

## SDK - HTMarch.dll Manual

### VB 6.0 IDE

Note:

HTMarch.dll was compiled under VC++6.0.

The following ifdef block is the standard way of creating macros which make exporting from a DLL simpler. All files within this DLL are compiled with the HTMARCH\_API symbol defined on the command line. this symbol should not be defined on any project that uses this DLL. This way any other project whose source files include this file see HTMARCH\_API functions as being imported from a DLL, whereas this DLL sees symbols defined with this macro as being exported.

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

```
#define WIN_API __stdcall
```

### Function Introduction

#### 1. Function declaration:

```
HTMARCH_API short WIN_API dsoOpenDevice(unsigned short DeviceIndex)
```

**Return value:** Return zero (0) indicates device isn't connected; return one (1) indicates device connected.

#### Parameter:

DeviceIndex

The first connected device index is 0, and others sequentially numbered.

#### Remark:

The device whose device index value is judged as DeviceIndex whether connected to PC or not.

#### Program example:

```
unsigned short nDev = 0;
if(dsoOpenDevice(0) == 1)
{
    ;// Device connected
}
Else
{
```

```
        ;// Not detect device  
    }
```

## 2. Function declaration:

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

**Return value:** Return zero (0) indicates failure; return one (1) indicates success.

### Parameter

#### DeviceIndex

indicates current device index value.

#### nType

0: logic analyzer Hantek6022BL

1: Hantek6022BE

### Remark:

Choose device

## 3. Function declaration:

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

**Return value:** one (0) for setup success and zero (0) for failure.

### Parameter:

#### DeviceIndex

indicates current device index value.

#### nCH:

Channel index value. 0 stands for CH1, and 1 stands for CH2.

#### nVoltDIV:

Voltage index value. Minimum voltage is 0, and following is the index values for corresponding voltages.

0: 20mV/DIV

1: 50mV/DIV

2: 100mV/DIV

3: 200mV/DIV

4: 500mV/DIV

5: 1V/DIV

6: 2V/DIV

7: 5V/DIV

### Remark:

The device whose device index value is judged as DeviceIndex whether connected to PC or not.

**Programme example:**

dsoSetVoltDIV(0,0,5); // Set the voltage of CH1 to 1V/DIV.

**4. Function declaration:**

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

**Return value:** one (0) for setup success and zero (0) for failure.

**Parameter**

nDeviceIndex

indicates current device index value.

nTimeDIV

indicates current sampling rate index value, following is the value.

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

**Remark:**

Setup device sampling rate.

**Programme example:****5. Function declaration:**

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);

```

**Return value:** Reading data, return “-1” for failure and non “-1” for success.

**Parameter:**

unsigned short DeviceIndex: Device index value  
 short\* pCH1Data: CH1 data storage buffer pointer  
 short\* pCH2Data: CH2 data storage buffer pointer  
 unsigned long nReadLen: The length of reading data  
 short\* pCalLevel: Proofreading level (reference function dsoGetCalLevel)  
 int nCH1VoltDIV: The voltage of CH1  
 int nCH2VoltDIV: The voltage of CH2  
 short nTrigSweep: SWP MODE-0: AUTO; 1: Normal; 2: Single  
 short nTrigSrc: Trigger source - 0: CH1; 1: CH2  
 short nTrigLevel: Trigger level - 0 ~ 255  
 short nSlope: Trigger Slope - 0: Rise; 1: Fall  
 int nTimeDIV: Sampling rate  
 short nHTrigPos: Horizontal trigger position -0 ~ 100  
 unsigned long nDisLen: The length of the display data  
 unsigned long \* nTrigPoint: The index value of returned trigger point  
 short nInsertMode: D-value mode - 0: Step D-value; 1: Line D-value; 2: SinX/X D-value

**Remark:**

Call this function to read data.

## 6. Function declaration:

```

HTMARCH_API unsigned short WIN_API dsoGetCalLevel(unsigned short
DeviceIndex,short* level,short nLen);

```

**Return value:** return zero (0) for success and non-zero for failure.

**Parameter:**

DeviceIndex  
 indicates current device index value.  
 level  
 Proofreading data storage buffer.  
 nLen  
 The length of proofreading data, here=32.

**Remark:**

Acquire proofreading data from device.

**Programme example:**

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

**7. Function declaration:**

```
HTMARCH_API short WIN_API dsoCalibrate(unsigned short nDeviceIndex, int nTimeDIV, int  
nCH1VoltDIV, int nCH2VoltDIV, short* pCalLevel);
```

**Return value:** return zero (0) for success and non-zero for failure.

**Parameter:**

nDeviceIndex

indicates current device index value.

nTimeDIV

Sampling rate

nCH1VoltDIV

The voltage of CH1

nCH2VoltDIV

The voltage of CH2

pCalLevel

Proofreading data memory area

**Remark:**

When any channel's zero reference offset, it's able to call this function to calibrate, and the calibration information stored at pCalLevel. If no offset, no need to call this function.

**8. Function declaration:**

```
HTMARCH_API unsigned short dsoSetCalLevel(unsigned short DeviceIndex, short* level, short  
nLen);
```

**Return value:** return zero (0) for success and non-zero for failure.

**Parameter:**

DeviceIndex

indicates current device index value.

level

Proofreading data memory area

nLen

The length of proofreading data, here is 32.

**Remark:**

After calling the function-dsoCalibrate to calibrate, it's able to call this function to store the acquired calibration data to device for future directly reading purpose.

**Programme example:**

```
short nLevel[32];  
dsoCalibrate(0,11,5,5,nLevel); // When zero reference offset, calibrate it first, then acquire  
the calibration data.  
dsoSetCalLevel(0, nLevel,32); // Store the calibration data to device.
```