**写盘器进程通讯协议**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改内容 | 修改人 | 日期 |
| 1.0 | 初始版本 | Lc | 2021-04-26 |
| 1.1 | 添加标签离场指令 4.2章节  添加联网状态指令 4.3章节 | Lc | 2021-05-10 |
| 1.2 | 添加工厂检测指令 5.1 章节 | Lc | 2021-08-31 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[一、编写目的 3](#_Toc81318821)

[二、 通讯方式 3](#_Toc81318822)

[2.1、通讯方式 3](#_Toc81318823)

[2.2、消息队列结构体 3](#_Toc81318824)

[2.3、数据效验 4](#_Toc81318825)

[三、接口定义（QT>>GOLANG） 5](#_Toc81318826)

[3.1、QT发送写盘信息到golang 5](#_Toc81318827)

[3.1.1、指令说明 5](#_Toc81318828)

[3.1.2、QT >> golang (0x01) 5](#_Toc81318829)

[3.2、QT发送停止写盘到golang 6](#_Toc81318830)

[3.2.1、指令说明 6](#_Toc81318831)

[3.2.2、QT >> golang (0x02) 7](#_Toc81318832)

[3.3、QT发送配网指令到golang 8](#_Toc81318833)

[3.3.1、指令说明 8](#_Toc81318834)

[3.3.2、QT >> golang (0x03) 8](#_Toc81318835)

[3.4、QT发送现场绑定指令到golang 9](#_Toc81318836)

[3.4.1、指令说明 9](#_Toc81318837)

[3.4.2、QT >> golang (0x04) 9](#_Toc81318838)

[3.5、QT发送解绑指令到golang 11](#_Toc81318839)

[3.5.1、指令说明 11](#_Toc81318840)

[3.5.2、QT >> golang (0x05) 11](#_Toc81318841)

[四、接口定义（GOLANG>>QT） 12](#_Toc81318842)

[4.1、golang发送写盘状态到QT 12](#_Toc81318843)

[4.1.1、指令说明 12](#_Toc81318844)

[4.1.2、golang >> QT（0xA1） 13](#_Toc81318845)

[4.2、golang发送标签离场状态到QT 14](#_Toc81318846)

[4.2.1、指令说明 14](#_Toc81318847)

[4.2.2、golang >> QT（0xA2） 14](#_Toc81318848)

[4.3、golang发送联网状态到QT 15](#_Toc81318849)

[4.3.1、指令说明 15](#_Toc81318850)

[4.3.2、golang >> QT（0xA3） 16](#_Toc81318851)

[4.4、golang发送closetid到QT,QT已二维码显示 17](#_Toc81318852)

[4.4.1、指令说明 17](#_Toc81318853)

[4.4.2、golang >> QT（0xA4） 17](#_Toc81318854)

[4.5、golang发送设备状态到QT 18](#_Toc81318855)

[4.5.1、指令说明 18](#_Toc81318856)

[4.5.2、golang >> QT（0xA5） 19](#_Toc81318857)

[4.6、golang发送获取菜单状态到QT 20](#_Toc81318858)

[4.6.1、指令说明 20](#_Toc81318859)

[4.6.2、golang >> QT（0xA6） 20](#_Toc81318860)

[4.7、golang发送固件更新状态到QT 22](#_Toc81318861)

[4.7.1、指令说明 22](#_Toc81318862)

[4.7.2、golang >> QT（0xA7） 22](#_Toc81318863)

[五、工厂检测接口定义（GOLANG>>QT） 23](#_Toc81318864)

[5.1、golang发送工厂检测指令到QT 23](#_Toc81318865)

[5.1.1、指令说明 23](#_Toc81318866)

[5.1.2、golang >> QT（0xFF） 24](#_Toc81318867)

# 一、编写目的

美餐写盘器QT与golang进程间通讯

# 通讯方式

## 2.1、通讯方式

进程间采用消息队列方式

## 2.2、消息队列结构体

**struct mymesg**

**{**

**int mtype; // Linux系统通信消息类型 （代码编写的消息类型 已固定）**

**char mtext[100];**

**};**

**golang发送QT接收队列结构体定义：**

Snd\_msg.mtype = 88;

key\_t key = ftok("/tmp", 88);

int id = msgget(key, IPC\_CREAT | 0666);

**QT发送golang接收队列结构体定义：**

Snd\_msg.mtype = 66;

key\_t key = ftok("/tmp", 66);

int id = msgget(key, IPC\_CREAT | 0666);

## 2.3、数据效验

需要效验的数据（数据长度+CMD+DATA）

*//异或效验*

static unsigned char Cal\_Crc(unsigned char \**Data*, unsigned char *Len*)

{

  unsigned char Crc = 0, i = 0;

  for(i=0; i<*Len*; i++)

  {

    Crc = Crc ^ *Data*[i];

  }

  return Crc;

}

# 三、接口定义（QT>>GOLANG）

## 3.1、QT发送消费额到golang

### 3.1.1、指令说明

1、QT发送消费额（消费金额）到QT

Golang收到本条指令后才允许写盘.

### 3.1.2、QT >> golang (0x01)

#### 发送数据:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| 数据长度 | Uint8\_t | 6 |  |
| CMD | Uint8\_t | 0x01 |  |
| 消费额 | Uint8\_t \*4 |  | ID==1234(小端模式)  D2 04 00 00 |
| CRC8 | Uint8\_t |  |  |
| |  | | --- | | 示例：  06 01 D2 04 00 00 CRC | | | | |

#### 返回参数:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| 数据长度 | Uint8\_t | 3 |  |
| CMD | Uint8\_t | 0x01 |  |
| 状态 | Uint8\_t | 00 | 成功===0x00  失败===0x01 |
| CRC8 | Uint8\_t |  |  |
| |  | | --- | | 示例：  03 01 00 CRC | | | | |

## 3.2、QT发送消费取消到golang

### 3.2.1、指令说明

1、QT发送消费取消命令到golang

### 3.2.2、QT >> golang (0x02)

#### 发送数据:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| 数据长度 | Uint8\_t | 2 |  |
| CMD | Uint8\_t | 0x02 |  |
| CRC8 | Uint8\_t |  |  |
| |  | | --- | | 示例：  02 02 CRC | | | | |

#### 返回参数:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| 数据长度 | Uint8\_t | 3 |  |
| CMD | Uint8\_t | 0x02 |  |
| 状态 | Uint8\_t | 00 | 成功===0x00  失败===0x01 |
| CRC8 | Uint8\_t |  |  |
| |  | | --- | | 示例：  03 02 00 CRC | | | | |

# 四、接口定义（GOLANG>>QT）

## 4.1、golang发送消费状态到QT

### 4.1.1、指令说明

1、golang发送消费状态到QT

### 4.1.2、golang >> QT（）

#### 发送数据:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| cmd | string | “getConsume” | 消费结果 |
| cardNum | String | “11223344” | 卡号 |
| consumeStatus | String | “消费成功”  “黑名单卡”  “批次错误”  “金额不足”  “次数不足”  “超过限额” | 消费结果 |
| name | string | “张三” | 用户名字 |
| currentMoney | int | 10000 | 10.000 |
| balance | int | 9999 | 9.999 |
| |  | | --- | | 示例：  {  “cmd”: “getConsume”,  “cardNum”: “11223344”,  “consumeStatus”: “消费成功”,  “name”: “张三”,  “currentMoney”: 10000,  “balance”:9999  } | | | | |

## 4.2、golang发送查询结果到QT

### 4.1.1、指令说明

1、golang发送查询结果到QT

### 4.1.2、golang >> QT（）

#### 发送数据:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| cmd | string | “getBalance” | 查询结果 |
| cardNum | String | “11223344” | 卡号 |
| consumeStatus | String | “查询成功”  “黑名单卡”  “批次错误”  “金额不足”  “次数不足”  “超过限额” | 消费结果 |
| name | string | “张三” | 用户名字 |
| balance | int | 9999 | 9.999 |
| |  | | --- | | 示例：  {  “cmd”: “getBalance”,  “cardNum”: “11223344”,  “consumeStatus”: “查询成功”,  “name”: “张三”,  “balance”:9999  } | | | | |

## 4.3、golang发送卡片离场到QT

### 4.3.1、指令说明

1、golang发送卡片离场到QT

### 4.3.2、golang >> QT（）

#### 发送数据:

| **字段名** | **类型** | **数据** | **备注** |
| --- | --- | --- | --- |
| cmd | string | “removeCard” | 卡片离场 |
| |  | | --- | | 示例：  {  “cmd”: “removeCard”,  } | | | | |