

# 驱动软件对外接口

## 1 DIO 卡

### 1.1 参数类型枚举

```
typedef enum _EnableStatus
{
    UNABLE_STATUS = 0, 不使能
    ENABLE_STATUS = 1 使能
} EnableStatus;
```

说明：使能枚举

```
typedef enum _OutputMode{
    NULL_OUTPUT_MODE = 0x00, 无模式
    LEVEL_OUTPUT_MODE = 0x01, 电平模式
    PWM_OUTPUT_MODE = 0x02, PWM 模式
    BIT_OUTPUT_MODE = 0x03 BIT 数据流模式
} OutputMode;
```

说明：DIO 输出模式枚举

```
typedef enum _OutputImpedanceMode
{
    HIGH_IMPEDANCE_MODE = 0, 高阻模式
    PULL_MODE = 0x01, 拉模式
    PUSH_MODE = 0x02, 推模式
    PUSH_AND_PULL_MODE = 0x03 推拉模式
} OutputImpedanceMode;
```

说明：输出阻抗模式

```
typedef enum _VoltageLevel
{
    LOW_LEVEL = 0x00, 低电平
    HIGH_LEVEL = 0x01, 高电平
    INVALID_LEVEL
} VoltageLevel;
```

说明：电平枚举

```
typedef enum _OutputReferenceVoltage
{

```

```

    OUTPUT_5V_REF = 0X01, 输出 5v
    OUTPUT_12V_REF = 0X02, 输出 12v
    OUTPUT_OUTSIDE_REF = 0X03 输出外部参考
} OutputReferenceVoltage;
说明：输出参考电压

```

```

typedef enum _ReferenceClock
{
    REFCLK_100_MHZ = 0, 100M 参考时钟
    REFCLK_20_MHZ = 1 20M 参考时钟
} ReferenceClock;
说明：参考时钟

```

## 1.2 参数数据结构体

```

typedef struct _DOLevelConfigure
{
    OutputImpedanceMode eMode; 输出阻抗
    OutputReferenceVoltage eRef; 输出参考电压
} DOLevelConfigure;
说明：输出电平配置

```

```

typedef struct _DOPWMConfigure
{
    double dFreq; 频率 Hz
    double dDuty; 占空比
    OutputImpedanceMode eMode; 输出阻抗
    OutputReferenceVoltage eRef; 输出参考电压
} DOPWMConfigure;
说明：输出 PWM 配置

```

```

typedef struct _DIPWMConfigure
{
    double dDurationTime; 采样时间
    ReferenceClock eRefClk; 参考时钟（保留，目前为自适应频率）
} DIPWMConfigure;
说明：输入 PWM 配置

```

```

typedef struct _PWMProperty
{
    double dFreq; 频率 Hz
    double dDuty; 占空比
} PWMProperty;
说明：PWM 属性

```

```

typedef struct _DOBitConfigure

```

```
{
    OutputImpedanceMode eMode; 输出阻抗
    long lRate;          速率
    OutputReferenceVoltage eRef; 输出参考电压
} DOBitConfigure;
```

说明：输出 BIT 流配置

### 1.3 函数接口

```
int OpenDio (const char* strDev, void **handle);
```

说明： 打开 DIO 对应设备文件，获取对应设备文件指针

参数： strDev DIO 对应设备文件，如 “/dev/dio\_3\_in”，表示打卡 3 槽 DIO 输入部分

handle 对应设备文件指针

返回值： 正常为 0

备注：

```
int GetDioVersion (void *handle,
                  char* pStr,
                  int iLength,
                  int* pActLength);
```

说明： 获取当前 DIO 卡逻辑固件版本信息

参数： handle 资源设备对应指针

pStr 日期版本字符串首地址，用户申请字符数组之数组地址

iLength 用户申请的字节数组长度

pActLength 获取的版本信息实际有效字节数组长度

返回值： 正常为 0

备注：

```
int CloseDio (void* handle);
```

说明： 关闭 DIO 对应设备文件

参数： handle 资源设备对应指针

返回值： 正常为 0

备注：

```
int SetDoChannelEnable (void *handle,
                       int iChannel,
                       EnableStatus eStatus);
```

说明： 设置 DIO 卡指定通道输出使能

参数： handle 资源设备对应指针

iChannel 输出通道号, 范围 0~23

eStatus 使能状态, ENABLE\_STATUS 使能输出

返回值： 正常为 0

备注：

```
int SetDoAllChannelEnable (void *handle, unsigned int iStatus);
```

说明： 设置 DIO 卡全通道输出使能

参数： handle 资源设备对应指针

iStatus 全通道使能状态，通道按位表示，0bit 位为 0 通道，1 为使能，0 为未使能，如参数 0x03 表示输出通道 0 与 1 使能输出，其余 30 个输出通道均未使能输出

返回值： 正常为 0

备注：

```
int GetDoAllChannelEnable (void *handle, unsigned int* pStatus);
```

说明： 获取全通道使能状态

参数： handle 资源设备对应指针

pStatus 全通道使能状态，通道按位表示，0bit 位为 0 通道，1 为使能，0 为未使能，如获取 0x03 表示输出通道 0 与 1 为使能输出状态，其余 30 个输出通道均未使能输出

返回值： 正常为 0

备注：

```
int SetDoMode (void *handle, int iChannel, OutputMode eMode);
```

说明： 设置 DIO 卡指定输出通道模式

参数： handle 资源设备对应指针

iChannel 输出通道号，范围 0~23

eMode 输出模式，主要有电平模式，PWM 模式以及 BIT 数据流模式

返回值： 正常为 0

备注：

```
int GetDoAllMode (void *handle,  
                  unsigned int* pHiMode,  
                  unsigned int* pLoMode);
```

说明： 获取所有输出通道模式

参数： handle 资源设备对应指针

pHiMode 通道 16~23 输出模式指针，如获取 0x01，表示输出通道 16 模式为电平模式

pLoMode 通道 0~15 输出模式指针 如获取 0x02, 表示输出通道 0 模式为 PWM 模式

返回值： 正常为 0

备注：

```
int SetDiReferenceVoltage (void *handle,  
                           int iChannel,  
                           int iVoltage);;
```

说明： 设置输入通道比较参考电压

参数： handle 资源设备对应指针

iChannel 通道号，范围 0~31，0~7 为一组比较电压参考值，8~15 为一组比较电压参考值，16~23 为一组比较电压参考值，24~31 为一组比较电压参考值，如输入通道参数 1 与 7，其实下发的是同一组比较电压值

iVoltage 参考电压，单位毫伏，数值在 0~25V 之间

返回值：正常为 0

备注：

```
int GetDiLevel (void *handle, int iChannel, int* iVoltage);
```

说明： 获取指定输入通道电平值

参数： handle 资源设备对应指针

iChannel 通道号

iVoltage 输入电平，0 为低，1 为高

返回值：正常为 0

备注：

```
int GetDiAllLevel (void *handle, int* iLevel);
```

说明： 获取所有输入通道输入电平

参数： handle 资源设备对应指针

iLevel 所有通道的输入电平，通道按位表示，0bit 位为 0 通道，1 为高电平，0 为低电平，如获取 0x01 表示 0 通道为高电平输入，其余 31 个通道为低电平输入

返回值：正常为 0

备注：

```
int SetDoLevelConfigure (void *handle,
                          int iChannel,
                          DOLevelConfigure stDOLevelCfg);
```

说明： 设置指定输出通道电平配置

参数： handle 资源设备对应指针

iChannel 输出通道号，0~23

stDOLevelCfg 输出电平配置结构

返回值：正常为 0

备注：

```
int SetDoLevel (void *handle, int iChannel, VoltageLevel eLevel);
```

说明： 设置指定输出通道电平值

参数： handle 资源设备对应指针

iChannel 通道号，范围 0~23

eLevel 输出电平

返回值：正常为 0

备注：

```
int SetDiPWMCaptureEnable (void *handle,
                            int iChannel,
```

`EnableStatus eStatus);`

说明： 设置指定输入通道 PWM 捕获使能

参数： `handle` 资源设备对应指针  
`iChannel` 通道号，范围 0~31  
`eStatus` 使能标志

返回值： 正常为 0

备注：

`int SetDiPWMAllCaptureEnable (void *handle, unsigned int iStatus);`

说明： 设置所有输入通道 PWM 捕获使能

参数： `handle` 资源设备对应指针  
`iStatus` 所有输入通道 PWM 采集使能，0bit 位为 0 通道，1 为使能，0 为不使能，如 0x03 表示输入通道 0、1 PWM 捕获使能，其余 30 个通道捕获未使能

返回值： 正常为 0

备注：

`int GetDiPWMAllCaptureEnable (void *handle, unsigned int* pStatus);`

说明： 获取所有输入通道 PWM 捕获使能状态

参数： `handle` 资源设备对应指针  
`pStatus` 所有输入通道 PWM 采集使能状态指针，0Bit 位 0 通道，1 位使能，0 为不使能，如 0x01 表示输入通道 0 为捕获使能状态，其余输入通道未捕获使能

返回值： 正常为 0

备注：

`int SetDoPWMConfigure (void *handle,  
int iChannel,  
DOPWMConfigure stDOPWMCfg);`

说明： 设置指定输出通道 PWM 波形配置

参数： `handle` 资源设备对应指针  
`iChannel` 输出通道号，范围 0~23  
`stDOPWMCfg` 输出 pwm 配置数据结构

返回值： 正常为 0

备注：

`int SetDiPWMConfigure (void *handle,  
int iChannel,  
DIPWMConfigure stDIPWMCfg);`

说明： 设置指定输入通道 PWM 波形采集设置

参数： `handle` 资源设备对应指针  
`iChannel` 通道号，范围 0~31  
`stDIPWMCfg` 输入 pwm 配置数据结构

返回值： 正常为 0

备注：

```
int GetDiPWMCapture (void *handle, PWMProperty arrPWMProper[32]);
```

说明： 获取所有输入通道 PWM 波形采集数据

参数： handle 资源设备对应指针  
arrPWMProper 波形数据结构体

返回值： 正常为 0

备注：

```
int ClearDoOverProtectionStatus (void *handle);
```

说明： 清除板卡过流保护状态标志，过流指示灯恢复为绿色正常

参数： handle 资源设备对应指针

返回值： 正常为 0

备注：

```
int SetDoBitConfigure (void *handle,  
                       int iChannel,  
                       DOBitConfigure stDOBitCfg);
```

说明： 设置指定输出通道 BIT 数据流配置

参数： handle 资源设备对应指针  
iChannel 通道号，范围 0~23  
stDOBitCfg 数据流配置结构

返回值： 正常为 0

备注：

```
int WriteDoBITData (void *handle, uint32_t* pArr, int iLength);
```

说明： 指定 DIO 卡发送 BIT 数据流

参数： handle 资源设备对应指针  
pArr 数据流缓冲区首地址  
iLength 数据流缓冲区长度，长度须为 2 的整数倍，

返回值： 正常为 0

备注：数据流为 uint32\_t 类型数组，每 uint32\_t 数据，表示在指定通道配置为 bit 流模式下，24 路输出通道同时输出 1bit 数据。数据从低 uint32\_t 元素至高 uint32\_t 依次输出，如已配置好 0、1 通道输出模式为 bit 流模式，下发数据 0x03, 0x01, 0x02, 0x01，表示通道 0 输出 BIT 流信息 1101，通道 1 输出 BIT 流信息 1010，如下图所示：

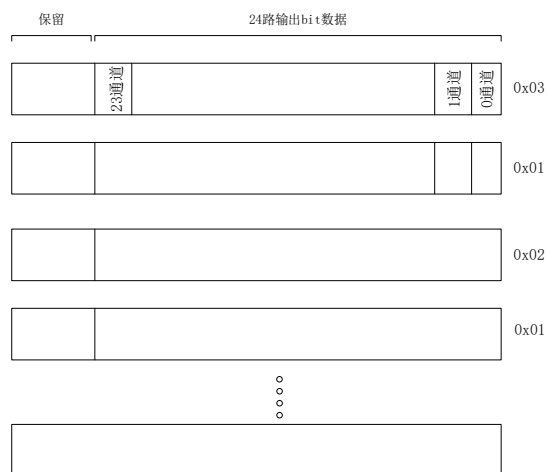


图 1Bit 流

```
int SetDioPpsUpdateCount(void* handle, uint32_t sec);
```

说明： 设置 PPS 秒记数配置

参数： handle 资源设备对应指针  
sec pps 秒计数值

返回值： 正常为 0

备注：

```
int GetDioBitStreamSendTime (void* handle, uint64_t *mic_sec);
```

说明： 获取 BIT 流发送时间戳

参数： handle 资源设备对应指针  
mic\_sec 时间戳微秒计数值

返回值： 正常为 0

备注：

```
int GetDioPpsSecCount (void* handle, uint32_t *sec);
```

说明： 获取当前 pps 秒计数值

参数： handle 资源设备对应指针  
sec pps 秒计数值

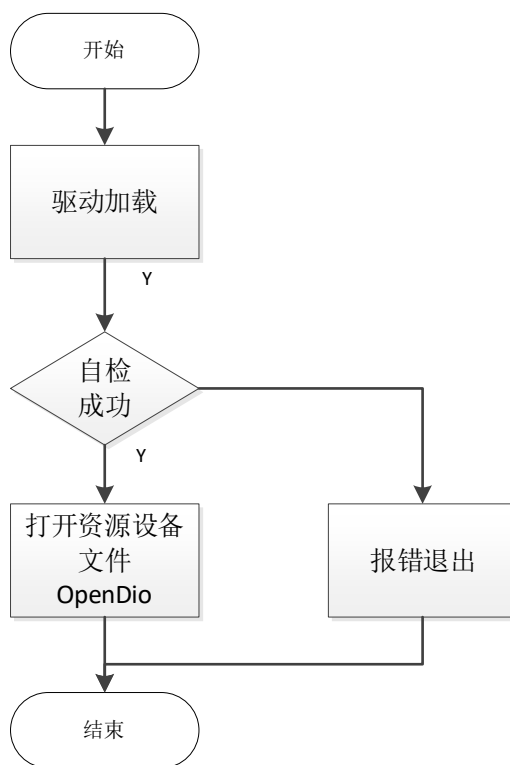
返回值： 正常为 0

备注：

## 1.4 工作流程操作

### 1.4.1 设备初始化与反初始化





1  
图 2 设备初始化流程图

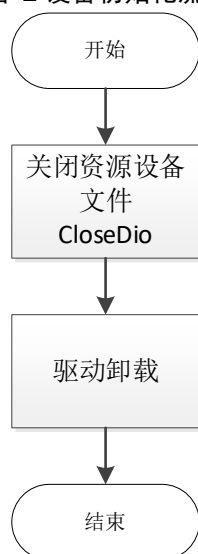


图 3 设备反初始化流程图

#### 1.4.2 DIO 电平输出



图 4 电平输出流程图

#### 1.4.3 DIO 电平输入

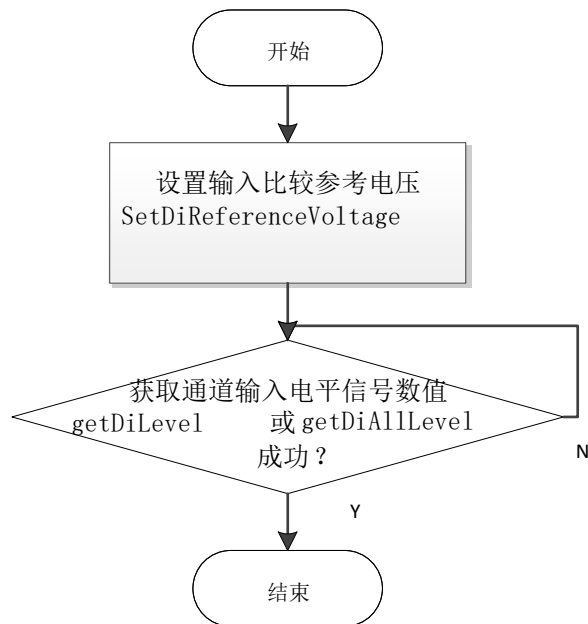


图 5 电平输入流程图

#### 1.4.4 DIO PWM 波形输入

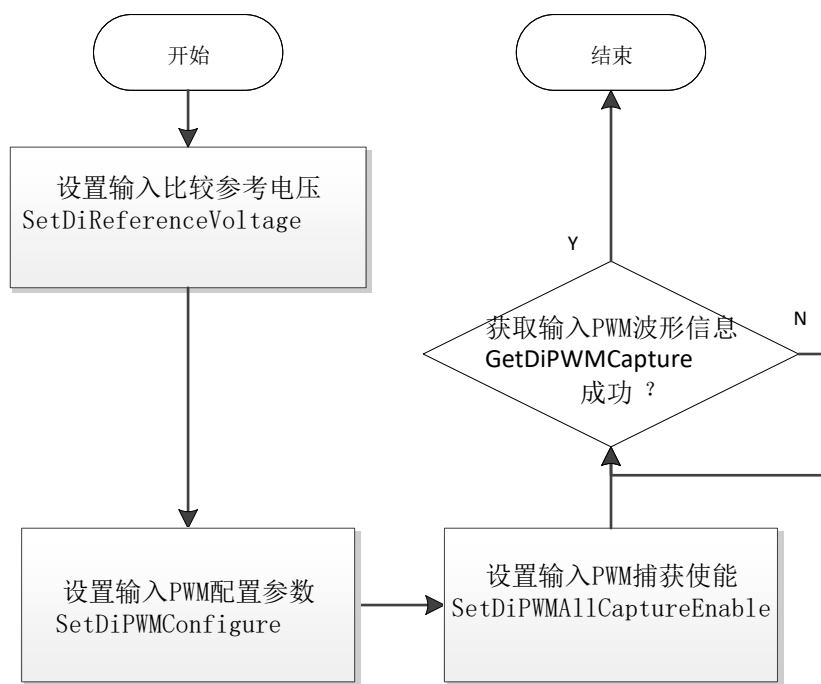


图 6PWM 输入流程图

#### 1.4.5 DIO PWM 波形输出



图 7PWM 输出流程图

#### 1.4.6 DIO Bit 流信息输出

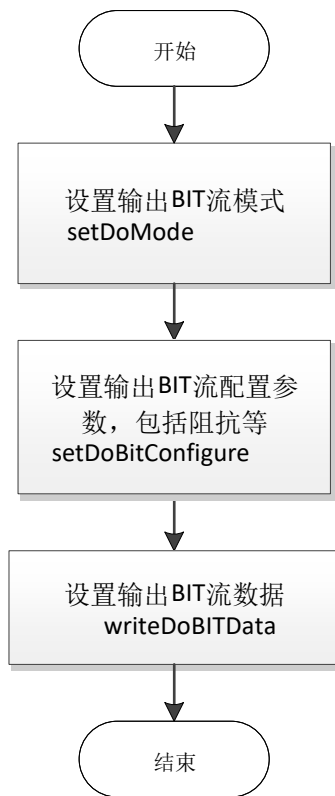


图 8BIT 流输出流程图