68 30 32 38 30 30 30 30 31 69 02 66 00 AB 08 01 F4 00 00 41 81 DC 72 08 01 F5 00 00 40 EA 1C 71 08 01 F6 00 00 40 46 84 BE 08 01 F7 00 00 3C 3D A1 30 08 01 F8 00 00 40 00 00 00 08 01 F9 00 00 41 F2 F6 85 08 01 FA 00 00 41 EE E3 8E 08 01 FB 00 00 3C 6D 09 7B 08 01 FC 00 00 00 00 00 00 08 01 FD 00 00 42 AA 17 B5 08 01 FE 00 00 42 66 5E D0 08 01 FF 00 00 3B BD A1 30 08 02 00 00 00 40 00 00 00 08 02 01 00 00 41 DE F6 85 08 02 02 00 00 41 EA D0 98 08 02 03 00 00 3C 6D 09 7B 08 02 04 00 00 00 00 00 00 08 02 05 00 00 42 AA 47 1D 08 02 06 00 00 42 42 D0 98 0D E4 0D

68 固定帧

00 AB 数据长度 高位在前 低位在后 len=0x00\*256+0xAB;

08 01 F4 00 00 41 81 DC 72 一个数据结构体

08 表示 除了08 这个结构体的长度

01 F4 参数编号

00 表示flaot 型

00 表示小数点位数

41 81 DC 72 表示 真实的数据 （flaot型数据有可能会倒叙）

0D E4 为crc校验位 从0x68 到 0D前面的 98 crc实现为下面的函数：

const u16 code wCRCTalbeAbs[16] =

{

0x0000, 0xCC01, 0xD801, 0x1400, 0xF001, 0x3C00, 0x2800, 0xE401, 0xA001, 0x6C00, 0x7800, 0xB401, 0x5000, 0x9C01, 0x8801, 0x4400,

};

u16 CRC16(u8 \*puchMsg, u16 usDataLen)

{

static u16 wCRC;

static u16 i;

static u8 chChar;

static u8 Chi;

static u8 Cli;

wCRC = 0xFFFF;

i=0;

for (i = 0; i < usDataLen; i++)

{

chChar = \*puchMsg++;

wCRC = wCRCTalbeAbs[(chChar ^ wCRC) & 15] ^ (wCRC >> 4);

wCRC = wCRCTalbeAbs[((chChar >> 4) ^ wCRC) & 15] ^ (wCRC >> 4);

}

Chi=wCRC%256;

Cli=wCRC/256;

wCRC=(Chi<<8)|Cli;

return wCRC;

}