Embedded – System – Library

Version 1.0

(Stand: 17.02.2023)

Prof. Dr.-Ing. Thomas Breuer

Bonn-Rhein-Sieg University
of Applied Sciences
Department of Computer Science

thomas.breuer@h-brs.de

Inhaltsverzeichnis

1 Einleitung....3

2 Installation....4

2.1 Quellcode....4
2.2 Umgebungsvariable....4
2.3 Compiler, IDE & Co....4
2.4 Dokumentation....6
2.5 Example....6
2.6 Eigenes Projekt....7

3 Architektur....8

3.1 Verzeichnisstruktur....8

4 Tools & Scripte....9

4.1 Bmp2Cpp....9
4.2 Bmp2Font....9
4.3 VirtualDeviceServer....10
4.4 USBhost....10
4.5 USB Descriptor....11
4.6 Pin Configuration....11

5 Virtual-Server....12

6 USB Descriptor....15

6.1 Schlüsselworte und Gruppen....15

7 Anhang....20

7.1 Abkürzungen....20
7.2 Änderungshistorie....20

1 Einleitung

Die Embedded-System-Library (EmbSysLib) ist eine objektorientierte, in C++ implementierte Library für eingebettete Systeme, die eine weitgehend controller-unabhängige Entwicklung von Embedded-Applikationen ermöglicht. Sie stellt zahlreiche Klassen mit Schnittstellen zu prozessorinternen und externen Komponenten zur Verfügung. Darüber hinaus umfasst die EmbSysLib weitere Software-Komponenten für embedded-typische Aufgaben.

Die Bibliothek ist im Rahmen einiger Vorlesungen an der Hochschule Bonn-Rhein-Sieg entstanden und wird in Programmier-Praktika und Übungen eingesetzt. Zu Gunsten eines einheitlichen und einfachen Interfaces, das die elementaren Grundfunktionen abbildet, wurde auf eine Unterstützung komplexer und prozessorspezifischer Features verzichtet. In einigen Projekten aus den Bereichen Smart-Home und Robotik hat sich dieser Ansatz bereits bewährt.

2 Installation

Kapitel

2.1 EmbSysLib herunterladen und entpacken
2.2 Umgebungsvariable setzen
2.3 Compiler bzw. Integrierte Entwicklungsumgebung (IDE) sowie weitere Tools installieren
2.4 Dokumentation erstellen
2.5 Example verwenden
2.6 Eigenes Projekt anlegen

2.1 Quellcode

Die EmbSysLib kann in ein beliebiges Verzeichnis kopiert bzw. entpackt werden. Dieses Verzeichnis wird im weiteren Verlauf der Dokumentation als "**`EmbSysLib\`**" bezeichnet. In Kap. 3.1 Verzeichnisstruktur wird die Struktur der EmbSysLib beschrieben.

2.2 Umgebungsvariable

Die Umgebungsvariable **`EMBSYSLIB`** wird verwendet, um auf Tools bzw. Quellcode zu verweisen. Die Umgebungsvariable muss dazu auf das Verzeichnis verweisen, in dem die EmbSysLib installiert wurde. Falls die EmbSysLib in ein anderes Verzeichnis verschoben wurde, muss die Umgebungsvariable erneut gesetzt werden.

In das Verzeichnis **`EmbSysLib\`** wechseln und dort **`_setEnv.bat`** ausführen, um die Umgebungsvariable **`EMBSYSLIB`** auf den aktuellen Pfad zu setzen.

2.3 Compiler, IDE & Co

Die EmbSysLib enthält zahlreiche Beispiel-Projekte (siehe auch Kap. 2.5 Example), einschließlich der Projektdateien für verschiedene Entwicklungsumgebungen. Um diese Beispielprojekte nutzen zu können, müssen die entsprechenden Entwicklungsumgebungen installiert werden. Auch weitere Tools, z.B. zur Generierung der Dokumentation oder zur Ausführung von Skripten sind ggf. zu installieren. Die folgende Tabelle listet auf, welche Tools zu welchem Zweck erforderlich sind:

Tool	Beschreibung	Anwendung
EmBitz www.embitz.org	Compiler/IDE	Projekte mit ARM-Cortex Mikrocontroller
µVision / MDK-ARM www.keil.com *(Registrierung erforderlich)*	Compiler/IDE	Projekte mit ARM-Cortex Mikrocontroller
STM32CubeIDE www.st.com *(Registrierung erforderlich)*	Compiler/IDE	Projekte mit STM32-Mikrocontroller
Code::Blocks www.codeblocks.org	Compiler/IDE	Projekte als Windows-Anwendung

Tool	Beschreibung	Anwendung
Doxygen doxygen.nl	Software-Dokumentations-werkzeug	Generierung der EmbSysLib-Dokumentation
PHP www.php.net	Script-Interpreter	Code-Generierung, z.B. für USB-Descriptoren
PuTTY www.putty.org	Terminal für Serielle Schnittstellen	In-/Output-Terminal für Example-Projekte

Tabelle 2.1:

2.3.1 EmBitz

`EmBitz` gemäß Beschreibung des Anbieters installieren, ggf. Debug-Tool **`EBLink`** ebenfalls installieren. Nach dem ersten Programmstart muss ein Compiler ausgewählt werden, hier "GNU Arm Embedded Toolchain" verwenden.

2.3.2 µVision / MDK

`MDK` gemäß Beschreibung des Anbieters installieren, ggf. Default-Einstellungen übernehmen. Nach dem erstmaligem Öffnen eines Projektes in µVision werden zunächst einige erforderliche Packages nachgeladen, dies kann einige Zeit in Anspruch nehmen. Falls dies nicht erfolgreich ist, können die Packages direkt unter Keil - MDK Device List bezogen und manuell nachinstalliert werden (Package Manager - Menü 'File'-'Import ...'). Es werden nur die "Device Support"-Packages der verwendeten Controller-Familie benötigt, "Board Support"-Packages sind nicht erforderlich.

2.3.3 STM32CubeIDE

Setup ausführen um **`STM32CubeIDE`** zu installieren, dabei Standardeinstellungen verwenden und Treiber für Debugger ebenfalls installieren.

2.3.4 Code::Blocks

Es wird die Variante **`codeblocks-*mingw-setup.exe`** empfohlen, die neben einem Setup zusätzlich einen GCC-Compiler und GDB-Debugger des MinGW-W64 Projektes enthält.

Setup ausführen um **`Code::Blocks`** zu installieren, dabei Standardeinstellungen verwenden. Nach dem ersten Programmstart muss ein Compiler ausgewählt werden, hier "GNU GCC Compiler" angeben. Ggf. Einstellung im Menü 'Settings' - 'Compiler' - "Reset defaults' zurücksetzen.

2.3.5 Doxygen

Setup ausführen um **`Doxygen`** zu installieren, anschließend das Installationsverzeichnis der Windows-Systemvariable "Path" hinzufügen.

2.3.6 PHP

Downloaddatei in ein geeignetes Verzeichnis entpacken. Anschließend das Verzeichnis der Windows-Systemvariable "Path" hinzufügen

2.3.7 PuTTY

Installationsdatei **putty-...-installer.msi** herunterladen und installieren. In **PuTTY** lassen sich verschiedene Konfiurationen speichern. Dazu nach Anschluss des Controller-Boards im Gerätemanager den betreffenden COM-Port ermitteln. Weitere Standardeinstellungen für die Beispiele:

- Connection – Serial
  Speed: 9600 baud, Data bits: 8, Stop bits: 1, Parity: None, Flow control: None
- Terminal
  Local line editing: Force off

2.4 Dokumentation

Die Dokumentation der EmbSysLib lässt sich mit **doxygen** generieren. In Verzeichnis **Doc** befindet sich dazu die Batch-Datei **_build.bat**, die **doxygen** mehrfach aufruft um die Dokumentation der EmbSysLib (controller-spezifischer und controller-unabhängiger Teil) zu erstellen.

Doc_build.bat aufrufen, um die Dokumentation zu erstellen.
Anschließend **Doc\EmbSysLib.html** im Browser starten, um die Dokumentation anzuzeigen.

2.5 Example

Die Projekt-Beispiele umfassen einen hardwareunabhängigen Teil (**Example\Src\Main**), die hardwareabhängige Konfigurationen (**Example\Src\Board*<Board>*\config*.h**) sowie hardware- und compilerspezifische Projektdateien (**Example\Project*<Board>**<IDE>***).

Um ein Projekt zu starten, passende Projektdatei aus **Examle/Project/*<Board>*/*<IDE>*/** auswählen und in der IDE öffnen.

In **Example\Src\main.cpp** sind alle Beispiele aufgelistet, s.d. das gewünschte Beispiel durch Entfernen der Kommentarzeichen aktiviert werden kann. Die Datei **Example\Src\lib.cpp** enthält Verweise auf Module der EmbSysLib, die in den Beispielen verwendet werden.

Hardware-Implementierungen **Src/Hardware/MCU/**	**Board** **Example/Project/**	**Mikrocontroller**
STM32L1xx	STM32L100-Discovery	STM32L100RC
STM32L4xx	STM32-Nucleo32-L432	STM32L432KC
Virtual	Hardware-Simulation mit VirtualDeviceServer	*Windows-Application*

Tabelle 2.2:

Die hardware-spezifischen Konfigurationen für die einzelnen Mikrocontroller bzw. Boards befinden sich im Verzeichnis **Example\Src\Board*<Board>*\config*.h** und werden von den jeweiligen Beispielen und in Abhängigkeit von der verwendeten Hardware (Board) includiert. In den Konfigurationsdateien ist auch angegeben, ob und welche zusätzlichen Komponenten an das Board angeschlossen werden müssen.

Vor Ausführung der Beispiele die Hinweise in der jeweiligen Konfigurationsdatei **config*.h** beachten.

2.5.1 Virtual

Das Target "Virtual" bezeichnet die Simulation von Hardware, s.d. die Software nicht nur für einen Mikrocontroller sondern auch als Windows-Applikation übersetzt werden kann. Einige typische Controller-Komponenten (ADC, DAC, GPIO, aber auch Display mit Touch-Funktion) werden durch den VirtualDeviceServer simuliert und visualisiert, siehe auch Kap. 4.3 VirtualDeviceServer.

Die EmbSysLib unterstützt auch die Entwicklung eines Servers, siehe dazu Kap. 5 Virtual-Server.

Vor Ausführung der Beispiele "Virtual" den Server mit
`Example\Project\Virtual\_VirtualDeviceServer.bat` starten.

2.6 Eigenes Projekt

Die Beispiele lassen sich als Vorlage für eigene Projekte nutzen:

Verzeichnis **`Example\`** in ein beliebiges Verzeichnis kopieren und ggf. beliebig umbenennen.

Anschließend kann im neuen Verzeichnis der Inhalt von **`Src\main.cpp`** beispielsweise durch **`Src\Main\Demo\Blinky\main.cpp`** oder einem anderen Beispiel aus **`Src\Main\...`** ersetzt werden.
Ggf. Konfigurationen in **`Src\Board\...`** anpassen und nicht mehr benötigte Dateien und Verzeichnisse löschen.
Falls weitere Module aus der EmbSysLib verwendet werden sollen, können diese in **`Src\lib.cpp`** eingetragen werden.

3 Architektur

3.1 Verzeichnisstruktur

EmbSysLib	Hauptverzeichnis
Doc	Dokumentation
_Doxygen	Doxygen-Konfigurationsdateien und weitere Ressourcen für Doxygen
Main	Dokumentation des Hauptteils der EmbSysLib sowie in weiteren Verzeichnissen die controllerspezifischen Dokumentationen. Die Verzeichnis werden von Doxygen angelegt.
Example	Beispiel-Projekte
Project	Projekt-Dateien für unterschiedliche Boards und IDE's
Src	Quelldateien
Board	Hardwarespezifische Konfigurationen
Main	Gemeinsame Quelldateien der Beispiele
Resource	Verschiedene Ressourcen (Font, Bitmap, Color etc.), die in den Beispielprojekten verwendet werden
Src	Quellcode
Control	User Interface Kontrolelemente (e.g. Button, LED)
Device	'High-Level' Hardware-Schicht
Hardware	'Low-Level' Hardware-Schicht
Common	Abstrakte Schnittstellen zur Controller- und Peripherie-Hardware.
MCU	Enthält in den jeweiligen Unterverzeichnissen die controllerspezifische Implementierung
Peripheral	Implementierung verschiedener Peripheriekomponenten
Module	Sammlung verschiedener SW-Module
Std	Sammlung "nützlicher" Klassen (FIFO etc.)
Tools	Weitere PHP-Scripte, Build-Tools sowie Tools, die für die Beispiele benötigt werden

Tabelle 3.1: Verzeichnisstruktur der EmbSysLib.

Das Unterverzeichnis **Example** kann als Vorlage für eigene Projekte genutzt und dazu in ein beliebiges Verzeichnis kopiert/verschoben werden.

In der IDE können zwar Umgebungsvariablen als Bestandteil von Include-Pfade angegeben werden, dies ist jedoch in µVision nicht für die zu übersetzenden Projekt-Dateien möglich. µVision sieht dazu nur absolute oder relative Pfade vor. Damit Projekte unabhängig vom Pfad zur EmbSysLib strukturiert werden können, werden alle Verweise auf die EmbSysLib über Include-Pfade gehandhabt (vergl. **main.cpp**, **lib.cpp** und **startup.s**).

4 Tools & Scripte

4.1 Bmp2Cpp

Programm-Datei:

`Tools/Bmp2Cpp.exe`
Das Programm konvertiert ein Bitmap-Bild in eine Binär- sowie eine Header-Datei.
Siehe auch: **`Example/Src/Resource/Bitmap`**.

Input:

Das Bild muss als Windows-Bitmap (BMP-Datei) mit 24-Bit Farbauflösung vorliegen.

Output:

Erzeugt wird eine H-Datei, die eine Objekt-Instanz der Template-Klasse **`EmbSysLib::Hw::Bitmap::Data`** enthält. Der Objektname setzt sich dabei aus "**`bitmap`**" und dem Namen der Input-Datei zusammen. Weiterhin wird eine BIN-Datei erzeugt, die das Image der Objekt-Instanz enthält. Die Dateinamen sind mit dem Dateinamen der Input-Datei identisch.
Die Objekt-Instanz kann in ein Bitmap-Objekt konvertiert und zur Darstellung von Bildern genutzt werden. Siehe Methode **`putBitmap()`** im Beispiel **`HwDisplayGraphic.cpp`** .

4.2 Bmp2Font

Programm-Datei:

`Tools/Bmp2Font.exe`
Dieses Programm konvertiert ein Bitmap-Bild in eine Header-Datei, die einen Bitmap-Font enthält.
Siehe auch: **`Example/Src/Resource/Font`**.

Input:

Der Font muss als Pixel-Font in einer Windows-BMP-Datei mit einer 4-Bit Farbauflösung (16-Farben) vorliegen. Das Bild muss alle 256 Glyphen enthalten, die durch einen 1-Pixel-Rand voneinander getrennt sind. Der Font lässt sich mit einem einfachen Bitmap-Zeichentools erstellen und bearbeiten.

Output:

Das Flag **`-a`** verändert auch die Input-Datei, indem die Ränder zwischen den Glyphen eingetragen werden.
Erzeugt wird eine H-Datei, die eine Objekt-Instanz der Template-Klasse **`EmbSysLib::Hw::Font::Data`** enthält. Der Objektname setzt sich dabei aus "**`font`**" und dem Namen der Input-Datei zusammen. Der Dateiname der Ausgabedatei (*.h) ist mit dem Dateinamen der Input-Datei (*.bmp) identisch.
Die Objekt-Instanz kann in ein Font-Objekt konvertiert und in der Klasse **`DisplayGraphic`** zur Darstellung von Texten genutzt werden. Siehe Beispiel **`HwDisplayGraphic.cpp`** .

4.3 VirtualDeviceServer

Programm-Datei:

`Tools/VirtualDeviceServer.exe`

Der VirtualDeviceServer simuliert typische Peripherie-Komponenten eines eingebetteten Systems:

- Taster
- Schalter
- LED
- Analog Input (ADC)
- Analog Output (DAC)
- Zeichen-Display (Font: 16x30)
- Graphik-Display

Die Display-Größe beträgt 320x240 Pixel bzw. 20 Spalten x 8 Zeilen (default). Analoge In- und Outputs sind als Kanal 0 bis 3 des virtuellen ADCs bzw. DACs verfügbar. Die GUI-Elemente Taster, Schalter und LED werden im Bitmuster des IO-Ports dargestellt.

GUI-Element	I/O-Bit	Direction
Taste <<	0	In
Taste o	1	In
Taste >>	2	In
Taste +	3	In
Taste -	4	In
Taste[A:E]	5:9	In
Switch[0:3]	10:13	In
LED[0:7]	16:23	Out

Tabelle 4.1: Abbildung der digitale GUI-Elemente auf den Bits des I/O-Ports

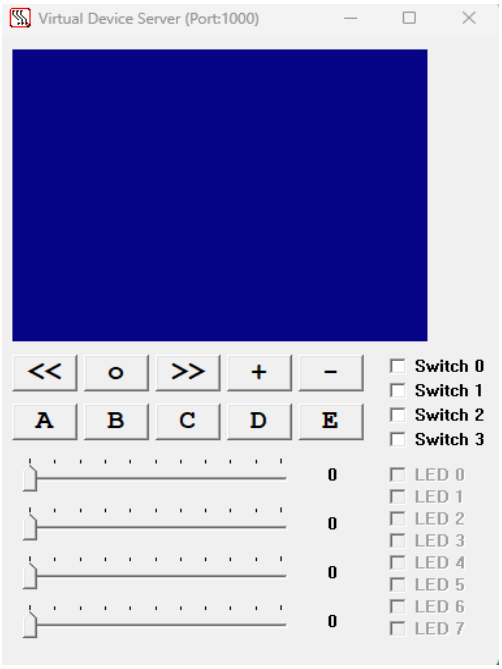

Abbildung 4.1: Bedienoberfläche

Mit dem Kommandozeilenparameter **`-d`** kann die Größe des virtuellen Displays und mit **`-p`** der UDP-Port eingestellt werden (z.B. **`VirtualDeviceServer.exe -d 800x480 -p 1100`**).
Der verwendete Port wird in der Kopfzeile der Bedienoberfläche angezeigt.

Configuration file .....

4.4 USBhost

Programm-Datei:

`Tools/USBhost.exe`
Mit diesem Programm kann das Beispiel **`HwUSBdevice`** host-seitig getestet werden.
Die Vendor- sowie die Product-ID können per Kommandozeile eingestellt werden. Nach dem Programmstart bzw. einem Kommunikationsfehler wird erneut versucht, eine Verbindung zum USB-Device aufzubauen. Sofern eine Verbindung besteht, kann per Tastatur ein Interrupt- oder Control-Transfer initiiert werden.

4.5 USB Descriptor

PHP-Skript:

`Src\Hardware\Common\USB\USBdevice\USB_Descriptor_Script.php`

Erzeugt eine Klasse mit USB-Descriptoren, die von der Klasse **`USBdevice`** benötigt werden.

Ausführung:

```
php -f USB_Descriptor_Script.php <inputFile> <className>
```

`<inputFile>`: Textdatei mit Descriptor-Spezifikation, siehe Kap. 6 USB Descriptor
`<className>`: Name der generierten Klasse, die von **`USBdeviceDescriptor`** erbt.

4.6 Pin Configuration

PHP-Skript:

`Src\Hardware\MCU\...\PinConfig.php`

Erzeugt eine Klasse **`PinConfig`**, die die Zuordnung der Alternate Functions zu den verfügbaren GPIO-Pins eines µControllers beschreibt. **`PinConfig`** wird von einigen hardwarespzifischen Klassen zur Konfiguration der GPIOs verwendet. Das Skript ist nur dann erforderlich, wenn ein neuer Controllertyp der EmbSysLib hinzugefügt wird.

Ausführung:

```
php -f PinConfig.php <inputFile>
```

`<inputFile>`: CSV-Datei mit der Spezifikation der Alternate Functions aller GPIO-Pins

5 Virtual-Server

Um einen Server zu erstellen, der eine oder mehrere virtuelle Schnittstelle unterstützt, muss die entsprechende Handler-Klasse konkretisiert werden. Die abgeleitete Klasse muss dazu die Schnittstellenmethoden implementieren (Call-Back-Methoden):

Basisklasse	Callback-Methode
Port_Virtual::Handler	
	DWORD onValueRequest(DWORD dir, DWORD value) Client setzt Port-Ausgabe und fordert Port-Eingabezustand an. **dir**: Maske der Bits, die als Ausgabe gesetzt werden sollen **value**: Zustand der Ausgabebits **return**: Zustand der Eingabebits
Adc_Virtual::Handler	
	WORD onGetValue(BYTE ch) Client fordert Analogeingabe an **ch**: Eingabekanal **return**: Wert des Eingabekanals
Dac_Virtual::Handler	
	void onSetValue(BYTE ch, WORD value) Client setzt Analogausgabe **ch**: Ausgabekanal **value**: Wert des Ausgabekanals
DisplayChar_Virtual::Handler	
	void onClear(void) Client fordert Löschung des Bildschirms an
	void onWrite(WORD column, WORD line, char c) Client setzt Ausgabe eines Zeichens **column**: Spalte **line**: Zeile **c**: Zeichen (ASCII-Code)
DisplayGraphic_Virtual::Handler	
	void onClear(void) Client fordert Löschung des Bildschirms an
	void onSetPixel(WORD xPos, WORD yPos, WORD color)

	Client setzt Farbe eines einzelnen Pixels **xPos**, **yPos**: x/y-Position des Pixels **color**: Farbwert (RGB565)
	void onSetArea(WORD xPos, WORD yPos, WORD width, WORD height) Client setzt Bildschirmbereich, der mit nachfolgenden Aufrufen von **onSetPixel** sukzessive gefüllt wird **xPos**, **yPos**: x/y-Position des Bildschirmbereiches **width**,**height**: Breite und Höhe des Bildschirmbereiches
	void onSetPixel(WORD color) Client gibt an, mit welcher Farbe der nächste Bildpunkt im Bildschirmbereich (siehe **onSetArea**) gefüllt werden soll **color**: Farbwert (RGB565)
	void onSetRectangle(WORD xPos, WORD yPos, WORD width, WORD heigth, WORD color) Client fordert Zeichnen eines Rechteckes an **xPos**, **yPos**: x/y-Position des Bildschirmbereiches **width**, **height**: Breite und Höhe des Bildschirmbereiches **color**: Farbwert (RGB565)
Touch_Virtual::Handler	
	Data onGetTouch(void) Client fordert Zustandsdaten des Touch-Displays an **return**: Zustandsdaten (x,y,isTouched)

Tabelle 5.1: Call-Back-Methoden der Handler im Virtual-Server

Das folgende Beispiel zeigt exemplarisch einen Server zur Simulation eines DAC. In Zeile 3ff wird dazu eine Handler-Klasse definiert, dessen Konstruktor das UDPserver-Objekt an die Basisklasse weitergibt. Die Call-Back-Methode **`onSetValue()`** gibt hier den DAC-Wert auf dem Display aus.
Hinweis: Der Port 1000 wird standardmäßig auch vom VirtualDeviceServer bereitgestellt, daher sollte dieser nicht gleichzeitig laufen.

```
1:   #include "EmbSysLib.h"
2:
3:   using namespace EmbSysLib;
1:
2:   //*****************************************************************
3:   class MyDacHandler : public Hw::Dac_Virtual::Handler
4:   {
5:     public:
6:       //-------------------------------------------------------------
7:       MyDacHandler( Hw::UDPserver &s )
8:       : Hw::Dac_Virtual::Handler( s )
9:       {
10:      }
11:
12:      //-------------------------------------------------------------
13:      virtual void onSetValue( BYTE ch, WORD value )
14:      {
15:        printf("DAC (%d) %8d \r", ch, value );
16:      }
17:  };
18:
19:  //*****************************************************************
20:  int main()
21:  {
22:    Hw::UDPserver udp( 1000 /*port*/ );
23:
24:    MyDacHandler handler( udp );
25:
26:    while( 1 )
27:    {
28:      udp.run();
29:    }
30:  }
```

Listing 5.1: Beispielhaft Implementierung eines Servers für das Beispiel ***Example/Src/Main/Hardware/HwDac.cpp***

6 USB Descriptor

PHP-Skript:

Src/Hardware/USB/USBdevice/USB_Descriptor_Script.php

Das PHP-Skript generiert aus einer USB-Descriptor-Spezifikation einen CPP-Code, der die USB-Descriptoren für die Klasse **USBdevice** beinhaltet. Dazu wird die Spezifikation (Text-Datei) zunächst in ein JSON-Format überführt und anschließend interpretiert. Die vom Interpreter erzeugt CPP-Datei kann schließlich im Projekt eingebunden werden kann.

Die Spezifikation in Form enthält pro Zeile eine Zuweisung an ein Schlüsselwort oder definiert eine Gruppe (Configuration/ Interface/Endpoint etc.). Diese werden mit dem Gruppenbezeichner und einem **<BEGIN>** eingeleitet und mit **<END>** abgeschlossen. Falls mehrere gleichartige Gruppen (z.B. mehrere Endpoints) nacheinander definiert werden sollen, müssen nachfolgende Gruppen mit **<NEXT>** eingeleitet werden.

Die Schlüsselworte und Gruppierungen orientieren sich an den USB-Spezifikationen "USB 2.0 Specification" (Kap. 9.5ff), "Device Class Definition for HID" und "HID Usage Tables", siehe www.usb.org.

Strings, die unverändert übernommen werden sollen, werden mit ' eingeleitet und beendet.

Kommentare werden mit '**//**' eingeleitet und enden automatisch mit dem nächsten Zeilenumbruch.

Tabulatoren sind zur besseren Lesbarkeit empfehlenswert, haben keine weitere Bedeutung und werden vom Interpreter ignoriert.

Beispiele: **Example/Src/Main/Module/USB/..../descriptor.txt**
Example/Src/Main/Hardware/HwUSBdevice/descriptor.txt

Zur Ausführung der Skripte siehe auch Batch-Dateien **_build.bat** in den entsprechenden Verzeichnissen.

6.1 Schlüsselworte und Gruppen

6.1.1 Device

Key	Value	mandatory
Version	Number	y
Class	**RESERVED_CLASS** **AUDIO_CLASS** **COMMUNICATIONS_CLASS** **HUMAN_INTERFACE_DEVICE** **MONITOR_CLASS** **PHYSICAL_INTERFACE_CLASS** **POWER_CLASS** **PRINTER_CLASS** **STORAGE_CLASS** **HUB_CLASS** **MISCELLANEOUS** **VENDOR_SPECIFIC_CLASS**	y
SubClass	Number	y
Protocol	Number	y

MaxPacketSize	Number	y
VendorID	Number	y
ProductID	Number	y
Device	Number	y
ManufacturerStr	String	n
ProductStr	String	n
SerialNumberStr	String	n
CONFIGURATION	Group	Min. 1

6.1.2 Group 'CONFIGURATION'

Key	Value	mandatory
Name	String	n
SelfPowered	**yes | no**	n
RemoteWakeup	**yes | no**	n
MaxPower	Number	y
INTERFACE	Group	min. 1

6.1.3 Group 'INTERFACE'

Key	Value	mandatory
Name	String	n
Class	**VENDOR_SPECIFIC_CLASS** **CDC_COMMUNICATION** **CDC_DATA** **HUMAN_INTERFACE_DEVICE**	y
SubClass	Number	y
Protocol	Number	y
ENDPOINT	Group	n
HID	Group	n
COMPAT	Group	n
IAD	Group	n

6.1.4 Group 'ENDPOINT'

Key	Value	mandatory
Address	Number	n
Direction	**IN | OUT**	n
Attributes	**CONTROL** **ISOCHRONOUS** **BULK** **INTERRUPT**	y
MaxPacketSize	Number	y
Interval	Number	y

6.1.5 Group 'HID'

Key	Value	mandatory
Version	Number	y
Country	**NOT_SUPPORTED** **GERMAN** **INTERNATIONAL**	y
REPORT	Gruppe	min 1

6.1.6 Group 'REPORT'

Gruppe beginnt mit **<BEGIN_ARRAY>** und endet mit **<END_ARRAY>**

Key	Value
Input Output Feature	Any comma separated combination of: **DATA** **CONSTANT** **ARRAY** **VARIABLE** **ABSOLUTE** **RELATIVE** **WRAP** **NON_LINEAR** **NON_PREFERED** **NULL_STATE** **VOLATILE**
Collection	**PHYSICAL** **APPLICATION** **LOGICAL** **REPORT** **NAMED_ARRAY** **USAGE_SWITCH** **USAGE_MODIFIER**
EndCollection	---
ReportSize	Number (Byte)
ReportID	Number (Byte)
ReportCount	Number (Byte)
Push	---
Pop	---
UsagePage	**GenericDesktop** **KeyCodes** **Consumer** **LED** **Button**
Usage	**UsagePage: GenericDesktop**
	POINTER **MOUSE** **JOYSTICK** **GAME_PAD** **KEYBOARD** **KEYPAD** **MULTI_AXIS** **TABLET_PC** **X** **Y** **Z**

	RX RY RZ SLIDER DIAL WHEEL HAT_SWITCH COUNTED_BUFFER BYTE_COUNT MOTION_WAKEUP START SELECT VX VY VZ VBRX VBRY VBRZ VNO FEATURE_NOTIFICATION RESOLUTION_MULTIPLIER SYSTEM_CTL SYSCTL_POWER SYSCTL_SLEEP SYSCTL_WAKE SYSCTL_CONTEXT_MENU SYSCTL_MAIN_MENU SYSCTL_APP_MENU SYSCTL_HELP_MENU SYSCTL_MENU_EXIT SYSCTL_MENU_SELECT SYSCTL_MENU_RIGHT SYSCTL_MENU_LEFT SYSCTL_MENU_UP SYSCTL_MENU_DOWN COLD_RESTART WARM_RESTART D_PAD_UP D_PAD_DOWN D_PAD_RIGHT D_PAD_LEFT SYSTEM_DOCK SYSTEM_UNDOCK SYSTEM_SETUP SYSTEM_BREAK SYSTEM_DEGUGGER_BREAK APPLICATION_BREAK APPLICATION_DEBUG_BREAK SYSTEM_SPEAKER_MUTE SYSTEM_HIBERNATE SYSTEM_DISPLAY_INNVERT SYSTEM_DISPLAY_INTERNAL SYSTEM_DISPLAY_EXTERNAL SYSTEM_DISPLAY_BOTH SYSTEM_DISPLAY_DUAL SYSTEM_DISPLAY_TOGGLE
	UsagePage: Consumer
	CONSUMER_CONTROL VOLUME VOLUME_MUTE VOLUME_INCREMENT VOLUME_DECREMENT
UsageMin	Number (Byte)
UsageMax	Number (Byte)
LogicalMin	Number (Byte)

LogicalMinS	Number (Word)
LogicalMax	Number (Byte)
LogicalMaxS	Number (Word)
PhysicalMin	Number (Byte)
PhysicalMinS	Number (Word)
PhysicalMax	Number (Byte)
PhysicalMaxS	Number (Word)
UnitExponent	Number (Byte)
Unit	Number (Byte)
UnitS	Number (Word)

6.1.7 Group 'COMPAT'

OS Compatibility ID Feature, weist ein Gerät als 'WinUSB'-Device aus, das von Windows automatisch installiert wird. Siehe https://learn.microsoft.com/en-us/windows-hardware/drivers/usbcon/building-usb-devices-for-windows.

Key	Value	mandatory
ID	String 'WINUSB'	y
SubID	empty String	n

Hinweis:

Falls Probleme bei der automatischen Installation eines USB-Devices auftreten, könnte eine Ursache darin liegen, dass ein vorheriger Versuch gescheitert ist, den Decriptor zu lesen. Dieser Fehlversuch wird in der Registry eingetragen und Windows nimmt danach keine weiteren Leseversuche mehr vor, s.d. das Gerät nicht mehr automatisch als WinUSB installiert wird.
Lösung: In der Registry folgenden Eintrag löschen:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbflags\<vvvvppppprrrr>

*mit **vvvv** = vendor id, **pppp** = product id und **rrrr** = revision id*

Siehe auch: „https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/usb-device-specific-registry-settings“

6.1.8 Group 'IAD'

Interface Association Descriptor,
siehe: https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/usb-interface-association-descriptor

Key	Value	mandatory
Name	String	n
Count	Number	n
Class	**SPECIFIC_CLASS** **VENDOR_SPECIFIC_CLASS** **CDC_COMMUNICATION** **CDC_DATA** **HUMAN_INTERFACE_DEVICE**	y
SubClass	Number	y
Protocol	Number	y

7 Anhang

7.1 Abkürzungen

ADC	Analog Digital Converter
DAC	Digital Analog Converter
I2C	eigentlich „IIC“, Inter Integrated Circuit
MCU	Mikrocontroller Unit
RTC	Real Time Clock
SPI	Serial Peripheral Interface
UART	Universal Asychronous Receiver Transmitter
IDE	Intergrated Development Enviroment

Tabelle 7.1: Abkürzungen

7.2 Änderungshistorie

01.12.2022 Initiale Version
06.01.2023 Struktur verbessert, Hinweise ergänzt, neue IDE