安卓 SDK 使用手册

Version 1.3

微目电子科技

2025-3-26

http://www.vimu.top/

升级记录

V1.0 (2023.9.20)

　　初始版本

V1.1 (2023.11.20)

　　增加 MSO10 和 MSO20 支持

　　增加 DDS API

V1.1 (2024.7.18)

　　修复 DDS 打开导致 aar 崩溃 bug

　　增加 MSO41 支持

　　修改说明书描述

V1.3 (2025.3.26)

　　增加重新扫描设备功能

目录

1. 简介

MSO 混合信号示波器配备的安卓 aar 接口，通过这个接口可以直接控制混合信号示波器。

该接口可以在支持 USB Host 的安卓系统上面使用。

2. 权限申请

AndroidManifest.xml 文件中添加如下信息：

2.1. USB 权限

```
<uses-feature
        android:name="android.hardware.usb.host"
        android:required="true" />

    <uses-permission android:name="android.hardware.usb.host"/>
    <uses-permission android:name="android.permission.HARDWARE_TEST"/>
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>

<intent-filter>
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>

<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
```

将 device_filter.xml 拷贝到 res/xml 目录下面。

2.2. 大堆栈权限

因为采集卡最大支持 32MB 存储深度，为了让 app 申请更多的内存，加入下面的内容。

```
android:largeHeap="true"
```

3. UsbDevMng

UsbDevMng用来管理设备的插入和拔出检测，并通过UsbDevMng.UsbDevDetectLister接口来通知。

3.1. 创建和初始化

```
usbManger = new UsbDevMng(UsbDevDetectLister UsbDevDetectLister);
usbManger.intiDetect(Context context);
```

3.2. 设备状态改变通知处理

```
public void UsbDevDetectCallback(UsbDevMng.DEVICE_DETECT_STATE state, boolean success, BasicUsbDev dev) {
    if (state == UsbDevMng.DEVICE_DETECT_STATE.DEVICE_ADD) {
        //设备插入处理
    }
    else if (state == UsbDevMng.DEVICE_DETECT_STATE.DEVICE_REMOVE) {
        //设备拔出处理
    }
```

```
        else if (state == UsbDevMng.DEVICE_DETECT_STATE.NEED_PERMISSION) {
            //没有权限处理
        }
        else if (state == UsbDevMng.DEVICE_DETECT_STATE.NEED_RSCAN) {
        }
    }
```

3.3. 重新扫描已经插入设备

有一些系统，授权以后，广播信息获取会失败，需要手动重新扫描设备，并连接。

boolean scanDevice(Context context, boolean requestDialog);

Description: Re-scan the USB device plugged into the system

Input: **context**

requestDialog If there is no permission, whether to reapply

Output: **Return value** success or failed

4. OscDdsFactory

OscDdsFactory 用来根据 BasicUsbDev 设备，创建示波器、DDS 或其他对应功能的控制类。

CreateSbqCardWave 创建示波器的控制类

BasicSbqUsbCardVer12 CreateSbqCardWave(BasicSbqUsbCardVer12.WaveReceiveLister callback, BasicUsbDev dev)

Description: Create an oscilloscope's control class.

Input: **BasicSbqUsbCardVer12.WaveReceiveLister** Waveform update notification

BasicUsbDev MSO USB device class

Output: **Return value** oscilloscope's control class

CreateDDSWave 创建 DDS 信号源的控制类

BasicHsfUsbWaveV12 CreateDDSWave(BasicUsbDev dev)

Description: Create an dds control class.

Input: **BasicUsbDev** MSO USB device class

Output: **Return value** DDS control class

5. 示波器

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。SDK 会根据设置的采集范围，自动的调整前级的增益放大器。

int SetRange(int channel, double minv, double maxv);

Description: Set the range of input signal.

Input: **channel** the set channel

0 channel 1

1 channel 2

minv the minimum voltage of the input signal (V)

maxv the maximum voltage of the input signal (V)

Output **Return value** 0 Success

-1 Device Not Open
-3 Not Support

double GetRangeMinV(byte channel)**;**
Description: Get the min range of input signal.
Input: **channel** the set channel
0 channel 1
1 channel 2
Output **Return value** **minv** the minimum voltage of the input signal (V)
-1 Device Not Open
-3 Not Support

double GetRangeMaxV(byte channel)**;**
Description: Get the max range of input signal.
Input: **channel** the set channel
0 channel 1
ss**1** channel 2
Output **Return value** **minv** the maximum voltage of the input signal (V)
-1 Device Not Open
-3 Not Support

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。

5.2. 采样率

int GetSamplesNum ();
Description: Get the number of samples that the equipment support.
Input: -
Output **Return value** the support sample number

int GetSamples(int[] sample, int maxnum);
Description: 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 SetSample(int sample);
Description: Set the sample.
Input: **sample** the set sample
Output **Return value** 0 Success
-1 Device Not Open
-3 Not Support

int GetSample();

Description　Get the sample.

Input:　　-

Output　　**Return value** sample

5.3. 触发(硬件触发)

触发模式

```
enum TRIGGER_MODE {
        AUTO(0),
        LIANXU(1)
};
```

触发条件

```
enum TRIGGER_STYLE {
        NONE(0),          //not trigger
        RISE_EDGE(1),     //Rising edge
        FALL_EDGE(2), //Falling edge
        EDGE(4), //Edge
        PULSE_P_MORE(8), //Positive Pulse width(>)
        PULSE_P_LESS(16), //Positive Pulse width(<)
        PULSE_P(32), //Positive Pulse width(<>)
        PULSE_N_MORE(64), //Negative Pulse width(>)
        PULSE_N_LESS(128), //Negative Pulse width(<)
        PULSE_N(256); //Negative Pulse width(<>)
};
```

TRIGGER_MODE GetTriggerMode();

Description:　Get the trigger mode.

Input:　　-

Output　　**Return value** TRIGGER_MODE

void SetTriggerMode(TRIGGER_MODE mode);

Description:　Set the trigger mode.

Input:　　**mode**　TRIGGER_MODE

Output	**Return value**	0	Success
		-1	Device Not Open
		-3	Not Support

TRIGGER_STYLE GetTriggerStyle();

Description:　Get the trigger style.

Input:　　-

Output　　**Return value**　　TRIGGER_STYLE

void SetTriggerStyle(TRIGGER_STYLE style);

Description:　Set the trigger style.

Input:　　**style**　　TRIGGER_STYLE

Output	**Return value**	0	Success
		-1	Device Not Open
		-3	Not Support

int GetTriggerPulseWidthNsMin();

Description: Get the min time of pulse width.

Input: -

Output Return min time value of pulse width(ns)

int GetTriggerPulseWidthNsMax();

Description: Get the max time of pulse width.

Input: -

Output Return max time value of pulse width(ns)

int GetTriggerPulseWidthDownNs();

Description: Get the down time of pulse width.

Input: -

Output Return down time value of pulse width(ns)

int GetTriggerPulseWidthUpNs();

Description: 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: Set the up time of pulse width.

Input:	**down_ns**		
	up_ns	up time value of pulse width(ns)	
Output	**Return value**	0	Success
		-1	Device Not Open
		-3	Not Support

TRIGGER_SOURCE GetTriggerSource();

Description: Get the trigger source.

Input: -

Output **Return value**

TRIGGER_SOURCE.CH1	0x0000000000000001L	//CH1
TRIGGER_SOURCE.CH2	0x0000000000000002L	//CH2
TRIGGER_SOURCE.D0	0x0000000000010000L	//Logic 0
TRIGGER_SOURCE.D1	0x0000000000020000L	//Logic 1
TRIGGER_SOURCE.D2	0x0000000000040000L	//Logic 2
TRIGGER_SOURCE.D3	0x0000000000080000L	//Logic 3
TRIGGER_SOURCE.D4	0x0000000000100000L	//Logic 4
TRIGGER_SOURCE.D5	0x0000000000200000L	//Logic 5

TRIGGER_SOURCE.D6	0x0000000000400000L	//Logic 6
TRIGGER_SOURCE.D7	0x0000000000800000L	//Logic 7

void SetTriggerSource(TRIGGER_SOURCE source);

Description: Set the trigger source.

Input: **source**

TRIGGER_SOURCE.CH1	0x0000000000000001L	//CH1
TRIGGER_SOURCE.CH2	0x0000000000000002L	//CH2
TRIGGER_SOURCE.D0	0x0000000000010000L	//Logic 0
TRIGGER_SOURCE.D1	0x0000000000020000L	//Logic 1
TRIGGER_SOURCE.D2	0x0000000000040000L	//Logic 2
TRIGGER_SOURCE.D3	0x0000000000080000L	//Logic 3
TRIGGER_SOURCE.D4	0x0000000000100000L	//Logic 4
TRIGGER_SOURCE.D5	0x0000000000200000L	//Logic 5
TRIGGER_SOURCE.D6	0x0000000000400000L	//Logic 6
TRIGGER_SOURCE.D7	0x0000000000800000L	//Logic 7

Output **Return value**

0	Success
-1	Device Not Open
-3	Not Support

注意：如果逻辑分析仪和IO是复用的（例如MSO10、MSO20、MSO21），需要将对应的IO打开，并设置为输入状态。

int GetTriggerLevel();

Description: Get the trigger level.

Input: -

Output **Return value** level (V)

void SetTriggerLevel(int level);

Description: Set the trigger level.

Input: level (V)

Output **Return value**

0	Success
-1	Device Not Open
-3	Not Support

int GetTriggerSenseDiv();

Description: Get the trigger sense.

Input: -

Output **Return value** Sense (0-1 div)

void SetTriggerSenseDiv(int sense, double y_interval_v);

Description: Set the trigger sense.

Input: Sense (0-1 div)

Interval(V)

Output **Return value**

0	Success
-1	Device Not Open
-3	Not Support

说明：sense 触发灵敏度的范围为 0.1 Div-1.0 Div。

y_interval_v 示波器软件使用的是垂直灵敏度设置，即每个格的电压值。

SDK 可以使用采集范围除以 10 来设置，即 (m_osc_range_maxv - m_osc_range_minv)/10.0，最后设置的灵敏度电压就是 (m_osc_range_maxv - m_osc_range_minv)/10.0*sense。

int GetTriggerFrontPercent ();

Description: Get the Pre-trigger Percent.

Input: -

Output Return value Percent (5-95)

void SetTriggerFrontPercent (int front);

Description: Set the Pre-trigger Percent.

Input: Percent (5-95)

Output **Return value** 0 Success

-1 Device Not Open

-3 Not Support

int IsSupportTriggerForce();

Description: Get the equipment support trigger force or not.

Input: -

Return value 1 support

0 not support

void SetTriggerForce();

Description: Force capture once.

Input: -

Output: 0 Success

-1 Device Not Open

-3 Not Support

5.4. AC/DC

int IsSupportAcDc(int channel);

Description: 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(int channel, int ac);

Description: Set the device AC coupling.

Input: **channel** 0 :channel 1

1 :channel 2

ac 1 : set AC coupling

0 : set DC coupling

Output　　0　Success
　　　　　-1　Device Not Open
　　　　　-3　Not Support

int GetAcDc(int channel);

Description:　Get the device AC coupling.

Input:　　**channel**　0 :channel 1
　　　　　　　　　1 :channel 2

Output　　**Return value**　1 : AC coupling
　　　　　　　　　　　0 : DC coupling

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, short capture_channel, byte force_length);

Description:　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**　>0 Success 返回采集长度/1024
　　　　　　　　　　　-1　Device Not Open
　　　　　　　　　　　-2　Mem Allocate Failed

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:　Set the abort capture

Input:

Output　　Return value 1:success 0:failed

int GetHardMemoryDepth();

Description:　Get memory depth of equipment (KB).

Input:　　-

Output　　memory depth of equipment(KB)

5.6.采集完成通知

当数据采集完成时，通过 **BasicSbqUsbCardVer12.WaveReceiveLister** 回掉通知主程序。

```
public boolean WaveReceiveCallBack(boolean success, int length){
   if(success) {
         //更新 UI
         runOnUiThread(new Runnable() {
               public void run() {
                     WaveReceive(length); //UI 数据处理
               }
         });
    }
    return true;
}
```

说明：通知回调函数不能访问安卓的 UI 数据，所以需要使用 runOnUiThread 来运行 UI 相应的数据处理函数。

5.7.数据读取

int ReadVoltageDatas(byte channel, double[] buffer, int length);

Description:	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		

int IsVoltageDatasOutRange(byte channel);

Description:	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(byte channel);

Description: 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	

int ReadLogicDatas(byte[] buffer, int length);

Description: Read the logic data of mso.

Input:

	buffer	the buffer to store logic datas
	length	the buffer length
Output	Return value the read length	

6. DDS

int GetDepth();

Description: Get DDS depth

Input:

Output: **Return value** depth

void SetOutMode(int channel_index, DDS_OUT_MODE out_mode);

Description: 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		

DDS_OUT_MODE GetOutMode(int channel_index);

Description: 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

void SetBoxing(int channel_index, BOXING_STYLE boxing);

Description: Set wave style

Input:	**channel_index**	0 :channel 1
		1 :channel 2
	boxing	**W_SINE = 0x0001,**
		W_SQUARE = 0x0002,
		W_RAMP = 0x0004,
		W_PULSE = 0x0008,
		W_NOISE = 0x0010,
		W_DC = 0x0020,
		W_ARB = 0x0040
Output:	**-**	

void UpdateArbBuffer(int channel_index, short[] arb_buffer, int arb_buffer_length);

Description: 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 SetFreq (int channel_index, int freq);

Description:	Set frequence	
Input:	**channel_index**	0 :channel 1
		1 :channel 2
	freq	frequence
Output:	-	

void SetDutyCycle(int channel_index, int cycle);

Description:	Set duty cycle	
Input:	**channel_index**	0 :channel 1
		1 :channel 2
	cycle	duty cycle
Output:	-	

int GetCurBoxingAmplitudeMv(BOXING_STYLE boxing);

Description:	Get DDS amplitdude of wave	
Input:	**boxing**	BX_SINE~BX_ARB
Output:	Return the amplitdude(mV) of wave	

void SetAmplitudeMv(int channel_index, int amplitdude);

Description:	Set DDS amplitdude(mV)	
Input:	**channel_index**	0 :channel 1
		1 :channel 2
	amplitdude	amplitdude(mV)
Output:	-	

int GetAmplitudeMv(int channel_index);

Description:	Get DDS amplitdude(mV)	
Input:	**channel_index**	0 :channel 1
		1 :channel 2
Output:	return amplitdude(mV)	

int GetCurBoxingBiasMvMin(BOXING_STYLE boxing);
int GetCurBoxingBiasMvMax(BOXING_STYLE boxing);

Description:	Get DDS bias of wave	
Input:	**boxing**	BX_SINE~BX_ARB
Output:	Return the bias(mV) range of wave	

void SetBiasMv(int channel_index, int bias);

Description: Set DDS bias(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

bias bias(mV)

Output: -

int GetBiasMv(int channel_index);

Description: Get DDS bias(mV)

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: Return the bias(mV) of wave

void SetSweepStartFreq(int channel_index, double freq);

Description: Set DDS sweep start freq

Input: **channel_index** 0 :channel 1

1 :channel 2

freq

Output: -

double GetSweepStartFreq(int channel_index);

Description: Get DDS sweep start freq

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **freq**

void SetSweepStopFreq(int channel_index, double freq);

Description: Set DDS sweep stop freq

Input: **channel_index** 0 :channel 1

1 :channel 2

freq

Output: -

double GetSweepStopFreq(int channel_index);

Description: Get dds sweep stop freq

Input: **channel_index** 0 :channel 1

1 :channel 2

Output: **freq**

void SetSweepTime(int channel_index, long time_ns);

Description: Set DDS sweep time

Input: **channel_index** 0 :channel 1

1 :channel 2

time/ns

Output: -

long GetSweepTime(int channel_index);

Description: Get DDS sweep time

Input:	**channel_index**	0 :channel 1
		1 :channel 2
Output:	**time/ns**	

void SetTriggerSource(int channel_index, DDS_TRIGGER_SOURCE src);

Description: Set DDS trigger source

Input:	**channel_index** 0 : channel 1	
		1: channel 1
	src	0: internal 2
		0: INTERNAL
		1: EXTERNAL
		2: MANUAL
Output:	-	

int GetTriggerSource(int 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 SetTriggerSourceIo(int channel_index, int io);

Description: 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

int GetTriggerSourceIo(int channel_index);

Description: Get DDS trigger source io

Input:	**channel_index**	0	: channel 1
		1	: channel 2
Output:	**trigger source io**	0 :	DIO0
			.....
		7 :	DIO7

void SetTriggerSourceEnge(int channel_index, DDS_ENGE enge);

Description: Set DDS trigger source enge

Input:	**channel_index**	0 : channel 1

		1 : channel 2
	enge	0 : rising
		1 : falling
Output:	-	

int GetTriggerSourceEnge(int channel_index);

Description: Get DDS trigger enge

Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	**enge**	0 : rising
		1 : falling

void SetOutputGateEnge(int channel_index, DDS_OUTPUT_ENGE enge);

Description: Set DDS output gate enge

Input:	**channel_index**	0 : channel 1
		1 : channel 2
	enge	0 : close
		1 : rising
		2 : falling
Output:	-	

int GetOutputGateEnge(int channel_index);

Description: Get DDS output gate enge

Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	**enge**	0 : close
		1 : rising
		2 : falling

void ManualTrigger(int channel_index);

Description: Manual trigger DDS

Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	-	

void ChannelStart (int channel_index);

Description: Enable DDS output or not

Input:	**channel_index**	0 : channel 1
		1 : channel 2
Output:	-	

boolean ChannelIsStart (int channel_index);

Description: Get DDS output enable or not

Input: -

Output **Return value** DDS enable or not