思路:

常把单片机系统的复位分为**冷启动和热启动**。所谓冷启动，也就是一般所说的**上电复位，冷启动后片内外 RAM 的内容是随机的**，通常是 0x00 或 0xFF；单片机的热启动是通过外部电路给运行中的单片机的复位端一复位电平而实现的，也就是所说的**按键复位或看门狗复位**。复位后，**RAM 的内容都没有改变。在某些场合**，必须区分出设备的重启是热重启还是冷重启。常用的方法是：确定某内存单位为标志位(如 0x40003FF4～0x40003FF7 RAM 单元)，启动时首先读该内存单元的内容，如果它等于一个特定的值(例如为 0xAA55AA55)，就认为是热启动，否则就是冷启动。

根据以上的设计思路思路定义一个变量：

uint32 unStartFlag；

在程序启动时判断：

if（unStartFlag==0xAA55AA55）

{

//热启动处理

}

else

{

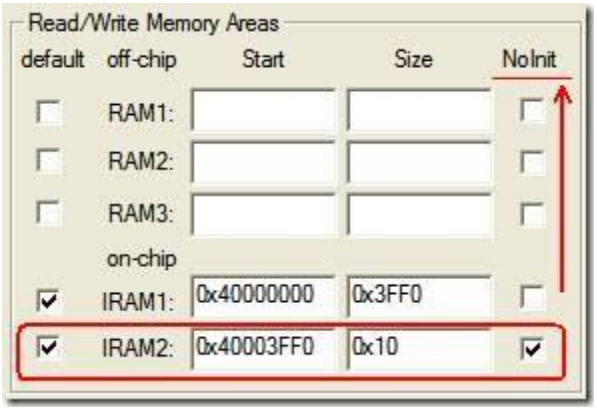
//冷启动处理

unStartFlag=0xAA55AA55；

}

然而实际调试中发现，无论是热启动还是冷启动，**开机后所有内存单元的值都被复位为 0**，当然也实现不了热启动的要求。通过看 keil MDK 自带的启动代码 Startup.s，在这个启动代码中也并没有发现将整个 RAM 区域清零的语句。反汇编程序，发现从启动代码执行结束到跳转到 main 函数过程中，编译器还执行了很多库函数，**其中__scatterload_zeroinit 函数将所有 W/R RAM 都初始化为 0（默认设置下）。**为了判断冷、热启动，必须人为控制某些

特定 RAM 在复位时不被编译器初始化为 0。通过查找编译器手册，在为处理器的 RAM 中分出一块小片 RAM，设置为 NoInit 格式（不对其初始化为 0），如下图：



然后使用__at 关键字将冷、热启动标志位定位到这个 NoInit 区域：

uint32 unStartFlag __at (0x40003FF4);

这样，当热启动时，变量 unStartFlag 所在的内存区域就不会被初始化为 0，也实现了冷热启动的判断。

定义铁电 0xFF7～0xFF8 区域存储冷启动次数

0xFF9～0xFFA 区域存储热启动次数

0xFFB～0xFFC 区域存储总启动次数