SDK API 使用手册

Version 1.20

微目电子科技

2025-08-12

http://www.vimu.top/

升级记录

V1.0 (2023.3.31)

初始版本

V1.1 (2023.5.22)

逻辑分析仪通道触发支持 s

V1.2 (2023.6.26)

增加 watchdog 开关

V1.3 (2023.8.19)

增加 MSO21 设备支持

增加 DDS ARB 和门控 API

V1.4 (2023.11.06)

Linux 系统增加稳定性

DLLTest 支持重新拔插，自动连接并采集

V1.5 (2023.12.14)

增加 MSO10 和 MSO21 V2 设备支持

Linux 读取 4MB 以上数据

V1.6 (2024.03.17)

DDS 增加 brust APIs

V1.7 (2024.05.15)

增加示波器和 logic 触发位置读取 API

V1.8 (2024.07.15)

MSO41 系列支持

V1.9 (2024.09.29)

MSO21 V3 和 MSO10 V2 支持

V1.10 (2024.11.21)

修复不是全通道采集，读取数据错位问题

修复 usb3.0 linux 系统 bug

V1.11 (2025.03.09)

增加系统稳定性

增加占空比计算

V1.20 (2025.07.28)

增加 DDS 和 IO API

目录

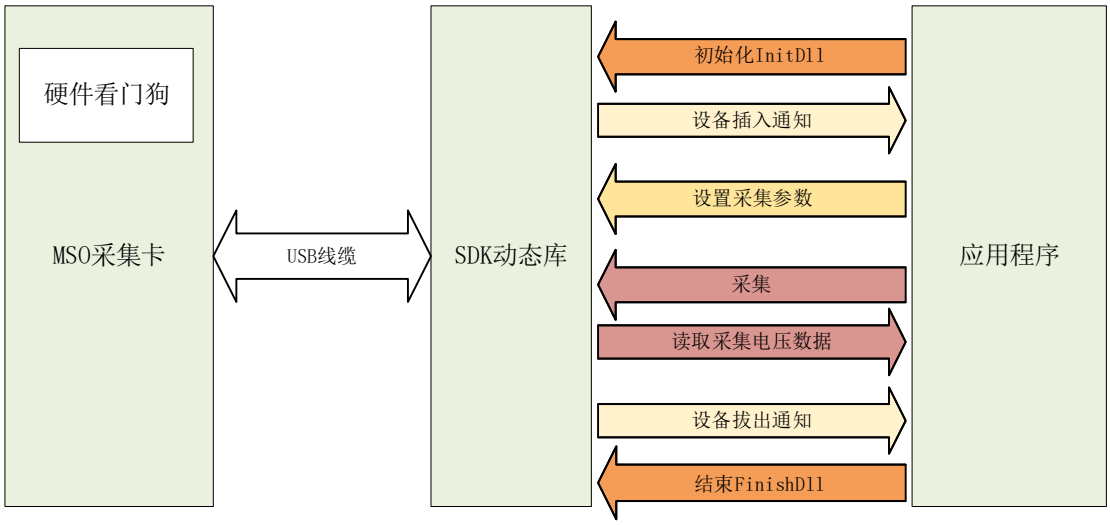
1. 简介 1
2. 系统工作框图 1
3. 循环检测模式 1
4. 回调函数模式 3
5. API 说明 4
5.1. 初始化和结束 4
5.2. 设备 ID 5
5.3. 设备复位 5
5.4. 设备监测 5
5.4.1. 回调函数 5
5.4.2. Event 6
5.4.3. 循环检测 6
5.5. 示波器 6
5.5.1. 采集范围设置 6
5.5.2. 采样率 7
5.5.3. 触发 7
5.5.4. AC/DC 11
5.5.5. 采集 11
5.5.6. 采集完成通知 12
5.5.6.1. 回调函数 12
5.5.6.2. Event 13
5.5.6.3. 循环检测 13
5.5.7. 数据读取 13
5.6. DDS 14
5.7. IO 21
回调函数 21
Event 21
循环检测 22

1. 简介

作为 MOS 混合信号示波器配备的标准 DLL 接口，通过这个接口可以直接控制混合信号示波器。

该接口支持 widows 系统(X86，X64 和 ARM64)和 linux 系统(X64，ARM32 和 ARM64)。

2. 系统工作框图



MSO 采集卡：硬件采集卡部分，通过 USB 和主机通信

SDK 动态库：负责和硬件通信；将应用程序的参数传输给硬件卡，将硬件卡的采集数据返回给应用程序

应用程序：用户设计的程序。

说明：

1、 InitDll 的参数 **watchdog enable** 可以启动采集卡的看门狗，最后交付的程序记得将看门狗启动。调试的时候，可以根据需要将看门狗关闭；

2、 SDK 动态库线程是一个单独的线程。使用回调模式的时候，在回调函数中，不能处理复杂的任务。如果回调函数占用时间太长，超过看门狗的阈值时间，将会导致硬件采集卡重启。

3、 采集卡的工作模式，大致可以分为“循环检测”和“回调函数”模式。开发应用的时候可以根据实际的需要来选择，也可以根据实际的需求混合使用。

3. 循环检测模式

循环检测模式，就是通过 api 轮询来查询采集卡的状态，然后相应的做出处理。DllTest-Polling 就是一个循环检测的例子。代码如下：

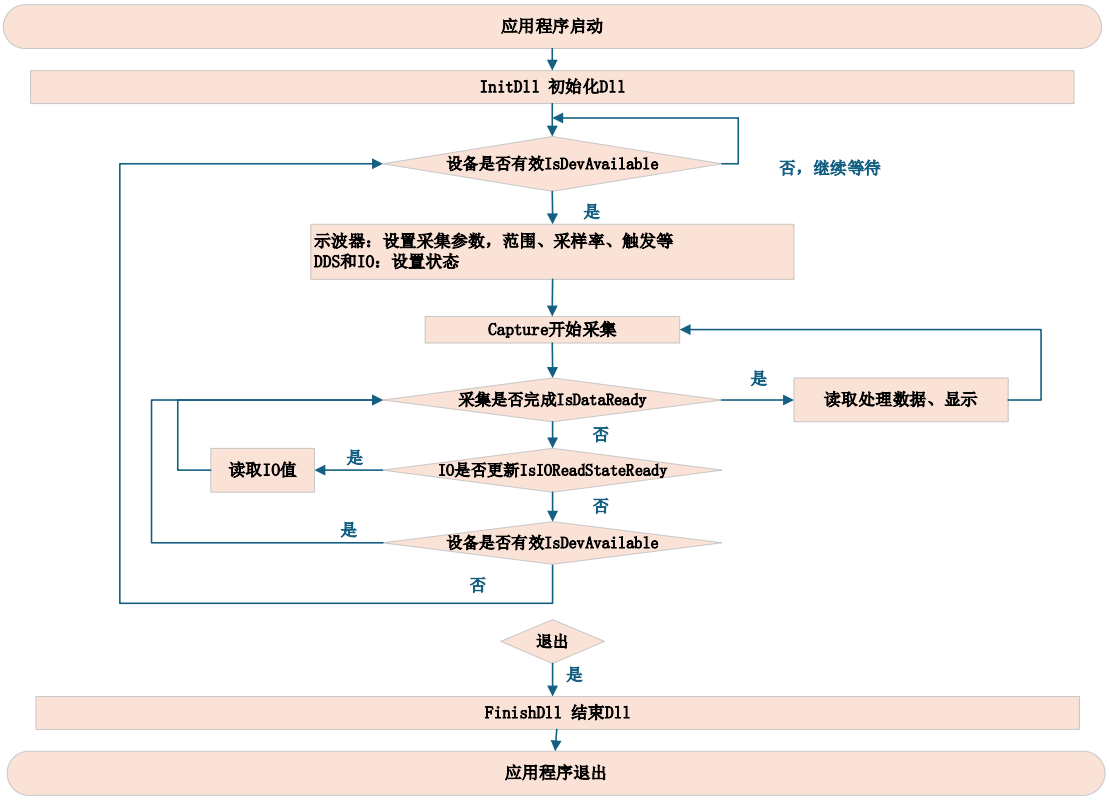
```
int main()
{
    std::cout << "Vdso Test..." << std::endl;
    InitDll(1, 1);
    std::this_thread::sleep_for(std::chrono::milliseconds(500));
     ScanDevice();

    while (true)
```

```
    {
        //device connection is successful?
        if(!devAvailable)
        {
            devAvailable = IsDevAvailable();

            //init functions
            if(devAvailable)
            {
                std::cout << "devAvailable start init" << std::endl;
                initFunction();
            }
            else
                std::this_thread::sleep_for(std::chrono::milliseconds(500));
        }

        if(devAvailable)
        {
            if (IsDataReady())
            {
                ReadDatas();
                NextCapture();
            }
            else if(IsIOReadStateReady())
            {
                std::cout << "io state " << std::hex << GetIOInState() <<" \n";
            }
            else
            {
                //test device is active
                devAvailable = IsDevAvailable();
                if(!devAvailable)
                    std::cout << "devAvailable is false" << std::endl;

                std::this_thread::sleep_for(std::chrono::milliseconds(200));
            }
        }
    };
    FinishDll();
    std::cout << "...Vdso Test" << std::endl;
    return 0;
}
```

开始，先检测设备是否插入，如果没有插入，线程休眠 500ms，然后重新检测；检测到设备插入就开始各个功能的初始化，并调用 Capture 开始采集数据。

然后，开始轮询检测是不是示波器是否采集完成，IO 是否有更新。如果都没有更新，顺便检测一下设备是否还在插入状态。

示波器或者 IO 有更新了，就读取数据做相应的处理工作，然后进行下一次采集。如果检测到设备拔出，就重新进入设备检测轮询。



4. 回调函数模式

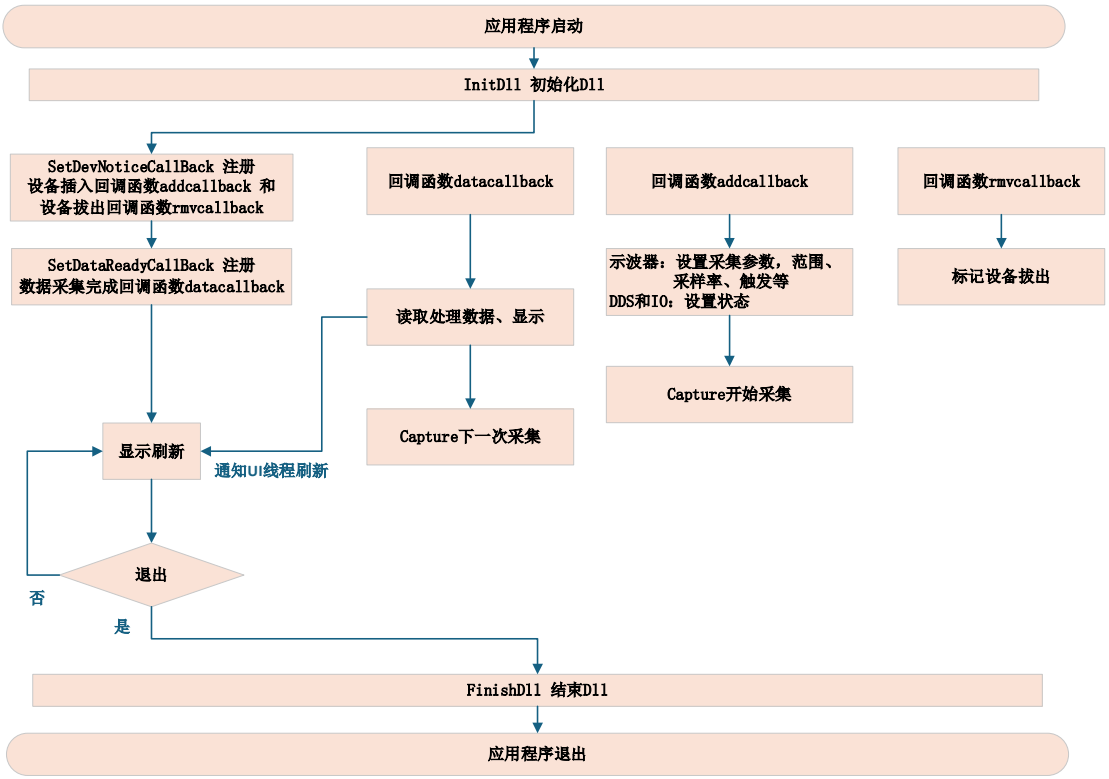
回调函数模式，就是通过注册回调函数来获取采集卡的状态，然后相应的做出处理。

回调函数模式，涉及到采集卡线程和 UI 线程的同步，开发程序的时候需要根据实际的情况来处理线程间的同步问题。

DllTest-Callback 就是一个回调函数的例子。代码如下：

```
int main()
{
    std::cout << "Vdso Test..." << std::endl;

    InitDll(1, 1);

    //OSC
    SetDevNoticeCallBack(NULL, DevNoticeAddCallBack, DevNoticeRemoveCallBack);
    SetDataReadyCallBack(NULL, DevDataReadyCallBack);

     std::this_thread::sleep_for(std::chrono::milliseconds(500));
     ScanDevice();

    while (true)
```

```
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    };

    FinishDll();
    std::cout << "...Vdso Test" << std::endl;
    return 0;
}
```



回调函数 addcallback：负责设备插入后各个功能的初始化，然后开始采集。
回调函数 rmcallback：负责设备拔出后的标记和及状态处理。
回调函数 datacallback：负责采集数据的读取，然后进行下一次采集。

5. API 说明

5.1. 初始化和结束

调用InitDll()来完成动态库的初始化，初始化的时候会分配内存和资源用于设备监测和数据读取用。

int InitDll(unsigned int en_log , unsigned int en_hard_watchdog);

Description　Dll initialization

Input:　**log enable**　1 Enable Log
　　0 Not Enable Log
　　watchdog enable 1 Enable hard watchdog
　　0 Not Enable hard watchdog

Output:　**Init Status**
　　Return value 1 Success

0 Failed

调用FinishDll()来完成动态库的结束，结束的时候，会时释放初始化中申请的内存和相关资源。

int FinishDll(void);

Description Dll finished

Input: -

Output: -**Finished Status**

Return value 1 Success

0 Failed

5.2. 设备 ID

每个设备都有一个 64 位的 ID 码。

int GetOnlyId0(void);

Description This routines return device id(0-31)

Input: -

Output: - **Device ID(0-31)**

int GetOnlyId1(void);

Description This routines return device id(32-63)

Input: -

Output: - **Device ID(32-63)**

5.3. 设备复位

int ResetDevice(void);

Description This routines reset device

Input: -

Output: - **Return value** 1 success

0 failed

5.4. 设备监测

当 DLL 检测到有设备接入时，有 3 种方式通知主程序，回掉函数、触发 Event 和主程序循环检测。

5.4.1. 回调函数

当检测到设备插入时，如果主程序注册了回掉函数**"addcallback"**，它就会被调用；当检测到设备拔出时，如果主程序注册了回掉函数**"rmvcallback"**，它就会被调用。Dll 有一个函数专门用于设置这个 2 个回掉函数

void SetDevNoticeCallBack(void* ppara, AddCallBack addcallback, RemoveCallBack rmvcallback);

Description This routines sets the callback function of equipment status changed.

Input: **ppara** the parameter of the callback function

addcallback a pointer to a function with the following prototype:
void AddCallBack(void * **ppara**)

rmvcallback a pointer to a function with the following prototype:
Void RemoveCallBack(void * **ppara**)

Output -

5.4.2. Event

当检测到设备插入时，如果主程序注册了 Event 句柄**"addevent"**，它就会被设置；当检测到设备拔出时，如果主程序注册了回掉函数**"rmvevent"**，它就会被设置。需要注意的是，主程序检测到 Event 后，需要将 Event 复位。Dll 有一个函数专门用于设置这 2 个 Event 句柄

void SetDevNoticeEvent(HANDLE addevent, HANDLE rmvevent);

Description This routines set the event handle, these will be set, when equipment status changed.

Input: **addevent** the event handle

rmvevent the event handle

Output -

5.4.3. 循环检测

int IsDevAvailable();

Description This routines return the device is available or not.

Input: -

Output **Return value** 1 available

0 not available

说明：3 方式只要使用其中的一种就可以了，回掉函数和 Event 都是异步的处理方式，更加的高效；循环检测需要主程序过一定时间就检测设备是否插入或者拔出。

5.4.4. 扫描设备

软件启动完成，可以使用 ScanDevice 来扫描已经插入的 USB 设备。

int ScanDevice();

Description Rescan Device

Input: -

Output: -Rescan Status

Return value 1 Successs

0 Failed

5.5. 示波器

5.5.1. 采集范围设置

设备的前级带有程控增益放大器，当采集的信号小于 AD 量程的时候，增益放大器可以把信号放大，更多的利用 AD 的位数，提高采集信号的质量。Dll 会根据设置的采集范围，自动的调整前级的增益放大器。

int SetOscChannelRangemV(int channel, int minmv, int maxmv);

Description This routines set the range of input signal.

Input: **channel** the set channel

0 channel 1

1 channel 2

minmv the minimum voltage of the input signal (mV)

maxmv the maximum voltage of the input signal (mV)

Output **Return value** 1 Success

0 Failed

说明：最大的采集范围为探头 X1 的时候，示波器可以采集的最大电压。比如 MSO20 为 [-12000mV,12000mV]。

注意：为了达到更好波形效果，一定要根据自己被测波形的幅度，设置采集范围。必要时，可以动态变化采集范围。

5.5.2. 采样率

int GetOscSupportSampleRateNum();

Description This routines get the number of samples that the equipment support.

Input: -

Output **Return value** the support sample number

int GetOscSupportSampleRates(unsigned int* sample, int maxnum);

Description This routines get support samples of equipment.

Input: **sample** the array store the support samples of the equipment

maxnum the length of the array

Output **Return value** the sample number of array stored

int SetOscSampleRate(unsigned int sample);

Description This routines set the sample.

Input: **sample** the set sample

Output **Return value** 0 Failed

other value new sample

unsigned int GetOscSampleRate();

Description This routines get the sample.

Input: -

Output **Return value** sample

5.5.3. 触发

该功能需要设备硬件触发支持。硬件触发的触发点都是采集数据的最中间，比如采集 128K 数据，触发点就是第 64K 的点。

触发模式

```
#define TRIGGER_MODE_AUTO 0
#define TRIGGER_MODE_LIANXU 1
```

触发条件

```
#define TRIGGER_STYLE_NONE 0x0000          //not trigger
#define TRIGGER_STYLE_RISE_EDGE 0x0001     //Rising edge
#define TRIGGER_STYLE_FALL_EDGE 0x0002     //Falling edge
#define TRIGGER_STYLE_EDGE 0x0004          //Edge
#define TRIGGER_STYLE_P_MORE 0x0008        //Positive Pulse width(>)
#define TRIGGER_STYLE_P_LESS 0x0010        //Positive Pulse width(>)
#define TRIGGER_STYLE_P      0x0020        //Positive Pulse width(<>)
#define TRIGGER_STYLE_N_MORE 0x0040        //Negative Pulse width(>)
#define TRIGGER_STYLE_N_LESS 0x0080        //Negative Pulse width(>)
```

#define TRIGGER_STYLE_N 0x0100 //Negative Pulse width(<>)

int IsSupportHardTrigger();

Description This routines get the equipment support hardware trigger or not .

Input: **-**

Output **Return value** 1 support hardware trigger

0 not support hardware trigger

unsigned int GetTriggerMode();

Description This routines get the trigger mode.

Input: **-**

Output **Return value** TRIGGER_MODE_AUTO

TRIGGER_MODE_LIANXU

void SetTriggerMode(unsigned int mode);

Description This routines set the trigger mode.

Input: **mode** TRIGGER_MODE_AUTO

TRIGGER_MODE_LIANXU

Output -

unsigned int GetTriggerStyle();

Description This routines get the trigger style.

Input: **-**

Output **Return value** TRIGGER_STYLE_NONE

TRIGGER_STYLE_RISE_EDGE

TRIGGER_STYLE_FALL_EDGE

TRIGGER_STYLE_EDGE

TRIGGER_STYLE_P_MORE

TRIGGER_STYLE_P_LESS

TRIGGER_STYLE_P

TRIGGER_STYLE_N_MORE

TRIGGER_STYLE_N_LESS

TRIGGER_STYLE_N

void SetTriggerStyle(unsigned int style);

Description This routines set the trigger style.

Input: **style** TRIGGER_STYLE_NONE

TRIGGER_STYLE_RISE_EDGE

TRIGGER_STYLE_FALL_EDGE

TRIGGER_STYLE_EDGE

TRIGGER_STYLE_P_MORE

TRIGGER_STYLE_P_LESS

TRIGGER_STYLE_P

TRIGGER_STYLE_N_MORE

TRIGGER_STYLE_N_LESS
TRIGGER_STYLE_N

Output -

int GetTriggerPulseWidthNsMin();

Description This routines get the min time of pulse width.

Input: -

Output Return min time value of pulse width(ns)

int GetTriggerPulseWidthNsMax();

Description This routines get the max time of pulse width.

Input: -

Output Return max time value of pulse width(ns)

int GetTriggerPulseWidthDownNs();

Description This routines get the down time of pulse width.

Input: -

Output Return down time value of pulse width(ns)

int GetTriggerPulseWidthUpNs();

Description This routines set the down time of pulse width.

Input: down time value of pulse width(ns)

Output -

void SetTriggerPulseWidthNs(int down_ns, int up_ns);

Description This routines set the up time of pulse width.

Input: up time value of pulse width(ns)

Output _

unsigned int GetTriggerSource();

Description This routines get the trigger source.

Input: **-**

Output **Return value** TRIGGER_SOURCE_CH1 0 //CH1
TRIGGER_SOURCE_CH2 1 //CH2
TRIGGER_SOURCE_LOGIC0 16 //Logic 0
TRIGGER_SOURCE_LOGIC1 17 //Logic 1
TRIGGER_SOURCE_LOGIC2 18 //Logic 2
TRIGGER_SOURCE_LOGIC3 19 //Logic 3
TRIGGER_SOURCE_LOGIC4 20 //Logic 4
TRIGGER_SOURCE_LOGIC5 21 //Logic 5
TRIGGER_SOURCE_LOGIC6 22 //Logic 6
TRIGGER_SOURCE_LOGIC7 23 //Logic 7

void SetTriggerSource(unsigned int source);

Description　This routines set the trigger source.

Input:　**source**　TRIGGER_SOURCE_CH1 0　//CH1
TRIGGER_SOURCE_CH2 1　//CH2
TRIGGER_SOURCE_LOGIC0 16　//Logic 0
TRIGGER_SOURCE_LOGIC1 17　//Logic 1
TRIGGER_SOURCE_LOGIC2 18　//Logic 2
TRIGGER_SOURCE_LOGIC3 19　//Logic 3
TRIGGER_SOURCE_LOGIC4 20　//Logic 4
TRIGGER_SOURCE_LOGIC5 21　//Logic 5
TRIGGER_SOURCE_LOGIC6 22　//Logic 6
TRIGGER_SOURCE_LOGIC7 23　//Logic 7

Output　-

注意：如果逻辑分析仪和 IO 是复用的（例如 MSO20、MSO21），需要将对应的 IO 打开，并设置为输入状态。

int GetTriggerLevelmV();

Description　This routines get the trigger level.

Input:　**-**

Output　**Return value**　level (mV)

void SetTriggerLevelmV (int level, int sense);

Description　This routines set the trigger level and sense.

Input:　level (mV)
sense (mV)

Output　**-**

int GetTriggerSensemV();

Description　This routines get the trigger sense.

Input:　**-**

Output　**Return value**　Sense (mV)

说明：触发灵敏度可以让触发稳定性更好，比如设置触发为上升沿，触发电平为 1V，触发灵敏度为 100mV。采集卡检测到 1V 以后，会继续检测 1.1V，如果达到了就认为触发满足条件；如果没有检测到 1.1V 就会认为没有满足，应该是毛刺干扰。

bool IsSupportPreTriggerPercent();

Description　This routines get the equipment support Pre-trigger Percent or not .

Input:　-

Output　Return value 1 support
0 not support

int GetPreTriggerPercent();

Description　This routines get the Pre-trigger Percent.

Input:　-

Output　Return value Percent (5-95)

void SetPreTriggerPercent(int front);

Description This routines set the Pre-trigger Percent.

Input: Percent (5-95)

Output -

int IsSupportTriggerForce();

Description This routines get the equipment support trigger force or not.

Input: **-**

Return value 1 support

0 not support

void TriggerForce();

Description This routines force capture once.

Input: **-**

Output: **-**

5.5.4. AC/DC

int IsSupportAcDc(unsigned int channel);

Description This routines get the device support AC/DC switch or not.

Input: channel 0 :channel 1

1 :channel 2

Output **Return value** 0 : not support AC/DC switch

1 : support AC/DC switch

void SetAcDc(unsigned int channel, int ac);

Description This routines set the device AC coupling.

Input: channel 0 :channel 1

1 :channel 2

ac 1 : set AC coupling

0 : set DC coupling

Output **-**

int GetAcDc(unsigned int channel,);

Description This routines get the device AC coupling.

Input: channel 0 :channel 1

1 :channel 2

Output **Return value** 1 : AC coupling

0 : DC coupling

5.5.5. 采集

调用**Capture**函数开始采集数据，**length**就是你想要采集的长度，以K为单位，比如**length**=10,就是10K 10240个点。对于采样率的大于等于存储深度的采集长度，取**length**和存储深度的最小值；对于采样率小于存储深度，取**length**和1秒采集数据的最小值。函数会返回实际采集数据的长度。**force_length**可以强制取消只能采集1秒的限制。

int Capture(int length, unsigned short capture_channel,char force_length);

Description This routines set the capture length and start capture.

Input: **length** capture length(KB)

capture_channel

ch1=0x0001 ch2=0x0002 ch3=0x0004 ch4=0x0008 logic=0x0100

ch1+ch2 0x0003

ch1+ch2+ch3 0x0007

ch1+logic 0x0101

force_length 1: force using the length, no longer limits the max collection 1 seconds

Output **Return value** the real capture length(KB)

使用正常触发模式（TRIGGER_MODE_LIANXU）的时候。发送了采集命令，还没有收到采集完成数据通知。现在，想要停止软件。

1、推荐方式：你把触发模式改成TRIGGER_MODE_AUTO，等待收到采集完成数据通知，再停止软件。

2、使用 **AbortCapture.**

DLL_API int WINAPI AbortCapture();

Description This routines set the abort capture

Input:

Output Return value 1:success 0:failed

unsigned int GetMemoryLength();

Description This routines get memory depth of equipment (KB).

Input: -

Output memory depth of equipment(KB)

Roll Mode: 该模式下，采样率被固定的设置为最小采样率，采集长度也是固定的设置为 1 秒采集数据长度。正常的调用 **Capture**，把每次采集的数据连接在一起显示就是完整的波形。

int IsSupportRollMode();

Description This routines get the equipment support roll mode or not .

Input: -

Output **Return value** 1 support roll mode

0 not support roll mode

int SetRollMode(unsigned int en);

Description This routines enable or disenable the equipment into roll mode.

Input: -

Output **Return value** 1 success

0 failed

5.5.6. 采集完成通知

当数据采集完成时，有 3 种方式通知主程序，回掉函数、触发 Event 和主程序循环检测。

5.5.6.1. 回调函数

当数据采集完成时，如果主程序注册了回掉函数**"datacallback"**，它就会被调用。Dll 有一个函数专门用于设置这个回掉函数

void SetDataReadyCallBack(void* ppara, DataReadyCallBack datacallback);

Description　This routines sets the callback function of capture complete.

Input:　**ppara**　the parameter of the callback function

datacallback　a pointer to a function with the following prototype:

void **DataReadyCallBack** (void * **ppara**)

Output　-

5.5.6.2. Event

当数据采集完成时，如果主程序注册了 Event 句柄**"dataevent"**，它就会被设置。需要注意的是，主程序检测到 Event 后，需要将 Event 复位。Dll 有一个函数专门用于设置这个 Event 句柄

void SetDevDataReadyEvent(HANDLE dataevent);

Description　This routines set the event handle, these will be set, when capture complete

Input:　**dataevent**　the event handle

Output　-

5.5.6.3. 循环检测

int IsDataReady();

Description　This routines return the capture is complete or not.

Input:　-

Output　**Return value** 1 complete

0 not complete

说明：3 方式只要使用其中的一种就可以了，回掉函数和 Event 都是异步的处理方式，更加的高效；循环检测需要主程序开始采集以后，过一定时间就检测是否采集完成。

5.5.7. 数据读取

unsigned int ReadVoltageDatas(char channel, double* buffer,unsigned int length);

Description　This routines read the voltage datas. (V)

Input:　**channel**　**read channel**　0 :channel 1

1 :channel 2

buffer　the buffer to store voltage datas

length　the buffer length

Output　**Return value** the read length

unsigned int ReadVoltageDatasTriggerPoint();

Description　This routines read the trigger location where the data collected

Input:

Output　**Return value** the trigger points

int IsVoltageDatasOutRange(char channel);

Description　This routines return the voltage datas is out range or not.

Input:　**channel**　**read channel**　0 :channel 1

1 :channel 2

Output　**Return value**　0 :not out range

1 :out range

double GetVoltageResolution(char channel);

Description This routines return the current voltage resolution value

One ADC resolution for the voltage value:

Full scale is 1000mv

the ADC is 8 bits

voltage resolution value = 1000mV/256

Input: **channel** **read channel** 0:channel 1

1:channel 2

Output **Return value** voltage resolution value

unsigned int ReadLogicDatas(unsigned char* buffer, unsigned int length);

Description This routines read the logic data of mso.

Input:

buffer the buffer to store logic datas

length the buffer length

Output Return value the read length

unsigned int ReadLogicDatasTriggerPoint();

Description This routines read the trigger location where the data collected

Input:

Output **Return value** the trigger points

5.6. DDS

int IsSupportDDSDevice();

Description This routines get support dds or not

Input: **-**

Output **Return value** support dds or not

int GetDDSDepth();

Description This routines set dds depth

Input:

Output: **Return value** depth

void SetDDSOutMode(unsigned char channel_index, unsigned int out_mode);

Description This routines set dds out mode

Input: **channel_index** 0 :channel 1

1 :channel 2

out_mode DDS_OUT_MODE_CONTINUOUS 0x00

DDS_OUT_MODE_SWEEP 0x01

DDS_OUT_MODE_BURST 0x02

Output

unsigned int GetDDSOutMode(unsigned char channel_index);

Description This routines get dds out mode

Input: channel_index 0 :channel 1

1 :channel 2

Output **mode** DDS_OUT_MODE_CONTINUOUS 0x00

DDS_OUT_MODE_SWEEP 0x01

DDS_OUT_MODE_BURST 0x02

int GetDDSSupportBoxingStyle(int* style);

Description This routines get support wave styles

Input: **style** array to store support wave styles

Output **Return value** if style==NULL return number of support wave styles

else store the styles to array, and return number of wave styles

void SetDDSBoxingStyle(unsigned char channel_index, unsigned int boxing);

Description This routines set wave style

Input: channel_index 0 :channel 1

1 :channel 2

Input: **boxing** **W_SINE = 0x0001,**

W_SQUARE = 0x0002,

W_RAMP = 0x0004,

W_PULSE = 0x0008,

W_NOISE = 0x0010,

W_DC = 0x0020,

W_ARB = 0x0040

Output: -

void UpdateDDSArbBuffer(unsigned char channel_index, unsigned short* arb_buffer, uint32_t arb_buffer_length);

Description This routines update arb buffer

Input: **channel_index** 0 :channel 1

1 :channel 2

arb_buffer the dac buffer

arb_buffer_length the dac buffer length need equal to the dds depth

Output: -

void SetDDSPinlv(unsigned char channel_index, unsigned int pinlv);

Description This routines set frequence

Input: **channel_index** 0 :channel 1

1 :channel 2

Input: **pinlv** frequence

Output: -

void SetDDSDutyCycle(unsigned char channel_index, int cycle);

Description This routines set duty cycle

Input: **channel_index** 0 :channel 1

1 :channel 2

Input: **cycle** duty cycle

Output: -

int GetDDSCurBoxingAmplitudeMv(unsigned int boxing);

Description This routines get dds amplitdude of wave

Input: **boxing** BX_SINE~BX_ARB

Output: Return the amplitdude(mV) of wave

void SetDDSAmplitudeMv(unsigned char channel_index, int amplitdude);

Description This routines set dds amplitdude(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

amplitdude amplitdude(mV)

Output: -

int GetDDSAmplitudeMv(unsigned char channel_index);

Description This routines get dds amplitdude(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: return amplitdude(mV)

int GetDDSCurBoxingBiasMvMin(unsigned int boxing);

int GetDDSCurBoxingBiasMvMax(unsigned int boxing);

Description This routines get dds bias of wave

Input: **boxing** BX_SINE~BX_ARB

Output: Return the bias(mV) range of wave

void SetDDSBiasMv(unsigned char channel_index, int bias);

Description This routines set dds bias(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

bias bias(mV)

Output: -

int GetDDSBiasMv(unsigned char channel_index);

Description This routines get dds bias(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: Return the bias(mV) of wave

void SetDDSSweepStartFreq(unsigned char channel_index, double freq);

Description This routines set dds sweep start freq

Input: **channel_index** 0 :channel 1

1 :channel 2

freq

Output: -

double GetDDSSweepStartFreq(unsigned char channel_index);

Description This routines get dds sweep start freq

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **freq**

void SetDDSSweepStopFreq(unsigned char channel_index, double freq);

Description This routines set dds sweep stop freq

Input: **channel_index** 0 :channel 1

1 :channel 2

freq

Output: -

double GetDDSSweepStopFreq(unsigned char channel_index);

Description This routines get dds sweep stop freq

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **freq**

void SetDDSSweepTime(unsigned char channel_index, unsigned long long int time_ns);

Description This routines set dds sweep time

Input: **channel_index** 0 :channel 1

1 :channel 2

time/ns

Output: -

unsigned long long int GetDDSSweepTime(unsigned char channel_index);

Description This routines get dds sweep time

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **time/ns**

void SetDDSBurstStyle(unsigned char channel_index, int style);

Description This routines set dds burst style

Input: **channel_index** 0 :channel 1
1 :channel 2
style 0--n loops
1--gate
Output: -

int GetDDSBurstStyle(unsigned char channel_index);
Description This routines get dds burst style
Input: **s** 0 :channel 1
1 :channel 2
Output: **style** 0--n loops
1--gate

void SetDDSLoopsNum(unsigned char channel_index, unsigned long long int num);
Description This routines set dds loops num
Input: **channel_index** 0 :channel 1
1 :channel 2
num
Output: -

unsigned long long int GetDDSLoopsNum(unsigned char channel_index);
Description This routines get dds loops num
Input: **channel_index** 0 :channel 1
1 :channel 2
Output: **num**

void SetDDSLoopsNumInfinity(unsigned char channel_index, int en);
Description This routines set dds loops num infinity
Input: **channel_index** 0 :channel 1
1 :channel 2
en
Output: -

int GetDDSLoopsNumInfinity(unsigned char channel_index);
Description This routines get dds loops num infinity
Input: **channel_index** 0 :channel 1
1 :channel 2
Output: **loops num infinity**

void SetDDSBurstPeriodNs(unsigned char channel_index, unsigned long long int ns);
Description This routines set dds burst period(ns)
Input: **channel_index** 0 :channel 1
1 :channel 2

ns

Output: -

unsigned long long int GetDDSBurstPeriodNs(unsigned char channel_index);

Description This routines get dds burst period(ns)

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **ns**

void SetDDSBurstDelayNs(unsigned char channel_index, unsigned long long int ns);

Description This routines set dds burst delay time(ns)

Input: **channel_index** 0 :channel 1

1 :channel 2

ns

Output: -

unsigned long long int GetDDSBurstDelayNs(unsigned char channel_index);

Description This routines get dds burst delay time(ns)

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **ns**

void SetDDSTriggerSource(unsigned char channel_index, unsigned int src);

Description This routines set dds trigger source

Input: **channel_index** 0 : channel 1

1 : channel 2

src 0 : internal

1 : external

2 : manual

Output: -

unsigned int GetDDSTriggerSource(unsigned char channel_index);

Description This routines get dds trigger source

Input: **channel_index** 0 : channel 1

1 : channel 2

Output: **trigger source** 0 : internal

1 : external

2 : manual

void SetDDSTriggerSourceIo(unsigned char channel_index, uint32_t io);

Description This routines set dds trigger source io

Input: **channel_index** 0 : channel 1

1 : channel 2

	io	0 : DIO0
		.....
		7 : DIO7
Output:	-	

Note：需要使用DIO API，将对应的DIO设置为输入/输出状态

uint32_t GetDDSTriggerSourceIo(unsigned char channel_index);

Description	This routines get dds trigger source io	
Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	**trigger source io**	0 : DIO0
		.....
		7 : DIO7

void SetDDSTriggerSourceEnge(unsigned char channel_index, unsigned int enge);

Description	This routines set dds trigger source enge	
Input:	**channel_index**	0 : channel 1
		1 : channel 2
	enge	0 : rising
		1 : falling
Output:	-	

unsigned int GetDDSTriggerSourceEnge(unsigned char channel_index);

Description	This routines get dds trigger enge	
Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	**enge**	0 : rising
		1 : falling

void SetDDSOutputGateEnge(unsigned char channel_index, unsigned int enge);

Description	This routines set dds output gate enge	
Input:	**channel_index**	0 : channel 1
		1 : channel 2
	enge	0 : close
		1 : rising
		2 : falling
Output:	-	

unsigned int GetDDSOutputGateEnge(unsigned char channel_index);

Description	This routines get dds output gate enge	
Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	**enge**	0 : close
		1 : rising

2 : falling

void DDSManualTrigger(unsigned char channel_index);

Description This routines manual trigger dds

Input: **channel_index** 0 : channel 1

1 : channel 2

Output: -

void DDSOutputEnable(unsigned char channel_index , int enable);

Description This routines enable dds output or not

Input: **channel_index** 0 : channel 1

1 : channel 2

enable 1 enable

0 not enable

Output: -

int IsDDSOutputEnable(unsigned char channel_index);

Description This routines get dds output enable or not

Input: **channel_index** 0 : channel 1

1 : channel 2

Output **Return value** dds enable or not

5.7. IO

int IsSupportIODevice();

Description This routines get support IO ctrl or not

Input: -

Output Return value support io ctrl or not

int GetSupportIoNumber();

Description This routines get support io nums of equipment.

Output Return value the sample number of io nums

当 IO 设置为输入时，有 3 种方式读取 IO 状态，回掉函数、触发 Event 和主程序循环检测。

回调函数

SDK 会定时读取 IO 状态，如果主程序注册了回掉函数**"datacallback"**，它就会被调用。Dll 有一个函数专门用于设置这个回掉函数

void SetIOReadStateCallBack(void* ppara, IOReadStateCallBack callback);

Description This routines sets the callback function of read io status.

Input: **ppara** the parameter of the callback function

callback a pointer to a function with the following prototype

Event

SDK 会定时读取 IO 状态，如果主程序注册了 Event 句柄**"dataevent"**，它就会被设置。需要注意的是，主程序检测到 Event 后，需要将 Event 复位。Dll 有一个函数专门用于设置这个 Event 句柄

void SetIOReadStateReadyEvent(HANDLE dataevent);

Description This routines set the event handle, these will be set, when capture complete

Input: **dataevent** the event handle

Output -

循环检测

int IsIOReadStateReady();

Description This routines return read io is complete or not.

Input: -

Output **Return value** 1 complete

0 not complete

说明：3 方式只要使用其中的一种就可以了，回掉函数和 Event 都是异步的处理方式，更加的高效；循环检测需要主程序开始采集以后，过一定时间就检测是否采集完成。

void IOEnable(unsigned char channel, unsigned char enable);

Description This routines set io enable or not

Input: **channel** dio0 0

dio1 1

dio2 2

.....

enable not enable 0

enable 1

Output: -

unsigned char IsIOEnable(unsigned char channel);

Description This routines get io enable or not

Input: **channel** dio0 0

dio1 1

dio2 2

.....

Output: **return** not enable 0 or enable 1

void SetIOInOut(unsigned char channel, unsigned char inout);

Description This routines set io in or out

Input: **channel** dio0 0

dio1 1

dio2 2

.....

inout in 0

out 1

Output: -

unsigned char GetIOInOut(unsigned char channel);

Description This routines get io in or out

Input: **channel** dio0 0
dio1 1
dio2 2
.....

Output: **return** in 0
out 1

void SetIOOutState(unsigned char channel, unsigned char state);

Description This routines set io state

Input: **channel** dio0 0
dio1 1
dio2 2
.....
state 0--0
1--1
2--z
3--pulse
4—dds gate

Output: -

void SendAutoReadIOInTimeMs(uint32_t ms);

Description This routines set auto read thread io in state time interval

Input: ms the time min is 100ms

IO 的输入状态读取，默认的读取定时是 500ms，可以通过这个函数修改读取的定时时间。最小间隔是 100ms；

unsigned int SendReadIOInStateCmd();

Description This routines send read hardware io cmd

Note:When sent and the hardware has returned, IsIOReadStateReady returns true

如果 100ms 不能满足实时读取 IO 的需求，可以使用这个函数，手动发送读取硬件 IO 输入状态命令，读取完成后，IsIOReadStateReady 变为真状态；使用 GetIOInState 读取当前的 IO 输入状态。

unsigned int GetIOInState();

Description This routines get io state

If the SetIOReadStateCallBack setting callback function is used, IOReadStateCallBack will directly notify the IO input status; If use SetIOReadStateReadyEvent and IsIOReadStateReady to read the query, you need to call GetIOState to get the IO input status

Output: return io state

void SetIOPulseData(unsigned char channel, double freq, double duty);

Description This routines set io pulse freq and duty
Input: freq freq(hz)
duty 5~95

double GetIOPulseFreq(unsigned char channel);
Description This routines get io pulse freq
Output: freq

double GetIOPulseDuty(unsigned char channel);
Description This routines get io pulse duty
Output: duty 5~95

void SetIOPulseSyn();
Description This routines set io pulse syn output
Input:
Output: -