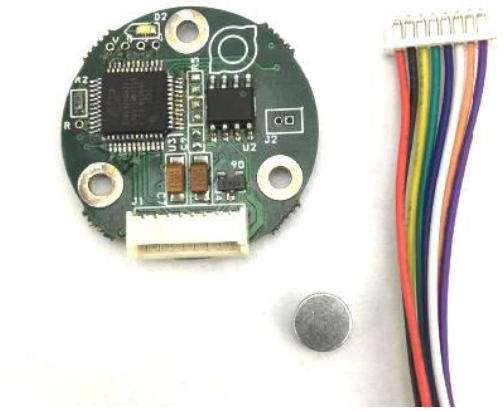
嵌入式绝对值编码器 EA28S06



接口

接口	RS485 / RS422 / SPI
通讯协议	MODBUS RTU/ASCII(RS485) / SSI(RS422)格雷码或二进制码 / SPI
诊断	磁场、内存、接口自诊断
编程功能	节点号，波特率，总圈数，分辨率，预设(原始点)，计数方向，温度
传输速率	RS485: 9600~921600bps RS422: 1Mbps SPI: 4Mbps
数据刷新时间	<10us

输出

输出驱动器	RS422/RS485总线差分

电气数据

主电源电压	5VDC（电源符合EN 50178）
工作电流消耗	< 30mA
后备电池规格	3.7V/500mA可充电锂电池
静止电流消耗	< 10uA （主电源断开且静止状态）
反极性保护	是
短路保护	是
EMC:发射干扰	EN 61000-6-4
EMC:抗干扰	EN 61000-6-2
MTTF	50000小时，在40 ℃下

传感器

技术	磁电
单圈分辨率	Max.15bits
圈数	Max.30bits
多圈技术	电子方式计圈数（后备电池）
精度(INL)	±0.1°
码制	二进制码 / 格雷码

环境规格

工作温度	-40°C～+105°C
存储温度	-40°C～+125°C
湿度	98%相对湿度，无凝结状态

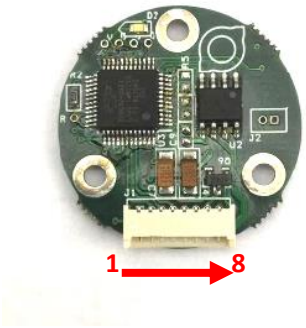
机械数据

轴的类型	分离实心轴套，内嵌ø6x2.5mm磁铁
产品直径	Ø28 mm
产品厚度	3.5 mm
最高机械速度	15000 rpm
重量	3g（不包含磁铁和线）

电气连接

连接方向	轴向或径向

信号芯线定义



引脚号	RS485	SSI	SPI
1	VCC	VCC	VCC
2	NC.	CLOCK+	SETPOS
3	NC.	CLOCK-	VBAT
4	SETPOS	SETPOS	MOSI
5	RS485A	DATA+	MISO
6	RS485B	DATA-	CLOCK
7	VBAT	VBAT	CS
8	GND	GND	GND

SETPOS 是外部复用输入引线，具备两种功能：

1. 恢复出厂默认设置：

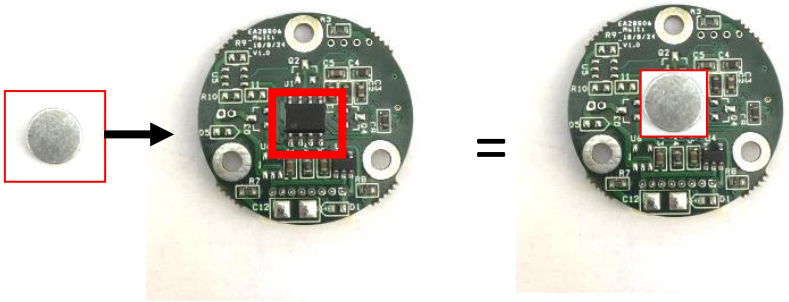
编码器上电前，预先将 SETPOS 与 GND 短接，上电 3 秒后，将 SETPOS 与 GND 断开并悬空。

2.设置原始位置：

编码器通电时，将 SETPOS 与 GND 短接 1 秒后，将 SETPOS 与 GND 断开并悬空。

离轴磁铁安装方式

将磁铁对中(偏差<0.2mm)编码器背面红框内芯片，并位于其上方 1mm(偏差<0.5mm)位置处。



注意：由于编码器上电时会检测磁铁的位置，确认磁场正常后才能进行数据通讯

板载 LED 指示灯：
1. 慢速闪烁：未检测到磁铁正确位置
2. 熄灭：正常待机中
3. 高频亮灯：数据通讯进行中

主动模式只适合单机运行模式
被动模式适合单机/联网模式
出厂默认通讯参数为:
被动等待命令模式
从地址：0x01
数据首地址：41800
波特率：115200/8/N/1

一、实时数据通讯方式（MODBUS RTU 主动/被动模式）

I)编码器位置数据请求命令：

1. 主控端发送 MODBUS RTU 命令帧：
发送数据(HEX)：0x01 0x03 0xA3 0x48 0x00 0x02 0x66 0x59
其中：
0x01: 从设备地址(出厂默认为0x01,可设置范围0x01~0xFE)
0x03: 读寄存器功能
0xA3 0x48: 数据寄存器首地址(出厂默认为41800,可设置范围40000~49999)
0x00 0x02：读取数据字个数(2个16 bits数据，分别对应圈数值和角度值)
0x66 0x59: CRC 校验

2. 主控端接收来自编码器的数据帧：
接收数据(HEX)：0x01 0x03 0x04 0x07 0x08 0x09 0x0A 0xFC 0xD2
其中：
0x01: 从设备地址
0x03: 读寄存器功能
0x04: 寄存器字节数量
0x07 0x08: 圈数值 = 0x0708 = 1800 (max.4095)
0x09 0x0A: 角度分辨率值 = 0x090A = 2314 (max.16383)
0xFC 0xD2: CRC 校验

II)编码器内部温度读出命令：

1. 主控端发送 MODBUS RTU 命令帧：
发送数据(HEX)：0x01 0x03 0xA3 0x4A 0x00 0x01 0x87 0x98
其中：
0x01: 从设备地址
0x03: 读寄存器功能

0xA3 0x4A: 温度值寄存器地址(即数据首地址+2)
0x00 0x01：读取数据字个数(1个16 bits数据，对应温度值)
0x87 0x98: CRC 校验

2. 主控端接收来自编码器的数据帧：
接收数据(HEX)：0x01 0x03 0x02 0x00 0x35 0x78 0x53
其中：
0x01: 从设备地址
0x03: 读寄存器功能
0x02: 寄存器字节数量
0x00 0x35: 温度值 = 0x0035 = 53(℃)
0x78 0x53: CRC 校验

二、实时数据通讯方式（MODBUS ASCII 主动/被动模式）

I)编码器位置数据请求命令：

1. 主控端发送 MODBUS ASCII 命令帧：
发送数据(HEX)：0x3A 0x01 0x03 0xA3 0x48 0x00 0x02 0x0F 0x0D 0x0A
其中：
0x3A: ASCII 帧起始标识
0x01: 从设备地址(出厂默认为0x01,可设置范围0x01~0xFE)
0x03: 读寄存器功能
0xA3 0x48: 数据寄存器首地址(出厂默认为41800,可设置范围40000~49999)
0x00 0x02：读取数据字个数(2个16 bits数据，分别对应圈数值和角度值)
0x0F: LRC 校验
0x0D 0x0A: ASCII 帧结束标识

2. 主控端接收来自编码器的数据帧：
接收数据(HEX)：0x3A 0x01 0x03 0x04 0x07 0x08 0x09 0x0A 0xD6 0x0D 0x0A
其中：
0x3A: ASCII 帧起始标识
0x01: 从设备地址
0x03: 读寄存器功能
0x04: 寄存器字节数量
0x07 0x08: 圈数值 = 0x0708 = 1800 (max.4095)
0x09 0x0A: 角度分辨率值 = 0x090A = 2314 (max.16383)
0xD6: LRC 校验
0x0D 0x0A: ASCII 帧结束标识

II)编码器内部温度读出命令：

发送数据(HEX)：0x3A 0x01 0x03 0xA3 0x4A 0x00 0x01 0x0E 0x0D 0x0A
其中：
0x3A: ASCII 帧起始标识

0x01: 从设备地址(出厂默认为0x01,可设置范围0x01~0xFE)
0x03: 读寄存器功能
0xA3 0x4A: 温度值寄存器地址(即数据首地址+2)
0x00 0x01：读取数据字个数(1个16 bits数据，对应温度值)
0x0E: LRC 校验
0x0D 0x0A: ASCII 帧结束标识

2. 主控端接收来自编码器的数据帧：
接收数据(HEX)：0x3A 0x01 0x03 0x02 0x00 0x35 0xC5 0x0D 0x0A
其中：
0x3A: ASCII 帧起始标识
0x01: 从设备地址
0x03: 读寄存器功能
0x02: 寄存器字节数量
0x00 0x35: 温度值 = 0x0035 = 53(℃)
0xC5: LRC 校验
0x0D 0x0A: ASCII 帧结束标识

三、参数设置通讯方式（无协议 主动/被动模式）

以下命令只适合单机设置时运行，并且默认处于上电锁定状态，须先执行解锁命令才能进行后续各项参数的设置

1. 设置解锁命令：
发送命令(HEX)：0x55 0xAA 0xF7 0x7F 0x0D 0x0A
接收数据(ASCII): 如设置成功板载指示灯会闪烁 1 次。

2. 参数读取命令：
发送命令(HEX)：0x55 0xA0 0xXX 0xXX 0x0D 0x0A (0xXX 表示可以是任意数，下同)
接收数据(ASCII): 依次收到可记圈数和分辨率、波特率、子地址、数据地址、计数方向、圈数预设值、角度预设值、主动模式、时间间隔、圈数间隔、角度间隔，如设置成功板载指示灯会闪烁 1 次。

3. 波特率设置命令：
发送命令(HEX)：0x55 0xA1 Baudrate 0xXX 0x0D 0x0A
Baudrate(Byte)取值范围为 0~7，分别对应波特率:
0--9600bps, 1--19200bps, 2--38400bps, 3--57600bps, 4--115200bps,
5--230400bps, 6--460800bps, 7--921600bps
接收数据(ASCII): 收到设定的波特率值，如设置成功板载指示灯会闪烁 1 次。

4. 子地址设置命令：
发送命令(HEX)：0x55 0xA2 Node_address 0xXX 0x0D 0x0A
Node_address(Byte)取值范围为 1~254
接收数据(ASCII): 收到设定的子地址值，如设置成功板载指示灯会闪烁 1 次。

5. 数据地址设置命令：
发送命令(HEX)：0x55 0xA3 Modbus_ADDH Modbus_ADDL 0x0D 0x0A
Modbus_ADD (Int)取值范围为 00001~49999
接收数据(ASCII): 收到设定的数据地址值，如设置成功板载指示灯会闪烁 1 次。

6. 计数方向设置命令：
发送命令(HEX)：0x55 0xA4 Direction 0xXX 0x0D 0x0A
Direction (Byte)取值范围为 0 或 1，分别对应计数方向: 0--CW, 1--CCW
接收数据(ASCII): 收到设定的计数方向，如设置成功板载指示灯会闪烁 1 次。

7. 圈数预设置命令：
发送命令(HEX)：0x55 0xA5 Preset_Turns_H Preset_Turns_L 0x0D 0x0A
Preset_Turns (Int)取值范围为 0~Max.Turns -1
接收数据(ASCII): 收到设定的圈数预设值，如设置成功板载指示灯会闪烁 1 次。

8. 角度预设置命令：
发送命令(HEX)：0x55 0xA6 Preset_Angle_H Preset_Angle_L 0x0D 0x0A
Preset_Angle (Int)取值范围为 0~Max.Angle -1
接收数据(ASCII): 收到设定的角度预设值，如设置成功板载指示灯会闪烁 1 次。

9. 软加载预设值命令：
发送命令(HEX)：0x55 0xA7 0xXX 0xXX 0x0D 0x0A
接收数据(ASCII): 无，如设置成功板载指示灯会闪烁 1 次。

10. 编码器主动发送数据启动命令：
发送命令(HEX)：0x55 0xB0 Node_address OTP_Send 0x0D 0x0A
Node_address 为需要启动的编码器
OTP_Send 默认值为 0xFF,如果被设置为特定值 **0x95**,则此编码器被设置为强主动模式，一上电即开始主动发送数据（可通过 SETPOS 引线恢复出厂默认设置）
接收数据(HEX): 收到 MOBUS RTU 响应数据帧（具体可参照前页描述），如发送成功板载指示灯会闪烁 1 次。该命令执行后，编码器即进入主动连续发送数据状态，若要转入被动模式，需重新上电。

11. 编码器主动发送模式设置命令：
发送命令(HEX)：0x55 0xB1 Master_Mode 0xXX 0x0D 0x0A
Master_Mode(Byte)取值范围为 0 或 1，分别对应两种主动模式: 0--按照时间间隔, 1--按照空间间隔
接收数据(ASCII): 收到设定的主动模式，如设置成功板载指示灯会闪烁 1 次。

12. 主动发送时间间隔设置命令：
发送命令(HEX)：0x55 0xB2 Interval_TH Interval_TL 0x0D 0x0A
Interval_T (int)取值范围为 0~65534，单位为毫秒
接收数据(ASCII): 收到设定的时间间隔值，如设置成功板载指示灯会闪烁 1 次。

13. 主动发送空间间隔圈数设置命令：
发送命令(HEX)：0x55 0xB3 Interval_RH Interval_RL 0x0D 0x0A
Interval_R (int)取值范围为 0~4094，单位为圈
接收数据(ASCII): 收到设定的圈数间隔值，如设置成功板载指示灯会闪烁 1 次。

14. 主动发送空间间隔角度设置命令：
发送命令(HEX)：0x55 0xB4 Interval_AH Interval_AL 0x0D 0x0A
Interval_A (int)取值范围为 0~16382，单位为最小分辨率
接收数据(ASCII): 收到设定的角度间隔值，如设置成功板载指示灯会闪烁 1 次。

****

CRC 生成函数代码示例（MODBUS RTU 模式）

```
//unsigned char *puchMsg ; /* message to calculate CRC upon */
//unsigned int usDataLen ; /* quantity of bytes in message */

unsigned int CRC16(unsigned char *puchMsg, unsigned int usDataLen)
{
    unsigned char uchCRCHi = 0xFF ; /* high CRC byte initialized */
    unsigned char uchCRCLo = 0xFF ; /* low CRC byte initialized */
    unsigned char uIndex ; /* will index into CRC lookup table*/
    while (usDataLen--) /* pass through message buffer*/
    {
        uIndex = uchCRCHi ^ *puchMsg++ ; /* calculate the CRC*/
        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
        uchCRCLo = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High Order Byte Table

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;

Low Order Byte Table

/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 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
} ;

RS485 伺服兼容通讯方式：

I） 编码器单圈数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x02

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x02 0x20 0x03 0x02 0x01 0x22

其中：

0x02: 返回相同命令CF

0x20: 状态字节SF 定义如下(低位在前)

Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
0	0	0	0	EA0	EA1	CA0	CA1

EA0=1 单圈计数错误

EA1=1 超温、计圈错误、电池报警、电池错误之一(具体错误可查看故障内容字节ALMC)

CA0=1 通讯奇偶校验错误

CA1=1 通讯停止位错误

0x03 0x02 0x01: 单圈数据值DF(低位在前) = 0x010203 = 66051 (max.262143)

0x16: CRC校验(将前面所有字节进行异或运算)

II）编码器多圈数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x8A

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x8A 0x20 0x06 0x05 0x04 0xAD

其中：

0x8A: 返回相同命令CF

0x20: 状态字节SF

0x06 0x05 0x04: 圈数数据值DF(低位在前) = 0x040506 = 263430 (max.8388351)

0xAD: CRC校验(将前面所有字节进行异或运算)

III） 编码器 ID 数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x92

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x92 0x20 0x11 0xA3

其中：

0x92: 返回相同命令CF

0x20: 状态字节SF

0x11: 编码器ID，固定值=0x11

0xA3: CRC校验(将前面所有字节进行异或运算)

IV）编码器所有数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x1A

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x1A 0x20 0x03 0x02 0x01 0x11 0x06 0x05 0x04 0x22 0x0E

其中：

0x1A: 返回相同命令CF

0x20: 状态字节SF

0x03 0x02 0x01: 单圈数据值DF(低位在前) = 0x010203 = 66051 (max.262143)

0x11: 编码器ID，固定值=0x11

0x06 0x05 0x04: 圈数数据值DF(低位在前) = 0x040506 = 263430 (max.8388351)

0x22: 故障内容字节ALMC 定义如下(低位在前)

Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
超速	低分辨率状态	单圈计数错误	多圈计数溢出	超温	多圈计数错误	电池错误	电池报警

0x0E: CRC校验(将前面所有字节进行异或运算)

V） 编码器错误复位请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0xBA -- 复位指令至少 40us 间隔重复发 10 次，复位所有的错误报警位信息

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0xBA 0x20 0x03 0x02 0x01 0x9A

其中：

0xBA: 返回相同命令CF

0x20: 状态字节SF

0x03 0x02 0x01: 单圈数据值DF(低位在前) = 0x010203 = 66051 (max.262143)

0x9A: CRC校验(将前面所有字节进行异或运算)

VI）编码器单圈复位请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0xC2 -- 复位指令至少 40us 间隔重复发 10 次，复位单圈角度值

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0xC2 0x20 0x00 0x00 0x00 0xE2

其中：

0xC2: 返回相同命令CF

0x20: 状态字节SF

0x00 0x00 0x00: 单圈数据值DF(低位在前)

0xE2: CRC校验(将前面所有字节进行异或运算)

VII） 编码器圈数复位请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x62 -- 复位指令至少 40us 间隔重复发 10 次，复位多圈圈数值

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x62 0x20 0x00 0x00 0x00 0x42

其中：

0x62: 返回相同命令CF

0x20: 状态字节SF

0x00 0x00 0x00: 圈数数据值DF(低位在前)

0x42: CRC校验(将前面所有字节进行异或运算)

VIII）写入 EEPROM 数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0x32 0x11 0x22 0x01

其中：

0x32: 写EEPROM命令

0x11: EEPROM地址(低位在前，0~127,最高字节127为页地址)

0x22: EEPROM数据(低位在前)

0x01: CRC校验(将前面所有字节进行异或运算)

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0x32 0x11 0x22 0x01

即返回相同命令帧

IX）读出 EEPROM 数据请求命令：

1. 主控端发送命令帧：

发送数据(HEX)：0xEA 0x11 0xFB

其中：

0xEA: 读EEPROM命令

0x11: EEPROM地址(低位在前，0~127,最高字节127为页地址)

0xFB: CRC校验(将前面所有字节进行异或运算)

2. 主控端接收来自编码器的数据帧：

接收数据(HEX)：0xEA 0x11 0x22 0xD9

其中：

0xEA: 返回相同命令CF

0x11: EEPROM地址(低位在前，0~127,最高字节127为页地址)

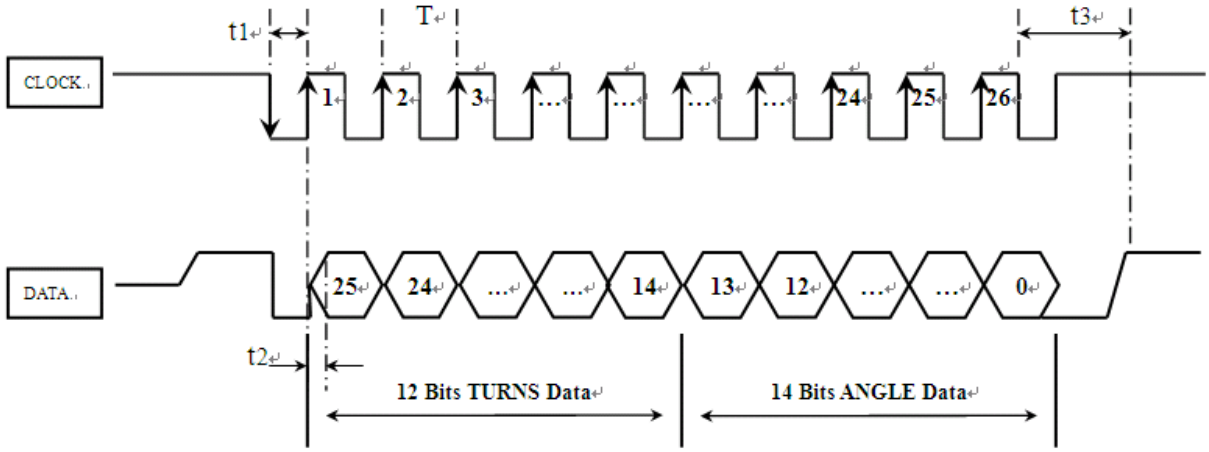
0x22: EEPROM数据(低位在前)

0xD9: CRC校验(将前面所有字节进行异或运算)

RS422 SSI 接口通讯方式

时钟频率范围：250Kbps -- 1Mbps

SSI 时序图及示例代码(示例为 12 位多圈+14 位单圈)



Time Description

t1 > 0.45us

t2 < 0.40us

t3 = 12us ~ 30us

T = 1us ~ 11us

C Code Example

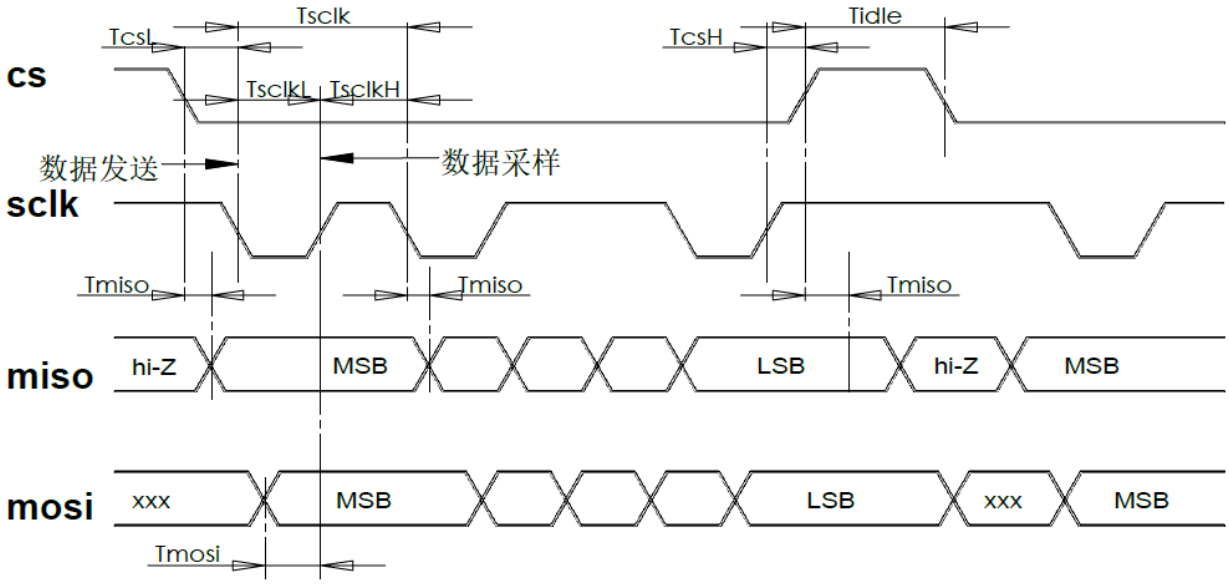
```
#define databit 26
uint32 Getdata;
uint16 TURNS,ANGLE;
void main(void)
{
   uint i;
   SSI_CLOCK = 1;              //SSI clock wire
   Delay_us(1);                //Delay 1us
   SSI_CLOCK = 0;              //Latch data into output shift register
   Delay_us(1);                //Delay 1us
   for(i=0;i<databit;i++)      //Get all of data from encoder
   {
        Getdata <<= 1;         //Shift one higher bit left
        SSI_CLOCK =0;
        SSI_CLOCK =1;      //Get one bit of data at clock rising edge
        Getdata |= SSI_DATA; //Read one bit from SSI data wire
   }
   TURNS = Getdata >> 14; //Get 12 bit Turns
   ANGLE = Getdata & 0x3fff; //Get 14 bit Angle
   Delay_us(30);               //Delay 30us
}
```

SPI 通讯协议

SPI 四线全双工通讯方式：(SLAVE MODE, CPOL=1 CPHA = 1)

SPI 数据位数：16bit 最高支持 8M 通讯速率

（CPOL=1 CPHA=1 时钟闲态为高电平，时钟下降沿数据发送，上升沿数据采样）



参数	描述	最小值	最大值	单位

Tidle	两个字节序列传输间隔时间	150		ns
	两次寄存器读或写间隔时间	800		ns
	两次存储器读或写间隔时间	15		ms
TcsL	cs 和 sclk 下降沿时间间隔	90		ns
TcsH	cs 和 sclk 上降沿时间间隔	35		ns
Tsclk	sclk 时钟周期	100		ns
TsclkL	sclk 时钟低电平时间	50		ns
TsclkH	sclk 时钟高电平时间	50		ns
Tmiso	时钟设置边沿到有效数据输出时间		25	ns
Tmosi	有效数据输入到时钟读取边沿时间	25		ns

SPI 指令定义：(*所有数据均为 16 位格式)

1.指令列表（表 1）

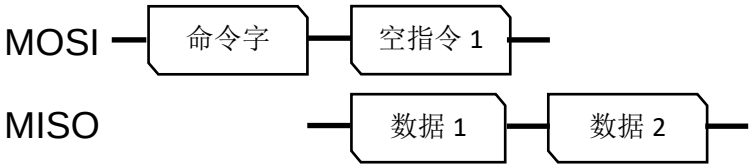
指令 名称	指令 描述	指令 字	地址 字	数据 字	正常 模式	设置 模式
ChangeMode	切换到正常/参数模式	0x01	0x00~0x01	N/A	valid	valid
ReadPosition	读取圈数和角度位置	0x02	N/A	N/A	valid	valid
ReadTurns	读取圈数信息	0x03	N/A	N/A	valid	valid
ReadAngle	读取角度信息	0x04	N/A	N/A	valid	valid
SetPosition	置位圈数和角度值	0x05	0x01~0x03	N/A	invalid	valid
ReadReg	读取内部寄存器	0x06	0x00~0x7F	N/A	valid*	valid
WriteReg	设置内部寄存器	0x07	0x00~0x7F	0x00~0xFF	invalid	valid
ReadMemory	读数据从内部存储器	0x08	0x00~0x7F	N/A	invalid	valid
WriteMemory	写数据到内部存储器	0x09	0x00~0x7F	0x00~0xFF	invalid	valid
WriteNVmem	写入到非易失存储区	0x0A	0x00~0x01	N/A	invalid	valid
RenewPara	所有参数恢复初始值	0x0B	N/A	N/A	invalid	valid

(N/A:保留，valid:可操作，invalid:不能操作，valid*:仅寄存器 0x00~0x10 可操作)

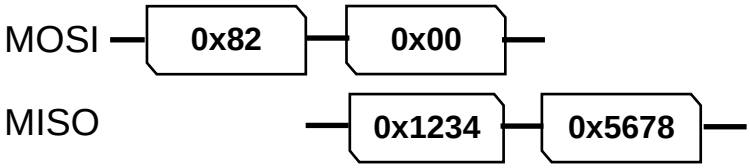
2.指令格式

一条指令帧由命令字、地址字、数据字和空指令组成，并且命令字的第 8 位(即 bit0~15 中的 bit7)为整帧的补奇偶位，即确保整帧逻辑 1 的个数始终为偶数，如果接收到的有效数据个数大于发送命令字、地址字、数据字之和，则需在发送帧末尾补充对应个数的空指令，如果接收到的有效数据个数小于发送帧命令字、地址字、数据字之和，则会在接收帧头出现补充数量的空数据。总之，须保证发送帧字个数与接收帧字个数完全相等。针对不同功能的指令，可分为单字/双字/三字三种形式：

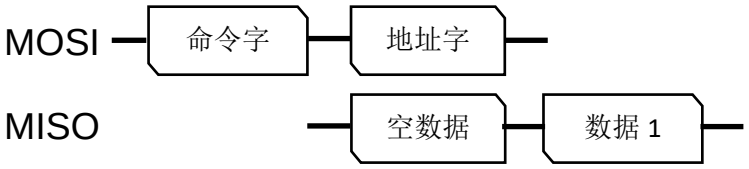
单字指令：



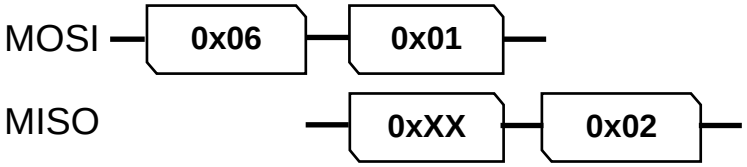
例如 ReadPosition 指令，读取圈数和角度值分别为 0x1234,0x5678：



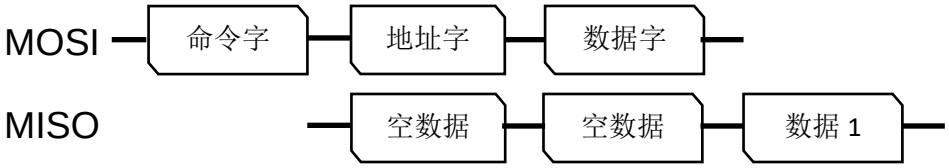
双字指令：



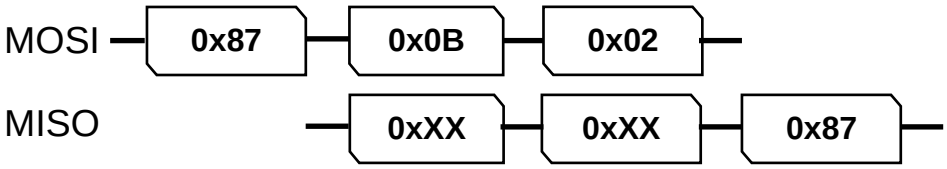
例如 ReadReg 指令，读取寄存器 0x01 的值为 0x02:



三字指令：



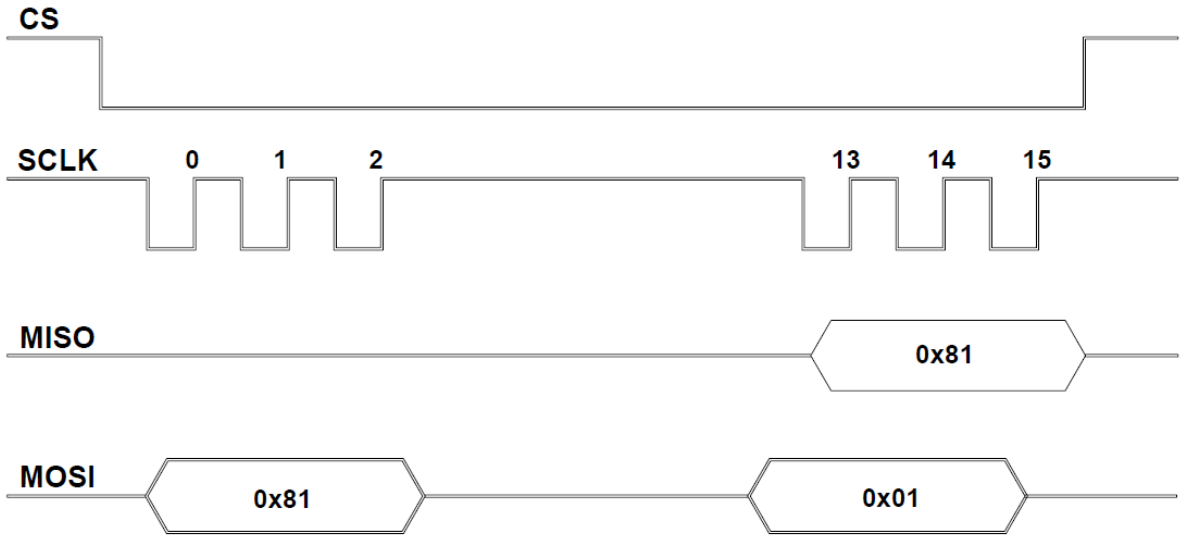
例如 WriteReg 指令，对寄存器 0x0B 写入值 0x02，返回 0x87 应答:



3.指令描述

ChangeMode 切换到正常/参数模式：0x0001(bit7 位补偶后为 0x0081)

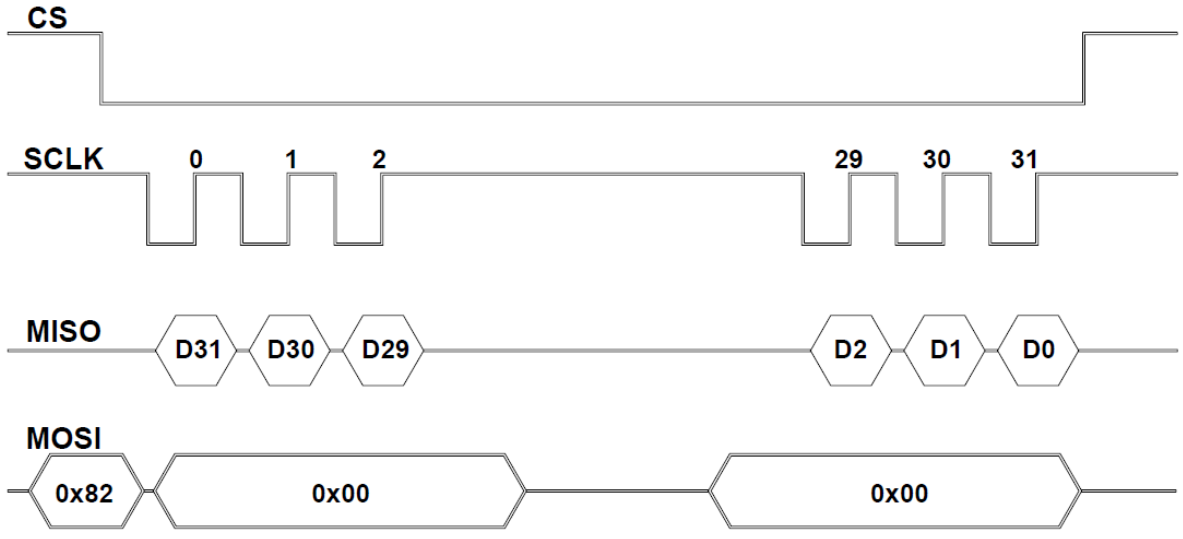
切换当前模式为正常或参数模式，以避免在正常工作模式下，误操作相关设定参数
地址字 0x00：设置为正常模式
地址字 0x01：设置为参数模式



ChangeMode Command

ReadPosition 读取圈数和角度位置：0x0002(bit7 位补偶后为 0x0082)

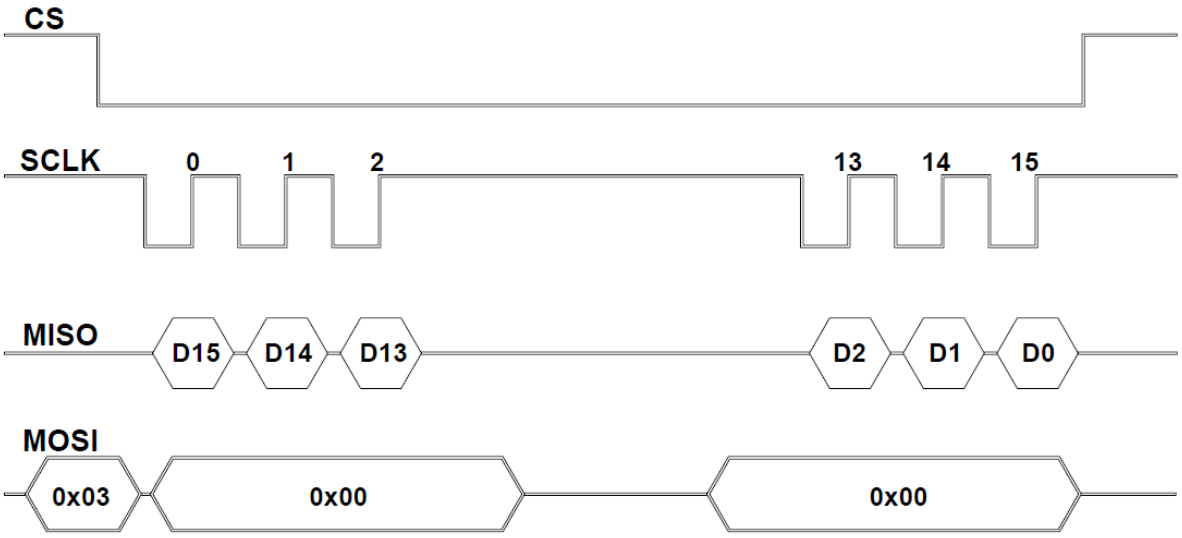
一次连续读取 32 位数据，其中高 16 位为圈数值，低 16 位为角度值，整个读取过程中，需保证 CS 一直为低电平。可以分 2 次(16 位宽,如下图)SPI 读操作。



ReadPosition Command

ReadTurns 读取圈数信息：0x0003

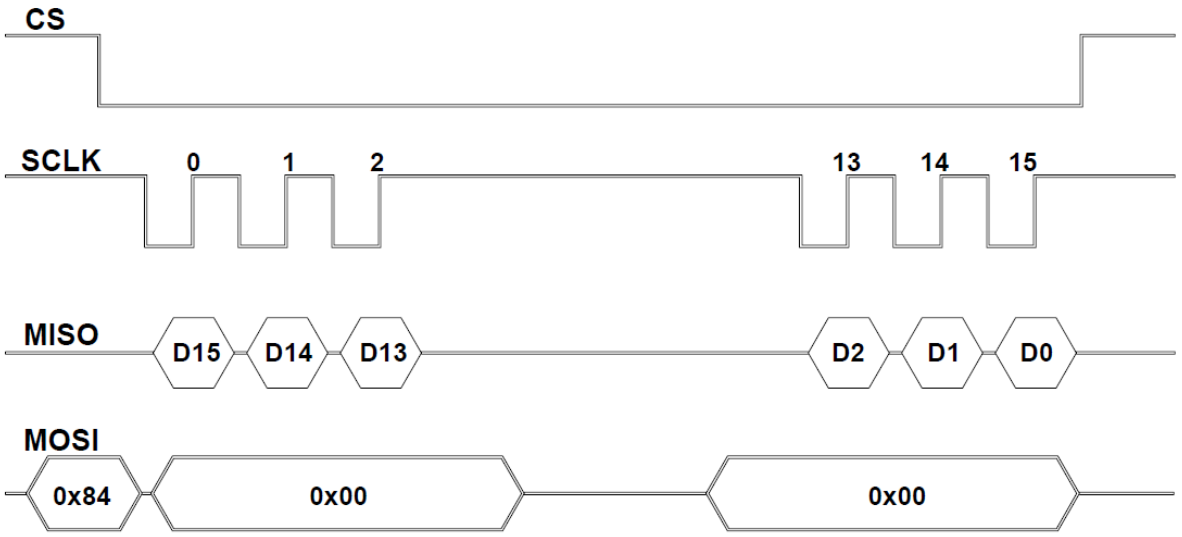
一次连续读取 16 位圈数数据，整个读取过程中，需保证 CS 一直为低电平。并执行 1 次(16 位宽,如下图)SPI 读操作。



ReadTurns Command

ReadAngle 读取角度信息：0x0004(bit7 位补偶后为 0x0084)

一次连续读取 16 位角度数据，整个读取过程中，需保证 CS 一直为低电平，并执行 1 次(16 位宽,如下图)SPI 读操作。

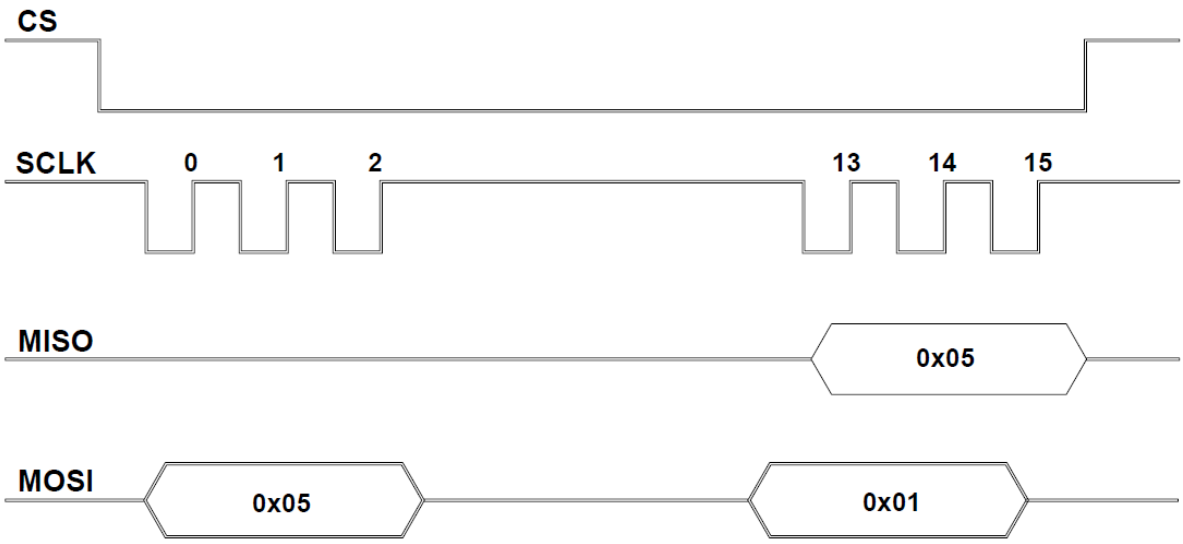


ReadAngle Command

SetPosition 置位圈数和角度值: 0x0005

将当前位置设置成预设值寄存器(0x13~0x16)中指定的值，该指令只在参数模式下有效，

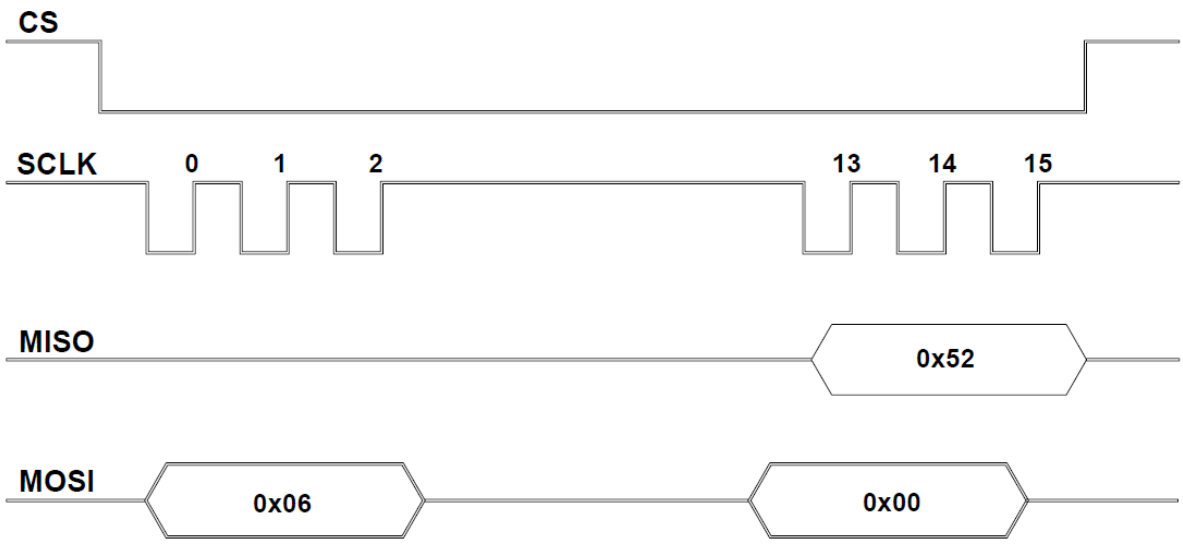
视情况，可配合 WriteNVmem 指令使用。假如预设值寄存器内的值为 0，就相当于将当前位置设置成零位，并返回相同指令字节作为应答。
地址字 0x01：同时设置圈数和角度
地址字 0x02：仅设置圈数
地址字 0x03：仅设置角度



SetPosition Command

ReadReg 读取内部寄存器：0x0006

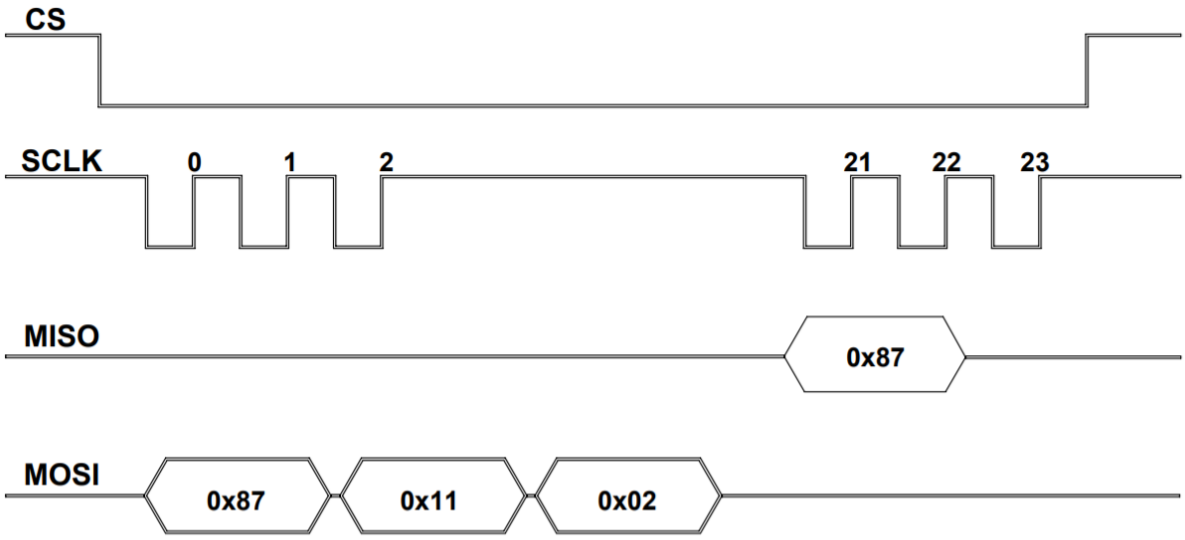
读取内部寄存器的值，该指令大部分只在参数模式下有效，具体详见（表 2）。
下图为读取传感器类别号寄存器地址 0x00 中的值，返回值为 0x52。



ReadReg Command

WriteReg 设置内部寄存器：0x0007(bit7 位补偶后为 0x0087)

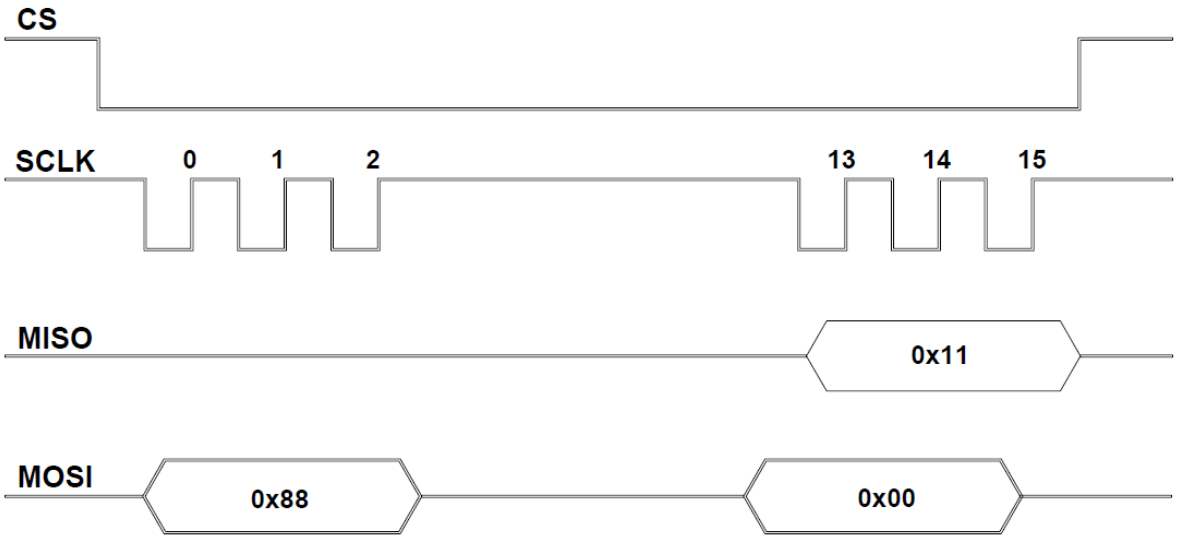
设置内部寄存器的值，该指令大部分只在参数模式下有效，具体详见（表 2），视情况，可配合 WriteNVmem 指令使用。下图为设置 CNTL 寄存器地址 0x11 中的值为 0x02，并返回相同指令字节作为应答。



WriteReg Command

ReadMemory 读数据从内部存储器：0x0008(bit7 位补偶后为 0x0088)

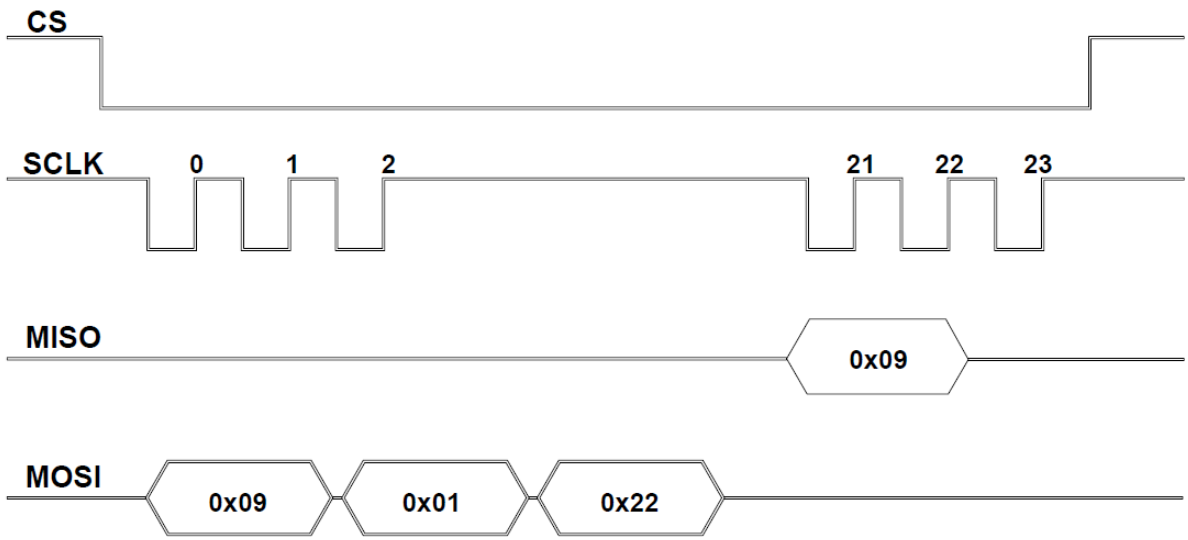
读取内部存储器数据，该指令只在参数模式下有效。
下图为读取存储器地址 0x00 中的值，返回值为 0x11。



ReadMemory Command

WriteMemory 写数据到内部存储器：0x0009

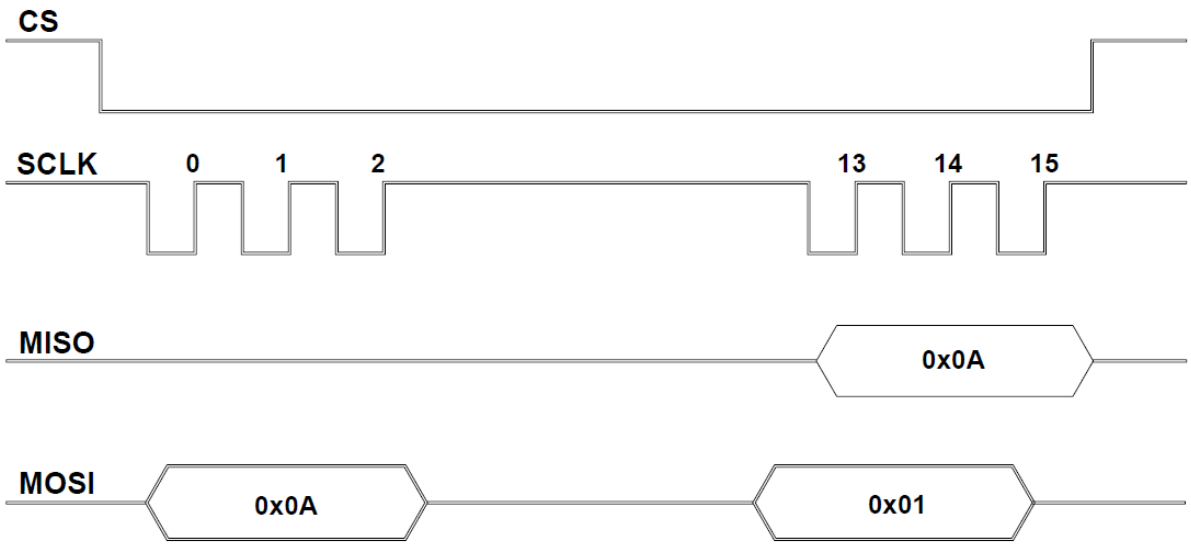
写入内部存储器数据，该指令只在参数模式下有效，视情况，可配合 WriteNVmem 指令使用。下图为对存储器地址 0x01 写入数据 0x22，并返回相同指令字节作为应答。



WriteMemory Command

WriteNVmem 写入到非易失存储区：0x000A

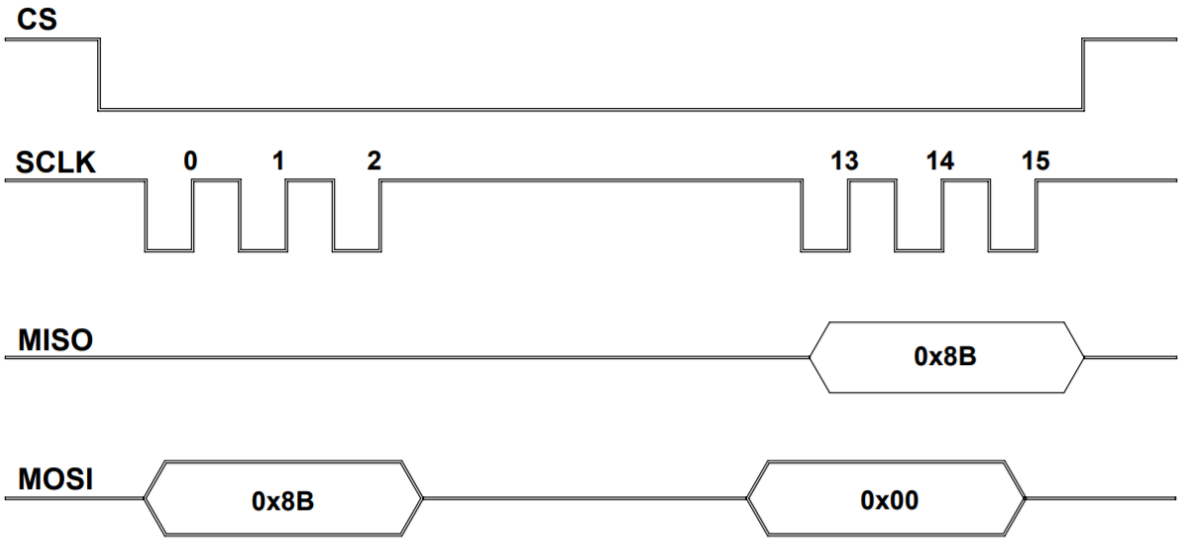
配合寄存器或存储区写指令使用，将相关参数或数据写入掉电非易失存储区中，可以支持最多 10000 次寄存器或存储器写入，每次写入须间隔 200ms(注意：为避免意外连续写入，不要在无限循环程序中执行该指令)，并返回相同指令字节作为应答。
地址字 0x00：将所有参数和数据写入掉电非易失存储区
地址字 0x01：将寄存器参数写入掉电非易失存储区
地址字 0x02：将存储器数据写入掉电非易失存储区



WriteNVmem Command

RenewPara 所有参数恢复初始值：0x000B(bit7 位补偶后为 0x008B)

将所有参数恢复成出厂初始值，仅在参数模式下有效，并返回相同指令字节作为应答，视情况，可配合 WriteNVmem 指令使用。



RenewPara Command

SPI 寄存器定义：(*所有地址和数据都为 16 位，目前都只用到低 8 位)

1.寄存器列表（表 2）

地址	名称	寄存器描述	正常模式	参数模式
0x00	CID	传感器类别号	R	R
0x01	TurnsH	圈数高 8 位	R	R
0x02	TurnsL	圈数低 8 位	R	R
0x03	AngleH	角度高 8 位	R	R

0x04	AngleL	角度低 8 位	R	R
0x05	STATUS	状态寄存器（空/忙）	R	R
0x06	ALARM	报警寄存器（奇偶校验/弱磁/过压）	R	R
0x07~0x0F		保留	N/A	N/A
0x10	CMODE	模式设置（正常/设置模式）	R/W	R/W
0x11	CNTL	控制寄存器（置位/方向/总圈数/总分辨率）	invalid	R/W
0x12	SDMODE	发送模式寄存器（时间/位置/命令）	invalid	R/W
0x13	PTurnsH	圈数预设值高 8 位	invalid	R/W
0x14	PTurnsL	圈数预设值低 8 位	invalid	R/W
0x15	PAngleH	角度预设值高 8 位	invalid	R/W
0x16	PAngleL	角度预设值低 8 位	invalid	R/W
0x17~0x1F		保留	N/A	N/A
0x20	MParam1	标定系数 1	invalid	R/W
0x21	MParam2	标定系数 2	invalid	R/W
0x22	MParam3	标定系数 3	invalid	R/W
0x23	MParam4	标定系数 4	invalid	R/W
0x24	MParam5	标定系数 5	invalid	R/W

(N/A:保留，invalid:不能操作，R:只读，W:只写，R/W:可读写)

2.寄存器描述

CID: 传感器类别号，固定值用户不可修改　　地址：0x00　　初始值：0x52

7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	0

TurnsH：圈数高 8 位　　地址：0x01　　初始值：xxx

7	6	5	4	3	2	1	0
Turns[15:8]							

TurnsL：圈数低 8 位　　地址：0x02　　初始值：xxx

7	6	5	4	3	2	1	0
Turns[7:0]							

AngleH：角度高 8 位　　地址：0x03　　初始值：xxx

7	6	5	4	3	2	1	0
Angle[15:8]							

AngleL：角度低 8 位　　地址：0x04　　初始值：xxx

7	6	5	4	3	2	1	0
Angle[7:0]							

STATUS：状态寄存器　　地址：0x05　　初始值：0x00

7	6	5	4	3	2	1	0
ready	updata	alarm	n/a	n/a	n/a	n/a	n/a

ready：0 – 非空闲，1-- 空闲
updata：0 – 无更新数据， 1 – 有更新数据
alarm：0 – 无报警， 1 – 有报警

ALARM：报警寄存器 地址：0x06 初始值：0x00

7	6	5	4	3	2	1	0
n/a	iadc	mag_bal	ov_vol	un_angle	mag_weak	err_check	err_pol

正常为 0，错误为 1，每次读取寄存器时，会自动整体清零。
iadc: 内部 ADC 错误 mag_bal：磁场失衡 ov_vol：电压异常 un_angle：角度值无效 mag_weak：磁场太弱 err_check：校验和错误 err_pol：偶校验错误

CMODE：模式设置 地址：0x10 初始值：0x00

7	6	5	4	3	2	1	0
CMODE [7:0]							

为了防止误操作，分成两种权限模式，即正常模式：0x00，设置模式：0x01，不同模式下的命令执行和寄存器操作对应不同的限制，具体见表 1、表 2 说明。

CNTL：控制寄存器 地址：0x11 初始值：0x00

7	6	5	4	3	2	1	0
n/a	corrfcn	n/a	n/a	n/a	dir	setturn	setangle

corrfcn：线性度校正方式选择 0 – 多项式方程，1 – 指数方程，2—高斯方程
dir：0 – CW，1—CCW
setturn：0 – 不使能， 1 – 将预设圈数置为当前圈数
setangle：0 – 不使能， 1 – 将预设角度置为当前角度

SDMODE：发送模式寄存器 地址：0x12 初始值：0x01

7	6	5	4	3	2	1	0
active	n/a	n/a	n/a	n/a	time	pos	cmd

active：0 -- 停止主动发送，1 -- 启动主动发送
time：0 – 不使能 ，1 -- 以设定的时间间隔主动发送
pos：0 – 不使能， 1 – 以设定的位置间隔主动发送
cmd：0 – 不使能， 1 – 被动等待命令后发送

PTurnsH：圈数预设值高 8 位 地址：0x13 初始值：0x00

7	6	5	4	3	2	1	0
Pre_Turns[15:8]							

PTurnsL：圈数预设值低 8 位　　地址：0x14　　初始值：0x00

7	6	5	4	3	2	1	0
Pre_Turns[7:0]							

PTurnsH，PTurnsL 为用户预设置的圈数值，当 SetPosition 命令被执行，或者控制寄存器 CNTL 中的 setturn 位置 1，即会将当前圈数变换成用户预设值，假如预设值为零，就相当于是零位设置。

PAngleH：角度预设值高 8 位　　地址：0x15　　初始值：0x00

7	6	5	4	3	2	1	0
Pre_Angle[15:8]							

PAngleL：角度预设值低 8 位　　地址：0x16　　初始值：0x00

7	6	5	4	3	2	1	0
Pre_Angle[7:0]							

PAngleH，PAngleL 为用户预设置的角度值，当 SetPosition 命令被执行，或者控制寄存器 CNTL 中的 setangle 位置 1，即会将当前角度变换成用户预设值，假如预设值为零，就相当于是零位设置。

MParam1~5：标定系数 1~5　　地址：0x20~0x24　　初始值：0x00

7	6	5	4	3	2	1	0
sign	MParam X [6:0]						

Bit7 位为符号位，用于存储根据线性校正方程获取的各项系数。

主控端 SPI 读写 C 代码示例 1：读取圈数和角度位置

主程序:

```
unsigned int RxBuffer[10];

void main(void)
{
    SPIM_WriteTxData(0x82);          //单字命令---同时读取圈数和角度位置
    SPIM_WriteTxData(0x00);          //发送空指令，将有效数据移位出来
    while(SPIM_GetRxBufferSize() < 2); //等待接收缓冲区接收到2个有效数据
    RxBuffer[0] = SPIM_ReadRxData();    //读取圈数
    RxBuffer[1] = SPIM_ReadRxData();    //读取角度
}
```

主控端 SPI 读写 C 代码示例 2：读取角度位置

主程序:

```
unsigned int RxBuffer[10];

void main(void)
{
    SPIM_WriteTxData(0x84);          //单字命令---读取角度位置
    while(SPIM_GetRxBufferSize() < 1); //等待接收缓冲区接收到1个有效数据
    RxBuffer[0] = SPIM_ReadRxData();    //读取角度
}
```

主控端 SPI 读写 C 代码示例 3：读取 CID(0x00)寄存器内容

主程序:

```
unsigned int RxBuffer[10];

void main(void)
{
    SPIM_WriteTxData(0x06);          //双字命令---读取寄存器
    SPIM_WriteTxData(0x00);          //读取寄存器0x00内容
    while(SPIM_GetRxBufferSize() < 2); //等待接收缓冲区接收到2个有效数据
    RxBuffer[0] = SPIM_ReadRxData();    //读取空数据
    RxBuffer[1] = SPIM_ReadRxData();    //读取寄存器0x00内容，应为0x52
}
```

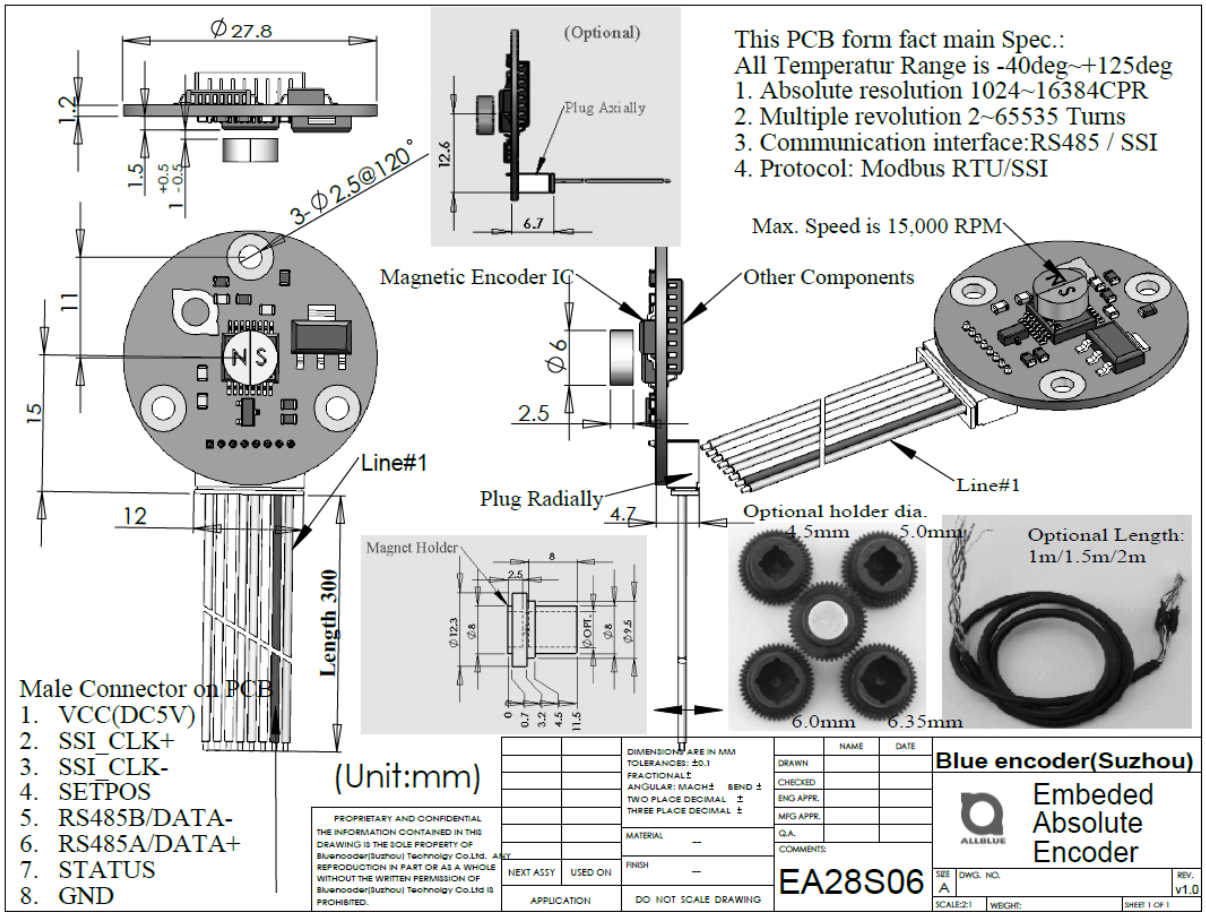
主控端 SPI 读写 C 代码示例 4：设置 CNTL(0x11)寄存器之旋转方向位

主程序:

```
unsigned int RxBuffer[10];

void main(void)
{
    SPIM_WriteTxData(0x87);          //三字命令---设置寄存器CMOD为0x01,
    SPIM_WriteTxData(0x10);          //进入参数模式
    SPIM_WriteTxData(0x01);
    DelayUs(100);

    SPIM_WriteTxData(0x87);          //三字命令---设置寄存器CNTL为0x04
    SPIM_WriteTxData(0x11);
    SPIM_WriteTxData(0x04);          //对应方向位dir bit = 1
```

```
    DelayUs(100);

    SPIM_WriteTxData(0x87);          //三字命令---设置寄存器CMOD为0x00,
    SPIM_WriteTxData(0x10);          //恢复到正常模式
    SPIM_WriteTxData(0x00);
    DelayUs(100);

    SPIM_WriteTxData(0x0A);          //双字命令---将寄存器值写入到非易失FLASH中,
    SPIM_WriteTxData(0x01);          //写入到寄存器FLASH区
}
```

产品尺寸图：



订货选择信息：

订货代码	描述
EA28S06-SVO	RS485接口/伺服兼容协议
EA28S06-RTU	RS485接口/RTU协议
EA28S06-ASCII	RS485接口/ASCII协议
EA28S06-SSI-G	RS422接口/SSI协议/格雷码
EA28S06-SSI-B	RS422接口/SSI协议/二进码
EA28S06-SPI	RS422接口/SPI协议

联系方式：

苏州傲蓝电子科技有限公司
地址：上海市浦东新区金皖路 399 号产业园 5 楼 508 室
电话：021-50373236；13901668071
公司网址：www.bluencoder.com； 电子邮箱：newmoon002@bluencoder.com