建立第一个工程

一、前期软件要求

需要预先安装如下软件：

1. MDK522：KEIL5.22 安装软件（Arm 开发软件[Keil for Arm]）
2. MDKCM522：KEIL LEGACY 安装软件，兼容 5 以前版本
3. Keil.TM4C_DFP.1.1.0：TM4C 系列芯片及板的 DFP(device family pack)安装包
4. SW-EK-TM4C1294XL-2.1.4.178：TM4C1294XL 驱动及样例程序
5. ICDI 调试驱动：调试驱动（代码下载）

默认安装完成后，有两个目录【根据安装的目录设置不同，目录允许有所不同】：

1. *C:\Keil_v5* KEIL 可执行文件目录
2. *C:\ti\TivaWare_C_Series-2.1.4.178* TIVA 系列驱动及样例

图 1 C:\ti\TivaWare_C_Series-2.1.4.178 目录内容

说明：图 1 包含开发基于 TM4C1294XL 系列芯片的电路板的各种资源，其中目录\inc 中包含针对 TM4C1294XL 系列芯片开发的.h 文件；目录\driverlib 中包含针对 TM4C1294XL 系列芯片开发的各种功能的.c 源文件以及对应的.h 文件，这些文件被编译一个.lib 库文件[不同的开发软件所编译的库文件格式有所不同，Keil 所生成的库文件位于目录\driverlib\rvmdk]。

二、硬件要求

WIN7 及以上操作系统，2G 内存

TM4C1294XL 板及 TM4C1294XL_SUBBOARD 组合板，即 A2000TM4 板

Micro-USB 数据线一根

三、新建用户目录设为 *C:\A2000TM4\EXP0*

因为需要使用 TM4C1294 芯片的硬件定义以及固件库，因此从 *C:\ti\TivaWare_C_Series-2.1.4.178* 中将 *INC* 及 *DRIVERLIB* 两个子目录拷贝到用户目录中。

四、STEP-BY-STEP 建立一个项目

1. 将 MICRO-USB 数据线一端接电脑，一端接 TM4C1294XL 的数据口（非网口端）
2. 打开 KEILuVision5，Project→New uVisionProject，新建一个项目。选择目录 *C:\A2000TM4\EXP0* 后，建立新项目 exp0。选择设备如图 2 所示：

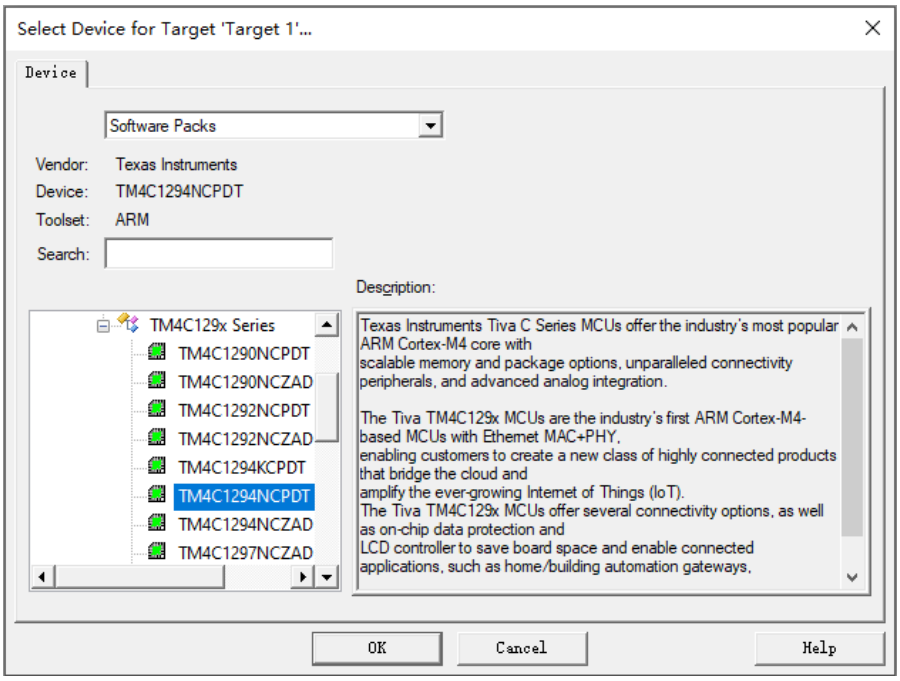

图 2 选择设备 TM4C1294NCPDT

3. 让 KEIL 代我们生成启动代码，点击图 3 中 Software Component 项下的 Device 项，勾选其中的 STARTUP 选项。点击“OK”。

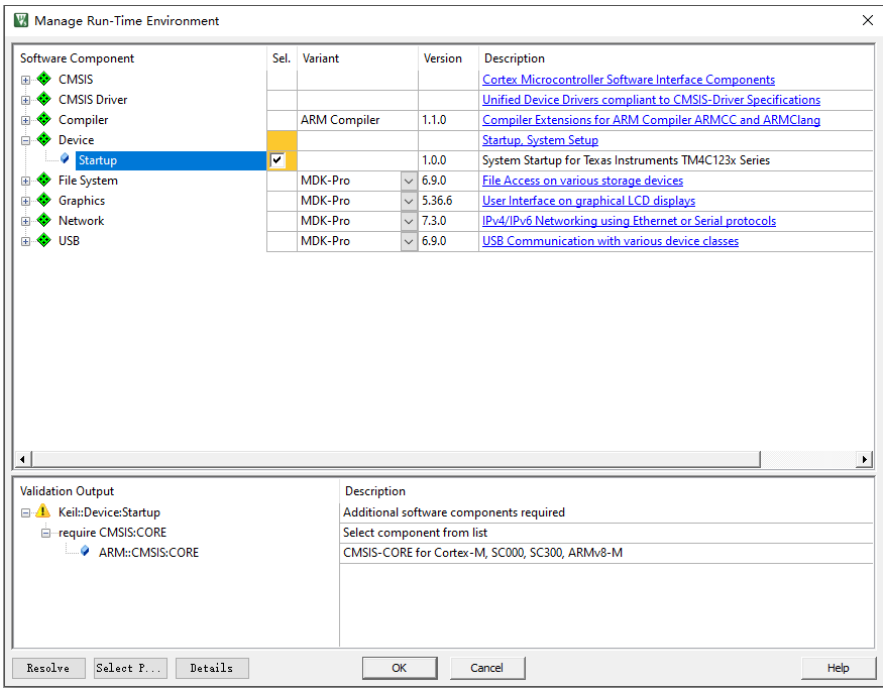

图 3 勾选其中的 STARTUP 选项

4. 这样系统自动生成一个项目，项目中包括了一个 Source Group 1，用来放置源文件，目前为空；以及一个设备目录 Device，包括 Startup_TM4C129.s 以及 System_TM4C129.c 两个文件。

   Startup_TM4C129.s　　配置了堆栈和中断函数名称以及从复位到 main 函数前的处理过程

   System_TM4C129.c　　默认系统初始化，配置了默认时钟

5. 单点 Source Group 1，右键，选择 Add New Item to Group “Source Group 1”如图 4 所示，选择生成的文件类型为.c 文件，在 Name 栏输入“exp0”，这样就生成一个源文件 exp0.c。编辑该文件并完成一个最简单的主函数如下：

```
int main(void)
{
    while (1);
}
```

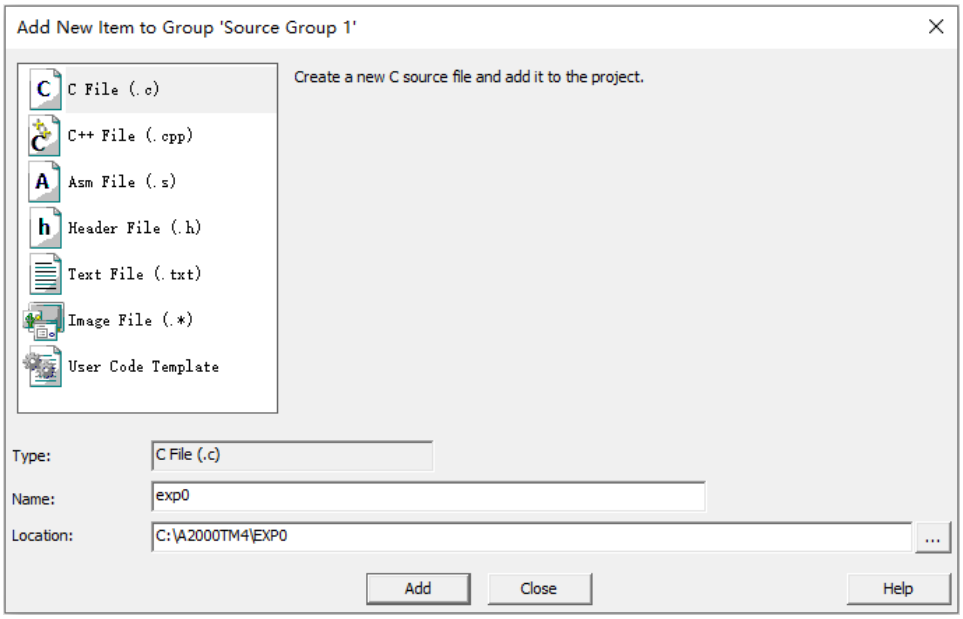

图 4 创建源文件 exp0.c

6. 这样建立了一个最简单的项目，可以试着编译一下 Project→Build Target，可以看到应该是无错误。

```
Build Output
*** Using Compiler 'V5.06 update 4 (build 422)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'Target 1'
compiling exp0.c...
linking...
Program Size: Code=1312 RO-data=544 RW-data=4 ZI-data=612
FromELF: creating hex file...
".\Objects\exp0.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

图 5 编译项目

7. 现在我们设计一个利用按键按下与松开控制 LED 灯快闪和慢闪的项目。芯片管脚 PJ0 外接按键（USR_SW1）,按键没有按下时，PJ0 管脚为高电平；按键按下时，PJ0 管脚为低电平。芯片管脚 PF0 外接 LED（D4），管脚 PF0 置 1，LED 点亮；管脚 PF0 置 0 时，LED 熄灭。USR_SW1 未按下时，点亮 LED 约 500 毫秒，然后熄灭 LED 约 500 毫秒，形成慢闪效果。当 USR_SW1 按下时，点亮 LED 约 50 毫秒，然后熄灭 LED 约 50 毫秒，形成快闪效果。

表 1 资源表

资源	管脚名称	有效电平
LED（D4）	PF0	高
按键（USR_SW1）	PJ0	低

首先对项目进行进一步配置，将驱动库文件添加到项目中。点在 Target 1，右键，Add Group；点在 New Group，右键，Add Existing File to Group “New Group”，将 \driverlib\rvmdk\driverlib.lib 库文件添加到此组中，如图 6 所示。注意选择文件类型为 LIB。

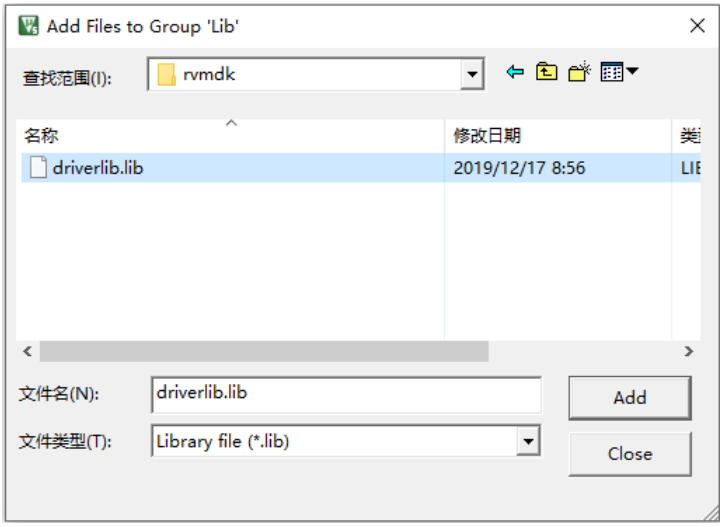

图 6 添加 driverlib.lib 库

点在 New Group 上，修改名称为 Lib。

点在 Target 1，右键，选择 Option for Target “Target 1”，需要变动的项目如下表 2 所示。

表 2 Option for Target “Target 1” 变动的项目

栏目名称	动作	说明
Output	勾选 creat HEX File	生成供 uniflash 使用的文件
Debug	Use Stellaris ICDI	在线 DEBUG 仿真器
C/C++	Preprocessor Symbols-Define **PART_TM4C1294NCPDT**	CPU 型号预定义，因为 driverlib 中某些头文件需要根据 CPU 类型进行不同预定义
C/C++	.\inc;.\driverlib	指定头文件目录

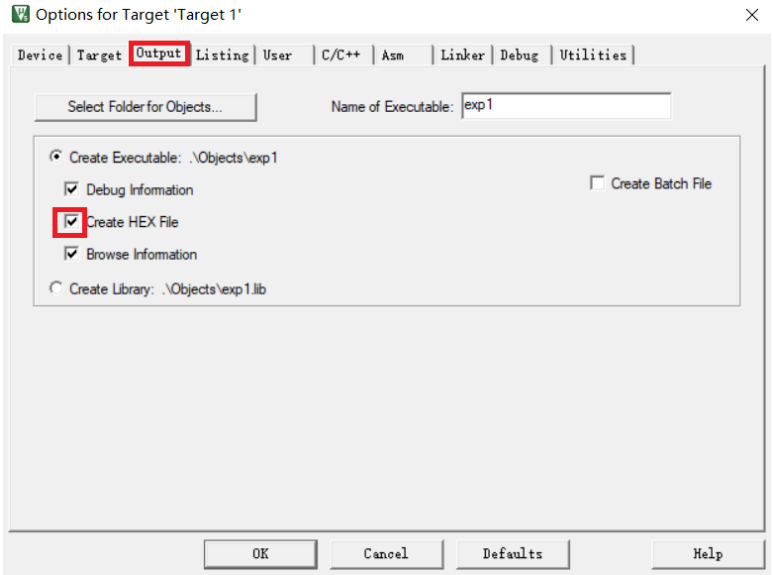

图 7 设置 Output 页

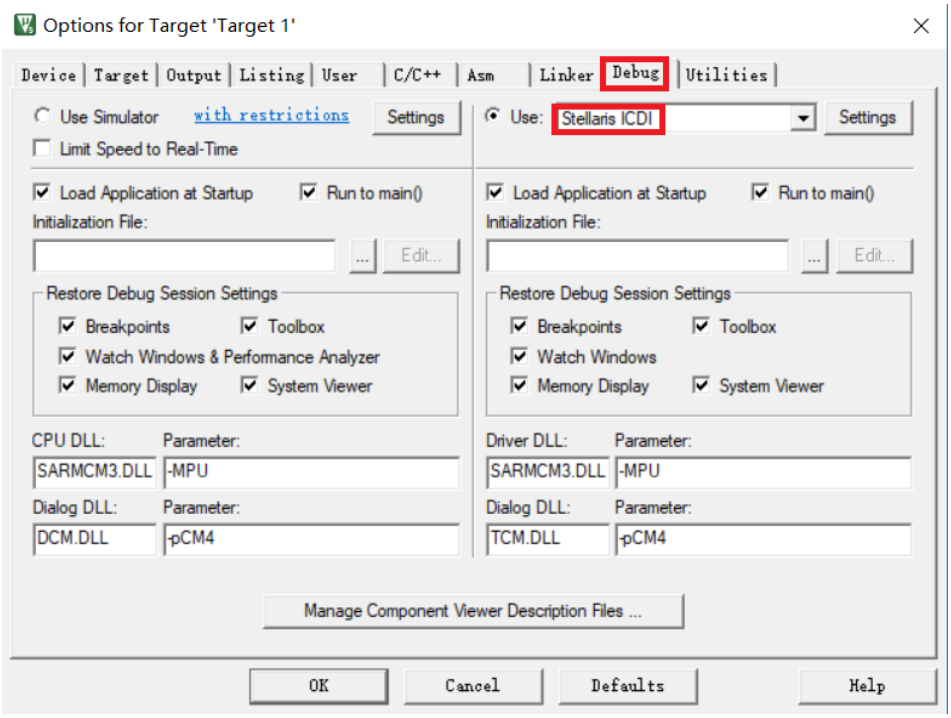

图 8 设置 Debug 页

设置 Debug 页，设置完成后，点击“Settings”继续设置，如图 9 所示。

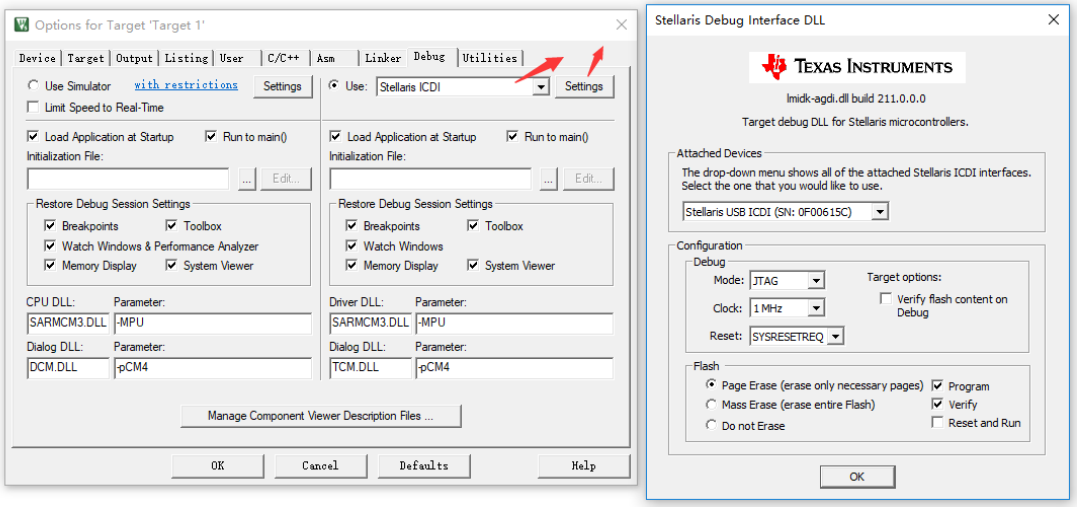

图 9 Stellaris ICDI 的设置

点击“C/C++”继续设置，如图 10 所示。

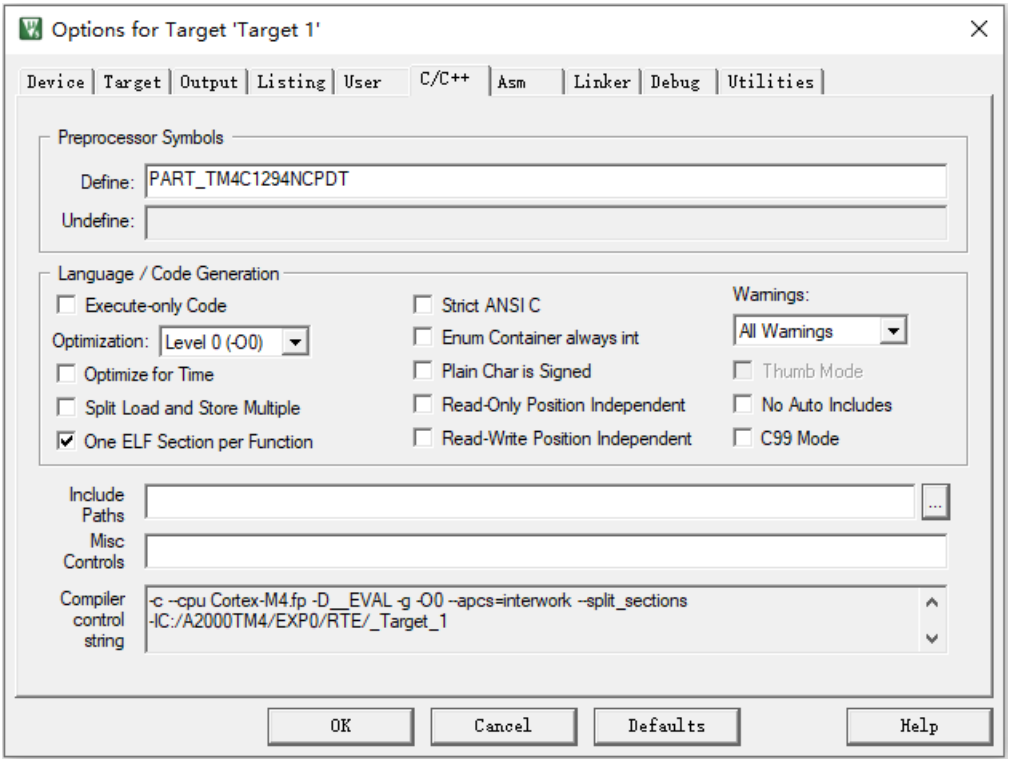

图 10 C/C++页的设置

8. 将 exp0.c 文件改成如下所示

```
//*****************************************************************************
//
// Copyright: 2019-2020，上海交通大学工程实践与科技创新 II-A 教学组
// File name: exp0.c
// Description: LED（D4-PF0）以 1000 毫秒为周期缓慢闪烁；
//   当按下 USR-SW1 键，LED（D4-PF0）以 100 毫秒为周期快速闪烁；
//   松开 USR-SW1 键，LED（D4-PF0）恢复以 1000 毫秒为周期缓慢闪烁。
// Author:        上海交通大学工程实践与科技创新 II-A 教学组
// Version: 1.0.0.20191230
// Date：2019-12-30
// History：
//
//*****************************************************************************

//*****************************************************************************
//
// 头文件
//
//*****************************************************************************
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"      // 基址宏定义
#include "inc/hw_types.h"       // 数据类型宏定义，寄存器访问函数
#include "driverlib/debug.h"    // 调试用
#include "driverlib/gpio.h"     // 通用 IO 口宏定义
#include "driverlib/pin_map.h"  // TM4C 系列 MCU 外围设备管脚宏定义
#include "driverlib/sysctl.h"   // 系统控制宏定义

//*****************************************************************************
//
// 宏定义
//
//*****************************************************************************
```

```
#define   MilliSecond          4000      // 形成 1ms 时延所需循环次数
#define   FASTFLASHTIME        50                  // 短延时（50ms）
#define   SLOWFLASHTIME        500                 // 长延时（500ms）

//*****************************************************************************
//
// 函数原型声明
//
//*****************************************************************************
void   DelayMilliSec(uint32_t ui32DelaySecond); // 延迟一定时长，单位为毫秒
void    GPIOInit(void);                          // GPIO 初始化
void   PF0Flash(uint8_t ui8KeyValue);   // 根据传入的按键值，决定 PF0 快闪或慢闪

//*****************************************************************************
//
// 主程序
//
//*****************************************************************************
int main(void)
{
    uint8_t ui8KeyValue;

    GPIOInit();                  // GPIO 初始化

    while(1)                     // 无限循环
    {

        // 读取 PJ0 键值   0-按下 1-松开
        ui8KeyValue = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0);
        PF0Flash(ui8KeyValue);   // 根据传入的按键参数，决定 PF0 快闪或慢闪
    }
}

//*****************************************************************************
//
// 函数原型：void DelayMilliSec(uint32_t ui32DelaySecond)
// 函数功能：延迟一定时长，单位为毫秒
// 函数参数：ui32DelaySecond：延迟毫秒数
//
//*****************************************************************************
void DelayMilliSec(uint32_t ui32DelaySecond)
{
    uint32_t ui32Loop;

    ui32DelaySecond = ui32DelaySecond * MilliSecond;
    for(ui32Loop = 0; ui32Loop < ui32DelaySecond; ui32Loop++){ };
}

//*****************************************************************************
//
// 函数原型：void GPIOInit(void)
// 函数功能：GPIO 初始化。使能 PortF，设置 PF0 为输出；
//           使能 PortJ，设置 PJ0 为输入
// 函数参数：无
//
//*****************************************************************************
void GPIOInit(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);            // 使能端口 F

    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF));   // 等待端口 F 准备完毕

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);              // 使能端口 J
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOJ)){};   // 等待端口 J 准备完毕

    // 设置端口 F 的第 0 位（PF0）为输出引脚
   GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);

    // 设置端口 J 的第 0 位（PJ0）为输入引脚
```

```
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE,GPIO_PIN_0);

    // 端口 J 的第 0 位作为按键输入，类型设置成“推挽上拉”
    GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0,GPIO_STRENGTH_2MA,
                     GPIO_PIN_TYPE_STD_WPU);
}

//*****************************************************************************
//
// 函数原型：void PF0Flash(uint8_t ui8KeyValue)
// 函数功能：根据传入的按键值，决定 PF0 快闪或慢闪。0-快闪，1-慢闪
// 函数参数：ui8KeyValue：按键值
//
//*****************************************************************************
void PF0Flash(uint8_t ui8KeyValue)
{
  uint32_t ui32DelayTime;

  if (ui8KeyValue == 0)                                 // USR_SW1-PJ0 按下
    ui32DelayTime = FASTFLASHTIME;
  else                                                  // USR_SW1-PJ0 松开
    ui32DelayTime = SLOWFLASHTIME;

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0); // 点亮 D4-PF0 LED
    DelayMilliSec(ui32DelayTime);                       // 延时 ui32DelayTime 毫秒

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);  // 关闭 D4-PF0 LED
    DelayMilliSec(ui32DelayTime);                       // 延时 ui32DelayTime 毫秒
}
```

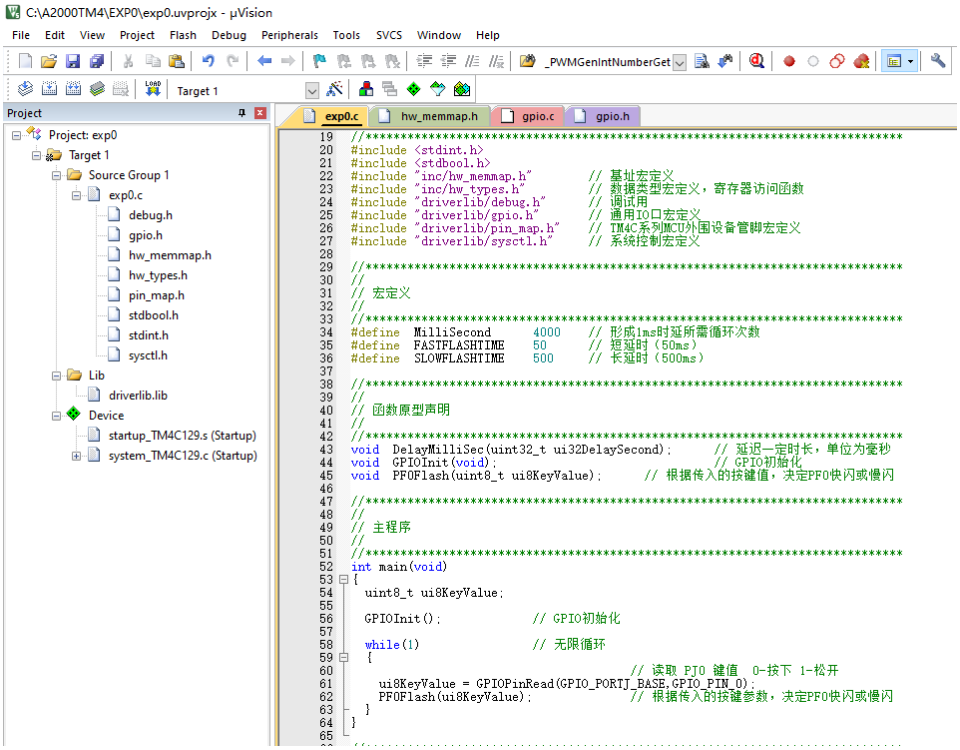

图 11 项目 exp0

9. 编译：Project→Build Target，没有错误。
10. 程序下载：Flash→Download，将程序下载到 MCU 中。
11. 程序运行：按板上 RESET 键，程序运行。TM4C1294XL 板上的 LED(D4)灯约以 1 秒为周期慢速闪烁；按下 USR_SW1，LED(D4)灯约以 100 毫秒为周期快速闪烁；松开 USR_SW1，LED(D4)灯恢复约以 1 秒为周期慢速闪烁。

五、显示汉字

如果程序中有汉字，Edit→Configuration…，将 Encoding 一栏选择为 Chinese GB2312(Simplified)即可，如图 12 所示。

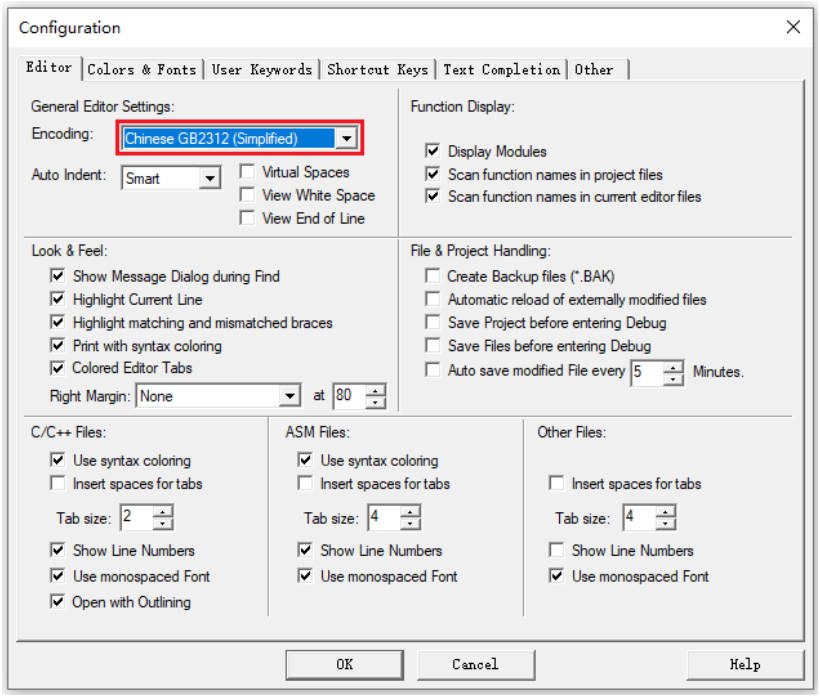

图 12 配置 Encoding