# 国赛qt库API文档

上海企想信息技术有限公司

软件开发部

|  |  |
| --- | --- |
| 作者 | 蒋慧 |
| 版本号 | V3.0 |
| 审核人 |  |
| 最后更新时间 | 2018.9.4 |

## 数据库类（Sql）

### bool sqlConnect(QString dbName)

函数功能：链接数据库，若数据库存在则打开数据库，否则新建数据库并打开

函数参数：dbname：数据库名

函数返回：若操作成功则返回ture，否则返回false

### bool sqlExec(QString sql)

函数功能：执行数据库的增，删，改操作

函数参数：sql：sql语句

函数返回：若操作成功则返回ture，否则返回false

举例：sqlExec(“create table if not exists tb\_eg(id integer primary key autoincrement, name varchar(15), passwd varchar(15))”);

### QVariant sqlQuery(QString sql)

函数功能：执行数据库的查询操作

函数参数：sql：sql 查询语句

函数返回：若操作成功则返回QVariant类型数据，可将其转为需要的类型

举例：QString pwd = sqlQuery(“select passwd from tb\_eg where id = 1”).toString();

### QSqlTableModel\* sqlTableMode(QString tb)

函数功能：执行数据库的查询整张表，并将表存入临时

函数参数：tb：表名

函数返回：若操作成功则返回tb表

举例：QSqlTableModel \* mode = new QSqlTableModel(this);

mode = sqlTableMode(“tb\_eg”);

delete mode;

mode = NULL;

### bool sqlDeleteTable(QString tb)

函数功能：执行删除数据表

函数参数：tb：表名

函数返回：若操作成功则返回ture，否则返回false

### bool sqlClearTable(QString tb)

函数功能：执行清空数据表，并将自增长字段重新置0

函数参数：tb：表名

函数返回：若操作成功则返回ture，否则返回false

### bool sqlDisconnect()

函数功能：关闭数据库

函数参数：无

函数返回：若操作成功则返回ture，否则返回false

### 表结构说明

|  |  |  |
| --- | --- | --- |
| 数据库名 | db\_modelRoom.db | |
| 表名 | 字段 | 说明 |
| tb\_gatewayInfo | username (char) | 用户名 |
| passwd (char) | 密码 |
| serverIp (char) | 服务器Ip |
| tb\_zigbeeInfo | sensorType (char) | 见附表1 |
| boardId (int) | 板号 |

注：qt中同一时间只可以打开一个数据库，若需使用多个数据库，则需每次数据库操作完成以后需关闭该数据库，若需要其他数据表，可以直接在db\_modelRoom.db数据库中创建。

## 串口数据处理

**SerialPortThread类：**

串口线程：串口参数配置（波特率，数据位，校验位等，串口接收方式为EventDriven）

EventDriven：事件驱动方式，当有数据收到，串口发射readyRead（）信号，用户将其与接收函数进行信号槽连接。

### void onReadyRead ()

函数功能：数据接收槽函数。

函数参数：无

函数返回：无

### int send (QByteArray ba)

函数功能：串口发送数据处理

函数参数：ba：待发送数据

函数返回：返回已发送的字节数，为-1时发送失败。

**SerialPortReceive类：**

串口数据接收处理类：将接收的数据处理为完整的一桢数据，并校验帧头和数据长度。

### void dataHandle(QByteArray temp)

函数功能：串口接收数据处理，将接收的数据处理为完整的一桢数据，并通过信号函数发给串口线程

函数参数：**temp**：接收的数据

函数返回：无

### void byteFinish(const QByteArray &);

函数功能：数据处理完成信号

函数参数：校验成功的一桢数据

函数返回：无

**SerialDataHandle类：**

串口数据处理类：1）将接收的一桢数据进行处理，储存；

2）将需要发送的数据进行转换。

### QByteArray sendingDataHandle(unsigned char opcodes, unsigned boardId, unsigned int \*pCommand,int len);

函数功能：发送数据处理

函数参数：1）opcodes（操作码）：0x06控制设备；0x07获取状态；

2）boardId（板号）：节点设备板号

3）\* pCommand（发送数据的数据部分）具体内容见demo。

eg：unsigned int pCmd[9];

pCmd[0] = 0; //预定义字段，默认为0

pCmd[1] = 0;//预定义字段，默认为0

pCmd[2] = 0; //设备类型低字节，见附表5,如继电器：0x01

pCmd[3] = 0;// 设备类型高字节，见附表5，如继电器：0xee

pCmd[4] = 0x03; //命令长度两个字节，低字节

pCmd[5] = 0; //命令长度两个字节，高字节

pCmd[6] = 0xXX; //命令码cmdCode，见附表2

pCmd[7] = 0xXX; //命令cmd，或红外设备通道低字节见附表3

pCmd[8] = 0xXX;//通道Channel，或红外设备通道高字节见附表4

4）len（数组长度）：pCommand数组长度。

函数返回：待发送的数据

### evt\_deviceState(const QString &,const QString & ,const QString & )

函数功能： 设备状态发生改变后发出信号

函数参数：参数1：见附表1传感器类型表；参数2:见附表2命令码表；参数3：设备值

函数返回： 无

### void receive(const QByteArray & str);

函数功能：串口数据接收处理函数，将由串口线程接收的数据进行解析

函数参数：str：接收的一桢数据

函数返回：无

### 设备信息结构体

typedef struct{

QString shortAddress; //设备短地址

QString deviceType; //设备类型

QString mac; //设备mac

}deviceInfoStruct;

deviceInfoStruct devInfoStruct;

QMap<unsigned int,deviceInfoStruct> deviceInfoMap; // 设备信息map

## TCP/IP客户端线程(TcpClientThread)

### void upload(QString type,QString cmdCode,QString value)

函数功能：传感器数据主动上传给服务器

函数参数：type: 见附表1传感器类型表；cmdCode:见附表2命令码表；value：设备状态值

函数返回：无

### void evt\_socket\_receive(QString type,unsigned int command,unsigned int cmdCode,unsigned int channel)

函数功能：信号函数，接收服务端数据处理后，若数据处理后为设备控制，则向主线程发出信号,使用时需建立信号槽连接

函数参数：type:见附表1传感器类型表；command:控制命令，见附表3设备命令表；cmdCode：见附表2命令码表；channel：见附表4设备通道表

函数返回：无

### void threadQuit()

函数功能：断开socket链接，并退出客户端线程

函数参数：无

函数返回：无

### QString serverIP

属性：链接服务器的ip，在主线程中对其进行设置，可以对其直接赋值，也可以读取数据库中保存的地址。

设置方法：pClient->serverIP = serverIp;

### bool getTcpClientFlag(){return tcpClientFlag;}

函数功能：获取客户端线程当前状态；

函数参数：无

函数返回：true:线程运行；false:线程退出

### void setTcpClentFlag(bool b){tcpClientFlag =b;}

函数功能：设置客户端线程当前状态标志；

函数参数：参数b:true线程运行；false线程退出

函数返回：无

## TCP/IP服务端线程(TcpServer)

说明：该线程只需开启服务监听和建立信号槽连接，无需其他操作，用于配置网关。

TcpServer.listen(IP,Port);

connect(pServer,SIGNAL(evt\_configBoardId()),this,SLOT(slot\_initBoardId()));

## 预处理定义(VariableDefinition.h)

详见头文件注释

## 系统定义(systemConfig.h)

### #define ARM

属性:当需要在ARM板上运行程序时，需定义ARM字段，若在虚拟机中运行程序则注释此条语句

## 附表：

### 传感器类型表

#define TEMP\_SERVE "0" //温度

#define HUM\_SERVE "1" //湿度

#define ILL\_SERVE "2" //光照

#define SMK\_SERVE "3" //烟雾

#define GAS\_SERVE "4" //燃气

#define PM25\_SERVE "5" //PM2.5

#define CO2\_SERVE "6" //二氧化碳

#define AP\_SERVE "7" //气压

#define HI\_SERVE "8" //人体红外

#define FAN\_SERVE "9" //风扇

#define LAMP\_SERVE "10" //射灯

#define WL\_SERVE "11" //报警灯

#define CT\_SERVE "12" //窗帘

#define INFRARED\_1\_SERVE "13" //红外发射1

#define DC\_SERVE "15" //门禁控制

### 命令码表

#define SWITCH 0x01 //开关

#define INFRAREDSTUDY 0x02 //红外学习

#define INFRAREDEXIT 0x00 //红外退出学习

#define RFID\_APP 0x01 //app开门

#define RFID\_CARD 0x02 //刷卡开门

#define RFID\_KEY 0x04 //外部按键开门

#### RFID设备：

App开门：0x01

刷卡开门：0x02

外部按键：0x04

#### 红外设备：

红外退出学习：0x00

红外发射：0x01

红外学习：0x02

#### 红外设备控制

#define AIR\_OPEN 0x01 //开空调

#define AIR\_CLOSE 0x02 //关空调

#define TV\_OPEN 0x03 //开电视

#define TV\_CLOSE 0x04 //关电视

#define DVD\_OPEN 0x05 //开DVD

#define DVD\_CLOSE 0x06 //关DVD

#### 其他设备：

默认为空

### 设备命令表

#define SWITCH\_ON 0x01 //设备开

#define SWITCH\_OFF 0x00 //设备关

### 设备通道表

#define CHANNEL\_1 0x01 //通道1

#define CHANNEL\_2 0x02 //通道2

#define CHANNEL\_3 0x04 //通道3

#define CHANNEL\_4 0x08 //通道4

#define CHANNEL\_ALL 0x07 //所有通道

### 上传状态

#define ON "1"

#define OFF "0"

#define STOP "2"

#define CLOSE "4"

### 设备类型表

#define RELAY 0XEE01 //继电器

#define BODYINFRARED 0XEE02 //人体红外

#define SMOKE 0XEE03 //烟雾

#define CO2 0XEE04 //二氧化碳

#define GAS 0XEE05 //燃气

#define PRESSURE 0XEE06 //气压

#define THI 0XEE07 //温湿度

#define ILL 0XEE08 //光照

#define PM25 0XEE09 //PM2.5

#define RFID 0XEE0A //门禁

#define INFRARED 0XEE0B //红外转发