设 计 文 件

名称	QTouch3 控件设计说明书
编号	
版本	V3.0

武汉舜通智能科技有限公司

编　制		工　艺	
校　核		标准化	
审　核		批　准	

版本号	更改人	更改日期	更改说明	变更编号
3.0	高伟锋	2020.6.16	完成 3.0 版本	

目　次

1 目的和范围

1.1 目的

本文档是对 QTouch3 绘制系统的扩展控件设计的详细说明。对开发设计者起一个引导作用，用户可根据自己的特殊需求设计独立专用的控件，只需一些简单的操作就可以将控件集成到 QTouch 组态软件中使用。

开发的环境需求：VS2012+Qt5.4。

配置完开发环境后，提供的示例代码里面有自带的 pro 工程文件，用户可以双机运行 make.bat 生成针对 vs 的工程。

2 开发环境部署

2.1 安装 VS2012

在微软官网上下载即可，安装过程中可以选择默认安装，也可以自定义；选择自定义安装需要勾选C++的编程语言。

2.2 安装 Qt5.4

Qt 下载地址 http://download.qt.io/new_archive/qt/5.4/5.4.2/，选择的Qt版本为qt-opensource-windows-x86-msvc2012_opengl-5.4.2.exe。

下载完后，直接双机exe进行安装，安装过程直接默认即可。

2.3 Qt 和 Vs 集成环境安装(不是必须)

为了方便在VS上直接开发Qt的项目工程，需安装qt-vsaddin集成工具，可以下载qt-vs-addin-1.2.2-opensource进行安装，下载地址：http://download.qt.io/archive/vsaddin/1.2.2/。

2.4 环境变量设置

在系统中需添加如下环境变量：（添加环境变量的方法：右键“我的电脑”->属性->高级系统设置->环境变量；可以在用户变量中添加）

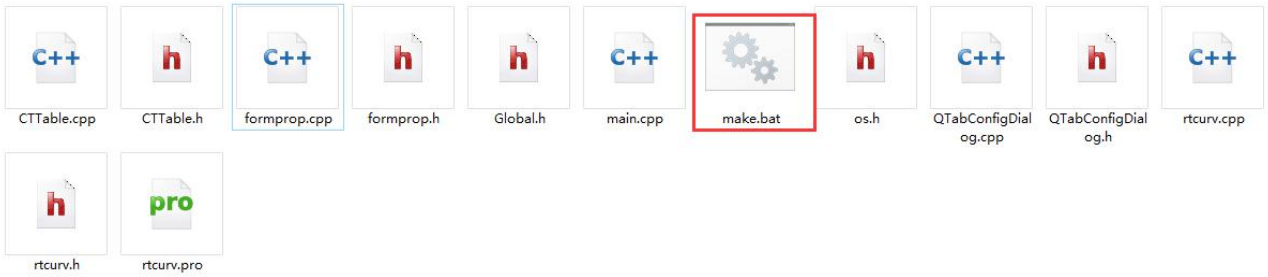
变量	值
QTDIR	QT的安装路径，比如：D:\Qt\Qt5.4.2\5.4\msvc2012_opengl
QMAKESPEC	win32-msvc2012
PATH（注意：PATH变量会存在多个值，win10可以直接新建增加一行，填入值即可，非win10的，添加的时候，在最后先写入分号再填值）	%QTDIR%\bin
include（注意事项同PATH）	%QTDIR%\include
lib（注意事项同PATH）	%QTDIR%\lib

环境变量设置完后，win10系统可以立即生效，非win10的可能需要重启才能生效；确认生效的方法，调出命令行cmd，然后输入qmake -v,有如下打印结果即可证明Qt的环境部署成功；

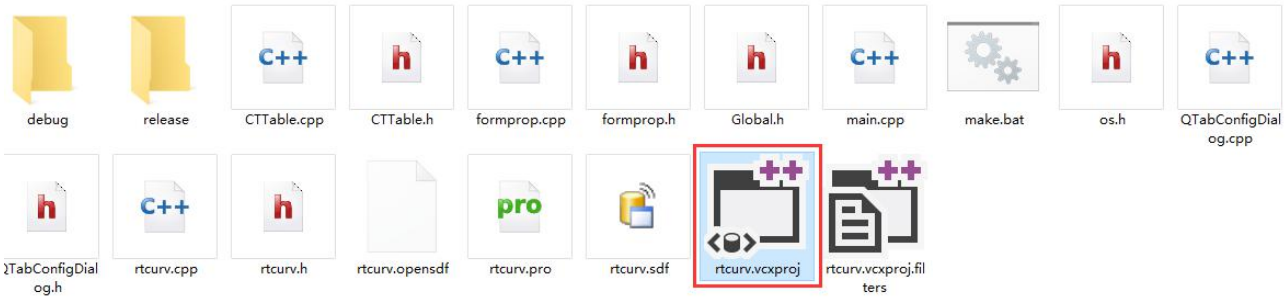
3　源码分析(示例代码提供了实时曲线、报表、表盘三个插件源码，本文以 rtcurv 实时曲线为例进行说明)

3.1　工程建立

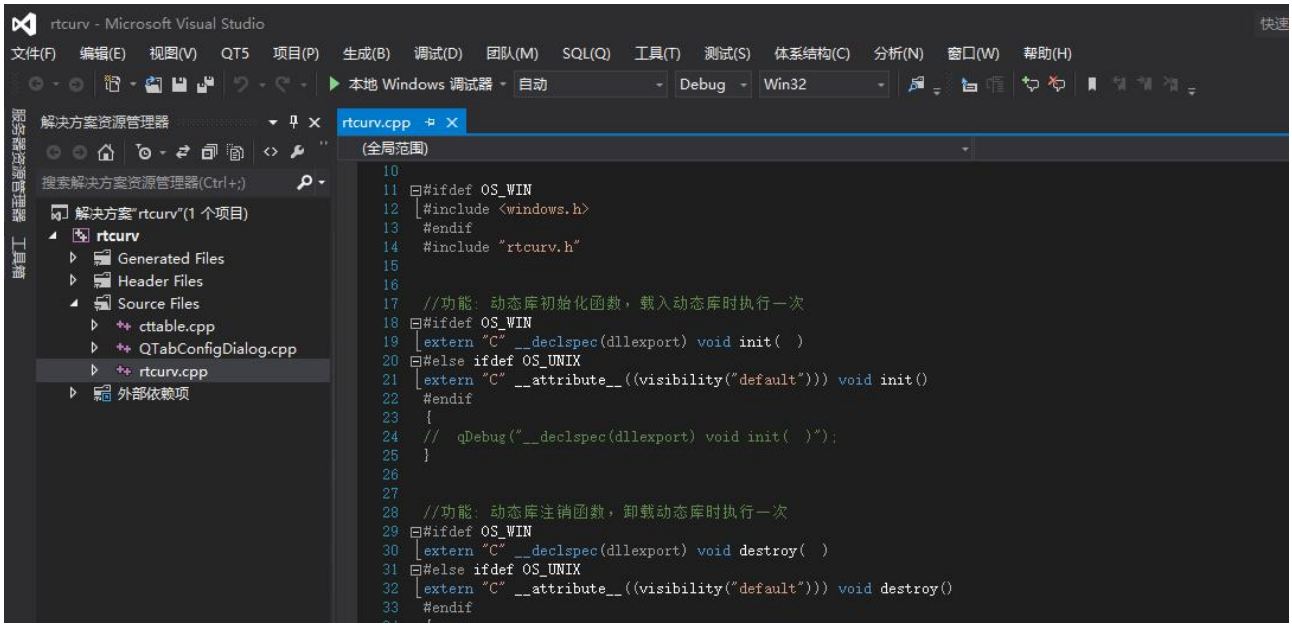
打开源码文件夹，可以双机make.bat生成VS的工程



执行make.bat后，文件夹下会生成相应的VS工程文件，下面标注的文件即为vs的工程文件；



用vs2012打开该文件即可



3.2 控件窗口基类

customwidget.h定义了控件动态窗口基类QCustomWidget，所有编写的控件必需继承该基类。QCustomWidget定义了几个重要的虚函数，这些虚函数根据具体的需要在子类中选择性的实现：

1、 virtual void Fresh(double dValue, char chChange = 0);

```
/*!
* 描述
*        数据刷新函数，在绘制系统中，当点击一个控制的实时关联属性，系统运行后，会根据画面的刷新周期实时刷新改关联的变量数据，用户可以根据这个函数接口获取关联变量的实时值
* 参数    dValue
*        关联变量的数值
* 参数    chChange
*        关联变量的状态
* 返回
*        无
* @note
*        接口类型: 阻塞式
*/
```

2、 virtual void save(QDataStream &stream);

```
/*!
* 描述
*        属性保存，基类QCustomWidget定义的公有属性外，设计者可以定义一些其它特有的属性，这些属性将以数据流(QDataStream)的形式保存起来
* 参数    stream
*        数据流
* 返回
*        无
```

```
    * @note
    *        接口类型：阻塞式
    */
```

3、 virtual void load(QDataStream &stream);

```
    /*!
    * 描述
    *        属性加载，当重新打开一个绘制画面时，画面的控件会首先调用该函数接口，加载属性并初始化该控件
    * 参数    stream
    *        数据流
    * 返回
    *        无
    * @note
    *        接口类型：阻塞式
    */
```

图8-1：绘制系统右下方的属性选择

4、 virtual void setLineColor(QColor color);

```
    /*!
    * 描述
    *        从图8-1的“线颜色”栏中选择，调用该函数接口就可以获取选择的颜色
    * 参数    color
    *        线的颜色
    * 返回
    *        无
    * @note
    *        接口类型：阻塞式
    */
```

5、 virtual void setLineWidth(int width);

```
/*!
* 描述
*       从图8-1的“线宽”栏中输入，调用该函数接口就可以获取输入的内容
* 参数   width
*       线的宽
* 返回
*       无
* @note
*       接口类型：阻塞式
*/
```

6、 virtual void setFillColor(QColor color);

```
/*!
* 描述
*       从图8-1的“填充颜色”栏中选择，调用该函数接口就可以获取选择的颜色
* 参数   color
*       填充色颜色
* 返回
*       无
* @note
*       接口类型：阻塞式
*/
```

7、 virtual void setTextValue(QString txt);

```
/*!
* 描述
*       从图8-1的“文本”栏中输入，调用该函数接口就可以获取输入的内容
* 参数   txt
*       文本内容
* 返回
*       无
* @note
*       接口类型：阻塞式
*/
```

8、 virtual void setTextColor(QColor color);

```
/*!
* 描述
*       从图8-1的“文本颜色”栏中选择，调用该函数接口就可以获取选择的颜色
* 参数   color
*       文本的颜色
* 返回
*       无
```

```
    * @note
    *        接口类型: 阻塞式
    */
```

9、 virtual void setFont(QFont font);

```
    /*!
    * 描述
    *        从图8-1的“字体”栏中选择，调用该函数接口就可以获取选择的字体
    * 参数    font
    *       设置的字体
    * 返回
    *        无
    * @note
    *        接口类型: 阻塞式
    */
```

10、 virtual void sendMsg(const QString msg);

```
    /*!
    * 描述
    *       自定义插件给主窗体发消息
    * 参数    msg
    *       发送的消息
    * 返回
    *        无
    * @note
    *        接口类型: 阻塞式
    */
```

11、 virtual void recvMsg(const QString msg);

```
    /*!
    * 描述
    *       自定义插件收到主窗体发送的消息
    * 参数    msg
    *       接收的消息
    * 返回
    *        无
    * @note
    *        接口类型: 阻塞式
    */
```

3.3 控件封装接口

编写好的控件都将以dll的形式集成到QTouch中。

```
//功能: 动态库初始化函数，载入动态库时执行一次
#ifdef OS_WIN
```

```
extern "C" __declspec(dllexport) void init(  )
#else ifdef OS_UNIX
extern "C" __attribute__((visibility("default"))) void init()
#endif
{
}


//功能: 动态库注销函数，卸载动态库时执行一次
#ifdef OS_WIN
extern "C" __declspec(dllexport) void destroy(  )
#else ifdef OS_UNIX
extern "C" __attribute__((visibility("default"))) void destroy()
#endif
{
}
//功能:
#ifdef OS_WIN
extern "C" __declspec(dllexport) QWidget * createRtCurv( QWidget * pWid )
#else ifdef OS_UNIX
extern "C" __attribute__((visibility("default"))) QWidget * createRtCurv( QWidget * pWid )
#endif
{
    // 设置字符集, GB2312 支持中文
    QTextCodec *codec = QTextCodec::codecForName("GB2312");
    QRtCurv *curv = new QRtCurv( pWid, "curv" );
    curv->setWindowTitle("realtime curv");
    curv->show();
    return curv ;
}
//功能: 列表自定义窗口类型
//在列表中顺序存入, 第一个为入口函数名称, 第二个为描述文字
#ifdef OS_WIN
extern "C" __declspec(dllexport) void listwidget( QStringList & list )
#else ifdef OS_UNIX
extern "C" __attribute__((visibility("default"))) void listwidget( QStringList & list )
#endif
{
    QTextCodec *codec = QTextCodec::codecForName("GB2312");
    list << "createRtCurv" << codec->toUnicode("实时曲线") ;
}
```

每一个编写的控件都需按照上面的形式进行封装。

3.4 控件主要函数：

```
//继承实时刷新，该函数是安装主窗体的画面刷新率周期执行的
//dValue 是给该插件配置的关联变量的数值，chChange 是关联变量的状态
//针对单个变量的关联可以直接采用该端口
//函数的实体需用户自行设计
void QRtCurv::Fresh( double  dValue, char chChange  )
{
}

//该函数是加载插件组态配置的属性；
//QTouch 的画面运行加载自定的插件首先构造完插件后，会调用插件的 load 方法加载属性执行；
//该方法是配合 save 函数一起使用的；
//该函数的设计需根据实际的插件功能进行设计
void QRtCurv::load( QDataStream &stream )
{
    //Q_INT32 iVersion ;
    qint32 iVersion;
    StructProperty info;

    stream >> iVersion ;
    stream >> info;
    if(iVersion>=3)
    {
        QFont proFont;
        stream >> proFont;
        info.proFont = proFont;
        int fontSizes;
        stream >> fontSizes;
        info.fontSizes = fontSizes;
        int b3DCheck;
        stream >> b3DCheck;
        if(b3DCheck==1)
        {
            info.b3DCheck = true;
        }
        else
        {
            info.b3DCheck = false;
        }
        QColor borderColor;
        stream >> borderColor;
        info.borderColor = borderColor;
    }

    stream >> nCrossLineWidth;
```

```
    if(iVersion >= 4)
{
    int MaxBaseLineY,MinBaseLineY;

    stream >> MaxBaseLineY;
    info.MaxBaseLineY = MaxBaseLineY;

    stream >> MinBaseLineY;
    info.MinBaseLineY = MinBaseLineY;


    QColor MaxBaseLineColor,MinBaseLineColor;

    stream >> MaxBaseLineColor;
    info.MaxBaseLineColor = MaxBaseLineColor;//上基准线颜色

    stream >> MinBaseLineColor;
    info.MinBaseLineColor = MinBaseLineColor;//下基准线颜色

    int MaxBaseLineWidth = 1;
    int MinBaseLineWidth = 1;

    stream >> MaxBaseLineWidth;
    info.MaxBaseLineYWidth = MaxBaseLineWidth;//上基准线线宽

    stream >> MinBaseLineWidth;
    info.MinBaseLineYWidth = MinBaseLineWidth;//下基准线线宽

}
m_propObj = info;
m_propBak = info;

if(iVersion>=5)
    stream >> sDataSourceList;

QString sInfo = "GetVarIndex/";
for (int i=0; i<info.nDataLineNums, i<sDataSourceList.count(); i++)
{
    sInfo += sDataSourceList[i];
    sInfo += "/";
}

emit sendMsg(sInfo);
```

```
    // 获得数据后立即刷新界面
    m_bUpdateInertiaPixmap = true;
    //refreshPixmap();
    update();
}

//save 函数是在组态画面过程中，调用绘制系统保存事件触发执行
//函数主要是保存插件在组态过程中针对插件属性的配置
//每个插件首先需设计版本号，建议大家延续该功能
//版本号有利于插件后续做升级，便于高版本的插件能够兼容低版本
void QRtCurv::save( QDataStream &stream )
{
    //Q_INT32 iVersion = 3 ;
    qint32 iVersion = 5;
    stream << iVersion ;    //先保存插件的版本号
    stream << m_propObj;
    stream << m_propObj.proFont;
    stream << m_propObj.fontSizes;
    int b3DCheck = 0;
    if(m_propObj.b3DCheck)
    {
        b3DCheck = 1;
    }
    else
    {
        b3DCheck = 0;
    }
    stream << b3DCheck;
    stream << m_propObj.borderColor;
    stream << nCrossLineWidth;

    stream << m_propObj.MaxBaseLineY;//上基准线纵坐标
    stream << m_propObj.MinBaseLineY;//下基准线纵坐标
    stream << m_propObj.MaxBaseLineColor;//上基准线颜色
    stream << m_propObj.MinBaseLineColor;//下基准线颜色
    stream << m_propObj.MaxBaseLineYWidth;//上基准线线宽
    stream << m_propObj.MinBaseLineYWidth;//下基准线线宽

    stream << sDataSourceList;
}
```

以上三个函数是每个插件必须要有，切需自行实现，其它的函数是根据插件的具体功能而设计，这个可以自行定义，不做具体要求

3.5 控件集成

将编译好的控件DLL拷贝到QTouch安装目录plugin的文件夹下

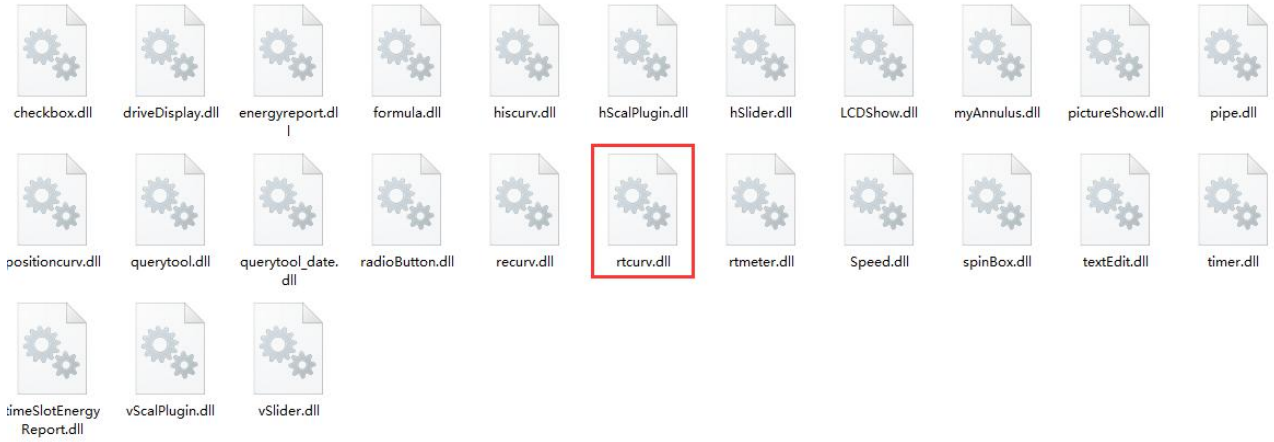


图8-2：控件集成

打开QTouch的画面，绘制系统的左侧下拉列表框就会看到刚才添加的控件组件，选择刚添加的控件，拖动鼠标就可以绘制控件了。右下角可以改变控件的公有属性。

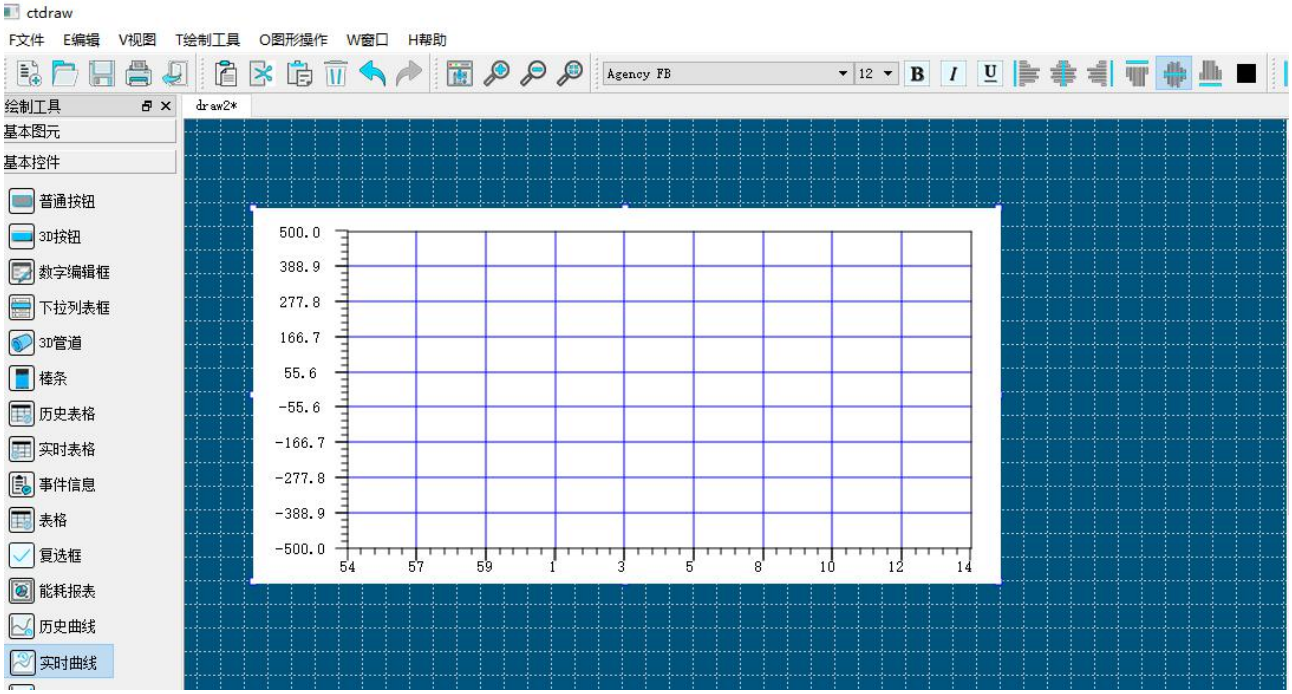


图8-3：集成效果

双击控件就会弹出控件自定义的属性设计编辑框

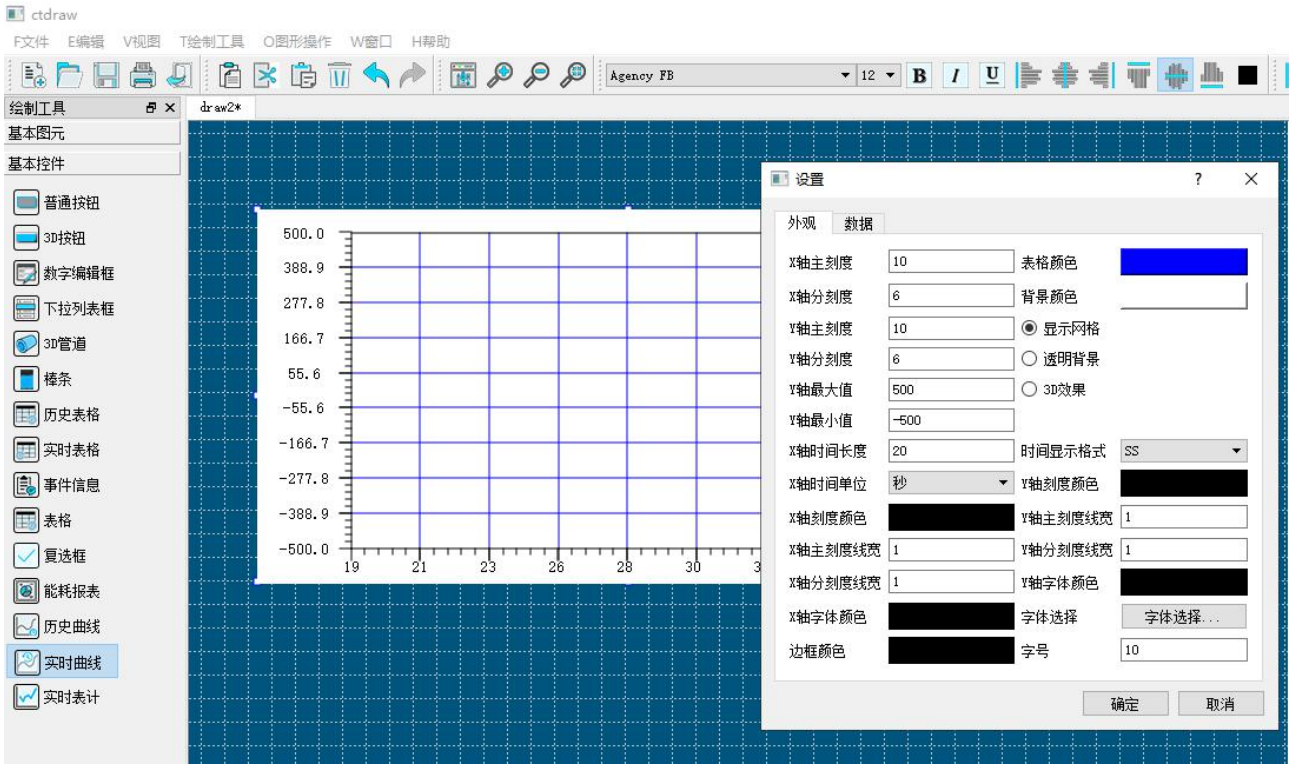


图8-4：控件自定义属性

改变插件属性，点击画面保存，画面关闭后，再查看插件的属性，即为上次修改后的属性；以上的操作即可完成插件在QTouch中的集成；