USB-Servo Reference Manual

Generated by Doxygen 1.4.7

Sat Oct 28 14:16:28 2006

Contents

Chapter 1

USB-Servo

1.1 Introduction

The USB-Servo is a device to control a servo via USB. A servo is a motorized device that is commonly used in remote controlled cars and planes. I built this device to activate a toy puppet. The puppet has a button on its bottom, if you press the button the puppet collapses. When the computer is able to press the button, I can use the puppet to signal information like someone's online-state in the Jabber-network: when my friend goes online, the puppet stands up, when he logs off it collapses.

Servos are connected with three-wire-cables. A red and a black one for the power, and a yellow one for the signal. Power has to be between 4.8 and 6 volts, so the 5 volts from the USB-port is in the range. The signal doesn't take much current, so you can connect it directly to the controller. The angle of the servo is controlled with pulse width modulation (PWM). It gets a signal of about 50Hz (one pulse every 20ms), the length of the pulse tells the servo the angle to adjust.

A problem that I didn't really solve is the power consumption: I don't know the current that runs through the motor. It seems to be low enough not to cause any problems, but I don't know how high it will get when the servo is blocked. **YOU HAVE BEEN WARNED**, I don't feel responsible for USB-ports catching fire... :-/

There are three parts included in the distribution: The firmware for an ATmega8 microcontroller, a commandline-client that can be run under Linux, and the circuits needed to build the device.

This project is based on my USB-LED-Fader, which itself is based on the PowerSwitch example application by Objective Development. Like those, it uses Objective Development's firmware-only USB driver for Atmel's AVR microcontrollers.

Objective Development's USB driver is a firmware-only implementation of the USB 1.1 standard (low speed device) on cheap single chip microcomputers of Atmel's AVR series, such as the ATtiny2313 or even some of the small 8 pin devices. It implements the standard to the point where useful applications can be implemented. See the file "firmware/usbdrv/usbdrv.h" for features and limitations.

1.2 Building and installing

Both, the firmware and Unix command line tool are built with "make". You may need to customize both makefiles.

1.2.1 Firmware

The firmware for this project requires avr-gcc and avr-libc (a C-library for the AVR controller). Please read the instructions at http://www.nongnu.org/avr-libc/user-manual/install_tools.html for how to install the GNU toolchain (avr-gcc, assembler, linker etc.) and avr-libc.

Once you have the GNU toolchain for AVR microcontrollers installed, you can run "make" in the sub-directory "firmware". You may have to edit the Makefile to use your preferred downloader with "make program". The current version is built for avrdude with a parallel connection to an stk200-compatible programmer.

If working with a brand-new controller, you may have to set the fuse-bits to use the external crystal:

```
avrdude -p atmega8 -P /dev/parport0 -c stk200 -U hfuse:w:0xC9:m -U lfuse:w:0x9F:m
```

Afterwards, you can compile and flash to the device:

```
make program
```

1.2.2 Commandline client and demo application

The command line tool and the demo application require libusb. Please take the packages from your system's distribution or download libusb from http://libusb.sourceforge.net/ and install it before you compile. Change to directory "commandline", check the Makefile and edit the settings if required and type

```
make
```

This will build the unix executable "usb-servo" which can be used to control the device, and the demo application "xservopointer".

1.3 Usage

Connect the device to the USB-port. If it isn't already, the servo will move to the 0-position.

1.3.1 Commandline client

Use the commandline-client as follows:

```
usb-servo status
usb-servo set <angle>
usb-servo test
```

1.3.1.1 Parameters

- *angle:* The angle you want to set the servo to. 0 is full left, 255 is full right.

1.3.1.2 Examples

Get the status of the servo:

```
usb-servo status
```

This will tell you the angle the servo is currently put to.

```
Current servo angle: 42
```

Set a new angle:

```
usb-servo set 23
```

This sets the servo to the angle 23. 0 is full left, 255 is full right, so with 23 the servo will be almost on the left side.

Test the device:

```
usb-led-fader test
```

This function sends many random numbers to the device. The device returns the packages, and the client looks for differences in the sent and the received numbers.

1.3.2 Demo application xservopointer

This is a pure fun thing, nobody will need it. That was reason enough to write it...

To use it, you have to position the servo centered above the screen (with a little tweaking in the source, you can change that position). Then, you attach a pointer to the servo and start the application.

You'll never ever have to search for your mouse cursor in the future. The pointer on the servo will always show you where to search.

1.4 Drawbacks

The main drawback is the mentioned power consumption. I tested it with my servo on my notebook, it is not sure to work on other systems. **THIS MAY BE HARMFUL FOR YOUR COMPUTER**, and nobody but yourself will be responsible for any damages.

Another, not so big problem is the crude implementation of the PWM. I got the timing-values by trial and error, and they might not fit on your servo. On the other hand, I think that servos should be interchangeable. But this is my first and only one, so I can't say anything about that.

1.5 Files in the distribution

- *Readme.txt:* Documentation, created from the htmldoc-directory.
- *firmware:* Source code of the controller firmware.
- *firmware/usbdrv*: USB driver – See Readme.txt in this directory for info
- *commandline:* Source code of the host software (needs libusb). Here, you find the pure commandline client (usb-servo) and the fun demo application (xservopointer).
- *common:* Files needed by the firmware and the commandline-client.

- *circuit:* Circuit diagrams in PDF and EAGLE 4 format. A free version of EAGLE is available for Linux, Mac OS X and Windows from http://www.cadsoft.de/.
- *License.txt:* Public license for all contents of this project, except for the USB driver. Look in firmware/usbdrv/License.txt for further info.
- *Changelog.txt:* Logfile documenting changes in soft-, firm- and hardware.

1.6 Thanks!

I'd like to thank **Objective Development** for the possibility to use their driver for my project. In fact, this project wouldn't exist without the driver.

1.7 About the license

Chapter 2

USB-Servo File Index

2.1 USB-Servo File List

Here is a list of all files with brief descriptions:

Chapter 3

USB-Servo File Documentation

3.1 commandline/usb-servo.c File Reference

Commandline-tool for the USB-Servo.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usb.h>
#include "usbdrv.h"
```

Functions

- void usage (char *name)

  Displays usage-information.

- int main (int argc, char **argv)

  Main function.

3.1.1 Detailed Description

Commandline-tool for the USB-Servo.

Author:

Ronald Schaten

Version:

usb-led-fader.c,v 1.2 2006/10/01 16:28:38 rschaten Exp

License: See documentation.

Definition in file usb-servo.c.

3.1.2 Function Documentation

3.1.2.1 int main (int *argc*, char ** *argv*)

Main function.

Initializes the USB-device, parses commandline-parameters and calls the functions that communicate with the device.

Parameters:

argc Number of arguments.

argv Arguments.

Returns:

Error code.

Definition at line 39 of file usb-servo.c.

References dev_set(), dev_status(), dev_test(), handle, usage(), USBDEV_SHARED_PRODUCT, USBDEV_SHARED_VENDOR, and usbOpenDevice().

3.1.2.2 void usage (char * *name*)

Displays usage-information.

This function is called if the parameters cannot be parsed.

Parameters:

name The name of this application.

Definition at line 23 of file usb-servo.c.

Referenced by main().

3.2 commandline/usbdrv.c File Reference

USB-driver-parts for implementing a client.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usb.h>
#include "usbdrv.h"
#include "usbservo.h"
```

Functions

- int usbGetStringAscii (usb_dev_handle *dev, int index, int langid, char *buf, int buflen)
  Reads and converts a string from USB.
- int usbOpenDevice (usb_dev_handle **device, int vendor, char *vendorName, int product, char *productName)
  Connect to the USB-device.
- int dev_test (usb_dev_handle *handle, int argc, char **argv)
  Test connection to the device.
- int dev_set (usb_dev_handle *handle, int argc, char **argv)
  Set the angle of the Servo.
- int dev_status (usb_dev_handle *handle, int argc, char **argv)
  Get the status of the device.

3.2.1 Detailed Description

USB-driver-parts for implementing a client.

Author:

Ronald Schaten

Version:

usb-led-fader.c,v 1.2 2006/10/01 16:28:38 rschaten Exp

License: See documentation.

Definition in file usbdrv.c.

3.2.2 Function Documentation

3.2.2.1 int dev_set (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Set the angle of the Servo.

Parameters:

handle Handle to talk to the device.

argc Number of arguments.

argv Arguments.

Definition at line 162 of file usbdrv.c.

References CMD_SET.

Referenced by main().

3.2.2.2 int dev_status (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Get the status of the device.

Status information is printed in detail (we dont't have too many details with only one servo).

Parameters:

handle Handle to talk to the device.

argc Number of arguments.

argv Arguments.

Definition at line 186 of file usbdrv.c.

References CMD_GET.

Referenced by main().

3.2.2.3 int dev_test (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Test connection to the device.

The test consists of writing 1000 random numbers to the device and checking the echo. This should discover systematic bit errors (e.g. in bit stuffing).

Parameters:

handle Handle to talk to the device.

argc Number of arguments.

argv Arguments.

Definition at line 129 of file usbdrv.c.

References CMD_ECHO.

Referenced by main().

3.2.2.4 int usbGetStringAscii (usb_dev_handle * *dev*, int *index*, int *langid*, char * *buf*, int *buflen*)

Reads and converts a string from USB.

The conversion to ASCII is 'lossy' (unknown characters become '?').

Parameters:

dev Handle of the USB-Device.

index Index of the required data.

langid Index of the expected language.

buf Buffer to contain the return-string.

buflen Length of buf.

Returns:

Length of the string.

Definition at line 19 of file usbdrv.c.

Referenced by usbOpenDevice().

3.2.2.5 int usbOpenDevice (usb_dev_handle ** *device*, int *vendor*, char * *vendorName*, int *product*, char * *productName*)

Connect to the USB-device.

Loops through all connected USB-Devices and searches our counterpart.

Parameters:

device Handle to address the device.

vendor USBDEV_SHARED_VENDOR as defined.

vendorName In our case "www.schatenseite.de".

product USBDEV_SHARED_PRODUCT as defined.

productName In our case "USB-Servo".

Returns:

Error code.

Definition at line 52 of file usbdrv.c.

References handle, USB_ERROR_ACCESS, USB_ERROR_IO, USB_ERROR_NOTFOUND, and usbGetStringAscii().

Referenced by main().

3.3 commandline/usbdrv.h File Reference

USB-driver-parts for implementing a client.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usb.h>
#include "usbservo.h"
```

Defines

- #define USBDEV_SHARED_VENDOR 0x16C0

  VOTI.

- #define USBDEV_SHARED_PRODUCT 0x05DC

  Obdev's free shared PID.

- #define USB_ERROR_NOTFOUND 1

  Error code if the device isn't found.

- #define USB_ERROR_ACCESS 2

  Error code if the device isn't accessible.

- #define USB_ERROR_IO 3

  Error code if errors in the communication with the device occur.

Functions

- int usbGetStringAscii (usb_dev_handle *dev, int index, int langid, char *buf, int buflen)

  Reads and converts a string from USB.

- int usbOpenDevice (usb_dev_handle **device, int vendor, char *vendorName, int product, char *productName)

  Connect to the USB-device.

- int dev_test (usb_dev_handle *handle, int argc, char **argv)

  Test connection to the device.

- int dev_set (usb_dev_handle *handle, int argc, char **argv)

  Set the angle of the Servo.

- int dev_status (usb_dev_handle *handle, int argc, char **argv)

  Get the status of the device.

3.3.1 Detailed Description

USB-driver-parts for implementing a client.

Author:

Ronald Schaten

VersIdn:

usb-led-fader.c,v 1.2 2006/10/01 16:28:38 rschaten Exp

License: See documentation.

Definition in file usbdrv.h.

3.3.2 Define Documentation

3.3.2.1 #define USB_ERROR_ACCESS 2

Error code if the device isn't accessible.

Definition at line 22 of file usbdrv.h.

Referenced by usbOpenDevice().

3.3.2.2 #define USB_ERROR_IO 3

Error code if errors in the communication with the device occur.

Definition at line 23 of file usbdrv.h.

Referenced by usbOpenDevice().

3.3.2.3 #define USB_ERROR_NOTFOUND 1

Error code if the device isn't found.

Definition at line 21 of file usbdrv.h.

Referenced by usbOpenDevice().

3.3.2.4 #define USBDEV_SHARED_PRODUCT 0x05DC

Obdev's free shared PID.

Use obdev's generic shared VID/PID pair and follow the rules outlined in firmware/usbdrv/USBID-License.txt.

Definition at line 18 of file usbdrv.h.

Referenced by main().

3.3.2.5 #define USBDEV_SHARED_VENDOR 0x16C0

VOTI.

Definition at line 17 of file usbdrv.h.

Referenced by main().

3.3.3 Function Documentation

3.3.3.1 int dev_set (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Set the angle of the Servo.

Parameters:

handle Handle to talk to the device.
argc Number of arguments.
argv Arguments.

Definition at line 162 of file usbdrv.c.

References CMD_SET.

Referenced by main().

3.3.3.2 int dev_status (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Get the status of the device.

Status information is printed in detail (we dont't have too many details with only one servo).

Parameters:

handle Handle to talk to the device.
argc Number of arguments.
argv Arguments.

Definition at line 186 of file usbdrv.c.

References CMD_GET.

Referenced by main().

3.3.3.3 int dev_test (usb_dev_handle * *handle*, int *argc*, char ** *argv*)

Test connection to the device.

The test consists of writing 1000 random numbers to the device and checking the echo. This should discover systematic bit errors (e.g. in bit stuffing).

Parameters:

handle Handle to talk to the device.
argc Number of arguments.
argv Arguments.

Definition at line 129 of file usbdrv.c.

References CMD_ECHO.

Referenced by main().

3.3.3.4 int usbGetStringAscii (usb_dev_handle * *dev*, int *index*, int *langid*, char * *buf*, int *buflen*)

Reads and converts a string from USB.

The conversion to ASCII is 'lossy' (unknown characters become '?').

Parameters:

dev Handle of the USB-Device.

index Index of the required data.

langid Index of the expected language.

buf Buffer to contain the return-string.

buflen Length of buf.

Returns:

Length of the string.

Definition at line 19 of file usbdrv.c.

Referenced by usbOpenDevice().

3.3.3.5 int usbOpenDevice (usb_dev_handle ** *device*, int *vendor*, char * *vendorName*, int *product*, char * *productName*)

Connect to the USB-device.

Loops through all connected USB-Devices and searches our counterpart.

Parameters:

device Handle to address the device.

vendor USBDEV_SHARED_VENDOR as defined.

vendorName In our case "www.schatenseite.de".

product USBDEV_SHARED_PRODUCT as defined.

productName In our case "USB-Servo".

Returns:

Error code.

Definition at line 52 of file usbdrv.c.

References handle, USB_ERROR_ACCESS, USB_ERROR_IO, USB_ERROR_NOTFOUND, and usbGetStringAscii().

Referenced by main().

3.4 commandline/xservopointer.c File Reference

Tool that uses a servo to point to the mouse cursor under X.

```
#include <math.h>
#include <time.h>
#include <X11/Xlib.h>
#include <usb.h>
#include "usbdrv.h"
```

Functions

- void update ()
  Determines the current cursor position and sets the servo angle.
- int main (int argc, char *argv[ ])
  Main function.

Variables

- Display * dpy
- Window root
  The display to use.
- int rootwidth
  The root-window of the display.
- int rootheight
- int servoposx
  Measurements of the desktop.
- int servoposy
- usb_dev_handle * handle = NULL
  Position of the mouse cursor.

3.4.1 Detailed Description

Tool that uses a servo to point to the mouse cursor under X.

Author:

Ronald Schaten

Version:

usb-led-fader.c,v 1.2 2006/10/01 16:28:38 rschaten Exp

License: See documentation.

Definition in file xservopointer.c.

3.4.2 Function Documentation

3.4.2.1 int main (int *argc*, char * *argv*[ ])

Main function.

Initializes the X-settings and the USB-device, starts the timer and calls the update-function.

Parameters:

argc Number of arguments.

argv Arguments.

Returns:

Error code.

Definition at line 68 of file xservopointer.c.

References dpy, handle, root, rootheight, rootwidth, servoposx, servoposy, update(), USBDEV_-SHARED_PRODUCT, USBDEV_SHARED_VENDOR, and usbOpenDevice().

3.4.2.2 void update ()

Determines the current cursor position and sets the servo angle.

Definition at line 30 of file xservopointer.c.

References CMD_SET, dpy, handle, root, servoposx, and servoposy.

Referenced by main().

3.4.3 Variable Documentation

3.4.3.1 Display* dpy

Definition at line 19 of file xservopointer.c.

Referenced by main(), and update().

3.4.3.2 usb_dev_handle* handle = NULL

Position of the mouse cursor.

Definition at line 25 of file xservopointer.c.

Referenced by main(), update(), and usbOpenDevice().

3.4.3.3 Window root

The display to use.

Definition at line 20 of file xservopointer.c.

Referenced by main(), and update().

3.4.3.4 int rootheight

Definition at line 22 of file xservopointer.c.

Referenced by main().

3.4.3.5 int rootwidth

The root-window of the display.

Definition at line 22 of file xservopointer.c.

Referenced by main().

3.4.3.6 int servoposx

Measurements of the desktop.

Definition at line 23 of file xservopointer.c.

Referenced by main(), and update().

3.4.3.7 int servoposy

Definition at line 23 of file xservopointer.c.

Referenced by main(), and update().

3.5 common/usbservo.h File Reference

Global definitions and datatypes, used by the firmware and the commandline-client.

`#include <stdint.h>`

Defines

- #define msgOK 0

  Return code for OK.

- #define msgErr 1

  Return code for Error.

- #define CMD_ECHO 0

  Command to echo the sent data.

- #define CMD_GET 1

  Command to fetch values.

- #define CMD_SET 2

  Command to send values.

3.5.1 Detailed Description

Global definitions and datatypes, used by the firmware and the commandline-client.

Also contains the main doxygen-documentation.

Author:

Ronald Schaten

Version:

usbledfader.h,v 1.2 2006/10/01 16:28:01 rschaten Exp

License: See documentation.

Definition in file usbservo.h.

3.5.2 Define Documentation

3.5.2.1 #define CMD_ECHO 0

Command to echo the sent data.

Definition at line 213 of file usbservo.h.

Referenced by dev_test(), and usbFunctionSetup().

3.5.2.2 #define CMD_GET 1

Command to fetch values.

Definition at line 214 of file usbservo.h.

Referenced by dev_status(), and usbFunctionSetup().

3.5.2.3 #define CMD_SET 2

Command to send values.

Definition at line 215 of file usbservo.h.

Referenced by dev_set(), update(), and usbFunctionSetup().

3.5.2.4 #define msgErr 1

Return code for Error.

Definition at line 210 of file usbservo.h.

Referenced by usbFunctionSetup().

3.5.2.5 #define msgOK 0

Return code for OK.

Definition at line 209 of file usbservo.h.

Referenced by usbFunctionSetup().

3.6 firmware/main.c File Reference

Firmware for the USB-Servo.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include "usbdrv.h"
#include "oddebug.h"
#include "usbservo.h"
```

Functions

- uchar usbFunctionSetup (uchar data[8])
  USB-Setup-Handler.
- int main (void)
  Main-function.

3.6.1 Detailed Description

Firmware for the USB-Servo.

Author:

Ronald Schaten

Version:

main.c,v 1.2 2006/09/29 21:51:07 rschaten Exp

License: See documentation.

Definition in file main.c.

3.6.2 Function Documentation

3.6.2.1 int main (void)

Main-function.

Initializes the hardware and starts the main loop of the application.

Returns:

An integer. Whatever... :-)

Definition at line 79 of file main.c.

3.6.2.2 uchar usbFunctionSetup (uchar *data*[8])

USB-Setup-Handler.

Handles setup-calls that are received from the USB-Interface.

Parameters:

data Eight bytes of data.

Returns:

The number of returned bytes (in replyBuffer[]).

Definition at line 46 of file main.c.

References CMD_ECHO, CMD_GET, CMD_SET, msgErr, and msgOK.

3.7 firmware/usbconfig.h File Reference

Configuration of the USB-driver.

Defines

- #define USB_CFG_IOPORTNAME D
- #define USB_CFG_DMINUS_BIT 0
- #define USB_CFG_DPLUS_BIT 2
- #define USB_CFG_HAVE_INTRIN_ENDPOINT 0
- #define USB_CFG_IMPLEMENT_HALT 0
- #define USB_CFG_INTR_POLL_INTERVAL 10
- #define USB_CFG_IS_SELF_POWERED 1
- #define USB_CFG_MAX_BUS_POWER 20
- #define USB_CFG_SAMPLE_EXACT 0
- #define USB_CFG_IMPLEMENT_FN_WRITE 0
- #define USB_CFG_IMPLEMENT_FN_READ 0
- #define USB_CFG_VENDOR_ID 0xc0, 0x16
- #define USB_CFG_DEVICE_ID 0xdc, 0x05
- #define USB_CFG_DEVICE_VERSION 0x00, 0x01
- #define USB_CFG_VENDOR_NAME 'w', 'w', 'w', '.', 's', 'c', 'h', 'a', 't', 'e', 'n', 's', 'e', 'i', 't', 'e', '.', 'd', 'e'
- #define USB_CFG_VENDOR_NAME_LEN 19
- #define USB_CFG_DEVICE_NAME 'U', 'S', 'B', '-', 'S', 'e', 'r', 'v', 'o'
- #define USB_CFG_DEVICE_NAME_LEN 9
- #define USB_CFG_SERIAL_NUMBER_LENGTH 0
- #define USB_CFG_DEVICE_CLASS 0xff
- #define USB_CFG_DEVICE_SUBCLASS 0
- #define USB_CFG_INTERFACE_CLASS 0
- #define USB_CFG_INTERFACE_SUBCLASS 0
- #define USB_CFG_INTERFACE_PROTOCOL 0
- #define USB_CFG_HID_REPORT_DESCRIPTOR_LENGTH 0

3.7.1 Detailed Description

Configuration of the USB-driver.

Version:

usbconfig.h,v 1.2 2006/09/29 21:51:07 rschaten Exp

Definition in file usbconfig.h.

3.7.2 Define Documentation

3.7.2.1 #define USB_CFG_DEVICE_CLASS 0xff

Definition at line 155 of file usbconfig.h.

3.7.2.2 #define USB_CFG_DEVICE_ID 0xdc, 0x05

Definition at line 114 of file usbconfig.h.

3.7.2.3 #define USB_CFG_DEVICE_NAME 'U', 'S', 'B', '-', 'S', 'e', 'r', 'v', 'o'

Definition at line 134 of file usbconfig.h.

3.7.2.4 #define USB_CFG_DEVICE_NAME_LEN 9

Definition at line 135 of file usbconfig.h.

3.7.2.5 #define USB_CFG_DEVICE_SUBCLASS 0

Definition at line 156 of file usbconfig.h.

3.7.2.6 #define USB_CFG_DEVICE_VERSION 0x00, 0x01

Definition at line 121 of file usbconfig.h.

3.7.2.7 #define USB_CFG_DMINUS_BIT 0

Definition at line 38 of file usbconfig.h.

3.7.2.8 #define USB_CFG_DPLUS_BIT 2

Definition at line 42 of file usbconfig.h.

3.7.2.9 #define USB_CFG_HAVE_INTRIN_ENDPOINT 0

Definition at line 62 of file usbconfig.h.

3.7.2.10 #define USB_CFG_HID_REPORT_DESCRIPTOR_LENGTH 0

Definition at line 165 of file usbconfig.h.

3.7.2.11 #define USB_CFG_IMPLEMENT_FN_READ 0

Definition at line 100 of file usbconfig.h.

3.7.2.12 #define USB_CFG_IMPLEMENT_FN_WRITE 0

Definition at line 95 of file usbconfig.h.

3.7.2.13 #define USB_CFG_IMPLEMENT_HALT 0

Definition at line 66 of file usbconfig.h.

3.7.2.14 #define USB_CFG_INTERFACE_CLASS 0

Definition at line 159 of file usbconfig.h.

3.7.2.15 #define USB_CFG_INTERFACE_PROTOCOL 0

Definition at line 161 of file usbconfig.h.

3.7.2.16 #define USB_CFG_INTERFACE_SUBCLASS 0

Definition at line 160 of file usbconfig.h.

3.7.2.17 #define USB_CFG_INTR_POLL_INTERVAL 10

Definition at line 72 of file usbconfig.h.

3.7.2.18 #define USB_CFG_IOPORTNAME D

Definition at line 33 of file usbconfig.h.

3.7.2.19 #define USB_CFG_IS_SELF_POWERED 1

Definition at line 77 of file usbconfig.h.

3.7.2.20 #define USB_CFG_MAX_BUS_POWER 20

Definition at line 81 of file usbconfig.h.

3.7.2.21 #define USB_CFG_SAMPLE_EXACT 0

Definition at line 86 of file usbconfig.h.

3.7.2.22 #define USB_CFG_SERIAL_NUMBER_LENGTH 0

Definition at line 139 of file usbconfig.h.

3.7.2.23 #define USB_CFG_VENDOR_ID 0xc0, 0x16

Definition at line 109 of file usbconfig.h.

3.7.2.24 #define USB_CFG_VENDOR_NAME 'w', 'w', 'w', '.', 's', 'c', 'h', 'a', 't', 'e', 'n', 's', 'e', 'i', 't', 'e', '.', 'd', 'e'

Definition at line 124 of file usbconfig.h.

3.7.2.25 #define USB_CFG_VENDOR_NAME_LEN 19

Definition at line 125 of file usbconfig.h.

Index