板卡通信接口子协议

*版本号：V1.0*

*2019年5月*

目录

[目录 1](#_Toc4884)

[1 概述 2](#_Toc9674)

[1.1 修改记录 2](#_Toc17642)

[1.2 名词及缩写解释 2](#_Toc15027)

[2 通信协议 3](#_Toc5382)

[2.1 协议格式 3](#_Toc29558)

[2.2 通信方式 4](#_Toc28317)

[2.3 帧命令(CMD)说明 4](#_Toc23984)

[2.4 指令格式 4](#_Toc17171)

[2.5 通信命令分配表 5](#_Toc16011)

[3 用户命令 (0xA000, 0xC000) 5](#_Toc16110)

[3.1 煲仔炉项目模块 5](#_Toc17491)

[3.1.1 获取煲仔炉单个板卡的状态信息 5](#_Toc17319)

[3.1.2 设置煲仔炉板卡的煮饭流程 7](#_Toc28050)

[3.1.3 查询当前菜单内容 8](#_Toc13515)

[3.1.4 从菜单服务器更新菜单 9](#_Toc6692)

[3.1.5 推送新的菜单内容 10](#_Toc23359)

[3.1.6 上位机获取煲仔炉所有板卡参数 10](#_Toc7655)

[4 返回码参数表 11](#_Toc30643)

[5 设备信息表 11](#_Toc10739)

[6 CRC校验算法示例 12](#_Toc9596)

# 概述

## 修改记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 修改人 | 说明 |
| 1.0 | 2019-12-24 | 屈孟 | 炒锅通信协议拆分初稿 |

## 名词及缩写解释

|  |  |
| --- | --- |
| 缩写 | 内容 |
| 下位机 | MS-TWO 或其它 |
| 上位机 | PC/手机/其它 |
| SOF | Start of frame， 数据通信起始符 |
| FRLEN | Frame length,通信帧长度 |
| DATA | Data， 通信交互数据 |
| RC | Return code, 返回码 |
| CRC | CRC 校验 |
| CRECTP | Current Encryption type, 当前数据加密类型 |

# 通信协议

## 协议格式

本文档所描述的数据均采用大端模式， 数据高字节保存在内存低地址，数据低字节保存在内存高地址。

请求报文为主动发起的命令，需要得到应答报文回复。

上下位机通信的单次data 数据最大长度为 1491 (1500 - 9)bytes, 单独指 DATA段， 不包含其它信息数据, 其它数据段最大 9 个字节.

上位机收发格式：

请求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项 | 字节 | 含义 | 格式 | 备注 |
| SOF | 1 | 数据包包头标识符 | HEX | 固定填0xAA |
| CMD | 2 | 命令码 | HEX | 指明当前通信报文的类型与功能 |
| FRLEN | 2 | 数据包内容长度 | HEX |  |
| DATA | N | 数据内容 | HEX | 具体的数据 |
| CRC | 2 | CRC16校验码 | HEX | 除去包头的其它所有数据进行CRC16校验运算 |

应答报文

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项 | 字节 | 含义 | 格式 | 备注 |
| SOF | 1 | 数据包包头标识符 | HEX | 固定填0xAA |
| CMD | 2 | 命令码 | HEX | 指明当前通信报文的类型与功能 |
| RC | 2 | 返回码 | HEX | 返回响应状态 |
| FRLEN | 2 | 数据包内容长度 | HEX |  |
| DATA | N | 数据内容 | HEX | 具体的数据 |
| CRC | 2 | CRC16校验码 | HEX | 除去包头的其它所有数据进行CRC16校验运算 |

说明：

SOF：起始域，1字节，值固定为0XAA；

CMD : 2 字节,指明当前通信报文的类型与功能;

RC : 2 字节， 指明返回帧状态， 参见 [返回码状态表](#_返回码参数表);

FRLEN：长度域，2字节，表示DATA数据的长度（仅为 DATA长度）；

DATA ：数据域，N字节，传输的有效数据

CRC：校验域，2字节，除去包头的其它所有数据进行CRC16校验运算, 参见 [CRC算法示例](#_CRC校验算法示例);

## 通信方式

采用同步和异步通讯方式。与上位机通过网络/串口/usb/串口等进行通信,具体待定。

上下位机通过命令类型判断通讯报文是否需要回复, 具体参考[帧命令说明](#_帧命令说明).

## 帧命令(CMD)说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 命令说明 | 命令值 | | | 12...0 |
| 15 | 14 | 13 |
| 请求命令, 上位机发送给下位机并且等待回复 | 0 | 0 | 0 | 0x0000 -- 0x1fff |
| 广播命令 | 0 | 0 | 1 | 0x0000 -- 0x1fff |
|  | 0 | 1 | 0 | 0x0000 -- 0x1fff |
|  | 0 | 1 | 1 | 0x0000 -- 0x1fff |
|  | 1 | 0 | 0 | 0x0000 -- 0x1fff |
| 用户命令区间, 由开发应用的相关人员定义 | 1 | 0 | 1 | 0x0000 -- 0x1fff |
|  | 1 | 1 | 0 | 0x0000 -- 0x1fff |
|  | 1 | 1 | 1 | 0x0000 -- 0x1fff |

## 指令格式

请求或通知报文:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 说明 | 帧头 | 帧命令 | 数据长度 | 数据 | 校验 |
| 长度 | 1Byte | 2Byte | 2Byte | (N)Byte | 2Byte |
| 数据 | 0xAA | CMD | FRLEN | DATA | CRC |

响应报文:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 说明 | 帧头 | 帧命令 | 返回码 | 数据长度 | 数据 | 校验 |
| 长度 | 1Byte | 2Byte | 2Byte | 2Byte | (N)Byte | 2Byte |
| 数据 | 0xAA | CMD | RC | FRLEN | DATA | CRC |

注意：

报文中数据长度的取值范围为 0 到 65535， 如果请求报文或者响应报文中没有数据，则数据长度的值为0 。 即当数据长度为0时，该报文中没有数据位， N 等于0。

例：测试通信

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 说明 | 帧头 | 帧命令 | 数据长度 | 数据 | 校验 |
| 长度 | 1Byte | 2Byte | 2Byte | 5Byte | 2Byte |
| 数据 | 0xAA | 0x0100 | 0x0005 | 0x01, 0x23, 0x45, 0x67, 0x89 | 0x0442 |

发送的数据为字符流(HEX)：AA010000050123456789C2C1

该指令格式为完整的数据传输格式， 所有数据必须以以上格式传输，否则为无效数据。

所有命令不可重复使用。

## 通信命令分配表

|  |  |  |
| --- | --- | --- |
| 命令模块 | 命令码(CMD) | 说明 |
| 预留命令  0x0000 |  | 暂时保留 |
| 系统模块  (0x0001, 0x01ff) |  | 用于系统控制的指令，内部使用，不对外开放 |
| 电机模块  (0x0200, 0x02ff) |  | 电机相关指令 |
| 输入输出模块  (0x0300, 0x03ff) |  | 输入输出指令 |
| 文件模块  (0x0400, 0x04ff) |  | 文件操作相关指令 |
| ... |  | ... |
| 应用模块  (0xA000, 0xC000) |  | 应用程序命令段 |

# 用户命令 (0xA000, 0xC000)

用户命令用于不同工程项目各自添加自己的通信命令, 仅仅对本身的应用项目有效.不同的应用项目新增的命令可以重复.

## 煲仔炉项目模块

### 获取煲仔炉单个板卡的状态信息

说明：获取煲仔炉板卡的状态信息

请求：(上位机)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 域 | 字节 | | 备注 | |
| 包头 | 1 | | 0xAA | |
| 包命令 | 2 | | 0xA000 | |
| 数据长度 | 2 | | 0x000E+X | |
| 数据内容 | 14+X | 2 | 炉头板卡当前的工艺状态信息（默认第一次发送值为0） | Bit0:煲头在线 |
| Bit1:煲头就绪 |
| Bit2:工作中 |
| Bit3:请求放煲 |
| Bit4:请求加油 |
| Bit5:请求取煲 |
| Bit6:炉头屏蔽 |
| Bit7:手动模式 |
| Bit8~Bit11:工作阶段 |
| Bit12~Bit15:报警信息 |
| 2 | 炉头板卡当前的工艺某状态的运行时间（单位秒） | 默认第一次发送值为0 |
| 2 | 炉头板卡当前烹饪剩余时间信息（单位秒） | 默认第一次发送值为0 |
| 4 | 炉头板卡菜谱ID | 默认第一次发送值为0 |
| 2 | 炉头板卡烹饪工艺步骤 | 默认第一次发送值为0 |
| 2 | 获取菜谱状态（0：空闲 1：正在获取 2：获取成功 3：获取失败） | 默认第一次发送值为0 |
| x | 菜单具体类容 | （当获取菜谱状态为2是才需要转发） |
| CRC校验 | 2 | | 除去包头的其它所有数据进行CRC16校验运算。 | |

应答：(下位机)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 域 | 字节 | | 备注 | |
| 包头 | 1 | | 0xAA | |
| 包命令 | 2 | | 0xA000 | |
| 返回码 | 2 | | RC | |
| 数据长度 | 2 | | 0x18 | |
| 数据内容 | **24** | 2 | 炉头板卡看门狗复位标志 | 0000表示复位了 |
| 2 | 炉头板卡当前的工艺状态信息 | Bit0:煲头在线 |
| Bit1:煲头就绪 |
| Bit2:工作中 |
| Bit3:请求放煲 |
| Bit4:请求加油 |
| Bit5:请求取煲 |
| Bit6:炉头屏蔽 |
| Bit7:手动模式 |
| Bit8~Bit11:工作阶段 |
| Bit12~Bit15:报警信息 |
| 4 | 炉头板卡当前菜谱id | 例如：0x0000000001----1号菜谱 |
| 2 | 炉头板卡烹饪工艺步骤 | 对应菜谱的流程序号  0-初始化 1-预热 2-预热超时时间 |
| **2** | 炉头板卡当前设置煮饭工艺命令 | Bit0:保留 |
| Bit1:煮饭需求 |
| Bit2:放煲完成 |
| Bit3:加油完成 |
| Bit4:取煲完成 |
| Bit5:保留 |
| Bit6:报警复位 |
| Bit7：流程复位 |
| Bit8：手动下单 |
| Bit9：屏蔽炉头 |
| Bit10~Bit15:预留信号 |
| 2 | 炉头板卡锅底实时温度值 | 举例：0x00 0xF0----代表温度值是：240摄氏度 |
| 2 | 炉头板卡当前某状态运行时间 | 举例：  0x01 0xE0表示(0x01 << 8) + 0xE0 = 480 秒（单位：秒） |
| 2 | 炉头板卡当前烹饪剩余时间 | 举例：  0x01 0xE0表示(0x01 << 8) + 0xE0 = 480 秒（单位：秒） |
| 2 | 炉头板卡离线标志 | 0001离线 |
| 2 | 炉头板卡人工复位标志 | 0001复位 |
| 2 | 炉头板卡离线时间 | 0001:表示1s |
| CRC校验 | 2 | | 除去包头的其它所有数据进行CRC16校验运算。 | |

### 设置煲仔炉板卡的煮饭流程

说明：设置煲仔炉板卡的煮饭流程

请求：(上位机)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 域 | 字节 | | 备注 | |
| 包头 | 1 | | 0xAA | |
| 包命令 | 2 | | 0xA001 | |
| 数据长度 | 2 | | 0x0002 | |
| 数据内容 | 2 | 2 | 设置当前煮饭工艺命令值 | Bit0:保留 |
| Bit1:煮饭需求 |
| Bit2:放煲完成 |
| Bit3:加油完成 |
| Bit4:取煲完成 |
| Bit5:保留 |
| Bit6:报警复位 |
| Bit7：流程复位 |
| Bit8：手动模式 |
| Bit9：屏蔽炉头 |
| Bit10~Bit15:预留信号 |
| CRC校验 | 2 | | 除去包头的其它所有数据进行CRC16校验运算。 | |

应答：(下位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA001 |
| 返回码 | 2 | RC |
| 数据长度 | 2 | 0 |
| 数据内容 | 0 | 无 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

### 查询当前菜单内容

说明：查询当前下位机的菜单内容

请求：(上位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA002 |
| 数据长度 | 2 | 0x0002 |
| 数据内容 | 4 | 菜单编号 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

应答：(下位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA002 |
| 返回码 | 2 | RC |
| 数据长度 | 2 | 菜单内容长度 |
| 数据内容 | N | 菜单具体类容 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

### 从菜单服务器更新菜单

说明：让下位机主动从菜单服务器去更新菜单

请求：(上位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA003 |
| 数据长度 | 2 | 0x0002 |
| 数据内容 | 4 | 菜单编号 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

应答：(下位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA003 |
| 返回码 | 2 | RC |
| 数据长度 | 2 |  |
| 数据内容 | 0 |  |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

### 推送新的菜单内容

说明：上位机推送菜单给下位机

请求：(上位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA004 |
| 数据长度 | 2 | 菜单数据长度 |
| 数据内容 | N | 菜单内容 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

应答：(下位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA004 |
| 返回码 | 2 | RC |
| 数据长度 | 2 | 解析成功后的菜单内容长度 |
| 数据内容 | N | 解析成功后的菜单内容 |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

### 上位机获取煲仔炉所有板卡参数

说明：获取煲仔炉所有板卡参数

请求：(上位机)

|  |  |  |
| --- | --- | --- |
| 域 | 字节 | 备注 |
| 包头 | 1 | 0xAA |
| 包命令 | 2 | 0xA005 |
| 数据长度 | 2 | 0x0002 |
| 数据内容 | 2 | 0x000X：表示随机ID |
| CRC校验 | 2 | 除去包头的其它所有数据进行CRC16校验运算。 |

应答：(下位机)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 域 | 字节 | | 备注 | |
| 包头 | 1 | | 0xAA | |
| 包命令 | 2 | | 0xA005 | |
| 返回码 | 2 | | RC | |
| 数据长度 | 2 | | 2 \* X + 4 | |
| 数据内容 | N | 2 | 随机ID | 0x000X（与请求的随机ID相同） |
| 2 | 炉头总数 | x |
| 2 \* x | 炉头当前状态信息 | Bit0:保留 |
| Bit1:煮饭需求 |
| Bit2:放煲完成 |
| Bit3:加油完成 |
| Bit4:取煲完成 |
| Bit5:保留 |
| Bit6:报警复位 |
| Bit7：流程复位 |
| Bit8：手动下单 |
| Bit9：屏蔽炉头 |
| Bit10~Bit15:预留信号 |
| 2 \* x | 炉头烹饪剩余时间 | 1号炉头烹饪剩余时用两个字节表示，例如：0x01 0xE0  表示(0x01 << 8) + 0xE0 = 480 秒（单位：秒）  如果没有烹饪则为0x00 0x00 |
| 2 | 炉头板卡锅底实时温度值 | 举例：0x00 0xF0----代表温度值是：240摄氏度 |
| 4 | 菜单编号 | 0X00000001---代表1号菜谱 |
| CRC校验 | 2 | | 除去包头的其它所有数据进行CRC16校验运算。 | |

# 返回码参数表

# 设备信息表

|  |  |
| --- | --- |
| 项目说明 | 项目唯一编码 |
| 板卡MS\_TWO V0.0 双层锅项目 | 0x01010001 |
| 板卡MS\_TWO V0.1 双层锅项目 | 0x01010101 |
| 板卡MS\_TWO V0.1 串口屏设置IP项目 | 0x01010104 |
| 板卡MS\_TWO V0.1 转轨项目 | 0x01010102 |
| 板卡MS\_TWO V0.1 下菜口项目 | 0x01010105 |
| 板卡JZ\_STMF407工控板AGV 项目 | 0x02020103 |
| 板卡PIZZA V0.0 pizza项目 | 0x03010006 |
| 板卡QIANXI\_AGV V0.0 AGV 项目 | 0x04010003 |

# CRC校验算法示例

/\*\* CRC-16-ANSI table values \*/

#defineCOMMAND\_CRC16\_TABLE { \

0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241, \

0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440, \

0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40, \

0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841, \

0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40, \

0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41, \

0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641, \

0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040, \

0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240, \

0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441, \

0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41, \

0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840, \

0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41, \

0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40, \

0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640, \

0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041, \

0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240, \

0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441, \

0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41, \

0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840, \

0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41, \

0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40, \

0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640, \

0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041, \

0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241, \

0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440, \

0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40, \

0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841, \

0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40, \

0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41, \

0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641, \

0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040 \

}

/\*\* CRC-16-ANSI table \*/

staticunsignedshort \_crc16\_table[] = COMMAND\_CRC16\_TABLE;

/\*\*

\* Calculate a CRC-16-ANSI checksum from data in the buffer and crc specified.

\* @param p Buffer

\* @param crc CRC base.

\* @param len Length

\* @return CRC calculated

\*/

unsignedshortcrc16(void \*p, unsignedshortcrc, unsignedintlen) {

unsignedchar\* ptemp = (unsignedchar \*)p;

inti = 0;

for (i = 0; i<len; i++) {

crc = (((crc / 256 ) &0xff) ^ \_crc16\_table[(crc ^ (ptemp[i])) &0xff]) &0x0000ffff;

}

return (crc&0xFFFF);

}