

版本	V1.0
日期	2024.8.1

古瑞瓦特固件升级协议（对外）



版权所有©深圳古瑞瓦特新能源有限公司 2024 保留一切权利

未经深圳古瑞瓦特新能源有限公司书面许可，不得以任何形式或手段如电子、机械、拍照、影印或其他方式复制、存储或传输本文件的任何部分。

深圳古瑞瓦特新能源有限公司

地址：深圳市宝安区西乡街道固兴社区航城大道中德欧产业示范园 A 栋 A401，4-13 楼

电话：400-931-9122

网址：www.growatt.com | service@ginverter.com

变更记录

项次	版次	变更内容	制定	制定日期
00	V1.0	首版制定	梁桂连	2024/8/1

目录

古瑞瓦特固件升级协议（对外）	1
1. 简介.....	5
1.1. 说明	5
1.2. 范围	5
1.3. 术语与缩略语.....	5
2. 串口升级协议.....	6
2.1. 串口总线规格.....	6
2.2. 数据模式	6
2.3. 通信模式	6
2.4. 名词解释	6
2.5. 外部串口升级协议	7
2.5.1. 外部串口升级协议规范	7
2.5.2. 外部串口升级步骤介绍	8
3. 附录.....	11
3.1. CRC16 计算方法.....	11
3.2. CRC32 计算方法.....	12

1. 简介

1.1. 说明

编写本文档的目的在于详细地说明古瑞瓦特公司固件对外升级协议。

1.2. 范围

适用于采集器与电池/逆变器/储能机的升级过程。

1.3. 术语与缩略语

术语/缩略语	英文	中文解释
BMS	Battery Management System	电池管理系统
CAN	Controller Area Network	控制器局域网络
SPI	Serial Peripheral Interface	外设串行接口
MCU	Microprogrammed Control Unit	微型控制器
FLASH	Flash	闪存
BOOT	BOOT	启动
APP	application program	应用程序
RS232	recommend standard 232	串行物理接口标准
RS485	recommend standard 485	串行物理接口标准

2. 串口升级协议

2.1. 串口总线规格

总线传输波特率：38400 bps，硬件对外接口为 RS485。

2.2. 数据模式

大端方式，数据高字节存放地址低字节。协议中会用到以下数据类型，定义如下：

表 2-1

序号	数据类型	类型定义	数据长度（字节）
1	Byte	unsigned char	1
2	Uint16	unsigned short int	2
3	Uint32	unsigned int	4
4	Sint8	signed char	1
5	Sint16	signed short int	2
6	Sint32	signed int	4
7	FP32	float	4

2.3. 通信模式

主机设备发送查询指令或者控制指令帧后，从机设备响应数据。

2.4. 名词解释

表 2-2

名词	解释说明
Modbus 协议	通用工业标准, 是应用于电子控制器上的一种通用协议
Address	地址，用于区分不同设备的标识
Fuction	功能，通过此码区分不同的功能作用，例如读，写等
CRC	循环冗余校验(Cyclical Redundancy Check)
波特率	表示每秒钟传送的码元符号的个数，它是对符号传输速率的一种度量，它用单位时间内载波调制状态改变的次数来表示，1 波特即指每秒传输 1 个符号
USART	通用同步异步收发器(Universal Synchronous Asynchronous Receiver Transmitter)
SCI	串行通信接口，是串行通信技术的一种总称，SCI 模块用于串行通讯，如 RS422、RS485、RS232；在 SCI 中，通信协议体现在 SCI 的数据格式上。通常将 SCI 的数据格式称之为可编程的数据格式

2.5. 外部串口升级协议

2.5.1. 外部串口升级协议规范

2.5.1.1. 格式内容

参照 Modbus 协议，格式如下：

Address	Fuction	Data	CRC Check
1 Byte	1 or 2Bytes	N *1 Byte	2 Bytes

命令及应答格式定义：“从机地址 命令码 寄存器 长度 数据 校验位 L 校验位 H”；Address 默认为 0x01。

2.5.1.2. 命令码说明

表 2-3 升级流程命令码

序号	命令码 (0x)	子命令码 (0x)	地址	说明
1	0x06	Null	0x0064	主机发送从机进入升级模式指令
2	0x15	0x22	Null	主机发送升级文件长度指令
3	0x15	0x23	Null	主机发送升级文件 CRC 校验值指令
4	0x15	0x24	Null	主机发送从机外部 Flash 擦除指令
5	0x15	0x25	Null	主机发送升级文件数据指令
6	0x15	0x26	Null	主机发送升级文件数据传输完成指令
7	0x03	Null	0x0064	进度查询指令

表 2-4 从机回应失败命令码

序号	异常码	说明
1	0x01	写入未定义功能码
2	0x02	读取或写入地址错误
3	0x03	写入数据超过规定范围
4	0x04	CRC 校验错误
5	0x05	设备地址错误

2.5.1.3. 升级流程适用范围

- (1) 采集器与电池/逆变器/储能机的升级过程；
- (2) 上位机与 STM32 烧录文件传输协议；
- (3) 适用机型：带有硬件串口对外接口并且带有升级功能的所有机型。

2.5.2. 外部串口升级步骤介绍

2.5.2.1. 主机发送从机进入升级模式指令

数据格式：从机地址(1Byte)+命令码(1Byte)+寄存器(2Bytes)+数据(2Bytes)+校验位(2Bytes)

数据说明：向 0x0064 号保持寄存器写入文件类型

0x0064：升级文件是.bin 文件

0x0065：升级文件是.hex 文件

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x01	0x06	0x00	0x64	0x00	0x64	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x01	0x06	0x00	0x64	0x00	0x64	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x86	0x01	CRC_L	CRC_H

2.5.2.2. 主机发送升级文件长度指令

数据格式：从机地址(1 Byte)+命令码(1 Byte)+子命令码(1 Byte)+数据长度(2 Bytes)+文件长度(4 Bytes)+校验位(2 Bytes)

数据说明：文件长度，以字节(Byte)为单位(M3 文件默认是 256KB，可支持 512KB 等)

例如： 128K 数据为 0x00, 0x02, 0x00, 0x00

256K 数据为 0x00, 0x04, 0x00, 0x00

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
0x01	0x15	0x22	0x00	0x04	0x00	0x04	0x00	0x00	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x22	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x95	0x01	CRC_L	CRC_H

2.5.2.3. 主机发送升级文件长度 CRC 校验值指令

数据格式：从机地址(1Byte)+命令码(1Byte)+子命令码(1Byte)+数据长度(2Bytes)+文件 CRC 校验(4Bytes)+校验位(2Bytes)

数据说明：文件 CRC 校验 = 整个文件 CRC32 位校验值, CRC 校验详见附录

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
0x01	0x15	0x23	0x00	0x04	0x00	0x00	0x00	0x00	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x23	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x95	0x01	CRC_L	CRC_H

2.5.2.4. 主机发送从机外部 Flash 擦除指令

数据格式：从机地址(1Byte)+命令码(1Byte)+子命令码(1Byte)+数据长度(2Bytes)+数据(2Bytes) +校验位(2Bytes)

数据说明：默认写入 0x0000

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x24	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x24	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x95	0x01	CRC_L	CRC_H

2.5.2.5. 主机发送升级文件数据指令

数据格式：从机地址(1Byte)+命令码(1Byte)+子命令码(1Byte)+数据长度(2Bytes)+帧编号 (2Bytes)+数据 (1~256Bytes)+校验位(2Bytes)

数据说明：数据长度 = 帧编号 + 数据 (字节的长度，默认是 0x0102)

帧编号 = 当前升级数据帧号

数据 = 发送 1~N 字节数据到从机(默认是 256Bytes, 可支持 1024Bytes)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7+N	Byte8+N	Byte9+N
0x01	0x15	0x25	0x01	0x02	0x00	0x00	Byte0...ByteN	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x25	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x95	0x01	CRC_L	CRC_H

2.5.2.6. 主机发送升级文件数据传输完成指令

数据格式：从机地址(1Byte)+命令码(1Byte)+子命令码(1Byte)+数据长度(2Bytes)+数据 (2Bytes)+校验位(2Bytes)

数据说明：默认为 0x0000, 表示升级文件发送完成

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x26	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(成功)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x01	0x15	0x26	0x00	0x02	0x00	0x00	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x95	0x01	CRC_L	CRC_H

2.5.2.7. 主机发送升级进度查询指令（主机主动发，1次/2S）

数据格式：从机地址(1Byte)+命令码(1Byte)+寄存器(2Bytes)+数据(2Bytes)+校验位(2Bytes)

数据说明：默认写入 0x0001，用于查询从机当前的升级进度，获取进度用于主机显示

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x01	0x03	0x00	0x64	0x00	0x01	CRC_L	CRC_H

从机回应(成功)（注：该示例为回复 95%，即 0x5F 的进度）

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
0x01	0x03	0x02	0x00	0x5F	CRC_L	CRC_H

从机回应(失败)

Byte0	Byte1	Byte2	Byte3	Byte4
0x01	0x83	0x01	CRC_L	CRC_H

2.5.2.8. 注意说明

(1) 操作异常处理：

获得“异常/擦数异常”后，重新发送上一组数据，连续发送 5 次后，提示烧录失败；

发送数据后没有得到正确回应，等待 5S 后，重新发送一次；连续发送 5 次后，提示烧录失败。

(2) 烧录成功判断标志：

烧录百分比等于 100%表示烧录成功；百分比大于 100%表示烧录失败（提示烧录失败，并将百分比作为提示信息传递）；百分比小于 100%表示正在烧录。

3.1.CRC16 计算方法

3.1.CRC16 计算方法

11

```

0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
0x41, 0x81, 0x80, 0x40
};

/*
函数名: usMBCRC16
形参: pucFrame: 计算缓存数据, usLen:计算长度
返回值: CRC 计算值
*/
Uint16 usMBCRC16(Uint8 * pucFrame, Uint16 usLen)
{
    Uint16 ucCRCHi = 0xFF;
    Uint8   ucCRCLo = 0xFF;
    Uint16 iIndex = 0;

    if((NULL == pucFrame)|| (usLen == 0))
    {
        return 0;
    }

    while(usLen--)
    {
        iIndex = (Uint8)(ucCRCLo ^ * pucFrame++);
        ucCRCLo = (Uint8)(ucCRCHi ^ aucCRCHi[iIndex]);
        ucCRCHi = aucCRCLo[iIndex];
    }
    return ((ucCRCHi << 8) | (Uint16)ucCRCLo);
}

```

3.2.CRC32 计算方法

```

/*
函数名: FuncFileCRC32CheckFlash_Cycle
形参: udFrameSerial_s 表示数据帧编号, udCrc_s 表示 CRC 校验值, pbBuffer_s 数据,
udSize_s 表示数据长度
返回值: CRC 计算值

```

*/

UInt32 FuncFileCRC32CheckFlash_Cycle(UInt32 udFrameSerial_s, UInt32 udCrc_s, UInt8
*pbBuffer_s, UInt32 udSize_s)

```
{
    UInt32 udCrcTmp = 0;
    static UInt32 sudFrameSerialTmp = 0;

    if((NULL == pbBuffer_s) || (0 == udSize_s) || (udSize_s > 1024))
    {
        return cFalse;
    }

    if((0 == udFrameSerial_s) || (udFrameSerial_s == (sudFrameSerialTmp + 1)))
    {
        sudFrameSerialTmp = udFrameSerial_s;
        if(0 == udFrameSerial_s)
        {
            udCrcTmp = 0xFFFFFFFF;
            FuncCrc32TableCalc();
        }
        else
        {
            udCrcTmp = udCrc_s;
        }
        udCrcTmp = udFuncFileCrc32(udCrcTmp, pbBuffer_s, udSize_s);
    }
    else
    {
        udCrcTmp = udCrc_s;
    }
    return udCrcTmp;
}
```

void FuncCrc32TableCalc(void)

```
{
    UInt32 c = 0;
    UInt32 i = 0, j = 0;
    memset(&udCrc32Table, 0, sizeof(udCrc32Table));
    for (i = 0; i < 256; i++)
    {
        c = i;
        for (j = 0; j < 8; j++)
        {
```

```

        if(c & 1)
        {
            c = 0xedb88320L ^ (c >> 1);
        }
        else
        {
            c = c >> 1;
        }
    }
    udCrc32Table[i] = c;
}
}

```

```

UInt32 udFuncFileCrc32(UInt32 crc, UInt8 *buffer, UInt32 size)
{
    UInt32 i = 0;
    UInt32 _crc = crc;

    if((NULL == buffer) || (size == 0))
    {
        return 0;
    }

    for (i = 0; i < size; i++)
    {
        _crc = ((_crc >> 8) & 0x00FFFFFF) ^ udCrc32Table[(UInt8)((_crc ^ buffer[i]) &
0xff)] ;/
    }
    return _crc ;
}

```