
CCS集成开发环境

朱雪秦

K103

实验与创新实践教育中心

CCS(Code Composer Studio): 一种集成开发环境 (IDE), 支持 TI 的微控制器和嵌入式处理器产品系列。具有环境配置、源文件编辑、程序调试、跟踪和分析等功能。帮助用户在一个软件环境下完成**编辑、编译、链接、调试和数据分析**等工作。

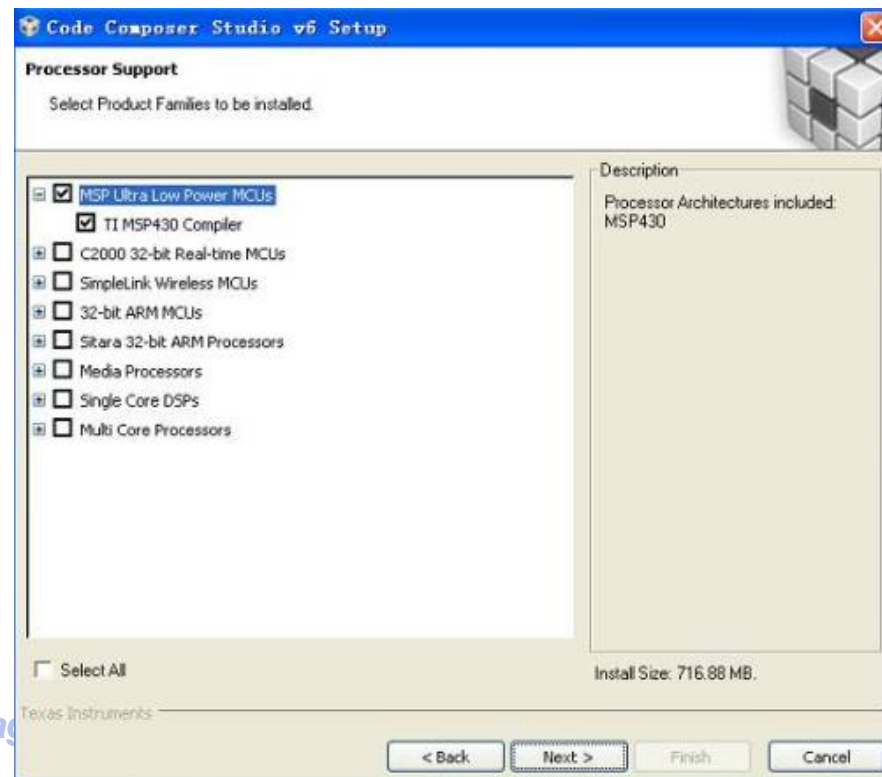
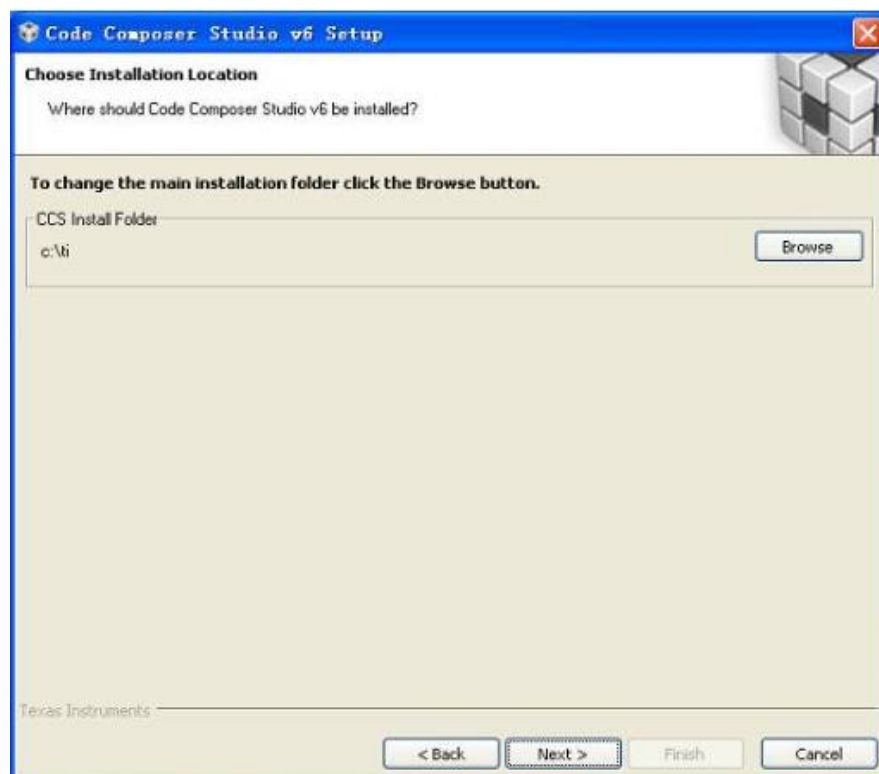
- 目前版本已更新到CCS10. x. x, 自己可以在官网上学着下载安装包, 选择Off-line Installers下的windows系统安装包。
- 注意: **所有名称 (计算机名和用户名) 和路径全部不包含中文字符。**

http://processors.wiki.ti.com/index.php/Download_CCS#Download_the_latest_CCS

-  1 **CCSv10.0的安装**.....●
-  2 **利用CCSv10.0导入已有工程**.....●
-  3 **利用CCSv10.0新建工程**.....●
-  4 **利用CCSv10.0调试工程**.....●
-  5 **CCSv10.0资源管理器应用**.....●
-  6 **CCS MSP430工程结构解析**.....●

CCS安装

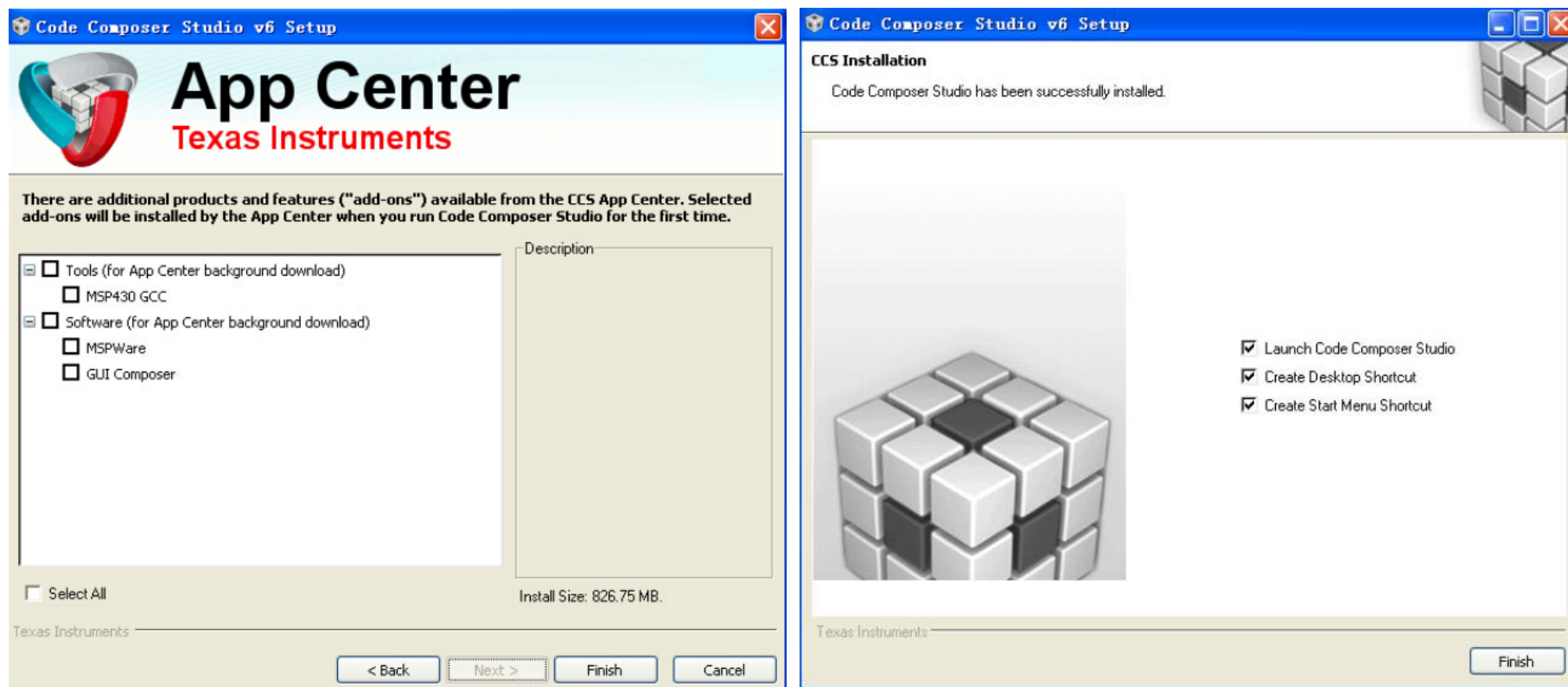
- (1) 运行安装程序 `ccs_setup_xxx.exe`，所有路径不能包含中文字符，只认英文字符，要特别注意。选择 CCS 安装路径，默认路径是 `c:\ti`，但如果 C 盘装有还原卡或是空间很小，请选择安装到其他硬盘分区。
- (2) 单击 **Next**，只选择支持 **MSP430 UltraLow Power MCUs** 的选项即可。单击 **Next**，继续安装。



CCS安装

(3) 在工具与软件安装选择页，不要勾选任何工具与软件。

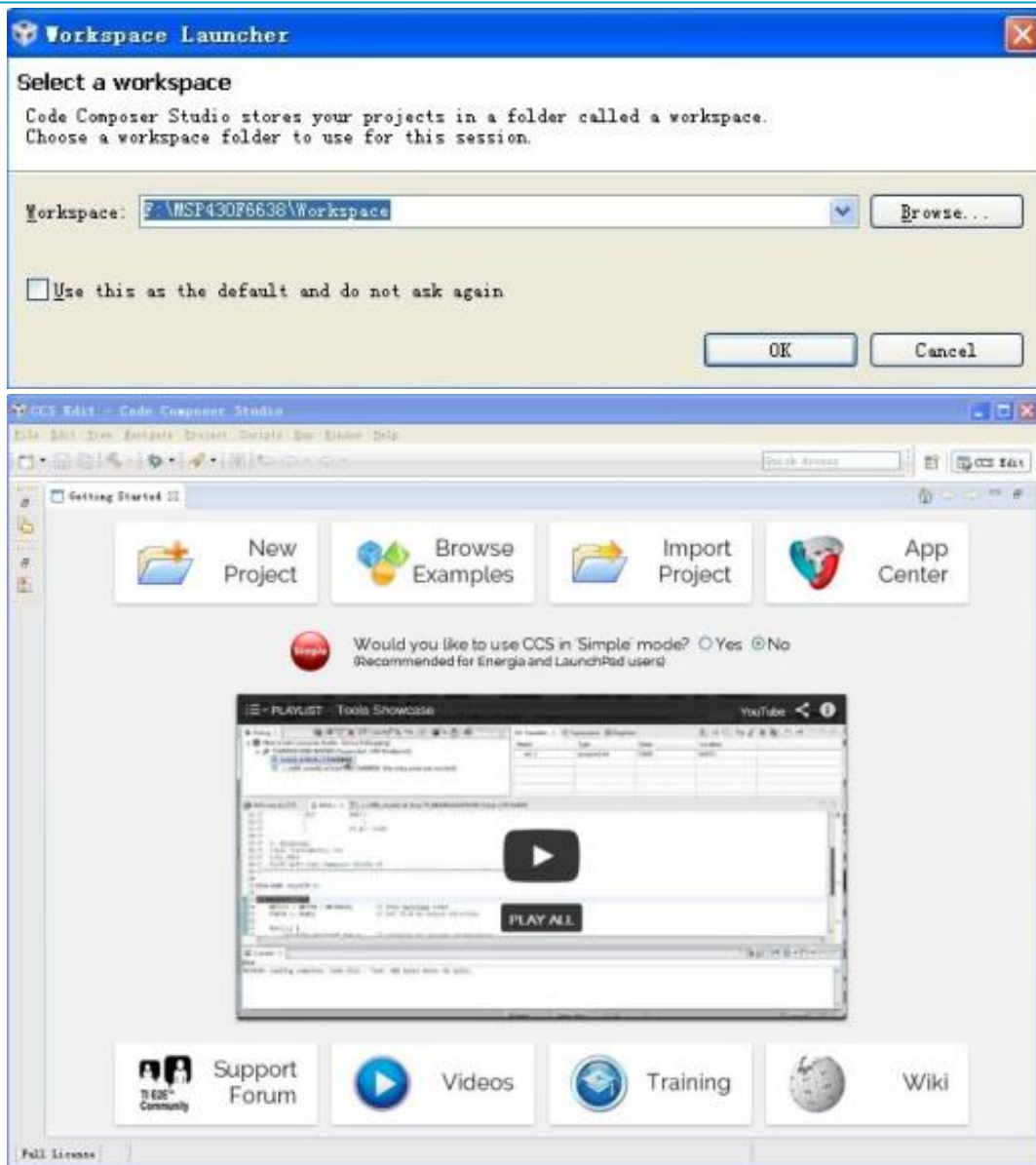
(4) 单击 **Finish**，将完成安装，同时运行 CCS。



CCS安装

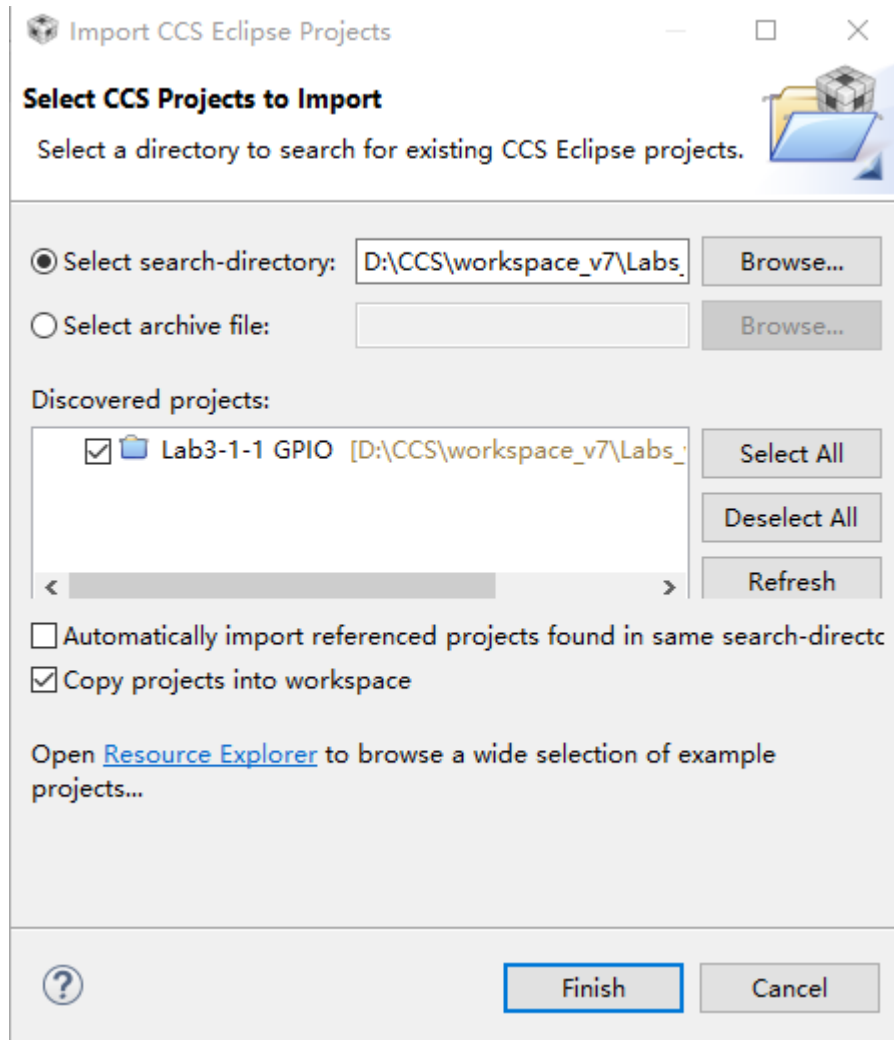
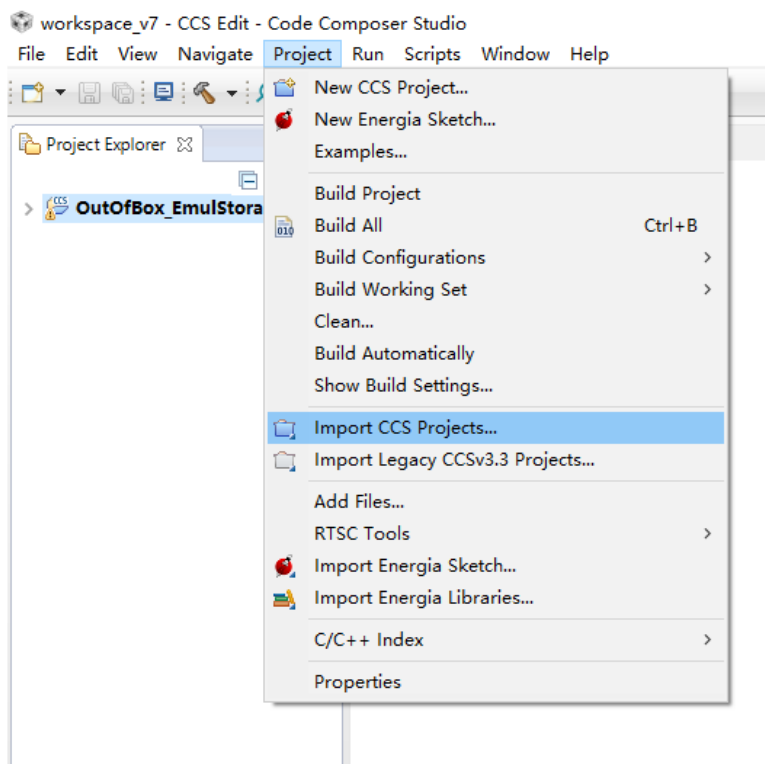
(5) 运行 CCS，弹出如右图所示窗口，我们可以在电脑中合适的区域建立工作空间（workspace），**最好不要**将“Use this as the default and do not ask again”选项勾选上，否则再更改workspace将会很不方便。

(6) 单击 OK，至此软件安装配置工作全部完成，如右图所示。



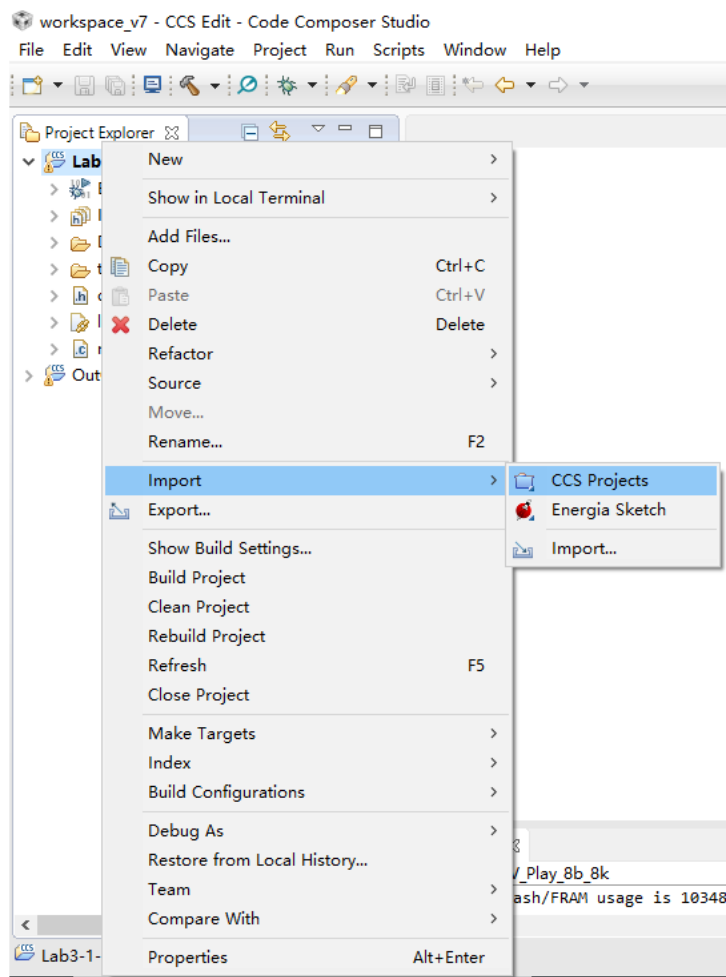
导入已有工程

(1) 打开 CCS 后，在 Project 菜单栏中选择 Import CCS Project，打开如下界面，按图示勾选相应选项。

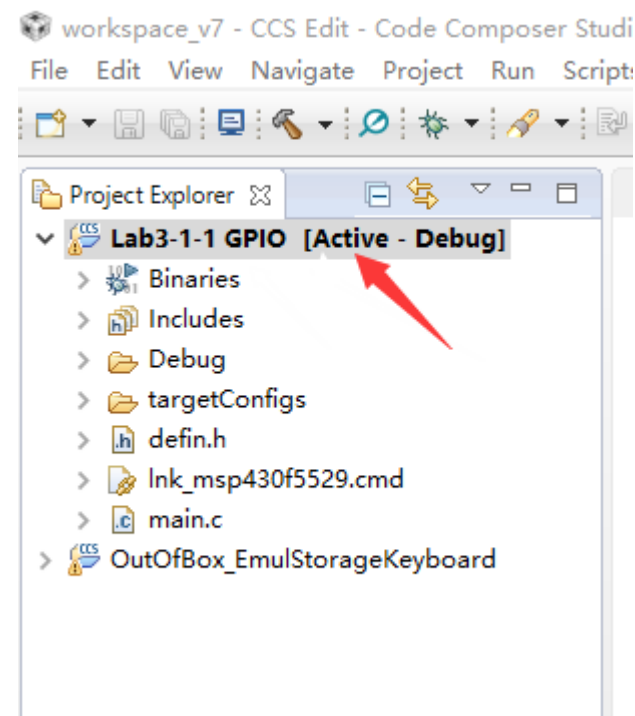


导入已有工程

(2) 另一种导入方法，即在工程窗口点击右键，选择如图

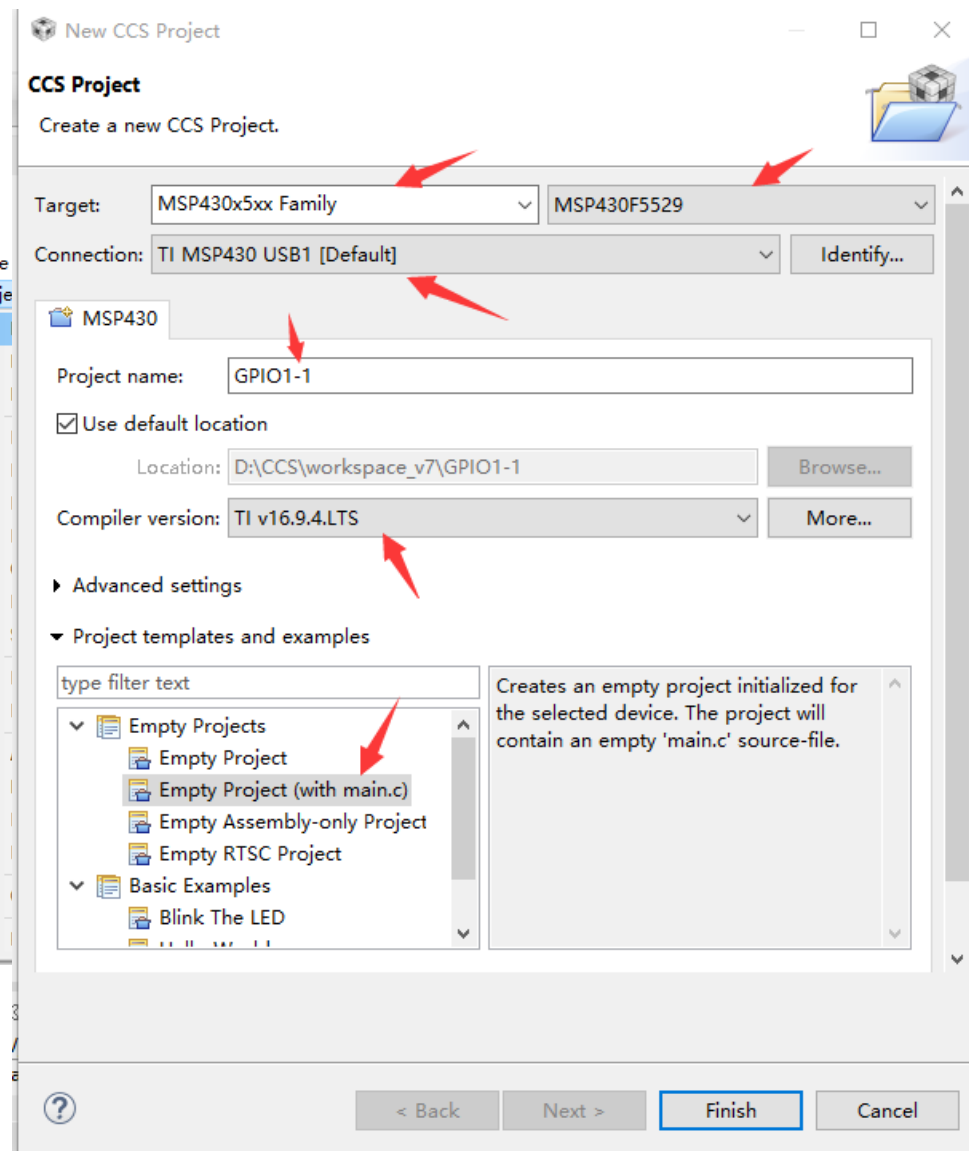
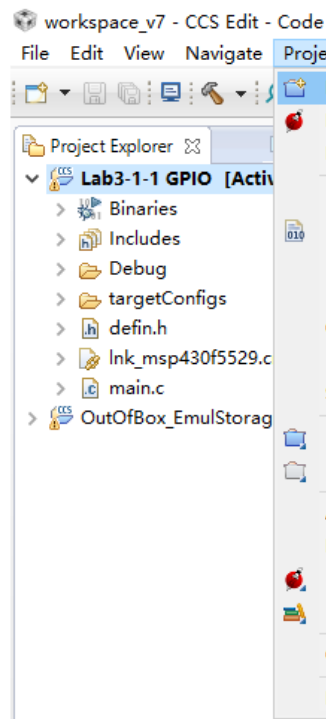
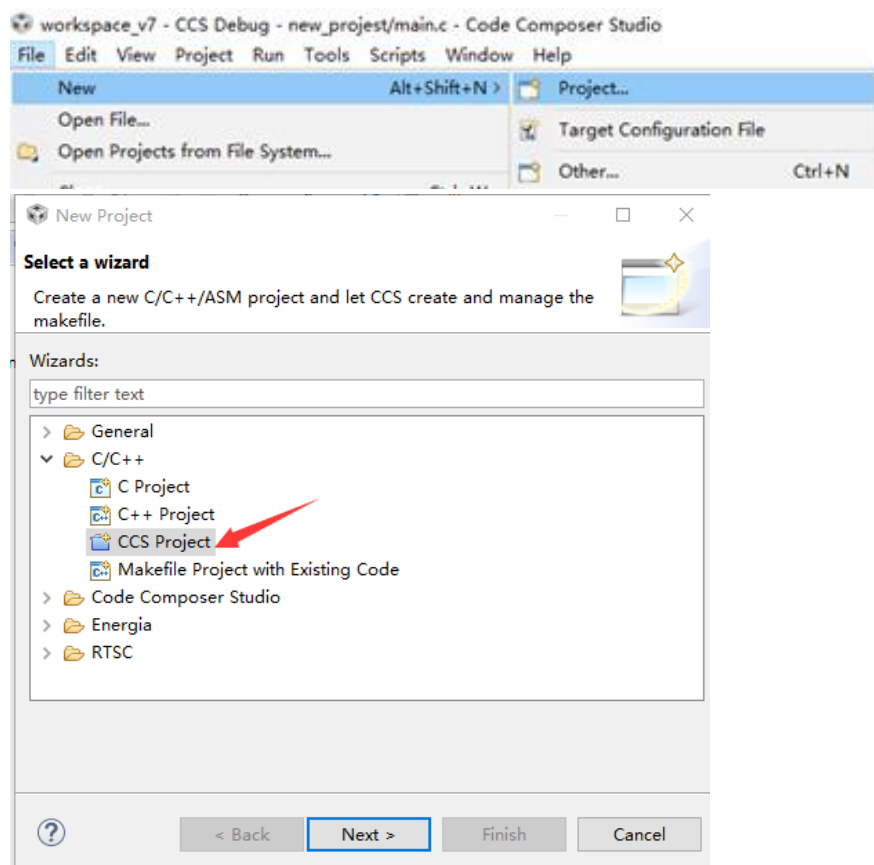


(3) 导入后，工程界面即出现导入的工程。

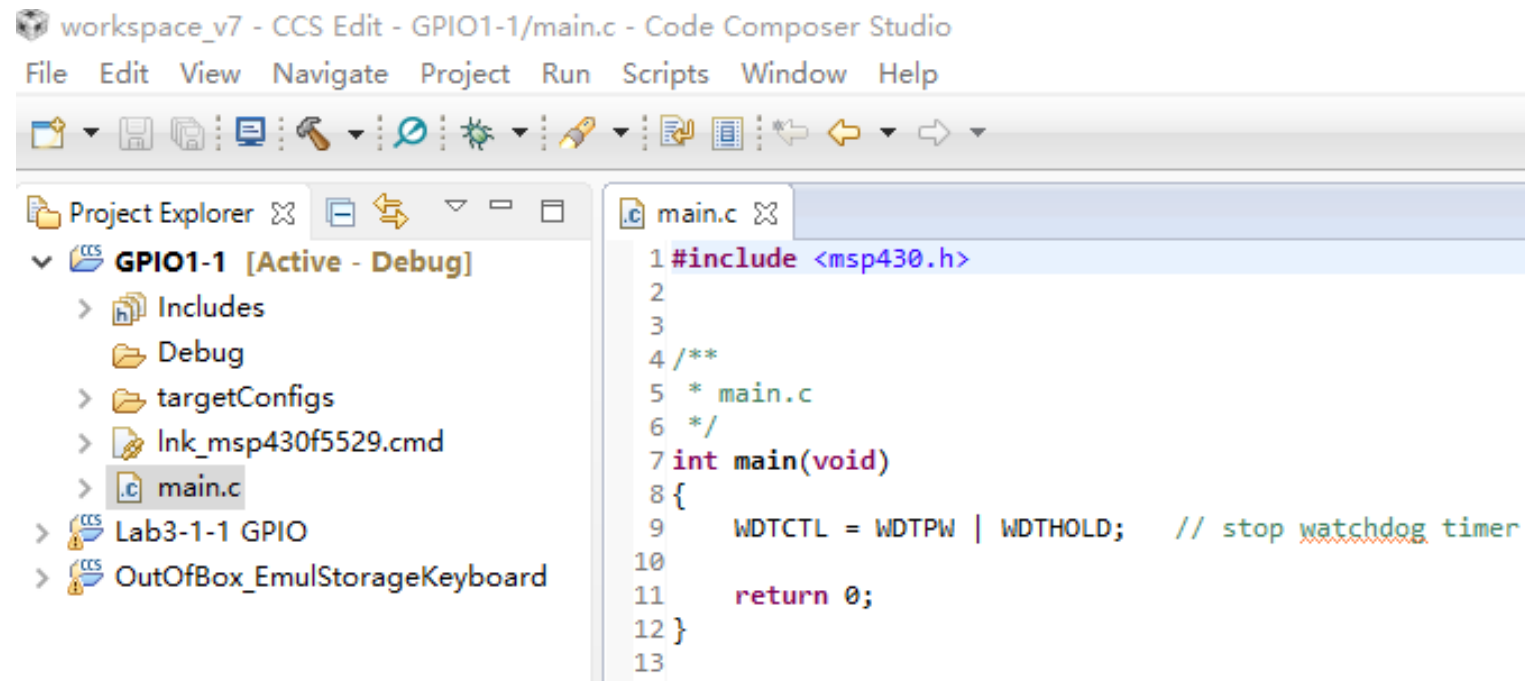


新建工程

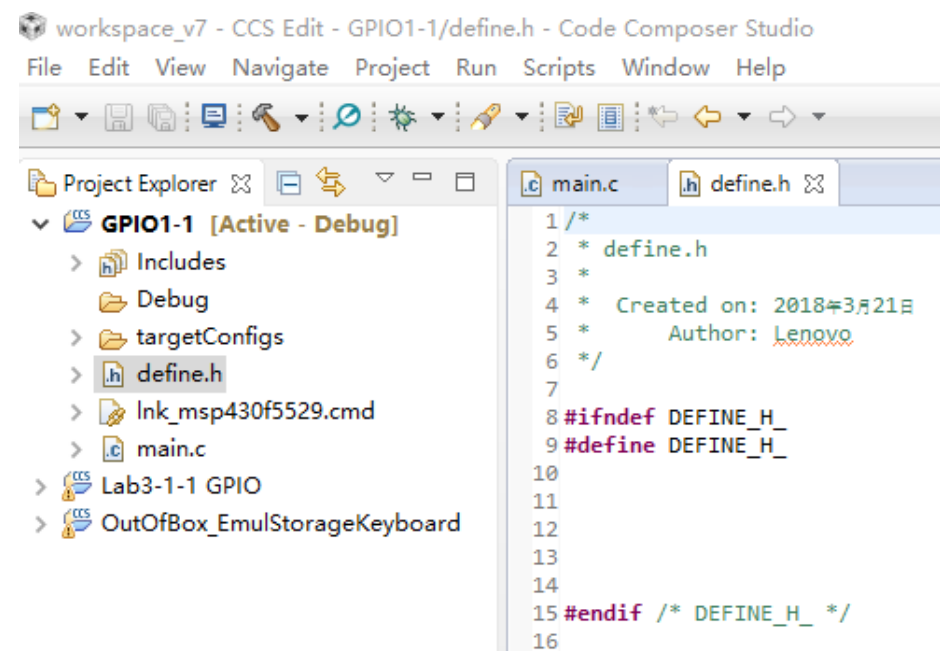
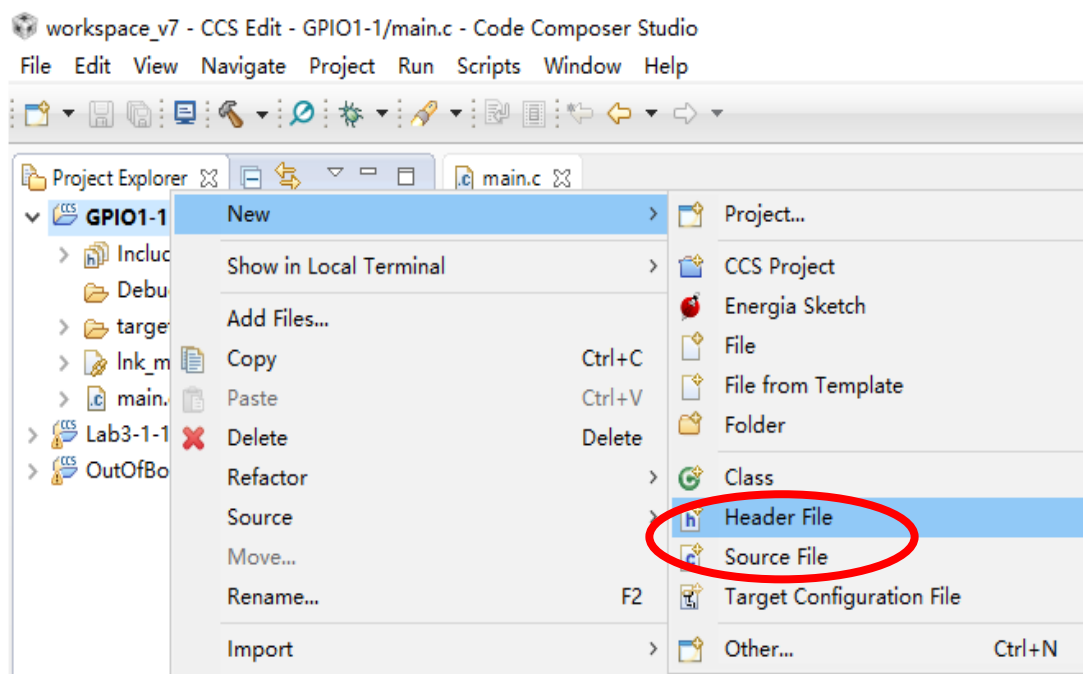
(1) 打开 CCS 后，在 File 菜单栏中选择 New CCS Project，打开界面，按图示选择相应选项。



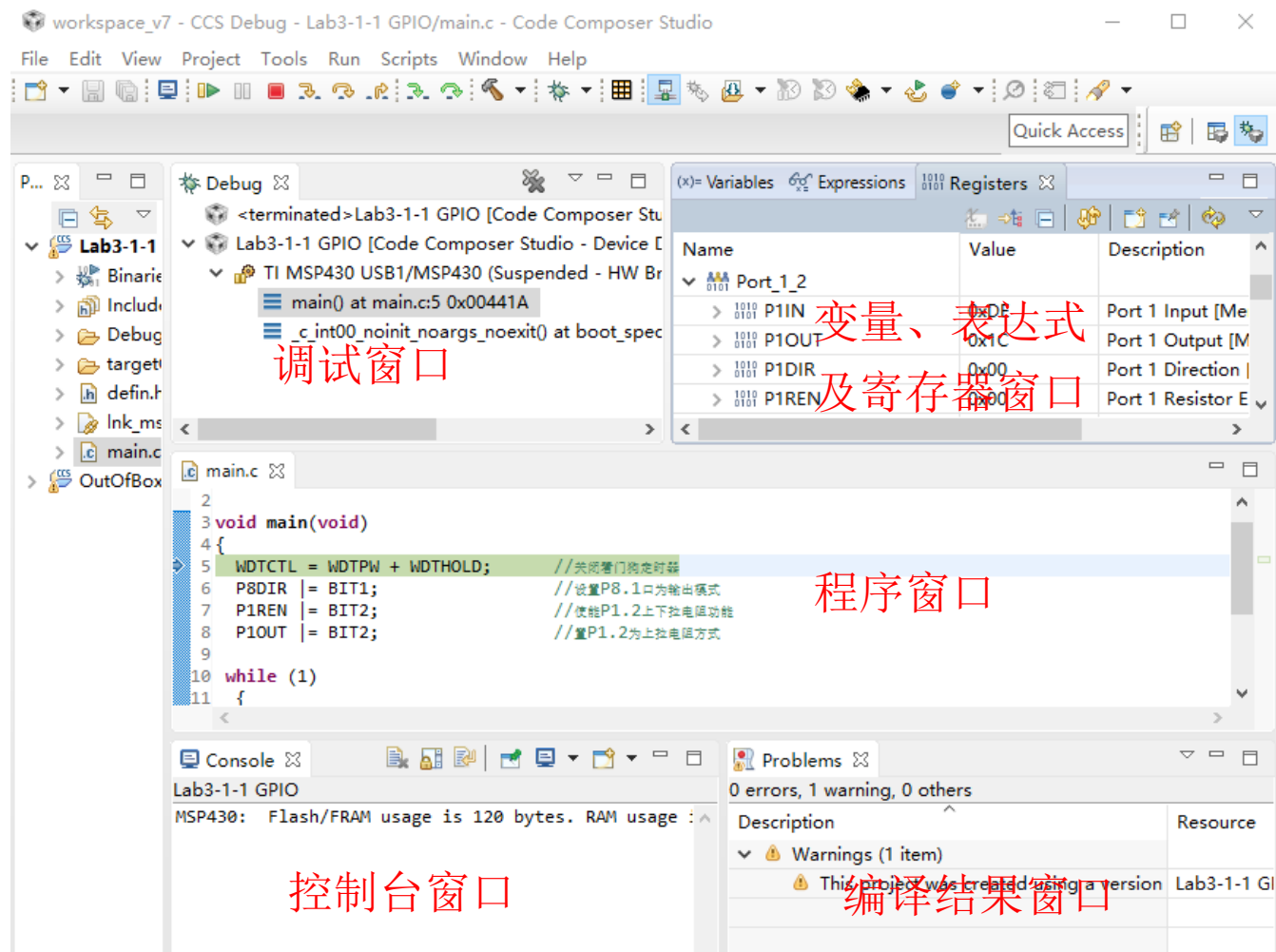
(2) 点击finish后，工程界面即出现新建的工程GPIO1-1，包含main.c文件，打开后，出现如下图所示编程界面。









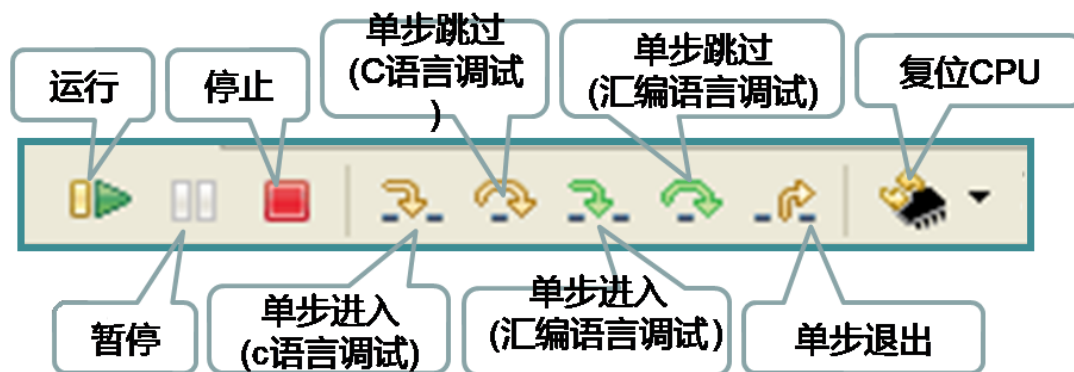
(3) 新建*.h头文件/*.c源文件，在工程界面选择工程名称，点击右键，选择新建头文件/源文件，命名后点开编辑即可。



- (1) 对已有工程，选择  **Project-->Build Project**，将工程进行编译通过。
- (2) 单击绿色的 **bug** 按钮  进行下载调试，得到下图所示的界面。

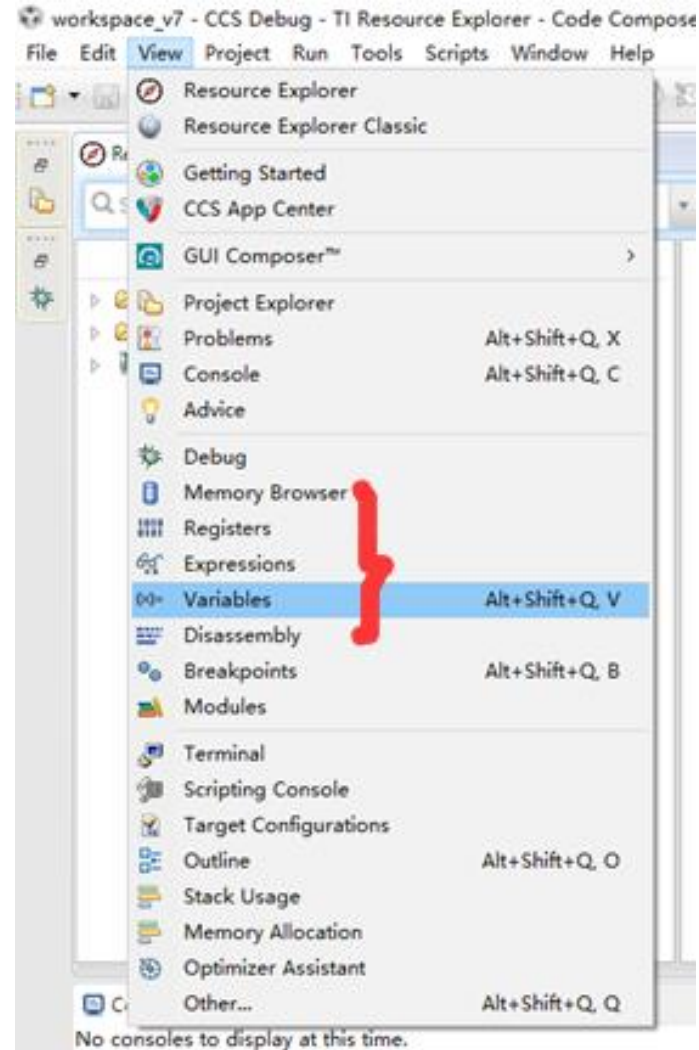


(3) 单击运行图标  运行程序，观察显示的结果。在程序调试的过程中，可通过设置断点来调试程序：选择需要设置断点的位置，右击鼠标选择 Breakpoints→Breakpoint，断点设置成功后将显示图标 ，可以通过双击该图标来取消该断点。程序运行的过程中可以通过单步调试按钮  配合断点单步的调试程序，单击重新开始图标  定位到 main()函数，单击复位按钮  复位。可通过中止按钮  返回到编辑界面。

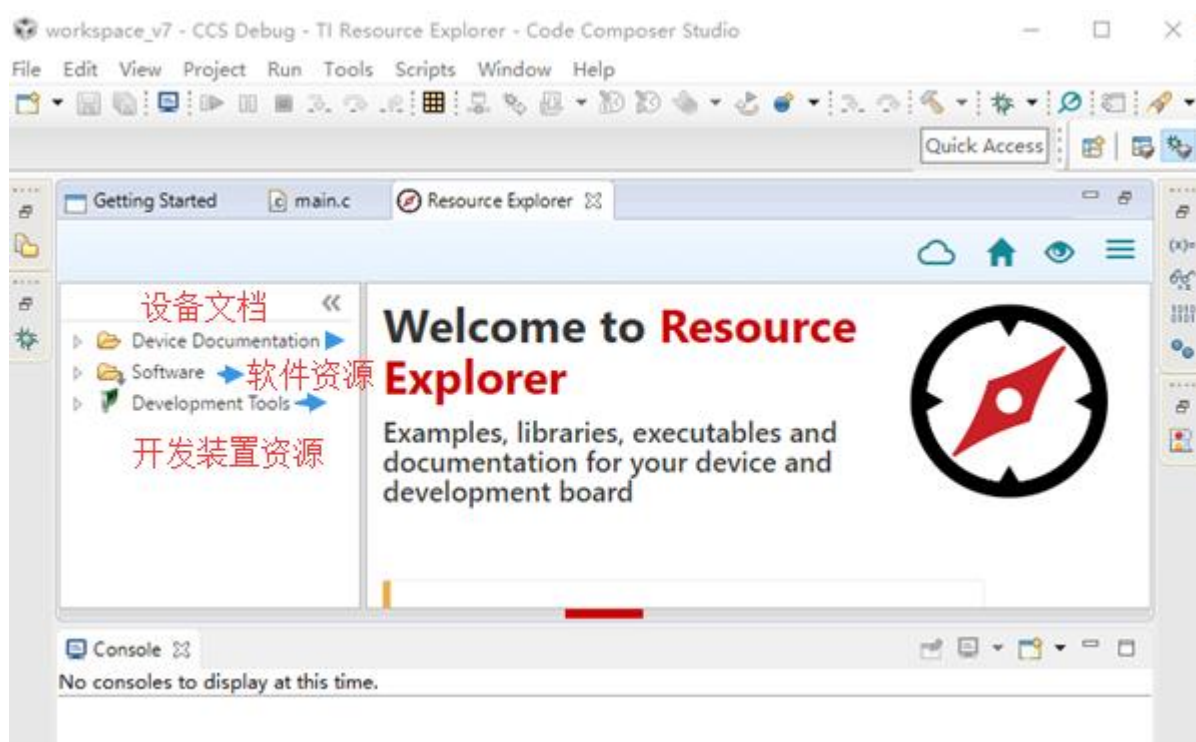


(4) 在单步调试时，根据需要可查看以下项目的数据。

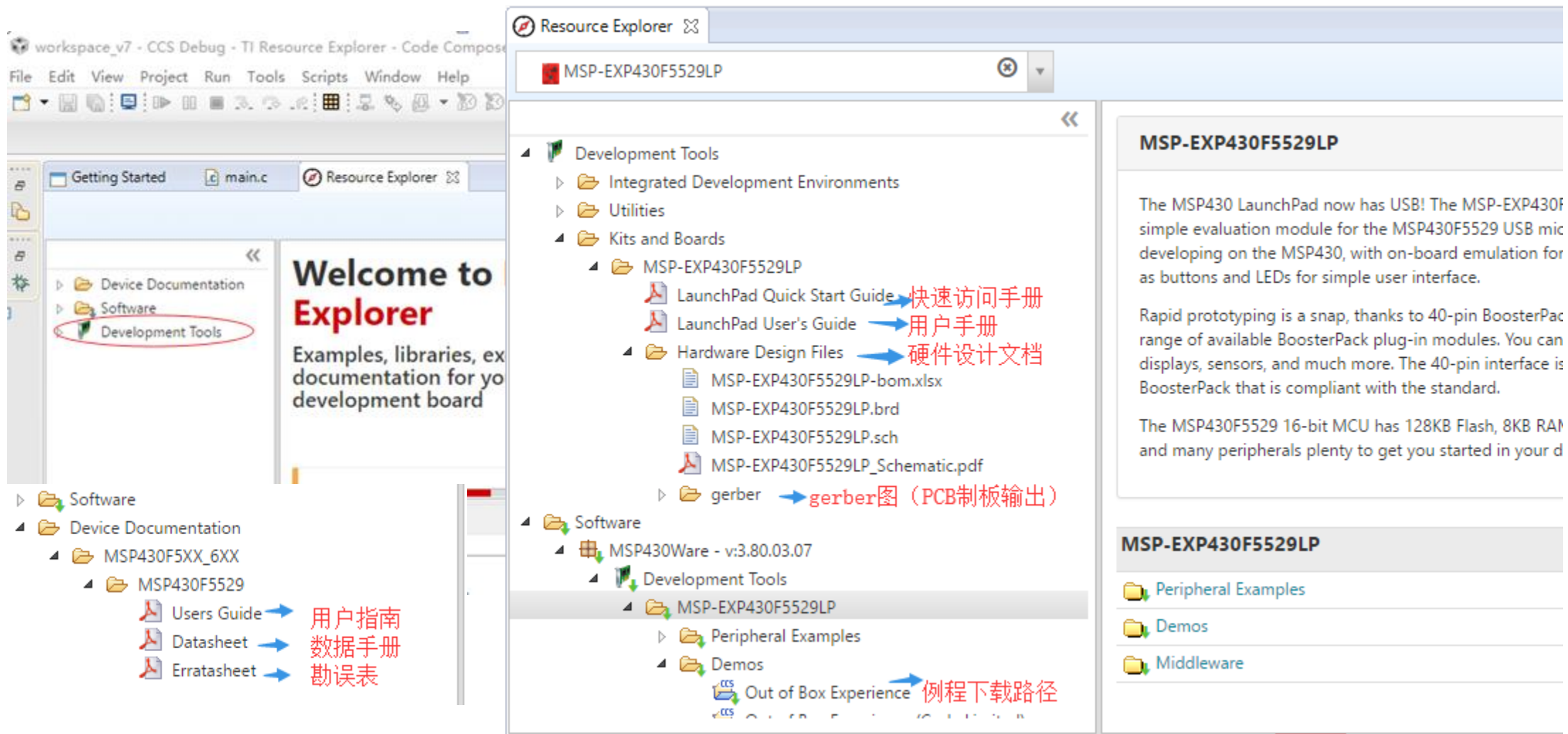
变量 (Variables)
寄存器 (Registers)
表达式 (Expressions)
内存 (Memory)
汇编程序观察窗口
(Disassembly)



(1) CCSv10.0具有很强大的功能, 并且其内部的资源也非常丰富, 利用其内部资源进行MSP430单片机开发, 将会非常方便。现在演示CCSv10.0资源管理器的应用。如图所示, 通过View-->Resource Explorer打开CCSv10.0的资源管理器。



(2) 展开MSP430的资源，得到下图所示的界面，其中包含F5/6系列的用户指导、数据手册、勘误表以及示例代码，装置开发资源中的硬件设计原理图和gerber图全部开源。



(3) CCS资源超级丰富，最新的装置开发资源都在Development Tools里面，我们实验用的例程就在里面。

The screenshot displays the TI Resource Explorer interface. On the left, a tree view shows the 'MSP430Ware - v:3.80.03.07' folder expanded, revealing 'Development Tools' and various example folders like 'MSP-EXP430FR2433', 'eZ430-F2013', 'eZ430-RF2500', 'MSP-EXP430G2', 'MSP-EXP430F5438', 'MSP-EXP430F5529LP', 'Peripheral Examples', 'Demos', and 'Middleware'. Red text labels '外设例程' (Peripheral Examples), '例程' (Examples), and '复杂外设例程' (Complex Peripheral Examples) are placed next to their respective folders. The main pane shows a list of development tools, including 'ELPROTRONIC FlashPro-43', 'EVM430-FR6989', 'GUI Composer', 'Hi-Lo Systems', 'IAR Embedded Workbench', 'LP-MSP430FR2476', 'MSP Flasher', 'MSP GDB', 'MSP-CAPT-FR2633', 'MSP-EXP430F5438', 'MSP-EXP430F5529', and 'MSP-EXP430F5529LP'. An 'Import' button is circled in the top right. A red text overlay states '首次下载需要安装430Ware程序' (First download requires installing 430Ware program). A dialog box titled 'Cannot import project' is shown, stating 'The package needs to be installed locally to import projects. Would you like to install now?' with 'Install' and 'Cancel' buttons. At the bottom, a progress bar indicates 'Installing MSP430Ware version 3.80.09.03 - Downloading...' with 'Downloaded 194.0 MB of 1.1 GB'.

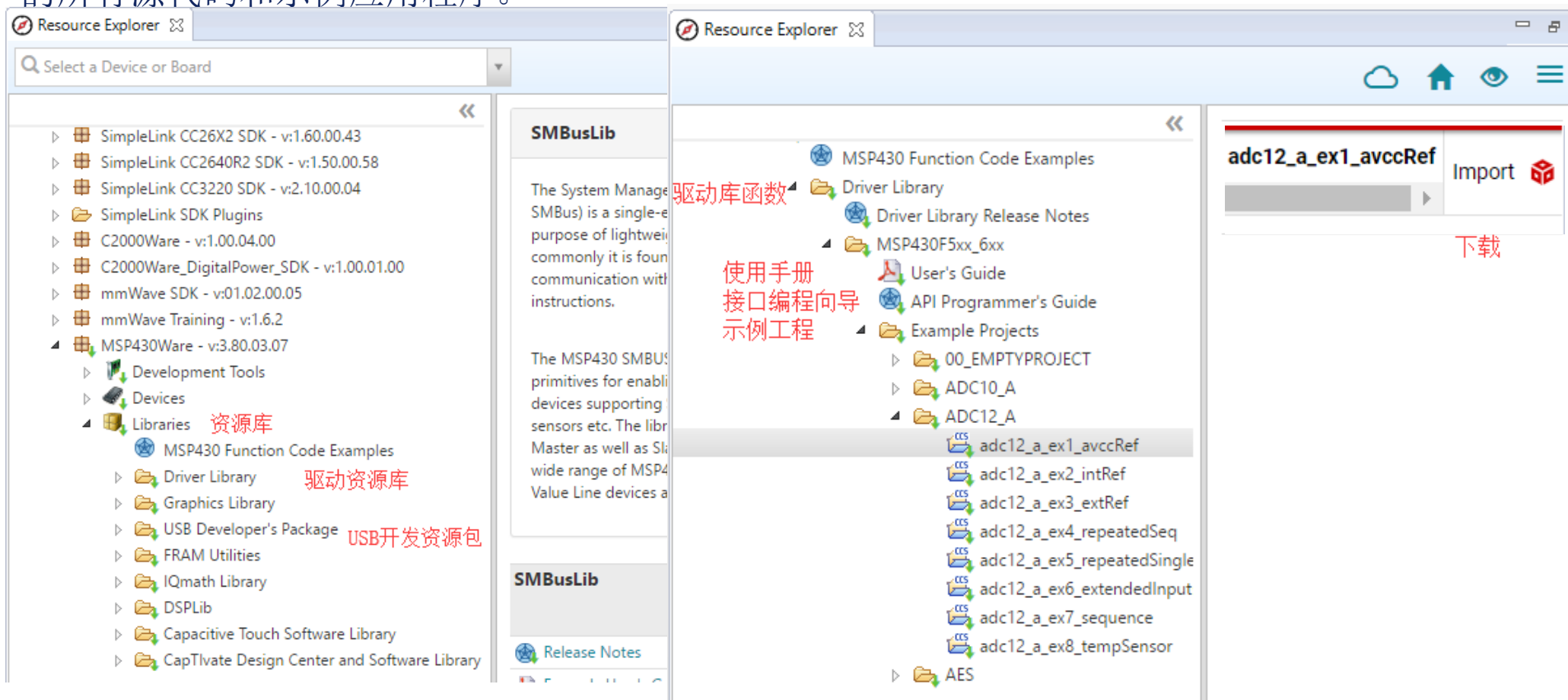
外设例程
例程
复杂外设例程

首次下载需要安装430Ware程序

Cannot import project
The package needs to be installed locally to import projects. Would you like to install now?
Install Cancel

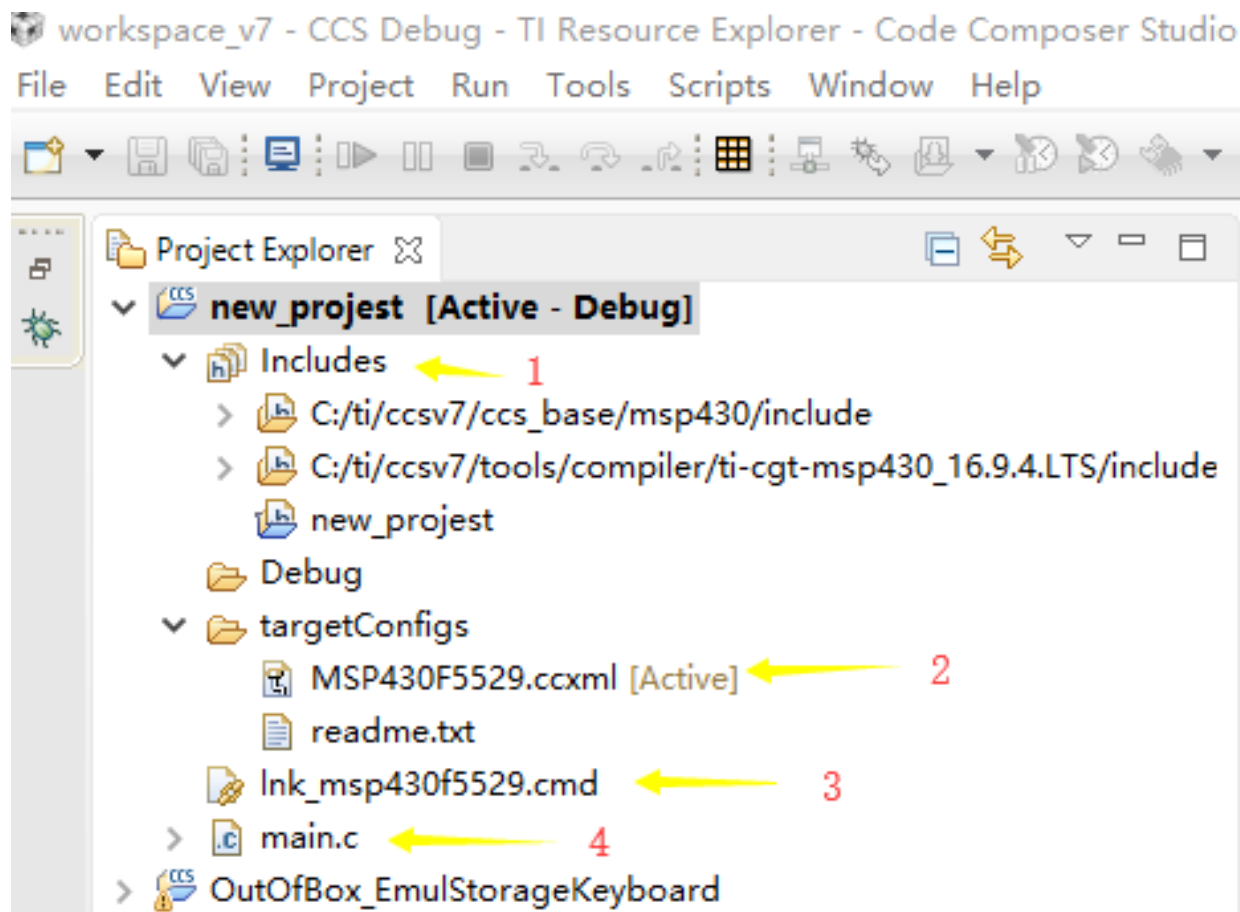
Installing MSP430Ware version 3.80.09.03 - Downloading...
Downloaded 194.0 MB of 1.1 GB

(4) 展开Libraries资源库，得到如下图所示的界面，其中包含MSP430驱动程序库以及USB的开发资源包。“MSP430驱动程序库”为全新高级API，这种新型驱动程序库能够使用户更容易地对MSP430硬件进行开发；MSP430USB开发资源包包含了开发一个基于USB的MSP430项目所需的所有源代码和示例应用程序。



CCS MSP430工程结构解析

前面对如何在**CCS**中新建一个工程做了详细的介绍，这里就一个完整的**MSP430**工程中包含的文件的作用做简单的介绍和说明。如图所示，从**CCS**窗口左侧的**Explorer**导航栏中观察工程，发现工程中的文件分为4种，为



1. Includes

2. ccxml配置文件

3. cmd配置文件

4. 源文件

◆ includes

在该目录下包含了用户设置的头文件路径下的所有头文件，如图所示为**CCS**默认的两个头文件路径。

(1) **MSP430**的头文件：

提供了不同型号的**MSP430**的头文件定义，包括寄存器定义，常用位定义等，与编译平台相关，不同的编译软件提供的头文件可能略有不同，在做平台间的移植的时候，注意要同时考虑到头文件间的差异；

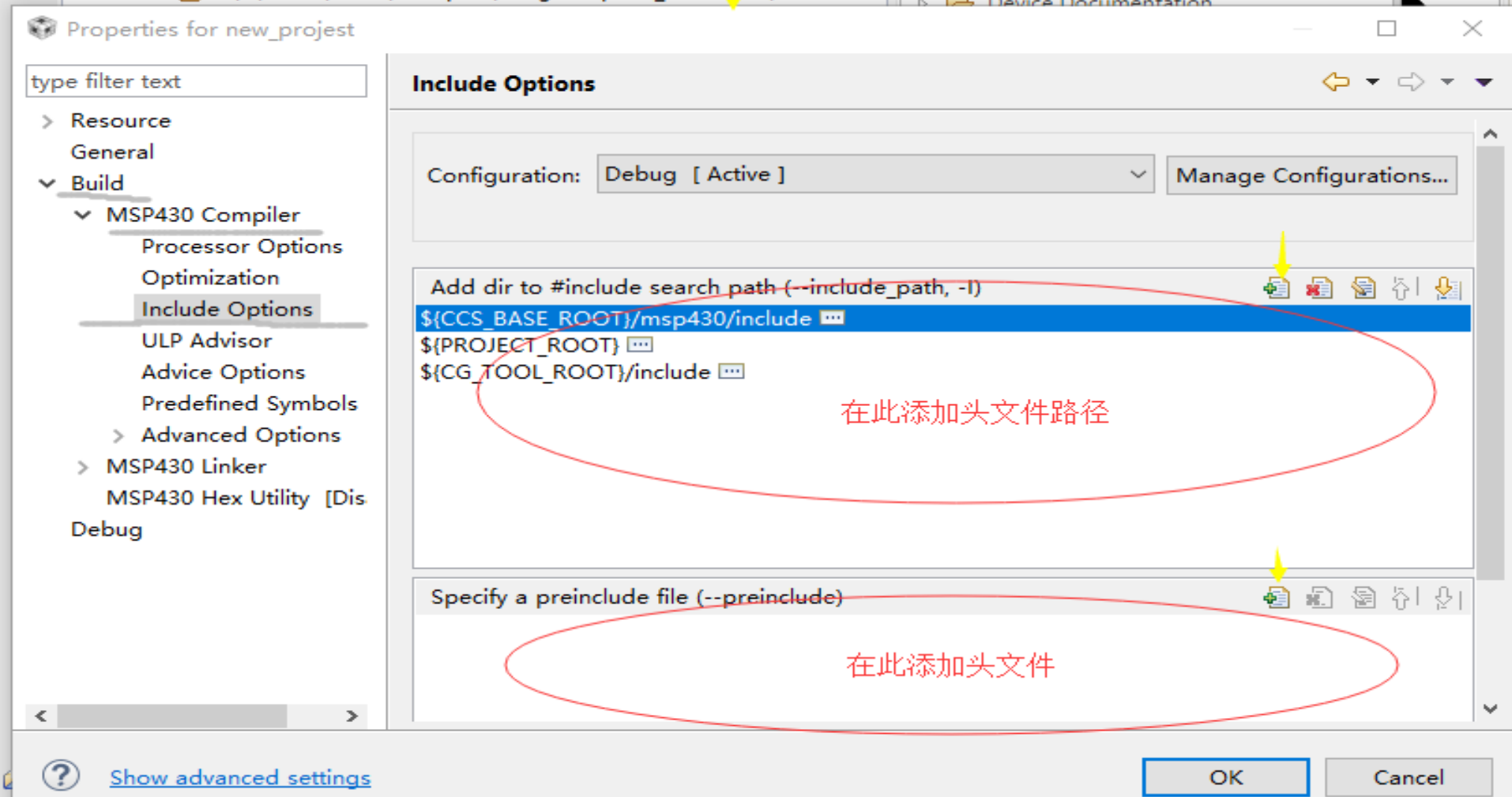
(2) **C**语言相关的头文件：

所有编译平台都具有的**C**的标准文件，如：**stdio.h**。

如何在工程中添加自定义的头文件呢？

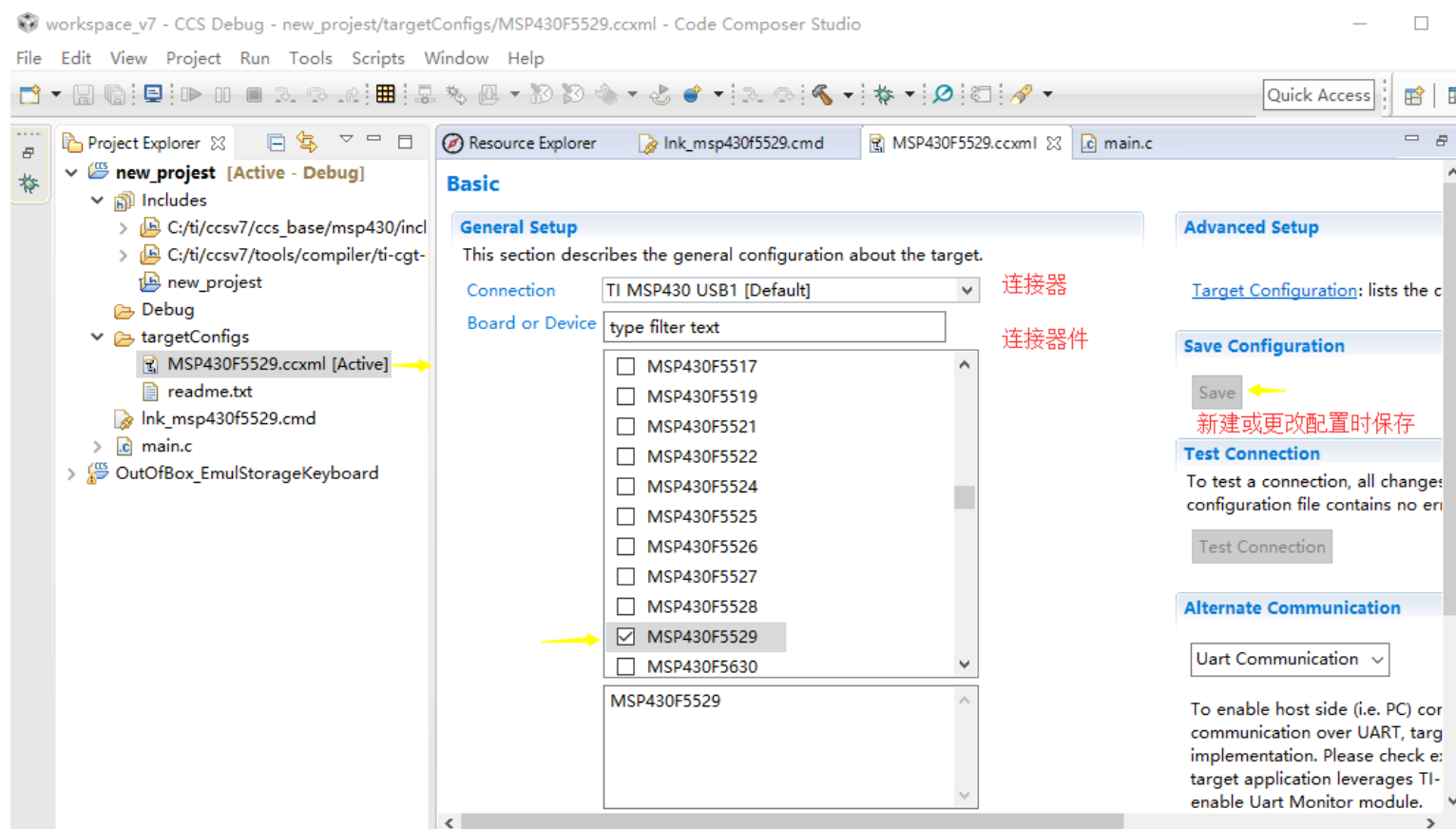


右键点击properties出现对话框



◆ ccxml配置文件

目标板配置文件，实现了对链接目标的定义和设置。一般会在创建工程时自动新建，且进行配置，包括**连接器**选择，**连接器件**的选择等，本文件也可以自己创建。



◆ cmd配置文件

默认cmd配置文件，主要用来分配430内部的FLASH和RAM空间，在link过程中告诉链接器怎样进行地址的计算和空间的分配。

文件开始，**MEMORY**段，对选择型号的芯片的存储单元映射进行定义，器件相关，不同型号的器件**FLASH**，**RAM**的大小及映射关系都不尽相同，所以对于不同型号的430会分别有不同的cmd文件，用户不会对该部分的内容进行修改操作。

SECTIONS的定义，主要是对程序的内容进行地址空间的分配。每个代码程序中都会包含有不同的段（**section**），默认对每个段的定义格式都以“.”开头，编译器对段的名称和定义有规定值。如下图的一个实例程序所示：定义的全局变量会储存在**.bss**段，在程序中初始化的值会存储在**.cinit**段，**.stack**段中则为程序中定义的局部变量，而书写的指令代码则会存储在**.text**段中。

以**MSP430G2553**为例子，展示一下文件。

CCS MSP430工程结构解析

```
/* **** */
/*  SPECIFY THE SYSTEM MEMORY MAP  */
/* **** */
```

MEMORY

```
{
    SFR                : origin = 0x0000, length = 0x0010
    PERIPHERALS_8BIT   : origin = 0x0010, length = 0x00F0
    PERIPHERALS_16BIT  : origin = 0x0100, length = 0x0100
    RAM                 : origin = 0x0200, length = 0x0200
    INFOA               : origin = 0x10C0, length = 0x0040
    INFOB               : origin = 0x1080, length = 0x0040
    INFOC               : origin = 0x1040, length = 0x0040
    INFOD               : origin = 0x1000, length = 0x0040
    FLASH               : origin = 0xC000, length = 0x3FE0
    INT00               : origin = 0xFFE0, length = 0x0002
    INT01               : origin = 0xFFE2, length = 0x0002
    .....
    .....
    INT13               : origin = 0xFFFA, length = 0x0002
    INT14               : origin = 0xFFFC, length = 0x0002
    RESET               : origin = 0xFFFE, length = 0x0002
}
```

CCS MSP430工程结构解析

```
/* **** */
/* SPECIFY THE SECTIONS ALLOCATION INTO MEMORY */
/* **** */
```

SECTIONS

```
{
    .bss      : {} > RAM          /* GLOBAL & STATIC VARS */
    .system   : {} > FLASH        /* DYNAMIC MEMORY ALLOCATION AREA */
    .stack    : {} > RAM (HIGH)   /* SOFTWARE SYSTEM STACK */

    .text     : {} > FLASH        /* CODE */
    .cinit     : {} > FLASH        /* INITIALIZATION TABLES */
    .const     : {} > FLASH        /* CONSTANT DATA */
    .cio       : {} > FLASH        /* C I/O BUFFER */

    .pinit     : {} > FLASH        /* C++ CONSTRUCTOR TABLES */

    .infoA     : {} > INFOA        /* MSP430 INFO FLASH MEMORY SEGMENTS */
    .infoB     : {} > INFOB
    .infoC     : {} > INFOC
    .infoD     : {} > INFOD

    .int00     : {} > INT00        /* MSP430 INTERRUPT VECTORS */
    .int01     : {} > INT01
    .....
    .....
    .int13     : {} > INT13
    .int14     : {} > INT14
    .reset     : {} > RESET        /* MSP430 RESET VECTOR */
}
```

◆ 源文件

可以在工程名上右击选择 “**add files...**” 向工程中添加文件，或者 “**New**” 新建文件，包括源文件。源文件的类型可以是**c**文件也可以是汇编文件。单击文件前的三角下拉菜单可以看到该文件中包含的头文件，全局变量和函数。

MSP430 C语言编程

运算符

操作符	说明
&&	逻辑与
	逻辑或
!	逻辑非
&	按位与
	按位或
^	按位异或
~	按位取反
>>	右移
<<	左移

P1OUT |= BIT0;
P1OUT &= ~BIT0;
P1OUT ^= BIT0;
P1IN & BIT0

```
#define BIT0 (0x0001)
#define BIT1 (0x0002)
#define BIT2 (0x0004)
#define BIT3 (0x0008)
#define BIT4 (0x0010)
#define BIT5 (0x0020)
#define BIT6 (0x0040)
#define BIT7 (0x0080)
```

执行次数	PxOUT	BIT0
0	0100 1110	0000 0001
1	0100 1111	0000 0001
2	0100 1110	0000 0001

与 (&)	0 & 0 = 0	1 & 0 = 0	0 & 1 = 0	1 & 1 = 1
或 ()	0 0 = 0	1 0 = 1	0 1 = 1	1 1 = 1
异或 (^)	0 ^ 0 = 0	1 ^ 0 = 1	0 ^ 1 = 1	1 ^ 1 = 0

常用的系统函数

```
Light
├── Binaries
└── Includes
    ├── D:/ti/ccs1000/ccs/ccs_base/msp430/include
    └── D:/ti/ccs1000/ccs/tools/compiler/ti-cgt-msp430 20.2.0.LTS/include
        ├── libcxx
        ├── machine
        ├── sys
        ├── xlocale
        └── _atomic.h
```

```
float.h
intrinsics_legacy_undefs.h
intrinsics.h
inttypes.h
iso646.h
```

```
#define _nop()          __no_operation()    //空操作
#define __no_operation()  __no_operation()

#define _enable_interrupt()  __enable_interrupt()  //开全局中断
#define __enable_interrupt() __enable_interrupt()

#define _disable_interrupt() __disable_interrupt() //关全局中断
#define __disable_interrupt() __disable_interrupt()

#define _set_interrupt_state(x) __set_interrupt_state(x)
#define __set_interrupt_state(x) __set_interrupt_state(x)

#define _get_interrupt_state() __get_interrupt_state()
#define __get_interrupt_state() __get_interrupt_state()

#define _delay_cycles(x)    __delay_cycles(x)    //延时
#define __delay_cycles(x)    __delay_cycles(x)
```




Q&A

