Interface Manual
USBLC3301x32.DLL
USBLC3301x64.DLL
(V1.00)

Coptonix GmbH
Falkentaler Steig 9
D – 13467 Berlin
Phone: +49 – (0)30 – 61 74 12 48
Fax: +49 – (0)30 – 61 74 12 47
www.coptonix.com
support@coptonix.com

Contents

Dynamic Link Library DLL....3

1 Functions....3
1.1 USB Functions....3
1.1.1 ls_GetErrorString....3
1.1.2 ls_Initialize....3
1.1.3 ls_SetPacketLength....3
1.1.4 ls_GetPacketLength....3
1.1.5 ls_EnumDevices....3
1.1.6 ls_OpenDeviceByIndex....4
1.1.7 ls_OpenDeviceBySerial....4
1.1.8 ls_CloseDevice....4
1.1.9 ls_DeviceCount....4
1.1.10 ls_CurrentDeviceIndex....4
1.1.11 ls_GetFWVersion....4
1.1.12 ls_GetVendorName....4
1.1.13 ls_GetProductName....4
1.1.14 ls_GetSerialNumber....4
1.1.15 ls_SetEPTimeOut....4
1.1.16 ls_GetEPTimeOut....5
1.2 Data Functions....5
1.2.1 ls_WaitForPipe....5
1.2.2 ls_GetPipe....5
1.2.3 ls_GetFPS....5
1.3 Camera Functions....5
1.3.1 ls_GetPixelCount....5
1.3.2 ls_GetSensorName....5
1.3.3 ls_SetMode....5
1.3.4 ls_SetState....5
1.3.5 ls_SetIntTime....6
1.3.6 ls_SetCFG1....6
1.3.7 ls_GetMode....6
1.3.8 ls_GetState....6
1.3.9 ls_GetIntTime....6
1.3.10 ls_GetCFG1....6
1.3.11 ls_SetGain....7
1.3.12 ls_GetGain....7
1.3.13 ls_SetOffSet....7
1.3.14 ls_GetOffSet....7
1.3.15 ls_SaveSettings....8

Dynamic Link Library DLL

1 Functions

1.1 USB Functions

1.1.1 ls_GetErrorString

function ls_geterrorstring(dwErr : DWORD) : PAnsiChar;

ls_GetErrorString converts the error code *dwErr* to a readable zero terminated string. If not specified, *dwErr* is the return value of most functions below.

1.1.2 ls_Initialize

function ls_initialize(dwPipeSize, dwPacketLength, dwThreadClass : DWORD;
iThreadPrio : Integer; pcMsgID : PAnsiChar): DWORD;

When starting the application, this function is called when the default values are not sufficient. The argument *dwPipeSize* defines the size of a ring buffer (pipe). If *dwPipeSize* is equal to 512, it means 512 bytes buffer and 67108864 is equal to 64 Mbytes. The default value is 4MB. *dwPacketLength* is the number of bytes to be read per read request from the hardware FIFO. The value of *dwPacketlength* must be equal to or multiple of number of pixels.
Use the function *ls_getpacketlength* (1.1.4) to read the number of bytes the device transfers per USB transaction, and set *dwPacketLength* to multiple of this value. The argument *dwThreadClass* and *iThreadPrio* gives the possibility to adapt the priority of the reading thread. *dwThreadClass* is the thread class and the default value is NORMAL_PRIORITY_CLASS. *iThreadPrio* is the priority of the thread. It's default value is THREAD_PRIORITY_NORMAL.

ls_Initialize defines a new window message that is guaranteed to be unique throughout the system. The argument ***pcMsgID*** (as PAnsiChar e.g. "My_USBLS_App" should be unique). If more than one application use the same ***pcMsgID***, they will share the same window message ID. If the function successes, it returns a message identifier in the range 0xC000 through 0xFFFF. If the function fails, the return value is zero. The returned value must be saved in a global valid variable in order to use it later in processing messages. For the registration of the new window message the window API function "RegisterWindowMessage" is used.

1.1.3 ls_SetPacketLength

function ls_setpacketlength(dwPacketLength : DWORD) : DWORD;

ls_SetPacketLength sets the value of *dwPacketLength*. *dwPacketLength* is described in section 1.1.2. Before calling this function, the device must be closed. If the function fails, the return value (*dwErr*) is non zero.

1.1.4 ls_GetPacketLength

function ls_getpacketlength(var dwPacketLength : DWORD) : DWORD;

ls_GetPacketLength reads the recommended value for *dwPacketLength* from the line sensor controller. *dwPacketLength* is described in section 1.1.2. If the function fails, the return value (*dwErr*) is non zero.

1.1.5 ls_EnumDevices

function ls_enumdevices : Integer;

ls_EnumDevices enumerates and creates a list of all connected devices and then returns the number of connected devices.

1.1.6 ls_OpenDeviceByIndex

function ls_opendevicebyindex(index : Integer) : DWORD;

ls_OpenDeviceByIndex connects a USB device and starts the reading thread. The argument *index* is 0-based. This means that the first device was connected has the index zero (0), the second one has the index 1, and so on. If the function fails, the return value (*dwErr*) is non zero.

1.1.7 ls_OpenDeviceBySerial

function ls_opendevicebyserial(pcserialnum : PAnsiChar) : DWORD;

ls_OpenDeviceBySerial connect a USB device and starts reading thread (see ls_OpenDeviceByIndex). The argument ***pcserialnum*** is the serial number (e.g. 160000) of a device. If the function fails, the return value (*dwErr*) is non zero.

1.1.8 ls_CloseDevice

function ls_closedevice : DWORD;

"ls_CloseDevice" disconnects the current opened device. If the function fails, the return value (*dwErr*) is non zero.

1.1.9 ls_DeviceCount

function ls_devicecount : Byte;

ls_DeviceCount returns the number of USB devices, which are currently connected to the system.

1.1.10 ls_CurrentDeviceIndex

function ls_currentdeviceindex : Integer;

ls_CurrentDeviceIndex returns the index of the opened USB device. The return value is -1 if no USB device is opened.

1.1.11 ls_GetFWVersion

function ls_getfwversion(index : Integer) : WORD;

ls_GetFWVersion returns the version of the firmware with index "***index***".

1.1.12 ls_GetVendorName

function ls_getvendorname(index : Integer) : PAnsiChar;

ls_GetVendorName returns the vendor's name of the device with index "***index***".

1.1.13 ls_GetProductName

function ls_getproductname(index : Integer) : PAnsiChar;

ls_GetProductName returns the product's name of the device with index "***index***".

1.1.14 ls_GetSerialNumber

function ls_getserialnumber(index : Integer) : PAnsiChar;

ls_GetSerialNumber the serial number of the device with index "***index***".

1.1.15 ls_SetEPTimeOut

function ls_seteptimeout(dwtimeout : DWORD) : DWORD;

ls_SetEPTimeOut sets the timeout value of the USB IN End Point. The time-out value will be expected in 1 ms units. A value of 1000 corresponds to 1 s. Before calling this function, the device must be closed. If the function fails, the return value (*dwErr*) is non zero.

1.1.16 ls_GetEPTimeOut

function ls_geteptimeout : DWORD;

ls_GetEPTimeOut reads the current timeout value of the USB IN End Point. The time-out value will be expected in 1 ms units. A value of 1000 corresponds to 1 s. If the function fails, the return value (*dwErr*) is non zero.

1.2 Data Functions

1.2.1 ls_WaitForPipe

function ls_waitforpipe(dwTimeOut : DWORD) : DWORD;

ls_WaitForPipe checks whether the pipe contains data for reading. If no data are available, the calling thread enters the wait state until data is received or the time-out interval elapses. *dwTimeOut* is the time out interval. The time-out value will be expected in 1 ms units. A value of 1000 corresponds to 1 s. If the function fails, the return value (*dwErr*) is non zero.

1.2.2 ls_GetPipe

function ls_getpipe(lpBuffer : Pointer; dwToRead: DWORD;
var dwRead: DWORD): DWORD;

ls_GetPipe reads data from the pipe (ring buffer). The argument *lpBuffer* points to the buffer, which has to include the data. *dwToRead* specifies the length of the data which must be read, and *dwRead* returns the actual number of bytes read. If *dwToRead* is specified with 0, then *dwRead* returns actual number of bytes available without reading data. If the function fails, the return value (*dwErr*) is non zero.

1.2.3 ls_GetFPS

function ls_getfps : DWORD;

ls_GetFPS reads the number of bytes transferred per second. To determine the speed (number of frames per second) the return value must be divided by the number of pixels.

1.3 Camera Functions

1.3.1 ls_GetPixelCount

function ls_getpixelcount : WORD;

ls_GetPixelCount returns the number of pixels of the sensor (102 pixels).

1.3.2 ls_GetSensorName

function ls_getsensorname : PAnsiChar;

ls_GetSensorName returns the name of the sensor “TSL3301”.

1.3.3 ls_SetMode

function ls_setmode(ucMode : Byte) : DWORD;

There are 3 operation modes available. The value for *ucMode* must be

ONE_SHOT	0x00	Acquisition is software triggered.
EXT_TRIGGER	0x01	Acquisition is done on external trigger.
FREE_RUNNING	0x02	Acquisition is done continuously.

If the function fails, the return value (*dwErr*) is non zero.

1.3.4 ls_SetState

function ls_setstate(ucState : Byte) : DWORD;

ls_SetState starts or stops data acquisition. If value passed to *ucState* is 0x01, acquisition starts. If value passed for *ucState* is 0x00, acquisition stops. If the function fails, the return value (*dwErr*) is non zero.

1.3.5 ls_SetIntTime

function ls_setinttime(dwIntTime : DWORD) : DWORD;

ls_SetIntTime sets the integration/exposure time *dwIntTime* in microseconds. If the function fails, the return value (*dwErr*) is non zero.

1.3.6 ls_SetCFG1

function ls_setcfg1(ucCFG1 : Byte) : DWORD;

ls_SetCFG1 sets the configuration register 1.
Note: Changes take effect after power off/on.

Bit number	Value	Description
Bit[0:3]		Number of images to be buffered before transferring to the host. This value deteremines the value of *dwPacketLength* used in functions *ls_initialize* and *ls_setpacketlength*.
	0000 = 0	1 image / USB transfer.
	0001 = 1	2 images / USB transfer
	….	
	1110 = 14	15 images / USB transfer
	1111 = 15	16 images / USB transfer
Bit[4:7]		Not used. Do not care.

If the function fails, the return value (*dwErr*) is non zero.

1.3.7 ls_GetMode

function ls_getmode(Var ucMode : Byte) : DWORD;

ls_GetMode returns the current mode "*ucMode*":

ONE_SHOT	0x00	Acquisition is software triggered.
EXT_TRIGGER	0x01	Acquisition is done on external trigger.
FREE_RUNNING	0x02	Acquisition is done continuously.

If the function fails, the return value (*dwErr*) is non zero.

1.3.8 ls_GetState

function ls_getstate(Var ucState : Byte) : DWORD;

ls_GetState returns the current state "*ucState*":

- 0x00 Acquisition is stopped
- 0x01 Acquisition is running

If the function fails, the return value (*dwErr*) is non zero.

1.3.9 ls_GetIntTime

function ls_getinttime(Var dwIntTime : DWORD) : DWORD;

ls_GetIntTime returns the integration/exposure time "*dwIntTime*" in microseconds. If the function fails, the return value (*dwErr*) is non zero.

1.3.10 ls_GetCFG1

function ls_getcfg1(var ucCFG1 : Byte) : DWORD;

ls_GetCFG1 reads the configuration register 1. For further information please refer to section 1.3.6. If the function fails, the return value (*dwErr*) is non zero.

1.3.11 ls_SetGain

function ls_setgain(ucleft, uccenter, ucright : Byte) : DWORD;

The sensor is divided into three 34-pixel zones (*ucleft, uccenter, ucright*), with each zone having programmable gain correction. The table below lists the gain settings and the corresponding pixel values. If the function fails, the return value (*dwErr*) is non zero.

Gain Code	Rel. Gain	% Increase	Gain Code	Rel. Gain	% Increase
0	1		**16**	1.52	3.23
1	1.02	2.17	**17**	1.57	3.33
2	1.05	2.22	**18**	1.62	3.45
3	1.07	2.27	**19**	1.68	3.57
4	1.09	2.33	**20**	1.74	3.70
5	1.12	2.38	**21**	1.81	3.85
6	1.15	2.44	**22**	1.88	4.00
7	1.18	2.50	**23**	1.96	4.17
8	1.21	2.56	**24**	2.05	4.35
9	1.24	2.63	**25**	2.14	4.55
10	1.27	2.70	**26**	2.24	4.76
11	1.31	2.78	**27**	2.35	5.00
12	1.34	2.86	**28**	2.48	5.26
13	1.38	2.94	**29**	2.61	5.56
14	1.43	3.03	**30**	2.77	5.88
15	1.47	3.13	**31**	2.94	6.25

1.3.12 ls_GetGain

function ls_getgain(var ucleft, uccenter, ucright : Byte) : DWORD;

ls_GetGain reads the values of all 3 gain registers of the sensor. For further information please refer to section 1.3.11. If the function fails, the return value (*dwErr*) is non zero.

1.3.13 ls_SetOffSet

function ls_setoffset(ucleft, uccenter, ucright : Byte) : DWORD;

The sensor is divided into three 34-pixel zones (*ucleft, uccenter, ucright*), with each zone having programmable gain correction. Codes 0h to 7Fh corresponds to positive offset values and codes 80h to FFh corresponds to negative offset values. Offset is affected by the gain settings and may have to be adjusted after gain changes are made. The offset correction is proportional to the gain setting. At minimal gain, one LSB of the offset corresponds to approximately 1/3 LSB of the device output, and at maximum gain, to about 1 LSB of the device output.

If the function fails, the return value (*dwErr*) is non zero.

1.3.14 ls_GetOffSet

function ls_getoffset(var ucleft, uccenter, ucright : Byte) : DWORD;

ls_GetOffSet reads the values of the Offset registers (*ucleft, uccenter, ucright*). For further information please refer to section 1.3.13. If the function fails, the return value (*dwErr*) is non zero.

1.3.15 ls_SaveSettings

function ls_savesettings : DWORD;

ls_SaveSettings saves all parameters / settings into EEPROM. If the function fails, the return value (*dwErr*) is non zero.