

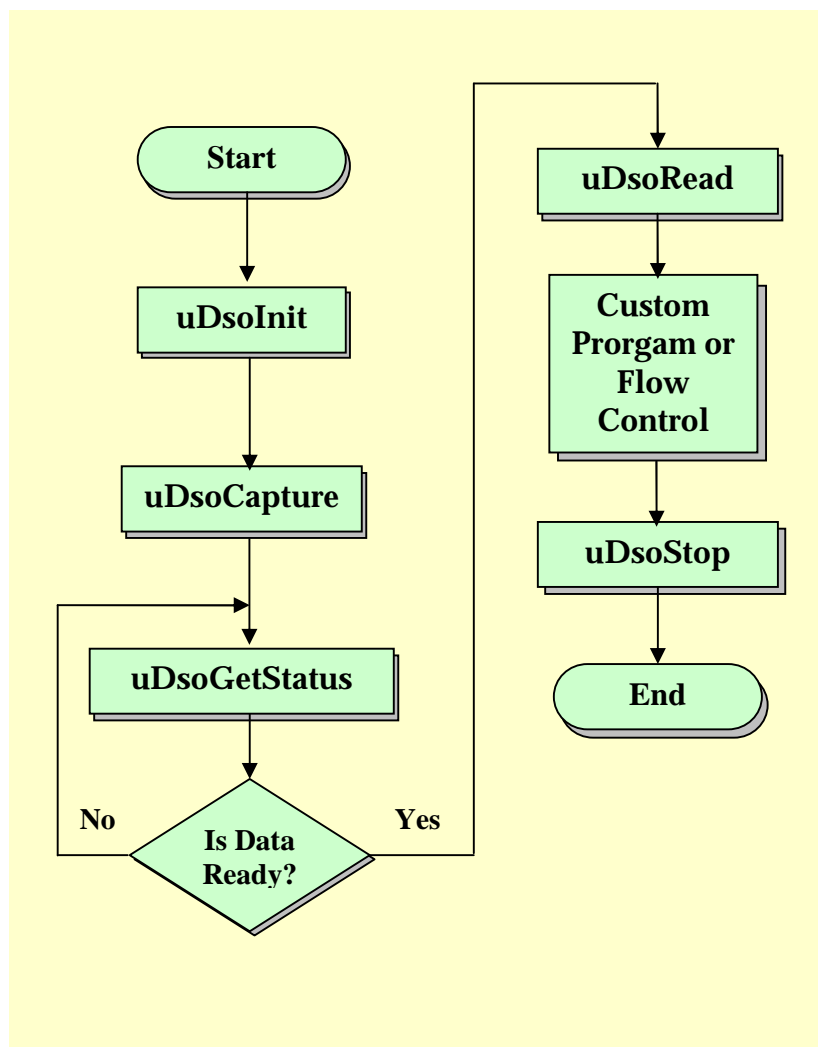
DSORUN.DLL Manual

: What is DSORUN.DLL ?

DSORUN.DLL is a dynamic-link library for Windows OS. It provides several function calls to control the Acute Digital Storage Oscilloscope (DS-1000 Series). You may use some languages that support DLL link function, such as Visual C or Visual Basic, to control Digital Storage Oscilloscope (DSO) with DSORUN.DLL library. Here, we illustrate some examples using Visual C and Visual Basic. For other languages, please refer to their respective description about DLL link application.

DSORUN.DLL includes the following function calls:

```
INT uDsoInit();  
BOOL uDsoGetHwInfo( int iDev, int iItem, LPVOID  
                    lpData );  
BOOL uDsoCapture( LPDSTRIGEX lpdstrex );  
INT uDsoGetErrorCode();  
BOOL uDsoStop();  
INT uDsoGetStatus( int iDev );  
BOOL uDsoDataReady();  
BOOL uDsoRead( LPDSDATAEX lpddex );
```

: *Flow Chart*: *Function Call Reference*

The DSORUN.DLL function call reference is C style. And this reference is valid for version 1.0.4.2 or later. You must run the DSO AP to calibrate the DSO hardware before to use the DSORUN.DLL.

uDsoInit

Initialize the digital oscilloscope.

```
INT uDsoInit();
```

Return values

The return value means how many DSO devices can be found. If the return value is 0 means no DSO was found.

Remarks

You must call the **uDsoInit** to initialize DSO before using any other function calls of DSORUN.DLL. The **uDsoInit** not only initialize DSO, but also check DSO hardware. You may call the **uDsoInit** only once.

uDsoGetHwInfo

Get DSO information.

```
BOOL uDsoCapture( int iDev, int iItem,
                  LPVOID lpData );
```

Parameters

nIndex:

- 0: Get vender name. The lpData points to the buffer that is to receive the copied string of the vender name. And the buffer size must greater than 32 bytes.
- 1: Get product name. The lpData points to the buffer that is to receive the copied string of the product name. And the buffer size must greater than 32 bytes.
- 2: Get serial number. The lpData points to the buffer that is to receive the copied string of the serial number. And the buffer size must greater than 16 bytes.
- 3: Get hardware version. The lpData points to a integer variable.
- 4: Get firmware version. The lpData

points to a integer variable.

Return values

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

uDsoCapture

Start capturing signals data.

```
BOOL uDsoCapture( LPDSTRIGEX lpdstrex );
```

Parameters

lpdstrex

A pointer of DSO Structure is defined as below:

```
typedef struct _DSTRIGEX
{
    int          iFlag;
    __int64      i64Sc;
    __int64      i64Thres;
    __int64      i64VoltDiv[12];
    __int64      i64TrPos;
    __int64      i64Wait;
    int          iBufLen;
```

```

int      iSpecial;
int      iCtrlPara;
int      iRes[11];
} DSTRIGEX, FAR *LPDSTRIGEX;

```

iFlag:

```

#define DSTEM_CH2      0x00000001
#define DSTEM_H2L      0x00000004
#define DSTEM_TYPE     0x000F0000
#define DSTEM_CH       0x00000000
#define DSTEM_EXT      0x00010000
#define DSTEM_SCANLINE 0x00020000
#define DSTEM_FIELD    0x00030000
#define DSTEM_ODDFIELD 0x00040000
#define DSTEM_EVENFIELD 0x00050000
#define DSTEM_WMMASK   0x00300000
#define DSTEM_WMQUICK  0x00000000
#define DSTEM_WMSLOW   0x00100000
#define DSTEM_WMFOREVER 0x00200000
#define DSTEM_WMCUSTOM 0x00300000

```

DSTEM_CH2 means that DSO's trigger input is CH2; otherwise, it is CH1. DSTEM_H2L means that trigger slope is "high to low"; otherwise, it is "low to high". DSTEM_TYPE is mask of trigger type in iFlag. The type can be any of DSTEM_CH, DSTEM_EXT, DSTEM_SCANLINE, DSTEM_ODDFIELD or DSTEM_EVENFIELD.

The DSTEM_WMMASK is mask of wait mode in iFlag. The mode can be any of DSTEM_WMQUICK, DSTEM_WM_SWLO, DSTEM_WMFOREVER or DSTEM_WMCUSTOM.

i64Sc:

The i64Sc is sampling rate of DSO. The unit is Hertz. And the value is just 5GHz, 2.5GHz, 100MHz, 50MHz, 20MHz, 10MHz, 5MHz, 2MHz, 1MHz, 500KHz, 200KHz, 100KHz, 50KHz, 20KHz, 10KHz, 5KHz, 2KHz, 1KHz, 500Hz, 200Hz or 100Hz.

i64Thres:

The i64Thres is threshold of trigger. The value can not be greater than +volt/div * 8 or less than -volt/div * 8. The unit is micro-volt.

i64VoltDiv:

The i64VoltDiv is voltage of division on DSO screen. The i64VoltDiv[0] means CH1's volt/div and so on. The unit is micro-volt. And the value is just 10V, 5V, 2V, 1V, 500mV, 200mV, 100mV, 50mV, 20mV, 10mV, 5mV or 2mV.

i64TrPos:

The i64TrPos is time of trigger position. When

the i64TrPos is 0, it means that the trigger position is in center of screen (buffer). The unit is pico-second.

Ps. In 512K memory depth version, when you set the value of iBufLen greater than 64k, the trigger position of buffer is 32768th point.

Ex1. iBufLen = 5000, i64TrPos = 0

The trigger position of buffer is 2500.

Ex2. iBufLen = 6000, i64TrPos = -10000ps, i64Sc = 100000000Hz

The trigger position of buffer is 2999.

Ex3. iBufLen > 65536, i64TrPos = 0

The trigger position of buffer is 32768.

Ex4. iBufLen > 65536, i64TrPos = 20000ps, i64Sc = 100000000Hz

The trigger position of buffer is 32770.

i64Wait:

The i64Wait is time to wait for trigger. If trigger doesn't occur until i64Wait, DSO will be time-out. The i64Wait unit is pico-second.

iBufLen:

The iBufLen is length of capturing. The maximum value is 64K or 512K (Please refer hardware specification for maximum memory depth).

iSpecial:

Just can be 0.

iCtrlPara:

```
#define SW_CH1_AC      0x00000001
#define SW_CH2_AC      0x00000002
#define SW_CH3_AC      0x00000004
#define SW_CH4_AC      0x00000008
#define SW_CH5_AC      0x00000010
#define SW_CH6_AC      0x00000020
#define SW_CH1_BWL     0x00010000
#define SW_CH2_BWL     0x00020000
#define SW_CH3_BWL     0x00040000
#define SW_CH4_BWL     0x00080000
#define SW_CH5_BWL     0x00100000
#define SW_CH6_BWL     0x00200000
```

The iCtrlPara can handle the coupling of channel.

Return values

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

uDsoGetErrorCode

You may use this function call to get the DSO error code when you got the error return after to call the other function calls.

```
INT uDsoGetErrorCode( ) ;
```

Return values

- 0: No Error.**
- 1: Sampling rate error.**
- 2: Volt/Div error.**
- 3: Buffer length setting error.**
- 4: No DSO hardware.**
- 5: Receive buffer can't write.**
- 6: Trigger position is out of range.**
- 7: Threshold is out of range.**
- 8: WaitCount is out of range.**
- 9: This hardware can't support TV trigger.**
- 10: In ETS mode can't use TV trigger.**
- 11: In ETS mode can't use WaitForever.**

uDsoGetStatus

Get DSO current status.

```
INT uDsoGetStatus( int iDev );
```

Parameters

iDev

The iDev parameter specifies the zero-based index of

the of DSO devices.

Return values

- Bit 0: In capture data.
- Bit 1: In capture post-trigger data.
- Bit 2: In wait for trigger.
- Bit 3: In capture pre-trigger data.
- Bit 4: Hardware button was pushed.

Remarks

When you call this function, the hardware button flag will be cleared

uDsoDataReady

Check DSO capturing status.

```
BOOL uDsoDataReady( );
```

Return

If the capturing finish, the return value is nonzero (TRUE).

If the capturing not finish, the return value is zero (FALSE).

uDsoRead

Read back the DSO captured data.

```
BOOL uDsoRead( LPDSDATAEX lpddex );
```

Parameters

lpddex: points to DSDATAEX structure.

```
typedef struct _DSDATAEX
{
    int      iType;
    int      iFlag;
    int      iTrigOfs;    // in ps
    void      *pvWaveData;
} DSDATAEX, FAR *LPDSDATAEX;
```

iType:

The iType can be set to the following value. It means which type of waveform data you need.

```
#define DDEX_RAW      0x0000
#define DDEX_INT_MV   0x0001
#define DDEX_INT_UV   0x0002
#define DDEX_DBL_MV   0x0003
#define DDEX_DBL_UV   0x0004
#define DDEX_I64_MV   0x0005
```

```
#define DDEX_I64_UV    0x0006
```

iFlag:

Bit 0: Time-out occurs.

iTrigOfs:

The iTrigOfs is only available at 100Mz sampling rate. The iTrigOfs unit is pico-second.

pvWaveData:

Points to buffer that is to receive the waveform data. The pvWaveData can be cast to other types, depending on iType value.

Return values

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

Remarks

You must prepare enough buffers to receive the waveform data. The formula is defined as below.

Buffer Size = *Device Counts* * 2 * *dstrigex.iBufLen* * *size of iType*

Device Counts: How many DSO devices you install on the PC.

2: Two Channels for each device.

dstrigex.iBufLen: How many horizontal pixels you want to show on screen.

size of iType: DDEX_INT_MV or DDEX_INT_UV = sizeof(int),

DDEX_DBL_MV or DDEX_DBL_UV = sizeof(double),

DDEX_I64_MV or DDEX_I64_UV = sizeof(__int64)