**STM32出现HardFault\_Handler故障的原因主要有两个方面：**  
  
1、内存溢出或者访问越界。  
2、堆栈溢出。  
最近遇到的问题是栈溢出，情况是这样的，举例说明：  
  
static char data[10000];  
void fun1(unsigned char \*buf)  
{  
int  i=0;  
for(i=0; i<5000; i++)  
{  
data = buf;  
}  
}  
  
void fun2(void)  
{  
unsigned char buf[5000];  
.........;  
fun1(buf); //执行完毕此函数出现硬件错误HardFault\_Handler  
printf("data: %s\r\n",buf);  
}  
  
int main()  
{  
.........();  
.........();  
.........();  
fun2();  
.........();  
.........();  
.........();  
while();  
}  
  
问题分析，通过断点代码跟踪，在进入fun1(buf);函数时，发现SP指向了数组data所开辟的空间，同时PC、等寄存器值压入栈，在循环执行data =buf;的时候修改了压入栈的数据，导致在退出函数fun1(buf);时PC指向了错误的位置。  
问题：为什么SP会指向数组data所开辟的空间？原因是发生了栈溢出。  
问题：那里导致了堆栈溢出呢？ 下面我们看下面的网络资料，认识一下堆栈。  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
int main()  
{  
while(1);  
}  
BUILD://Program Size: Code=340 RO-data=252 RW-data=0ZI-data=1632    
编译后，就会发现这么个程序已用了1600多的RAM，这1600多的RAM跑哪儿去了，分析map,你会发现是堆和栈占用的  
在startup\_stm32f10x\_md.s文件中，它的前面几行就有以上定义，这下该明白了吧。  
Stack\_Size     EQU    0x00000400  
Heap\_Size      EQU    0x00000200  
  
**理解堆和栈的区别**  
  
（1）栈区（stack）：由编译器自动分配和释放，存放函数的参数值、局部变量的值等，其操作方式类似  
     于数据结构中的栈。  
（2）堆区（heap）：一般由程序员分配和释放，若程序员不释放，程序结束时可能由操作系统回收。分配  
     方式类似于数据结构中的链表。  
（3）全局区（静态区）（static）：全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态  
     变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。程序结束后由系  
     统自动释放。  
（4）文字常量区：常量字符串就是存放在这里的。  
（5）程序代码区：存放函数体的二进制代码。  
  
例如：  
    int a=0;                    //全局初始化区  
    char \*p1;                   //全局未初始化区  
    main()  
    {  
     int b;                    //栈  
     char s[]="abc";              //栈  
     char \*p3= "1234567";          //在文字常量区  
     static int c =0 ;            //静态初始化区  
     p1= (char \*)malloc(10);         //堆区  
     strcpy（p1,"123456");          //"123456"放在常量区  
    }  
所以堆和栈的区别：  
    stack的空间由操作系统自动分配/释放，heap上的空间手动分配/释放。  
    stack的空间有限，heap是很大的自由存储区。  
    程序在编译期和函数分配内存都是在栈上进行，且程序运行中函数调用时参数的传递也是在栈上进行。  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
明白堆栈的分配原理后，我们也就明白了为什么说是栈溢出了，而没有说是堆栈溢出或者堆溢出，我们接下来再来分析什么导致了栈溢出，这会不难发现真凶是unsignedcharbuf[5000];，**buf的开辟占用了很大的栈空间，超出了startup\_stm32f10x\_md.s文件中定义的空间大小，导致了栈的溢出。**

问题总结：

**1、函数内部变量占用空间较大时，定义为全局变量或者静态变量，减少堆栈的占用。  
2、多使用指针解决数据的复制，同时减少内存的占用。**