gtcusbr.dll API Specification Sheet

[Device access API]

[Description]
This product is an API to provide input and output functions for devices that can be controlled by gtcusbr.sys. Input and output processing functions are performed via the handle obtained from the Device Open function. This handle is generated within the DLL, and therefore cannot be used directly by Windows API functions (ReadFile, CloseHandle, etc.).

In addition to the error codes described here, errors that occur in normal Windows API input and output operations are also returned as is.
Please use the Windows API GetLastError() function to obtain the error codes. If for any reason you cannot use the GetLastError() function (for example, when the GetLastError() function is being used within an interpreter module from LabView VI or a similar virtual instrument), please use the GtcUSBr_GetLastError() function.

[Usage Method]

- Please install gtcusbr.h to enable use of this API.

```
#include “gtcusbr.h”
```

- When performing a linking operation, please also link gtcusbr.lib.

- Place gtcusbr.dll in the Windows system directory, or else in a directory with a Run Application file.

GtcUSBr_OpenDevice()

Function:

Device Open

Description:

If there is a device among the devices controlled by gtcusbr.sys that is not being used, this function opens the device and returns that handle.
If there are multiple devices that have not been opened, the device found first will be opened. The device to be opened cannot be selected.

Declaration:

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

Returned values:

The device handle is returned if the Open operation was successful.
NULL is returned if the Open operation failed.

Error codes:

GetLastError() GtcUSBr_GetLastError()

ERROR_DEVICE_NOT_CONNECTED	:	A valid device is not connected
ERROR_NO_MORE_DEVICES	:	There are no devices that are not being used.
ERROR_NOT_ENOUGH_MEMORY	:	Insufficient memory

* To select a device to be controlled, obtain its ID number by using :IF:ID? command.

GtcUSBr_CloseDevice()

Function:

Device Close

Description:

The Device Close function closes the device with the specified handle and frees up all the associated resources.

Declaration:

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

Argument:

hDevice : The handle obtained by GtcUSBr_OpenDevice

Returned values:

TRUE is returned if the Close operation was successful, and FALSE if it failed.

Error codes:

GetLastError() GtcUSBr_GetLastError()

ERROR_INVALID_HANDLE : An invalid handle value was specified.

GtcUSBr_ReadDevice()

Function:

Synchronous reading

Description:

This function performs synchronous reading of data from the device.

Declaration:

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

Arguments:

hDevice :	The device handle obtained by GtcUSBr_OpenDevice
lpBuffer :	The data read destination buffer
nNumberOfBytesToRead :	The number of bytes to be read
lpNumberOfBytesRead :	The number of data bytes that were actually read
dwTimeOut :	The time interval specified (in ms) for reading of the number of bytes specified in nNumberOfBytesToRead. If Infinite is specified, return is not made until the number of specified bytes has been read.

Returned values:

TRUE is returned if the specified number of bytes could be read, and FALSE if the specified number could not be read.

FALSE being returned does not necessarily mean that the number of bytes read was 0.

Error codes:

GetLastError() GtcUSBr_GetLastError()

ERROR_INVALID_HANDLE	:	An invalid handle value was specified.
ERROR_TIMEOUT	:	A timeout occurred before the specified number of bytes could be read.
ERROR_BUSY	:	The specified device is already performing a read operation.

GtcUSBr_ReadDeviceEx()

Function:

Asynchronous reading

Description:

The ReadDevice() function performs asynchronous reading of data from the device.

It activates a thread in the function to perform asynchronous reading of data from the device.

The activated thread waits until the data is read, and then performs event notification with respect to the event handle indicated in the overlap structure (there are times when reading up to the specified number of bytes is not performed).

Declaration:

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

Arguments:

hDevice :	The device handle obtained by GtcUSBr_OpenDevice
lpBuffer :	The data read destination data buffer
nNumberOfBytesToRead :	The number of bytes to be read
lpGtcUSBr_Overlapped :	Pointer to the Overlap structure

Overlap structure:

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

dwErrorCode :	Error code
dwNumberOfBytesTransfered :	Actual number of bytes read
hEvent :	Event handle

Returned values:

TRUE if the thread for reading was able to be activated, FALSE if it wasn't.

Error codes:

GetLastError()　GtcUSBr_GetLastError()

ERROR_INVALID_HANDLE	: An invalid handle value was specified.
ERROR_BUSY	: The specified device is already performing a read operation.

Error codes received by the Overlap structure:

ERROR_SUCCESS	: Normal completion
ERROR_HANDLE_EOF	: The specified number of bytes could be read.

GtcUSBr_WriteDevice()

Function:

Synchronous writing

Description:

This function performs synchronous writing of data to the device.

Declaration:

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

Arguments:

hDevice :	The device handle obtained by GtcUSBr_OpenDevice
lpBuffer :	The data write destination buffer
nNumberOfBytesToWrite :	The number of bytes to be written
lpNumberOfBytesWrite :	The number of data bytes that were actually written
dwTimeOut :	The time interval specified (in ms) for writing of the number of bytes specified in nNumberOfBytesToWrite. If Infinite is specified, return is not made until the specified number of bytes has been written.

Returned values:

TRUE is returned if the specified number of bytes could be written, and FALSE if they could not be written

FALSE being returned does not necessarily mean that the number of bytes written was 0.

Error codes:

GetLastError()　GtcUSBr_GetLastError()

ERROR_INVALID_HANDLE	:	An invalid handle value was specified.
ERROR_TIMEOUT	:	A timeout occurred before the specified number of bytes could be written.
ERROR_BUSY	:	The specified device is already performing a write operation.

GtcUSBr_WriteDeviceEx()

Function:

Asynchronous writing

Description:

The WriteDeviceEX() function performs asynchronous writing of data to the device.

It activates a thread in the function to perform asynchronous writing of data to the device.

The activated thread waits until the data is written, and then performs event notification with respect to the event handle indicated in the overlap structure (there are times when writing up to the specified number of bytes is not performed).

Declaration:

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

Arguments:

hDevice :	The device handle obtained by GtcUSBr_OpenDevice
lpBuffer :	The data write destination data buffer
nNumberOfBytesToWrite :	The number of bytes to be written
lpGtcUSBr_Overlapped :	Pointer to the Overlap structure

Overlap structure:

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

dwErrorCode :	Error code
dwNumberOfBytesTransfered :	Actual number of bytes written
hEvent :	Event handle

Returned values:

TRUE if the thread for writing was able to be activated, FALSE if it wasn't.

Error codes:

GetLastError()　GtcUSBr_GetLastError()

ERROR_INVALID_HANDLE	: An invalid handle value was specified.
ERROR_BUSY	: The specified device is already performing a write operation.

Error codes received by the Overlap structure:

ERROR_SUCCESS	: Normal completion

GtcUSBr_GetLastError()

Function:

Gets extended error information

Description:

This function gets extended codes for the errors that occurred in each GtcUSBr API. Use this function if there is some reason that you cannot use the Windows API GetLastError() function. Error codes can be obtained for each function (but not for different threads).

*Available from Ver 1.21 onwards.

Declaration:

```
DWORD GtcUSBr_GetLastError();
```

Arguments: None

Returned values: Extended error codes

Refer to each API for further details.

January 20, 2003

Graphtec Corporation