P-Touch

使用手册

第 0.01 版

2020 年 6 月 17 日

Copyright © 2020 by PADAUK Technology Co., Ltd., all rights reserved.

6F-6, No.1, Sec. 3, Gongdao 5th Rd., Hsinchu City 30069, Taiwan, R.O.C.

TEL: 886-3-572-8688 www.padauk.com.tw

重要声明

应广科技保留权利在任何时候变更或终止产品，建议客户在使用或下单前与应广科技或代理商联系以取得最新、最正确的产品信息。

应广科技不担保本产品适用于保障生命安全或紧急安全的应用，应广科技不为此类应用产品承担任何责任。关键应用产品包括，但不仅限于可能涉及的潜在风险之死亡、人身伤害、火灾或严重财产损失。

应广科技不承担任何责任来自于因客户的产品设计所造成的任何损失。在应广科技所保障的规格范围内，客户应设计和验证他们的产品。为了尽量减少风险，客户设计产品时，应保留适当的产品工作范围安全保障。

提供本文档的中文简体版是为了便于了解，请勿忽视文中英文的部份，因为其中提供有关产品性能以及产品使用的有用信息，应广科技暨代理商对于文中可能存在的差错不承担任何责任。

©Copyright 2020, PADAUK Technology Co. Ltd

目　录

1. 简介 5
2. P-Touch 的安装 5
3. P-Touch 界面介绍 7
3.1. 方案配置说明 8
3.1.1. 工程方案 8
3.1.2. 芯片类型选择 8
3.1.3. T-Watch 9
3.1.4. 触摸通道选择 9
3.1.5. 滑条模块选择 9
3.1.6. 触摸寄存器设置 9
3.1.7. 基本防噪声设置 10
3.1.8. CS 应对策略 10
3.2. 菜单说明 10
3.3. 方案设定范例 10
3.3.1. 参数设定 10
3.3.2. 打开程序 11
3.4. 编写功能 12
3.4.1. 触摸库配置文件 PT_Lib.h 12
3.4.2. 用户功能文件 User_Function.c 21
3.4.3. 用户主工程文件 26
4. T-Watch 28
4.1. T-Watch - 仿真器测试 28
4.2. 工程方案 33
5. 仿真触摸板测试 34
5.1. 打开 34
5.2. 命名&显示方式 35
5.3. 点选待测试通道 35
5.4. 生成程序 37
6. CS 测试应对办法 39
6.1. 总述 39
6.2. 分述 39
6.2.1. PCB Layout 39
6.2.2. 注意事项 39

©Copyright 2020, PADAUK Technology Co. Ltd

修订历史:

修　订	日　期	描　　述
0.00	2020/04/23	初版
0.01	2020/06/17	修改第 6.1 节

©Copyright 2020, PADAUK Technology Co. Ltd

1. 简介

P-Touch 是 PMS164/PFC161 配套的触摸程序产生器视窗型软件（如图 1-1），用于用户触摸方案的前期设置和仿真器测试，支持 T-Watch 实时显示触摸变化。其使用简单，操作便捷，可视感强，用户能够轻松高效地生成并测试目标工程方案。

P-Touch 需搭配 0.88D7（或以上）版本 IDE 使用。

用户可至官网 http://www.padauk.com.tw/cn/technical/index.aspx 下载最新版 IDE。

欢迎扫描以下二维码，加入“应广触控单片机技术讨论群”。

本文仅介绍 P-Touch 的使用方法，请参考 IDE 的使用手册了解如何使用 IDE。

图 1-1: P-Touch 图标

2. P-Touch 的安装

注意：在执行下载安装程序前，部分杀毒软件可能会误判，请在安装完成前关闭杀毒软件，安装完成后可以重新打开。

步骤 1：下载软件

请从应广官网下载最新版软件使用：http://www.padauk.com.tw/cn/technical/index.asp

步骤 2：选择安装地址。

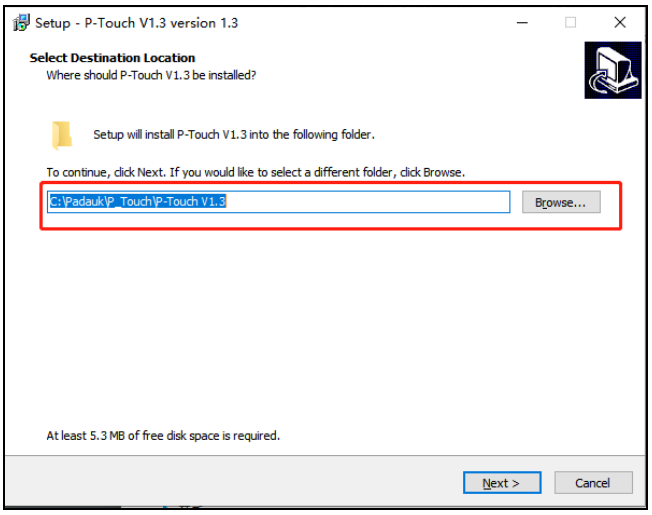


图 2-1: P-Touch 安装画面

步骤 3：确认安装信息，点击“Next”→ “Next” →“Install”安装软件。

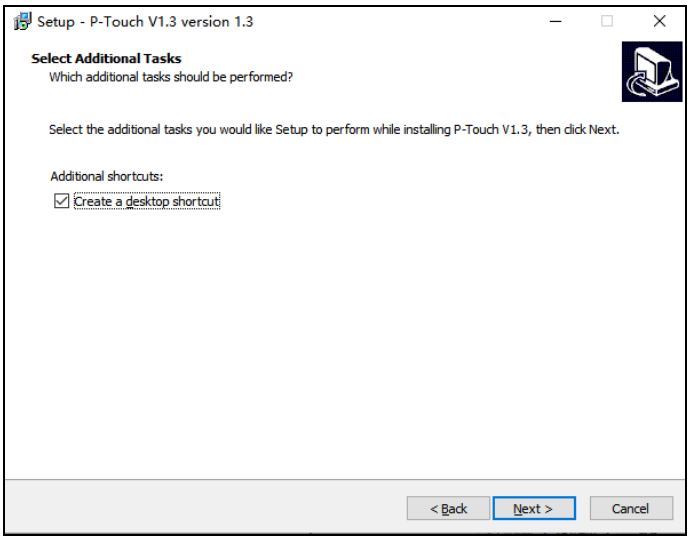


图 2-2: P-Touch 安装画面

步骤 4：完成安装。

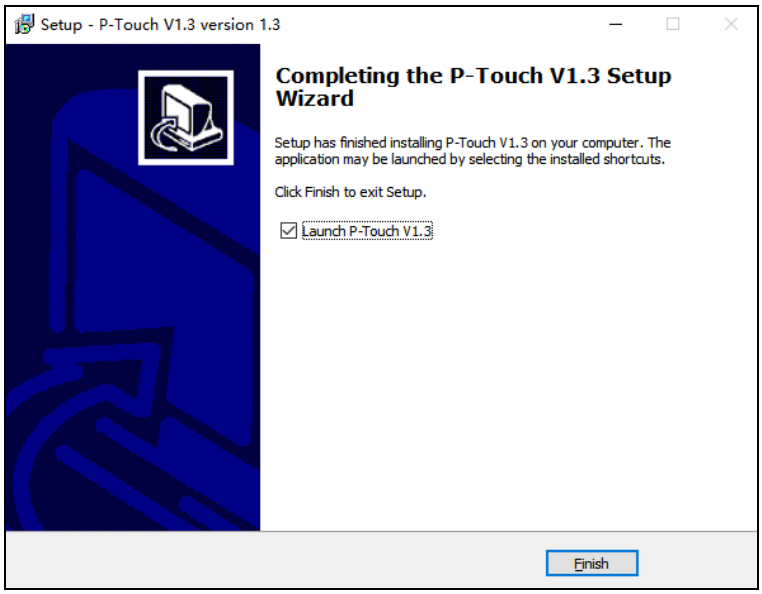


图 2-3: P-Touch 安装完成画面

3. P-Touch 界面介绍

点击打开该软件，界面如下图所示。可看到左上角有文件（F）、设置（S）、工具（T）和帮助（H）四种功能菜单，菜单栏下方方案设置被分为八个区块。在每个区块内选定所需功能后，用户便可在生成的程序框架内编写功能程序。程序除错结束后，如果之前选择了启用上位机还可连接 T_Watch 观察 TK 变化。下面将详细介绍 P-Touch 的使用。

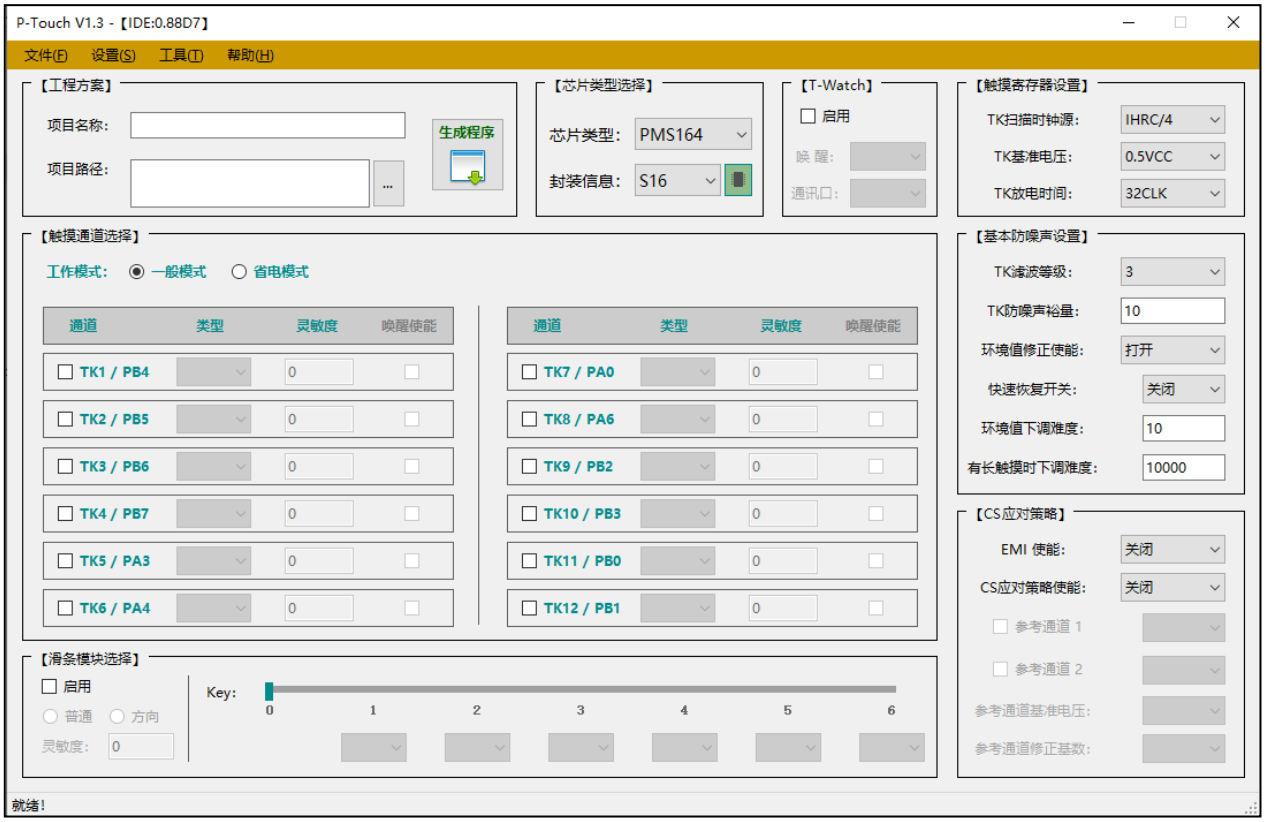


图 3-1: P-Touch 打开界面

3.1.方案配置说明

用户可点击左上角“帮助”→ “使用说明”→ “参数设置”查看每项参数的设置说明。

为了便于查看，以下也会对该部分进行简单介绍。

3.1.1. 工程方案

用户可在【工程方案】区块填写项目名称并选择项目存放位置。项目名称可以是中文汉字、英文字母和数字。

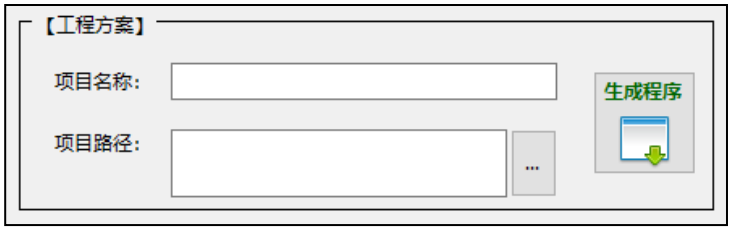
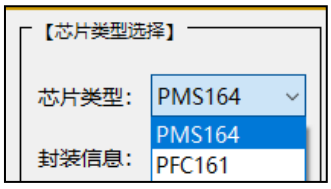


图 3-2: 工程方案设置界面

3.1.2. 芯片类型选择

1. 芯片类型选择：即选择客户需要的芯片类型，包括 PMS164，PFC161 的各种封装型号。创建工程前请先选择芯片型号。
2. 封装信息：根据需求选择合适的封装信息。
3. 芯片视图：封装信息右侧按钮  ，单击此按钮，可弹出对应芯片型号的封装引脚分配视图。方便用户选择脚位。



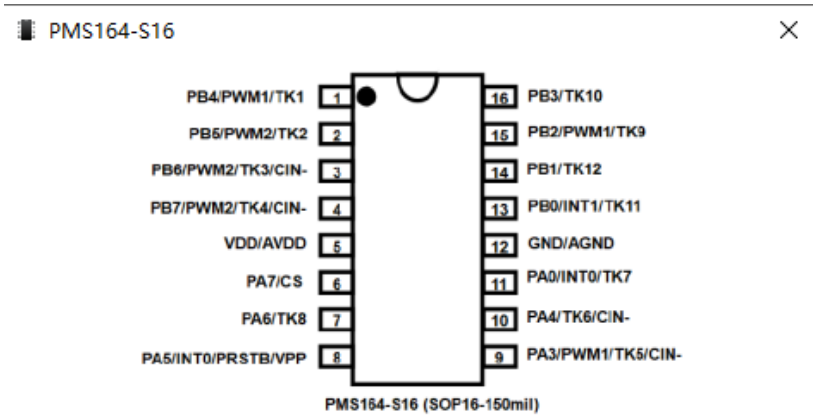


图 3-3: 芯片类型选择

3.1.3. T-Watch

用户可根据需求选择是否启用 T-Watch。

1. 启用选项：勾选启用上位机，一般项目处于调试阶段时需要开启此功能，实际 IC 可以通过 Uart 连 接上位机(T-Watch)，把触摸数据直观的反映在 T-Watch 上。
2. 唤醒方式：默认 T-Watch 唤醒；如果设为低电平唤醒，需要将 IO 对地短接大于 0.5 秒；
3. 通讯口：上位机功能需要一个通信 IO，且必须是外部中断引脚，仿真时此功能只支持 PA0 端口通信，实际 IC 可以是 PA0、PA5 和 PB0 中的任意一个。

实际使用时，请参考第 4 章 T-Watch 详细说明。

3.1.4. 触摸通道选择

1. 工作模式：包括一般模式和省电模式：
   (1) 一般模式：触摸灵敏，无休眠，一般用于不需计较功耗的应用，如交流电应用；
   (2) 省电模式：带睡眠，可唤醒，待机功耗低，一般用于电池应用。
2. 触摸通道：勾选可置能，此通道可用作触摸，否则除能。
3. 类型：即触摸类型，默认为短触摸，如需使用长触摸或长短触摸同时使用请选择类型为长触摸（兼容短触摸）。
4. 灵敏度：可调范围 0~210，默认 180，数值越大,灵敏度越高。
5. 唤醒使能：省电模式下选项，勾选可置能，可用此通道唤醒休眠，否则除能。

注意：选择省电模式时，右侧会出现一个 IO 唤醒设置的按钮，点击会弹框出唤醒设置，包括以下设置：

(1) 睡眠扫描间隔：睡眠时扫描按键的间隔范围 50-500ms，默认 100ms，此处一般保持默认，如需提高唤醒速度，可略微减少扫描间隔，如需降低待机电流也可略微增加扫描间隔。

(2) IO 唤醒设置：省电模式下除了用触摸唤醒系统外，也可在此处设置用外部 IO 唤醒系统。此处可选择唤醒端口以及唤醒电位，一般情况下低电平唤醒，如需高电平唤醒，请 IO 外接下拉电阻，以保持 IO 常态为低电平。

3.1.5. 滑条模块选择

1. 启用按钮：勾选为启用滑条模块，反之不启用。
2. 类型：包括普通状态滑条和方向滑条。方向滑条兼容普通状态滑条并且包含滑条滑动方向。
3. 灵敏度：范围为 0~210，数值越大，灵敏度越高。
4. 滑条通道共有 6 个，按照所需打开相应开关，选择对应触摸通道。

注意：滑条模块通道与触摸通道及 CS 应对策略中的参考通道不可重复。

3.1.6. 触摸寄存器设置

1. TK 扫描时钟源：即触摸功能扫描计数的时钟，此处时钟频率越高，TK 值越大。注意，此处并非是系统时钟。
2. TK 基准电压：又称 TK 参考电压，对 CS 电容大小和触摸灵敏度有影响，此处提高可降低外部基准电容。
3. TK 放电时间：启动触摸功能前的放电时间，一般 CS 电容越大需要的放电时间越长。

©Copyright 2020, PADAUK Technology Co. Ltd

3.1.7. 基本防噪声设置

1. TK 滤波等级：此处采用的是多次采样平均的滤波方式，等级由 0→6，分别对应采样次数 1 次、3 次、6 次、10 次、18 次、34 次和 66 次。此处等级越高防噪声能力越强，但同时触摸会略微变慢，所以用户要根据实际触摸个数和干扰情况选择滤波等级，默认等级 3。
2. TK 防噪声裕量：此处用于环境值向上修正余量，此值越大，当环境值需要上修时保留的裕量越大，使环境值比实际值略低，一定程度上应对 TK 实际值都对带来的干扰。此处一般保留默认值 10 即可。如要更改请注意此值取值范围：0≦防噪声裕量<（220—触摸灵敏度）。
3. 环境值修正：环境值修正总开关，用于触摸环境值实时修正以应对环境变化，一般保持开启。
4. 环境值下调难度：环境值下调修正难度 1~10000 值越大修正越慢，值为 100 代表至少采样 100 笔偏小资料才开始下调 1，一般保持默认即可。
5. 有长触摸时下调难度：有长触摸功能时环境值下修难度范围 100~65535，一般保持默认即可。

3.1.8. CS 应对策略

1. EMI 使能：打开后，IHRC 会左右跳动，如此在传导和辐射测试测试时，量测到的谐波功率 db 值会降低一些。
2. CS 应对策略使能：可开启和关闭 CS 应对策略。在产品需要过 CS 测试时可以选择开启此选项。
3. 参考通道 1、2：可选不需使用的触摸通道或隐藏在 IC 内的触摸通道（优先选隐藏在 IC 内的触摸通道），此通道不产生触摸信号所以不可与触摸通道混用。
4. 参考通道基准电压：参考按键准位，因参考按键通常为隐藏于 IC 内部或未使用的 TK 脚，从而计数值都相对较高，为了防止溢出，所以要降低位准（比 TK 基准电压低），防止溢出，默认 0.2 VCC。
5. 参考通道修正基数：通过参考通道修正环境值的基数，默认 1，一般保持默认即可。

3.2.菜单说明

1. **文件**：可点击“打开配置参数”查看之前的程序初始设定，或点击“保存配置参数”保存当前设定。
2. **设置**：从此处选择要使用的 IDE 版本。为使程序运行无误，请打开 P-Touch 后先选择可行版本 IDE 使用。
3. **工具**：从此处选择使用 T-Watch 或生成测试触摸仿真版程序。触摸仿真版的测试可以选择 T-Watch 显示或 IDE 显示，具体请查看第 4 章及第 5 章说明。
4. **帮助**：可点击“帮助”→“使用说明”查看 P-Touch 参数设置说明及触摸方案文件说明；
   或点击“帮助”→“关于”，进入应广触摸技术讨论 QQ 群。

3.3.方案设定范例

3.3.1. 参数设定

认真阅读各部分设置说明后，依序选择或填写用户方案需要的设定，确认无误后，点击“设置”选择最新版 IDE，最后方可点击“生成程序”。

后续说明以以下界面设定为例：

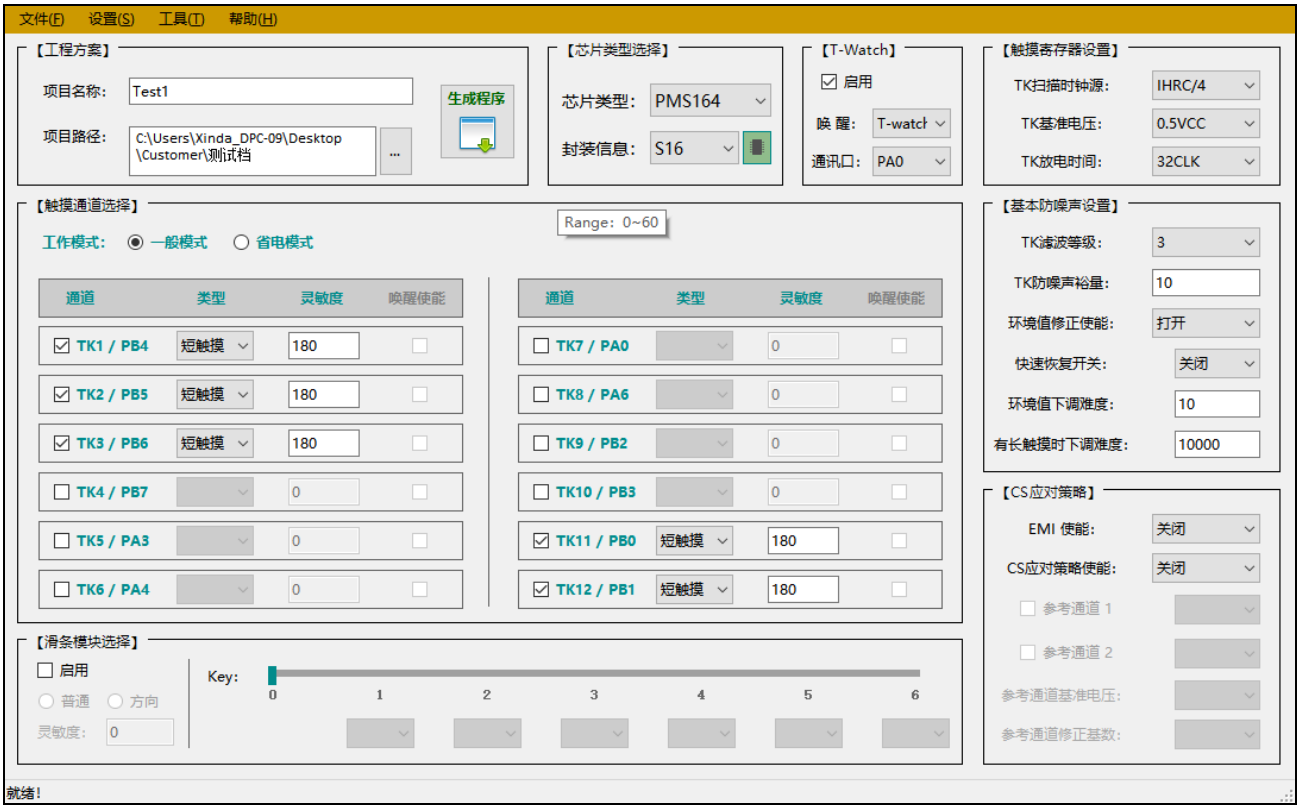


图 3-4: 参数设定界面

注意：勾选启用“上位机”（T_Watch），可通过T_Watch观察每个触摸按键的变化，具体说明请参考T_Watch章节。

3.3.2. 打开程序

在基本参数及IDE版本都设定好之后，点击“生成程序”，用户可以看到以下界面，选择用IDE运行程序。

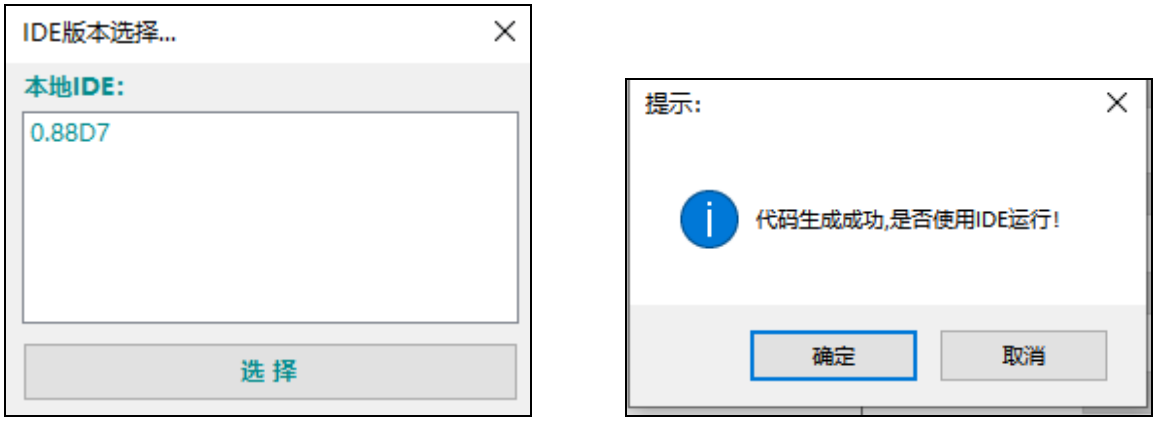


图3-5: 打开文件

3.4.编写功能

P-Touch 芯片类型选择分为PMS164和PFC161。它们在结构上大致存在以下差异：

	PMS164	PFC161
触摸通道数量	12 个(TK1-TK12)	8 个(TK4-TK11)
CS 脚	PA.7	PA.7 或 PB.7（M10 封装）
TK9	PB.2	PA.5
TK10	PB.3	PA.7

工程文件打开后，即可基于此Touch_Demo编写实际工程程序。下面便具体介绍Touch_Demo。

Touch_Demo的文件系统共包含6个文件（隐藏库文件不包括在内），其中提供客户直接修改应用的文件有3个，分别是PT_Lib.h、User_Function.c和Touch_Demo.c：

- PT_Lib.h 是触摸库配置文件，可以对触摸功能和 Uart Mode 进行设置和使能；
- User_Function.c 是用户的功能编写文件，用户可以在其中写入自己需要的功能；
- Touch_Demo.c 则是主程序所在文件，可以对上面两个文件中使用的 IO 脚进行输入输出和数字使能等相关操作，也可以选择主程序的工作模式。

下面分别进行详细说明。

3.4.1. 触摸库配置文件 PT_Lib.h

注意：

此文件说明旨在帮助客户理解程序，PT_Lib.h 内的参数设置在 P-Touch 生成程序时已经自动配置完成，如非必要，请勿修改此文件中的配置信息。

此文件旨在帮助用户学习如何配置触摸相关的设置，包括触摸引脚选择，灵敏度配置，唤醒引脚配置，环境修正参数以及 Uart 开关等。

1. 触摸通道使能设置：(**Const_EN_CH_T_Key**)
   通道使能设置为 1：通道置能；通道使能设置为 0：通道除能

```
    //T_Key通道使能设置
Touch_Channel_Selection:
//  Const_EN_CH_T_Key1  =>  1   //TK1(PB4)通道使能(自动T_Key_Scan_Reg置能)
//  Const_EN_CH_T_Key2  =>  1   //TK2(PB5)通道使能  1 : 置能  0 : 除能
//  Const_EN_CH_T_Key3  =>  1   //TK3(PB6)
//  Const_EN_CH_T_Key4  =>  1   //TK4(PB7)
//  Const_EN_CH_T_Key5  =>  1   //TK5(PA3)
//  Const_EN_CH_T_Key6  =>  1   //TK6(PA4)
//  Const_EN_CH_T_Key7  =>  1   //TK7(PA0)
//  Const_EN_CH_T_Key8  =>  1   //TK8(PA6)
//  Const_EN_CH_T_Key9  =>  1   //TK9(PB2)
//  Const_EN_CH_T_Key10 =>  1   //TK10(PB3)
//  Const_EN_CH_T_Key11 =>  1   //TK11(PB0)
//  Const_EN_CH_T_Key12 =>  1   //TK12(PB1)
//-----------------------------------
```

图 3-6: T_Key 通道使能设置

注意：

(1) 已经开启的 **T_Key** 通道对应的 IO 请保持模拟输入，并且无不含上拉电阻，请勿随意切换为输出 IO。

(2) 若程序运行前不确定触摸通道数量，或需在程序中关闭或打开触摸通道，请先将所有可能用到的通道使能。而程序中，只有 **T_Key_Scan_Reg** 寄存器对应的通道位为 1 时，**Const_EN_CH_T_KeyX**=1 通道使能语句才有效。如：

T_Key_Scan_Reg.1 = 0；//关闭 T_Key1 通道

T_Key_Scan_Reg.1 = 1；//开启 T_Key1 通道（Const_EN_CH_T_Key1==1 时有效）

2. 睡眠定时扫描和唤醒设置：(**Const_Wakeup_CH_T_Key**)
   触摸通道唤醒设置为 1：通道唤醒使能； 触摸通道唤醒设置为 0：通道唤醒除能

```
//  Const_Wakeup_Timing =>  100 //60-600睡眠时扫描按键的间隔,单位ms
//-----------------------------------
\******当主程序用省电模式(Const_Work_Mode=>2)时,程序会睡眠,需设置以下唤醒通道******/
    //T_Key唤醒通道使能设置
Touch_Wakeup_Channel_Selection:
//  Const_Wakeup_CH_T_Key1  =>  1   //TK1(PB4)通道唤醒使能,与Const_EN_CH_T_Keyx配合使用
//  Const_Wakeup_CH_T_Key2  =>  1   //TK2(PB5)通道唤醒使能 1 : 置能  0 : 除能
//  Const_Wakeup_CH_T_Key3  =>  1   //TK3(PB6)
//  Const_Wakeup_CH_T_Key4  =>  1   //TK4(PB7)
//  Const_Wakeup_CH_T_Key5  =>  1   //TK5(PA3)
//  Const_Wakeup_CH_T_Key6  =>  1   //TK6(PA4)
//  Const_Wakeup_CH_T_Key7  =>  1   //TK7(PA0)
//  Const_Wakeup_CH_T_Key8  =>  1   //TK8(PA6)
//  Const_Wakeup_CH_T_Key9  =>  1   //TK9(PB2)
//  Const_Wakeup_CH_T_Key10 =>  1   //TK10(PB3)
//  Const_Wakeup_CH_T_Key11 =>  1   //TK11(PB0)
//  Const_Wakeup_CH_T_Key12 =>  1   //TK12(PB1)
//-----------------------------------
```

图 3-7: T_Key 唤醒通道使能设置

3. 触摸灵敏度：(**Const_SEN_T_Key**)

触摸通道灵敏度设置，范围 0~210，默认为 180。设置为 0：灵敏度最低。设置为 210：灵敏度最高

```
    //T_Key灵敏度sensitivity设置
Touch_Sensitivity_Set:
//  Const_SEN_T_Key1        =>  180 //Max=210,Min=0,该应灵敏度设定,数值越大,灵敏度越高
//  Const_SEN_T_Key2        =>  180 //0:最不灵敏 210:最灵敏 default:180
//  Const_SEN_T_Key3        =>  180
//  Const_SEN_T_Key4        =>  180
//  Const_SEN_T_Key5        =>  180
//  Const_SEN_T_Key6        =>  180
//  Const_SEN_T_Key7        =>  180
//  Const_SEN_T_Key8        =>  180
//  Const_SEN_T_Key9        =>  180
//  Const_SEN_T_Key10       =>  180
//  Const_SEN_T_Key11       =>  180
//  Const_SEN_T_Key12       =>  180
//-------------------------------------
```

图 3-8: T_Key 灵敏度设置

4. 长触摸使能通道设置：(**Const_En_Long_T_Key**)

设定触摸信道是否有长触摸功能。**Const_En_Long_T_Key** 设置为 0：无长触摸；设置为 1：有长触摸

```
    //是否使用长触摸            //主要用于环境值修正参考，有长触摸环境值修正减慢
//  Const_En_Long_T_Key1    =>  1
//  Const_En_Long_T_Key2    =>  1
//  Const_En_Long_T_Key3    =>  1
//  Const_En_Long_T_Key4    =>  1
//  Const_En_Long_T_Key5    =>  1
//  Const_En_Long_T_Key6    =>  1
//  Const_En_Long_T_Key7    =>  1
//  Const_En_Long_T_Key8    =>  1
//  Const_En_Long_T_Key9    =>  1
//  Const_En_Long_T_Key10   =>  1
//  Const_En_Long_T_Key11   =>  1
//  Const_En_Long_T_Key12   =>  1
//-------------------------------------
```

图 3-9: 长触摸使能设置

5. 触摸方块滑条设置：(**Const_En_Slider**)

Const_En_Slider_A 设置为 1：方块滑条使能；设置为 0：方块滑条除能

Slider_T_Key 设置方块滑条的信道及顺序。

方块滑条最多支持 6 个通道。

触摸通道灵敏度设置：从 0~210 灵敏度逐渐升高，默认灵敏度为 180。

```
    //滑条设置部分
Touch_Slider_Set:
//  Const_En_Slider_A   =>  1   //6Key方块滑条使能
//  Slider_T_Key1   =>  'TK6';          //XDT_Slider_1.1
//  Slider_T_Key2   =>  'TK5';
//  Slider_T_Key3   =>  'TK4';
//  Slider_T_Key4   =>  'TK3';
//  Slider_T_Key5   =>  'TK2';
//  Slider_T_Key6   =>  'TK1';
//  Const_SEN_Slider_A=> 180    //滑条灵敏度设置, Max=210,Min=0,该应灵敏度设定,数值越大,灵敏度越高
                                ////0:最不灵敏 210:最灵敏 default:180
```

图 3-10: 滑条设置

©Copyright 2020, PADAUK Technology Co. Ltd

6. 触摸按键控制参数设置：

触摸时钟源设置：(**Const_Touch_Source_CLK**)

0：保留；1：保留；2：IHRC/4；3：IHRC/8；4：IHRC/16；5：IHRC/32；6：IHRC/64

7：IHRC/128；8：ILRC。默认为 2：IHRC/4

触摸 CS 电容参考电压设置：(**Const_Touch_VRef**)

0：0.5Vcc；1：0.4Vcc；2：0.3Vcc；3：0.2Vcc

触摸 CS 电容参考电压设置与触摸灵敏度有关，设置 0 灵敏度最高，设置 3：灵敏度最低

触摸转换前 CS 电容预放电时间设置：(**Const_Touch_Discharge**)

0：0_CLK 放电时间；1：32_CLK 放电时间；2：64_CLK 放电时间；3：128_CLK 放电时间

```
//实际触摸按键设置
Const_Touch_Source_CLK  =>  2;  //  Touch Pad clock selection
                                //0:IHRC/1, 1:IHRC/2, 2:IHRC/4, 3:IHRC/8,
                                //4:IHRC/16, 5:IHRC/32, 6:IHRC/64, 7:IHRC/128, 8:ILRC
                                //default:2
Const_Touch_VRef        =>  1;  //0:0.5*VCC, 1:0.4*VCC, 2:0.3*VCC, 3:0.2*VCC
                                //CS电容参考电压设定 0.2VCC-0.5VCC
                                //对CS电容大小和触摸灵敏度有影响
                                //此处提高可降低外部基准电容
                                //default:0
Const_Touch_Discharge   =>  1;  //0:CLK_0, 1:CLK_32, 2:CLK_64, 3:CLK_128
                                //Discharge time before starting the touch function
                                //default:1
```

图 3-11: 实际触摸按键设置

7. Touch 环境修正及抗干扰设置：

触摸采样一笔数据所需的滤波等级设置：(**Const_T_Key_Smooth_Rank**)

滤波等级为 1~ 6，数值愈大代表滤波越平滑，当然采样耗时也愈长。默认值为 3。

触摸防噪声裕量：(**Const_T_Key_Noise_Margin**)

防噪声裕量等级为 0~ 60，数值愈大代表防噪声越强。

触摸环境值修正使能：(**Const_Env_Fix**)

Const_Env_Fix 设置为 1：环境值修正使能 ； 设置为 0：环境值修正除能。

触摸按键环境值快速修复开关：（**Const_Fast_Recover**）

触摸按键释放后快速恢复环境值

Const_Fast_Cover 设置为 1：快速修复环境值使能；设置为 0：快速修复环境值除能。

短按触摸通道环境值往下修正梯度：(**Const_Env_Dw_Fix_Cnt**)

设定范围：1 ~ 10000。数值愈小修正愈快，数直愈大修正愈慢。

例如设值为 100，代表至少采样 100 笔偏小的实时环境值，才将参考触摸环境值往下调 1。

长按触摸通道环境值往下修正梯度：(**Const_Env_Dw_Fix_Cnt_Long_Touch**)

设定范围：100 ~ 65535。数值愈小修正愈快，数直愈大修正愈慢。

例如设值为 100，代表至少采样 100 笔偏小的实时环境值，才将参考触摸环境值往下调 1。

©Copyright 2020, PADAUK Technology Co. Ltd

```
//------Touch环境修正及抗干扰设置-----
Touch_Ref_and_Noise_Set:
    Const_T_Key_Smooth_Rank =>  3   //TK滤波等级(1-6),数值越大滤波越平滑,采样耗时越长
                                    //default:3
    Const_T_Key_Noise_Margin=>  10  //Tk防噪声裕量(0-60),数值越大代表防噪声越强
                                    //default:1
    Const_Env_Fix           =>  1   //环境值修正总开关  0 : 关闭 1 : 打开
    Const_Fast_Recover      =>  0   //触摸按键快速恢复开关(按键释放后快速恢复环境值)    1:开启 0:关闭
                                    //default:0
    Const_Env_Dw_Fix_Cnt    =>  10  //环境值下调修正难度 1 - 10000 值越大修正越慢    值为100代表至少采样100笔偏小资料才开始下调1
                                    //default:10
    Const_Env_Dw_Fix_Cnt_Long_Touch =>  10000
                                    //有长触摸功能时环境值下修难度   100-65535
                                    //default:10000
//-----------------------------------
```

图 3-12: Touch 环境值修正及抗干扰设置

注意：仿真时触摸会比较慢，是因为仿真器与 TouchKit 通信需要时间（实际 IC 不存在通信时间）。此时降低 **Const_T_Key_Smooth_Rank** 等级可减少触摸测量时间(以减少软件滤波，减少取样点为代价)。

8. 防噪声参考按键设置

防噪声信道参考电压设置：(**Const_Touch_Noise_VRef**)

```
    //防噪声参考按键设置
//  Const_Touch_Noise_Strategy  =>  0   //取值范围 0-3  分别对应关闭防噪声策略和开启防噪声策略1-开启防噪声策略3
                                        //主要应对电源干扰
                                    //防噪声策略1    简单粗暴，有噪声纯压制环境值，防止按键触发，但谨防PWM式连续噪声使环境值压制过度，按键难以触发
                                    //防噪声策略2    灵活修复环境值，按照基数与参考通道同时变大变小，效果比策略3偏差，但节省ROM和RAM
                                    //防噪声策略3    目前修复环境值表现最佳，按照基数与参考通道变化等比修正，但占用较多ROM和RAM
//  Const_T_Key_Noise_Ref_1 =>  'TK12'  //取值范围 TK1-TK12 ，为其他值时无效
//  Const_T_Key_Noise_Ref_2 =>  'TK11'  //取值范围 TK1-TK12 ，为其他值时无效
                                    //干扰较大时，请至少打开1个参考通道,且按顺序开启，开启顺序Const_T_Key_Noise_Ref_1>Const_T_Key_Noise_Ref_2
                                    //注意:此处设置的参考通道不产生触摸信号，请勿和使用的触摸通道混用
//  Const_Touch_Noise_VRef  =>  3   //参考按键准位
                                    //0:0.5*VCC, 1:0.4*VCC, 2:0.3*VCC, 3:0.2*VCC
                                    //因参考按键通常为隐藏于IC内部或未使用的TK脚，从而计数值都相对较高，为了防止溢出，所以要降低位准，防止溢出
                                    //一般比Const_Touch_VRef设置的电压低如 0.3*VCC
                                    //default:3
//  Const_T_Key_Noise_Fix_Cn=>  1   //范围:1-6,通过防噪声按键修正环境值的基数
                                    //default:1
```

图 3-13: 防噪声参考按键设置

9. 上位机

```
//-------上位机参数设定---------------
T_Watch_Set:
    Const_EN_Uart       =>  1       //Uart使能, 0:Uart除能  1:Uart置能
                                    //需要接上位机时请将Const_EN_Uart置能,不用上位机请除能
                                    //此处除能后其他上位机参数无效
                                    //default:0
    Const_Uart_Wakeup_Mode=>1       //上位机唤醒方式, 0:低电平唤醒(>0.5s) 1:上位机T-Watch唤醒
                                    //default:1
    Interrupt_Uart      =>  0       //UART中断及传输端口选择
                                    //0:PA0中断   1:PA5中断   2:PB0中断   default:0
                                    //注意:仿真时不支持PA5和PB0中断
                                    //实际IC烧录建议使用PA5
```

图 3-14: 上位机参数设置

注意：如果软件生成的 Code 已经选择过 Uart 通讯端口，但用户又想要在程序里重新更改此设置，除了更改 **Interrupt_Uart** 外，还需至 **FPPA_IO_Set(void)** 中将对应的通讯 IO 设置成输入、上拉以及数字输入使能。

示例如下：

已经在“3.3.1 参数设定”使能 Uart 且选择 PA0 为通讯端口，程序内 **T_Watch_Set:**部分如图 3-14 所示。PA0 端口对应的三个寄存器（*PAC/PAPH/PADIER*）设定如下：

```
void    IO_Init(void)
{
    //----------------------------
    PA      =   0b0000_0000;
    PAC     =   0b0000_0000;    //1:输出   0:输入
    PAPH    =   0b0000_0001;    //1:加上拉 0:不加上拉
    PB      =   0b0000_0000;
    PBC     =   0b0000_0000;
    PBPH    =   0b0000_0000;

    PADIER  =   0b0000_0001;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    PBDIER  =   0b0000_0000;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    //----------------------------
```

图 3-15: 修改 Uart 通讯端口

如果此时想更改 PA5 为通讯端口，但您的方案已经写了很多，所以不便从头打开 P-Touch 再设定新的程序框架，则在当前程序中，需先将 **T_Watch_Set:** 的 **Interrupt_Uart** 设为 1，如下图所示。

（如果想改为 PB0，需先确认 TK11(PB0)未被选为触摸通道后才能修改）

```
//-------上位机参数设定---------------
T_Watch_Set:
    Const_EN_Uart       =>  1       //Uart使能, 0:Uart除能   1:Uart置能
                                    //需要接上位机时请将Const_EN_Uart置能,不用上位机请除能
                                    //此处除能后其他上位机参数无效
                                    //default:0
    Const_Uart_Wakeup_Mode=>1       //上位机唤醒方式, 0:低电平唤醒(>0.5s) 1:上位机T-Watch唤醒
                                    //default:1
    Interrupt_Uart      =>  1       //UART中断及传输端口选择
                                    //0:PA0中断   1:PA5中断   2:PB0中断    default:0
                                    //注意:仿真时不支持PA5和PB0中断
                                    //实际IC烧录建议使用PA5
```

图 3-16: 修改 Uart 通讯端口

再将 pac.5、padier.5、paph.5 按如下修改即可：

```
void    IO_Init(void)
{
    //----------------------------
    PA      =   0b0000_0000;
    PAC     =   0b0000_0000;    //1:输出   0:输入
    PAPH    =   0b0000_0000;    //1:加上拉 0:不加上拉
    PB      =   0b0000_0000;
    PBC     =   0b0000_0000;
    PBPH    =   0b0000_0000;

    PADIER  =   0b0000_0000;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    PBDIER  =   0b0000_0000;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    //----------------------------
```

图 3-17: 修改 Uart 通讯端口

10. 触摸测试开关

```
//--------触摸测试开关-------------
//  T_Key_Debug         =>  1       //打开可在Log中看到已打开通道的触摸信号
//  Continuous_Debugging=>  1       //打开可连续输出测量讯号，关闭时仅仅讯号跳变会输出测量讯号到LOG窗口(如触摸)
//  Disable_Debug_Var   =>  1       //T_Key_Debug=1时，禁用Debug变量
                                    //当提示变量内存分配失败(即RAM不足)时使用，释放Debug变量内存
```

图 3-18: 触摸测试开关

- **T_Key_Debug** 是触摸测试的总开关，仅当其为 1 时下述两开关方有效。
- **Continuous_Debugging**

  Continuous_Debugging = 1： 持续扫描开启。

每次扫描完用户开启的全部通道之后，**Debug_Num** 递增 1 且输出扫描各通道触摸的环境值、实时值、触发状态。

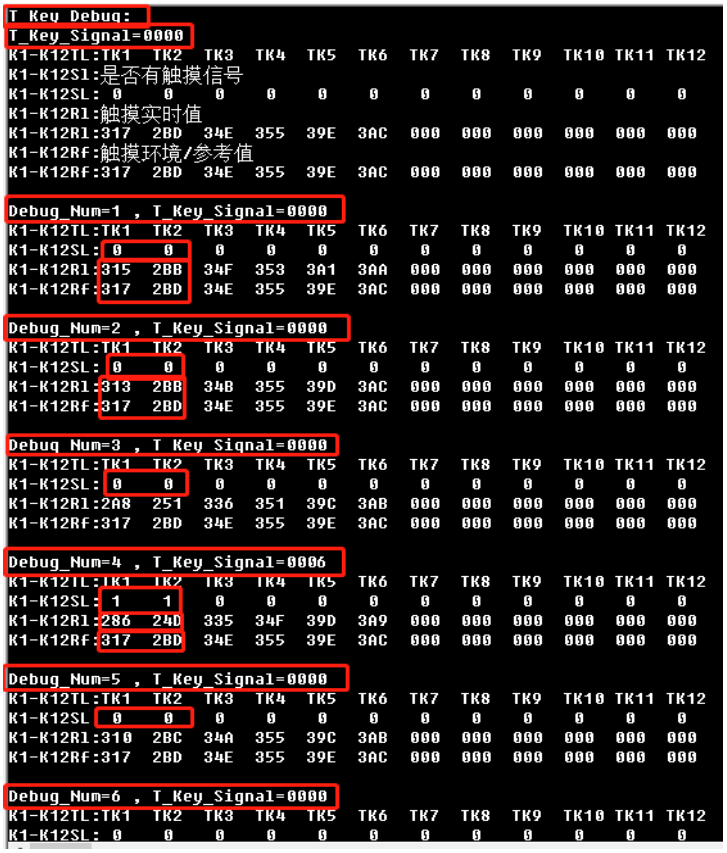


图 3-19: 持续 debug 时画面

Continuous_Debugging = 0：持续扫描关闭。

关闭后，仅触发状态改变时会输出相应测量讯号到 LOG 窗口。如下图：

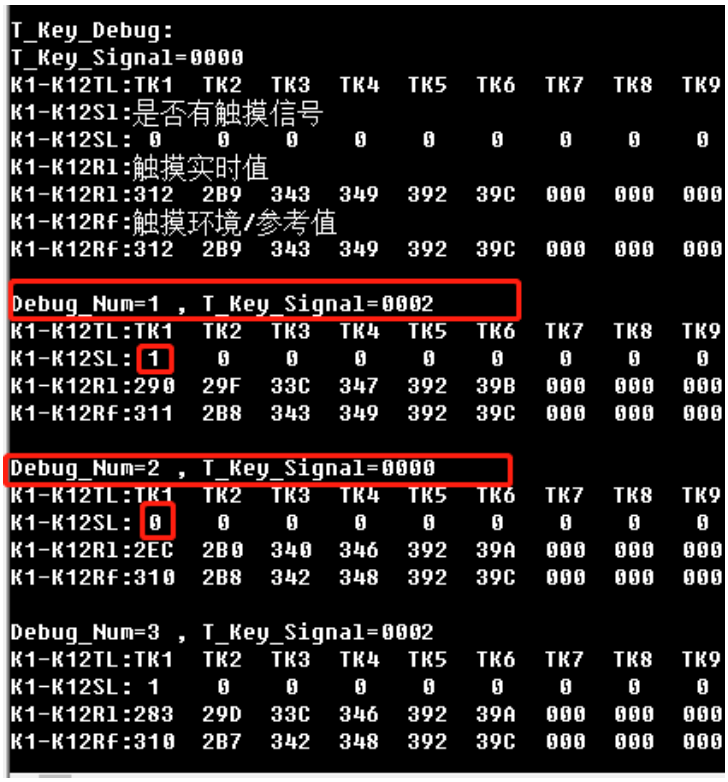


图 3-20: 仅打开 T_Key_Debug 画面

- **Disable_Debug_Var**

当 T_Key_Debug 为 1 时，若提示变量内存分配失败（即 RAM 不足）可打开 Disable_Debug_Var 来释放部分 RAM 解决该问题。

注意：烧录时以上三个开关（**T_Key_Debug**、**Continuous_Debugging**、**Disable_Debug_Var**）务必全部屏蔽。

11. 主要函数说明及相关寄存器定义

```
//  库内主要函数说明
//*********************************//
//T_Key_Channel_Setting T_Keyx  //手动T_Key端口设定，x表示常数
//Env_Fix    T_Keyx             //强制修复环境值到当前实际值，x表示常数
//void  TK_Init_Auto(void);     //根据Const_T_Key_VRef和Const_EN_CH_T_Keyx自动初始化T_Key相关寄存器
                                //如若中途改变通道请使用T_Key_Scan_Reg寄存器,对应Const_EN_CH_T_Keyx应置1
//void  T_Key_Scan(void);       //T_Key扫描函数(非阻塞式分步骤轮询)
//void  T_Key_Data_Ref_Initial(void);   //初始化第一笔触摸环境修正值
//void  Get_T_Key_Signal(void); //根据TK扫描结果判断按键是否成立并给出按键信号 T_Key1_Signal - T_Key12_Signal
//void  Uart_Auto(void);        //Uart自动初始化
//void  Sleep_Mode(void);       //睡眠模式,根据Const_Wakeup_CH_T_Keyx和Const_EN_CH_T_Keyx的设定唤醒
//T_Key_Scan_Reg                //TK扫描寄存器(16位)
//T_Key_Signal                  //按键标志寄存器(16位)
//---------------------------------*/
```

图 3-21: 触摸库内主要函数说明

- **T_Key_Scan_Reg** —— TK 扫描寄存器（16 位）

位	初始值	读/写	说明
15	0	-	预留（请设为 0）
14	0	-	预留（请设为 0）
13	0	-	保留用作中断标志
12	0	读/写	T_Key12 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key12 置 1 时有效）
11	0	读/写	T_Key11 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key11 置 1 时有效）
10	0	读/写	T_Key10 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key10 置 1 时有效）
9	0	读/写	T_Key9 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key9 置 1 时有效）
8	0	读/写	T_Key8 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key8 置 1 时有效）
7	0	读/写	T_Key7 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key7 置 1 时有效）
6	0	读/写	T_Key6 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key6 置 1 时有效）
5	0	读/写	T_Key5 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key5 置 1 时有效）
4	0	读/写	T_Key4 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key4 置 1 时有效）
3	0	读/写	T_Key3 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key3 置 1 时有效）

©Copyright 2020, PADAUK Technology Co. Ltd

位	初始值	读/写	说明
2	0	读/写	T_Key2 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key2 置 1 时有效）
1	0	读/写	T_Key1 扫描置能，0/1：禁用/启用 （Const_EN_CH_T_Key1 置 1 时有效）
0	0	-	T_Key 轮询扫描结束标志位，保留用作 Scan_End 标志

注意：程序中需要用到的 T_keyx 端要在 **PT_Lib.h** 档中置 1 才有效。

只有当中途需改变 T_keyx 脚时才需用到此寄存器，一般情况下只需要初始化时调用 **TK_Init_Auto()** 函数，系统便会自动根据 **Const_T_Key_VRef** 和 **Const_EN_CH_T_Keyx** 初始化 **T_Key** 相关寄存器 (**T_Key_Scan_Reg**)。

故一般情况不需要自行设置 **T_Key_Scan_Reg** 寄存器。

- **T_Key_Signal** —— 按键标志寄存器（16 位）

位	初始值	读/写	说明
15	0	-	预留（请设为 0）
14	0	-	预留（请设为 0）
13	0	-	预留（请设为 0）
12	0	读	T_Key12_Signal
11	0	读	T_Key11_Signal
10	0	读	T_Key10_Signal
9	0	读	T_Key9_Signal
8	0	读	T_Key8_Signal
7	0	读	T_Key7_Signal
6	0	读	T_Key6_Signal
5	0	读	T_Key5_Signal
4	0	读	T_Key4_Signal
3	0	读	T_Key3_Signal
2	0	读	T_Key2_Signal
1	0	读	T_Key1_Signal
0	0	-	预留（请设为 0）

©Copyright 2020, PADAUK Technology Co. Ltd

3.4.2. 用户功能文件 User_Function.c

1. 变量及 IO 的初始化程序

变量定义及初始化：

用户可在 **Variable_Define** 下定义所需变量，然后在 **void Variable_Init(void)**函数下完成对变量的初始化。

```
//****************************************************//
UserFile_TOP:
//****************************************************//
//User's Variable Define
Variable_Define:
//Insert variable define Code

//User's variable initialization Settings.
void    Variable_Init(void)
{
    //Insert variable Initial Code
}
```

图 3-22: 变量初始化示例图

IO 初始化：

函数 **void IO_Init(void)**用于客户配置触摸 IO 和其他 IO 的输入状态和数字/模拟使能。

对于没有用到的 IO 要设置为输出低或输入上拉（有利于系统稳定和省电）。

IO 初始化函数：

```
//IO设置, 部分IO XD-Touch已经设置
void    IO_Init(void)
{
    //----------------------------
    PA      =   0b0000_0000;
    PAC     =   0b0000_0000;    //1:输出   0:输入
    PAPH    =   0b0000_0000;    //1:加上拉 0:不加上拉
    PB      =   0b0000_0000;
    PBC     =   0b0000_0000;
    PBPH    =   0b0000_0000;

    PADIER  =   0b0000_0000;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    PBDIER  =   0b0000_0000;    //CS脚及对应T_Key设置为模拟IO(设为0),用上位机时UART_IO设置为数字IO(设为1)
    //----------------------------
    //User's Other IO Define

    //----------------------------
}
```

图 3-23: IO 设置示例图

2. 滑块模块程序（六键 11 阶）

滑块模块分为普通滑条和方向滑条。其中方向滑条分为正方向和负方向。

六键 11 阶是我们提供滑条模块的最大可行范围。其中滑条阶数 m 和按键数 n 的关系是 m=2n-1。单个按键无法做到滑条效果。以最低两键 3 阶为例，3 阶分别是：单独按到按键 1、同时按到按键 1 和按键 2、单独按到按键 2。

©Copyright 2020, PADAUK Technology Co. Ltd

```
#if Const_En_Slider_A==1
Void    Slider_Function(void)
{
#if Direction_switch==1
    if(Slide_Positive_shift)    //滑条正移
    {
        //User's Code
    }
    elseif(Slide_Negative_shift)//滑条负移
    {
        //User's Code
    }
    else
    {
    }
#endif
    byte    shift   =   SliderA_Gear_Range & 0x0F;
    switch(shift)       //根据档位信号产生动作
    {
    case 1: //User's Code
        break;
    case 2: //User's Code
        break;
    case 3: //User's Code
        break;
    case 4: //User's Code
        break;
    case 5: //User's Code
        break;
    case 6: //User's Code
        break;
    case 7: //User's Code
        break;
    case 8: //User's Code
        break;
    case 9: //User's Code
        break;
    case 10:    //User's Code
        break;
    case 11:    //User's Code
        break;
    default:
    case 0: //滑条释放  //User's Code
        break;
    }
}
#endif
```

图 3-24: 滑块程序

3. 触摸异常处理

如果程序运行中出现 Intrq.Tk_OV==1 的情况，表示触摸溢出异常，此时应主动减小外部 CS 电容或降低 CS 参考电压 **Const_T_Key_VRef** 的设定（在 PT_Lib.h 中）。为了有效降低硬件成本，请优先采取减小外部 CS 电容的方法来调整。

```
//**************************************************//
void    T_Key_Warning(void)     //处理Tkey计数溢出异常
{
    if(Intrq.Tk_OV)
    {
        Intrq.Tk_OV =   0;
        //此处处理TK_OV信号, Intrq.Tk_OV = 1表示TK计数器溢出
        //此时则需要减小外部VC电容
        //或降低XDT_Lib_1.0.h中对Const_Touch_VRef的设定
    }
}
```

图 3-25: 触摸异常处理

4. 用户功能编写

用户功能编写在 **void T_Key_Function(void)**函数中的**//User can add code** 中，对应功能写在对应触摸按键中。一共 12 路触摸按键，其使能设置在 PT_Lib.h 内。

以 T_Key1 为例，如下图，若程序跳入 if 语句内，则表示触摸按键被按下；若程序跳入 elseif 语句内，则表示触摸按键释放或无触摸。

```
void    T_Key1_Func(void)
{
    if(T_Key1_Signal==1 && Pre_T_Key1_Release==1)
    {
        Pre_T_Key1_Release  =   0;
    //-----------------------------------------//

        //User can add code

    //-----------------------------------------//
    }
    elseif(T_Key1_Signal==0)
    {
        Pre_T_Key1_Release  =   1;
    }
}
```

图 3-26: 按键功能示例

在两者之间加定时器，可以判断是长触摸还是短触摸。

```
elseif(T_Key1_Signal==0)
{
    //Stop  Timing  //结束定时
    //判断是否满足长按
    //满足长按加入长按动作
    Pre_T_Key1_Release  =   1;
}
#endif
```

图 3-27: 按键功能示例

用户其他功能编写在 **void User_other_Function(void)**函数中**//User's Code**。

```
void    User_other_Func(void)
{

    //User's Code

}
```

图 3-28: 其他功能示例

5. 睡眠函数编写

#define Sleep_Condition（0）：进入省电睡眠的条件

若用户在参数配置页面将工作模式选为省电模式，则程序生成后此数字默认配置为 1，用户再自行添加所需的睡眠判断条件，如 **#define Sleep_Condition** (PA.0==0)。在一般工作模式下，括号内数字为 0，代表无睡眠。

```
//进入省电模式条件设置
#define Sleep_Condition (0)          //如(PA.0==0 && 变量T = 100) 1代表无条件
```

图 3-29: 睡眠条件示例

进入省电模式前程序设置。

```
//进入省电模式前动作
void    Pre_sleep_set(void)
{
    nop;
    //进入省电模式前动作，如关灯等
    //唤醒条件在XDT_Lib.h中设置
}
```

图 3-30: 睡眠前功能设置示例

唤醒方式分为触摸按键唤醒和 IO 唤醒，方式不同对应唤醒后动作不同，用户根据需要自己设定。

```
//唤醒后动作
void    After_wakeup_set(void)
{
    if(Wakeup_Signal==1)
    {
        Wakeup_Signal = 0;  //及时清零，否则影响下次进入省电模式
                            //触摸唤醒，如有必要可由Pre_T_Key_Signal读出是哪个通道唤醒
    }
    else
    {
        //IO唤醒
    }
}
```

图 3-31: 唤醒后动作示例

中断函数 **void Interrupt(void)**，顾名思义，客户的中断程序就写在这里。

```
void    Interrupt (void)
{
    pushaf;

    if (Intrq.T16 && Inten.T16)
    {   //  T16 Trig
        //  User can add code
        Intrq.T16   =   0;
        //...
    }

    popaf;
}
```

图 3-32: 中断函数示例

6. 其他寄存器及变量

(1) **Pre_T_Key_Release** 按键释放标志寄存器（16 位）

位	初始值	读/写	说明
15	0	-	预留（请设为 0）
14	0	-	预留（请设为 0）
13	0	-	预留（请设为 0）
12	0	读/写	Pre_T_Key12_Release
11	0	读/写	Pre_T_Key11_Release
10	0	读/写	Pre_T_Key10_Release
9	0	读/写	Pre_T_Key9_Release
8	0	读/写	Pre_T_Key8_Release
7	0	读/写	Pre_T_Key7_Release
6	0	读/写	Pre_T_Key6_Release
5	0	读/写	Pre_T_Key5_Release
4	0	读/写	Pre_T_Key4_Release
3	0	读/写	Pre_T_Key3_Release
2	0	读/写	Pre_T_Key2_Release
1	0	读/写	Pre_T_Key1_Release
0	0	读/写	预留（请设为 0）

注意：此寄存器在库内并未做读写操作，仅在 User_Function.C 中的 **T_Key_Function()**函数中使用。客户可自行读写。

(2) **T_Key1_Data_Ref** - **T_Key12_Data_Ref**（word）：TK1-TK12 环境值变量（可读/写）

(3) **T_Key1_Data_Real** - **T_Key12_Data_Real**（word）：TK1-TK12 实际值变量（可读/写）

(4) **SliderA_Gear_Range**（byte）：滑条档位，范围 1-11（可读/写，不会自动清零）

©Copyright 2020, PADAUK Technology Co. Ltd

3.4.3. 用户主工程文件

以用户工程名为 Touch_Demo 为例，则代码生成的主程序文件为 Touch_Demo.c，其中包括 1 个主函数和 5 个子函数。提供 2 个可选工作模式及其他功能配置选项。

此主程序框架由 P-Touch 软件自动生成，工作模式也在 P-Touch 参数配置页面选择，不可程序中擅自更改。

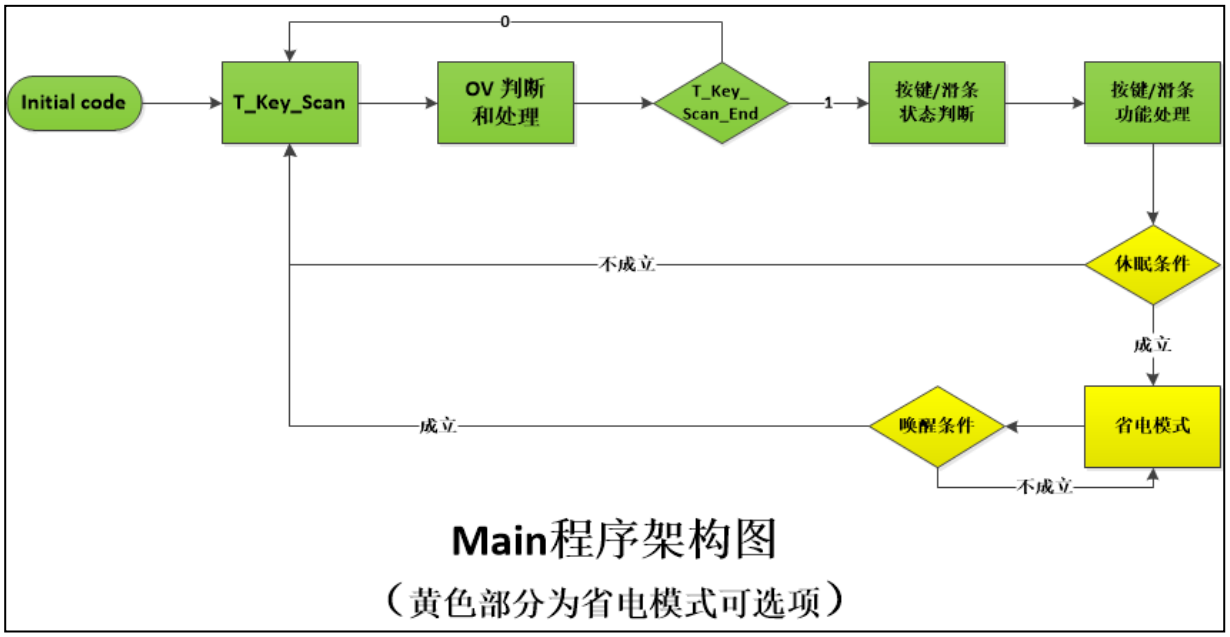


图 3-33: 睡眠和唤醒函数说明

1. 模式状态：

   工作模式：正常模式 / 省电模式

   主程序模式状态是在 P-Touch 中进行选择，然后在程序中生成。

   一旦选定工作模式后，不可二次更改此模式，如若想切换模式状态请在 P-Touch 重新生成新程序。

```
//-------主程序模式选择-----------------
Const_Work_Mode =>  2    //取值范围1-2
                         //1 :    mode 1  正常模式 : 触摸灵敏，无休眠，可加定时器
                         //2 :    mode 2  省电模式 : 带睡眠，可唤醒，睡眠前可加定时器
//-----------------------------------------
Const_En_Option_Detect  =>  0    //外部输入IO选择功能使能常量: 0:不启用 1:启用
//-----------------------------------------
```

图 3-34: 模式状态显示

2. 函数说明：

在正常模式：无休眠功能模式下，初始化函数包括 **IO_Init()**，**Variable_Init()**和 **TK_Init_Auto()**；主函数中包括 **T_Key_Scan()**，**T_Key_Warning()**，**T_Key_Process()**，**T_Key_Function()**和 **User_other_Function()**。其子函数程序请在 **User_Function.C** 文件下更改。

如若需要使用中断，在初始化中选择中断开关即可。

```
    IO_Init();      //IO Initial
    Variable_Init();
    //  Insert Initial Code
//  En_Interrupt;   //开启全局中断  //用UART时请使用这种写法开中断
//  Dis_Interrupt;  //关闭中断      //用UART时请使用这种写法关中断
    TK_Init_Auto(); //初始化Tk并自动根据预设常量Const_EN_CH_T_Keyx配置寄存器
                    //如若中途改变通道请使用T_Key_Scan_Reg寄存器,对应Const_EN_CH_T_Keyx应置1

    while (1)
    {
//      .wdreset;
        //------------------------------------------------------------//
        //mode 1    正常模式 : 无休眠功能
        T_Key_Scan();           //扫描T_Key
        T_Key_Warning();        //处理Tkey计数溢出异常
        if(T_Key_Scan_End)      //触摸按键轮询结束
        {
            T_Key_Process();        //处理Tkey数据并判断触摸状态(如果滑条开启,一同判断滑条状态)
            T_Key_Function();       //触摸按键成立后命令

            //----------------------
            User_other_Func();  //The user's function is called here.
            //----------------------
        }
        //------------------------------------------------------------------------------------//
//      ...
//      wdreset;

    }
}
```

图 3-35: 睡眠和唤醒函数说明

T_Key_Scan()源程序按键扫描示意图如下：

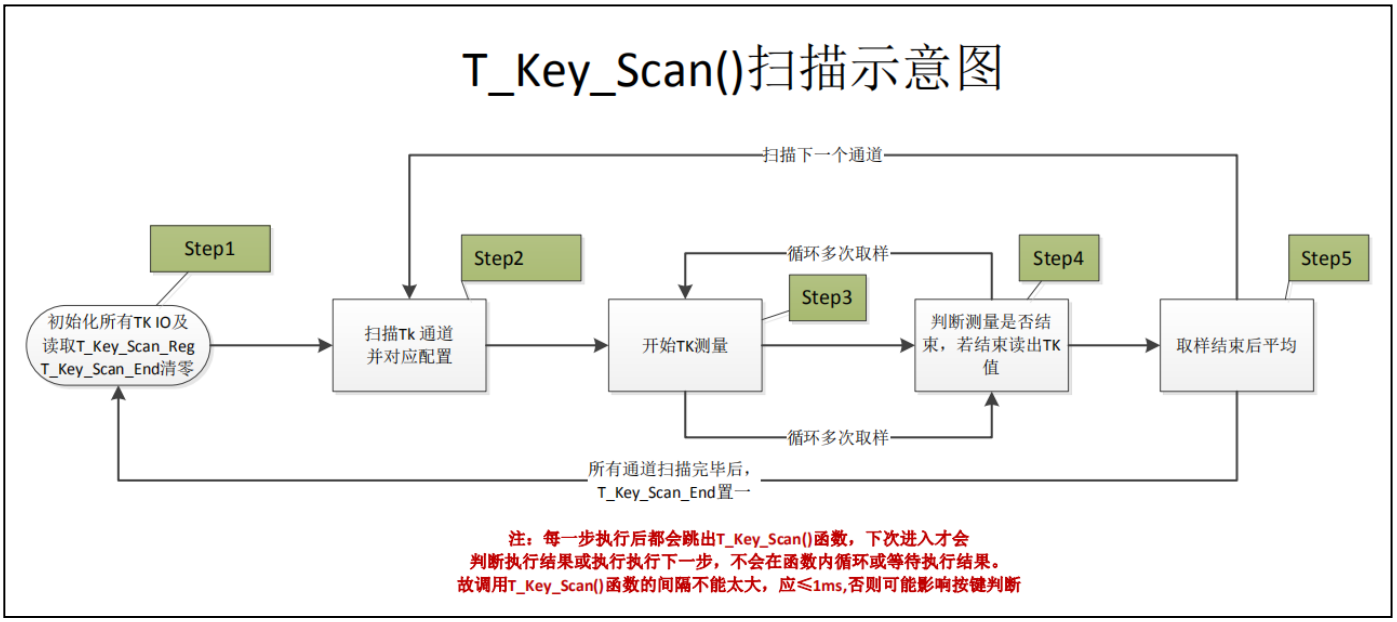


图 3-36: 按键扫描示意图

如若在省电模式下，会相较于正常模式多出睡眠和唤醒函数，包括子函数 **Pre_sleep_set()**，**Sleep_Mode()**和 **After_wake_up_set()**。子函数程序请在 **User_Function.C** 文件下更改。

```
if(T_Key_Signal==0 && Pre_T_Key_Signal==0 && Const_EN_Uart==0 && Sleep_Condition)
{                       //可加入其他限制条件，如关灯 if(T_Key_Signal==0 && LED==0)等
                        //无按键且无需连Uart发送数据时自动进入省电模式
    Pre_sleep_set();//先关闭PWM等其他耗电项
    Sleep_Mode();   //此处阻塞式睡眠，有唤醒才跳出睡眠
    After_wakeup_set();//先关闭PWM等其他耗电项
}
```

图 3-37: 睡眠和唤醒函数说明

4. T-Watch

T_Watch 即上位机。仿真器测试和实际工程方案测试时均可使用T_Watch来直观观察触摸按键的状态变化，这两种工程环境下打开和使用T_Watch的方法大同小异。下面将具体介绍T_Watch的使用方法与不同环境下通讯口的选择差异。

4.1. T-Watch - 仿真器测试

1. 使用T_Watch时要用串口传输数据，仿真器测试时PA0是唯一用于通讯的Uart IO。请依下图将串口与仿真器正确连接：

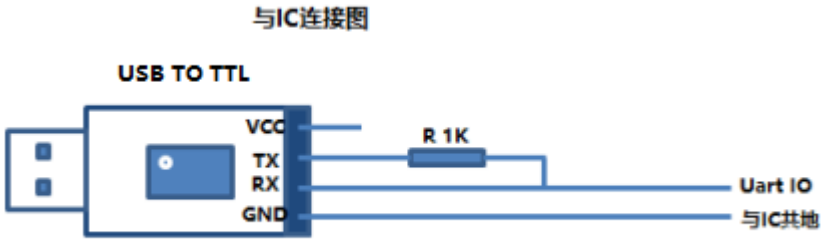


图 4-1: Uart 与仿真器连接图

注意： 请务必先将串口与仿真器连接，再打开T_watch。

2. 选择“工具” → “仿真器测试”→ “T_Watch显示”，如下图所示步骤：

图 4-2: 选择 T_Watch 显示

当选择T_Watch显示时，PA0被用于通讯，因此不可再作为触摸通道使用。

选择待测试通道，生成测试程序，然后打开并运行（Run）。可手动选择需要的测试通道，也可点击“Select all”选取除TK7外所有通道。

下图为程序运行时界面：

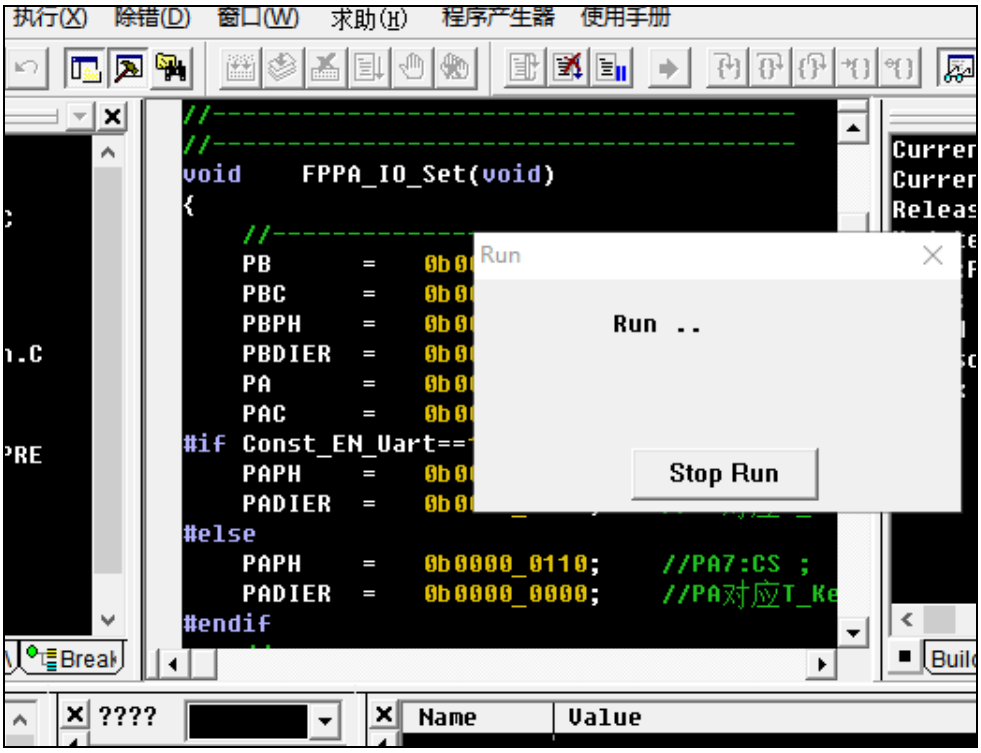


图 4-3: 程序运行(Run)

3. 点击“工具”→“T_Watch”

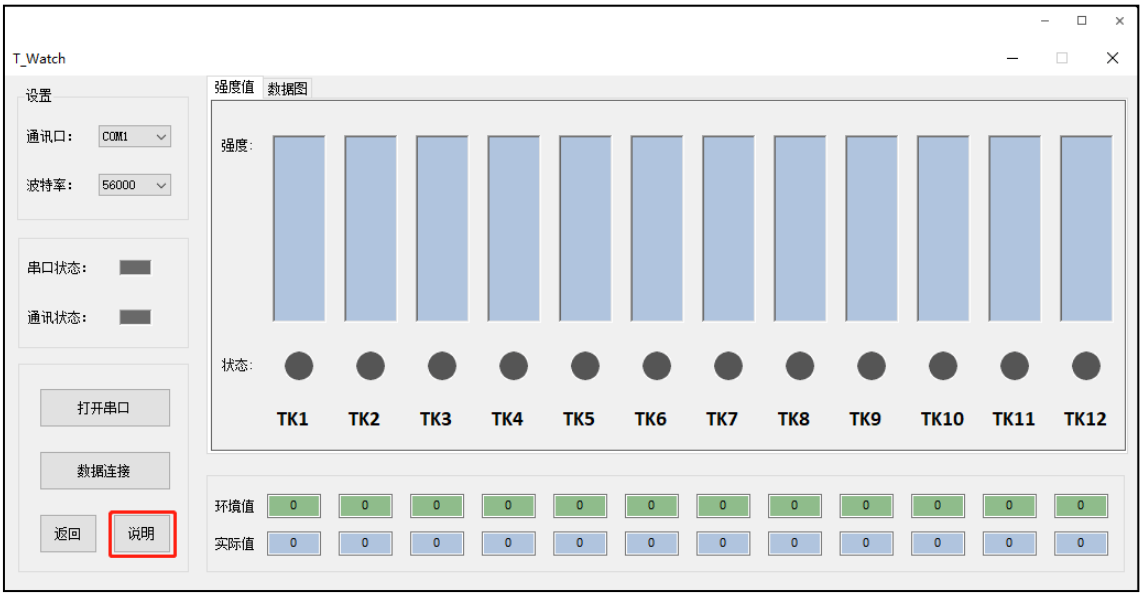


图4-4: 打开T_Watch界面

点击“说明”，可查看T-Watch使用须知。

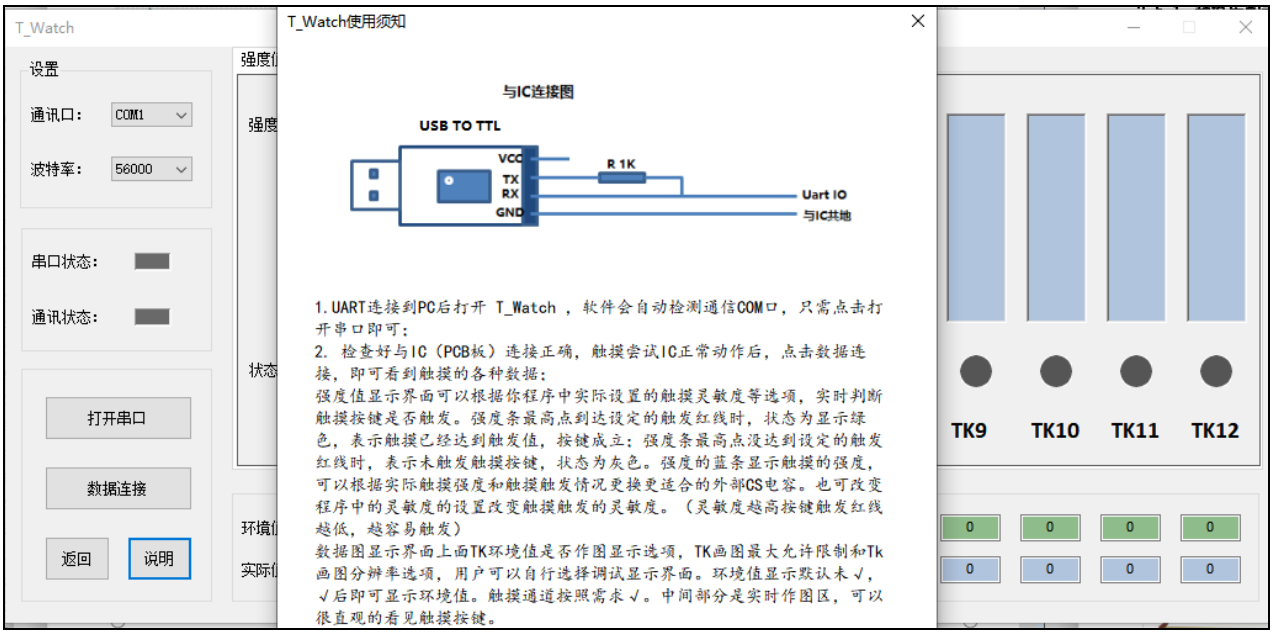


图 4-5: 查看 T_Watch 使用须知

4. 回到T_Watch设定界面，选择正确的通讯口。一般连接PC后，T_Watch 通讯口选择框会自动显示匹配的通讯口，此时，点击“打开串口”然后点击“数据连接”，串口成功打开后其状态显示为绿色，数据连接成功后通讯状态显示为以一定频率闪烁的黄色，如下图：

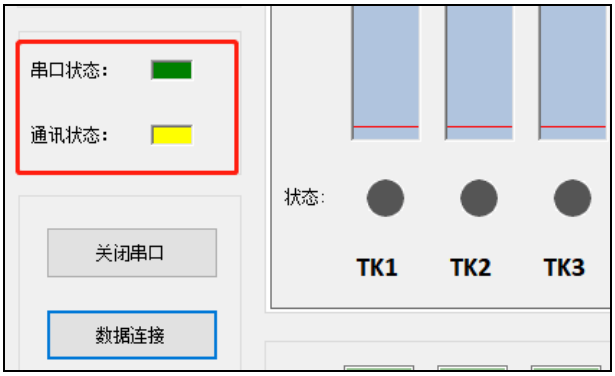


图 4-6: 数据连接成功画面

这时便可观察测试触摸通道时的状态变化。

但若无以上图示现象，则说明通讯口并未选择正确。请右击左下角Windows，选择“设备管理器”→“端口”，找到匹配通讯口。如下图：

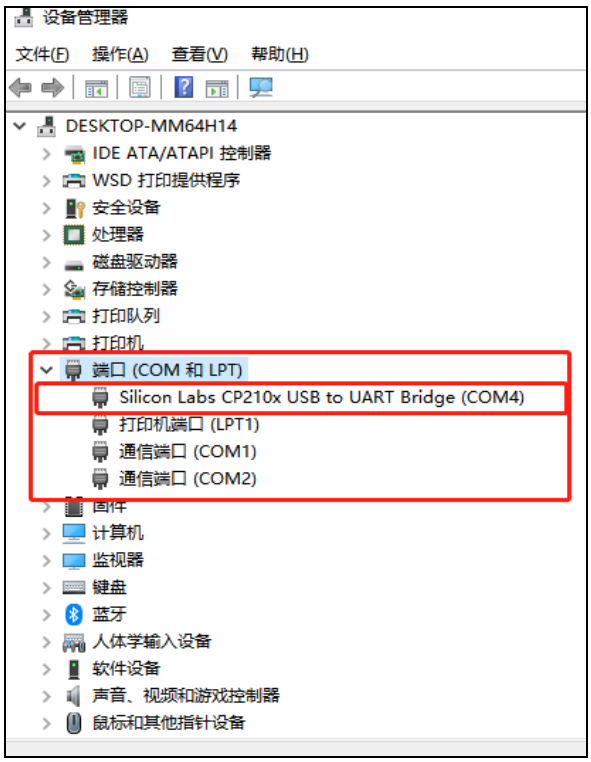


图 4-7: 选择匹配端口

5. T_Watch 有 “强度值”和“数据图”两种表现形式。图4-8为“强度值”画面，图4-9为“数据图”画面。

(1) 强度值

图 4-8: 强度值界面

强度值显示界面可以根据您程序中实际设置的触摸灵敏度等选项，实时判断触摸按键是否触发。

① 当有外部按键时，对应按键的“强度”和“状态”会以图示的鲜明颜色呈现，如上图红色方框内所示。强度的蓝条显示触摸的强度，状态小圆圈由灰色变为绿色时表示按键触发。

② 每个 TK 的强度显示框下方均有一条红色的线，为触发红线。如上图绿色方框标记。只有当蓝色强度条高过触发红线，状态小圆圈才会变为绿色表示存在外部触摸。

③ 在界面下端，显示对应的实际值，如蓝色方框内所示，它和触摸面积有关。

©Copyright 2020, PADAUK Technology Co. Ltd

(2) 数据图

图 4-9: 数据图界面

“数据图”除显示形式与“强度值”不同外，还有四处参数可供用户修改。

① 环境值显示：环境值显示默认为 √。当不勾选此选项时，显示界面只显示上图④中所选 TK 的实时实际值，如⑤所示；当勾选此选项时，环境值会显示在界面上，其颜色较实际值颜色更深更暗，如下图 4-10 所示：

图 4-10: 数据图界面

② TK 允许数量：最大允许同时显示 3 个触摸通道的值。

③ 显示分辨率： 有 10Bit,11bit,12bit 三种选择，用户可自己调整观看显示区别。

④ 勾选要显示的 TK 通道： 请根据②的数量选择勾选此处。

⑤ 显示：上图 4-9⑤处所示为没有外部触摸时的显示情况，当有触摸时，实时显示如下图 4-11。

©Copyright 2020, PADAUK Technology Co. Ltd

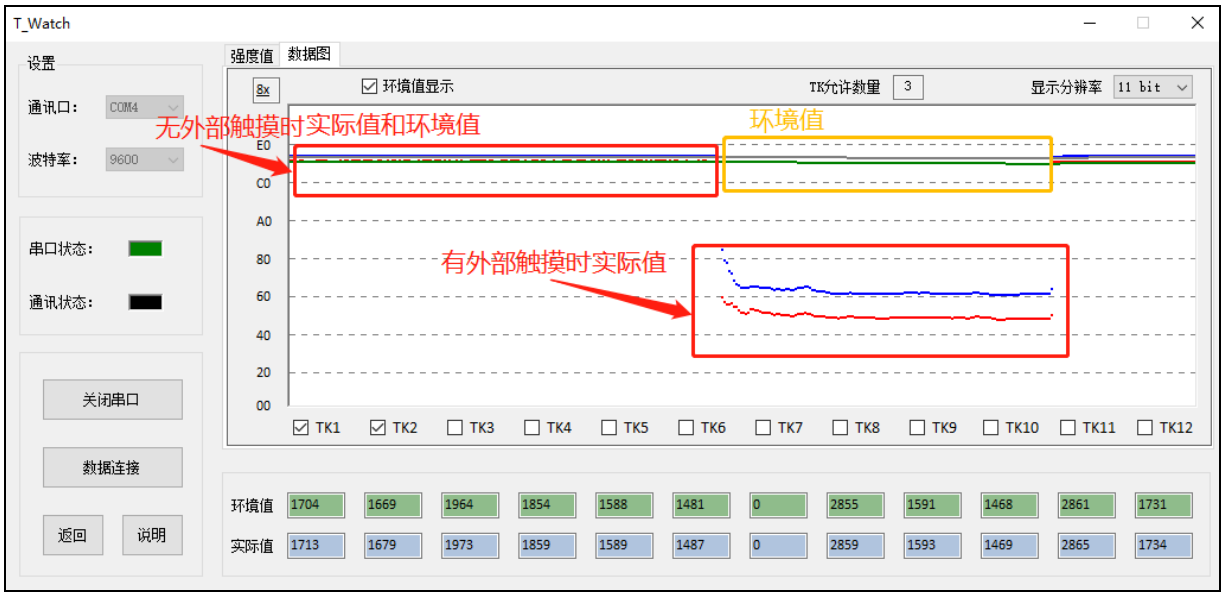


图 4-11: 数据图界面

⑥ 环境值与实际值：含义在“强度值”部分已经介绍，这里不再赘述。

4.2. 工程方案

由于在上一节 4.1 中详细介绍了 T_Watch，此处便只介绍“工程方案”与“仿真器测试”两个功能模块在使用 T_Watch 时前两个步骤的区别，其余操作便不再赘述。

1. 同样，首先依照下图将串口与仿真器/实际芯片正确连接。

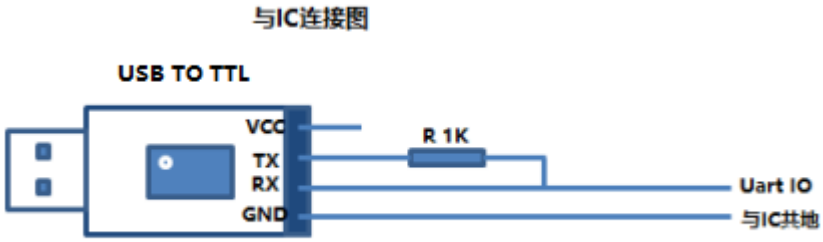


图4-12: Uart与仿真器/IC连接图

注意：对于工程方案，仿真时仅支持PA0做Uart IO，测试实际芯片时PA0中断、PA5和PB0均可。

2. 使用仿真器来测试工程方案时，打开已经编写好的工程方案，将仿真器与 PC 成功连接后，使用最新版 IDE 打开程序并运行（Run），打开 P-Touch → T_Watch。
   烧录实际芯片测试工程方案时，打开P-Touch →T_Watch。

3. 打开串口，待数据连接成功后，即可使用 T_Watch 观察每个 TK 的实际情况。

5. 仿真触摸板测试

该功能模块产生的程序仅用于测试触摸板。将仿真器与 PC、触摸板正确连接后，运行程序，用手指随意触摸选择的 TK 通道（触摸按键），便可通过观察通道数值变化来确认触摸按键能否正常工作。

通道数值显示有两种方式：IDE 显示 ，T_Watch 显示。

本节仅介绍测试程序生成步骤及如何使用 IDE 观测显示数据，有关“T_Watch 显示”的内容请参考 T_Watch 章节（第 4 章）。

注意：进行程序设置前，需先点击“设置”选择最新版 IDE。

5.1. 打开

打开桌面上的 P-Touch，点击“工具”→“仿真器测试”

图 5-1: P-Touch 打开界面

之后出现如下界面：

图 5-2: 触摸仿真板测试打开界面

5.2. 命名&显示方式

自定义“程序名称”及“保存路径”，选择触摸时按键数值变化的显示方式，默认“IDE 显示”（有关“T_Watch 显示”的内容请参考 T_Watch 章节）。

如下图：

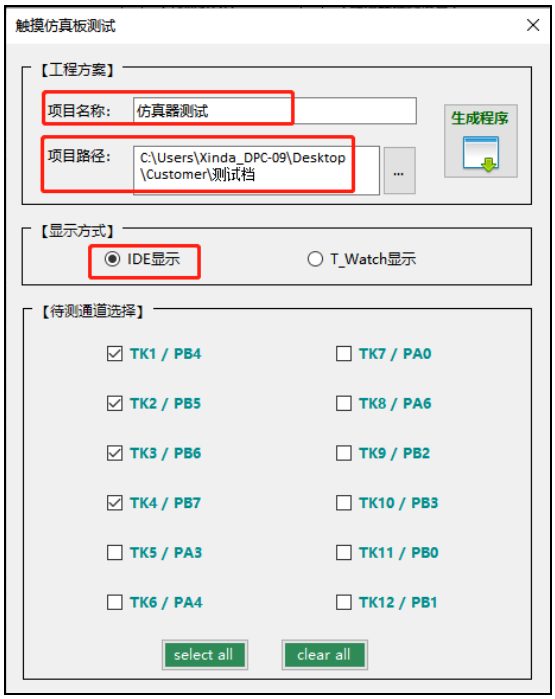


图 5-3: 仿真器测试设置界面

5.3. 点选待测试通道

用户可根据需要自由点选“**待测试通道**”，也可直接点击“**select all**”选择全部通道，或点击“**clear all**”取消所有已选择通道。具体界面参考下图，下文示范以此设置为例：

图 5-4: 自由点选测试通道

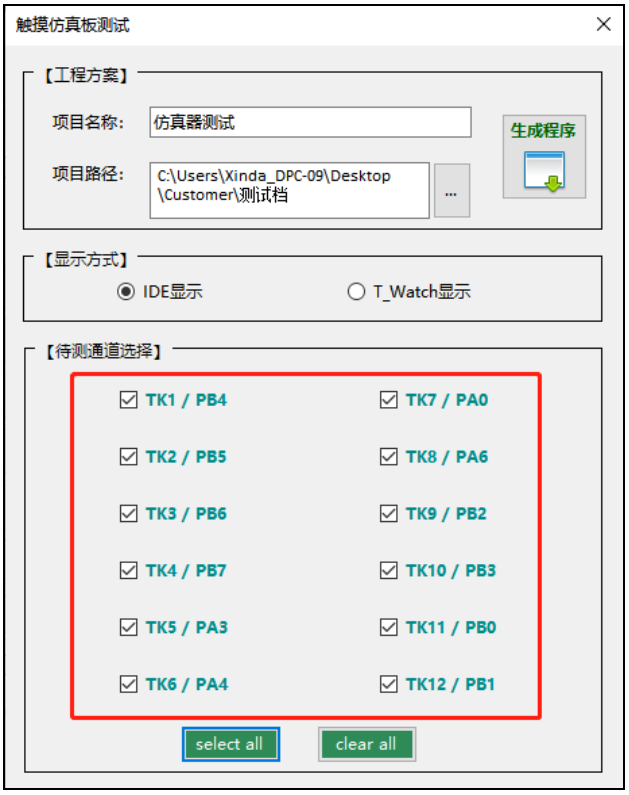


图 5-5: “一键全选”全部通道

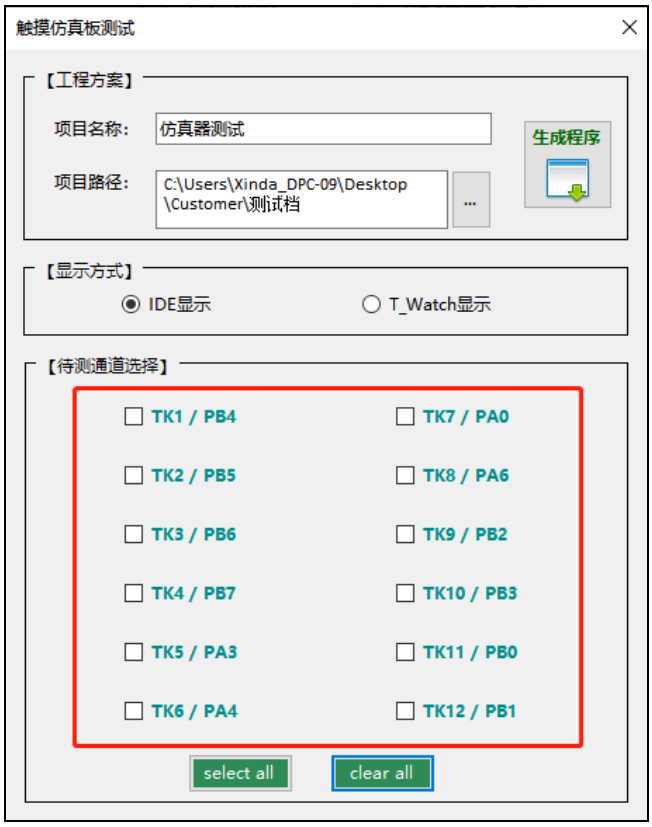


图 5-6: “一键反选”全部通道

5.4. 生成程序

1. 点击“生成程序”。出现如下界面：

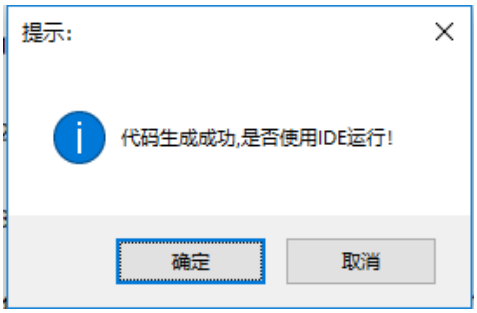


图 5-7: 代码生成后界面

用户可直接点击“确定”使用 IDE 运行程序（如图 5-7），或点击“取消”，从其所属文件夹中使用 IDE 打开.PRJ文件（如图 5-8）。

图 5-8: 代码文件夹打开界面

程序打开后，IDE 视窗画面如图 5-9 所示：

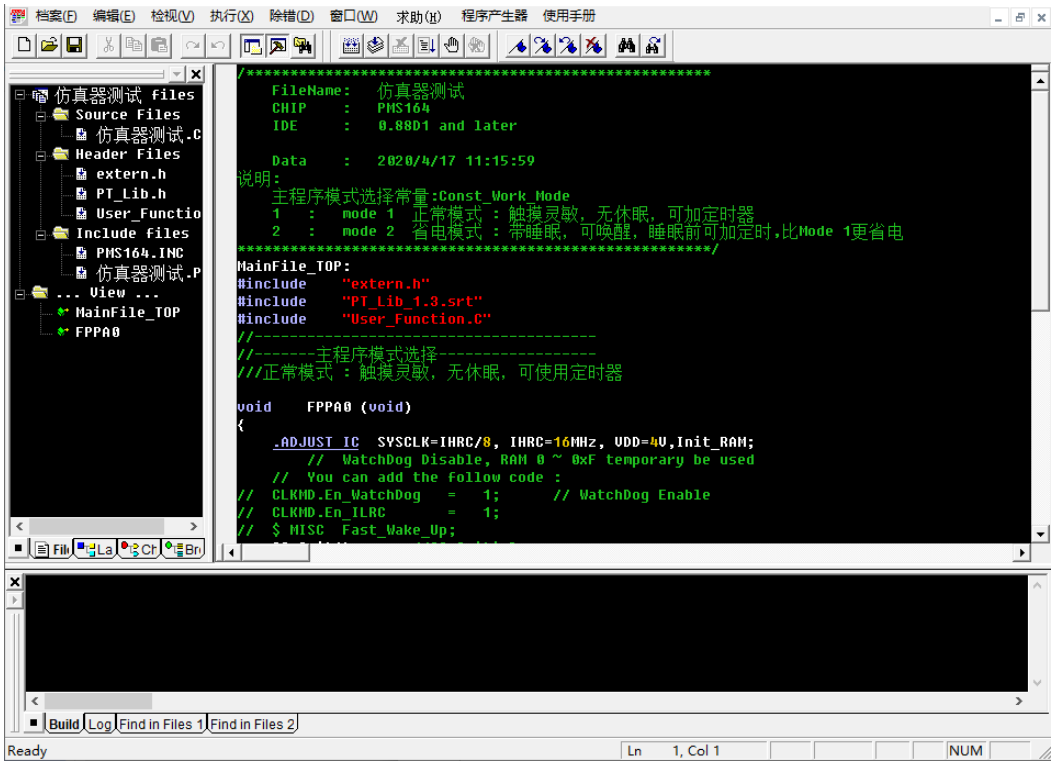


图 5-9: 代码打开后界面

2. 确保将仿真器与电脑、触摸板连接无误后，运行程序。

注意：连接方法请参考群文件 ICE-TouchKit User Manual 或者联系 FAE 以获得资料。

用户可从左下角的 log 窗口中获取版本信息，修改历史及触摸结果等，如下图 5-10。

当有外部触摸时，K1-K12SL 的值为 1，反之为 0。

每次有按键时都会更新所选通道的信息，包括：按键名称、触摸信号、触摸实时值及触摸环境值。

```
Current Version : PMS164 Touch Library V1.3
Current Version date : 2020/04/13
Release date : 2020/04/13
Update record:
*V1.0:
First release.
*V1.1:
1.Add reference channels,touch test,slider and other new functions.
2.Resolve conflicts with single-line debugging.
3.Fix some other bugs.
*V1.2:
1.Add Direction slider,power saving mode to wake up with IO,
Uart wake up at available level ,Coping strategies for CS
(Conducted Susceptibility) and other new functions.
2.Optimize the error triggerring of touch kit when it is
powered on at the first time.
3.Optimize some other problem.
*V1.2B:
1.Solve the accidental wake-up under power saving mode when enable CS(Conduct
*V1.2E:
1. Fix some holes where local variables affect touches
*V1.3:
1.PMS164 rename the release.
.....
T_Key_Debug:
T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Sl:是否有触摸信号
K1-K12SL: 0    0    0    0    0    0    0    0    0    0    0    0
K1-K12Rl:触摸实时值
K1-K12Rl:4A4  48E  559  503  000  000  4FF  521  000  000  000  000
K1-K12Rf:触摸环境/参考值
K1-K12Rf:4A4  48E  559  503  000  000  4FF  521  000  000  000  000

Debug_Num=1 , T_Key_Signal=0018
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12SL: 0    0    1    1    0    0    0    0    0    0    0    0
K1-K12Rl:4A3  489  269  281  000  000  4FE  51E  000  000  000  000
K1-K12Rf:4A3  48D  558  502  000  000  4FE  520  000  000  000  000

Debug_Num=2 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12SL: 0    0    0    0    0    0    0    0    0    0    0    0
K1-K12Rl:4A4  48D  557  502  000  000  4FE  520  000  000  000  000
K1-K12Rf:4A3  48D  557  502  000  000  4FE  520  000  000  000  000
```

图 5-10: IDE 信息显示窗口

6. CS测试应对办法

6.1. 总述

为了帮助 PMS164/PFC161 相关产品通过 CS 测试特制定此办法。

6.2. 分述

6.2.1. PCB Layout

要求按附件相关说明执行。

电容式触摸按键面板PCB设计方案-1.

6.2.2. 注意事项

关于软件参数上要求按以下说明操作：

(1) CS 应对策略使能要打开并至少选择一个参考通道。
(2) 参考通道基准电压要小于 Tk 基准电压，参考通道修正基数如无特殊要求请保持默认 1。
(3) 快速恢复开关请保持关闭状态。
(4) 相关 TK 触摸通道灵敏度不能太大，一般建议要求（90~180）。
(5) CS 电容不能过大，一般建议 10nf 左右。

特殊说明：

CS测试并未使用省电模式，如使用省电模式，为避免影响过大，不应频繁进入睡眠，一般系统待机（无操作，无输出）30s左右再进入睡眠模式即可。

当芯片的CS电容>10nF，且CS应对策略使能打开时，应将触摸寄存器设置（第3.1.6节）中的TK放电时间由默认值32CLK修改为64CLK或128CLK。

©Copyright 2020, PADAUK Technology Co. Ltd