USB-LED-Fader Reference Manual

Generated by Doxygen 1.4.7

Mon Oct 2 19:19:59 2006

Contents

1 USB-LED-Fader **1**

1.1 Introduction 1

1.2 Building and installing 1

1.3 Usage 2

1.4 Drawbacks 5

1.5 Files in the distribution 5

1.6 Thanks! 5

1.7 About the license 6

2 USB-LED-Fader Data Structure Index **7**

2.1 USB-LED-Fader Data Structures 7

3 USB-LED-Fader File Index **9**

3.1 USB-LED-Fader File List 9

4 USB-LED-Fader Data Structure Documentation **11**

4.1 S_fade_GlobalData Struct Reference 11

4.2 S_fade_LedState Struct Reference 12

4.3 S_fade_Waveform Struct Reference 14

4.4 S_messageQueue_GlobalData Struct Reference 16

4.5 S_pwm_Channels Struct Reference 18

4.6 S_pwm_Channels_ChannelBrightness Struct Reference 19

4.7 S_pwm_Channels_Message Struct Reference 20

4.8 S_pwm_Channels_Step Struct Reference 21

4.9 S_pwm_Timer_GlobalData Struct Reference 22

5 USB-LED-Fader File Documentation **25**

5.1 commandline/usb-led-fader.c File Reference 25

5.2 common/channels.h File Reference 31

5.3 common/usbledfader.h File Reference 32

5.4 firmware/boolean.h File Reference 36
5.5 firmware/config_message_queue.h File Reference 37
5.6 firmware/config_message_queue_impl.h File Reference 39
5.7 firmware/config_pwm_timer_impl.h File Reference 40
5.8 firmware/main.c File Reference 42
5.9 firmware/message_queue.c File Reference 45
5.10 firmware/message_queue.h File Reference 48
5.11 firmware/pwm_channels.c File Reference 51
5.12 firmware/pwm_channels.h File Reference 54
5.13 firmware/pwm_timer.c File Reference 57
5.14 firmware/pwm_timer.h File Reference 60
5.15 firmware/usbconfig.h File Reference 64

Chapter 1

USB-LED-Fader

1.1 Introduction

The USB-LED-Fader is a device to control a number of LEDs via USB. I built it to display the online-status of my internet-connection, the recording-status of my videorecorder, and warnings if the available disc-space is low. You can imagine an endless number of applications for this.

The LEDs are controlled with pulse width modulation (PWM). That way, they are not only on or off, it is possible to control the brightness. Included in the device is a number of ’waveforms’ that can be displayed on the LEDs. That way, one LED can display some kind of a sinus- or triangular wave without any interaction with the controlling host.

Every LED can be controlled individually, each one can display it’s own waveforms.

You can assign three different waves to every LED: two ’eternal’ waves (0 & 1). They are displayed alternating until anything different is required. The third wave (2) is only displayed once, afterwards the device will switch back to alternating between the first two waves.

One wave is described by three parameters: the waveform, the duration for one repetition of the wave and the number of repetitions before switching to the next wave.

This version supports four LEDs, it should be quite easy to change that number between one and eight. I have not tested any number greater than four, but I can imagine that the load on the controller can be too high to reliably communicate via USB.

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 the PowerSwitch example application by Objective Development. Like that, 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 subdirectory "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

The command line tool requires 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-led-fader" which can be used to control the device.

1.3 Usage

Connect the device to the USB-port. All LED should flash up to indicate that the device is initialized.

Then use the commandline-client as follows:

```
usb-led-fader status
usb-led-fader set <ledId> <waveId> <waveformId> <periodDuration> <repetitionCount>
usb-led-fader clear <ledId>
usb-led-fader reset
usb-led-fader show <waveformId>
usb-led-fader test
```

When using the set-function, it is possible to define several waves at once. You simply have to give the parameters for all waves. See examples below.

1.3.1 Parameters

- *ledId:* ID of the LED (0-n, depending on the number of LEDs in your circuit).
- *waveId:* ID of the wave (0-1: constant waves, 2: override).
- *waveformId:* ID of the waveform (0-31: brightness, 32-37: patterns). For a reference to the patterns, use the show-function.

- *periodDuration:* Time in sec/10 for one repetition of the waveform. A value of 0 can be used to reset the wave.
- *repetitionCount:* Number of repetitions before switching to the next wave. A value of 0 can be used to repeat this forever.

1.3.2 Examples

Get the status of all LEDs:

```
usb-led-fader status
```

This will result in an output similar to this:

```
LED 0          curid   curvalue     curpos     currep    nextupd
                   0          2         26          0         23
      wave  waveform     length     repeat   duration    updtime
         0        38         32          1         20         45
         1         0          1          1          0          1
         2         0          1          1          0          1
LED 1          curid   curvalue     curpos     currep    nextupd
                   0         14         19          0         19
      wave  waveform     length     repeat   duration    updtime
         0        38         32          1         20         45
         1         0          1          1          0          1
         2         0          1          1          0          1
LED 2          curid   curvalue     curpos     currep    nextupd
                   0         31         16          0         43
      wave  waveform     length     repeat   duration    updtime
         0        38         32          1         20         45
         1         0          1          1          0          1
         2         0          1          1          0          1
LED 3          curid   curvalue     curpos     currep    nextupd
                   0          6          9          0         39
      wave  waveform     length     repeat   duration    updtime
         0        38         32          1         20         45
         1         0          1          1          0          1
         2         0          1          1          0          1
```

In this output, the values curvalue, curpos, nextupd and updtime are for debugging purposes only. They shouldn't be of interest to the common user. The meaning of the other values should be clear.

Set the first LED to keep a middle brightness:

```
usb-led-fader set 0 0 15 10 1
```

So, on LED 0 the wave 0 is set to waveform 15. It will stay there for one second and will be repeated once before switching to the next wave. There is no next wave because we didn't define one, so this waveform will stay forever.

Now set a second wave on the first LED, a little brighter than the one before:

```
usb-led-fader set 0 1 25 10 1
```

This is wave 1 on LED 0, waveform 25 indicates a constant level of brightness. After setting the second wave, it will alternate with the first one after every second, because both waves have the same duration and the same number of repetitions.

Set a third wave on the first LED:

```
usb-led-fader set 0 2 36 20 5
```

This sets the third wave (wave 2) on the first LED. Waveform 36 is a nice sinus-like wave, so the LED starts to fade. One period of the fading takes 2 seconds, it is repeated for 5 times. Since this is the third wave, after the repetitions the LED returns to alternating between wave 0 and wave 1, this wave is discarded.

Set multiple waves at once:

```
usb-led-fader set 0 0 15 10 1 0 1 25 10 1 0 2 36 20 5
```

This will set all of the above waves at once. Thus, the first LED will first fade the sinus-wave five times, then start alternating between the two brightnesses in one-second-rhythm.

Clear the first LED:

```
usb-led-fader clear 0
```

This will clear all three waves on the first LED.

Reset the device:

```
usb-led-fader reset
```

All LEDs will flash once, to indicate that the device is reset and the LEDs are working.

Show a waveform on the screen:

```
usb-led-fader show 36
```

This will lead to an output like the following:

```
wave 36 - length 64
31:                              *****
30:                            *********
29:                           ***********
28:                         ***************
27:                        *****************
26:                       *******************
25:                       *******************
24:                      *********************
23:                     ***********************
22:                    *************************
21:                    *************************
20:                   ***************************
19:                  *****************************
18:                  *****************************
17:                 *******************************
16:                *********************************
15:               ***********************************
14:               ***********************************
13:              *************************************
12:             ***************************************
11:             ***************************************
10:            *****************************************
 9:           *******************************************
 8:          *********************************************
 7:          *********************************************
 6:         ***********************************************
 5:        *************************************************
 4:      *****************************************************
 3:     *******************************************************
 2:   ***********************************************************
 1: ****************************************************************
    ================================================================
```

Keep in mind that the width of the displayed wave corresponds to the length of the waveform. If you display a very simple one like the constant brightness levels (0-31), the length is 1. Therefore only one column is displayed.

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.

1.4 Drawbacks

As mentioned above, controlling the PWM for several LEDs is a lot of work for one small microcontroller. Speaking the USB protocol is so, either. Both combined result in a lot of load on the device, so the communication with the device is not 100% reliable. More than 99% though, at least in our tests.

SO BE WARNED: You should not use this device to control the state of your nuclear reactor. If you intend to use it in that way despite of this warning, please let me know... ;-)

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).
- *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.

And I'd like to give special credits to **Thomas Stegemann**. He wrote the PWM-stuff, and I guess it would have been nearly to impossible to me to write the rest of the project without his help since C isn't my natural language.

1.7 About the license

Our work - all contents except for the USB driver - are licensed under the GNU General Public License (GPL). A copy of the GPL is included in License.txt. The driver itself is licensed under a special license by Objective Development. See firmware/usbdrv/License.txt for further info.

(c) 2006 by Ronald Schaten - http://www.schatenseite.de

Chapter 2

USB-LED-Fader Data Structure Index

2.1 USB-LED-Fader Data Structures

Here are the data structures with brief descriptions:

S_fade_GlobalData (Contains the state of all four LEDs) 11
S_fade_LedState (The state of one LED) 12
S_fade_Waveform (Description of one waveform) 14
S_messageQueue_GlobalData (Structure of the global data of the queue) 16
S_pwm_Channels (Structure to contain the state of several channels) 18
S_pwm_Channels_ChannelBrightness (Structure to contain the state of one channel) 19
S_pwm_Channels_Message (Structure to contain an array of steps) 20
S_pwm_Channels_Step (Structure to contain one step) 21
S_pwm_Timer_GlobalData (Structure to contain the global data for the timer) 22

Chapter 3

USB-LED-Fader File Index

3.1 USB-LED-Fader File List

Here is a list of all files with brief descriptions:

commandline/usb-led-fader.c (Commandline-tool for the USB-LED-Fader) 25
common/channels.h (Global definitions, used by the firmware and the commandline-client) . . . 31
common/usbledfader.h (Global definitions and datatypes, used by the firmware and the commandline-client) 32
firmware/boolean.h (Provides boolean variables in C) 36
firmware/config_message_queue.h (Configures the message-queue) 37
firmware/config_message_queue_impl.h (Configures the implementation of the message-queue) 39
firmware/config_pwm_timer_impl.h (Configures the implementation of the PWM-timer) 40
firmware/main.c (Firmware for the USB-LED-Fader) 42
firmware/message_queue.c (A message queue used to exchange messages between two concurrent threads) 45
firmware/message_queue.h (A message queue used to exchange messages between two concurrent threads) 48
firmware/pwm_channels.c (Manages the values of the displayed channels) 51
firmware/pwm_channels.h (Manages the values of the displayed channels) 54
firmware/pwm_timer.c (Controls the actual PWM-output) 57
firmware/pwm_timer.h (Controls the actual PWM-output) 60
firmware/usbconfig.h (Configuration of the USB-driver) 64

Chapter 4

USB-LED-Fader Data Structure Documentation

4.1 S_fade_GlobalData Struct Reference

Contains the state of all four LEDs.

`#include <usbledfader.h>`

Data Fields

- fade_LedState led [4]

  Data for four LEDs.

4.1.1 Detailed Description

Contains the state of all four LEDs.

Definition at line 361 of file usbledfader.h.

4.1.2 Field Documentation

4.1.2.1 fade_LedState S_fade_GlobalData::led[4]

Data for four LEDs.

Definition at line 362 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

The documentation for this struct was generated from the following file:

- common/usbledfader.h

4.2 S_fade_LedState Struct Reference

The state of one LED.

`#include <usbledfader.h>`

Data Fields

- fade_Waveform wave [3]

  Three waveforms: base-function1, base-function2 and override-function.

- uint8_t waveCurrentId

  Which of the three waveforms is currently displayed?

- uint8_t waveCurrentValue

  The current brightness.

- uint8_t waveCurrentPosition

  Our position in the current waveform.

- uint8_t waveCurrentRepetition

  We are in the n-th repetition.

- int32_t waveNextUpdate

  Number of cycles till next update.

4.2.1 Detailed Description

The state of one LED.

Definition at line 351 of file usbledfader.h.

4.2.2 Field Documentation

4.2.2.1 fade_Waveform S_fade_LedState::wave[3]

Three waveforms: base-function1, base-function2 and override-function.

Definition at line 352 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.2.2.2 uint8_t S_fade_LedState::waveCurrentId

Which of the three waveforms is currently displayed?

Definition at line 353 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.2.2.3 uint8_t S_fade_LedState::waveCurrentPosition

Our position in the current waveform.

Definition at line 355 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.2.2.4 uint8_t S_fade_LedState::waveCurrentRepetition

We are in the n-th repetition.

Definition at line 356 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.2.2.5 uint8_t S_fade_LedState::waveCurrentValue

The current brightness.

Definition at line 354 of file usbledfader.h.

4.2.2.6 int32_t S_fade_LedState::waveNextUpdate

Number of cycles till next update.

Definition at line 357 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

The documentation for this struct was generated from the following file:

- common/usbledfader.h

4.3 S_fade_Waveform Struct Reference

Description of one waveform.

```
#include <usbledfader.h>
```

Data Fields

- uint8_t waveformId

  ID of this waveform.

- uint8_t waveformLength

  Length of this waveform.

- uint8_t waveformRepetition

  How often is this waveform to be repeated?

- uint8_t waveformDuration

  Duration for one cycle of this waveform, stored for status-output.

- uint32_t waveformUpdateTime

  Time between two waveform-samples in calls of timerInterrupt(), calculated from waveformDuration.

4.3.1 Detailed Description

Description of one waveform.

Definition at line 342 of file usbledfader.h.

4.3.2 Field Documentation

4.3.2.1 uint8_t S_fade_Waveform::waveformDuration

Duration for one cycle of this waveform, stored for status-output.

Definition at line 346 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.3.2.2 uint8_t S_fade_Waveform::waveformId

ID of this waveform.

Definition at line 343 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.3.2.3 uint8_t S_fade_Waveform::waveformLength

Length of this waveform.

Definition at line 344 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.3.2.4 uint8_t S_fade_Waveform::waveformRepetition

How often is this waveform to be repeated?

Definition at line 345 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

4.3.2.5 uint32_t S_fade_Waveform::waveformUpdateTime

Time between two waveform-samples in calls of timerInterrupt(), calculated from waveformDuration.

Definition at line 347 of file usbledfader.h.

Referenced by fade_globalData_init(), and fade_startWaveform().

The documentation for this struct was generated from the following file:

- common/usbledfader.h

4.4 S_messageQueue_GlobalData Struct Reference

Structure of the global data of the queue.

Data Fields

- messageQueue_QueuedMessage queue [messageQueue_Size]

  the data elements of the queue

- messageQueue_SizeType begin

  the current start of the queue

- messageQueue_SizeType end

  the current end of the queue, behind the last element

4.4.1 Detailed Description

Structure of the global data of the queue.

Definition at line 15 of file message_queue.c.

4.4.2 Field Documentation

4.4.2.1 messageQueue_SizeType S_messageQueue_GlobalData::begin

the current start of the queue

Definition at line 17 of file message_queue.c.

Referenced by messageQueue_init(), messageQueue_isEmpty(), messageQueue_isFull(), and message-Queue_read().

4.4.2.2 messageQueue_SizeType S_messageQueue_GlobalData::end

the current end of the queue, behind the last element

Definition at line 18 of file message_queue.c.

Referenced by messageQueue_init(), messageQueue_isEmpty(), messageQueue_isFull(), and message-Queue_write().

4.4.2.3 messageQueue_QueuedMessage S_messageQueue_GlobalData::queue[messageQueue_-Size]

the data elements of the queue

Definition at line 16 of file message_queue.c.

Referenced by messageQueue_read(), and messageQueue_write().

The documentation for this struct was generated from the following file:

- firmware/message_queue.c

4.5 S_pwm_Channels Struct Reference

Structure to contain the state of several channels.

`#include <pwm_channels.h>`

Data Fields

- pwm_Channels_Brightness channel [CHANNELS]

  Array of channels.

4.5.1 Detailed Description

Structure to contain the state of several channels.

Definition at line 30 of file pwm_channels.h.

4.5.2 Field Documentation

4.5.2.1 pwm_Channels_Brightness S_pwm_Channels::channel[CHANNELS]

Array of channels.

Definition at line 31 of file pwm_channels.h.

Referenced by pwm_Channels_show().

The documentation for this struct was generated from the following file:

- firmware/pwm_channels.h

4.6 S_pwm_Channels_ChannelBrightness Struct Reference

Structure to contain the state of one channel.

Data Fields

- pwm_Channels_Bitfield field

  Bitfield resembling one channel.

- pwm_Timer_Cycles cycle

  Number of on-cycles.

4.6.1 Detailed Description

Structure to contain the state of one channel.

Definition at line 18 of file pwm_channels.c.

4.6.2 Field Documentation

4.6.2.1 pwm_Timer_Cycles S_pwm_Channels_ChannelBrightness::cycle

Number of on-cycles.

Definition at line 20 of file pwm_channels.c.

Referenced by pwm_Channels_show().

4.6.2.2 pwm_Channels_Bitfield S_pwm_Channels_ChannelBrightness::field

Bitfield resembling one channel.

Definition at line 19 of file pwm_channels.c.

The documentation for this struct was generated from the following file:

- firmware/pwm_channels.c

4.7 S_pwm_Channels_Message Struct Reference

Structure to contain an array of steps.

`#include <pwm_timer.h>`

Data Fields

- pwm_Channels_Step step [pwm_Channels_StepCounter_Max]

  Array of steps.

4.7.1 Detailed Description

Structure to contain an array of steps.

Definition at line 61 of file pwm_timer.h.

4.7.2 Field Documentation

4.7.2.1 pwm_Channels_Step S_pwm_Channels_Message::step[pwm_Channels_StepCounter_-Max]

Array of steps.

Definition at line 62 of file pwm_timer.h.

Referenced by pwm_Timer_init(), and SIGNAL().

The documentation for this struct was generated from the following file:

- firmware/pwm_timer.h

4.8 S_pwm_Channels_Step Struct Reference

Structure to contain one step.

```
#include <pwm_timer.h>
```

Data Fields

- pwm_Timer_Cycles cycle

  Number of cycles to complete this step.

- pwm_Channels_Bitfield field

  The state of all channels.

4.8.1 Detailed Description

Structure to contain one step.

Definition at line 55 of file pwm_timer.h.

4.8.2 Field Documentation

4.8.2.1 pwm_Timer_Cycles S_pwm_Channels_Step::cycle

Number of cycles to complete this step.

Definition at line 56 of file pwm_timer.h.

Referenced by pwm_Timer_init(), and SIGNAL().

4.8.2.2 pwm_Channels_Bitfield S_pwm_Channels_Step::field

The state of all channels.

Definition at line 57 of file pwm_timer.h.

Referenced by pwm_Timer_init().

The documentation for this struct was generated from the following file:

- firmware/pwm_timer.h

4.9 S_pwm_Timer_GlobalData Struct Reference

Structure to contain the global data for the timer.

Data Fields

- pwm_Channels_Message message [2]

  Array of two messages.

- pwm_Channels_Message ∗ pActive

  Pointer to the active message.

- pwm_Channels_Message ∗ pRead

  Pointer to the message to read.

- pwm_Channels_StepCounter step

  Current step in the cycle.

- pwm_Timer_Cycles currentCycle

  Current cycle.

- Boolean readDone

  Indicates if something is read from the queue.

4.9.1 Detailed Description

Structure to contain the global data for the timer.

Definition at line 20 of file pwm_timer.c.

4.9.2 Field Documentation

4.9.2.1 pwm_Timer_Cycles S_pwm_Timer_GlobalData::currentCycle

Current cycle.

Definition at line 25 of file pwm_timer.c.

Referenced by pwm_Timer_init(), and SIGNAL().

4.9.2.2 pwm_Channels_Message S_pwm_Timer_GlobalData::message[2]

Array of two messages.

Definition at line 21 of file pwm_timer.c.

Referenced by pwm_Timer_init().

4.9.2.3 pwm_Channels_Message∗ S_pwm_Timer_GlobalData::pActive

Pointer to the active message.

Definition at line 22 of file pwm_timer.c.

Referenced by pwm_Timer_init(), and SIGNAL().

4.9.2.4 pwm_Channels_Message∗ S_pwm_Timer_GlobalData::pRead

Pointer to the message to read.

Definition at line 23 of file pwm_timer.c.

Referenced by pwm_Timer_init(), and SIGNAL().

4.9.2.5 Boolean S_pwm_Timer_GlobalData::readDone

Indicates if something is read from the queue.

Definition at line 26 of file pwm_timer.c.

Referenced by pwm_Timer_init(), and SIGNAL().

4.9.2.6 pwm_Channels_StepCounter S_pwm_Timer_GlobalData::step

Current step in the cycle.

Definition at line 24 of file pwm_timer.c.

Referenced by pwm_Timer_init(), and SIGNAL().

The documentation for this struct was generated from the following file:

- firmware/pwm_timer.c

Chapter 5

USB-LED-Fader File Documentation

5.1 commandline/usb-led-fader.c File Reference

Commandline-tool for the USB-LED-Fader.

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

- void usage (char *name)

Displays usage-informations.

- 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.

- void dev_test (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Test connection to the device.

- void dev_set (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Set waves.

- void dev_clear (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Clear all waves on one LED.

- void dev_status (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Get the status of the device.

- void dev_reset (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Reset the device.

- void dev_show (usb_dev_handle ∗handle, int argc, char ∗∗argv)

  Show a waveform.

- int main (int argc, char ∗∗argv)

  Main function.

5.1.1 Detailed Description

Commandline-tool for the USB-LED-Fader.

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 usb-led-fader.c.

5.1.2 Define Documentation

5.1.2.1 #define USB_ERROR_ACCESS 2

Error code if the device isn't accessible.

Definition at line 23 of file usb-led-fader.c.

Referenced by usbOpenDevice().

5.1.2.2 #define USB_ERROR_IO 3

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

Definition at line 24 of file usb-led-fader.c.

Referenced by usbOpenDevice().

5.1.2.3 #define USB_ERROR_NOTFOUND 1

Error code if the device isn't found.

Definition at line 22 of file usb-led-fader.c.

Referenced by usbOpenDevice().

5.1.2.4 #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 19 of file usb-led-fader.c.

Referenced by main().

5.1.2.5 #define USBDEV_SHARED_VENDOR 0x16C0

VOTI.

Definition at line 18 of file usb-led-fader.c.

Referenced by main().

5.1.3 Function Documentation

5.1.3.1 void dev_clear (usb_dev_handle $*$ *handle*, int *argc*, char $**$ *argv*)

Clear all waves on one LED.

Parameters:

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

Definition at line 261 of file usb-led-fader.c.

References CHANNELS, CMD_CLEAR, and usage().

Referenced by main().

5.1.3.2 void dev_reset (usb_dev_handle ∗ *handle*, int *argc*, char ∗∗ *argv*)

Reset the device.

Parameters:

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

Definition at line 330 of file usb-led-fader.c.

References CMD_RESET, and usage().

Referenced by main().

5.1.3.3 void dev_set (usb_dev_handle ∗ *handle*, int *argc*, char ∗∗ *argv*)

Set waves.

It is possible to set any number of waves at once.

Parameters:

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

Definition at line 204 of file usb-led-fader.c.

References CHANNELS, CMD_SET, and usage().

Referenced by main().

5.1.3.4 void dev_show (usb_dev_handle ∗ *handle*, int *argc*, char ∗∗ *argv*)

Show a waveform.

This will not send a command to the device, the waveform is only printed on the screen.

Parameters:

handle Handle to talk to the device (not needed).
argc Number of arguments.
argv Arguments.

Definition at line 351 of file usb-led-fader.c.

References fade_calculateWaveform(), and usage().

Referenced by main().

5.1.3.5 void dev_status (usb_dev_handle ∗ *handle*, int *argc*, char ∗∗ *argv*)

Get the status of the device.

Status information is printed in detail.

Parameters:

handle Handle to talk to the device.

argc Number of arguments.

argv Arguments.

Definition at line 286 of file usb-led-fader.c.

References CHANNELS, CMD_GET, and usage().

Referenced by main().

5.1.3.6 void 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 171 of file usb-led-fader.c.

References CMD_ECHO, and usage().

Referenced by main().

5.1.3.7 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 390 of file usb-led-fader.c.

References dev_clear(), dev_reset(), dev_set(), dev_show(), dev_status(), dev_test(), usage(), USBDEV_-SHARED_PRODUCT, USBDEV_SHARED_VENDOR, and usbOpenDevice().

5.1.3.8 void usage (char $*$ *name*)

Displays usage-informations.

This function is called if the parameters cannot be parsed.

Parameters:

name The name of this application.

Definition at line 31 of file usb-led-fader.c.

References CHANNELS.

Referenced by dev_clear(), dev_reset(), dev_set(), dev_show(), dev_status(), dev_test(), and main().

5.1.3.9 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 59 of file usb-led-fader.c.

Referenced by usbOpenDevice().

5.1.3.10 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-LED-Fader".

Returns:

Error code.

Definition at line 99 of file usb-led-fader.c.

References USB_ERROR_ACCESS, USB_ERROR_IO, USB_ERROR_NOTFOUND, and usbGetString-Ascii().

Referenced by main().

5.2 common/channels.h File Reference

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

Defines

- #define CHANNELS 4

  number of output channels

5.2.1 Detailed Description

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

Author:

Thomas Stegemann

VersIdn:

channels.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

Definition in file channels.h.

5.2.2 Define Documentation

5.2.2.1 #define CHANNELS 4

number of output channels

Definition at line 13 of file channels.h.

Referenced by dev_clear(), dev_set(), dev_status(), pwm_Timer_init(), and usage().

5.3 common/usbledfader.h File Reference

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

`#include <stdint.h>`

Data Structures

- struct S_fade_Waveform

  Description of one waveform.

- struct S_fade_LedState

  The state of one LED.

- struct S_fade_GlobalData

  Contains the state of all four LEDs.

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.

- #define CMD_CLEAR 3

  Command to switch off a certain LED.

- #define CMD_RESET 4

  Command to reset the whole device.

Typedefs

- typedef S_fade_Waveform fade_Waveform

  Description of one waveform.

- typedef S_fade_LedState fade_LedState

  The state of one LED.

- typedef S_fade_GlobalData fade_GlobalData

  Contains the state of all four LEDs.

Functions

- uint8_t fade_calculateWaveform (uint8_t waveformId, uint8_t waveformPosition)

  Calculate a waveform.

5.3.1 Detailed Description

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

Also contains the main doxygen-documentation.

Author:

Ronald Schaten & Thomas Stegemann

VersIdn:

usbledfader.h,v 1.3 2006/10/02 16:56:11 rschaten Exp

License: See documentation.

Definition in file usbledfader.h.

5.3.2 Define Documentation

5.3.2.1 #define CMD_CLEAR 3

Command to switch off a certain LED.

Definition at line 338 of file usbledfader.h.

Referenced by dev_clear(), and usbFunctionSetup().

5.3.2.2 #define CMD_ECHO 0

Command to echo the sent data.

Definition at line 335 of file usbledfader.h.

Referenced by dev_test(), and usbFunctionSetup().

5.3.2.3 #define CMD_GET 1

Command to fetch values.

Definition at line 336 of file usbledfader.h.

Referenced by dev_status(), and usbFunctionSetup().

5.3.2.4 #define CMD_RESET 4

Command to reset the whole device.

Definition at line 339 of file usbledfader.h.

Referenced by dev_reset(), and usbFunctionSetup().

5.3.2.5 #define CMD_SET 2

Command to send values.

Definition at line 337 of file usbledfader.h.

Referenced by dev_set(), and usbFunctionSetup().

5.3.2.6 #define msgErr 1

Return code for Error.

Definition at line 332 of file usbledfader.h.

Referenced by usbFunctionSetup().

5.3.2.7 #define msgOK 0

Return code for OK.

Definition at line 331 of file usbledfader.h.

Referenced by usbFunctionSetup().

5.3.3 Typedef Documentation

5.3.3.1 typedef struct S_fade_GlobalData fade_GlobalData

Contains the state of all four LEDs.

5.3.3.2 typedef struct S_fade_LedState fade_LedState

The state of one LED.

5.3.3.3 typedef struct S_fade_Waveform fade_Waveform

Description of one waveform.

5.3.4 Function Documentation

5.3.4.1 uint8_t fade_calculateWaveform (uint8_t *waveformId*, uint8_t *waveformPosition*)

Calculate a waveform.

Returns either the length of a given waveform or the output-level at a certain position in the wave.

Parameters:

waveformId ID of the waveform in question.

waveformPosition 0 or position in the given waveform.

Returns:

If the waveformPosition is 0, the number of steps in this waveform is returned. Otherwise the resulting output-level, an integer between 0 and 31.

Definition at line 374 of file usbledfader.h.

Referenced by dev_show(), fade_globalData_init(), and fade_startWaveform().

5.4 firmware/boolean.h File Reference

Provides boolean variables in C.

Typedefs

- typedef enum E_Boolean Boolean

  Possible boolean values.

Enumerations

- enum E_Boolean { False = 0, True = 1 }

  Possible boolean values.

5.4.1 Detailed Description

Provides boolean variables in C.

Author:

Thomas Stegemann

VersIdn:

boolean.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

Definition in file boolean.h.

5.4.2 Typedef Documentation

5.4.2.1 typedef enum E_Boolean Boolean

Possible boolean values.

5.4.3 Enumeration Type Documentation

5.4.3.1 enum E_Boolean

Possible boolean values.

Enumerator:

False	logical false
True	logical true

Definition at line 14 of file boolean.h.

5.5 firmware/config_message_queue.h File Reference

Configures the message-queue.

```
#include "pwm_timer.h"
```

Typedefs

- typedef pwm_Channels_Message messageQueue_QueuedMessage

Enumerations

- enum { messageQueue_Size = 3 }

5.5.1 Detailed Description

Configures the message-queue.

Author:

Thomas Stegemann

VersIdn:

config_message_queue.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- define the size of the messageQueue(messageQueue_Size) and the type of the messageQueue_-QueuedMessage
- check that messageQueue_SizeType can hold 0..messageQueue_Size+1
- the messageQueue buffers up to messageQueue_Size messages of the type messageQueue_Queued-Message
- currently the messageQueue is used by pwm_Channels and pwm_Timer with the pwm_Channels_-Message

Definition in file config_message_queue.h.

5.5.2 Typedef Documentation

5.5.2.1 typedef pwm_Channels_Message messageQueue_QueuedMessage

Definition at line 23 of file config_message_queue.h.

5.5.3 Enumeration Type Documentation

5.5.3.1 anonymous enum

Enumerator:

messageQueue_Size

Definition at line 24 of file config_message_queue.h.

5.6 firmware/config_message_queue_impl.h File Reference

Configures the implementation of the message-queue.

```
#include <stdint.h>
```

Typedefs

- typedef uint8_t messageQueue_SizeType

5.6.1 Detailed Description

Configures the implementation of the message-queue.

Author:

Thomas Stegemann

Version:

config_message_queue_impl.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- define the SizeType for the messageQueue
- the messageQueue_SizeType must hold 0..messageQueue_Size + 1, see config_message_queue.h
- the messageQueue_SizeType must be read/written by the processor in an atomic instruction

Definition in file config_message_queue_impl.h.

5.6.2 Typedef Documentation

5.6.2.1 typedef uint8_t messageQueue_SizeType

Definition at line 21 of file config_message_queue_impl.h.

5.7 firmware/config_pwm_timer_impl.h File Reference

Configures the implementation of the PWM-timer.

```
#include "pwm_channels.h"
```

Enumerations

- enum { pwm_Timer_Cycles_Max = pwm_Channels_Brightness_Max ∗ pwm_Channels_-Brightness_Max }
- enum { pwm_Timer_Cycles_ReadMin = 2 }
- enum { pwm_Timer_Cycles_SleepMax = 2 }

5.7.1 Detailed Description

Configures the implementation of the PWM-timer.

Author:

Thomas Stegemann

VersIdn:

config_pwm_timer_impl.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- pwm_Timer_Cycles_Max defines the number of (prescaled) processor cycles for a full pwm_Timer-Cycle
- pwm_Timer_Cycles_ReadMin defines the number of (prescaled) processor cycles the reading from the message queue may last
- pwm_Timer_Cycles_SleepMax defines the minimum number of (prescaled) processor cycles for which the timer is used. for less cycles the pwm_Timer waits active

Definition in file config_pwm_timer_impl.h.

5.7.2 Enumeration Type Documentation

5.7.2.1 anonymous enum

Enumerator:

pwm_Timer_Cycles_Max

Definition at line 23 of file config_pwm_timer_impl.h.

5.7.2.2 anonymous enum

Enumerator:

pwm_Timer_Cycles_ReadMin

Definition at line 24 of file config_pwm_timer_impl.h.

5.7.2.3 anonymous enum

Enumerator:

pwm_Timer_Cycles_SleepMax

Definition at line 25 of file config_pwm_timer_impl.h.

5.8 firmware/main.c File Reference

Firmware for the USB-LED-Fader.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include "usbdrv.h"
#include "oddebug.h"
#include "pwm_channels.h"
#include "usbledfader.h"
#include "channels.h"
```

Functions

- void fade_startWaveform (uint8_t ledId, uint8_t waveId, uint8_t waveformId, uint8_t periodDuration, uint8_t repetitionCount)

  Start displaying a certain waveform on a single LED.
- void fade_globalData_init (void)

  Fills fade_globalData.
- uchar usbFunctionRead (uchar ∗data, uchar len)

  USB-Data-Handler (device -> host).
- uchar usbFunctionWrite (uchar ∗data, uchar len)

  USB-Data-Handler (host -> device).
- uchar usbFunctionSetup (uchar data[8])

  USB-Setup-Handler.
- int main (void)

  Main-function.

5.8.1 Detailed Description

Firmware for the USB-LED-Fader.

Author:

Ronald Schaten & Thomas Stegemann

Version:

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

License: See documentation.

Definition in file main.c.

5.8.2 Function Documentation

5.8.2.1 void fade_globalData_init (void)

Fills fade_globalData.

The state of all LEDs is initialized to off. One signal is displayed on all LEDs to ensure they're working.

Definition at line 134 of file main.c.

References fade_calculateWaveform(), fade_startWaveform(), S_fade_GlobalData::led, S_fade_LedState::wave, S_fade_LedState::waveCurrentId, S_fade_LedState::waveCurrentPosition, S_fade_LedState::waveCurrentRepetition, S_fade_Waveform::waveformDuration, S_fade_Waveform::waveformId, S_fade_Waveform::waveformLength, S_fade_Waveform::waveformRepetition, S_fade_Waveform::waveformUpdateTime, and S_fade_LedState::waveNextUpdate.

Referenced by main(), and usbFunctionSetup().

5.8.2.2 void fade_startWaveform (uint8_t *ledId*, uint8_t *waveId*, uint8_t *waveformId*, uint8_t *periodDuration*, uint8_t *repetitionCount*)

Start displaying a certain waveform on a single LED.

Parameters:

ledId ID of the LED that is changed.

waveId ID of the wave that to be set: 0 and 1 are the base waves, 2 is the override wave.

waveformId ID of the Waveform that is to be assigned to the LED.

periodDuration How long should this wave stay on display? Time in seconds/10.

repetitionCount How many times should this wave be repeated while it is on display?

Definition at line 99 of file main.c.

References fade_calculateWaveform(), S_fade_GlobalData::led, S_fade_LedState::wave, S_fade_LedState::waveCurrentId, S_fade_LedState::waveCurrentPosition, S_fade_LedState::waveCurrentRepetition, S_fade_Waveform::waveformDuration, S_fade_Waveform::waveformId, S_fade_Waveform::waveformLength, S_fade_Waveform::waveformRepetition, S_fade_Waveform::waveformUpdateTime, and S_fade_LedState::waveNextUpdate.

Referenced by fade_globalData_init(), usbFunctionSetup(), and usbFunctionWrite().

5.8.2.3 int main (void)

Main-function.

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

Returns:

An integer. Whatever... :-)

Definition at line 247 of file main.c.

References fade_globalData_init(), and pwm_Channels_init().

5.8.2.4 uchar usbFunctionRead (uchar ∗ *data*, uchar *len*)

USB-Data-Handler (device -> host).

Handles data that is to be sent to the host via USB-Interface. In our case the data contains the current settings for the LEDs. This function is called until the returned length is shorter than the buffer (typically 8 bytes).

Parameters:

data Buffer for the data.

len Length of the buffer.

Returns:

Length of the returned buffer.

Definition at line 166 of file main.c.

5.8.2.5 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 203 of file main.c.

References CMD_CLEAR, CMD_ECHO, CMD_GET, CMD_RESET, CMD_SET, fade_globalData_-init(), fade_startWaveform(), msgErr, and msgOK.

5.8.2.6 uchar usbFunctionWrite (uchar ∗ *data*, uchar *len*)

USB-Data-Handler (host -> device).

Handles data that is received from the USB-Interface. In our case the data contains settings for the LEDs.

Parameters:

data The received data, up to 8 bytes.

len Length of the received data.

Returns:

1 if we have received the entire payload successfully, 0 if we expect more data. We don’t, so we always return 1.

Definition at line 184 of file main.c.

References fade_startWaveform().

5.9 firmware/message_queue.c File Reference

A message queue used to exchange messages between two concurrent threads.

```
#include <stdint.h>
#include "message_queue.h"
#include "config_message_queue_impl.h"
```

Data Structures

- struct S_messageQueue_GlobalData

  Structure of the global data of the queue.

Typedefs

- typedef S_messageQueue_GlobalData messageQueue_GlobalData

  Structure of the global data of the queue.

Functions

- void messageQueue_init (void)

  Initialize the queue.

- void messageQueue_cleanup (void)

  Clean up the queue.

- Boolean messageQueue_isEmpty (void)

  Test if the queue is empty.

- Boolean messageQueue_isFull (void)

  Test if the queue is full.

- Boolean messageQueue_read (messageQueue_QueuedMessage ∗pMessage)

  Read a message from the queue.

- Boolean messageQueue_write (messageQueue_QueuedMessage message)

  Write a message to the queue.

5.9.1 Detailed Description

A message queue used to exchange messages between two concurrent threads.

Author:

Thomas Stegemann

Version:

message_queue.c,v 1.2 2006/09/29 22:30:03 rschaten Exp

License: See documentation.

Definition in file message_queue.c.

5.9.2 Typedef Documentation

5.9.2.1 typedef struct S_messageQueue_GlobalData messageQueue_GlobalData

Structure of the global data of the queue.

5.9.3 Function Documentation

5.9.3.1 void messageQueue_cleanup (void)

Clean up the queue.

Currently this does nothing.

Definition at line 49 of file message_queue.c.

Referenced by pwm_Timer_cleanup().

5.9.3.2 void messageQueue_init (void)

Initialize the queue.

Definition at line 41 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by pwm_Timer_init().

5.9.3.3 Boolean messageQueue_isEmpty (void)

Test if the queue is empty.

Returns:

True if it is empty, otherwise false.

Definition at line 56 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by messageQueue_read().

5.9.3.4 Boolean messageQueue_isFull (void)

Test if the queue is full.

Returns:

True if it is full, otherwise false.

Definition at line 64 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by messageQueue_write().

5.9.3.5 Boolean messageQueue_read (messageQueue_QueuedMessage ∗ *pMessage*)

Read a message from the queue.

Parameters:

pMessage Pointer to a message variable that should be set to the message.

Returns:

True if an entry could be read, otherwise false.

Definition at line 74 of file message_queue.c.

References S_messageQueue_GlobalData::begin, messageQueue_isEmpty(), and S_messageQueue_-GlobalData::queue.

Referenced by SIGNAL().

5.9.3.6 Boolean messageQueue_write (messageQueue_QueuedMessage *message*)

Write a message to the queue.

Parameters:

message The message to append.

Returns:

True if the message could be appended, otherwise false.

Definition at line 88 of file message_queue.c.

References S_messageQueue_GlobalData::end, messageQueue_isFull(), and S_messageQueue_Global-Data::queue.

Referenced by pwm_Channels_show().

5.10 firmware/message_queue.h File Reference

A message queue used to exchange messages between two concurrent threads.

`#include "boolean.h"`

`#include "config_message_queue.h"`

Functions

- void messageQueue_init (void)

  Initialize the queue.

- void messageQueue_cleanup (void)

  Clean up the queue.

- Boolean messageQueue_isEmpty (void)

  Test if the queue is empty.

- Boolean messageQueue_isFull (void)

  Test if the queue is full.

- Boolean messageQueue_read (messageQueue_QueuedMessage ∗pMessage)

  Read a message from the queue.

- Boolean messageQueue_write (messageQueue_QueuedMessage message)

  Write a message to the queue.

5.10.1 Detailed Description

A message queue used to exchange messages between two concurrent threads.

Author:

Thomas Stegemann

VersIdn:

message_queue.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- exchange messages between two concurrent threads (e.g.: main thread and interrupt calls)
- before using any other function of the messageQueue, init must be called
- one thread must be data source (use isFull and write)
- the other thread must be the data sink (use isEmpty and read)
- two concurrent threads must not use both the write functions and two concurrent threads must not use both the read functions

- read/write return True on success and False if the message could not be read/written because the queue is empty/full
- the size of the messageQueue and the type of the messageQueue_QueuedMessage are defined in config_message_queue.h
- only one messageQueue can be used in a project

Definition in file message_queue.h.

5.10.2 Function Documentation

5.10.2.1 void messageQueue_cleanup (void)

Clean up the queue.

Currently this does nothing.

Definition at line 49 of file message_queue.c.

Referenced by pwm_Timer_cleanup().

5.10.2.2 void messageQueue_init (void)

Initialize the queue.

Definition at line 41 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by pwm_Timer_init().

5.10.2.3 Boolean messageQueue_isEmpty (void)

Test if the queue is empty.

Returns:

True if it is empty, otherwise false.

Definition at line 56 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by messageQueue_read().

5.10.2.4 Boolean messageQueue_isFull (void)

Test if the queue is full.

Returns:

True if it is full, otherwise false.

Definition at line 64 of file message_queue.c.

References S_messageQueue_GlobalData::begin, and S_messageQueue_GlobalData::end.

Referenced by messageQueue_write().

5.10.2.5 Boolean messageQueue_read (messageQueue_QueuedMessage ∗ *pMessage*)

Read a message from the queue.

Parameters:

pMessage Pointer to a message variable that should be set to the message.

Returns:

True if an entry could be read, otherwise false.

Definition at line 74 of file message_queue.c.

References S_messageQueue_GlobalData::begin, messageQueue_isEmpty(), and S_messageQueue_-GlobalData::queue.

Referenced by SIGNAL().

5.10.2.6 Boolean messageQueue_write (messageQueue_QueuedMessage *message*)

Write a message to the queue.

Parameters:

message The message to append.

Returns:

True if the message could be appended, otherwise false.

Definition at line 88 of file message_queue.c.

References S_messageQueue_GlobalData::end, messageQueue_isFull(), and S_messageQueue_Global-Data::queue.

Referenced by pwm_Channels_show().

5.11 firmware/pwm_channels.c File Reference

Manages the values of the displayed channels.

```
#include <stdlib.h>
#include "pwm_channels.h"
#include "pwm_timer.h"
#include "config_pwm_timer_impl.h"
#include "message_queue.h"
```

Data Structures

- struct S_pwm_Channels_ChannelBrightness
  Structure to contain the state of one channel.

Typedefs

- typedef S_pwm_Channels_ChannelBrightness pwm_Channels_ChannelBrightness
  Structure to contain the state of one channel.

Functions

- void pwm_Channels_init (void)
  Initialize channels.
- void pwm_Channels_cleanup (void)
  Clean up channels.
- pwm_Timer_Cycles pwm_Channels_BrightnessToCycles (pwm_Channels_Brightness brightness)
  Calculate number of cycles from a brightness.
- int pwm_Channels_CompareChannels (const void $*$cmp1, const void $*$cmp2)
  Compare the number of cycles in two channels.
- void pwm_Channels_show (pwm_Channels channels)
  Writes the current pattern to the message-queue.

5.11.1 Detailed Description

Manages the values of the displayed channels.

Author:

Thomas Stegemann

Version:

pwm_channels.c,v 1.2 2006/09/29 22:30:03 rschaten Exp

License: See documentation.

Definition in file pwm_channels.c.

5.11.2 Typedef Documentation

5.11.2.1 typedef struct S_pwm_Channels_ChannelBrightness pwm_Channels_ChannelBrightness

Structure to contain the state of one channel.

5.11.3 Function Documentation

5.11.3.1 pwm_Timer_Cycles pwm_Channels_BrightnessToCycles (pwm_Channels_Brightness *brightness*)

Calculate number of cycles from a brightness.

Parameters:

brightness The brightness.

Returns:

The number of cycles.

Definition at line 81 of file pwm_channels.c.

Referenced by pwm_Channels_show().

5.11.3.2 void pwm_Channels_cleanup (void)

Clean up channels.

Basically, the PWM-timer gets cleaned.

Definition at line 33 of file pwm_channels.c.

References pwm_Timer_cleanup().

5.11.3.3 int pwm_Channels_CompareChannels (const void $*$ *cmp1*, const void $*$ *cmp2*)

Compare the number of cycles in two channels.

This is needed for the qsort-call in pwm_Channels_show().

Parameters:

cmp1 First channel.

cmp2 Second channel.

Returns:

A value <0 if cmp1 is smaller than cmp2, 0 if they are of the same length and a value >0 if cmp1 is larger than cmp2.

Definition at line 93 of file pwm_channels.c.

Referenced by pwm_Channels_show().

5.11.3.4 void pwm_Channels_init (void)

Initialize channels.

Basically, only the PWM-timer is started.

Definition at line 26 of file pwm_channels.c.

References pwm_Timer_init().

Referenced by main().

5.11.3.5 void pwm_Channels_show (pwm_Channels *channels*)

Writes the current pattern to the message-queue.

The pattern is built from the state of all channels.

Parameters:

channels Array with the channel-states.

Definition at line 102 of file pwm_channels.c.

References S_pwm_Channels::channel, S_pwm_Channels_ChannelBrightness::cycle, messageQueue_-write(), pwm_Channels_BrightnessToCycles(), pwm_Channels_CompareChannels(), and pwm_Timer_-idle().

5.12 firmware/pwm_channels.h File Reference

Manages the values of the displayed channels.

```
#include <stdint.h>
#include "channels.h"
```

Data Structures

- struct S_pwm_Channels

  Structure to contain the state of several channels.

Typedefs

- typedef uint8_t pwm_Channels_Brightness

  Type to contain the brightness of one channel.

- typedef S_pwm_Channels pwm_Channels

  Structure to contain the state of several channels.

Enumerations

- enum { pwm_Channels_Brightness_Max = 31 }

  Definition of the maximum brightness.

Functions

- void pwm_Channels_init (void)

  Initialize channels.

- void pwm_Channels_cleanup (void)

  Clean up channels.

- void pwm_Channels_show (pwm_Channels channels)

  Writes the current pattern to the message-queue.

5.12.1 Detailed Description

Manages the values of the displayed channels.

Author:

Thomas Stegemann

Version:

pwm_channels.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- display the specified channels for a cycle of pwm_timer
- before using the function show, init must be called
- for every cycle of pwm_timer, show must be called
- show buffers the selected channels, so it returns immediatly, as long as the internal buffer is not full
- when the buffer is full the function blocks until another pwm_timer cycle has processed the current channels

Definition in file pwm_channels.h.

5.12.2 Typedef Documentation

5.12.2.1 typedef struct S_pwm_Channels pwm_Channels

Structure to contain the state of several channels.

5.12.2.2 typedef uint8_t pwm_Channels_Brightness

Type to contain the brightness of one channel.

Definition at line 24 of file pwm_channels.h.

5.12.3 Enumeration Type Documentation

5.12.3.1 anonymous enum

Definition of the maximum brightness.

Enumerator:

pwm_Channels_Brightness_Max

Definition at line 27 of file pwm_channels.h.

5.12.4 Function Documentation

5.12.4.1 void pwm_Channels_cleanup (void)

Clean up channels.

Basically, the PWM-timer gets cleaned.

Definition at line 33 of file pwm_channels.c.

References pwm_Timer_cleanup().

5.12.4.2 void pwm_Channels_init (void)

Initialize channels.

Basically, only the PWM-timer is started.

Definition at line 26 of file pwm_channels.c.

References pwm_Timer_init().

Referenced by main().

5.12.4.3 void pwm_Channels_show (pwm_Channels *channels*)

Writes the current pattern to the message-queue.

The pattern is built from the state of all channels.

Parameters:

channels Array with the channel-states.

Definition at line 102 of file pwm_channels.c.

References S_pwm_Channels::channel, S_pwm_Channels_ChannelBrightness::cycle, messageQueue_-write(), pwm_Channels_BrightnessToCycles(), pwm_Channels_CompareChannels(), and pwm_Timer_-idle().

5.13 firmware/pwm_timer.c File Reference

Controls the actual PWM-output.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include "boolean.h"
#include "message_queue.h"
#include "pwm_timer.h"
#include "config_pwm_timer_impl.h"
```

Data Structures

- struct S_pwm_Timer_GlobalData

  Structure to contain the global data for the timer.

Typedefs

- typedef S_pwm_Timer_GlobalData pwm_Timer_GlobalData

  Structure to contain the global data for the timer.

Functions

- void pwm_Timer_init (void)

  Initialize the PWM-Timer.
- void pwm_Timer_cleanup (void)

  Clean up the timer.
- void pwm_Timer_idle (void)

  Do nothing.
- SIGNAL (SIG_OUTPUT_COMPARE1A)

  Timer interrupt routine.

5.13.1 Detailed Description

Controls the actual PWM-output.

Author:

Thomas Stegemann

Version:

pwm_timer.c,v 1.2 2006/09/29 22:30:03 rschaten Exp

License: See documentation.

Definition in file pwm_timer.c.

5.13.2 Typedef Documentation

5.13.2.1 typedef struct S_pwm_Timer_GlobalData pwm_Timer_GlobalData

Structure to contain the global data for the timer.

5.13.3 Function Documentation

5.13.3.1 void pwm_Timer_cleanup (void)

Clean up the timer.

Basically, the message-queue is cleaned.

Definition at line 60 of file pwm_timer.c.

References messageQueue_cleanup().

Referenced by pwm_Channels_cleanup().

5.13.3.2 void pwm_Timer_idle (void)

Do nothing.

Definition at line 67 of file pwm_timer.c.

Referenced by pwm_Channels_show().

5.13.3.3 void pwm_Timer_init (void)

Initialize the PWM-Timer.

Sets basic values, starts the timer and initializes output-pins.

Definition at line 35 of file pwm_timer.c.

References CHANNELS, S_pwm_Timer_GlobalData::currentCycle, S_pwm_Channels_Step::cycle, False, S_pwm_Channels_Step::field, S_pwm_Timer_GlobalData::message, messageQueue_init(), S_-pwm_Timer_GlobalData::pActive, S_pwm_Timer_GlobalData::pRead, pwm_Channels_Brightness_Max, pwm_Timer_Cycles_Max, S_pwm_Timer_GlobalData::readDone, S_pwm_Timer_GlobalData::step, and S_pwm_Channels_Message::step.

Referenced by pwm_Channels_init().

5.13.3.4 SIGNAL (SIG_OUTPUT_COMPARE1A)

Timer interrupt routine.

Determines the pattern to set and handles the times to do PWM.

Definition at line 103 of file pwm_timer.c.

References S_pwm_Timer_GlobalData::currentCycle, S_pwm_Channels_Step::cycle, False, message-Queue_read(), S_pwm_Timer_GlobalData::pActive, S_pwm_Timer_GlobalData::pRead, pwm_-Channels_StepCounter_Max, pwm_Timer_Cycles_Max, pwm_Timer_Cycles_ReadMin, S_pwm_-Timer_GlobalData::readDone, S_pwm_Channels_Message::step, S_pwm_Timer_GlobalData::step, and True.

5.14 firmware/pwm_timer.h File Reference

Controls the actual PWM-output.

```
#include "pwm_channels.h"
```

Data Structures

- struct S_pwm_Channels_Step
  Structure to contain one step.
- struct S_pwm_Channels_Message
  Structure to contain an array of steps.

Typedefs

- typedef uint8_t pwm_Channels_Bitfield
  8-bit-field to contain the state of the channels.
- typedef uint8_t pwm_Channels_StepCounter
  Value to count the steps in one channel.
- typedef uint16_t pwm_Timer_Cycles
  Contains a number of controller-cycles.
- typedef S_pwm_Channels_Step pwm_Channels_Step
  Structure to contain one step.
- typedef S_pwm_Channels_Message pwm_Channels_Message
  Structure to contain an array of steps.

Enumerations

- enum { pwm_Channels_StepCounter_Max = CHANNELS + 1 }
  Definition of the maximum number of steps.

Functions

- void pwm_Timer_init (void)
  Initialize the PWM-Timer.
- void pwm_Timer_cleanup (void)
  Clean up the timer.
- void pwm_Timer_idle (void)
  Do nothing.

5.14.1 Detailed Description

Controls the actual PWM-output.

Author:

Thomas Stegemann

Version:

pwm_timer.h,v 1.1 2006/09/26 18:18:27 rschaten Exp

License: See documentation.

- read and process the pwm_Channels_Message from the messageQueue (written by pwm_Channels)
- use a timed interrupt to switch the led at a specified processor cycle
- init starts the processing and the timer
- idle is called by the pwm_Channels when the internal buffer is full
- at every pwm_timer cycle the leds can be switched in up to four steps every step defines which leds are switched on/off and up to which processor cycle the status is hold so the brightness for the three leds can be switched independently
- example:
  - start with all leds for 10 cycles:

    ```
    step[0]= {10, 1|2|4};
    ```

  - switch off the red led for further 10 cycles

    ```
    step[1]= {20,   2|4};
    ```

  - switch off the green led for further 10 cycles

    ```
    step[2]= {30,     4};
    ```

  - switch off all leds for the remaining time

    ```
    step[3]= {pwm_Timer_Cycles_Max, 0};
    ```

Definition in file pwm_timer.h.

5.14.2 Typedef Documentation

5.14.2.1 typedef uint8_t pwm_Channels_Bitfield

8-bit-field to contain the state of the channels.

Definition at line 43 of file pwm_timer.h.

5.14.2.2 typedef struct S_pwm_Channels_Message pwm_Channels_Message

Structure to contain an array of steps.

5.14.2.3 typedef struct S_pwm_Channels_Step pwm_Channels_Step

Structure to contain one step.

5.14.2.4 typedef uint8_t pwm_Channels_StepCounter

Value to count the steps in one channel.

Definition at line 46 of file pwm_timer.h.

5.14.2.5 typedef uint16_t pwm_Timer_Cycles

Contains a number of controller-cycles.

Definition at line 49 of file pwm_timer.h.

5.14.3 Enumeration Type Documentation

5.14.3.1 anonymous enum

Definition of the maximum number of steps.

Enumerator:

pwm_Channels_StepCounter_Max

Definition at line 52 of file pwm_timer.h.

5.14.4 Function Documentation

5.14.4.1 void pwm_Timer_cleanup (void)

Clean up the timer.

Basically, the message-queue is cleaned.

Definition at line 60 of file pwm_timer.c.

References messageQueue_cleanup().

Referenced by pwm_Channels_cleanup().

5.14.4.2 void pwm_Timer_idle (void)

Do nothing.

Definition at line 67 of file pwm_timer.c.

Referenced by pwm_Channels_show().

5.14.4.3 void pwm_Timer_init (void)

Initialize the PWM-Timer.

Sets basic values, starts the timer and initializes output-pins.

Definition at line 35 of file pwm_timer.c.

References CHANNELS, S_pwm_Timer_GlobalData::currentCycle, S_pwm_Channels_Step::cycle, False, S_pwm_Channels_Step::field, S_pwm_Timer_GlobalData::message, messageQueue_init(), S_-pwm_Timer_GlobalData::pActive, S_pwm_Timer_GlobalData::pRead, pwm_Channels_Brightness_Max, pwm_Timer_Cycles_Max, S_pwm_Timer_GlobalData::readDone, S_pwm_Channels_Message::step, and S_pwm_Timer_GlobalData::step.

Referenced by pwm_Channels_init().

5.15 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 1
- #define USB_CFG_IMPLEMENT_FN_READ 1
- #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', '-', 'L', 'E', 'D', '-', 'F', 'a', 'd', 'e', 'r'
- #define USB_CFG_DEVICE_NAME_LEN 13
- #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

5.15.1 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.

5.15.2 Define Documentation

5.15.2.1 #define USB_CFG_DEVICE_CLASS 0xff

Definition at line 155 of file usbconfig.h.

5.15.2.2 #define USB_CFG_DEVICE_ID 0xdc, 0x05

Definition at line 114 of file usbconfig.h.

5.15.2.3 #define USB_CFG_DEVICE_NAME 'U', 'S', 'B', '-', 'L', 'E', 'D', '-', 'F', 'a', 'd', 'e', 'r'

Definition at line 134 of file usbconfig.h.

5.15.2.4 #define USB_CFG_DEVICE_NAME_LEN 13

Definition at line 135 of file usbconfig.h.

5.15.2.5 #define USB_CFG_DEVICE_SUBCLASS 0

Definition at line 156 of file usbconfig.h.

5.15.2.6 #define USB_CFG_DEVICE_VERSION 0x00, 0x01

Definition at line 121 of file usbconfig.h.

5.15.2.7 #define USB_CFG_DMINUS_BIT 0

Definition at line 38 of file usbconfig.h.

5.15.2.8 #define USB_CFG_DPLUS_BIT 2

Definition at line 42 of file usbconfig.h.

5.15.2.9 #define USB_CFG_HAVE_INTRIN_ENDPOINT 0

Definition at line 62 of file usbconfig.h.

5.15.2.10 #define USB_CFG_HID_REPORT_DESCRIPTOR_LENGTH 0

Definition at line 165 of file usbconfig.h.

5.15.2.11 #define USB_CFG_IMPLEMENT_FN_READ 1

Definition at line 100 of file usbconfig.h.

5.15.2.12 #define USB_CFG_IMPLEMENT_FN_WRITE 1

Definition at line 95 of file usbconfig.h.

5.15.2.13 #define USB_CFG_IMPLEMENT_HALT 0

Definition at line 66 of file usbconfig.h.

5.15.2.14 #define USB_CFG_INTERFACE_CLASS 0

Definition at line 159 of file usbconfig.h.

5.15.2.15 #define USB_CFG_INTERFACE_PROTOCOL 0

Definition at line 161 of file usbconfig.h.

5.15.2.16 #define USB_CFG_INTERFACE_SUBCLASS 0

Definition at line 160 of file usbconfig.h.

5.15.2.17 #define USB_CFG_INTR_POLL_INTERVAL 10

Definition at line 72 of file usbconfig.h.

5.15.2.18 #define USB_CFG_IOPORTNAME D

Definition at line 33 of file usbconfig.h.

5.15.2.19 #define USB_CFG_IS_SELF_POWERED 1

Definition at line 77 of file usbconfig.h.

5.15.2.20 #define USB_CFG_MAX_BUS_POWER 20

Definition at line 81 of file usbconfig.h.

5.15.2.21 #define USB_CFG_SAMPLE_EXACT 0

Definition at line 86 of file usbconfig.h.

5.15.2.22 #define USB_CFG_SERIAL_NUMBER_LENGTH 0

Definition at line 139 of file usbconfig.h.

5.15.2.23 #define USB_CFG_VENDOR_ID 0xc0, 0x16

Definition at line 109 of file usbconfig.h.

5.15.2.24 #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.

5.15.2.25 #define USB_CFG_VENDOR_NAME_LEN 19

Definition at line 125 of file usbconfig.h.

Index

begin
  S_messageQueue_GlobalData, 16
Boolean
  boolean.h, 36
boolean.h
  Boolean, 36
  E_Boolean, 36
  False, 36
  True, 36

channel
  S_pwm_Channels, 18
CHANNELS
  channels.h, 31
channels.h
  CHANNELS, 31
CMD_CLEAR
  usbledfader.h, 33
CMD_ECHO
  usbledfader.h, 33
CMD_GET
  usbledfader.h, 33
CMD_RESET
  usbledfader.h, 33
CMD_SET
  usbledfader.h, 34
commandline/usb-led-fader.c, 25
common/channels.h, 31
common/usbledfader.h, 32
config_message_queue.h
  messageQueue_Size, 38
config_message_queue.h
  messageQueue_QueuedMessage, 37
config_message_queue_impl.h
  messageQueue_SizeType, 39
config_pwm_timer_impl.h
  pwm_Timer_Cycles_Max, 40
  pwm_Timer_Cycles_ReadMin, 40
  pwm_Timer_Cycles_SleepMax, 41
currentCycle
  S_pwm_Timer_GlobalData, 22
cycle
  S_pwm_Channels_ChannelBrightness, 19
  S_pwm_Channels_Step, 21

dev_clear
  usb-led-fader.c, 27
dev_reset
  usb-led-fader.c, 27
dev_set
  usb-led-fader.c, 28
dev_show
  usb-led-fader.c, 28
dev_status
  usb-led-fader.c, 28
dev_test
  usb-led-fader.c, 29

E_Boolean
  boolean.h, 36
end
  S_messageQueue_GlobalData, 16

fade_calculateWaveform
  usbledfader.h, 34
fade_GlobalData
  usbledfader.h, 34
fade_globalData_init
  main.c, 43
fade_LedState
  usbledfader.h, 34
fade_startWaveform
  main.c, 43
fade_Waveform
  usbledfader.h, 34
False
  boolean.h, 36
field
  S_pwm_Channels_ChannelBrightness, 19
  S_pwm_Channels_Step, 21
firmware/boolean.h, 36
firmware/config_message_queue.h, 37
firmware/config_message_queue_impl.h, 39
firmware/config_pwm_timer_impl.h, 40
firmware/main.c, 42
firmware/message_queue.c, 45
firmware/message_queue.h, 48
firmware/pwm_channels.c, 51
firmware/pwm_channels.h, 54
firmware/pwm_timer.c, 57

firmware/pwm_timer.h, 60
firmware/usbconfig.h, 64

led
    S_fade_GlobalData, 11

main
    main.c, 43
    usb-led-fader.c, 29
main.c
    fade_globalData_init, 43
    fade_startWaveform, 43
    main, 43
    usbFunctionRead, 43
    usbFunctionSetup, 44
    usbFunctionWrite, 44
message
    S_pwm_Timer_GlobalData, 22
message_queue.c
    messageQueue_cleanup, 46
    messageQueue_GlobalData, 46
    messageQueue_init, 46
    messageQueue_isEmpty, 46
    messageQueue_isFull, 46
    messageQueue_read, 47
    messageQueue_write, 47
message_queue.h
    messageQueue_cleanup, 49
    messageQueue_init, 49
    messageQueue_isEmpty, 49
    messageQueue_isFull, 49
    messageQueue_read, 49
    messageQueue_write, 50
messageQueue_cleanup
    message_queue.c, 46
    message_queue.h, 49
messageQueue_GlobalData
    message_queue.c, 46
messageQueue_init
    message_queue.c, 46
    message_queue.h, 49
messageQueue_isEmpty
    message_queue.c, 46
    message_queue.h, 49
messageQueue_isFull
    message_queue.c, 46
    message_queue.h, 49
messageQueue_QueuedMessage
    config_message_queue.h, 37
messageQueue_read
    message_queue.c, 47
    message_queue.h, 49
messageQueue_Size
    config_message_queue.h, 38
messageQueue_SizeType
    config_message_queue_impl.h, 39
messageQueue_write
    message_queue.c, 47
    message_queue.h, 50
msgErr
    usbledfader.h, 34
msgOK
    usbledfader.h, 34

pActive
    S_pwm_Timer_GlobalData, 22
pRead
    S_pwm_Timer_GlobalData, 23
pwm_Channels
    pwm_channels.h, 55
pwm_channels.c
    pwm_Channels_BrightnessToCycles, 52
    pwm_Channels_ChannelBrightness, 52
    pwm_Channels_cleanup, 52
    pwm_Channels_CompareChannels, 52
    pwm_Channels_init, 53
    pwm_Channels_show, 53
pwm_channels.h
    pwm_Channels_Brightness_Max, 55
pwm_channels.h
    pwm_Channels, 55
    pwm_Channels_Brightness, 55
    pwm_Channels_cleanup, 55
    pwm_Channels_init, 55
    pwm_Channels_show, 56
pwm_Channels_Bitfield
    pwm_timer.h, 61
pwm_Channels_Brightness
    pwm_channels.h, 55
pwm_Channels_Brightness_Max
    pwm_channels.h, 55
pwm_Channels_BrightnessToCycles
    pwm_channels.c, 52
pwm_Channels_ChannelBrightness
    pwm_channels.c, 52
pwm_Channels_cleanup
    pwm_channels.c, 52
    pwm_channels.h, 55
pwm_Channels_CompareChannels
    pwm_channels.c, 52
pwm_Channels_init
    pwm_channels.c, 53
    pwm_channels.h, 55
pwm_Channels_Message
    pwm_timer.h, 61
pwm_Channels_show
    pwm_channels.c, 53
    pwm_channels.h, 56

pwm_Channels_Step
  pwm_timer.h, 61
pwm_Channels_StepCounter
  pwm_timer.h, 62
pwm_Channels_StepCounter_Max
  pwm_timer.h, 62
pwm_timer.c
  pwm_Timer_cleanup, 58
  pwm_Timer_GlobalData, 58
  pwm_Timer_idle, 58
  pwm_Timer_init, 58
  SIGNAL, 58
pwm_timer.h
  pwm_Channels_StepCounter_Max, 62
pwm_timer.h
  pwm_Channels_Bitfield, 61
  pwm_Channels_Message, 61
  pwm_Channels_Step, 61
  pwm_Channels_StepCounter, 62
  pwm_Timer_cleanup, 62
  pwm_Timer_Cycles, 62
  pwm_Timer_idle, 62
  pwm_Timer_init, 62
pwm_Timer_cleanup
  pwm_timer.c, 58
  pwm_timer.h, 62
pwm_Timer_Cycles
  pwm_timer.h, 62
pwm_Timer_Cycles_Max
  config_pwm_timer_impl.h, 40
pwm_Timer_Cycles_ReadMin
  config_pwm_timer_impl.h, 40
pwm_Timer_Cycles_SleepMax
  config_pwm_timer_impl.h, 41
pwm_Timer_GlobalData
  pwm_timer.c, 58
pwm_Timer_idle
  pwm_timer.c, 58
  pwm_timer.h, 62
pwm_Timer_init
  pwm_timer.c, 58
  pwm_timer.h, 62

queue
  S_messageQueue_GlobalData, 16

readDone
  S_pwm_Timer_GlobalData, 23

S_fade_GlobalData, 11
S_fade_GlobalData
  led, 11
S_fade_LedState, 12
S_fade_LedState
  wave, 12
  waveCurrentId, 12
  waveCurrentPosition, 12
  waveCurrentRepetition, 13
  waveCurrentValue, 13
  waveNextUpdate, 13
S_fade_Waveform, 14
  waveformDuration, 14
  waveformId, 14
  waveformLength, 14
  waveformRepetition, 15
  waveformUpdateTime, 15
S_messageQueue_GlobalData, 16
S_messageQueue_GlobalData
  begin, 16
  end, 16
  queue, 16
S_pwm_Channels, 18
  channel, 18
S_pwm_Channels_ChannelBrightness, 19
S_pwm_Channels_ChannelBrightness
  cycle, 19
  field, 19
S_pwm_Channels_Message, 20
  step, 20
S_pwm_Channels_Step, 21
  cycle, 21
  field, 21
S_pwm_Timer_GlobalData, 22
S_pwm_Timer_GlobalData
  currentCycle, 22
  message, 22
  pActive, 22
  pRead, 23
  readDone, 23
  step, 23
SIGNAL
  pwm_timer.c, 58
step
  S_pwm_Channels_Message, 20
  S_pwm_Timer_GlobalData, 23

True
  boolean.h, 36

usage
  usb-led-fader.c, 29
usb-led-fader.c
  dev_clear, 27
  dev_reset, 27
  dev_set, 28
  dev_show, 28
  dev_status, 28
  dev_test, 29

main, 29
usage, 29
USB_ERROR_ACCESS, 26
USB_ERROR_IO, 27
USB_ERROR_NOTFOUND, 27
USBDEV_SHARED_PRODUCT, 27
USBDEV_SHARED_VENDOR, 27
usbGetStringAscii, 30
usbOpenDevice, 30
USB_CFG_DEVICE_CLASS
usbconfig.h, 64
USB_CFG_DEVICE_ID
usbconfig.h, 64
USB_CFG_DEVICE_NAME
usbconfig.h, 65
USB_CFG_DEVICE_NAME_LEN
usbconfig.h, 65
USB_CFG_DEVICE_SUBCLASS
usbconfig.h, 65
USB_CFG_DEVICE_VERSION
usbconfig.h, 65
USB_CFG_DMINUS_BIT
usbconfig.h, 65
USB_CFG_DPLUS_BIT
usbconfig.h, 65
USB_CFG_HAVE_INTRIN_ENDPOINT
usbconfig.h, 65
USB_CFG_HID_REPORT_DESCRIPTOR_-LENGTH
usbconfig.h, 65
USB_CFG_IMPLEMENT_FN_READ
usbconfig.h, 65
USB_CFG_IMPLEMENT_FN_WRITE
usbconfig.h, 65
USB_CFG_IMPLEMENT_HALT
usbconfig.h, 65
USB_CFG_INTERFACE_CLASS
usbconfig.h, 66
USB_CFG_INTERFACE_PROTOCOL
usbconfig.h, 66
USB_CFG_INTERFACE_SUBCLASS
usbconfig.h, 66
USB_CFG_INTR_POLL_INTERVAL
usbconfig.h, 66
USB_CFG_IOPORTNAME
usbconfig.h, 66
USB_CFG_IS_SELF_POWERED
usbconfig.h, 66
USB_CFG_MAX_BUS_POWER
usbconfig.h, 66
USB_CFG_SAMPLE_EXACT
usbconfig.h, 66
USB_CFG_SERIAL_NUMBER_LENGTH
usbconfig.h, 66
USB_CFG_VENDOR_ID
usbconfig.h, 66
USB_CFG_VENDOR_NAME
usbconfig.h, 66
USB_CFG_VENDOR_NAME_LEN
usbconfig.h, 67
USB_ERROR_ACCESS
usb-led-fader.c, 26
USB_ERROR_IO
usb-led-fader.c, 27
USB_ERROR_NOTFOUND
usb-led-fader.c, 27
usbconfig.h
USB_CFG_DEVICE_CLASS, 64
USB_CFG_DEVICE_ID, 64
USB_CFG_DEVICE_NAME, 65
USB_CFG_DEVICE_NAME_LEN, 65
USB_CFG_DEVICE_SUBCLASS, 65
USB_CFG_DEVICE_VERSION, 65
USB_CFG_DMINUS_BIT, 65
USB_CFG_DPLUS_BIT, 65
USB_CFG_HAVE_INTRIN_ENDPOINT, 65
USB_CFG_HID_REPORT_-DESCRIPTOR_LENGTH, 65
USB_CFG_IMPLEMENT_FN_READ, 65
USB_CFG_IMPLEMENT_FN_WRITE, 65
USB_CFG_IMPLEMENT_HALT, 65
USB_CFG_INTERFACE_CLASS, 66
USB_CFG_INTERFACE_PROTOCOL, 66
USB_CFG_INTERFACE_SUBCLASS, 66
USB_CFG_INTR_POLL_INTERVAL, 66
USB_CFG_IOPORTNAME, 66
USB_CFG_IS_SELF_POWERED, 66
USB_CFG_MAX_BUS_POWER, 66
USB_CFG_SAMPLE_EXACT, 66
USB_CFG_SERIAL_NUMBER_-LENGTH, 66
USB_CFG_VENDOR_ID, 66
USB_CFG_VENDOR_NAME, 66
USB_CFG_VENDOR_NAME_LEN, 67
USBDEV_SHARED_PRODUCT
usb-led-fader.c, 27
USBDEV_SHARED_VENDOR
usb-led-fader.c, 27
usbFunctionRead
main.c, 43
usbFunctionSetup
main.c, 44
usbFunctionWrite
main.c, 44
usbGetStringAscii
usb-led-fader.c, 30

usbledfader.h
    CMD_CLEAR, 33
    CMD_ECHO, 33
    CMD_GET, 33
    CMD_RESET, 33
    CMD_SET, 34
    fade_calculateWaveform, 34
    fade_GlobalData, 34
    fade_LedState, 34
    fade_Waveform, 34
    msgErr, 34
    msgOK, 34
usbOpenDevice
    usb-led-fader.c, 30

wave
    S_fade_LedState, 12
waveCurrentId
    S_fade_LedState, 12
waveCurrentPosition
    S_fade_LedState, 12
waveCurrentRepetition
    S_fade_LedState, 13
waveCurrentValue
    S_fade_LedState, 13
waveformDuration
    S_fade_Waveform, 14
waveformId
    S_fade_Waveform, 14
waveformLength
    S_fade_Waveform, 14
waveformRepetition
    S_fade_Waveform, 15
waveformUpdateTime
    S_fade_Waveform, 15
waveNextUpdate
    S_fade_LedState, 13