

USB 转 CAN&UART 模块说明书

USB 转 CAN&UART 模块是深圳市达妙科技有限公司，设计的一款 USB 转 CAN 设备调试模块。该模块可以同时支持 USB 转 CAN 和 UART 功能。支持 CAN 分析功能，并可以支持波特率设置功能。

1 性能参数

供电：USB5V

电流：<50mA

USB 转串口波特率：4800 ~ 921600bps

USB 转 CAN 波特率：5K ~ 1MBps

串口缓存：10KB

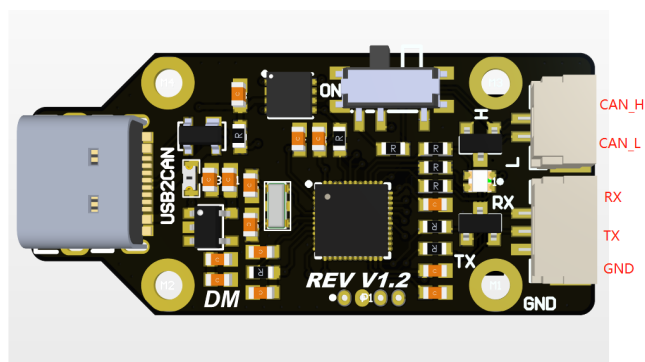
CAN 数据缓存：32KB 可容纳 1K 帧数据

ESD 性能：CAN 接口 $\pm 30\text{kV}$ (air) $\pm 30\text{kV}$ (contact)

ESD 性能：UART 接口 $\pm 15\text{kV}$ (air) $\pm 8\text{kV}$ (contact)

ESD 性能：USB 接口 $\pm 30\text{kV}$ (air) $\pm 30\text{kV}$ (contact)

2 引脚说明



GND GND 负极

CANL 连接 CAN 总线 CANL 信号线

CANH 连接 CAN 总线 CANH 信号线

RX 串口 RX 信号 (3.3V 电平, 5V 容忍)

TX 串口 TX 信号 (3.3V 电平, 5V 容忍)

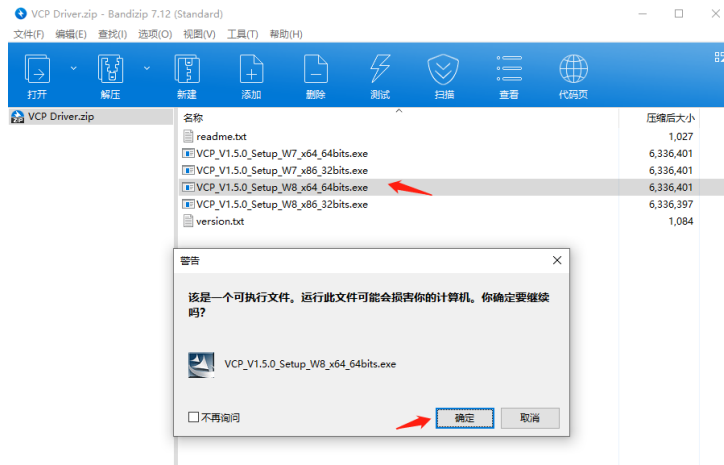
GND 参考地

拨动开关为 CAN 总线 120R 终端电阻接入选择开关，此处出厂默认接入 120R 中断电阻，拨动开关偏向丝印 ON 处，拨动到另一端为断开终端电阻的接入。

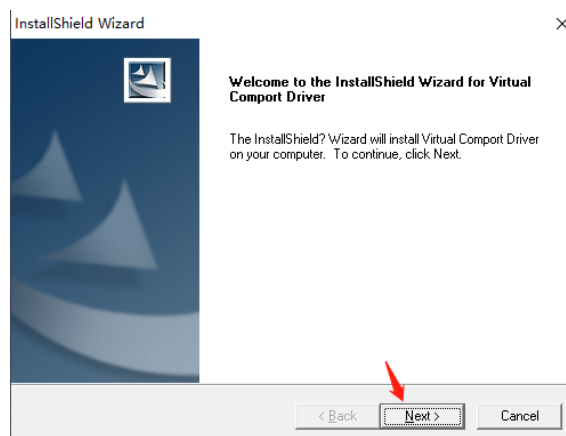
3 安装驱动 VCP Driver.zip

从论坛 (<http://www.dmbot.cn/forum.php?mod=viewthread&tid=328&page=1&extra=#pid572>) 下载虚拟串口驱动，并按照以下方式安装。

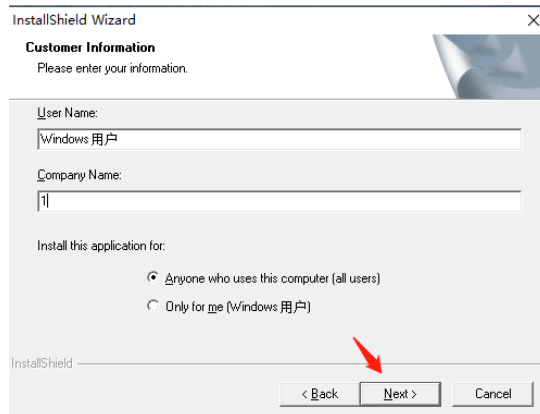
1) 根据系统选择合适的驱动包，下面以 W8 x64 为例（此驱动 win10 下可用）。



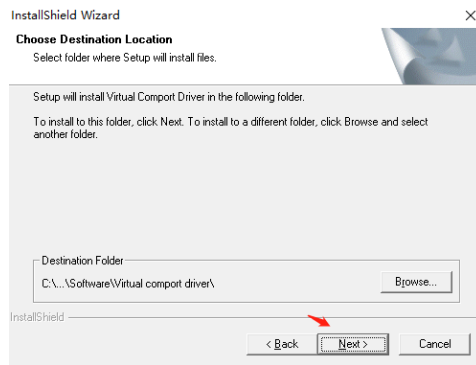
选择驱动并双击



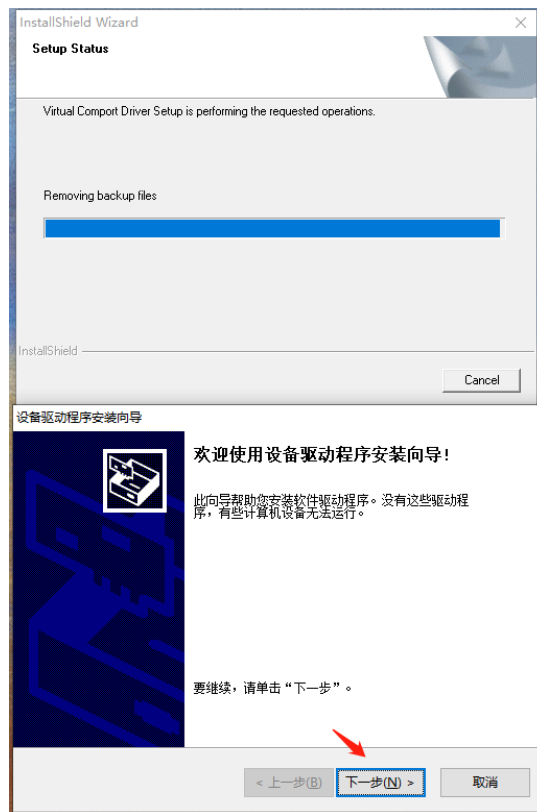
点击下一步



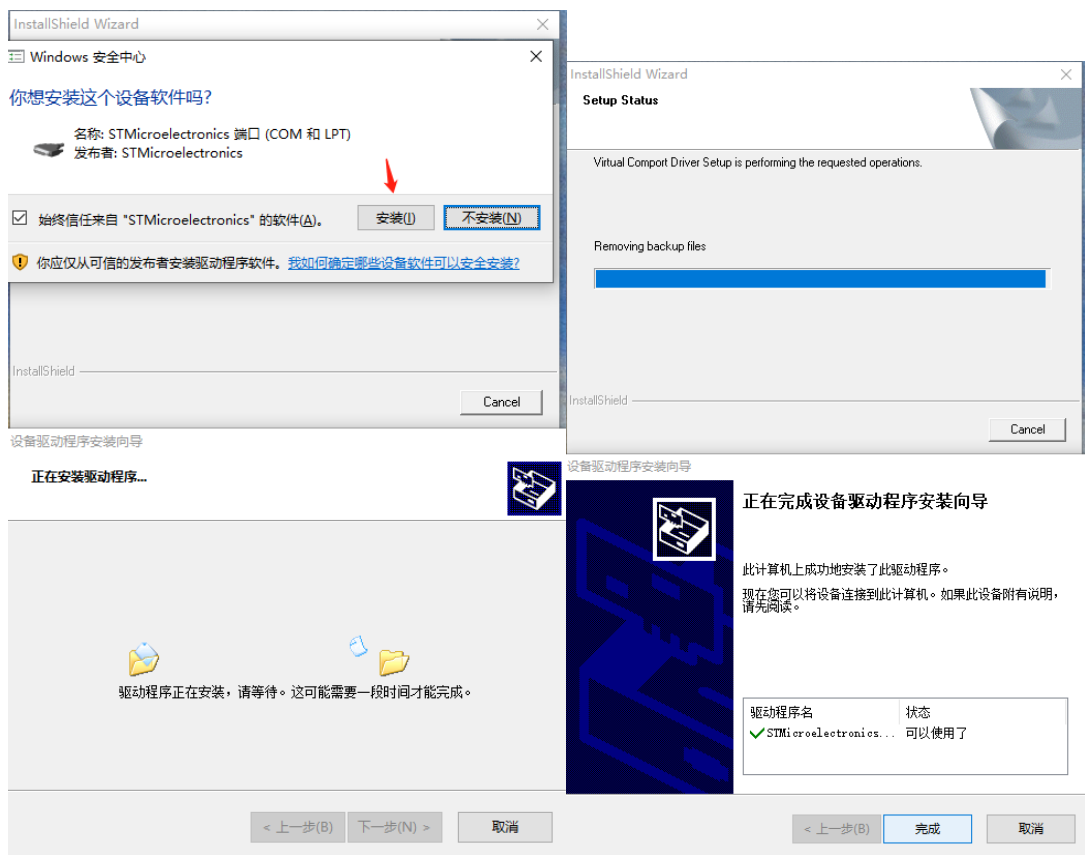
点击下一步



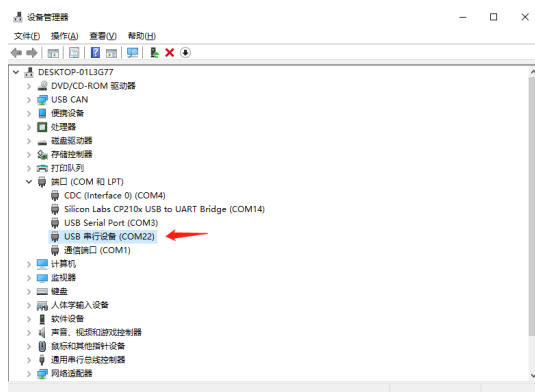
点击下一步



点击下一步




选择安装



安装完成后, 连接 USB 模块到电脑, 可以看到设备管理器增加 USB 穿行设备至此驱动安装结束。

4 使用

1) 使用单独的 USB 转 CAN&UART 工具

从论坛下载 USB2CAN 工具 **CAN** 分析仪上位机:  [mBotAcToolsV1.0.0.1.exe](#)。



具体使用方法可以参见视频教程 (<https://www.bilibili.com/video/BV1up4y1a7HX>)。

2) MIT 电机调试上位机

此处应配合达秒科技出品的 MIT 驱动板使用，其他驱动板请做出相应改动才能使用。

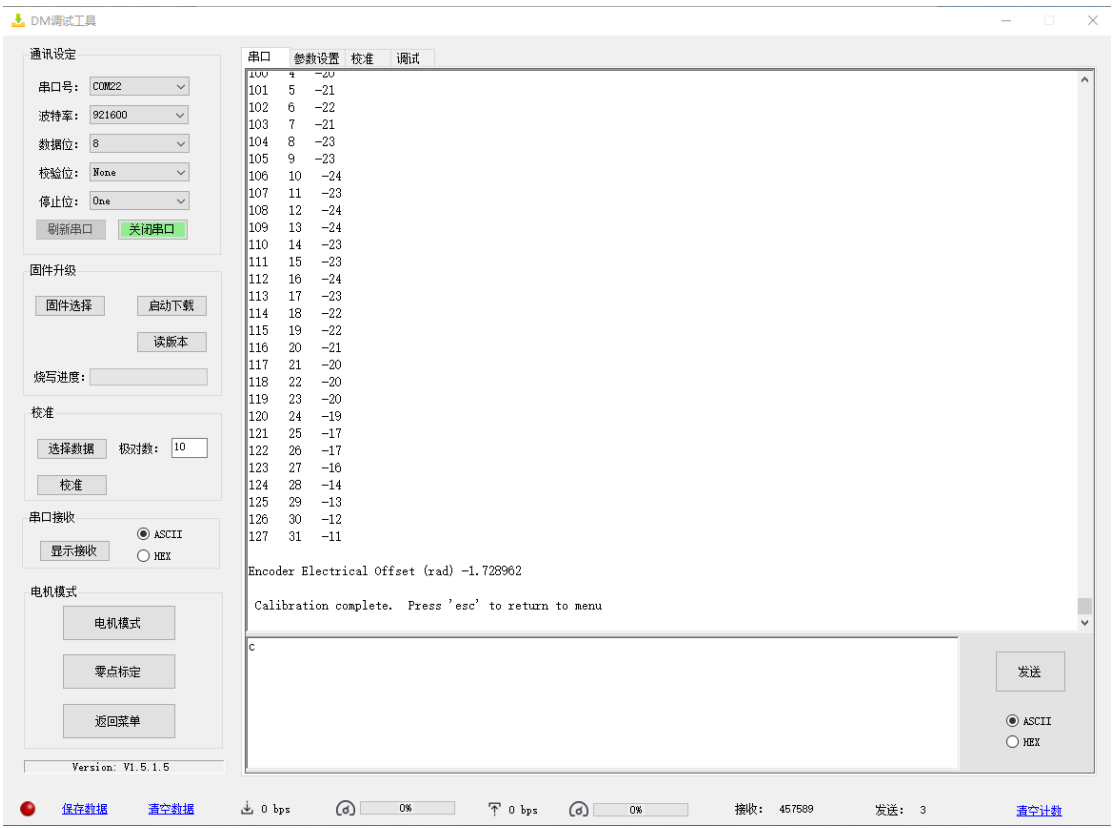
电机调试上位机： [dm_calibration tools-V1.5.1.4.exe](#)

具体使用方法可以参见视频教程 MIT 原版使用方法：(<https://www.bilibili.com/video/BV18y4y1t7Sw>)、
达秒科技驱动板使用方法 (<https://www.bilibili.com/video/BV17y4y1t7hn>)。

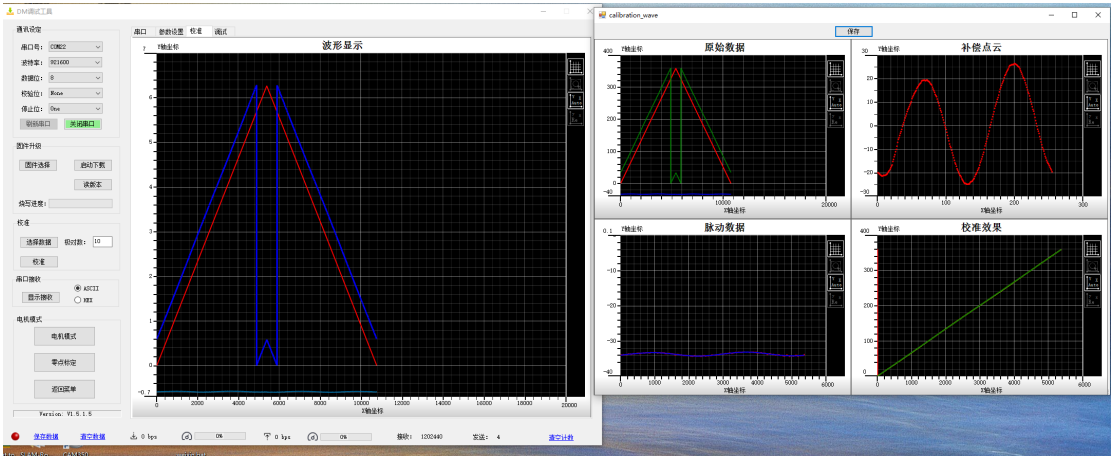


选择串口并打开设备

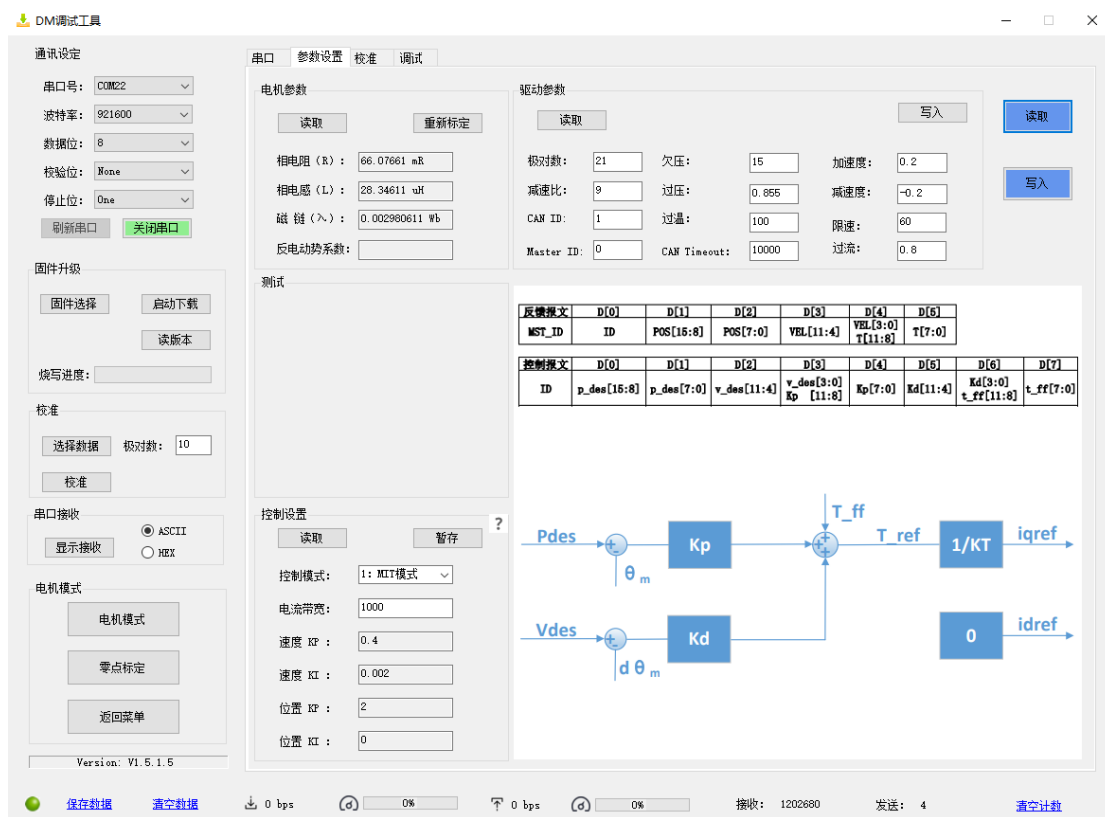
校准



校准后 误差补偿数据一般都比较小 如果较大请检查**极对数**是否与实际电机匹配。

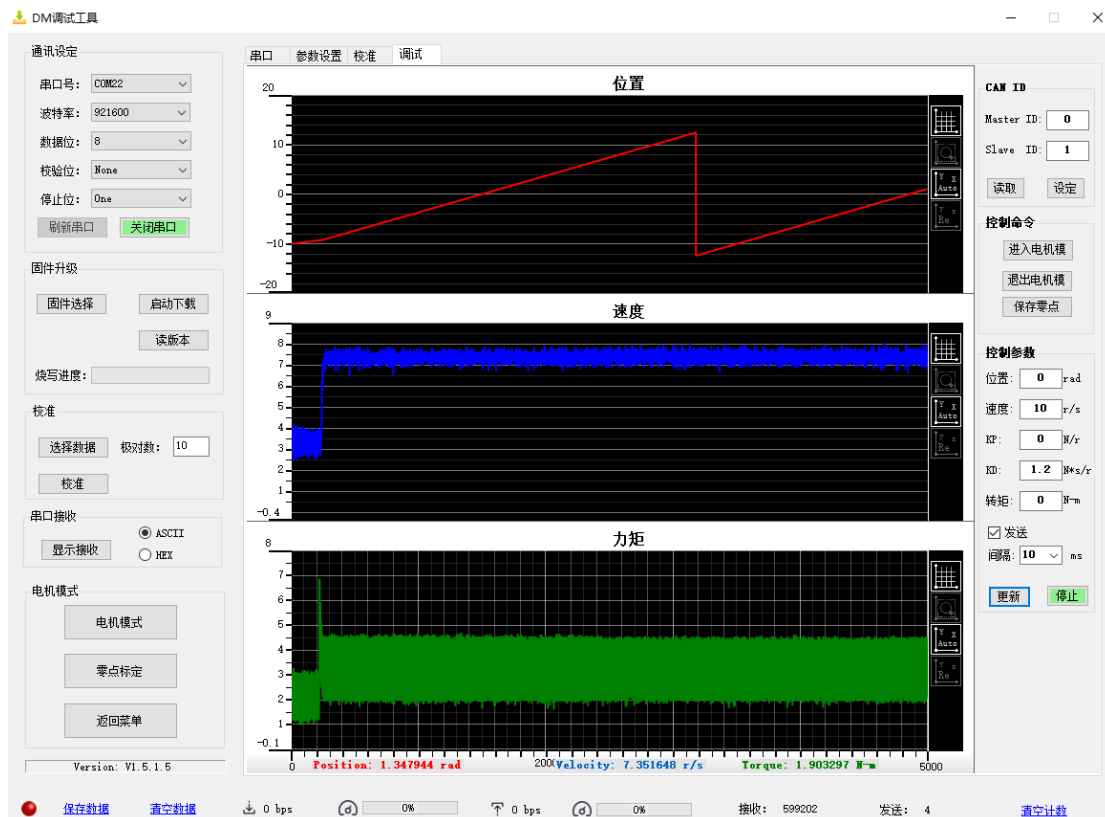


达妙科技关节电机校准示意 (MIT 原版无效)

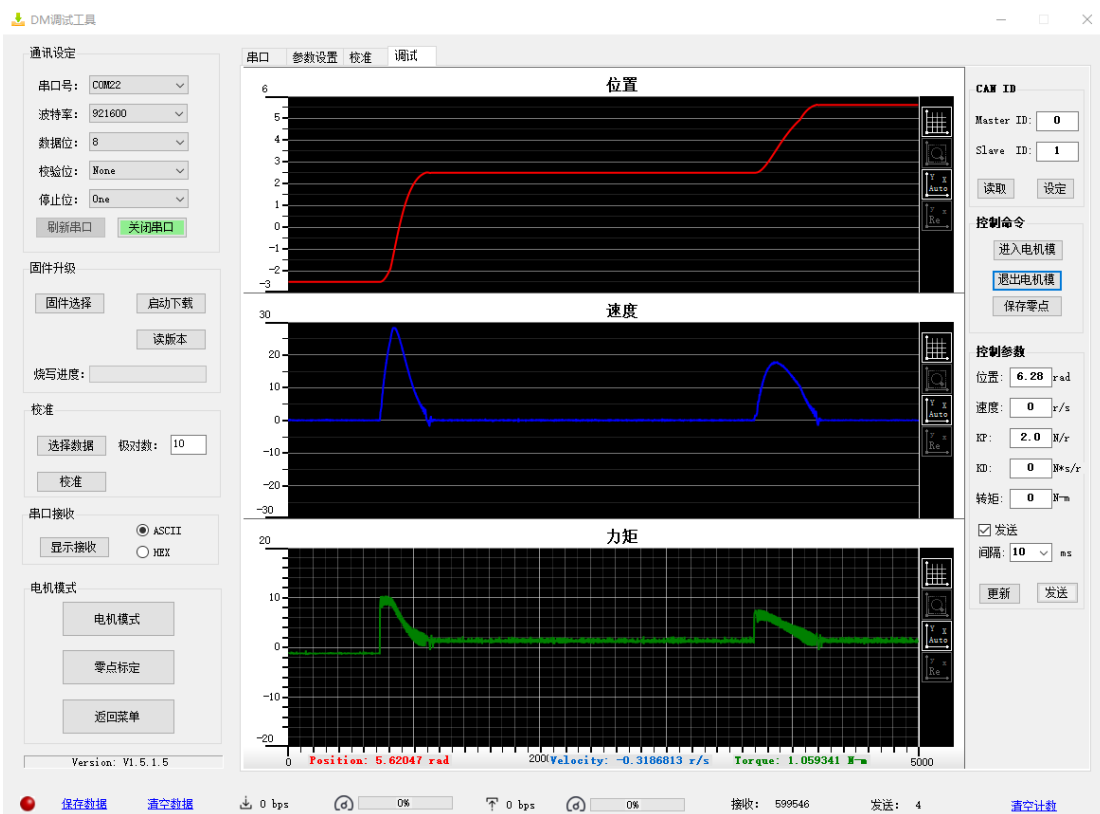


达秒科技驱动参数配置页面 (MIT 原版无效)

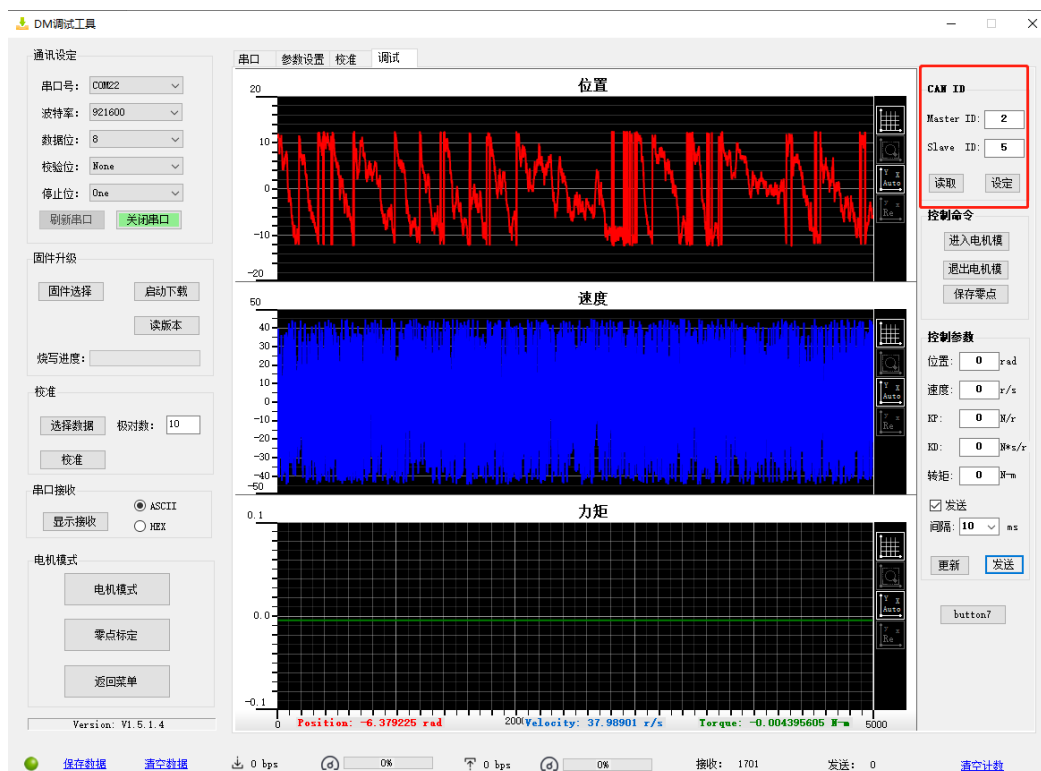
速度环控制



位置环控制



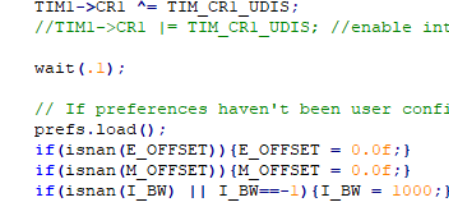
如何增加 CAN 通讯设置 ID 功能:



- 1、该功能仅在驱动处于 REST_MODE 模式下支持设置、读取 CAN_ID 的功能。
- 2、原版代码需要修改两处：

a) 在 main 函数增加 can 滤波器设置, 允许设置命令 0x7FF。位置及代码如下所示:

```
can.filter(0x7FF, 0x07FF, CANStandard, 1);
```



```
422 TIM1->CR1 |= TIM_CR1_UDIS;  
423 //TIM1->CR1 |= TIM_CR1_UDIS; //enable interrupt  
424  
425 wait(.1);  
426  
427 // If preferences haven't been user configured yet  
428 prefs.load();  
429 if(isnan(E_OFFSET)){E_OFFSET = 0.0f;}  
430 if(isnan(M_OFFSET)){M_OFFSET = 0.0f;}  
431 if(isnan(I_BW) || I_BW== -1){I_BW = 1000;}  
432 if(isnan(I_MAX) || I_MAX == -1){I_MAX=40;}  
433 if(isnan(I_FW_MAX) || I_FW_MAX == -1){I_FW_MAX=0;}  
434 if(isnan(CAN_ID) || CAN_ID== -1){CAN_ID = 1;}  
435 if(isnan(CAN_MASTER) || CAN_MASTER== -1){CAN_MASTER  
436 if(isnan(CAN_TIMEOUT) || CAN_TIMEOUT== -1){CAN_TIME  
437  
438 NVIC_SetPriority(TIM1_UP_TIM10_IRQn, 2);  
439  
440 NVIC_SetPriority(CAN1_RX0_IRQn, 3);  
441 // can.filter(CAN_ID<21, 0xFFFE0004, CANStandard,  
442 can.filter(CAN_ID, 0x07FF, CANStandard, 0);  
443 can.filter(0x7FF, 0x07FF, CANStandard, 1);  
444  
445 txMsg.id = CAN_MASTER;  
446 txMsg.len = 6;  
447 rxMsg.len = 8;
```

b) 在 CAN 中断处理函数 onMsgReceived() 中增加 CANID 读取、设置处理函数:

```

else if((rxMsg.id == 0x7FF) && (state ==REST_MODE))
{
    if((rxMsg.len == 4) && (rxMsg.data[0]==0xAA) && (rxMsg.data[3]==0x55) )
    {
        CAN_ID      = rxMsg.data[1] & 0x7F;    //0~127
        CAN_MASTER = rxMsg.data[2] & 0x7F;    //0~127
        //保存数据 并重载
        if (!prefs.ready()) prefs.open();
        prefs.flush();                          // Write
        new prefs to flash
        prefs.close();
        prefs.load();
        can.filter(CAN_ID, 0x07FF, CANStandard, 0); //重新设置滤波器
        txMsg.len = 0x04;
        txMsg.id = 0x7ff;
        memcpy(txMsg.data,rxMsg.data,4); //返回
        can.write(txMsg);
        txMsg.len = 0x06;
        txMsg.id = CAN_MASTER;
    }
    else if((rxMsg.len == 4) && (rxMsg.data[0]==0x55) && (rxMsg.data[3]==0xAA) )
    {
        can.filter(CAN_ID, 0x07FF, CANStandard, 0); //重新设置滤波器
        txMsg.id = 0x7ff;
        txMsg.len = 0x04;
        txMsg.data[0] = 0x55;
        txMsg.data[1] = CAN_ID&0xFF;
        txMsg.data[2] = CAN_MASTER&0xFF;
        txMsg.data[3] = 0xAA;
        can.write(txMsg);
        txMsg.len = 0x06;
        txMsg.id = CAN_MASTER;
    }
}
}

```

```

52 //DigitalOut drv_en_gate(PA_11);
53 DRV832x drv(&drv_spi, &drv_cs);
54
55 PositionSensorAM5147 spi(16384, 0.0, NPP);
56
57 volatile int count = 0;
58 volatile int state = REST_MODE;
59 volatile int state_change;
60
61 void onMsgReceived() {
62     //msgAvailable = true;
63     can.read(rxMsg);
64     if((rxMsg.id == CAN_ID)){
65         controller.timeout = 0;
66         if(((rxMsg.data[0]==0xFF) & (rxMsg.data[1]==0xFF) & (rxMsg.data[2]==0xFF) & (rxMsg.data[3]==0xFF) & (r
67             state = MOTOR_MODE;
68             state_change = 1;
69         })
70         else if(((rxMsg.data[0]==0xFF) & (rxMsg.data[1]==0xFF) & (rxMsg.data[2]==0xFF) & (rxMsg.data[3]==0xFF)
71             state = REST_MODE;
72             state_change = 1;
73             gpio.led->write(0);
74         })
75         else if(((rxMsg.data[0]==0xFF) & (rxMsg.data[1]==0xFF) & (rxMsg.data[2]==0xFF) & (rxMsg.data[3]==0xFF)
76             spi.ZeroPosition();
77         })
78         else if(state == MOTOR_MODE){
79             unpack_cmd(rxMsg, &controller);
80         }
81         pack_reply(&txMsg, controller.theta_mech, controller.dtheta_mech, controller.i_q_filt*KT_OUT);
82         can.write(txMsg);
83     }
84     else if((rxMsg.id == 0x7FF) && (state == REST_MODE))
85     {
86         if((rxMsg.len == 4) && (rxMsg.data[0]==0xAA) && (rxMsg.data[3]==0x55) )
87         {
88             CAN_ID = rxMsg.data[1] & 0x7F; //0~127
89             CAN_MASTER = rxMsg.data[2] & 0x7F; //0~127
90             //保存数据 并重载
91             if (!prefs.ready()) prefs.open();
92             prefs.flush(); // Write new prefs to flash
93             prefs.close();
94             prefs.load();
95             can.filter(CAN_ID, 0x07FF, CANStandard, 0); //重新设置滤波器
96             txMsg.len = 0x04;
97             txMsg.id = 0x7ff;
98             memcpy(txMsg.data, rxMsg.data, 4); //返回
99             can.write(txMsg);
100             txMsg.len = 0x06;
101             txMsg.id = CAN_MASTER;
102         }
103         else if((rxMsg.len == 4) && (rxMsg.data[0]==0x55) && (rxMsg.data[3]==0xAA) )
104         {
105             can.filter(CAN_ID, 0x07FF, CANStandard, 0); //重新设置滤波器
106             txMsg.id = 0x7ff;
107             txMsg.len = 0x04;
108             txMsg.data[0] = 0x55;
109             txMsg.data[1] = CAN_ID&0xFF;
110             txMsg.data[2] = CAN_MASTER&0xFF;
111             txMsg.data[3] = 0xAA;
112             can.write(txMsg);
113             txMsg.len = 0x06;
114             txMsg.id = CAN_MASTER;
115         }
116     }
117 }
118 }

```