

# SDK - HTMarch.dll 说明文档

## 中文版(VB 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.

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 HTMARCH_API
#define HTMARCH_API extern "C" __declspec(dllimport)
#endif
```

定义标准调用:

```
#define WIN_API __stdcall
```

## 函数介绍

### 1. 函数声明:

HTMARCH\_API **short** WIN\_API dsoOpenDevice(unsigned short DeviceIndex)

返回值:

返回 0 表示设备没有连接，返回 1 表示设备已连接。

参数:

DeviceIndex

设备索引值，第一台连接的设备索引值是 0，依次递增。

备注:

判断设备索引值为 DeviceIndex 的设备是否连接到 PC。

程序举例:

```
unsigned short nDev = 0;
if(dsoOpenDevice(0) == 1)
{
    ;//设备已连接
}
Else
{
    ;//没有发现设备
}
```

### 2. 函数声明:

HTMARCH\_API unsigned short WIN\_API dsoChooseDevice(unsigned short DeviceIndex, **short** nType);

**返回值：**失败返回 0，成功返回 1。

**参数：**

**DeviceIndex**

表示当前设备的索引值。

**nType**

0: 逻辑分析仪 Hantek6022BL

1: 示波器 Hantek6022BE

**备注：**

选择设备

### 3. 函数声明：

HTMARCH\_API **short** WIN\_API dsoSetVoltDIV(unsigned short DeviceIndex,**int** nCH,**int** nVoltDIV);

**返回值：**1 表示设置成功；0 表示设置失败

**参数：**

**DeviceIndex**

表示当前设备的索引值。

**nCH:**

信道索引值。0 表示 CH1，1 表示 CH2。

**nVoltDIV:**

电压档位索引值。最小电压档位为 0。以下是索引值代表的档位

0: 20mV/DIV

1: 50mV/DIV

2: 100mV/DIV

3: 200mV/DIV

4: 500mV/DIV

5: 1V/DIV

6: 2V/DIV

7: 5V/DIV

**备注：**

判断索引值为 DeviceIndex 的设备是否连接。

**程序举例：**

dsoSetVoltDIV(0,0,5); //设置 CH1 的电压档位为 1V/DIV.

### 4. 函数声明：

HTMARCH\_API **short** WIN\_API dsoSetTimeDIV(unsigned short DeviceIndex,**int** nTimeDIV);

**返回值：**1 表示设置成功；0 表示设置失败

**参数:**

nDeviceIndex

表示当前设备的索引值。

nTimeDIV

表示当前的采样率档位的索引值，以下是取值

0 ~ 10 : 48MSa/s

11: 16MSa/s

12: 8MSa/s

13: 4MSa/s

14 ~ 24: 1MSa/s

25: 500KSa/s

26: 200KSa/s

27: 100KSa/s

**备注:**

设置设备的采集率档位

**程序举例:**

**5. 函数声明:**

```
HTMARCH_API short WIN_API dsoReadHardData(  
    unsigned short DeviceIndex,  
    short* pCH1Data,  
    short* pCH2Data,  
    unsigned long nReadLen,  
    short* pCalLevel,  
    int nCH1VoltDIV,  
    int nCH2VoltDIV,  
    short nTrigSweep,  
    short nTrigSrc,  
    short nTrigLevel,  
    short nSlope,  
    int nTimeDIV,  
    short nHTrigPos,  
    unsigned long nDisLen,  
    unsigned long * nTrigPoint,  
    short nInsertMode);
```

**返回值:** 读取数据，失败返回-1， 其他表示成功。

**参数:**

unsigned short DeviceIndex : 设备的索引值  
short\* pCH1Data: 存储CH1数据的缓冲区指针  
short\* pCH2Data: 存储CH2数据的缓冲区指针  
unsigned long nReadLen: 读取数据的长度  
short\* pCalLevel: 校对电平(参考函数dsoGetCalLevel)  
int nCH1VoltDIV: CH1的电压档位  
int nCH2VoltDIV: CH2的电压档位  
short nTrigSweep: 扫频模式—0: AUTO; 1: Normal; 2: Single  
short nTrigSrc: 触发信源--- 0: CH1; 1: CH2  
short nTrigLevel: 触发电平 – 0 ~ 255  
short nSlope: 触发沿方式—0: Rise; 1: Fall  
int nTimeDIV: 采样率档位  
short nHTrigPos: 水平触发位置---0 ~ 100  
unsigned long nDisLen: 显示数据的长度  
unsigned long \* nTrigPoint: 返回触发点的索引值  
short nInsertMode: 差值方式—0: Step 差值; 1: Line 差值; 2: SinX/X 差值

**备注:**

读取数据时调用此函数。

**6. 函数声明:**

HTMARCH\_API unsigned short WIN\_API dsoGetCalLevel(unsigned short DeviceIndex, short\* level, short nLen);

返回值: 失败返回 0, 成功返回非 0。

**参数:**

DeviceIndex

表示当前设备的索引值。

level

存储校对数据的缓冲区。

nLen

校对数据的长度, 这里=32。

**备注:**

获取设备的校对数据。

**程序举例:**

```
short nCal[32];  
dsoGetCalLevel(0, nCal, 32);
```

**7. 函数声明:**

HTMARCH\_API short WIN\_API dsoCalibrate(unsigned short nDeviceIndex, int

nTimeDIV,int nCH1VoltDIV,int nCH2VoltDIV,short\* pCalLevel);

**返回值：**失败返回 0，成功返回非 0。

**参数：**

nDeviceIndex

表示当前设备的索引值。

nTimeDIV

采样率档位

nCH1VoltDIV

CH1 的电压档位

nCH2VoltDIV

CH2 的电压档位

pCalLevel

存储校对数据的存储区

**备注：**

当任意通道的零基准发生偏移时，可调用此函数进行校正，校正信息存储在 pCalLevel 中。若没有发生偏移，则不需要调用此函数

**8. 函数声明：**

HTMARCH\_API unsigned short dsoSetCalLevel(unsigned short DeviceIndex,short\* level,short nLen);

**返回值：**失败返回 0，成功返回非 0。

**参数：**

DeviceIndex

表示当前设备的索引值。

level

校正数据存储区

nLen

校正数据的长度，这里是 32

**备注：**

当调用 dsoCalibrate 函数校正后，可调用此函数将获得的校正数据存储到设备中，以备以后直接读取使用。

**程序举例：**

```
short nLevel[32];
```

```
dsoCalibrate(0,11,5,5,nLevel); //零基准发生偏移，首先进行校对，获取校对数据
```

```
dsoSetCalLevel(0, nLevel,32); //将校对数据存储到设备中
```

