GWU2X Programming Guide_U2X_JTAG

UG1003-1.0E, 06/29/2021

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
06/29/2021	1.0E	Initial version published.

Contents

List of Tables

1 General Description

GWU2X can be used to create a USB to JTAG protocol interface to read/write JTAG interfaces. Its TCK clock frequency can be configured. It supports up to 15MHz TCK according to the current test result. It can send or read IR/DR data of any BIT length.

2 Driver Installation and Uninstallation

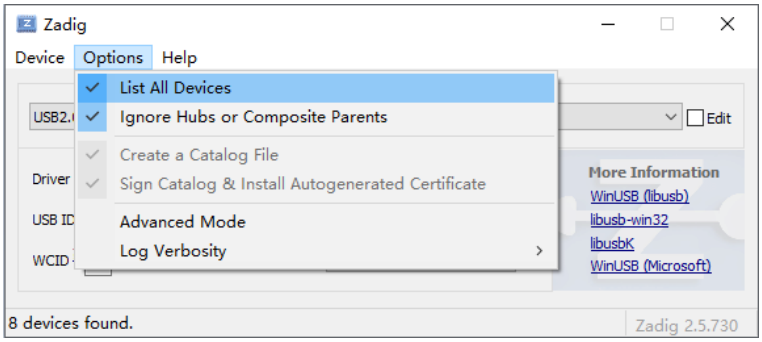
GWU2X _JTAG can be programmed using libusb, the open source USB function library. To program with this function library, the “WinUSB.sys” USB driver program needs to be installed.

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

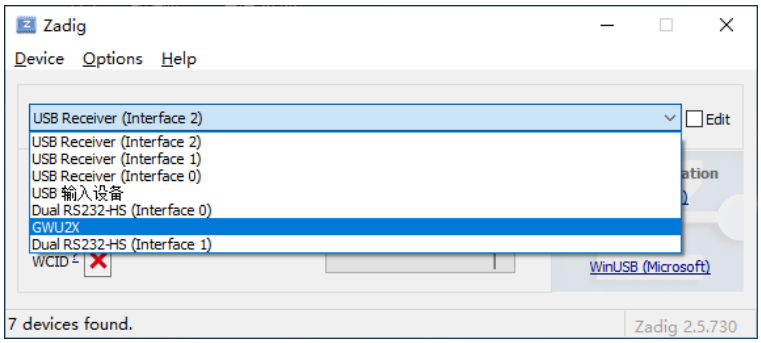
Connect GWU2X device to the computer USB interface, double-click to open Zadig (administrator privileges required), click Options, and check the "List All Device" option. All USB devices connected to the computer will be listed.

Figure 2-1 Check “List All Device” Option



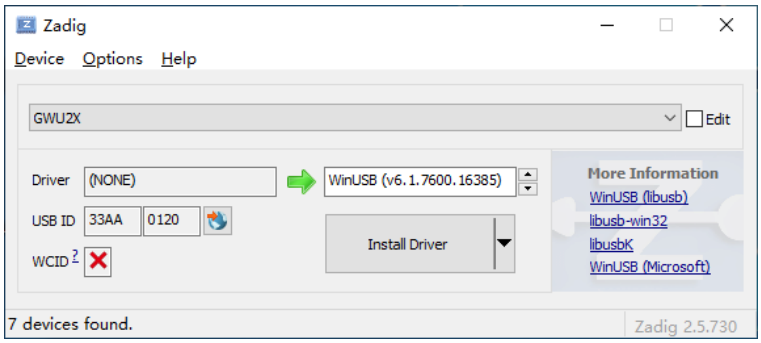
Select GWU2X, the device that requires driver installation.

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.

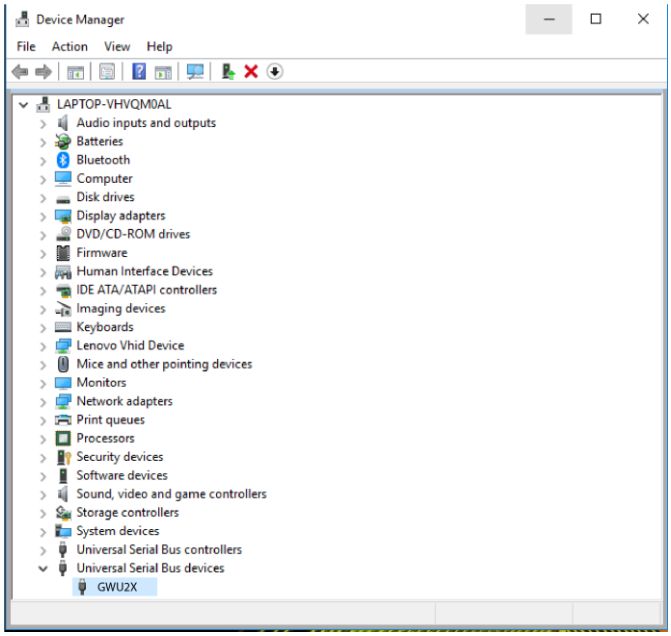
Note!

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

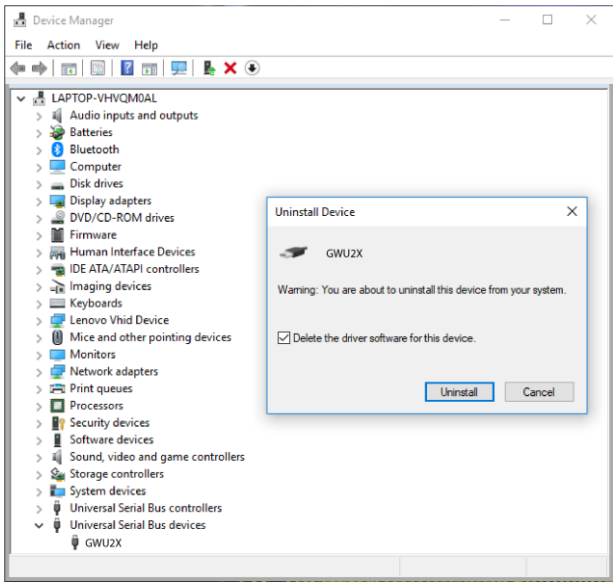
To uninstall the driver, connect GWU2X device to the computer, open the windows device manager, and find GWU2X device in the "Universal Serial Bus Devices" list. Right-click on the device name and select the "Uninstall Device" option in the pop-up menu.

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 Programming

3.1 API Functions

3.1.1 Get JTAG Status

```
const char * jtag_get_state_str(int iJtagStateCode);
```

Parameters:

iJtagStateCode: A variable of the JTAG state. For the variable definition, please refer to the enumerated value of JTAG_STATE defined in gw_usb2jtag.h.

Return Value:

A const char pointer to the address of a string that describes JTAG status. It can be used to output information for debugging.

3.1.2 Get Running Error Information

```
const char * usb2jtag_get_err_info(int errCode);
```

Parameters:

errCode: Returns the error code of API functions. Returns 0 if successful; returns a negative value if an error occurs in the function running.

Return Value:

A const char pointer to the address of a string that describes error Information. It can be used to output information for debugging.

3.1.3 Generate TMS Signal Array

```
int jtag_generate_tms_array(
int *pJtagState,
int iTargetState,
int *pTmsBitCnt,
unsigned int *pTmsBitArray);
```

Parameters:

pJtagState: A pointer to a variable that holds the current JTAG state. If running successful, the variable will be updated to the JTAG state after the TMS signal is sent.

iTargetState: Send a TMS signal to cause the JTAG device to jump to the target JTAG state.

pTmsBitCnt: A pointer to a variable that holds the length of the generated TMS signal array.

pTmsBitArray: A pointer to a variable that holds the value of the generated TMS signal array.

Return Value:

Returns 0 if the parameter setting is successful, otherwise an error code less than zero is returned

3.1.4 Set TCK Frequency

```
int usb2jtag_tckset(
libusb_device_handle *devh,
unsigned int uiFreqKiloHz,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

uiFreqKiloHz: TCK frequency to be set, in KHz;

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

Returns 0 if the parameter setting is successful, otherwise an error

code less than zero is returned

3.1.5 Set IO Ports

```
int usb2jtag_ioset(
libusb_device_handle *devh,
unsigned short usIODir,
unsigned short usIOLevelSet,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

uiIODir: Set the direction of IO ports. JTAG TCK/TMS/TDI needs to be set to output mode and TDO needs to be set to input mode. For the parameter definition, please refer to the macro definition in gw_usb2jtag.h.

uiIOLevelSet: Set the level of the IO port during its idle time. During idle time, TMS/TDI is usually set to high and TCK/TDO to low. For the parameter definition, please refer to the macro definition in gw_usb2jtag.h.

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

Returns 0 if the parameter setting is successful, otherwise an error code less than zero is returned

Example:

```
usIODir = JTAG_GPIO_TCK_OUT | JTAG_GPIO_TMS_OUT | JTAG_GPIO_TDI_OUT;
usIOLevelSet = JTAG_GPIO_TMS_HIGH | JTAG_GPIO_TDI_HIGH;
usb2jtag_ioset(devh, usIODir, usIOLevelSet, TIMEOUT);
```

3.1.6 JTAG Status Reset

When this function is executed, JTAG sends a series of TMS high level signals, causing the JTAG device state to jump to TEST LOGIC RESET.

```
int usb2jtag_reset(
libusb_device_handle *devh,
int *pJtagCurrentState,
```

```
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

pJtagCurrentState: A pointer to a variable that holds the current JTAG status. After the function is executed, the variable will change to TEST_LOGIC_RESET (0x0).

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

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

3.1.7 JTAG Status Jumping to IDLE

When this function is executed, JTAG sends TMS signals, causing the JTAG device state to jump to RUN TEST IDLE.

```
int usb2jtag_goto_idle(
libusb_device_handle *devh,
int *pJtagCurrentState,
unsigned short usIdleTckLen,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

pJtagCurrentState: A pointer to a variable that holds the current JTAG status. After the function is executed, the variable will change to RUN TEST IDLE (0x0).

usIdleTckLen: After jumping to Run Test Idle, it continues to generate a TCK with the specified length. If it is not required, this parameter is passed to 0.

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

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

3.1.8 Send IR Data

This function sends TMS signals first to cause JTAG status to jump to SHIFT IR, and then sends the specified IR data. After that, sends TMS signals to cause JTAG status to jump to RUN TEST IDLE, and continues to send TCK signals with specified cycles if necessary.

```
int usb2jtag_shift_ir(
libusb_device_handle *devh,
int *pJtagCurrentState,
int iTdiBitCnt,
unsigned char *ucTdiData,
unsigned char *ucTdoReadBack,
unsigned short usIdleTckLen,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

pJtagCurrentState: A pointer to a variable that holds the current JTAG status. After the function is executed, the variable will change to RUN TEST IDLE (0x0).

iTdiBitCnt: The BIT length of IR data sent by TDI.

TransData: A pointer to the IR data to be sent by TDI.

ucTdoReadBack: A pointer to the address that holds the TDO readback data. If the TDO readback data is not required to be stored, this parameter is passed to NULL.

usIdleTckLen: After jumping to RUN TEST IDLE, it continues to generate a TCK with the specified length. If it is not required, this parameter is passed to 0.

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

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

3.1.9 Send DR Data

This function sends TMS signals first to cause JTAG status to jump to SHIFT DR, and then sends the specified DR data. After that, sends TMS signals to cause JTAG status to jump to RUN TEST IDLE, and continues to send TCK signals with specified cycles if necessary.

```
int usb2jtag_shift_dr(
libusb_device_handle *devh,
int *pJtagCurrentState,
int iTdiBitCnt,
unsigned char *ucTdiData,
unsigned char *ucTdoReadBack,
unsigned short usIdleTckLen,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

pJtagCurrentState: A pointer to a variable that holds the current JTAG status. After the function is executed, the variable will change to RUN TEST IDLE (0x0).

iTdiBitCnt: The BIT length of DR data sent by TDI.

ucTdiData: A Pointer to the DR data to be sent by TDI.

ucTdoReadBack: A pointer to the address that holds the TDO readback data. If the TDO readback data is not required to be stored, this parameter is passed to NULL.

usIdleTckLen: After jumping to RUN TEST IDLE, it continues to generate a TCK with the specified length. If it is not required, this parameter is passed to 0.

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

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

3.1.10 Readback TDO Data

This function reads back the TDO data currently stored.

```
int usb2jtag_readback_tdo(
libusb_device_handle *devh,
int iBitsToRead,
unsigned char *ucTdoReadBack,
unsigned int uiTimeOut);
```

Parameters:

Devh: USB device handle in libusb, used to operate the USB2JTAG device.

iBitsToRead: The BIT length of TDO that needs to be read back.

ucTdoReadBack: A pointer to the address that holds the TDO readback data.

uiTimeout: Timeout parameter, in ms. It's infinite waiting if set to “0”.

Return Value:

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

3.1.11 Generate Commands to Send DR Data

This function generates the commands of sending DR data according to the incoming parameters, but the DR data will not be sent until a set of several instructions to send DR data are generated and stored in a continuous address and then the “libusb_bulk_transfer” function in libusb is used to sent all at once, which can improve the transmission efficiency.

```
int usb2jtag_build_shift_dr_cmd(
int *pJtagCurrentState,
int iTdiBitCnt,
unsigned char *ucTdiData,
unsigned short usIdleTckLen,
unsigned char ucIsTdoRdbk,
unsigned char *ucCmdMem,
int *piCmdMemByteLeft,
```

```
int *piCmdByteCnt);
```

Parameters:

pJtagCurrentState: A pointer to a variable that holds the current JTAG status. After the function is executed, the variable will change to RUN_TEST_IDLE.

iTdiBitCnt: The BIT length of the DR data sent by TDI.

ucTdiData: A Pointer to the address that holds the DR data to be sent by TDI.

usIdleTckLen: How many TCK cycles needs to be sent after sending and jumping to IDLE. If no TCK cycle is required, this parameter is passed as 0.

ucIsTdoRdbk: Whether the TDO readback data needs to be saved. If so, this parameter is passed as 1; otherwise, this parameter is passed as 0.

ucTdoReadBack: A pointer to the address that holds the generation commands.

piCmdMemByteLeft: A pointer to a variable that indicates how much byte space is left in the address that holds the generated commands.Upon successful execution of this function, the value of the variable is automatically updated.

piCmdByteCnt: A pointer to a variable that holds the BYTE length of the comands generated by this execution of the function.

Return Value:

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

3.2 Programming Example

```
#define TIMEOUT  1000

usIODir = JTAG_GPIO_TCK_OUT | JTAG_GPIO_TMS_OUT |
JTAG_GPIO_TDI_OUT;

usIOLevelSet = JTAG_GPIO_TMS_HIGH |
JTAG_GPIO_TDI_HIGH;


usb2jtag_ioset(devh, usIODir, usIOLevelSet, TIMEOUT);

usb2jtag_tckset(devh, 1330, TIMEOUT);

usb2jtag_reset(devh, &iJtagCurrentState, TIMEOUT);
```

```
   usb2jtag_goto_idle(devh, &iJtagCurrentState, 0,
TIMEOUT);
   rc = usb2jtag_shift_ir(devh, &iJtagCurrentState, 5,
&ir_data, &ir_rdbk, 10, TIMEOUT);
   if(rc != 0) {
     printf("usb2jtag shift ir test failed...\n");
     goto out;
   }
   printf("ir tdo read back: 0x%02x\n", ir_rdbk);

   rc = usb2jtag_shift_dr(devh, &iJtagCurrentState, 58,
(unsigned char *)(&dr_data), (unsigned char *)(&dr_rdbk),
20, TIMEOUT);
   if(rc != 0) {
     printf("usb2jtag shift dr test failed...\n");
     goto out;
   }
   printf("dr tdo read back: %016llx\n", dr_rdbk);
```

4 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 4-1 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	U2J_ERROR_USBTRANS_ERR	USB data transmitting error
-14	U2J_ERROR_INVALID_PARAM	Invalid U2X_JTAG parameter setting

Value	Enumerator	
-15	U2J_ERROR_TIMEOUT	U2X_JTAG transmitting timeout
-16	U2J_ERROR_CMD_ERR	U2X_JTAG command error
-99	ERROR_OTHER	Other errors

Terminology and Abbreviations

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

Table A-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
USB	Universal Serial Bus
JTAG	Joint Test Action Group

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