OWON Oscilloscope PC Guidance Manual 1.3

Communication Protocols of USB and Introduction of data format, open source for the third party

WWW.OWON.COM.CN

1. USB communications interface

Introduction

OWON instruments, also the digital oscilloscope, mixed LA-oscilloscope with the function of logic analyzer, support using USB communication protocols to transmit data with PC, which version is 1.1. The followings are referred to OWON products without special remarks.

Device information

The followings are the printed parameter information when connecting OWON instruments with PC and installing driver, in which some parameter information (the parameters for USB communication are similar in different products) could be used in USB communication, or acquired by the USB driver of the third party:

Usb_Device_Descriptor

blenght: 0x12
bDescriptorType: 0x1
bcdUSB: 0x110
bDeviceClass: 0x0
bDeviceSubClass: 0x0
bDeviceProtocol: 0x0
bMaxPacketSize0: 0x8 (8)
idVendor: 0x5345
idProduct: 0x1234
bcdDevice: 0x0
iManufacturer: 0x1
iProduct: 0x2
iSerialNumber: 0x0
bNumConfigurations: 0x1

String descriptors

iManufacturer: System MCU
iProduct: LILLIPUT S3C2410A SPQ SYSTEM

Usb_Config_Descriptor

blenght: 0x9
bDescriptorType: 0x2
bNumInterfaces: 0x1
bConfigurationValue: 0x1
iConfiguration: 0x0

bmAttributes: 0xc0

MaxPower [mA]: 0x19 (25)
extralen: 0x0
extra: null

Usb_Interface_Descriptor

blenght: 0x9
bDescriptorType: 0x4
bInterfaceNumber: 0x0
bAlternateSetting: 0x0
bNumEndpoints: 0x2
bInterfaceClass: 0x5
bInterfaceSubClass: 0x6
bInterfaceProtocol: 0x50
iInterface: 0x0
extralen: 0x0
extra: null

Usb_Endpoint_Descriptor

blenght: 0x7
bDescriptorType: 0x5
bEndpointAddress: 0x81
bmAttributes: 0x2
wMaxPacketSize: 0x40 (64)
bInterval: 0x0
bRefresh: 0x0
bSynchAddress: 0x0
extralen: 0x0
extra: null

Usb_Endpoint_Descriptor

blenght: 0x7
bDescriptorType: 0x5
bEndpointAddress: 0x3
bmAttributes: 0x2
wMaxPacketSize: 0x40 (64)
bInterval: 0x0
bRefresh: 0x0
bSynchAddress: 0x0
extralen: 0x0
extra: null

The parameters may be used in communication such as the following enumerating (based on instruments)

idVendor: 0x5345
idProduct: 0x1234
iSerialNumber: 0x0

WriteEndpoint: 0x03
ReadEndpoint: 0x81

bConfigurationValue: 0x1
bInterfaceNumber: 0x0
bAlternateSetting: 0x0(without setting in operating)

1.3: Data acquisition protocols
Predefined variable:
START_COMMAND = "START"
For SDS Serials START_COMMAND can be "STARTBIN", "STARTBMP", "STARTMEMDEPTH", for get vector data file, bitmap file, or deep memory vector data file.
For deep memory Oscilloscopes START_COMMAND can be "STARTBIN", "STARTBMP", for get vector data file, bitmap file.
RESPONSE_START_LENGTH = 12
BMP_FILE_FLAG = 1
BMP_FILE_EXTENTION = "bmp"
BIN_FILE_EXTENTION = "bin"

The process is as follows :(The steps besides [] are the standard defined by communication protocols, while only the steps in [] was customized in OWON USB data acquisition protocols.)
)
Initialize USB
Searching for all Buses
Searching for all USB

Enumerate instruments and find out the instrument matching with **idVendor** and **idProduct** (if there are several instruments appeared, it could be differentiated by ordinal descriptor obtained through index value of **iSerialNumber**.)

Use the instrument to communicate with USB
Set configuration information by **bConfigurationValue**
Register communication interface by **bInterfaceNumber**
Set other information by **bAlternateSetting**, while OWON products is no need to do so.)

[
(Contents obtained from OWON USB data acquisition protocols are as follows, in which the communication could use interrupt transfer or bulk transfers.)
Starting command **START_COMMAND , ASCII** code
Little Endian encode format is used on all 4 bytes int data.
The instrument will be back to the data of byte **RESPONSE_START_LENGTH,** the first int indicates length of data file, the second int for no use, and the third int use as a flag, when flag == 0, then it is vector data file(.bin), when flag == 1,then it is bitmap file(.bmp), then the length of data will be sent and communication is over.

WHEN flag >=128, it means the data is a deep memory vector data file, and because of the huge data, transmission will be separated to several parts. The (flag - 128) is the number of channel, after the **RESPONSE_START_LENGTH** bytes of data received**,** the length of data will be sent; and after that, if the (flag-128) is not yet 0, **RESPONSE_START_LENGTH** will be sent back again and the format is similar except flag decreases 1 which indicates the another channel, and until (flag -128)==0, the communication is over.
]

Release the communication interface
Close the communication with USB.

Beside, for some types of oscilloscopes, there is a LAN port for transmission. It uses the same protocol and data file format as USB communications interface.

2. RS232 communication interface
2.1 Introduction
OWON instruments, also the digital oscilloscope, mixed LA-oscilloscope with the function of logic analyzer, support using RS232 communication protocols to transmit data with PC. The followings are referred to OWON products without special remarks.

2.2 Device information
RS232 communication protocols use **YModem** with transmission parameter as (baud rate:115200, data bit:8, even-odd check: nothing, stop bit:1).

2.3 data acquisition protocols:
Send out a command to the instrument to transmit using **C** of **ASCII** and get the data packet in accordance with **RS232 YModem**, then analyze the data packet by responding, the serial number of data packet and **CRC** check value which of the three are not the information format of **YModem** protocols: the first data packet instruct the information of the transmitted data including that the string of **ASCII** which indicates file name (with suffix indicating file type), and then the 0x00 with 1 byte, the following an **int** coding by **Little-Endian** about file size, the last ending with 0x00. The persistent

data input get from the second data packet is the data file.

RS232 communication only supports "START" Command, and get vector data file or bitmap file controlled by Oscilloscope.

3. Oscilloscope data use

3.1 Pre-introduction

The oscilloscope displays waveform and operates according to itself collecting rules and drawing process. Some principles help to correctly analyze the data file obtained from communication and achieve the function of upper computer. The following introductions are about the development of upper computer.

3.1.1 Model

OWON products include two types: Bench Digital Storage Oscilloscope—PDS Series and Handheld Digital Storage Oscilloscope--- HDS Series. In the data files, the head of string of the files (such as **"SPBxxx"**) correspond to the machine model. The relationship is as follow:

SPBW01 correspond to PDS6062S, PDS6062T, PDS7102T
SPBW11 correspond to HDS2062M
SPBW10 correspond to HDS2062M-N

SPBV01 correspond to PDS5022S, MSO5022
SPBV10 correspond to HDS1022M-N
SPBV11 correspond to HDS1022M
SPBV12 correspond to HDS1021M

SPBX01 correspond to MSO7102, PDS8102T
SPBX10 correspond to HDS3102M-N

SPBM01 correspond to MSO8202, PDS8202T

SPBS01 correspond to SDS6062
SPBS02 correspond to SDS7102
SPBS03 correspond to SDS8202
SPBS04 correspond to SDS9302 (SDS8202 and SDS9302 may change,because they are not produced yet)

In which the head characters function marking.(start from 0)
The [3] character indicates sample rate, which V indicates 20M and W indicates 60M.
The [4] character indicates the type of machine, which O indicates PDS series and 1 indicates HDS series.
The [3] character with the character before indicates the machine model in accordance with different detail of data communication protocols.

P.S: **SPBbin** only appears in earlier products and correspond to all models, but it

phased out already because unable to defined the data files analyticity. The [2] character sometimes used to indicate.

3.1.2: Screen drawing:
The screen drawing in the official software is as follow:

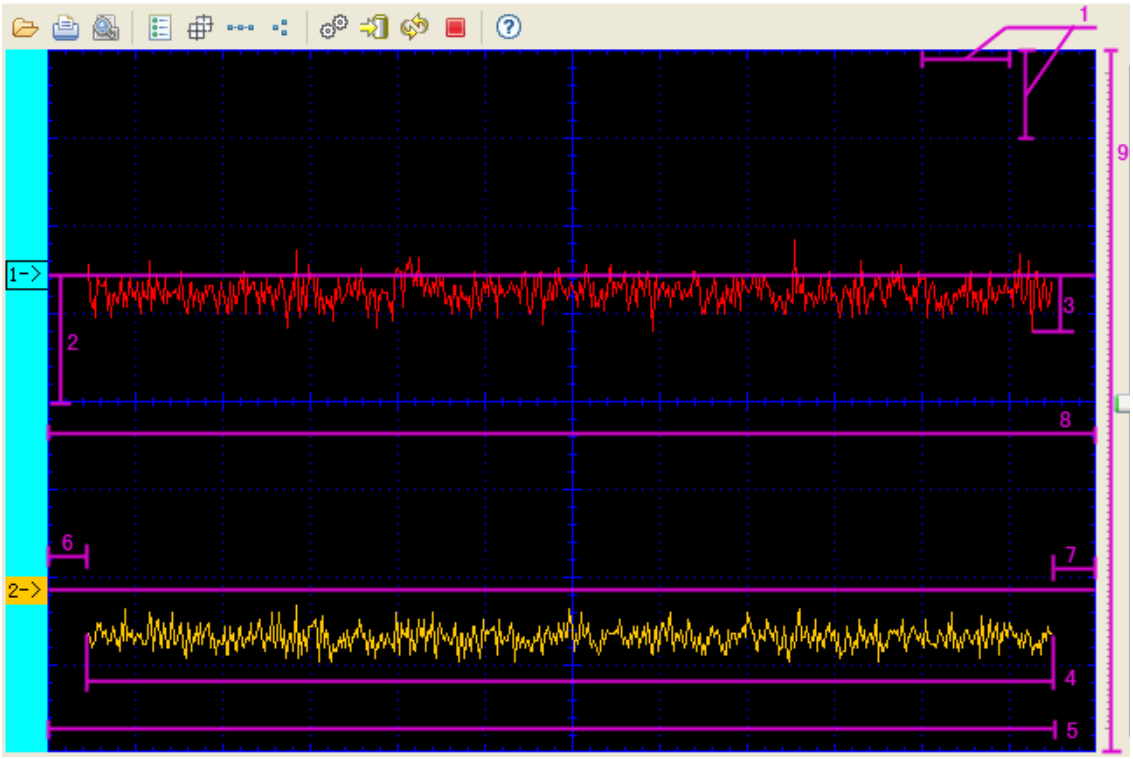

Fig.1-1 the drawing of upload data of OWON HDS series

Explain the concepts shown as 1-1:

Describable point:
8 and 9 shown in 1-1 is the range of full screen (the pixel value of PDS series is 500*400, which of HDS series is 300*200) in the horizontal-vertical axis on the screen, among which (only visibility region) the most drawing points is called for Describable point. The concept of the describable point is only the base unit different from that of actual collection dot. The number of the describable points of oscilloscope is a fixed value, PDS series 250*200 points, while 300*200 points of HDS series.

Block: There is the range of describable points in the X axis and the Y axis by scale and the width of every 25 describable points as a block as shown in 1-1. Take the Y axis for example, the screen is divided into blocks of 200/25 = 4*2. In the similar way, PDS series are divided into some blocks of 250/25 = 5*2 and HDS series are of 300/25 = 6*2 in the X axis.

Zero position: A waveform channel should take a position in the Y axis as the value of

zero voltage named as zero position, which value is the zero position in the Y axis by the horizontal-half axis, upwards is positive, downwards is negative, and unit is the number of describable points as shown in the 2 of 1-1. For the zero position of CH“1” in the 36th position up the horizontal-half axis, the value of its zero position is 36.

Data of Collection dot: Shown as 3 in the 1-1, the zero position of the waveform is as the zero position in the Y axis, upwards is positive, downwards is negative, every collection dot could be gridded north as value, that is to obtain the data of every collection dot from data files. For example, the value of a collection dot is -20, it means that the point is in the interval 20 describable points below the zero position of waveform. Data files also supply” the interval value of voltage of describable points, which multiply with the value of the collection dot to get the value of voltage.

Note: The instrument display the above drawing value with the number of describable point as unit. In the development of software of upper computer, it could customize the size of the drawing range by set up the actual width of the pixel. The describable points in the Y axis of the two product series are different. To appoint the width of the axis of the different describable point by making the width of the drawing range as the lowest common multiple can simplify the branch of program.

3.1.3 Numerical value setting

The oscilloscope has the voltage level and time base level, in which one block corresponds to the magnitude of voltage and time in Y axis and X axis separately. There are index of level value in the data file, so the corresponded value could be indexed by the two numbers, such as that 5 is the supplied voltage level index number, the voltage level is **chVv[5]=0.1V.** If the supplied time base level index number is 10, the time base level is **chHs[10]=0.01mS**

The group of the voltage level (Unit:V)

Double chVv[21];
chVv[0] = 0.002;
chVv[1] = 0.005;
chVv[2] = 0.01;
chVv[3] = 0.02;
chVv[4] = 0.05;
chVv[5] = 0.1;
chVv[6] = 0.2;
chVv[7] = 0.5;
chVv[8] = 1d;
chVv[9] = 2d;
chVv[10] = 5d;
chVv[11] = 10d;
chVv[12] = 20d;
chVv[13] = 50d;

chVv[14] = 100d;
chVv[15] = 200d;
chVv[16] = 500d;
chVv[17] = 1000d;
chVv[18] = 2000d;
chVv[19] = 5000d;
chVv[20] = 10000d;

The group of time base level: (unit:ms)

double chHs[34];
chHs[-2] = 0.000001;SPBMxx , SPBS03,SPBS04 start chHs[0] from here
chHs[-1] = 0.000002;SPCX01,SPBNxx start chHs[0] from here
chHs[0] = 0.000005; SPBVxx, SPBWxx, SPBXxx,SPBVxx ,SPBS01,SPBS02 start chHs[0] from here
chHs[1] = 0.00001;
chHs[2] = 0.000025;
chHs[3] = 0.00005;
chHs[4] = 0.0001;
chHs[5] = 0.00025;
chHs[6] = 0.0005;
chHs[7] = 0.001;
chHs[8] = 0.0025;
chHs[9] = 0.005;
chHs[10] = 0.01;
chHs[11] = 0.025;
chHs[12] = 0.05;
chHs[13] = 0.1;
chHs[14] = 0.25;
chHs[15] = 0.5;
chHs[16] = 1;
chHs[17] = 2.5;
chHs[18] = 5;
chHs[19] = 10;
chHs[20] = 25;
chHs[21] = 50;
chHs[22] = 100;
chHs[23] = 250;
chHs[24] = 500;
chHs[25] = 1000;
chHs[26] = 2500;
chHs[27] = 5000;
chHs[28] = 10000;
chHs[29] = 25000;

chHs[30] = 50000;
chHs[31] = 100000;

Notice: some MachineType such as MSO8202, timebases extended to 1ns as the bandwith expanded to 200M/s, the first element of array for each MachineType has been indicated up.

The group of time base level should be adjusted according to the type of machine model when using. The next to third from last character of the file head is as judgment base, "S", “W” and “X” change as the step of 1,2 and 5, “V” changes as the step of 1, 2.5 and 5. The followings are the pseudocode of adjustment:

```
switch (MachineType) {
case ‘S’:
case ‘W’:
case ‘X’: {
            chHs[-1] = 0.000002;
            chHs[2] = 0.00002;
            chHs[5] = 0.0002;
            chHs[8] = 0.002;
            chHs[11] = 0.02;
            chHs[14] = 0.2;
            chHs[17] = 2;
            chHs[20] = 20;
            chHs[23] = 200;
        chHs[26] = 2000;
            chHs[29] = 20000;

            }
            break;

        case ‘V’: {
            chHs[-1] = 0.0000025;
            chHs[2] = 0.000025;
            chHs[5] = 0.00025;
            chHs[8] = 0.0025;
            chHs[11] = 0.025;
            chHs[14] = 0.25;
            chHs[17] = 2.5;
            chHs[20] = 25;
            chHs[23] = 250;
            chHs[26] = 2500;
    chHs[29] = 25000
            }
            break;
```

}

The new products appeared in the future will expand the level of time base setting and the voltage setting, but the old value of level still available, which only expand the corresponded value of **chHs** and **chVv.**

In the vertical of the screen, the waveforms will be unavailable by changing zero position which of the variable range could be obtained by using the index of the voltage level in the following group of numbers. One block as unit, index 1 is in force, for example, if the index of the voltage level is 3, the variable range of zero position of the voltage level is:

Block of blockNums[3]=50*2 (include of 4*2 of available range)

int[] blockNums = int[] { -1, 400, 200, 100, 40, 20, 10, 100, 50, 25, 10 };

P.S.:

The index of voltage level is not only about the acquirement of the value of voltage level, but also affect the variable range of zero position, which the index could be obtained with another method, that is to obtain the interval voltage value of the describal point and multiply 25 (one block has 25 describal point.)to get the voltage value, and then look up reversely by chVv groups. Compared with the index of the voltage level from data file, if different, the latter is correct.

3.1.4 Data acquisition

Collection dots in full screen:

Of which is the range value instead of the actual collection value and indicating the actual acceptable collection dots in the horizontal direction, not related to the referred describable points.

The number of collection dots: Shown as 4 in 1-1, the actual collection dots of the instrument.

The data of collection dots: Data string of collection dots with the length of the number of collection dots

Slow scanning: When time base level is >=100ms(the index of time base level>=22), the instrument is in slowing scanning. The two blank just shown as 6 and 7in Fig1-1 existed in the waveform serial which are different from the range still not achieved yet when the waveform move from left to right after refreshing the screen, is the actual collection dots without be shown in the screen and the corresponding dots number is 0.08 of the number of the collection dots of full screen deemed as data of collection dots included in the number of collection dots

Slow moving: When slow scanning, the range from the left of screen to the rightmost of the first collection dots serial includes the number of actually collection dots so as the

blank which shown as 5 in Fig.1-1

In accordance with the following to analyze and program the screen drawing:
The fist step is to check whether slow scanning or not by the index of time base level to distinguish the way of ordinary drawing and the drawing just only cover collection dots data from left to right. Then, check whether the number of collection dots < whole-screen collection dots only for slowing scanning to distinguish the may existed range not yet achieved when first moving from left to right which the second drawing step the same as fist is just only to cover collection dots data from left to right.. The last step is to draw from the position of slow moving number to left only for other conditions when slow scanning. If slow moving number >= the whole-screen collection dots(1-0.08), the drawing should be a continuous collection dots serial otherwise the two ranges divided by the blank.

3.1.5 Attenuation
The special waveform could be displayed by attenuating voltage in the instrument. In the data files, the parameters concern with the voltage excepting attenuation multiplying power index, the others is not number in attenuation.

Attenuation multiplying power index: 10 is as the base, attenuation multiplying power index, for example, 1 is x10, 2 is X100.

The voltage value, the voltage level, the average value and the peak-to-peak value, the voltage value of the point of the cursor line, which displayed in the official pc software are all numbered the attenuation multiplying power.

3.2 Oscilloscope file format
Instruction:
The following value type of **int, float, short** code with the method of **Little-Endian** (that is prior to read the low byte)

A; the file header: **"SPBxxx" (six bytes, ASCII)** the last three bytes are according to the type of machine.
B: the file length: **(4 bytes ,int)** if the file length is negative value, it indicates that the type of machine is the customized machine, which absolute value indicates the file length.

C: the specification of the channel of the waveform
(1) Name of waveform: **CHx" (3 bytes, ASCII)**
X indicates the number of channel, the available is {1, 2, A, B, C, D};
(2) the length of block: **(4 bytes, int)**
The length from **"CHx"** to the data block is to divide the waveform channel.
(2.0)if the length is Positive,it is a normal wave.

if the length is Negative,it is a deep memory wave,or other extend wave.

(2.1) extended value (**4 bytes,int**)

When the length of block is Negative,this value is readable. Two bit of the value are usedful,other bits are reserves

bit0: the flag of deep memory wave,0—normal wave，1—deep memory wave

bit1: the flag of machine have deep memory，0--no，1--yes

(2.2)offset(**4 bytes, int. only for SDS series,others skip**)

Draw the wave from which points

(3) Whole screen collecting points: **(4 bytes, int)**

The collecting point contained in the horizontal line of the screen. e.g. take the 4 bytes value as int 10048; which means the bytes between "CH1" and "CH2"(or the next channel) is 10048

(4) the number of collecting point: **(4 byte,int)**

The actual number of collecting dots of the instrument

In case the deep memory，(4) the number of collecting point is all of the wave，more than(3) Whole screen collecting points，

When draw current screen wave,from the point of offset,draw (3)whole screen collection points

(5) slow moving number: **(4 byte,int)**

When slow scanning, the range from the left screen to the right point of the last collecting point could contain how many collecting points.

(6) time base level: **(4 byte,int)**

To gain the index number by looking up group numbers, the time range of a block is indicated horizontal line.

(7) zero point: **(4 byte,int)**

Take the vertical axis as middle location is 0, upper is positive, lower is neggtive. This channel waveform lays somewhere, and each point has its voltage value, the zero point location is the voltage value 0 position.

e.g. take the 4 bytes value as int +100, this channel's 0 voltage value is at this position +100, between 0 and +100 can hold 100 different sample value of voltage vertically.

(8) voltage level **(4 byte,int)**

To gain the index number by looking up group numbers, the time range of a block is indicated vertical line. e.g. take the 4 bytes value as int 5; refer to 3.1.3 Double chVv[21], then the voltage base is chVv[5] = 0.1V

(9) Attenuation multiplying power index: **(4 byte,int)**
(this value not exist in the earlier machine of **"SPBbin"**)
10 is as the base, Attenuation multiplying power index, for example, 1 is ×10, 2 is X100.

(10) The spacing interval of the describal point: **(4 bytes, float)** , unit is uS
The spacing interval of the describal point of the horizontal direction changes according to the time base level.
[Notice: this value maybe wrong, it is much better computed as: (10) = timebase uS value computed from (6) / 25, such as HDS 2062M-N]

(11) Frequency: **(4 bytes, int)** unit is Hz
Gathering frequency

(12) cycle: **(4 bytes, int)** unit is uS
Gathering cycle

(13) voltage value per point **(4 bytes, float)** , unit is mV
voltage value per point.
(14) the data of sampling point **(1.normal wave, 2bytes array, short[]**
2.deep momery wave,1byte array,byte[])
)
An array of sampling data with the length of (4)the number of collection dots

e.g. take the (13)voltage value per point as 20, and one of (14) the data of sampling point as 30, then the voltage of this sampling point is 20*30 = 600 mV; for more, take the (7)

4. Appendix
It could be convenient to program communication with USB (include OWON USB) by USB **libusb** driver.
The following will introduce the API connection which marked in the **usb.h** headline of the file and support to compile USB communication program by calling implement of the corresponding platform. The example of **C** language is in the end.

4.1 Introduction of **libusb** driver
Libusb has a series of outside **API** to call for program, by which could operate hardware. These **APIS** call for the LLI of the kernel from the source code of **libusb** which achieved function is similar to the function of the kernel driver, while **libusb** is more close to the standard of USB, which is easy for using **libusb** than developing the kernel driver.

USB driver **libusb** (open source platforms):
(1) program: **libusb** (USB 1.0- 2.0)

(2) Website: http://libusb.sourceforge.net/
(3) Word: http://libusb.sourceforge.net/doc/
(4) Version: 0.1.12
(5) Source code download:(including examples)
http://sourceforge.net/project/showfiles.php?group_id=1674
(6) compile source code and method of driver installation
http://www.linuxfromscratch.org/blfs/view/stable/general/libusb.html
note: Not windows USB driver could use the code supplied by the program.

The following introduction is about the windows program libusb-win32:
(1) Program: libusb-win32
(2) Website: http://libusb-win32.sourceforge.net/
(3) Obtain the source code and the file of binary system
http://sourceforge.net/project/showfiles.php?group_id=78138
(4) Version: 0.1.12.1
(5) Type of setup:
Upzip the downloaded **libusb-win32-device-bin-0.1.12.1.tar.gz**, find out bin catalogue, in which **libusb0.dll, libusb0.sys** are used for common driver files for all USB devices. Insert USB, use the **inf-wizard.exe** of bin catalogue to result **inf** and **cat files**. Driver installation will finish in appointing the **libusb0.dll, libusb0.sys** and inf catalogue in checking the driver installation guide of new hardware. (_x64 is for the version of 64 bit)
Note: After installation, “Safely Remove Hardware Program “will not appear in the taskbar.((**libusb-win32 0.1.8** could display such sample, if need for use, could choose it. Although the program interface of the two is the same, they could not exist at the same time. **LibUSB-Win32 Devices** could be found by device manager, also the driver registration.

If allowed to use JAVA, **Java libusb / libusb-win32 wrapper** could be used at the same time.
(1) Program: **libusbjava**
(2) Website: http://libusbjava.sourceforge.net
(3) obtain the source code and the file of binary system
http://sourceforge.net/project/showfiles.php?group_id=188245
(4) Version: 0.5.7
(5) Method:
In **classpath**, it include **ch.ntb.usb-0.5.7.jar**,
In runtime environment, it includes **LibusbJava.dll** (it could obtain the source code of **LibusbJava** to compile with **Ant** for no windows.)
(6) other information:
http://inf.ntb.ch/infoportal/help/index.jsp?topic=/ch.ntb.infoportal/tools.html

.Net platform of **libusb** also achieve **The USB library for .NET:**
(1) program: The USB library for **.NET**

(2) website: http://www.icsharpcode.net/opensource/sharpusblib/

4.2 API interface supported by USB specification in libusb
The following are introduction about the API interface, which could be ignored temporarily. After read the last example, you could look up the API introduction again.

(1) initialization interface:
These interfaces are named as kernel functions, which are used for initialization instrument and looking for some instruments.

usb_init
Function Definition: **void usb_init(void);**
For initialization related data.

usb_find_busses
Function Definition: **int usb_find_busses(void);**
Look for USB bus line in the system, which connect the communication of the computer and USB, to achieve communication with other instruments. The function is back to the bus line number.

usb_find_devices
Function Definition: **int usb_find_devices(void);**
Look for USB in bus line, the function use after calling **usb_find_busses()**, then back to instrument. The above three functions call by sequencing when initialization.

usb_get_busses
Function Definition: **struct usb_bus *usb_get_busses(void);**
The function is back to the list of bus line, which is no available in some advanced versions. It could use **usb_busses** as shown in example.

(2) Operating equipment interface
usb_open
Function Definition: **usb_dev_handle *usb_open(struct *usb_device dev);**
call **usb_open** to open the instrument before operating the hardware, **usb_dev_handle** and **usb_device** have definitions in **usb.h** and **usbi.h** of **libusb. usb_dev_handle** is for the handle of opening instrument, while **usb_device** contain the information about opening instrument.

usb_close
Function Definition: **int usb_close(usb_dev_handle *dev);**
With corresponding to **usb_open**, close the instrument, call when communication is finished. if back to O, it succeed. If <0, it fail.

usb_set_configuration
Function Definition: **int usb_set_configuration(usb_dev_handle *dev, int configuration);**
Install **configuration** which use **bConfigurationValue** of **configurtation descriptoes**, if back to O, it succeed. If <0, it fail.(An instrument contains several configurations, such as the supported highway instrument and the low way instrument have two different configurations, for detail, please refer to USB standard.)

usb_set_altinterface
Function Definition: **int usb_set_altinterface(usb_dev_handle *dev, int alternate);**
The function has interface descriptor, in which of **bAlternateSetting** could be used parameter alternate. if back to O, it succeed. If <0, it fail.

usb_resetep
Function Definition: **int usb_resetep(usb_dev_handle *dev, unsigned int ep);**
Endpoint of appointed reset, **ep** is reffered to **bEndpointAddress** which could replace **usb_resetep.**

usb_clear_halt
Function Definition: **int usb_clear_halt(usb_dev_handle *dev, unsigned int ep);**
Endpoint of appointed reset, **ep** is referred to **bEndpointAddress** which could replace **usb_resetep.**

usb_reset
Function Definition: **int usb_reset(usb_dev_handle *dev);**
The function is not often available, so **usb_close** could be satisfied.

usb_claim_interface
Function Definition: **int usb_claim_interface(usb_dev_handle *dev, int interface);**
The function should be called to register the interface of the operating communication. Only registered interface firstly and then to operate.

usb_release_interface
Function Definition: **int usb_release_interface(usb_dev_handle *dev, int interface);**
Log out the interface of called function **usb_claim_interface** and release resources. Then use corresponding to **usb_claim_interface** as shown in example.

(3) Control transmission interface.
usb_control_msg
Function Definition: **int usb_control_msg(usb_dev_handle *dev, int requesttype, int request, int value, int index, char *bytes, int size, int timeout);**
Transfer and accept data from default channel.

usb_get_string

Function Definition: **int usb_get_string(usb_dev_handle *dev, int index, int langid, char *buf, size_t buflen);**
Obtain the descriptor of the character string by appointed **index** and **langid.**

usb_get_string_simple
Function Definition: **int usb_get_string_simple(usb_dev_handle *dev, int index, char *buf, size_t buflen);**
Obtain the descriptor of the character string by appointed **index**. Default use is **langid.**

usb_get_descriptor
Function Definition: **int usb_get_descriptor(usb_dev_handle *dev, unsigned char type, unsigned char index, void *buf, int size);**
Obtain the descriptor of the character string from default channel by appointed type and index.

usb_get_descriptor_by_endpoint
Function Definition: **int usb_get_descriptor_by_endpoint(usb_dev_handle *dev, int ep, unsigned char type, unsigned char index, void *buf, int size);**
Obtain the descriptor of the character string from **ep** channel by appointed **type** and **index.**

(4) Batch transmission interface

usb_bulk_write
Function Definition: **int usb_bulk_write(usb_dev_handle *dev, int ep, char *bytes, int size, int timeout);**
Do a block write in appointed **EP.**

usb_bulk_read
Function Definition: **int usb_bulk_read(usb_dev_handle *dev, int ep, char *bytes, int size, int timeout);**
Do a block read in appointed **EP.**

(5) Interrupting transmission interface

usb_interrupt_write
Function Definition: **int usb_interrupt_write(usb_dev_handle *dev, int ep, char *bytes, int size, int timeout);**
Do an interrupt write in appointed **EP.**

usb_interrupt_read
Function Definition: **int usb_interrupt_read(usb_dev_handle *dev, int ep, char *bytes, int size, int timeout);**
Do an interrupt read in appointed **EP.**

4.3 Example code

The following is about the use of **libusb API** in **C** language and data acquirement of OWON .

Program environment is **Windows XP, Dev-Cpp, Gcc. libusb0.dll** and **usb.h** could be obtained in **libusb-win32**, which add to **lib** and **include** separately to ensure the successful installation of **ibusb-win32** driver and **.inf** in OWON USB.

```
/*must contain usb.h */
#include "usb.h"
#include <stdio.h>

/* VENDOR_ID and PRODUCT_ID of OWON USB */
#define VENDOR_ID            0x5345
#define PRODUCT_ID           0x1234

/* WRITE_ENDPOIN and READ_ENDPOINT of OWON USB */
#define WRITE_ENDPOINT       0x03
#define READ_ENDPOINT        0x81

/*interface parameter of OWON USB */
#define _CONFIGURATION       1
#define _INTERFACE           0
#define _ALTINTERFACE        -1

/*parameter used in data acquirement protocols of OWON USB */
#define START_COMMAND        "START"
#define RESPONSE_START_LENGTH 12
#define BMP_FILE_FLAG        1
#define BMP_FILE_EXTENTION    "bmp"
#define BIN_FILE_EXTENTION    "bin"
#define USB_TIMEOUT          2000

/*several types and parameters could be defined by itself when programming*/
#define NOT_OPENED_DEVICE    -1
#define WRITE_ERROR          -2
#define READ_ERROR           -3
#define SUCCESS              1
#define DEVICE_MINOR         16

/*the mapping relation of usb_device*and usb_dev_handle*are saved as
device_descript*/
typedef struct
```

```
{
   struct usb_device*  udev;
   usb_dev_handle*     device_handle;

} device_descript;

/*device_descript arrays*/
device_descript g_list[ DEVICE_MINOR ];
int g_num;

/* enumerate available USB, which could be match with VENDOR_ID and
PRODUCT_ID */
int Device_Find()
{
   struct usb_bus      *bus;
   struct usb_device   *dev;
   g_num = 0;

   /* initialization before using of libusb */
   usb_init();
   /*look for bus line*/
   usb_find_busses();
   /*look for instrument*/
   usb_find_devices();

   /*usb_busses have the instruments, in usb.h to find the matching instrument with
VENDOR_ID and PRODUCT_ID */
   for (bus = usb_busses; bus; bus = bus->next) {
      for (dev = bus->devices; dev; dev = dev->next) {
         if(dev->descriptor.idVendor==VENDOR_ID&& dev->descriptor.idProduct
== PRODUCT_ID) {

            if (g_num < DEVICE_MINOR) {
               g_list[g_num].udev = dev;
               g_num ++;
            }
         }
      }
   }

   return g_num;
}
```

```
/*through device_descript*to open the instrument*/
int Device_Open(device_descript* descript){
   /*open USB for communication and back to handle*/
   usb_dev_handle* device_handle = descript->device_handle =
usb_open(descript->udev);

   /*use three instrument parameters to set up communication*/
   if (usb_set_configuration(device_handle, _CONFIGURATION) < 0) {
      return NOT_OPENED_DEVICE;
   }
   if (usb_claim_interface(device_handle, _INTERFACE) < 0) {
      return NOT_OPENED_DEVICE;
   }
   if(_ALTINTERFACE>=0){
      if (usb_set_altinterface(device_handle, _ALTINTERFACE) < 0) {
         usb_release_interface(device_handle, _ALTINTERFACE);
         return NOT_OPENED_DEVICE;
   }}

   return 0;
}

/*use device_descript*to close USB communication*/
int Device_Close(usb_dev_handle* device_handle){
   /*release interface*/
   usb_release_interface(device_handle, _INTERFACE);
   /*close USB communication*/
   usb_close(device_handle);
}

/*use the instrument acquirement protocols of OWON USB to accept data*/
int Device_COMM(usb_dev_handle* device_handle)
{
   /*the array for XMT*/
   char   send_data[] = START_COMMAND;
   /*the array for accept transmission information*/
   char   recv_data[RESPONSE_START_LENGTH];

   int          send_len;
   int          recv_len;

   memset(recv_data, 0 , sizeof(recv_data));
```

```
   /*XMT by block transmission, or interrupt transmission */
   send_len = usb_bulk_write(device_handle,
      WRITE_ENDPOINT, send_data, sizeof(send_data), USB_TIMEOUT);

   /*accept bits*/
   if (send_len < sizeof(send_data)) {
      return WRITE_ERROR;
   }

   /*accept transmission information by block transmission, or interrupt transmission */
   recv_len = usb_bulk_read(device_handle,
      READ_ENDPOINT, recv_data, sizeof(recv_data), USB_TIMEOUT);

   /* accept bit */
   if (recv_len < sizeof(recv_data)) {
      return READ_ERROR;
   }

   /*the first three bytes by accepting data files is the file size, the eighth byte indicates
picture.*/
   int file_len = ((recv_data[2]&0xff)<<16)+((recv_data[1]&0xff)<<8)+
(recv_data[0]&0xff);
   int bmp_flag = recv_data[8]&0xff;
   printf("file_len: %d\n", file_len);
   /*1 is jpg, suffix is.bmp; 0 is data file, suffix is .bin*/
   printf("file_type: %s\n", bmp_flag==BMP_FILE_FLAG?
BMP_FILE_EXTENTION:BIN_FILE_EXTENTION);

   /*the number group for accepting file directly, when programming please
considering about repeatedly accepting by buffer to write to the end of file*/
   char data[file_len];
   memset(data, 0 , sizeof(data));

   /*accept data files, suitable for block transmission*/
   recv_len = usb_bulk_read(device_handle,
      READ_ENDPOINT, data, file_len, USB_TIMEOUT);

   /*accept bit*/
   if (recv_len < file_len) {
      printf("usb_strerror(): %s\n", usb_strerror());
      printf("recv_len: %d\n", recv_len);
      return READ_ERROR;
   }
```

```
    /*save as file*/
    FILE* fp;
    fp = fopen("c:\\ddd", "wb");/*wb*/
    fwrite(data, sizeof(char), file_len, fp);
    fclose(fp);
    printf("save file here   c:\\ddd\n");

    return SUCCESS;
}

int main(int argc, char *argv[])
{
    /*look up*/
    int num = Device_Find();
    printf("dev_num: %d\n", num);
    if(num <= 0)
        return -1;

    /*open*/
    int flag = Device_Open(&g_list[0]);
    printf("open device: %d\n", flag);

    usb_dev_handle* device_handle = g_list[0].device_handle;
    /*transmission*/
    flag = Device_COMM(device_handle);

    printf("communication device: %d\n", flag);

    /*closed*/
    Device_Close(device_handle);

    getchar();
    return 0;
}
```