ARM Cortex-M底层技术（九）KEIL MDK 分散加载示例1-更改程序运行基址 - weixin_39118482的博客

本文链接：https://blog.csdn.net/weixin_39118482/article/details/80066566
KEIL MDK 分散加载示例1-更改程序运行基址

小编我一向主张在实战中学习，不主张直接去去学习规则&定义，太枯燥，在实际应用中去摸索，才会真正理解具体的技术细节，下面我们就通过实际的简单用例来搞清楚分散加载。

更改程序运行基址

我们先来做一个最简单的示例，然后逐步深入。

这里选择一颗简单一些的MCU，LPC824，一颗M0+作为示例（因为其比较简单，作为示例比较合适），我们先来看其默认的分散加载：

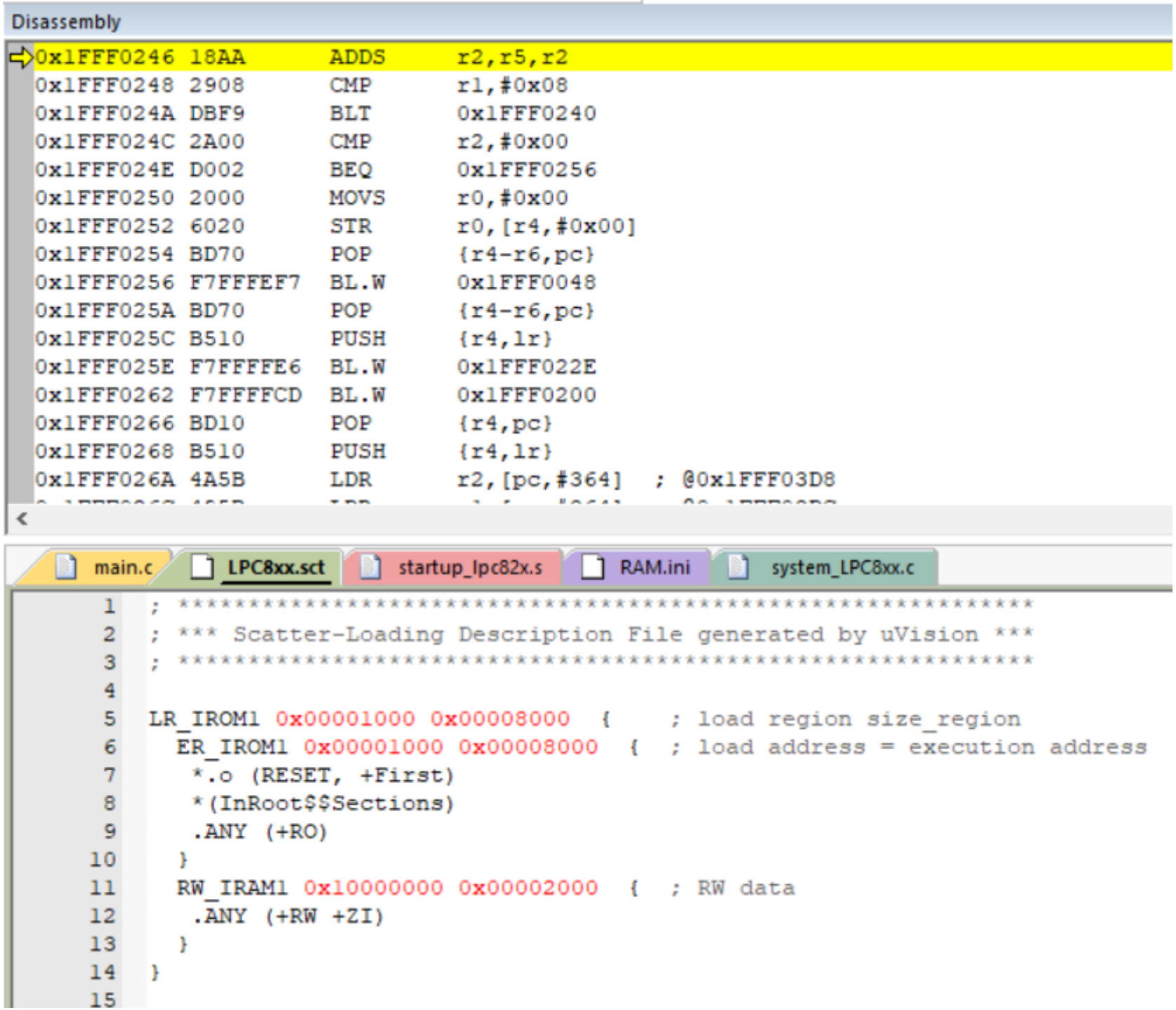
```
LR_IROM1 0x00000000 0x00008000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00008000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00002000  {  ; RW data
   .ANY (+RW +ZI)
  }
}
```

加载域地址从0x00000000地址开始，大小为0x8000（32KB），运行域（RO）从0x00000000开始，运行域（RW+ZI）从片内SRAM地址开始0x10000000。

我们尝试让程序从0x00001000地址上开始运行，我们该如何修改？加载域以及RO运行域直接改成0x00001000？？？可行吗？我们试试，我们把分散加载修改过来试试，看程序能不能运行。

```
LR_IROM1 0x00001000 0x00008000  {    ; load region size_region
  ER_IROM1 0x00001000 0x00008000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x10000000 0x00002000  {  ; RW data
   .ANY (+RW +ZI)
  }
}
```

按照上图修改分散加载，我们点击DEBUG，看程序是否能够正常调试，程序进入如下状态：

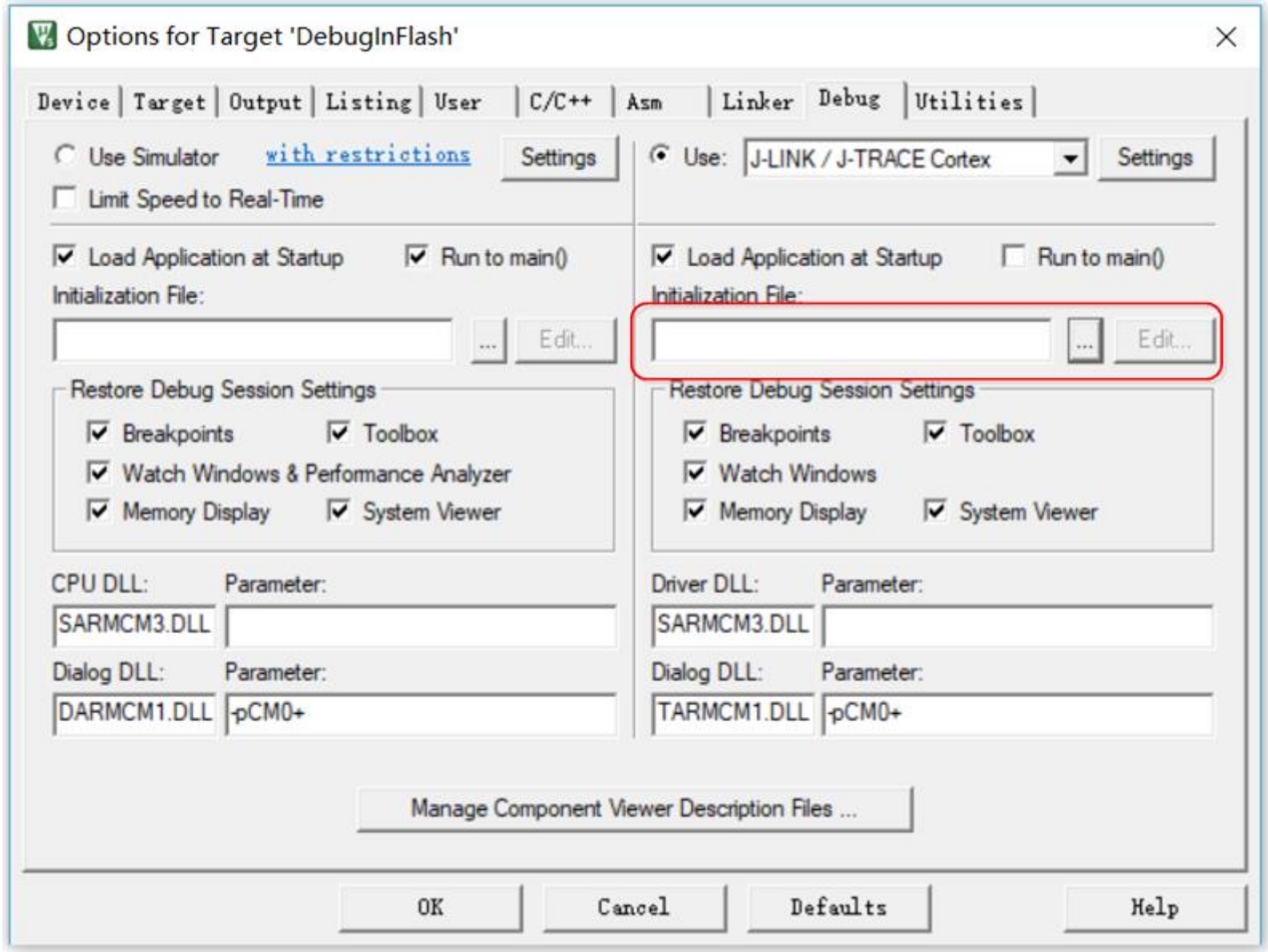

Disassembly
0x1FFF0246 18AA ADDS r2,r5,r2
0x1FFF0248 2908 CMP r1,#0x08
0x1FFF024A DBF9 BLT 0x1FFF0240
0x1FFF024C 2A00 CMP r2,#0x00
0x1FFF024E D002 BEQ 0x1FFF0256
0x1FFF0250 2000 MOVS r0,#0x00
0x1FFF0252 6020 STR r0,[r4,#0x00]
0x1FFF0254 BD70 POP {r4-r6,pc}
0x1FFF0256 F7FFFEF7 BL.W 0x1FFF0048
0x1FFF025A BD70 POP {r4-r6,pc}
0x1FFF025C B510 PUSH {r4,lr}
0x1FFF025E F7FFFFE6 BL.W 0x1FFF022E
0x1FFF0262 F7FFFFCD BL.W 0x1FFF0200
0x1FFF0266 BD10 POP {r4,pc}
0x1FFF0268 B510 PUSH {r4,lr}
0x1FFF026A 4A5B LDR r2,[pc,#364] ; @0x1FFF03D8
main.c
LPC8xx.sct
startup_lpc82x.s
RAM.ini
system_LPC8xx.c
1 ; ***
2 ; *** Scatter-Loading Description File generated by uVision ***
3 ; ***
4
5 LR_IROM1 0x00001000 0x00008000 { ; load region size_region
6 ER_IROM1 0x00001000 0x00008000 { ; load address = execution address
7 *.o (RESET, +First)
8 *(InRoot$$Sections)
9 .ANY (+RO)
10 }
11 RW_IRAM1 0x10000000 0x00002000 { ; RW data
12 .ANY (+RW +ZI)
13 }
14 }
15

```
16
```

我们再之前的文章里面介绍过，你可以去查一下芯片的Memory MAP，这个地址空间其实是BOOT ROM（之前文章有过介绍。不知道BOOT ROM是啥的出门左转）你调试一下就会发现程序在里面死循环无法正常运行。

纳尼？不能跑？？WHY？？？分散加载不好使了？？？？？我们仔细分析一下，哪里出了问题，我们之前的文章说过MCU上电从BOOTROM启动起来之后会默认去0x00000000地址上去找MSP，从0x00000004地址上去找PC，但现在程序的加载地址以及运行地址都变成0x00001000了，而不是0x00000000地址了，我们需要告诉MCU，默认地址变了，不是0x00000000。那怎么告诉MCU呢？

其实在M0+/M3/M4内核里面有一个叫VTOR的寄存器（M0核里面我记得是没有），地址是0xE000ED08，这个寄存器用于设置异常&中断向量表默认地址，我们需要在程序运行起来之前设置VTOR寄存器，告诉MCU，地址变成0x00001000了，那怎么做呢？在这里，如下图：



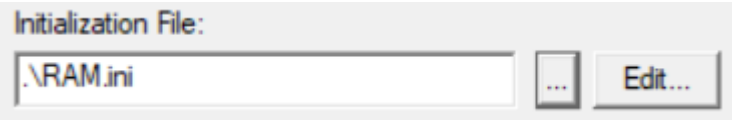
这里有一个仿真器初始化文件，可以在程序下载之前写一些寄存器做一些初始化之类的工作，是一个扩展名为.ini的文件，也有其自己的语法，按照下图编写这个ini文件：

```
FUNC void Setup () {
  SP = _RDWORD(0x00001000);                          // Setup Stack Pointer
  PC = _RDWORD(0x00001004);                          // Setup Program Counter
  _WDWORD(0xE000ED08, 0x00001000);                   // Setup Vector Table Offset Register
}

LOAD .\Out\DebugInRAM\LPC8xx.axf INCREMENTAL         // Download

Setup();                                           // Setup for Running
```

这里简单解释下这个ini文件的意思：FUNC的意思是定义一个函数，Setup()是对应的函数，内容大家都看得懂，直接给MSP以及PC赋值，然后写0xE000ED08（VTOR寄存器）地址，告知MCU地址启动地址改变了，最后一行代码的意思是调用Setup()函数。然后把写好的ini文件放到刚才的位置上，如下图：



我们再次运行一次程序，如下，我们的程序完美运行：

main.c | LPC8xx.sct | startup_lpc82x.s | RAM.ini | system_LPC8xx.c

```
60      LPC_GPIO_PORT->PIN[0] |=  (1 << 10);
61  }
62
63  /*********************************************************************************************************
64  ** Function name:       main
65  ** Descriptions:        GPIO Int 例程:
66  **                      LED连接P0.16。
67  **                      运行程序，LED闪烁。
68  ** input parameters:    无
69  ** output parameters:   无
70  *********************************************************************************************************/
71  int main (void)
72  {
73      SystemInit();                                   /* 初始化目标板，切勿删除       */
74      GPIOInit();                                     /* GPIO初始化                   */
75
76      while (1) {
77          LPC_GPIO_PORT->PIN[0] &= ~LED;              /* 点亮LED                      */
78          myDelay(500);                               /* 延时 500ms                   */
79          LPC_GPIO_PORT->PIN[0] |=  LED;              /* 熄灭LED                      */
80          myDelay(500);                               /* 延时 500ms                   */
81      }
82  }
```

这只是一个最简单的分散加载使用示例，我们会尝试把代码加载到RAM中运行，就像电脑和手机一样；我们也会尝试把单独的一个函数或者一个变量加载到固定地址上；我们还会有把一个Flash空间加载多个image的示例等等。

当然上面提到的.ini文件能做的事情也远不止于此，以后我们还会多次涉及这个文件的用法。