

GWU2X プログラミング ガイド_U2X_JTAG

UG1003-1.0J, 2021-06-16

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
2021/06/16	1.0J	初版。

目次

目次.....	i
図一覧	ii
表一覧	iii
1 機能の紹介	1
2 ドライバーのインストールとアンインストール.....	2
2.1 Zadig を使用してドライバーをインストール	2
2.2 ドライバーのアンインストール	4
3 プログラミングの説明	6
3.1 API 関数.....	6
3.1.1 JTAG ステータスの文字列の取得	6
3.1.2 関数実行のエラーメッセージの文字列の取得	6
3.1.3 TMS 信号列の生成	6
3.1.4 TCK 周波数の設定.....	7
3.1.5 IO ポートの設定	7
3.1.6 JTAG ステータスのリセット	8
3.1.7 JTAG ステータスの IDLE へのジャンプ	9
3.1.8 IR データの送信	10
3.1.9 DR データの送信.....	11
3.1.10 TDO データのリードバック	12
3.1.11 DR データを送信するための命令(送信しない)の生成	12
3.2 プログラミング例.....	13
3.3 エラーコード	14
用語、略語.....	16
テクニカル・サポートとフィードバック	17

図一覧

図 2-1 “List All Device”オプションを選択.....	2
図 2-2 デバイスを選択	3
図 2-3 ドライバーを選択.....	3
図 2-4 デバイスマネージャーを開く	4
図 2-5 ドライバーのアンインストール.....	5

表一覧

表 3-1 エラーコード一覧.....	14
表 A-1 用語、略語	16

1 機能の紹介

GWU2X は、USB - JTAG Master モードを実装し、JTAG デバイスの読み出しと書き込みに使用できます。その TCK クロック周波数は構成可能であり、最大は 15MHz(現在のテスト結果)です。任意のビット長の IR/DR データを送信または読み出すことができます。

2 ドライバーのインストールとアンインストール

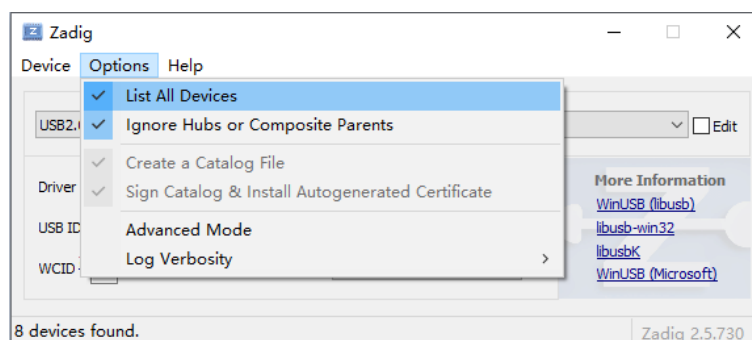
オープンソースの USB 関数ライブラリ **libusb** を使用して、**GWU2X_JTAG** をプログラミングすることができます。この関数ライブラリを使用してプログラミングする場合は、**WINDOWS** での USB ドライバーの **WinUSB.sys** をインストールする必要があります。

ドライバーをインストールするには、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。

2.1 Zadig を使用してドライバーをインストール

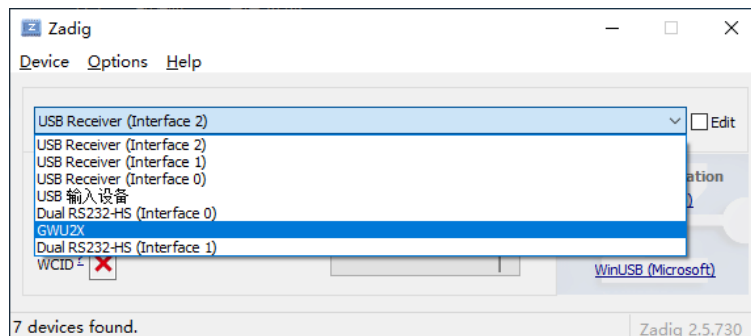
GWU2X デバイスをコンピュータの USB インターフェースに接続し、Zadig をダブルクリックして開き(管理者権限が必要)、**Options > List All Device** をチェックすると、コンピュータに接続されているすべての USB デバイスが一覧表示されます。

図 2-1 “List All Device” オプションを選択



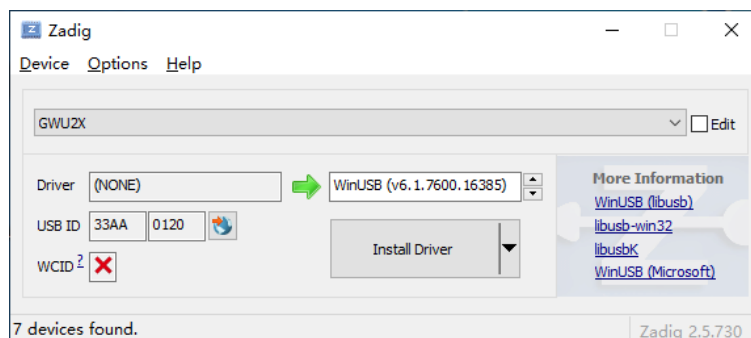
デバイス **GWU2X** を選択して、ドライバーをインストールします。

図 2-2 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 2-3 ドライバーを選択



“Install Driver”^[1] 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。

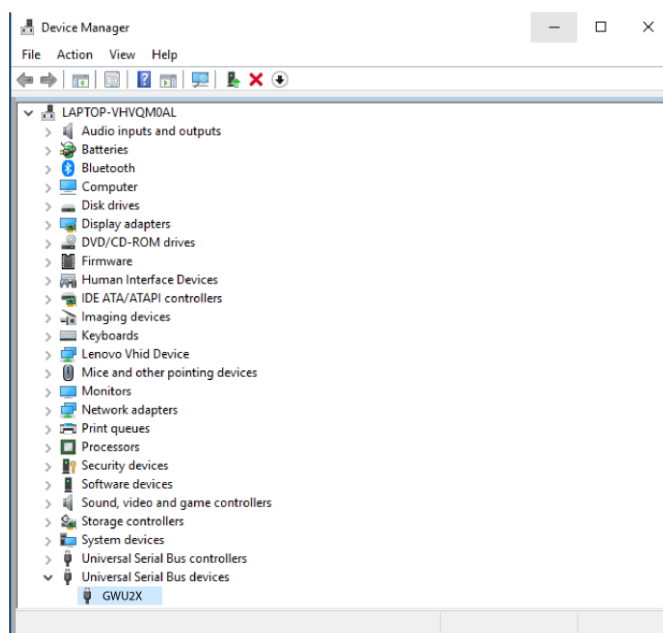
注記：

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。

2.2 ドライバーのアンインストール

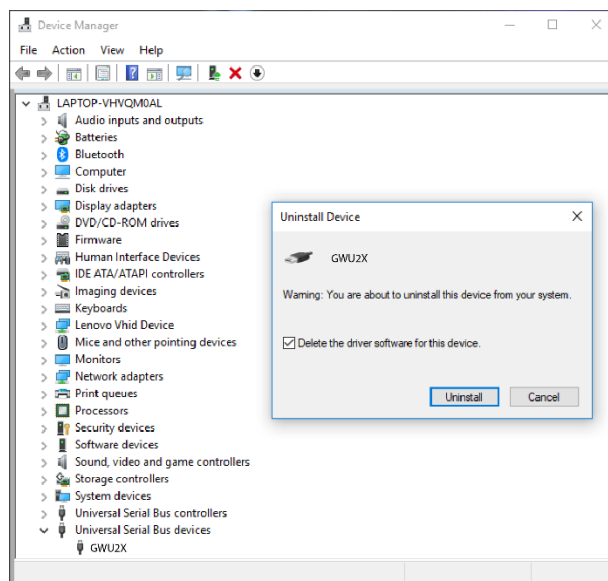
ドライバーをアンインストールするときは、GWU2X デバイスをコンピュータに接続し、**Windows** デバイスマネージャーを開いて、“ユニバーサル シリアル バス デバイス” リストで **GWU2X** デバイスを特定します。デバイス名を右クリックして、右クリックメニューの“削除”オプションを選択します。

図 2-4 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 2-5 ドライバーのアンインストール



3 プログラミングの説明

3.1 API 関数

3.1.1 JTAG ステータスの文字列の取得

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

パラメータ :

iJtagStateCode : JTAG ステータスを表す変数。変数の定義については、**gw_usb2jtag.h** で定義されている列挙値 **JTAG_STATE** を参照してください。

戻り値 :

デバッグ用の情報を出力するために使用できる、**JTAG** ステータスを説明する文字列のアドレスを指す **const char** ポインタ。

3.1.2 関数実行のエラーメッセージの文字列の取得

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

パラメータ :

errCode : API 関数によって返されるエラーコード。**0** は、実行が成功したことを意味し、負の値は、関数の実行にエラーがあることを意味します。

戻り値 :

デバッグ用の情報を出力するために使用できる、エラーメッセージを説明する文字列のアドレスを指す **const char** ポインタ。

3.1.3 TMS 信号列の生成

```
int jtag_generate_tms_array(  
  
int *pJtagState,
```

```
int iTargetState,  
  
int *pTmsBitCnt,  
  
unsigned int *pTmsBitArray);
```

パラメータ：

pJtagState：現在の JTAG ステータスを一時的に保存する変数を指すポインタ。実行が成功すると、この変数は TMS 信号が送信された後の JTAG ステータスに更新されます。

iTargetState：TMS 信号を送信して JTAG デバイスをジャンプさせるターゲット JTAG ステータス。

pTmsBitCnt：生成された TMS 信号列の長さが格納される変数を指すポインタ。

pTmsBitArray：生成された TMS 信号列の値が格納される変数を指すポインタ。

返回值：

パラメータの設定が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.4 TCK 周波数の設定

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

パラメータ：

devh：USB2JTAG デバイスを操作するために使用される libusb の USB デバイスハンドル。

uiFreqKiloHz：設定する TCK 周波数(KHz)。

uiTimeOut：タイムアウトパラメータ。単位はミリ秒です。0 に設定すると、無期限に待機することになります。

返回值：

パラメータの設定が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.5 IO ポートの設定

```
int usb2jtag_ioreset(  
  

```

```
libusb_device_handle *devh,

unsigned short usIODir,

unsigned short usIOLevelSet,

unsigned int uiTimeout);
```

パラメータ :

devh : USB2JTAG デバイスを操作するために使用される libusb の USB デバイスハンドル。

uiIODir : IO ポートの方向の設定。JTAG の TCK/TMS/TDI を出力モードに設定し、TDO を入力モードに設定する必要があります。そのパラメータの定義については、**gw_usb2jtag.h** のマクロ定義を参照してください。

uiIOLevelSet : アイドル時の IO ポートのレベル(High または Low)を設定します。通常、アイドル時は TMS/TDI を High に設定し、TCK/TDO を Low に設定する必要があります。そのパラメータの定義については、**gw_usb2jtag.h** のマクロ定義を参照してください。

uiTimeout : タイムアウトパラメータ。単位はミリ秒です。0 に設定すると、無期限に待機することになります。

返り値 :

パラメータの設定が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

例 :

```
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 ステータスのリセット

この関数が実行されると、JTAG は一連の High レベルの TMS 信号を送信して、JTAG デバイスのステータスを TEST LOGIC RESET にジャンプさせます。

```
int usb2jtag_reset(

libusb_device_handle *devh,

int *pJtagCurrentState,

unsigned int uiTimeout);
```

パラメータ :

devh : USB2JTAG デバイスを操作するために使用される libusb の USB

デバイスハンドル。

pJtagCurrentState : 現在の JTAG ステータスを格納する変数を指すポインタ。関数が実行されると、この変数は **TEST_LOGIC_RESET(0x0)** になります。

uiTimeout : タイムアウトパラメータ。単位はミリ秒です。0 に設定すると、無期限に待機することになります。

返回值 :

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.7 JTAG ステータスの IDLE へのジャンプ

この関数が実行されると、JTAG は TMS 信号を送信して、JTAG デバイスのステータスを **RUN TEST IDLE** にジャンプさせます。

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

パラメータ :

devh : USB2JTAG デバイスを操作するために使用される libusb の USB デバイスハンドル。

pJtagCurrentState : 現在の JTAG ステータスを格納する変数を指すポインタ。関数が実行されると、この変数は **RUN TEST IDLE(0x0)** になります。

usIdleTckLen : **RUN TEST IDLE** にジャンプした後、指定された長さの TCK を生成し続けます。必要がない場合は、このパラメータを 0 として渡します。

uiTimeout : タイムアウトパラメータ。単位はミリ秒です。0 に設定すると、無期限に待機することになります。

返回值 :

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.8 IR データの送信

この関数を実行するときは、まず **TMS** 信号を送信して **JTAG** ステータスを **SHIFT IR** にジャンプさせ、次に指定した **IR** データを送信します。送信が完了したら、**TMS** 信号を送信して **JTAG** ステータスを **RUN TEST IDLE** にジャンプさせます。必要があれば、指定されたサイクル数の **TCK** 信号を送信し続けます。

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

パラメータ：

devh : **USB2JTAG** デバイスを操作するために使用される **libusb** の **USB** デバイスハンドル。

pJtagCurrentState : 現在の **JTAG** ステータスを格納する変数を指すポインタ。関数が実行されると、この変数は **RUN TEST IDLE(0x0)** になります。

iTdiBitCnt : **TDI** によって送信される **IR** データのビット長。

ucTdiData : **TDI** によって送信される **IR** データを指すポインタ。

ucTdoReadBack : **TDO** リードバックデータが保存されるアドレスを指すポインタ。**TDO** リードバックデータを保存する必要がない場合、このパラメータは **NULL** として渡されます。

usIdleTckLen : **RUN TEST IDLE** にジャンプした後、指定された長さの **TCK** を生成し続けます。必要がない場合は、このパラメータを **0** として渡します。

uiTimeout : タイムアウトパラメータ。単位はミリ秒です。**0** に設定すると、無期限に待機することになります。

返回值：

実行が成功した場合、**0** が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.9 DR データの送信

この関数を実行するときは、まず **TMS** 信号を送信して **JTAG** ステータスを **SHIFT DR** にジャンプさせ、次に指定した **DR** データを送信します。送信が完了したら、**TMS** 信号を送信して **JTAG** ステータスを **RUN TEST IDLE** にジャンプさせます。必要があれば、指定されたサイクル数の **TCK** 信号を送信し続けます。

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

パラメータ：

devh : **USB2JTAG** デバイスを操作するために使用される **libusb** の **USB** デバイスハンドル。

pJtagCurrentState : 現在の **JTAG** ステータスを格納する変数を指すポインタ。関数が実行されると、この変数は **RUN TEST IDLE(0x0)** になります。

iTdiBitCnt : **TDI** によって送信される **DR** データのビット長。

ucTdiData : **TDI** によって送信される **DR** データを指すポインタ。

ucTdoReadBack : **TDO** リードバックデータが保存されるアドレスを指すポインタ。**TDO** リードバックデータを保存する必要がない場合、このパラメータは **NULL** として渡されます。

usIdleTckLen : **RUN TEST IDLE** にジャンプした後、指定された長さの **TCK** を生成し続けます。必要がない場合は、このパラメータを **0** として渡します。

uiTimeout : タイムアウトパラメータ。単位はミリ秒です。**0** に設定すると、無期限に待機することになります。

返回值：

実行が成功した場合、**0** が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.10 TDO データのリードバック

この関数を実行すると、現在一時的に保存されている TDO データがリードバックされます。

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

パラメータ：

devh：USB2JTAG デバイスを操作するために使用される libusb の USB デバイスハンドル。

iBitsToRead：リードバックする必要がある TDO データのビット長。

ucTdoReadBack：TDO リードバックデータが保存されているアドレスを指すポインタ。

uiTimeout：タイムアウトパラメータ。単位はミリ秒です。0 に設定すると、無期限に待機することになります。

返り値：

実行が成功した場合、0 が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.1.11 DR データを送信するための命令(送信しない)の生成

この関数を実行すると、入力パラメータに従って DR データを送信する命令が生成されますが、送信されません。複数の DR データ送信命令のグループを生成して連続したアドレスに保存し、libusb の libusb_bulk_transfer 関数を使用して一度に送信すると、送信効率を向上させることができます。

```
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);
```

パラメータ：

pJtagCurrentState：現在の JTAG ステータスを格納する変数を指すポインタ。関数が実行されると、この変数は **RUN_TEST_IDLE** になります。

iTdiBitCnt：TDI によって送信される DR データのビット長。

ucTdiData：TDI によって送信される DR データを指すポインタ。

usIdleTckLen：IDLE にジャンプした後、指定された長さの TCK を生成し続けます。必要がない場合は、このパラメータを **0** として渡します。

uclTdoRdbk：TDO リードバックデータを保存するかどうかを設定します。保存する必要がある場合、このパラメータは **1** として渡されます。それ以外の場合、このパラメータは **0** として渡されます。

ucCmdMem：生成命令が保存されるアドレスを指すポインタ。

piCmdMemByteLeft：生成命令が保存されたアドレスの残りのバイト数を示す変数を指すポインタ。この関数が正常に実行された後、変数の値は自動的に更新されます。

piCmdByteCnt：今回関数によって生成された命令のバイト長を示す変数を指すポインタ。

返り値：

実行が成功した場合、**0** が返されます。成功しなかった場合、ゼロ未満のエラーコードが返されます。

3.2 プログラミング例

```
#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);

```

3.3 エラーコード

API 関数が正常に実行されている場合、返り値は **0** です。それ以外の場合は、負の数を返します。

0 以外の返り値で表されるエラーの意味を次の表に示します。

表 3-1 エラーコード一覧

Value	Enumerator	
0	SUCCESS	エラーなし
-1	USB_ERROR_IO	USB 入出力エラー
-2	USB_ERROR_INVALID_PARAM	USB パラメータエラー
-3	USB_ERROR_ACCESS	デバイスにアクセスするための権限が不十分です
-4	USB_ERROR_NO_DEVICE	USB デバイスが見つかりません(デバイスが切断されています)
-5	USB_ERROR_NOT_FOUND	エンティティ(Entity)が見つかりません
-6	USB_ERROR_BUSY	USB デバイスがビジーです
-7	USB_ERROR_TIMEOUT	タイムアウト
-8	USB_ERROR_OVERFLOW	メモリオーバーフロー
-9	USB_ERROR_PIPE	パイプエラー
-10	USB_ERROR_INTERRUPTED	システム関数が中断されました

Value	Enumerator	
-11	USB_ERROR_NO_MEM	メモリ不足
-12	USB_ERROR_NOT_SUPPORTED	この操作は現在のプラットフォームではサポートされていません
-13	U2J_ERROR_USBTRANS_ERR	USB データ転送エラー
-14	U2J_ERROR_INVALID_PARAM	U2X_JTAG パラメータ値無効
-15	U2J_ERROR_TIMEOUT	U2X_JTAG 転送タイムアウト
-16	U2J_ERROR_CMD_ERR	U2X_JTAG 命令エラー
-99	ERROR_OTHER	その他のエラー

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
JTAG	Joint Test Action Group	ジョイント・テスト・アクション・グループ

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

