P-Touch_V1.8

使用手册

第 1.00 版

2022 年 4 月 14 日

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。

目 录

修订历史:

修　订	日　期	描　　述
0.00	2021/12/8	初版
1.00	2022/4/14	第二版，新增 PMS160

1. 简介

P-Touch 是应广触摸系列芯片配套的程序框架产生器软件，用于用户触摸方案的前期开发和仿真测试，软件内有触摸调试工具 T-Watch，可实时显示触摸的强度和变化。其使用简单，操作便捷，可视感强，用户能够轻松高效的生成程序框架并测试目标工程方案。

P-Touch_V1.8 相较于以前的版本有稍微大一点的更动，新增滑条与滑环功能模块，从主界面分类、功能视窗到各模块分布等都做出了优化调整，该使用手册将针对 P-Touch_V1.8 作出详细的描述和介绍。

图 1-1：P-Touch_V1.8 图标

P-Touch_V1.8 需搭配 0.93 及以上版本 IDE 使用。

用户可至应广官网下载 P-Touch_V1.8 及 IDE。

欢迎扫描一下二维码，加入“应广触控单片机技术讨论群”。

群名称:應廣芯達 觸控單片機技術討...
群　号:710107052

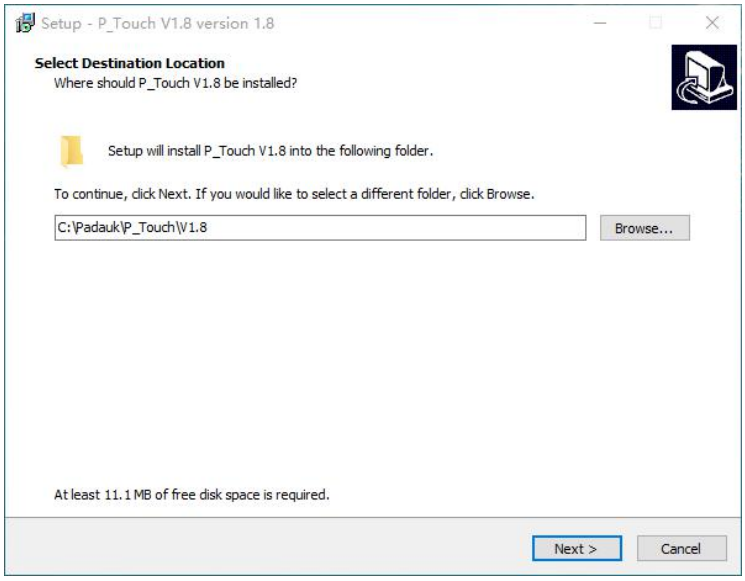
2. P-Touch_V1.8 安装

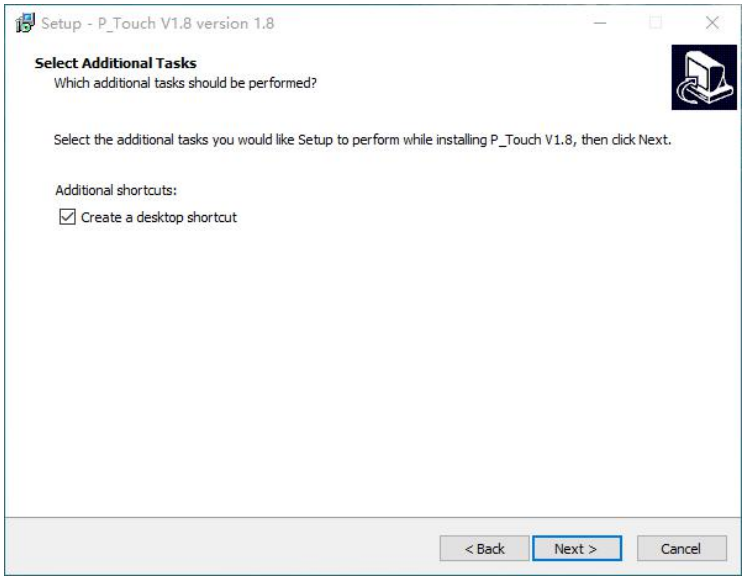
注意：在执行下载或安装过程中，可能会有部分杀毒软件误判，若遇到此情况，请添加信任或关闭杀毒软件即可。

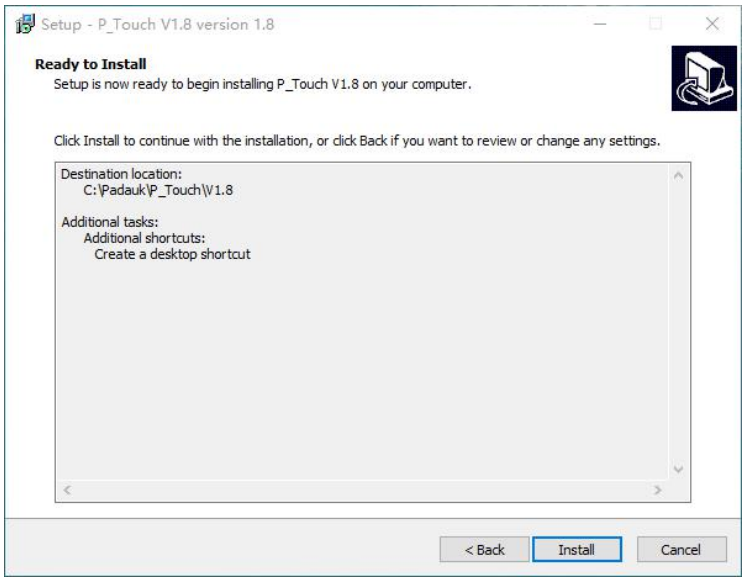
步骤 1：下载软件安装包

请至应广官网下载最新版本：http://www.padauk.com.tw/cn/technical/index.aspx

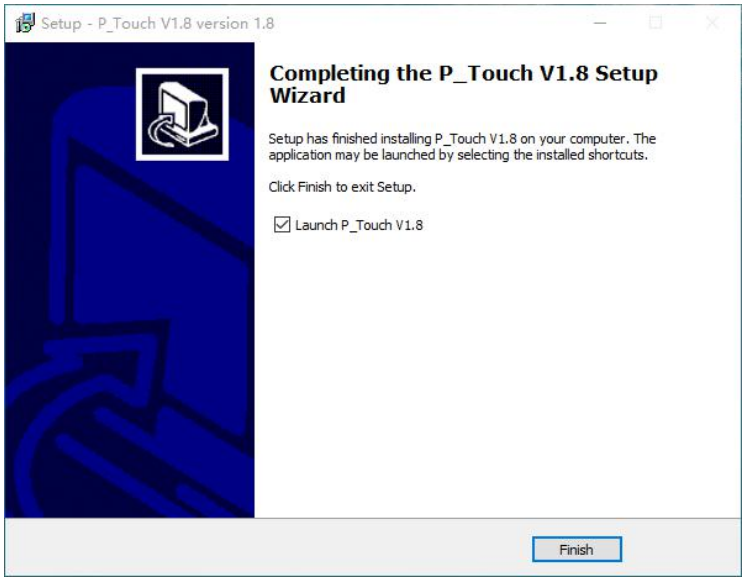
步骤 2：双击安装软件，并根据步骤提示一步一步执行；


Setup - P_Touch V1.8 version 1.8
Select Destination Location
Where should P_Touch V1.8 be installed?
Setup will install P_Touch V1.8 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Padauk\P_Touch\V1.8
Browse...
At least 11.1 MB of free disk space is required.
Next >
Cancel


Setup - P_Touch V1.8 version 1.8
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing P_Touch V1.8, then click Next.
Additional shortcuts:
Create a desktop shortcut
< Back
Next >
Cancel


Setup - P_Touch V1.8 version 1.8
Ready to Install
Setup is now ready to begin installing P_Touch V1.8 on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Padauk\P_Touch\V1.8
Additional tasks:
Additional shortcuts:
Create a desktop shortcut
< Back
Install
Cancel

步骤 4：安装完成。



3. P-Touch_V1.8 主界面介绍

运行 P-Touch_V1.8，如下图，软件分为三个功能模块：触摸工程方案、滑条与滑环方案、触摸标准品选型表，下面会一一介绍。

图 3-1：P-Touch_V1.8 主界面

点击右上方向下箭头展开主页菜单，可以切换软件中英文显示及关于软件版本更新的历史记录信息；

图 3-2：主页菜单

4. 触摸工程方案介绍

此模块用于生成触摸类程序框架，使用时请根据软件上方箭头索引【工程信息】、【通道选择】、【参数设置】等一一配置，然后生成程序，用户可在框架中对应模块完成自己的功能。

4.1. 方案配置说明

4.1.1. 工程信息

需先设置基本的工程信息，包括工程名称、工程路径；选择芯片名称、封装信息、应用类型，是否启用状态滑条，是否启用 T-Watch 以及运行的 IDE 版本；封装信息右侧的按钮可以查看当前选择的封装示意图，在开启 T-Watch 时，需配置唤醒方式和通讯口，T-Watch 介绍在后续会讲到。右侧为当前可选择的触摸 MCU 简介，MCU 详细说明还需参考规格书。

图 4-1：工程信息

4.1.2. 通道选择

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，一般用于不需要计较功耗的应用，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；

CS 引脚：若选择的芯片有多个可选择的外部电容脚位，可根据需求选择合适的外部电容脚位；

触摸通道选择：先单击选择需要的触摸通道（Step1），再点击其右侧的右移箭头（Step2），然后中间的通道列表中就会显示所选择使用的所有通道及其初始参数配置（Step3）；如果选择的是省电模式，还可配置 IO 唤醒设置（Step4）；

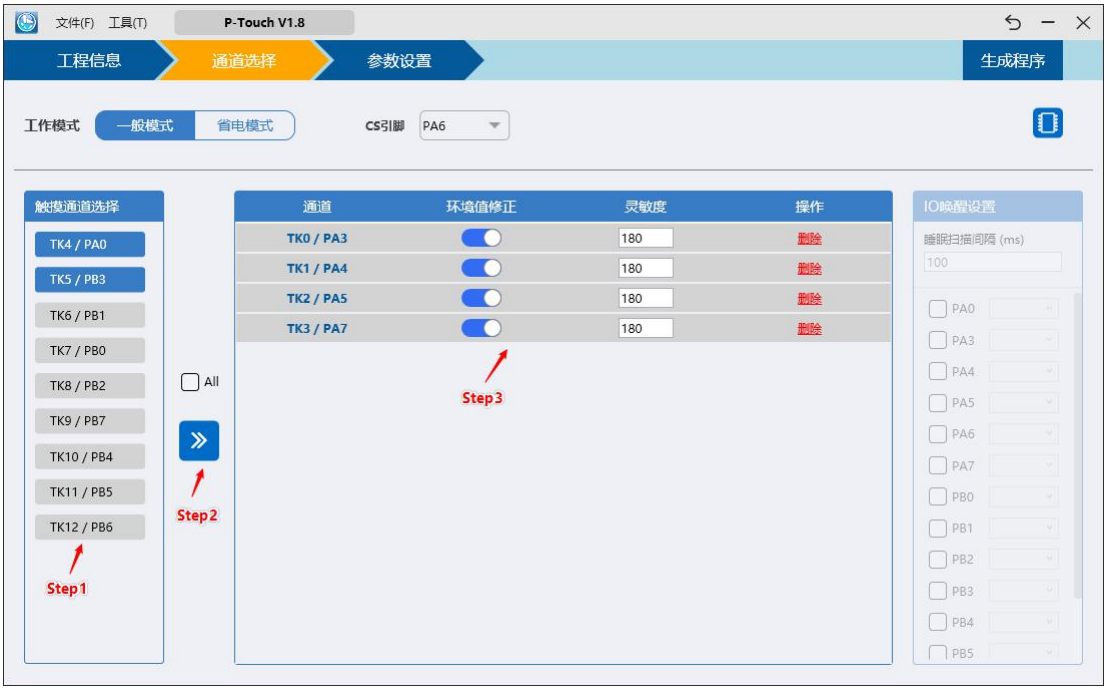


图 4-2：一般模式通道选择

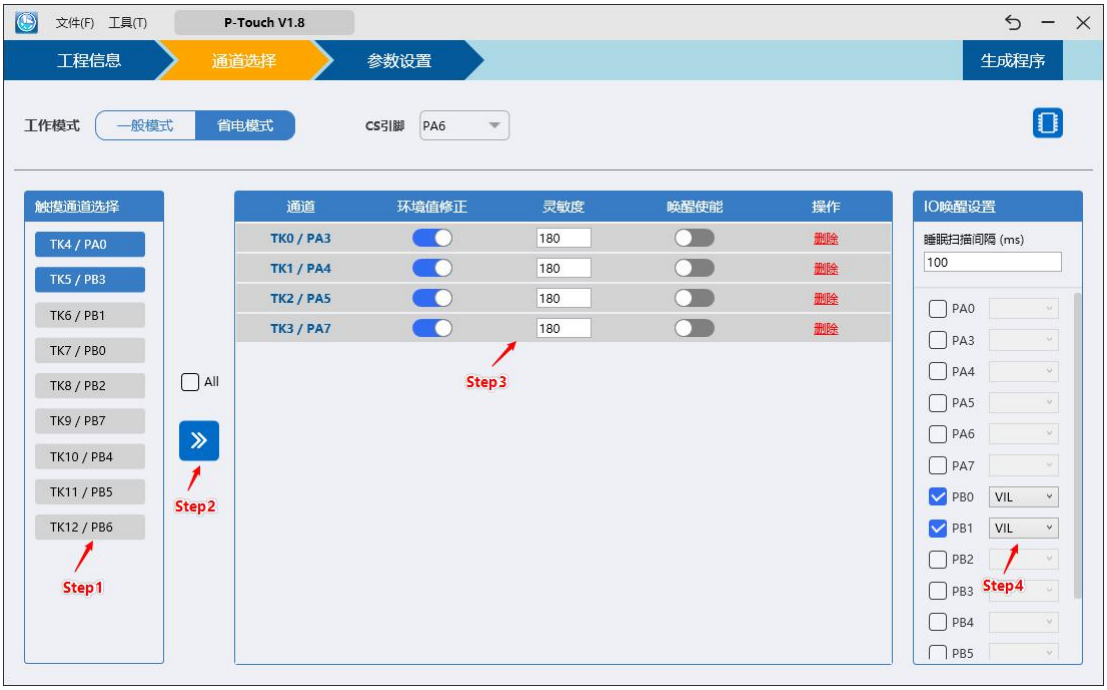


图 4-3：省电模式通道选择

参数介绍：

环境值修正：表示触摸环境值在按键被触发时是否仍会实时修复；

灵敏度：可调范围为 0~210，一般默认值为 180，但 PMS161 默认值 150，其数值越大,表示灵敏度越高；

唤醒使能：省电模式下的选项，选择后表示可用此通道来唤醒休眠；

睡眠扫描间隔：省电模式下的参数，表示睡眠时扫描按键的时间间隔，可调范围为 50~500ms，一般默认值为 100ms；如果需要提高唤醒速度，可适当微调减少扫描间隔，如果需要降低待机电流可适当微调增加扫描间隔；

IO 唤醒设置：在省电模式下除了用触摸唤醒系统外，也可以在此处设置用外部的 IO 唤醒系统。所以此处可选

择 IO 唤醒的端口及唤醒的电位，一般情况下为低电平唤醒；如需高电平唤醒，请将 IO 外接下拉电阻，以保持 IO 常态为低电平。

4.1.3. 状态滑条

程序中目前可提供 6 键 11 阶的状态滑条功能，其中 PMS164 可最多支持 12 键 23 阶，其按键数（N）和滑条阶数（M）的关系为：M=2N-1；状态滑条至少要用到两个按键，例如：两键三阶滑条，其状态为：单独按到键一、同时按到键一与键二、单独按到键二。另外注意滑条模块通道与触摸通道及后面的 CS 应对策略中的参考通道不可重复使用。如要使用到状态滑条，需在工程信息栏中勾选启用状态滑条，则会多出此一栏；

滑条类型：包括普通滑条和方向滑条，方向滑条兼容普通状态滑条并且包含滑条滑动方向，且方向滑条仅在正方向和负方向滑动时有效；

灵敏度：滑条灵敏度范围及默认值和触摸通道相同，数值越大，灵敏度越高；

唤醒使能：如果在通道选择栏选择的是省电模式，则可以在滑条每个键位下设置唤醒功能；

图 4-4：一般模式状态滑条

图 4-5：省电模式状态滑条

4.1.4. 参数设置

因芯片型号不同，参数会有所差异；PMS163、PFC460、PMS164、PMS161、PFC161 参数设置：

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，触摸实际值越大。注意，此处并非是系统时钟；

(2) **TK 基准电压值：**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) **CS 放电时间值：**触摸前的 CS 放电时间，一般 CS 电容越大需要的放电时间越长；

(4) **CS 强制放电时间值：**触摸前 CS 放电时间扩展选择，当 CS 过大，放电 128CLK 仍然不能完全放电/充电时，可以打开此选项，进行 CS 强制放电/充电，其效率要高于上个参数 CS 放电时间值，默认为 50us；

基本防噪声设置

(1) **TK 滤波等级：**程序中采用的是多次采样取平均值的滤波方式，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，滤波越平滑，防噪声能力越强，但同时采样耗时越长，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来选择滤波等级，默认等级为 3；

(2) **TK 释放裕量：**当触摸被触发时，触摸值回调至低于触发阈值一定值(TK 释放裕量)才认为是触摸释放，此参数需谨慎调整，若设置过大，可能会导致触摸被触发后无法释放，取值范围 0~20，默认值为 10；

(3) **环境值修正总使能：**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) **快速恢复开关：**触摸按键快速恢复开关，开启后在按键释放后快速恢复环境值；

(5) **环境值下调难度：**环境值下调修正难度，值越大修正越慢，值为 100 表示至少采样 100 笔偏小的资料才开始下调 1，取值范围 1~10000，默认值为 10；

(6) **高灵敏度使能：**对于应用在超厚隔板的触摸环境中（隔板大于 8mm），一般灵敏度无法满足需求时需开启高灵敏度使能；

图 4-6：参数设置

CS 应对测试设置

(1) **EMI 使能**：开启后，IHRC 会左右跳动，如此在传导和辐射测试时，量测到的谐波功率 db 值会降低一些；

(2) **CS 应对测试使能**：在产品需要过 CS 测试时可以选择开启此选项；

(3) **参考通道一、二**：在干扰较大时，请至少打开一个参考通道，需按顺序开启；可选择不需要使用的通道或隐藏在 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，请勿和使用的触摸通道混用；

(4) **参考通道基准电压**：触摸参考电压选择，对触摸灵敏度的表现和 CS 电容大小的选择有影响，理论上参考电压越大，灵敏度越高；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；

(5) **参考通道修正基数**：通过参考通道修正环境值的基数，取值范围 1~6，默认为 1；

PMS160 参数设置：

触摸寄存器设置

(1) **TK 时钟频率**：IFC 触摸时钟选择，取值选项： 0:32MHz，1:16MHz，默认选项 16MHz；

(2) LDO 电压：IFC 触摸 LDO 电压选择，取值选项: 0:1.8V, 1:reserved, 2:1.7V, 3:1.6V，默认选项 1.8V；

(3) TK 参考电容系数：IFC 触摸参考电容系数选择,此处选择影响 IFC 读数，取值选项: 0:*1, 1:*2, 2:*3, 3:*4, …… ,N:*(N+1) N<=15，默认选项为 8；

(4) **IFC 模式**：IFC 模式选择，取值范围：0: Mode0, 1: Mode1，默认选项 Mode0；

(5) IFC 脉冲计数分频：Mode0 参数，IFC 脉冲计数分频，取值范围: 0:/1, 1:/2, 2:/4, 3:/8, 4:/16, 5:/32, 6:/64, 7:/128, 8:/256，默认选项：0；

(6) IFC 脉冲计数上限：Mode0 参数，IFC 脉冲计数上限选择,此处选择影响 IFC 读数，取值范围：1~255，默认取值：50；

(7) IFC 读数时间：Mode1 参数，IFC 读数时间选择，单位 us，精准度 10us (启动 IFC 多久读取 IFC 值,等待时间越长，IFC 值越大，影响灵敏度)，取值范围: 100 ~ 3000，默认选值: 1000；

参数复烧设置

(1) **多次复烧参数设置**：多次复烧参数使能设置，可复烧参数 Const_SEN_T_KeyX(灵敏度)、

Const_IFC_Count(Mode0 IFC 计数上限)、Const_IFC_Timing(Mode1 IFC 读数时间)最大允许复烧次数；

(2) **最大允许复烧次数：**最大允许复烧次数设置，取值范围：1~8，默认选值：3；

基本防噪声设置

(1) TK 滤波等级：TK 采样移动滤波等级,数值越大滤波越平滑,采样耗时越长,按键触发越慢，取值选项: 1:1/2, 2:1/4, 3:1/8，默认选值：2；

(2) **TK 释放裕量：**TK 释放裕量，当 TK 触发时，触摸值回调至低于触发阈值一定值(TK 释放裕量)才认为是触摸释放，专业参数，谨慎调整，若裕量设置过大，可能会导致 TK 被触发后，无法释放，取值范围: 0-20，默认选值: 10；

(3) 环境值修正总使能：环境值修正总开关，默认开启；

(4) **无按键时环境值修复计数：**空闲时，环境值修复延后计数，即相对与采样多少次修复一次环境值，取值范围: 1 ~ 100，默认选值: 10；

(5) 有按键时环境值修复计数：繁忙(按下)时，环境值修复延后计数，即相对与采样多少次修复一次环境值，取值范围: 1 ~ 100，默认选值: 10；

(6) **无按键时环境值修复速度：**空闲时，环境值修复速度，值越小修复速度越快，取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64，默认选项: 5；

(7) **有按键时环境值修复速度：**繁忙(按下)时，环境值修复速度，值越小修复速度越快，取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64，/默认选项: 5；

图 4-8：PMS160 参数设置

4.1.5. 应对策略

此应对策略仅支持 PMS164 及 PMS161，主要应对抗强干扰与超长类触摸应用（例如抗对讲机干扰与鱼漂类应用）；

抗强干扰对策

(1) **抗强干扰对策使能：**开启即启用此应对策略；

(2) **参考通道一、二：**用于侦测强干扰的触摸通道环境值修复速度；

(3) **环境值修复速度：**检测为无干扰的次数大于 该值时，修复环境值；

(4) **环境值修复比例：**环境值以一定的速度靠近实际值，值越大，修复速度越慢，具体为 上一次环境值与实际值的差值除以 2^n；

(5) **参考通道抖动阀值：**参考通道的实际值抖动超过 该值,则判定异常；

(6) **触摸深度阈值设置：**表示触摸通道受到强干扰时的误触发阈值，具体为不能超过 （220 - 参考通道的灵敏度）×该值；

超长触摸对策

(1) 超长触摸对策使能：开启即启用此应对策略；

(2) **异常按键释放检测速率：**判定为达到释放点的次数大于该值时，修复环境值；

(3) 异常按键释放检测灵敏度：设置释放点，则释放位置为 原释放位置下移 环境值与实际值的差值除以 2^n；

图 4-9：应对策略

4.1.6. 菜单栏介绍

1. **文件：**包含打开配置和保存配置，在用户设定配置及参数生成程序框架后都会在程序包中生成一个与工程同名的 ini 格式的配置文件，用于记录用户设定的配置及参数。当然，用户也可以手动点击保存配置文件来保存当前的配置及参数；点击打开配置可以用此软件打开配置文件中的设定；
2. **工具：**点击可打开 T-Watch 触摸调试工具，此工具使用说明会在后续讲到；

4.2. 生成方案框架

如上述介绍，按需求配置完后，点击软件右上角生成程序按钮，即可生成程序框架，然后会弹出如下窗口，提示是否直接用 IDE 运行程序。

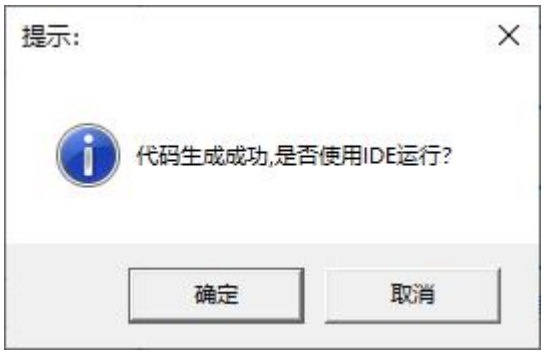


图 4-10：提示生成程序框架

用 IDE 运行生成的方案后，即可基于此框架编写实际的工程程序。下面具体介绍各个工程文件（举例我们生成一个 Touch_Demo 框架，程序中的文件架构图如下）：

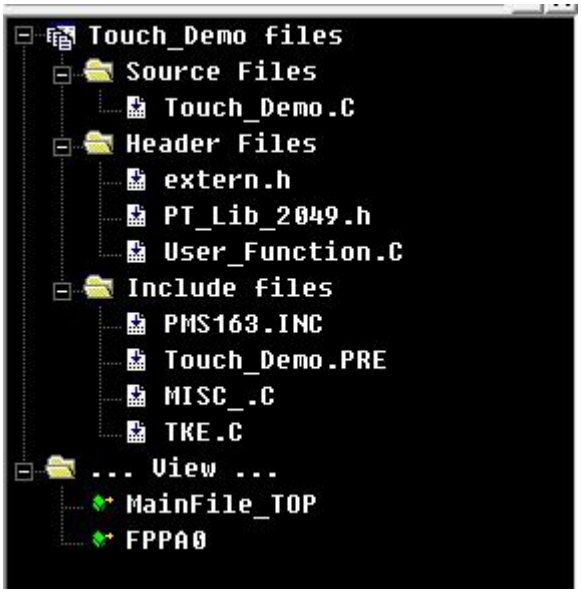


图 4-11：程序文件架构

- Touch_Demo.C 是主程序文件，可以对使用的 IO 脚进行输入输出和数字使能等相关操作及选择主程序的工作模式；
- Extern.h 是扩展的程序头文件，里面包含芯片名称、CS 引脚、系统时钟和 UART 时钟等信息；
- PT_Lib_xxxx.h 是触摸库的配置文件，里面记录了所使用的触摸通道信息、参数配置、UART 设置等信息；
- User_Function.C 是用户的功能编写文件，用户可在其中对应模块根据自身需求写入需要的功能；

下面来详细介绍其中几个主要程序文件。

4.2.1. 触摸库配置文件

触摸库配置文件 PT_Lib_xxxx.h 文件用于配置触摸相关的设定，包括触摸引脚选择，灵敏度配置，唤醒引脚配置，环境修正参数及 Uart 使能等。

注意：PT_Lib_xxxx.h 文件说明旨在帮助用户理解程序，文件内的参数设置在 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。

1. **触摸通道使能设置：（Const_EN_CH_T_Key）**

通道使能设置为 1：开启通道；通道使能设置为 0 或注释掉：关闭通道

```
    //T_Key通道使能设置
Touch_Channel_Selection:
    Const_EN_CH_T_Key0  =>  1
    Const_EN_CH_T_Key1  =>  1
    Const_EN_CH_T_Key2  =>  1
    Const_EN_CH_T_Key4  =>  1
    Const_EN_CH_T_Key5  =>  1
```

图 4-12：TK 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，并且关闭上拉电阻，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，可先将所有可能用到的通道打开。

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：开启通道唤醒；触摸通道唤醒设置为 0 或注释掉：关闭通道唤醒；

```
\******当主程序用省电模式时,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_Key0  =>  1
    Const_Wakeup_CH_T_Key1  =>  1
    Const_Wakeup_CH_T_Key2  =>  1
    Const_Wakeup_CH_T_Key3  =>  1
```

图 4-13：TK 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，范围为 0~210；一般默认为 180，但 PMS161 默认值 150；数值越大，灵敏度越高；

```
    //T_Key灵敏度sensitivity设置
Touch_Sensitivity_Set:
    Const_SEN_T_Key0        =>  180 //触摸灵敏度设定,数值越大,灵敏度越高
    Const_SEN_T_Key1        =>  180 //取值选项: 0-210 default: 180
    Const_SEN_T_Key2        =>  180
    Const_SEN_T_Key4        =>  180
    Const_SEN_T_Key5        =>  180
```

图 4-14：TK 灵敏度设置

4. 环境值修正使能设置：（Const_Press_Fix_T_Key）

触摸按键触发后，是否继续修正环境值，设置为 0：继续修正；设置为 1 或注释掉：不修正；

```
//触摸按键触发后，是否继续修正环境值//0: 继续修正;  1: 关闭
Const_Press_Fix_T_Key0  =>  0
Const_Press_Fix_T_Key1  =>  0
Const_Press_Fix_T_Key2  =>  0
Const_Press_Fix_T_Key3  =>  0
```

图 4-15：环境值修正使能设置

5. 状态滑条设置：（Const_En_Slider）

Const_En_Slider_A 设置为 1：启用状态滑条；设置为 0：不启用状态滑条

Slider_T_Key 设置状态滑条的通道及顺序，目前最多支持 6 个通道，灵敏度范围及默认值与触摸通道相同；

```
Touch_Slider_Set:
    Const_En_Slider_A    =>  1    //6Key方块滑条使能
    Slider_T_Key1    =>  'TK0';
    Slider_T_Key2    =>  'TK1';
    Slider_T_Key3    =>  'TK2';
    Slider_T_Key4    =>  'TK3';
    Const_SEN_Slider_A=>    180    //滑条灵敏度设置, Max=210,Min=0,数值越大,灵敏度越高
                                ////0:最不灵敏 210:最灵敏 default:180
```

图 4-16：滑条设置

6. 触摸寄存器设置：

因芯片型号不同，参数可选范围也有所不同；下面分别以 PMS164 和 PMS160 为例：

PMS164 参考如下：

(1) 触摸扫描时钟源设置：（**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。默认值为 3：IHRC/8；

(2) 触摸 CS 电容参考电压设置：（**Const_Touch_VRef**）

0：0.5*VCC；1：0.4*VCC；2：0.3*VCC；3：0.2*VCC

触摸 CS 电容参考电压与触摸灵敏度有关，设置为 0 灵敏度最高，设置为 3 灵敏度最低；

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

0：保留；1：CLK_32；2：CLK_64；3：CLK_128；

(4) 触摸转换前 CS 电容放电时间扩展选择设置：（**Const_Touch_Discharge2**）

此为新增加参数，因为但 CS 过大，放电时间为 CLK_128 仍然不能完全充放电时，可打开此选项，进行 CS 强制充放电，其效率高于上个参数，单位为 us，取值范围：0~1000；

```
//实际触摸按键设置
Const_Touch_Source_CLK  =>  3;  //触摸时钟选择
                                //0:reserved, 1:reserved, 2:IHRC/4, 3:IHRC/8,
                                //4:IHRC/16, 5:IHRC/32, 6:IHRC/64, 7:IHRC/128, 8:ILRC
                                //推荐选值: 3
Const_Touch_VRef        =>  0;  //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
                                //推荐选值:0
Const_Touch_Discharge   =>  0;  //触摸前CS放电时间选择, 时间越长, 放电越干净
                                //0:reserved, 1:CLK_32, 2:CLK_64, 3:CLK_128
                                //默认选项: 3
Const_Touch_Discharge2  =>  50; //触摸前CS放电时间扩展选择, 当CS过大, 放电CLK_128任然不能完全放电/充电时,
                                //可以打开此选项, 进行CS强制放电/充电
                                //放电/充电效率高于Const_Touch_Discharge, 单位 US , 取值范围 0 - 1000 us
                                //默认选项: 50
```

图 4-17：触摸寄存器设置

PMS160 参考如下：

(1) IFC 触摸时钟设置：（**Const_IFC_CLK**）

取值选项:0:32MHz, 1:16MHz

默认选项: 1

(2) IFC 触摸 LDO 电压设置：（**Const_IFC_LDO**）

取值选项: 0:1.8V, 1:reserved, 2:1.7V, 3:1.6V

默认选项: 0

(3) IFC 触摸参考电容系数：（**Const_IFC_CAP**）

取值选项: 0:*1, 1:*2, 2:*3, 3:*4, …… ,N:*(N+1) N<=15

默认选项: 8

(4) IFC 模式选择：（**Const_IFC_Mode**）

取值选项: 0: Mode0, 1: Mode1

默认选项: 0

(5) IFC 脉冲计数分频：（**Const_IFC_Scalar**）

Mode0 参数，取值范围: 0:/1, 1:/2, 2:/4, 3:/8, 4:/16, 5:/32, 6:/64, 7:/128, 8:/256

默认选项: 0

(6) IFC 脉冲计数上限：（**Const_IFC_Count**）

Mode0 参数，取值范围: 1~255

默认选值：50

(7) IFC 读数时间：（**Const_IFC_Timing**）

Mode1 参数，取值范围: 100 ~ 3000

默认选值: 1000

```
    //实际触摸按键设置
    Const_IFC_CLK    =>  0;  //IFC触摸时钟选择
                             //取值选项: 0:32MHz, 1:16MHz
                             //默认选项: 1
    Const_IFC_LDO    =>  0;  //IFC触摸LDO电压选择
                             //取值选项: 0:1.8V, 1:reserved, 2:1.7V, 3:1.6V
                             //默认选项: 0
    Const_IFC_CAP    =>  8;  //IFC触摸参考电容系数选择,此处选择影响IFC读数
                             //取值选项: 0:*1, 1:*2, 2:*3, 3:*4, …… ,N:*(N+1) N<=15
                             //默认选项: 8
    Const_IFC_Mode   =>  0;  //IFC Mode 选择
                             //取值选项: 0: Mode0, 1: Mode1
                             //默认选项: 0

C_IFC_Mode0:
    Const_IFC_Scalar=>  0;  //IFC脉冲计数分频
                             //取值范围: 0:/1, 1:/2, 2:/4, 3:/8, 4:/16, 5:/32, 6:/64, 7:/128, 8:/256
                             //默认选项: 0
    Const_IFC_Count =>  50;//IFC脉冲计数上限选择,此处选择影响IFC读数
                             //取值范围: 1~255
                             //默认选值: 50
```

7. 触摸环境修正及抗干扰设置

(1) 触摸采样一次所需的滤波等级设置：（**Const_T_Key_Smooth_Rank**）

滤波等级为 1~6，分别对应采样次数 1 次、3 次、6 次、10 次、18 次、34 次和 66 次，数值越大代表滤波越平滑，当然采样耗时也越长，默认值为 3；

(2) 触摸释放裕量设置：（**T_Key_Release_Margin**）

当触摸触发时，触摸值回调至低于触发阈值，及触摸释放裕量值，才认为是触摸释放；此值需谨慎调整，不可设置过大，否则可能会导致触摸触发后无法释放；取值范围为 0~20，默认值为 10；

(3) 触摸按键环境值修正总开关：（**Const_Env_Fix**）

设置为 1：开启环境值修正总使能；设置为 0：关闭环境值修正总使能；

(4) 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

设置为 1：开启；设置为 0：关闭；

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

设定范围：1~10000；数值越小修正越快，数值越大修正越慢；

例如设定值为 100，表示至少采样 100 笔偏小的实时环境值，才将参考触摸环境值往下调 1；

```
//------Touch环境修正及抗干扰设置-----
Touch_Ref_and_Noise_Set:
    Const_T_Key_Smooth_Rank =>  3   //TK采样滤波等级,数值越大滤波越平滑,采样耗时越长
                                    //取值选项: 1:4次, 2:6次, 3:10次, 4:18次, 5:34次, 6:66次,
                                    //推荐选值: 3
    T_Key_Release_Margin    =>  10  //TK释放裕量, 当TK触发时, 触摸值回调至低于触发阈值一定值(TK释放裕量)才认为是触摸释放
                                    //专业参数, 谨慎调整, 若裕量设置过大, 可能会导致TK被触发后, 无法释放
                                    //取值范围: 0-20
                                    //推荐选值: 10
    Const_Env_Fix           =>  1   //环境值修正总开关
                                    //取值选项: 0:关闭, 1:打开
                                    //推荐选值: 1
    Const_Fast_Recover      =>  1   //触摸按键快速恢复开关(按键释放后快速恢复环境值)
                                    //取值选项: 1:开启 0:关闭
                                    //推荐选值: 1
    Const_Env_Dw_Fix_Cnt    =>  10  //环境值下调修正难度,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
                                    //取值范围: 1 - 10000
                                    //推荐选值: 10
```

图 4-18：环境值修正及抗干扰设置

PMS160 参考部分：

```
Const_Env_Fix_Count_Free=>  10  //空闲时, 环境值修复延后计数, 即相对与采样多少次修复一次环境值
                                //取值范围: 1 ~ 100
                                //默认选值: 10
Const_Env_Fix_Count_Busy=>  10  //繁忙(按下)时, 环境值修复延后计数, 即相对与采样多少次修复一次环境值
                                //取值范围: 1 ~ 100
                                //默认选值: 10
Const_Env_Fix_Speed_Free=>  5   //空闲时, 环境值修复速度, 值越小修复速度越快
                                //取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64   //12禁用
                                //默认选值: 5
Const_Env_Fix_Speed_Busy=>  5   //繁忙(按下)时, 环境值修复速度, 值越小修复速度越快
                                //取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64   //12禁用
                                //默认选值: 5
```

图 4-19：PMS160 环境修正部分参数

(6) 空闲时，环境值修复延后计数：（**Const_Env_Fix_Count_Free**）

相对与采样多少次修复一次环境值，取值范围: 1 ~ 100

默认选值: 10

(7) 繁忙(按下)时，环境值修复延后计数：（**Const_Env_Fix_Count_Busy**）

相对与采样多少次修复一次环境值，取值范围: 1 ~ 100

默认选值: 10

(8) 空闲时，环境值修复速度：（**Const_Env_Fix_Speed_Free**）

值越小修复速度越快，取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64 //12 禁用

默认选项: 5

(9) 繁忙(按下)时，环境值修复速度：（**Const_Env_Fix_Speed_Busy**）

值越小修复速度越快，取值选项: 0:关闭 1:1/2 2:1/4 3:1/8 4:1/16 5:1/32 6:1/64 //12 禁用

默认选项: 5

8. 防噪声参考按键设置

(1) CS 应对策略设置：（**Const_Touch_Noise_Strategy**）

主要应对电源干扰，取值范围为：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；防噪声策略 1 与防噪声策略 3 保留；

防噪声策略 2 针对 CS 测试的应对策略，需要 1 至 2 个参考通道，当干扰较大时，请至少打开一个参考通道，且按顺序打开；

(2) 参考按键准位设置：（**Const_Touch_Noise_Vref**）

因参考按键通常为隐藏与 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，所以要降低准位，防止溢出；

0：0.5*VCC；1：0.4*VCC；2：0.3*VCC；3：0.2*VCC，默认值为 3；

(3) 参考通道修正基数：（**Const_T_Key_Noise_Fix_Cn**）

通过防噪声按键修正环境值的基数，取值范围：1~6，推荐值：1

```
//防噪声参考按键设置
Const_Touch_Noise_Strategy  =>  2   //主要应对电源干扰
                                    //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
                                    //防噪声策略1   保留
                                    //防噪声策略2   针对CS测试的应对策略，需要1至2个参考通道
                                    //防噪声策略3   保留
Const_T_Key_Noise_Ref_1 =>  'TK1'
Const_T_Key_Noise_Ref_2 =>  'TK2'

                                    //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
                                    //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
                                    //取值范围：0:0.5*VCC, 1:0.4*VCC, 2:0.3*VCC, 3:0.2*VCC
                                    //推荐选值:1
Const_T_Key_Noise_Fix_Cn=>  1       //通过防噪声按键修正环境值的基数
                                    //取值范围：1-6
                                    //推荐选值：1
```

图 4-20：防噪声参考按键设置

9. 上位机参数设置

(1) 上位机使能：（**Const_EN_Uart**）

需要开启上位机时，请将 Const_EN_Uart 置能，不用上位机请除能；

(2) 上位机唤醒方式设置：（**Const_Uart_Wakeup_Mode**）

0：低电平唤醒，需将通讯口对地短接大于 0.5s；1：T_Watch 唤醒，点击上位机上的连接按钮即会发送唤醒信号给芯片进行唤醒；

(3) 上位机波特率：（**UART_BaudRate**）

UART 波特率范围：9600~56000，默认值是 38400；需注意记得如果修改此处波特率值，T_Watch 上的波特率需要一起修改；

(4) 上位机通讯口：（**Interrupt_Uart**）

此处选择 UART 中断及传输端口，不同芯片可选端口会有所区别；需注意仿真时不支持 PA5 和 PB0 中断，实际 IC 烧录建议使用 PA5；另外如果芯片选择 PMS160 则可开启在板调试模式，后续上位机使用中会介绍；

```
//-------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
    UART_BaudRate       =>  38400   //UART波特率:9600-56000
                                    //default:38400
    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 4-21：上位机参数设置

注意：如果软件生成的程序已经选择开启 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 函数中将对应的 IO 设置成输入、上拉及数字输入使能。

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       //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 default:0
```

图 4-22：触摸测试开关设置

(1) 触摸测试总开关：（**T_Key_Debug**）

开启后可在 Log 窗口中看到已打开通道的触摸信号；

(2) 继续扫描开关：（**Continuous_Debugging**）

开启后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；

```
T_Key_Debug:
T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:触摸实时值
K1-K12Rl:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:触摸环境/参考值
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000

Debug_Num=1 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  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-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000

Debug_Num=3 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000

Debug_Num=4 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000

Debug_Num=5 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000

Debug_Num=6 , T_Key_Signal=0000
K1-K12TL:TK1  TK2  TK3  TK4  TK5  TK6  TK7  TK8  TK9  TK10 TK11 TK12
K1-K12Rl:1D3  1E1  1D3  1D3  000  000  000  000  000  000  000  000
K1-K12Rf:1D2  1E1  1D3  1D3  000  000  000  000  000  000  000  000
```

图 4-23：Debug Log 显示

(3) 释放部分 RAM：（**Disable_Debug_Var**）

当开启触摸测试时，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：烧录时以上三个使能务必全关闭。

11. 主要函数说明及相关寄存器定义

```
Lib_H_Bottom:
//************************************///
//  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 4-24：触摸库中主要函数说明

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）

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_xxxx.h 文件中置 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）,所以一般情况下不需要自行设置 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	-	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）
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

12. PMS164、PMS161 应对策略说明

主要适用于应对强抗干扰与超长触摸类应用，例如抗对讲机干扰、鱼漂类超长触摸应用；

抗强干扰对策：

(1) 抗强干扰对策使能：（**ANTIJAM_Enable**）

(2) 抗强干扰参考通道一、二：（**Const_T_Key_Antijam_Ref_1**、**Const_T_Key_Antijam_Ref_2**）

(3) 环境值修复速度：（**CK_Fix_Count**）

检测为无干扰的次数大于 该值时，修复环境值，范围 0-255；

(4) 环境值修复比例：（**Antijam_Fix_Speed**）

环境值以一定的速度靠近实际值，值越大，修复速度越慢具体为 上一次环境值与实际值的差值除以 2^n；

(5) 参考通道的抖动的阈值：（**ANTIJAM_REFERENCE_VALUE**）

参考通道的实际值抖动超过 该值,则判定异常

(6) 触摸深度阈值设置：（**ANTIJAM_REFERENCE_SNECEx**）

表示触摸通道受到强干扰时的误触发阈值，具体为不能超过 （220 - 参考通道的灵敏度）×该值；

超长触摸对策：

(1) 超长触摸对策使能：（**LongLong_Touch_Release_Enable**）

(2) 异常按键释放检测速率：（**LongLong_Touch_Fix_Count**）

判定为达到释放点的次数大于该值时，修复环境值；

(3) 异常按键释放检测灵敏度：（**LongLong_Touch_Fix_Sensi**）

设置释放点，则释放位置为 原释放位置下移 环境值与实际值的差值除以 2^n；

```
//抗干扰参数设置
ANTIJAM_Enable              => 1        //抗干扰对策开关  1:使能  0:除能
Const_T_Key_Antijam_Ref_1   => 'TK5'
Const_T_Key_Antijam_Ref_2   => 'TK6'
CK_Fix_Count                => 20       //环境修复速度，范围0-255
Antijam_Fix_Speed           => 4        //环境值修复比例，环境值向实际值以 1/(2^n) 的差值速度接近
ANTIJAM_REFERENCE_VALUE     => 40       //参考通道的抖动的阈值设定

ANTIJAM_REFERENCE_SNECE1    => 5
ANTIJAM_REFERENCE_SNECE2    => 5
ANTIJAM_REFERENCE_SNECE3    => 5

//-------------------------------------
//超长触摸释放对策
LongLong_Touch_Release_Enable   => 1        //超长触摸释放使能
LongLong_Touch_Fix_Count        => 20       //异常按键释放检测速率
LongLong_Touch_Fix_Sensi        => 1        //异常按键释放检测灵敏度 数值越大越灵敏，过大容易误释放，请斟酌调整，建议[1-4]
```

图 4-25：PMS164、PMS161 应对策略说明

13. PMS160 参数复烧说明

(1) 多次复烧参数使能：（**En_Mult_Write_Para**）

可复烧参数 Const_SEN_T_KeyX(灵敏度)、Const_IFC_Count(Mode0 IFC 计数上限)、Const_IFC_Timing(Mode1 IFC 读数时间)

(2) 最大允许复烧次数设置：（**Mult_Write_Para_MaxCnt**）

取值范围 1-8，默认选值: 3；

(3) 当前复烧次数：（**Mult_Write_Para_Cnt**）

当前复烧次数，0 代表没有进行复烧，1 代表第一次进行复烧，取值范围 0-Mult_Write_Para_MaxCnt，默认选值: 0

```
//参数复烧设置
En_Mult_Write_Para      => 1        //多次复烧参数使能设置，可复烧参数Const_SEN_T_KeyX(灵敏度)、Const_IFC_Count(Mode0 IFC计数上限)、Const_IFC_Timing(Mode1 IFC读数时间)
                                    //默认选项：0                 注：其他参数不可修改，否则可能会导致复烧失败
Mult_Write_Para_MaxCnt  => 3        //最大允许复烧次数设置
                                    //取值范围1-8
                                    //默认选值：3
Mult_Write_Para_Cnt     => 0        //当前复烧次数，0代表没有进行复烧，1代表第一次进行复烧
                                    //取值范围0-Mult_Write_Para_MaxCnt
                                    //默认选值：0
```

图 4-26：PMS160 参数复烧设置

4.2.2. 用户功能文件

用户功能文件 User_Function.C 主要用于 IO 及用户变量初始化，用户自定义功能模块的实现。

1. 用户变量的定义与初始化：

用户可在 Variable_Define 下定义自己所需的变量，然后在 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 4-22：变量定义及初始化

2. IO 初始化：

函数 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设置，部分IO P-Touch已经设置
void    IO_Init(void)
{
    //----------------------------
    PA      =   0b0000_0000;
    PAC     =   0b0111_1111;    //1:输出   0:输入
    PAPH    =   0b0000_0000;    //1:加上拉 0:不加上拉
    PB      =   0b0000_0000;
    PBC     =   0b0000_1111;
    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;
    //----------------------------
    //User's Other IO Define
    //----------------------------
}
```

图 4-23：IO 初始化设置

3. 用户自定义功能

用户自定义功能写在 T_Key_Func(void)函数中，对应的按键功能写在对应模块内。其使能配置在 PT_Lib_xxxx.h 文件中；

以 T_Key1 为例，如下图，若程序跳入 if 语句中，则表示触摸按键被触发；若程序跳入 elseif 语句内，则表示触摸按键被释放或者无触摸。

```
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 4-24：按键功能设置

用户其他功能可编写在 User_other_Func(void)函数中；

```
void    User_other_Func(void)
{
    //User's Code
}
```

图 4-25：其他功能设置

4. 滑条模块程序

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。

```
Void    Slider_Function(void)
{
    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;
    default:
    case 0: //滑条释放
        //User's Code
        break;
    }
}
```

图 4-26：滑条设置

5. 省电模式下睡眠函数编写

#define Sleep_Condition (0)：进入省电睡眠的条件

如果用户选择的工作模式为省电模式，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；省电模式前的程序设置可在 Pre_sleep_set(void) 函数中完成；

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

图 4-27：睡眠前功能设置

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 4-28：唤醒后动作设置

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）；

4.2.3. 用户主工程文件

1. 工作模式：一般模式和省电模式

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 =>  1    //取值范围1-2
                         //1 :    mode 1  正常模式 : 触摸灵敏，无休眠，可使用定时器
                         //2 :    mode 2  省电模式 : 带睡眠，可唤醒，可使用定时器
```

图 4-29：工作模式设置

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
        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 4-30：工作模式内函数说明

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 && Sleep_Condition)   //无按键且无需连Uart发送数据时自动进入省电模式
{                   //可加入其他限制条件，如关灯 if(T_Key_Signal==0 && LED==0)等
    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 4-31：睡眠和唤醒函数说明

4.2.4. T-Watch 的使用

T-Watch 是触摸灵敏度调试的上位机，可以来直观的观察触摸按键的状态变化。下面来具体介绍 T-Wacth 的使用方法：

1. 首先在使用 P-Touch 生成触摸程序框架时选择启用 T-Watch，并设置唤醒方式及合适的通讯口；

图 4-32：启用 T-Watch 设置

注意：对于工程方案，仿真时仅支持 PA0 做 Uart 通讯，实际芯片可支持 PA0、PA5 及 PB0；

2. 在 P-Touch 菜单中选择“工具”，点击打开 T-Watch，如下图，使用时用 Uart 连接触摸仿真板；

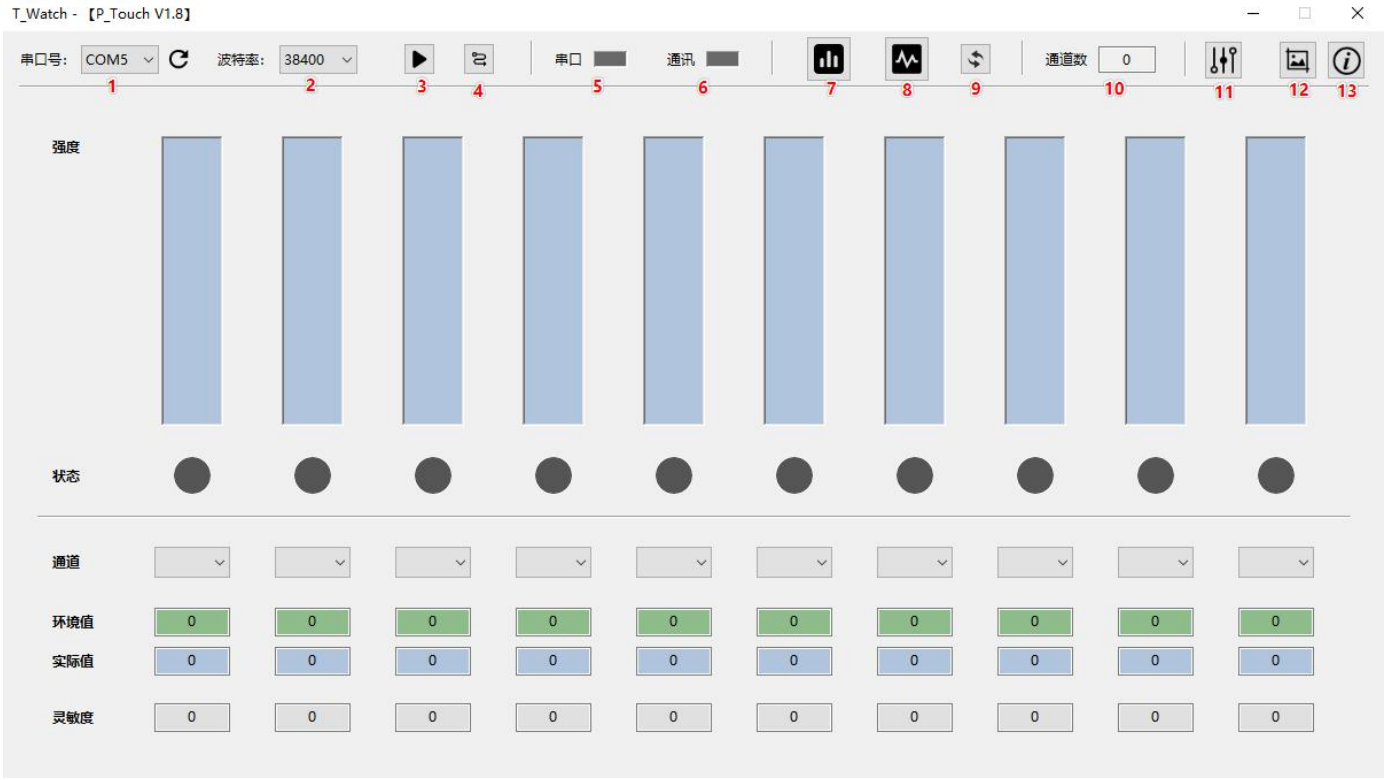


图 4-33：T-Watch 初始界面

下面对图中主菜单红色标号功能一一介绍：

标号 1 显示通讯口，一般情况在电脑连接 UART 后，再打开 T-Watch 会自动识别串口号，如果未识别到可点击旁边的刷新通讯口按钮重新获取；用户也可以在电脑设备管理器中查看端口号；

标号 2 显示串口通讯波特率，默认设置为 38400，如果需要修改，请务必注意要保持和 P-Touch 生成的程序框架中的波特率保持一致，否则将无法进行通讯；

标号 3 为开始按钮，即打开通讯串口，如果配置和线路正常，点击后就具备通讯条件；

标号 4 此按钮仅在设置唤醒方式为 T-Wacth 唤醒时才起作用，点击后会发出一个信号，用于唤醒单片机；如果选择的是低电平唤醒，则可忽略此按钮功能；

标号 5 用于显示电脑端通讯口是否连接正常，如果正常显示绿色；

标号 6 用于显示 T-Watch 和单片机通讯状况，如果显示黄色闪烁表示通讯正常；

标号 7 显示方式一，及上图的强度显示；
标号 8 显示方式二，以波形图的方式显示；
标号 9 初始化显示，点击后会重新初始化通信；
标号 10 显示程序中开启的触摸通道数量；
标号 11 PMS160 在板调试设置模块，需先开启串口通讯，才可使用；
标号 12 快速截图，图片自动保存在用户电脑桌面上；
标号 13 为 T-Watch 的使用说明；

注意：使用 UART 时需要将 TX 端串联 1K 欧姆电阻，且只支持 CP2101 和 CH340(G)模块，详细介绍请参考 T-Watch 中的使用说明。

3. 下面介绍两种显示方式

（1）显示一：强度显示

图 4-34：触发强度显示

强度显示界面可以根据程序中设置的灵敏度等参数选项，实时判断触摸按键是否触发及触发状况；当有外部按键时，图中标示 1 蓝色注形状态条会往上涨，其长度越高，表示触摸的强度越高；当其高度超过表示 2 红线高度时，标示 3 按键状态灯变亮，表示按键触发；每个状态条下方显示对应的按键通道、实时环境值、实时实际值及灵敏度；因为软件同时最多只能显示 12 个通道，所以如果涉及的通道比较多，可以手动通过通道的下拉框选择需要显示的触摸通道。

（2）显示二：图形显示

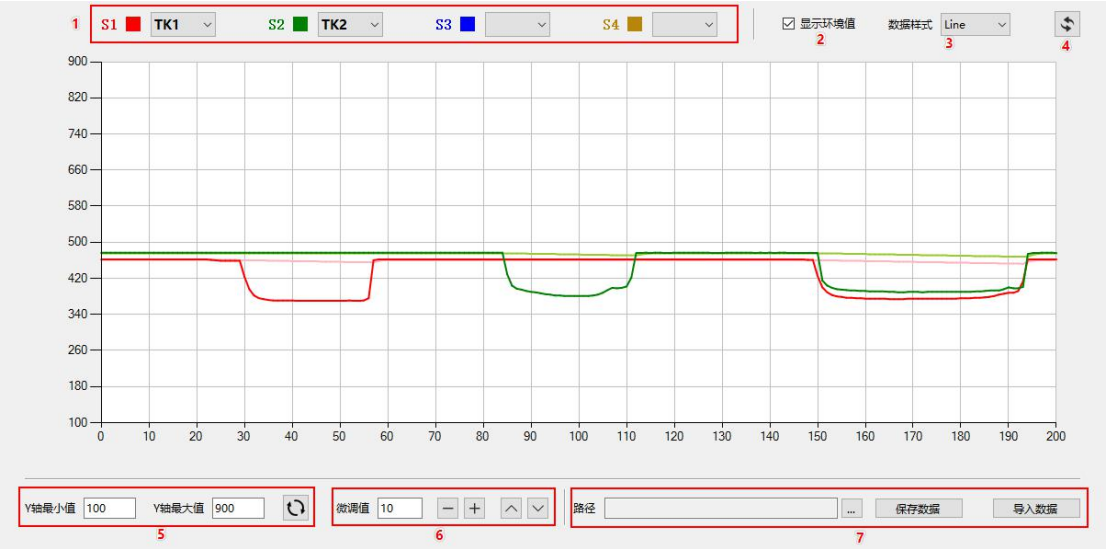


图 4-35：触发图形显示

标示 1 为当前显示的通道选择，不同颜色对应下面显示区的线形颜色，同时最多可显示四个通道触摸数据；

标示 2 为环境值显示，每个通道的环境值线形颜色为相对暗一点颜色表示；

标示 3 为显示样式切换，可选择线性、波形、点形、条形显示；

标示 4 为数据刷新显示，点击后会重新显示图形；

标示 5 用于调试 Y 轴起始值，程序运行后默认会根据你的初始触摸值设置一个相对合适初始起始值，当然你也可以自己修改适合自己的起始值；

标示 6 用于微调图形显示上下位置及图形大小缩放；

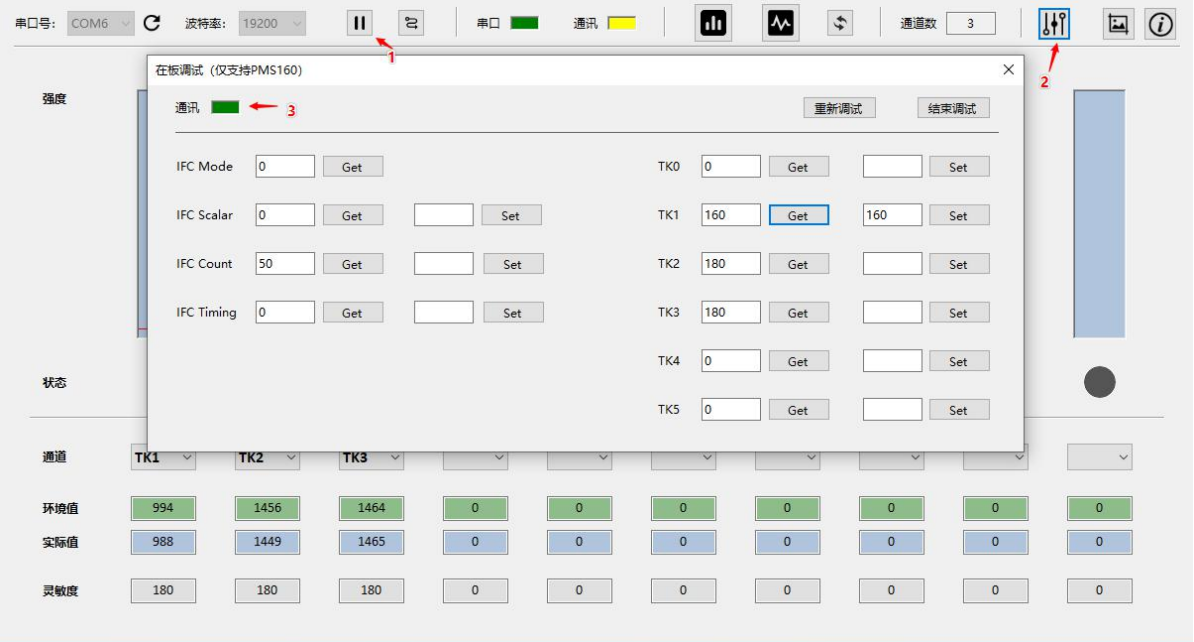
标示 7 可将触摸数据保存下来或者用 T-Watch 打开显示触摸数据；

4. PMS160 在板调试功能的使用

(1) 使用 PMS160 在板调试需在 P_Touch 中启用 T_Watch，并勾选在板调试功能，目前仅 PMS160 支持此功能；

图 4-35：开启 PMS160 在板调试功能

(2) 如果需要此功能修改参数，需先保证串口通讯是开启状态（如下图标示 1），需特别注意检查通讯波特率是否一致，并打开 T_Watch 中 PMS160 在板调试窗口（如下图标示 2），此时 T_Watch 处于等待模式状态，然后接通测试板，并上电，测试板上电后会与 T_Watch 通讯是否进入在板调试模式，如果在板调试窗口中的通讯指示灯有灰变成绿（如下图标示 3），则表示已进入在板调试模式，用户可通过 Get 获取参数，通过 Set 修改参数；



4.3. CS 测试应对办法

4.3.1. 总述

为了帮助相关触摸产品通过 CS 测试特制定此办法；

4.3.2. 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-1.0.pdf

4.3.3. 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修改为64CLK或128CLK。

5. 滑条与滑环方案

此模块用于生成滑条与滑环程序框架，使用时请根据软件上方箭头索引【工程信息】、【通道选择】、【参数设置】

等一一配置，然后生成程序，用户可在框架中对应模块完成自己的功能。

5.1.方案配置说明

5.1.1. 工程信息

需先设置基本的工程信息，包括工程名称、工程路径；选择芯片名称、封装信息；目前只支持选择 PFC460（可根据需求是否启用 FPPA 多核）以及运行的 IDE 版本；封装信息右侧的按钮可以查看当前选择的封装示意图。在开启上位机时时，需配置通讯口，上位机的介绍在后续会讲到。右侧为滑条与滑环库的使用简介。

图 5-1：滑条与滑环方案界面

5.1.2. 通道选择

先在通道列表中选择对应的通道(Step1)，然后点击对应的右移按键(Step2)，将通道设置为滑条、滑环或按键(Step3),然后设置滑条或滑环的分辨率及连接处润滑系数；选择通道时请参考下面注意事项：


文件(F) 工具(T)
P-Touch V1.8
工程信息
通道选择
参数设置
生成程序
通道列表
TK13 / PA1
TK14 / PA2
TK15 / PC0
TK16 / PC2
TK17 / PA5
TK18 / PC1
TK19 / PC3
TK20 / PC4
TK21 / PC5
TK22 / PC6
TK23 / PC7
Step1
滑条
滑环
按键
Step2
滑 条
通道 灵敏度 操作
TK0 / PD0 30 删除
TK1 / PB4 30 删除
TK2 / PB5 30 删除
TK3 / PB6 30 删除
滑 环
通道 灵敏度 操作
TK4 / PB7 30 删除
TK5 / PA3 30 删除
TK6 / PA4 30 删除
TK7 / PA0 30 删除
按 键
通道 灵敏度 操作
TK8 / PA6 50 删除
TK9 / PB2 50 删除
TK10 / PB3 50 删除
TK11 / PB0 50 删除
TK12 / PB1 50 删除
Step3
Step4
滑条分辨率
30
连接处润滑处理系数
1
滑环分辨率
32
连接处润滑处理系数
1
注意:
程序中的按键在触发后默认会自动修复环境值，如果不想按键在触发后修复环境值，可在程序里关闭!

图 5-2：通道选择

注意事项：

滑条与滑环设置：

滑条或滑环每个部分最多可选择 6 个通道，灵敏度数值越小，触摸越灵敏，取值范围：10~255；

连接处润滑系数是保证滑条或滑环在滑动时取值连续的重要参数，但滑条或滑环分辨率较高时，滑条或滑环连接处可能会出现取值不连续的情况，此时可以设置此润滑系数，使连接处取值紧密相连，取值范围：0~10；

滑条与滑环分辨率的值与选择按键的个数及连接处润滑系数有一定的计算关系，如果设定的值不合理，程序会对应给出提示建议，可根据提示建议重新设置一个分辨率，分辨率较低时润滑系数可以设为 0；

按键设置：

最多可设置 15 个触摸按键，每个按键灵敏度灵敏度单独设置，按键触发后默认会自动修复，如果不想自动修复可在程序中设置。灵敏度数值越小，触摸越灵敏，取值范围：10~255；

5.1.3. 参数设置

通用设置

(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，触摸实际值越大。注意，此处并非是系统时钟；

(2) **TK 基准电压值：**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) **CS 放电时间值：**触摸前的 CS 放电时间，一般 CS 电容越大需要的放电时间越长；

(4) **系统时钟：**设置系统时钟频率；

(5) **滤波等级：**滤波次数等于 2 的滤波等级次幂再加 2 次，例如：滤波等级为 2 表示滤波次数为 6 次；

(6) **抖动允许范围：**如果抖动超过此设定范围且低于灵敏度，环境值将开始修正；

(7) **CS 引脚：**若选择的芯片封装有多个可选择的外部电容脚位，可根据需求选择合适的外部电容脚位；

图 5-3：参数设置

5.2. 生成方案框架

如上述介绍，按需求配置完后，点击软件右上角生成程序按钮，即可生成程序框架，然后会弹出如下窗口，提示是否直接用 IDE 运行程序。

图 5-4：提示生成程序框架

用 IDE 运行生成的方案后，即可基于此框架编写实际的工程程序。下面具体介绍各个工程文件（举例我们生成一个 Touch_Slider 框架，程序中的文件架构图如下）：

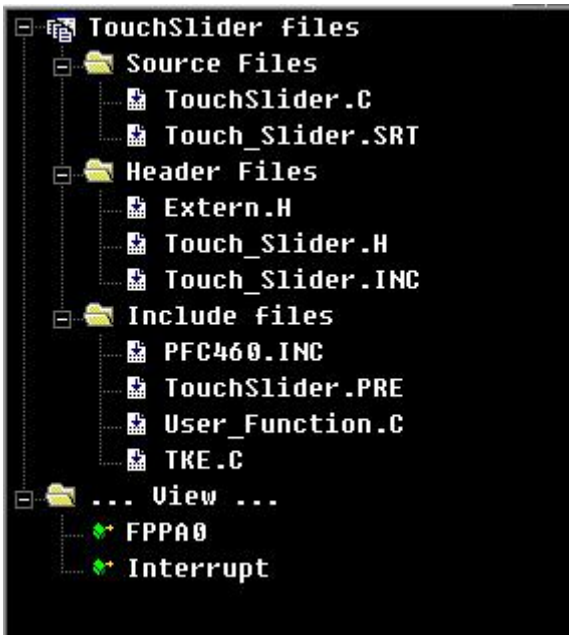


图 5-5：程序文件框架

- Touch_Demo.C 是主程序文件，主要包含一些初始化函数及功能函数的调用；
- Touch_Slider.H 是滑条、滑环、按键的配置文件，里面记录了所使用的触摸通道信息、参数配置、UART 设置等信息；
- User_Function.C 是用户的功能编写文件，用户可在其中对应模块根据自身需求写入需要的功能；

5.2.1. 滑条与滑环库配置文件

触摸库配置文件 Touch_Slider.H 文件用于配置滑条、滑环及按键相关的设定，包括通道选择，灵敏度配置，通用参数配置及 Uart 使能等。

1. 触摸按键设置（T_Keyx_Set）

先配置按键通道，然后再设置每个对应通道的灵敏度；程序默认在按键触发后会自动修复环境值；如果不想修复，可以在程序中开启不修复，取消对应按键 Disable_Press_Fix_T_Key 注释即可；

```
T_Keyx_Set:         //触摸按键(T_Keyx)设置
.C_T_Keyx    PA3,PA4,PA1,PA2,PA5     //独立触摸键，最多选择，最多选择15个触摸键
ENUM
{
    //触摸按键(T_Keyx)灵敏度设置
    C_Sen_T_Key1    =   50,             //触摸灵敏度,数值越小,灵敏度越高;取值范围 (10 ~ 255)
    C_Sen_T_Key2    =   50,
    C_Sen_T_Key3    =   50,
    C_Sen_T_Key4    =   50,
    C_Sen_T_Key5    =   50,
//Disable_Press_Fix_T_Key1  =   1,  //开启后，关闭按键触发修复，常用于长触摸
//Disable_Press_Fix_T_Key2  =   1,
//Disable_Press_Fix_T_Key3  =   1,
//Disable_Press_Fix_T_Key4  =   1,
//Disable_Press_Fix_T_Key5  =   1,
}   ;
```

图 5-6：触摸按键设置

2. 触摸滑条设置（Draw_Slip_Set）

先配置滑条通道，再设置每个对应通道的灵敏度及滑条分辨率及连接处润滑系数；

```
Draw_Slip_Set:  //触摸滑条(Draw_Slip)设置
.C_Draw_Slip    PB4,PB5,PB6,PB7 //触摸滑条(Draw_Slip)设置
ENUM
{
    //触摸滑条(Draw_Slip)灵敏度设置
    C_Sen_Draw_Slip_T1  =   30,     //触摸滑条灵敏度,数值越小,灵敏度越高;取值范围 (10 ~ 255)
    C_Sen_Draw_Slip_T2  =   30,
    C_Sen_Draw_Slip_T3  =   30,
    C_Sen_Draw_Slip_T4  =   30,
    //-----------------------
    C_Draw_Slip_Resolution= 30, //滑条分辨率
    C_Draw_Slip_Libe            =   1   //连接处润滑处理,分辨率比较低可以设为0
    //-----------------------
}   ;
```

图 5-7：触摸滑条设置

3. 触摸滑环设置（Slip_Ring_Set）

先配置滑环通道，再设置每个对应通道的灵敏度及滑环分辨率及连接处润滑系数；

```
Slip_Ring_Set:  //触摸滑环(Slip_Ring)设置
.C_Slip_Ring    PC0,PC2,PC1,PC3 //滑环触摸键选择，最多选择6个，建议使用4键滑环
ENUM
{
    //触摸滑环(Slip_Ring)灵敏度设置
    C_Sen_Slip_Ring_T1  =   30,     //触摸滑环灵敏度,数值越小,灵敏度越高;取值范围 (10 ~ 255)
    C_Sen_Slip_Ring_T2  =   30,
    C_Sen_Slip_Ring_T3  =   30,
    C_Sen_Slip_Ring_T4  =   30,
    //-----------------------
    C_Slip_Ring_Resolution= 32, //滑环分辨率
    C_Slip_Ring_Libe            =   1   //连接处润滑处理,分辨率比较低可以设为0
    //-----------------------
}   ;
```

图 5-8:触摸滑环设置

4. 通用参数设置

(1) 触摸时钟（C_Touch_Source_CLK）

取值选项: 0:ILRC, 1:IHRC/2, 2:IHRC/4, 3:IHRC/8, 4:IHRC/16, 5:IHRC/32, 6:IHRC/64, 7:IHRC/128 (1:reserved)

默认选项：3

(2) 触摸参考电压（C_Touch_VRef）

取值选项: 0:0.8*Touch Power(简称 TP), 1:0.7*TP, 2:0.6*TP, 3:0.5*TP, 4:0.4*TP, 5:0.3*TP, 6:0.2*TP

默认选项: 0

参考电压的选择对触摸灵敏度和 CS 电容的大小选择有影响，理论上参考电压越大，灵敏度越高；

(3) CS 放电时间值（C_Touch_Discharge）

取值选项：0:reserved, 1:CLK_32, 2:CLK_64, 3:CLK_128

默认选项: 3

CS 放电时间越长，放电越干净；

(4) 滤波等级（C_Smooth_Rank）

可选范围：1~6，滤波次数等于 2 的滤波等级次幂加 2；

(5) 抖动允许范围（C_Shake_Rang）

如果抖动超过此设定范围且低于灵敏度，环境值将开始修正；

```
General_Set:    //通用(General)设置
ENUM
{
    //----------------------
    C_Touch_Source_CLK= 3,          //触摸时钟选择
                                    //取值选项: 0:ILRC, 1:IHRC/2, 2:IHRC/4, 3:IHRC/8, 4:IHRC/16, 5:IHRC/32, 6:IHRC/64, 7:IHRC/128 (1:reserved)
                                    //默认选项: 3
    C_Touch_VRef            =   0,  //触摸参考电压选择，对触摸灵敏度的表现和CS电容大小的选择有影响，理论参考电压越大，灵敏度越高
                                    //取值选项: 0:0.8*Touch Power(简称TP), 1:0.7*TP, 2:0.6*TP, 3:0.5*TP, 4:0.4*TP, 5:0.3*TP, 6:0.2*TP
                                    //默认选项: 0
    C_Touch_Discharge   =   2,  //触摸前CS放电时间选择，时间越长，放电越干净
                                //0:reserved, 1:CLK_32, 2:CLK_64, 3:CLK_128
                                //默认选项: 3
    C_Smooth_Rank   =   2,      //滤波等级，滤波次数 = 2^滤波等级
    C_Shake_Rang    =   5,      //抖动允许范围，超过范围且低于灵敏度，环境值将开始修正
    //----------------------
}   ;
```

图 5-9：通用参数设置

5. 上位机设置

如果在软件上启用上位机使能，则会根据设定配置如下上位机参数，System_Clock 保持和系统时钟一致；En_Uart 为上位机使能；因为 PFC460 是多核芯片，FPPA 表示设置 Uart 位置；Baud_Rate 为 Uart 波特率，如果需要修改注意同步上位机软件上的波特率；UART_Out 设置通讯口；

```
Debug_Set:  //除错/调试设置
ENUM
{
    //------------------------
    //C_Debug_printf    =   1,  //Debug printf 触摸滑条/滑环值
    //------------------------
    System_Clock    =   2000000,    //  SYSCLK,Used at UART
    En_Uart     =   1,
    FPPA            =   0,                      //  Uart所在FPPA
    Baud_Rate   =   19200,              //  Uart波特率
    //------------------------
}   ;
UART_Out        BIT PA.0
```

图 5-10：上位机设置

5.2.2. 用户功能文件

用户功能文件 User_Function.C 主要用于 IO 及用户变量初始化，用户自定义功能模块的实现。

1. IO 初始化：

```
Void    IO_Init(void)
{
//  PA   =   0x00;
//  PAC  =   0x00;
//  PAPH=    0x00;
//  PAPL=    0x00;
//  PADIER=  0x00;
}
```

图 5-11：IO 初始化设置

2. 自定义滑条、滑环、按键功能：

在函数 T_Key_Function(void)中会持续扫描滑条、滑环、按键的触发信号，用户可以在每个触发及释放的条件判断中来自定义自己的功能；

自定义按键功能设置

```
if(T_Key_Signal)    //T_Key_Signal.1~T_Key_Signal.15    用于储存按键信号
{
    if(T_Key_Signal.1)  //按键1触发
    {
    }
    else    //释放
    {
    }
    if(T_Key_Signal.2)  //按键2触发
    {
    }
    else    //释放
    {
    }
    if(T_Key_Signal.3)  //按键3触发
    {
    }
    else    //释放
    {
    }
    if(T_Key_Signal.4)  //按键4触发
    {
    }
    else    //释放
    {
    }
    if(T_Key_Signal.5)  //按键5触发
    {
    }
    else    //释放
    {
    }
}
else            //所有按键释放
{
}
```

图 5-12：自定义按键功能设置

自定义滑条功能：功能写在对应 Case 中；

```
if(Draw_Slip_Press)          //滑条触发标志
{
    switch(Draw_Slip_Shift)
    {
    Case 1: NULL;
        break;
    Case 2: NULL;
        break;
    Case 3: NULL;
        break;
    Case 4: NULL;
        break;
    Default:NULL;
        break;
    }
}
else    //滑条释放
{
    NULL;
}
```

图 5-13：自定义滑条功能设置

自定义滑环功能：功能写在对应 Case 中；

```
if(Draw_Slip_Press)          //滑环触发标志
{
    switch(Draw_Slip_Shift)
    {
    Case 1: NULL;
        break;
    Case 2: NULL;
        break;
    Case 3: NULL;
        break;
    Case 4: NULL;
        break;
    Default:NULL;
        break;
    }
}
else    //滑环释放
{
    NULL;
}
```

图 5-14：自定义滑环功能设置

5.2.3. 用户主工程文件

主工程文件主要包含一些初始化函数及滑环、滑块、按键的扫描函数；

```
IO_Init();
Uart_Auto();
TK_Init_Auto();
Timer16_Init();


//  CLKMD.En_WatchDog   =   1;      // WatchDog Enable
while (1)
{
    T_Key_Scan();
    Uart_Auto();
    T_Key_Function();
    //      ...
    wdreset;
}
```

图 5-15：主工程文件

5.2.4. 上位机的使用

滑条与滑环方案中的上位机是用于直观显示各个模块触发状况；

1. 首先在使用 P-Touch_V1.8 生成程序框架时选择启用上位机，并设置合适的通讯口，默认采用低电平方式唤醒；

图 5-15：启用上位机设置

2. 在 P-Touch_V1.8 菜单中选择“工具”，点击打开上位机，如下图，使用时使用 Uart 连接滑条、滑环仿真板，M1 与 M2 为两种触发显示效果；

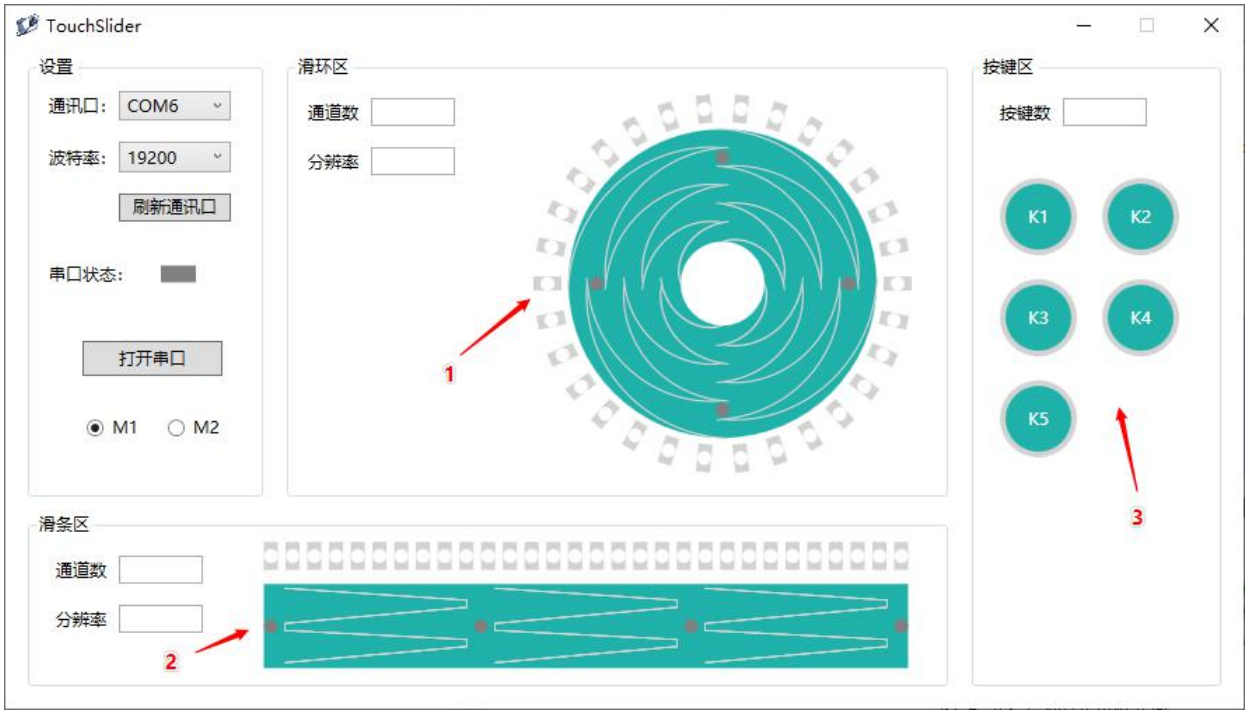


图 5-16：滑条与滑环上位机

标号 1 为滑环触发显示区；
标号 2 为滑条触发显示区；
标号 3 为按键触发显示区；

3. 触发显示效果：

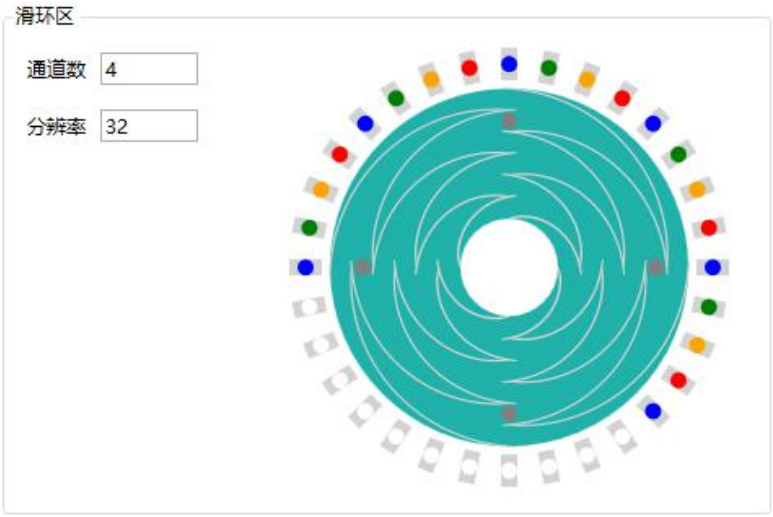


图 5-17：滑环触发显示

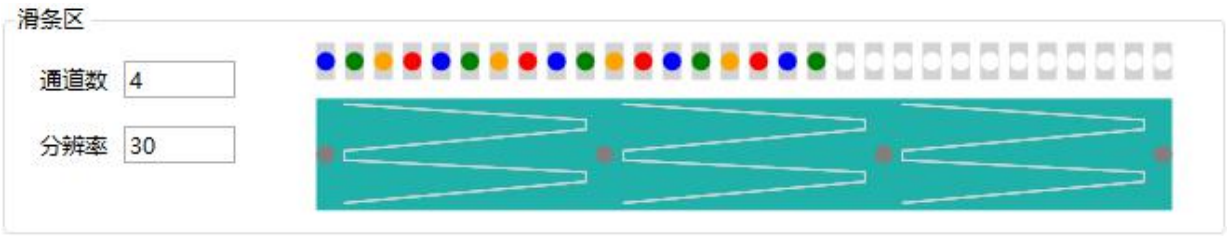


图 5-18：滑条触发显示

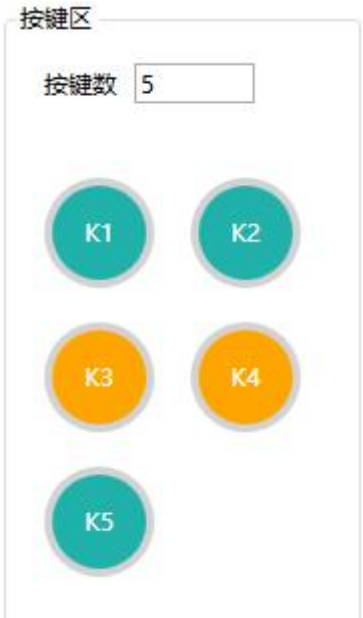


图 5-19：按键触发显示

6. 触摸标准品选型表

下面是触摸标准品选型表，用户可根据其功能描述如果满足自己需求可直接使用，另外也可自行下载对应的PDK烧录档；

注意：标准品程序使用前请用户务必详细验证功能，本公司不承担任何软件责任。

触摸系列标准品

触摸标准品选型手册

温馨提示：标准品程序使用前请详细验证功能，本公司不承担任何软件责任！ 2020/12/04

产品名称	触控通道	输出型态	封装	功能描述	校验码	PDK下载
XDT8001C-S08B	1	PWM	SOP8	单键单色调光	0x1D2474	
XDT8002B-S08B	2	CMOS	SOP8	双键开关类型-翻转模式	0x3124F5	
XDT8101B-U06	1	CMOS	SOT23-6	单键开关类型-直通模式	0x4539A4	
XDT8101B-2N06	1	CMOS	DFN-6	单键开关类型-直通模式	0x4539A4	
XDT8102B-U06	1	CMOS	SOT23-6	单键开关类型-翻转模式	0xCC0B5C	
XDT8103-S08B	2	CMOS	SOP8	双键开关类型-直通模式	0xE43510	
XDT8104-S08B	2	CMOS	SOP8	双键开关类型-翻转模式	0x1404A0	
XDT8105-EY10	3	CMOS	ESSOP10	双键开关类型-直通模式	0xA50624	
XDT8106-EY10	3	CMOS	ESSOP10	双键开关类型-翻转模式	0xAC3B6A	
XDT8107-EY10	4	CMOS	ESSOP10	BCD 码输出	0xFD1245	
XDT8108-EY10	1	PWM	ESSOP10	单键双色调光	0x653E03	