



WIFI脂肪秤通信协议

——MCU与CSM64F02串口协议

V0.11

芯海科技（深圳）股份有限公司

版权所有不得复制

修订记录

日期	修订版本	描述	修改人
2017-01-06	V0.0	初稿	陈华辉

目录

1	硬件接口	4
2	通信格式	5
3	通信协议(SOC->WIFI)	6
3.1	发送重量数据	6
3.2	进入 SmartConfig 模式	9
4	通信协议(WIFI->SOC)	10
4.1	设置 RTC 参数	10
4.2	设置 UNIT 参数	11
4.3	发送 MCU 升级信息	11
4.4	MCU 请求数据包	11
5	测试模式	14
5.1	测试 WIFI	14
6	通信异常处理	15
6.1	UART 超时重发	15
6.2	UART 接收错误重发	15
7	通信流程	16
7.1	Master 主流程图	16
7.2	Master 与 Slave 交互流程	16

1 硬件接口

UART 通讯以 SOC 为主机，WIFI 为从机。

通信速率暂定 9600bps。

^{注1}：SOC 型号为 CSU18M8x 系列。

^{注2}：WIFI 型号为 CSM64F02。

系统仅在称重模式使能 WIFI 模块的电源，其它模式（标定、设置等）则主动关闭 WIFI。低电压状态不开启 WIFI。

整机测试时也需要打开 wifi 的电源，wifi 需要扫描周围 AP，确定 wifi 是否 OK。需要添加测试 wifi 的指令，设备标定时发送该指令到 wifi，之后 wifi 开始扫描周围 AP，成功之后应答 SOC 成功，扫描超时时应答 SOC 失败。

2 通信格式

(1) 发送(Transmit)

B0	B1	B2	B3 ~ B _{N-1}	B _N
帧头	长度	命令	数据	校验
1、帧头：固定值 0xC5 2、长度：表示命令与数据的长度(N-2) 3、命令：表示发送指令的用途 4、数据：具体通信的内容 5、校验：B0 ~ B _{N-1} 的异或值				

(2) 应答(Response)

B0	B1	B2	B3 ~ B _{N-1}	B _N
帧头	长度	命令	数据	校验
1、帧头：固定值 0xC6 2、长度：表示命令与数据的长度(N-2) 3、命令：与发送指令一致 4、数据：具体通信的内容 5、校验：B0 ~ B _{N-1} 的异或值				

接收错误应答(Receive Error)

B0	B1	B2	B3	B4
帧头	长度	命令	EVENT	校验
0xC6	0x02	0xEE	0x--	0x--
1、帧头：固定值 0xC6 2、长度：数据长度 0x02 3、命令：接收错误命令 0xEE 4、 EVENT ：错误原因。具体含义如下所示： 0x10 → 命令错误 0x11 → 参数越界 0x12 → 校验码错误 0x13 → 指令运行错误；如系统未初始化完等 Other → 保留 5、校验：B0 ~ B3 的异或值				

3 通信协议(SOC->WIFI)

3.1 发送重量数据

(1) 发送: SOC → WIFI

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20~B31	B32
帧头	长度	命令	重量值		电压值		时间戳							SOC 状态	MCU 软件版本	产品硬件版本		测试标志	测试结果	校验	
0x C5	0x1 E	0x1 0	Wg tH	Wg tL	Vol tH	Vol tL	Yea r	Mo n	Day	Ho ur	Mi n	Sec	We ek	STATU S	SW H	SW L	HW H	HW L	TS_ F	Res Val	0x--

注¹: SOC 发送重量数据的频率为 7.5Hz。

注²: 重量值采用十六进制，数据长度为 2Bytes。为了便于计算人体成分，重量值统一采用 kg 单位。重量值放大 100 倍。

举例: 120.00kg 表示为 0x2EE0, 即 WgtH=0x2E、WgtL=0xE0
51.05kg 表示为 0x13F1, 即 WgtH=0x13、WgtL=0xF1

WgtH: 重量值高 8 位

WgtL: 重量值低 8 位

注³: 电压值采用十六进制，数据长度为 2Bytes。电压值为锂电池电压（放大 100 倍）。

举例: 4.00V 表示为 0x0190, 即 VoltH=0x01、VoltL=0x90
3.67V 表示为 0x016F, 即 VoltH=0x01、VoltL=0x6F

VoltH: 电压值高 8 位

VoltL: 电压值低 8 位

注⁴: 时间戳统一采用 24 小时制，所有时间和日期信息由 BCD 码表示，举例: Year = 0x17 表示 2017 年，Year = 0x20 表示 2020 年。

Year: 年 (0x00 ~ 0x99)

Mon: 月 (0x01 ~ 0x12)

Day: 日 (0x01 ~ 0x31)

Hour: 时 (0x00 ~ 0x23)

Min: 分 (0x00 ~ 0x59)

Sec: 秒 (0x00 ~ 0x59)

Week: 星期 (0~6)，其中 0 表示星期日，1~6 分别表示星期一~星期六

注⁵: SOC 状态为系统主控的状态标志

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	DC	FP	LVD	UT	--	OTA	RE	VD

Bit7 **DC**: 外部电源接入标志

0 = 系统使用电池供电

1 = 系统使用外部电源供电

Bit6 **FP**: 电池充满标志（注：DC 标志为 0 时，该标志无效）

0 = 电池正在充电

1 = 电池已经充满

Bit5 **LVD**: 电量状态

0 = 电量正常

1 = 电量不足（注：系统尚未显示“Lo”，用于 APP 提示用户需要充电）

Bit4 **UT**: 重量单位

0 = 公斤（kg）

1 = 英磅（lb）

Bit3: 保留

Bit2 **OTA**: 固件支持 OTA 的标志

0 = 该固件不支持 OTA

1 = 该固件支持 OTA

Bit1 **RE**: 阻抗测量

0 = 阻抗测量关闭

1 = 阻抗测量使能，1258 开始测量以这个标志位为准

Bit0 **VD**: 有效重量数据

0 = 临时重量数据

1 = 有效重量数据（锁定重量值）

注⁶: WIFI 未连接状态只有接收到有效数据（重量+电压+时间戳+状态）才进入离线存储操作。

SW_H、SW_L: 从 1 开始，不支持 OTA 的软件版本号默认为 0;

HW_H、HW_L: 从 1 开始，该项不能被 OTA，硬件确定之后，该项不变。

TS_F: 整机测试的完成标志;

注⁷: 测脂结果字段共占 12 字节，依次为

Z34_H, Z34_L, Z12_H, Z12_L, Z13_H, Z13_L, Z14_H, Z14_L, Z23_H, Z23_L, Z24_H, Z24_L,

当 Z34_H = 0xFF, Z34_L=0xFF 时，表示由 WIFI 模式负责测脂，否则表示由 MCU 负责测脂，并以该 12 字节表示测脂结果，其中当为 4 电极时，Z12_H, Z12_L, Z13_H, Z13_L, Z14_H, Z14_L, Z23_H, Z23_L, Z24_H, Z24_L 均为 0x00

(2) 应答: WIFI → SOC

B0	B1	B2	B3	B4	B5	B6
帧头	长度	命令	WIFI 状态			校验

0xC6	0x04	0x10	DEV Status	SYS Status	00	0x--
------	------	------	------------	------------	----	------

注¹: WIFI 状态为 WIFI 模块的状态标志

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DEV Status								
System Status								
00								

[DEV Status](8bits)

- bit0: 0: 四电极 1: 8 电极 //默认是 4 电极,
- bit1: 0: 未测试或正在测试 CS1258 1: 测试 CS1258 完成 //默认未测试
- bit2: 0: 未连接或正在连接服务器 1: 连接服务器 OK //为 1wifi 指示灯常亮, 否则闪烁
- bit3: 0: smartconfig 未进行或正在进行 1: smartconfig 配网完成 //默认 SmartConfig 未进行
- bit4: 0: 未配网,首次使用或恢复出厂设置
1: 已配网, 上电后会自动连接路由器、服务器
- bit5: 0: 未接收到有效重量、正在上传重量信息到服务器、正在保存重量信息到 flash。
1: 上传数据完毕或保存离线数据完毕, 可以关闭电源
- 注意: 如果接收不到有效重量, 该 bit 会一直为 1**
- Bit6: 0: 正常工作模式 1: PCBA 或整机测试模式
- bit7: 0: 未进行 OTA 升级或升级完成 1: 正在 OTA 升级

[SYSStatus]

- Bit0: 0: 未进行加热 1: 正在加热
- Bit1: 0: 硬件未解密 1: 硬件已解密
- Bit2: 0: PCBA 测试模式 1: 整机测试模式 //DEV Status.bit6 为 1 时检测该标志
- Bit3: 0: wifi 正在测试或未测试 1: wifi 测试完成
- Bit4: 0: wifi 测试未通过 1: wifi 测试通过// SYSStatus.bit3 为 1 时检测该标志
- Bit5: 0: 1258 正在测试或未测试 1: 1258 测试完成
- Bit6: 0: 1258 测试未通过 1: 1258 测试通过// SYSStatus.bit5 为 1 时检测该标志
- Other: 保留

[00]

预留

注²: WIFI 连接超时后, 同时把 SOC 发送的有效数据与人体阻抗保存在数据缓冲区。如果 SOC 没有发送有效数据 (例如称重不稳定, 等等), 则 WIFI 无需保存当前的数据。

3.2 进入 SmartConfig 模式

(1) 发送: SOC → WIFI

B0	B1	B2	B3
帧头	长度	命令	校验
0xC5	0x01	0x12	0x--

(2) 应答: WIFI → SOC

B0	B1	B2	B3
帧头	长度	命令	校验
0xC6	0x01	0x12	0x--

注¹: (前提: WIFI 连接服务器成功) Master 发送该命令使 WIFI 进入 Standby 模式 (保持连接路由器, 系统进入低功耗模式, 以节省功耗), 当 Master 需要再次发送数据时, 先发送字符串 (例如 0x000000) 来唤醒 WIFI 模块, 然后再正常通信。

4 通信协议(WIFI->SOC)

4.1 设置 RTC 参数

(1) 发送: WIFI→SOC

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
帧头	长度	命令	时间戳							校验
0xC5	0x08	0xA0	Year	Mon	Day	Hour	Min	Sec	Week	0x--

注¹: 时间戳统一采用 24 小时制, 所有时间和日期信息由 BCD 码表示,, 举例: Year = 0x17 表示 2017 年, Year = 0x20 表示 2020 年。

Year: 年 (0x00 ~ 0x99)

Mon: 月 (0x01 ~ 0x12)

Day: 日 (0x01 ~ 0x31)

Hour: 时 (0x00 ~ 0x23)

Min: 分 (0x00 ~ 0x59)

Sec: 秒 (0x00 ~ 0x59)

Week: 星期 (0~6), 其中 0 表示星期日, 1~6 分别表示星期一~星期六

注²: 每次 WIFI 连接 (或绑定) 都同步 1 次时间戳。

(2) 应答: SOC→WIFI

B0	B1	B2	B3
帧头	长度	命令	校验
0xC6	0x01	0xA0	0x67

4.2 设置 UNIT 参数

(1) 发送: WIFI→SOC

B0	B1	B2	B3	B4
帧头	长度	命令	单位	校验
0xC5	0x02	0xA1	UNIT	0x--

注¹:

UNIT = 0x00: 公斤 (kg)

UNIT = 0x01: 英磅 (lb)

UNIT = Other: 保留

注²: 为了便于计算人体成分, SOC 发送的重量值统一采用 kg 单位。

(2) 应答: SOC→WIFI

B0	B1	B2	B3
帧头	长度	命令	校验
0xC6	0x01	0xA1	0x66

4.3 发送 MCU 升级信息

(3) 发送: WIFI→SOC

B0	B1	B2	B3	B4	B5	B6
帧头	长度	命令	Bin 文件大小		数据包个数	校验
0xC5	0x--	0xA2	bin_h	bin_L	package_num	0x--

注¹:

bin_h、bin_L: bin 文件的大小, 以字节为单位;

package_num: 分割数据包的个数, 以 100 字节为单位分割;

(4) 应答: SOC→WIFI

B0	B1	B2	B3	B4	B5	B6
帧头	长度	命令	Bin 文件大小		数据包个数	校验
0xC6	0x--	0xA2	bin_h	bin_L	package_num	0x--

4.4 MCU 请求数据包

(5) 发送: SOC→WIFI

B0	B1	B2	B3	B4
帧头	长度	命令	数据包序列号	校验
0xC5	0x--	0xA3	Index	0x--

注¹:

Index: 请求数据包的序列号, 从 0 开始, 到 package_num-1;

(6) 应答: WIFI→SOC

B0	B1	B2	B3	...	B102	B103	B104
帧头	长度	命令	数据			CRC 检验	校验
0xC6	0x--	0xA3	data			CRC	0x--

注¹:

data: 一个数据包的 100 字节数据。

CRC: 100 字节数据的 CRC 检验值;, 检验算法如下。

```
static const unsigned char crc_table[] =
{
```

```
0x00,0x31,0x62,0x53,0xc4,0xf5,0xa6,0x97,0xb9,0x88,0xdb,0xea,0x7d,0x4c,0x1f,0x2e,
0x43,0x72,0x21,0x10,0x87,0xb6,0xe5,0xd4,0xfa,0xcb,0x98,0xa9,0x3e,0x0f,0x5c,0x6d,
0x86,0xb7,0xe4,0xd5,0x42,0x73,0x20,0x11,0x3f,0x0e,0x5d,0x6c,0xfb,0xca,0x99,0xa8,
0xc5,0xf4,0xa7,0x96,0x01,0x30,0x63,0x52,0x7c,0x4d,0x1e,0x2f,0xb8,0x89,0xda,0xeb,
0x3d,0x0c,0x5f,0x6e,0xf9,0xc8,0x9b,0xaa,0x84,0xb5,0xe6,0xd7,0x40,0x71,0x22,0x13,
0x7e,0x4f,0x1c,0x2d,0xba,0x8b,0xd8,0xe9,0xc7,0xf6,0xa5,0x94,0x03,0x32,0x61,0x50,
0xbb,0x8a,0xd9,0xe8,0x7f,0x4e,0x1d,0x2c,0x02,0x33,0x60,0x51,0xc6,0xf7,0xa4,0x95,
0xf8,0xc9,0x9a,0xab,0x3c,0x0d,0x5e,0x6f,0x41,0x70,0x23,0x12,0x85,0xb4,0xe7,0xd6,
0x7a,0x4b,0x18,0x29,0xbe,0x8f,0xdc,0xed,0xc3,0xf2,0xa1,0x90,0x07,0x36,0x65,0x54,
0x39,0x08,0x5b,0x6a,0xfd,0xcc,0x9f,0xae,0x80,0xb1,0xe2,0xd3,0x44,0x75,0x26,0x17,
0xfc,0xcd,0x9e,0xaf,0x38,0x09,0x5a,0x6b,0x45,0x74,0x27,0x16,0x81,0xb0,0xe3,0xd2,
0xbf,0x8e,0xdd,0xec,0x7b,0x4a,0x19,0x28,0x06,0x37,0x64,0x55,0xc2,0xf3,0xa0,0x91,
0x47,0x76,0x25,0x14,0x83,0xb2,0xe1,0xd0,0xfe,0xcf,0x9c,0xad,0x3a,0x0b,0x58,0x69,
0x04,0x35,0x66,0x57,0xc0,0xf1,0xa2,0x93,0xbd,0x8c,0xdf,0xee,0x79,0x48,0x1b,0x2a,
0xc1,0xf0,0xa3,0x92,0x05,0x34,0x67,0x56,0x78,0x49,0x1a,0x2b,0xbc,0x8d,0xde,0xef,
0x82,0xb3,0xe0,0xd1,0x46,0x77,0x24,0x15,0x3b,0x0a,0x59,0x68,0xff,0xce,0x9d,0xac
};
```

```
/**
 * 计算 CRC16 校验
 * @param ptr      需要计算的数组
 * @param len      长度
 * @param old_crc  之前的校验值
 * @return CRC8    校验值
 */
unsigned char calcCrc8_base(unsigned char *ptr,unsigned int len,unsigned char
old_crc)
{
```

```
    unsigned char  crc = old_crc;
    while (len--)
    {
        crc = crc_table[crc ^ *ptr++];
    }
    return (crc);
}

/**
 * 计算 CRC16 校验
 * @param data    需要计算的数组
 * @param offset  起始位置
 * @param len      长度
 * @return  CRC16 校验值
 */
int calcCrc8(char *data, int len)
{
    return calcCrc8_base(data, len, 0x00);
}

int main(int argc, char *argv[]) {
    char testData[] = {
        0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,
        0x00,0x00,0x00,0x00,0x00,0x00,0x5C,0xCF,0x7F,0x01,
        0x78,0xF2,0x00,0x00,0x40,0x10,0xC0,0xA8,0x09,0x64,
        0x1D,0x87,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
        0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,
        0x00,0x00,0x00,0x00,0x00,0x00,0x5C,0xCF,0x7F,0x01,
        0x78,0xF2,0x00,0x00,0x40,0x10,0xC0,0xA8,0x09,0x64,
        0x1D,0x87,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
        0x78,0xF2,0x00,0x00,0x40,0x10,0xC0,0xA8,0x09,0x64,
        0x1D,0x87,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    };
    printf("testDataLen:%d\r\n",sizeof(testData));
    int crc = calcCrc8(testData,sizeof(testData));
    printf("0x%02x\r\n",crc);
    getchar();
}
```

5 测试模式

5.1 测试 WIFI

(1) 发送: WIFI → SOC

B0	B1	B2	B3
帧头	长度	命令	校验
0xC5	0x01	0xCC	0x08

(2) 应答: SOC → WIFI

B0	B1	B2	B4
帧头	长度	命令	校验
0xC6	0x02	0xCC	0x--

注¹: 当 WIFI 进入测试模式, 发送该命令给 MCU, MCU 应答对应指令, 用于检测 UART 焊接是否正常。

6 通信异常处理

6.1 UART 超时重发

如果 **Transmit** 超过规定时间（例如 100ms）没有接收到 **Response** 返回的数据则重发 1 遍上次的数据。
Response 接收超时则丢弃该指令。

6.2 UART 接收错误重发

无论 Master 或 Slave 接收到对方返回的 **Receive Error**（数据帧格式见 PAGE 5——接收错误应答），则重发 1 遍上次的数据，如果超过 1 次则表示该指令无效。

7 通信流程

7.1 Master 主流程图

7.2 Master 与 Slave 交互流程

