



GWU2U Driver (libusb based)

ユーザーガイド

UG1006-1.1J, 2021-12-23

著作権について(2021)

著作権に関する全ての権利は、**Guangdong Gowin Semiconductor Corporation** に留保されています。

GOWIN高云、Gowin、及びGOWINSEMIは、当社により、中国、米国特許商標庁、及びその他の国において登録されています。商標又はサービスマークとして特定されたその他全ての文字やロゴは、それぞれの権利者に帰属しています。何れの団体及び個人も、当社の書面による許可を得ず、本文書の内容の一部もしくは全部を、いかなる視聴覚的、電子的、機械的、複写、録音等の手段によりもしくは形式により、伝搬又は複製をしてはなりません。

免責事項

当社は、GOWINSEMI Terms and Conditions of Sale (GOWINSEMI取引条件)に規定されている内容を除き、(明示的か又は黙示的かに拘わらず)いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。本文書における全ての情報は、予備的情報として取り扱われなければなりません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ (不具合情報)については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2021/06/29	1.0J	初版。
2021/12/23	1.1J	3 libusb プログラミングの説明を更新。

目次

目次.....	i
図一覧	ii
表一覧	iii
1 GWU2U の libusb + WinUSB ドライバー	1
1.1 Zadig を使用してドライバをインストール	1
2 ドライバのアンインストール	3
3 libusb プログラミングの説明.....	5
3.1 パラメータの構成.....	5
3.2 SW API の説明	6
3.3 libusb 関数ライブラリの開発プロセス	8
3.3.1 libusb の初期化と終了	8
3.3.2 指定された USB デバイスを開きます	8
3.3.3 インターフェース宣言	11
3.3.4 シリアルポートのパラメータの設定	11
3.3.5 同期データ送信	12
3.3.6 同期データ受信	12
3.3.7 非同期データ転送	13
4 プログラミング例	15
用語、略語	20
サポートされる OS	21
テクニカル・サポートとフィードバック	22

図一覧

図 1-1 List All Device	2
図 1-2 デバイスを選択	3
図 1-3 ドライバーを選択.....	3
図 2-1 デバイスマネージャーを開く	3
図 2-2 ドライバーのアンインストール.....	4

表一覧

表 A-1 用語、略語	20
表 A-2 サポートされる OS	21

1 GWU2U の libusb + WinUSB ドライバー

オープンソースの USB 関数ライブラリ **libusb** を使用して、GWU2U をプログラムすることができます。

Windows でこの関数ライブラリを使用してプログラムする場合、**Windows** での USB ドライバーの **WinUSB.sys** をインストールする必要があります。データの受送信や、ボーレート、パリティビット、ストップビット、ビット幅などのパラメータの設定を実現できます。

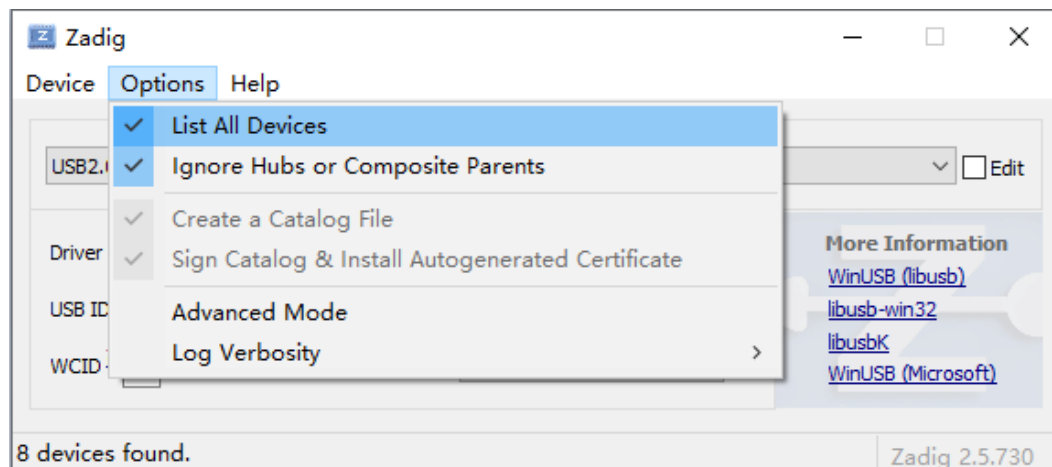
ドライバーをインストールするには、**Gowin** が提供するドライバー・インストール・ツールまたはオープンソースのドライバー・インストール・ツール **Zadig**(<https://zadig.akeo.ie/>)を使用できます。ドライバーをインストールするには、管理者権限が必要です。

Linux でこの関数ライブラリを使用してプログラムする場合、ドライバーをインストールする必要がなく、第 1 章と第 2 章を無視してください。

1.1 Zadig を使用してドライバーをインストール

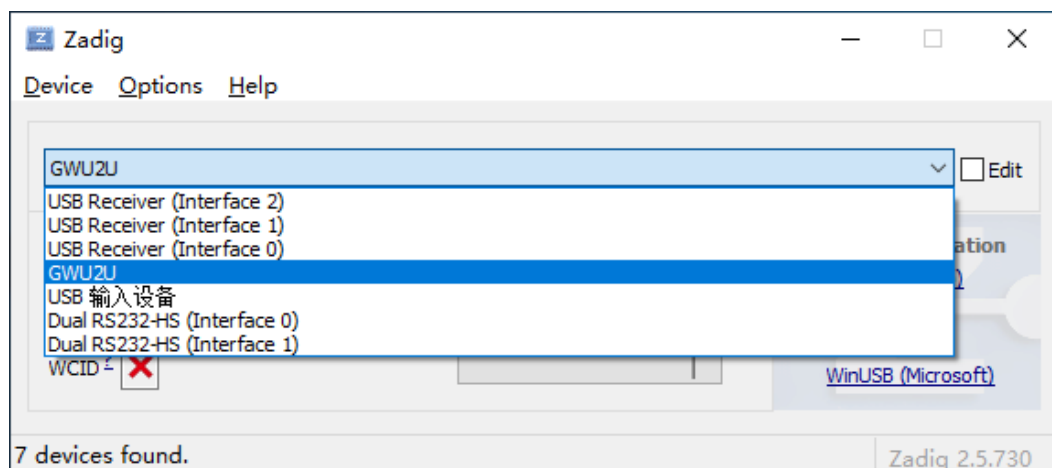
GWU2U デバイスをコンピュータの USB インターフェースに接続し、**Zadig** をダブルクリックして開き(管理者権限が必要)、**Options > List All Device** をチェックすると、コンピュータに接続されているすべての USB デバイスが一覧表示されます。

図 1-1 List All Device



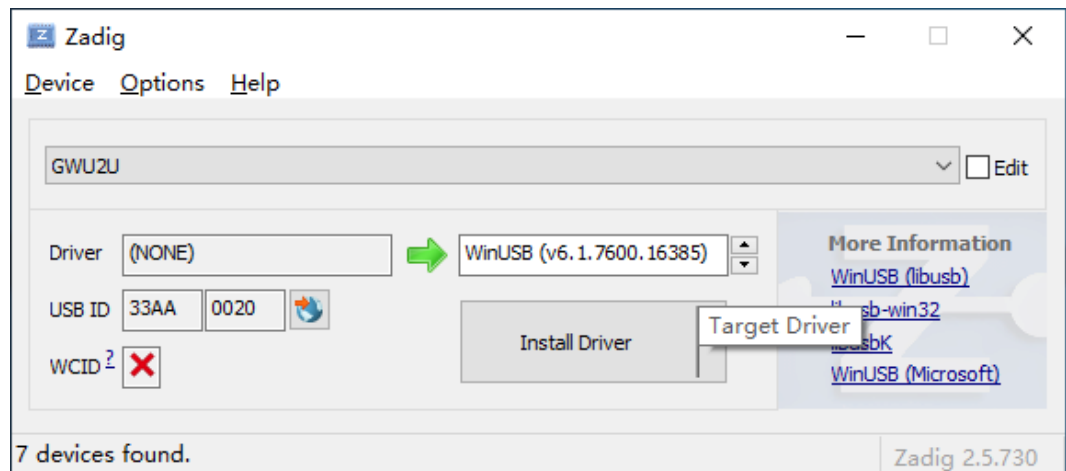
デバイス GWU2U を選択して、ドライバーをインストールします。

図 1-2 デバイスを選択



インストールするドライバーを選択します。libusb + WinUSB を使用する
場合、WinUSB を選択してください。

図 1-3 ドライバーを選択



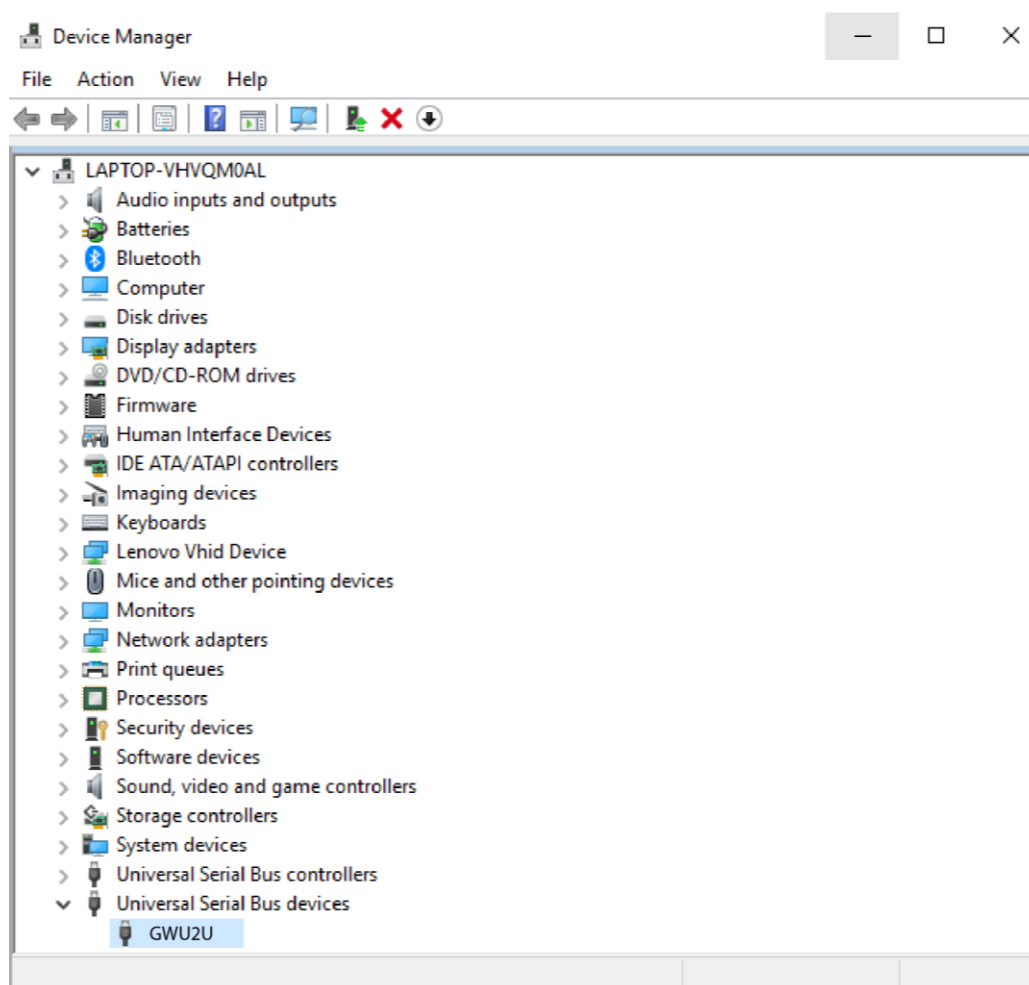
“Install Driver” ボタンをクリックして、ドライバーをインストールします。

なお、ドライバーが現在インストールされていない場合は“Install Driver”、他のドライバーがインストールされている場合は“Replace Driver”と表示されます。

2 ドライバーのアンインストール

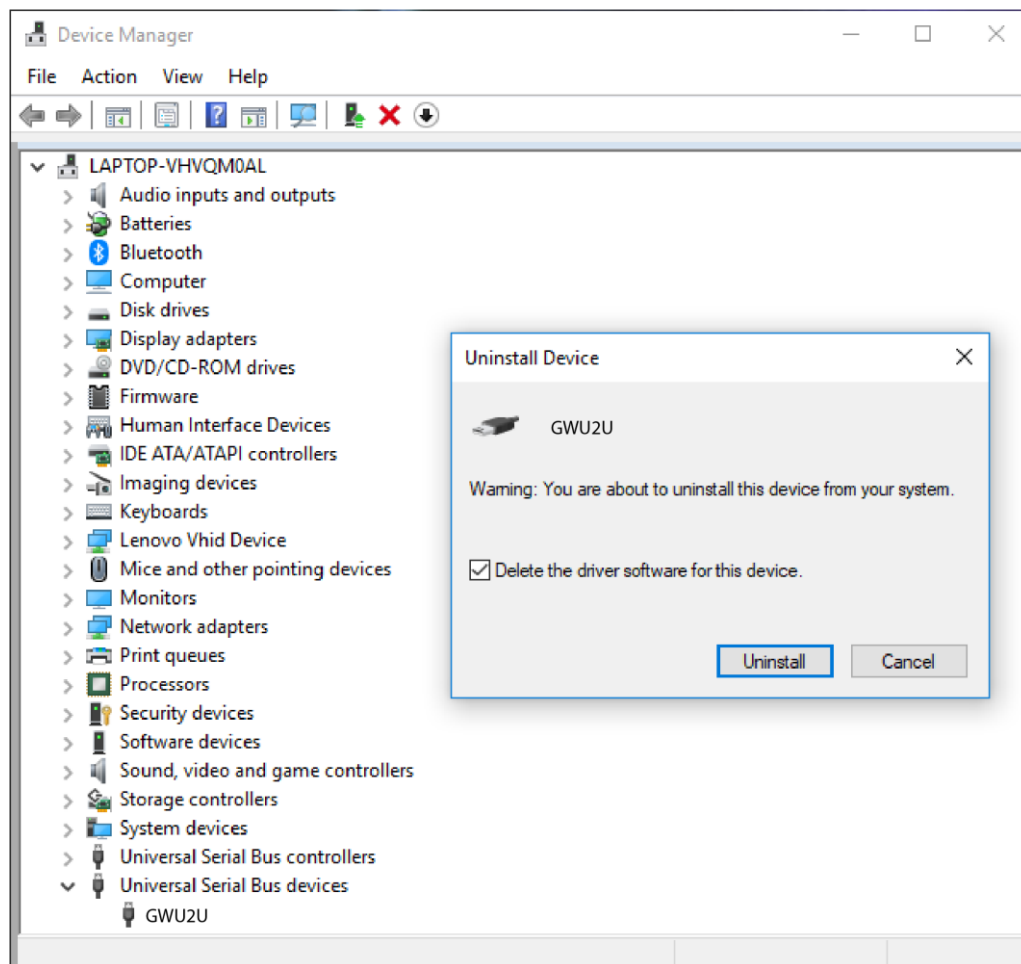
ドライバーをアンインストールするときは、GWU2U デバイスをコンピュータに接続し、Windows デバイスマネージャーを開いて、“ユニバーサル シリアル バス デバイス” リストで GWU2U デバイスを特定します。デバイス名を右クリックして、右クリックメニューの“削除”オプションを選択します。

図 2-1 デバイスマネージャーを開く



ポップアップダイアログボックスで、” このデバイスのドライバー ソフトウェアを削除する” にチェックを入れてアンインストールします。

図 2-2 ドライバーのアンインストール



3 libusb プログラミングの説明

libusb はオープンソースの USB 関数ライブラリです。

その公式ウェブサイトは <https://libusb.info> です。

そのソースコードは <https://github.com/libusb/libusb> から入手できます。

動的ライブラリと静的ライブラリを含む、コンパイル済みバージョン、公式 GCC バージョン、および VS バージョンは、公式 Web サイトからダウンロードできます。github からソースコードをダウンロードして、自分でコンパイルすることもできます。

libusb 関数宣言については、公式リファレンスを参照してください。

<http://libusb.sourceforge.net/api-1.0>

3.1 パラメータの構成

usb2uart_config_setting 構造体を介して UART パラメータを構成します。構造体は次のように定義されています。

```
typedef struct _usb2uart_config_setting {  
    unsigned char BaudRate;  
    unsigned char StopBits;  
    unsigned char Parity;  
    unsigned char DataBits;  
};
```

BaudRate : シリアル通信のボーレート

StopBits : シリアル通信のストップビット

StopBits の値の意味は次のとおりです :

```
#define STOPBITS_1    0
```

```
#define STOPBITS_1_5 1
```

```
#define STOPBITS_2 2
```

Parity : シリアル通信のパリティ

Parity の値の意味は次のとおりです :

```
#define GW_PARITY_NONE 0 //パリティなし
```

```
#define GW_PARITY_ODD 1 //奇数パリティ
```

```
#define GW_PARITY_EVEN 2 //偶数パリティ
```

```
#define GW_PARITY_MARK 3 //パリティビットは常に 1
```

```
#define GW_PARITY_SPACE 4 //パリティビットは常に 0
```

DataBits : シリアルデータビット

3.2 SW API の説明

```
int usb2uart_set_config(libusb_device_handle * devh,
usb2uart_config_setting * pusb2uart_config, unsigned char
interface, unsigned int uiTimeout);
```

UART パラメータ構成用構造体を使用して、**USB2UART** デバイスのパラメータを構成します。

パラメータ :

devh : USB デバイス操作ハンドル

pusb2uart_config : **pusb2iic_config** : UART パラメータ構成用構造体を指すポインター

interface : **GWU2U** デバイスのインターフェース

uiTimeout : タイムアウトパラメータ

返回值 :

実行が正しければ **0** を返し、エラーがあれば **0** 未満のエラーコードを返します。

```
int print_usb2uart_config(usb2uart_config_setting *
pusb2uart_config);
```

UART パラメータ構成用構造体の内容を印刷します。

パラメータ :

pusb2uart_config : **pusb2iic_config** : UART パラメータ構成用構造体を指すポインター

返り値：

実行が正しければ **0** を返し、エラーがあれば **0** 未満のエラーコードを返します。

```
int usb2uart_send_data(libusb_device_handle * devh, unsigned char
endpoint_out, int DataCnt, unsigned char * TransData, unsigned int
uiTimeout);
```

同期データ送信。

パラメータ：

devh：USB デバイス操作ハンドル

endpoint_out：GWU2U デバイスの出力エンドポイント

DataCnt：送信されるデータのバイト数

TransData：送信されるデータの格納アドレスを指すポインタ

uiTimeout：タイムアウトパラメータ

返り値：

実行が正しければ **0** を返し、エラーがあれば **0** 未満のエラーコードを返します。

```
int usb2uart_receive_data(libusb_device_handle * devh, unsigned
char endpoint_in, int * DataCnt, unsigned char * TransData, unsigned
int uiTimeout);
```

同期データ送信。

パラメータ：

devh：USB デバイス操作ハンドル

endpoint_in：GWU2U デバイスの入力エンドポイント

DataCnt：受信されるデータのバイト数の格納アドレスを指すポインタ

TransData：受信されるデータの格納アドレスを指すポインタ

uiTimeout：タイムアウトパラメータ

返り値：

実行が正しければ **0** を返し、エラーがあれば **0** 未満のエラーコードを返します。

3.3 libusb 関数ライブラリの開発プロセス

3.3.1 libusb の初期化と終了

libusb を使用してプログラムする場合は、関数 `libusb_init()` を呼び出して初期化する必要があります。使用後は、関数 `libusb_exit()` を呼び出して終了させる必要があります。

関数宣言は次のとおりです。

```
int libusb_init (libusb_context ** context)

void libusb_exit (libusb_context * ctx)
```

パラメータ `libusb_context` は `libusb` コンテキスト構造体であり、`libusb` のいくつかの構成パラメータを保存するために使用されます。`libusb_context` が指定されていない場合、デフォルトのコンテキスト構造が作成されます。コンテキスト構造がすでに存在する場合、その構造は再初期化せずに直接使用されます。

プログラミング例：

```
int rc = libusb_init(NULL);

if (rc != LIBUSB_SUCCESS)

{

    return rc;

}

libusb_exit(NULL);
```

3.3.2 指定された USB デバイスを開きます

`libusb_open_device_with_vid_pid()` 関数を使用して、Vendor ID および Product ID に従って指定されたデバイスを開くことができます。`libusb_get_device_list()` 関数を使用してすべての USB デバイスを取得し、それらから必要なデバイスを選択し、`libusb_open()` 関数を使用して後続の操作のためにデバイスのハンドルを取得することもできます。関数の宣言は次のとおりです。

(1) Vendor ID および Product ID を使用してデバイスを開きます。

```
libusb_device_handle* libusb_open_device_with_vid_pid(

    libusb_context * ctx,

    uint16_t vendor_id,
```

```
uint16_t product_id
```

```
)
```

その中で、パラメータ **ctx** は、**libusb** が初期化されるときに生成されるコンテキスト構造のアドレスです。デフォルトのコンテキストが使用される場合、**NULL** が使用されます。**vendor_id** と **product_id** は、それぞれ USB デバイスの Vendor ID および Product ID です。Gowin USB デバイスの Vendor ID は **0x33aa** であり、GWU2U デバイスの Product ID は **0x0020** です。

返り値は、このコンピュータで **libusb** によって最初に一致したデバイスの操作ハンドルポインターです。それ以外の場合は、**NULL** ポインタが返されます。

使用例は次のとおりです。

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

(2) すべての **USB** デバイスを取得した後、指定したデバイスを選択します。

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

その中で、パラメータ **ctx** は、**libusb** が初期化されるときに生成されるコンテキスト構造のアドレスです。デフォルトのコンテキストが使用される場合、**NULL** が使用されます。**list** はデバイスリストへのポインタです。使用後、**libusb_free_device_list()**関数を使用してそのメモリを解放する必要があります。

関数が正しく実行された場合、返り値はデバイスの数であり、見つかったデバイスのリストが **list** に保存されます。それ以外の場合は、ゼロ未満の

libusb_error 値が返されます。

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

その中で、パラメータ **dev** はデバイスリスト内のデバイスであり、**dev_handle** は返されるデバイスハンドルへのポインタのアドレスです。

実行が成功した場合、**LIBUSB_SUCCESS** が返されます。成功しなかった場合、ゼロ未満の **libusb_error** 値が返されます。

使用例は次のとおりです。

```
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);
    }
}
```

3.3.3 インターフェース宣言

USB デバイスには通常、1 つ以上のインターフェースが含まれています。libusb がインターフェースを使用する場合、最初にインターフェースを宣言 (claim interface) する必要があります。宣言が成功すると、インターフェースが正常に開かれ、インターフェースに含まれるエンドポイントに対して受送信できることになります。

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

その中で、パラメータ **dev_handle** はデバイスハンドルであり、**interface_number** はインターフェース番号です。GWU2U デバイスでは、インターフェース番号は **0** です。インターフェースの宣言が成功した場合は **LIBUSB_SUCCESS** が返され、そうでない場合はゼロ未満の **libusb_error** 値が返されます。

プログラミング例：

```
rc = libusb_claim_interface(devh, 0);

if (rc != LIBUSB_SUCCESS)
{
    printf("Error claiming interface: %s\n",
libusb_error_name(rc));
    goto out;
}

else
{
    printf("Claiming interface\n");
}
```

3.3.4 シリアルポートのパラメータの設定

GWU2U シリアルデバイスでは、ボーレート、パリティビット、ストップビットを設定するには、SW API **usb2uart_set_config()**関数を使用してください。

プログラミング例：

```
usb2uart_config.BaudRate = 115200;

usb2uart_config.Parity = GW_PARITY_NONE;

usb2uart_config.StopBits = STOPBITS_1;

usb2uart_config.DataBits = 8;

rc = usb2uart_set_config(devh, &usb2uart_config, UART_INTERFACE,
0);

if(rc != 0)
{
    printf("Error : %s", libusb_strerror(rc));
    return -1;
}
```

3.3.5 同期データ送信

GWU2U 同期データ送信を実行するには、SW API `usb2uart_send_data()` 関数を使用してください。

プログラミング例：

```
rc = usb2uart_send_data(devh, ENDPOINT_OUT, sizeof(tx_data),
tx_data, 1000);

if(rc != 0)
{
    printf("Error: %s", libusb_strerror(rc));
}

else
{
    printf("GWU2U device sends %d bytes data, return code: %d\n",
(int)(sizeof(tx_data)), rc);
}
```

3.3.6 同期データ受信

GWU2U 同期データ受信を実行するには、SW API `usb2uart_receive_data()` 関数を使用してください。

プログラミング例：

```
rc = usb2uart_receive_data(devh, ENDPOINT_IN, &size, rx_data,
1000);

if(rc != 0)

{

    printf("Error: %s", libusb_strerror(rc));

}

else

{

    printf("GWU2U device receives %d bytes data, return code: %d¥n",
size, rc);

    size = 0;

}
```

3.3.7 非同期データ転送

libusb は非同期データ転送を実現できます。

その実装方法は以下のとおりです。

- (1) libusb_alloc_transfer()関数を使用して転送を割り当てます。
- (2) libusb_fill_bulk_transfer()関数を使用して、転送コンテンツを入力し、コールバック関数を登録します。
- (3) libusb_submit_transfer()関数を使用して、転送要求を送信します。
- (4) 転送が完了すると、コールバック関数が自動的に呼び出され、転送結果が処理されます。
- (5) 転送が終了したら、libusb_free_transfer()関数を使用してメモリを解放する必要があります。

非同期データ受信を例に紹介します。

```
static uint8_t buf[PACK_SIZE];

static struct libusb_transfer *xfr; //転送を定義する構造

xfr = libusb_alloc_transfer(0); //転送を割り当てます。バルク転送の場合、パラメータは0です。

if (!xfr)

{

    return -ENOMEM;
```

```
    }

    libusb_fill_bulk_transfer(
        xfr, //転送定義構造
        devh, //デバイスハンドル
        ep, //転送エンドポイント
        buf, //データを受信するバッファ
        sizeof(buf), //受信 BUFFER の長さ
        cb_xfr_in, //登録されたコールバック関数
        NULL, //コールバック関数に渡されるデータへのポインタ。void 型です。
        0 //タイムアウト定義(単位はミリ秒)
    );

    libusb_submit_transfer(xfr);

    while(1)
    {
        rc = libusb_handle_events(NULL); //保留中のイベントを処理します
    }
```

4 プログラミング例

以下は、簡単な **GWU2U** シリアルプログラミングの例です。

最初に同期送信モードを使用してデータを送信し、次に非同期受信モードに入り、受信したデータを文字形式でコンソールに出力します。

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include "libusb.h"

#include "gw_usb2uart.h"

static struct libusb_device_handle *devh = NULL;

unsigned char tx_data[] = "This is a usb2uart test string sent
through usb¥r¥n" ;

unsigned int rd_cnt = 0;

usb2uart_config_setting usb2uart_config;

//非同期受信用のコールバック関数。データを受信すると、文字列の形式でコンソールに出力します。

void LIBUSB_CALL cb_xfr_in(struct libusb_transfer *xfr)
{
    int i;

    rd_cnt++;

    if (xfr->status != LIBUSB_TRANSFER_COMPLETED)
    {
        fprintf(stderr, "transfer status %d¥n", xfr->status);

        libusb_free_transfer(xfr);
    }
}
```

```
        exit(3);
    }

    for (i = 0; i < xfr->actual_length; i++)
    {
        printf("%c", xfr->buffer[i]);
    }

    // 今回データを受信した後、送信要求を再送信し、次の受信イベントを待ちます。
    // 送信に失敗した場合、メモリは解放されます。

    if(libusb_submit_transfer(xfr) < 0)
    {
        libusb_free_transfer(xfr);
    }
}

static int benchmark_in(uint8_t ep)
{
    static uint8_t buf[PACK_SIZE];
    static struct libusb_transfer *xfr;

    printf("(SUBMIT FUNC) Starting a new asynchronous bulk
transfer¥n");

    printf("Receive data :¥n");
    xfr = libusb_alloc_transfer(0);
    if (!xfr)
    {
        return -ENOMEM;
    }

    libusb_fill_bulk_transfer (
        xfr,
        devh,
        ep,
        buf,
        sizeof(buf),
```

```
        cb_xfr_in,
        NULL,
        0
    );
    return libusb_submit_transfer(xfr);
}

int main(int argc, char *argv[])
{
    int rc = 0;
    //Initializes libusb
    rc = libusb_init(NULL);
    if (rc != LIBUSB_SUCCESS)
    {
        return rc;
    }

    //Open USB device
    devh = libusb_open_device_with_vid_pid(NULL, VENDOR_ID,
PRODUCT_ID);

    if(NULL == devh)
    {
        printf("Open USB device failed\n");
        goto out;
    }
    else
    {
        printf( "Open USB device\n" );
    }

    rc = libusb_claim_interface(devh, UART_INTERFACE);
    if (rc != LIBUSB_SUCCESS)
    {
        printf("Error claiming interface: %s\n",
```



```
libusb_error_name(rc));
    goto out;
}
else
{
    printf( "Claiming interface¥n" );
}

printf("DEVICE READY...¥n¥n");
//Configuration settings
usb2uart_config.BaudRate  = 115200;
usb2uart_config.Parity    = 0;
usb2uart_config.StopBits  = 0;
usb2uart_config.DataBits  = 8;
//Set configurations
rc = usb2uart_send_data(devh, &usb2uart_config,
UART_INTERFACE, 0);
if(rc != 0)
{
    printf("Error: %s", libusb_strerror(rc));
    return -1;
}
//Send data synchronously
rc = usb2uart_send_data(devh, ENDPOINT_OUT, sizeof(tx_data),
tx_data, 1000);
if(rc != 0)
{
    printf("Error: %s", libusb_strerror(rc));
}
else
{
    printf("GWU2U device sends %d bytes, return code: %d¥n",
(int)(sizeof(tx_data)), rc);
}
```

```
    }

    benchmark_in(ENDPOINT_IN);
    while(1)
    {
        rc = libusb_handle_events(NULL);
    }

    libusb_release_interface(devh, UART_INTERFACE);
out:
    if(devh)
    {
        libusb_close(devh);
    }
    libusb_exit(NULL);
    return 0;
}
```

用語、略語

表 A-1 に、本マニュアルで使用される用語、略語、及びその意味を示します。

表 A-1 用語、略語

用語、略語	正式名称	意味
USB	Universal Serial Bus	ユニバーサル・シリアル・バス
UART	Universal Asynchronous Receiver/Transmitter	汎用非同期送受信回路

サポートされる OS

表 A-2 に GWU2U SW API をサポートする OS を示します。

表 A-2 サポートされる OS

Windows	Windows 7/10(32bit/64bit)
Linux	Ubuntu 20.04 LTS

テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

Web サイト : www.gowinsemi.com

E-mail : support@gowinsemi.com

