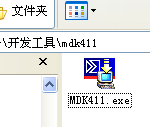
**【ARM】ARM接口技术**

**[TOC]**

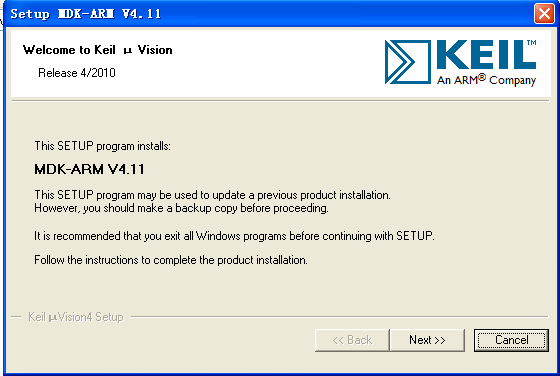
# #0开发工具的使用

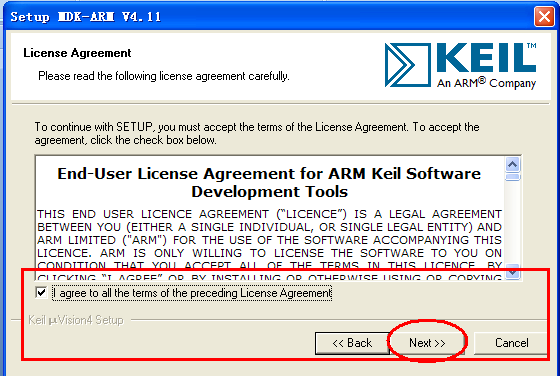
## ##0.1安装keil4

在arm资料1压缩文件中找到mdk411.rar，解压，双击MDK411.exe进行安装

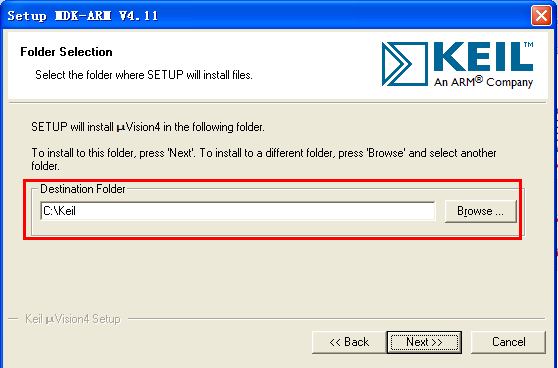


## ##0.2运行MDK411.EXE安装

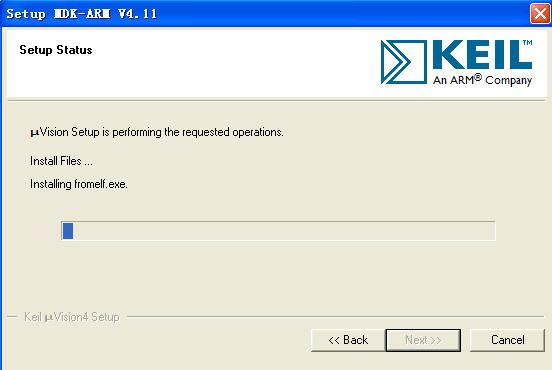




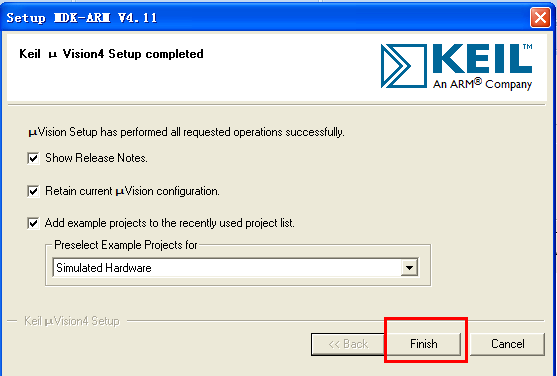
## ##0.3选择好要安装的目的位置



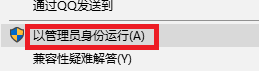
点NEXT，安装界面如下



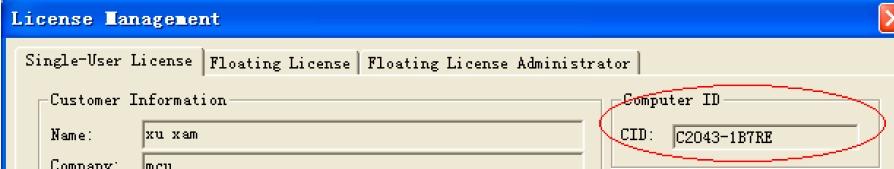
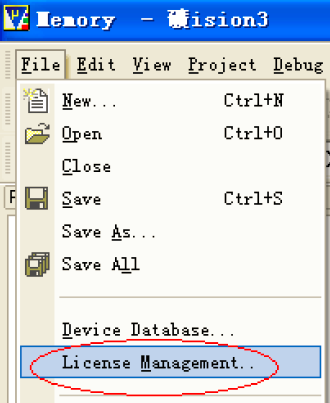
## ##0.4安装完成并运行



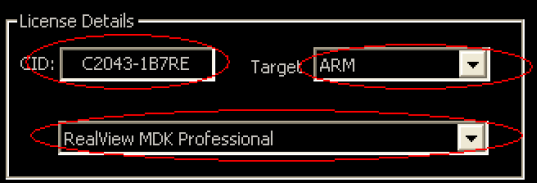
安装完成后，鼠标右键单击该程序，以管理员身份运行该程序：



把CID拷贝下来：



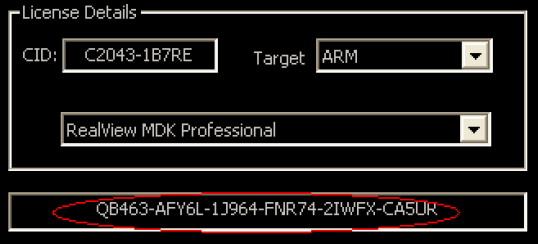
复制Computer ID：C2043-1B7RE ，然后运行

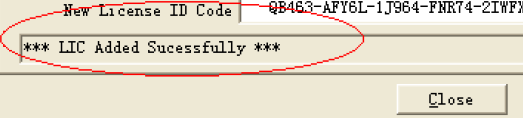


把CID填入上图，然后：



复制注册码：



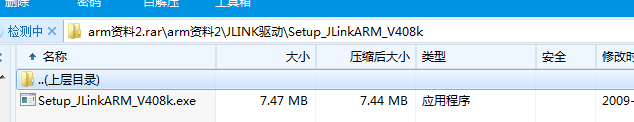
复制注册码填入：点ADD LIC后，提示：

提示：\*\*\* LIC Added Sucessfully \*\*\*

说明注册成功了。

## ##0.5安装JLink驱动

在资料2中可以找到Setup\_JLinkARM\_V408k.exe文件，双击安装JLink驱动

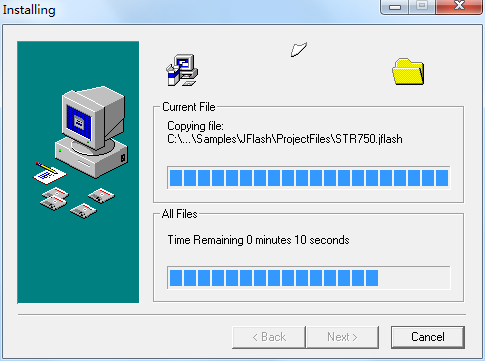


将会弹出如下窗口

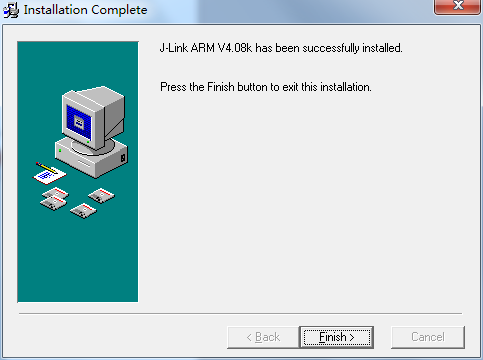




然后一直点击next

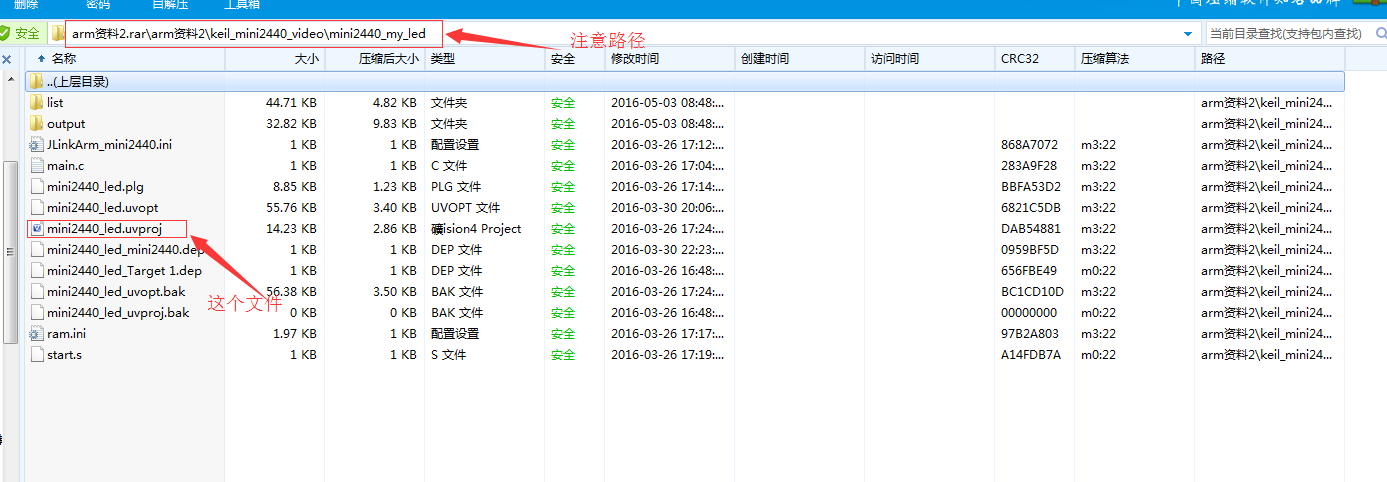


最后提示出如下界面

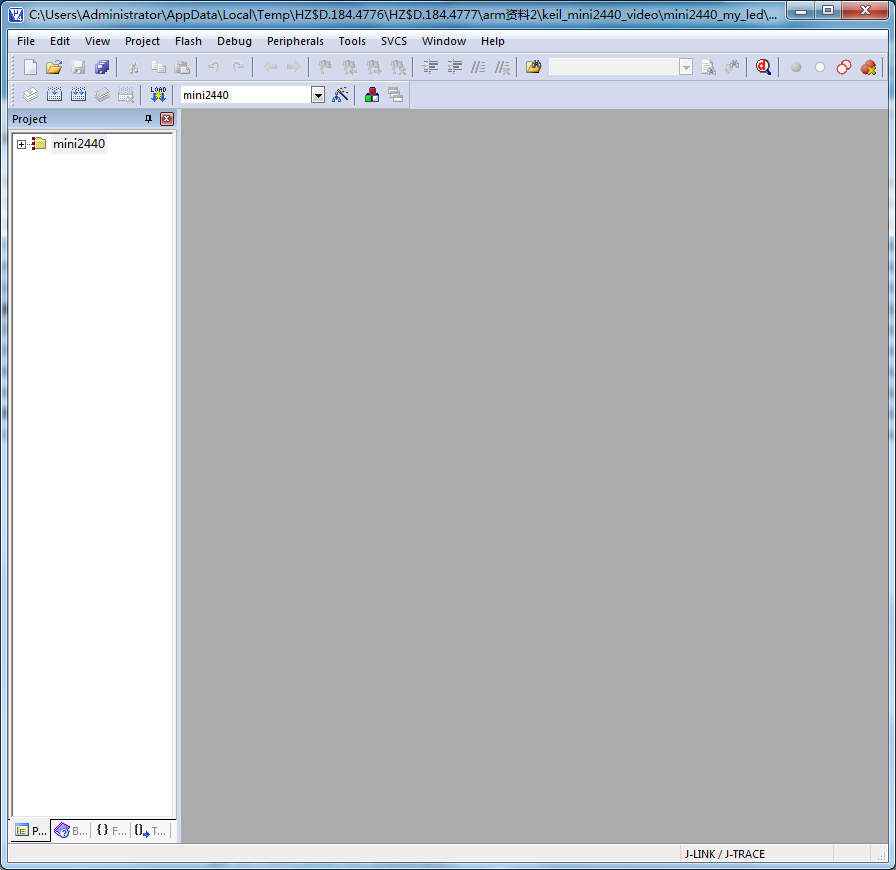


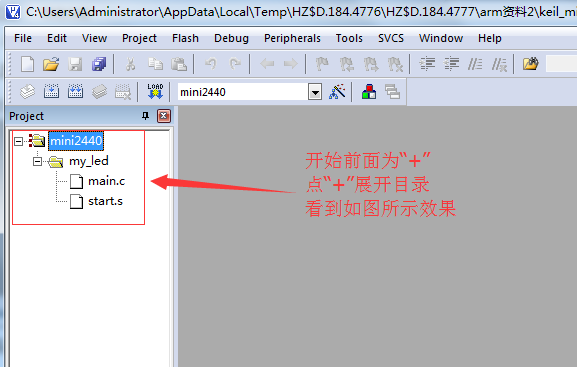
## ##0.6运行测试例程，验证开发板是否OK

运行老师给的mini2440\_led工程文件，即双击mini2440\_led.uvproj文件

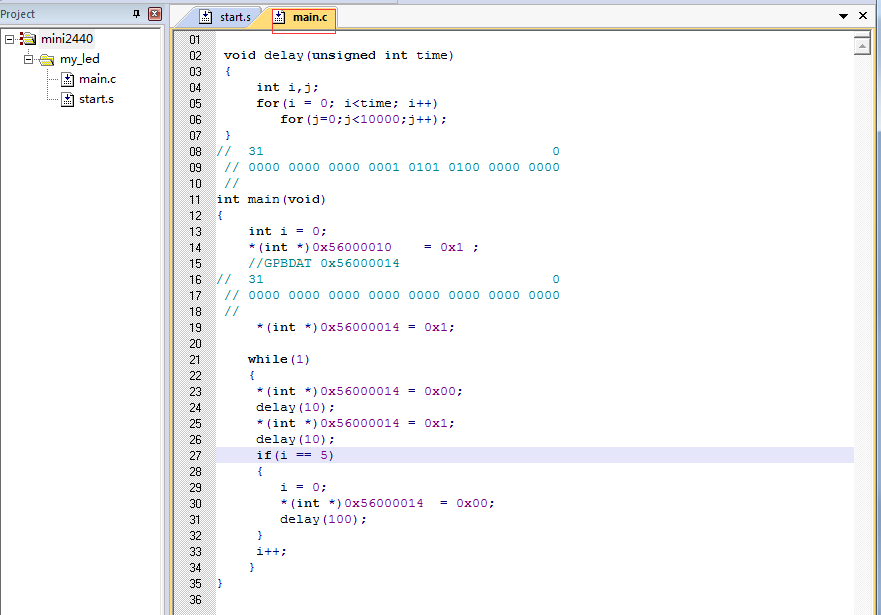


可以看到以下界面：



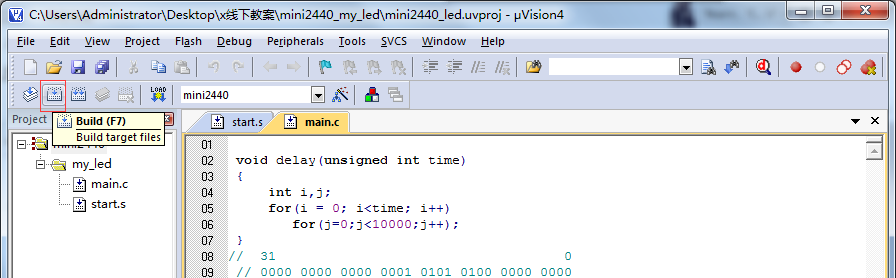


双击点开main.c，看看源码

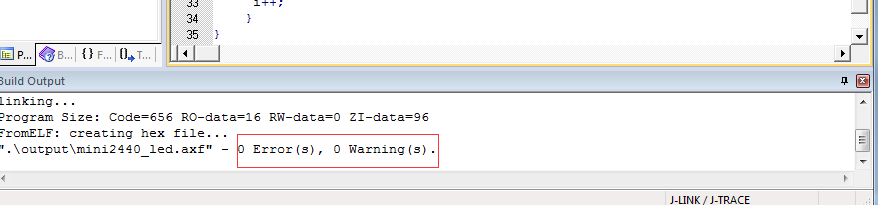


看不懂，没关系，直接编译，看结果

点【】

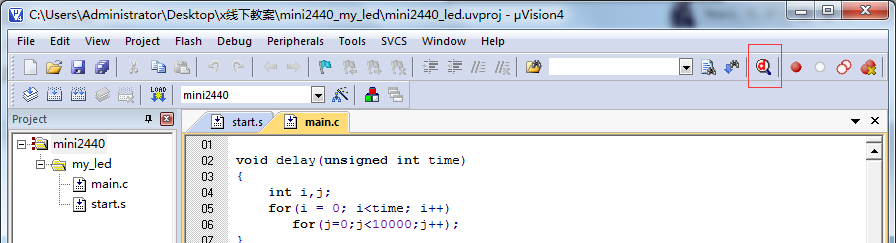


在最下方提示出如下结果

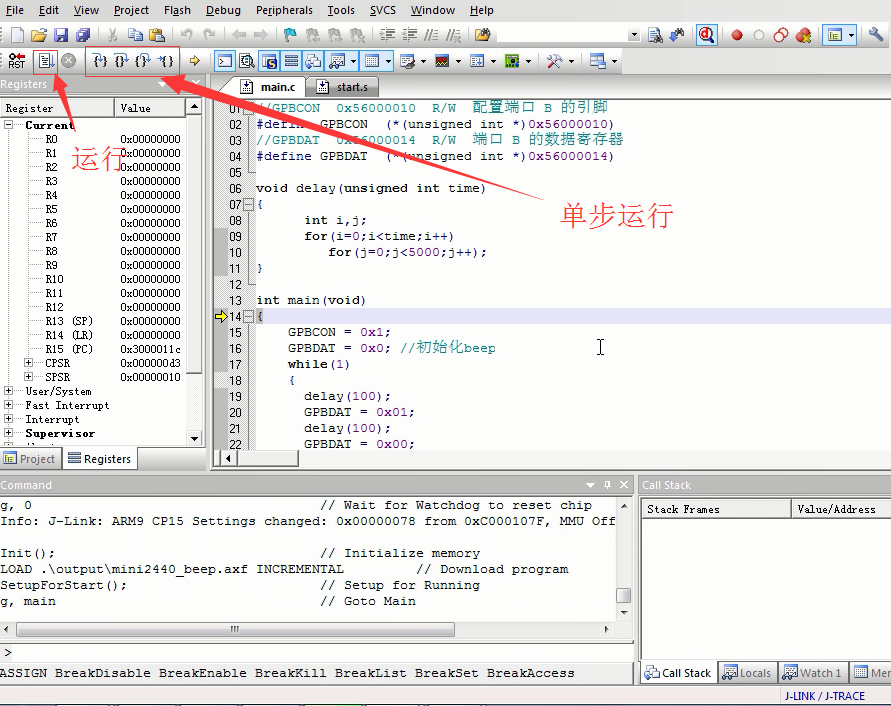


紧接着点【】

进入调试界面

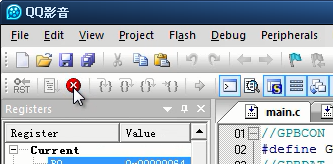


将会调出调试界面



点击运行程序，听开发板蜂鸣器的运行状态！

（蜂鸣器一会响，一会停）

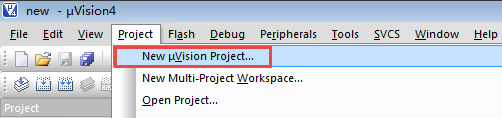


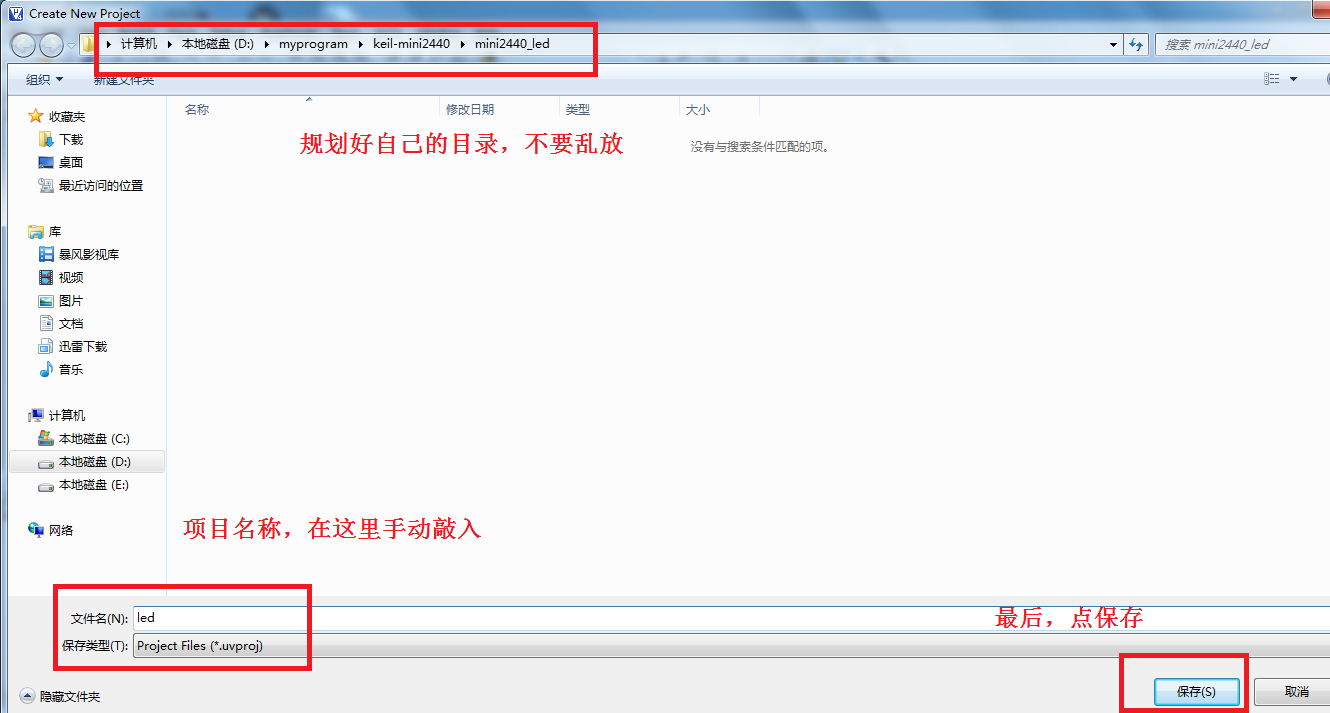
点击【】停止运行

再次点击点【】退出调试状态。

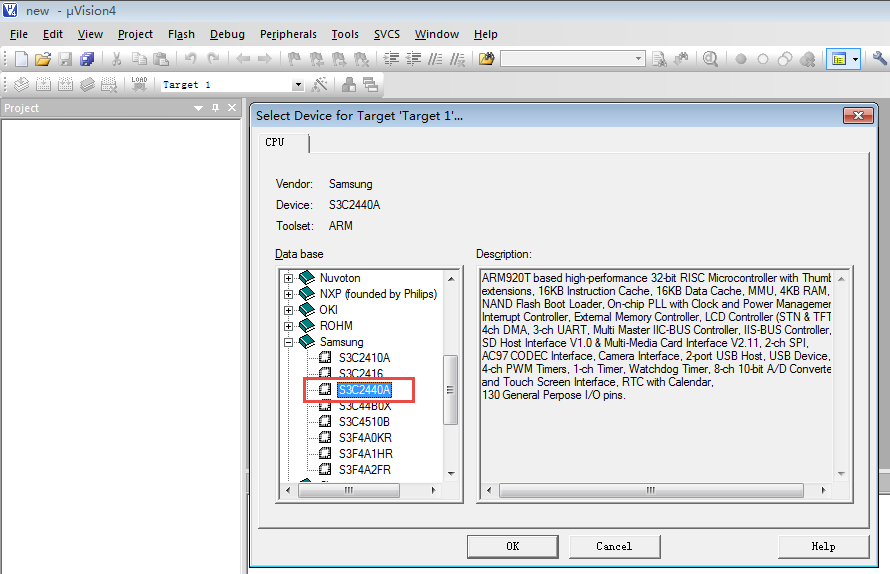
# #1自己做一个例子

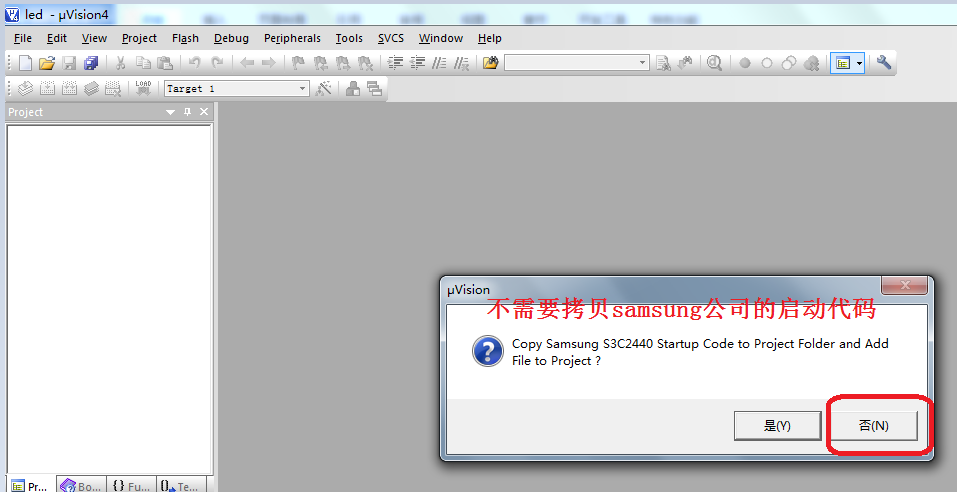
## ##1.1新建一个项目



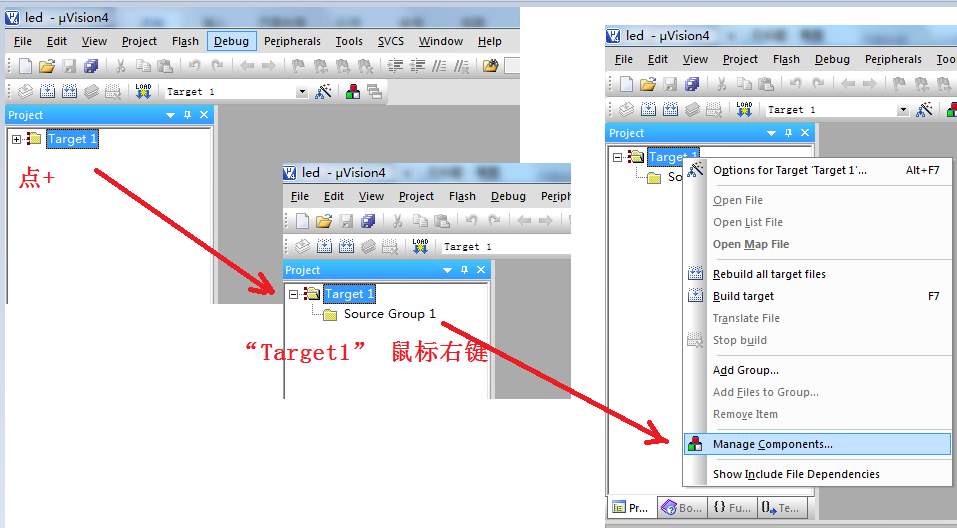


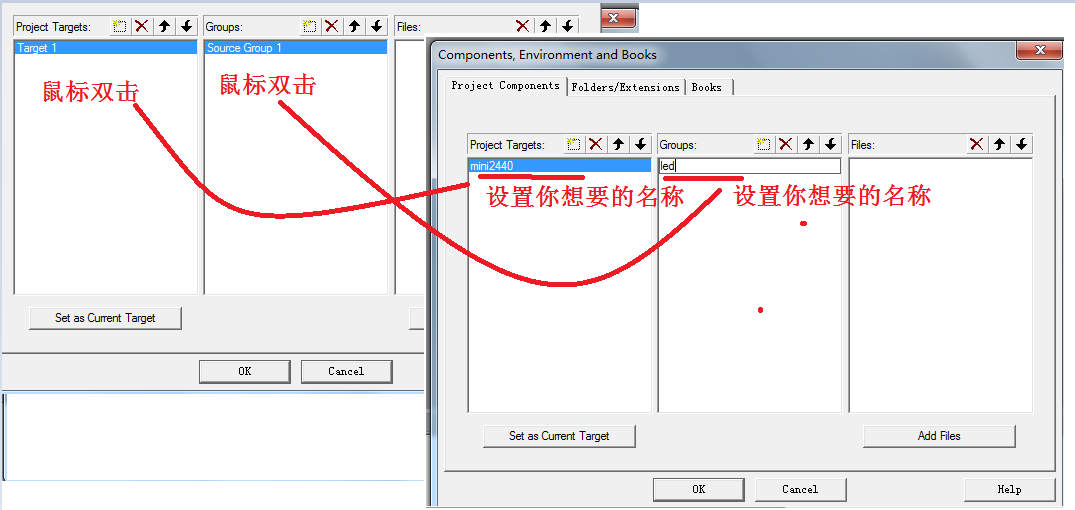
## ##1.2选择与开发板对应的CPU型号



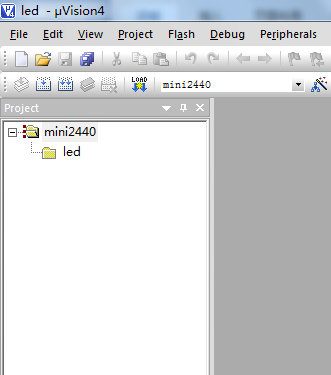


## ##1.3设置一个好的目标和组

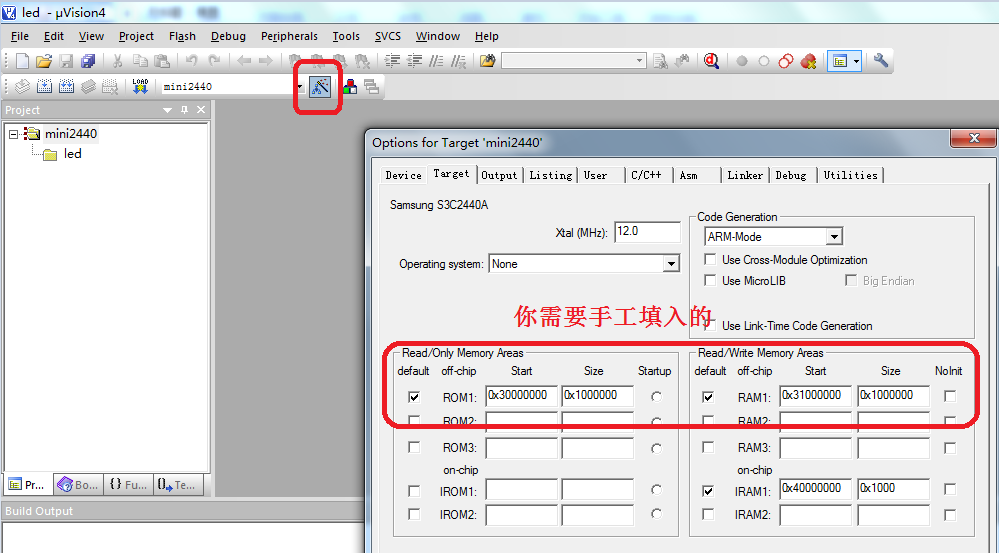


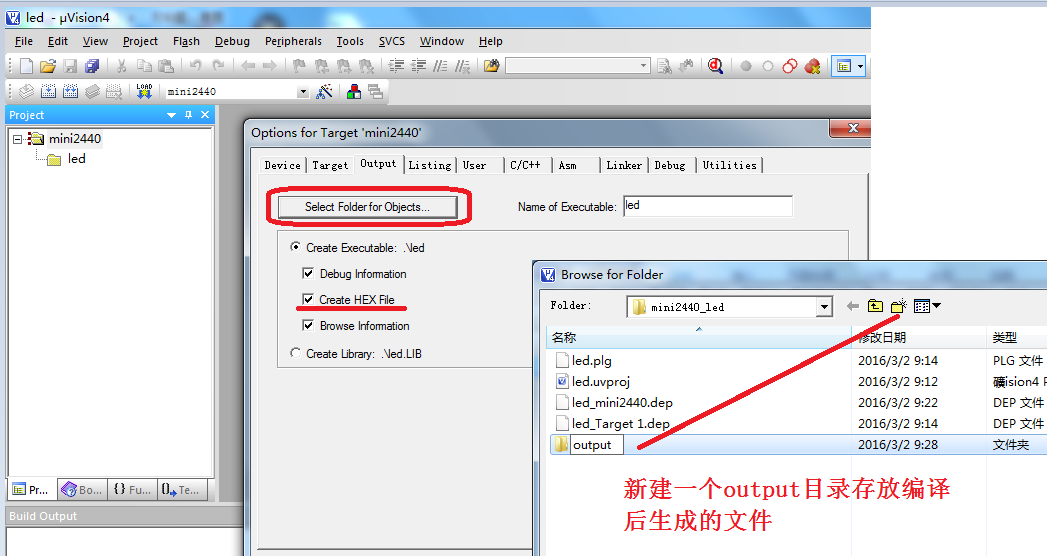


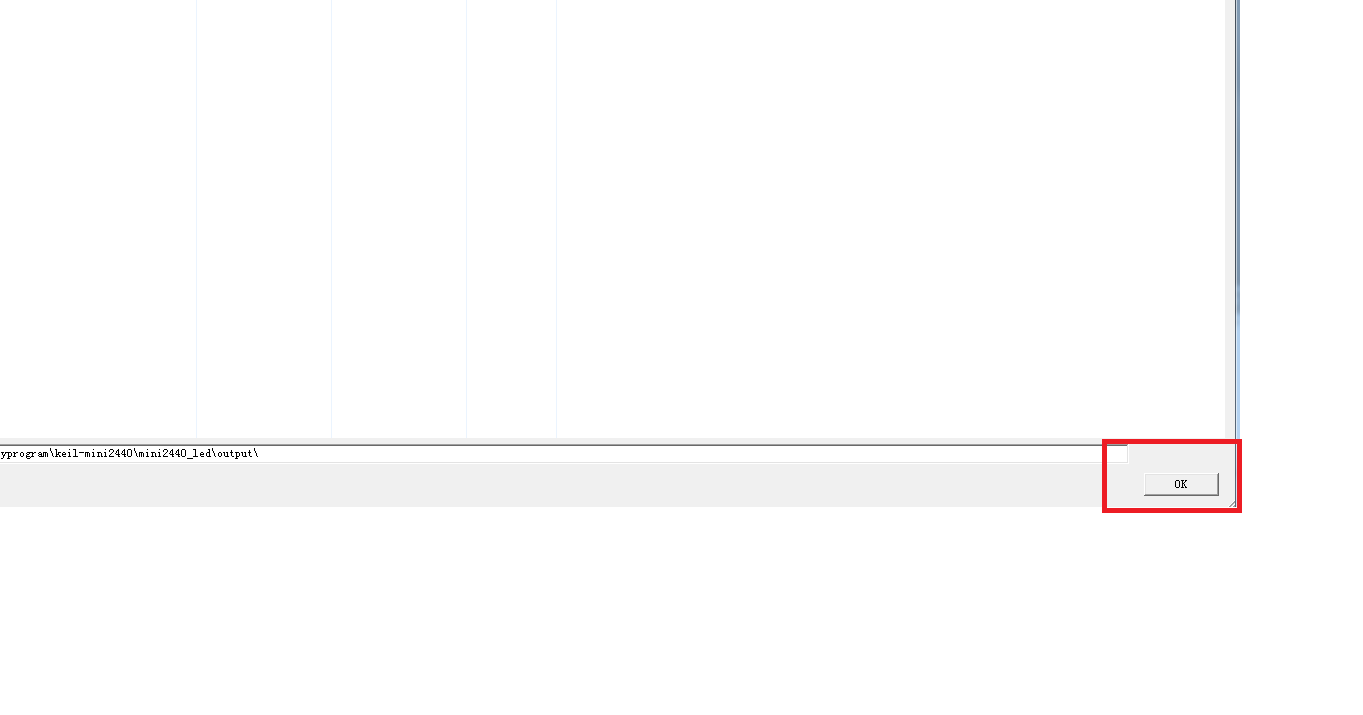
结果如下：

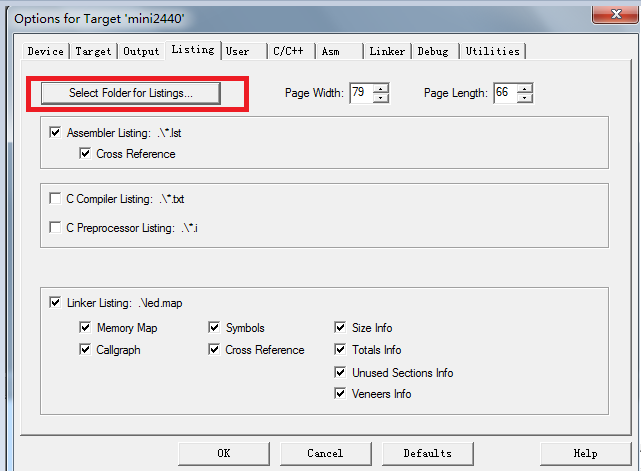


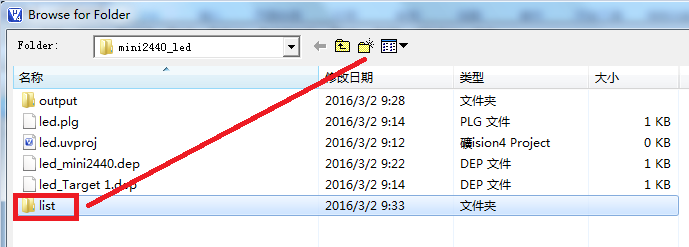
## ##1.4配置目标







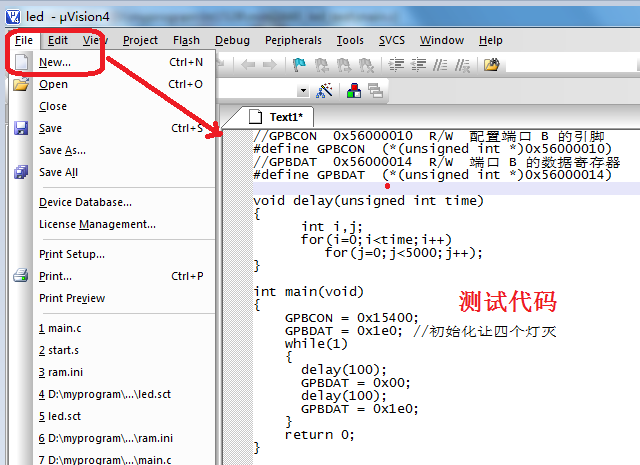


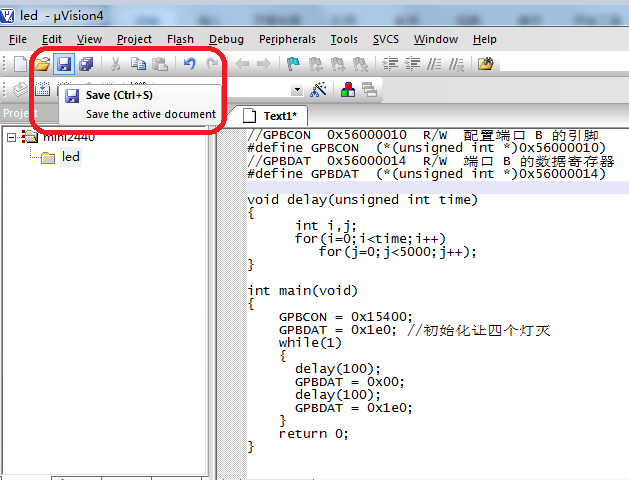


点OK！

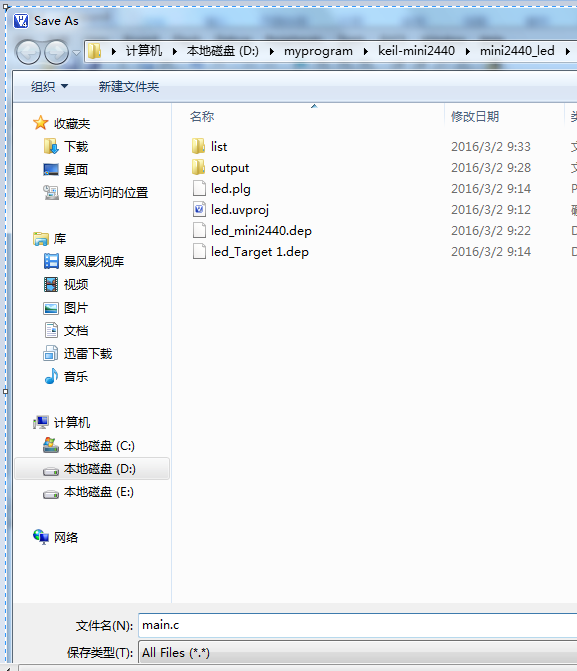
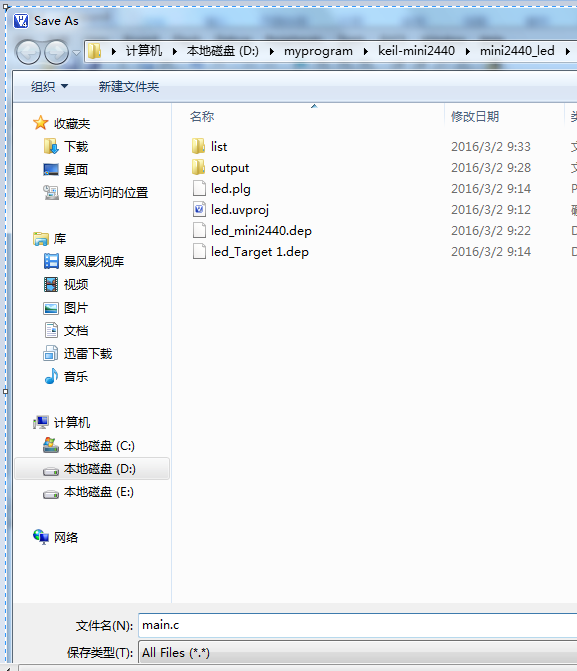
## ##1.5创建一个测试文件

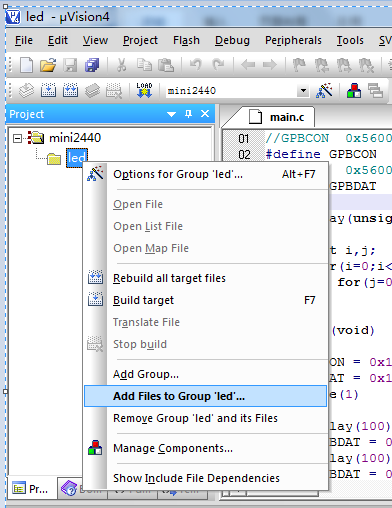
### ###1.5.1创建main.c文件



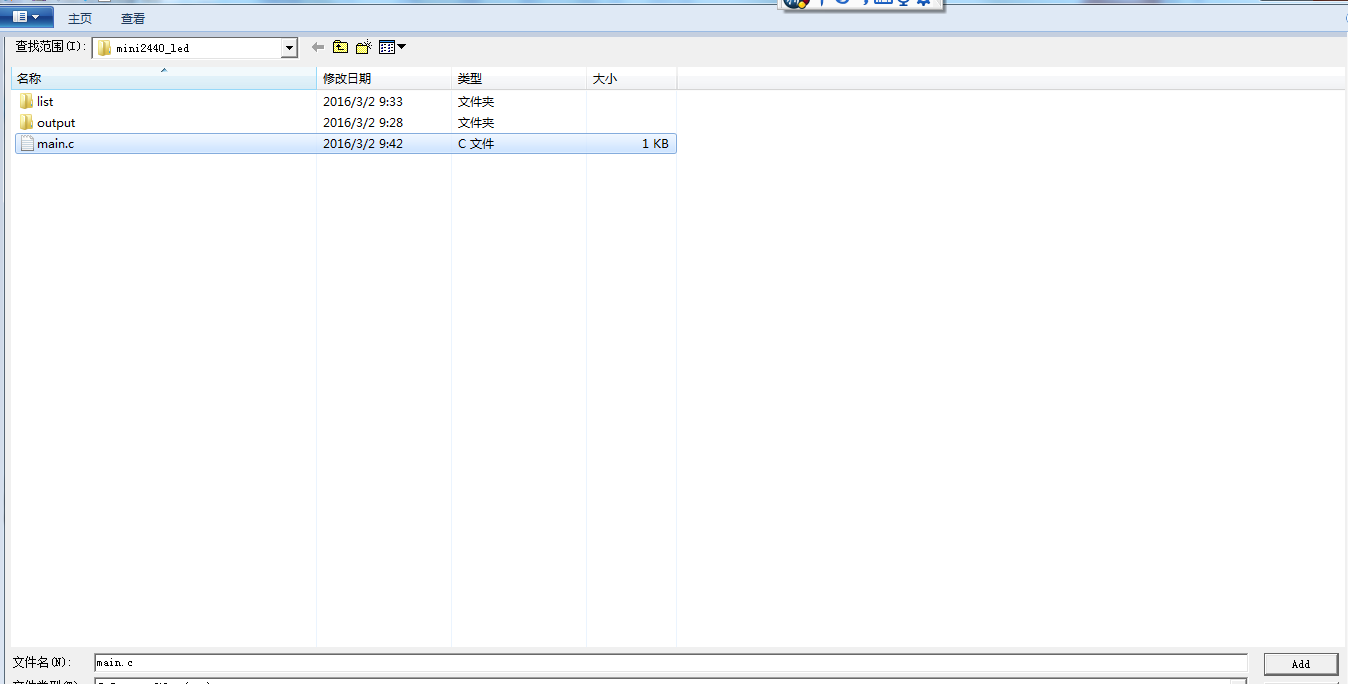


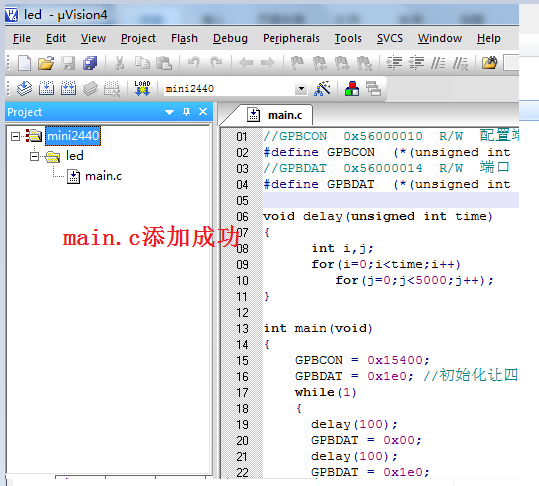
保存为main.c





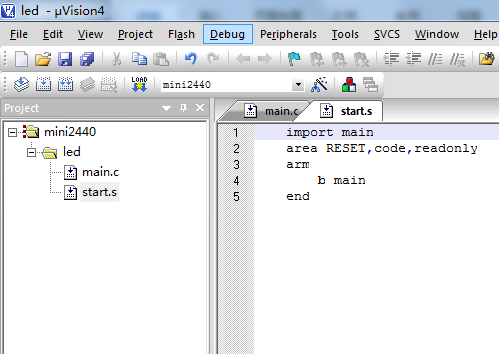
把main.c添加到工程

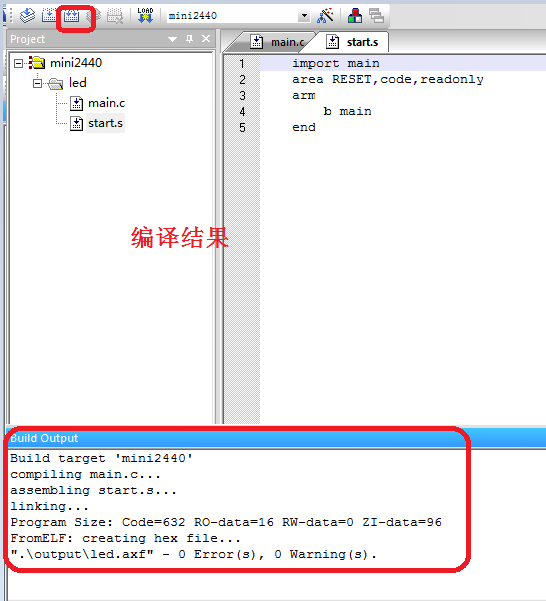
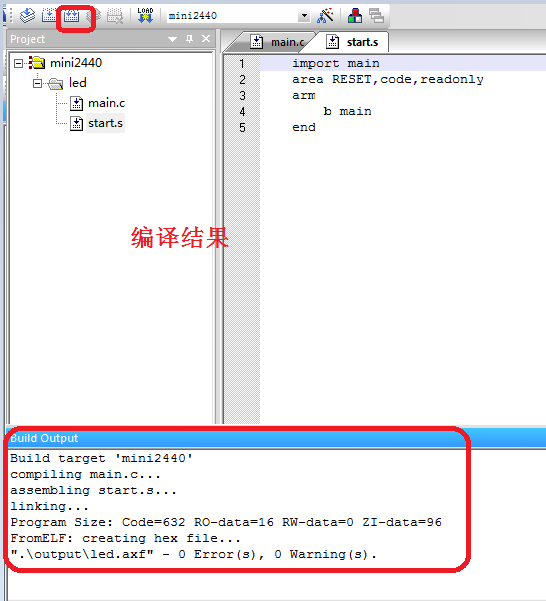




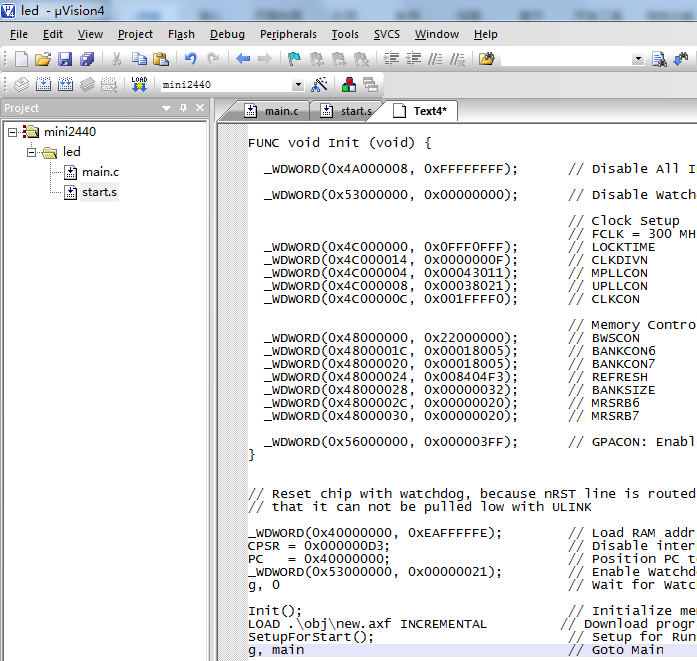
### ###1.5.2建一个汇编文件

同样的步骤建一个汇编文件，保存，添加！





### ###1.5.3建立一个ini文件



内容如下：

说明：倒数第三行文件要根据实际情况进行设置

```

FUNC void SetupForStart (void) {

// <o> Program Entry Point

PC = 0x30000000;

}

FUNC void Init (void) {

\_WDWORD(0x4A000008, 0xFFFFFFFF); // Disable All Interrupts

\_WDWORD(0x53000000, 0x00000000); // Disable Watchdog Timer

// Clock Setup

// FCLK = 300 MHz, HCLK = 100 MHz, PCLK = 50 MHz

\_WDWORD(0x4C000000, 0x0FFF0FFF); // LOCKTIME

\_WDWORD(0x4C000014, 0x0000000F); // CLKDIVN

\_WDWORD(0x4C000004, 0x00043011); // MPLLCON

\_WDWORD(0x4C000008, 0x00038021); // UPLLCON

\_WDWORD(0x4C00000C, 0x001FFFF0); // CLKCON

// Memory Controller Setup for SDRAM

\_WDWORD(0x48000000, 0x22000000); // BWSCON

\_WDWORD(0x4800001C, 0x00018005); // BANKCON6

\_WDWORD(0x48000020, 0x00018005); // BANKCON7

\_WDWORD(0x48000024, 0x008404F3); // REFRESH

\_WDWORD(0x48000028, 0x00000032); // BANKSIZE

\_WDWORD(0x4800002C, 0x00000020); // MRSRB6

\_WDWORD(0x48000030, 0x00000020); // MRSRB7

\_WDWORD(0x56000000, 0x000003FF); // GPACON: Enable Address lines for SDRAM

}

// Reset chip with watchdog, because nRST line is routed on hardware in a way

// that it can not be pulled low with ULINK

\_WDWORD(0x40000000, 0xEAFFFFFE); // Load RAM addr 0 with branch to itself

CPSR = 0x000000D3; // Disable interrupts

PC = 0x40000000; // Position PC to start of RAM

\_WDWORD(0x53000000, 0x00000021); // Enable Watchdog

g, 0 // Wait for Watchdog to reset chip

Init(); // Initialize memory

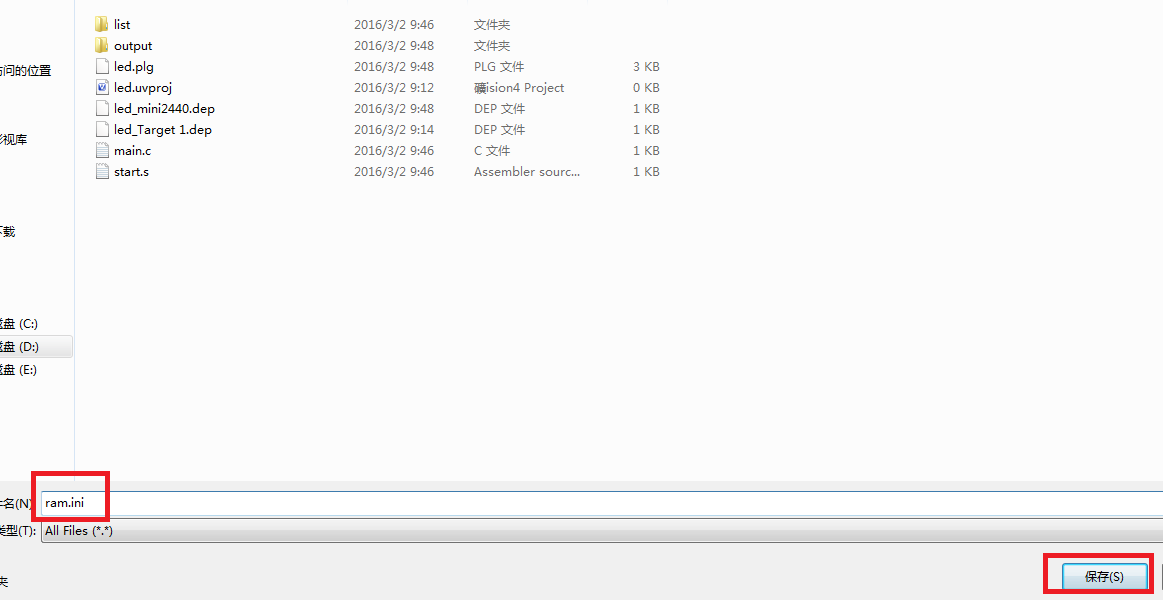
LOAD .\obj\new.axf INCREMENTAL // Download program

SetupForStart(); // Setup for Running

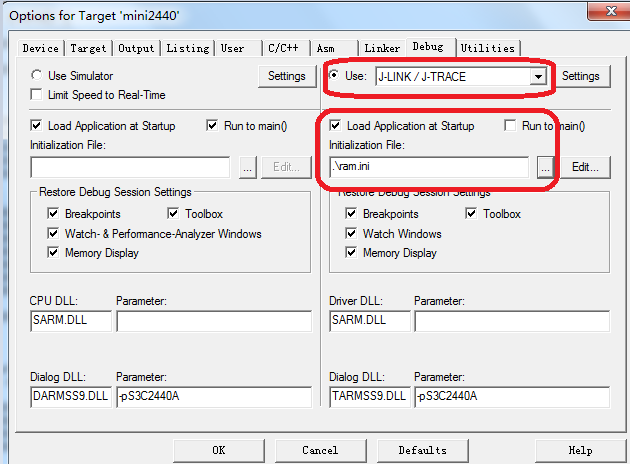
g, main // Goto Main

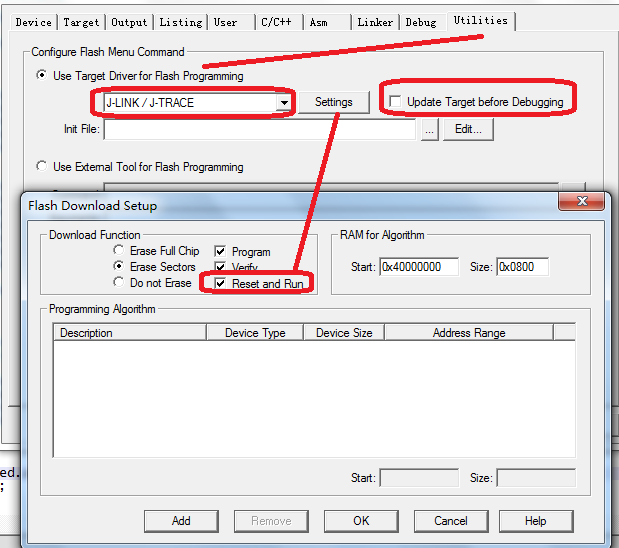
```

保存为ram.ini

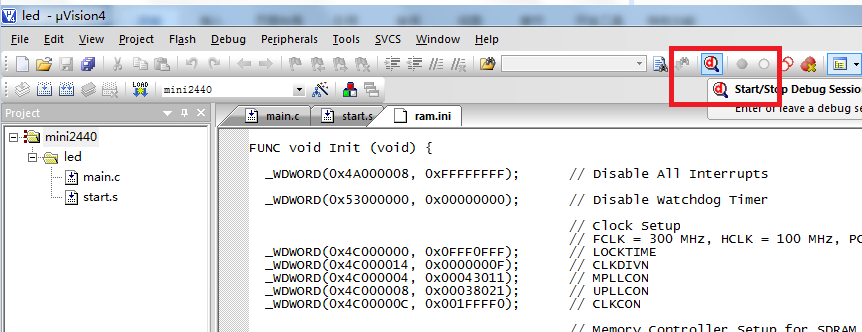


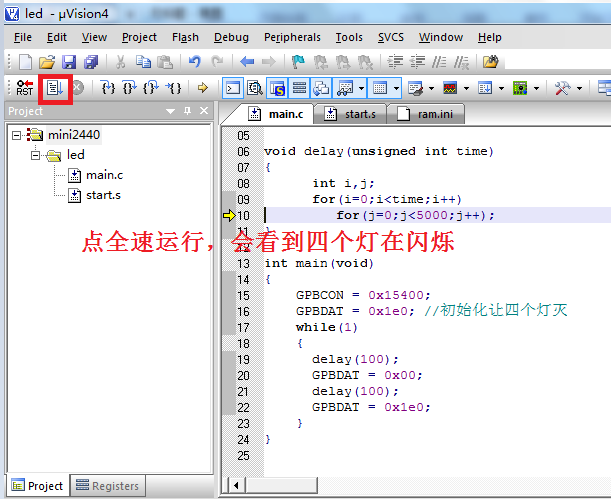
## ##1.6设置参数

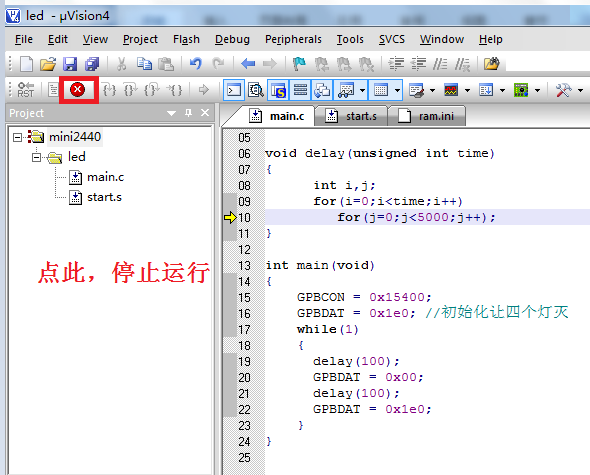




## ##1.7上板子调试





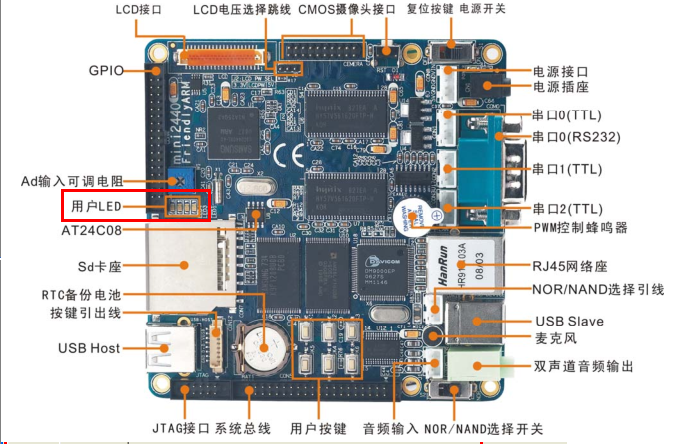


# #2ARM接口原理

## ##2.1查看原理图

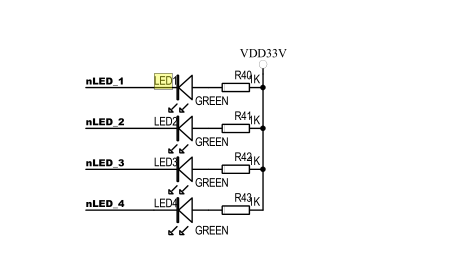
思路：

要控制一个设备，首先想到的是搞清楚这个硬件设备在什么位置。 怎么办？



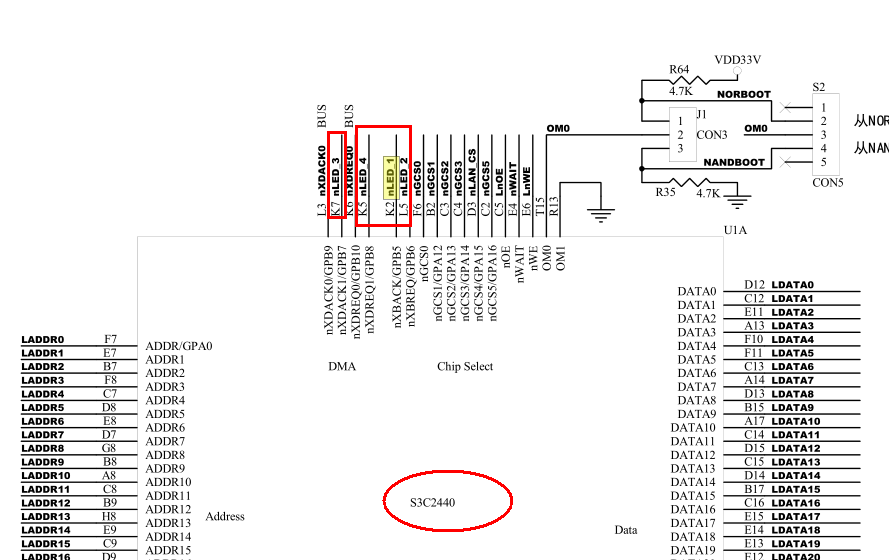
参考《mini2440-um-20090223.pdf》第13页

找到丝印，再找原理图



参考原理图《mini2440原理图.pdf》第3页。

通过网络表nLED\_1，在原理图中找到控制这个灯的cpu引脚。



找到cpu引脚与LED灯的对应关系如下

GPB5----------------- nLED \_1

GPG6----------------- nLED \_2

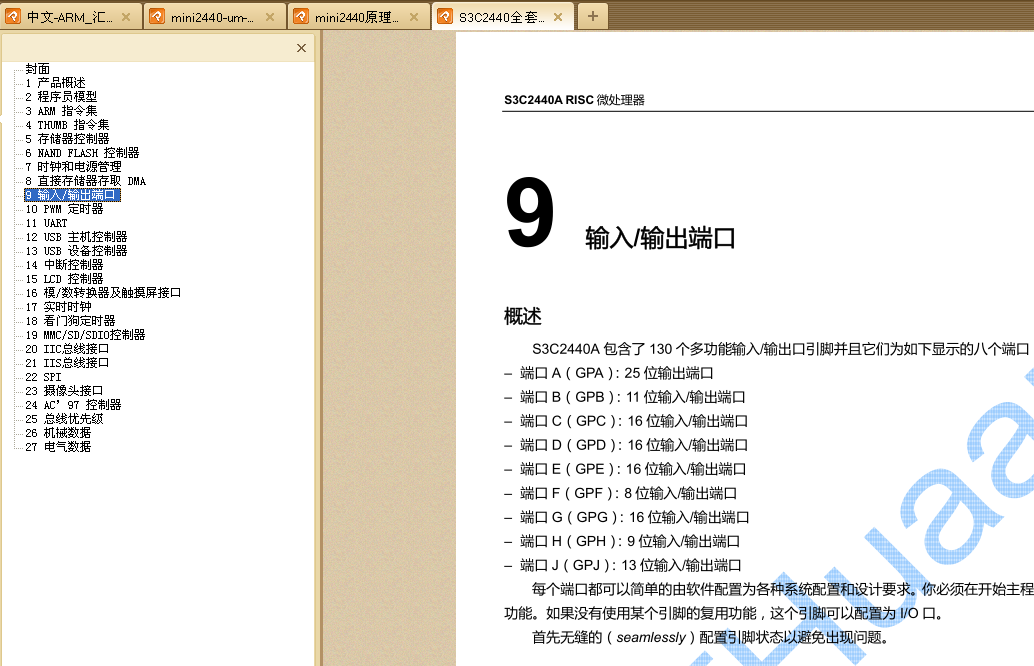
GPB7----------------- nLED \_3

GPB8----------------- nLED \_4

接下来怎么办？

接下来就要看cpu的说明书了，就像我们买了一个微波炉，使用前就要看一下它的说明书，我们CPU的说明书就是s3c2440的datasheet。即《S3C2440全套中文手册.PDF》

## ##2.2GPIO控制原理





### ##2.2.1GPBCON

GPBCON xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

GPB10 9 8 7 6 5 4 3 2 1 0

两个位控制一个引脚，所以GPB0~GPB10共11各引脚，需要0~21共22各位来设置

00----该引脚配置为输入

01----该引脚配置为输出

10------该引脚配置为其他特殊功能

11------一般引脚表示保留，未使用

要让我们四个LED亮灭，GPBCON寄存器该如何设置呢？

GPBCON===？？？？？

GPBDAT是数据寄存器，用来控制（或者反应）引脚的高低电平，该寄存器是一个位对应一个引脚，如果某个位为0那么对应引脚为低电压，如果为1则引脚为高电平

### ##2.2.2GPBDAT

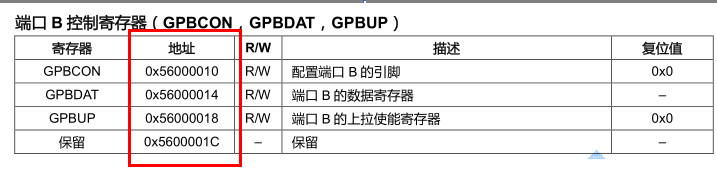
GPBDAT xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

GPB 10 9 8 7 6 5 4 3 2 1 0

原理搞清楚了，程序怎么写？

## ##2.3程序怎么写

我们要把某个引脚设置为输出，也就是我们的GPBCON寄存器要配置为0x15400，那么这个值怎么给我们的cpu呢？



这里的地址，就是cpu提供给我们控制它的接口。

知道了这个接口，代码怎么操作呢？

这里涉及到怎么访问可读性的绝对地址问题？

```

void delay(int time)

{

int i,j;

for(i=0; i<time; i++)

for(j=0; j<1000; j++);

}

int main(void)

{

//GPBCON 0x56000010 R/W 配置端口B 的引脚

//31 0

//0000 0000 0000 0001 0101 0100 0000 0001

// 0 0 0 1 5 4 0 1 GPB0

\*(int \*)0x56000010 = 0x15401; //设置GPB5/6/7/8 to output and 1 beep

//GPBDAT 0x56000014 R/W 端口B 的数据寄存器

//31 0

//0000 0000 0000 0000 0000 0001 1110 0001

// 0 0 0 0 0 1 e 1 GPB0

\*(int \*)0x56000014 = 0x01; //让GPB第5/6/7/8 设置为低电平，第1位为高：为响亮

while(1)

{

\*(int \*)0x56000014 = 0x00000001; //响亮

delay(1000);

\*(int \*)0x56000014 = 0x000001e0;

delay(1000);

}

}

```

学生动手实现延时函数，宏定义，led配置，led亮，led灭函数。

## ##2.4练习，自己实现beep

# #附录

工具下载链接：

https://github.com/1040003585/Mini2440/tree/master/Tools

\*\*Wu\_Being 博客声明\*\*：本人博客欢迎转载，请标明博客原文和原链接！谢谢！

《【ARM】ARM接口技术》

http://blog.csdn.net/u014134180/article/details/72877048

![Wu\_Being 吴兵博客接受赞助费二维码](http://img.blog.csdn.net/20170305211231929?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvdTAxNDEzNDE4MA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast)

如果你看完这篇博文，觉得对你有帮助，并且愿意付赞助费，那么我会更有动力写下去。