Realview MDK中链接脚本详细解析

使用 Realview MDK 时不可避免的要涉及到链接脚本文件，特别是编译链接那些大的工程文件时更是如此。在链接脚本中可以指定代码的存储布局，可以将代码段、只读数据段、可读写的数据段分别存放，甚至可以精确地指定代码放置的位置，这一点是很关键的，譬如说启动代码就必须放在可知型文件的开始位置。由于链接脚本重要性，开发者必须掌握其编写的方法。

Realview MDK 链接程序使用了两种方式控制程序的链接，即链接控制命令选项和链接脚本文件。当使用链接控制命令选项时，链接器定义了 Image$$RW$$Base、Image$$RW$$Limit、Image$$RO$$Base、Image$$RO$$Limit、Image$$ZI$$Base 和 Image$$ZI$$Limit 等 6 个段地址描述符。这 6 个描述符可以直接在程序中引用。而在使用链接脚本文件后，这 6 个描述符号没有了，取而代之的是链接脚本文件中的段描述符，格式为：Image$$段名$$Base 和 Image$$段名$$Limit。下面将结合 3 个具体的例子说明链接脚本文件的使用。

例 1 一个加载区域，多个连续的执行区域。

```
LR_1 0x040000           ; 定义载入区域 LR_1 的起始地址为 0x040000。
{
    ER_RO +0            ; 执行区域 ER_RO 的起始地址紧接载于区域 LR_1 的起始地址，即为 0x040000.
    {
        * (+RO)         ; 所有的只读代码段都连续地放在这个区域。
    }
    ER_RW +0        ; 可读写数据段 ER_RW 紧接 ER_RO 段的尾地址存放，即 0x040000 + ER_RO 的容量。
    {
        * (+RW)      ; 所有的可读写的程序都连续地放在这个区域。
    }
    ER_ZI +0         ; 清零数据段 ER_ZI 紧接 ER_RW 段的尾地址存放。
    {
        * (+ZI)      ; 所有清零数据都连续地放在这个区域。
    }
}
```

例 2 一个加载区域，多个非连续的执行区域。

```
LR_1 0x010000     ; 定义载入区域 LR_1 的起始地址为 0x010000。
{
    ER_RO +0      ; 执行区域 ER_RO 的起始地址紧接载于区域 LR_1 的起始地址，即为 0x010000.
    {
        * (+RO)   ; 所有的只读代码段都连续地放在这个区域。
    }
    ER_RW 0x040000 ; 定义可读写数据段 ER_RW 的起始地址为 0x040000.
    {
        * (+RW)     ; 所有的可读写的程序都连续地放在这个区域。
    }
    ER_ZI +0; 清零数据段 ER_ZI 紧接 ER_RW 段的尾地址存放，即为 0x040000 + ER_RW 的容量。
    {
         * (+ZI)    ; 所有清零数据都连续地放在这个区域。
    }
}
```

例 3 二个加载区域，多个非连续的执行区域。

```
LR_1 0x010000     ; 载入区域 LR_1 的起始地址为 0x010000。
```

```
{
    ER_RO +0        ; ER_RO 段的起始地址为 0x010000.
    {
        * (+RO)
    }
}
LR_2 0x040000     ; 载入区域 LR_2 的起始地址为 0x040000。
{
    ER_RW +0        ;  ER_RO 段的起始地址为 0x010000.
    {
        * (+RW)     ; 所有可读写的数据段都放在这里。
    }
    ER_ZI +0        ; 清零段 ER_Z 的起始地址为 0x040000 + ER_RW 段的容量。
    {
        * (+ZI)     ; 所有清零段 ZI 的数据都连续的放在这里。
    }
}
```

上面三个例子中，载入区域和执行区域的名字是可以任意命名的，对这些段地址的引用可以使用如 Image$$ LR_1$$Base 、Image$$ LR_1$$Limit、Image$$ ER_RW $$Base 和 Image$$ ER_RW $$Limit 等。