引言

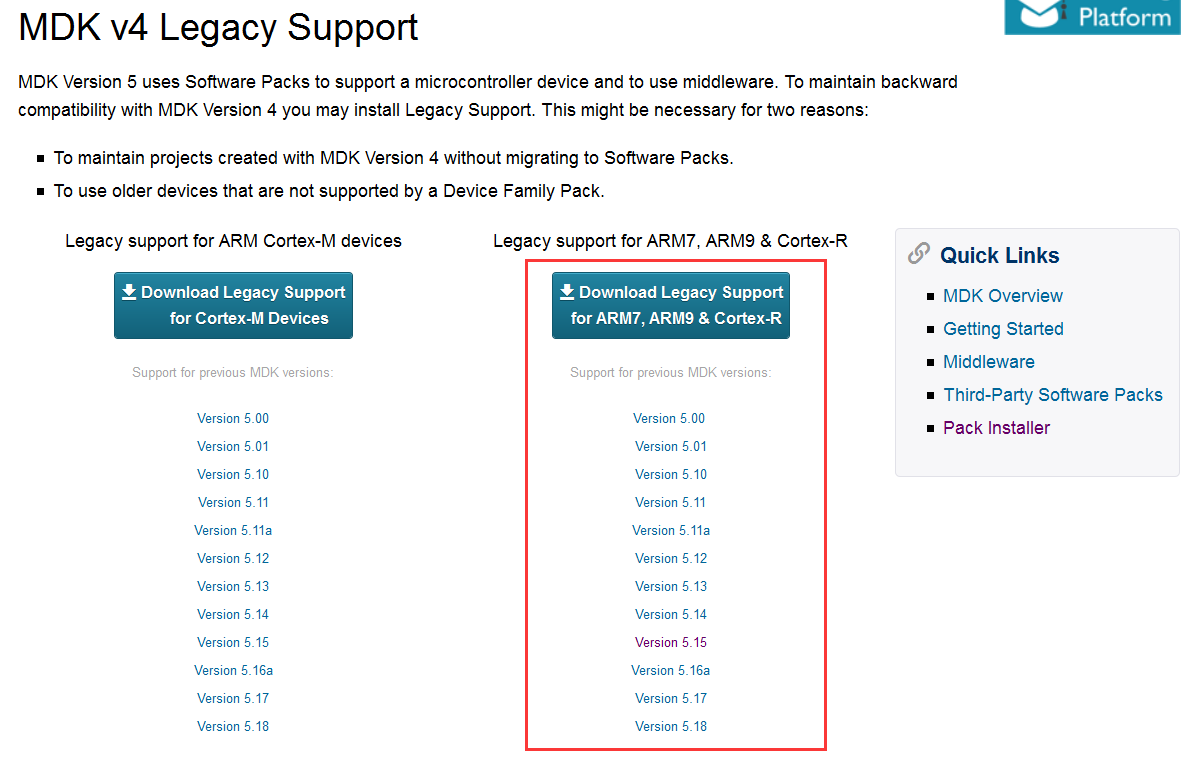
用于arm裸机程序开发的IDE基本有 以下3个：MDK或者称Keil，IAR，还有ADS。其中ADS已经停止更新，不建议使用了。它们的具体情况在这里我就不多说了，百度一下就明白了。

由于之前开发c51的时候都使用了MDK开发环境，而且MDK的界面确实看起来舒服多了，所以我选择了MDK作为我的s3c2440裸机开发的IDE。以下主要介绍一下如何使用MDK配合J-link来调试基于s3c2440的开发板。

MDK、J-link的安装

MDK5软件包请自行百度下载，官方网站上的需要注册下载略显麻烦。安装后需要注册码，否则使用有限制。个人学习研究可尝试破解，安装和破解方法请自行百度。

这里我只提醒需要特别注意的地方：MDK5以后的版本不在直接支持arm7、arm9的开发。因此我们需要下载相应的arm7、arm9的软件支持包。

打开这个网页 [MDK Version 5 - Legacy Support](http://www2.keil.com/mdk5/legacy) 下载相应的软件支持包就可以了。截图如下，注意要选择自己MDK对应版本的支持包。   
[](https://img-blog.csdn.net/20160621004447146)

软件工具下载

首先，我们需要下载的有以下3样：   
MDK软件包   
arm7、arm9的软件支持包   
J-link 驱动   
我的开发环境：windows 7 64位，MDK5.15版本，J-linkv8，S3C2440开发板一块。J-link v8的版本J-linkARM v4.081

MDK的大部分版本可以到以下链接下载(也可到官方网站下载)：

<http://www.mcuzone.com/down/List.asp?SA=1009>

J-link驱动的最新版本可以到以下链接下载：

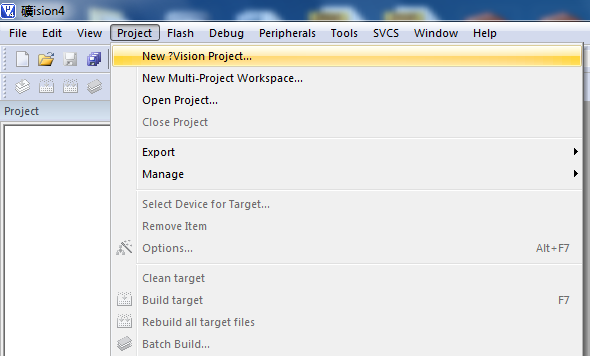
<http://www.segger.com/jlink-software.html>

安装MDK,J-link驱动,安装过程就不多说了,跟着提示走就行了,安装MDK时后要记得注册,安装J-link驱动后插上J-link系统会自动设别安装。

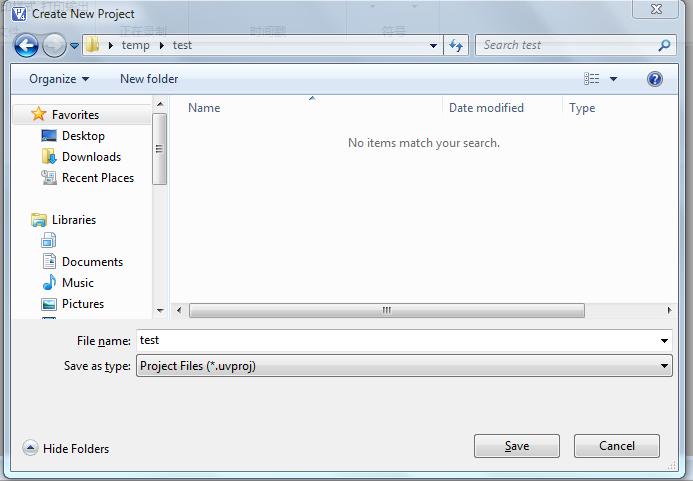
MDK工程的建立、烧写

MDK工程的建立,代码的烧写（基于s3c2440）：

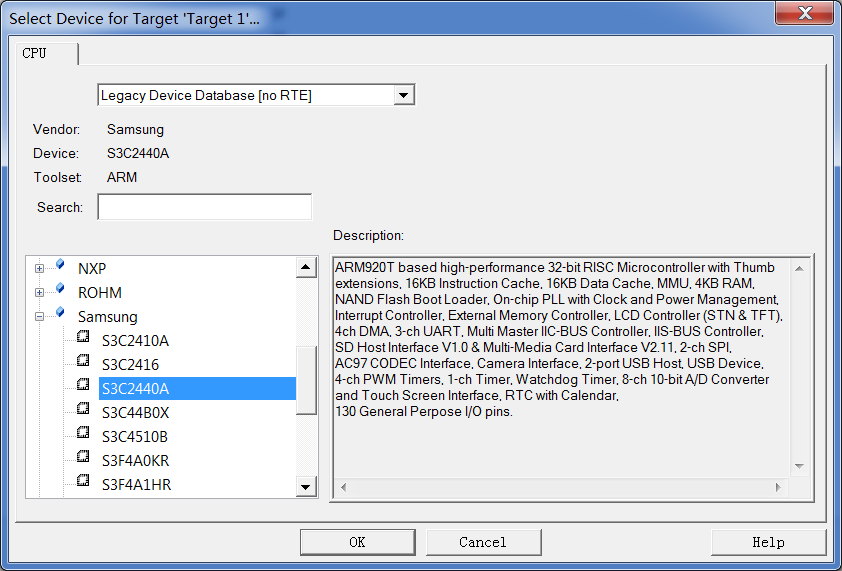
1）打开Keil ，新建一个工程：

[](https://img-blog.csdn.net/20160513231357237)

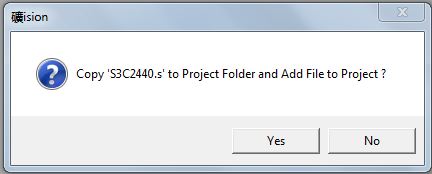
2）新建一个工程目录，用于存放一个工程的所有文件：并且取工程名为s3c2440\_t1并保存,然后创建工程文件,你可以自己命名为test：

[](https://img-blog.csdn.net/20160513231816047)

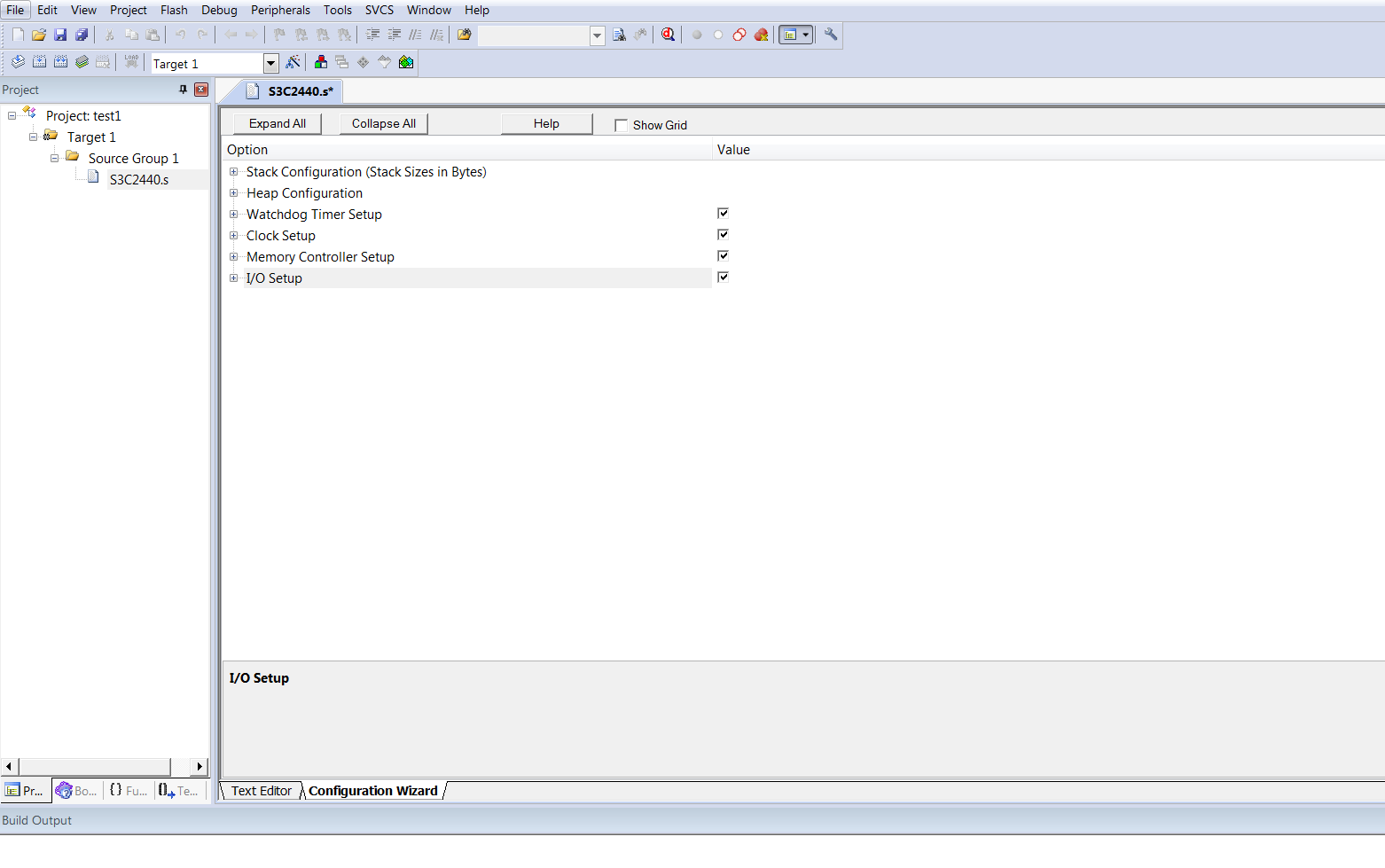
3）接下来选择CUP database：我们选择samsung的S3C2440A，点击OK：

[](https://img-blog.csdn.net/20160513232122428)

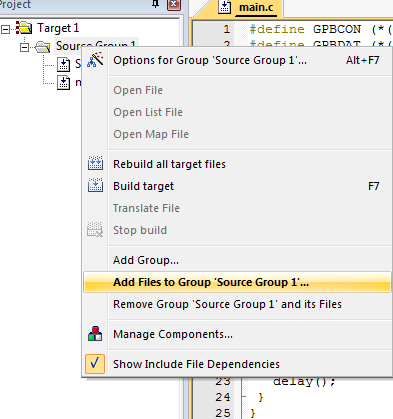
4）之后会出现第二个图问你是否拷贝’S3C2440.s’到工程文件夹并加入到工程里，S3C2440.s是启动代码我们选择YES：

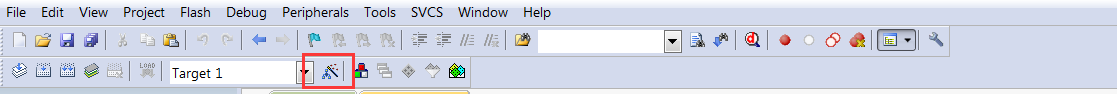
[](https://img-blog.csdn.net/20160513232256647)

5）然后我们需要对S3C2440.s进行一些必须的配置，打开此文件选择编辑界面下面的Configuration wizard进行配置，这里我把全部都打上勾，这样比较保险：

[](https://img-blog.csdn.net/20160513232526430)

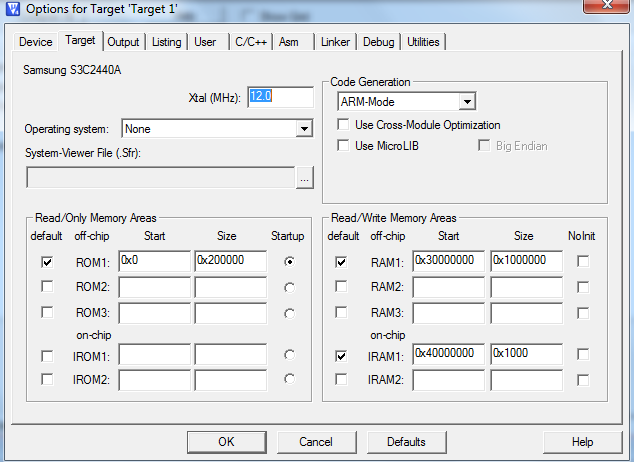
6）之后新建一个c文件并编写代码，我习惯新建一个main目录用来保存这个存放入口函数的文件，然后把这个文件加到工程里:

[](https://img-blog.csdn.net/20160514235557720)

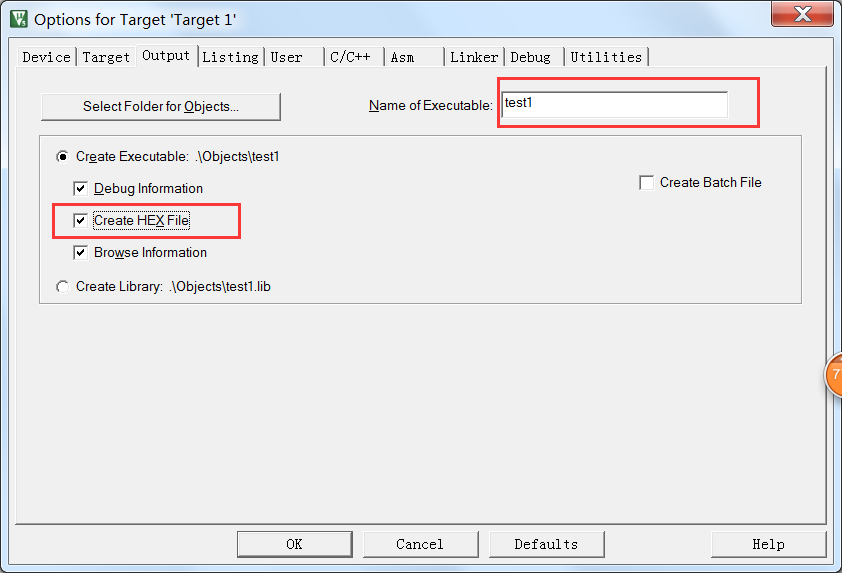
7）把所有c源文件加进工程之后,再选择Target 1的Optionsfor Target ’ Target 1’进行相应的配置   
[](https://img-blog.csdn.net/20160621002151788)

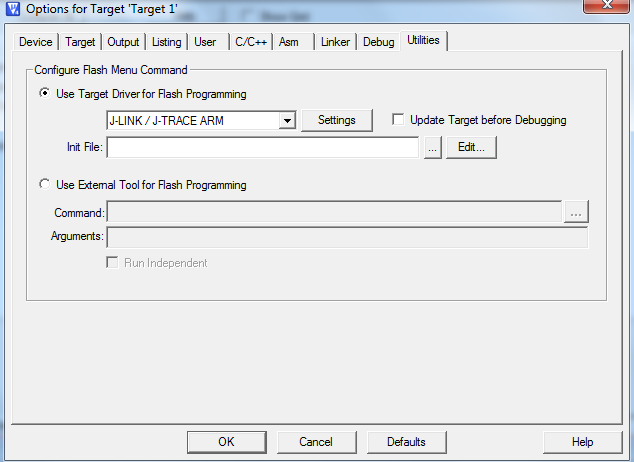
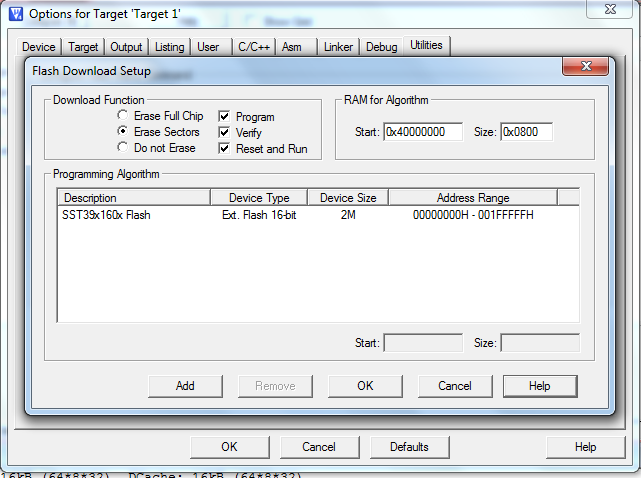
8）因为只是单单要生成可以在norflash运行的HEX可执行文件，需要配置的有以下几个：

①Target选项，修改如下图：

[](https://img-blog.csdn.net/20160514235952944)

②Output选项，修改如下图：

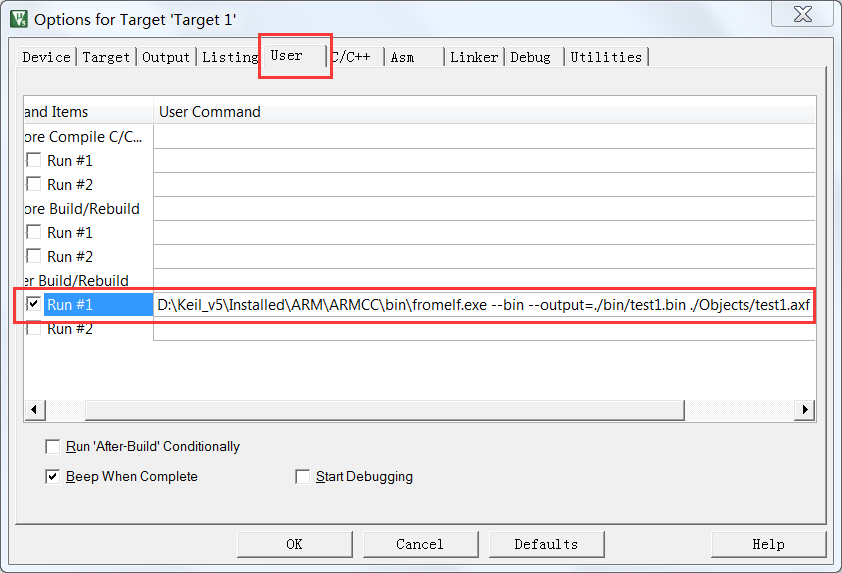
[](https://img-blog.csdn.net/20160621002246259)   
③Utilities选项，修改如下第一图：这里需要特别注意的是，Utilities里面的UpdateTarget before Debugging前面的勾去掉，然后进入Settings，选择你要烧到的flash型号的编程算法，这个如果没有对应的，可以找相似的，如下第二图：

[](https://img-blog.csdn.net/20160515000342684)   
[](https://img-blog.csdn.net/20160515000420450)

9）配置好了之后，都选择OK这样就可以开始rebuild了，正常就可以生成HEX文件(bin格式文件后面讲)，然后连接好J-Link跟开发板上电，把开关拨到要烧进去的flash，点击keil里的download按钮，没有异常的话程序就烧进去了，然后复位一下开发板就可以看到运行效果了。

[这里写图片描述](https://img-blog.csdn.net/20160515000550936)   
10）在Realview MDK的集成开发环境中，默认情况下可以生成.axf格式的调试文件和.hex格式的可执行文件。虽然这两个格式的文件非常有利于ULINK2仿真器的下载和调试，但是有一些特殊情况：

* ADS的用户更习惯于使用\*.bin格式的文件；
* 有些嵌入式软件开发者已经拥有了\*.bin格式文件的调试或烧写工具；
* 有些开发者采用uboot直接将\*.bin格式文件加载到SRAM中执行。

为了充分地利用现有的工具，同时发挥Realview MDK集成开发环境的优势，可以将.axf格式文件或.hex格式文件转换成\*.bin格式的文件。   
首先，打开User选项，编辑内容如下图：   
[](https://img-blog.csdn.net/20160621003107425)   
在选项“run#1”中的命令解释：   
1.–bin 生成Plain Binary格式的文件   
2.–output file 输出文件(默认的输出为文本格式,需加–bin命令)   
这句意思就是使用fromelf.exe工具把Objests文件夹下的test1.axf转成test1.bin文件,转换后的文件放在bin目录下。

进行以上操作后，重新编译工程，在./bin/文件夹下生成了test1.bin文件，开发者就可以将bin文件烧写到开发板了。

## 参考资料

1. MDK5 IDE的帮助文档。
2. zhzht19861011的博客： <http://blog.csdn.net/zhzht19861011/article/details/5927924>