QTouch 2.0
HMI/SCADA SOLUTIONS

设 计 文 件

名称	QTouch 控件设计说明书
编号	
版本	V2.2.0

武汉舜通智能科技有限公司

标准模板：Q/ST41.34-2010

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

目　次

1 目的和范围

1.1 目的

本文档是对 QTouch 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。

1.2 范围

开发者：软件应用开发人员/客户相关开发人员

开发的控件集成到 QTouch 中，供用户使用

2 规范性引用文件

表1

序号	标准/文件号	标准/文件名称	备注
1		软件编码规范-C++语言	
2		C 语言编码规范使用手册	

3 参考资料

表2

序号	标题	版权信息	编号
1	QT 帮助文档		
2	显示器组态软件需求分析说明书		
3			
4			
5			

4 术语和缩略语

表3

序号	术语/缩略语	描述
1	Qt	一种图形系统
2	Linux	嵌入式操作系统
3		
4		

5 设计规约

5.1 文件命名约定

根据《编码规范》中的命名规则命名文件。

5.2 项目编码命名约定

根据《编码规范》编码。

命名规范：顶层包采用 3 字母大写缩写，其他采用驼峰命名方式。必须采用顶层包名称作为前缀。

进一步的命名约定，项目组在编写文档时需进一步详细具体化。

5.3 文档排版约定

本文档对结构设计中包和组件的描述采用深度优先的顺序进行。

6 关联关系

7 设计策略

1) 以 Qt 图形库为基础，调用 QTouch 提供的外部接口进行设计；
2) 整个控件将从编写，封装，集成几个步骤来完成；
3) 用户可根据接口自定义控件。

8 设计说明

8.1 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：

virtual void Fresh(double dValue, char chChange = 0){ ; }数据刷新函数，在绘制系统中，当点击一个控件的实时关联属性，系统运行起来后，会根据画面的刷新周期实时的刷新该关联的数据。用户可根据这个函数接口获取关联变量的实时值。

virtual void save(QDataStream &stream){ ; }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。

virtual void load(QDataStream &stream){ ; }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。

图8-1：绘制系统右下方的属性选择

virtual void setLineColor(QColor color){ this->m_lineColor = color ;} 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。

virtual void setLineWidth(int width){ this->lineWidth = width;} 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。

virtual void setFillColor(QColor color){ this->m_fillColor = color ;} 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。

virtual void setTextValue(QString txt){ this->m_strText = txt;} 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。

virtual void setTextColor(QColor color){ this->m_textColor = color ;}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。

virtual void setFont(QFont font){ this->m_font = font ;} 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。

8.2　实时数据库接口

ramrt.h定义了QTouch实时数据库的接口，通过这些函数接口，用户可以方面的调用整个运行系统的实时数据。其中有几个经常会用到的接口：

double GetItemValue(int iStationid, int id) ;获取实时变量id的值

void SetItemValue(int iStationid , int id , double dValue) ;设置实时变量id的值为value

double GetSysItem(int id) { return *(pSysItem + id) ; }获取系统变量id的值

void SetSysItem(int id , double dValue) { *(pSysItem + id) = dValue ; }设置系统变量id的值为dValue

void SetKcData(int iStationid, int id,QString data);针对设备，向设备写值，值的信息包含在data里面

void SetKcFlag(int iStationid,int flag);向设备写值时，要通知下面的设备通信驱动程序将写的信息发给设备，此时就应将写的标志flag置1；

int GetKcFlag(int iStationid);通信驱动程序如果正常响应了下发的控制信息就会将控制的标志flag置0，通过此函数接口可以判断下发的的情况。

8.3 控件封装

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: 创建控件窗口
#ifdef OS_WIN
extern "C" __declspec(dllexport) QWidget * createHisReport( QWidget * pWid )
#else ifdef OS_UNIX
```

```
    extern "C" __attribute__((visibility("default"))) QWidget * createHisReport( QWidget *
pWid )
    #endif
    {
        // 设置字符集，GB2312支持中文
        QTextCodec *codec = QTextCodec::codecForName("GB2312");
        QHisReport *hisreport = new QHisReport( pWid, "hisreport" );
        hisreport->setCaption("hisreport");
        return hisreport ;
    }

    //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(与上面的函数接口保持一致)，第二个为描述文字
    //(绘制系统中下拉列表中显示的内容)
    #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
    {
        list << "createHisReport" << "历史报表" ;
    }
```

每一个编写的控件都需按照上面的形式进行封装。

8.4 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

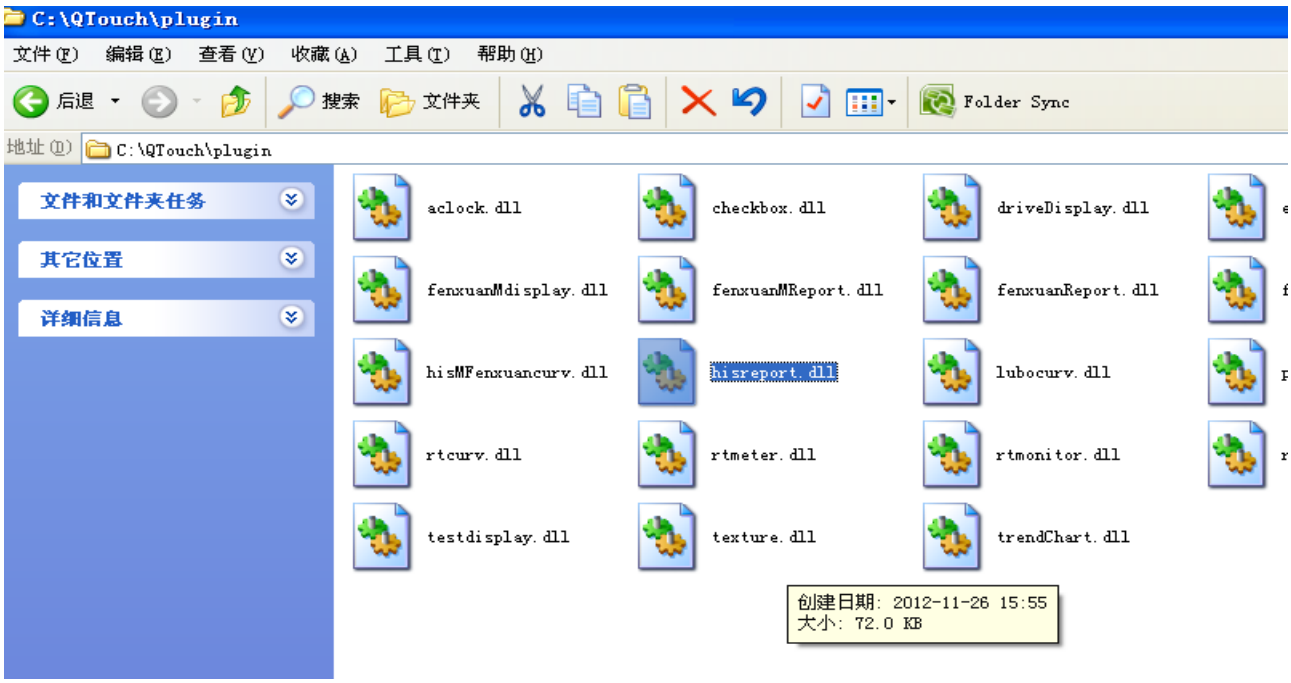

图8-2：控件集成

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。

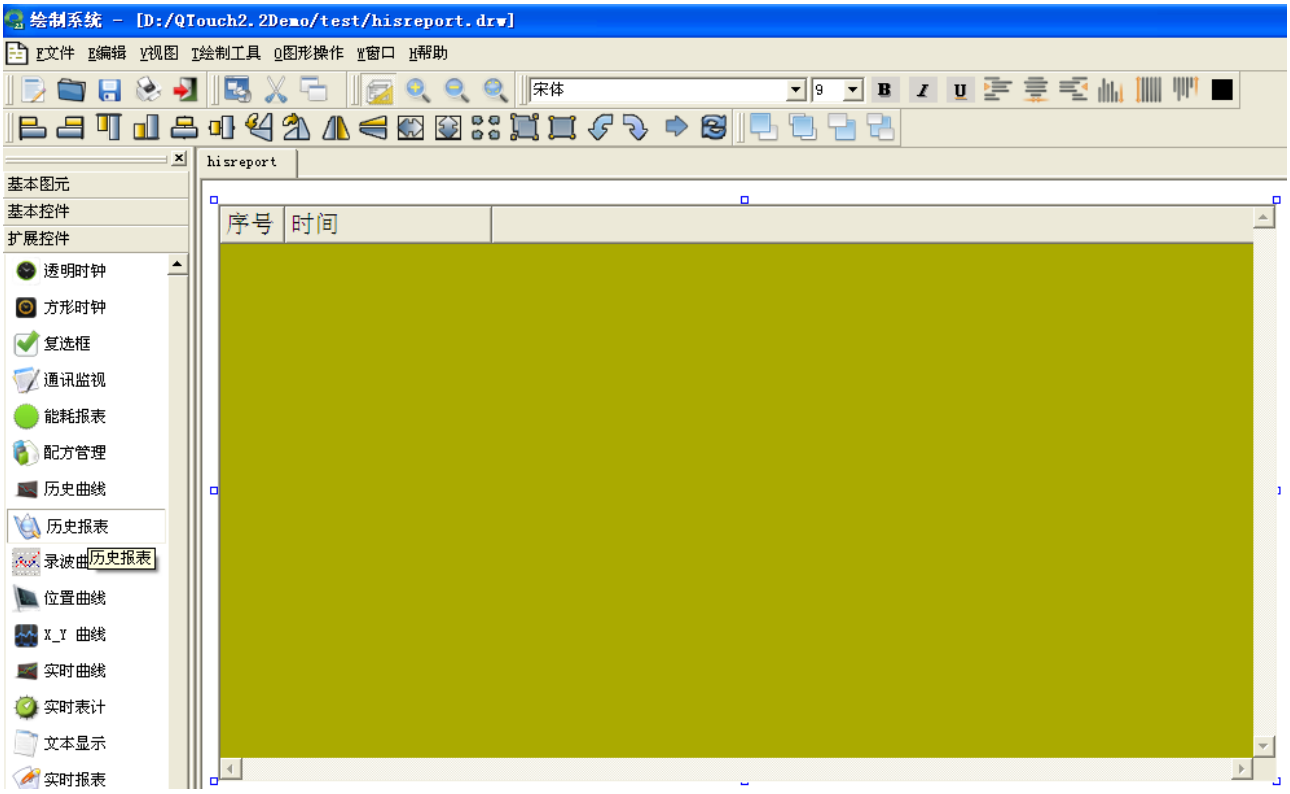

图8-3：集成效果

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