高精度、高稳定性、低噪声和
振动抑制特性的移动机器人全场定位模块

VEGA-V1.0

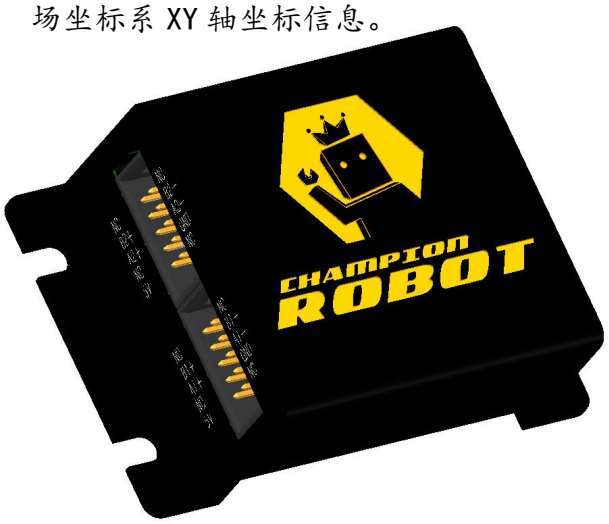
产品特性

陀螺仪+两路正交编码器——全场定位
可直接输出全场定位坐标信息
陀螺仪长时零漂：0.06°
陀螺仪零偏稳定度：0.05°
陀螺仪测量精度：0.09°/round
两路标准 ABZ 信号正交编码器接口
电源电压：16~26V 宽电压供电
可使用串口根据实际环境进行参数矫正
CAN 总线读取模块数据
体积小、质量轻

概述

“VEGA”是一款移动机器人（移动平台）全场定位传感检测系统的核心部件，用于实时获取机器人精确航向角度及全场坐标，实现全场定位、路径跟踪、路径规划等智能导航控制。

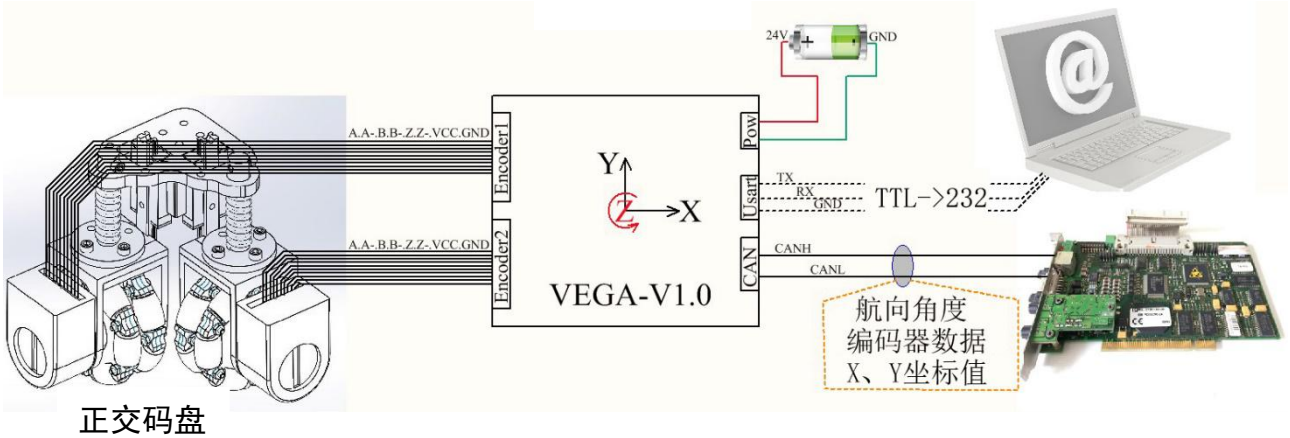
“VEGA”传感定位模块内部集成了一颗高精度单轴陀螺仪和双通道正交编码器接口以及位姿解算 CPU。机器人主控系统可通过 CAN 总线与本模块进行高速、稳定数据传输，可直接获取航向角度、两路编码器数据、全场坐标系 XY 轴坐标信息。



应用

室内移动机器人定位
竞技机器人全场定位
其他高精度移动平台定位系统

应用示例



正交码盘

文档反馈
E-Mail:service@champion-robot.com

联系方式：
成都电科创品机器人科技有限公司
咨询客服 QQ:1147927964

技术规格

除非另有说明，测试条件为：室温 25℃，电源电压 24V，工作电流 45mA，角速度 0°/S，加速度 1g。

表 1.

参　　数	测试条件/说明	Min	Typ	Max	Unit
角速度范围	角度检测失真临界值	0		±250	°/s
测量精度	陀螺仪以各转速旋转 360° 角度误差平均值	0.02	0.09	0.24	°/r
角度零漂	陀螺仪静止 1h 角度漂移平均值	0.01	±0.06	±0.3	°
角度温漂	陀螺仪升温 29-47° C 角度漂移平均值	*见陀螺仪温度-漂移关系曲线			
非线性度	最佳拟合直线	0.01			%of FS
数据输出率	CAN 总线读取数据	200			Hz
启动时间	初始化-数据输出	2			s
电源电压	电压纹波≤200mV	16	24	26	V
电源电流		80.35	55.44	51.84	mA
温度范围	*以芯片手册为准	-40	25	105	° C

注：该表中参数均为实验室环境下测试结果，实际系统可能存在较小差异。

绝对最大额定值

表 2.

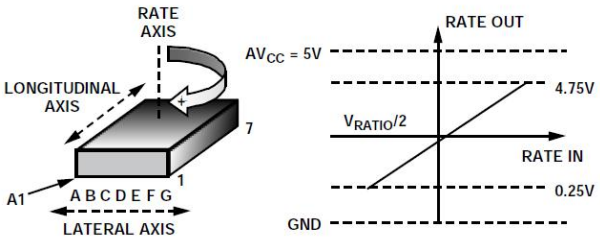
参　　数		最大额定值
加速度冲击	未上电	10,000g
	上电	10,000g
电源电压		30V
工作温度范围		-55° C 至+125° C
存储温度范围		-65° C 至+150° C

注意，超出上述绝对最大额定值可能会导致器件永久性损坏。不能以这些条件或者在任何其它超出本技术规范操作章节中所示规格的条件下，推断器件能否正常工作。长期在绝对最大额定值条件下工作会影响器件的可靠性。

掉在坚硬表面上可能会引起高于 10,000g 的冲击，甚至超过器件绝对最大额定值。搬运时应小心，避免损坏器件。

角度敏感轴

这是 z 轴速率检测器件(也称为航向角速度检测器件)。当它绕封装顶部的法线轴(即俯视封装盖)顺时针旋转时，可产生正输出角度。



ESD警告

ESD(静电放电)敏感器件。

带电器件和电路板可能会在没有察觉的情况下放电。尽管本产品具有专利或专有保护电路，但在遇到高能量ESD时，器件可能会损坏。因此，应当采取适当的ESD防范措施，以避免器件性能下降或功能丧失。

接口配置和功能描述

接口功能描述

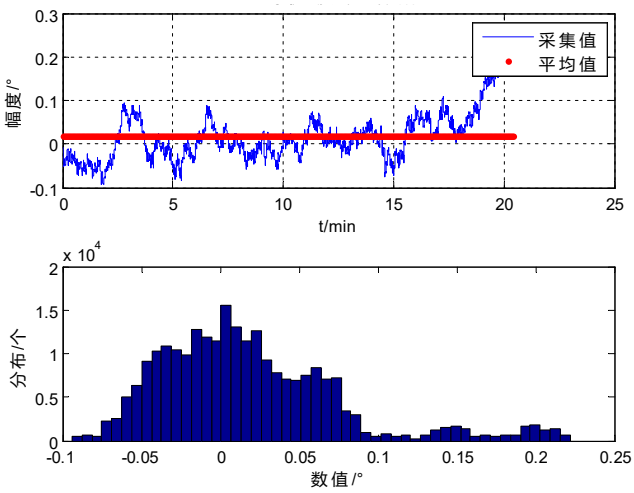
接口名称	引脚	功能描述
Encoder1	A1+	编码器 1 差分输出：A 通道(推荐使用 HEDL-5540 500 线编码器)
	A1-	
	B1+	编码器 1 差分输出：B 通道(推荐使用 HEDL-5540 500 线编码器)
	B1-	
	5V	编码器 1 输入电压：5V
	GND	编码器 1 接地：GND
	NC	预留引脚
Encoder2	A2+	编码器 2 差分输出：A 通道(推荐使用 HEDL-5540 500 线编码器)
	A2-	
	B2+	编码器 2 差分输出：B 通道(推荐使用 HEDL-5540 500 线编码器)

	B2-	
	5V	编码器 2 输入电压：5V

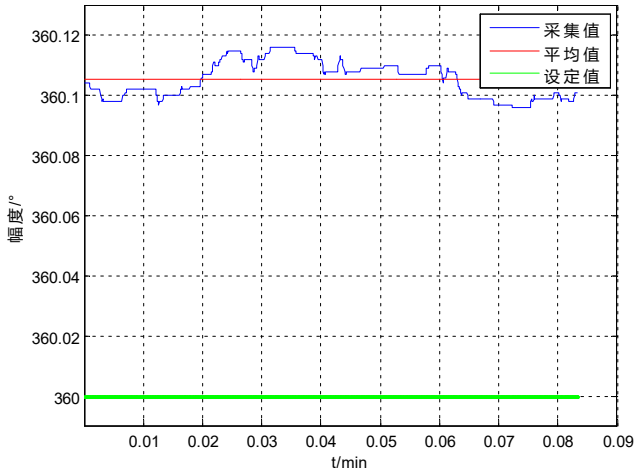
续上表

	GND	编码器 2 接地：GND
	NC	预留引脚
POWER & CAN	GND	电源地
	24V	电源电压（16V-26V）
	CANH	CAN_H （两组 CAN 接口连接在同一总线，仅方便转接使用）
	CANL	CAN_L （两组 CAN 接口连接在同一总线，仅方便转接使用）
	CANH	CAN_H （两组 CAN 接口连接在同一总线，仅方便转接使用）
	CANL	CAN_L （两组 CAN 接口连接在同一总线，仅方便转接使用）
USART	GND	串口地
	3V3	串口供电电压
	RX	串口 RX
	TX	串口 TX

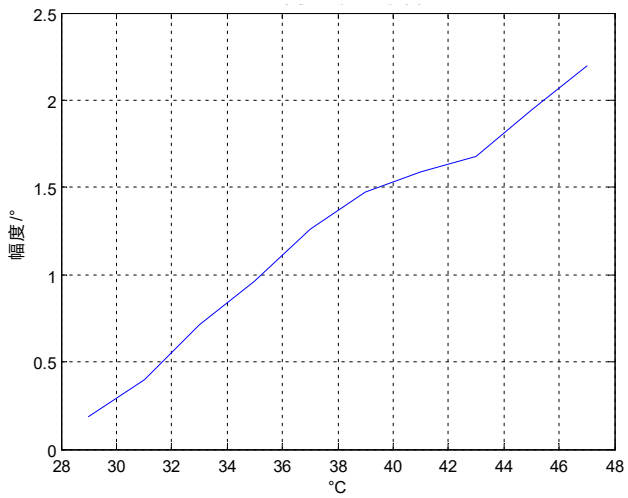
典型性能参数



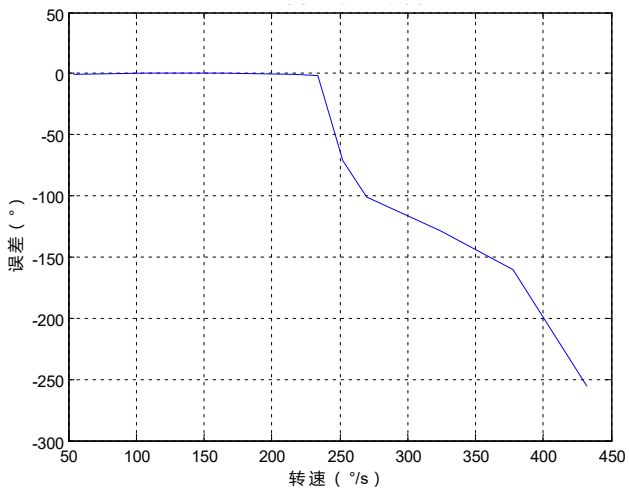
陀螺仪零漂性能



陀螺仪精度测试数据

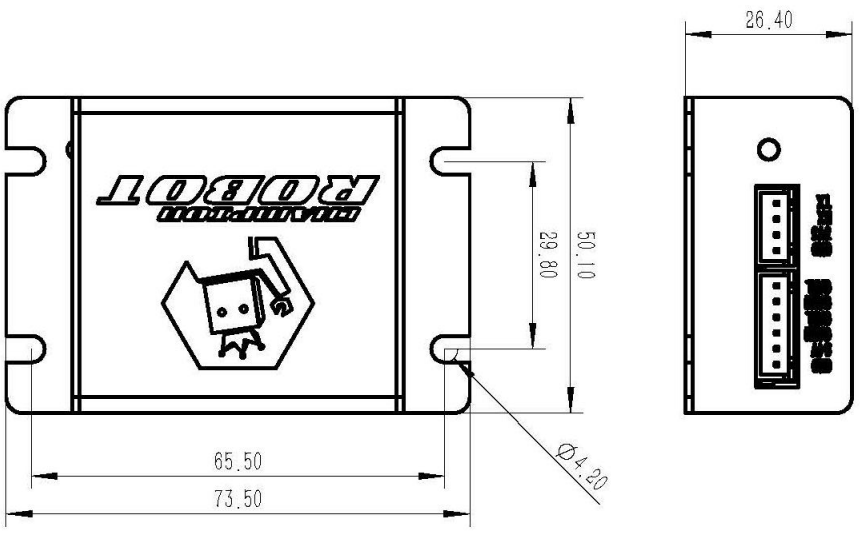


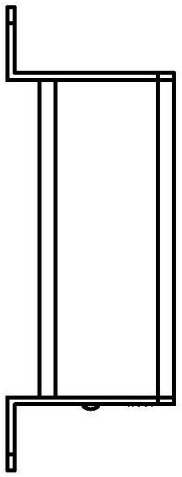
陀螺仪温度-漂移曲线



陀螺仪最大角速度-误差曲线

外形尺寸





名称	移动机器人全场定位模块		
出品公司	成都电科创品机器人科技有限公司		
比例	1：1	型号	Vega-v1.0
重量	60g	日期	2016年3月

软件及使用说明

1 通信概述:

本品采用 CAN 通信，用户需对 CAN 通信协议有一定的了解。用户可直接读取当前角度及坐标值（x,y）。也可以发送相应的指令来设置当前角度及坐标以适应需求。并且控制器可复位本模块。

通信协议如下:

通信速率	1Mbps
时间特性设置建议	同步段：1Tq 传播时间段+相位缓冲段 1：9Tq 相位缓冲段 2：4Tq
本模块 ID(CAN)	0x11
用户控制器 ID(CAN)	0x12

注：本模块 ID，用户控制器 ID 都是可通过指令重新配置的，详见 POS 指令集。

2 读取角度，坐标值

2.1.通过 CAN 接收中断,接收 ID 为 0x11 的消息(Rx_Message)。Rx_Message 里的数据即为角度，坐标信息。

2.2. 通过判断 Rx_Message 的数据长度（DLC）来区分是角度信息还是坐标。If(DLC == 4) Data[4]里存的是角度信息，else Data[8]存储的是坐标信

息，前四字节为 x 坐标值，后四字节为 y 坐标值。

2.3 示例：

```
void CAN2_RX0_IRQHandler(void)
{
    DataConvertTypeDef temp;
    if (CAN_GetITStatus(CAN2,CAN_IT_FMP0) != RESET)
    {
        CAN_Receive(CAN2, CAN_FIFO0, &RxMessage);
        switch (RxMessage.StdId)
        {
               case POSITION_ID_POS:
                if(RxMessage.DLC == 8)
                {
                  temp.u8_form[0] = RxMessage.Data[0];
                  temp.u8_form[1] = RxMessage.Data[1];
                  temp.u8_form[2] = RxMessage.Data[2];
                  temp.u8_form[3] = RxMessage.Data[3];
                  memcpy((void*)&G_Param.original_pos_x,
             &temp.s32_form, 4);//得到了 X 坐标
                  temp.u8_form[0] = RxMessage.Data[4];
                  temp.u8_form[1] = RxMessage.Data[5];
                  temp.u8_form[2] = RxMessage.Data[6];
                  temp.u8_form[3] = RxMessage.Data[7];
                  memcpy((void *)&G_Param.original_pos_y,
             &temp.s32_form, 4); //得到了 Y 坐标
                  G_Param.pos_rec_cnt++;
                }
                else
                {
                  temp.u8_form[0] = RxMessage.Data[0];
                  temp.u8_form[1] = RxMessage.Data[1];
                  temp.u8_form[2] = RxMessage.Data[2];
                  temp.u8_form[3] = RxMessage.Data[3];
                  memcpy((void *)&G_Param.original_pos_a,
             &temp.float_form, 4);//得到了角度值
                  G_Param.gyro_rec_cnt++;
                }
                break;
        default : break;
        }
          CAN_ClearITPendingBit(CAN2, CAN_IT_FMP0);
```

```
        }
    }
```

3 设置当前角度(基值)

3.1 设置角度只需要将角度信息以一定格式(CAN 的 ID,DLC 等)将要设置的角度打包到一个 CAN 邮箱，然后发送出去就 OK。

3.2 示例

```
Angle_float_to_char.float_form=angle;//angle 即为要发送的 float 型数据。
TxMessage.StdId = 0x12;//与控制器 ID 一致，如果控制器 ID 改变，修改这里。
TxMessage.DLC = 4;//必须为 4
TxMessage.Data[0]=Angle_float_to_char.uchar_form[0];
TxMessage.Data[1]=Angle_float_to_char.uchar_form[1];
TxMessage.Data[2]=Angle_float_to_char.uchar_form[2];
TxMessage.Data[3]=Angle_float_to_char.uchar_form[3];
```

内容及格式打包完毕，发送即可。

4 设置当前坐标(基值)

4.1 设置坐标原理同角度设置。

4.2 示例

```
s32 xx=0,yy=0;
xx=car_x;    //car_x 即为要发送的 x 坐标
yy=car_y;    //car_y 即为要发送的 y 坐标
TxMessage.StdId = 0x12;//与控制器 ID 一致，如果控制器 ID 改变，修改这里。
TxMessage.DLC = 8;//必须为 8
TxMessage.Data[0]=(u8)(xx>>0);
TxMessage.Data[1]=(u8)(xx>>8);
TxMessage.Data[2]=(u8)(xx>>16);
TxMessage.Data[3]=(u8)(xx>>24);
TxMessage.Data[4]=(u8)(yy>>0);
TxMessage.Data[5]=(u8)(yy>>8);
TxMessage.Data[6]=(u8)(yy>>16);
TxMessage.Data[7]=(u8)(yy>>24);
```

内容及格式打包完毕，发送即可。

5 复位此陀螺仪模块

5.1 通过控制器发送 DLC = 2 且 Data[0] = 0x55,Data[1] = 0xff 的邮箱包来复位陀螺仪。

5.2 示例

```
TxMessage.StdId = 0x12;//如果控制器 ID 改变，修改这里。
TxMessage.DLC = 2;//必须为 2
TxMessage.Data[0]=0x55;
TxMessage.Data[1]=0xff;
```

内容及格式打包完毕，发送即可。

附件 1：相对完整的 CAN 读取及控制代码

以 STM32F4 系列芯片为例：

```
#include <string.h>
//数据存储变量
float angle = 0;
int32_t pos_x = 0,pos_y = 0;
//数据转换结构体：
typedef union
{
    uint8_t  u8_form[4];
    int32_t  s32_form;
    float    float_form;
}DataConvertTypeDef;
//CAN2 配置函数：
void Can_Init(void)
{
    GPIO_InitTypeDef       Gpio_Structure;
    CAN_InitTypeDef        Can_Structure;
    NVIC_InitTypeDef       NVIC_InitStructure;
    CAN_FilterInitTypeDef  Can_filter_Structure;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1| RCC_APB1Periph_CAN2, ENABLE);
    GPIO_PinAFConfig(GPIOB, GPIO_PinSource5, GPIO_AF_CAN2);
    GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_CAN2);
    Gpio_Structure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6 ;
    Gpio_Structure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_Init(GPIOB, &Gpio_Structure);
    NVIC_InitStructure.NVIC_IRQChannel = CAN2_RX0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    CAN_ITConfig(CAN2,CAN_IT_FMP0,ENABLE);
```

```
    CAN_StructInit(&Can_Structure);
    Can_Structure.CAN_TTCM = DISABLE;
    Can_Structure.CAN_ABOM = DISABLE;
    Can_Structure.CAN_AWUM = DISABLE;
    Can_Structure.CAN_NART = DISABLE;
    Can_Structure.CAN_RFLM = DISABLE;
    Can_Structure.CAN_TXFP = ENABLE;
    Can_Structure.CAN_Mode = CAN_Mode_Normal;
    Can_Structure.CAN_SJW  = CAN_SJW_1tq;
    Can_Structure.CAN_BS1 = CAN_BS1_9tq;
    Can_Structure.CAN_BS2 = CAN_BS2_4tq;
    Can_Structure.CAN_Prescaler = 3;
    CAN_Init(CAN2, &Can_Structure);
    Can_filter_Structure.CAN_FilterNumber=14;
    Can_filter_Structure.CAN_FilterMode=CAN_FilterMode_IdMask;
    Can_filter_Structure.CAN_FilterScale=CAN_FilterScale_32bit;
    Can_filter_Structure.CAN_FilterIdHigh=0x0000;
    Can_filter_Structure.CAN_FilterIdLow=0x0000;
    Can_filter_Structure.CAN_FilterMaskIdHigh=0x0000;
    Can_filter_Structure.CAN_FilterMaskIdLow=0x0000;
    Can_filter_Structure.CAN_FilterFIFOAssignment=0;
    Can_filter_Structure.CAN_FilterActivation=ENABLE;
    CAN_FilterInit(&Can_filter_Structure);
}
//CAN2 接收中断:
void CAN2_RX0_IRQHandler(void)
{
    DataConvertTypeDef temp;
    CanRxMsg RxMessage;
    if (CAN_GetITStatus(CAN2,CAN_IT_FMP0) != RESET)
    {
        CAN_Receive(CAN2, CAN_FIFO0, &RxMessage);
        switch (RxMessage.StdId)
        {
            case POSITION_ID_POS:
                if(RxMessage.DLC == 8)
                {
                    temp.u8_form[0] = RxMessage.Data[0];
                    temp.u8_form[1] = RxMessage.Data[1];
                    temp.u8_form[2] = RxMessage.Data[2];
                    temp.u8_form[3] = RxMessage.Data[3];
                    memcpy((void *)&pos_x, &temp.s32_form, 4);
                    temp.u8_form[0] = RxMessage.Data[4];
                    temp.u8_form[1] = RxMessage.Data[5];
                    temp.u8_form[2] = RxMessage.Data[6];
```

```
                temp.u8_form[3] = RxMessage.Data[7];
                memcpy((void *)&pos_y, &temp.s32_form, 4);
            }
            else if(RxMessage.DLC == 4)
            {
                temp.u8_form[0] = RxMessage.Data[0];
                temp.u8_form[1] = RxMessage.Data[1];
                temp.u8_form[2] = RxMessage.Data[2];
                temp.u8_form[3] = RxMessage.Data[3];
                memcpy((void *)&angle, &temp.float_form, 4);
            }
        break;
        default:break;
      }
      CAN_ClearITPendingBit(CAN2, CAN_IT_FMP0);
   }
}
//设置角度
void setAngle(float angle)
{
    DataConvertTypeDef temp;
    CanTxMsg TxMessage;
    temp.float_form = angle;
   TxMessage.StdId = 0x12;//如果控制器 ID 改变，修改这里。
   TxMessage.DLC = 4;//必须为 4
   TxMessage.IDE=CAN_Id_Standard;
   TxMessage.RTR=CAN_RTR_Data;
   TxMessage.Data[0]=temp.u8_form[0];
   TxMessage.Data[1]=temp.u8_form[1];
   TxMessage.Data[2]=temp.u8_form[2];
   TxMessage.Data[3]=temp.u8_form[3];
   CAN_Transmit(CAN2, &TxMessage);
}
//设置 x,y 坐标
void setPosition(s32 pos_x, s32 pos_y)
{
   CanTxMsg TxMessage;
   s32 xx=0,yy=0;
   xx=pos_x;   //car_x 即为要发送的 x 坐标
   yy=pos_y;   //car_y 即为要发送的 y 坐标
   TxMessage.StdId = 0x12;//如果控制器 ID 改变，修改这里。
   TxMessage.DLC = 8;//必须为 8
   TxMessage.IDE=CAN_Id_Standard;
   TxMessage.RTR=CAN_RTR_Data;
   TxMessage.Data[0]=(u8)(xx>>0);
```

```
    TxMessage.Data[1]=(u8)(xx>>8);
    TxMessage.Data[2]=(u8)(xx>>16);
    TxMessage.Data[3]=(u8)(xx>>24);
    TxMessage.Data[4]=(u8)(yy>>0);
    TxMessage.Data[5]=(u8)(yy>>8);
    TxMessage.Data[6]=(u8)(yy>>16);
    TxMessage.Data[7]=(u8)(yy>>24);
    CAN_Transmit(CAN2, &TxMessage);
}
//模块软件复位
void resetModule(void)
{
    CanTxMsg TxMessage;

    TxMessage.StdId = 0x12;//如果控制器 ID 改变，修改这里。
    TxMessage.DLC = 2;//必须为 2
    TxMessage.IDE=CAN_Id_Standard;
    TxMessage.RTR=CAN_RTR_Data;
    TxMessage.Data[0]=0x55;
    TxMessage.Data[1]=0xff;
    CAN_Transmit(CAN2, &TxMessage);
}
```

附件 2： POS 指令集说明

一、 陀螺仪模块

命令 1：测试连接命令

命令	应答	参数
POS	OK.	无

命令 2：查询——查询程序版本号

命令	应答	参数
POS+VERSION	+VERSION=< Para1>	<Para1>：固件版本号

举例：

POS+VERSION\r\n

VERSION: Firmware V1.0

命令 3：软件复位/重启

命令	应答	参数
POS+RESET	The module has been restarted.	无

命令 4：恢复默认设置

命令	应答	参数

POS+DEFAULT	The module will restore factory default. Restore successful. Current parameters: LID:0x0011 CID:0x0012 DX: 0 mm DY: 0 mm D: 100 mm CNT: 2000	无

命令 5：查询——查询帮助信息

命令	应答	参数
POS+HELP	---------- POS HELP ---------- Please do not enter multiple instruction within 5 ms. - Command Description POS Check the connect. …… ……	无

命令 6：查询/设置——本模块通过 CAN 发送数据所用 ID（LID）

命令	应答	参数
POS+LID	+LID=< Para1>	< Para1>: LID (16进制，0x0000~0x07FF) 输入时不需输入“0x” 默认：0x0011
POS+LID< Para1>	+LID=< Para1>, OK	

举例：

POS+LID0005\r\n

+LID=0x0005, OK

命令 7：查询/设置——本模块通过 CAN 接收指令所用 ID（CID）

命令	应答	参数
POS+CID	+CID=< Para1>	< Para1>:CID (16进制，0x0000~0x07FF) 输入时不需输入“0x” 默认：0x0012
POS+CID< Para1>	+CID=< Para1>, OK	

命令 8：查询/设置——陀螺仪校准

命令	应答	参数
POS+ANG	+ANG=< Para1>	< Para1>: 当前陀螺仪校准系数
POS+ANGS	Please rotate the robot with this module for 5 circles in horizontal precisely. Input "POS+ANGE" when finished above-mentioned action. Current angle: < Para1>	< Para1>: 当前陀螺仪输出角度

POS+ANGE	Current angle: < Para1> +ANG=< Para2>, OK	< Para1>: 当前陀螺仪输出角度 < Para2>: 当前陀螺仪校准系数

校准步骤

1、输入指令“POS+ANGS\r\n”启动陀螺仪校准

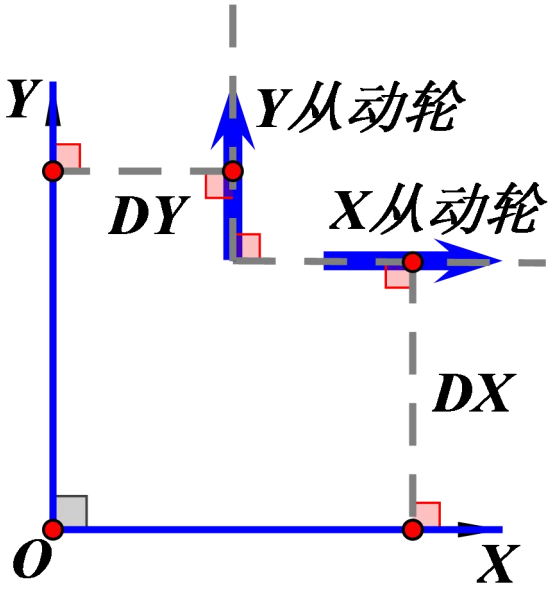
2、将装有陀螺仪模块的机器人或其它平台按某一方向（可顺时针，也可逆时针）精确旋转 5 圈

3、输入指令“POS+ANGE\r\n”结束陀螺仪校准

4、输入“POS+WRITE\r\n”将校准参数写入 Flash

二、 编码器模块

推荐使用 HEDL-5540 500 线编码器



OXY坐标系：全场定位坐标系

O点：移动机器人底盘的旋转中心
DX：装X轴编码器的正交码盘从动轮到X轴的距离
DY：装Y轴编码器的正交码盘从动轮到Y轴的距离

注意：
X从动轮须与Y从动轮垂直
正方向及几何关系如图所示

参数说明

命令 9：查询/设置——X 轴编码器的正方向

命令	应答	说明
POS+DIRX	Please forward the X wheel within 5 seconds. 5s …… Set DIRX successful.	按提示在5秒内将X从动轮往所定义的正方向旋转（正常速度即可）。

命令 10：查询/设置——Y 轴编码器的正方向

命令	应答	说明
POS+DIRY	Please forward the Y wheel within 5 seconds. 5s …… Set DIRY successful.	按提示在5秒内将Y从动轮往所定义的正方向旋转（正常速度即可）。

命令 11：查询/设置——X 轴编码器从动轮相对于坐标原点的偏移量 DX

命令	应答	参数
POS+DX	+DX=< Para1>< Para2>	< Para1>：+/-（DX的符号） 默认：- < Para2>：DX的数值大小 [10进制(0~9999)，单位：mm] 默认：0
POS+DX< Para1>< Para2>	+DX=< Para1>< Para2>, OK	

命令 12：查询/设置——Y 轴编码器从动轮相对于坐标原点的偏移量 DY

命令	应答	参数
POS+DY	+DY=< Para1>< Para2>	< Para1>：+/-（DX的符号） 默认：- < Para2>：DY [10进制(0~9999)，单位：mm] 默认：0
POS+DY< Para1>< Para2>	+DY=< Para1>< Para2>, OK	

命令 13：查询/设置——编码器从动轮的直径 D

命令	应答	参数
POS+D	+D=< Para1>	< Para1>：D [10进制(0~9999)，单位：mm] 默认：100
POS+D< Para1>	+D=< Para1>, OK	

命令 14：查询/设置——编码器转一圈的计数值 CNT

命令	应答	参数
POS+CNT	+ CNT=< Para1>	< Para1>：CNT 默认：2000
POS+CNT< Para1>	+ CNT =< Para1>, OK	

三、 读取/保存参数

命令 15：查询——读取当前参数

命令	应答	参数
POS+READ	Current parameters: LID: < Para1>	< Para1>: LID (16进制) < Para2>: CID (16进制)

	CID: < Para2> DX: < Para3> mm DY: < Para4> mm D: < Para5> mm CNT: < Para6>	< Para3>: DX (mm) < Para4> : DY (mm) < Para5> : D (mm) < Para6> : CNT

命令 16：设置——将当前参数写入 Flash

命令	应答	参数
POS+WRITE	Write current parameters to flash. Write successful. Current parameters: LID: < Para1> CID: < Para2> DX: < Para3> mm DY: < Para4> mm D: < Para5> mm CNT: < Para6>	< Para1>: LID (16进制) < Para2>: CID (16进制) < Para3>: DX (mm) < Para4> : DY (mm) < Para5> : D (mm) < Para6> : CNT

四、 注意事项

1、POS 命令不分大小写，均以回车、换行字符结尾：\r\n

2、LID、CID 均以十六进制（无需加“0x”）形式进行设置

范围：0x0000~0x07FF，即输入 0000~07ff

3、DX、DY、D、CNT 均以十进制形式进行设置

范围：0000~9999

4、设置完参数后，使用“POS+WRITE”将当前参数写入 Flash，否则一切设置无效！