QTouch 2.0
HMI/SCADA SOLUTIONS

设 计 文 件

名称　Qtouch 驱动文件设计说明

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

1 目的和范围

1.1 目的

本文档详细介绍了 Qtouch 驱动文件设计的基本格式和接口，使用这些接口函数可实现与设备的通讯、QTouch 通讯配置的连接和实时数据库的交互。

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 术语和缩略语

表2

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	QT3	Qt 的版本号
4		

4 驱动开发包组成（以 modbusRTU 为例）

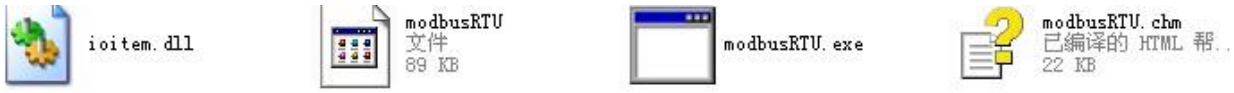
表3

序号	一级文件	二级文件
1	Bin	ioitem.dll io 连接属性设置 dll
2		modusRTU linux 下执行程序
3		modbusRTU.exe windows 下执行程序
4		modbusRTU.chm chm 格式帮助文档

序号	一级文件	二级文件
5	Source	modbusRTU_linV2.2　lin 下源程序
6		modbusRTU_winV2.2　win 下源程序

驱动包文件组成如下图：

Bin 文件夹下组成如下图：



Source 文件夹下组成如下图：

5　驱动执行依赖关系（以 modbusTCP 为例）

表4

序号	文件	依赖关系
1	modbusTCP	增加一个 modusTCP 驱动之后,从 drive 包中拷贝出来的 linux 下执行程序
2	modbusTCP.exe	增加一个 modusTCP 驱动之后,从 drive 包中拷贝出来的 win 下执行程序
3	modbusTCP.xml	增加一个 modusTCP 驱动之后,由驱动设置窗口配置的初始化参数
4	modbusTCP_Reg.xml	由数据库管理配置的 IO 关联参数,根据这个文件进行发送帧配置

工程文件包组成如下：

6　结构分解

6.1　顶层包

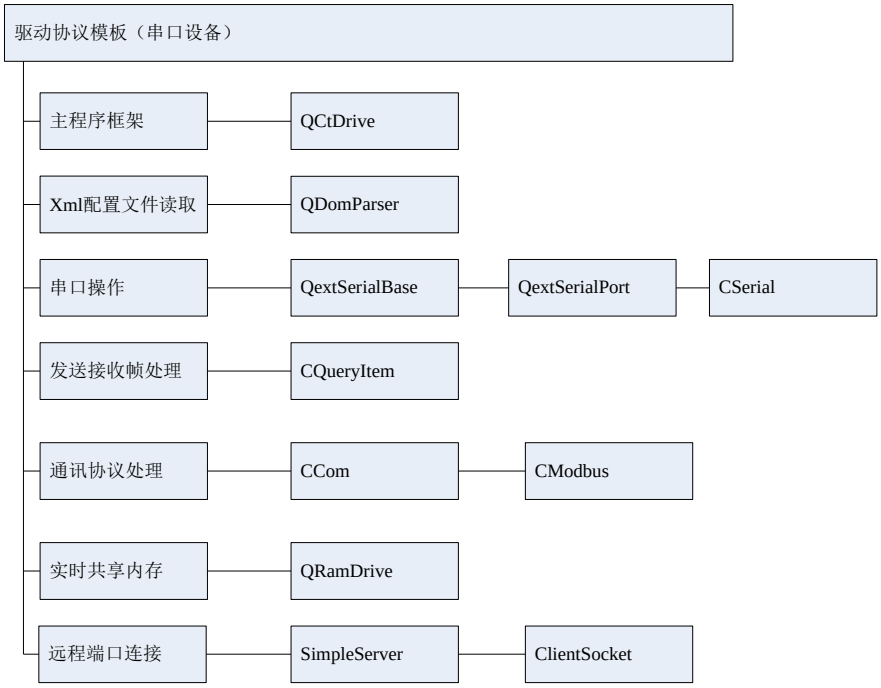
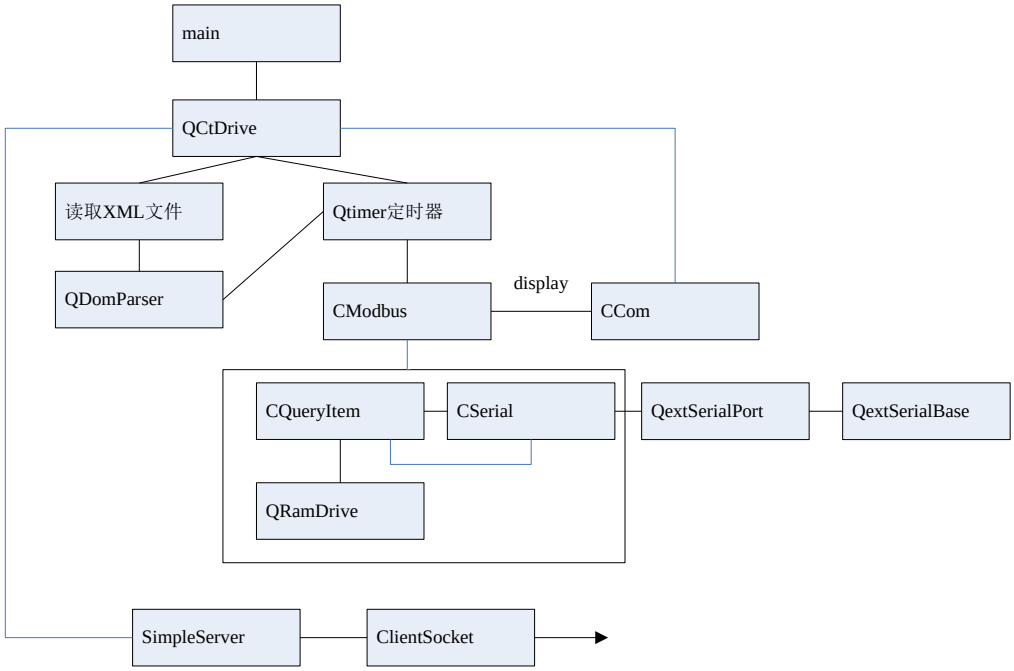


表5

序号	主要接口文件	描述
1	ramrt.h ， ramrt.cpp	实时数据库系统函数封装
2	domparser.h ， domparser.cpp	XML 文件读取，获取用户配置信息的接口
3	serial.h，serial.cpp	串口操作封装，对串口读写之调用
4	qextserialbase.h，qextserialbase.cpp qextserialport.h，qextserialport.cpp	串口操作基本类
5	win_qextserialport.h，win_qextserialport.cpp	Windows 下串口设置和操作类，与 linux 下而选一
6	posix_qextserialport.h，posix_qextserialport.cpp	Linux 下串口设置和操作类，与 windows 下而选一
7	ctsocket.h ， ctsocket.cpp	TCP 服务器端封装类
8	socketclient.h，socketclient.cpp	TCP 客户端封装类
9	trayicon.h，trayicon.cpp trayicon_win.h，trayicon_win.cpp	Windows 下驱动程序显示图标类，仅 windows 下使用
10	comm.h ， comm.cpp	对驱动程序处理的父类，共有特性初始化
11	Modbus.h ， Modbus.cpp	特定 Modbus 程序通讯处理类
12	ctdrive.h，ctdrive.cpp	主框架处理类

序号	主要接口文件	描述
13	main.cpp	程序 main 入口

6.2 执行流程



6.3 主要函数说明

1、 bool CreateRam(int iStationNum);

```
/*!
* 描述
*        打开实时数据库共享内存，在操作实时数据库之前必须先打开共享内存
* 参数    iStationNum
*        站号，默认为0
* 返回
*        返回true表示成功，否则表示失败
* @note
*        接口类型：阻塞式
*/
```

2、double GetItemValue(int iStationid , int id)

```
/*!
* 描述
*        获取实时数据库指定编号的数据
* 参数    iStationid
*        站号，默认为0
* 参数    id
```

```
*          指定数据在内存中编号
* 返回
*          返回数据值
* @note
*          接口类型：阻塞式
*/
```

3、**void SetItemValue(int iStationid , int id , double dValue)**

```
/*!
* 描述
*          设置实时数据库指定类型指定编号的数据值
* 参数     iStationid
*          站号，默认为0
* 参数     id
*          指定数据在内存中编号
* 参数     dValue
*          设置的数值
* 返回
*          无返回值
* @note
*          接口类型：阻塞式
*/
```

4、**int GetKcFlag(int iStationid)**

```
/*!
* 描述
*          获取实时数据库下发控制标志
* 参数     iStationid
*          站号，默认为0
* 返回     int
*          返回实时数据库下发控制标志，为1表示有控制要下发，否则为空
* @note
*          接口类型：阻塞式
*/
```

5、**StructKcData GetKcData(int iStationid)**

```
/*!
* 描述
*          获取实时数据库下发控制信息，一般在GetKcFlag( 0 )返回1后调用
* 参数     iStationid
*          控制量编号
* 返回     StructKcData
*          struct StructKcData
           {
               int kcNo;          //下发控制的数据内存序号
```

```
            char dataKC[200];  //传递下来的数值
    *       };
    * @note
    *       接口类型：阻塞式
    */
```

6、**void SetKcFlag(int iStationid, int flag)**

```
    /*!
    * 描述
    *       设置实时数据库下发控制信息，一般在GetKcData（0）获取到控制信息后清空标志
    * 参数    iStationid
    *       站号，默认为0
    * 参数    flag
    *       需要被设置的值
    * 返回
    *       无返回值
    * @note
    *       接口类型：阻塞式
    */
```

7、**void SetKcClear(int iStationid,int flag)**

```
    /*!
    * 描述
    *       清除数据库下发控制信息
    * 参数    iStationid
    *       站号，默认为0
    * 参数    flag
    *       为0时，表示清空，否则无效
    * 返回
    *       无返回值
    * @note
    *       接口类型：阻塞式
    */
```

8. bool readEtherNetLinkLayer_Reg(QString path ,QString sname)

```
    /*!
    * 描述
    *       打开XML文件，并读取信息
    * 参数    path
    *       文件路径
    * 参数    sname
    *       文件名
    * 返回
    *       返回true表示成功，否则表示失败
    * @note
```

```
*        接口类型：阻塞式
*/
```

9. bool CSerial::OpenCom( )

```
/*!
* 描述
*        打开串口，此前需要先配置好“PortSettings settings”结构的串口参数
* 参数
*        无
* 返回
*        返回true表示成功，否则表示失败
* @note
*        接口类型：阻塞式
*/
```

10. int CSerial::ReadCom(unsigned char * lpBuf, int dataLen)

```
/*!
* 描述
*        读取串口信息
* 参数    lpBuf
*        接收数据的结构
* 参数    dataLen
*        接收数据的最大长度，函数会在接收到此长度的数据或1S未响应后返回
* 返回    int
*        接收到的数据字节长度
* @note
*        接口类型：阻塞式
*/
```

11. int CSerial::WriteCom(const void* lpBuf, int dwCount)

```
/*!
* 描述
*        写入串口信息
* 参数    lpBuf
*        写入的数据结构
* 参数    dwCount
*        被写入的数据串长度
* 返回    int
*        写入的数据字节长度
* @note
*        接口类型：阻塞式
```

```
*/
```

12. int QSocketClient::readSocketBlock(unsigned char* Buf,int maxLen)

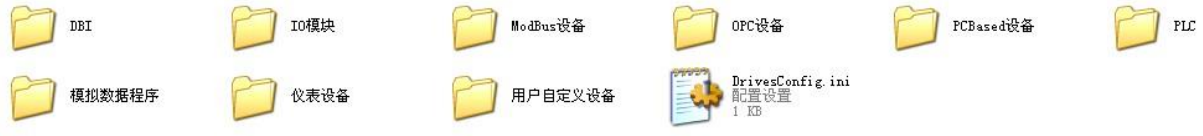
```
/*!
* 描述
*        读取网口接收到的数据
* 参数    Buf
*        用以接收数据的结构
* 参数    maxLen
*        接收最大长度
* 返回    int
*        接收到的数据字节长度
* @note
*        接口类型：阻塞式
*/
```

13. int QSocketClient::writeSocketBlock(unsigned char *Buf,int iLen)

```
/*!
* 描述
*        写入数据到网口
* 参数    Buf
*        传入的数据结构
* 参数    iLen
*        传入的数据最大长度
* 返回    int
*        写入数据字节长度
* @note
*        接口类型：阻塞式
*/
```

7 用户自定义驱动方法

用户可以将自己定义的驱动文件放到QTouch的drives目录下，按照一定的规格保存，即可以实现自定义驱动的添加，QTouch的参数系统就会找到这个文件。



规则文件为 DrivesConfig.ini，主要对 drives 下面的文件夹进行管理，用户可以增加一个专用的文件夹。文件夹分为 2 级管理，如上图所示为一级文件夹，主要是对设备的分类，一般用户可以不新建一级文件夹，在一级文件夹之下，可以新建自己的文件夹，用于将自定义的程序保存进来。

以 ModBus 设备为例：

```
[Drive4]
DriveName=ModBus设备
SonNum=5
SonDrive0=modbusRTU
SonDrive1=modbusTCP
SonDrive2=modbusRTU_slav
SonDrive3=modbusASCII
SonDrive4=modbusTCP_server
```

如上表明：在 ModBus 设备这个文件夹下面有 5 个子文件夹，如用户要增加一个专用文件夹则可以如下操作：

```
[Drive4]
DriveName=ModBus设备
SonNum=6
SonDrive0=modbusRTU
SonDrive1=modbusTCP
SonDrive2=modbusRTU_slav
SonDrive3=modbusASCII
SonDrive4=modbusTCP_server
SonDrive5=自定义文件
```

完成这个设置之后，在 Modbus 设备下就可以建立一个自定义文件夹了：

在自定义文件夹下，用户就可以添加自定义驱动，需要增加一个配置文件 CONFIG.ini：

```
[Drive]
DrivesNum=1

[Drive0]
DriveName=自定义
DriveType=2
ProtocolNum=1
Protocol0=自定义
```

文件夹内容如下：

完成后在 QTouch 的设备选择中就会出现自定义的选项：

协议选择中就会出现自定义协议的选项：