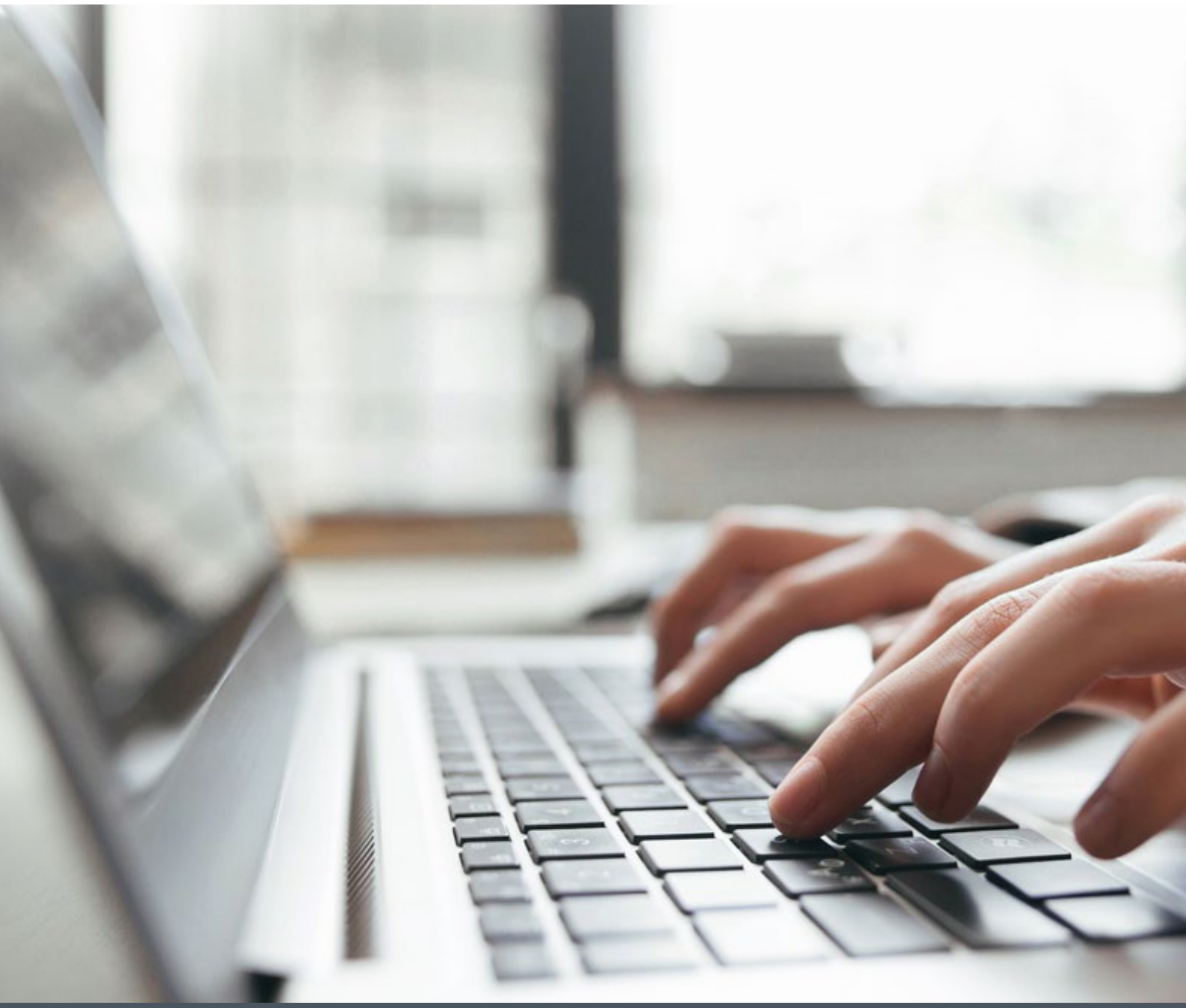

Application Note

Library for NanoJ Programming

Version 1.0.0

www.yanlanmc.com

一、功能说明

下面将描述 nanoj 库文件的函数说明

1.1 DigitalInputHigh(x);

参数：1,2,3,4,5 或者 6
返回：true/false
描述：如果 input 1,2,3,4,5 或者 6 被触发高电平，返回 true
功能：返回 60FD 的 bit16,17,18,19,20 或者 21
用法举例：
If(DigitalInputHigh(1))　　input1 被触发高电平
while(DigitalInputHigh(1))　　input1 高电平状态下一直循环执行某段程序

1.2 DigitalInputLow(x);

参数：1,2,3,4,5 或者 6
返回：true/false
描述：如果 input 1,2,3,4,5 或者 6 被触发低电平，返回 true
功能：返回 60FD 的 bit16,17,18,19,20 或者 21
用法举例：
If(DigitalInputLow(1))　　input1 被触发低电平
while(DigitalInputLow(1))　　input1 低电压状态下一直循环执行某段程序

1.3 TrigerControlbit(U08 controlbit);

参数：bit 位触发对应的十进制数
返回：true/false
描述：如果 bit 位被触发，返回 true
功能：判断 2400:01 某一位是否被置 1
用法举例：
If(TrigerControlbit(1))　判断 bit1 是否被触发

1.4 NontrigerControlbit(U08 controlbit);

参数：bit 位对应的十进制数
返回：true/false
描述：如果 bit 位不被触发，返回 true
功能：判断 2400:01 某一位是否被置 0
用法举例：

If(NontrigerControlbit(1))　判断 bit1 是否==0

1.5 AbsoluteMode();

参数：

返回：

描述：执行绝对位置模式，在执行该函数前，需要配置好速度和目标位置。执行该函数后，需要紧跟一个 do{ } while ((In.StatusWord & 0x400)!=0x400);　do 循环中加入各种用户逻辑。执行完 while ((In.StatusWord & 0x400)!=0x400); 代表电机运行到位。

内部功能：模式 6060 设置 1　控制字 6040 写入 6,7,0xF,0x1F

1.6 RelativeMode();

参数：

返回：

描述：执行相对位置模式，在执行该函数前，需要配置好速度和目标位置。执行该函数后，需要紧跟一个 do{ } while ((In.StatusWord & 0x400)!=0x400); do 循环中加入各种用户逻辑。执行完 while ((In.StatusWord & 0x400)!=0x400); 代表电机运行到位。

内部功能：模式 6060 设置 1　控制字 6040 写入 6,7,0x4F,0x5F

1.7 VelocityMade();

参数：

返回：

描述：执行速度模式，需要先配置好速度和方向。函数后需要紧跟 while(true) { }，　循环中加入停止运行的条件。

内部功能：模式 6060 设置 3　控制字 6040 写入 6,7,0xF

1.8 HomingMade();

参数：

返回：

描述：执行找零模式，需要先配置找零模式，执行该函数后，需要紧跟 do{ } while ((In.StatusWord & 0x1400)!=0x1400); do 循环中加入各种停止或者跳出条件

内部功能：模式 6060 设置 6　控制字 6040 写入 6,7,0xF,0x1F

1.9 AutoSetup (int controlbits ,int urgentstop,int encoder,int stepbldc);

参数：controlbits 与进入该模式的触发数据位保持一致，urgentstop 急停 bit 位；编码器线束*4；stepbldc：0-步进 1-直流无刷

返回：执行完毕后，检查 2500:0x20 如果等于 6666 代表找零成功

描述：自适应模式，自动检测编码器信号

1.10 SetDigitalOutput(U08 outputs);

参数：1、2、3

返回：

描述：Sets the digital output 1, 2 or 3

功能: Sets Bit 16, 17, 18 or 19 of 0x60FE:01

1.11 ClearDigitalOutput(U08 outputs);

参数：1、2、3

返回：

描述：关闭数字输出 1, 2 or 3

功能: Clears Bit 16, 17, 18 or 19 of 0x60FE:01

1.12 Analogposition(int Target_Pos);

参数：Target_Pos 最大位移量对应的脉冲数

返回：

描述：走模拟量定位

功能: 定位模式下，由模拟量决定目标定位

1.13 GetAnalogVelocity(int max_vel)；

参数：max_vel 最大模拟量值对应的最大速度

返回：

描述：走模拟量调速

功能: 速度或者定位模式下的模拟量调速

1.14 AbsoluteMove(int controlbits,int halt,int urgentstop,int leftlimit_port,int rightlimit_port);

参数：设置触发位，暂停位，急停位，左限位 Input 端口，右限位 Input 端口

返回：

描述：自动走绝对定位模式

备注：该模式适合于能够外部触发状态变化的应用场景，否则无法跳出函数中的循环

1.15 VelocityMove (int controlbits ,int halt , int urgentstop,int digitallimit_port);

参数：设置触发位，暂停位，急停位，限位 Input 端口

返回：

描述：自动速度模式

备注：该模式适合于能够外部触发状态变化的应用场景，否则无法跳出函数中的循环

1.16 RelativeMove(int controlbits,int halt,int urgentstop,int leftlimit_port,int rightlimit_port);

参数：设置触发位，暂停位，急停位，左限位 Input 端口，右限位 Input 端口

返回：

描述：自动相对定位模式

备注：该模式适合于能够外部触发状态变化的应用场景，否则无法跳出函数中的循环

1.17 HomingMove (int controlbits ,int urgentstop,int digitallimit_port);

参数：设置触发位，暂停位，急停位，限位 Input 端口

返回：

描述：自动找零模式

备注：该模式适合于能够外部触发状态变化的应用场景，否则无法跳出函数中的循环

二、范例说明

2.1 相对定位

```
if( TrigerControlbit(1) && NontrigerControlbit(11) )   //相对定位
{
    InOut.Control = 0;
    Out.TargetPosition = Target_Pos1;
    RelativeMode();
    do  {
        yield();
        if( TrigerControlbit(11) )    //急停  2400:01=0x800
            {
                yield();
                Out.ControlWord = Out.ControlWord& 0xFF70;
                break;
            }
            if( TrigerControlbit(12) )    //暂停  2400:01=0x1000
               {
                  Out.ControlWord = Out.ControlWord | 0x0100;

                  while( TrigerControlbit(12) )
                      {
                        yield();
                      }
                   while(true)
                      {
                         yield();
                         if( TrigerControlbit(1) && NontrigerControlbit(12) )
                         {
                            InOut.Control = 0;
                            Out.ControlWord = Out.ControlWord & 0xFEFF;
                            yield();
                            od_write(0x2500,0x20,11);
                            yield();
                            break;

                         }
                      }
                  }
        }
    while ( ( In.StatusWord & 0x400)!=0x400);
    //InOut.Control = 0x04;
}
```

利用控制位 bit1 做为启动位，bit11 做为急停位　bit12 做为暂停位