MPI204x

惯性测量传感器

使用说明书

(Version: 2.3)

2019.11.29

www.memsplus.com

MPI204x 惯性测量传感器

➢ 产品描述

曼普拉斯 MPI204x 系列惯性测量传感器集成了 32 位 MCU、高性能加速度传感器和高性能工业级单轴陀螺传感器。采用 MEMSPlus 自主研发的高性能惯性传感器融合算法。具有高灵敏度、高稳定性和高精度等特性。不依赖于外界环境而提供精确稳定的水平方位角度信息，适用于各种需要导航或标定的场合。

➢ 产品特点

- 极高灵敏度
- 优异的长期稳定性
- RS485/CAN 工业总线
- 12 ~ 30V 宽电压输入
- 使用寿命长
- 性价比高
- 出厂已标定校准
- 低功耗

➢ 主要应用领域

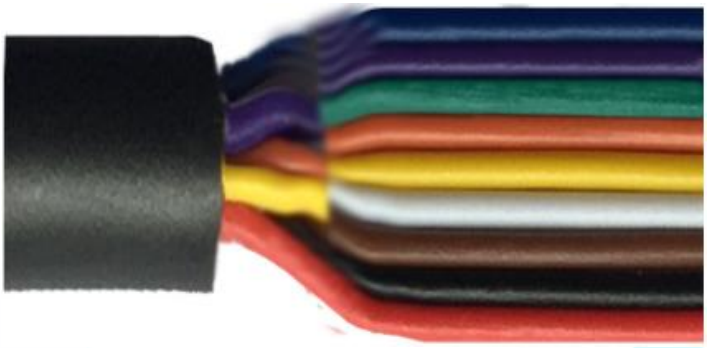
- 工业机器人
- 惯性导航组合
- 服务机器人
- 工业设备移动监测

➢ 技术指标

- Noise：< 0.1 °/s
- Bias Drift: < 20 °/hr
- Input Range: < ±400 °/s
- 工作电压：12 ~ 30V(DC)
- 电气接口：CAN/RS485
- 工作电流：≤ 45mA
- 工作温度：0℃ ~ 50℃
- 存储湿度：≤ 95%RH
- 存储温度：－20℃ ~ 60℃
- 工作湿度：≤ 60%RH
- 外形尺寸：51 x 91 x 26(mm)

目录

一、 接口定义



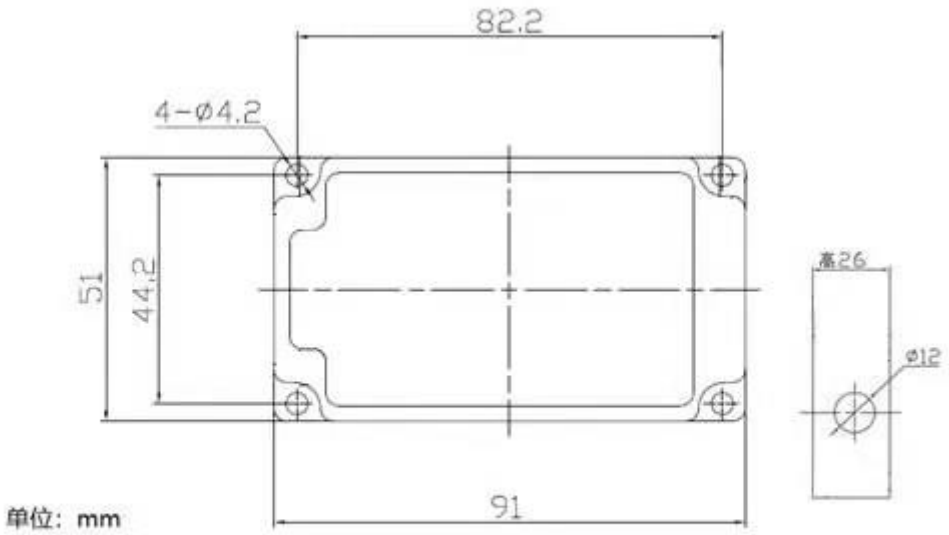
No.	MPI204A	MPI204C	No.	MPI204A	MPI204C
1	VCC	VCC	6	RS485-A/+	RS485-A/+
2	GND	GND	7	CAN-H	CAN-H
3	GND	Reserved①	8	CAN-L	CAN-L
4	RS485-GND②	RS485-GND	9	CAN-GND	CAN-GND
5	RS485-B/-	RS485-B/-			

① MPI204C 可与 MPSH208 连接，用于磁钉导航系统方案。除此以外请让该接线悬空。

② MPI204x 系列 RS485、CAN 均为工业级隔离接口，4、9 均为隔离信号地，不等同于 GND。如主控设备接口无信号地，请保持该接线悬空。不可与电源地相连。

二、 安装说明

1) 务必将传感器固定牢固，以减少抖动带来的测量误差。但使用螺丝固定时最好加装韧性垫片，避免直接刚性接触。
2) 固定时请保持模块测量面与被测物运动平面相平行。
3) 请务必保证模块不被弯曲变形，以免增加模块测量误差甚至损坏。
4) 传感器尺寸及安装孔位置如下图所示：



三、 通讯接口

1) RS-485

- **通信参数**

	参数	默认值
传输方式	Modbus-RTU	
波特率	115200，38400，19200，9600，4800	115200
DataBits	8	
StopBits	1	
Parity	None, Odd, Even	None
校验方式	CRC-16	
设备地址	1 - 99	8

- **通信指令**

1) 数据读取指令：

08 03 00 04 00 04 05 51

(08: Modbus ID; 03:功能码; 00 04:寄存器地址; 00 04:数据长度(字); 05 51: CRC16 校验码)

指令返回：

08 03 08 FC 88 FF 56 00 00 00 00 E0 51

保留字节内容随机
角速度值低字节
角速度值高字节
角度值低字节
角度值高字节

数据解析：

当前角度 Angle = (float) ((short)(0xFC << 8) | 0x88) / 10.0 = -88.8°

当前角速度 AngleRate = (float) ((short) (0xFF << 8) | 0x56) / 10.0 = -17.0°/s

（以上 C 程序仅作参考，请注意强制类型转换）

2) 角度清零指令：

08 10 00 04 00 02 04 43 4C 52 00 35 F3

指令返回：

08 10 00 04 00 02 00 90 （执行成功）

2) CAN（Raw CAN）

- **通信参数**

功能	参数	默认值
物理接口	CAN BUS 2.0A	
节点 ID	1~127	8
波特率	125Kbps, 250Kbps, 500Kbps, 800Kbps, 1000Kbps	1000Kbps
帧类型	标准帧	
帧格式	数据帧	

- **通信指令**

CAN 接口数据协议有三种，**主动输出**、**被动查询**和 **CANopen**，传感器默认 CAN 接口为主动输出模式。该模式下传感器以设定频率输出数据帧。被动查询模式下 CAN 主设备发送询问帧，传感器返回数据帧。

1) 主动模式输出数据帧：

CAN-ID	Payload（TLC = 4）			
0x588	0xFC	0x88	0xFF	0x56
0x580+NodeID	角度值高字节	角度值低字节	角速度值高字节	角速度值低字节

当前角度 Angle = (float) ((short)(0xFC << 8) | 0x88) / 10.0 = -88.8°
当前角速度 AngleRate = (float) ((short) (0xFF << 8) | 0x56) / 10.0 = -17.0°/s
（以上 C 程序仅作参考，请注意强制类型转换）

2) 查询模式下主设备查询帧如下：

CAN-ID	Payload（TLC = 2）	
0x608	0x4D	0x04
0x600+NodeID		

传感器应答帧：

CAN-ID	Payload（TLC = 6）					
0x588	0x4D	0x04	0xFC	0x88	0xFF	0x56
0x580+NodeID	查询命令		角度值高字节	角度值低字节	角速度值高字节	角速度值低字节

3) 角度清零帧：

	CAN-ID	Payload (TLC = 5)				
发送帧	0x608	0xCE	0x04	0x43	0x4C	0x52
	0x600+NodeID	清零指令				
	CAN-ID	Payload (TLC = 5)				
返回帧	0x588	0xCE	0x04	0x43	0x4C	0x52
	0x580+NodeID	指令执行成功角度清零，接收的命令原样返回，失败则无返回				

3) CANopen

CANopen 是以 CAN 为基础的多主机的现场总线系统，本设备实现的 CANopen 协议完美支持 DS301 v4.02 版本，可通过 CiA（CAN in Automation）一致性测试。

1) **EDS**（**E**lectronic **D**ata **S**heet）

EDS 文件是描述 CANopen 节点设备功能参数的电子说明书，大多数 CANopen 的主控设备需要导入节点设备的 EDS 文件才能操作该节点。请联系销售或技术支持获取。

2) **对象字典**（**O**bject **D**ictionary）

对象字典（简称 OD）详细描述了 CANopen 设备的所有对象，是 CANopen 设备的核心。

Index	Sub Index	Description	Default	Type	R/W
0x1000		Device Type	0	UINT32	R
0x1001		Error Register	0	UINT8	R
0x1005		SYNC COB ID	0	UINT32	RW
0x1008		Manufacture Device Name	MPI204x	STRING	R
0x1009		Manufacture Hardware Version		STRING	R
0x100A		Manufacture Software Version		STRING	R
0x100C		Guard Time	0	UINT16	RW
0x100D		Life Time Factor	0	UINT8	RW
0x1010		Store Parameters			
	0	Number of Entries	3	UINT8	R
	1	Save All Parameters	0	UINT32	RW
	2	Save Communication Parameters	0	UINT32	RW
	4	Save Manufacture Parameters	0	UINT32	RW
0x1011		Restore Default Parameters			
	0	Number of Entries	3	UINT8	R
	1	Restore All Default Parameters	0	UINT32	RW
	2	Restore Manufacture Default Parameters	0	UINT32	RW
	4	Restore Manufacture Defined Default Parameters	0	UINT32	RW
0x1014		Emergency COB ID	0x88	UINT32	RW
0x1017		Producer Heartbeat Time	0	UINT16	RW
0x1018		Identity			
	0	Number of Entries	4	UINT8	R
	1	Vendor ID	77	UINT32	R
	2	Product Code	0	UINT32	R
	3	Revision Number	0	UINT32	R
	4	Serial Number	0	UINT32	R
0x1200		Server SDO Parameter			
	0	Number of Entries	2	UINT8	R
	1	COB ID Client to Server	&NODEID+0x600	UINT32	R

	2	COB ID Server to Client	&NODEID+0x580	UINT32	R
0x1400		Receive PDO 1 Parameter			
	0	Number of Entries	2		
	1	COB ID used by PDO	0x208	UINT32	RW
	2	Transmission Type	254	UINT8	RW
0x1600		Receive PDO 1 Mapping of Input			
	0	Number of Entries	2	UINT8	R
	1	Mapping for Angle Reset	0x22000108	UINT32	RW
	2	Reserved	0x22000208	UINT32	RW
0x1800		Transmit PDO 1 Parameter			
	0	Highest Sub Index Supported	5	UINT8	R
	1	COB ID used by PDO	0x184	UINT32	RW
	2	Transmission Type	254	UINT8	RW
	3	Inhibit Time	100	UINT16	RW
	5	Event Timer	50	UINT16	RW
0x1A00		Transmit PDO 1 Mapping of output			
	0	Number of Entries	2	UINT8	R
	1	Mapping for Angle	0x20000120	UINT32	RW
	2	Mapping for Angle Rate	0x20000220	UINT32	RW
0x2000		Sensor Output			
	0	Number of Entries	2	UINT8	R
	1	Angle	0	FLOAT	R
	2	Angle Rate	0	FLOAT	R
0x2200		Sensor Input			
	0	Number of Entries	2	UINT8	R
	1	Angle Reset	0	UINT8	R
	2	Reserve	0	UINT8	R
0x3000		Device Parameters			
	0	Number of Entries	10	UINT8	R
	1	Data Interface Index	0	UINT8	RW
	2	Sensing Polarity Index	0	UINT8	RW
	3	RS-485 Modbus ID	4	UINT8	RW
	4	RS-485 Baudrate Index	4	UINT8	RW
	5	RS-485 Parity Index	0	UINT8	RW
	6	RS-232 Baudrate Index	4	UINT8	RW
	7	CAN Node ID	4	UINT8	RW
	8	CAN Baudrate Index	4	UINT8	RW
	9	CAN Data Protocol Index	0	UINT8	RW
	10	RawCAN Data Rate Index	2	UINT8	RW

3) 启动节点

根据 OD 或 EDS 文件可知设备默认使用 TPDO1 来发送传感器数据（OD 中黄色区域），无需改动即可快速使用（当然你也可以修改 TPDO1 的参数和映射来组网），过程如下：

a) 上电节点。

b) 节点上电完毕后，会上报 Boot Up 帧告诉 NMT Master 节点已经准备完毕，等待 NMT 命令。

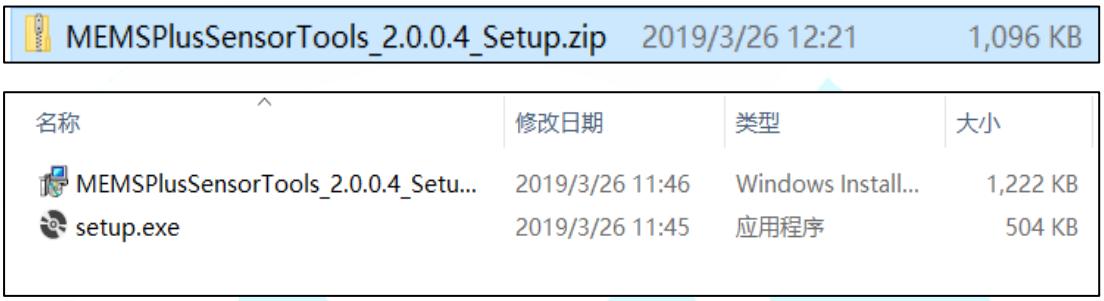
c) NMT Master 发送 NMT 命令（如 START、STOP 等）即可启动或停止数据传输。

四、 软件工具

如需修改传感器的默认配置，请使用曼普拉斯传感器配置工具软件 MEMSPlusSensorTools（请联系销售或技术支持获取最新版本）。

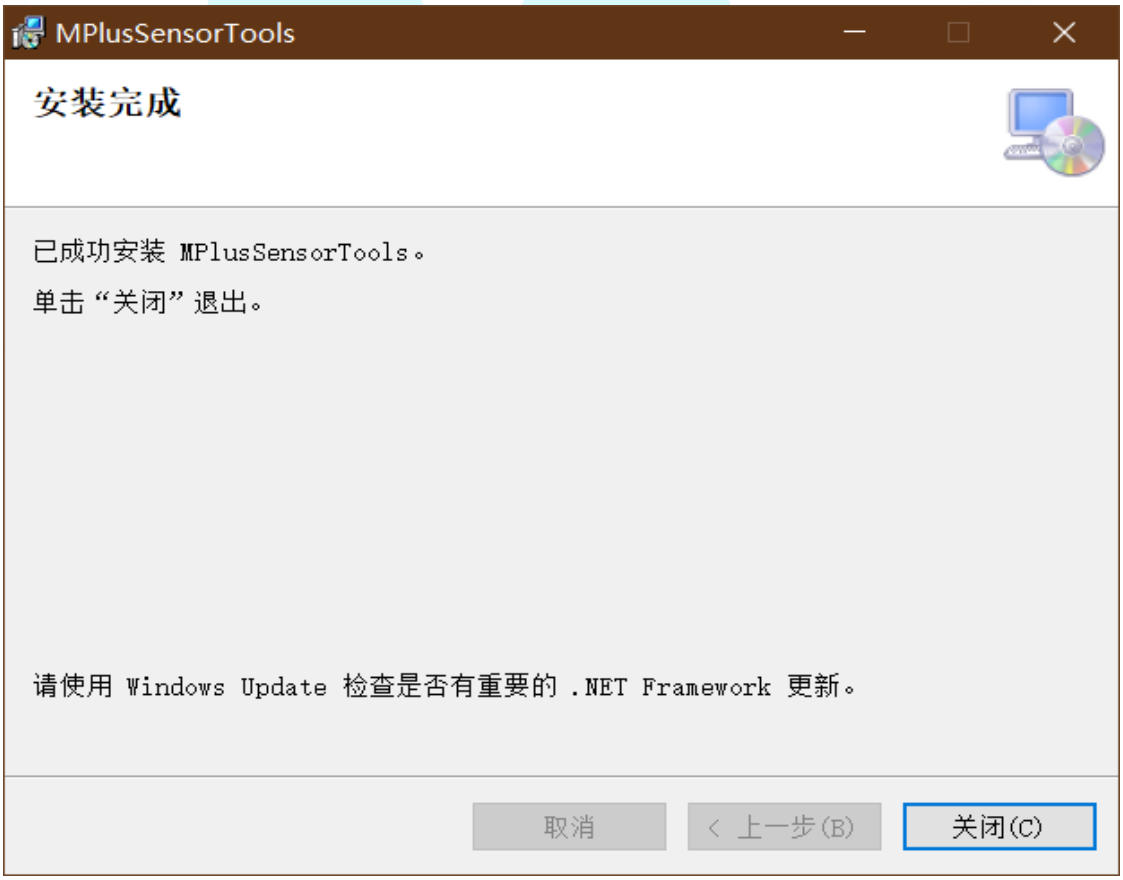
1) 软件安装

a) 解压安装包



b) 启动安装

双击 setup.exe 安装开始后根据安装引导程序完成整个安装流程。



注：该软件适用于 Windows 7 及以上操作系统。

2) 软件使用

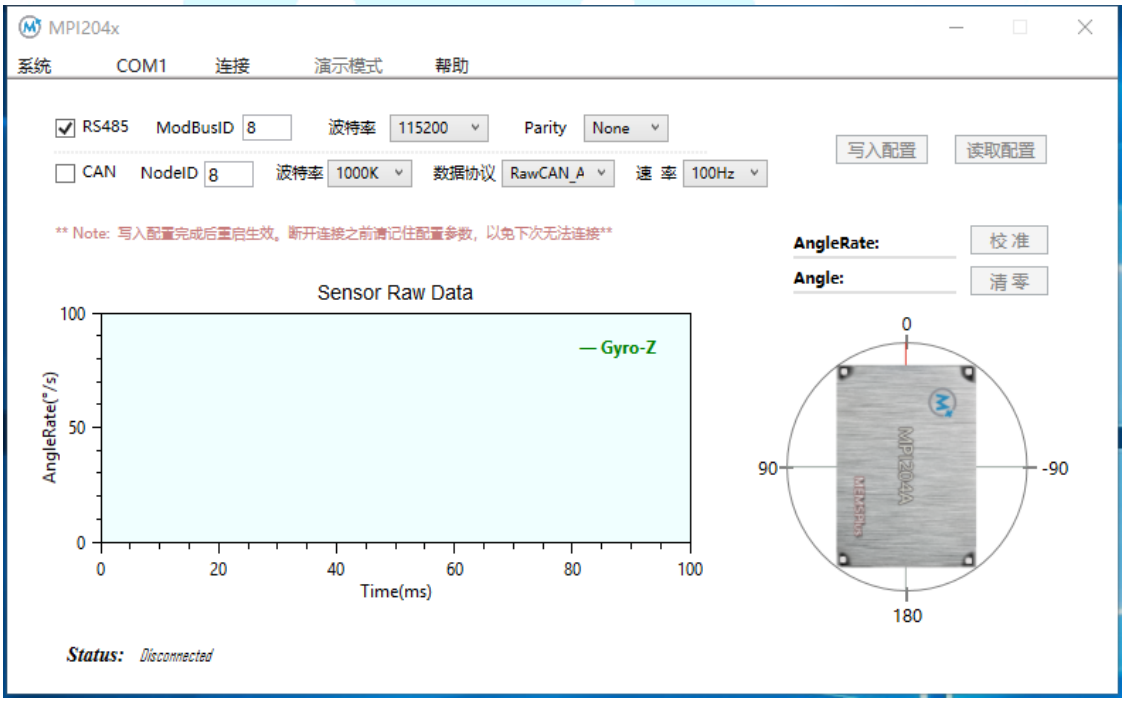
a) 启动软件

软件安装完成后会在桌面生成启动图标  ，也可以再开始菜单里面找到MEMSPlusSensorTool 启动图标。

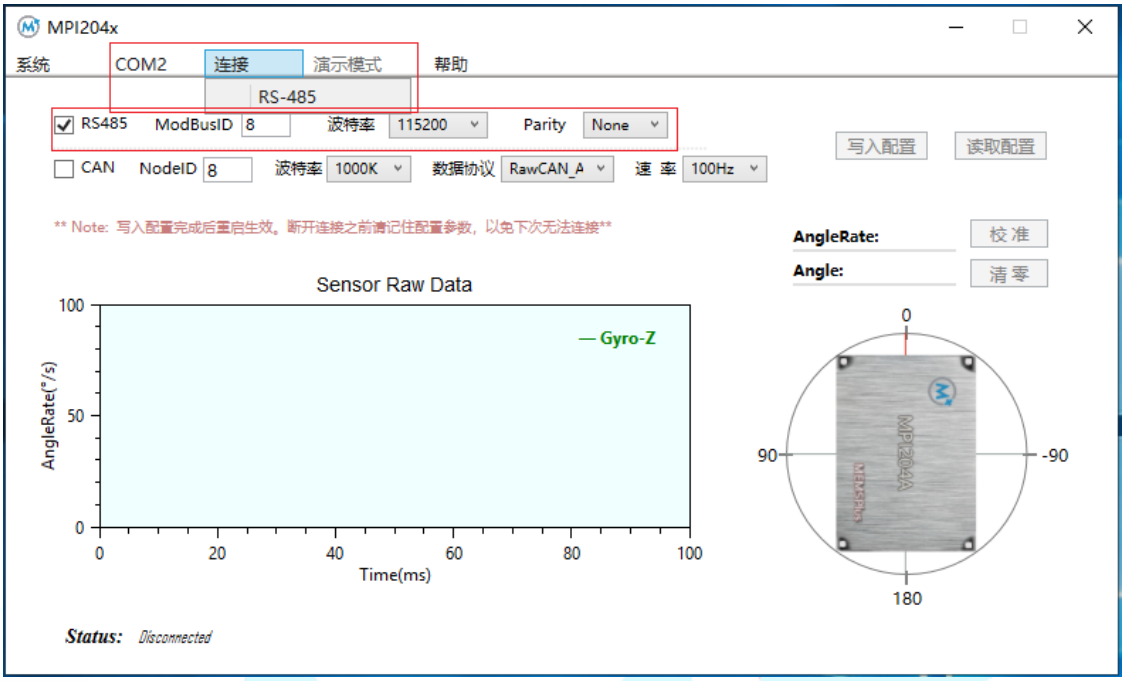
启动之后选择相应的产品名称，然后点击启动按钮。

b) 连接设备

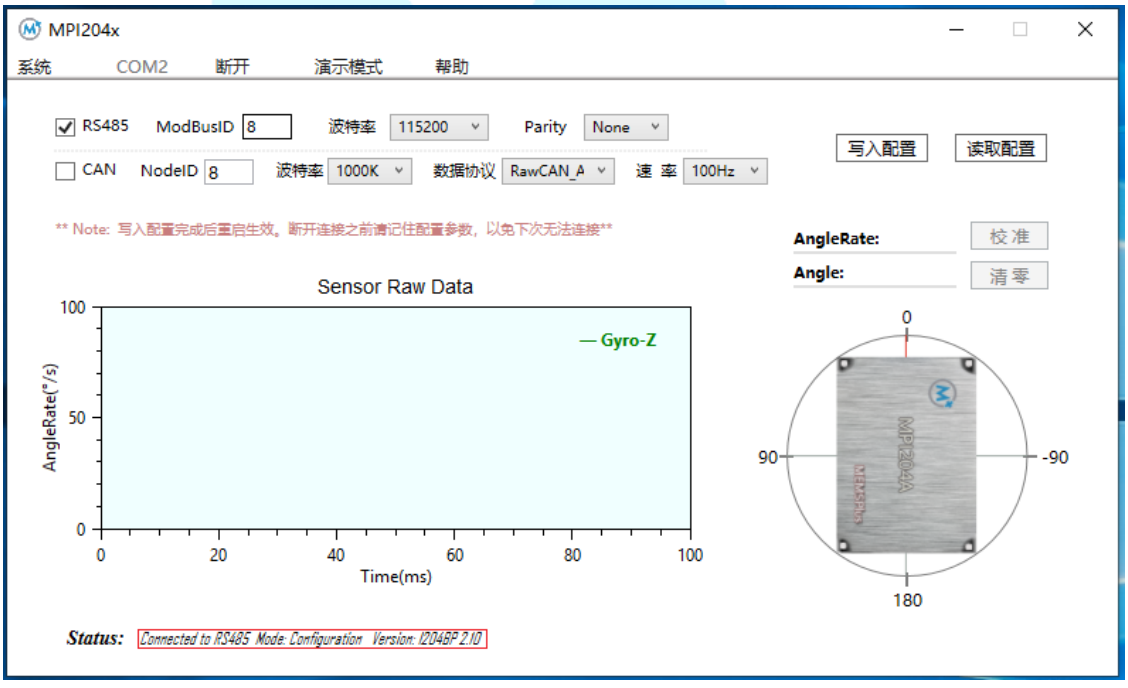
启动产品界面如下（该软件支持 RS-485 方式与设备连接）



请根据设备的连接方式和端口在软件界面选择相应的设置，如下图所示：

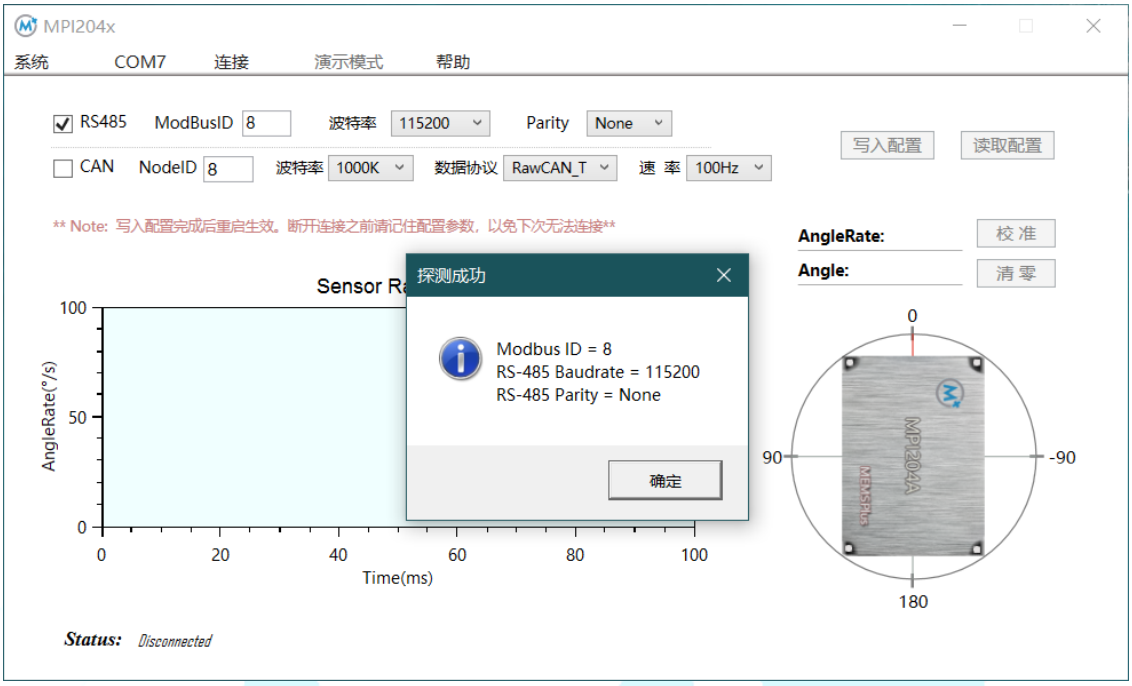


RS-485 连接方式，对应的端口为 COM2，首次连接 RS-485(Modbus)选项卡的参数保持默认即可。连接成功如下图所示：



c) 连接助手

如果不确定 RS485 和 CAN 接口的参数，无法连接上设备的时候，请尝试“帮助”菜单里面的“连接助手”。连接助手会自动帮您找回设备的参数配置（只支持 RS485）。如下图：

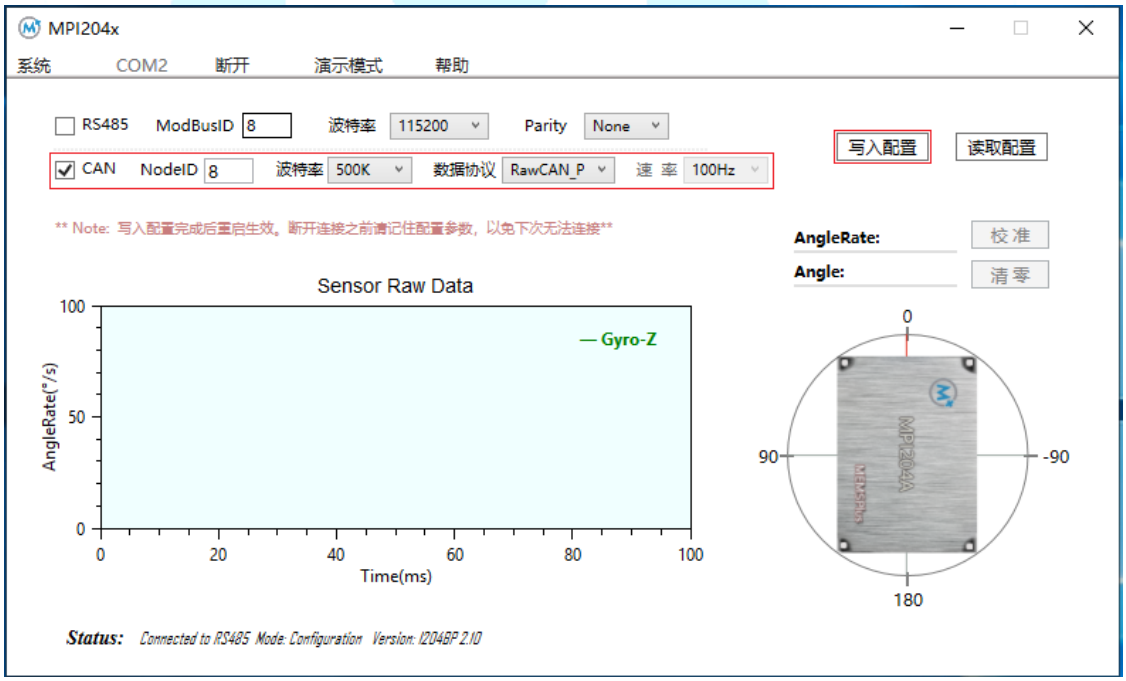


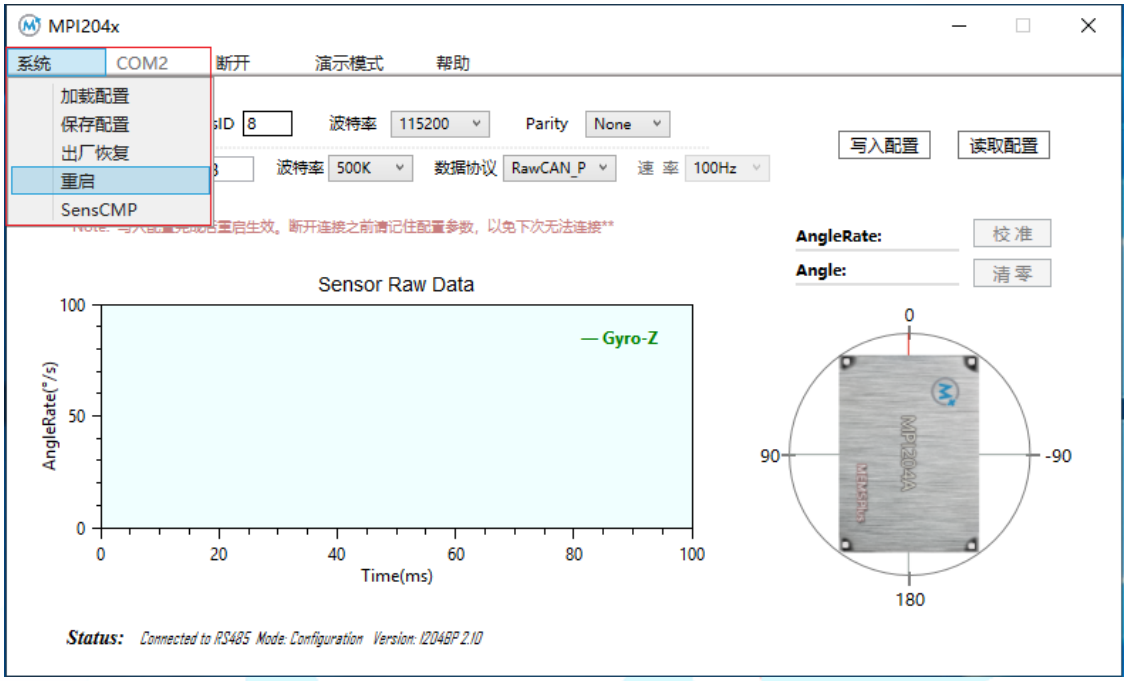
（注：使用该功能时请断开总线上其他设备）

d) 修改参数

连接成功之后，软件默认处于配置模式，此时可对设备进行参数修改。

例如：某客户需要使用 CAN 接口通讯，CAN 节点地址 6，波特率 500Kbps，数据传输协议为自定义被动查询模式（RawCAN_P），则修改界面如下：

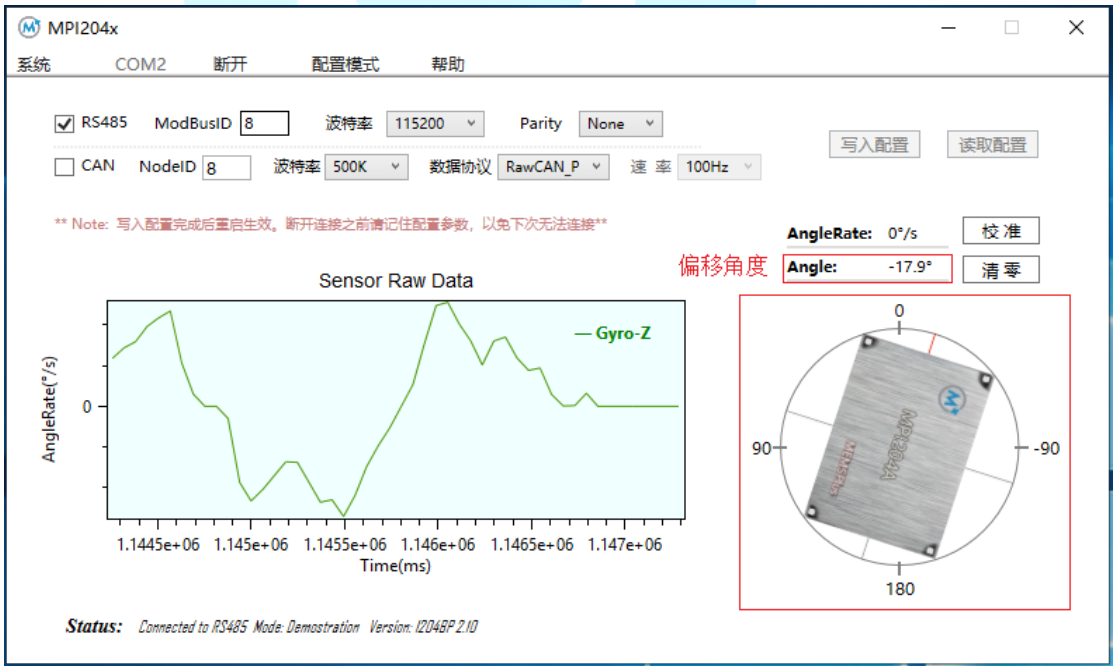




点击“写入配置”按钮即可完成参数修改，然后点击“重启”按钮或掉电重启即可使修改的配置生效。

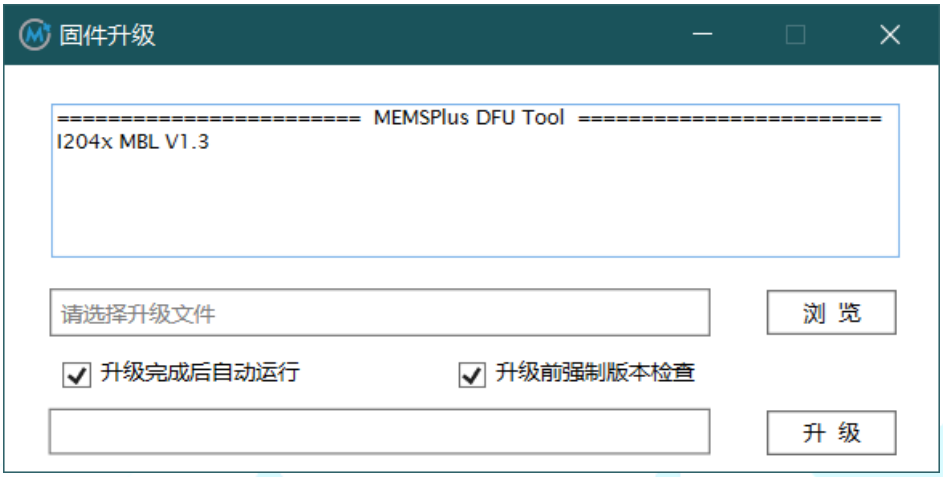
e) 演示模式

演示模式可以实时演示设备工作时的状态。点击菜单栏的“演示模式”可进入演示模式。

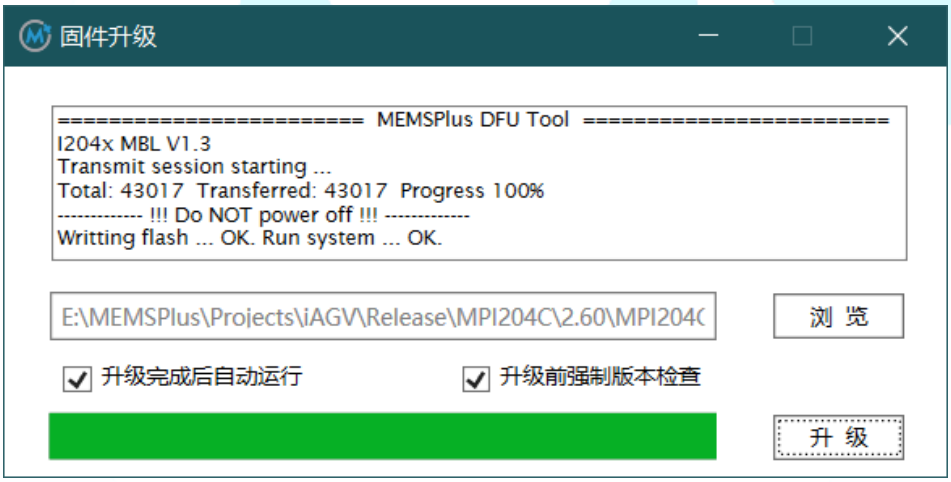


f) 固件升级

请首先连接上设备，然后点击“系统”菜单，选择“固件升级”选项。弹出升级窗口如下：



选择升级文件后点击“升级”按钮，然后等待升级完成即可。如下图：



(注：升级期间切勿断开连接或掉电)

五、 售后

1) 技术支持

公司网站：www.memsplus.com

电话：0512 – 67573040

附录:

C 语言 CRC-16 校验算法

```
/* 该算法适用于 Modbus 的 CRC-16 校验的快速计算 */

static const unsigned char aucCRCHi[] = {

    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
};
```

```
static const unsigned char aucCRCLo[] = {

        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
};

/* 返回 16 位 CRC 校验值 */
unsigned short usMBCRC16( unsigned char * pucFrame, unsigned short usLen )
{
        unsigned char               ucCRCHi = 0xFF;
        unsigned char               ucCRCLo = 0xFF;
        int                         iIndex;

        while( usLen-- )
        {
                iIndex = ucCRCLo ^ *( pucFrame++ );
                ucCRCLo = ( unsigned char)( ucCRCHi ^ aucCRCHi[iIndex] );
                ucCRCHi = aucCRCLo[iIndex];
        }
    return ( unsigned short)( ucCRCHi << 8 | ucCRCLo );
}
```