

LARUN.DLL Manual

(Version 2.3.0.8)

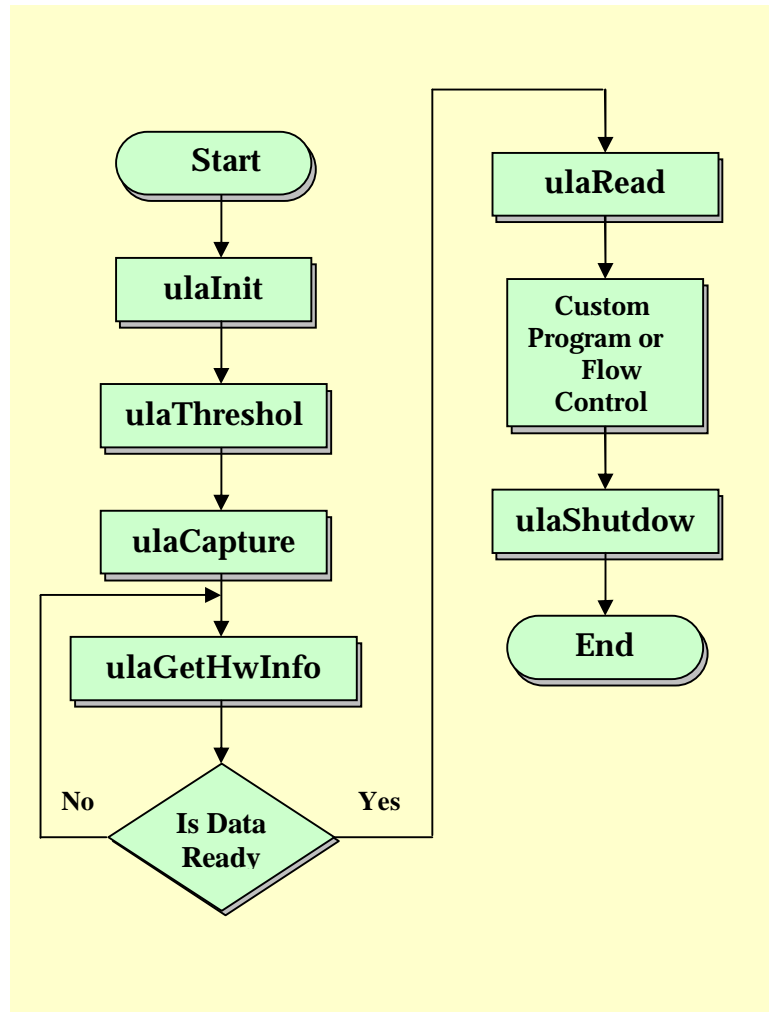
: *What is the LARUN.DLL ?*

The LARUN.DLL is a dynamic-link library for Windows OS. It provides several function calls to control the Acute Logic Analyzer (LA1000P/LA2000P Series and PKLA1000 Series). You may use some language that support DLL link function, such as Visual C or Visual Basic, to control Logic Analyzer (LA) with LARUN.DLL library. Here, we illustrate some examples using Visual C and Visual Basic. The other languages please refer to their description about DLL link application.

LARUN.DLL includes following function calls:

```
BOOL ulaInit();  
BOOL ulaShutdown();  
BOOL ulaThreshold( int iPod, int iMilliVolt );  
INT ulaGetHwInfo( UINT nIndex );  
BOOL ulaSetHwInfo( UINT nIndex, int iValue );  
BOOL ulaGetModelName( LPSTR szModel, int iSize );  
BOOL ulaCapture( LPTRIG lptr, int *lpi, LPBYTE lpb );  
BOOL ulaRead( LPBYTE lpb, int iFlag );
```

: Flow Chart



: Function Call Reference

The LARUN.DLL function call reference is in C language. And this reference is valid for version 2.3.0.8 or later.

ulaInit

Initialize the logic analyzer

```
BOOL ulaInit();
```

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 call the **ulaInit** to initialize the LA before using any other function calls of LARUN.DLL. The **ulaInit** is not only to initialize LA, but it will check the LA hardware. You may call the **ulaInit** just once.

ulaShutdown

Turn off the logic analyzer.

```
BOOL ulaShutdown();
```

Return values

If the LA turns off in normal procedure, this function will return nonzero value (TRUE), else return zero (FALSE).

Remarks

Before you closing your application, please call this function to release the connection of LA and PC. Thus, it will not affect anything to enter LA in next time. Otherwise, it maybe causes the LA unstable. And, only reboot your PC can solve this problem.

ulaThreshold

To set logic threshold voltage of LA.

```
BOOL ulaThreshold(  
    int iPod,           // Pod Number  
    int iMilliVolt      // Voltage of Pod  
);
```

Parameters

iPod

Pod number.

iMilliVolt

Threshold voltage. (Unit: mV)

LA will pick a nearby value as no your value.

Return values

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

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

ulaCapture

Start capturing signals data.

```
BOOL ulaCapture( LPTRIG lptr, int *lpi,
                LPBYTE lpb );
```

Parameters

lptr

A pointer of Trigger Structure define as following:

```
typedef struct _TRIG
{
    int iFlag;           //Trigger Flag
    int iCh;             //Fill 64
    int iLvl;            //Trigger level
    int iDelay;          //Delay time of
                        //Delay Trigger

    int iWidth;          //Trigger width
    int iPassCount;      //Trigger pass time
    int iFreq;           //Sampling rate
    int iExtend;         //Fill 1
    int iTrMode;         //Trigger mode
    int iExtClk;         //External clock
    int iTrPos;          //Trigger position
    int iFreqHi;         //Combine iFreq & iFreqHi
                        //into 64bits Frequency
```

```
int iRes[2];           //Must fill 0
int *lpiCont;          //Fill NULL
BYTE *lpbTrigData;     //Fill NULL

}TRIG, FAR *LPTRIG;

iFlag:
```

```
#define TR_DBLMODE      0x0001
#define TR_PRETRIG      0x0002
#define TR_XCHG         0x0004
```

TR_DBLMODE is for double capturing mode. (Refer to the Capture mode of LA manual.); TR_PRETRIG is Pre-Trigger function switch (only for LA2000P series); TR_XCHG is Dual-Condition trigger mode switch (only for LA2000P series).

iTrMode:

```
#define TR_ONELEVEL      0
#define TR_MULTILEVEL    1
#define TR_DUALCOND      2
#define TR_WIDTHMODE     3
#define TR_NOTRIG        4
```

TR_ONELEVEL is single trigger level mode; TR_MULTILEVEL is multi-trigger level mode; TR_DUALCOND is dual condition trigger mode; TR_WIDTH is trigger width condition mode; TR_NOTRIG is all trigger setting disabled. (only one trigger level setting in non-LA2000P series)

iExtClk:

```

Bit0: using Internal sampling clock
Bit1: using External sampling clock
Bit2: External clock path in CH15
Bit3: External clock path in CH31
Bit4: External clock path in CH63
Bit5: 0=> Internal AND External
      1 => Internal OR External
Bit6: 0 => Sampling at External-clock rising edge
      1 => Sampling at External- clock falling
      edge
Bit7: Use External Clock (TravelLogic series only),
      ExternalClock Channel = Channel 35

Bit8~Bit15: External Clock Mode (Include !Channel
           35, Channel134 & Channel135, Channel134 #
           Channel135 mode, TravelLogic series only)

```

lpi

is a index to point 16 arrays of integer. Each data presents their relationship of between trigger levels. Ex. *(lpi+0) means the relationship of 1st trigger level and 2nd trigger level; *(lpi+3) means the 4th trigger level and 5th trigger level relationship. While there value filled as TR_CONTINUE, it means this two trigger levels belong to continual type: Trigger success only at the two trigger levels condition appeared in two sequential sampling. (Please refer to the principle of LA section in LA2000P manual.);While there value filled as TR_DISCONTINUE, it means this two trigger

levels belong to discontinuous type. If there value filled as TR_END, it means this is the last trigger level. As the LA inspected TR_END, it will start capturing.

```

#define TR_CONTINUE      0
#define TR_DISCONTINUE  1
#define TR_END           2

```

lpb

is a index to point trigger data buffer. This buffer size is 1024(64x16) bytes. There are 16 trigger levels in LA2000P series. Each level has 64 trigger channels. So, it needs amount of 1024 bytes to store trigger data. Trigger data define as following:

```

#define TRIG_LOW         0
#define TRIG_HIGH        6
#define TRIG_DONTCARE    8

```

The other value is illegal to trigger data.

Return values

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

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

ulaGetHwInfo

Get LA information and check now status.

```
INT ulaGetHwInfo(UINT nIndex);
```

Parameters

nIndex:

- 0: get LA ID code.
- 1: check LA capturing status. (True or False means finish or not)
- 2: get the LA capturing memory size (Bytes) for creating a buffer. This value is not equal to the LA memory depth, but to reserve a space for placing captured data. (See [ulaRead](#))

Return values

Return values depend on different parameters.

ulaSetHwInfo

Set LA H/W parameter.

```
BOOL ulaSetHwInfo(UINT nIndex, int iValue);
```

Parameters

nIndex:

- 2: Set the LA capturing memory size (Bytes) for creating a buffer. This value is not equal to the LA memory depth, but to reserve a space for placing captured data. (See [ulaRead](#))
- 5: Set Travel Logic Mode. (See below table)

H/W Mode	Frequency	Channels	P1	P2	P3
0	4 GHz	36	64	36	36
1	4 GHz	18	64	18	18
2	4 GHz	9	64	9	9
3	4 GHz	4	64	4	4
100	2 GHz	36	64	36	36
200	1.6 GHz	4	64	32	4
300	800 MHz	9	64	36	9
400	400 MHz	18	64	36	18
500	<= 200MHz	36	64	36	36
501	<= 200MHz	18	64	36	18
502	<= 200MHz	12	64	36	12
503	<= 200MHz	9	64	36	9
504	<= 200MHz	6	64	36	6
505	<= 200MHz	4	64	36	4
506	<= 200MHz	2	64	36	2
507	<= 200MHz	1	64	36	1

600	ExternalClock	36	64	36	36
601	ExternalClock	18	64	36	18
602	ExternalClock	12	64	36	12
603	ExternalClock	9	64	36	9
604	ExternalClock	6	64	36	6
605	ExternalClock	4	64	36	4
606	ExternalClock	2	64	36	2
607	ExternalClock	1	64	36	1

iValue:

The iValue is H/W Mode. In this DLL version, you need a LaStream class. (LaStream.cpp & LaStream.h) When initial the class need P1, P2 and P3 parameter.(See above table)

P1: a sample clock data width: 64bits.

P2: data width available.

P3: data width used actually.

Return values

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

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

ulaGetModelName

Get LA model name.

```
BOOL ulaGetModelName(LPSTR szModel, int iSize);
```

Parameters

szModel: the buffer for saving LA model name.

iSize: size of the buffer to save LA model name.

Return values

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

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

ulaRead

Read back the LA captured data.

```
BOOL ulaRead(LPBYTE lpb, int iFlag);
```

Parameters

lpb: point a buffer, which is to store captured data. The buffer size may get by [ulaGetHwInfo](#).

iFlag: set as -1

Return values

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

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

Remarks

If we use [ulaGetHwInfo\(1\)](#) to check capturing status and get FALSE back, the [ulaRead](#) can only get parts of captured data which is prior of trigger position.

: *Memo*