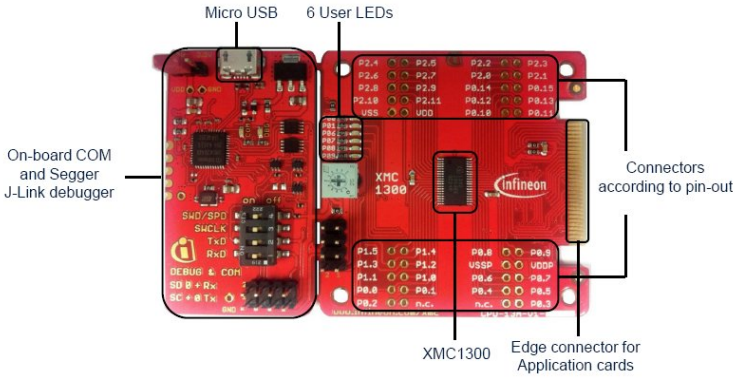
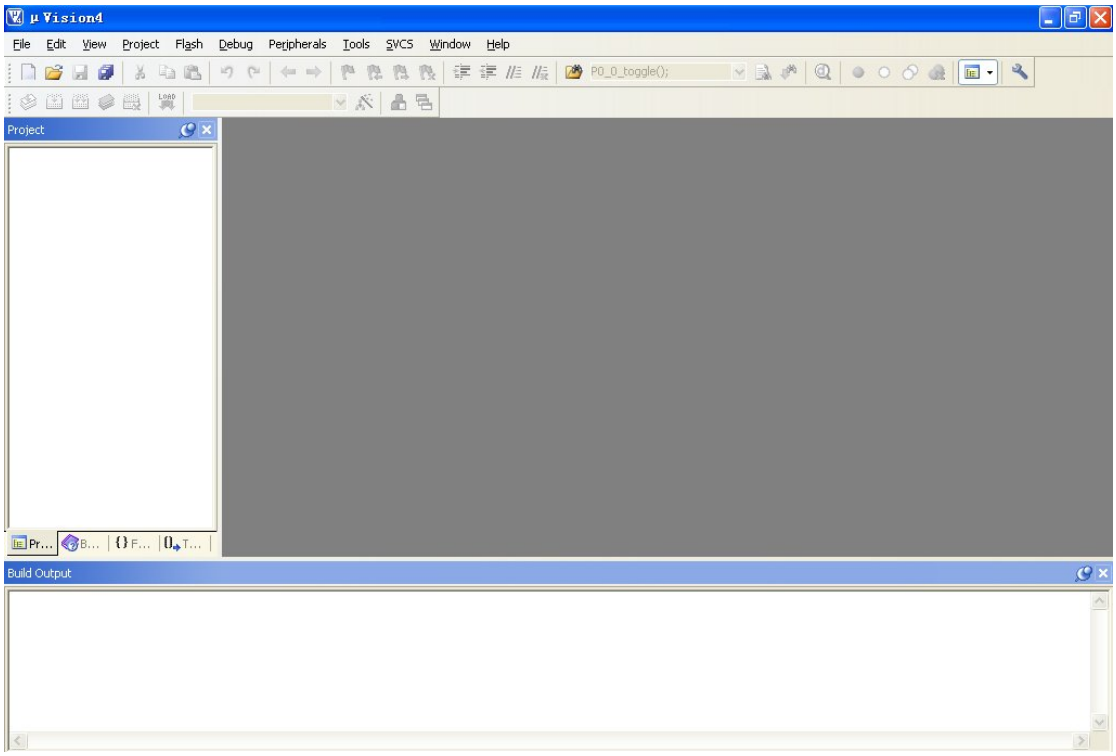
# 教你一步步使用 KEIL-MDK 开发 XMC1300

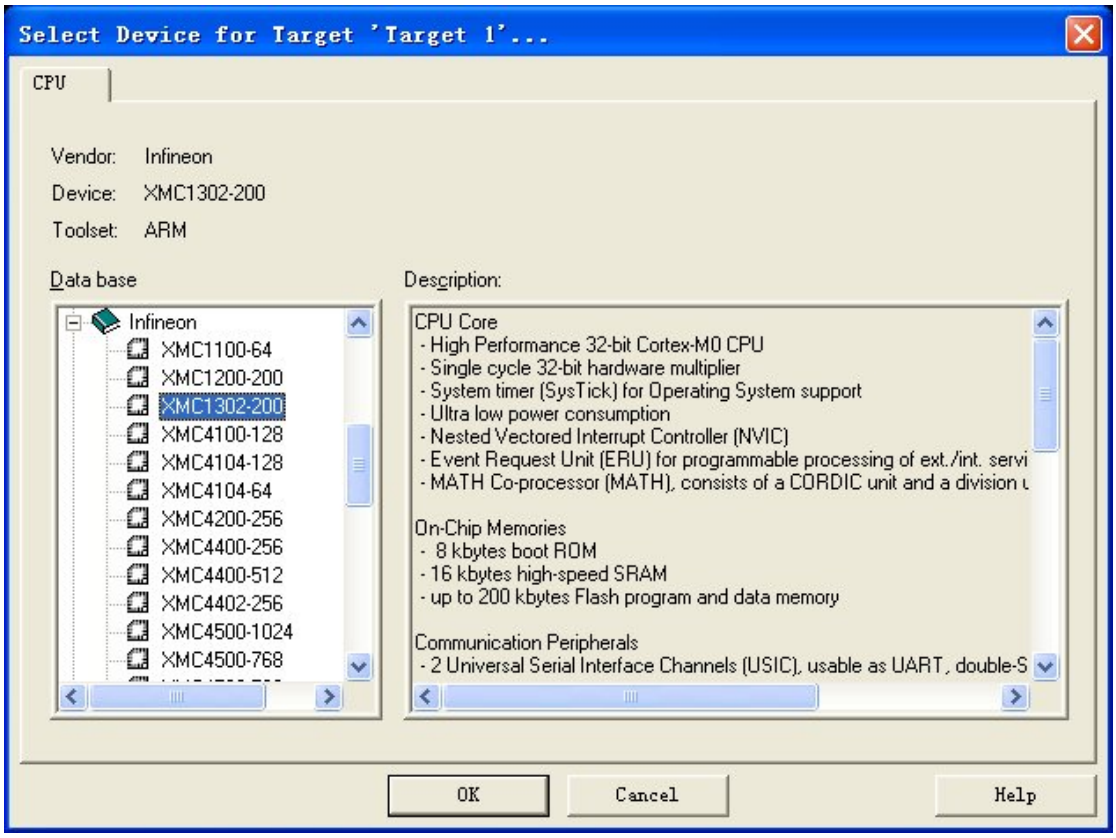
目前 ,手里有一块 XMC1302-Boot Kit 开发板，该开发板采用 XMC1302-038-200MCU，是 XMC1300 系列中最高级的型号，外设资源丰富，采用 M0 内核，并有辅助浮点处理。官网使用手册和例程中，大部分都是采用 DAVE 平台来讲解的，采用 KEI-MDK 环境的资料非常少，下面就用此板子作为讲解平台，以一个翻转板上 6 路 LED 灯为 DEMO 一步步讲解如何建立 MDK 工程，及编写外设底层驱动的过程！硬件平台的外观如下，板载一个 J-LINK 仿真器，采用 XMC4200MCU 实现，如果你有这个板子，可以对照着实践一下：



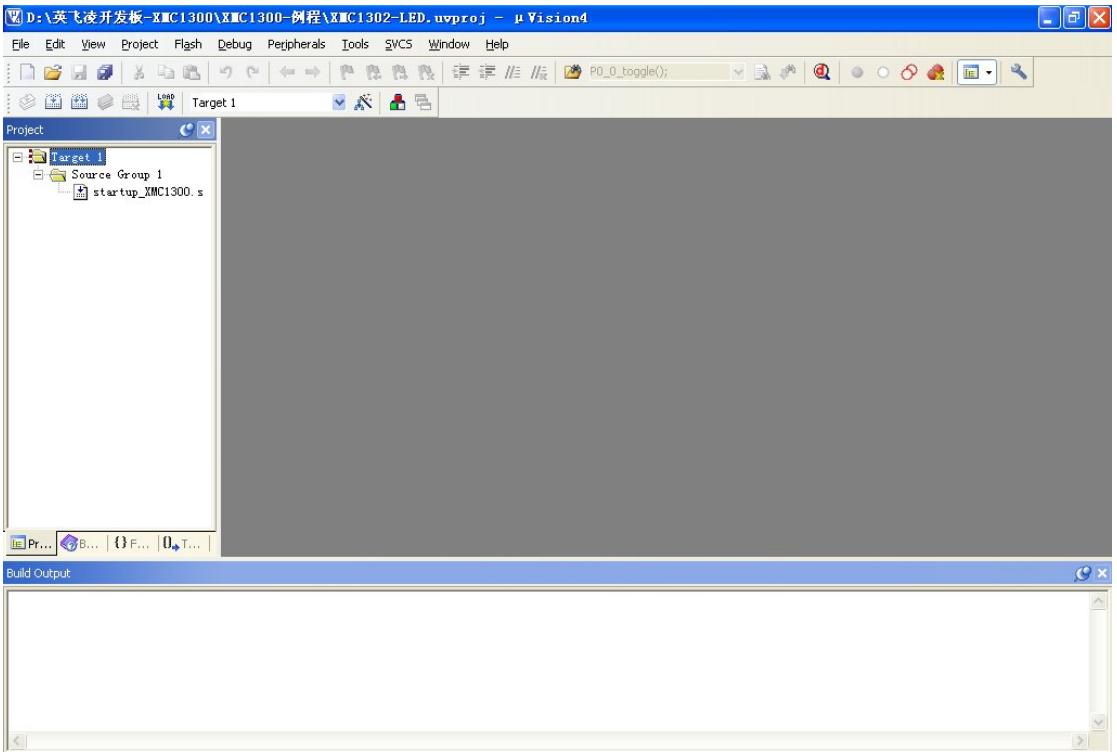

首先打开 KEIL-MDK4,这里用的是 V4.72A 版本，这个版本是带有 XMC1300 系列型号的，采用 MDK5 版本也可以，需要安装 XMC1300 的补丁包。



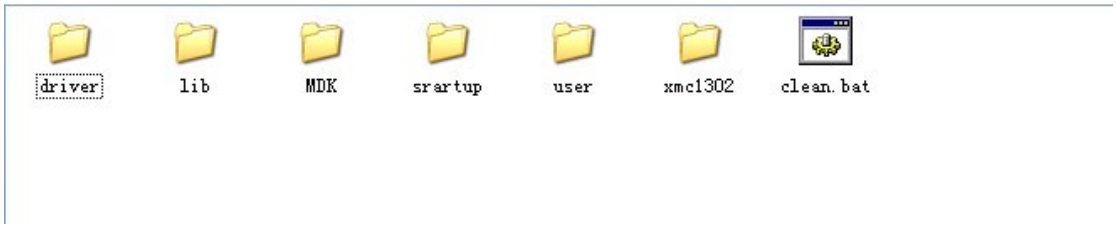

建立新工程，XMC1302-LED，然后选项 MCU 型号




在这个版本里面，选择 XMC1302-200 这个型号，点击确定后，KEIL 会自动增加 startup_XMC1300.s 的启动文件。



关闭 KEIL-MDK 工程，返回到建立工程的文件夹，在文件夹下面建立如下文件夹，并增加文件。

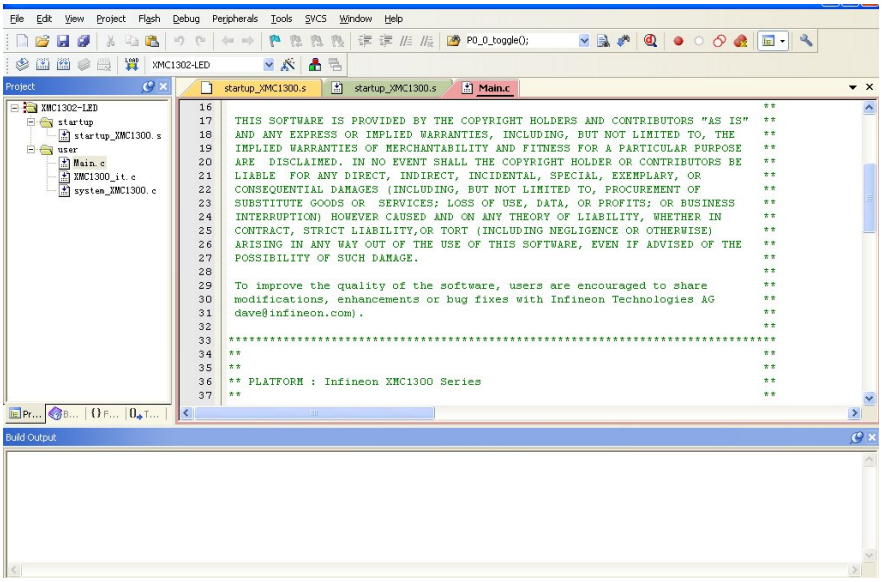



其中 driver 放置底层外设驱动，lib 放置 XMC1300 的库文件，有些需要自己实现。

MDK 放置工程文件，startup 放置启动文件.s，user 文件夹放置主函数和预编译文件，XMC1302 放置必要的系统文件，比如 XMC1300.h 文件，时钟配置文件，中断文件，如图：


Clean.bat 是一个批处理命令，用于清理工程编译产生的额外文件，要不会占用很多空间。

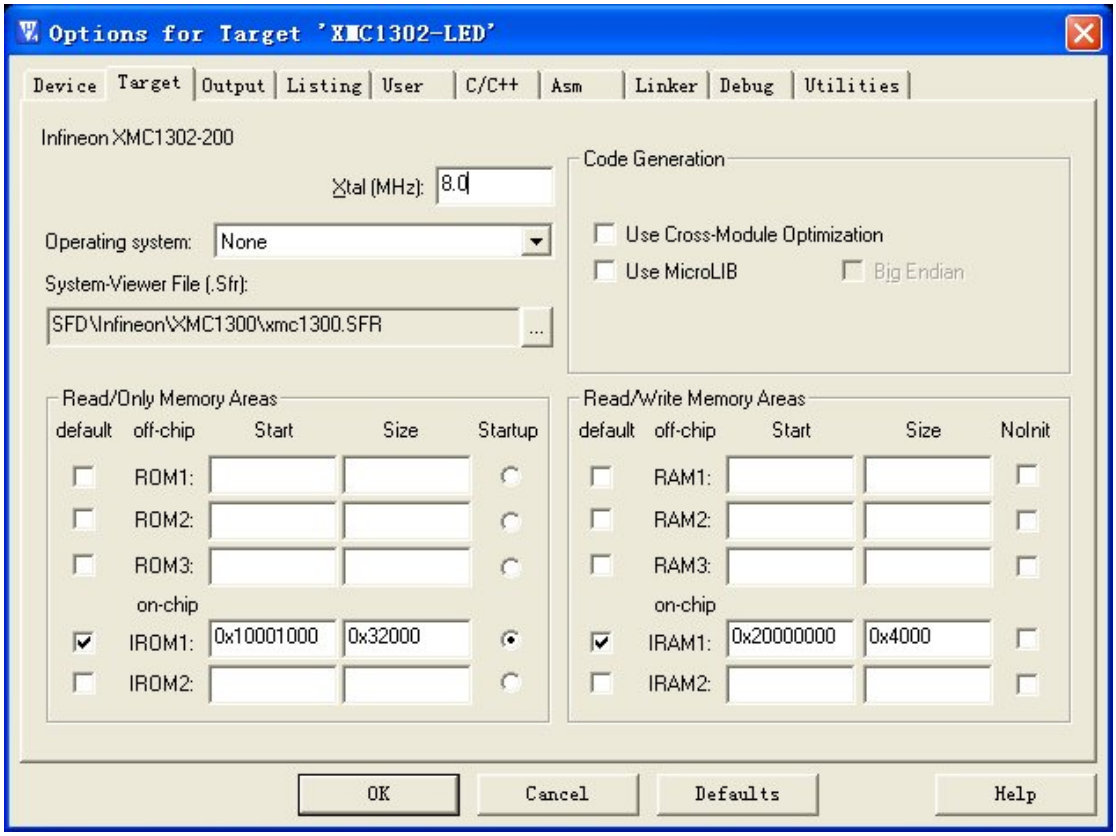
建立好工程文件夹，在 MDK 文件夹里面，找到工程文件名，再打开，之前，把刚才建立的工程文件已经放置到里面了。在工程里面，修改 MCU 配置，组名称，并建立一个组为 user，吧 main.c、xmc1300_it.c 和 system_XMC1300.c 加入，其中 system_XMC1300.c 文件在 MDK 里面自带，可以从安装文件里面找到，xmc1300_it.c 中断文件需要自己实现，这个根据启动文件名来实现一下，不难



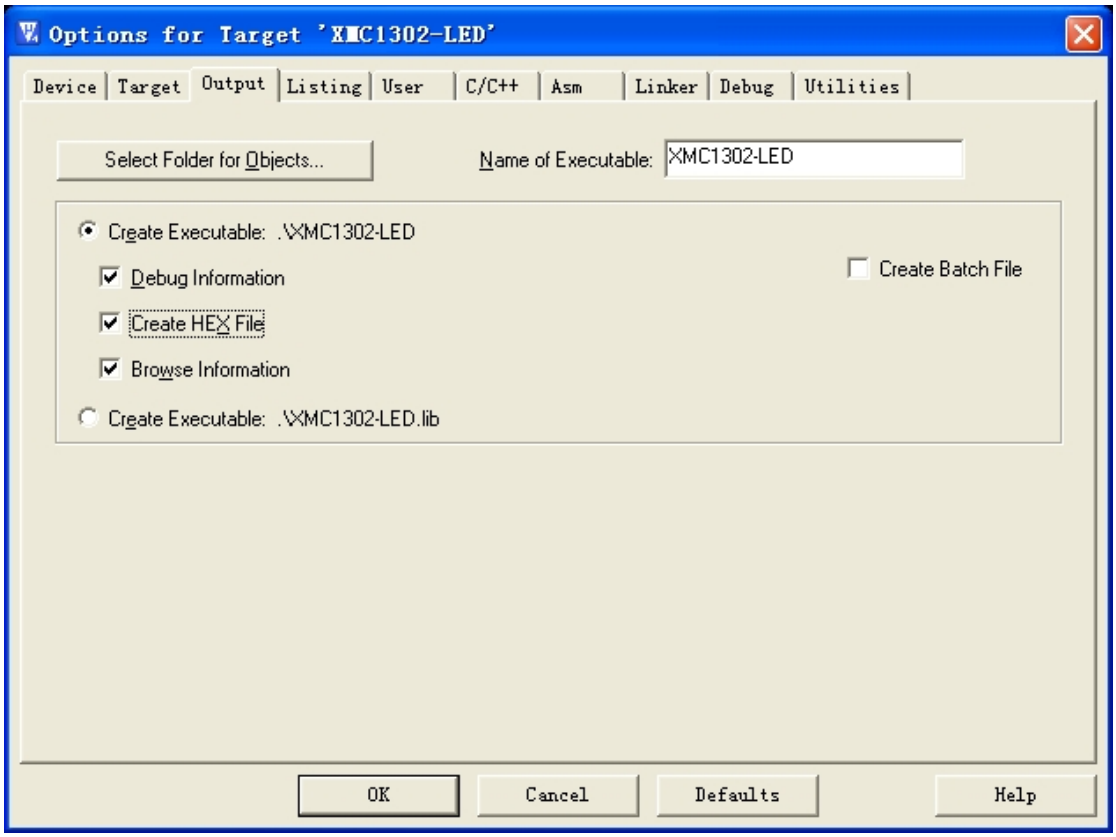

建立好文件组和添加文件后，就需要对 MDK 的工程进行配置，配置后才能顺利实现程序下载。

```
**                                                                            **
**                                                                            **
** PLATFORM : Infineon XMC1300 Series                                         **
**                                                                            **
** AUTHOR : TomasoAt                                                  **
**                                                                            **
** version 1.0.0                                                      **
**                                                                            **
** MODIFICATION DATE : 21 Feb, 2013                                           **
**                                                                            **
********************************************************************************/


#include <XMC1300.h>    //SFR declarations of the selected device
#include <gpio_xmc1300_tssop38.h>
#include "system_XMC1300.h"
#include "XMC1300_it.h"

unsigned short int systick_num=0;

int main(void)
{

  //----CLOCK-SETUP--------------------------------------------------------------------------------
  SCU_GENERAL->PASSWD = 0x000000C0UL;
  SCU_CLK->CLKCR = 0x3FF00400UL;      // 8 MHz MCLK, 8 MHz PCLK
  while((SCU_CLK->CLKCR)&0x40000000UL); // wait for VDDC to stabilize
  SCU_GENERAL->PASSWD = 0x000000C3UL;
```

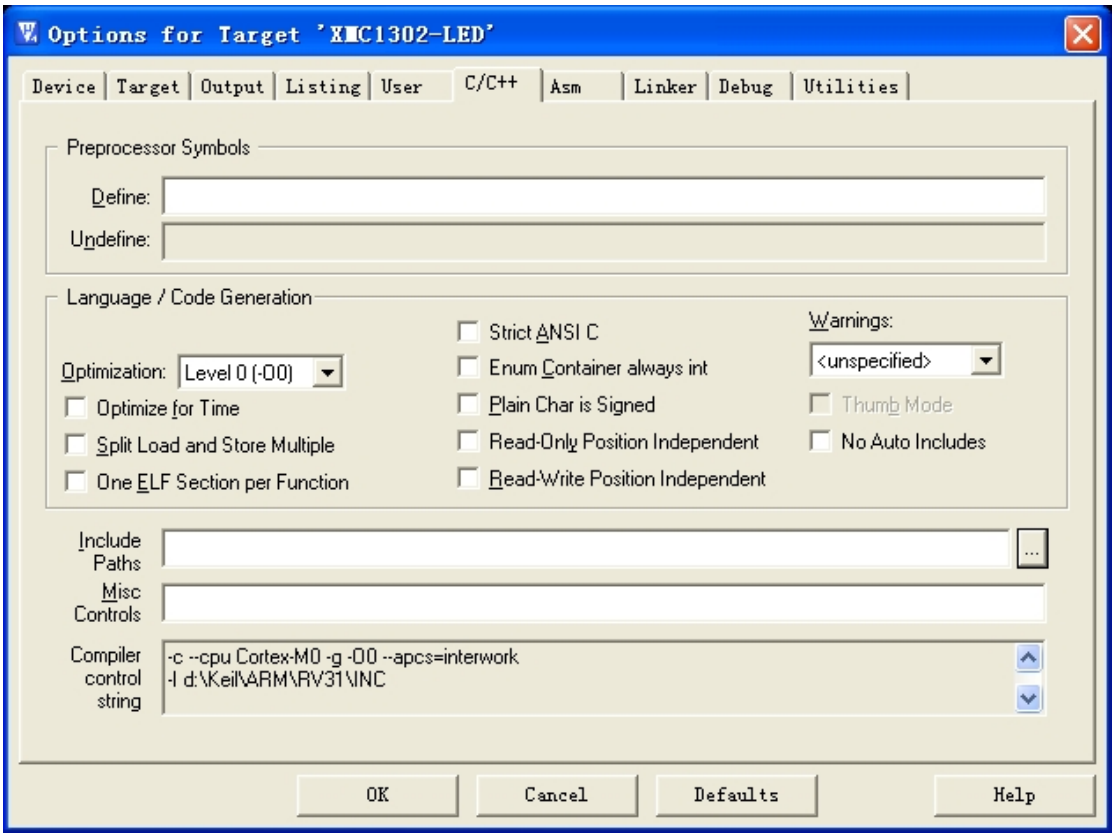
在函数中，因为要实现翻转 LED 灯，需要用到 GPIO 函数，这里使用自带的宏定义.h 文件。不设置工程路径，KEIL 会提示错误。



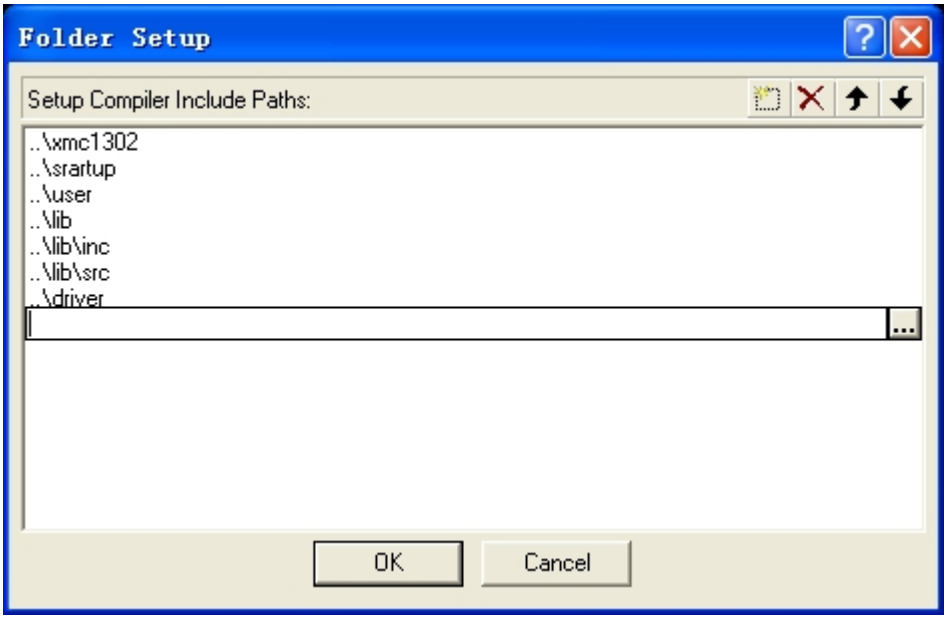

在 KEIL 工程选项中，首先修改晶体频率，开发板采用的是内部晶振，8MHZ，原来默认 12MHZ。



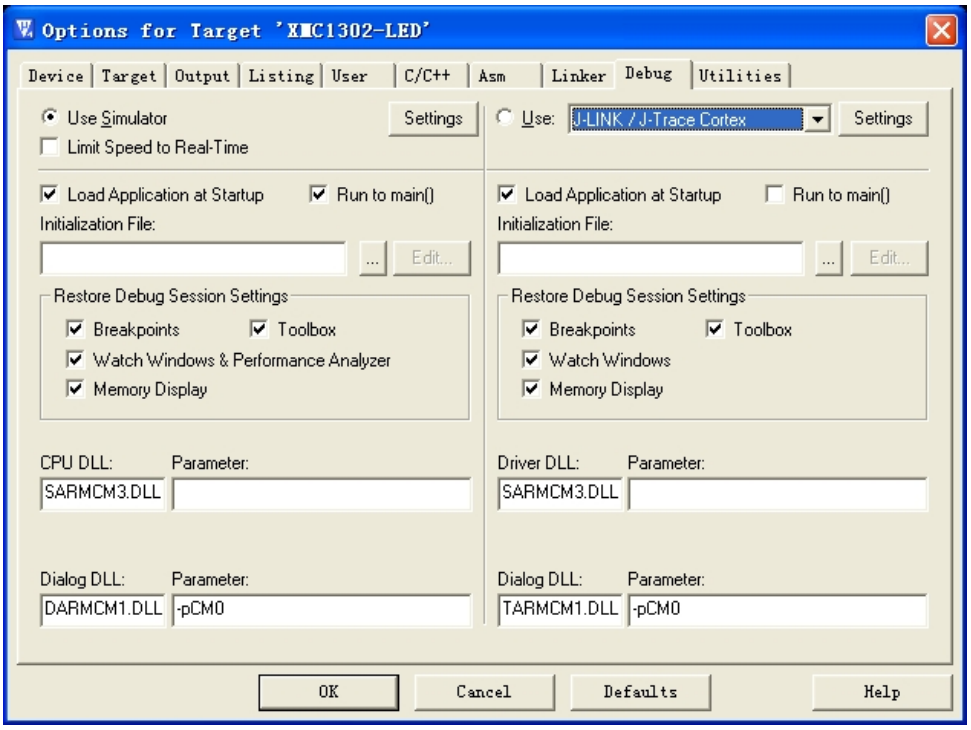
在输出设置里面，选择创建 hex 文件选项，方便通过其它方式下载



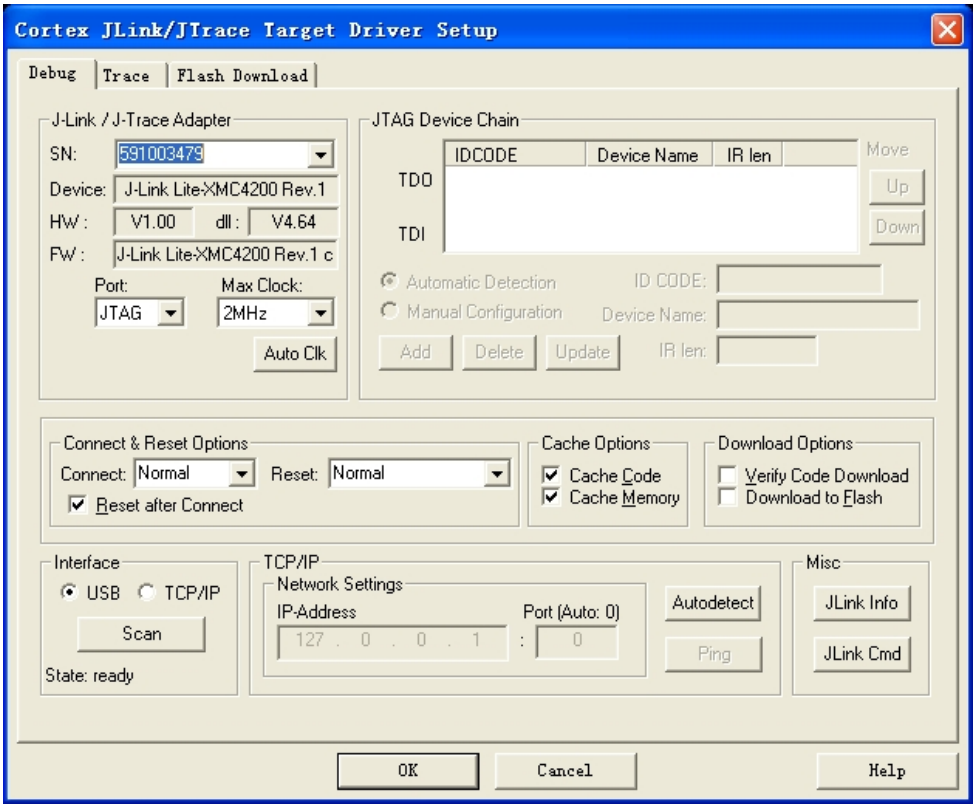

在【C++】选项卡里面，增加包含的工作路径，设置好的路径如下



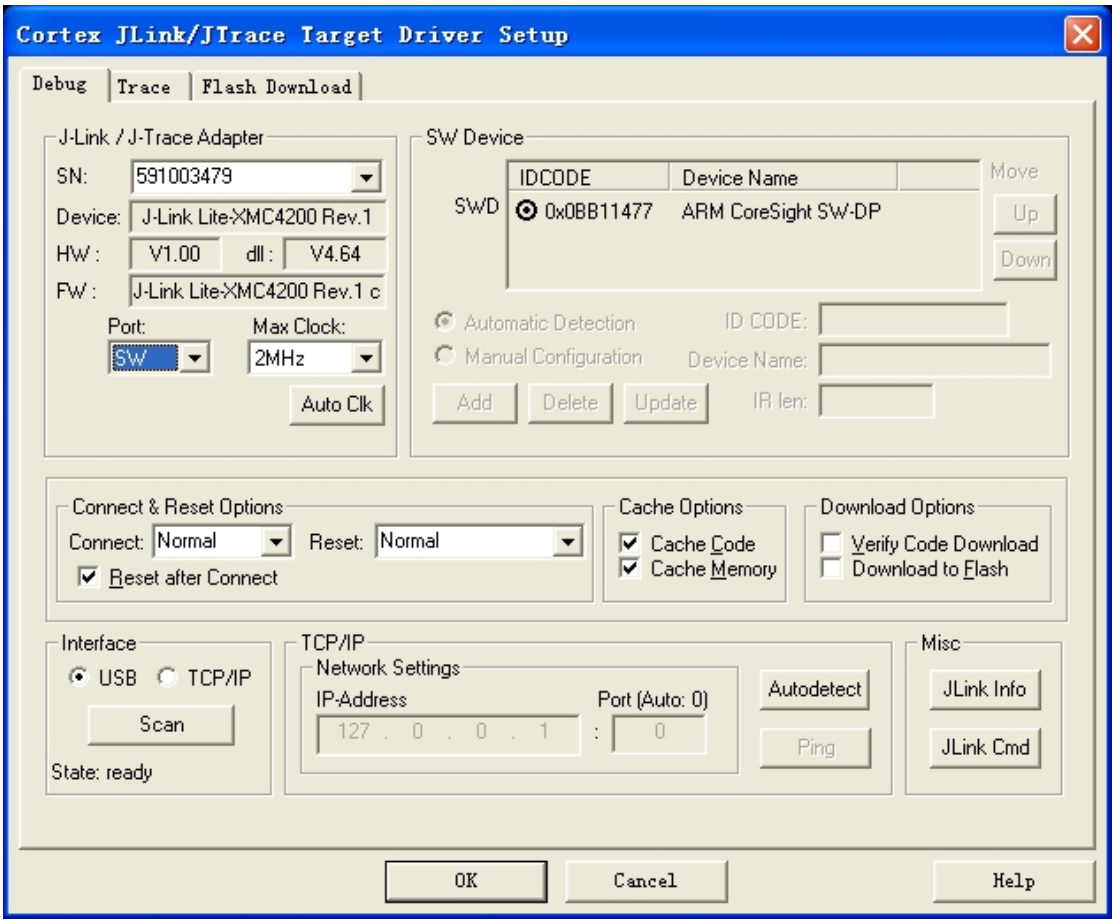

最重要的是，设置调试环境，选择 J-LINK/J-Trace Cortex 仿真器，默认是 ULINK2，点击【Settings】




这里系统已经识别到了仿真器型号，可以看到 J-LINK 的详细信息，默认是 JTAG 方式，但是没有发现 MCU 内核。这个是很多初学者容易迷惑的地方，其实 XMC1302 开发板的 J-LINK 仅支持 SW 方式，不支持 JTAG 方式，所以，需要修改端口为【SW】

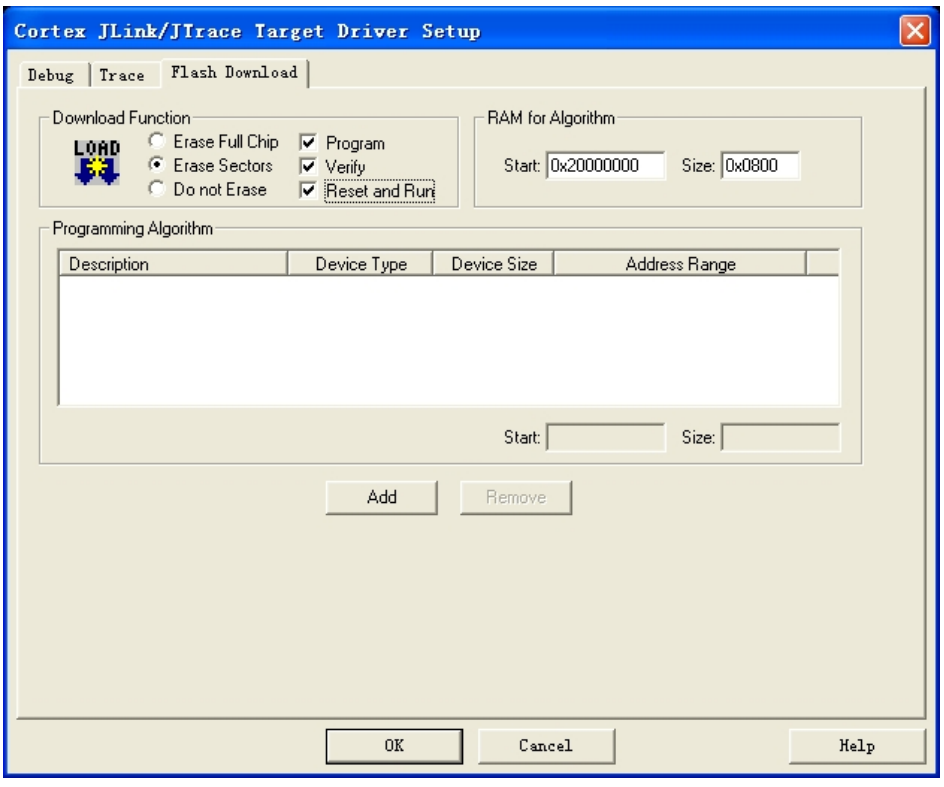



修改后，J-LINK 仿真器就识别到了

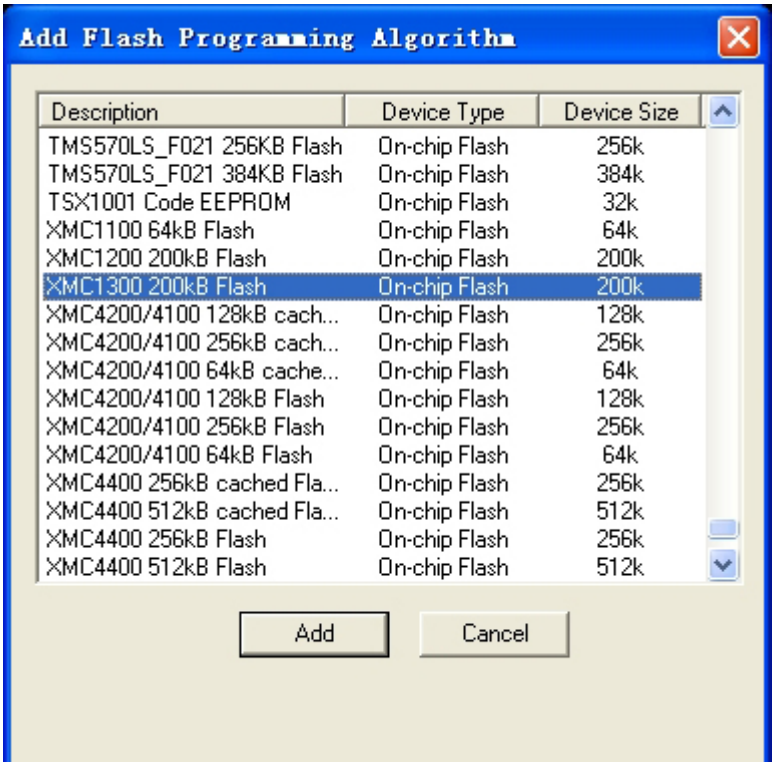



如果仿真器界面出来这个信息，恭喜，你的板子硬件连接是成功的！

下面需要增加下载算法了，不增加下载算法，KEIL 会给出 DLL 错误的提示

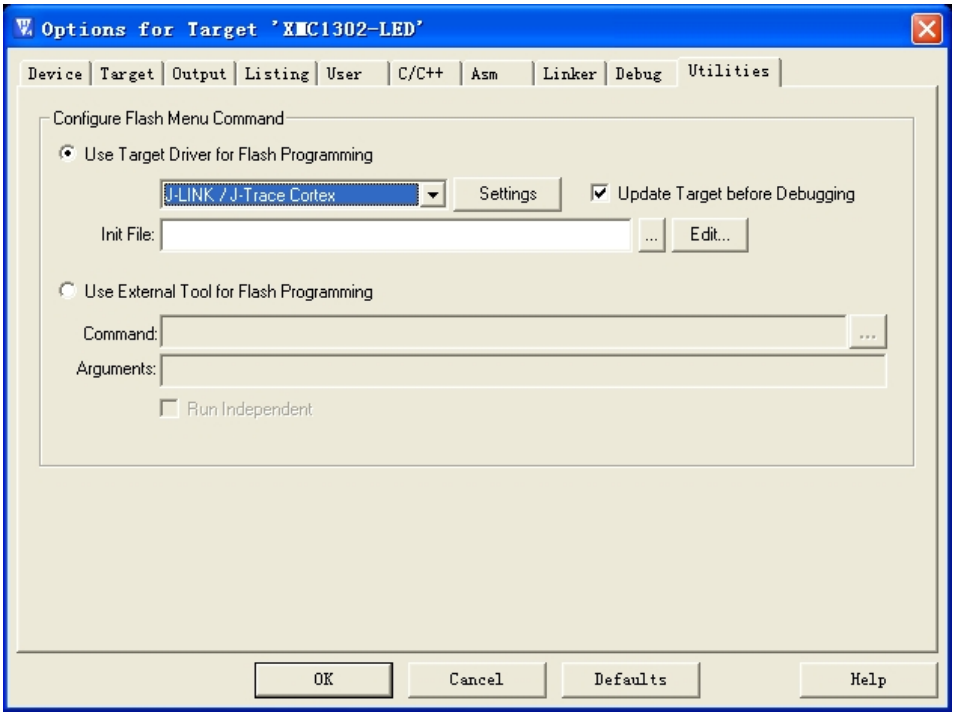


找到该选项卡，点击【ADD】增加下载算法




增加后，在下载功能中，勾选【复位并运行】


除此之外，还需要在【Utilities】中，设置一下仿真器，和【Debug】的设置对应




这样设置后，仿真和下载程序环境就设置好了。

在主函数 main.c 里面实现如下

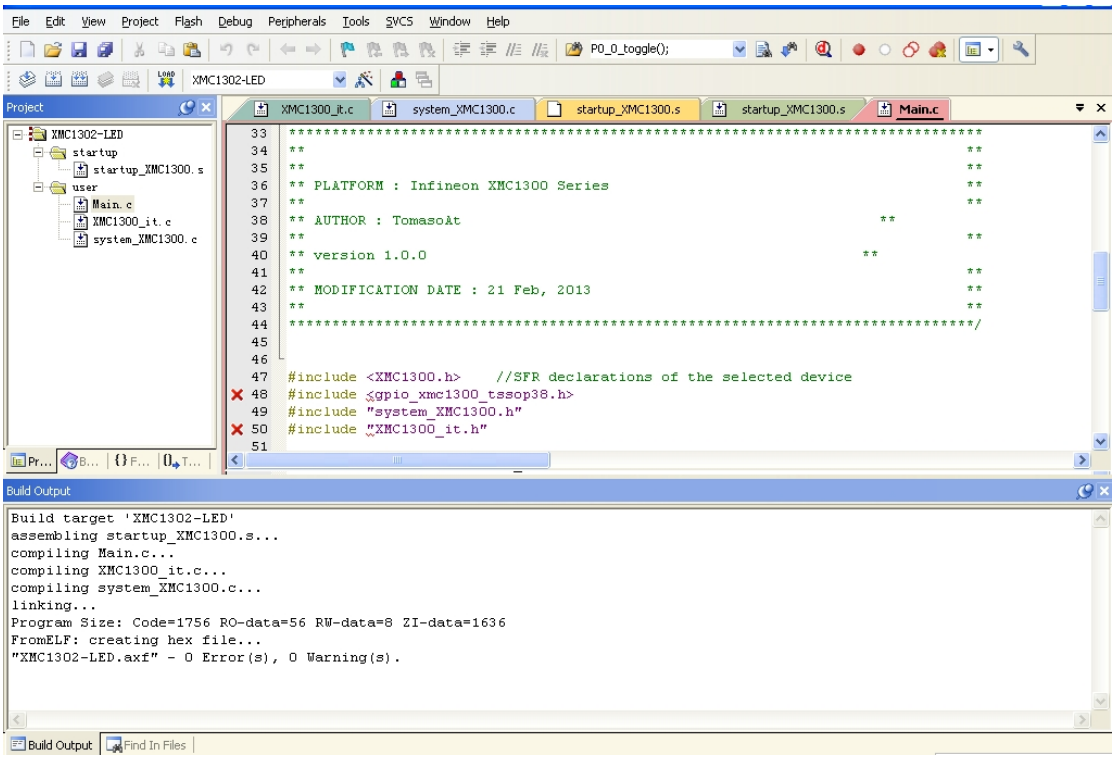
前面是时钟配置，设置为 8MHZ,内部时钟，中间是将 GPIO 配置为推挽输出，再针对 systick 定时器进行设置，定时 10ms 的基准

```
//----CLOCK-SETUP--------------------------------------------------------------
SCU_GENERAL->PASSWD = 0x000000C0UL;
SCU_CLK->CLKCR = 0x3FF00400UL;      // 8 MHz MCLK, 8 MHz PCLK
while((SCU_CLK->CLKCR)&0x40000000UL); // wait for VDDC to stabilize
SCU_GENERAL->PASSWD = 0x000000C3UL;



//------------------------------------------------------------------------------

//----PIN-SETUP----------------------------------------------------------------
P0_0_set_mode(OUTPUT_PP_GP);
P0_1_set_mode(OUTPUT_PP_GP);
P0_6_set_mode(OUTPUT_PP_GP);
P0_7_set_mode(OUTPUT_PP_GP);
P0_8_set_mode(OUTPUT_PP_GP);
P0_9_set_mode(OUTPUT_PP_GP);
//------------------------------------------------------------------------------

//----SYSTICK-SETUP------------------------------------------------------------
SystemCoreClockUpdate();
SysTick_Config(SystemCoreClock / 10); // 0.1s interrupt base
//------------------------------------------------------------------------------
```

在中断里面，计数器，每计数到 5，也就是 50ms 定时，翻转 6 个 GPIO

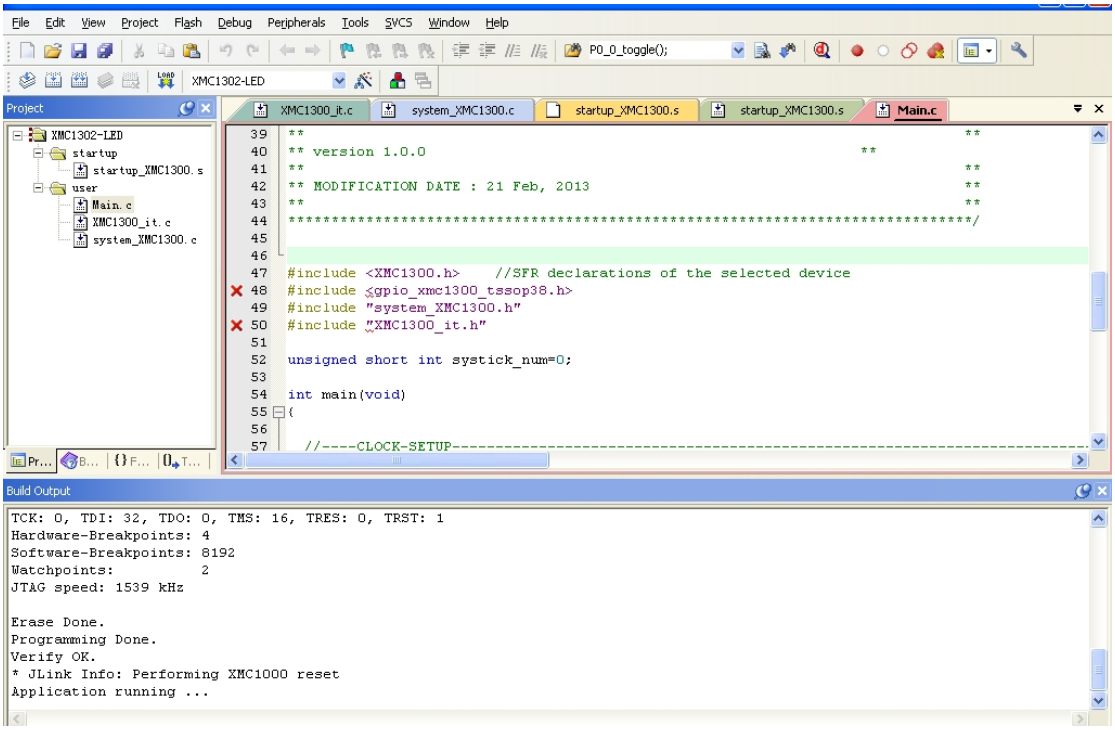
```
/**
  * @brief  This function handles SysTick Handle
  * @param  None
  * @retval None
  */
void SysTick_Handler(void)
{

      systick_num ++;
      if(systick_num==5)
      { P0_0_toggle();
        P0_1_toggle();
        P0_6_toggle();
        P0_7_toggle();
        P0_8_toggle();
        P0_9_toggle();
        systick_num=0;
      }

}
```

将程序编译一下，在 MDK 里面经常会出现包含文件的错误提示，这个对程序不影响，可以忽略

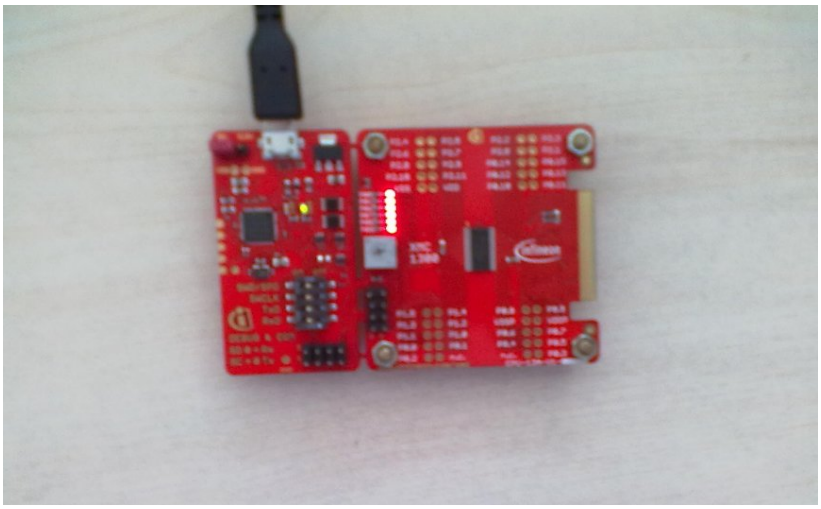



经过上面的设置，一个可以运行的工程环境就建立起来了。

把程序下载一下




FLASH 被擦除，程序运行了，运行现象：



在熟悉 KEIL-MDK 建立工程环境和设置仿真器后，可以在上面增加需要的底层驱动。和相关的库文件，这里可以结合 DAVE 产生的驱动文件源码来实现!

Fengye5340