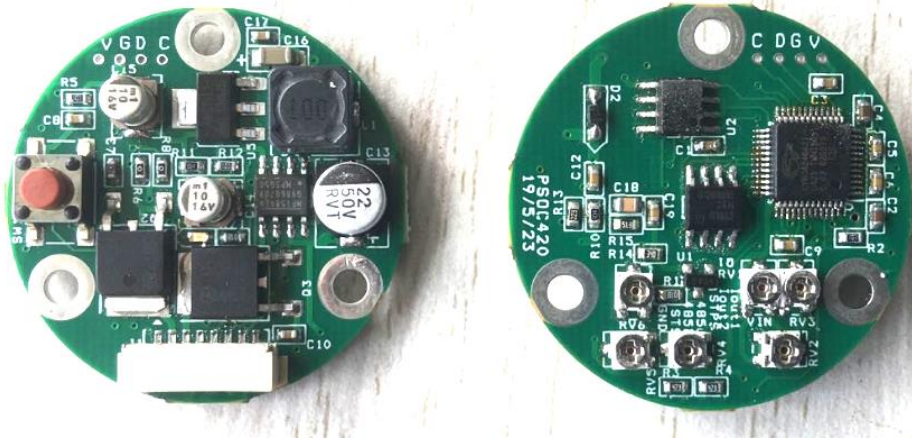




数字模拟混合编码器 EA36S06A



接口

接口	4~20mA & RS485
通讯协议	MODBUS RTU/ASCII(RS485)
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
接口响应时间	<1ms

输出

输出驱动器	4~20mA NPN & PNP, RS485总线差分
-------	-----------------------------

电气数据

电源电压	DC8~26V (电源符合EN 50178)
电流消耗	< 60 mA
功耗	$\leq 0.1W$
启动时间	< 100ms
模拟输出驱动负载阻抗	< 75 Ω
反极性保护	是
EMC:发射干扰	EN 61000-6-4
EMC:抗干扰	EN 61000-6-2
MTTF	50000小时, 在40 °C下

传感器

技术	磁电
最高单圈分辨率	32768cpr
最高可计圈数	65536r
多圈技术	电子方式计圈数(后备电池)
精度(INL)	$\pm 0.1^\circ$
码制	二进制码

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

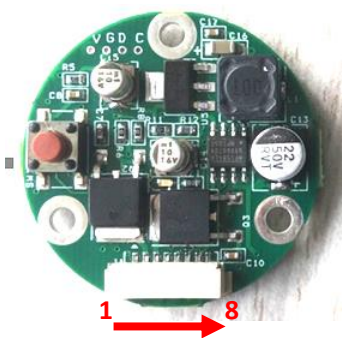


轴的类型	分离实心轴套，内嵌 $\varnothing 6 \times 2.5 \text{mm}$ 磁铁
产品直径	$\varnothing 36 \text{ mm}$
产品厚度	7 mm
最高机械速度	15000 rpm
重量	10g（不包含磁铁和线）

电气连接

连接方向	轴向或径向
------	-------

信号芯线定义



引脚号	引脚定义	说明
1	VDD	编码器电源 DC8~26V
2	IOUT1	模拟电流 NPN 无源输出
3	IOUT2	模拟电流 PNP 有源输出
4	SETPOS	位置设置输入(5V 高电平有效)
5	RS485A	RS485 接口 A
6	RS485B	RS485 接口 B
7	STATUS	状态指示输出
8	GND	电源地

SETPOS 是外部复用输入引线，具备两种功能(板载按钮与 SETPOS 有相同功能)：

1. 恢复出厂默认设置：

编码器上电前，预先将 SETPOS 与 5V 短接(或按下按钮)，上电 3 秒后，将 SETPOS 与 5V 断开(或释放按钮)。

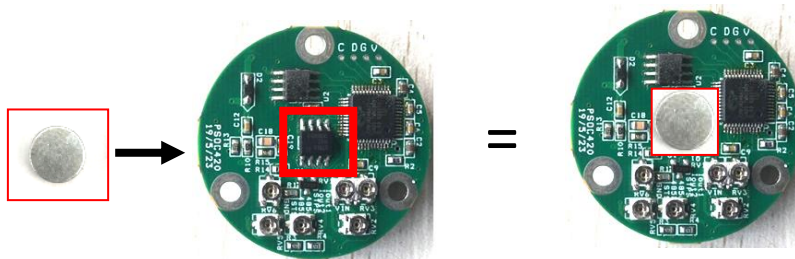
2. 设置原始位置：

编码器通电时，将 SETPOS 与 5V 短接 1 秒后(或按下按钮)，将 SETPOS 与 5V 断开(或释放按钮)。

STATUS 是编码器状态输出引线：1--Ready 0--Busy

离轴磁铁安装方式

将磁铁对中(偏差 $<0.2 \text{mm}$)编码器背面红框内芯片，并位于其上方 1mm(偏差 $<0.5 \text{mm}$)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 = $0x0035 = 53(^{\circ}\text{C})$

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(°C)

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 0xFF 0xFF 0x0D 0x0A (0xFF 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 0xFF 0x0D 0x0A

Baudrate(Byte)取值范围为 0~4, 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 0xFF 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 0xFF 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 0xFF 0xFF 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 MODBUS 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 0xFF 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 ^ uchCRCHi[uIndex] ;
```

```
        uchCRCLo = uchCRCLo[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

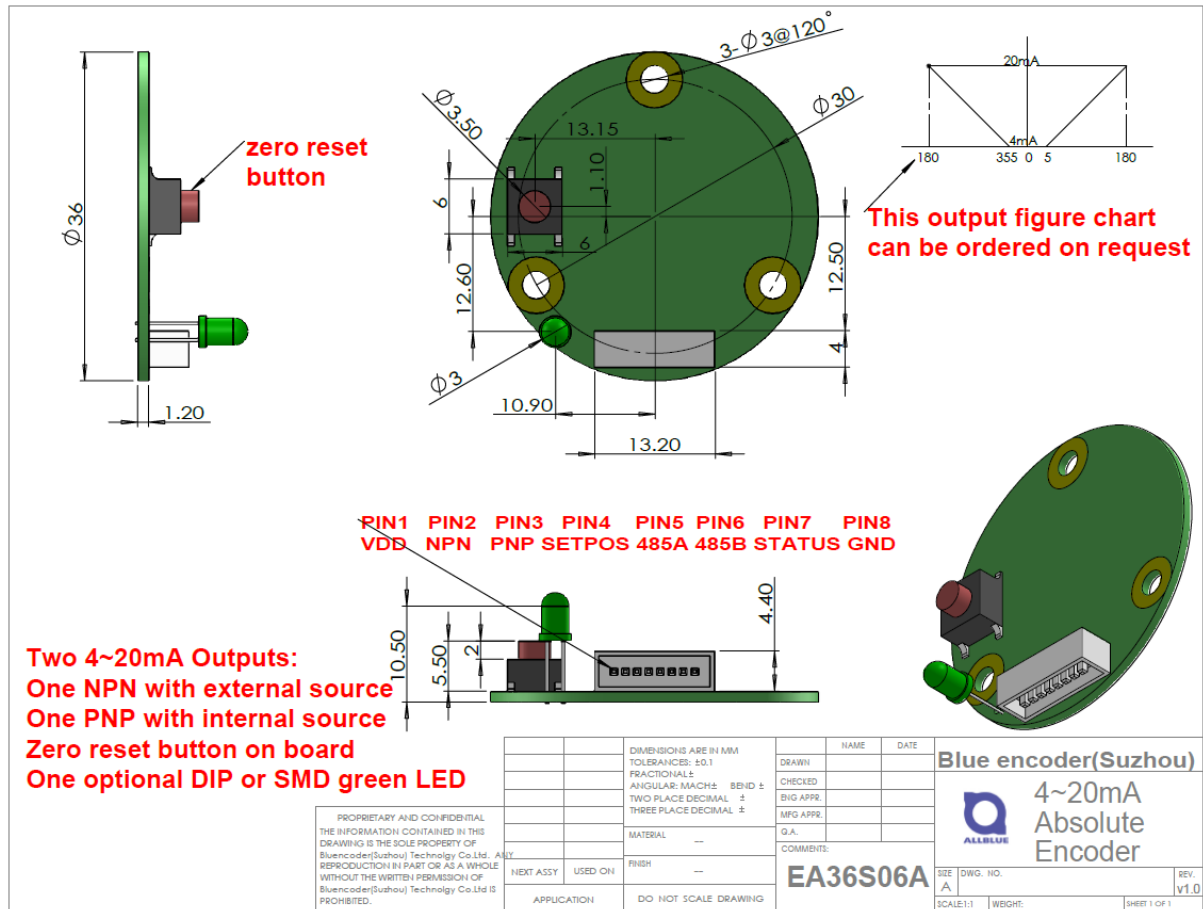
/* Table of CRC values for low-order byte */

```
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
```

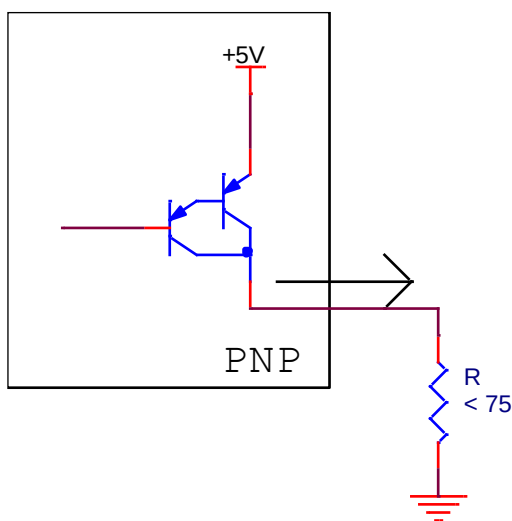
```
};
```



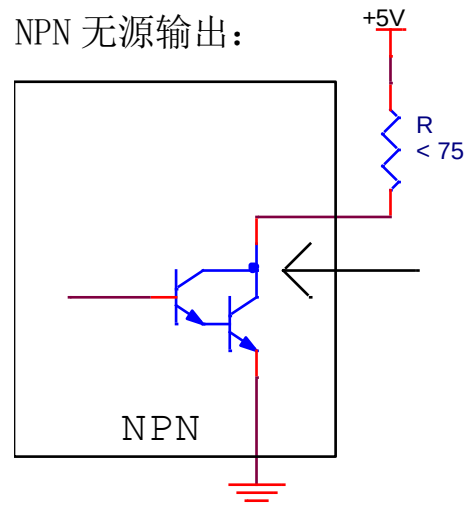

产品尺寸图:



PNP 有源输出:



NPN 无源输出:



注:

PNP 输出型的+5V 电源由编码器提供, NPN 输出型的+5V 电源由外部提供, 两者驱动负载电阻须小于 75 欧姆



订货选择信息:

订货代码	描述
EA36S06A-RTU	4~20mA / RS485接口 / RTU协议
EA36S06A-ASCII	4~20mA / RS485接口 / ASCII协议

联系方式:

苏州傲蓝电子科技有限公司

地址: 中国浙江省海宁市双联路 128 号 1 栋 5 楼东

电话: +86 573 80771560/80771561; +86 13901668071

公司网址: www.blucoder.com; 电子邮箱: newmoon002@blucoder.com

所有尺寸均为毫米, 此信息和图纸的目的旨在于一般性的介绍, 请参阅“下载”部分技术图纸进行了解, 傲蓝 Allblue 版权所有, 对于技术性误差或疏漏, 我们不承担责任, 产品规格的变更恕不另行通知。