GWU2X Programming Guide_U2X_IIC

UG1002-1.0E, 6/29/2021

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. All information in this document should be treated as preliminary. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
6/29/2021	1.0E	Initial version published.

Contents

List of Figures

List of Tables

1 General Description

GWU2X is a USB to multiple protocol converter that enables USB2IIC function conversion and supports up to 500 kHz SCL clock frequency (current test result).

It supports IIC host mode data transceiver function and supports 7-bit address mode and 10-bit address mode.

2 Driver Installation and Uninstallation

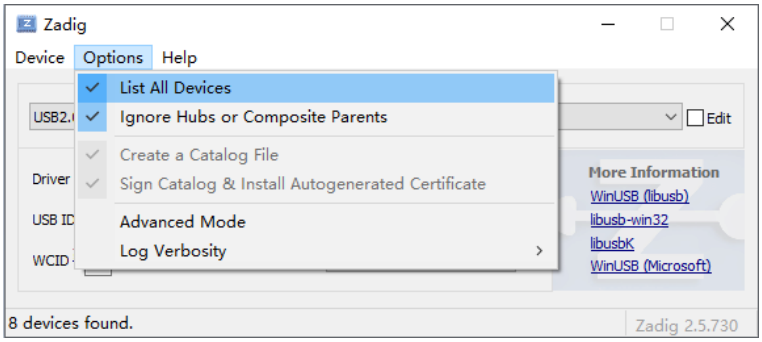
GWU2X_IIC can be programmed using libusb, the open source USB function library. To program with this function library, the “WinUSB.sys” USB driver program needs to be installed.

You can use Zadig(https://zadig.akeo.ie/), the open source driver installation tool, to install driver. The driver installation requires administrator privileges.

2.1 Use Zadig to Install Driver

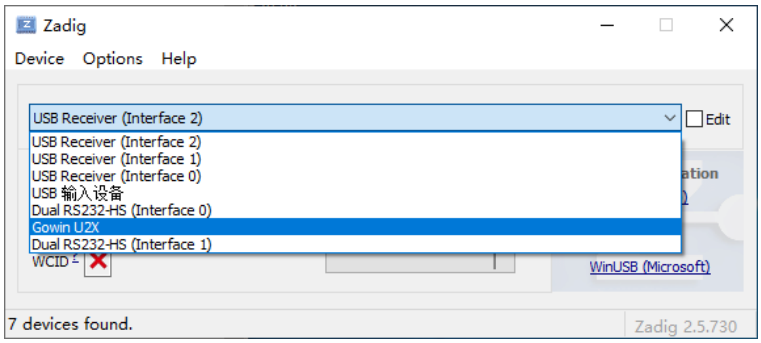
Connect GWU2X device to the computer USB interface, double-click to open Zadig (administrator privileges required), click Options, and check the "List All Device" option. All USB devices connected to the computer will be listed.

Figure 2-1 Check “List All Device” Option



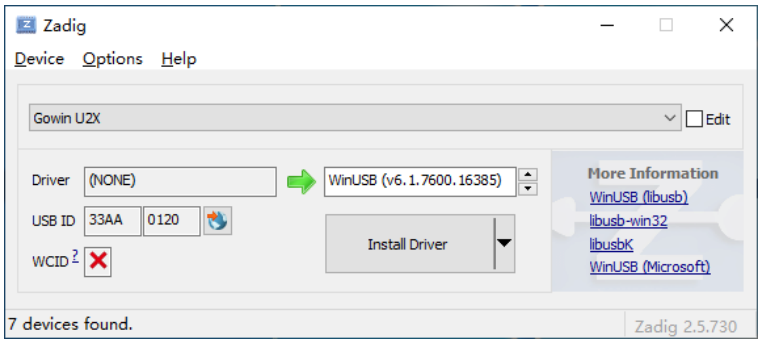
Select GWU2X, the device that requires driver installation.

Figure 2-2 Select the Device that Requires Driver Installation



Select the driver to be installed, use libusb+WinUSB, and select WinUSB.

Figure 2-3 Select the Driver Program to be Installed



Click “Install Driver”. The driver will be installed after a few moments.

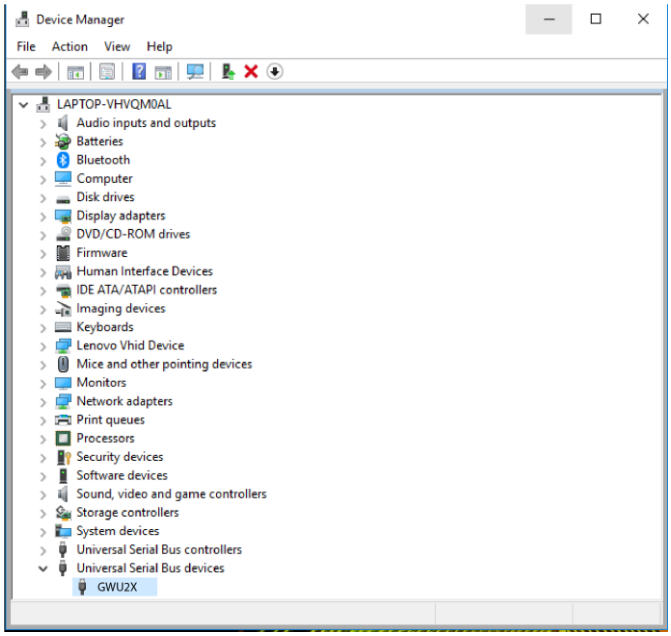
Note!

The button displays "Install Driver" if the driver is not currently installed, and "Replace Driver" if another driver is currently installed.

2.2 Uninstall Driver

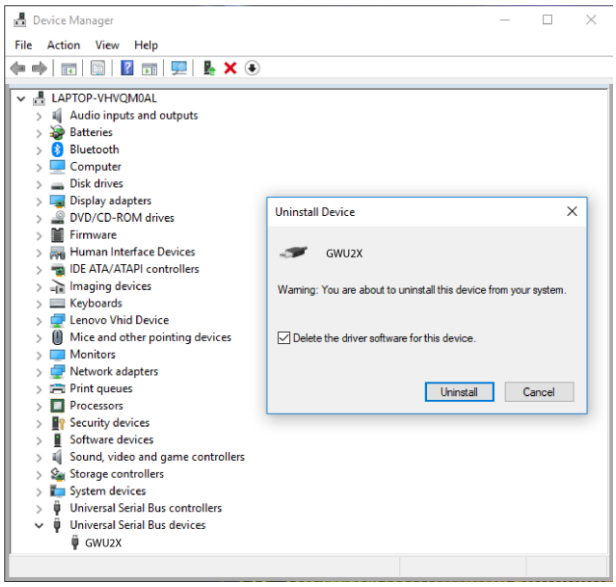
To uninstall the driver, connect GWU2X device to the computer, open the windows device manager, and find GWU2X device in the "Universal Serial Bus Devices" list. Right-click on the device name and select the "Uninstall Device" option in the pop-up menu.

Figure 2-4 Open Device Manager



In the pop-up dialog box, first check "Remove driver software for this device", and then click the “Uninstall” button to uninstall the driver.

Figure 2-5 Uninstall Device



3 libusb_WinUSB Programming

libusb is an open source USB function library.

The official website is: https://libusb.info

The source code is hosted on github at: https://github.com/libusb/libusb

You can download the pre-compiled version, the official GCC version and VS version, including dynamic and static libraries, through the official website. You can also download the source code through github and compile it on your own.

For libusb function descriptions, see the official reference at: http://libusb.sourceforge.net/api-1.0

3.1 Libusb Initialization and Exit

When programming with libusb, you need to call the function libusb_init() to initialize it first, and at the end of use, you should call the function libusb_exit() to exit it from the system.

Function declarations are as follows:

```
int libusb_init (libusb_context ** context)

void libusb_exit (libusb_context * ctx)
```

The parameter libusb_context is a libusb context struct that saves some configuration parameters for libusb. If libusb_context is not specified, a default context struct will be created, or if one already exists, it will be used directly and not reinitialized.

Programming examples are as follows:

```
int rc = libusb_init(NULL);

if (rc < 0)

    return rc;
```

```
libusb_exit(NULL);
```

3.2 Open the Specified USB Device

You can use the libusb_open_device_with_vid_pid() function to open the specified device based on VID/PID. You can also use the libusb_get_device_list() function to get all the USB devices, select the desired device from them, and use the libusub_open() function to get the handle of the device for subsequent operations. The function declaration is as follows:

Open the device with VID/PID:

```
libusb_device_handle* libusb_open_device_with_vid_pid(
   libusb_context * ctx,
   uint16_t vendor_id,
   uint16_t product_id
);
```

The parameter ctx is the address of the context struct generated when initializing libusb. If the default context is used, use NULL. vendor_id and product_id are the VID and PID of the USB device, respectively. The VID of Gowin USB device is 0x33aa, and the PID of the GWU2X device is 0x0020.

The return value is the pointer to the operation handle of the first matching device found by libusb on this computer, otherwise it returns the null pointer NULL.

Examples of use are as follows:

```
devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0020);
if(NULL == devh) {
   printf("Open USB device failed\n");
   goto out;
}
```

Select the specified device after getting all USB devices.

```
ssize_t libusb_get_device_list (
   libusb_context * ctx,
   libusb_device *** list
)
```

The parameter ctx is the address of the context struct generated when

initializing libusb. If the default context is used, use NULL. “list” is the pointer to storage device list.

At the end of use, the memory should be freed using the libusb_free_device_list() function.

If the function is executed correctly, the return value is the number of devices and the list saves the list of found devices. Otherwise, a libusb_error value less than zero is returned.

```
int libusb_open (
    libusb_device *dev,
    libusb_device_handle **dev_handle
    )
```

The parameter dev is the device in the device list, and dev_handle is the address that saves the pointer of the returned device handle.

If the device is opened successfully, the return value is zero, otherwise an libusb_error value less than zero is returned.

Examples of use are as follows:

```
cnt = libusb_get_device_list(NULL, &devs);
if(cnt < 0) {
    // get device list failed
    return -1;
}
for(int i = 0; i < cnt; i++) {
    libusb_open(dev[i], dev_handle);
    if(/*the wanted device is opened*/) {
        break;
    } else {
        //the current device is not wanted, close it and check the next one.
        libusb_close(dev_handle);
    }
}
```

3.3 Interface Declaration

USB devices usually contain one or more interfaces. libusb needs to

declare the interface first when using the interface, and when the declaration is successful, it means that the interface is successfully opened and the endpoint contained in the interface can be received/transmitted.

```
int libusb_claim_interface(
    libusb_device_handle * dev_handle,
    int interface_number
)
```

The parameter dev_handle is the device handle; interface_number is the number of interface. In GWU2U device, the interface number is 0. If the interface is declared successfully, the return value is zero, otherwise an libusb_error value less than zero is returned.

Programming Examples:

```
rc = libusb_claim_interface(devh, 0);
if (rc < 0) {
    printf("Error claiming interface: %s\n", libusb_error_name(rc));
    goto out;
}
```

4 U2X_IIC API Functions

4.1 U2X_IIC Initialization

This function is used to initialize U2X_IIC and to configure the SCL clock frequency. If you need to modify the SCL clock frequency, you need to call this function again for configuration.

```
int u2x_iic_init(
libusb_device_handle *devh,
unsigned int uiFreqKiloHz,
unsigned int uiTimeOut);
```

Parameters:

- devh: device handle of libusb;
- uiFreqKiloHz: SCL clock Frequency, in Khz;
- uiTimeout: timeout parameter, in milliseconds;

Return Value:

Returns 0 if the function runs successful, otherwise an error code less than zero is returned.

4.2 Data Transmission

Sends data via IIC to a specified IIC Slave device, with a maximum length of 256 bytes at a time.

```
int u2x_iic_send_bytes(
 libusb_device_handle *devh,
 unsigned int uiAddr,
 unsigned int uiAddrBit,
```

```
unsigned char *pucData,
unsigned int uiDataLen,
unsigned int uiTimeout);
```

Parameters:

- devh: device handle of libusb;
- uiAddr: Address of IIC Slave device;
- uiAddrBit: address bits. Set this parameter to 7 if 7-bit address is used; Set this parameter to 10 if 10-bit address is used. Other values are not available.
- pucData: A pointer to the Memory address that holds the data to be sent.
- uiDataLen: The count of data bytes to be transmitted；
- uiTimeout: timeout parameter, in milliseconds;

Return Value:

Returns 0 if the function runs successful, otherwise an error code less than zero is returned.

4.3 Data receiving

Sends data via IIC to a specified IIC Slave device, with a maximum length of 256 bytes at a time.

```
int u2x_iic_read_bytes(
libusb_device_handle *devh,
unsigned int uiAddr,
unsigned int uiAddrBit,
unsigned char *pucData,
unsigned int uiDataLen,
unsigned int uiTimeout);
```

Parameters:

- devh: device handle of libusb;
- uiAddr: Address of IIC Slave device;
- uiAddrBit: address bits. Set this parameter to 7 if 7-bit address is used; Set this parameter to 10 if 10-bit address is used. Other values are not available.
- pucData: A pointer to the memory address that holds the received data.
- uiDataLen: The count of data bytes to be received；
- uiTimeout: timeout parameter, in milliseconds;

Return Value:

Returns 0 if the function runs successful, otherwise an error code less than zero is returned.

4.4 Programming Example

```
int main(int argc, char *argv[])
{
unsigned char txdata[TEST_BYTE];
    unsigned char rxdata[TEST_BYTE];
    int i = 0;
    int rc = 0;
    // Open the device
    rc = libusb_init(NULL);
    if (rc < 0)
    return rc;
    devh = libusb_open_device_with_vid_pid(NULL, 0x33aa, 0x0120);
    if(NULL == devh) {
        printf("Open USB device failed\n");
        goto out;
    }
    rc = libusb_claim_interface(devh, 0);
    if (rc < 0) {
        printf("Error claiming interface: %s\n", libusb_error_name(rc));
        goto out;
    }
    txdata[0] = 0x55;
    for(i = 1; i < TEST_BYTE; i++) {
        txdata[i] = i;
    }
    // Initialize U2X IIC
    u2x_iic_init(devh, 400, 1000);
    //Sends 4 bytes of data to an IIC Slave device at address 0x50 in 7-bit mode
    u2x_iic_send_bytes(devh, 0x50, 7, txdata, 4, 1000);
    // Reads 4 bytes of data from an IIC Slave device at address 0x50 in 7-bit mode
```

```
        u2x_iic_read_bytes(devh, 0x50, 7, rxdata, 4, 1000);
       // Close the device and Exit
       libusb_release_interface(devh, 0);
out:
   if(devh)
                libusb_close(devh);
   libusb_exit(NULL);
   return 0;
            }
```

5 Error Code

If the API function runs well, it returns 0; otherwise, it returns a negative number.

The meaning of the non-zero return values are shown in the table as below.

Table 5-1 Error Code List

Value	Enumerator	
0	SUCCESS	Runs correctly
-1	USB_ERROR_IO	USB input/output error
-2	USB_ERROR_INVALID_PARAM	USB parameter error
-3	USB_ERROR_ACCESS	No permission to access the device
-4	USB_ERROR_NO_DEVICE	No USB device (device disconnected)
-5	USB_ERROR_NOT_FOUND	No Entity
-6	USB_ERROR_BUSY	USB device busy
-7	USB_ERROR_TIMEOUT	Timeout
-8	USB_ERROR_OVERFLOW	Memory overflow
-9	USB_ERROR_PIPE	Pipe error
-10	USB_ERROR_INTERRUPTED	The system function was interrupted
-11	USB_ERROR_NO_MEM	Out of memory
-12	USB_ERROR_NOT_SUPPORTED	Not supported on the current platform
-13	U2X_IIC_ERROR_USBTRANS_ERR	USB data transmitting error
-14	U2X_IIC_ERROR_INVALID_PARAM	Invalid parameter setting
-15	U2X_IIC_ERROR_TIMEOUT	U2X_IIC transmitting timeout

Value	Enumerator	
-16	U2X_IIC_ERROR_CMD_ERR	U2X_IIC command error
-17	U2X_IIC_ERROR_NO_SPEC_SLV	The specified slave device not found by U2X_IIC
-99	ERROR_OTHER	Other errors

Terminology and Abbreviations

The abbreviations and terminology used in this manual are as shown in Table A -1.

Table A- 1 Terminology and Abbreviations

Terminology and Abbreviations	Full Name
USB	Universal Serial Bus
IIC	Inter—Integrated Circuit

Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail:support@gowinsemi.com


GOWIN
PROGRAMMING FOR THE FUTURE