# SY-COM 系列模块 DLL 函数说明书

# Developer's Guide

Made By Shuangyi

# SY-COM 系列模块 DLL 函数说明书

# 1.List of Function 函数列表说明

根据所需各项功能提供相应的接口。

## 1.1 通用函数接口列表

| Function Name(函数名称) | 说明 |
|---|---|
| SY_MVD_ComPort_Connect | 串口连接 |
| SY_MVD_Module_Disconnect | 释放端口 |
| SY_MVD_Module_ConnectSts | 模块连接状态查询 |
| SY_MVD_Module_Set_SlaveIP | 改变模块 ID 号 |
| SY_MVD_Module_Get_SlaveIP | 得到当前模块 ID 号 |
| SY_MVD_Module_Get_AllSlaveIP | 查询连接的模块 ID |
| SY_MVD_Module_SaveParamToFlash | 参数保存到 Flash |
| SY_MVD_Module_Get_ManuInfo | 获取生产信息 |
| SY_MVD_Module_Get_CurrentVer | 获取模块版本 |
| SY_MVD_Module_Set_Baudrate | 设置串口波特率 |
| SY_MVD_Module_Get_Baudrate | 得到当前设定的串口波特率 |
| SY_MVD_Module_Get_CardInfo | 获取板卡信息 |

# 1.2 通用端口函数说明

### 1.2.1 串口连接

long long long long SY_MVD_ComPort_Connect(int PortNum, int Baudrate_type)
说明：串口端口连接
输入参数：
PortNum：串口号，数值 1~9
Baudrate_type：波特率，数值为 0/1/2/3.

| Baudrate_type(波特率) | 描述 |
|---|---|
| 0 | 2400bps |
| 1 | 9600bps |
| 2 | 38400bps |
| 3 | 115200bps |

输出参数：资源句柄（long long hSMDevice）

### 1.2.2 释放端口

BOOL SY_MVD_Module_Disconnect（long long hSMDevice）
说明：释放端口
输入参数：资源句柄（long long hSMDevice）
输出参数：TRUE ：成功； FALSE ：失败

### 1.2.3 设置模块 ID 号

BOOL SY_MVD_Module_Set_SlaveIP (int BoardType,long long hSMDevice, int SlaveIP,int NewIP)
说明：设置串口端口号
输入参数：
BoardType：模块类型

| BoardType | 描述 |
|---|---|
| 0x00 | Board_Light:带光源控制的控制板 |
| 0x01 | Board_DIO：仅有 IO 功能的控制板 |

hSMDevice：资源句柄
SlaveIP：当前 ID ，ID 数值范围 1~32
NewIP:设置新 ID ，ID 数值范围 1~32
输出参数:
返回 True 成功；返回 False 表示端口设置失败

### 1.2.4 获得当前使用模块 ID 号

BOOL SY_MVD_Module_Get_SlaveIP (int BoardType,long long hSMDevice, int *SlaveIPIndex)
说明：得到当前使用串口 ID 号
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
输出参数：TRUE ：成功； FALSE ：失败

### 1.2.5 查询模块链接状态

bool SY_MVD_Module_ConnectSts (int BoardType,long long hSMDevice, int SlaveIP)
说明：查询端口链接状态
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：当前 ID
输出参数；返回 True 成功；返回 False 表示端口设置失败

### 1.2.6 保存到 Flash

BOOL SY_MVD_Module_SaveParamToFlash(int BoardType,long long hSMDevice, int SlaveIP)
说明：将设置的参数存入 Flash
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
输出参数：
TRUE ：操作成功； FALSE ：失败

### 1.2.7 获得链路上所有连接的模块 ID 号

BOOL SY_MVD_ Module_Get_AllSlaveIP (int BoardType,long long hSMDevice, int *AllIP , int *IPcount)
说明：获得链路上所有连接的模块 ID 号。
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
AllIP：总共允许接驳的模块数量号，可以接驳 63 个模块;
IPcount：得到当前连接模块的数量;
输出参数：TRUE ：获取成功； FALSE ：获取失败

### 1.2.8 设置通讯波特率

BOOL SY_MVD_Module_Set_Baudrate(int BoardType,long long hSMDevice, int SlaveIP,int Baudrate_type);

说明：设置通讯波特率

输入参数：

BoardType：模块类型

hSMDevice：资源句柄

SlaveIP：控制器的 ID（1~32）

Baudrate_type：波特率速率，数值为0/1/2/3.

| Baudrate_type(波特率) | 描述 |
|---|---|
| 0 | 2400bps |
| 1 | 9600bps |
| 2 | 38400bps |
| 3 | 115200bps |

输出参数：

TRUE ：操作成功； FALSE ：失败

### 1.2.9 得到通讯波特率

BOOL SY_MVD_Module_Get_Baudrate(int BoardType,long long hSMDevice, int SlaveIP,int *Baudrate_type);

说明：设置通讯波特率

输入参数：

BoardType：模块类型

hSMDevice：资源句柄

SlaveIP：控制器的 ID（1~32）

Baudrate_type：波特率速率，数值为0/1/2/3.

| Baudrate_type(波特率) | 描述 |
|---|---|
| 0 | 2400bps |
| 1 | 9600bps |
| 2 | 38400bps |
| 3 | 115200bps |

输出参数：

TRUE ：操作成功； FALSE ：失败

### 1.3.0 得到模块生产信息

BOOL SY_MVD_DIO_Get_ManuInfo(int BoardType,long long hSMDevice, int SlaveIP,int *BoardNumb ,unsigned char *BoardVersion ,unsigned char *BoardReversion, int *ManuYear, unsigned char *ManuMonth, unsigned char *ManuDay,int *BatchNumb );

说明：获得生产信息
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
BoardNumb:PCB 板型号
BoardVersion: PCB 硬件版本大版本号
BoardReversion: PCB 硬件版本小版本号
ManuYear:生产年份
ManuMonth:生产月份
ManuDay:生产日
BatchNumb: 生产批次

### 1.3.1 得到模块 MCU 信息

BOOL SY_MVD_DIO_Get_CurrentVer(int Connect_Num , int SlaveIP, BYTE *MCUVersion ,BYTE *MCUReversion, BYTE *PLDVersion ) ;
说明：获取模块 MCU 版本信息

输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
MCUVersion:MCU 大版本号
MCUReversion: MCU 小版本号
PLDVersion: PLD 版本号

### 1.3.2 获得串口模块信息

BOOL SY_MVD_Module_Get_CardInfo(int BoardType,long long hSMDevice, int SlaveIP , unsigned char *ProductType, unsigned char *ProductNo ,int *BoardNumb )
说明：获得光源控制器板卡信息
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：模块 ID 号，控制器的 ID（1~32）
ProductType:

| ProductType | 描述 |
|---|---|
| 1 | 光源 SYLight 系列 |
| 2 | IO 模块 SYCOM 系列 |

ProductNo: 产品的编号, SYLight 31 、SYCOM02 等...
BoardNumb: PCB 板的编号，和 Get_ManuInfo 里面的板号一样 。如：SN261
输出参数：TRUE ：成功； FALSE ：失败

例如:

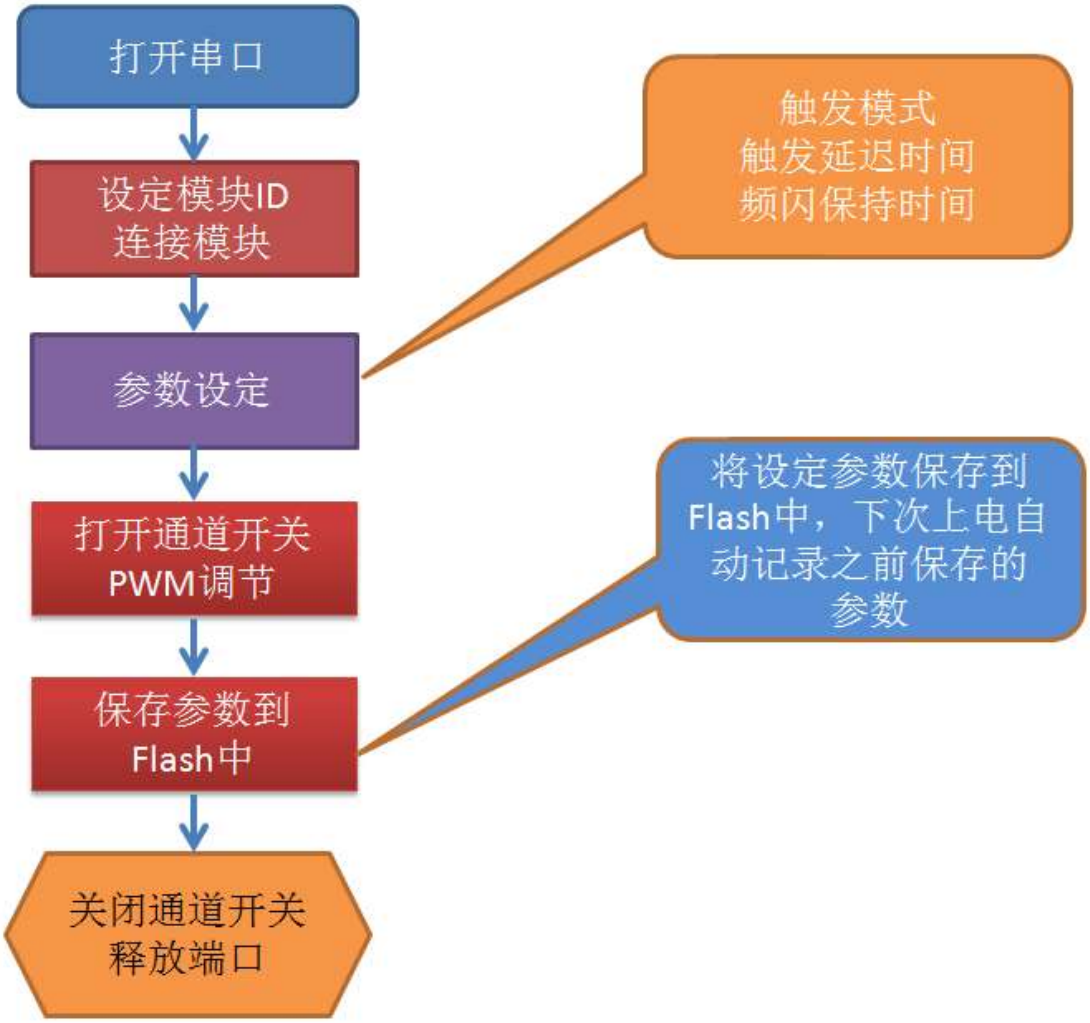
```
SY_MVD_Light_Get_CardType(BoardType , hSMDevice ,m_nEdtSlaveIP,
&ProductType ,&ProductNo)  ;
if(ProductType == 1)
{
    strTmp.Format("产品型号：        SYLight%02d   \r\n" , ProductNo ) ;
}

if(ProductType == 2)
{
    strTmp.Format("产品型号：        SYCOM%02d   \r\n" , ProductNo ) ;
}
```

# 2.光源控制器命令列表

| Function Name(函数名称) | 说明 |
|---|---|
| SY_MVD_Light_Slave_Init | 模块初始化 |
| SY_MVD_Light_Set_Intensity | 设置单通道单亮度 |
| SY_MVD_Light_Get_Intensity | 得到单通道亮度值 |
| SY_MVD_Light_SetChStsOnOff | PWM 模式下打开或者关闭某个通道，或者频闪 |
| SY_MVD_Light_SetCHMode | 光源通道模式(PWM 或者触发模式) |
| SY_MVD_Light_GetCHMode | 得到光源通道模式(PWM 或者触发模式) |
| SY_MVD_Light_Set_TriggerDelay | 光源通道触发延迟时间 |
| SY_MVD_Light_Get_TriggerDelay | 得到光源通道触发延迟时间 |
| SY_MVD_Light_Set_StrobeDuration | 光源通道频闪保持时间 |
| SY_MVD_Light_Get_StrobeDuration | 得到光源通道频闪保持时间 |
| SY_MVD_Light_Set_CommMod | 设置窜口通讯模式 |
| SY_MVD_LoadLightParam | 将参数存到文件中 |
| SY_MVD_SaveLightParam | 从文件中读取参数 |
| SY_MVD_Light_SelChn | 软件切换模块的通道 |
| SY_MVD_Light_Set_AllIntensity | 设定 4 个通道亮度值 |
| SY_MVD_Light_Set_AllChStsOnOff | 设定打开或者关闭 4 个通道状态 |
| SY_MVD_Light_Get_AllChStsOnOff | 获得设定 4 个通道的状态 |
| SY_MVD_Light_Set_Intensity_ChanelSts | 独立控制 4 个通道 PWM 值和开关状态 |
| SY_MVD_Light_Set_CurrentOut | 设置通道电流输出值 |
| SY_MVD_Light_Get_CurrentOut | 得到当前通道电流输出值 |
| SY_MVD_Light_Set_AllCurrentOut | 设置所有通道电流输出值 |
| SY_MVD_Light_Set_CurrentOut_ChanelSts | 独立控制 4 个通道电流值和开关状态 |
| SY_MVD_Light_GetTrigInStatus | 获得每个通道触发信号状态 |

## 2.1 光源控制器编程流程说明

### 2.1.1 光源控制器的编程流程图




程序框架图

### 2.1.2 打开串口

| Function Name(函数名称) | 说明 |
| --- | --- |
| SY_MVD_ComPort_Connect | 串口连接 |

```
//串口号为 3
int PortNum = 3;
//函数返回的资源句柄
long long    hSMDevice；
//波特率
int Baudrate_type = 1；
```

Baudrate_type：波特率速率

| Baudrate_type(波特率) | 描述 |
| --- | --- |
| 0 | 2400bps |
| 1 | 9600bps |
| 2 | 38400bps |
| 3 | 115200bps |

```
hSMDevice= SY_MVD_ComPort_Connect (PortNum， Baudrate_type);
if (0==hSMDevice)
{
    MessageBox(_T("连接失败！ "));
    return;
}
else
{
    CString    strTmp;
    MessageBox(_T("连接成功!!! "));
    strTmp.Format(_T("连接成功!!! \r\n 当前 Handle =    %d"),hSMDevice)；
}
```

### 2.1.3 设定模块 ID&连接光源控制器模块

| Function Name(函数名称) | 说明 |
| --- | --- |
| SY_MVD_Module_Set_SlaveIP | 改变模块 ID 号 |
| SY_MVD_Module_ConnectSts | 模块连接状态查询 |

```
//模块类型（光源控制器 or 串口 IO 模块）
int BoardType=0;（光源控制器）
//连接模块返回的句柄
long long hSMDevice;
hSMDevice= SY_MVD_ComPort_Connect (PortNum， Baudrate_type);
//模块 ID 默认为 10
int   SlaveNum=10;
//设置新的 ID
int m_nEDtNewIP;
if (SY_MVD_Module_Set_SlaveIP(m_BoardType,hSMDevice,SlaveNum,m_nEDtNewIP)
{
      SlaveNum= m_nEDtNewIP;
}
```

### 2.1.4 参数设定

| Function Name(函数名称) | 说明 |
| --- | --- |
| SY_MVD_Light_SetCHMode | 光源通道模式(PWM 或者触发模式) |
| SY_MVD_Light_Set_TriggerDelay | 光源通道触发延迟时间 |
| SY_MVD_Light_Set_StrobeDuration | 光源通道频闪保持时间 |

```
//设定通道 0 参数，模式为 PWM 常亮模式，触发延迟时间为 10000us
//频闪保持时间为 20000us
//模块类型（光源控制器 or 串口 IO 模块）
int BoardType=0;（光源控制器）
//连接模块返回的句柄
long long hSMDevice;
//模块 ID 默认为 10
int   SlaveIP =10;
SY_MVD_Light_SetCHMode(BoardType, hSMDevice, SlaveIP, 0,0);
SY_MVD_Light_Set_TriggerDelay(BoardType, hSMDevice, SlaveIP,0,10000);
SY_MVD_Light_Set_StrobeDuration( BoardType, hSMDevice, SlaveIP ,0,20000);
```

### 2.1.5 打开通道开关&PWM 调节

| Function Name(函数名称) | 说明 |
|---|---|
| SY_MVD_Light_SetChStsOnOff | PWM 模式下打开或者关闭或频闪 |
| SY_MVD_Light_Set_Intensity | 设置亮度 |
| SY_MVD_Light_Set_AllChStsOnOff | 同步设定打开或者关闭 4 个通道 |
| SY_MVD_Light_Set_AllIntensity | 同步设定 4 个通道的 PWM 值 |

```
//通道 0 打开开关，并设置通道 0 亮度值为 100
SY_MVD_Light_SetChStsOnOff(BoardType, hSMDevice, SlaveIP,0,1);
SY_MVD_Light_Set_Intensity(BoardType, hSMDevice, SlaveIP,, 0, 100);  //设置通道 0 的光源亮度

//同时打开 4 个通道
SY_MVD_Light_Set_AllChStsOnOff(BoardType, hSMDevice, SlaveIP,1);
//同时设定 4 个通道的亮度值,亮度值分别为通道 0:100；通道 1:99;通道 2:98;通道 3:97;
int chvalue0 = 100;
int chvalue1 =99;
int chvalue2 =98;
int chvalue3 =97;
SY_MVD_Light_Set_AllIntensity(BoardType, hSMDevice,SlaveIP,,chvalue0,chvalue1,chvalue2,
chvalue3);
```

### 2.1.6 保存参数到 Flash 中

| Function Name(函数名称) | 说明 |
|---|---|
| SY_MVD_Module_SaveParamToFlash | 参数保存到 Flash |

```
//将参数表保存到 Flash 中
SY_MVD_Light_SaveParamToFlash(BoardType, hSMDevice,SlaveIP);
```

### 2.1.7 关闭通道开关&释放端口

| Function Name(函数名称) | 说明 |
| --- | --- |
| SY_MVD_Light_SetChStsOnOff | PWM 模式下打开或者关闭 |
| SY_MVD_Light_Set_AllChStsOnOff | 同步设定打开或者关闭 4 个通道 |
| SY_MVD_Light_Disconnect | 释放端口 |

```
//通道 0 关闭
SY_MVD_Light_SetChStsOnOff(BoardType, hSMDevice,SlaveIP,0,0);

//同时关闭 4 个通道
SY_MVD_Light_Set_AllChStsOnOff(BoardType, hSMDevice,SlaveIP,0);

//释放端口
SY_MVD_Module_Disconnect (hSMDevice);
```

## 2.2 光源控制器的相关函数

### 2.2.1 光源控制器模块初始化

BOOL SY_MVD_Light_Slave_Init(int BoardType,long long hSMDevice , int SlaveIP)
说明：光源控制器模块初始化，执行此函数后，光源控制器所有参数为出厂设定值。
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：模块 ID 号，控制器的 ID（1~32）
输出参数：TRUE ：成功； FALSE ：失败

### 2.2.2 设置单通道亮度值

BOOL SY_MVD_Light_Set_Intensity(int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int Intesity)
说明：设定光源控制器某个通道的亮度值
输出参数：TRUE ：设置成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
Intesity：通道的亮度值（0~255）

### 2.2.3 得到单通道亮度值

BOOL SY_MVD_Light_Get_Intensity(int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int *Intesity)
说明：查询光源控制器某个通道的亮度值
输出参数：Intesity：通道的亮度值（0~255）
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）

### 2.2.4 设置通道触发延迟时间

BOOL SY_MVD_Light_Set_ TriggerDelay (int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int StrobeTime)
说明：设置触发延迟时间
输出参数：TRUE ：操作成功； FALSE ：失败

输入参数:
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
StrobeTime：频闪延迟时间值（1~4096），单位为 5 微秒，即最小触发延迟时间为 5us，最大触发延迟时间为 4096*5=20480 微秒。

| 类型 | 单位（us） | 数值限定 |
|---|---|---|
| 简易型控制器 | 80 | 1~65535(即 1*80us~65535us),默认参数为 480us |
| 全功能型控制器 | 5 | 1~4096(即 1*5us~4096*5us) ,默认参数为 500us |

注明：使用简易型控制器时，4 个通道共用通道 0 的触发延迟时间，所以仅需要设置 0 通道的触发延迟时间即可；采用全功能型控制器时，则需要每个通道都单独设置。

### 2.2.5 得到通道触发延迟时间

BOOL SY_MVD_Light_Get_TriggerDelay(int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int *StrobeTime)
说明：得到触发延迟时间
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
StrobeTime：频闪延迟时间值（1~4096），单位为 5 微秒，即最小触发延迟时间为 5us，最大触发延迟时间为 4096*5=20480 微秒。

| 类型 | 单位（us） | 数值限定 |
|---|---|---|
| 简易型控制器 | 80 | 1~65535(即 1*80us~65535us) ,默认参数为 480us |
| 全功能型控制器 | 5 | 1~4096(即 1*5us~4096*5us) ,默认参数为 500us |

注明：使用简易型控制器时，4 个通道共用通道 0 的触发延迟时间，所以仅需要设置 0 通道的触发延迟时间即可；采用全功能型控制器时，则需要每个通道都单独设置。

### 2.2.6 设置通道频闪保持时间

BOOL SY_MVD_Light_Set_ StrobeDuration (int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int HoldTime)
说明：设置频闪保持时间
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄

SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
HoldTime：频闪保持时间值（1~4096），单位为 5 微秒，即最小触发延迟时间为 5us，最大触发延迟时间为 4096*5=20480 微秒。

| 类型 | 单位（us） | 数值限定 |
|---|---|---|
| 简易型控制器 | 80 | 1~65535(即 1*80us~65535us),默认参数 **48000us** |
| 全功能型控制器 | 5 | 1~4096(即 1*5us~4096*5us),默认参数 **20000us** |

注明：使用简易型控制器时，4 个通道共用通道 0 的频闪保持时间，所以仅需要设置 0 通道的频闪保持时间即可；采用全功能型控制器时，则需要每个通道都单独设置。

### 2.2.7 得到通道频闪保持时间

BOOL SY_MVD_Light_Get_StrobeDuration(int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int *HoldTime)
说明：得到频闪保持时间
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
HoldTime：频闪保持时间值

| 类型 | 单位（us） | 数值限定 |
|---|---|---|
| 简易型控制器 | 80 | 1~65535(即 1*80us~65535us) ,默认参数 **48000us** |
| 全功能型控制器 | 5 | 1~4096(即 1*5us~4096*5us) ,默认参数 **20000us** |

注明：使用简易型控制器时，4 个通道共用通道 0 的频闪保持时间，所以仅需要设置 0 通道的频闪保持时间即可；采用全功能型控制器时，则需要每个通道都单独设置。

### 2.2.8 设置光源控制模式

BOOL SY_MVD_Light_SetCHMode(int BoardType,long long hSMDevice, int SlaveIP, int ChNum,int Mode)
说明：光源通道模式，Strobe 或者 PWM 模式等
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~4）
Mode：模式有 5 种模式

工作模式 0，将该通道设置为软件触发常亮模式。可由按键触发和通信触发。

工作模式 1，将改通道设置为软件触发闪光模式。可由按键触发和通信触发。

工作模式 2，将该通道设置为外部触发闪光模式，上升沿触发。只能由输入端口触发。

工作模式 3，将该通道设置为外部触发闪光模式，下降沿触发。只能由输入端口触发。

工作模式 4，将该通道设置为外部开关量触发模式，输入信号高电平有效，光源常亮；信号无效时候，光源关闭。

### 2.2.9 得到光源控制模式

BOOL SY_MVD_Light_GetCHMode(int BoardType,long long hSMDevice, int SlaveIP, int ChNum,int *Mode)
说明：得到光源通道模式，Strobe 或者 PWM 模式等
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~4）
Mode：模式有 5 种模式
工作模式 0，将该通道设置为软件触发常亮模式。可由按键触发和通信触发。
工作模式 1，将改通道设置为软件触发闪光模式。可由按键触发和通信触发。
工作模式 2，将该通道设置为外部触发闪光模式，上升沿触发。只能由输入端口触发。
工作模式 3，将该通道设置为外部触发闪光模式，下降沿触发。只能由输入端口触发。
工作模式 4，将该通道设置为外部开关量触发模式，输入信号高电平有效，光源常亮；信号无效时候，光源关闭。

### 2.2.10 设置光源开关状态

BOOL SY_MVD_Light_SetChStsOnOff(int BoardType,long long hSMDevice, int SlaveIP,int ChNum,int CHstatus)
说明：PWM 模式下打开或者关闭某个通道，或者频闪
输出参数：TRUE ：设置成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
CHstatus：2：频闪；1：打开； 0：关闭

### 2.2.11 载入配置文件

BOOL SY_MVD_LoadLightParam(int BoardType,long long hSMDevice, int SlaveIP,char *FilePath)
说明：从 ini 文件中读取当前光源控制&COM 参数
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
FilePath：ini 文件的路径

### 2.2.12 保存配置文件

BOOL SY_MVD_SaveLightParam(int BoardType,long long hSMDevice, int SlaveIP,char *FilePath)
说明：当前光源控制&COM 参数存入 ini 文件
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
FilePath：ini 文件的路径

### 2.2.13 软件切换通道

BOOL SY_MVD_Light_SelChn (int BoardType,long long hSMDevice, int SlaveIP, int ChNumSet)
说明：软件切换模块的通道
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNumSet：控制器的通道号（0~3）

### 2.2.14 设置通讯模式

BOOL SY_MVD_Light_Set_CommMod (int BoardType,long long hSMDevice, int SlaveIP, int CommMod)
说明：软件切换模块的通讯方式
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数:
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CommMod：光源控制器模块通讯模式

| 光源控制器模块通讯模式 | 描述 |
|---|---|
| 0 | 采用 RS-232 通讯模式 |
| 1 | 采用 RS-485 通讯模式 |

### 2.2.15 设置 4 通道亮度值

BOOL SY_MVD_Light_Set_AllIntensity(int BoardType,long long hSMDevice, int SlaveIP,int CH1Intesity,int CH2Intesity,int CH3Intesity,int CH4Intesity)
说明：同步设定 4 个通道的 PWM 值
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CH1Intesity：通道 1 的亮度值（0~255）
CH2Intesity：通道 2 的亮度值（0~255）
CH3Intesity：通道 3 的亮度值（0~255）
CH4Intesity：通道 4 的亮度值（0~255）

### 2.2.16 设置 4 通道开关状态

BOOL BOOL SY_MVD_Light_Set_AllChStsOnOff(int BoardType,long long hSMDevice, int SlaveIP,int CHstatus);
说明：同步设定打开或者关闭 4 个通道
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型，hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CHstatus：

| 光源控制器所有通道状态（CHstatus） | 描述 |
|---|---|
| 0 | 所有通道关闭 |
| 1 | 所有通道打开 |

### 2.2.17 得到 4 通道开关状态

BOOL SY_MVD_Light_Get_AllChStsOnOff(int BoardType,long long hSMDevice, int SlaveIP,int *pstatus);

说明：获得 4 个通道的开关状态

输出参数：TRUE ：操作成功； FALSE ：失败

输入参数：

BoardType：模块类型，hSMDevice：资源句柄

SlaveIP：控制器的 ID（1~32）

pstatus：通道开关状态

pstatus 各个 bit 定义如下所示

| 位 | 定义 | 描述 |
|---|---|---|
| 0 | 通道 1 状态 | 值为 1 时候表示通道 1 状态是打开的；值为 0 时候表示通道 1 状态是关闭的 |
| 1 | 通道 2 状态 | 值为 1 时候表示通道 2 状态是打开的；值为 0 时候表示通道 2 状态是关闭的 |
| 2 | 通道 3 状态 | 值为 1 时候表示通道 3 状态是打开的；值为 0 时候表示通道 3 状态是关闭的 |
| 3 | 通道 4 状态 | 值为 1 时候表示通道 4 状态是打开的；值为 0 时候表示通道 4 状态是关闭的 |

### 2.2.18 设置 4 通道亮度和通道状态

BOOL SY_MVD_Light_Set_Intensity_ChanelSts（ int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int Intesity）
说明：独立控制 4 个通道 PWM 值和 4 个通道开关命令
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CHSts:分别控制 4 个通道的开关状态。
如 CHSts =0X01，则表示通道 0 打开，其余通道关闭；如 CHSts =0X05,则表示通道 0 和通道 2 同时打开，其余通道关闭。
CH1Intesity:通道 1 的 PWM 值
CH2Intesity:通道 2 的 PWM 值
CH3Intesity:通道 3 的 PWM 值
CH4Intesity:通道 4 的 PWM 值

### 2.2.19 设置单通道电流输出值

BOOL SY_MVD_Light_Set_CurrentOut(int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int CurrentOut)
说明：设定光源控制器某个通道的电流输出值，恒流源光源控制器专用。
输出参数：TRUE ：设置成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）
CurrentOut：通道的电流输出值，根据恒流源的光源控制器实际输出来设定。

### 2.2.20 得到单通道电流输出值

BOOL SY_MVD_Light_Get_CurrentOut (int BoardType,long long hSMDevice, int SlaveIP,int ChNum, int *CurrentOut )
说明：查询光源控制器某个通道的电流输出值，恒流源光源控制器专用。
输出参数：CurrentOut：通道的电流输出值.
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
ChNum：控制器的通道号（0~3）

### 2.2.21 设置 4 通道电流输出值

BOOL SY_MVD_Light_Set_AllCurrentOut(int BoardType,long long hSMDevice, int SlaveIP,int CH1CurrentOut,int CH2CurrentOut,int CH3CurrentOut,int CH4CurrentOut);
说明：独立控制4个通道电流输出值和4个通道开关命令
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CH1CurrentOut:通道 1 的电流输出值
CH2CurrentOut:通道 2 的电流输出值
CH3CurrentOut:通道 3 的电流输出值
CH4CurrentOut:通道 4 的电流输出值

### 2.2.22 设置 4 通道电流输出值

BOOL SY_MVD_Light_Set_AllCurrentOut(int BoardType,long long hSMDevice, int SlaveIP,int CH1CurrentOut,int CH2CurrentOut,int CH3CurrentOut,int CH4CurrentOut);
说明：独立控制4个通道电流输出值和4个通道开关命令
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CHSts:分别控制 4 个通道的开关状态。
如 CHSts =0X01，则表示通道 0 打开，其余通道关闭；如 CHSts =0X05,则表示通道 0 和通道 2 同时打开，其余通道关闭。
CH1CurrentOut:通道 1 的电流输出值
CH2CurrentOut:通道 2 的电流输出值
CH3CurrentOut:通道 3 的电流输出值
CH4CurrentOut:通道 4 的电流输出值

### 2.2.23 设置 4 通道电流输出值和通道状态

BOOL SY_MVD_Light_Set_CurrentOut_ChanelSts (int BoardType,long long hSMDevice, int SlaveIP,int CHSts,int CH1CurrentOut,int CH2CurrentOut,int CH3CurrentOut,int CH4CurrentOut);
说明：独立控制 4 个通道 PWM 值和 4 个通道开关命令
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型

hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
CHSts:分别控制 4 个通道的开关状态。如 CHSts =0X01，则表示通道 0 打开，其余通道关闭；如 CHSts =0X05,则表示通道 0 和通道 2 同时打开，其余通道关闭。
CH1CurrentOut:通道 1 的电流输出值
CH2CurrentOut:通道 2 的电流输出值
CH3CurrentOut:通道 3 的电流输出值
CH4CurrentOut:通道 4 的电流输出值

### 2.2.24 获得每个通道触发信号状态

bool CALL_TYPE SY_MVD_Light_GetTrigInStatus(int BoardType,INT64 hSMDevice ,int SlaveIP, int *TrigInStatus);

说明：获取光源四个通道的触发信号状态
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~32）
TrigInStatus:分别控制 4 个通道的触发信号状态。TrigInStatus 的 bit[3:0] 分别表示 TrigIN3~0 Status，1:Enable；0:Disable。

# 3.串口 IO 模块相关命令

| Function Name(函数名称) | 说明 |
|---|---|
| SY_MV_ DI_ReadPort | 单个输入端口 DI 状态查询 |
| SY_MV_ DI_ReadLine _dW | 所有端口 DI 状态查询，支持 32 路 DI 状态查询 |
| SY_MV_DI_SetFilter | 设置输入端口滤波参数 |
| SY_MV_DI_GetFilter | 得到输入端口滤波参数 |
| SY_MV_ DO_WritePort | 设置单个输出端口 |
| SY_MV_ DO_ReadBackPort | 读取单个输出端口状态 |
| SY_MV_ DO_ WriteLine_dW | 支持同时设置 32 位 DO 状态 |
| SY_MV_DO_ReadBackLine_dW | 支持同时查询 32 路 DO 状态 |
| SY_MVD_DO_CFG_DOutMode | 设置 DO 输出模式状态 |
| SY_MVD_DO_Get_DOutMode | 读取 DO 输出模式状态 |
| SY_MVD_DI_CFG_DinMode | 设置 DI 输出模式状态 |
| SY_MVD_DI_Get_DinCount | 读取 DI 输出模式状态 |

### 3.1 单个输入端口 DI 状态查询

BOOL SY_MVD_DI_ReadPort(int BoardType,long long hSMDevice, int SlaveIP,int DIPortNum, unsigned short *PortStatus)
说明：单个输入端口 DI 状态查询
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
DIPortNum：控制器的通道号（0~15）
PortStatus：返回的通道状态。1 表示有效；0 表示无效；

### 3.2 所有端口 DI 状态查询，支持 32 路 DI 状态查询

BOOL SY_MVD_DI_ReadLine_dW(int BoardType,long long hSMDevice, int SlaveIP, unsigned long long *PortStatus) ;
说明：输出端口 DO 状态设置，支持同时设置 32 位 DO 状态
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
PortStatus：所有通道要设置的状态。
一个 bit 对应一个通道. 即 bit0 对应通道 0，bit1 对应通道 1…bit31 对应通道 31 。如果对应的位为 0，则表示要关闭该通道输出；对应的位为 1，则表示打开该通道输出。

### 3.3 设置输入端口滤波

BOOL SY_MVD_DI_SetFilter(int BoardType,long long hSMDevice, int SlaveIP, unsigned short Timer）
说明：设置输入端口滤波
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
Timer:滤波参数，数值范围为 1:~255ms，默认值为 10ms

### 3.4 得到输入端口滤波

BOOL SY_MVD_DI_GetFilter(int BoardType,long long hSMDevice, int SlaveIP, unsigned short *Timer)
说明：得到输入端口滤波
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
Timer:滤波参数，数值范围为 1:~255ms，默认值为 10ms

### 3.5 单个输出端口 DO 状态设置

BOOL SY_MVD_DO_WritePort(int BoardType,long long hSMDevice, int SlaveIP,int DOPortNum, unsigned short PortStatus)
说明：单个输出端口 DO 状态设置
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
DOPortNum：控制器的通道号（0~15）
PortStatus：输出通道状态。1 表示打开；0 表示关闭；

### 3.6 输出端口 DO 状态查询

BOOL SY_MVD_DO_ReadBackPort(int BoardType,long long hSMDevice, int SlaveIP,int DOPortNum, unsigned short *PortStatus)
说明：输出端口 DO 状态查询
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
DOPortNum：控制器的通道号（0~15）
PortStatus：返回的通道状态。1 表示有效；0 表示无效；

### 3.7 输出端口 DO 状态设置，支持同时设置 32 位 DO 状态

BOOL SY_MVD_DO_WriteLine_dW(int BoardType,long long hSMDevice, int SlaveIP, unsigned long long PortStatus)
说明：输出端口 DO 状态设置，支持同时设置 32 位 DO 状态
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
PortStatus：所有通道要设置的状态。
一个 bit 对应一个通道. 即 bit0 对应通道 0，bit1 对应通道 1…bit31 对应通道 31 。如果对应的位为 0，则表示要关闭该通道输出；对应的位为 1，则表示打开该通道输出。

### 3.8 输出端口 DO 状态查询，支持 32 路 DO 状态查询

BOOL SY_MVD_DO_ReadBackLine_dW(int BoardType,long long hSMDevice, int SlaveIP , unsigned long long *PortStatus)
说明：输出端口 DO 状态查询，支持 32 路 DO 状态查询
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
PortStatus：所有通道的设置状态。
一个 bit 对应一个通道. 即 bit0 对应通道 0，bit1 对应通道 1…bit31 对应通道 31 。如果对应的位为 0，则表示当前该通道输出关闭；对应的位为 1，则表示当前该通道输出打开。

### 3.9 设置 DO 输出模式状态

BOOL SY_MVD_DO_CFG_DOutMode(int BoardType, INT64 hSMDevice, int SlaveIP, U32 ChannelNumber, U32 DOutMode, U32 DoutPara1, U32 DoutPara2)
说明：设置当前通道输出端口模式
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
ChannelNumber：当前设置通道号
DOutMode：输出模式
0x0：正常模式。
0x1：输入触发输出
0x2：单脉冲输出

0x3：连续脉冲输出

0x4：上升沿输入触发延时脉冲输出模式

0x104:下降沿输入触发延时脉冲输出模式

DoutPara1：参数 1

模式 1：输出脉宽

模式 2：输出脉宽

模式 3：输出脉宽高电平

模式 4：延时输出时间。单位 1ms。设置范围 1-1000ms。

DoutPara2：参数 2

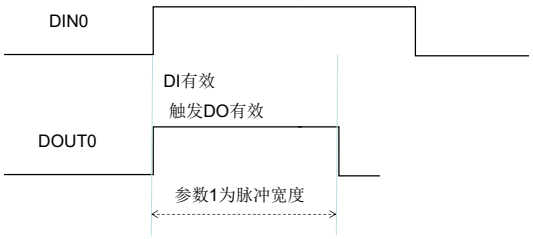
模式 1：不涉及

模式 2：不涉及

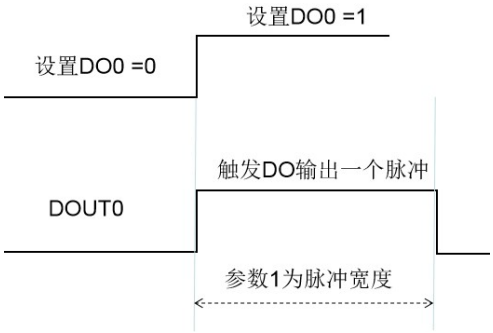
模式 3：输出脉宽低电平

模式 4：输出脉宽。单位 1ms。设置范围 1-1000ms。

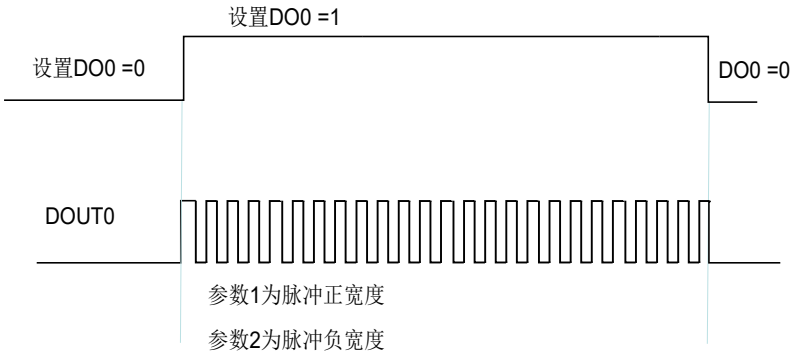
模式1：输入触发输出模式



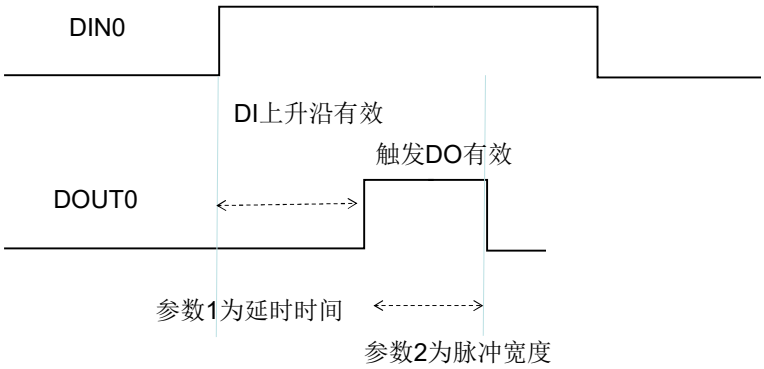

模式2：单脉冲输出模式

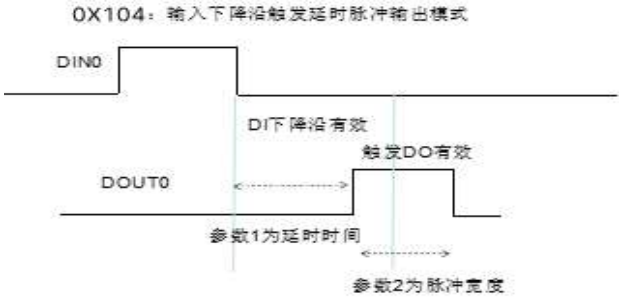



模式3：连续冲输出模式



模式4：输入上升沿触发延时脉冲输出模式




| 模式范围 | 通道号范围 | 参数 1 范围 | 参数 2 范围 | 模式功能说明 |
|---|---|---|---|---|
| 0 | 0~31 | 不涉及，填 0 | 不涉及，填 0 | 正常 DO 功能；SY_MC_Write_d_Channel_output 和 SY_MC_Write_d_output 正常操作 |
| 1 | 0~3 | 1~255 | 不涉及，填 0 | 输入触发输出模式。只有输出端口 0~3 支持。设置成功后。当对应的输入信号 DIN0~3（DIN0 对应 DOUT0、DIN1 对应 DOUT1...）有效时，在输出端口产生一个输出脉冲，脉冲宽度 1~255ms 可设 。输入触发是上升沿触发，即 DIN 从无效到有效时产生一次输出 |

| | | | | |
|---|---|---|---|---|
| 2 | 0~7 | 1~255 | 不涉及，填 0 | 单脉冲输出模式。只有端口 0~7 支持。<br>设置成功后。当对该 Dout_n 通道设置为 1 的时候，如果原来的通道是 0，则在该端口产生一个输出脉冲，脉冲宽度 1~255ms 可设 。输出指令是一次有效。再次产生输出指令，需要先将该口输出清 0，再置 1。<br>如端口 0 模式设置为 2 ；需要在 DO0 产生一个脉冲，则需要：<br>SY_MC_Write_d_Channel_output(Ch_No =0 ， DO_Data=0); //先关闭该端口<br>SY_MC_Write_d_Channel_output(Ch_No =0 ， DO_Data=1); //打开该端口，这时输出端口产生一个脉冲。<br>SY_MC_Write_d_output 对该位的操作，也是同样的效果 |
| 3 | 0~7 | 1~9000 | 1~9000 | 连续脉冲输出模式。只有端口 0~7 支持。<br>设置成功后。当对应的输出 Dout_n 通道设置为 1 的时候，该端口输出连续脉冲，脉冲宽度的高低电平 1~9000ms 可设 。<br>如端口 0 模式设置为 3 ；需要在 DO0 产生连续脉冲，则只需：<br>SY_MC_Write_d_Channel_output(Ch_No =0 ， DO_Data=1); //打开该端口，这时输出端口产生连续脉冲。<br>如果想关闭脉冲输出，则只需：<br>SY_MC_Write_d_Channel_output(Ch_No =0 ， DO_Data=0); //先关闭该端口<br>SY_MC_Write_d_output 对该位的操作，也是同样的效果 |
| 4 | 0~3 | 1~1000 | 1~1000 | 输入上升沿触发延时脉冲输出模式。只有输出端口 0~3 支持。<br>设置成功后。当对应的输入信号 DIN0~3（DIN0 对应 DOUT0、DIN1 对应 DOUT1...）有效时，先延时 DoutPara1 时间后，在输出端口产生一个输出脉冲，脉冲宽度为 DoutPara2 。输入触发是上升沿触发，即 DIN 从无效到有效时产生一次输出 |

| | | | | |
|---|---|---|---|---|
| 260<br>/0X104 | 0~3 | 1~1000 | 1~1000 | 输入下降沿触发延时脉冲输出模式。只有输出端口 0~3 支持。<br>设置成功后。当对应的输入信号 DIN0~3（DIN0 对应 DOUT0、DIN1 对应 DOUT1...）有效时，先延时 DoutPara1 时间后，在输出端口产生一个输出脉冲，脉冲宽度为 DoutPara2 。输入触发是下降沿触发，即 DIN 从有效到无效时产生一次输出 |

### 3.10 读取 DO 输出模式状态

BOOL SY_MVD_DO_Get_DOutMode(int BoardType, INT64 hSMDevice, int SlaveIP, U32 ChannelNumber, U32 *DOutMode, U32 *DoutPara1, U32 *DoutPara2)

说明：读取当前通道输出端口模式

输出参数：TRUE ：查询成功； FALSE ：失败

输入参数：

BoardType：模块类型

hSMDevice：资源句柄

SlaveIP：控制器的 ID（1~31）

ChannelNumber：当前设置通道号

DOutMode：输出模式

0x0：正常模式。

0x1：输入触发输出

0x2：单脉冲输出

0x3：连续脉冲输出

0x4：上升沿输入触发延时脉冲输出模式

0x104:下降沿输入触发延时脉冲输出模式

DoutPara1：参数 1

模式 1：输出脉宽

模式 2：输出脉宽

模式 3：输出脉宽高电平

模式 4：延时输出时间。单位 1ms。设置范围 1-1000ms。

DoutPara2：参数 2

模式 1：不涉及

模式 2：不涉及

模式 3：输出脉宽低电平

模式 4：输出脉宽。单位 1ms。设置范围 1-1000ms。

### 3.11 设置当前输入端口模式

BOOL SY_MVD_DI_CFG_DinMode(int BoardType, INT64 hSMDevice, int SlaveIP, U32 ChannelNumber, U32 DinMode)
说明：设置输入端口模式
输出参数：TRUE ：查询成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
ChannelNumber：当前设置通道号
DinMode：输入模式
0x0：正常模式。
0x1：上升沿计数
0x2：下降沿计数

### 3.12 读取输入端口模式和数据结果

BOOL SY_MVD_DI_Get_DinCount(int BoardType, INT64 hSMDevice, int SlaveIP, U32 ChannelNumber, U32 *DinMode, U32 *DinCount);
说明：设置 DI 中断的模式
输出参数：TRUE ：操作成功； FALSE ：失败
输入参数：
BoardType：模块类型
hSMDevice：资源句柄
SlaveIP：控制器的 ID（1~31）
ChannelNumber：当前设置通道号
DinMode：输入模式
DinCount：输入脉冲数量

# 修订记录

| Rev | Data | Author | Description |
|---|---|---|---|
| 1.0 | 20150121 | Shuangyi | 初稿 |
| 1.1 | 20150206 | Shuangyi | 增加 SY_MVD_Light_Set_Current 函数 |
| 1.2 | 20150315 | Shuangyi | 增加 SY_MVD_Light_SetCHMode 函数<br>增加 SY_MVD_Light_Set_TriggerDelay<br>增加 SY_MVD_Light_Set_StrobeDuration |
| 1.2 | 20150530 | Shuangyi | 增加 SY_MVD_Light_Get_AllSlaveIP<br>查询连接的模块 ID |
| 1.3 | 20150601 | Shuangyi | 增加 SY_MVD_Light_SelChn，软件切换模式 |
| 1.4 | 20150701 | Shuangyi | 增加光源频闪电流峰值函数<br>SY_MVD_Light_Set_PeakCurrent<br>SY_MVD_Light_Get_PeakCurrent |
| 1.5 | 20150721 | Shuangyi | 函数更名为：SY_MVD_Light_xxxxxxx |
| 1.5 | 20150722 | Shuangyi | 修复 SY_MVD_Light_SetCHMode 设置模式为 3 的异常状况 |
| 1.6 | 20160622 | Shuangyi | 增加设置和查询通讯模式函数<br>SY_MVD_Light_Set_CommMod<br>SY_MVD_Light_Get_CommMod |
| 1.7 | 20160623 | Shuangyi | 增加设置和查询设置输出电压函数<br>SY_MVD_Light_Set_OutVolt<br>SY_MVD_Light_Get_OutVolt |
| 1.8 | 20170811 | Shuangyi | 增加设置开关量触发模式(模式 4)<br>SY_MVD_Light_SetCHMode |
| 1.9 | 20171207 | Shuangyi | *增加一条指令同时设置 4 个同道的 PWM 值<br>SY_MVD_Light_Set_AllIntensity<br>*增加一条指令同时控制 4 个通道的开关状态<br>SY_MVD_Light_Set_AllChStsOnOff<br>*增加一条读取 4 个通道开关状态的命令<br>SY_MVD_Light_Get_AllChStsOnOff<br>*修改波特率设置命中波特率的值<br>SY_MVD_Light_Set_Baudrate<br>*删除不使用的命令 |
| 2.0 | 20171216 | Shuangyi | 增加编程流程图 |
| 3.0 | 20180317 | Shuangyi | 将触发延迟和频闪保持时间设定值先定为 1ms~20ms |
| 3.1 | 20180427 | Shuangyi | 独立控制 4 个通道 PWM 值和 4 个通道开关命令 |
| 3.2 | 20180730 | Shuangyi | 增加电流输出函数<br>SY_MVD_Light_Set_CurrentOut<br>SY_MVD_Light_Get_CurrentOut |
| 3.3 | 20180801 | Shuangyi | 增加一条同时设置 4 个通道恒流电流值的命令。<br>增加了同时设置 4 通道电流和开关的命令 |

| | | | |
|---|---|---|---|
| 4.2 | 20200618 | Shuangyi | 光源控制器和串口 IO 控制器整合成统一函数库 |
| 4.3 | 20201021 | Shuangyi | 对简易光源控制器的触发延迟时间和频闪保持时间函数使用说明 |
| 4.4 | 20210411 | Shuangyi | 新增建议光源控制器的触发延迟时间和频闪保持时间默认参数说明 |
| 4.5 | 20220212 | Shuangyi | 删除老版本 IO 模块增加输入输出 4 模式接口<br>增加光源控制器四个通道触发信号状态 |