

# gtcusbr.dll API 仕様書

## 【 デバイスアクセス API 】

### 【 概要 】

grusb.sys で制御可能なデバイスに対する入出力を提供する API です。

デバイスオープンで得たハンドル経由で入出力処理を行います。

ハンドルは DLL 内部で生成しているものなので、WindowsAPI(ReadFile,CloseHandle 等)で直接使用することはできません。

エラーコードは、ここに記述しているものの他に通常の WindowsAPI の入出力におけるエラーをそのまま返す場合もあります。

エラーコードの取得には、WindowsAPI の GetLastError()関数を使用してください。

何らかの原因により GetLastError()関数が使用出来ない(例: LabView の Vi から使用等、一種のインタプリターの動作を行うモジュールで、インタプリター内部で GetLastError()関数が使用されている場合等)場合は、GtcUSBr\_GetLastError()関数を使用してください。

### 【 使用方法 】

- ・ API 使用時には gtcusbr.h をインクルードして使用してください。

```
#include "gtcusbr.h"
```

- ・ リンク時には、gtcusbr.lib もリンクしてください。

- ・ gtcusbr.dll 本体はウィンドウズのシステムディレクトリ下、もしくは、アプリケーションの実行ファイルの有るディレクトリ下に置いてください。

## GtcUSBr\_OpenDevice()

機能：

デバイスオープン

概要：

gtcusbr.sys で制御されるデバイスの中で、使われていないデバイスがあればオープンしてそのハンドルを返す。

複数の未オープンデバイスが接続されている場合、その中から最初に見つかったデバイスをオープンする。オープンするデバイスを選択することはできない。

宣言：

```
HANDLE GtcUSBr_OpenDevice(void);
```

戻値：

オープンに成功したらデバイスのハンドルを返す。

オープンに失敗したら NULL を返す。

エラーコード：

```
GetLastError() GtcUSBr_GetLastError()
```

ERROR\_DEVICE\_NOT\_CONNECTED : 有効なデバイスが接続されていない

ERROR\_NO\_MORE\_DEVICES : 未使用のデバイスがない

ERROR\_NOT\_ENOUGH\_MEMORY : メモリー不足

## GtcUSBr\_CloseDevice()

機能：

デバイスクローズ

概要：

指定されたハンドルのデバイスをクローズして、関連するリソースを開放する。

宣言：

```
BOOL GtcUSBr_CloseDevice(  
    HANDLE hDevice  
);
```

引数：

hDevice : GtcUSBr\_OpenDevice で取得したデバイスのハンドル

戻値：

クローズに成功したら真(TRUE)、失敗したら偽(FALSE)を返す。

エラーコード：

GetLastError() GtcUSBr\_GetLastError()

ERROR\_INVALID\_HANDLE : 無効なハンドル値が指定された。

## GtcUSBr\_ReadDevice()

機能：

同期読み込み

概要：

デバイスからデータを同期読み込みする。

宣言：

```
BOOL GtcUSBr_ReadDevice(  
    HANDLE hDevice,  
    LPVOID lpBuffer,  
    DWORD nNumberOfBytesToRead,  
    LPDWORD lpNumberOfBytesRead,  
    DWORD dwTimeout  
);
```

引数：

hDevice	GtcUSBr_OpenDevice で取得したデバイスのハンドル
lpBuffer	読み込み先データバッファ
nNumberOfBytesToRead	読み取りバイト数
lpNumberOfBytesRead	実際に読み取れたデータのバイト数
dwTimeout	nNumberOfBytesToRead で指定したバイト数を読み込み までの待機時間(ミリ秒)。INFINITE を指定すると指定 バイト数を読み込むまで戻らない。

戻値：

指定バイト数読み込めたら真(TRUE)、読み込めなかったら偽(FALSE)。  
(偽でも、読み込みバイト数が0とは限らない)

エラーコード：

GetLastError() GtcUSBr\_GetLastError()

ERROR_INVALID_HANDLE	無効なハンドル値が指定された。
ERROR_TIMEOUT	指定バイト数読み込む前に Timeout 時間になった。
ERROR_BUSY	指定されたデバイスはすでに読み込み処理中である。

## GtcUSBr\_ReadDeviceEx()

機能：

非同期読み込み

概要：

デバイスからデータを非同期に読みを行う。

関数内にてデバイスからデータを非同期に読み込むスレッドを起動する。

起動されたスレッドはデータを読み取れるまで待機し、読み取ったらオーバーラップ構造体で示されるイベントハンドルに対しイベント通知を行う（指定されたバイト数に満たない場合もある）。

宣言：

```
BOOL GtcUSBr_ReadDeviceEx(  
    HANDLE                hDevice,  
    LPVOID                lpBuffer,  
    DWORD                 nNumberOfBytesToRead,  
    LPGtcUSBr_OVERLAPPED lpGtcUSBr_Overlapped  
);
```

引数：

hDevice : GtcUSBr\_OpenDevice で取得したデバイスのハンドル  
lpBuffer : 読み込み先データバッファ  
nNumberOfBytesToRead : 読み取りバイト数  
lpGtcUSBr\_Overlapped : オーバーラップ構造体へのポインタ

オーバーラップ構造体：

```
typedef struct {  
    DWORD dwErrorCode;  
    DWORD dwNumberOfBytesTransferred;  
    HANDLE hEvent;  
} GtcUSBr_OVERLAPPED, *LPGtcUSBr_OVERLAPPED;
```

dwErrorCode：	エラーコード
dwNumberOfBytesTransferred：	実際に読み取れたバイト数
hEvent：	イベントハンドル

戻値：

読み込みスレッドが起動できたら真(TRUE)、できなかったら偽(FALSE)。

エラーコード：

GetLastError() GtcUSBr\_GetLastError()

ERROR\_INVALID\_HANDLE：無効なハンドル値が指定された。

ERROR\_BUSY：指定されたデバイスはすでに読み込み処理中である。

オーバーラップ構造体が受け取るエラーコード：

ERROR\_SUCCESS：正常終了

ERROR\_HANDLE\_EOF：指定されたバイト数読み取れなかった。

## GtcUSBr\_WriteDevice()

機能：

同期書き込み

概要：

デバイスにデータを同期書き込みする。

宣言：

```
BOOL GtcUSBr_WriteDevice(  
    HANDLE hDevice,  
    LPVOID lpBuffer,  
    DWORD nNumberOfBytesToWrite,  
    LPDWORD lpNumberOfBytesWrite,  
    DWORD dwTimeOut  
);
```

引数：

hDevice : GtcUSBr\_OpenDevice で取得したデバイスのハンドル  
lpBuffer : 書き込み元データバッファ  
nNumberOfBytesToWrite : 書き込みバイト数  
lpNumberOfBytesWrite : 実際書き込めたデータのバイト数  
dwTimeOut : nNumberOfBytesToWrite で指定したバイト数を書き込めるまでの待機時間(ミリ秒)。INFINITE を指定すると指定バイト数を読み込むまで戻らない。

戻値：

指定バイト数書き込めたら真(TRUE)、書き込めなかったら偽(FALSE)。  
(偽でも、書き込みバイト数が0とは限らない)

エラーコード：

GetLastError() GtcUSBr\_GetLastError()

ERROR\_INVALID\_HANDLE : 無効なハンドル値が指定された。

ERROR\_TIMEOUT : 指定バイト数書き込む前に TimeOut 時間になった。

ERROR\_BUSY : 指定されたデバイスはすでに読み込み処理中である。

## GtcUSBr\_WriteDeviceEx()

機能：

非同期書き込み

概要：

デバイスからデータを非同期に書き込みを行う。

関数内にてデバイスにデータを非同期に書き込むスレッドを起動する。

起動されたスレッドはデータが書き込めるまで待機し、書き込めたらオーバーラップ構造体で示されるイベントハンドルに対しイベント通知を行う。

宣言：

```
BOOL GtcUSBr_WriteDeviceEx(  
    HANDLE hDevice,  
    LPVOID lpBuffer,  
    DWORD nNumberOfBytesToWrite,  
    LPGtcUSBr_OVERLAPPED lpGtcUSBr_Overlapped  
);
```

引数：

hDevice：	GtcUSBr_OpenDevice で取得したデバイスのハンドル
lpBuffer：	書き込み元データバッファ
nNumberOfBytesToRead：	書き込みバイト数
lpGtcUSBr_Overlapped：	オーバーラップ構造体へのポインタ



オーバーラップ構造体：

```
typedef struct {  
    DWORD dwErrorCode;  
    DWORD dwNumberOfBytesTransferred;  
    HANDLE hEvent;  
} GtcUSBr_OVERLAPPED, *LPGtcUSBr_OVERLAPPED;
```

dwErrorCode：	エラーコード
dwNumberOfBytesTransferred：	実際に書き込めたバイト数
hEvent：	イベントハンドル

戻値：

書き込みスレッドが起動できたら真(TRUE)、できなかったら偽(FALSE)。

エラーコード：

GetLastError() GtcUSBr\_GetLastError()

ERROR\_INVALID\_HANDLE：無効なハンドル値が指定された。

ERROR\_BUSY：指定されたデバイスはすでに書き込み処理中である。

オーバーラップ構造体を受け取るエラーコード：

ERROR\_SUCCESS：正常終了

## GtcUSBr\_GetLastError()

機能：

拡張エラー情報を取得する。

概要：

各種 GtcUSBr API にて発生した拡張エラーコードを取得する。

WindowsAPI の GetLastError()関数が何らかの原因で使用出来ない場合に使用。

エラーコードはスレッド毎で取得が可能（異なるスレッドからは取得できない）

V e r 1 . 2 1 以降対応

宣言：

```
BOOL GtcUSBr_GetLastError();
```

引数： なし

戻値： 拡張エラーコード

詳細は各 A P I 参照のこと。