

RX 族

R20AN0398CC0100

Rev.1.00

Workbench6 生成项目的导入指南

2016.12.31

要点

本应用说明介绍如何将 Workbench6 生成的 Touch Program 导入到用户项目中。

对象 MCU

RX113 群

将本篇应用说明应用于其他单片机时，需结合单片机规格进行变更，并进行详细评价。

目录

1. 概要	3
2. 动作确认条件	3
3. 相关应用说明	3
4. 导入至用户项目	4
4.1 必要文件的导入	4
4.1.1 必要文件	4
4.1.2 导入至用户项目	5
4.2 用户程序的编辑	23
4.2.1 时钟的设定	23
4.2.2 主程序的编辑	24
4.2.3 定周期 Timer 的使用	25
4.2.4 DTC 向量表的设定	30
5. 利用 Workbench6 确认动作	31
5.1 通过综合开发环境 CS+确认动作的设定步骤	31
5.2 通过 USB 确认动作的设定步骤	42
6. 注意事项	57
6.1 时钟的设定	57
6.2 Touch 控制处理	57
6.3 和 Workbench6 的连接	57
7. 参考文献	58

1. 概要

本应用说明介绍如何将 Workbench6 生成的 Touch Key 相关程序导入到用户开发的应用项目中。

2. 动作确认条件

本应用说明所述步骤，是以以下假定条件为前提的。

表 2.1 动作确认条件

项目	内容
所用 MCU	RX113
综合开发环境	瑞萨电子开发 CS+ for CC V3.00.00
Workbench6	瑞萨电子开发 Workbench Version:1.03.0000
C 编译器	瑞萨电子开发 C/C++ Compiler Package for RX Family V2.02.00
字节存储次序	Little Endian
动作模式	单芯片模式

3. 相关应用说明

使用本应用说明时，请同时参考以下相关的文档。

- CTSU API 参考指南（R30AN0215C）

4. 导入至用户项目

本节内容说明将 Workbench6 生成的 Touch 处理程序导入至用户项目的步骤。

4.1 必要文件的导入

4.1.1 必要文件

在 Workbench6 生成的文件中，需要导入到用户项目的文件，如表 4.1 所示。

表 4.1 文件构成一览

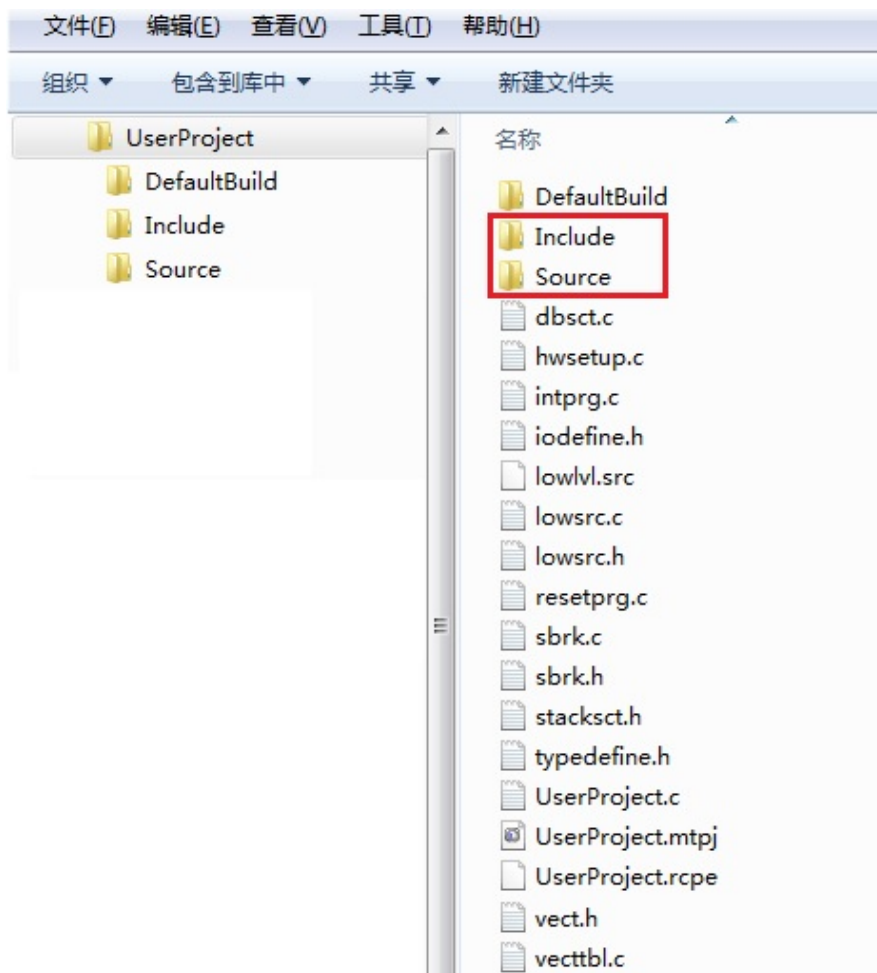
文件名	内容
Source\r_touch_API.c	API 源程序
Source\CTSU\r_ctsu.c	CTSU 源程序
Source\CTSU\r_ctsu_physical_driver.c	CTSU 寄存器访问源程序
Source\DTCL\r_dtc.c	DTC 源程序
Source\HwResource\r_cg_cgc_user.c	工作时钟的用户设定程序
Source\HwResource\r_cg_cmt.c※	CMT 源程序
Source\HwResource\r_cg_cmt_user.c※	CMT 的用户设定程序
Source\HwResource\r_mpc.c	MPC 设定源程序
Source\Touch\r_touch.c	静电电容 Touch 源程序
Include\r_touch_API.h	API 头文件
Include\r_cg_userdefine.h	用户定义设定头文件
Include\CTSU\r_ctsu.h	CTSU 头文件
Include\DTCL\r_dtc.h	DTC 头文件
Include\HwResource\r_cg_cmt.h※	CMT 头文件
Include\HwResource\r_cg_macrodriver.h	宏定义头文件
Include\HwResource\r_cg_cgc.h	工作时钟头文件
Include\HwResource\r_mpc.h	MPC 头文件
Include\Touch\r_touch.h	静电电容 Touch 头文件

注： 用户系统包含其他的定周期处理程序时，无需导入。详细内容请参考“4.2.3 定周期 Timer 的使用”。

4.1.2 导入至用户项目

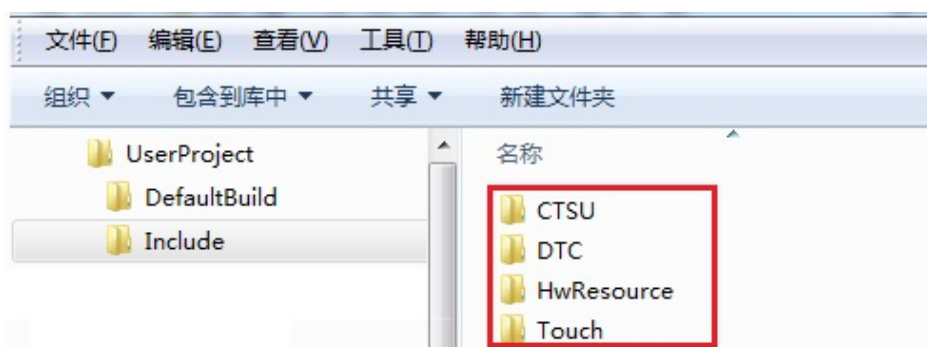
本节内容阐述将 Workbench6 生成的程序导入用户项目的步骤。

- 1) 首先，在用户项目中创建和 Workbench6 生成的文件夹结构相同的文件夹，用于存放必要的文件。
 - 利用资源管理器，在包含用户项目文件（扩展名.mtpj）的文件夹中，创建 Source 文件夹和 Include 文件夹，存放用于 Touch 处理的源程序和头文件。



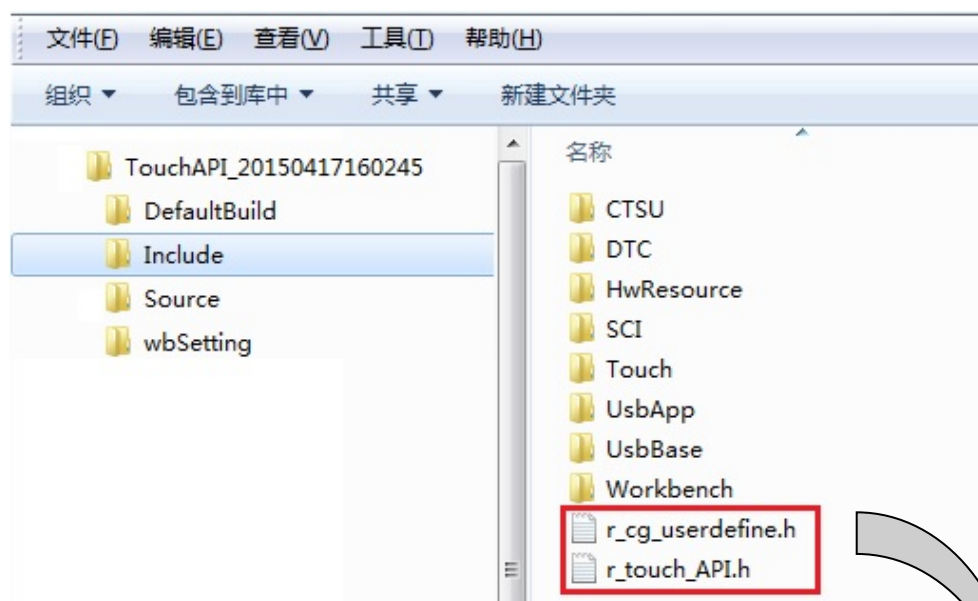
注： 用户项目中既存 Source 和 Include 文件夹时，无需重新创建。

- 在以上创建的 Source 文件夹和 Include 文件夹中，分别创建 CTSU、DTC、HwResource 和 Touch 共计 4 个文件夹。

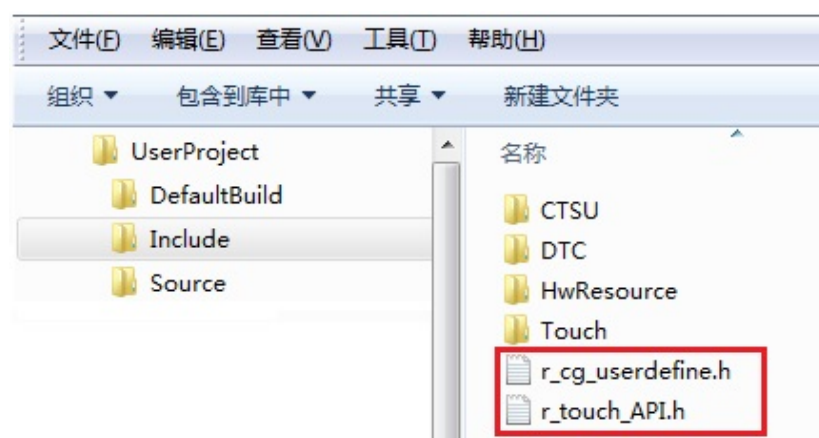


- 2) 其次，将必要的文件拷贝至用户项目中。
- 将 Workbench6 生成的 Include 文件夹中的头文件，拷贝至用户项目的 Include 文件夹中。

Workbench6 生成的文件夹 Include



用户项目中的文件夹 Include

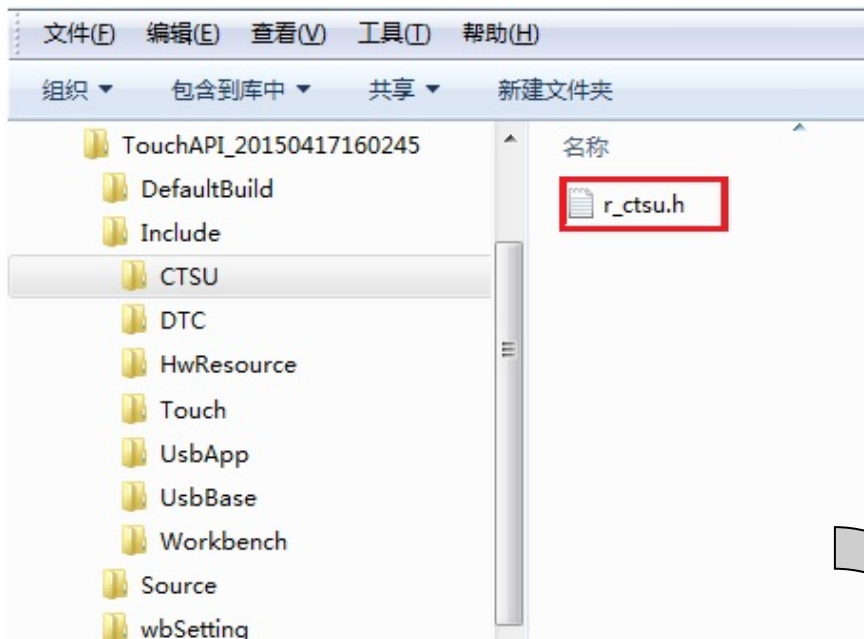


拷贝文件

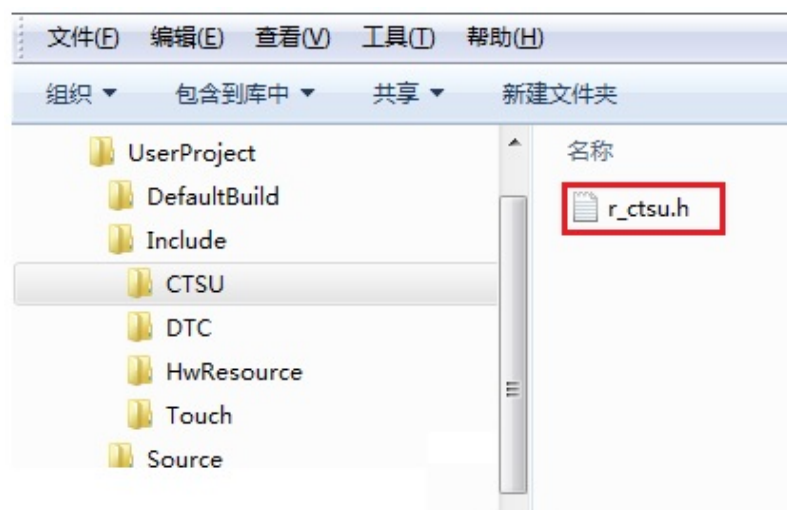


- 将 Workbench6 生成的 Include\CTSU 文件夹中的头文件，拷贝至用户项目的 Include\CTSU 文件夹中。

Workbench6 生成的文件夹 Include\CTSU



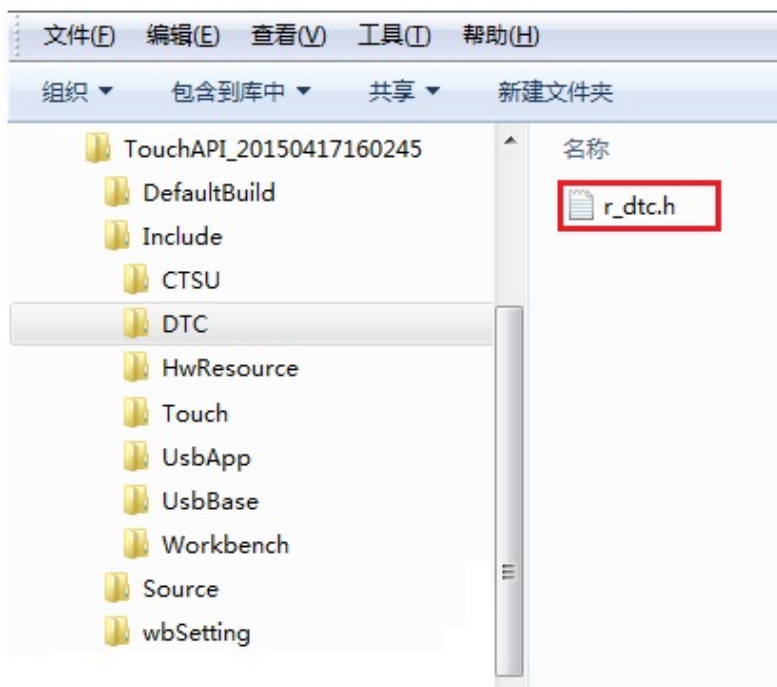
用户项目中的文件夹 Include\CTSU



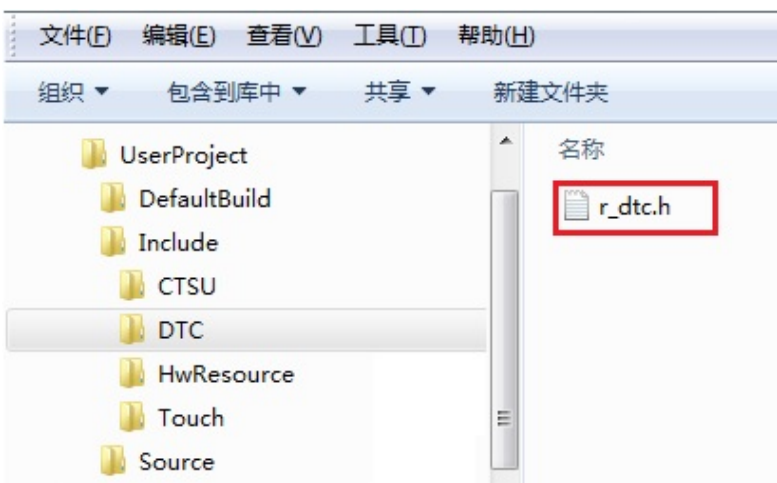
拷贝文件

- 将 Workbench6 生成的 Include\DTC 文件夹中的头文件，拷贝至用户项目的 Include\DTC 文件夹中。

Workbench6 生成的文件夹 Include\DTC



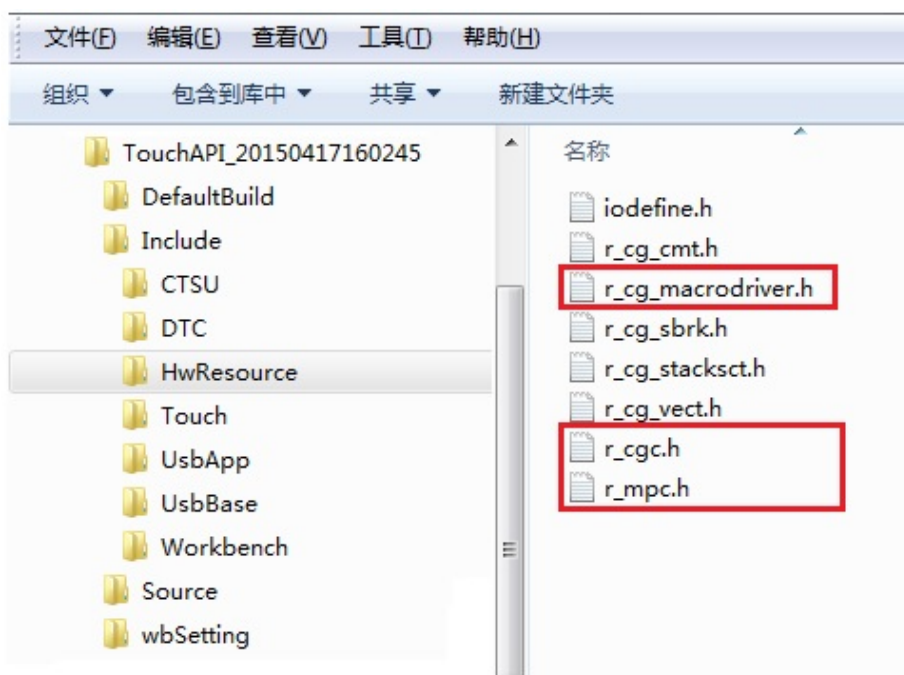
用户项目中的文件夹 Include\DTC



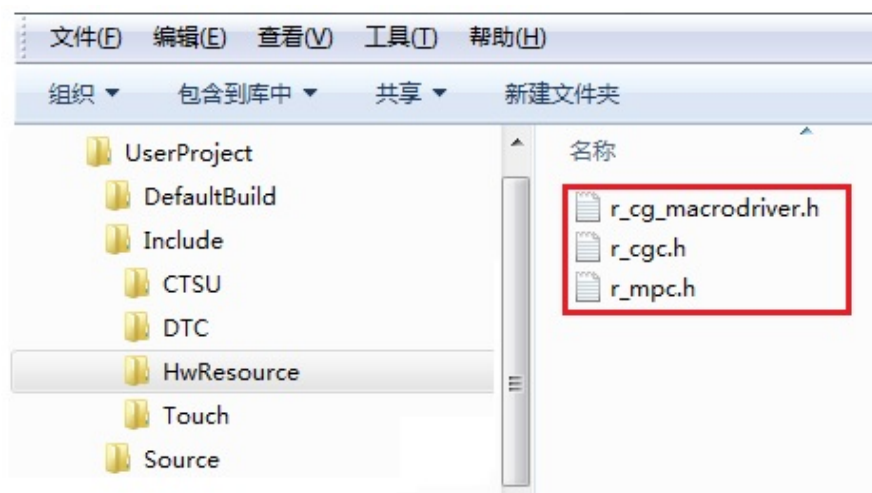
拷贝文件

- 将 Workbench6 生成的 Include\HwResource 文件夹中的头文件 r_cg_macrodriver.h、r_cg.h、r_mpc.h，拷贝至用户项目的 Include\HwResource 文件夹中。

Workbench6 生成的文件夹 Include\HwResource



用户项目中的文件夹 Include\HwResource

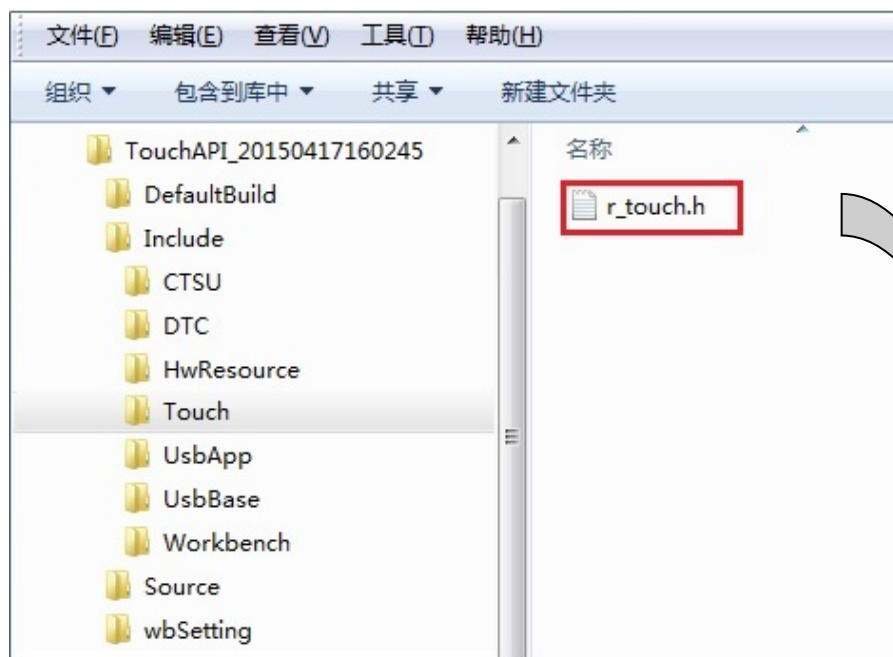


拷贝文件



- 将 Workbench6 生成的 Include\Touch 文件夹中的头文件，拷贝至用户项目的 Include\Touch 文件夹中。

Workbench6 生成的文件夹 Include\Touch



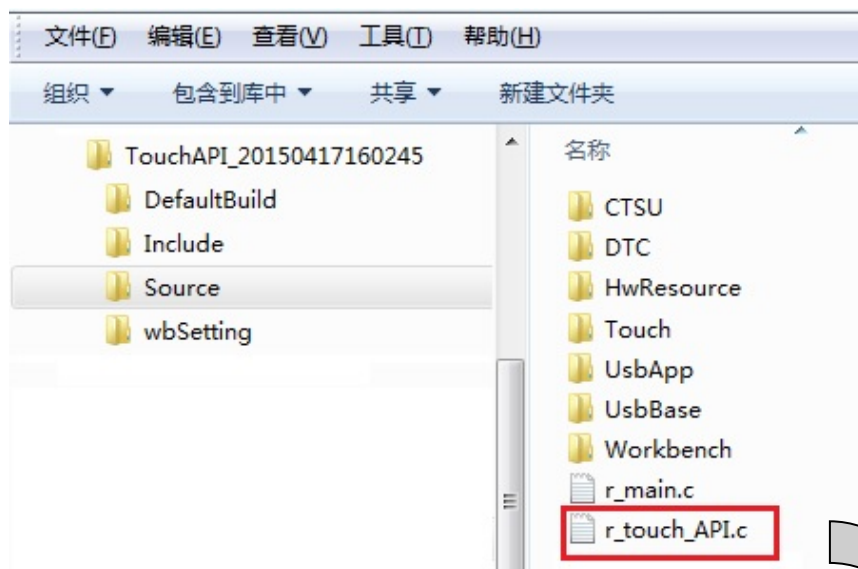
拷贝文件

用户项目中的文件夹 Include\Touch

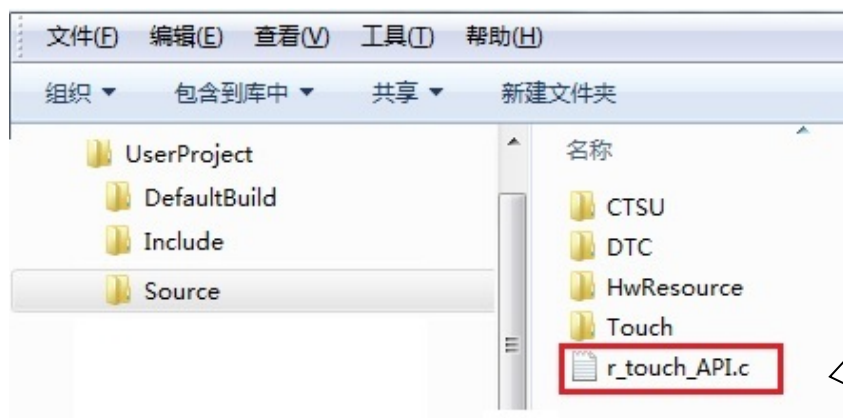


- 将 Workbench6 生成的 Source 文件夹中的源程序 r_touch_API.c，拷贝至用户项目的 Source 文件夹中。

Workbench6 生成的文件夹 Source



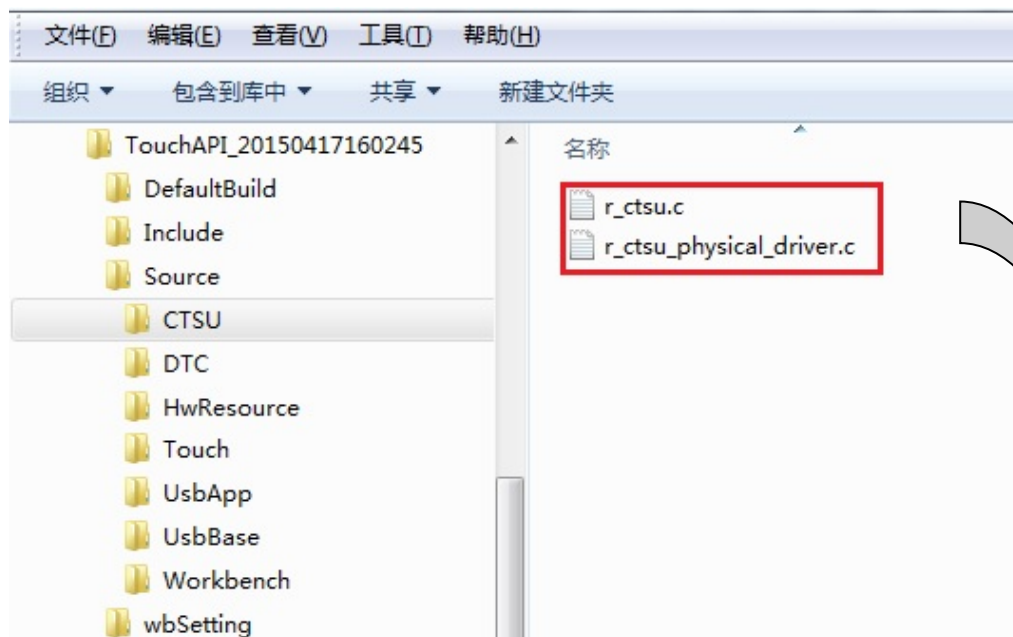
用户项目中的文件夹 Source



拷贝文件

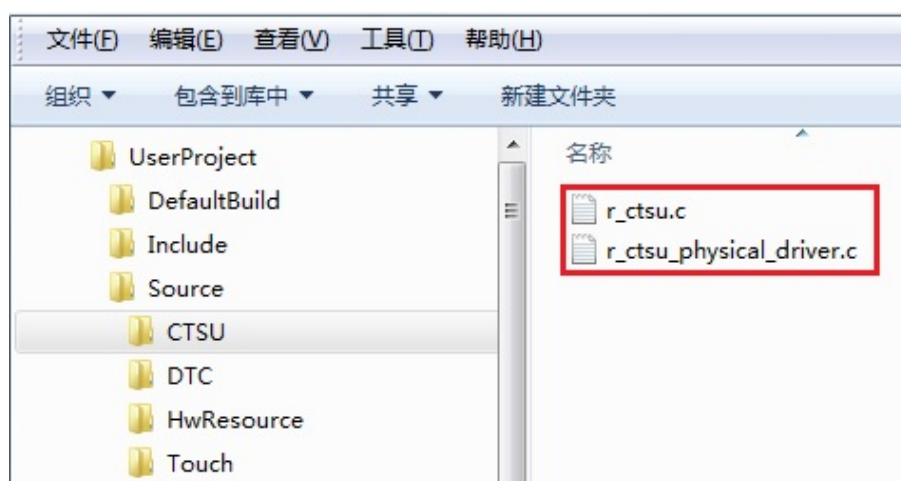
- 将 Workbench6 生成的 Source\CTSU 文件夹中的源程序，拷贝至用户项目的 Source\CTSU 文件夹中。

Workbench6 生成的文件夹 Source\CTSU



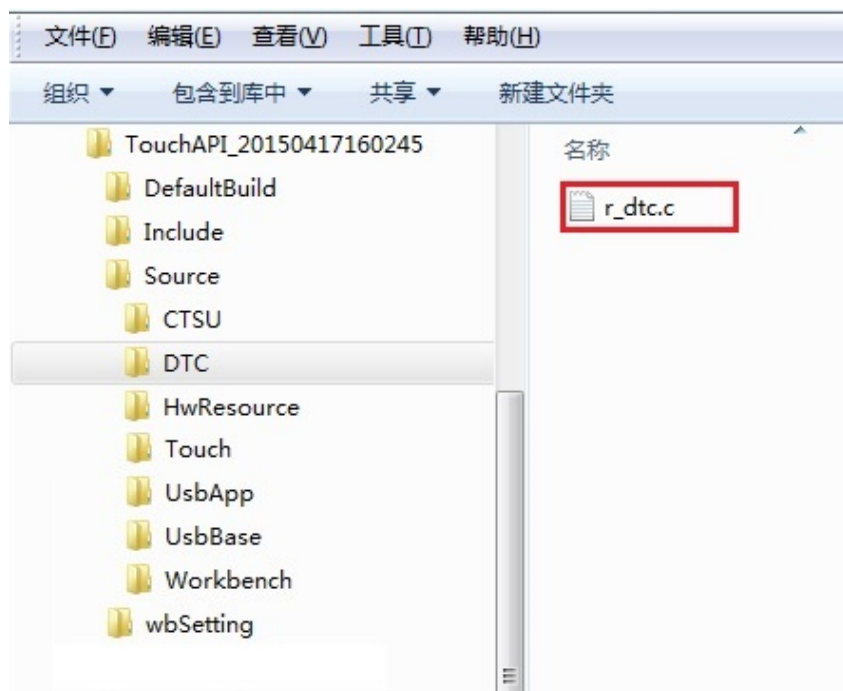
拷贝文件

用户项目中的文件夹 Source\CTSU

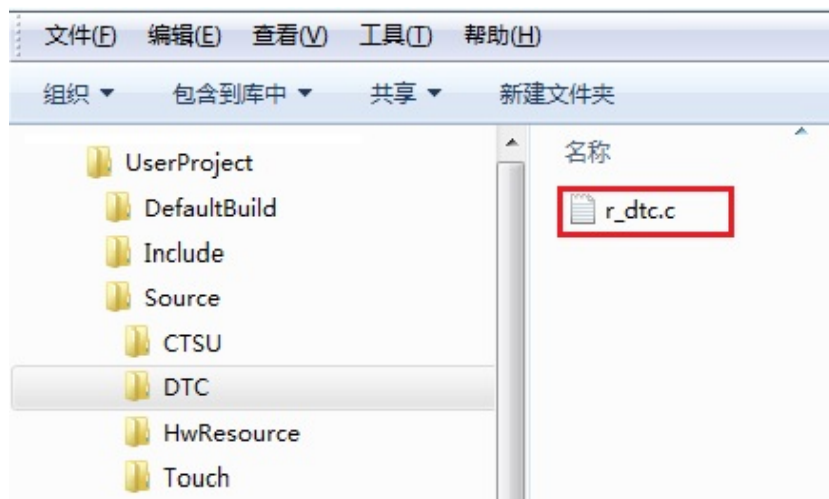


- 将 Workbench6 生成的 Source\DTC 文件夹中的源程序，拷贝至用户项目的 Source\DTC 文件夹中。

Workbench6 生成的文件夹 Source\DTC



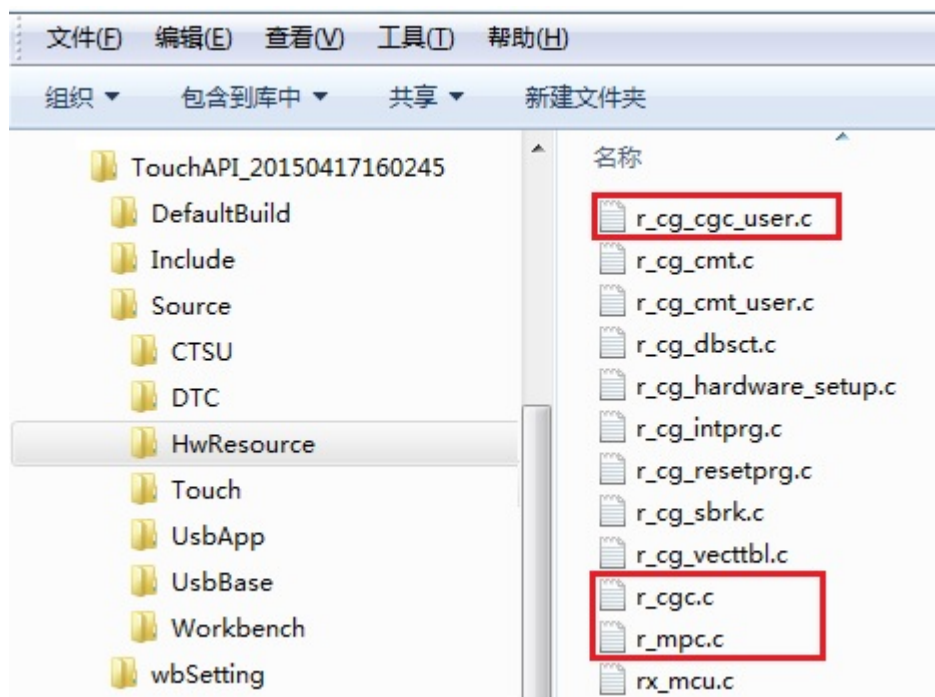
用户项目中的文件夹 Source\DTC



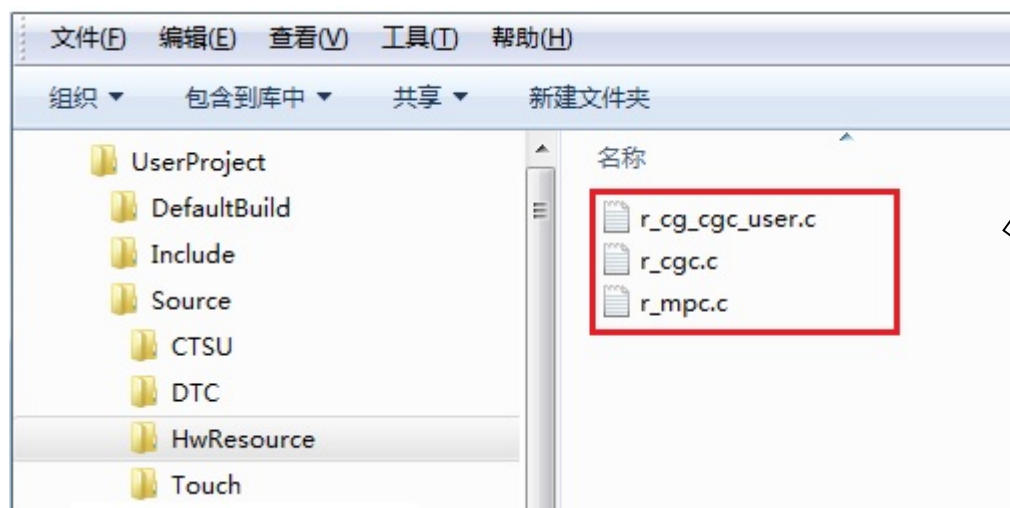
拷贝文件

- 将 Workbench6 生成的 Source\HwResource 文件夹中的源程序，拷贝至用户项目的 Source\HwResource 文件夹中。

Workbench6 生成的文件夹 Source\HwResource



用户项目中的文件夹 Source\HwResource

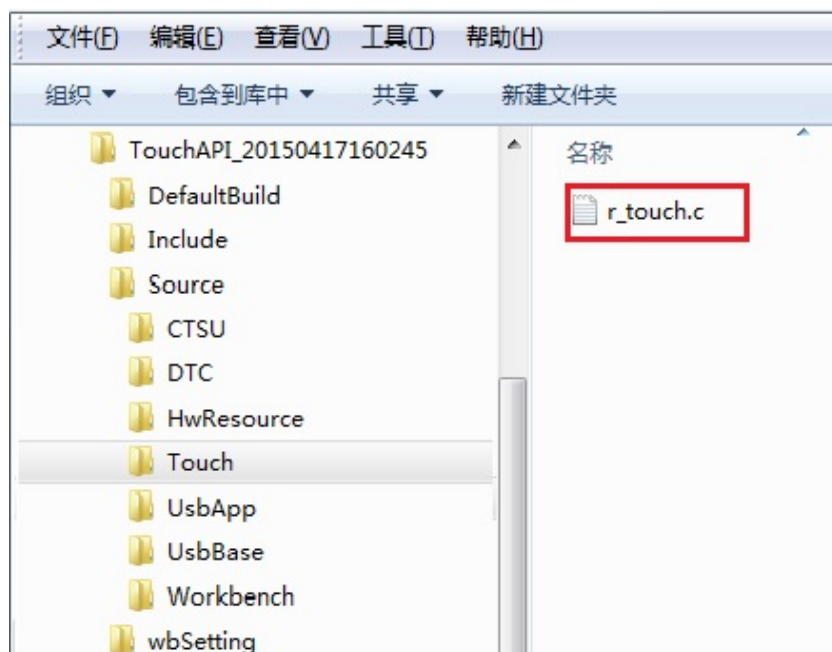


拷贝文件

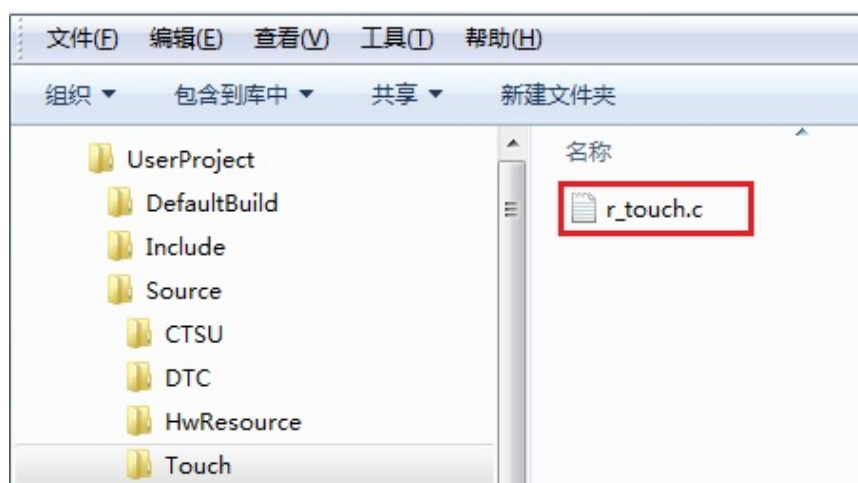


- 将 Workbench6 生成的 Source\Touch 文件夹中的源程序，拷贝至用户项目的 Source\Touch 文件夹中。

Workbench6 生成的文件夹 Source\Touch

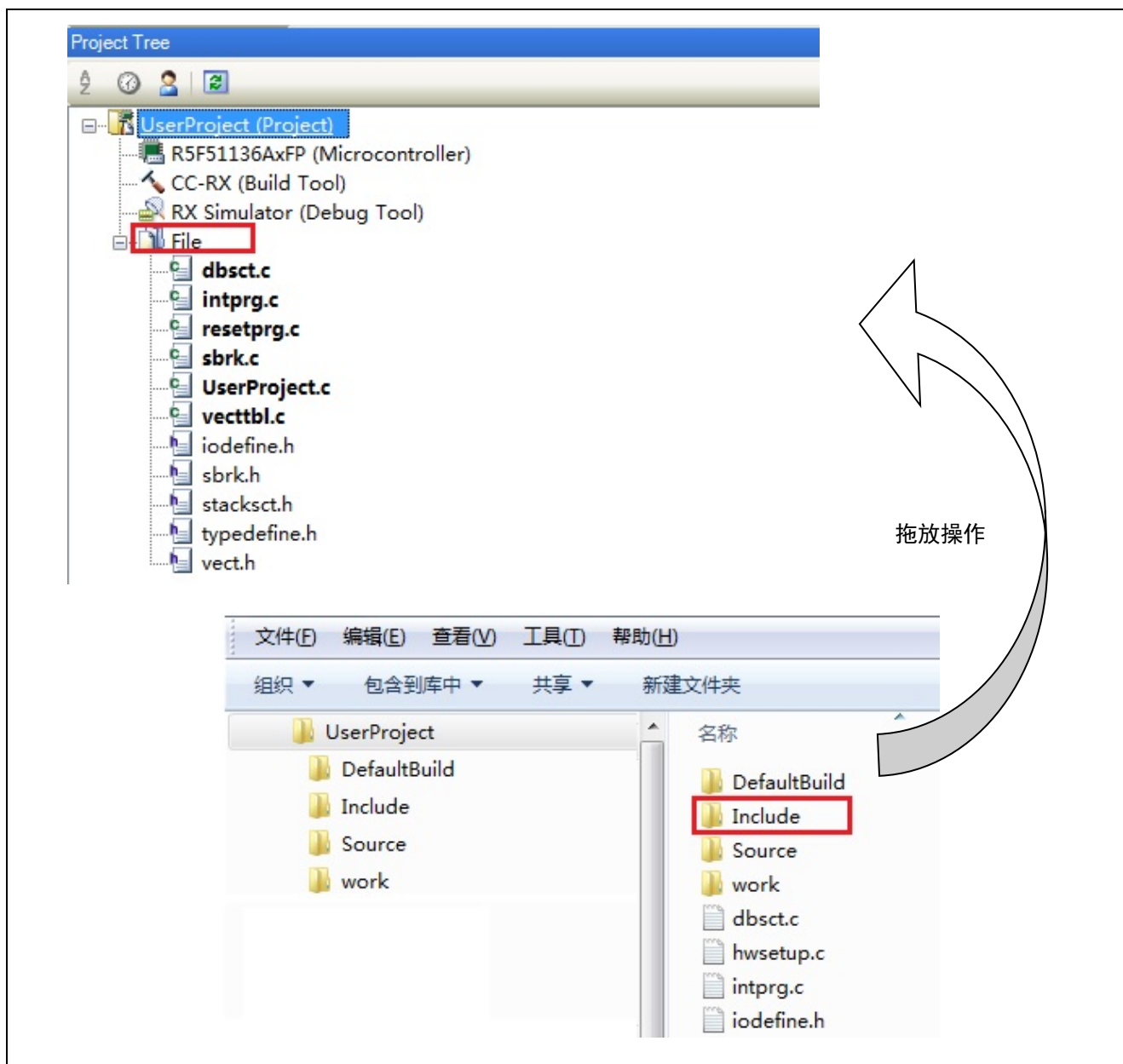


用户项目中的文件夹 Source\Touch

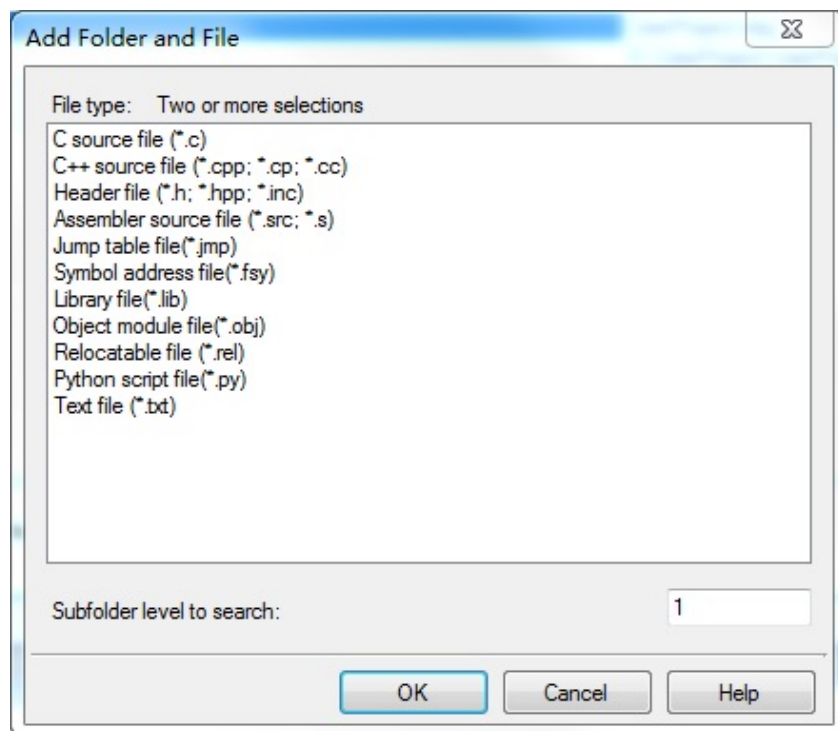


拷贝文件

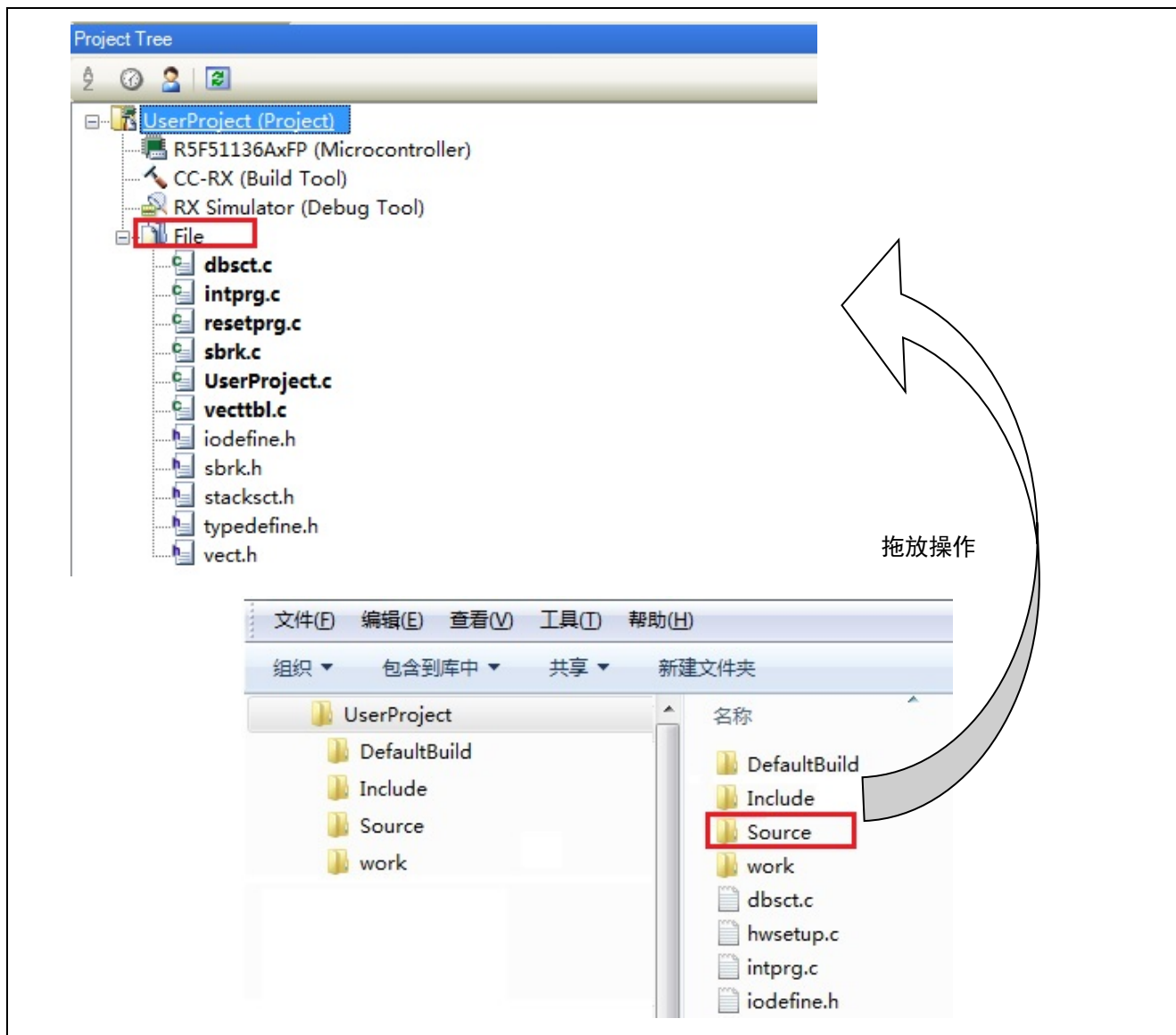
- 3) 将步骤 2) 中拷贝的文件，添加到综合开发环境 CS+ 中。
- 从资源管理器中，将 Include 文件夹拖放至综合开发环境 CS+ 中。



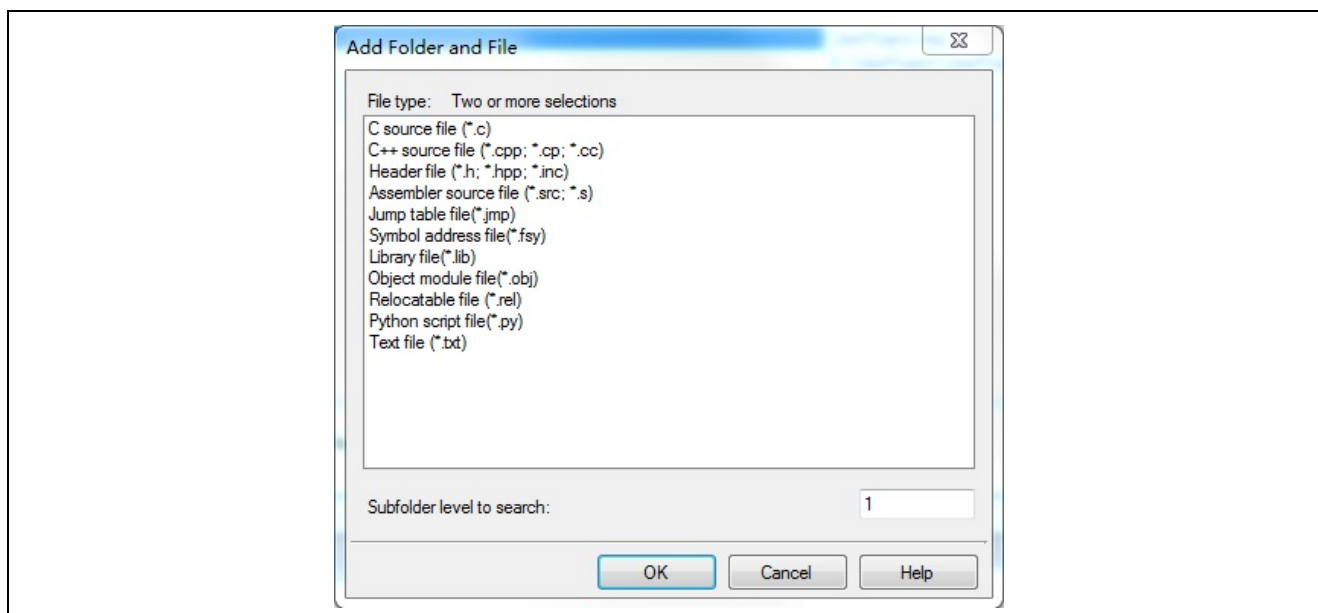
- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。



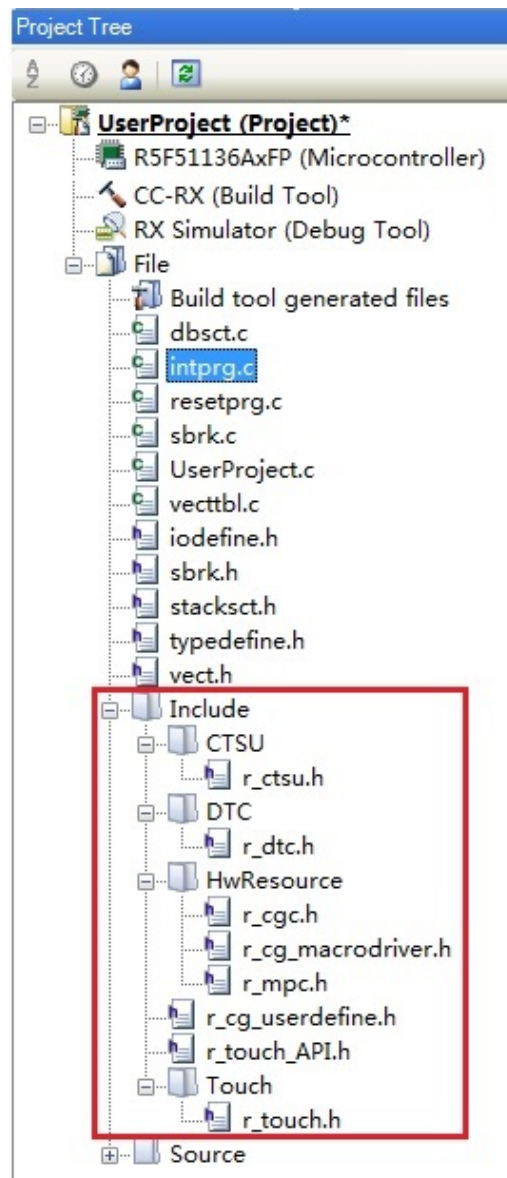
- 从资源管理器中，将 Source 文件夹拖放至综合开发环境 CS+中。



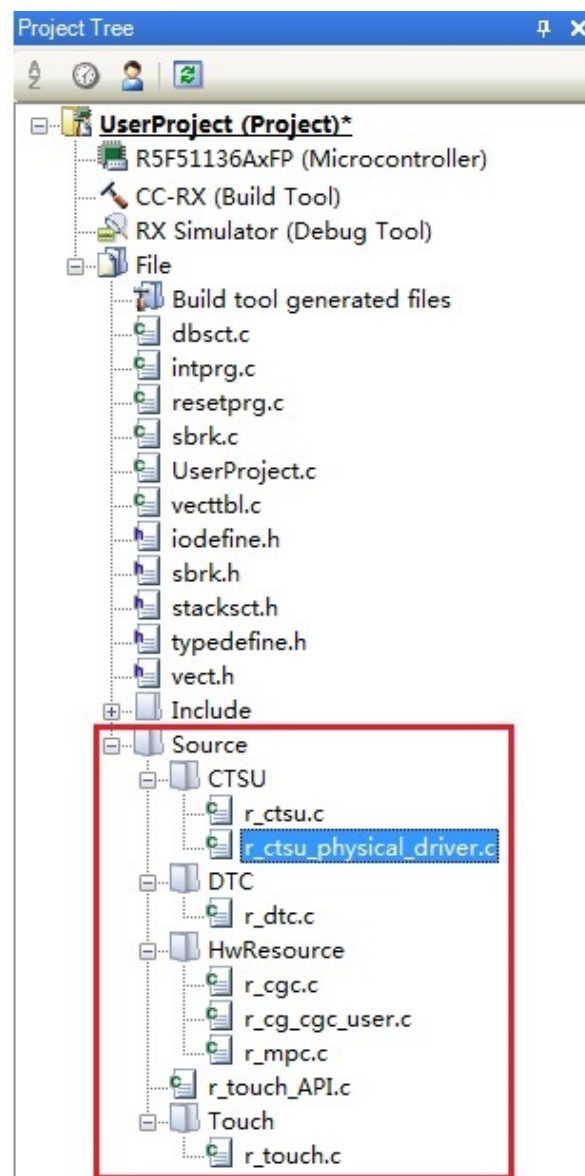
- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。



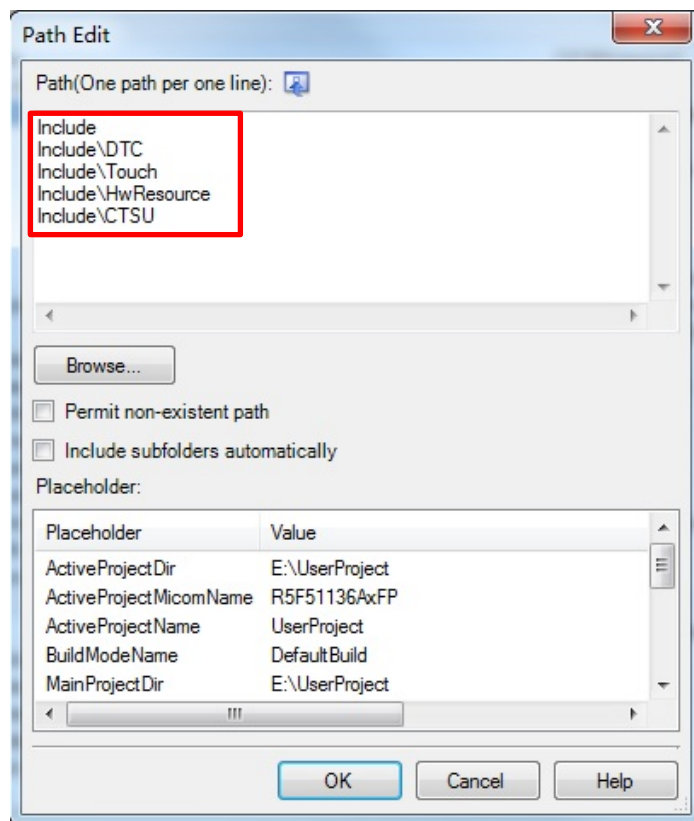
- 确认拷贝到 Include 文件夹中的头文件是否添加到了 CS+ IDE 中。



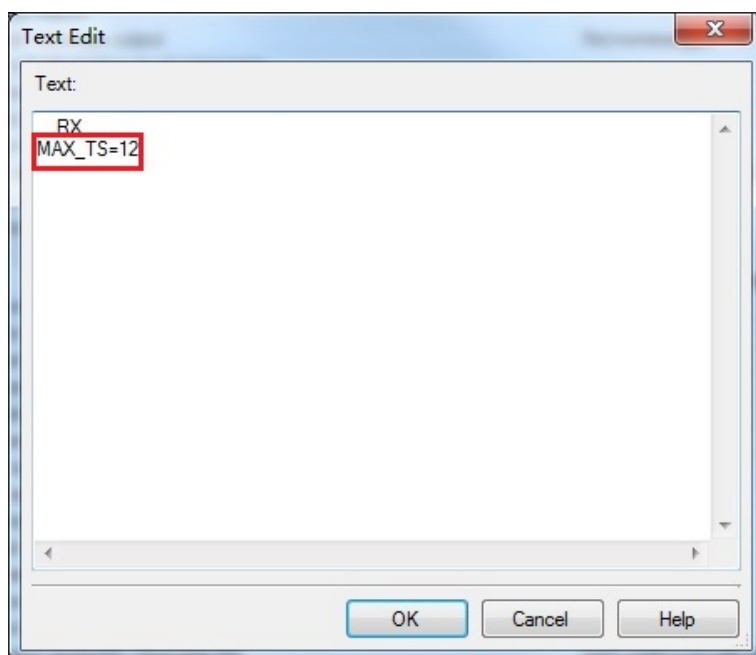
- 确认拷贝到 Source 文件夹中的源程序，是否添加到了 CS+ IDE 中。



- 4) 在综合开发环境 CS+中，设定编译选项。
- 打开 CC-RX 的「Property」窗口，单击「Common Options」选项卡，进入「Additional include paths」，确认下图红框内所示的路径是否存在。如果不存在，添加这些路径。

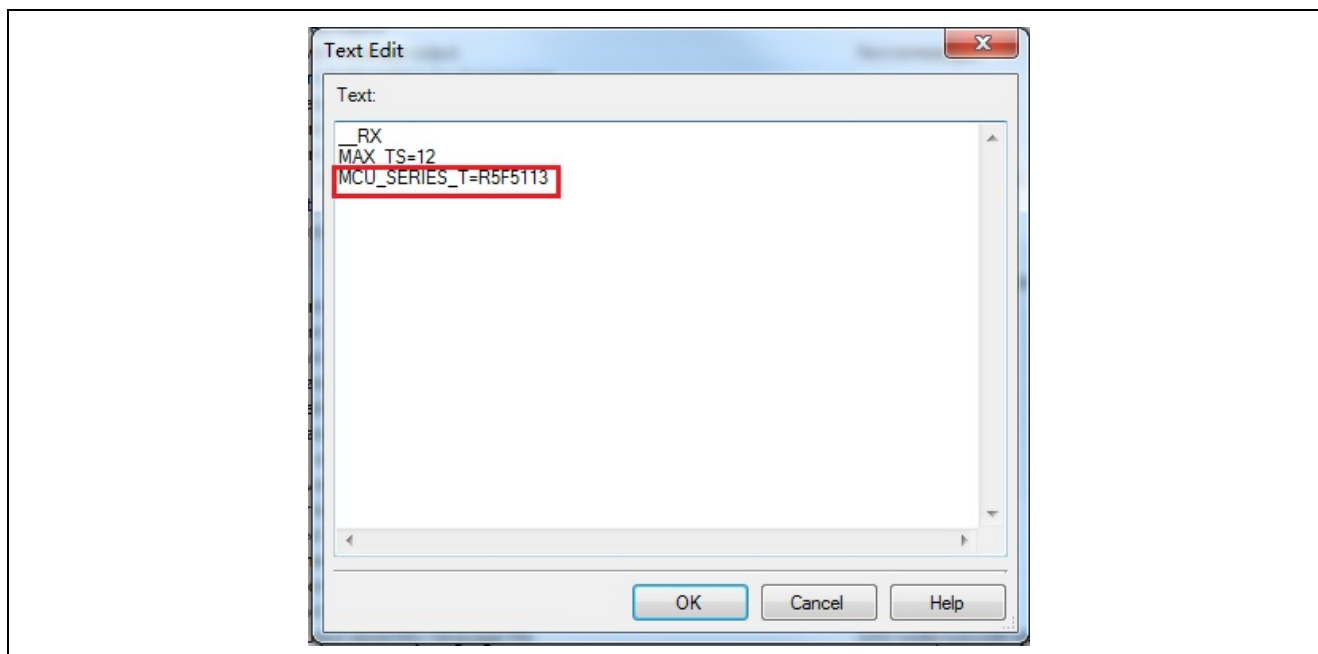


- 在 CC-RX 的「Property」窗口中，通过「Common Options」>>「Macro definition」，定义 MAX_TS = 12^{*}。



注： MAX_TS 的具体数值，请通过 Workbench6 生成的项目确认。MAX_TS 的数值，因 MCU 型号不同而异。MAX_TS 的数值，等于所用 MCU 可以提供的 TS 端子总数。

- 在 CC-RX 的「Property」窗口中，通过「Common Options」>>「Macro definition」，定义 MCU_SERIES_T 。



注： MCU_SERIES_T 的数值，用于设定所用 MCU 的系列名。对于 RX113 来说，应设定为 R5F5113。

- 编辑用户项目中包含中断处理函数的源文件。
 - 屏蔽 Touch 处理程序使用的中断处理函数。下面示例中所编辑的源文件，是 CS+自动生成的 intprg.c 文件。

（省略）

```
// CTSU CTSUWR
void Excep_CTSU_CTSUWR(void) { }

// CTSU CTSURD
void Excep_CTSU_CTSURD(void) { }

// CTSU CTSUFN
void Excep_CTSU_CTSUFN(void) { }
```

屏蔽处理

```
// RTC CUP
void Excep_RTC_CUP(void) { }
```

（省略）

- 编辑 Source\CTSU 文件夹中的 r_cts.c 源文件。
 - 屏蔽#include "r_cg_vect.h"预处理指令，代之以包含用户项目中向量表所用的头文件。下面的示例中，包含了"vect.h"头文件。

（省略）

```
/* H/W include header */
#include "iodefine.h"
#include "r_cg_vect.h"
#include "vect.h"
#include "r_mpc.h"
#include "r_dtc.h"
#include "r_cts.h"
```

屏蔽旧的向量表头文件，包含用户项目中向量表所用的头文件。

（省略）

4.2 用户程序的编辑

4.2.1 时钟的设定

Touch 处理程序，使用了外围模块时钟 B (PCLKB)。

- 1) 外围模块时钟 B 的设定，使用 Workbench6 生成程序中的设定值。

如果用户系统所用时钟不同于 Workbench6 生成程序中的时钟设定，请将用户系统时钟变更为 Workbench6 生成程序中的时钟设定值。

时钟设定的示例，如下图所示。

r_cgc.h

```
/* Start user code for function. Do not edit comment generated here */

/* Select the clock condition */
#define TOUCH_CFG_CLOCK_SOURCE      (4)
#define TOUCH_CFG_HOCO_FREQUENCY    (0)
#define TOUCH_CFG_XTAL_HZ           (16000000)

/* PLL Input Frequency Division Ratio Select/Frequency Multiplication Factor Select */
#define TOUCH_CFG_UPLL_DIV           (2)      /* USB-dedicated PLL Input Frequency Division Ratio Select */
#define TOUCH_CFG_UPLL_MUL           (6)      /* Frequency Multiplication Factor Select */
#define TOUCH_CFG_PLL_DIV            (4)      /* PLL Input Frequency Division Ratio Select */
#define TOUCH_CFG_PLL_MUL            (8)      /* Frequency Multiplication Factor Select */

/* System Clock Control Register (SCKCR) */
#define TOUCH_CFG_ICLK_DIV            (1)      /* System Clock (ICLK) Select */
#define TOUCH_CFG_PCKB_DIV            (1)      /* Peripheral Module Clock B(PCLKB) Select */

#define TOUCH_CFG_FCK_DIV TOUCH_CFG_ICLK_DIV /* FlashIF Clock (FCLK) Select */
#define TOUCH_CFG_PCKD_DIV TOUCH_CFG_PCKB_DIV /* Peripheral Module Clock D(PCLKD) Select */
```

(省略)

```
#define TOUCH_SCKCR    TOUCH_PCKD_DIV | TOUCH_PCKB_DIV | TOUCH_ICLK_DIV | TOUCH_FCK_DIV
```

用户的时钟设定处理

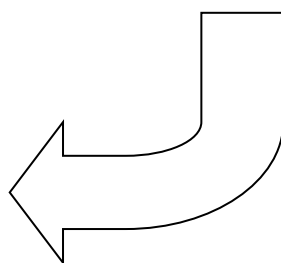
(省略)

```
uint32_t sckcr_dummy;

/* Set system clock */
sckcr_dummy = TOUCH_SCKCR;
SYSTEM_SCKCR_LONG = sckcr_dummy;

while (SYSTEM_SCKCR_LONG != sckcr_dummy);
```

(省略)



设定为 TOUCH_SCKCR

4.2.2 主程序的编辑

- 1) 在用户项目的主程序中，追加 Touch API 头文件的包含指令。具体如下所示。

(省略)

```
#include "r_touch_API.h"
```

包含 Touch API 头文件

(省略)

- 2) 主函数中追加 Touch 处理程序。

- 追加程序的示例，如下图所示。

(省略)

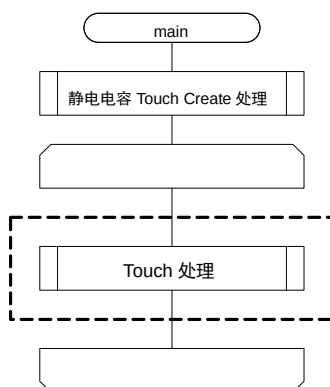
```
void main(void)
{
    uint8_t method;
    method = 0U;
    R_Set_Cap_Touch_Create(method);
    g_otsu_soft_mode = OTSU_READY_MODE;
    while(1U)
    {
        R_Set_Cap_Touch_measurement_start(method);
        if (GET_OK == R_Get_Cap_Touch_Data_Check(method))
        {
            if (INITIAL_END == R_Get_Cap_Touch_Initial_status())
            {
                if (GET_OK == R_Get_Cap_Touch_Function_Data(method))
                {
                }
            }
            else
            {
                R_Set_Cap_Touch_Initial_Tuning(method);
            }
        }
        method = R_Set_Cap_Touch_Next_Method_Change(method);
    }
}
```

追加内容

(省略)

注： 初始化函数 R_Set_Cap_Touch_Create()之外的 Touch 处理程序，必须以 2ms 为周期循环执行。

- 示例程序的大致流程，如下图所示。虚线框包围的内容，必须在 2ms 内执行完成。



4.2.3 定周期 Timer 的使用

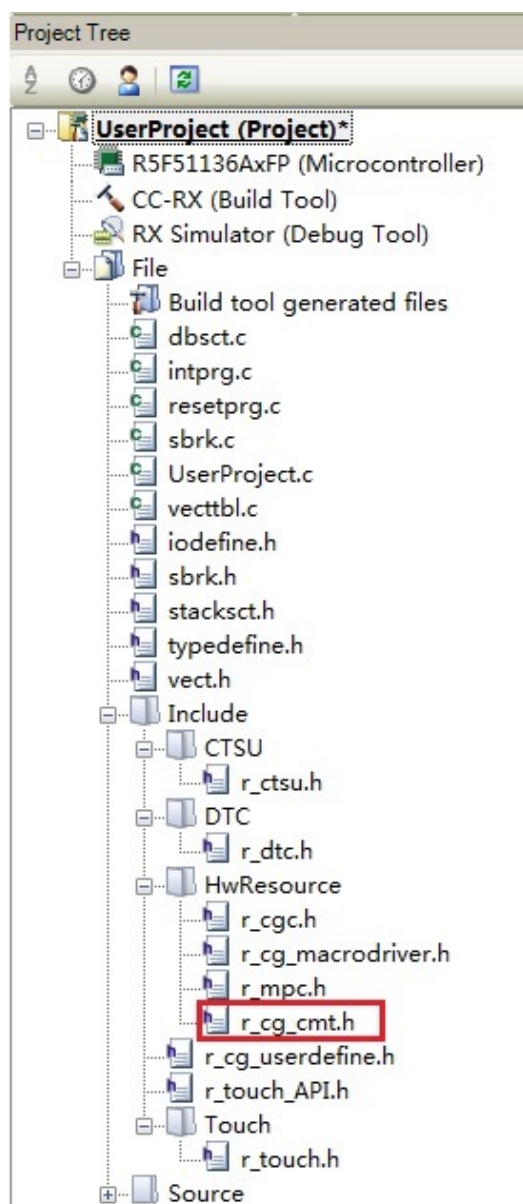
Touch 处理程序必须以 2ms 的固定周期执行。

在 Workbench6 生成的程序中，使用了比较一致定时器（CMT0）。如果用户系统中未使用周期固定的定时程序，请按照下面 1)~3)的步骤进行处理。

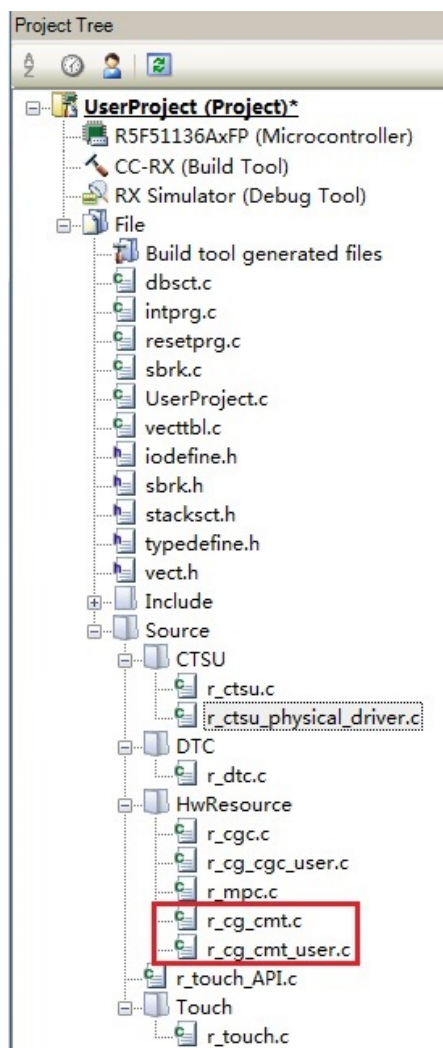
如果用户系统中已经使用了 2ms 周期的定时程序，可以使用既存的定时程序。但是，在使用既存定时程序时，需在定时程序内执行以下 4)的内容。

1) 在用户项目中追加以下文件。

- 将 Workbench6 生成的 Include\HwResource 文件夹中的 r_cg_cmt.h 文件，拷贝至用户项目的 Include\HwResource 文件夹中，并添加到综合开发环境 CS+中。



- 将 Workbench6 生成的 Source\HwResource 文件夹中的 r_cg_cmt.c、r_cg_cmt_user.c，拷贝至用户项目的 Source\HwResource 文件夹中，并将这 2 个文件添加到综合开发环境 CS+中。



- 编辑用户项目中包含中断处理函数的源文件。

- 屏蔽用户项目中缺省的 CMT0 中断处理函数。以下示例中所编辑的文件，是通过综合开发环境 CS+自动生成的 intprg.c 文件。

(省略)

```
// ICU SWINT
void Excep_ICU_SWINT(void) { }

// CMT0 CM10
void Excep_CMT0_CM10(void) { }

// CMT1 CM11
void Excep_CMT1_CM11(void) { }

// CMT2 CM12
void Excep_CMT2_CM12(void) { }
```

屏蔽处理

(省略)

- 3) 主程序中追加固定周期定时器的初始化函数和启动函数。
- 包含 r_cg_cmt.h 头文件。
 - 追加 CMT0 的初始化函数和 CTM0 的启动函数。

(省略)

```
#include "r_touch_API.h"  
#include "r_cg_cmt.h"
```

追加内容

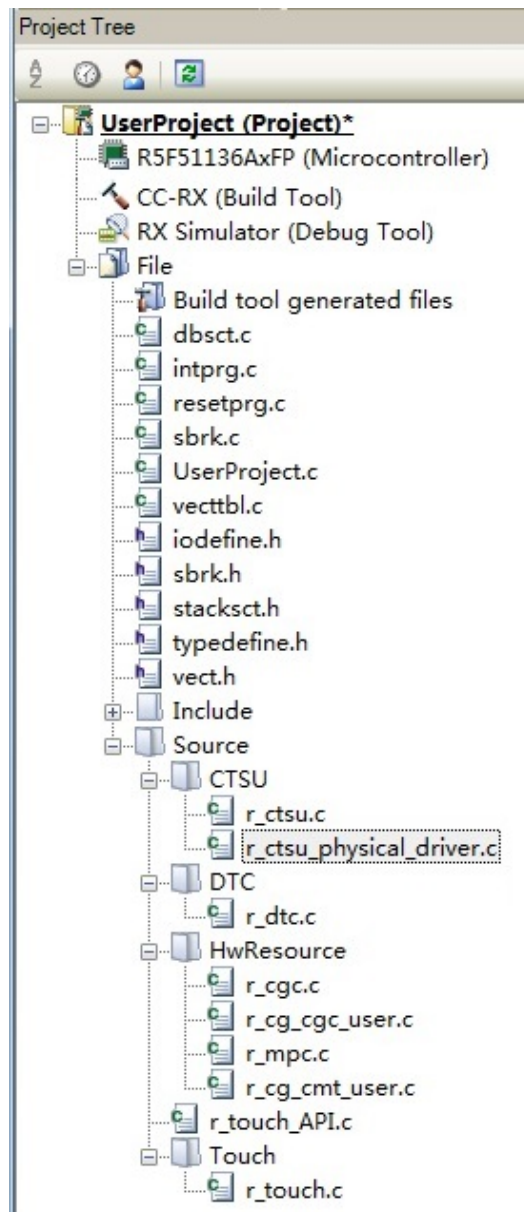
(省略)

```
void main(void)  
{  
    uint8_t method;  
    method = 0U;  
    R_Set_Cap_Touch_Create(method);  
    R_CMT0_Create();  
    R_CMT0_Start();  
    while(1U)
```

追加内容

(省略)

- 4) 用户系统中已经存在定周期定时程序时，进行如下处理：
- 将 Workbench6 生成的 Source\HwResource 文件夹中的 r_cg_cmt_user.c 源程序，拷贝至用户项目的 Source\HwResource 文件夹中，并添加到综合开发环境 CS+中。



- 编辑 r_cg_cmt_user.c 文件中的函数 r_cmt_cmi0_interrupt(), 使其可以外部调用。

(省略)

```

3/*****
 * Function Name: r_cmt_cmi0_interrupt
 * Description  : None
 * Arguments    : None
 * Return Value : None
 *****/
//if FAST_INTERRUPT_VECTOR == VECT_CMT0_CMI0
//pragma interrupt r_cmt_cmi0_interrupt(vect=VECT(CMT0, CMI0), fint)
//else
//pragma interrupt r_cmt_cmi0_interrupt(vect=VECT(CMT0, CMI0))
//endif
//static void r_cmt_cmi0_interrupt(void)
void r_cmt_cmi0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
3/*
    if( debug_cnt == 0 )
    {
        PORT1.PMR  = 0x00;
        PORT1.PODR.BYTE = 0x08;
        PORT1.PDR.BYTE = 0x08;
        debug_cnt = 1;
    }
    else
    {
        PORT1.PODR.BYTE = 0x00;
        PORT1.PDR.BYTE = 0x08;
        debug_cnt = 0;
    }
3*/
    g_touch_system.flag.cmt_timing = 1; /* CTSU measurement timing */
    /* End user code. Do not edit comment generated here */
}

```

屏蔽处理

将函数的 static 属性变更为 global

(省略)

- 在定周期处理程序中，调用函数 r_cmt_cmi0_interrupt()。

(省略)

```

{
    r_cmt_cmi0_interrupt();
}

```

(省略)

4.2.4 DTC 向量表的设定

Touch 处理程序中使用了 DTC 功能。如果用户系统中也使用了 DTC 功能，请按照以下方法处理。如果用户系统中未使用 DTC 功能，请忽略以下的处理。

- 1) 确认 DTC 向量表寄存器的设定值。
 - 确认 Source\DTC\r_dtc.c 文件中 DTC_Set_Initial_of_CTSU()函数内寄存器的设定值。
 - 如果 DTC 向量表寄存器的设定值和用户系统使用的设定值不同，请执行下一步骤的内容。

(省略)

```
/* Set transfer data start address in DTC vector table */  
dtc_vector60 = (uint32_t)&g_dtc_info_ctsu_wr;  
  
/* Set transfer data start address in DTC vector table */  
dtc_vector61 = (uint32_t)&g_dtc_info_ctsu_rd;
```

DTC.DTCVBR = (void *)0x00007C00U; 确认设定值

(省略)

- 2) 按照用户系统的设定，变更各向量的设定值。
 - 下图的示例中，用户系统使用 0000 8000H 地址。

(省略)

```
//#pragma address dtc_vector60=0x00007CF0U 变更  
#pragma address dtc_vector60=0x000080F0U  
uint32_t dtc_vector60;  
DTC_EXTERN dtc_register_data_t g_dtc_info_ctsu_wr;  
  
//#pragma address dtc_vector61=0x00007CF4U 变更  
#pragma address dtc_vector61=0x000080F4U  
uint32_t dtc_vector61;  
DTC_EXTERN dtc_register_data_t g_dtc_info_ctsu_rd;
```

(省略)

```
/* Set transfer data start address in DTC vector table */  
dtc_vector60 = (uint32_t)&g_dtc_info_ctsu_wr;  
  
/* Set transfer data start address in DTC vector table */  
dtc_vector61 = (uint32_t)&g_dtc_info_ctsu_rd;
```

```
// DTC.DTCVBR = (void *)0x00007C00U;  
DTC.DTCVBR = (void *)0x00008000U; 变更
```

(省略)

5. 利用 Workbench6 确认动作

程序导入后，Workbench6 可以利用以下 2 种连接方法实现动作确认：

- 通过综合开发环境 CS+和用户系统连接；
- 通过 USB*和用户系统连接。

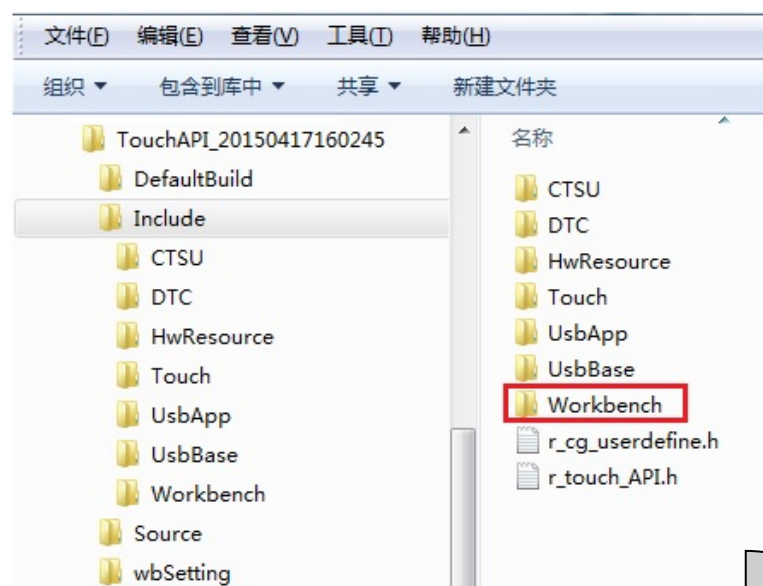
本节内容分别说明利用上述两种方法连接时，必要程序的追加步骤以及综合开发环境 CS+的设定。

注： 仅适用于内置 USB 功能的 MCU。对于无 USB 功能的 MCU，不可将 USB 程序导入用户项目。
此外，Touch 处理程序中包含的 USB 处理程序，使用了 IRQ0 引脚。因此，如果用户系统也使用了 IRQ0 引脚，那么不可将 USB 处理程序导入用户项目中。

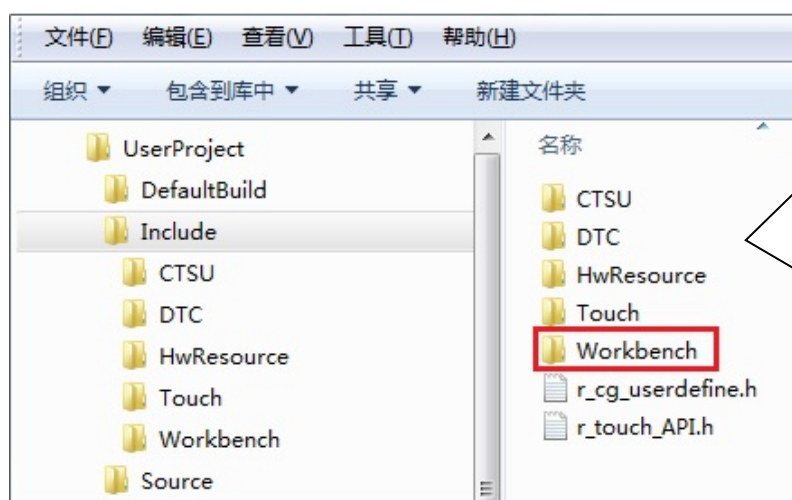
5.1 通过综合开发环境 CS+确认动作的设定步骤

- 1) 将 Workbench6 生成的 Include 文件夹中的 Workbench 文件夹，拷贝到用户项目的 Include 文件夹中。

Workbench 生成的文件夹 Include



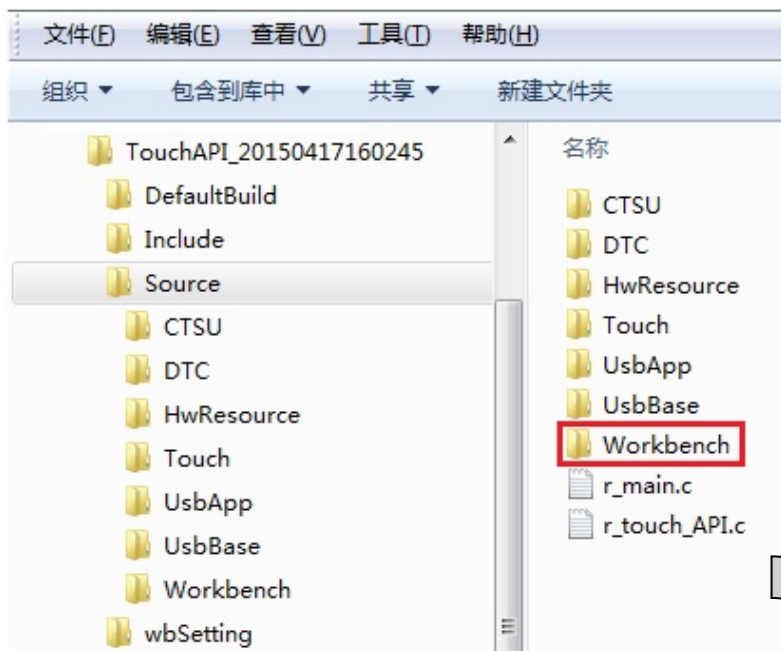
用户项目的文件夹 Include



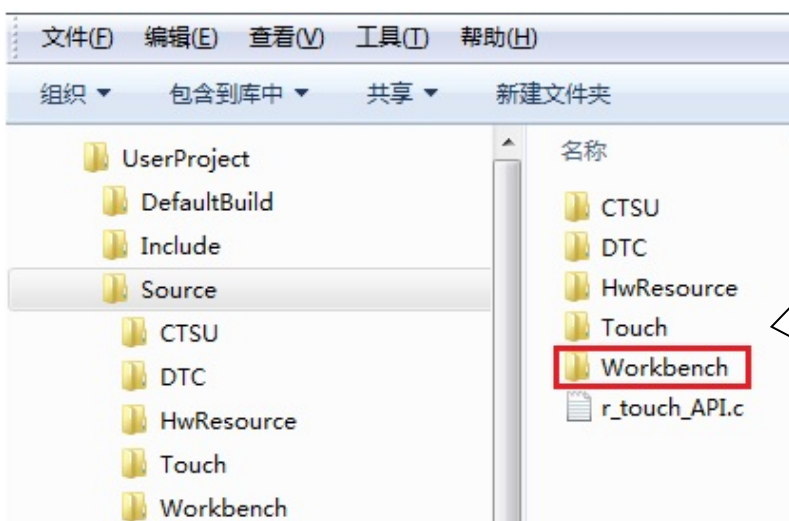
文件夹拷贝

- 2) 将 Workbench6 生成的 Source 文件夹中的 Workbench 文件夹，拷贝到用户项目的 Source 文件夹中。

Workbench 生成的文件夹 Source

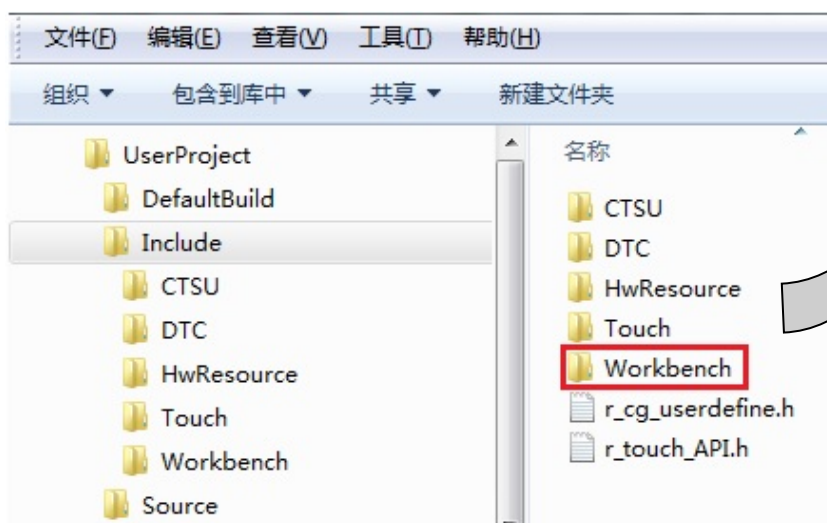
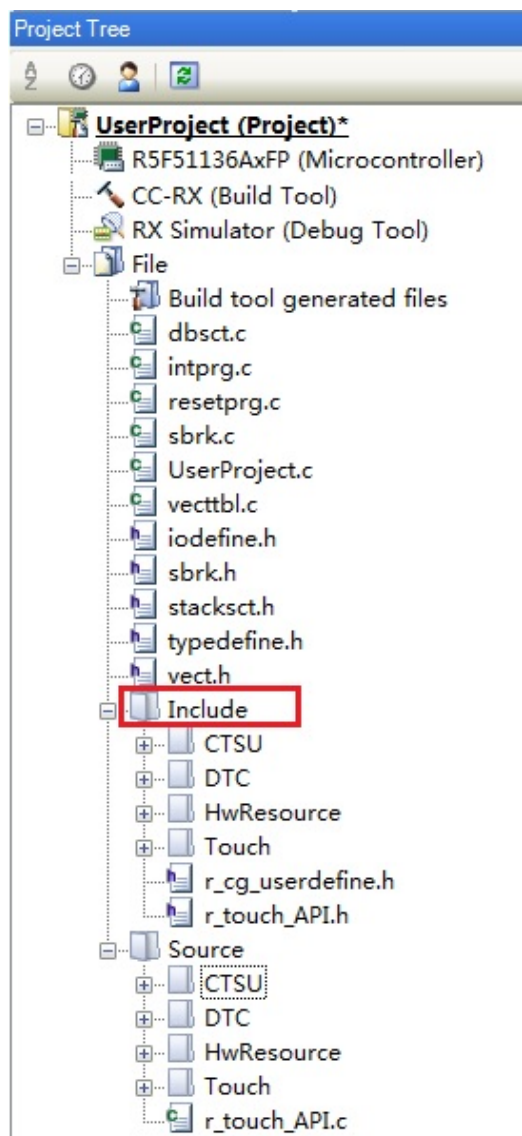


用户项目中的文件夹 Source



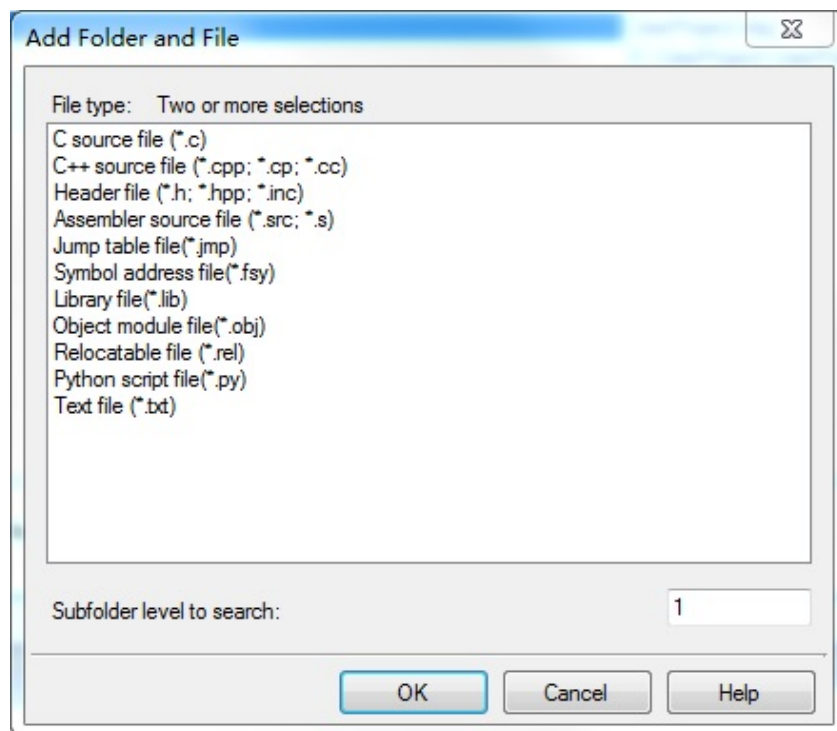
拷贝文件夹

- 3) 将以上拷贝的文件夹添加到综合开发环境 CS+中。
- 将 Include\Workbench 文件夹从资源管理器中拖放至综合开发环境 CS+中。

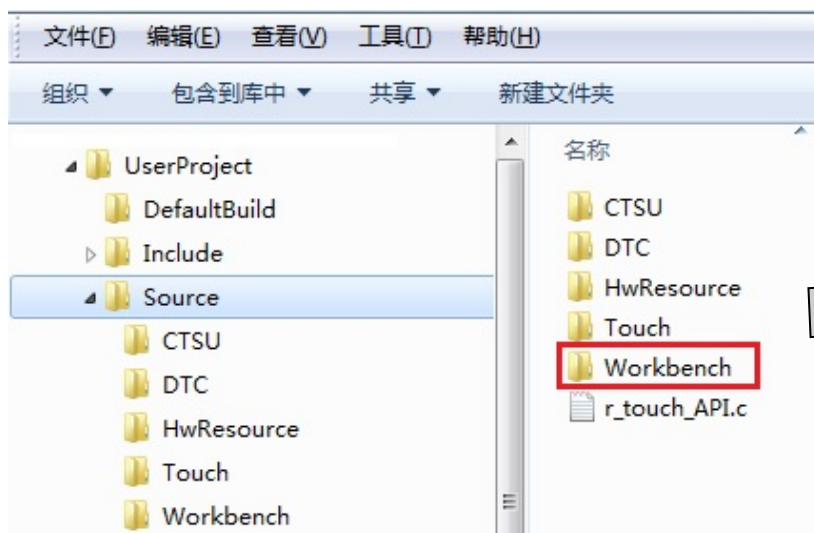
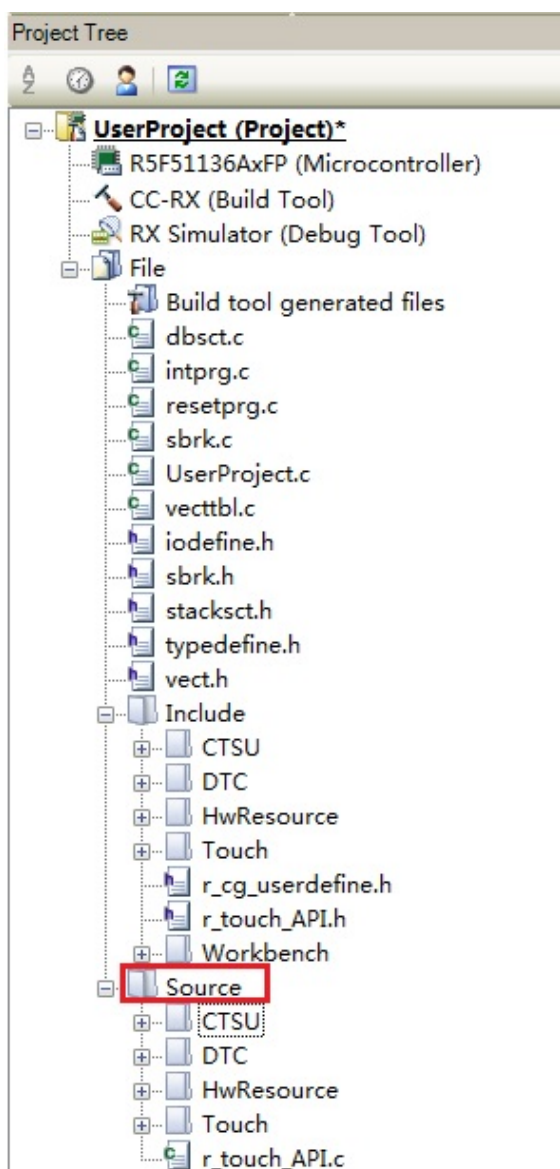


拖放操作

- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。

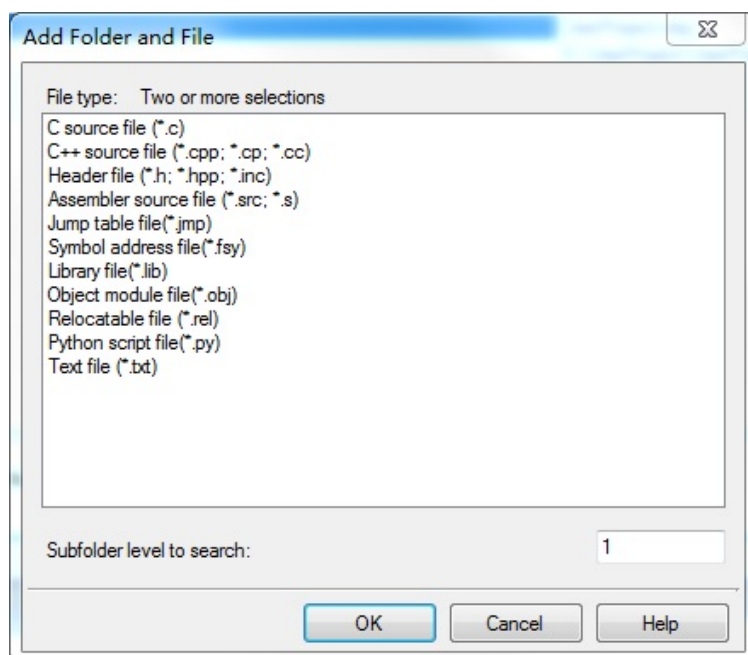


- 将 Source\Workbench 文件夹从资源管理器中拖放至综合开发环境 CS+中。

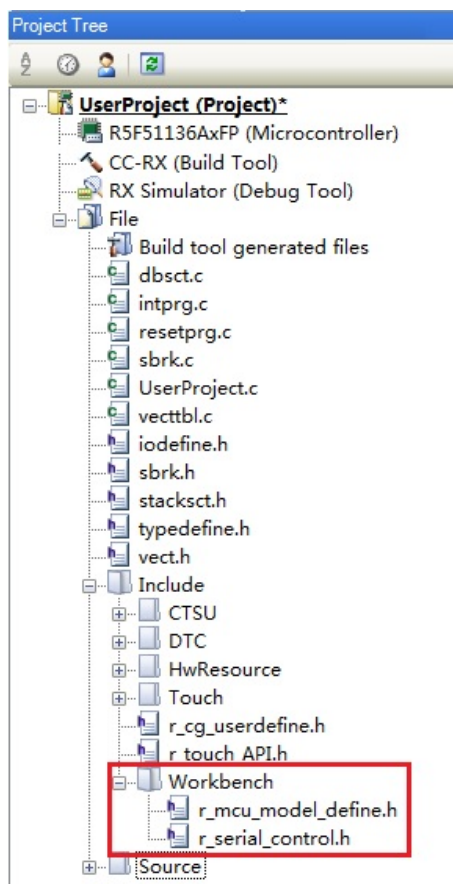


拖放操作

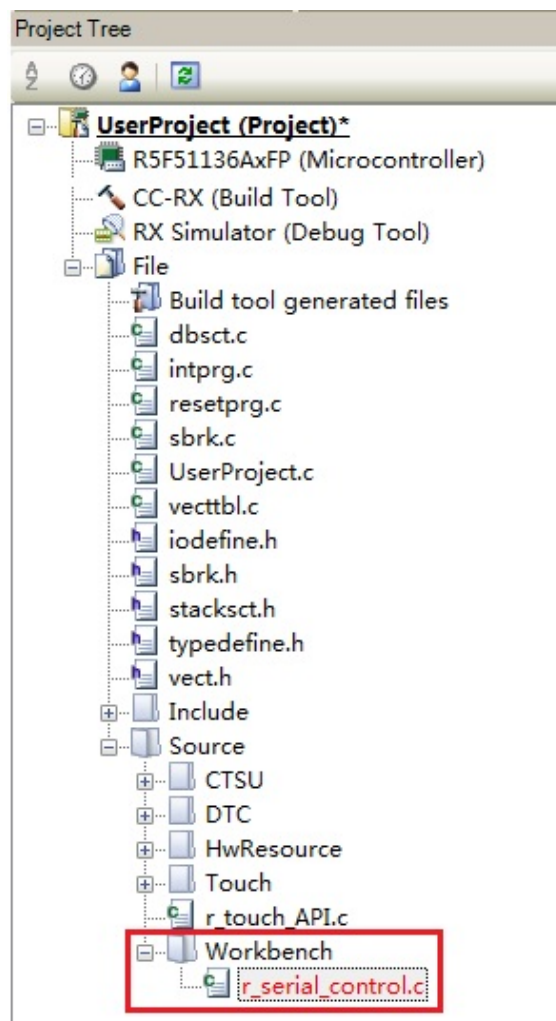
- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。



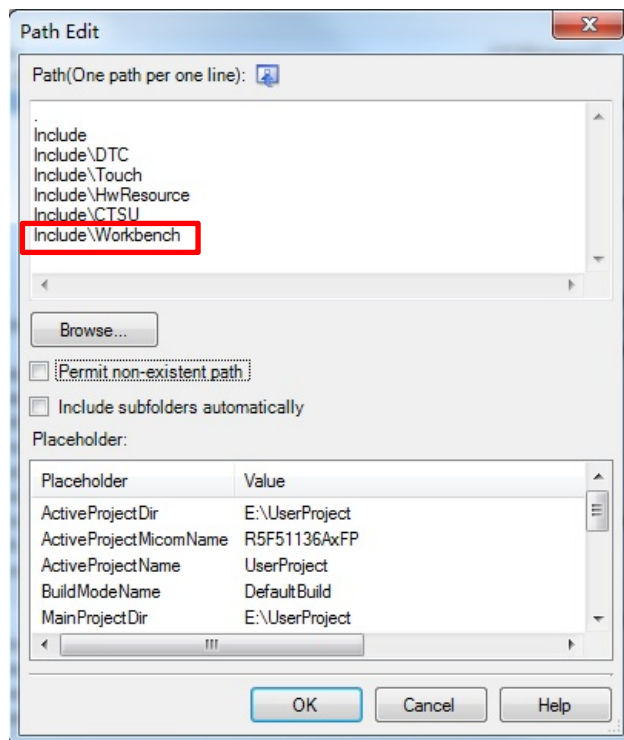
- 确认拷贝到 Include 文件夹中的头文件，是否添加到了 CS+ IDE 中。



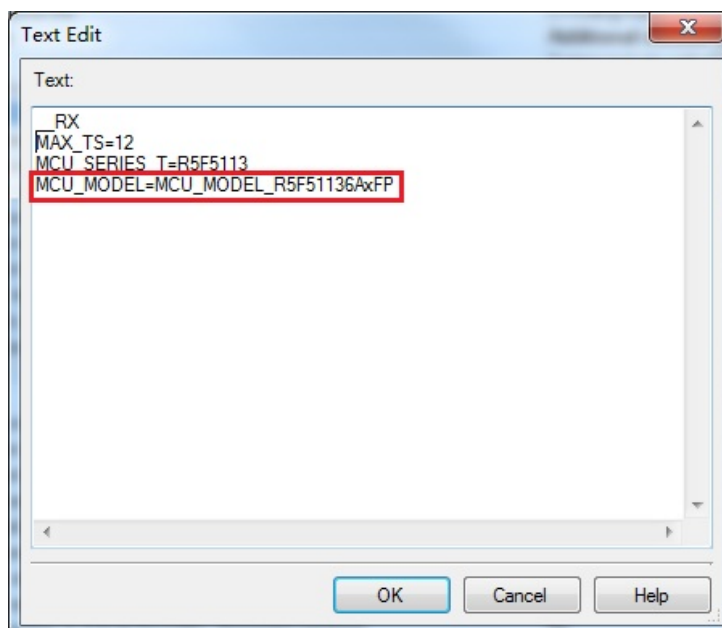
- 确认拷贝到 Source 文件夹中的源程序，是否添加到了 CS+ IDE 中。



- 4) 在综合开发环境 CS+中，设定编译选项。
- 打开 CC-RX 的「Property」窗口，单击「Common Options」选项卡，进入「Additional include paths」，确认是否存在 Include\Workbench 路径。

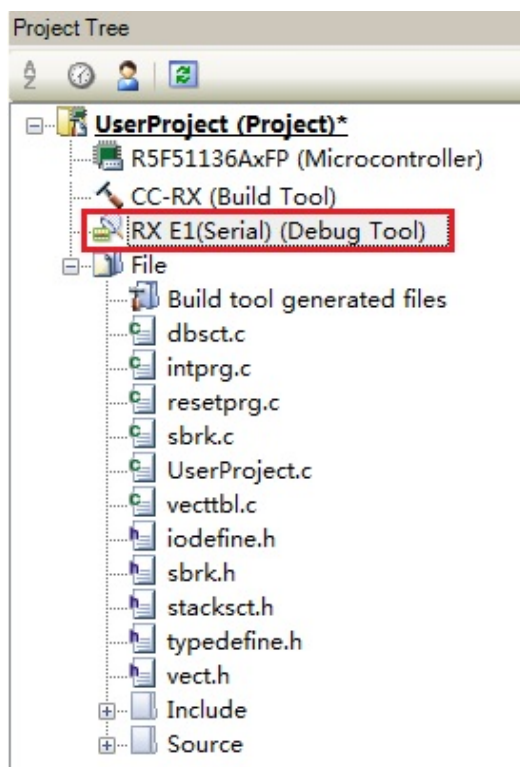


- 在 CC-RX 的「Property」窗口中，通过「Common Options」>>「Macro definition」，设定 MCU_MODEL 的值。
MCU_MODEL 的设定值，根据所用 MCU 的型号，在文件 Include\Workbench\r_mcu_model_define.h 中选择。

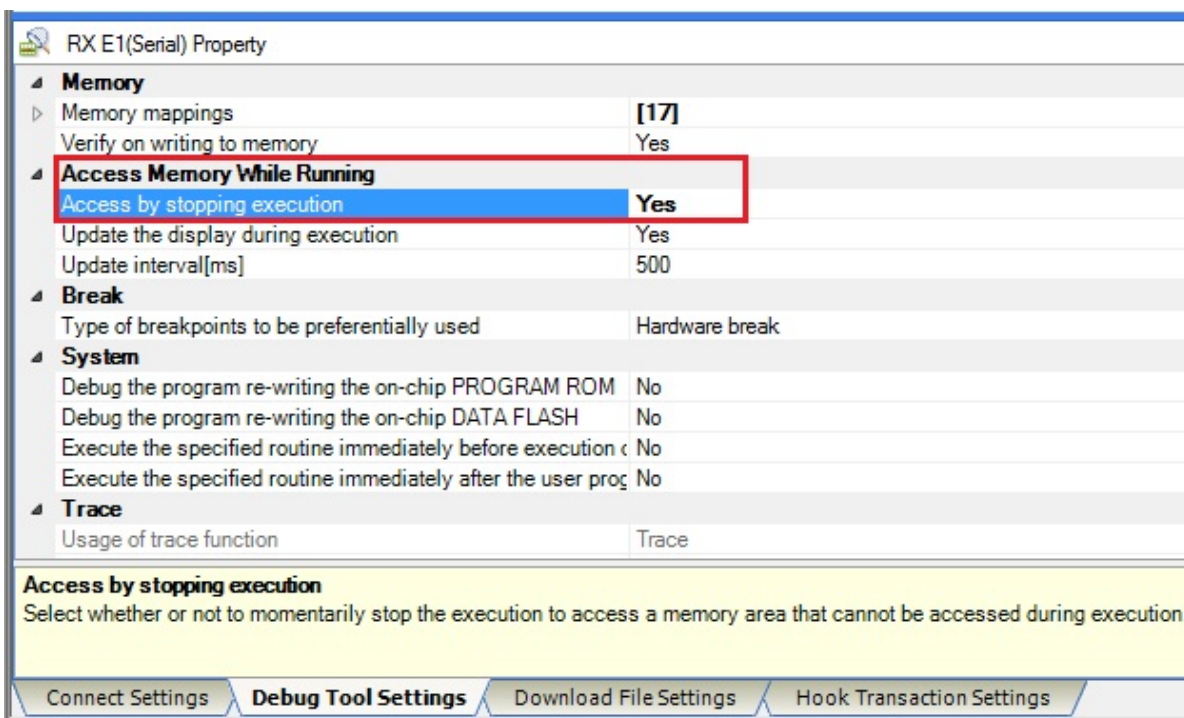


注： 利用 Workbench6 生成的项目，可以确认 MCU_MODEL 的值。

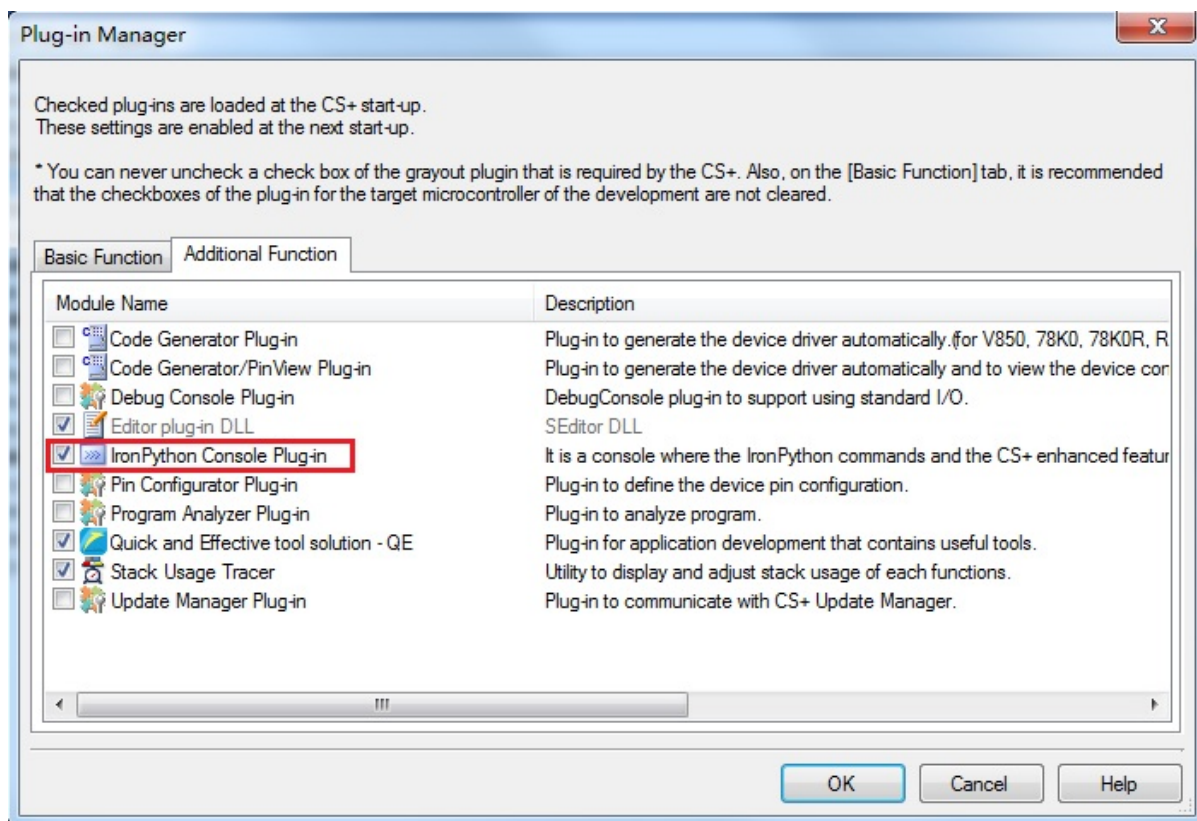
- 5) 在综合开发环境 CS+中，设定 Debug Tool。
- 确认所用调试工具是否为 RX E1(Serial)。如果不是 RX E1(Serial)，设定为 RX E1(Serial)。



- 在 Debut Tool 的「Property」窗口中，将「Debug Tool Settings」>>「Access Memory While Running」>>「Access by stopping execution」设定为「Yes」。



- 6) 在综合开发环境 CS+ 中，设定 IronPython Console Plug-in。
- 选择菜单「Tool」>>「Plug-in Setting」，显示「Plug-in Manager」窗口，选择「Additional Function」选项卡中「IronPythonConsole Plug-in」，然后单击「OK」。



- 7) 编辑用户项目的主程序。
- 追加以下的 2 个头文件包含指令。

(省略)

```
#include "r_serial_control.h"
#include "r_touch.h"
```

追加内容

(省略)

- 追加用于和综合开发环境 CS+通讯的初始化函数和通信处理函数。

(省略)

```
void main(void)
{
    uint8_t method;

    method = 0U;
    R_Set_Cap_Touch_Create(method);

    g_ctsu_soft_mode = CTSU_READY_MODE;

    R_CMT0_Create();
    R_CMT0_Start();

    SerialCommandInitial(); /* Initialize serial command communication task */

    while(1U)
    {
        R_Set_Cap_Touch_measurement_start(method);

        if (GET_OK == R_Get_Cap_Touch_Data_Check(method))
        {
            if (INITIAL_END == R_Get_Cap_Touch_Initial_status())
            {
                if (GET_OK == R_Get_Cap_Touch_Function_Data(method))
                {
                }
            }
            else
            {
                R_Set_Cap_Touch_Initial_Tuning(method);
            }
        }
        method = R_Set_Cap_Touch_Next_Method_Change(method);

        PrepareReplayMessage(); /* Make the replay message of the serial command from Workbench */
    }
}
```

追加内容

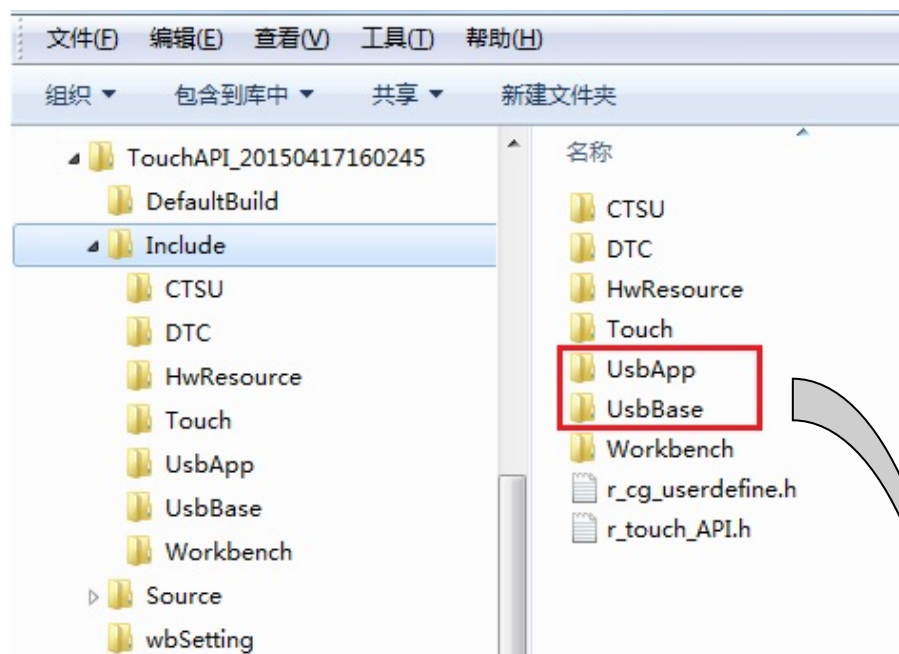
追加内容

(省略)

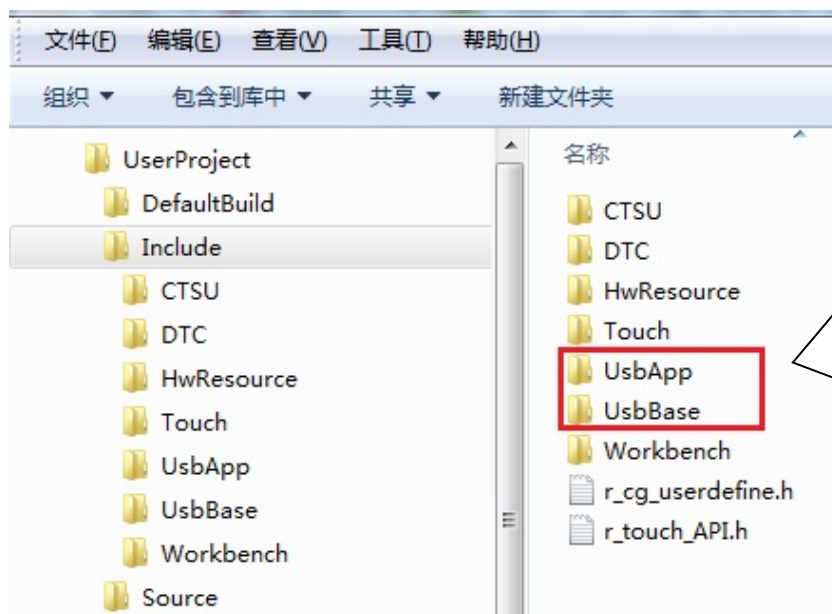
5.2 通过 USB 确认动作的设定步骤

- 1) 执行“5.1 通过综合开发环境 CS+确认动作的设定步骤”中的设定内容。
- 2) 将 Workbench6 生成的 Include 文件夹中的 UsbApp 和 UsbBase 文件夹，拷贝至用户项目的 Include 文件夹中。

Workbench 生成的文件夹 Include



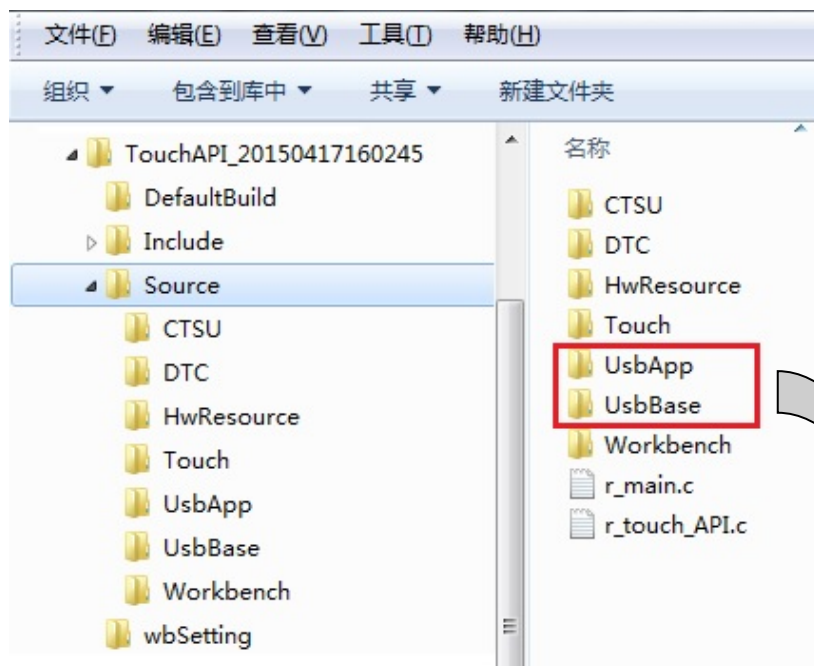
用户项目中的文件夹 Include



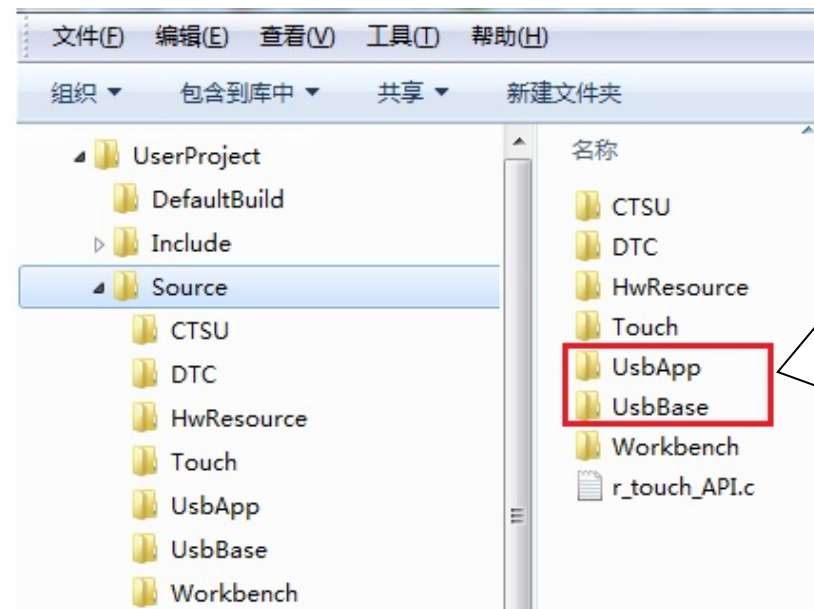
拷贝文件夹

- 3) 将 Workbench6 生成的源程序拷贝至用户项目中。
- 将 Workbench6 生成的 Souce 文件夹中的 UsbApp 和 UsbBase 文件夹，拷贝至用户项目的 Source 文件夹中。

Workbench 生成的文件夹 Source



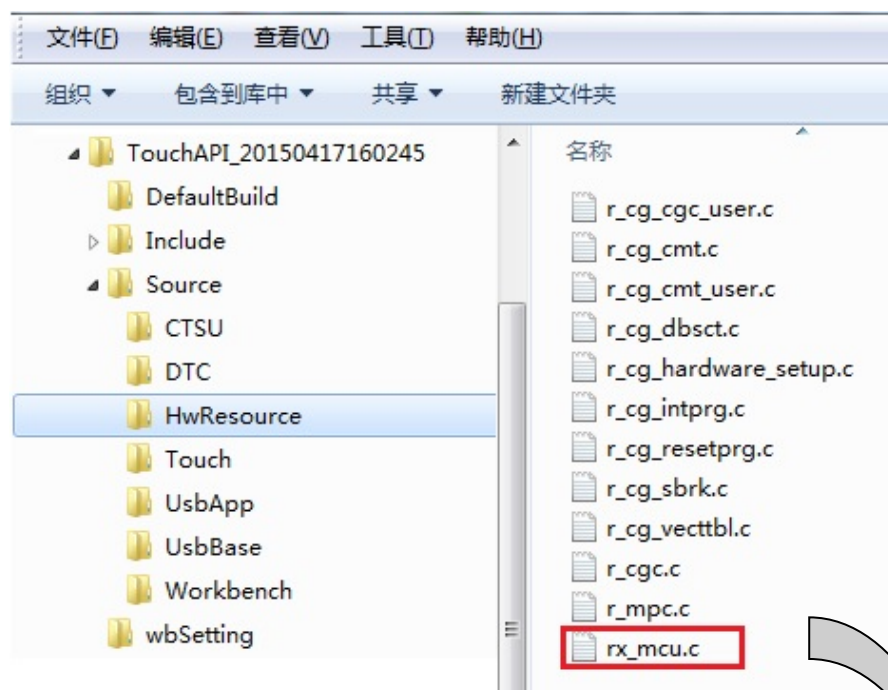
用户项目中的文件夹 Source



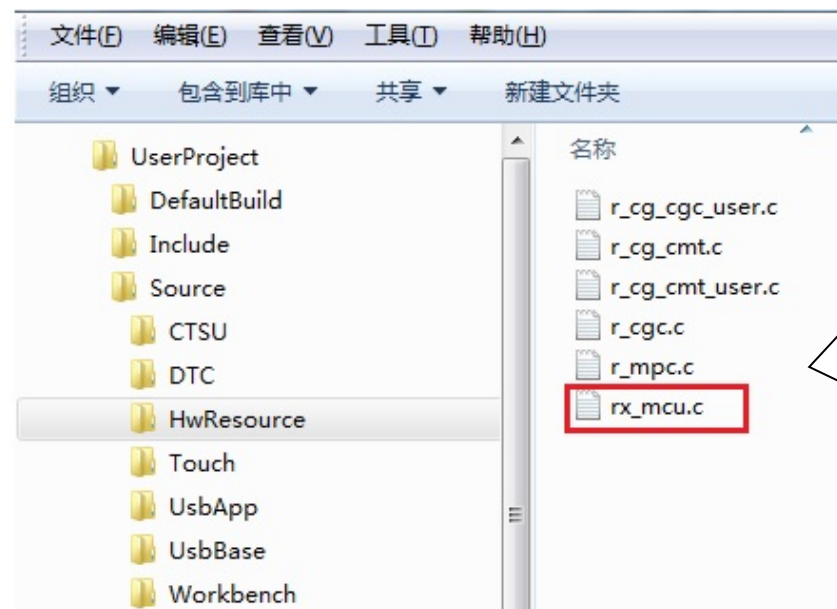
拷贝文件夹

- 将 Workbench6 生成的 Souce\HwResource\rx_mcu.c 源程序，拷贝至用户项目的 Source\HwResource 文件夹中。

Workbench 生成的文件夹 Source\HwResource

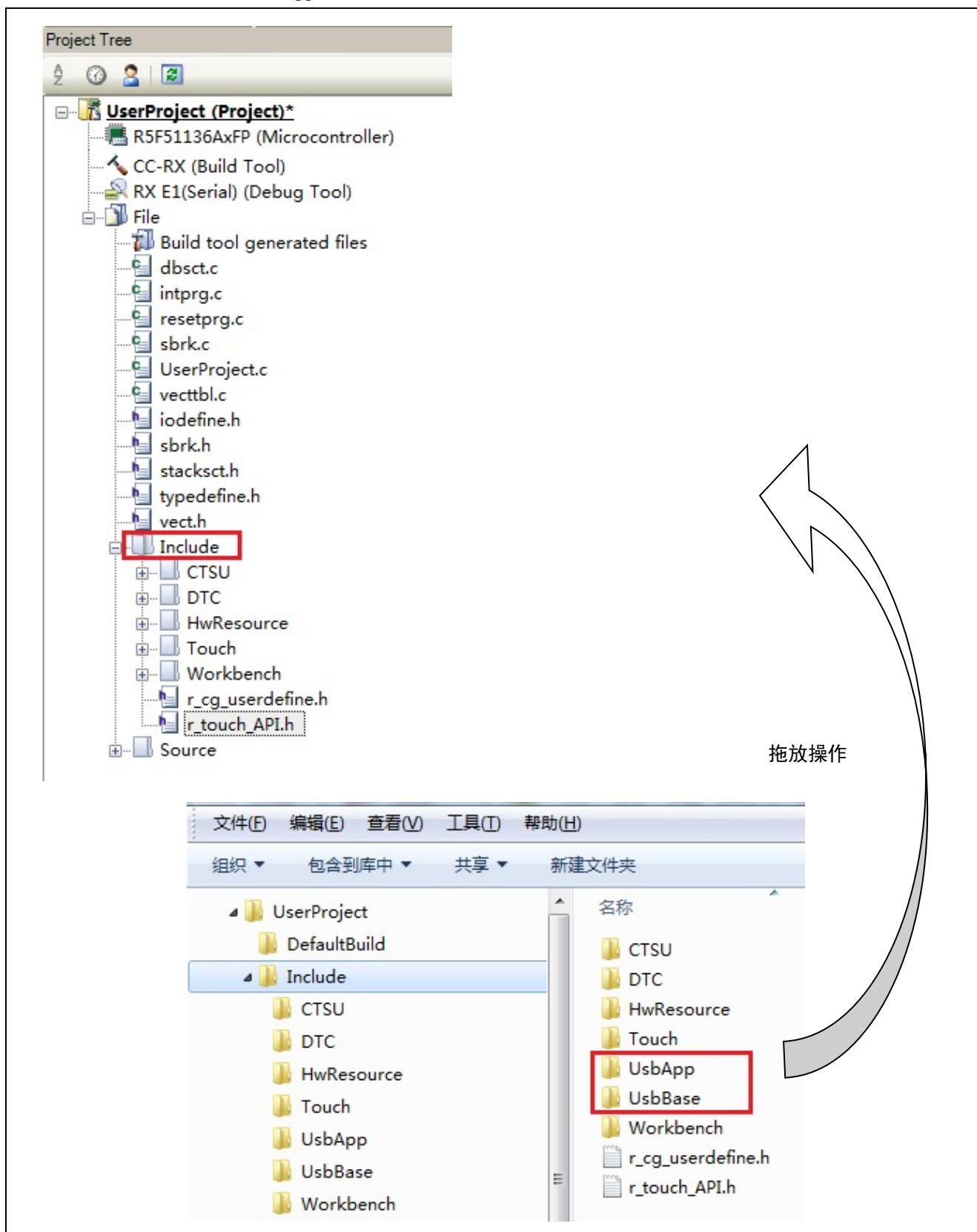


用户项目中的文件夹 Source\HwResource

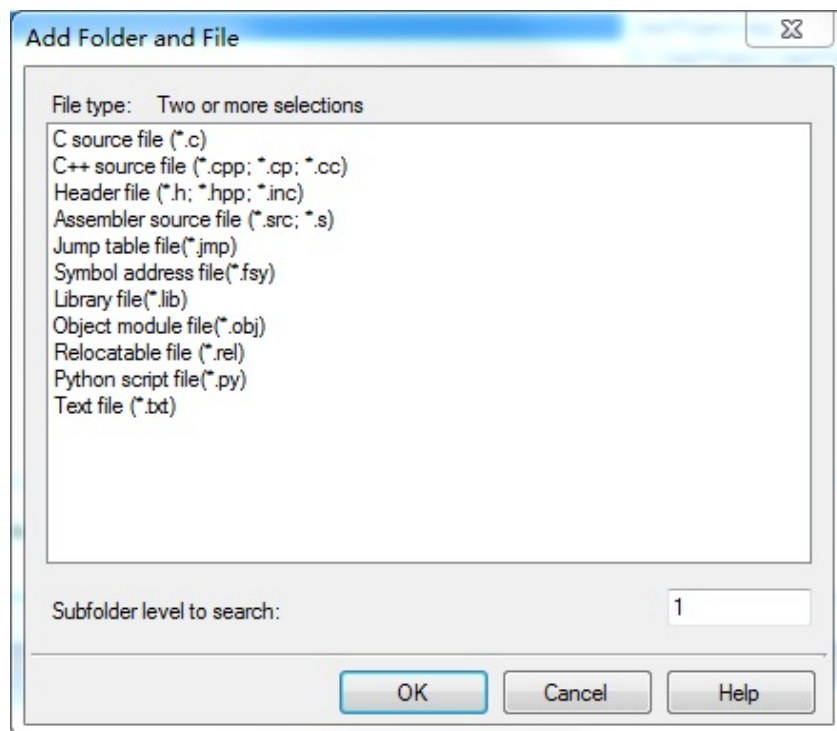


拷贝文件

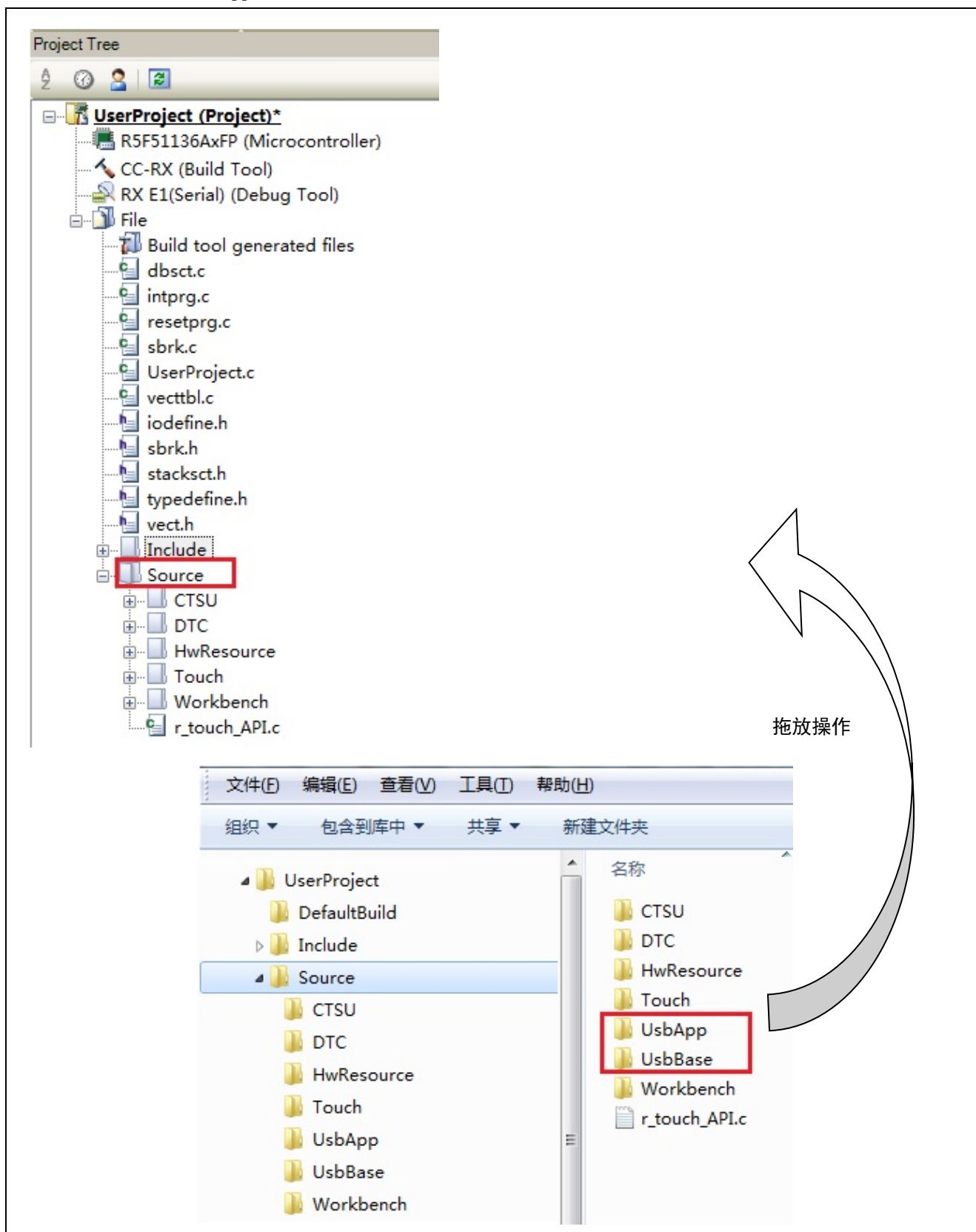
- 4) 将以上拷贝的文件夹添加到综合开发环境 CS+中。
- 将文件夹 Include\UsbApp 和 Include\UsbBase, 从资源管理器中拖放至综合开发环境 CS+中。



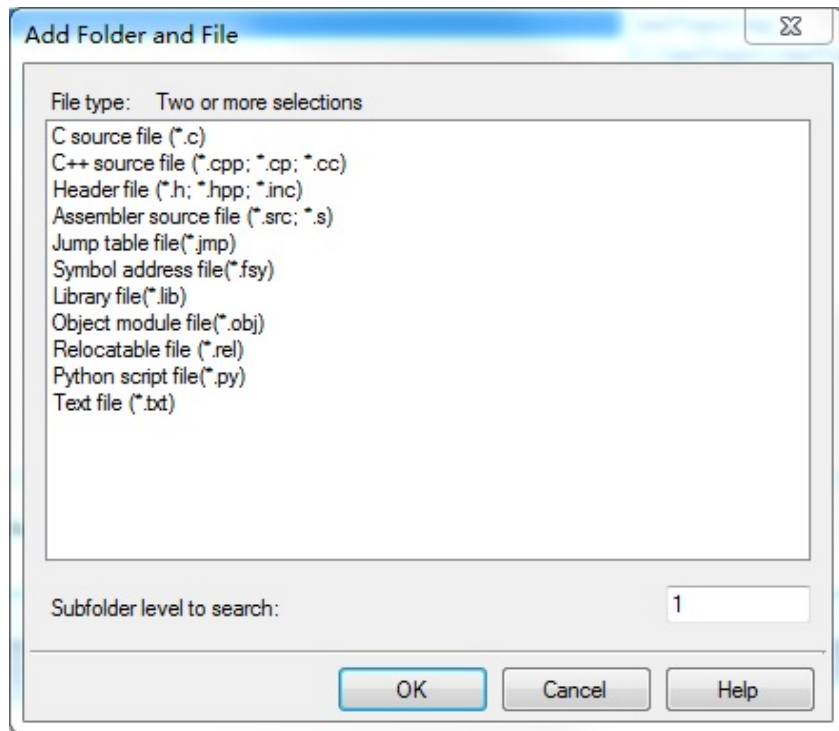
- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。



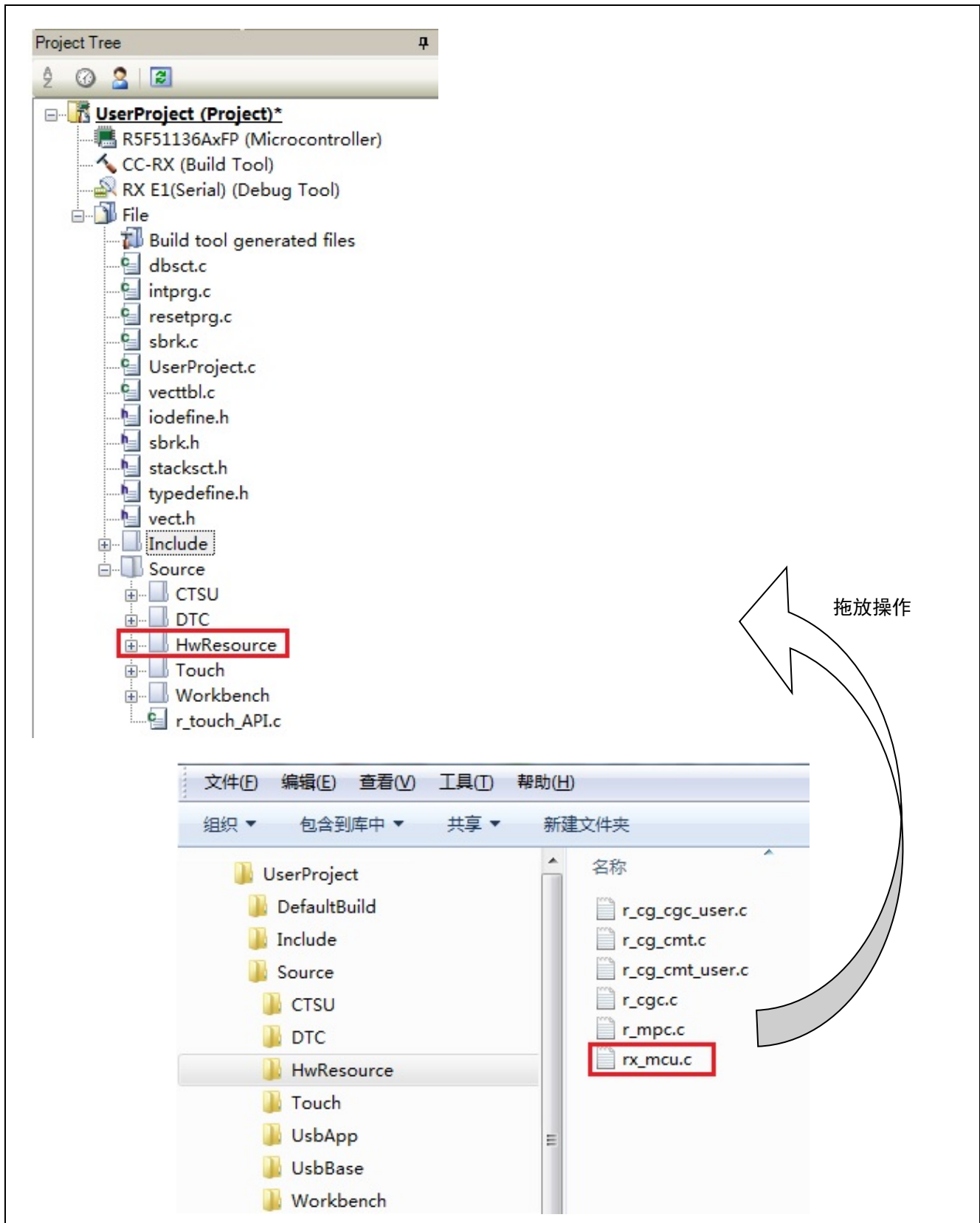
- 将 Source\UsbApp 和 Source\UsbBase 文件夹，从资源管理器中拖放至综合开发环境 CS+中。



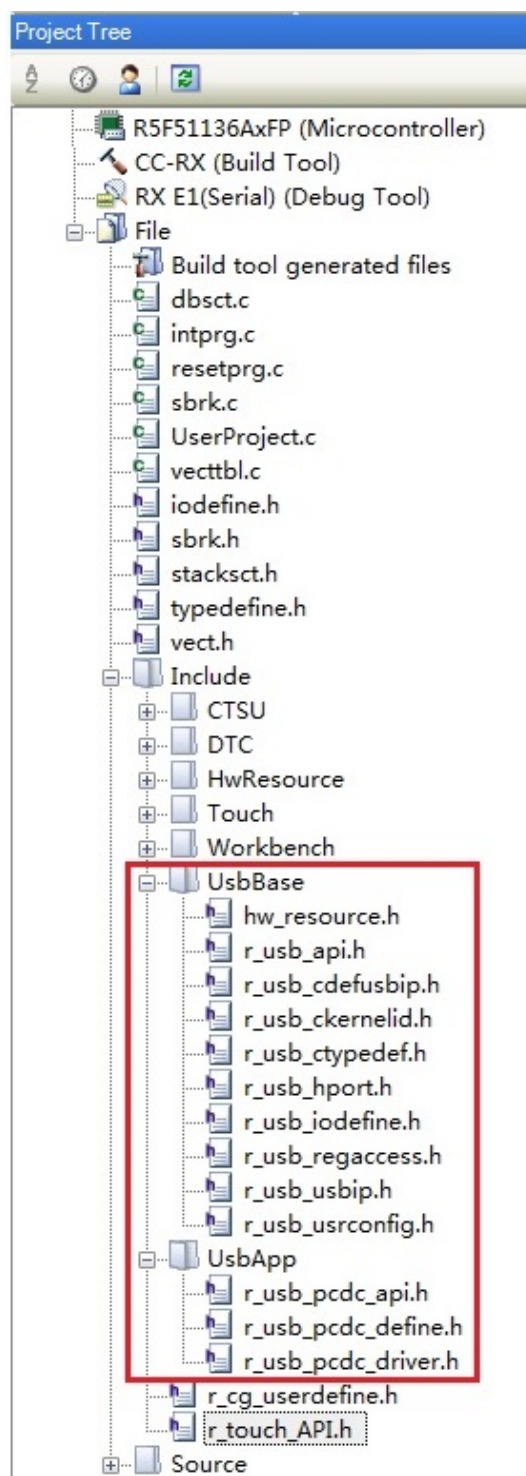
- 拖放后，弹出「Add Folder and File」窗口，单击「OK」键关闭该窗口。



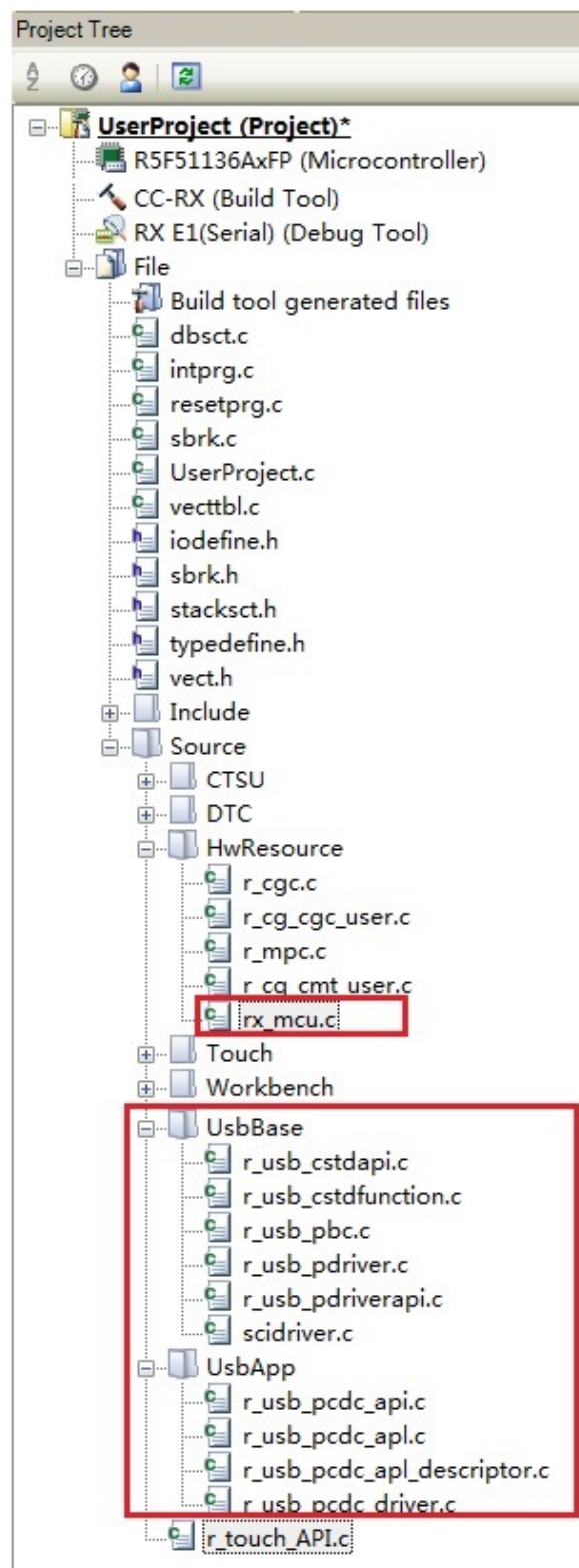
- 将 Source\HwResource\rx_mcu.c 源程序，从资源管理器中拖放至综合开发环境 CS+ 中。



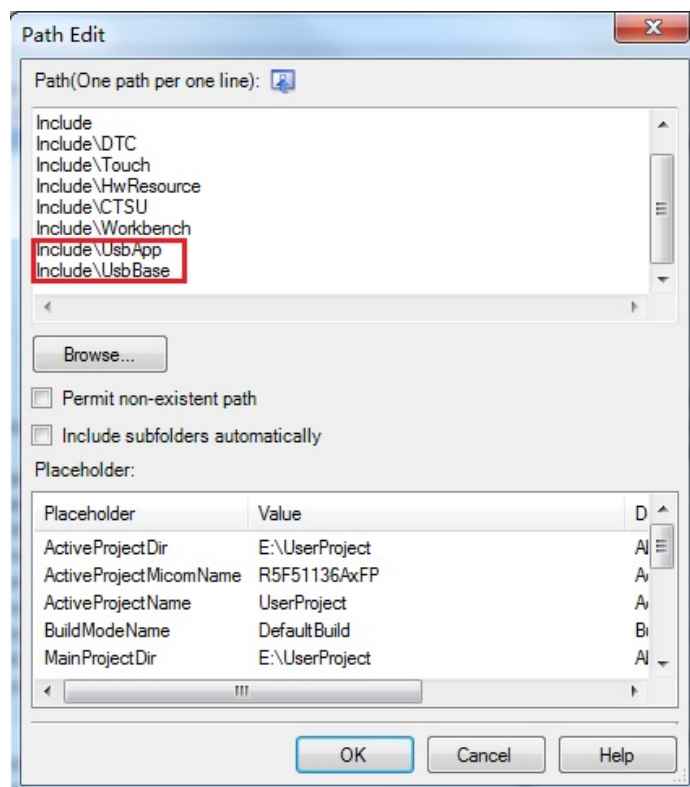
- 确认拷贝到 Include 文件夹中的头文件，是否添加到了 CS+ IDE 中。



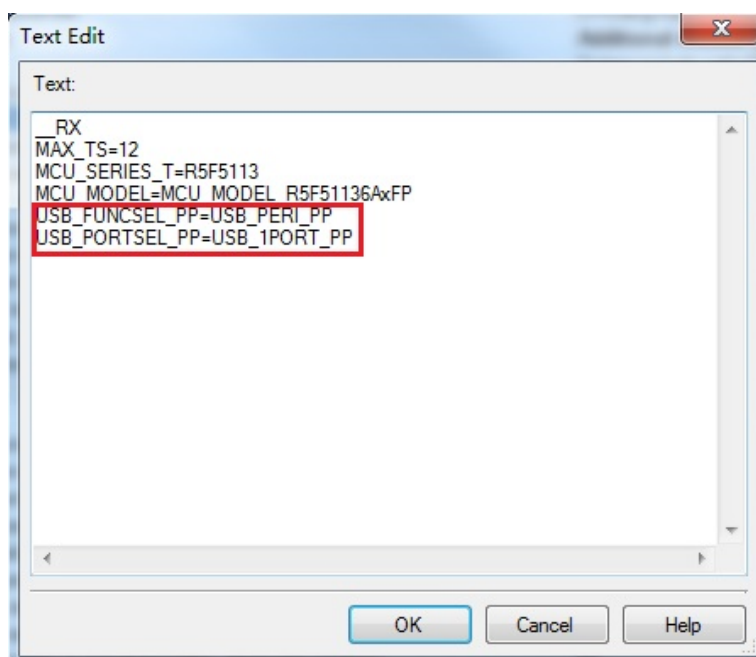
- 确认拷贝到 Source 文件夹中的源程序，是否添加到了 CS+ IDE 中。



- 5) 在综合开发环境中，设定编译选项。
- 打开 CC-RX 的「Property」窗口，单击「Common Options」选项卡，进入「Additional include paths」，确认是否存在 Include\UsbApp 和 Include\UsbBase 路径。



- 在 CC-RX 的「Property」窗口中，通过「Common Options」>>「Macro definition」，将 USB_FUNCSEL_PP 和 USB_PORTSEL_PP 分别设定为 USB_PERI_PP 和 USB_1PORT_SEL。



- 6) 编辑用户项目的主程序。
- 包含以下红框内所示的头文件。

(省略)

```
#include "r_cgc.h"
#include "r_mpc.h"

#include "r_cg_userdefine.h"
#include "r_ctsu.h"
#include "r_usb_ctypedef.h"           /* Type define */
#include "r_usb_kernelid.h"         /* Kernel ID definition */
#include "r_usb_usrconfig.h"        /* System definition */
#include "r_usb_cdefusbip.h"        /* USB-FW Library Header */
#include "r_usb_usrconfig.h"        /* System definition, USB-FW Library Header */
#include "r_usb_api.h"              /* USB API public function header */

#include "r_usb_pcdc_driver.h"
#include "r_usb_pcdc_define.h"
#include "r_usb_pcdc_api.h"
```

追加内容

(省略)

- 声明变量和函数原型。

(省略)

```
USB_STATIC volatile uint8_t g_usb_suspend_flag;
USB_STATIC volatile uint8_t g_usb_int_key_flag;

void      R_USBPcdc_Create(void);
void      R_USBPcdc_Apl_Task_Switch(void);
void      usb_smpl_set_suspend_flag(uint8_t data);
uint8_t   usb_smpl_get_suspend_flag(void);
void      usb_smpl_set_intkey(uint8_t data);
uint8_t   usb_smpl_get_intkey(void);

extern void usb_cpu_target_init(void);
extern void usb_cstd_task_start(void);
extern void usb_pcdc_task_start(void);
extern void usb_apl_task_switch(void);
```

追加内容

(省略)

- 在主程序中追加 USB 处理程序。

(省略)

```
void main(void)
{
    uint8_t method;

    method = 0U;
    R_Set_Cap_Touch_Create(method);

    g_ctsu_soft_mode = CTSU_READY_MODE;

    R_USBPcdc_Create();           追加内容
    R_CMT0_Create();
    R_CMT0_Start();

    SerialCommandInitial(); /* Initialize serial command communication task */

    while(1U)
    {
        R_USBPcdc_Apl_Task_Switch(); /* Switch task for nonOS */ 追加内容
        R_Set_Cap_Touch_measurement_start(method);
    }
}
```

(省略)

- 在主程序中追加用于 USB 处理的相关函数。
需要追加的函数，如下所示。

(省略)

```
void R_USBPDCD_Create(void)
{
#ifdef USB_SERIAL_USE
    SYSTEM_PRCR_WORD = PRCR1_ENA;      /* Enables writing to the registers */
    SYSTEM_MSTPCRB.BIT.MSTPB19 = 0;    /* Module stop control register (Enable USB0 module(MSTPB19)) */
    SYSTEM_PRCR_WORD = PRCR1_DIS;      /* Disables writing to the registers */
    usb_cpu_target_init();
    usb_smpl_set_suspend_flag(USB_NO);
    R_usb_pstd_PcdChangeDeviceState(USB_DO_INITHWFUNCTION);
    R_usb_pstd_PcdOpen();
    usb_pcdc_task_start();              /* Start Peripheral USB driver */
#endif // USB_SERIAL_USE
}

void R_USBPDCD_Apl_Task_Switch(void)
{
#ifdef USB_SERIAL_USE
    usb_apl_task_switch();              /* Switch task for nonOS */
#endif // USB_SERIAL_USE
}

#ifdef USB_SERIAL_USE
void usb_smpl_set_suspend_flag(uint8_t data)
{
    g_usb_suspend_flag = data;
} /* End of function usb_smpl_set_suspend_flag() */

uint8_t usb_smpl_get_suspend_flag(void)
{
    return g_usb_suspend_flag;
} /* End of function usb_smpl_get_suspend_flag() */

void usb_smpl_set_intkey(uint8_t data)
{
    g_usb_int_key_flag = data;
} /* End of function usb_smpl_set_intkey() */

uint8_t usb_smpl_get_intkey(void)
{
    return g_usb_int_key_flag;
} /* End of function usb_smpl_get_intkey() */
#endif // USB_SERIAL_USE
```

追加内容

(省略)

- 编辑用户项目中包含中断处理函数的源程序。下面示例中所编辑的源程序，是 CS+自动生成的 intprg.c 文件。
- 屏蔽 D0FIFO0 中断处理函数。
- 屏蔽 USBI0 中断处理函数。
- 屏蔽 IRQ0 中断处理函数。
- 屏蔽 USBR0 中断处理函数。

(省略)

```
// USB0 D0FIFO0  
//void Excep_USB0_D0FIFO0(void) { }
```

屏蔽处理

```
// USB0 D1FIFO0  
void Excep_USB0_D1FIFO0(void) { }
```

```
// USB0 USBI0  
//void Excep_USB0_USBI0(void) { }
```

屏蔽处理

(省略)

```
// ICU IRQ0  
//void Excep_ICU_IRQ0(void) { }
```

屏蔽处理

(省略)

```
// USB0 USBR0  
//void Excep_USB0_USBR0(void) { }
```

屏蔽处理

(省略)

- 7) 如果用户系统中使用了 DTC 功能, 请进行如下处理。
- 在 Source\HwResource\rx_mcu.c 源程序中, 根据用户系统的设定, 变更各设定值。
 - 下图示例中, 用户系统使用的向量地址为 0000 8000H。

(省略)

```
// #pragma address dtc_vector36=0x00007C90U  
#pragma address dtc_vector36=0x00008090U  
uint32_t dtc_vector36;  
dtc_register_data_t      usb_dtcreg;
```

变更内容

(省略)

```
void usb_cpu_DmaintInit(void)  
{  
    /* Enable DTC module(MSTPA28) Stop  
    b8-b0   Reserved 0  
    b9      MSTPA9   Multifunction timer unit0 stop bit  
    b14-10  Reserved 0  
    b15     MSTPA15  Compare timer unit0 stop bit  
    b16     Reserved 0  
    b17     MSTPA17  12bit AD stop bit  
    b18     Reserved 0  
    b19     MSTPA19  DA stop bit  
    b27-20  Reserved 0  
    b28     MSTPA28  DTC stop bit  
    b31-29  Reserved 0  
    */  
    SYSTEM.MSTPCRA.BIT.MSTPA28 = 0;  
  
    dtc_vector36 = (uint32_t)&usb_dtcreg;  
  
    /* DTC vector register  
    b21-b0 DTCVBR Vector table address  
    */  
    // DTC.DTCVBR = (void *)&dtc_vect_table;  
    // DTC.DTCVBR = (void *)0x00007C00U;  
    DTC.DTCVBR = (void *)0x00008000U;
```

变更内容

(省略)

6. 注意事项

6.1 时钟的设置

Touch 处理程序使用外围模块时钟 B (PCLKB)。外围模块时钟 B 的频率，必须设定为和 Workbench6 生成项目相同的频率。

如果用户系统所用时钟不同于 Workbench6 生成程序中的时钟设定，必须将用户系统时钟变更为 Workbench6 生成程序中所设定的时钟。

6.2 Touch 处理程序

Touch 处理程序应以 2ms 为周期循环执行。

Touch 处理程序中使用了 DTC 功能。如果用户系统也使用了 DTC 功能，需要注意 DTC 向量表中所配置的地址。

6.3 和 Workbench6 的连接

所用 MCU 无内置 USB 功能时，不可执行“5.2 通过 USB 确认动作的设定步骤”中的内容。

用户系统中使用 IRQ0 引脚时，不可执行“5.2 通过 USB 确认动作的设定步骤”中的内容。

用户系统使用了 DTC 功能，并且执行“5.2 通过 USB 确认动作的设定步骤”的内容时，需要注意 DTC 向量表的配置地址。

7. 参考文献

RX113 群 用户手册 硬件篇 Rev.1.02 (R01UH0448J)

(最新版本请从瑞萨电子网页上取得)

技术信息/技术更新

(最新信息请从瑞萨电子网页上取得)

公司主页和咨询窗口

瑞萨电子主页

- <http://www.renesas.com/zh-cn/>

咨询

- <http://www.renesas.com/zh-cn/support/contact.html>

修订记录

Rev.	发行日	修订内容	
		页	要点
1.00	2016.12.31	—	初版发行

所有商标及注册商标均归其各自拥有者所有。

产品使用时的注意事项

本文对适用于单片机所有产品的“使用时的注意事项”进行说明。有关个别的使用时的注意事项请参照正文。此外，如果在记载上有与本手册的正文有差异之处，请以正文为准。

1. 未使用的引脚的处理

【注意】将未使用的引脚按照正文的“未使用引脚的处理”进行处理。

CMOS产品的输入引脚的阻抗一般为高阻抗。如果在开路的状态下运行未使用的引脚，由于感应现象，外加LSI周围的噪声，在LSI内部产生穿透电流，有可能被误认为是输入信号而引起误动作。未使用的引脚，请按照正文的“未使用引脚的处理”中的指示进行处理。

2. 通电时的处理

【注意】通电时产品处于不定状态。

通电时，LSI内部电路处于不确定状态，寄存器的设定和各引脚的状态不定。通过外部复位引脚对产品进行复位时，从通电到复位有效之前的期间，不能保证引脚的状态。

同样，使用内部上电复位功能对产品进行复位时，从通电到达到复位产生的一定电压的期间，不能保证引脚的状态。

3. 禁止存取保留地址（保留区）

【注意】禁止存取保留地址（保留区）

在地址区域中，有被分配将来用作功能扩展的保留地址（保留区）。因为无法保证存取这些地址时的运行，所以不能对保留地址（保留区）进行存取。

4. 关于时钟

【注意】复位时，请在时钟稳定后解除复位。

在程序运行中切换时钟时，请在要切换成的时钟稳定之后进行。复位时，在通过使用外部振荡器（或者外部振荡电路）的时钟开始运行的系统中，必须在时钟充分稳定后解除复位。另外，在程序运行中，切换成使用外部振荡器（或者外部振荡电路）的时钟时，在要切换成的时钟充分稳定后再进行切换。

5. 关于产品间的差异

【注意】在变更不同型号的产品时，请对每一个产品型号进行系统评价测试。

即使是同一个群的单片机，如果产品型号不同，由于内部ROM、版本模式等不同，在电特性范围内有时特性值、动作容限、噪声耐量、噪声辐射量等不同。因此，在变更不同型号的产品时，请对每一个型号的产品进行系统评价测试。

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

以下"注意事项"为从英语原稿翻译的中文译文，仅作参考译文，英文版的"Notice"具有正式效力。

注意事项

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件 and 相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，瑞萨电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，瑞萨电子已尽量做到合理注意，但是，瑞萨电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，瑞萨电子不承担任何责任。
3. 对于因使用本文档中的瑞萨电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，瑞萨电子不承担任何责任。本文档所记载的内容不应视为对瑞萨电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 用户不得更改、修改、复制或者以其他方式部分或全部地非法使用瑞萨电子的任何产品。对于用户或第三方因上述更改、修改、复制或其他方式非法使用瑞萨电子产品的行为而遭受的任何损失，瑞萨电子不承担任何责任。
5. 瑞萨电子产品根据其质量等级分为两个等级：“标准等级”和“高质量等级”。每种瑞萨电子产品的推荐用途均取决于产品的质量等级，如下所示：
标准等级： 计算机、办公设备、通讯设备、测试和测量设备、视听设备、家用电器、机械工具、个人电子设备以及工业机器人等。
高质量等级： 运输设备（汽车、火车、轮船等）、交通控制系统、防灾系统、预防犯罪系统以及安全设备等。
瑞萨电子产品无意用于且未被授权用于可能对人类生命造成直接威胁的产品或系统及可能造成人身伤害的产品或系统（人工生命维持装置或系统、植埋于体内的装置等）中，或者可能造成重大财产损失的产品或系统（核反应堆控制系统、军用设备等）中。在将每种瑞萨电子产品用于某种特定应用之前，用户应先确认其质量等级。不得将瑞萨电子产品用于超出其设计用途之外的任何应用。对于用户或第三方因将瑞萨电子产品用于其设计用途之外而遭受的任何损害或损失，瑞萨电子不承担任何责任。
6. 使用本文档中记载的瑞萨电子产品时，应在瑞萨电子指定的范围内，特别是在最大额定值、电源工作电压范围、移动电源电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用瑞萨电子产品而产生的故障或损失，瑞萨电子不承担任何责任。
7. 虽然瑞萨电子一直致力于提高瑞萨电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，瑞萨电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当瑞萨电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。由于难于对微机软件单独进行评估，所以请用户自行对最终产品或系统进行安全评估。
8. 关于环境保护方面的详细内容，例如每种瑞萨电子产品的环境兼容性等，请与瑞萨电子的营业部门联系。使用瑞萨电子产品时，请遵守对管制物质的使用或含量进行管理的所有相应法律法规（包括但不限于《欧盟RoHS指令》）。对于因用户未遵守相应法律法规而导致的损害或损失，瑞萨电子不承担任何责任。
9. 不可将瑞萨电子产品和技术用于或者嵌入日本国内或海外相应的法律法规所禁止生产、使用及销售的任何产品或系统中。也不可将本文档中记载的瑞萨电子产品或技术用于与军事应用或者军事用途有关的目的（如大规模杀伤性武器的开发等）。在将本文档中记载的瑞萨电子产品或技术进行出口时，应当遵守相应的出口管制法律法规，并按照上述法律法规所规定的程序进行。
10. 向第三方分销或处分产品或者以其他方式将产品置于第三方控制之下的瑞萨电子产品买方或分销商，有责任事先向上述第三方通知本文档规定的内容和条件；对于用户或第三方因非法使用瑞萨电子产品而遭受的任何损失，瑞萨电子不承担任何责任。
11. 在事先未得到瑞萨电子书面认可的情况下，不得以任何形式部分或全部转载或复制本文档。
12. 如果对本文档所记载的信息或瑞萨电子产品有任何疑问，或者用户有任何其他疑问，请向瑞萨电子的营业部门咨询。
(注1) 瑞萨电子：在本文档中指瑞萨电子株式会社及其控股子公司。
(注2) 瑞萨电子产品：指瑞萨电子开发或生产的任何产品。



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadistrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhichunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-6235-1155, Fax: +86-10-6235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Lianqiao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852-2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #05-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141