

SDK - HTHardDll.dll 说明文档

中文版(VC++ 6.0)

阅读须知:

本 DLL 在 VC++ 6.0 环境下编译生成。所以数据类型符合 VC++ 6.0 标准.

WORD : **unsigned short**, 无符号 16bit 整型。

BOOL : **bool**, 布尔类型。

ULONG: **unsigned long**, 无符号 32bit 整型。

此 DLL 中的所有文件都是用命令行上定义的 **DLL_API** 符号编译的。在使用此 DLL 的任何其他项目上都不应定义此符号。这样，源文件中包含此文件的任何其他项目都会将 **DLL_API** 函数视为是从 DLL 导入的。

```
#ifndef DLL_API
#define DLL_API extern "C" __declspec(dllimport)
#endif
```

定义标准调用:

```
#define WIN_API __stdcall
```

结构体介绍

结构体 **_HT_RELAY_CONTROL** 包含了所有控制继电器状态所需要的信息。

```
typedef struct _HT_RELAY_CONTROL
{
    BOOL bCHEnable[4];
    WORD nCHVoltDIV[4];
    WORD nCHCoupling[4];
    BOOL bCHBWLimit[4];
    WORD nTrigSource;
    BOOL bTrigFilt;
    WORD nALT;
}RELAYCONTROL,*PRELAYCONTROL;
```

说明:

bCHEnable[4]: 大小为 4(CH 的总数)的数组, 表示 CH 的开/关。取值: 1 为开; 0 为关。

nCHVoltDIV[4]: 大小为 4(CH 的总数)的数组, 表示 CH 的电压档位。电压档位以索引值形式表示。以最小电压档位为 0 开始依次递加 1 计算。

nCHCoupling[4]: 大小为 4(CH 的总数)的数组, 表示 CH 的耦合。耦合以索引值形式表示。取值: DC 为 0; AC 为 1; GND 为 2;

bCHBWLimit[4]: 大小为 4(CH 的总数)的数组, 表示 CH 的带宽限制。取值: 1 为打开带宽限制; 0 为关闭带宽限制。

nTrigSource: 表示触发源, 以索引值形式取值。假设现在为 4CH 示波器, 则内部触发取值为: CH1 为 0; CH2 为 1; CH3 为 2; CH4 为 3; 如果有外部触发, 则 EXT

为 5；如果有 EXT/10 触发，则取值为 6。

bTrigFilt：表示高频抑制。取值：1 表示打开高频抑制，0 表示关闭高频抑制。

nALT：表示是否交替。取值：1 为交替，0 为非交替。

举例：

声明一个变量：`RELAYCONTROL myRelayControl;`

声明一个指针：`PRELAYCONTROL pRelayControl;`

结构体 `_HT_CONTROL_DATA` 包含了某些函数需要的一些控制信息。

`typedef struct _HT_CONTROL_DATA`

```
{
    WORD nCHSet; //CH 开关---//第 0 位：表示 CH1 开或者关. 0:关, 1 开
                                   //第 1 位：表示 CH2 开或者关. 0:关, 1 开
                                   //第 2 位：表示 CH3 开或者关. 0:关, 1 开
                                   //第 3 位：表示 CH4 开或者关. 0:关, 1 开

    WORD nTimeDIV; //时基
    WORD nTriggerSource; //触发源
    WORD nHTriggerPos; //水平触发位置
    WORD nVTriggerPos; //垂直触发位置
    WORD nTriggerSlope; //边沿触发触发沿
    ULONG nBufferLen; //内存长度
    ULONG nReadDataLen; //读取数据长度
    WORD nALT; //是否交替

}CONTROLDATA,*PCONTROLDATA;
```

举例：

声明一个变量：`CONTROLDATA myControlData;`

声明一个指针：`PCONTROLDATA pControlData;`

函数介绍

1. 函数声明：`DLL_API WORD WINAPI dsoHTSearchDevice(short* pDevInfo)`

返回值：

返回连接的设备总数。1 台 PC 支持最大连接 32 台设备。

参数：

`pDevInfo`

`short` 型数组指针，用于存储有无设备信息。数组大小为 32。

备注：

获取 PC 已经连接上的设备总数，总数不超过 32。

程序举例：

`short DevInfo[32];`

`WORD nConnectedDevNum = 0;`

```
//DevInfo 初始化
//...
//调用函数
nConnectedDevNum = dsoHTSearchDevice(DevInfo);
如果 DevInfo[n] = 0 有设备，DevInfo[n] = -1 无设备.
```

2. 函数声明: DLL_API WORD WINAPI dsoHTDeviceConnect(WORD nDeviceIndex)

返回值: 1 表示设备已经连接; 0 表示无设备

参数:

nDeviceIndex

WORD 型变量, 表示当前设备的索引值。

备注:

判断索引值为 nDeviceIndex 的设备是否连接.

程序举例:

假设目前 PC 上连接两台设备, 需要判断第 2 台设备是否连接。

```
WORD nDeviceIndex = 1; //0 为第 1 台设备
```

```
WORD nRe = 0;
```

```
//调用函数
```

```
nRe = dsoHTDeviceConnect(nDeviceIndex);
```

```
if (nRe == 1)
```

```
    ;//设备连接
```

```
else
```

```
    ;//无设备连接
```

3. 函数声明: DLL_API WORD WINAPI dsoHTSetCHPos(

WORD nDeviceIndex,

WORD* pLevel,

WORD nVoltDIV,

WORD nPos,

WORD nCH

)

返回值: 失败返回 0, 成功返回非 0。

参数:

nDeviceIndex

WORD 型变量, 表示当前设备的索引值。

pLevel

WORD 型变量指针, 指向校对电平。

nVoltDIV

WORD 型变量, 表示电压档位索引值。

nPos

WORD 型变量，表示 CH 零电平（参考电平）位置，取值范围 0 ~ 255（8bit 精度）。

nCH

WORD 型变量，表示当前设置的 CH。CH1 为 0，CH2 为 1，CH3 为 2，CH4 为 3。

备注：

当零电平（参考电平）发生变化时，调用此函数进行设置。

程序举例：

假设要将 CH1 的零电平设置为 128。

```
WORD nDeviceIndex = 0;
```

```
WORD CHLevel[256]; //大小最小为 256，通过 dsoHTReadCalibrationData 附值。
```

```
WORD nVoltDIV = 6; //假设 1V/div 的索引值为 6。
```

```
WORD nPos = 128; //零电平 128。
```

```
WORD nCH = 0; //CH1。
```

```
//调用函数
```

```
if ( 0 == dsoHTSetCHPos(nDeviceIndex,CHLevel,nVoltDIV,nPos,nCH) )
```

```
    ;//失败
```

```
else
```

```
    ;//成功
```

4. 函数声明：DLL_API WORD WINAPI dsoHTSetVTriggerLevel(

WORD nDeviceIndex,

WORD* pLevel,

WORD nPos)

返回值：失败返回 0，成功返回非 0。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

pLevel

WORD 型变量指针，指向校对电平。

nPos

WORD 型变量，表示垂直触发电平位置，取值范围 0 ~ 255（8bit 精度）。

备注：

当需要设置触发电平时，调用此函数设置。

程序举例：

```
WORD nDeviceIndex = 0;
```

```
WORD CHLevel[256]; //大小最小为 256，通过 dsoHTReadCalibrationData 附值。
```

```
WORD nPos = 128; //触发电平为 128
```

```
//调用函数
```

```
if ( 0 == dsoHTSetVTriggerLevel( nDeviceIndex, CHLevel, nPos) )
```

```
    ;//失败
```

```
else
```

```
    ;//成功
```

5. 函数声明: DLL_API WORD WINAPI dsoHTSetHTriggerLength(
WORD nDeviceIndex,
ULONG nBufferLen,
WORD HTriggerPos,
WORD nTimeDIV,
WORD nYTFormat
)

返回值: 失败返回 0, 成功返回非 0。

参数:

nDeviceIndex

WORD 型变量, 表示当前设备的索引值。

nBufferLen

ULONG 型变量, 表示内存长度。

HTriggerPos

WORD 型变量, 表示水平触发位置。取值返回 0 ~ 100 。

nTimeDIV

WORD 型变量, 表示时基的索引值, 以时间最短时基为 0 依次递增 1 计算。

nYTFormat

WORD 型变量, 3 种采样格式。0: Normal, 1: Scan, 2: Roll

备注:

设置触发和预触发长度。

程序举例:

假设当前设备内存为 10K, 水平触发位置在 50%, 时基为 40us (索引值为 12)。

```
WORD nDeviceIndex = 0;  
ULONG nBufferLen = 10240;//10K  
WORD HTriggerPos = 50; //50%  
WORD nTimeDIV = 12; //40us  
//调用函数  
if ( 0 == dsoHTSetHTriggerLength(nDeviceIndex,nBufferLen,HTriggerPos,  
nTimeDIV) )  
    ;//失败  
else  
    ;//成功
```

6. 函数声明: DLL_API WORD WINAPI dsoHTSetCHAndTrigger(
WORD nDeviceIndex,
RELAYCONTROL RelayControl)

返回值: 失败返回 0, 成功返回非 0。

参数:

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

RelayControl

结构体，具体见结构体说明部分。

备注：

设置 CH 和 Trigger

程序举例：

```
WORD nDeviceIndex = 0;
RELAYCONTROL RelayControl;
//RelayControl 各变量附值
//调用函数
if ( 0 == dsoHTSetCHAndTrigger(nDeviceIndex,RelayControl) )
    ;//失败
else
    ;//成功
```

7. 函数声明：DLL_API WORD WINAPI dsoHTSetTriggerAndSyncOutput(
WORD nDeviceIndex,
WORD nTriggerMode,
WORD nTriggerSlope,
WORD nPWCondition,
ULONG nPW,
USHORT nVideoStandard,
USHORT nVedioSyncSelect,
USHORT nVideoHsyncNumOption,
WORD nSync
)

返回值：失败返回 0，成功返回非 0。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

nTriggerMode

WORD 型变量，表示触发类型。取值：边沿触发为 0，脉冲触发为 1。

nTriggerSlope

WORD 型变量，表示上升沿/下降沿。取值：上升沿为 0，下降沿为 1。

nPWCondition

WORD 型变量，表示脉冲触发条件。（脉冲触发下有效）

取值：

+ Less 为 0：当正脉宽 小于 nPW 时进行触发。

+ Equal 为 1：当正脉宽 等于 nPW 时进行触发。

+ More 为 2：当正脉宽 大于 nPW 时进行触发。

- Less 为 3：当负脉宽 小于 nPW 时进行触发。

- Equal 为 4 : 当负脉宽 等于 nPW 时进行触发。
- More 为 5 : 当负脉宽 大于 nPW 时进行触发。

nPW

ULONG 型变量, 表示脉冲宽度设置。脉冲宽度时间范围为 10ns ~ 10s。nPW 的取值范围为 2 ~ 2000000000(10ns/5 ~ 10000000000ns/5)。(脉冲触发下有效)

nVideoStandard

USHORT 变量, 表示视频标准设置。取值 0 为 PAL/SECAM, 1 为 NTSC

nVedioSyncSelect

USHORT 变量, 表示同步选项设置。取值:

- 所有行: 0
- 所有场: 1
- 奇数场: 2
- 偶数场: 3
- 指定行: 4

nVideoHsyncNumOption

USHORT 变量, 表示指定行数设置。当同步参数 (nVedioSyncSelect) 设置为指定行时有效。最小值为 1。

nSync

WORD 型变量, 表示同步输出。取值: 0 关闭同步输出, 1 打开同步输出。

备注:

设置 Trigger 和同步输出。如果是边沿触发, 参数 nPWCondition 和 nPW 无效。

程序举例:

```
WORD nDeviceIndex = 0;
WORD nTriggerMode = 0; //当前是边沿触发。
WORD nTriggerSlope = 0; //当前是上升沿触发。
WORD nPWCondition = 0; //
ULONG nPW = 2; //10ns, nPW = 10ns/5。
WORD nSync = 0; //关闭同步输出。
//调用函数
if (0 == dsoHTSetTriggerAndSyncOutput(nDeviceIndex, nTriggerMode,
                                       nTriggerSlope, nTriggerCondition, nPW, nSync) )
    ; //失败
else
    ; //成功
```

8. 函数声明: DLL_API WORD WINAPI dsoHTSetSampleRate(
 WORD nDeviceIndex,
 WORD nTimeDIV
)

返回值: 失败返回 0, 成功返回非 0。

参数:

nDeviceIndex

WORD 型变量, 表示当前设备的索引值。

nTimeDIV

WORD 型变量，表示时基的索引值，以时间最短时基为 0 依次递加 1 计算，

nYTFormat

WORD 型变量，3 种采样格式。0: Normal, 1: Scan, 2: Roll

备注：

设置时基（采样率）。

程序举例：

如果我们想要把时基设置到 40us.

```
WORD nDeviceIndex = 0;
```

```
WORD nTimeDIV = 12; //假设 40us 的索引值为 12。
```

```
//调用程序
```

```
if (0 == dsoHTSetSampleRate(nDeviceIndex, nTimeDIV) )
```

```
    ;//失败
```

```
else
```

```
    ;//成功
```

9. 函数声明：DLL_API WORD WINAPI dsoHTStartCollectData(WORD nDeviceIndex)

返回值：失败返回 0，成功返回非 0。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注：

通知设备准备采集数据。当各种设置都设备完毕，进入采集数据状态时，第 1 步就是调用此函数通知设备，准备采集数据。这里要注意，调用此函数之后，只有在设备采集完本次数据后，且进入新一轮数据采集时，才能再次调用，也就是采集一次数据只能调用一次本函数。

程序举例：

```
WORD nDeviceIndex = 0;
```

```
if (0 == dsoHTStartCollectData(nDeviceIndex) )
```

```
    ;//失败
```

```
else
```

```
    ;//成功
```

10. 函数声明：DLL_API WORD WINAPI dsoHTStartTrigger(WORD nDeviceIndex)

返回值：失败返回 0，成功返回非 0。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注：

通知设备准备检测触发信号。在采集状态下，调用函数 dsoHTStartCollectData 成功之后，调用本函数启动触发信号检测。

程序举例：

```
WORD nDeviceIndex = 0;
if (0 == dsoHTStartTrigger(nDeviceIndex) )
    ; //失败
else
    ; //成功
```

11. 函数声明： DLL_API WORD WINAPI dsoHTForceTrigger(WORD nDeviceIndex)

返回值： 失败返回 0，成功返回非 0。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注：

调用此函数可以进行一次强制触发采集数据。触发点随机确定。

程序举例：

```
WORD nDeviceIndex = 0;
if (0 == dsoHTForceTrigger(nDeviceIndex) )
    ; //失败
else
    ; //成功
```

12. 函数声明： DLL_API WORD WINAPI dsoHTGetState(WORD nDeviceIndex)

返回值：

当返回值为 0x07 时，表示设备已经采集完数据，可以进行读取数据。

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注：

判断设备是否采集完数据。

程序举例：

```
WORD nDeviceIndex = 0;
while ( dsoHTGeState(nDeviceIndex) != 0x07)
{
    continue;
}
```

```
//读取数据
//.....
```

13. 函数声明: DLL_API WORD WINAPI dsoSDGetData (

```
    WORD nDeviceIndex,
    WORD* pCH1Data,
    WORD* pCH2Data,
    WORD* pCH3Data,
    WORD* pCH4Data,
    PCONTROLDATA pControl,
    WORD nInsertMode
);
```

返回值: 失败返回 0，成功返回非 0。

参数:

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

pCH1Data

WORD 型指针变量。指向保存 CH1 数据的缓冲区的指针。

pCH2Data

WORD 型指针变量。指向保存 CH2 数据的缓冲区的指针。

pCH3Data

WORD 型指针变量。指向保存 CH3 数据的缓冲区的指针。

pCH4Data

WORD 型指针变量。指向保存 CH4 数据的缓冲区的指针。

pControl

见结构体说明部分。

nInsertMode

WORD 型变量，表示差值方式。0 表示台阶，1 表示直线，2 表示正弦。只有时间最短的 4 个时基才会产生效果。

备注:

读取数据

程序举例:

```
WORD nDeviceIndex = 0;
WORD nInsertMode = 2; //正弦差值
if ( 0 == dsoSDGetData ( nDeviceIndex, pCH1Data, pCH2Data, pCH3Data, pCH4Data
                        , pControl, nInsertMode) )
    ; //读取数据失败
else
    ; //读取数据成功
```

14. 函数声明: DLL_API ULONG WINAPI dsoHTGetHardFC(WORD nDeviceIndex)

返回值:

返回硬件频率计或计数器的数值。

参数:

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注:

当设置为硬件频率计时，返回的是频率值，单位是 Hz，当设置为计数器时，返回的是触发信号的个数。注意，当没有触发信号时，返回的值是不正确的。

15. 函数声明: DLL_API WORD WINAPI dsoHTSetHardFC (WORD nDeviceIndex,
WORD nType
)

返回值:

设置硬件频率计或计数器。

参数:

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

nType

WORD 型变量， 0: 打开硬件频率计,1: 打开硬件计数器,2: 关闭

备注:

硬件频率计和计数器同一时刻只能打开一个。

16. 函数声明: DLL_API WORD WINAPI dsoHTReadCalibrationData (WORD nDeviceIndex,
WORD* pLevel,
WORD nLen
)

返回值:

返回 0 表示失败，返回 1 表示成功

参数:

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

pLevel nType

WORD 型指针变量。指向外部缓冲区，保存设备的校对值。

nLen

WORD 型变量，表示缓冲区的大小，具体值见代码。

备注:

每次初始化设备，都要首先读取设备的校对值，以确保通道的零电平位置正确。

17. 函数声明: DLL_API BOOL WINAPI dsoSetUSBBus (WORD nDeviceIndex)

返回值:

返回 FALSE 表示失败，返回 TRUE 表示成功

参数：

nDeviceIndex

WORD 型变量，表示当前设备的索引值。

备注：

设置 USB 通讯总线。切换到 USB 通讯模式后，这是必须第一个被调用的操作硬件函数

控制流程图如下：

