关于430单片机的二次升级

作者:佚名　来源:本站原创　点击数：960　更新时间：2012年06月01日　【字体：大 中 小】

注：　在近两个多月的工作与学习中，关于430单片机的二次升级我有了新的理解并通过编写程序代码在MSP430F448单片机中成功实现了程序的二次升级。就此心得与体会愿与曾经帮助过我的网友一起分享！

一、430单片机的升级方案

　针对TI公司430单片机的二次升级，网上也有不少网友在探讨这个问题，也给出了相关的设计方案，总结归纳有这么三种：

①　利用JTAG接口

② 利用BSL固件

③ 利用用户自定义的升级固件

二、本篇对①②不做讨论，针对第③种设计方案本人通过搜集资料、修改程序代码，理清思路，总结升级过程如下：

1、　　划分FLASH空间，

将FLASH空间划分为两段互不相干的程序存储空间，一段为升级引导程序固化区（同嵌入式ARM中的bootloader）另一段为应用程序区，本例以MSP430F448为例，划分空间如下：

4000-F5DF:　用户应用程序代码空间

F5E0-F5FF:　用户应用程序虚拟中断向量表

F600-FFDF:　升级引导程序代码空间

FFE0-FFFF:　升级引导程序中断向量表（即：硬件中断响应区，见方案<二>）

2、　　修改.XCK文件

如何划分FLASH空间是通过修改.XCK文件来实现的。打开C:\ProgramFiles\IAR Systems\Embedded Workbench Evaluation 5.0\430\config\目录以上地址（根据安装目录不同而定）。本例用MSP430F44

8　　单片机，所以要找到Ink430F448.xck文件，然后分别复制出来两个文件。并分别改名为Ink430F448_FlashUpdate.xcl（用于升级引导程序使用）和Ink430F448_FLASHAPP.xcl（用于用户应用程序使用）

①升级引导程序定位：
【1】在WE430编译引导程序工程之前，需要修改刚才一个Ink430F448_FlashUpdate.xcl文件
修改的目的：
首先要修改Ink430F448_FlashUpdate.xcl文件的部分数据如下红色所示。原单片机默认的程序的开始flash地址为4000H，现在由于要放置一段升级引导程序的需要，那么必须将引导程序放到一个指定的地址中去。本例划分后的引导程序空间为F600H-FFFFH，F600H地址开始就是我们的引导程序放置地址。
修改Ink430F448_FlashUpdate.xcl文件内容如下：

```
// ------------------------------------------------------
// Constant data
// ------------------------------------------------------

-Z(CONST)DATA16_C,DATA16_ID,DIFUNCT=F600-FFDF          //原为4000-FFDF
-Z(CONST)DATA20_C,DATA20_ID=F600-FFDF

// ------------------------------------------------------
// Code
// ------------------------------------------------------

-Z(CODE)CSTART,ISR_CODE=F600-FFDF                       //原为4000-FFDF
-P(CODE)CODE=F600-FFDF                                  //原为 4000-FFDF

// ------------------------------------------------------
// Interrupt vectors
// ------------------------------------------------------

-Z(CODE)INTVEC=FFE0-FFFF
-Z(CODE)RESET=FFFE-FFFF

// ------------------------------------------------------
// The end
// ------------------------------------------------------
```

注意：引导程序用的Ink430F448_FlashUpdate.xcl文件中的中断向量和复位地址是没有改变的，也就是说单片机上电的起始地址和中断向量的实质是以引导程序为主的。用户程序才是被调用的
【2】 编译操作：　在WE430建立一个工程文件，将引导程序所在的文Ink430F448_FlashUpdate.xcl加入工程并进行相关设置。其中在IAR WE430的项目选择中，设置：Options/linker/Config/Linker command file/Override default/Ink430F448_FlashUpdate.xcl文件。然后进行编译，当编译好的程序的

起始地址就会在F600H地址。此时可以在WE430环境中打开：View/Memory/Flash窗口，移到大概在FA00附近的地址处可以看到代码数据。

②应用程序定位：

【1】　　同样，要修改上面建立的Ink430F448_FLASHAPP.xcl文件。

修改目的：此次修改不同于上面的引导程序修改。原为4000-FFDF地址现改为4000-F5DF地址。从地址数上看程序空间变小了，这样可以保证应用程序空间不和引导程序空间相冲突。换句话说将原来的4000-FFDF空间分成了两部分，一小部分用于放置引导程序，一大部分空间用于分给用户应用程序空间使用。

另外，应用程序的中断时虚拟出来的，这个地址是由用户修改Ink430F448_FLASHAPP.xcl文件分配的。用户的应用程序的应用程序中断向量地址是虚拟的，再也不是由原来的FFE0-FFFF了。而被修改为F5E0-F5FF。当单片机接收到中断时单片机中断先跳去真实中断向量中，然后再通过内嵌汇编将PC转移到用户应用程序的虚拟中断向量地址中。

修改的Ink430F448_FLASHAPP.xcl的内容如下：

```
// ------------------------------------------------------------
// Constant data
// ------------------------------------------------------------

-Z(CONST)DATA16_C,DATA16_ID,DIFUNCT=4000-F5DF              //原为4000-FFDF
-Z(CONST)DATA20_C,DATA20_ID=4000-F5DF

// ------------------------------------------------------------
// Code
// ------------------------------------------------------------

-Z(CODE)CSTART,ISR_CODE=4000-F5DF                           //原为4000-FFDF
-P(CODE)CODE=4000-F5DF                                      //原为4000-FFDF



// ------------------------------------------------------------
// Interrupt vectors
// ------------------------------------------------------------

-Z(CODE)INTVEC=F5E0-F5FF                                    //原为FFE0-FFFF
-Z(CODE)RESET=F5FE-F5FF                                     //  原为FFFE-FFFF



// ------------------------------------------------------------
// The end
// ------------------------------------------------------------
```

【2】编译操作：在WE430建立用户应用程序工程，并进行相关的选项设置。其中，在IAR WE430的工程选项中：

[2-1] 设置：Options/linker/Config/Linker command file/Override default/Ink430F448_FLASHAPP.xcl文件

[2-2]设置： Options/Linker/Output/将输出文件选择为Other,使其输出为.txt文件。

[2-3] 编译：进行工程编译，并自动生成了.txt烧录文件在工程目录中。.txt文件是生成在工程文件夹的Debug\Exe\位置。

3、 生成有规律的用户应用程序代码.TXT件。

例：下为LED显示程序 .txt烧录文件

```
   @4000
31 40 00 0A B0 12 0C 40 B0 12 24 40 B2 40 80 5A
20 01 F2 43 1A 00 5E 42 19 00 C2 43 19 00 F2 43
19 00 FD 3F 30 40 28 40 30 40 2C 40 FF 3F
@F5FE
00 40
q
```

注：@为地址引导符，其后数据为地址，即接下来代码所存入的FLASH地址起始位置。

如：@后面的4000表示下面的数据内容需要写入4000开始的地址中，@后面的F5FE 是应用程序中断向量表中的复位地址，下面的数据是复位后PC的执行位置

q 为结束字符，表明用户应用程序已下载完毕

4、 编写升级引导程序（bootloader）

升级引导程序需完成的任务是将用户应用程序通过通过串口准确的接收，并通过处理将数据存入相应的FLASH地址空间中，接收完毕后能够执行用户应用程序，而与升级引导程序无任何影响。执行用户应用程序中可以通过特殊字符跳出用户应用程序，执行升级引导程序，从而实现用户应用程序的二次升级。

① 接收数据

单片机接收到串口发送过来的是有一定意义的ASCII码，因此在写入数据时首先要进行ASCII码与16进制数转换，另外串口发送数据过程中是以0X0A结束一行ASCII的发送的，因此我们采用单行处理写入模式，这样只需设置50个存储缓冲区就可以了，大大的节省FLASH的存储空间。当处理写入一行数据后，50个存储空间被释放以供下一行数据接收存储。

② 数据处理：

当接收到数据0X0A 表示一行数据接收完毕，接下来判断这一行数据的第一个字符是“q(结束标志)”，是“@（地址标志）”还是数据ASCII码

如果是“q”表示应用程序代码发送完毕，PC转移至应用程序复位向量跳出升级程序 执行应用程序

如果是“@”表示@后面的存储单元存入的是地址，则后面的存入的数据先进行ASCII码与16进制数转换，然后保存地址数据

如果不是以上两字符，则为数据字符先进行ASCII码与16进制数转换，然后写入到FLASH中的相应位置

③ 设置中断转移地址

数据接收完毕，也按照之前的约定保存到了FLASH空间的相应位置，还是不够的，因为PC无法自动跳入用户应用程序中，并且用户应用程序产生中断后，硬件响应的中断向量在FFE0-FFFF区中无法执行用户

应用程序的虚拟中断向量表，因此要在升级引导程序中内嵌汇编语言，当产生中断时将PC转移到虚拟中断向量表中，从而执行用户应用程序的中断子程序。

5、 实现用户应用程序的二次升级

通过前面四个步骤基本完成了430单片机程序的一次升级，完全可以通过串口发送用户应用程序从而执行应用程序。那么如何从应用程序中跳出，如何下载其他的用户应用程序呢，这就是用户应用程序的二次升级了，

实现用户应用程序的二次升级，必须在升级引导程序和用户应用程序中进行改进。

① 用户应用程序可以通过接收特殊字符如“$”“@”等，执行内嵌汇编语言改变PC值，从而跳出用户应用程序，跳出应用程序前必须擦除虚拟中断向量表以防止执行升级引导程序时再次进入用户应用程序

② 升级引导程序中必须能接收PC从用户应用程序中跳出，当从用户应用程序跳出执行升级引导程序时，此时是不可以发送第二套用户应用程序的，因为在划分好的flash用户应用程序地址空间被之前的用户应用程序所占用了，在430单片机中是不可以进行数据替换的，这就要求在进行第二套程序升级前必须通过串口接收特殊的字符来进行Flash数据擦除工作，只有将用户应用程序flash地址空间擦除空了才能正常的接收第二套用户应用程序。

<到此，基本可以实现430单片程序的二次升级了，

程序代码不便公布>

注：

这种升级方案是网上讨论蛮多的，经本人的添枝加叶总算是实现430单片机程序的二次升级，但这种升级的方案存在着许多弊端，其一、串口接收用户应用程序的波特率要求设置极低，以致接收处理数据过慢，升级一次程序需等待3-4分钟，如果波特率设置过高，数据接收过虽快，数据处理不过来容易遗漏数据，使用户应用程序无法正常运行。

其二、升级引导程序不可用中断，如果用上中断会与内嵌汇编语言的转中断向量表产生中断冲突，从而使程序跑飞，无法正常运行。

方案<二>

基于以上方案存在的缺陷，本人通过实验得出的经验而设想：

在进入用户应用程序，执行用户虚拟复位向量后，在用户应用程序开始将升级引导程序向量表读出并保存，将用户的虚拟中断向量表读出并拷贝到硬件中断响应区(FFE0-FFFF)中，

当退出用户应用程序前将升级引导程序的中断向量表拷贝到硬件中断响应区。这样就不需要在升级引导程序中设置内嵌汇编语言转移中断向量表了，用户应用程序的中断和升级引导程序的中断将不会起冲突了，也解决数据处理慢的问题！

经过多次实验证明，这种方案的确是可行的！

此方案的具体步骤同上所述，在上面的基础上去掉内嵌汇编语言转中断向量表，升级引导程序可设置串口数据接收中断、波特率能设置到115200已达到快速完整的程序升级之目的。由于升级引导程序去掉了内嵌汇编语言转中断向量表的功能，使得应用程序无法正常响应中断。这时必须在用户应用程序中加一段中断向量表处理的程序的.c文件，中断向量表处理文件需实现一下功能：

1.、进入应用程序，先保存硬件中断响应区（MSP430F448的硬件中断响应区为0FFE0-0FFFF）中的引导程序中断向量表，并写入用户应用程序FLASH地址中保存；然后擦除硬件中断响应区中的引导程序中

断向量表；最后从应用程序FLASH地址中读出用户应用程序中断向量表，并写入到硬件中断响应区的FLASH地址中。完成此过程后，用户应用程序与升级引导程序完全无任何关联，用户应用程序产生中断不再通过执行引导程序的内嵌汇编语言转中断向量表了，而是直接由硬件中断响应区跳入用户应用程序的中断入口，从而执行用户应用程序的中断子程序。

2、退出应用程序与进入应用程序功能相反，在退出用户应用程序前，先保存硬件中断响应区（MSP430F448的硬件中断响应区为0FFE0-0FFFF）中的用户应用程序中断向量表并写入到用户应用程序的FLASH地址中保存；然后擦除硬件中断响应区中的用户应用程序中断向量表；最后从用户应用程序FLASH地址中读取在第一步保存的引导程序中断向量表，并写入硬件中断响应区中以还原引导程序的中断功能，通过开门狗复位来跳出用户应用程序执行升级引导程序。

注：用户应用程序中断向量表和引导程序中断向量表要区分开，分别保存在用户应用程序的FLASH地址中不同的位置。

3、同样在执行二次程序升级前必须擦除用户应用程序FLASH地址中的程序代码

4、在以上步骤后基本完成了二次升级，但在经试验过程中发现，在执行升级过程中没有意外断掉的情况下，是完全可行的。当执行用户应用程序中无意断电，单片机重新启动时，用户应用程序无法正常运行。发送返回命令无效，程序进入死机状态。经过试验和测试以及理论分析，在进入用户应用程序时，程序先拷贝硬件中断响应区的中断向量表，并保存。而执行用户应用程序时，硬件中断响应区中断向量表为用户应用程序的中断向量表，程序误将用户应用程序中断向量表当成引导程序中断向量表保存到了引导程序中断向量表的FLASH地址中，再发送命令跳出用户应用程序时拷贝过来的实际是用户应用程序的中断向量表而并非引导程序的中断向量表。因此退出后无法正常执行引导程序。

解决此过程中造成的误操作，需在进入用户应用程序时设置一标志置位，并将其状态写入FLASH地址中保存，断电或重启后先读此标志，如果是置位状态，则不再处理中断向量表问题，直接执行程序。

剩余字数：400 提交发表 查看全部评论

站长：胡琴 站长信箱：aaf1@21cn.com