目 录

第 1 章 MSP430 快速入门

因为最近转入 MPS430 的技术支持工作，所以现在开始学习 430 的开发。

由于之前用过 51,也用过 TI 的 ARM CORTEX-M3,但是就是没有用过 TI 的 430，所以将我学习 430 的过程写出来，给像我一样之前没有 430 开发经验的工程师快速入门做个参考。

1.1 准备工作

如果你准备学习 MSP430，在开始之前需要准备如下资料与工具:

1. 430 的开发板与调试仿真工具

我公司有个 TI 430 LaunchPan 开发板，我正好拿它入门。该开发板上自带一个 MSP-EXP430G2 的调试仿真工具。MSP-EXP430G2 适用于所有带 2 线调制功能的 430。

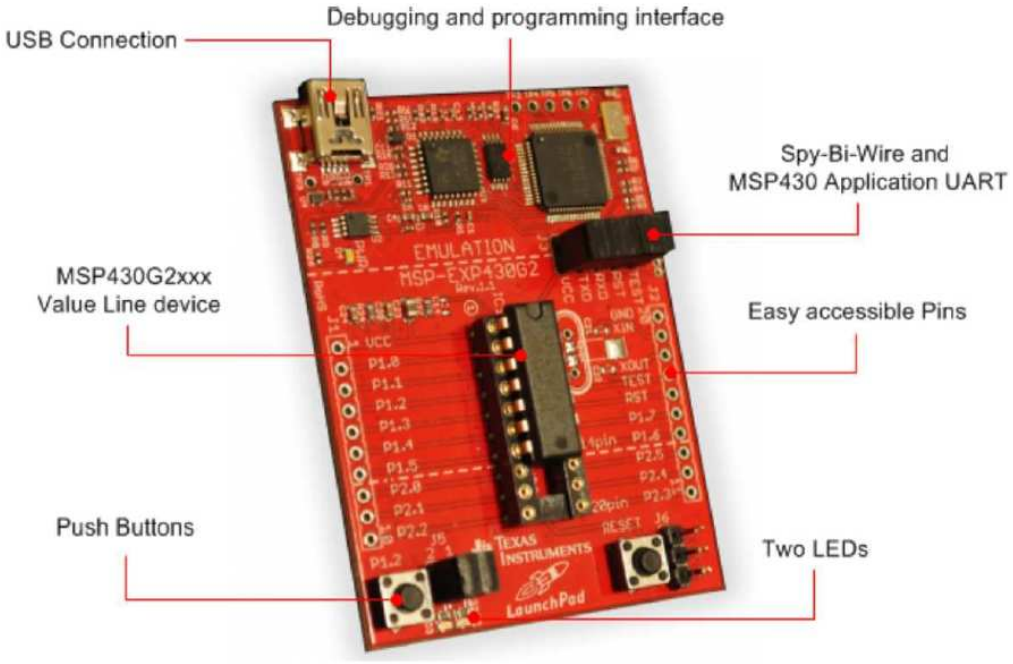


图 1.1 LaunchPan 开发板

2. 开发环境

MSP430 的开发平台，主要有 IAR 和 TI 的 CCS。

IAR 是一款非常优秀的编译调试开发环境，具有业界最高的代码编译效率，而且几乎支持目前所有主流的 MCU，如 8051\430\ARM\AVR\PIC 等。

CCS 是一款 TI 公司自己的开发环境，支持 TI 所有系列的处理器，如 DSP\ARM\430 等。

从性能和使用人数来看，IAR 是无可争议的主流，所以我一般推荐使用 IAR，且我后面的介绍也是基于 IAR 开发平台的。

3. 如何获得 IAR 的 430 版本

IAR 是一款收费软件，但是 IAR 公司有提供 30 天免费试用版和 4K/8K 代码限制永久免费版，可以登陆 IAR 公司官网 http://www.iar.com 获得。需要注意的是，需要先注册一个账号，登陆以后才能下载。

如果你仅仅是用于个人学习，可以使用 30 天免费试用版+和谐文件，获得无限制使用。IAR 的和谐文件，可以从网上获得。

4. 获得 430 相关资料

登陆 TI 官网 http://www.ti.com/ ,进入产品中心，选择 MCU，选择你使用 430 型号所属系列，比如 LaunchPan 开发板，使用的是 MSP430G2231,属于 G2XX 系列，选择 G2XX 系列，在 430 型号列表中找到 MSP430G2231,双击进入。

在这里你可以获得很多关于 MSP430G2231 的资料，你必须获得的重点资料如下：

- DataSheet
- MSP430xxxx Family User's Guide
- MSP430 Optimizing C/C++ Compiler User's Guide

DataSheet，是 CPU 所包含资源的概述，引脚的详细定义，芯片的电气特性等。

MSP430xxxx Family User's Guide，是 430 时钟系统、外设及外设寄存器的详细介绍，适用于某一个系列的 430 芯片，比如 MSP430x2xx Family User's Guide，适用于 MSP430x2xx 系列的 430 芯片。

MSP430 Optimizing C/C++ Compiler User's Guide，是基于该系列 MCU 的代码例程。有适用于 IAR 汇编语言版本，适用于 CCS 汇编语言版本，以及 C 语言版本等 3 个版本。

MSP430 Optimizing C/C++ Compiler User's Guide，包含了系统时钟, GPIO, TIMER, ADC, UART, USI, COMP 等各种外设的具体应用配置例程。其中有一个 Readme.txt 文件，简要的介绍了每个例程的设置。

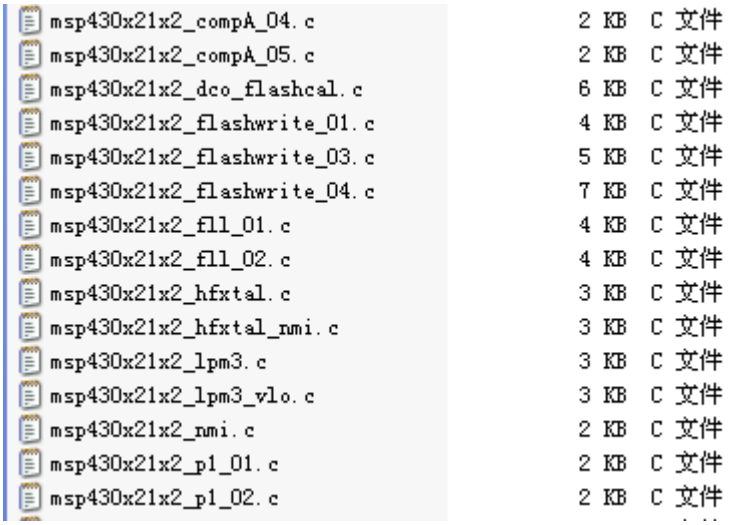


图 1.2 C 代码例程

1.2 MSP430 的时钟系统

要学习 MPS430，首先需要搞清楚他的时钟系统。我们知道 MSP430 是一个超低功耗的 MCU，他的时钟系统以及外设都为实现超低功耗而设计。

我们以 MSP430G2231 为例进行说明。

MSP430G2231 系列有 4 个时钟源，分别是： LFXT1CLK\ XT2CLK\ DCOCLK\ VLOCLK。

LFXT1CLK: 由低频时钟晶体或外接 32768hz 时钟源产生的低频/高频振荡器或由标准晶体、振荡器，或外部 400KHZ--16MHZ 的外部时钟源提供。

XT2CLK: 可供选择的高频振荡器，由标准晶体、振荡器，或外部 400KHZ--16MHZ 的外部时钟源提供。

DCOCLK: 片内可数字控制的振荡器。

VLOCLK: 片内超低功耗、12KHZ 的低频振荡器。

MSP430G2231 可提供 3 个时钟信号，分别是： ACLK\ MCLK\ SMCLK。

ACLK：辅助时钟。ACLK 有软件选择来自 LFXT1CLK 和 VLOCLK 之一的时钟信号。ACLK 经 1,2,3,4 分频后得到。ACLK 可由软件选作各个外围模块时钟。

MCLK: 主时钟。MCLK 有软件选择来自 LFXTICLK，VLOCLK,XT2CLK(如果片内提供)，DCOLCK 之一的时钟信号。MCLK 由 1,2,4,8 分频得到。MCLK 用于 CPU 和系统。

SMCLK：子系统时钟。SMCLK 由软件选择来自 LFXT1CLK，VLOCLK, XT2CLK（片内提供），DCOCLK 之一的时钟。SMCLK 有 1,2,4,8 分频得到。由 SMCLK 看有软件选作各个外围模块时钟。

NOTE: MSP430G2231 在复位上电后，MCLK 和 SMCLK 来自 DCOCLK 的 1.1MHZ，ACLK 来自内部集成 6pF 电容的 LFXT1CLK 的高频模式。使用 DCOCLK 作为 MCLK 也是最常用的时钟设置模式。

我们搞清楚了时钟系统后，接下里需要搞清楚各个外设功能模块，包括 GPIO, TIMER, ADC, UART, USI, COMP 等。由于这篇文章只是快速入门，我在这里不做详细叙述。

1.3 DCO 时钟校准

在上节中我们提到，在上电复位后 MSP430G2231 默认使用 DCOCLK 作为 MCLK 和 SMCLK。

DCO 数字可控的振荡器，误差较大，一般需要校准。MSP430G2231 在出厂时有提供一个 1MHZ 的校准值。为获得较高的时钟精度，一般推荐使用校准时钟。

当然你也可以自己校准，在 MSP430 Optimizing C/C++ Compiler User's Guide 软件包中有一个 msp430x20xx_dco_flashcal.c 的程序，该例程需要外接一个 32768 的晶体作为 LFXT1CLK，然后将校准后值保存在 FLASH 的数据空间。

使用出厂校准值配置 DCO，请看程序清单 1.1。

```
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;                    // Stop watchdog timer

    //=====================================================================
    if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
    {
        while(1);                                // If calibration constants erased
                                                 // do not load, trap CPU!!
    }

    //1Mhz
    BCSCTL1 = CALBC1_1MHZ;                       // Set range
    DCOCTL = CALDCO_1MHZ;                        // Set DCO step + modulation */
    //=====================================================================

    while(1)
    {
    }
}
```

程序清单 1.1 使用出厂校值值配置 DCO

1.4 硬件最小系统

MSP430 的最小系统其实很简单，因为上电后默认使用 DCO，所以不需要外部振荡器，只需要最基本的电源，地，复位 3 个信号就可工作。

如果需要使用 2 线调试，只需将 GND, RST,TEST 3 个引脚与 MSP-EXP430G2 仿真器相连。

1.5 在 IAR 下新建工程

在安装 IAR 后，下面就手把手的教你如何新建一个工程。

1. 打开 IAR

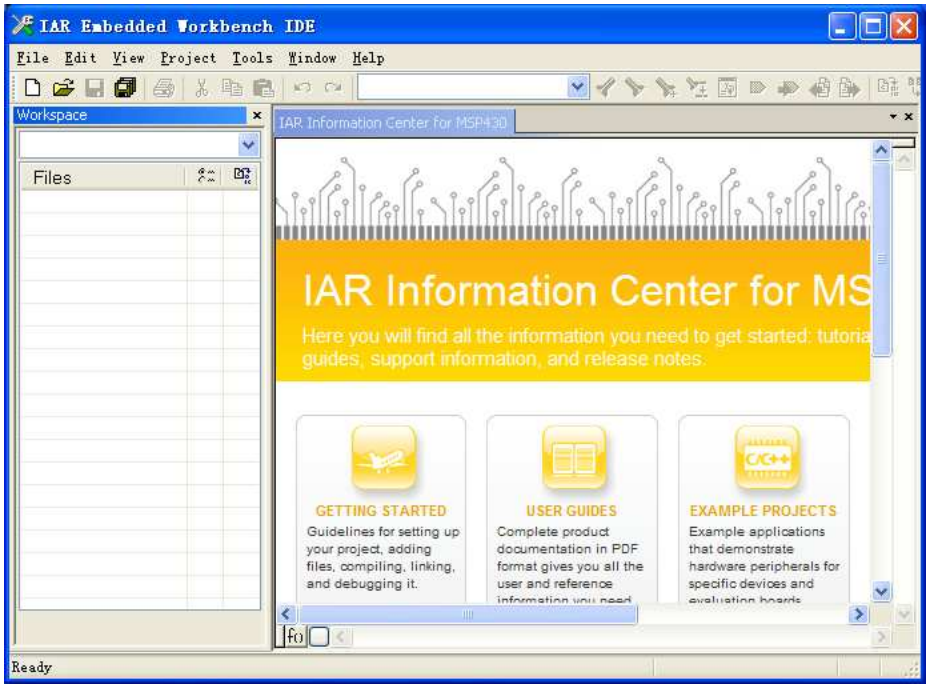


图 1.3 打开 IAR

2. 新建工程

选择菜单 Project/ Create New Project... ，弹出如下菜单，点击 OK 确认，

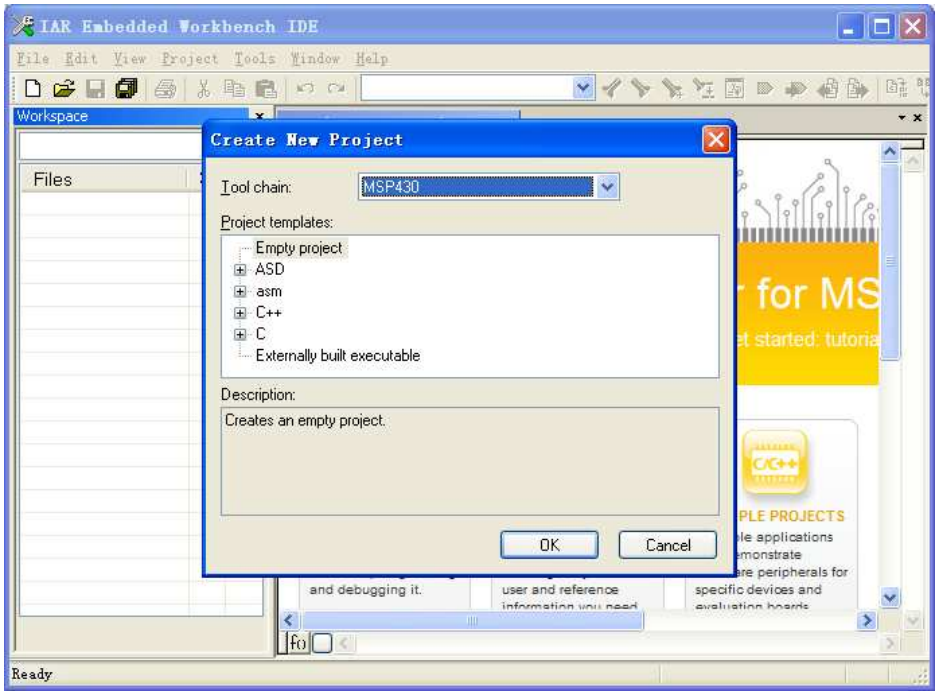


图 1.4 新建工程

3. 新建一个目录

在步骤 2 后，会弹出如下菜单，需要为你的新工程新建一个目录，然后填入工程名。

图 1.5 新建工程目录

4. 工程创建完成

工程创建完成后，回到项目工程界面，如图所示。

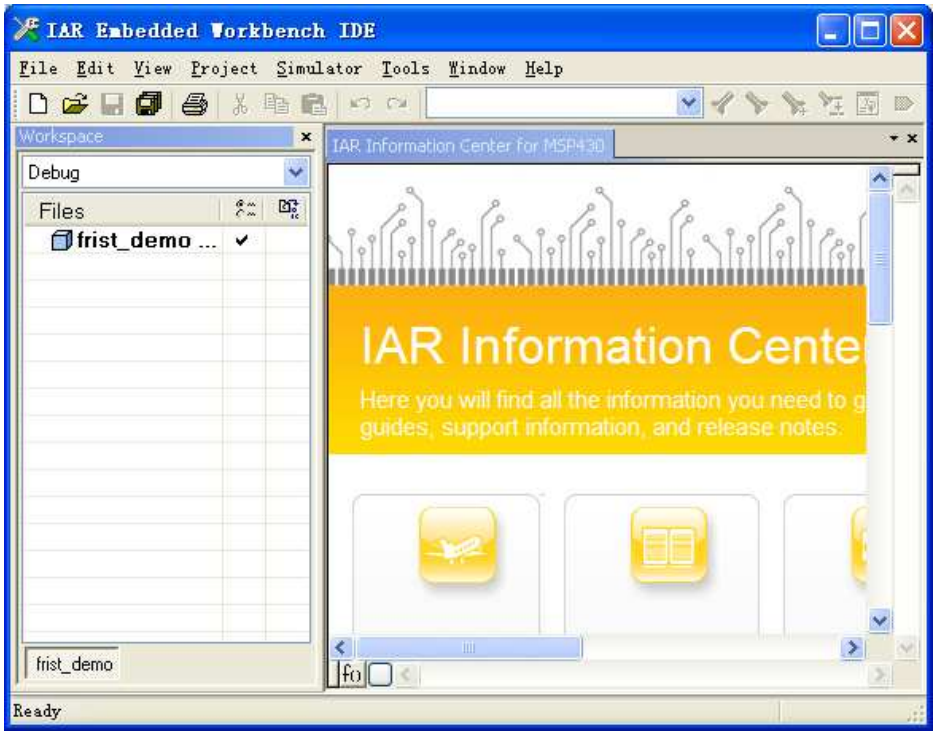
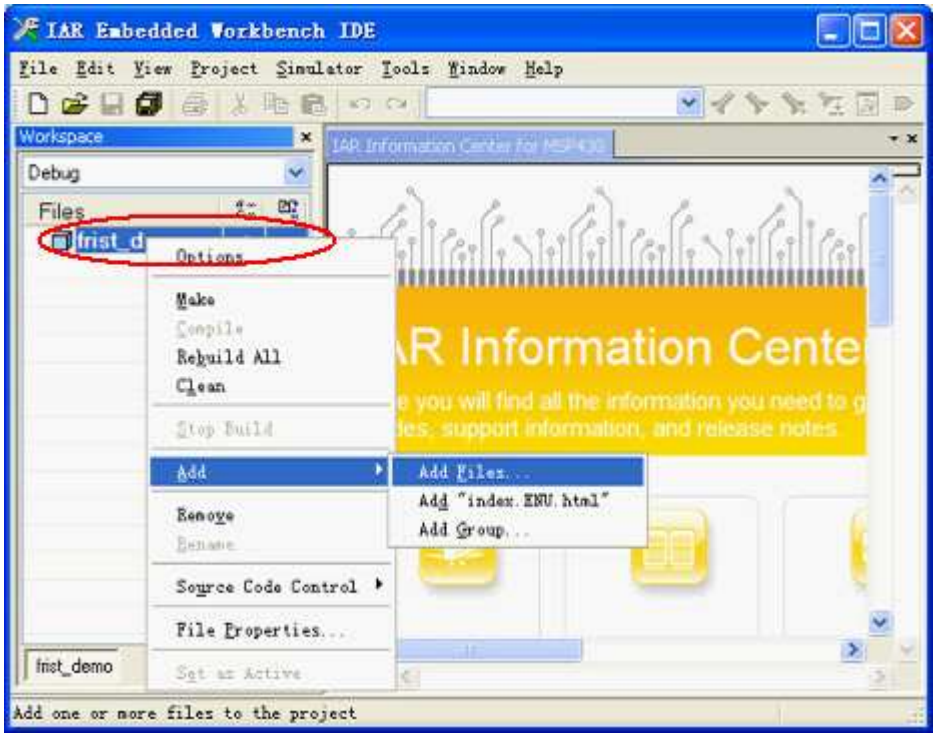


图 1.6 工程创建完成

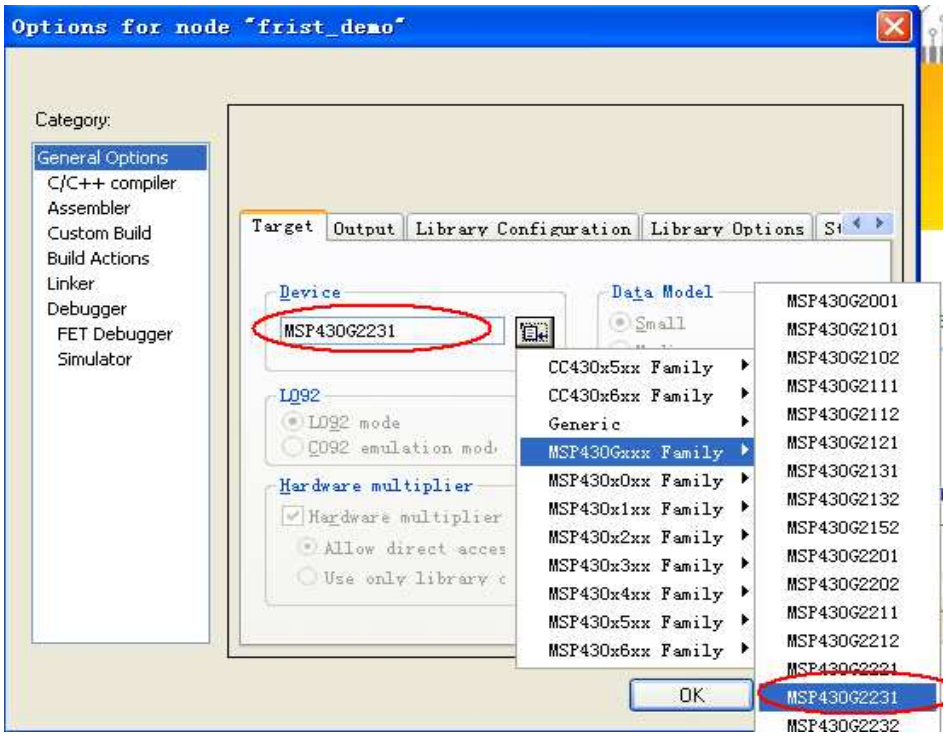
5. 添加 c 文件

如图所示，鼠标右键左上角的工程，选择菜单 Add/ Add File，然后把你自己的 C 文件添加进工程。



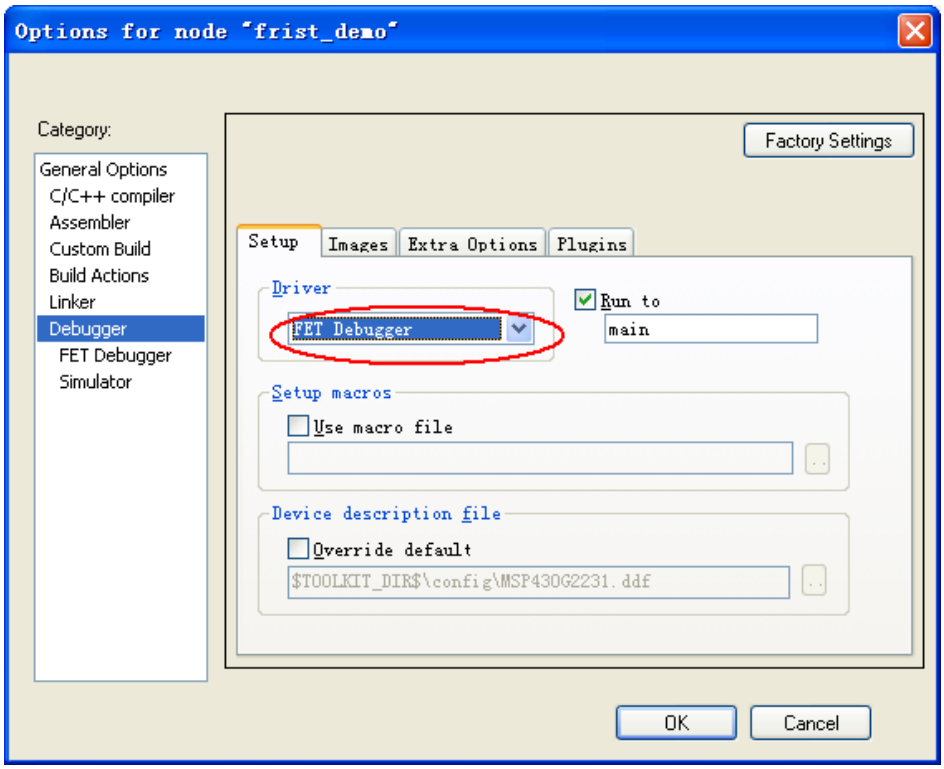
6. 配置工程---之选择芯片型号

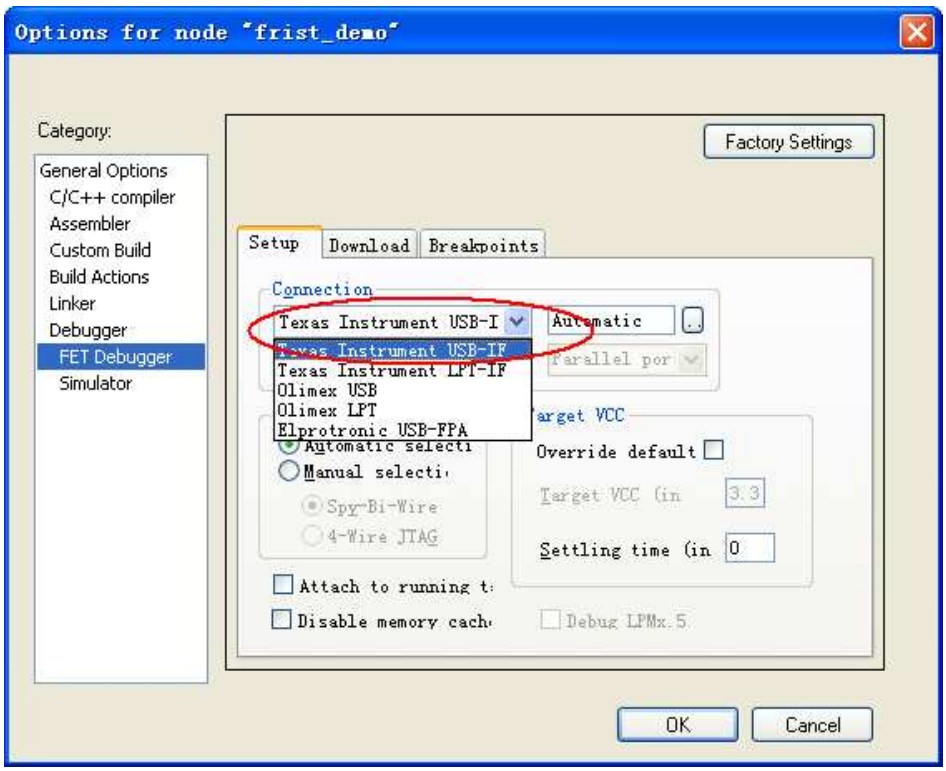
加入你自己的 C 文件后，需要配置工程选项。鼠标右键左上角的工程，选择菜单 Options...，在弹出的 Options 对话框中选择菜单 General Options。在 Device 选项中选择你芯片的型号。



7. 配置工程---之选择调试工具

工程配置 Options 内容很多，不过一般按默认值就可以了，只是需要根据你的使用调试工具，需要在 Debugger 选项中，选中你的调试工具型号。





1.6 运行你的第一个程序

在工程创建与设置完成之后，就可以开始运行你的第一个程序了。忙活了这么久，很值得期待吧？

程序清单 1.2 为我的一个测试程序 frist_demo，该程序用出厂校正值配置 DCO，配置 P1.0 与 P1.6 为 IO 输出（这两个 IO 有连接的两个 LED），然后在主循环里翻转 LED，两个 LED 一闪一闪。

程序清单 1.2---L1，加入 MSP430G2231 所需要的头文件，该头文件里包含了 MSP430G2231 的寄存器地址定义。

程序清单 1.2---L2，关闭看门狗，MSP430 上电复位后，默认看门狗开启。

程序清单 1.2---L3，L4，L5 使用出厂校正值配置 DCO 为 1MHZ。

程序清单 1.2---L6 配置两个 LED 引脚为输出功能。

程序清单 1.2---L8，LED 引脚置高电平。

程序清单 1.2---L9，延时 0.5 秒，void__delay_cycles(unsigned long cycles) 为 IAR 编译器的库函数（CCS 也有），该函数的功能是精确延时 n 个系统时钟。

程序清单 1.2---L10，LED 引脚置低电平。

```
#include   <msp430g2231.h>                                                    L1
#define       SYS_CLK            1000000
#define       MS                 (SYS_CLK/4/1000)

#define       LED_1              (1<<0)
#define       LED_2              (1<<6)
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;                  // Stop watchdog timer          L2

    //=============================================================================
    if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)                             L3
    {
        while(1);                                  // If calibration constants erased
                                                   // do not load, trap CPU!!
    }

    //1Mhz
    BCSCTL1 = CALBC1_1MHZ;                     // Set range                    L4
    DCOCTL = CALDCO_1MHZ;                      // Set DCO step + modulation */ L5
    //=============================================================================

    P1DIR = LED_1|LED_2;                       // LED 为输出                   L6
    P1OUT =   LED_1|LED_2;                     // 设置 LED 为 1                L7

    while(1)
    {
        P1OUT   =   LED_1|LED_2;               // LED == 1                     L8
```

```
        __delay_cycles(SYS_CLK/2);                                          L9

        P1OUT &= ~(LED_1|LED_2);            // LED == 0                     L10
        __delay_cycles(SYS_CLK/2);
    }
}
```

程序清单 1.2 frist_demo 程序