EasyUSB Application Programming Interface

Preliminary Rev 0.1 – Mar/2010

1. Description

The EasyUSB API can be embedded on the device memory and it's accessible by software applications. Through the library functions it's easy to communicate directly to the UART interface and set parameters of the device.

In the current version the EasyUSB API is distributed as a static or dynamic link library (DLL). If you need to use native functions, library customizations or other operating system implementations, please contact the EasyUSB development team.

2. Functions

The prototypes of the functions implemented on the current version are:

```
char* DllVersion(void);
HANDLE Connect(char driveletter);
void Disconnect(HANDLE dev);
int Read(HANDLE dev, unsigned char *buf);
void Write(HANDLE dev, unsigned char *buf, int length);
void GetConfig(HANDLE dev, unsigned char *buf);
void SetConfig(HANDLE dev, unsigned char *buf);
```

2.1. DllVersion

Description: obtains the library version that's currently in use
Prototype: char* DllVersion(void)
Parameters: none
Return Value: string with dll version

2.2. Connect

Description: connects to the EasyUSB device
Prototype: HANDLE Connect(char driveletter)
Parameters: driveletter – drive where is mounted the EasyUSB device
Return Value: handle to the EasyUSB device

2.3. Disconnect

Description: disconnects from the EasyUSB device
Prototype: void Disconnect(HANDLE dev)
Parameters: dev – device handle
Return Value: none

2.4. Read

Description: reads a buffer of data sent from the device via UART
Prototype: int Read(HANDLE dev, unsigned char *buf);
Parameters: dev – device handle
*buf – buffer pointer where read data will be copied
Return Value: number of bytes read

2.5. Write

Description: writes a buffer of data to the device via UART
Prototype: void Write(HANDLE dev, unsigned char *buf, int length)
Parameters: dev – device handle
*buf – buffer pointer where data to be written is stored
length – number of bytes to be written
Return Value: none

2.6. GetConfig

Description: gets configuration data block
Prototype: void GetConfig(HANDLE dev, unsigned char *buf)
Parameters: dev – device handle
*buf – buffer pointer where read data will be copied
Return Value: none

2.7. SetConfig

Description: sets configuration data block
Prototype: void SetConfig(HANDLE dev, unsigned char *buf);
Parameters: dev – device handle
*buf – buffer pointer where read data will be copied
Return Value: none

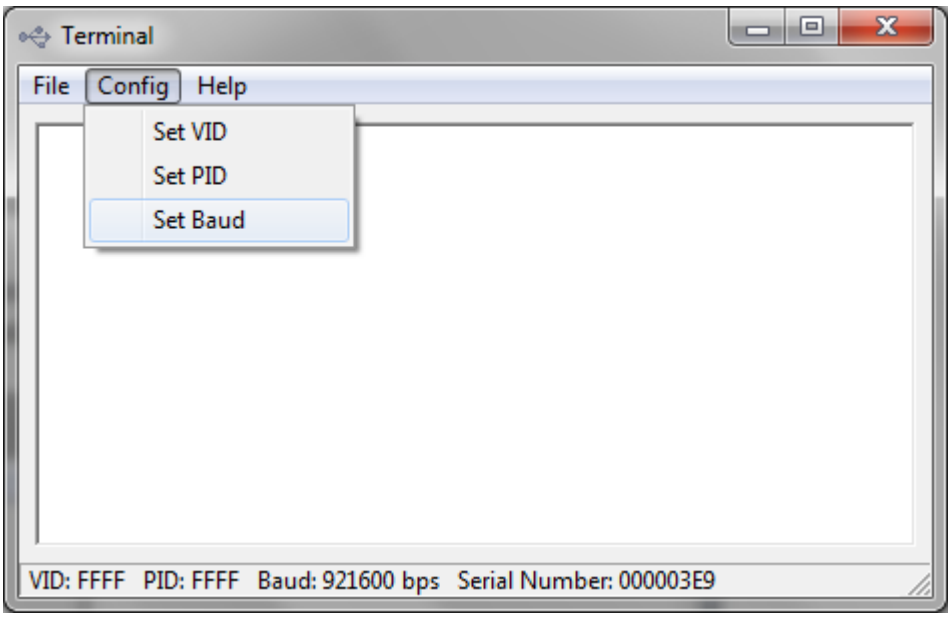
3. Configuration data block

The configuration data is a 512 bytes block that stores the parameters of the device that must be appropriately set by the manufacturer, according to the following table:

Config[0..1] = word with the VID (Vendor ID) of the USB interface
default = 0xFFFF
Config[2..3] = word with the PID (Product ID) of the USB interface
default = 0xFFFF
Config[4..7] = reserved
Config[8] = byte that indicates the baud rate of the UART interface:
0 = 9600bps
1 = 19200bps
2 = 38400bps
3 = 57600bps
4 = 115200bps
5 = 921600bps
default = 9600bps
Config[510..511] = signature: 0x55AA

3. Demonstration software

For demonstrations purposes, it's freely available the Terminal software with all the described functions implemented:



Contacts

This is a preliminary release of the EasyUSB project.

For technical questions, software, documentation and for information about producing, distributing, licensing, please contact the developer:

easyusb@brondani.com

Disclaimer