



STSL 低功耗倾角传感器

产品说明书

V1.8-20260918

✔ 技术亮点

- ❖ 三轴测量；
- ❖ 量程 $\pm 30^\circ$ ；
- ❖ 分辨率 0.001°；
- ❖ 测量精度： $< \pm 15^\circ$: 0.01° / $< \pm 30^\circ$: 0.05°；
- ❖ IP67 防护等级。



✔ 产品介绍

STSL 是瑞惯科技面向工业现场控制领域推出的紧凑型、低功耗、多参数双轴倾角传感器，测量精度较好，可同步监测目标的倾斜角度、摆动频率、摆动幅度和中心偏移量，可用于铁塔、桥梁监测等工业工程场景。该产品角度输出稳定性较好，测量精度表现较佳，温漂特性较优，长期稳定性指标小于 0.02°。其采用五阶滤波算法对重力信号进行综合运算，可解析传感器 X、Y 轴角度。每台产品出厂前均借助多面转台完成交叉轴误差校正，有助于在较宽量程下降低测量轴间的相互影响。产品可选配 RS485 或 TTL 接口，接口集成保护电路，有利于减少对总线通信的干扰。供电部分采用多级稳压及过压、过流保护设计。该产品面向工业领域，运行稳定性较好，扩展能力较强，可用于多种严苛工业控制环境。具体参数以产品规格书为准。

✔ 应用范围

该系列产品典型应用场景包括：基坑测斜监测、工业自动化、工程机械平台调平系统，光伏跟踪支架控制系统，航空航天姿态参考系统，精密仪器水平校准装置，平台调平等高精度倾角测量领域。

工程机械设备测控



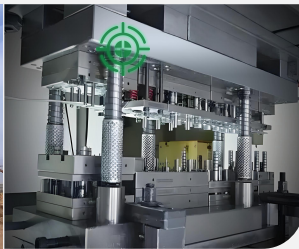
桥梁大坝安全监测



挖掘机角度测控



高精度大型治具



 性能参数

STSL	条件	参数	单位
量程		±30	°
测量轴		X Y Z	轴
分辨率		0.001	°
测量精度	@25℃	<±15°: 0.01° / <±30°: 0.05°	° (RMS)
长期稳定性		<0.02	°
零点温度系数	-40 ~ 85℃	0.0008	°/℃
灵敏度温度系数	-40 ~ 85℃	≤0.01	%/℃
摆动频率测量范围	<12.5Hz		
摆动频率测量精度	0.20Hz		
中心偏移量	通过安装高度与 Z 轴与中垂线的夹角换算		
动态摆幅	通过安装高度与 Z 轴与中垂线的动态夹角换算		
上电启动时间	2.5s		
响应时间	<0.1s		
响应频率	>10HZ		
输出信号	RS485/TTL 可选		
通信协议	MODBUS RTU 协议		
电磁兼容性	依照 EN61000 和 GBT17626		
绝缘电阻	≥100 兆欧		
抗冲击	100g@11ms、三轴向(半正弦波)		
抗振动	10grms、10 ~ 1000Hz		
防水等级	IP67		
电缆线	标配 2 米电缆线 4*0.2mm²		
重量	≤150g(含 2 米电缆线)		
分辨率：指传感器在测量范围内能够检测和分辨出的被测量的最小变化值。			
测量精度：在常温条件下对角度多次测量(16 次以上)，取测量值与实际角度误差的均方根差。			
长期稳定性：指传感器在常温条件下，经过一年的长期工作下最大值与最小值之间的偏差。			
响应时间：指传感器在一旦的角度变化时，传感器输出达到标准值所需的时间。			

 电气参数

参数	条件	最小值	典型值	最大值	单位
供电电压	标准	9	12	36	V
	可定制		5		V
工作电流	无负载		6.5 (12V)		mA
工作温度		-40		+85	℃
存储温度		-40		+85	℃

订购选型

型号	封装	信号输出
STSL	3:PCBA 板	485: RS485
	6:标准外壳	TTL: TTL
例：STSL6-485：标准外/RS485 输出。		

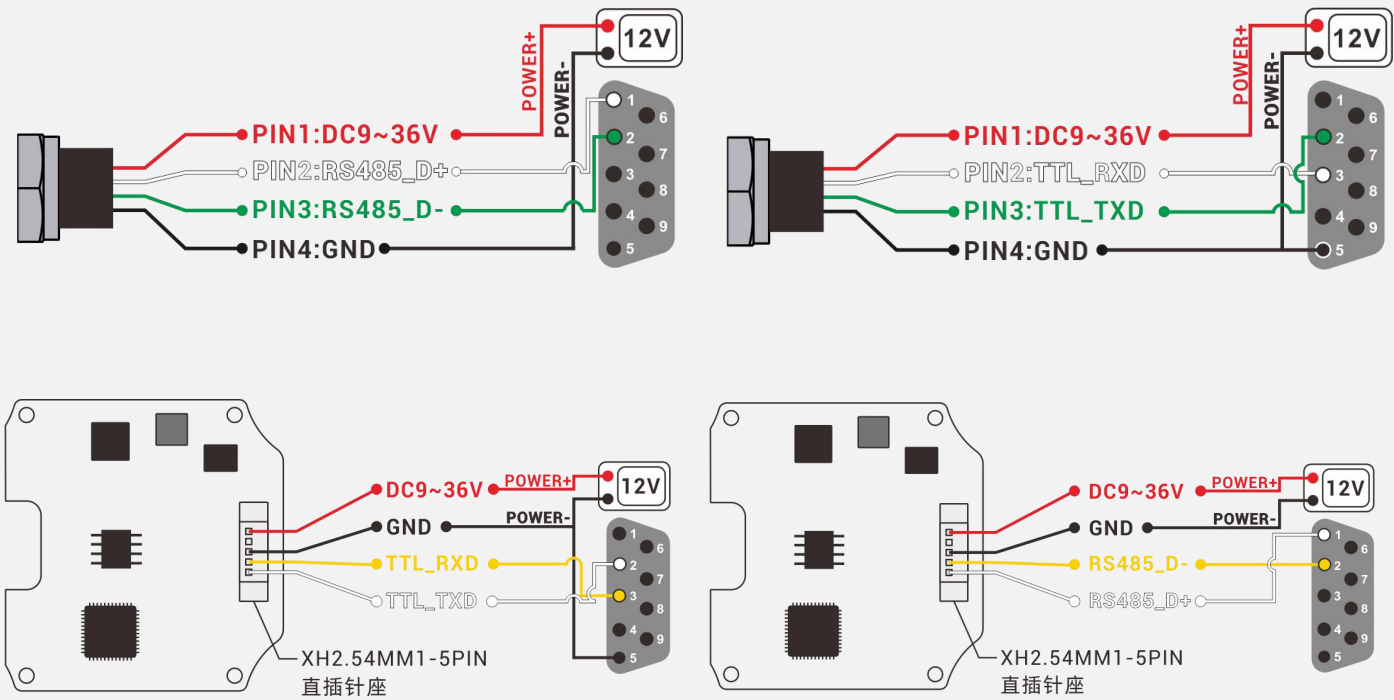
指示灯工作说明

产品上电后红色指示灯常亮，成功建立连接并传输数据时红灯转为闪烁状态。若操作连接过程中红灯持续常亮，表明连接失败，需执行重新连接操作。

接口定义

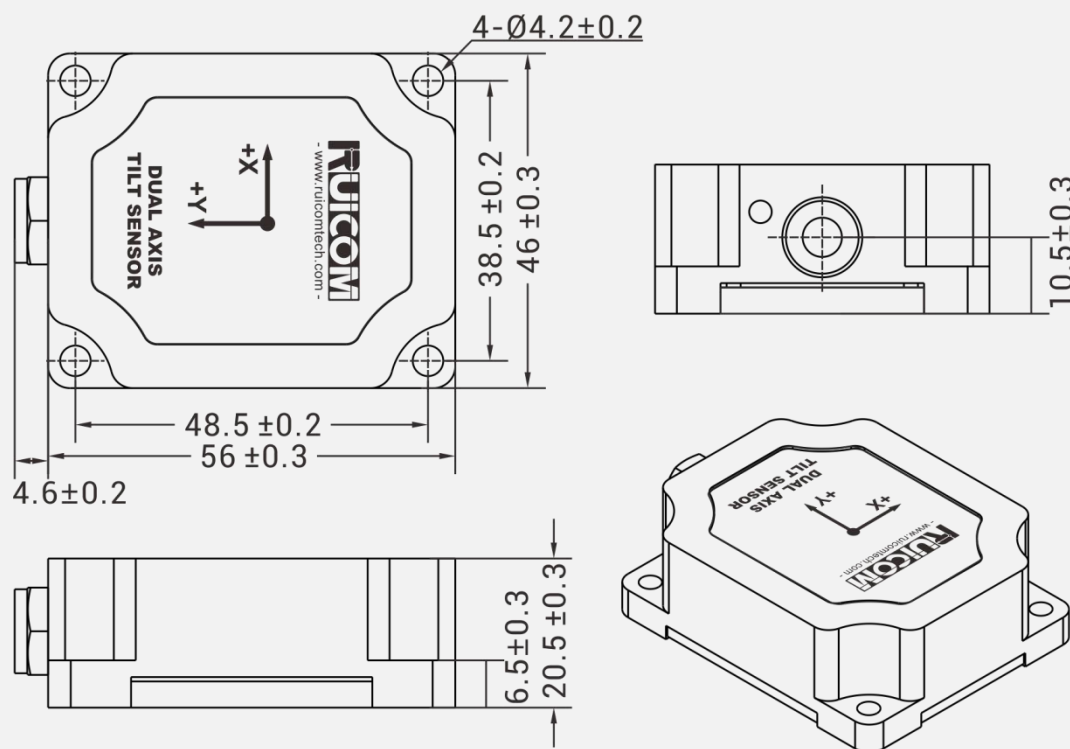
外壳线色	PCBA 线色	定义	描述
红色	红色	DC9~36V	电源正极
白色	白色	TTL_RXD、RS485_D+	接收数据
绿色	黄色	TTL_TXD、RS485_D-	发送数据
黑色	黑色	GND	电源负极

+



产品尺寸

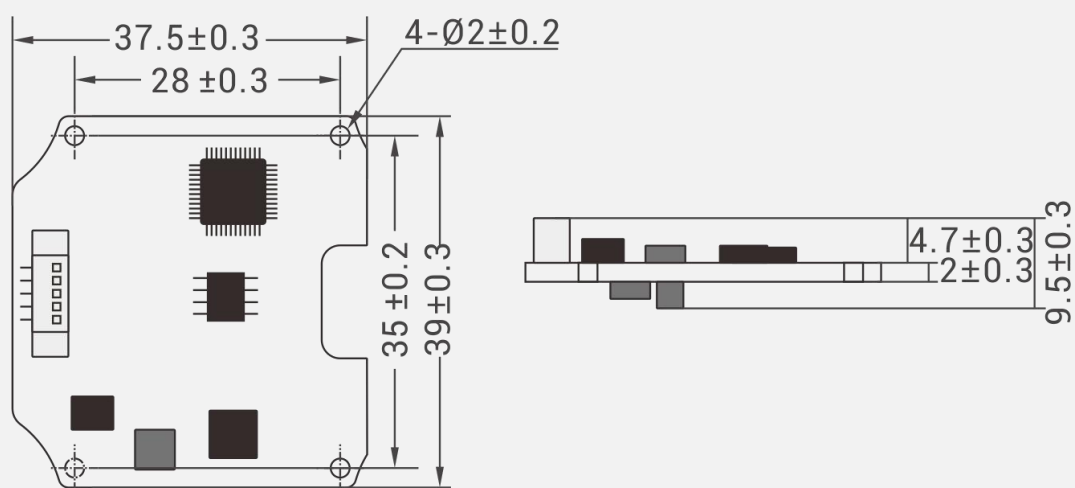
外壳封装尺寸：



外壳尺寸：L60.6×W46×H20.5mm

建议螺丝：4 颗 M4 螺丝

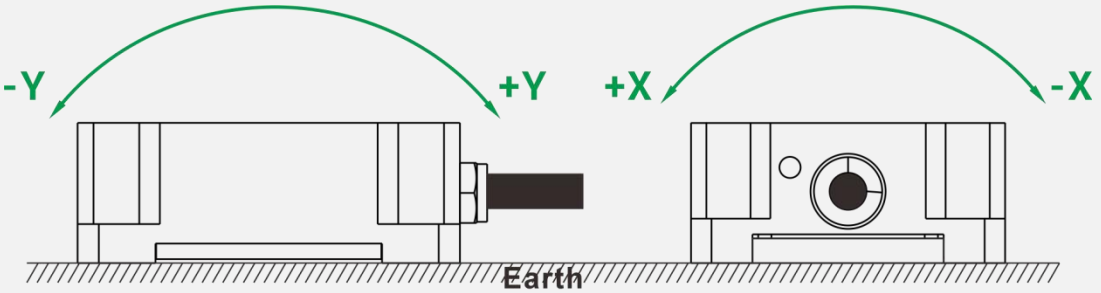
PCBA 尺寸：



外壳尺寸：L37.5×W39×H9.5mm

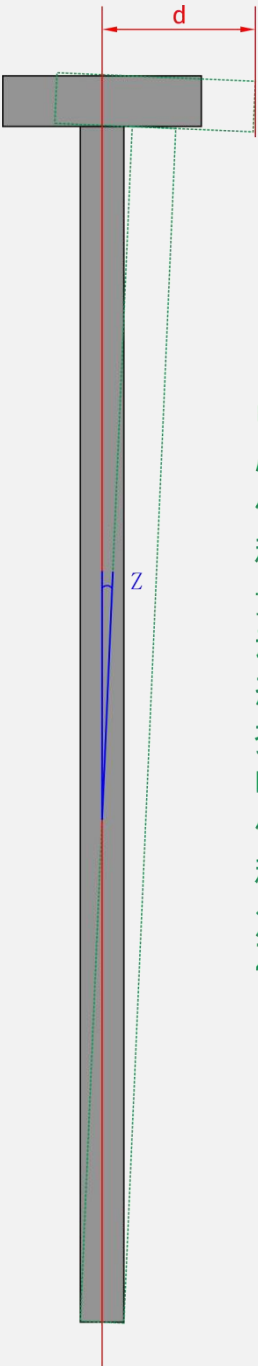
建议螺丝：4 颗 M2 螺丝

测量方向



水平安装

偏移量动态摆幅测量图例



中心偏移量动态摆幅偏移角Z

 MODBUS RTU 通信协议

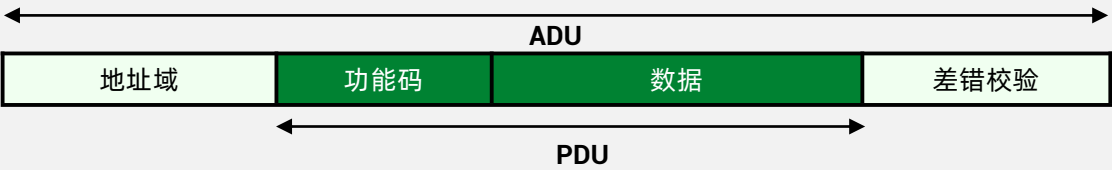
默认参数：校验位：N（无校验），波特率：9600bps。

1. 协议类型：MODBUS 协议

通信格式：1 个起始位，8 个数据位，1 个停止位。

地址范围：1-247，0 为通用广播地址，写命令地址为 0 广播地址时，所有传感器响应命令，但不回复。

设置命令（功能码 06）：所有设置命令操作之前要进行密码认证，设置一次先认证一次；否则无响应。



2. 传输格式

(1)命令报文格式

地址	功能码	数据起始 地址高位	数据起始 地址低位	数据个数 高位	数据个数 低位	CRC16 位校验
高位在前						

发送读数据命令返回：

地址	功能码	字节长度	数据 1 输入	数据 2 输入	...	CRC16 位校验
高位在前	高位在前					

发送写数据命令：

地址	功能码	数据起始 地址高位	数据起始 地址低位	数据高位	数据低位	位校验 CRC16
高位在前						

发送写数据命令返回—命令原样返回：

地址	功能码	数据起始 地址高位	数据起始 地址低位	数据高位	数据低位	位校验 CRC16
高位在前						

(2)异常应答返回

非法功能：

从站地址	功能码	异常码	CRC16 校验
	80H+原功能码	01	

非法数据地址：

从站地址	功能码	异常码	CRC16 校验
	80H+原功能码	02	

非法数据值:

从站地址	功能码	异常码	CRC16 校验
	80H+原功能码	03	

请求数据值为空（未有值）：

从站地址	功能码	异常码	CRC16 校验
	80H+原功能码	0F	

3、模块 RTU 寄存器地址指令及说明信息

寄存器地址	寄存器名称	数据类型	字数	寄存器类型	支持功能码及内容
0x0002-0x0003	密码认证	UINT32	2	WO	0x10(只支持写 2 个寄存器) 对寄存器进行相关设置时，须首先进行密码认证，认证一次设置一次。
0x0004	波特率	UINT16	1	WO	0x06（写单个寄存器） 0x00A0: 2400bps 0x00A1: 4800bps 0x00A2: 9600bps(默认) 0x00A3: 19200bps 0x00A4: 38400bps 0x00A5: 115200bps 设置成功后，上电有效
0x0006	奇偶校验位设置寄存器	UINT16	1	WO	0x06（写单个寄存器）；写入以下内容进行奇偶校验位设置。 1) 0x0000: 偶校验； 2) 0x0001: 无校验（默认）； 3) 0x0002: 奇校验；
0x0007	软件复位寄存器	UINT16	1	WO	0x06（写单个寄存器）；写入以下内容进行软件复位。 1) 0xFFFF: 执行软件复位操作；
0x0008-0x0009	安装高度寄存器（cm）	UINT32	2	R/W	0x03/0x10(读/写 2 个寄存器)； 默认值: 3000cm
0x000D	倾角报警阈值	Uint16	1	R/W	0x03/0x06(读/写单个寄存器)； Angle=uint16*0.01(°) 默认值: 0.50°
0x000E	摆幅报警阈值（mm）	Uint16	1	R/W	0x03/0x06(读/写单个寄存器)； 默认值: 500mm
0x000F	摆动频率报警阈值	Uint16	1	R/W	0x03/0x06(读/写单个寄存器)； Hz = uint16*0.01(hz) 默认值: 0.50Hz
0x0010	水平偏移量报警阈（mm）	Uint16	1	R/W	0x03/0x06(读/写单个寄存器)； 默认值: 500mm

0x0011	Z 角度相对/ 绝对零点设置	Uint16	1	R/W	0x03/0x06(读/写单个寄存器); 0x0000: 绝对零点(默认) ; 0x0001: 相对零点;
0x0012	XY 角度相对/ 绝对零点设置	Uint16	1	R/W	0x03/0x06(读/写单个寄存器); 0x0000: 绝对零点(默认) ; 0x0001: 相对零点;
0x0016-0x0017	X 轴倾斜角度	float	2	RO	0x03(读输出寄存器)
0x0018-0x0019	Y 轴倾斜角度	float	2	RO	0x03(读输出寄存器)
0x001A-0x001B	Z 轴夹角 (Z 轴与中垂 线的夹角)	float	2	RO	0x03(读输出寄存器)
0x001C-0x001D	温度 Tc	float	2	RO	0x03(读输出寄存器)
0x001E	AM 摆动幅度	Uint16 (mm)	1	RO	0x03(读输出寄存器)
0x001F	FR 摆动频率	Uint16 (*0.01Hz)	1	RO	0x03(读输出寄存器) 注意: *0.01 变比(Hz)
0x0020	中心偏移量	Uint16 (mm)		RO	0x03(读输出寄存器)
0x0021	报警状态	Uint16	1	RO	0x03(读输出寄存器) Bit0: 倾角报警; Bit1: 摆动频率报警; Bit2: 摆动幅度报警; Bit3: 中心偏移量报警; Bit = 1 报警; 读取后状态自动清零;
0x0030	传感器地址 (Modbus Address)	UINT16	1	W/R	0x03(支持读单个寄存器); 0x06 (支持写单个寄存器) ; 默认地址: 0x01

3.1 读当前测量值指令

Modbus 功能码 03H, 寄存器首地址: 0x0016, 寄存器数目: 0x000C 读取数据命令应用举例:

主机查询命令:		从机响应:	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器 首地址	00H 16H	数据域长度	18H
寄存器数目	00H 0CH	数据域	BFH 3AH B0H AEH BFH
CRC	A40BH		X 轴数据 (ft32)
			Y 轴数据

			45H	(ft32)
			66H	
			0AH	
			3FH	Z 轴数据 (ft32)
			87H	
			D8H	
			00H	
			41H	T 温度 (ft32)
			F3H	
			16H	
			14H	
			16H	Am 摆幅 (uint16)
			94H	
			02H	freq 振动频率 (uint16)
			36H	
			16H	offset 水平偏移 量(uint16)
			94H	
			00H	报警标志位 (uint16)
			01H	
		CRC	DDF1H	

注：从机回复帧的数据域为 BF 3A B0 AE BF 45 66 0A 3F 87 D8 00 41 F3 16 14 16 94 02 36 16 94 00 01

X 轴为数据域的第 1-4 字节，Y 轴为数据域的第 5-8 字节，Z 轴为数据域的第 9-12 字节，T 温度为数据域的第 13-16 字节，Am 为数据域的第 17-18 字节，freq 为数据域的第 19-20 字节，offset 为数据域的第 21-22 字节，所有数据为大端模式，高字节在前，低字节在后。XYZ 角度和 T 温度为 ft32 浮点存储°, Am, freq 和 offset 为 uint16 存储，Am 和 offset 为 mm, freq 除以 100 后为频率 Hz。

以上述数据帧为例：角度的转换过程如下：

- 1) X 轴为：BF 3A B0 AEH=-0.729°;
- 2) Y 轴为：BF 45 66 0AH=-0.771°;
- 3) Z 轴为：3F 87 D8 00H=1.601°;
- 4) T 温度为：41 F3 16 14H=30.386°;
- 5) Am 摆幅为：16 94H=5780mm;
- 6) Freq 振动频率为 0236H=566，566/100=5.66Hz;
- 7) Offset 水平偏移量为：1694H=5780mm;
- 8) 报警标志：0001H 有倾角报警;

3.2 地址码指令

3.2.1 读地址指令 寄存器地址：0030H 寄存器数目：0001H

主机查询命令：		从机响应：		
传感器地址	00H	传感器地址	01H	
功能码	03H	功能码	03H	
访问寄存器地址	00H	数据域长度	02H	
	30H	数据域	00H	传感器地址
寄存器数目	00H		01H	
	01H	CRC	7984H	
CRC	85D4H			

主机发送的命令帧：

Addr	03h	0h	30h	0h	1h	CRC16L	CRC16H
------	-----	----	-----	----	----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1~Data2	CRC16L	CRC16H
------	-----	----------	-------------	--------	--------

指令返回数据为 2 字节整型数据，模块地址编码 1~247；0x00 是广播地址码。

可用作读地址码指令的前始地址，读取模块读取地址码。

发送时 Addr 使用 0x00，返回时，Addr 和 Data1~Data2 都是模块当前地址码。

Send: 00 03 00 30 00 01 85 D4

Recv: 01 03 02 00 01 79 84

3.2.2 设置地址码指令

设置传感器地址码命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	30H		30H
传感器的新地址	00H	传感器的新地址	00H
	04H		04H
CRC	8806	CRC	8806

主机发送的命令帧：

Addr	06h	0h	30h	0h	NewAddr	CRC16L	CRC16H
------	-----	----	-----	----	---------	--------	--------

模块返回的数据帧：

NewAddr	06h	0h	30h	0h	NewAddr	CRC16L	CRC16H
---------	-----	----	-----	----	---------	--------	--------

NewAddr：模块地址编码 1~247。Addr 为 0x00 时，修改地址成功后，无回复。

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：

设置 ID 功能码 06 寄存器地址 30 字节数 2

Send: 01 06 00 30 00 04 88 06

Recv: 01 06 00 30 00 04 88 06

3.3 MBRTU 模式密码认证指令

寄存器地址：0002H 寄存器数目：0002H

MBRTU 模式密码认证命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	10H	功能码	10H
寄存器地址	00H	寄存器地址	00H
	02H		02H
数据字长	00H	数据字长	00 H
	02H		02H
数据字节数	04H	CRC	E008
数据	50H		
	53H		
	57H		
	44H		
CRC	AD64		

主机发送的命令帧：

Addr	10h	0h	02h	0h	2h	04h	Data1~Data4	CRC16L	CRC16H
------	-----	----	-----	----	----	-----	-------------	--------	--------

模块返回的数据帧：

Addr	10h	0h	02h	0h	02h	CRC16L	CRC16H
------	-----	----	-----	----	-----	--------	--------

Data1~Data4: 50 53 57 44。

设置成功返回以上指令，否则出错码。

注：MBRTU 模式，所有写指令和调校指令都需密码认证。

示例：

功能：密码认证

发送：01 10 00 02 00 02 04 50 53 57 44 AD 64

接收：01 10 00 02 00 02 E0 08

密码认证- 成功！

功能：修改地址 地址 1 修改 2

发送：01 06 00 30 00 02 08 04

接收：02 06 00 30 00 02 08 37

地址已修改为：【2】

3.4 安装高度

3.4.1 设置安装高度(默认高度为：3000cm)

寄存器地址：0008H 寄存器数目：0002H

设置安装高度命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	10H	功能码	10H
寄存器地址	00H	寄存器地址	00H
	08H		08H
数据字长	00H	数据字长	00H

	02H		02H
数据字节数	04H	CRC	C00A
高度数据(cm)	00H		
	00H		
	0BH		
	B8H		
CRC	F54B		

高度数据(cm)为 uint32: 00 00 0B B8=3000cm

主机发送的命令帧:

Addr	10h	0h	08h	00h	02h	04h	Data1~Data4	CRC16L	CRC16H
------	-----	----	-----	-----	-----	-----	-------------	--------	--------

模块返回的数据帧:

Addr	10h	0h	08h	0h	02h	CRC16L	CRC16H
------	-----	----	-----	----	-----	--------	--------

发送: 01 10 00 08 00 02 04 00 00 0B B8 F5 4B

接收: 01 10 00 08 00 02 C0 0A

高度数据(cm)Data1~Data4 为 uint32: 00 00 0B B8=3000cm。

设置成功返回以上指令, 否则无响应。

3.4.2 读取安装高度

主机查询命令:			从机响应:		
传感器地址	01H		传感器地址	01H	
功能码	03H		功能码	03H	
访问寄存器地址	00H		数据域字节数	04H	
	08H		高度数据域(cm)	00H	
寄存器数目	00H			00H	
	02H			0BH	
CRC	45C9H			B8H	
			CRC	FD71H	

注意: 高度数据为 4 字节的 uint32; 高字节在前, 低字节在后。

主机发送的命令帧:

Addr	03h	0h	08h	0h	02h	CRC16L	CRC16H
------	-----	----	-----	----	-----	--------	--------

模块返回的数据帧:

Addr	03h	Length=4	Data1 ~ Data4	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

指令返回数据为 4 字节的 uint32; 高字节在前, 低字节在后。

3.5 XY 角度相对/绝对零点指令

3.5.1 查询当前 XY 角度的零点状态指令

寄存器地址: 0012H 寄存器数目: 0001H

主机查询命令:			从机响应:		
传感器地址	01H		传感器地址	01H	
功能码	03H		功能码	03H	

访问寄存器地址	00H	数据域长度	02H
	12H	数据域	00H
寄存器数目	00H		01H
	01H	CRC	7984H
CRC	240FH		

注意：数据域：0000H：绝对零点； 0001H：相对零点；

例：主机发送的命令帧：

Addr	03h	00h	12h	00h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

Send: 01 03 00 12 00 01 24 0F

Recv: 01 03 02 00 01 79 84

3.5.2 设置 XY 角度绝对与零点指令

设置 XY 角度绝对与零点命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	12H		12H
数据内容	00H	数据内容	00H
	01H		01H
CRC	E80FH	CRC	E80FH

注意：数据内容：0x0000：绝对零点 0x0001：相对零点

主机发送设置相对零点命令帧：

Addr	06h	0h	12h	00h	01h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	12h	00h	01h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令，否则无响应。

示例：

Send: 01 06 00 12 00 01 E8 0F

Recv: 01 06 00 12 00 01 E8 0F

3.6 Z 角度相对/绝对零点指令

3.6.1 查询当前 Z 轴角度零点状态指令

寄存器地址：0011H 寄存器数目：0001H

主机查询命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器地址	00H	数据域长度	02H
	11H	数据域	00H

寄存器数目	00H	CRC	00H
	01H		B844H
CRC	D40FH		

注意：数据域：0000H：绝对零点； 0001H：相对零点；

主机发送的命令帧：

Addr	03h	00h	11h	00h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

Send: 01 03 00 11 00 01 D4 0F

Recv: 01 03 02 00 00 B8 44

3.6.2 设置当前 Z 轴零点指令

设置当前 Z 轴零点命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	11H		11H
数据内容	00H	数据内容	00 H
	01H		01H
CRC	180FH	CRC	180FH

注意：数据内容：0x0000:绝对零点 0x0001：相对零点

主机发送设置相对零点命令帧：

Addr	06h	0h	11h	00h	01h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	11h	00h	01h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：

Send: 01 06 00 11 00 01 18 0F

Recv: 01 06 00 11 00 01 18 0F

3.7 设置通讯波特率指令

寄存器地址：0004h 寄存器长度：0001H

设置通讯波特率命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	04H		04H
数据内容	00H	数据内容	00 H
	A1H		A1H
CRC	09CBH	CRC	09CBH

数据内容：0x00A0: 2400bps 0x00A1: 4800bps 0x00A2: 9600bps
 0x00A3: 19200bps 0x00A4: 38400bps 0x00A5: 115200bps

注意：重新上电有效。

主机发送设置波特率命令帧：

Addr	06h	0h	04h	00h	A1h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	04h	00h	A1h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：

Tx: 01 06 00 04 00 A1 09 CB

Rx: 01 06 00 04 00 A1 09 CB

3.8 设置通讯奇偶校验位指令

寄存器地址：0006h 寄存器长度：0001H

设置通讯奇偶校验位命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	06H		06H
数据内容	00H	数据内容	00H
	00H		00H
CRC	69CBH	CRC	69CBH

数据内容：0x0000：偶校验 0x0001：无校验 0x0002：奇校验

注意：重新上电有效。

主机发送设置奇偶校验位命令帧：

Addr	06h	0h	06h	00h	00h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	06h	00h	00h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：

Send: 01 06 00 06 00 00 69 CB

Recv: 01 06 00 06 00 00 69 CB

3.9 设置倾角报警阈值指令

3.9.1 查询当前倾角报警阈值指令

寄存器地址：000DH 寄存器数目：0001H

主机查询命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器	00H	数据域长度	02H

地址	0DH	数据域	00H
寄存器数目	00H		14H
	01H	CRC	B84BH
CRC	15C9H		

注意：数据域：0014H = 20/100= 0.20°；

默认报警角度为：0.5°。

主机发送的命令帧：

Addr	03h	00h	0Dh	00h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

TX: 01 03 00 0D 00 01 15 C9

RX: 01 03 02 00 14 B8 4B

3.9.2 设置倾角报警阈值指令

设置倾角报警阈值命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	0DH		0DH
数据内容	00H	数据内容	00H
	32H		32H
CRC	99DCH	CRC	99DCH

注意：数据内容：0x0032 = 50/100=0.50°

主机发送命令帧：

Addr	06h	0h	0Dh	00h	32h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	0Dh	0Dh	32h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：TX: 01 06 00 0D 00 32 99 DC

RX: 01 06 00 0D 00 32 99 DC

3.10 设置摆幅报警阈值指令

3.10.1 查询当前摆幅报警阈值指令

寄存器地址：000EH 寄存器数目：0001H

主机查询命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器地址	00H	数据域长度	02H
寄存器数目	0EH	数据域	00H
	00H		64H

	01H	CRC	B9AFH
CRC	E5C9H		

注意：数据域：0064H = 100mm；

默认摆幅报警阈值为：500mm

主机发送的命令帧：

Addr	03h	00h	0Eh	00h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

TX: 01 03 00 0E 00 01 E5 C9

RX: 01 03 02 00 64 B9 AF

3.10.2 设置摆幅报警阈值指令

设置摆幅报警阈值命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器	00H
	0EH	地址	0EH
数据内容	01H	数据内容	01H
	F4H		F4H
CRC	E81EH	CRC	E81EH

注意：数据内容：0x01F4 = 500mm

主机发送命令帧：

Addr	06h	0h	0Eh	01h	F4h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	0Eh	01h	F4h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令，否则无响应。

示例：TX: 01 06 00 0E 01 F4 E8 1E

RX: 01 06 00 0E 01 F4 E8 1E

3.11 设置摆动频率报警阈值指令

3.11.1 查询当前摆动频率报警阈值指令

寄存器地址：000FH 寄存器数目：0001H

主机查询命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器	00H	数据域长度	02H
地址	0FH	数据域	00H
寄存器数目	00H		14H
	01H	CRC	B84BH
CRC	B409H		

注意：数据域：0014H = 20/100 = 0.20Hz；

默认摆动频率报警阈值为：0.5Hz;

主机发送的命令帧：

Addr	03h	00h	0Fh	00h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

TX: 01 03 00 0F 00 01 B4 09

RX: 01 03 02 00 14 B8 4B

3.11.2 设置摆动频率报警阈值指令

设置摆动频率报警阈值命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	0FH		0FH
数据内容	00H	数据内容	00H
	32H		32H
CRC	381CH	CRC	381CH

注意：数据内容：0x0032 = 50 = 0.50Hz

主机发送命令帧：

Addr	06h	0h	0Fh	00h	32h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	0Fh	00h	32h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：TX: 01 06 00 0F 00 32 38 1C

RX: 01 06 00 0F 00 32 38 1C

3.12 设置水平偏移量报警阈值指令

3.12.1 查询当前偏移量报警阈值指令

寄存器地址：0010H 寄存器数目：0001H

主机查询命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	03H	功能码	03H
访问寄存器地址	00H	数据域长度	02H
寄存器数目	10H	数据域	01H
	00H		F4H
	01H	CRC	B853H
CRC	85CFH		

注意：数据域：01F4H = 500mm;

默认水平偏移量报警阈值为：500mm

主机发送的命令帧：

Addr	03h	00h	10h	01h	01h	CRC16L	CRC16H
------	-----	-----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	03h	Length=2	Data1 ~ Data2	CRC16L	CRC16H
------	-----	----------	---------------	--------	--------

TX: 01 03 00 10 00 01 85 CF

RX: 01 03 02 01 F4 B8 53

3.12.2 设置偏移量报警阈值指令

设置偏移量报警阈值命令：		从机响应：	
传感器地址	01H	传感器地址	01H
功能码	06H	功能码	06H
地址	00H	寄存器地址	00H
	10H		10H
数据内容	00H	数据内容	00H
	64H		64H
CRC	89E4H	CRC	89E4H

注意：数据内容：0x01F4 = 500mm

主机发送命令帧：

Addr	06h	0h	10h	00h	64h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

模块返回的数据帧：

Addr	06h	0h	10h	00h	64h	CRC16L	CRC16H
------	-----	----	-----	-----	-----	--------	--------

操作前需密码认证，操作成功返回原指令， 否则无响应。

示例：TX: 01 06 00 10 00 64 89 E4

RX: 01 06 00 10 00 64 89 E4

● CRC 码的计算方法是：

1. 预置 1 个 16 位的寄存器为十六进制 FFFF（即全为 1）；称此寄存器为 CRC 寄存器；
2. 把第一个 8 位二进制数据（既通讯信息帧的第一个字节）与 16 位的 CRC 寄存器的低 8 位相异或，把结果放于 CRC 寄存器；
3. 把 CRC 寄存器的内容右移一位（朝低位）用 0 填补最高位，并检查右移后的移出位；
4. 如果移出位为 0：重复第 3 步（再次右移一位）；
- 如果移出位为 1：CRC 寄存器与多项式 A001（1010 0000 0000 0001）进行异或；
5. 重复步骤 3 和 4，直到右移 8 次，这样整个 8 位数据全部进行了处理；
6. 重复步骤 2 到步骤 5，进行通讯信息帧下一个字节的处理；
7. 将该通讯信息帧所有字节按上述步骤计算完成后，得到的 16 位 CRC 寄存器的高、低字节进行交换；
8. 最后得到的 CRC 寄存器内容即为：CRC 码。

unsigned char const 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
};
```

```
unsigned char const 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
};
```

```
unsigned short MB_Make_CRC16( unsigned char *puchMsg, unsigned short usDataLen )
{
    unsigned char uchCRCHi = 0xFF;
    unsigned char uchCRCLo = 0xFF;
    unsigned uIndex;
    while (usDataLen--)
    {
        uIndex = uchCRCLo^*puchMsg++;
        uchCRCLo = uchCRCHi^auchCRCHi[uIndex];
        uchCRCHi = auchCRCLo[uIndex];
    }
    return (uchCRCHi<<8|uchCRCLo);
}
```