Acute Logic Analyzer Software development kit (SDK) Programming guide

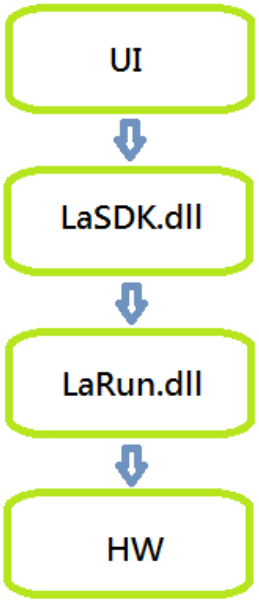
Version: 1.8

(For DLL version 1.0.4.0 or higher)

Publish: 2018/12/03

Contents

LASDK Control Flow and simple introduction



LaSDK is a warper of the LaRun.dll, translate those complicated parameters in LaRun.dll to an user-friendly format. The LaSDK also check the possible error for invalid parameters, users can read the error code and check from LASDK_Err.h if any error happened.

LPSDK_TRIG Structure Introduction

```
typedef struct _SDKTRIG
{
    int iFlag;                  //Trigger flag
    int iDelay;                 //Trigger delay for N sample clocks
    int iWidth;                 //Set trigger width to N sample clocks
                                //Ex：Sample Rate = 200MHz, iDelay = 5, iWidth = 10
                                //Delay time        5 * (1 / 200MHz) = 25 ns
                                //Trigger width    10 * (1 / 200MHz) = 50 ns
    int iPassCount;             //Trigger condition passcount
    int iFreq;                  //Sample rate (Hz)
    int iFreqHi;                //Reserved, fill with zero
    int iExtClk;                //External clock flag
    int iTrPos;                 //Trigger position
    int lpiCont[16];            //Trigger condition buffer (Do not change the content manually)
    BYTE lpbTrigData[1024];     //Trigger data buffer (Do not change the content manually)
    } SDK_TRIG, FAR *LPSDK_TRIG;
```

iFlag:

Value		Description
#define TR_DBLMODE	0x0001	Double capturing mode (For PKLA and LA2000 series)
#define TR_PRETRIG	0x0002	Enable Pre-Trigger function (For LA2000P and TravelLogic series)
#define TR_XCHG	0x0004	Dual-Condition trigger mode switch (For LA2000P and TravelLogic series)

iExtClk:

Value		Description
#define TRF_INT	0x0001	Using internal sampling clock
#define TRF_EXT	0x0002	Using external sampling clock
#define TRF_CH15	0x0004	External clock path in CH15
#define TRF_CH31	0x0008	External clock path in CH31
#define TRF_CH63	0x0010	External clock path in CH63
#define TRF_OR	0x0020	0: Internal AND External 1: Internal OR External
#define TRF_NOT	0x0040	0: Sampling at External clock's rising edge. 1: Sampling at External clock's falling edge.
#define TRF_CUSTOM	0x0080	Use External Clock (Channel 35) (For TravelLogic series)
#define TRF_TL_CH35	0x000	External clock mode:
#define TRF_TL_NOT_CH35	0x100	Channel 35
#define TRF_TL_CH34_AND_CH35	0x200	! Channel 35
#define TRF_TL_CH34_OR_CH35	0x300	Channel 34 & Channel 35 Channel 34 | Channel 35 (For TravelLogic series)

LASDK Function Definitions

BOOL ulaSDKInit()

Search and initial all LA devices connect on the PC, Ex: TravelLogic, PKLA1616

Return value

Return true if device initialized successful。

Remarks

If there are multiple devices connect on the PC, this function will initial all devices in the same time.

After the initializations, the dll will select the index 0 device as the default selected device.

BOOL ulaSDKSelectDevice(char* szSerialNo)

Select LA device by specified Serial number.

Parameters

szSerialNum[in/out]:

Type：char*

Contains the serial number of the specified LA device.

Return value

Return true when device selected successful.

Remarks

To control multiple LAs, you should use this function to specify the target LA before assigning the parameters.

Ex:

```
ulaSDKSelectDevice("TL22360000");
ulaSDKSetSamplesNum(1000);          //Set device TL22360000's sample points to 1000
ulaSDKSelectDevice("TL22360001");
ulaSDKSetSamplesNum(2000);          //Set device TL22360001's sample points to 2000
```

INT ulaSDKGetLaID()

Get LA device ID to check the LA model.

Return value

Return the LA device ID. Ex: 0x2136 stand for model TL2136.

Remarks

ID	Model
0x2036	TL2036
0x2136	TL2136
0x2236	TL2236
0x1032	LA1032
0x1064	LA1064
0x2032	LA2032
0x2064	LA2064
0x2132	LA2132
0x2164	LA2164
0x1116	PKLA1116
0x1216	PKLA1216
0x1616	PKLA1616

BOOL ulaSDKGetSerialNumber(char *szSerialNum, int iSize)

Get current selected device serial number.

Parameters

szSerialNum[in/out]：

Type：**char***

Contains the device's serial number, Ex：TL22061234. Buffer size must greater than 35 bytes.

iSize[in]：

Type：**int**

Buffer size in byte.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKSetHwInfo(int iIndex, LPVOID lpv)

Set LA hardware mode, including sample rate and available channels.

Parameters

iIndex [in]

Type：int

Input SET_TL_MODE = 5 to set the hardware mode. Other value is reserved.

lpv [in]:

Type：LPVOID

When user input SET_TL_MODE in **iIndex** parameter, this parameter will be recognized as an int pointer indicates the selected hardware mode.

Index	Sample Rate	Available Channel(s)
HW_4G_36CH	4GHz	36
HW_4G_18CH	4GHz	18
HW_4G_9CH	4GHz	9
HW_4G_4CH	4GHz	4
HW_2G_36CH	2GHz	36
HW_1600M_4CH	1.6GHz	4
HW_800M_9CH	800MHz	9
HW_400M_18CH	400MHz	18
HW_200M_36CH	1Hz - 200MHz	36
HW_200M_18CH	1Hz - 200MHz	18
HW_200M_12CH	1Hz - 200MHz	12
HW_200M_9CH	1Hz - 200MHz	9
HW_200M_6CH	1Hz - 200MHz	6
HW_200M_4CH	1Hz - 200MHz	4
HW_200M_2CH	1Hz - 200MHz	2
HW_200M_1CH	1Hz - 200MHz	1
HW_EXT_35CH	External Clock	35
HW_EXT_18CH	External Clock	18
HW_EXT_12CH	External Clock	12
HW_EXT_9CH	External Clock	9
HW_EXT_6CH	External Clock	6
HW_EXT_4CH	External Clock	4
HW_EXT_2CH	External Clock	2
HW_EXT_1CH	External Clock	1
HW_TR200M_32CH	200MHz	32
HW_TR200M_8CH	200MHz	8

Hardware Parameter Setting Table

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

Ex: For Sample Rate 200MHz, available channel number 9

```
int iHWMode = HW_200M_9CH;
if (!ulaSDKSetHwInfo(SET_TL_MODE, &iHWMode))
    AfxMessageBox("Hw Info Set Error!");
```

The HW_TR200M_32CH and HW_TR200M_8CH are provided as Transitional storage mode, the

timestamped waveform must be read from ulaSDKGetTrWfm, and the Conventional data read function are not supported in Transitional storage mode.

Except External Clock mode and 200MHz Mode, other modes are fixed at specific sample rate!

BOOL ulaSDKSetChTrigger(LPSDK_TRIG lpSDKtr, int iLevel, int iCh, int iTrig, int iCondLogic)

Set the Trigger condition to LPSDK_TRIG struct.

Parameters

lpSDKtr [in, out]

Type：LPSDK_TRIG

Trigger structure, must be passed to **ulaSDKCapture** function in order to capture data.

iLevel / iCh /iTrig [in]：

Type：int

Trigger level, type and channel select. **iTrig** should be one of the value described below.

Value		Trigger condition
#define LA_TRIG_DONT_CARE	0x08	Don't care, triggered at any condition.
#define LA_TRIG_LOW	0x00	Triggered at Low Signal.
#define LA_TRIG_HIGH	0x06	Triggered at High Signal.
#define LA_TRIG_RISING	0x04	Triggered at rising edge.
#define LA_TRIG_FALLING	0x02	Triggered at falling edge.
#define LA_TRIG_CHANGE	0x0A	Triggered at any edge.

iCondLogic [in]：

Type：int

Value		Description
#define TR_NEXT	0x00	Two trigger conditions must be located in two continuous samples.
#define TR_NEXTIF	0x01	Two trigger conditions can be separated by other samples.
#define TR_TRIGGER	0x02	Trigger the LA after trigger condition goes true.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

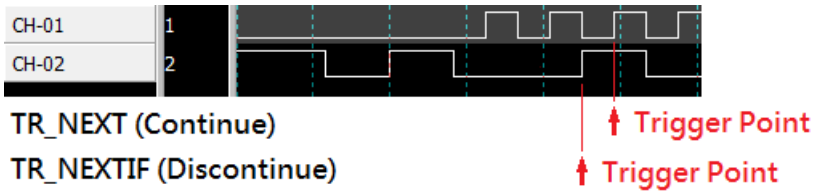
Ex1: Set Trigger condition to Channel 3 must be High at trigger level 2.

iLevel = 2,

iCh = 3,

iTrig = LA_TRIG_HIGH

Ex2: Set Trigger Level 0 to Channel 1 Rising and Trigger Level 1 to Channel 2 HIGH with different trigger continue conditions:



The maximum number of trigger levels is 16, but it was also limited by the trigger condition.

If any level in the trigger struct used **LA_TRIG_CHANGE**, the trigger level will be limited to 4.

If any level in the trigger struct used **LA_TRIG_RISING / LA_TRIG_FALLING**, the number will be limited to 8.

Users can use this function to create a multi-level trigger condition.

It is available to call **ulaSDKClearTrigger**() to initialize the trigger settings.

It is recommended to set the trigger channels within the available channels selected by HW mode, in order to make the reading data more readable with the trigger channel's data.

BOOL ulaSDKClearTrigger(LPSDK_TRIG lpSDKtr)

Initial and clear all trigger condition.

Parameters

lpSDKtr[in, out]：

Type：LPSDK_TRIG

Trigger Structure.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

This function will initialize all parameters to the default setting.

The default settings of the LPSDK_TRIG are described at below:

```
lpSDKtr->iDelay      = 0;          //Trigger delay
lpSDKtr->iExtClk     = 1;          //Use internal clock
lpSDKtr->iFlag       = TR_PRETRIG; //Enable Pre-Trigger function
lpSDKtr->iFreq       = 200000000;  //Sample Rate = 200MHz
lpSDKtr->iFreqHi     = 0;          //Sample Rate = 200MHz
lpSDKtr->iPassCount  = 0;          //Trigger Pass Count
lpSDKtr->iTrPos      = 360;        //Trigger position
                  //Trigger signal may not appear on the waveform if this value is too small to keep them.
lpSDKtr->iWidth      = 0;          //Trigger Width   (for LA 2000)
```

BOOL ulaSDKSetSamplesNum(int iSize)

Set the sample numbers to be captured by LA.

Maximum value = (LA total memory / all opening channel number).

Parameters

iSize[in]：

Type：int

The sample number will be automatically set to a valid value if the user input a value greater than the maximum available value.

The maximum value can be read by calling **ulaSDKGetMaxSamplesNum**() function.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKThreshold(int iPod, int iMilliVolt)

Set LA voltage threshold. This value is recommended to set to the mid of target signal.

Parameters

iPod [in]：

Type：int

The Pod number was varying from the LA models. Each Pod has its independent voltage threshold

Model	Pod	Channel
PKLA series	0	CH0 – CH15
TravelLogic series	0	CH0 – CH17
	1	CH18 – CH35
LA 1000、2000 series	0	CH0 – CH15
	1	CH16 – CH31
	2	CH32 – CH47
	3	CH48 – CH63

iMilliVolt [in]：

Type：int

Voltage threshold in millivolt.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKSetChannelMask(BYTE *lpbChMask, int iSize)

Set 32 bytes signal mask to the logic analyzer, the signal activity of masked channel will be ignored and all data in these masked channels will be recorded as zero.

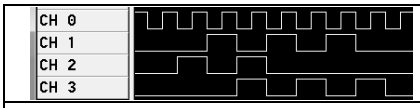
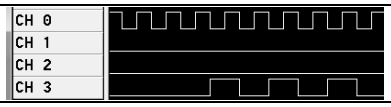
If input array size < 32 bytes, only the specified channel mask will be set, and the channel mask on other channels will be cleared.

Parameters

lpbChMask [in]

Type：BYTE *

Input the channel mask to ignore signal activity for specified channels.

	
Mask = 0x00 (Default)	Mask = 0x06 (Ignore CH1 & CH2)

iSize [in]

Type：int

Input the size of the lpbChMask array.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKSetHwFilter(__int64 i64ChOnOff, int iFilterTime_ns)

Set H/W glitch filter for each channel in bit-field to the logic analyzer, the maximum available Filter time is 35ns.

Parameters

i64ChOnOff [in]

Type：__int64

Input the H/W glitch filter on/off option for each channel in bit-field.

Enable H/W Filter on CH0	Enable H/W Filter on CH0 & CH5
i64ChOnOff = 0x01	i64ChOnOff = 0x01 | 0x10

iFilterTime_ns [in]

Type：int

Input the glitch filter value, unit: ns. Maximum available input: 35ns.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKCapture(LPSDK_TRIG lpSDKtr)

Send capture command to the LA.

Parameters

lpSDKtr [in]

Type：LP SDK_TRIG

Input the trigger structure to be used to capture data.

Return value

If the function succeeded, the return value is a nonzero value. The LED indicator on the LA turns to red when the LA is capturing data.

If the function failed, the return value is zero.

Remark

EX:
```
if (!ulaSDKCapture(lpSDK_TrigSet))
        AfxMessageBox("Capture Error!");
    else
        SetTimer(ID_TM_CAPTURE, 100, NULL);
```

After send the capture command, user have to wait the **ulaSDKIsCaptureReady**() ready before retrieving the data from LA.

BOOL ulaSDKIsTriggerReady()

Check the LA trigger status.

Return value

If the trigger condition is matched, the return value is a nonzero value.

Return zero if the LA is still waiting for trigger event.

BOOL ulaSDKIsCaptureReady()

Check the LA capture status.

Return value

If captured succeessful, the return value is a nonzero value. User can call **ulaSDKGetChData**() or **ulaSDKGetBusData**() to get data from LA.

Return zero if the LA is still capturing data.

BOOL ulaSDKStopCapture()

Send stop capture command to the LA. The LA will keep those data in pre-trigger section.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

After sent stop capture command, user can call **ulaSDKGetChData**() or **ulaSDKGetBusData**() to get the pre-trigger data.

BOOL ulaSDKGetChData(int iCh,UINT* pUserData, int* lpiSize, int iStartSamplePos)

Get one channel data from Logic Analyzer for Conventional Storage data, Transitional Storage data should call ulaSDKGetTrWfm to get data with timestamp.

Parameters

iCh [in]：

Type：int

LA channel.

pUserData [out]：

Type：UINT*

Data buffer. Required buffer size could be read from **ulaSDKGetSamplesNum**().

lpiSize [in, out]：

Type：int*

Input data buffer size; return the size of buffer used to store the data after execute.

iStartSamplePos[in]：

Type：int

The number of the first sample point stored in the data buffer.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

This function will store the data at Bit 0 of the data buffer.

EX: LA captured data are (CH7 - CH0) 0x02、0x00

```
int iSize = ulaSDKGetSamplesNum();
UINT *pUserData = new UINT[iSize];
int iStartSample = 0;     //start read from sample point 0
ulaSDKGetChData (1, pUserData, lpiSize, iStartSample);
```

After execute the function pUserData[0] will be filled with 0x01, and pUserData[1] will be filled with 0x00.

BOOL ulaSDKGetBusData(int iMSB, int iLSB, UINT* pUserData, int* lpiSize, int iStartSamplePos)

Get multiple channels' Data from LA after capture. The maximum channel number is limited to 32. This function is built for Conventional Storage data, Transitional Storage data should call ulaSDKGetTrWfm to get data with timestamp.

Parameter

iMSB [in]：

Type：int

The most significant bit channel of the data read from the LA.

iLSB [in]：

Type：int

The least significant bit channel of the data read from the LA.

pUserData [out]：

Type：UINT*

Data buffer. Required buffer size could be read from **ulaSDKGetSamplesNum**().

lpiSize [in, out]：

Type：int

Data buffer size, return used buffer size after read.

iStartSamplePos[in]：

Type：int

Start read data from specified sample position.

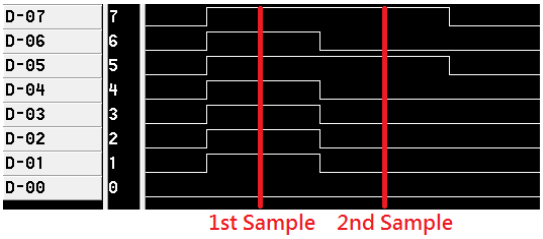
Return value

Return true means successful read from LA. Return false if data reading fail.

Remarks

Example: LA captured two sample points 0xFE, 0xA0 as picture below.

```
int iSize = ulaSDKGetSamplesNum();
UINT *pUserData = new UINT[iSize];
int iStartSample = 0;     //Get data start from sample point 0
ulaSDKGetBusData(7, 0, pUserData, lpiSize, iStartSample);
```



The function will fill pUserData[0] with 0xFE as 1st sample, and pUserData[1] with 0xA0 as 2nd sample.

Bit / Samples	31 - 8	7	6	5	4	3	2	1	0	Data
pUserData[0]	0	1	1	1	1	1	1	1	0	0x000000FE
pUserData[1]	0	1	0	1	0	0	0	0	0	0x000000A0

BOOL ulaSDKGetTransStoreInfo(__int64 & i64TrHead, __int64 & i64TrPos)

Get Transitional Storage captured waveform information, including first timestamp and trigger position.

This function is built for Transitional Storage mode only.

Parameter

i64TrHead [out]：

Type：__int64

The first timestamp captured by logic analyzer.

i64TrPos [out]：

Type：__int64

The timestamp of trigger position captured by logic analyzer.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

I64 ulaSDKGetTrWfm(__int64 i64Index, LPBYTE lpb)

Get Transitional Storage waveform with next timestamp information from specified timestamp.

This function is built for Transitional Storage mode only.

Parameter

i64Index [in]：

Type：__int64

Specified timestamp position for data read.

lpb [out]：

Type：LPBYTE

The data read buffer for current timestamp, the size of this array must be greater than size of __int64, below is the example of channel data layout in this parameter for the first 2 array cells.

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
lpb[0]	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
lpb[1]	CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8

Return value

If the function succeeded, function returns next timestamp. Function will return -1 when there's no more transitional data after current timestamp.

BOOL ulaSDKSaveAsLawFile(char* szFilePathName)

Store current captured waveform as LAViewer compatible .LAW file.

Parameters

szFilePathName[in]：

Type：char *

File pathname.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

The .LAW file contents the last captured data and parameter settings, so this function is only available after successful execute the **ulaSDKCapture**().

BOOL ulaSDKSaveAsLawFileWithTemplate(char* szFilePathName, char* szFileTemplate)

Store current captured waveform as LAViewer compatible .LAW file, and copy the Channel Label settings with Decode settings from template .LAW file or channel setting .csv/.txt file.

Parameters

szFilePathName[in]：

Type：char *

File pathname.

szFileTemplate[in] ：

Type：char *

Template .LAW file pathname, or channel setting .csv/.txt file pathname.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Remark

This function works exactly the same with ulaSDKSaveAsLawFile, but instead of creating default 36 channel labels in the target save file, this function will copy the Channel Label settings from the template file and save to the new .law file.

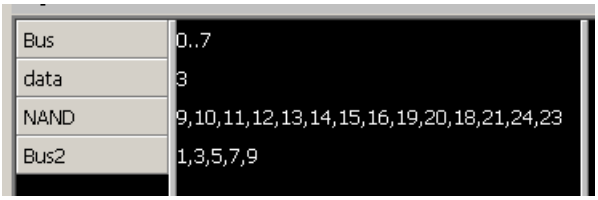
Input template .law file:

Template File	Current SDK data	Output .law file
Waveform	Waveform	Waveform
Waveform capture settings	Waveform capture settings	Waveform capture settings
Channel display settings	Channel display settings	Channel display settings

Input channel setting .csv/.txt file:

```
Bus,0..7
data,3
NAND,"9,10,11,12,13,14,15,16,19,20,18,21,24,23"
Bus2,1,3,5,7,9
```

Result:



BOOL ulaSDKClose ();

Shutdown LA

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKGetLastError()

Get error code from the **LASDK.dll**, the error number meaning and reasons are marked at **LASDK_Err.h**.

Return value

Return error code.

Trigger Setting Save/Load by File

BOOL ulaSDKSaveTriggerByFile(LPSDK_TRIG lpSDKtr, char* szFileName)

BOOL ulaSDKReadTriggerByFile(LPSDK_TRIG lpSDKtr, char* szFileName)

Users can save/load trigger settings to files by calling these functions.

Parameters

lpSDKtr[in/out]：

Type：LPSDK_TRIG

Trigger storage structure.

szFileName[in]：

Type：char*

File full pathname.

EX：D:\LA_SDK\TriggerSet.aqr

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

BOOL ulaSDKSetHWByPrj(char* szFileName, LPSDK_TRIG lpSDKtr, int iUserMaxMemSize);

Load Hardware and trigger settings from the LAViewer project file.

Parameters

szFileName[in]：

Type：char*

File full pathname

EX：C:\Users\UserName\Documents\Acute\LA\Project\Project1.aqr

lpSDKtr[in/out]：

Type：LPSDK_TRIG

Trigger storage structure.

iUserMaxMemSize[in]：

Type：int

The sample number will be captured by LA.

Return value

If the function succeeded, the return value is a nonzero value.

If the function failed, the return value is zero.

Acute Technology Inc.

www.acute.com.tw

Address：6F-7, #12, Ln. 609, Sec. 5, Chongxin Rd., Sanchong Dist., New Taipei City 24159, Taiwan

Tel：+886-2-2999-3275

Fax：+886-2-2999-3276

E-mail: service@acute.com.tw