

ChipON TSTool V3 用户使用手册

在使用本手册前，请您认真阅读以下使用许可协议。只有在同意以下使用许可协议的情况下，方能使用本手册中介绍的产品。

版权公告

未经上海芯旺微电子有限公司书面允许，任何公司、个人不得以任何形式复制本使用手册的全部或部分内容。

重要声明

上海芯旺微电子有限公司努力使本手册中提供的信息准确和适用，然而，产品及手册可能包括技术或印刷上的错误。上海芯旺微电子有限公司保留在不事先通知的情况下改变本使用手册全部或部分内容的权力。

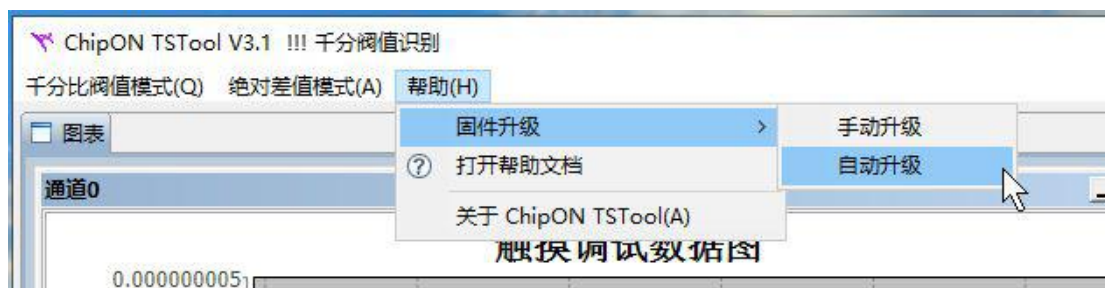
目录

触摸调试软件使用说明.....	4
一、帮助与固件管理.....	4
二、连接 退出 调压.....	4
三、开始接受数据.....	5
四、暂停接受数据.....	6
五、清零并坐标自适应.....	6
六、更新速度.....	6
七、对曲线进行放大和还原.....	7
八、历史数据功能.....	7
九、其他说明.....	9
CHIPON 触摸调试配置使能说明.....	10
一、概述.....	10
二、上位机使用说明.....	10
三、调试代码的使用方法.....	10
四、其他说明.....	11

触摸调试软件使用说明

一、帮助与固件管理

可点击菜单栏“帮助”—“固件升级”进行更新。连接时会自动检测版本进行提示更新。



该软件未提供编程器的驱动程序，可以从 ChipON IDE 、ChipON Pro 安装或者官网下载单独的驱动安装包。

帮助为 pdf 文档，需要对应的阅读器支持。可以在安装目录处打开或这里点击打开。

二、连接 退出 调压

软件打开后，工作之前需要首先进行连接。

显示连接成功后，方可进行后续操作，同样进行编程复位后再次使用需要再次点击连接。

当前软件集成了连接功能，即点击基于通道工作时，如果未进行连接会优先进行连接，后开启通道采样。

支持了不同的工作电压，但为粗略电压，根据 1.0~5.0 (V) 的数据输入，具体供电值应进行实测。该功能需要使用的固件版本不应该低于当前软件自带，否则按照 5.0V 上电，实际约 4.7V。

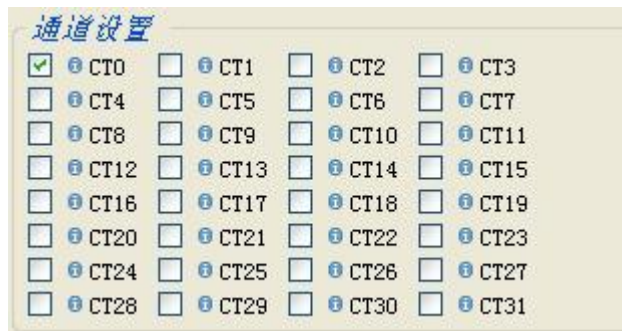


软件的退出，可以通过应用程序的右上角的关闭按钮，但应该处于不工作状态下，即未连接、暂停接收数据，释放连接后均可。否则需要使用这里“退出软件”按钮功能。

三、开始接受数据



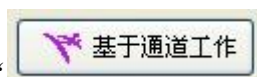
显示数据获取周期和曲线内容，一般默认即可。



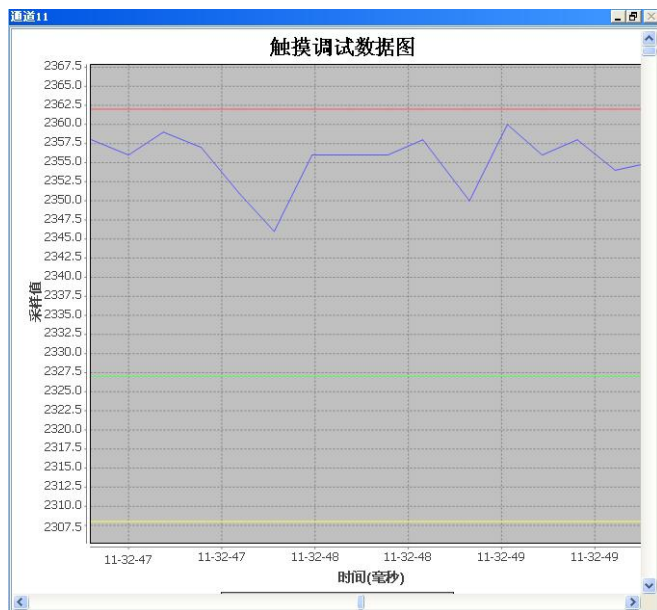
与芯片对应的通道观察选择。观察时进行勾选。



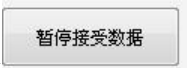
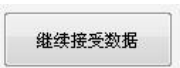
针对不同通道可以设置不同，可以在工作期间随时调整。支持负数的设入。



然后点击“基于通道工作”这个按钮，便可在图上看到绘制出来的曲线。如下图。



四、暂停接受数据

点击  按钮，则停止接受数据，并且暂停按钮变为 ，当需要再次接受数据时，则点击继续接受数据按钮。

该功能暂停触摸数据的接收，但继续供电，并因数据的暂停传输，较少在传输上的时间消耗。另外暂停接受数据也将影响监控查看，即恢复数据后恢复时刻为可记录的历史起始时间。

五、清零并坐标自适应

- 清除按钮的功能主要为从零开始接受数据，更多的作用在于根据当前的基准和采样数据，以及计算的阈值进行动态调整数据轴。
- 默认坐标轴从 0 到最大值的长度进行显示，自适应后，基于监控曲线的最小值自动调整，变化情况观察更明显。

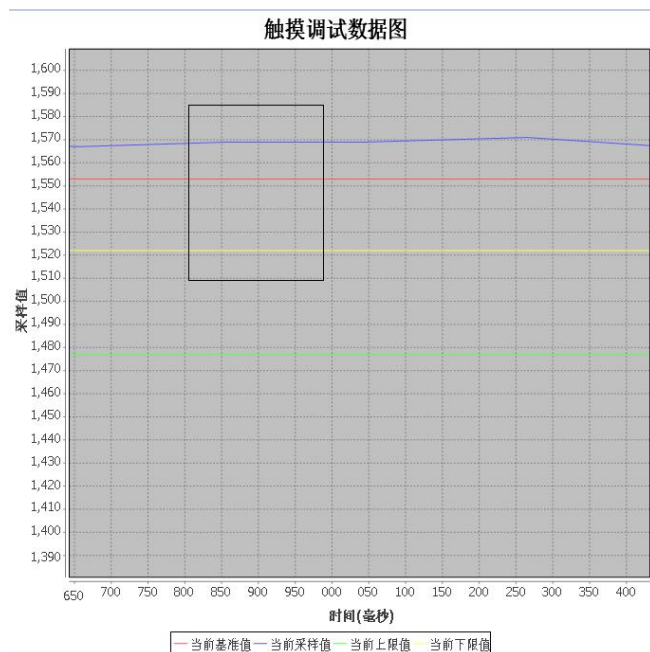
六、更新速度

更新速度可以进行选择或输入，决定了抽样数据的周期。

当缺失数据时，软件不会停止工作，仅更改通道前图标做标记。因此一般默认速度即可。

七、对曲线进行放大和还原

- 对曲线进行放大，只需要按住鼠标左键，向右下角拖动鼠标，则可以将曲线放大。
- 还原曲线，则是只按住鼠标左键，向左上角拖动鼠标，则可以还原曲线。
- 同时也只有还原曲线后才可继续进行实时曲线的显示。
- 该观察区间的放大与还原同样适用于历史数据观察窗口。

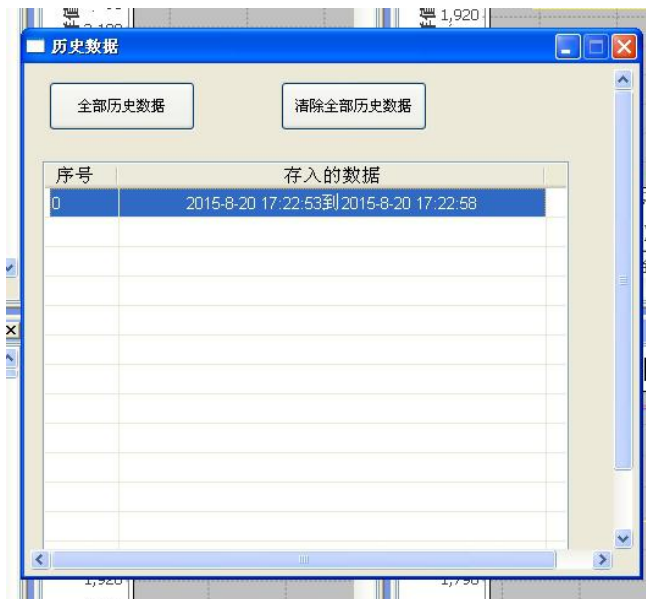


八、历史数据功能

- 在数据接收前应该勾选“☐ 使能监控查看”选项。基于通道工作后按钮不可选择，通过释放连接后可以恢复选择并基于通道工作开展新的监控。
- 当 或 后，可以点击 进行选择查看。
- 在通道的下的触摸数据查看，可以拖动进行数据曲线的浏览。
- 默认完整显示监控时间内的数据。可以通过针对曲线片段的选择进行放大观察即可。并通过还原回到默认状态，即左上到右下鼠标点击拖动矩形下放大，右下往左上还原。
- 可以点击“全部历史数据”按键或在时间分类的序号行右键选择清除或查看。需要说明的时全部历史查看打开所有系统支持的通道窗，有选择的查看仅显示当时记

录时的监控的通道。

- 历史记录针对当前打开有效，软件关闭后，信息丢弃。



选择窗口



观察窗口

- 查看窗口下可以进行窗口大小的调整，也可以拖动滚动条进行查看时间下的数据曲线。同时可以使用曲线的放大和还原操作。

九、其他说明

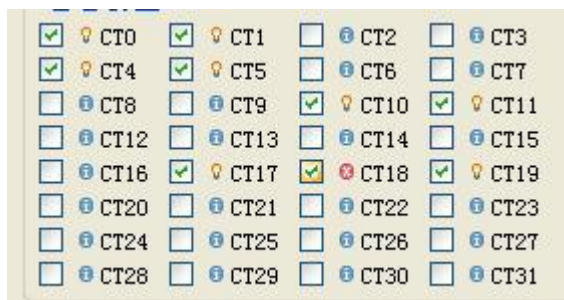
■ 软件提供千分阈值识别和绝对差值模式识别 2 种模型，应选择与算法一致。可以通过直接点击菜单栏的菜单进行切换，也可以快捷键切换。

■ 针对有限的桌面大小，当选择较多的通道时，窗口分配的空间较小，不利于数据的观察，可以最大化窗口进行查看，一般在窗口的标题栏上进行双键点击即可完成切换。

■ 针对通道的数据观察，可以在界面上进行选择后进行实时观察，按键的状态基于阈值的模拟计算，不代表实际的芯片状态。如下图。



针对通道获取数据失败，可以通过通道的图标进行区分，如下图所示，CT0 正常获取到，CT18 获取数据失败，CT22 未进行数据获取。



一、概述

调试代码参考资源使用：20 BYTE 数据区； 300 WORD 程序区。占用 2 个双向 I/O 口，可自行定义，默认使用编程 DAT 和 CLK 即可；具体与实现芯片型号和资源有关。

三、调试代码的使用方法

因 KF8 LIBf touch process () 为触摸的处理函数, 处理当前通道, 寻找下一通道,

[illegible]

四、其他说明

C4 进入调试模式，返回 68 55 00 00 BD 42 16
A5 A5 允许通道数据上传，返回 68 55 00 00 BD 42 16
YY A7 电压控制，返回 68 55 00 00 BD 42 16，应在 A5 A5 之前传输，YY（10~50）
68 3A 00 00 A2 5D 16 暂停
68 A3 00 00 0B F4 16 继续
68 33 0M 00 XX xx YY yy tt ~tt16 通道指令，无返回
其中 M 包含通道操作的个数，数据有效个数，因此为通道数的 2 倍
XX、YY 为操作的通道号（支持 0~31 共 32 个通道）
xx、yy 为通道要设定的值 1 为开、0 为关
tt，~tt 代表数据帧的校验和、和校验和反码，以 68 为开始，最后 1 个数据为结尾。

2、上位机的通道设定采用只下发、编程器采用仅上传通道触摸数据模式，因此存在上位机命令被传输错误的概率，可以通过再次点击接收来进行。触摸数据上传格式为：0xAA 通



3、编程器指示灯复用来指示当前运行状态。工作忙指示灯表示存在触摸通道数据上传。编程器结果正确时灯表示接收上位机命令包正确，编程器结果错误红灯表示接收上位机命令包错误。收到上位机命令包头 0x68 时熄灭编程器结果灯，等待命令包结尾。异常造成不完整的命令包下发会造成 2 次命令包的丢失，原因为第二个的包头被补充在了第一个包尾。可以根据命令接收指示灯进行重试操作。

```
* 文件名: debug_touch.h
* 版 本:   v1.1
* 日 期:   2015-6-01
* 作 者:   上海芯旺微电子有限公司
* 说明:     电容触摸库调试输出函数文件声明
```

```
//{ // 芯片端口操作类型 1
```

ChipON

```
    __asm
    MOV PCH,R1
    MOV PCL,R0
    __endasm;
}
void TOUCH_DEBUG_TRS_DEAL()
{
    unsigned char j,step;
    // 判断当前通道允许上传时进行上传过程
    #if 0
        TOUCH_CH_TRS_EN[_KF8_TOUCH_CH_EN[Touch_Channel_Protect]]=1;
    #else
        __asm
        MOV R1,#High(__KF8_TOUCH_CH_EN+0)
        MOV R0,#(Touch_Channel_Protect)
        ADD R0,#Low(__KF8_TOUCH_CH_EN+0)
        JNB PSW,0
        INC R1
        PAGESEL __R2PCHPCL
        CALL __R2PCHPCL
        PAGESEL $
        ADD R0,#(_TOUCH_CH_TRS_EN+0)
        MOV R1,#0x01
        BANKSEL _TOUCH_CH_TRS_EN
        ST [R0],R1
        __endasm;
    #endif

    //////////////////////////////////////
    #if 0
        if(TOUCH_CH_TRS_EN[_KF8_TOUCH_CH_EN[_KF8_LIBc_channel_]])
    #else
        __asm
        BANKSEL __KF8_LIBc_channel_
        MOV R0,__KF8_LIBc_channel_
        ADD R0,#Low(__KF8_TOUCH_CH_EN+0)
        MOV R1,#High(__KF8_TOUCH_CH_EN+0)
        JNB PSW,0
        INC R1
        PAGESEL __R2PCHPCL
        CALL __R2PCHPCL
        PAGESEL $
        ADD R0,#(_TOUCH_CH_TRS_EN+0)
        BANKSEL _TOUCH_CH_TRS_EN
        LD R1,[R0]
        XOR R1,#0x00
        JNB PSW,2
        JMP Label_x_y_z_a
        __endasm;
    #endif
    {
        CLK_SET_IN;
        DAT_SET_IN;
        Touch_Delay1;

        //获取上传权限,高 clk有效允许传输

        if(CLK_GET_STATE)
        {
            Touch_Delay1;
        }
    }
}
```

```
        if(!CLK_GET_STATE)
            return;
    }
    else if(!CLK_GET_STATE)
    {
        return;
    }
    else
        return;
//发送启动信号，等待接收准备就绪
CLK_SET_OUT;

CLK_SET_LOW;
Touch_Delay1;

CLK_SET_HIGH;          //上升沿

Touch_Delay1;

if(!DAT_GET_STATE)
{
    return;
}

CLK_SET_LOW;          //下降沿

//开始数据的上传，优先传输通道值，基线低位，基线高位，采样低位，采样高位
Touch_Delay1;

DAT_SET_OUT;
step=0;
j=5;
//=====
for(step=0;step<5;step++)
{
    switch(step)
    {
        case 0:
        {
            j=5;

            #if 0
                T_buf=_KF8_TOUCH_CH_EN[_KF8_LIBc_channel_];
            #else
                __asm
                BANKSEL __KF8_LIBc_channel_
                MOV R0, __KF8_LIBc_channel_
                ADD R0,#Low(__KF8_TOUCH_CH_EN+0)
                MOV R1,#High(__KF8_TOUCH_CH_EN+0)
                JNB PSW,0
                INC R1
                PAGESEL __R2PCHPCL
                CALL __R2PCHPCL
                PAGESEL $
                BANKSEL _T_buf
                MOV _T_buf,R0
                __endasm;
            #endif

        }break;
        case 1:
        {
```



```
                                j=8;

    #if 0
                                T_buf=(unsigned
char) (_KF8_LIBi_buff_baseline[_KF8_LIBc_channel_]);
    #else
                                __asm
                                BANKSEL __KF8_LIBc_channel_
                                CLR PSW,0
                                RLCR __KF8_LIBc_channel_
                                BANKSEL __KF8_LIBi_buff_baseline_
                                ADD R0,#(__KF8_LIBi_buff_baseline_+0)
                                LD R1,[R0]
                                BANKSEL _T_buf
                                MOV _T_buf,R1
                                __endasm;
    #endif
                                }break;
                                case 2:
                                {
                                    j=8;

                                    #if 0
                                        T_buf=(unsigned
char) (_KF8_LIBi_buff_baseline[_KF8_LIBc_channel_]>>8);
                                    #else
                                        __asm
                                        BANKSEL __KF8_LIBc_channel_
                                        CLR PSW,0
                                        RLCR __KF8_LIBc_channel_
                                        BANKSEL __KF8_LIBi_buff_baseline_
                                        ADD R0,#(__KF8_LIBi_buff_baseline_+1)
                                        LD R1,[R0]
                                        BANKSEL _T_buf
                                        MOV _T_buf,R1
                                        __endasm;
                                    #endif

                                    }break;
                                    case 3:
                                    {
                                        j=8;

                                        #if 0
                                            T_buf=(unsigned
char) (_KF8_LIBi_buff_hit[_KF8_LIBc_channel_]);
                                        #else
                                            __asm
                                            BANKSEL __KF8_LIBc_channel_
                                            CLR PSW,0
                                            RLCR __KF8_LIBc_channel_
                                            BANKSEL __KF8_LIBi_buff_hit_
                                            ADD R0,#(__KF8_LIBi_buff_hit_+0)
                                            LD R1,[R0]
                                            BANKSEL _T_buf
                                            MOV _T_buf,R1
                                            __endasm;
                                        #endif

                                        }break;
                                        case 4:
                                        {
                                            j=8;
```



```
#if 0
                                T_buf=(unsigned
char) (_KF8_LIBi_buff_hit[_KF8_LIBc_channel_]>>8);
#else
                                __asm
                                BANKSEL __KF8_LIBc_channel_
                                CLR PSW,0
                                RLCR __KF8_LIBc_channel_
                                BANKSEL __KF8_LIBi_buff_hit_
                                ADD R0,#(__KF8_LIBi_buff_hit_+1)
                                LD R1,[R0]
                                BANKSEL _T_buf
                                MOV _T_buf,R1
                                __endasm;
#endif

                                }break;
                                }
//-----
    for(i_T_D=0;i_T_D<j;i_T_D++)
    {
        if(T_buf&0x01)
            DAT_SET_HIGH;
        else
            DAT_SET_LOW;

        CLK_SET_HIGH;
        Touch_Delay1;

        CLK_SET_LOW;
        Touch_Delay1;

        T_buf=T_buf>>1;
    }
//-----
    Touch_Delay2;
}
//=====
//获取有无通道设置下发需求
DAT_SET_IN;
Touch_Delay2;    //编程器判断需要时间

CLK_SET_HIGH;
Touch_Delay1;
Touch_Delay1;
Touch_Delay1;
//;;;;
if(DAT_GET_STATE)
{
    CLK_SET_LOW;
    T_buf=0;
    Touch_Delay1;
    for(i_T_D=0;i_T_D<6;i_T_D++)
    {
        CLK_SET_HIGH;
        Touch_Delay1;

        if(i_T_D!=5)
        {
            T_buf=T_buf>>1;
            if(DAT_GET_STATE)
```

ChipON