Ftd2xxドライバー説明

はじめに

このドライバー説明書はFTDI社の資料を基に青色の部分を弊社が和訳したものです。
無断転記及び他への流用は御遠慮お願い致します。

（株）西日本常盤商行

Introduction

An FTD2XX device is an FT8U232 or FT8U245 running with FTDI's direct driver FTD2XX.SYS. Thedevice has a programming interface exposed by the dynamic link library FTD2XX.DLL, and this documentdescribes that interface.

はじめに

ＦＴ８Ｕ２３２やＦＴ８Ｕ２４５は、デバイスドライバーＦＴＤ２ＸＸ．ＳＹＳで制御します。
そのプログラムインターフェースのＤＬＬライブラリーであるＦＴＤ２ＸＸ．ＤＬＬの解説書です。

Overview

Before the device can be accessed, it must first be opened. **FT_Open** and **FT_OpenEx** return a handlewhich is used by all functions in the programming interface to identifiy the device. When the device hasbeen opened successfully, I/O can be performed using **FT_Read** and **FT_Write**. When operations arecomplete, the device is closed using **FT_Close**.

概要

デバイスにアクセスする前に、FT_OpenかFT_OpenEx関数を使って
デバイスをオープンしてデバイス固有のハンドル値を得なければなりません。デバイスが正常にオープンできると　FT_ReadやFT_Writeを使って、デバイスを制御することが出来ます。制御が終わると、FT_Closeを使ってデバイスをクローズします。

Functions are available to reset the device **(FT_ResetDevice)**; purge receive and transmit buffers **(FT_Purge)**; set receive and transmit timeouts **(FT_SetTimeouts)**; get receive queue status **(FT_GetQueueStatus)**; get device status **(FT_GetStatus)**; set and reset break condition **(FT_SetBreakOn, FT_SetBreakOff)**; set conditions for event notification **(FT_SetEventNotification)**.

この他に以下の関数も用意されています

FT_ResetDevice	デバイスをリセット
FT_Purge	デバイスの送受信バッファをクリア
FT_SetTimeouts	デバイスの送受信タイムアウト時間を設定
FT_GetQueueStatus	デバイスのキューの状態を得る
FT_GetStatus	デバイスの状態を得る
FT_SetBreakOn	ブレークの状態をｏｎ／ｏｆｆ
FT_SetBreakOff	

FT_SetEventNotification　通知するデバイスのイベントの条件を設定
FT_ListDevices　現在接続されているデバイスの情報を得る

For FT8U232 devices, functions are available to set the baud rate **(FT_SetBaudRat**e), and set a non-standard baud rate **(FT_SetDivisor**); set the data characteristics such as word length, stop bits and parity**(FT_SetDataCharacteristic**s); set hardware or software handshaking **(FT_SetFlowControl**); set modem control signals **(FT_SetDTR**, **FT_ClrDTR**, **FT_SetRTS**, **FT_ClrRTS**); get modem status **(FT_GetModemStatus**); set special characters such as event and error characters **(FT_SetChars**).

ＦＴ８Ｕ２３２の制御用に以下の関数も用意されています
FT_SetBaudRate　ボーレートの設定
FT_SetDivisor　非標準ボーレートの設定
FT_SetDataCharacteristics　キャラクターのワード長、ストップビット・パリティの設定
FT_SetFlowControl　ハードフローやソフトフローのハンドシェークの設定
FT_SetDTR　モデム制御線のｏｎ／ｏｆｆ
FT_ClrDTR,
FT_SetRTS
FT_ClrRTS
FT_GetModemStatus　モデムステータスを得る
FT_SetChars　エラーやイベントの特別な制御キャラクターを設定

Reference

The functions which comprise the FTD2XX programming interface are described in this section. Excerpts from the header file FTD2XX.H are included to explain any references in the descriptions of the functions below.

リファレンス
　ＦＴＤ２ＸＸのプログラミング・インターフェースをこの章で解説します
以下の参考プログラムにはヘッダーファイルのＦＴＤ２ＸＸ．Ｈをインクルードしてください。

FT_HANDLE

```
typedef DWORD FT_HANDLE
```

FT_STATUS

```
FT_OK = 0
FT_INVALID_HANDLE = 1              // 無効なハンドル
FT_DEVICE_NOT_FOUND = 2            // デバイスが見つからない
FT_DEVICE_NOT_OPENED = 3           // デバイスがオープンされていない
FT_IO_ERROR = 4                    // Ｉ／Ｏエラー
```

```
FT_INSUFFICIENT_RESOURCES = 5       // リソースが不足
FT_INVALID_PARAMETER = 6            // 無効なパラメータ
```

Flags (see FT_OpenEx)　FT_OpenEx　フラグ

```
FT_OPEN_BY_SERIAL_NUMBER = 1        // デバイスのシリアル番号でオープン
FT_OPEN_BY_DESCRIPTION = 2          // デバイスの製品情報でオープン
```

Flags (see FT_ListDevices)　FT_ListDevices　フラグ

```
FT_LIST_NUMBER_ONLY = 0x80000000    // 番号リストのみ
FT_LIST_BY_INDEX = 0x40000000       // インデックスによるリスト
FT_LIST_ALL = 0x20000000            // 全て表示
```

Word Length (see FT_SetDataCharacteristics)　キャラクター長

```
FT_BITS_8 = 8                       // キャラクター　８ビット
FT_BITS_7 = 7                       // キャラクター　７ビット
```

Stop Bits (see FT_SetDataCharacteristics)　ストップビット

```
FT_STOP_BITS_1 = 0                  // ストップビット　１ビット
FT_STOP_BITS_2 = 2                  // ストップビット　２ビット
```

Parity (see FT_SetDataCharacteristics)　パリティ

```
FT_PARITY_NONE = 0                  // パリティ無し
FT_PARITY_ODD = 1                   // 奇数パリティ
FT_PARITY_EVEN = 2                  // 偶数パリティ
FT_PARITY_MARK = 3                  // マーク
FT_PARITY_SPACE = 4                 // スペース
```

Flow Control (see FT_SetFlowControl)　フロー制御

```
FT_FLOW_NONE = 0x0000               // フロー制御無し
FT_FLOW_RTS_CTS = 0x0100            // RTS/CTS 制御
FT_FLOW_DTR_DSR = 0x0200            // DTR/DSR 制御
FT_FLOW_XON_XOFF = 0x0400           // XON/XOFF 制御
```

Purge RX and TX buffers (see FT_Purge)

```
FT_PURGE_RX = 1                     // 受信バッファをクリア
FT_PURGE_TX = 2                     // 送信バッファをクリア
```

Notification Events (see FT_SetEventNotification)　イベント通知

```
FT_EVENT_RXCHAR = 1                 // キャラクターを受信
FT_EVENT_MODEM_STATUS = 2           // モデムステータス
```

FT_ListDevices

Description

Get information concerning the devices currently connected. This function can return such information as the number of devices connected, and device strings such as serial number and product description.

説明

現在接続されているデバイスの情報を得ます。この関数は接続されているデバイスの数やシリアル番号や製品情報の文字列を返します

書式

Syntax FT_STATUS **FT_ListDevices** (PVOID *pvArg*1,PVOID *pvArg*2, DWORD *dwFlags*)

Parameters

pvArg1 PVOID: Meaning depends on *dwFlags*

pvArg2 PVOID: Meaning depends on *dwFlags*

dwFlags DWORD: Determines format of returned information

パラメータ

pvArg1 PVOID: *dwFlags*の値により意味が決まる

pvArg2 PVOID: *dwFlags*の値により意味が決まる

dwFlags DWORD: 得る情報の種類を決める

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は FT エラーコードを表します

戻り値はのこの形式は、他のほとんどの関数でもこの形式です。

Remarks

This function can be used in a number of ways to return different types of information. In its simplest form, it can be used to return the number of devices currently connected. If *FT_LIST_NUMBER_ONLY* bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as a pointer to a DWORD location to store the number of devices currently connected. It can be used to return device string information. If *FT_OPEN_BY_SERIAL_NUMBER* bit is set in *dwFlags*, the serial number string will be returned from this function. If *FT_OPEN_BY_DESCRIPTION* bit is set in *dwFlags*, the product description string will be returned from this function. If neither of these bits is set, the serial number string will

be returned by default.

注釈
　この関数は、３種類の情報を得る方法があります。
初めは最も単純な、現在接続されているデバイスの数を求める方法です
dwFlags に　*FT_LIST_NUMBER_ONLY*　を設定して*pvArg1*　に　現在接続数を格納する
変数のアドレス　（DWORDのポインター）をセットします。
dwFlags に　*FT_OPEN_BY_SERIAL_NUMBER*　を設定してすると、シリアル番号の文字列を求めることが出来ます。

dwFlags に　*FT_OPEN_BY_DESCRIPTION*　を設定してすると、デバイス製品情報の文字列を求めることが出来ます。
いずれかのフラグがセットされない時は、デフォルトとしてシリアル番号の文字列が返されます。

It can be used to return device string information for a single device. If *FT_LIST_BY_INDEX* bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as the index of the device, and the parameter *pvArg2* is interpreted as a pointer to a buffer to contain the appropriate string. Indexes are zero-based, and the error code *FT_DEVICE_NOT_FOUND* is returned for an invalid index.

dwFlags に　*FT_LIST_BY_INDEX*　を設定すると　*pvArg1*　は０を始めとするデバイスのインデックスを指定します。*PvArg2*　には　結果の文字列を格納する領域のアドレスを指定します。無効なインデックスを指定するとエラーコードとして　*FT_DEVICE_NOT_FOUND*　が返されます。

It can be used to return device string information for all connected devices. If *FT_LIST_ALL* bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as a pointer to an array of pointers to buffers to contain the appropriate strings, and the parameter *pvArg2* is interpreted as a pointer to a DWORD location to store the number of devices currently connected. Note that, for *pvArg*1, the last entry in the array of pointers to buffers should be a NULL pointer so the array will contain one more location than the number of devices connected.

dwFlags に　*FT_LIST_ALL*　を設定すると　*pvArg1*　は接続されている全てのデバイスの情報文字列の配列を格納する領域へのポインターを設定ます。*PvArg2*　には　現在接続数を格納する変数のアドレス　（DWORDのポインター）をセットします。*pvArg1*　の最後はＮＵＬＬポインターが設定されます。

Examples　　例
Sample code shows how to get the number of devices currently connected.
サンプルは現在接続されているデバイスの数を求めます

```
FT_STATUS ftStatus;
```

```
DWORD numDevs;
ftStatus = FT_ListDevices(&numDevs,NULL,FT_LIST_NUMBER_ONLY);
if (ftStatus == FT_OK) {
// FT_ListDevices OK, number of devices connected is in numDevs
//   FT_ListDevices  == FT_OKなら  numDevs  に有効な接続デバイスの数が入っています
}
else {
// FT_ListDevices failed
//   失敗
}
```

This sample shows how to get the serial number of the first device found. Note that indexes are zero-based. If more than one device is connected, incrementing devIndex will get the serial number of each connected device in turn.

このサンプルは接続されているデバイスのシリアル番号を得る方法を示しています、２ケ以上のデバイスが接続されている場合は、**devIndex**　の値を０から増やしながら、接続されている全てのシリアル番号を得ることができます。

```
FT_STATUS ftStatus;
DWORD devIndex = 0;
char Buffer[16];
ftStatus =
FT_ListDevices((PVOID)devIndex,Buffer,FT_LIST_BY_INDEX|FT_OPEN_BY_SERIAL_N
UMBER);
if (FT_SUCCESS(ftStatus)) {
// FT_ListDevices OK, serial number is in Buffer
else {
// FT_ListDevices failed
}
```

This sample shows how to get the product descriptions of all the devices currently connected.

このサンプルは接続されている全てのデバイス製品情報を得る方法です。

```
FT_STATUS ftStatus;
char *BufPtrs[3]; // pointer to array of 3 pointers 3つのポインターの配列へのポインター
char Buffer1[64]; // buffer for product description of first 1位の製品記述用バッファー
device found
```

```
char Buffer2[64]; // buffer for product description of second第2の製品記述用バッファー
device
DWORD numDevs;
// initialize the array of pointersポインターの配列を初期化します。
BufPtrs[0] = Buffer1;
BufPtrs[1] = Buffer2;
BufPtrs[2] = NULL; // last entry should be NULL最後のエントリーはNULLであるべきです。
ftStatus =
FT_ListDevices(BufPtrs,&numDevs,FT_LIST_ALL|FT_OPEN_BY_DESCRIPTION);
if (FT_SUCCESS(ftStatus)) {
// FT_ListDevices OK, product descriptions are in Buffer1 and Buffer2,
and
// numDevs contains the number of devices connected
//FT_ListDevices、問題なく、製品記述はBuffer1とBuffer2にあります、そして//numDevsは、接続し
ている装置の数を含んでいます。
}
else {
// FT_ListDevices failed
}
```

FT_Open

Description Open the device and return a handle which will be used for subsequent accesses.

説明

デバイスとの通信をオープンしてこの後のアクセスのためのハンドルを返します。

書式

Syntax FT_STATUS **FT_Open** (PVOID *pvDevic*e, FT_HANDLE **ftHandle*)

Parameters

pvDevice ULONG: Must be 0 if only one device is attached. For multiple devices 1, 2 etc.

ftHandle FT_HANDLE *: Pointer to a variable of type FT_HANDLE where the handle will be stored. This handle must be used to access the device.

パラメータ

pvDevice ULONG:　通常０

ftHandle FT_HANDLE ＊:　デバイスにアクセスするためのハンドルを格納する領域へのポインター

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

Remarks

Although this function can be used to open multiple devices by setting *pvDevice* to 0, 1, 2 etc. there is no ability to open a specific device. To open named devices, use the function **FT_OpenE**x.

注意

pvDevice　に０以上の数字をセットすることが出来ますが、複数のデバイスを

オープンするには　　FT_OpenEx　関数を使用してください。

Example　　例

This sample shows how to open a device.　このサンプルは、装置を開く方法を教えます。

```
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
ftStatus = FT_Open((PVOID) 0,&ftHandle);
if (ftStatus == FT_OK) {
// FT_Open OK, use ftHandle to access device FT_Open OK、装置にアクセスする使用ftHandle
}
else {
// FT_Open failed
}
```

FT_OpenEx

Description

Open the named device and return a handle which will be used for subsequent accesses. The device name can be its serial number or device description.

説明

デバイスとの通信をオープンしてこの後のアクセスのためのハンドルを返します。

シリアル番号やデバイス製品情報を指定して複数のデバイスから選択したものをオープンします

Syntax　FT_STATUS **FT_OpenEx** (PVOID *pvArg*1, DWORD *dwFlag*s, FT_HANDLE　**ftHandle*)

書式

Parameters

pvArg1 PVOID: Meaning depends on *dwFlag*s, but it will normally be interpreted as a pointer to a null terminated string.

dwFlags DWORD: FT_OPEN_BY_SERIAL_NUMBER or FT_OPEN_BY_DESCRIPTION.

ftHandle FT_HANDLE *: Pointer to a variable of type FT_HANDLE where the handle will be stored. This handle must be used to access the device.

パラメータ

pvArg1 PVOID:　　通常はNullターミネーションのＣ文字列が格納されている
アドレスを指していますが、*dwFlags*の値によって意味が変わります

dwFlags DWORD:　　FT_OPEN_BY_SERIAL_NUMBER か FT_OPEN_BY_DESCRIPTIONを指定します

ftHandle FT_HANDLE *:ハンドルが格納される　FT_HANDLE　型変数へのポインター

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

Remarks

This function should be used to open multiple devices. Multiple devices can be opened at the same time if they can be distinguished by serial number or device description.

注釈

この関数は、複数のデバイスをオープンするのに使用します。

同時に複数デバイスを使用する際に、シリアル番号かデバイス製品情報を使用して区別します。

Example　例

These samples show how to open two devices simultaneously.
Suppose one device has serial number "FT000001", and the other has serial number "FT999999".

以下のサンプルは、デバイスを同時に２個オープンする方法です。
ひとつのデバイスシリアル番号がＦＴ０００００１で他方がＦＴ９９９９９の場合の例題です。

```
FT_STATUS ftStatus;
FT_HANDLE ftHandle1;
FT_HANDLE ftHandle2;
ftStatus = FT_OpenEx("FT000001",FT_OPEN_BY_SERIAL_NUMBER,&ftHandle1);
if (ftStatus == FT_OK) {
// FT_OpenEx OK, device with serial number "FT000001" is open
}
else {
```

```
// FT_OpenEx failed
}
ftStatus = FT_OpenEx("FT999999",FT_OPEN_BY_SERIAL_NUMBER,&ftHandle2);
if (ftStatus == FT_OK) {
// FT_OpenEx OK, device with serial number "FT999999" is open
}
else {
// FT_OpenEx failed
}
```

Suppose one device has description "USB HS SERIAL CONVERTER", and the other has description "USB PUMP CONTROLLER".

ひとつのデバイス製品情報が"USB HS SERIAL CONVERTER"で他方が"USB PUMP CONTROLLER"の場合の例題です。

```
FT_STATUS ftStatus;
FT_HANDLE ftHandle1;
FT_HANDLE ftHandle2;
ftStatus = FT_OpenEx("USB HS SERIAL
CONVERTER",FT_OPEN_BY_DESCRIPTION,&ftHandle1);
if (ftStatus == FT_OK) {
// FT_OpenEx OK, device with description "USB HS SERIAL CONVERTER"is open
}
else {
// FT_OpenEx failed
}

ftStatus = FT_OpenEx("USB PUMPCONTROLLER",FT_OPEN_BY_DESCRIPTION,&ftHandle2);
if (ftStatus == FT_OK) {
// FT_OpenEx OK, device with description "USB PUMP CONTROLLER" is  open
}
else {
// FT_OpenEx failed
}
```

FT_Close

Description Close an open device.

説明 オープンされたデバイスをクローズします。

Syntax FT_STATUS **FT_Close** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to close.

パラメータ

ftHandle FT_HANDLE: クローズするデバイスのハンドル

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_Read

Description Read data from the device.

説明　デバイスからデータを読み込む。

Syntax FT_STATUS **FT_Read** (FT_HANDLE *ftHandle*, LPVOID *lpBuffer*, DWORD *dwBytesToRead*, LPDWORD *lpdwBytesReturned*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to read.

lpBuffer LPVOID: Pointer to the buffer that receives the data from the device.

dwBytesToRead DWORD: Number of bytes to be read from the device.

lpdwBytesReturned LPDWORD: Pointer to a variable of type DWORD which receives the number of bytes read from the device.

パラメータ

ftHandle FT_HANDLE:　データを読み込むデバイスのハンドル

lpBuffer LPVOID:　データを読み込むバッファへのポインター

dwBytesToRead DWORD:　デバイスから読み込むデータ数.

lpdwBytesReturned LPDWORD: 実際にデバイスから読み込んだデータ数を格納する変数のアドレス　（DWORDのポインター）をセットします。

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_Write

Description Write data to the device.

説明　デバイスにデータを書き込む。

Syntax FT_STATUS **FT_Write** (FT_HANDLE *ftHandle*, LPVOID *lpBuffer*, DWORD *dwBytesToWrite*, LPDWORD *lpdwBytesWritten*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to write.

lpBuffer LPVOID: Pointer to the buffer that contains the data to be written to the device.

dwBytesToWrite DWORD: Number of bytes to write to the device.

lpdwBytesWritten LPDWORD: Pointer to a variable of type DWORD which receives the number of bytes written to the device.

パラメータ

ftHandle FT_HANDLE:　データを書き込むデバイスへのハンドル

lpBuffer LPVOID:　書き込むデータバッファへのポインター

dwBytesToRead DWORD:　デバイスに書き込むデータ数.

lpdwBytesReturned LPDWORD: 実際にデバイスに書き込んだデータ数を格納する変数のアドレス　（DWORDのポインター）をセットします。

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_ResetDevice

Description This function sends a reset command to the device.

説明　デバイスをリセットするコマンドを送る。

Syntax FT_STATUS **FT_ResetDevice** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to reset.

パラメータ

ftHandle FT_HANDLE:　リセットするデバイスへのハンドル

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetBaudRate

Description　This function sets the baud rate for the device.

説明　シリアル通信のボーレートを設定します。

Syntax　FT_STATUS **FT_SetBaudRate** (FT_HANDLE *ftHandle*, DWORD *dwBaudRate*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

dwBaudRate DWORD: Baud rate.

パラメータ

ftHandle FT_HANDLE: デバイスへのハンドル

dwBaudRate DWORD:　ボーレート

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetDivisor

Description

This function sets the baud rate for the device. It is used to set non-standard baud rates.

説明　シリアル通信の非標準のボーレートを設定します。

Syntax　FT_STATUS **FT_SetDivisor** (FT_HANDLE *ftHandle*, USHORT *usDivisor*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

usDivisor USHORT: Divisor.

パラメータ

ftHandle FT_HANDLE:　　デバイスへのハンドル

usDivisor USHORT:　　クロックの除数

Return Value　　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

Remarks

The application note "Setting Baud rates for the FT8U232AM", which is available on our web site http://www.ftdichip.com , describes how to calculate the divisor for a non

standard baud rate.

注釈

クロックの除数の計算方法はアプリケーションノートの　"Setting Baud rates for the FT8U232AM" で説明ししています、インターネット上の　http://www.ftdichip.com　から　ダウンロードして下さい。

Example　例

Suppose we want to set a baud rate of 5787 baud. A simple calculation suggests that a divisor of 4206 should work.

ボーレートに５７８７ボーを設定するためには、クロック除数は

４２０６です

```
FT_HANDLE ftHandle; // handle of device obtained from FT_Open or
FT_OpenEx
FT_STATUS ftStatus;
ftStatus = FT_SetDivisor(ftHandle,0x4206);
if (ftStatus == FT_OK) {
// FT_SetDivisor OK, baud rate has been set to 5787 baud
}
else {
// FT_SetDivisor failed
}
```

FT_SetDataCharacteristics

Description　　This function sets the data characteristics for the device.

説明　　シリアル通信のキャラクタ長・ストップビットパリティを設定します。

Syntax　　FT_STATUS **FT_SetDataCharacteristics** (FT_HANDLE *ftHandle*, UCHAR *uWordLength*, UCHAR *uStopBits*,UCHAR *uParity*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

uWordLength UCHAR: Number of bits per word - must be FT_BITS_8 or FT_BITS_7.

uStopBits UCHAR: Number of stop bits - must be FT_STOP_BITS_1 or FT_STOP_BITS_2.

uParity UCHAR: FT_PATITY_NONE, FT_PARITY_ODD, FT_PARITY_EVEN, FT_PARITY_MARK, FT_PARITY_SPACE.

パラメータ

ftHandle FT_HANDLE:　　デバイスへのハンドル

uWordLength UCHAR:　　キャラクター長　FT_BITS_8 か FT_BITS_7

uStopBits UCHAR:　　ストップビット長 FT_STOP_BITS_1 か FT_STOP_BITS_2

uParity UCHAR:　　パリティビット

FT_PATITY_NONE,　パリティ無し

FT_PARITY_ODD,　奇数パリティ

FT_PARITY_EVEN,　偶数パリティ

FT_PARITY_MARK,　パリティビット常に１

FT_PARITY_SPACE.　パリティビット常にＯ

Return Value　　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetFlowControl

Description　　This function sets the flow control for the device.

説明　　シリアル通信のフロー制御の設定

Syntax　　FT_STATUS **FT_SetFlowControl** (FT_HANDLE *ftHandle*, USHORT us*FlowControl*, UCHAR *uXon*,UCHAR *uXoff*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

usFlowControl USHORT: Must be one of FT_FLOW_NONE, FT_FLOW_RTS_CTS, FT_FLOW_DTR_DSR, FT_FLOW_XON_XOFF

uXon UCHAR: Character used to signal XON. Only used if flow control is FT_FLOW_XON_XOFF.

uXoff UCHAR: Character used to signal XOFF. Only used if flow control is FT_FLOW_XON_XOFF.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

*usFlowContro*l USHORT:　フロー制御のモード設定

FT_FLOW_NONE,　フロー制御

FT_FLOW_RTS_CTS,　ＲＴＳ／ＣＴＳ制御

FT_FLOW_DTR_DSR,　ＤＴＲ／ＤＳＲ制御

FT_FLOW_XON_XOFF　ＸＯＮ／ＸＯＦＦ制御

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetDTR

Description　This function sets DTR.

説明　ＤＴＲをＯＮにする

Syntax　FT_STATUS **FT_SetDTR** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_ClrDTR

Description This function clears DTR.

説明 ＤＴＲをＯＦＦにする

Syntax FT_STATUS **FT_ClrDTR** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetRTS

Description This function sets RTS.

説明 ＲＴＳをＯＮにする

Syntax FT_STATUS **FT_SetRTS** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_ClrRTS

Description This function clears RTS.

説明 ＲＴＳをＯＦＦにする

Syntax FT_STATUS **FT_ClrRTS** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_GetModemStatus

Description　Gets the modem status from the device.

説明　モデムステータスを得る

Syntax　FT_STATUS **FT_GetModemStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwModemStatus*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to read.

lpdwModemStatus LPDWORD: Pointer to a variable of type DWORD which receives the modem status from the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

lpdwModemStatus LPDWORD: モデムのステータスを格納する変数のアドレス
（DWORDのポインター）をセットします。

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetChars

Description　This function sets the special characters for the device.

説明　デバイスにイベント通知やエラーの特別な文字を割当設定します

Syntax　FT_STATUS **FT_SetChars** (FT_HANDLE *ftHandle*, UCHAR *uEventCh*, UCHAR *uEventChEn*, UCHAR *uErrorCh*, UCHAR *uErrorChEn*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

uEventCh UCHAR: Event character,

uEventChEn UCHAR: 0 if event character is disabled, non-zero otherwise.

uErrorCh UCHAR: Error character,

uErrorChEn UCHAR: 0 if error character is disabled, non-zero otherwise.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

uEventCh UCHAR:　　イベント発生時の文字

uEventChEn UCHAR:　0の時は、イベント文字の発生を禁止、それ以外は発生

uErrorCh UCHAR:　　エラー発生時の文字

uErrorChEn UCHAR:　0の時は、エラー文字の発生を禁止、それ以外は発生

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_Purge

Description　This function purges the receive and transmit buffers in the device.

説明　デバイスの送受信バッファをクリアします

Syntax　FT_STATUS **FT_Purge** (FT_HANDLE *ftHandle*, DWORD *dwMask*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

dwMask DWORD: Any combination of FT_PURGE_RX and FT_PURGE_TX.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

dwMask DWORD:　クリアーするバッファの指定FT_PURGE_RX受信バッファ

FT_PURG送信バッファ組み合わせて使用できます

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetTimeouts

Description　This function sets the read and write timeouts for the device.

説明　デバイスの送受信タイムアウトを設定します

Syntax　FT_STATUS **FT_SetTimeouts** (FT_HANDLE *ftHandle*, DWORD *dwReadTimeout*, DWORD *dwWriteTimeout*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

dwReadTimeout DWORD: Read timeout in milliseconds.

dwWriteTimeout DWORD: Write timeout in milliseconds.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

dwReadTimeout DWORD:　読込のタイムアウト　ｍｓ単位で指定

dwWriteTimeout DWORD:　書込みのタイムアウト　ｍｓ単位で指定

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_GetQueueStatus

Description　Gets the number of characters in the receive queue.

説明　受信キュー（バッファ）の文字数を得ます

Syntax　FT_STATUS **FT_GetQueueStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwAmountInRxQueue*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to read.

lpdwAmountInRxQueue LPDWORD: Pointer to a variable of type DWORD which receives the number of characters in the receive queue.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

lpdwAmountInRxQueue LPDWORD: 受信キュー（バッファ）の文字数を格納する変数の

アドレス　（DWORDのポインター）をセットします。

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetBreakOn

Description　Sets the BREAK condition for the device.

説明　デバイスをブレーク状態にします

Syntax　FT_STATUS **FT_SetBreakOn** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_SetBreakOff

Description　Resets the BREAK condition for the device.

説明　デバイスのブレークを解除します

Syntax　FT_STATUS **FT_SetBreakOff** (FT_HANDLE *ftHandle*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

FT_GetStatus

Description

Gets the device status including number of characters in the receive queue, number of characters in the transmit queue, and the current event status.

説明　　デバイス受信バッファの中のデータ数や送信バッファのデータ数とイベント情報を得ます

Syntax　　FT_STATUS **FT_GetStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwAmountInRxQueu*e, LPDWORD *lpdwAmountInTxQueu*e, LPDWORD *lpdwEventStatus*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device to read.

lpdwAmountInRxQueue LPDWORD: Pointer to a variable of type DWORD which receives the number of characters in the receive queue.

lpdwAmountInTxQueue LPDWORD: Pointer to a variable of type DWORD which receives the number of characters in the transmit queue.

lpdwEventStatus LPDWORD: Pointer to a variable of type DWORD which receives the current state of the event status.

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

lpdwAmountInRxQueue LPDWORD: 受信キュー（バッファ）の文字数を格納する変数のアドレス　（DWORDのポインター）をセットします。

lpdwAmountInTxQueue LPDWORD: 送信キュー（バッファ）の文字数を格納する変数のアドレス　（DWORDのポインター）をセットします。

lpdwEventStatus LPDWORD: 現在の受信イベントの状態を格納する変数のアドレス（DWORDのポインター）をセットします。

Return Value　　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

Remarks

For an example of how to use this function, see the sample code in *FT_SetEventNotificatio*n.

注釈

この関数の使い方は　*FT_SetEventNotificatio*n　のサンプルプログラムを参考にして下さい。

FT_SetEventNotification

Description　　Sets conditions for event notification.

説明　　イベントを発生させる条件を設定

Syntax　　FT_STATUS **FT_SetEventNotification** (FT_HANDLE *ftHandle*, DWORD *dwEventMask*, PVOID *pvArg*)

書式

Parameters

ftHandle FT_HANDLE: handle of the device.

dwEventMask DWORD: conditions that cause the event to be set.

pvArg PVOID: interpreted as a handle of an event

パラメータ

ftHandle FT_HANDLE:　デバイスへのハンドル

dwEventMask DWORD:　イベントを発生させる条件を設定

pvArg PVOID:　イベントハンドラーを設定

Return Value　FT_STATUS: FT_OK if successful, otherwise the return value is an FT error code.

戻り値

FT_OK：正常終了

それ以外の値は　FT　エラーコードを表します

Remarks

An application can use this function to setup conditions which allow a thread to block until one of the conditions is met. Typically, an application will create an event, call this function, then block on the event. When the conditions are met, the event is set, and the application thread unblocked.

dwEventMask is a bit-map that describes the events the application is interested in.

pvArg is interpreted as the handle of an event which has been created by the application.

If one of the event conditions is met, the event is set.

If *FT_EVENT_RXCHAR* is set in *dwEventMas*k, the event will be set when a character has been received by the device. If *FT_EVENT_MODEM_STATUS* is set in *dwEventMas*k, the event will be set when a change in the modem signals has been detected by the device.

注釈

アプリケーションプログラムは、この関数を使って設定した条件が揃うまで　スレッドをブロックすることが出来ます。通常アプリケーションはイベントを生成し、この関数を呼びスレッドブロックをします、条件が揃うとイベントが発生しアプリケーションのスレッドブロックが解かれます。

*dwEventMask*はアプリケーションで必要なイベントの指定をします。

*pvArg*はアプリケーションで生成したイベントへのハンドルを指定します

設定された条件になるとイベントがセットされます。

*FT_EVENT_RXCHAR*を*dwEventMask*にセットすると、データが受信されるたびにイベントがセットされます。*FT_EVENT_MODEM_STATUS*を*dwEventMask*にセットすると、モデムの状態の変化をデバイスが捉えるたびにイベントがセットされます。

Example　例

This example shows how to wait for a character to be received or a change in modem status.　First, create the event and call *FT_SetEventNotificatio*n.

このサンプルプログラムは、文字を受信するか、モデムのステータスが変化を捕らえます。
先にイベントを生成してから*FT_SetEventNotification*nを呼んでください

```
FT_HANDLE ftHandle; // handle of an open device
FT_STATUS ftStatus;
HANDLE hEvent;
DWORD EventMask;
hEvent = CreateEvent(NULL,False, // auto-reset eventfalse, // non-signalled state"");
EventMask = FT_EVENT_RXCHAR | FT_EVENT_MODEM_STATUS;
ftStatus = FT_SetEventNotification(ftHandle,EventMask,hEvent);
```

Sometime later, block the application thread by waiting on the event, then when the event has occurred, determine the condition which caused the event, and process it accordingly.

アプリケーションスレッドがイベントを待つためにプロックし、イベントが発生した時は、イベント発生理由を探して適切に処理をします

```
WaitForSingleObject(hEvent,INFINITE);
DWORD EventDWord;
DWORD RxBytes;
DWORD TxBytes;
FT_GetStatus(ftHandle,&RxBytes,&TxBytes,&EventDWord);
if (EventDWord & FT_EVENT_MODEM_STATUS) {
        // modem status event detected, so get current modem status
        FT_GetModemStatus(ftHandle,&Status);
        if (Status & 0x00000010) {
        // CTS is high
        }
        else {
        // CTS is low
        }
        if (Status & 0x00000020) {
        // DSR is high
        }
        else {
        // DSR is low
        }
}
if (RxBytes > 0) {
// call FT_Read() to get received data from device
}
```