PWM-PSO

控

制

器

使

用

手

册

版本： V20210623(V3.0)

目 录

一、串口命令说明

1、串口通讯参数：

请使用直连或者 USB 转 RS232 串口线；

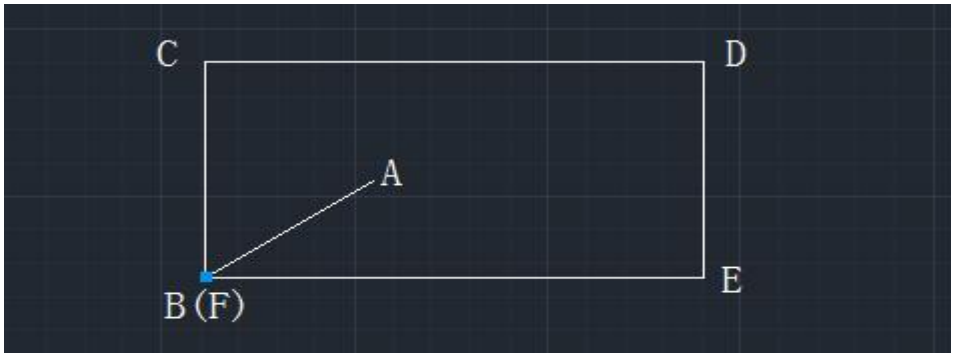
串口参数为：

波特率：115200；奇偶校验：N；数据位：8；停止位：1；

2、高级 PWM 原理解释：

高级 PWM 模式：在激光轨迹加工过程中，不管是在加速、减速、匀速过程，等距离同步输出 PWM 激光控制信号，确保每个点的能量一致，达到最佳切割品质。

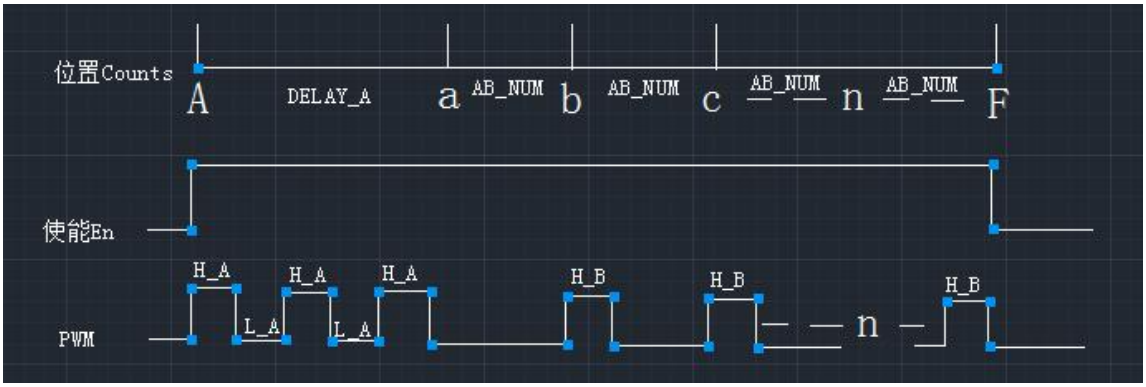
如下图示一个切割路径为 A->B->C->D->E->F：



A->B 段控制参数，解决首点击穿问题（根据需要来设定参数）：

参数	指令	说明
1	0x55 aa 10 xx xx xx xx	PARA_pwm_h_a 开光首点高电平时间（单位：us）
2	0x55 aa 12 xx xx xx xx	PARA_pwm_l_a 开光首点低电平时间（单位：us）
3	0x55 aa 14 xx xx xx xx	PARA_dly_a 开光首点延时时间（单位：us）

时序图如下：

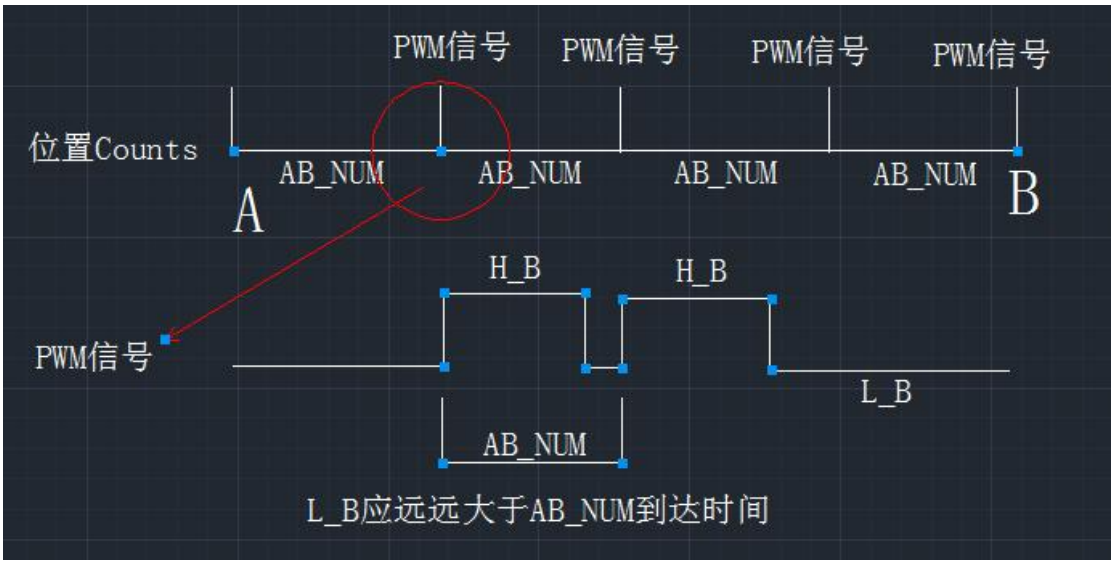


说明：

设置好间距参数（H_A，L_A，Delay_A,H_B，L_B,AB_NUM）等，使能 EN 导通，准备就绪，再平台移动有编码器计数，触发产生 5/24V 的 PWM 信号，来控制激光器出光;

B->C->D->E->F 段控制参数：

参数	指令	说明
1	0x55 aa 16 xx xx xx xx	PARA_AB_NUM 点间距（单位：count）
2	0x55 aa 18 xx xx xx xx	PARA_pwm_h_b 脉冲输出高电平时间（单位：us）
3	0x55 aa 1a xx xx xx xx	PARA_pwm_l_b 脉冲输出低电平时间（单位：us）



说明：按设定的点间距，到位置优先出光，假如速度非常慢的情况，超过低电平还没到位置，低电平时间到了，就出一个高电平脉冲，这样保证有一个最低频率的 PWM 信号。

目前此卡支持四种 PWM 信号输出：

四种 PWM 信号通道切换指令（**十六进制**）：

模式	指令	说明
PWM_1	0x55 aa 42 00 10	高级脉冲 PSO 输出，支持开始段按固定频率的脉冲同步输出，适用于金属切割等，保证首段击穿
PWM_2	0x55 aa 42 00 20	标准 PWM，按设定频率、脉宽出光，适用用于测试
PWM_3	0x55 aa 42 00 30	标准脉冲同步输出，适用于皮秒玻璃切割等
PWM_4	0x55 aa 42 00 40	带首脉冲抑制的脉冲同步输出，适用于 CO2 激光器等

A: PWM_1 模式(首点穿孔 PSO)参数：

参数	指令	说明
1	0x55 aa 10 xx xx xx xx	PARA_pwm_h_a 开光首点高电平时间（单位：us）
2	0x55 aa 12 xx xx xx xx	PARA_pwm_l_a 开光首点低电平时间（单位：us）
3	0x55 aa 14 xx xx xx xx	PARA_dly_a 开光首点延时时间（单位：us）
4	0x55 aa 16 xx xx xx xx	PARA_AB_NUM 点间距（单位：count）
5	0x55 aa 18 xx xx xx xx	PARA_pwm_h_b 脉冲输出高电平时间（单位：us）
6	0x55 aa 1a xx xx xx xx	PARA_pwm_l_b 脉冲输出低电平时间（单位：us）

B: PWM_2 模式（标准 PWM）参数：

参数	指令	说明
1	0x55 aa 20 xx xx xx xx	PARA_pwm_h_a 高电平时间（单位：us）
2	0x55 aa 22 xx xx xx xx	PARA_pwm_l_a 低电平时间（单位：us）

C: PWM_3 模式(标准 PSO)参数：

参数	指令	说明
1	0x55 aa 16 xx xx xx xx	PARA_AB_NUM 点间距（单位：count）
2	0x55 aa 18 xx xx xx xx	PARA_pwm_h_b 脉冲输出高电平时间（单位：us）
3	0x55 aa 1a xx xx xx xx	PARA_pwm_l_b 脉冲输出低电平时间（单位：us）

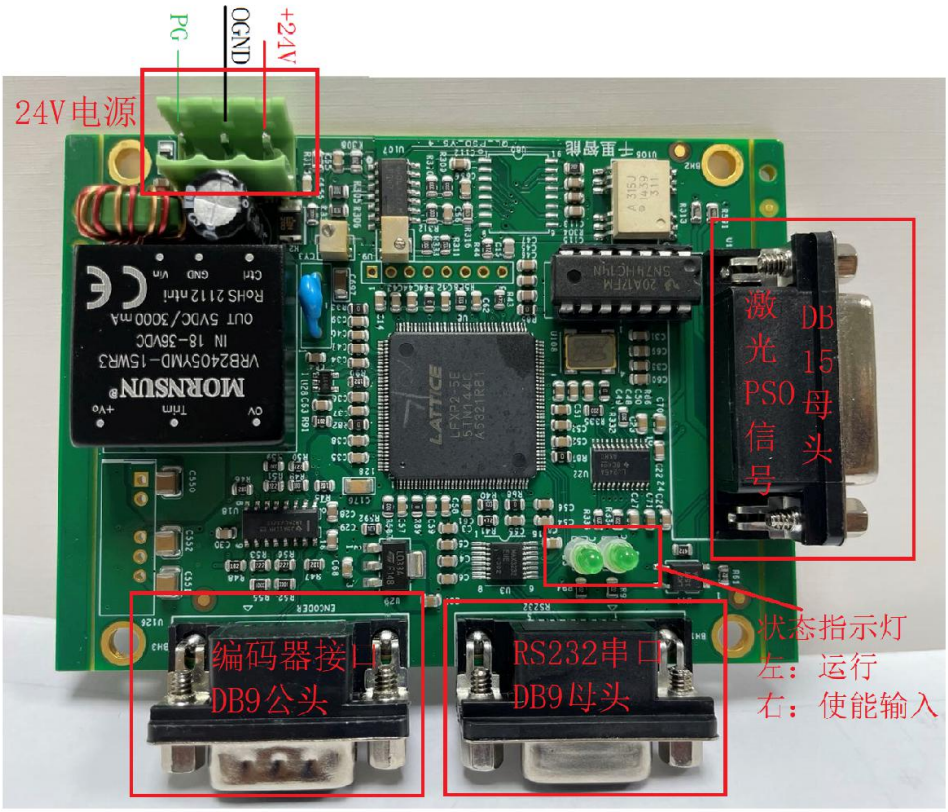
D: PWM_4 模式(带首脉冲抑制的 PSO)参数：

参数	指令	说明
1	0x55 aa 10 xx xx xx xx	PARA_pwm_h_a 首脉冲高电平时间（单位：us）
2	0x55 aa 12 xx xx xx xx	PARA_pwm_l_a 首脉冲步长（单位：count）
3	0x55 aa 16 xx xx xx xx	PARA_AB_NUM 点间距（单位：count）
4	0x55 aa 18 xx xx xx xx	PARA_pwm_h_b 脉冲输出高电平时间（单位：us）
5	0x55 aa 1a xx xx xx xx	PARA_pwm_l_b 脉冲输出低电平时间（单位：us）

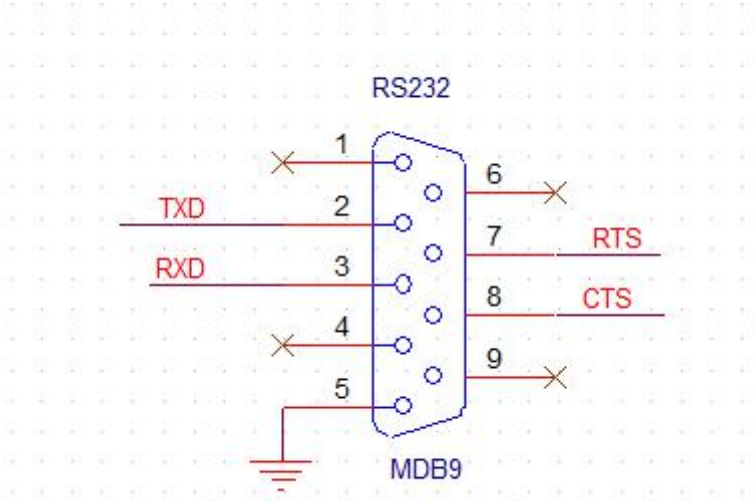
注意：不同模式下，参数的含义不一样，最后出来的效果不一样。

电压 DAC 输出指令

参数	指令	说明
1	0x55 aa 40 xx xx	例如： 0V 对应命令 0x55 aa 40 00 00；10V 对应命令 0x55 aa 40 7F FF

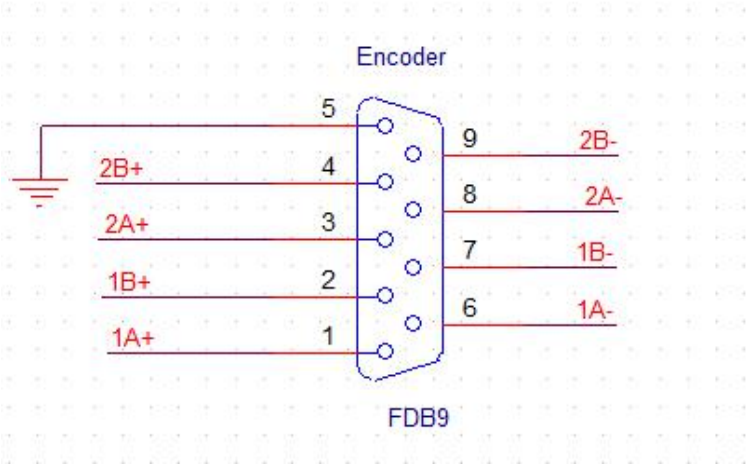


二．RS232 串口通信端口定义(9 PIN 母头)



注：请做直连串口线，DB9 的 2、3、5 脚

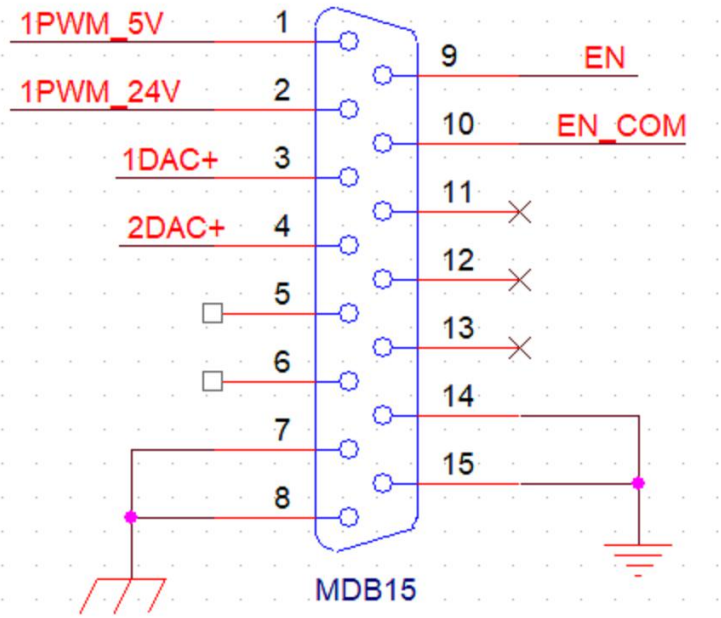
三．ENCODER 端口定义(9PIN 公头)



注：仅支持增量式 AB 正交的编码器信号输入，1 为 1#轴；2 为 2#轴；

管脚	名称	说明
1、6	1A+/1A-	1#轴编码器 A+/编码器 A-
2、7	1B+/1B-	1#轴编码器 B+/编码器 B-
3、8	2A+/2A-	2#轴编码器 A+/编码器 A-
4、9	2B+/2B-	2#轴编码器 B+/编码器 B-
5	GND	接地脚（内部地）

四. 激光 PSO 信号接口(15PIN 母头)



管脚	名称	说明
1	1PWM_5V	输出为 5V 的 TTL 信号
2	1PWM_24V	输出为 24V 的 TTL 信号
3	1DAC+	输出第 1 路±10V 的模拟量
4	2DAC+	输出第 2 路±10V 的模拟量
9	EN	触发使能信号，输入，内部为光耦
10	EN_COM	触发使能信号的公共端，可以实现高低电平切换
7/8	OGND	接地脚（外部电源地）
14/15	GND	接地脚（内部地）
5/6/11/12/13	---	悬空未用

注：使能信号（5~24V）：

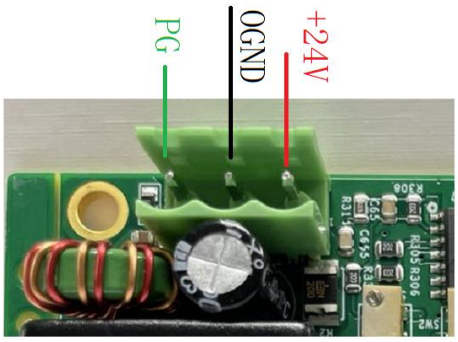
一般是由运动控制器的 IO 发出，我们卡内为光耦，导通时指示灯 D4 会亮，设置好间距参数等，使能 EN 导通，准备就绪，再使平台移动，编码器开始计数，触发产生 5/24V 的 PWM 信号，来控制激光器出光，接 1/2、14 脚。

根据激光器控制方式不一样，请按需要接线、区分 5V/24V 模式：

一般 CO2 激光器、SPI 连续光纤激光器只接 PWM 信号，接 1、14 脚；

IPG 连续光纤激光器需要 PWM 和 DAC（电压）配合使用，PWM 接 1、14，电压 3、15 脚；

五. 电源端口定义



电源 24V 电流不小于 1A，请注意方向和顺序！

管脚	名称	说明
1	+24V	+24V 输入，电流大于 1A
2	OGND	+24V 输入地（外部电源地）
3	PG	外壳大地（建议不接）

六．参考程序界面及例子程序：

建议，集成到工艺操作软件参数设置界面。

计算公式：

f=KV/AB_NUM

f:频率；K：编码器分辨率，counts；
V：速度，mm/s;AB_NUM：位置间距，counts；

例如：K=10000，即 0.1um 光栅尺，V=30mm/s,AB_NUM=60
f=10000*30/60=5000Hz
最小频率为 5000=1/(H_B+L_B)

注意：使用高级 PWM 模式时，L_B 应尽量大，远远大于 AB_NUM 到达时间

程序代码：PWM_TEST_BASIC.h

```
#pragma once
#include "HIPWM.h"
```

```
#include "HIRS232.h"

class PWM_TEST_BASIC : public HIPWM
{
public:
    PWM_TEST_BASIC(void);
    ~PWM_TEST_BASIC(void);

    //实现 CDevice 虚方法
    virtual BOOL InitDevice();
    virtual BOOL OpenDevice();
    virtual void CloseDevice();

    //实现 IConnectedDevice 接口
    virtual void SetCommPara(const TConnectorPara& aCommPara);
    virtual CConnector& GetConnector();

    //实现 HIPWM 虚方法
    //设置初始化参数
    virtual BOOL SetInitPara(const TPWMPara& aPara);
    //设置频率（单位：hz）
    virtual BOOL SetFreq(double fFreq);
    //设置脉宽（单位：us）
    virtual BOOL SetPulseWidth(double fPulseWidth);
    //设置模拟电压（单位：v）
    virtual BOOL SetAnalogVoltage(double fAnalogVolt);
    //启用 SPC 模式（若不支持此功能，则返回 FALSE）
    virtual BOOL EnableSPC(BOOL bEnable);
    //设置 SPC 参数
    virtual BOOL SetSPCPara(int nPWM_H_A,int nPWM_L_A,int nDELAY_A,int
nAB_NUM,int nPWM_H_B,int nPWM_L_B);

private:
    //执行命令
    BOOL ExecuteCmd(CByteArray& ba);
private:
    HIRS232 m_serialPort;

    double m_fFreq;          //频率（hz）
    double m_fPulseWidth;//脉宽（us）

    BOOL m_bEnableSPC;//是否启用 SPC（启用后所有对标准 PWM 的调用直接
```

返回成功，以兼容上层应用）

```
    int m_nPWM_H_A;//开光首点高电平时间（单位：us）
    int m_nPWM_L_A;//开光首点低电平时间（单位：us）
    int m_nDELAY_A;//开光首点延时时间（单位：us）
    int m_nAB_NUM; //SPC 步长（单位：count）
    int m_nPWM_H_B;//SPC 高电平时间（单位：us）
    int m_nPWM_L_B;//SPC 低电平时间（单位：us）
};
```

PWM_TEST_BASIC.cpp

```
#include "StdAfx.h"
#include "PWM_TEST_BASIC.h"

//****pwm_module*****************************************************
//0x55 aa 10 xx xx xx xx----PARA_pwm_h_a----
//0x55 aa 12 xx xx xx xx----PARA_pwm_l_a----
//0x55 aa 14 xx xx xx xx----PARA_dly_a    ----
//0x55 aa 16 xx xx xx xx----PARA_AB_NUM ----
//0x55 aa 18 xx xx xx xx----PARA_pwm_h_b----
//0x55 aa 1a xx xx xx xx----PARA_pwm_l_b----
//****pwm_only_module************************************************
//0x55 aa 20 xx xx xx xx----PARA_pwm_h_a----//55aa 20 00 00 00 08//02 fa f0 80
//0x55 aa 22 xx xx xx xx----PARA_pwm_l_a----//55aa 22 00 00 00 08//02 fa f0 80
//****ad1868_in_nopll************************************************
//0x55 aa 40 xx xx----adc data in----
//0x55 aa 42 xx xx----control parameters: 8'h10 = pwm_spc;8'h20 = pwm_std;
//****pwm_module*****************************************************
//0x55 aa 80
//PARA DATA LEN：32bit
//ADC DATA IN：    16bit
//Freq:50MHZ(Period:20ns)
//*******************************************************************
#define BASICTEST_PERIOD          10 //unit:ns(50MHZ->20ns->hi 10s lo 10s)
#define MAX_ANALOG_VOLT_ABS     10 //unit:v

PWM_TEST_BASIC::PWM_TEST_BASIC(void)
{
    //设置默认频率及脉宽
    m_fFreq=1.0;m_fPulseWidth=1.0;
```

```
    m_bEnableSPC = FALSE;//默认不启用
}

PWM_TEST_BASIC::~PWM_TEST_BASIC(void)
{
}

BOOL PWM_TEST_BASIC::SetInitPara(const TPWMPara& aPara)
{
    //检查参数型号
    if(TPWMPara::PWM_BASICTEST != aPara.GetModel())
        return FALSE;

    //检查参数范围并保存所设置参数
    //TBasicTestPara& para = (TBasicTestPara&)aPara;
    //TODO:copy para

    return TRUE;
}

BOOL PWM_TEST_BASIC::InitDevice()
{
    //按照文档中串口参数进行设置（替代默认设置）
    m_serialPort.SetSettings(_T("115200,N,8,1"));

    //初始化串口
    if(!m_serialPort.Init())
        return FALSE;

    return TRUE;
}

BOOL PWM_TEST_BASIC::OpenDevice()
{
    //打开串口
    if(!m_serialPort.IsOpen())
    {
        if(!m_serialPort.Open())
        {
            return FALSE;
        }
    }
```

```
    return TRUE;
}

void PWM_TEST_BASIC::CloseDevice()
{
    //关闭串口
    m_serialPort.Close();
}

void PWM_TEST_BASIC::SetCommPara(const TConnectorPara& aCommPara)
{
    m_serialPort.SetPara((const TRS232Para&)aCommPara);
}

CConnector& PWM_TEST_BASIC::GetConnector()
{
    return m_serialPort;
}

//执行单个命令
BOOL PWM_TEST_BASIC::ExecuteCmd(CByteArray& ba)
{
    //必须确保通讯端口打开
    if(!m_serialPort.IsOpen())
        return FALSE;

    //执行命令（不查看返回值）
    CByteArray recvData;
    m_serialPort.SendData(ba,50);

    return TRUE;
}

//设置频率（单位：hz）
BOOL PWM_TEST_BASIC::SetFreq(double fFreq)
{
    BOOL bRet1=TRUE,bRet2=TRUE;

    //检查频率参数范围
    int nFreq=(int)fFreq;
    if(nFreq<=0) return FALSE;
```

```
//保存频率
m_fFreq = fFreq;

//检查周期是否覆盖脉宽
char szCmd[256]={0};
int nPulseWidth=(int)m_fPulseWidth;
int nPeriod = 1000000 / nFreq;//周期（单位：us）
if(nPeriod<=nPulseWidth || nPulseWidth<=0)
{
    return FALSE;
}

//SPC 模式直接返回成功
if(m_bEnableSPC)
{
    return TRUE;
}

//计算参数
m_nPWM_H_A = nPulseWidth;
m_nPWM_L_A = nPeriod - nPulseWidth;

//构造命令(PWM_H_A)
CByteArray ba;
ba.Add(0x55);
ba.Add(0xaa);
ba.Add(0x20);
int n_h_a =m_nPWM_H_A*1000/BASICTEST_PERIOD;//us->times of Period
ba.Add((n_h_a>>24)&0xFF);
ba.Add((n_h_a>>16)&0xFF);
ba.Add((n_h_a>>8)&0xFF);
ba.Add(n_h_a&0xFF);
//执行命令（PWM_H_A)
bRet1=ExecuteCmd(ba);

//构造命令(PWM_L_A)
ba.RemoveAll();
ba.Add(0x55);
ba.Add(0xaa);
ba.Add(0x22);
int n_l_a =m_nPWM_L_A*1000/BASICTEST_PERIOD;//us->times of Period
```

```
    ba.Add((n_l_a>>24)&0xFF);
    ba.Add((n_l_a>>16)&0xFF);
    ba.Add((n_l_a>>8)&0xFF);
    ba.Add(n_l_a&0xFF);
    //执行命令（PWM_L_A)
    bRet2=ExecuteCmd(ba);

    return bRet1 && bRet2;
}

//设置脉宽（单位：us）
BOOL PWM_TEST_BASIC::SetPulseWidth(double fPulseWidth)
{
    char szCmd[256]={0};
    m_fPulseWidth = fPulseWidth;

    return SetFreq(m_fFreq);
}

//设置模拟电压（单位：V，范围-10~+10V）
BOOL PWM_TEST_BASIC::SetAnalogVoltage(double fAnalogVolt)
{
    //检查并处理参数
    if(fAnalogVolt<0.0)
    {
        fAnalogVolt=0.0;
    }else if(fAnalogVolt>MAX_ANALOG_VOLT_ABS){
        fAnalogVolt=MAX_ANALOG_VOLT_ABS;
    }else if(fAnalogVolt< -MAX_ANALOG_VOLT_ABS){
        fAnalogVolt= -MAX_ANALOG_VOLT_ABS;
    }

    //构造命令
    CByteArray ba;
    ba.Add(0x55);
    ba.Add(0xaa);
    ba.Add(0x42);
    short n_dac=(short) (fAnalogVolt*32767/MAX_ANALOG_VOLT_ABS);//16 bit
dac
    ba.Add((n_dac>>8)&0xFF);
    ba.Add(n_dac&0xFF);
```

```
    //执行命令
    return ExecuteCmd(ba);
}

//启用 SPC 模式
BOOL PWM_TEST_BASIC::EnableSPC(BOOL bEnable)
{
    CByteArray ba;
    ba.Add(0x55);
    ba.Add(0xaa);
    ba.Add(0x42);
    ba.Add(0x00);
    if(bEnable)
    {//pwm_spc
        ba.Add(0x10);
    }else{//pwm_std
        ba.Add(0x20);
    }

    BOOL bRet= ExecuteCmd(ba);
    m_bEnableSPC = bEnable;

    return TRUE;
}

//设置 SPC 参数
BOOL PWM_TEST_BASIC::SetSPCPara(int nPWM_H_A,int nPWM_L_A,int
nDELAY_A,int nAB_NUM,int nPWM_H_B,int nPWM_L_B)
{
    BOOL bRet[6]={TRUE};
    BOOL bRetVal=TRUE;

    //保存参数
    m_nPWM_H_A = nPWM_H_A;//0x10
    m_nPWM_L_A = nPWM_L_A;//0x12
    m_nDELAY_A = nDELAY_A;//0x14
    m_nAB_NUM   = nAB_NUM; //0x16
    m_nPWM_H_B = nPWM_H_B;//0x18
    m_nPWM_L_B = nPWM_L_B;//0x1a

    CByteArray ba;
    int
```

```
ar[6]={nPWM_H_A,nPWM_L_A,nDELAY_A,nAB_NUM,nPWM_H_B,nPWM_L_
B};
    for(int i=0;i<6;i++)
    {
        //构造命令
        ba.RemoveAll();
        ba.Add(0x55);
        ba.Add(0xaa);
        char cCmd = 0x10+i*2;
        ba.Add(cCmd);
        int n=0;
        if(cCmd == 0x16){
            n = ar[i];//count
        }else if(cCmd == 0x14){
            int nPWM_T = nPWM_H_A+nPWM_L_A;
            if(nPWM_T<=0) nPWM_T=1;
            n = nDELAY_A / nPWM_T;
        }else{
            n = ar[i] * 1000/BASICTEST_PERIOD;//us->times of Period
        }
        ba.Add((n>>24)&0xFF);
        ba.Add((n>>16)&0xFF);
        ba.Add((n>>8)&0xFF);
        ba.Add(n&0xFF);
        //执行命令
        bRet[i]=ExecuteCmd(ba);

        bRetVal = bRetVal && bRet[i];
    }

    return bRetVal;
}
```

（结束）