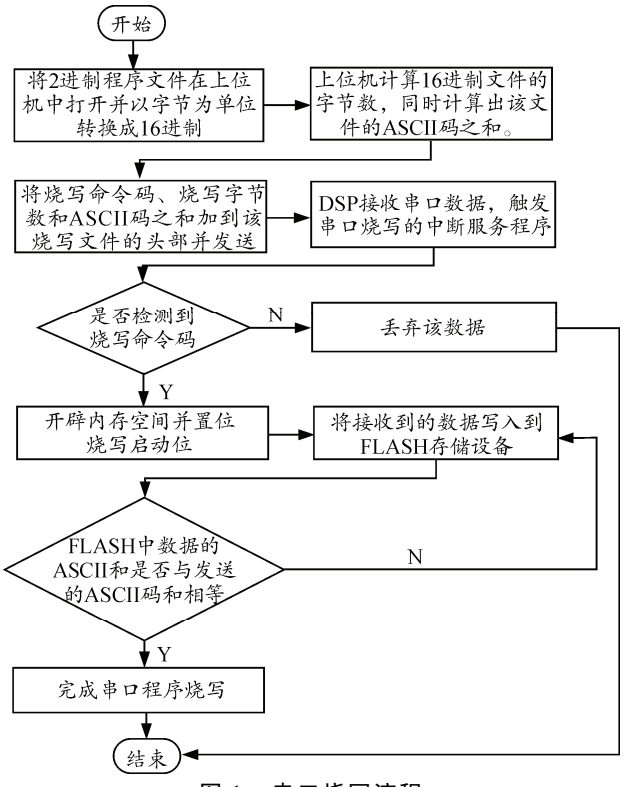

图 1 串口烧写流程

从上述流程图可以看出，DSP 通过串口烧写程序可分为以下几个步骤：

1) DSP 通过串口接收上位机发送的数据，触发 DSP 的串口中断服务程序。

上位机软件(串口调试助手或用户自己编写的串口收发软件)将待烧写的 DSP 二进制程序文件打开，并以字节为单位转换成十六进制，按照字节逐个相加，计算出待烧写数据的 ASCII 码之和。上位机按照事先定义好的数据帧格式将待烧写数据通过串口发送到 DSP 中。文中所定义的数据发送帧格式为烧写命令码(4 字节)+烧写数据字节数(4 字节)+烧写数据 ASCII 码之和(4 字节)+待烧写数据。

通过中断源、中断优先级和中断入口函数的设定，TMS320C6747 通过串口接收数据时将触发串口接收中断，程序进入 DSP/BIOS 中定义好的用于串口烧写的中断服务函数。

2) 在中断服务程序中根据烧写命令码判断该数据帧是否需要烧写数据，若为烧写命令，置位烧写启动位，则相应的烧写代码会把待烧写数据写入到 Flash。

通过串口接收到的前 4 字节数据判断是否是烧写命令，若检测到烧写命令码，则根据收到的烧写数据字节数在 DSP 的内存空间中开辟相应的存储空间用于烧写数据的存储，同时置位烧写启动位，该烧写启动标志位将触发主程序中的烧写代码，该代码通过 Flash 的写操作时序将存储的烧写数据按块写入到启动 Flash 中[5]。

3) 将写入到启动 Flash 中的烧写数据读出并进行 ASCII 码计算，判断是否与 DSP 串口接收到的烧写数据 ASCII 码之和相等。

当程序烧写完成后，为了防止数据接收过程和烧写过程中出现错误导致烧写失败，应对烧写到

Flash 中的数据进行校验。笔者通过 ASCII 码之和的方式进行了数据正确性的校验，校验方式可根据实际情况灵活选择。

需要特别说明的是为保证能触发串口接收中断完成烧写。DSP 第一次使用时，需要通过 JTAG 口将该段程序写入到 DSP 中。以后烧写时，将不再需要 JTAG 口，只需要将该段串口烧写的代码集成到用户应用程序中即可。

3　关键步骤的具体实现

根据串口烧写的实现流程可以看到，整个烧写过程大致可分为 3 个阶段：中断触发、数据烧写和数据验证。其具体实现流程如下：

首先通过 DSP/BIOS 设定 TMS320C6747 的中断源、中断优先级及中断服务程序的入口函数，如图 2 所示。

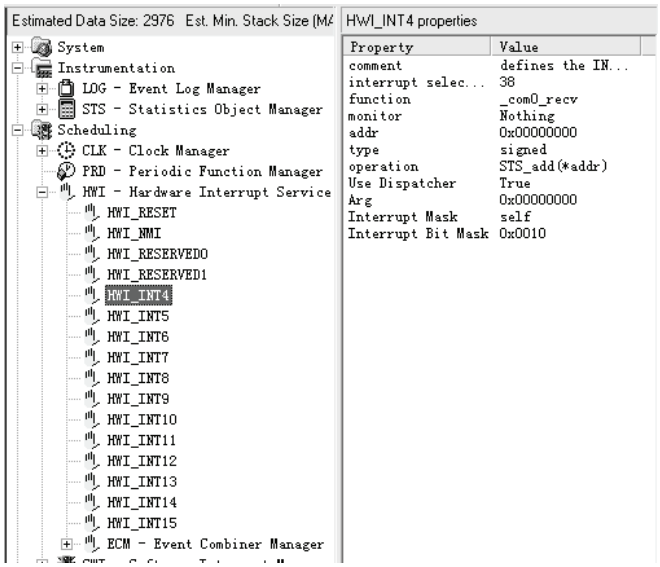

图 2　DSP/BIOS 中断设置

DSP/BIOS 是 CCS 中集成的一个简易的嵌入式实时操作系统，能够大大方便用户编写多任务应用程序。DSP/BIOS 拥有很多实时嵌入式操作系统的功能，如任务的调度，任务间的同步和通信，内存管理，实时时钟管理，中断服务管理等[6]。

通过如图 2 所示的设置，设置中断号为 38，当相应的串口接收数据时，将会产生串口中断，触发中断服务程序，进入 com0_revc 函数。

在中断服务程序中，首先判断该帧数据是否为烧写数据，在文中，4 字节串口烧写命令码定义为 CKSX。当检测到烧写命令码后，将烧写命令标志 flash_command 置为 true；接收随后的 8 个字节的数据，将烧写数据的字节数和 ASCII 码之和存到 counter_byte 和 counter_ascii 中；当把整个数据接收完成后将烧写启动标志 flash_write 置为 true，启动烧写。部分程序如下：

```
void com0_recv()
{
    uint32_t rtn;
    uint8_t   rcv_data = 1;
    while(CHKBIT(UART0->LSR, DR))
    {
        rtn = UART_rxByte(UART0, &rcv_data);
        xx[ii]=rcv_data;
        if(rtn == ERR_NO_ERROR)
            UART_txByte(UART0, rcv_data);
        if((xx[0]=='C') && (xx[1]=='K') &&
        (xx[2]=='S') && (xx[3]=='X')
        && (ii == 3))
        {
            flash_command=true;
        }
        if(flash_command && (ii==12))
        {
            memcpy(&counter_byte,&xx[4],4);
            memcpy(&counter_ascii,&xx[8],4);
            flash_length=true;
        }
        ii++;
        if((ii == (counter_byte + 12)) &&
flash_length)
        {
            flash_write = true;
            flash_command=false;
            flash_length=false;
        }
    }
}
```

在主函数中有一个 while 循环，一旦当烧写启动标志 flash_write 为 true 时，将开始执行 flash 烧写代码。

```
while(1)
{
  if(flash_write)
    {
    SPIFLASH_erase(0x00000000, counter_byte, 1);
    SPIFLASH_write(0x00000000, &xx[12],
    counter_byte, 1);
    }
}
```

通过 SPIFLASH_write 完成烧写后，需要对烧

写的数据进行校验，保证烧写数据的正确性。笔者通过 ASCII 码之和的方式进行数据验证，将 Flash 中的数据读出并进行 ASCII 码的累加后与 counter_ascii 的值进行对比，如相等则认为烧写数据正确。至此完成 Flash 的烧写工作，将 flash_write 重新置位为 false。

需要说明的是，笔者是对串口的烧写提供了一种方法。可根据具体需要重新定义串口烧写命令码，以防止误操作，同时也可以定义新的校验方式，进一步确保数据烧写的正确性。

4　结束语

笔者提出的一种基于串口实现 TMS320C6747 程序烧写的方法，提高了 DSP 程序烧写的灵活性，尤其是在系统中由于结构的限制、接口的需求等导致 DSP 部件装入设备后无法将 JTAG 接口引出时将有很大的作用。同时，该方法也可以引申到其余接口完成 DSP 程序烧写的方法，具有很大的灵活度，较好地解决了 DSP 程序烧写的问题。

参考文献：

[1] 李方慧, 王飞, 王佩琨, 等. TMS320C6000 系列 DSPs 原理与应用[M]. 北京: 电子工业出版社, 2003: 35-38.
[2] 汪安民, 周慧, 蔡湘平, 等. TMS320C674X DSP 应用开发[M]. 北京: 北京航空航天大学出版社, 2011: 361-363.
[3] Urmil Parikh and Joseph Coombs. SPRABB1C: Using the TMS320C6747/45/43 Bootloader[DB]. 2012-06.
[4] 官琴. 一种基于 TMS320C6713 串口烧写 FLASH 实现自启动的方法[J]. 兵工自动化, 2014, 33(2): 91-92.
[5] 陈雅芳. 宏程序编程与 Vericut 仿真技术的结合应用[J]. 机电工程, 2013, 30(8): 963-966.
[6] 彭启琮, 管庆. CCS 及 DSP/BIOS 的原理与应用[M]. 北京: 电子工业出版社, 2004: 46-47.

****

（上接第 58 页）

```
 for (int i = 0; i < dt.Rows.Count; i++)
{
   value = dt.Rows[i]["byjb"].ToString();
   if (value.Trim() == "一级保养")
   {
      _yjby++;
   }
……//统计“二级保养、三级保养”的次数
DataTable table = new DataTable("staticsby");
table.Columns.Add("byjb", typeof(string));
table.Columns.Add("value", typeof(decimal));
table.Rows.Add(new object[] { "一级保养", _yjby });
table.Rows.Add(new object[] { "二级保养", _ejby });
table.Rows.Add(new object[] { "三级保养", _sjby });
return table;
}
```

3　总结

结果表明：该系统不仅实现了对装备技术信息的快速查询、装备故障的交互性诊断，提高了现场维修能力，而且实现了对装备履历信息(保养记录、小中大修记录等信息)的统计，为装备保障下一步工作提供了数据参考与支持，具有广阔的应用前景。

参考文献：

[1] 于大海, 刘浩. 基于故障模式的交互式电子手册的设计与实现[J]. 四川兵工学报, 2010(6): 28-30.
[2] 梁晨, 糜玉林, 林瑜, 等. 基于 PMA 的 IETM 故障诊断系统设计[J]. 学术研究, 2011(10): 46.
[3] 朱兴动. 武器装备电子技术手册-IETM[M]. 北京: 国防工业出版社, 2009: 12.
[4] 高万春. 基于 Web 的协同性 IETM 结构模型及其应用研究[D]. 武汉: 华中科技大学, 2007: 1.
[5] 于守淼. 基于 Web 的军用飞机维修专家指导系统的设计[D]. 沈阳: 东北大学, 2008: 6.
[6] 倪绍徐, 张裕芳, 易宏, 等. 基于故障树的智能故障诊断方法[J]. 上海交通大学学报, 2008, 42(8): 1372-1375.
[7] 章涛, 张立新, 刘卫东. 基于电子履历的车辆管理电子档案系统研究[J]. 现代电子技术, 2011, 34(19): 198-200.