RoboWay

Generated by Doxygen 1.7.6.1

Wed Mar 26 2014 01:44:37

Contents

Chapter 1

This is the documentation for developers and people interested in functionality of RoboWay - Inverted Pendulum

RoboWay is an educational project.

Its body and its wheels are 3d printed.

The two controller boards are self made.

I've chosen this project because of the limitations of micro controllers.

I think today you can realize almost everything if you have enough resources.

So normally you don't have to care about the efficiency of your algorithms and programs.

But an inverted pendulum such as RoboWay is very time-critical and you have to write very efficient and lightweight code because of the limitations of embedded hardware.

A video is available here

Used electronic components

- **Sonar sensor SRF08**

  Actually not assembled

  See srf08.h

  See srf08.c

- **9 degrees of freedom IMU GY80**

  Three axis accelerometer ADXL354

  See adxl354.h

See adxl354.c

Three axis gyroscope L3G4200D

See l3g4200d.h

See l3g4200d.c

- **Micro controller** `AtMaga 328P`

  i2cmaster.h

  i2cmaster.c

  usart.h

  usart.c

  avrConstants.h

  avrConstants.c

- **DC motor driver L293D**

  motorControll.h

  motorControll.c

Wiring diagrams:

Voltage regulator and motor controller Micro controller board and sensors

How does RoboWay works

As you can see in the diagram below RoboWay uses an accelerometer and a gyroscope with a complementary filter to calculate its angular position.

The angular position and a reference (BALANCE_REF) value is the input for a PID controller.

The output of the PID is used as PWM output for the motor control (motorControll.h).

Future versions of RoboWay

This is just a prototype of a self balancing robot.

So there are many little bugs in hardware and software.

As example:

- The mechanical characteristics of motors with a gear are not as good as they should be.

  I think this must be changed at first for newer versions

- The controller board could be much better. As example the ISP-Header and three potentiometers for the PID settings are missing

- The settings for the pid controller is not perfect

Source disclosure

- http://www.rn-wissen.de/index.php/Regelungstechnik (09.03.2014)
- https://github.com/astromaf/VertiBOT/tree/master/-Firmwares/PID_VertiBOT_V3_2 (09.03.2014)
- http://www.pieter-jan.com/node/11 (09.03.2014)
- AVR-Mikrocontroller-Kochbuch (Entwurf und Programmierung praktischer - Anwendungen) - Franzis
- Powerprojekte mit Arduino und C - Franzis (ISBN 978-3-645-65131-8)
- Data sheet of gyroscope L3G4200D (rev. 3)
- Data sheet of accelerometer ADXL354 (Alaog Devices)
- http://www.ti.com/lit/ds/symlink/l293.pdf (09.03.2014)
- Data sheet of micro controller AtMega328P (Atmel rev. Rev. 8161D ? 10/09)
- Data sheet of sonar sensor SRF08

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 Alias names of ADXL354 Registers (refer to site 24-27 of the ADXL354 datasheet)

Defines

- #define ADXL354_REG_DEVID 0x00

  The DEVID holds a fixed device ID (see ADXL354_STD_DEVID)
- #define ADXL354_REG_THRESH_TAP 0x1D

  The THRESH_TAP register holds the threshold value for tap interrupts Data format is unsigned.
- #define ADXL354_REG_OFSX 0x1E

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_OFSY 0x1F

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_OFSZ 0x20

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_TAP_DUR 0x21

  Maximum time than an event must be above the THRESH_TAP to qualifiy as a tap event (625us/LSB) A value of 0 disables the single / double tap events.
- #define ADXL354_REG_TAP_LATENT 0x22

  Latency from detection of a tap to start the time window (1.25 ms/LSB)
- #define ADXL354_REG_TAP_WINDOW 0x23

  Time window from one to a second valid tap.
- #define ADXL354_REG_THRESH_ACT 0x24

  Threshold value for detecting activity unsigned (62.5mg/LSB)
- #define ADXL354_REG_THRESH_INACT 0x25

  Threshold value for detecting inactivity unsigned (62.5mg/LSB)
- #define ADXL354_REG_TIME_INACT 0x26

Time value that acceleration must be under the value of ADXL354_REG_THRESH_INACT to generate an inactivity interrupt unsigned (1s/LSB)

- #define ADXL354_REG_ACT_INACT_CTL 0x27

  Control register control activity and inactivity interrupts.
- #define ADXL354_REG_THRESH_FF 0x28

  This register holds the threshold value for free-fall detection unsigned (62.5mg/LSB) a value between 300 and 600mg is recommended.
- #define ADXL354_REG_TIME_FF 0x29

  Minimum time that the value of all axis must be less than ADXL354_REG_THRESH_FF unsigned (5ms/LSB)
- #define ADXL354_REG_TAP_AXES_CTL 0x2A

  Enable/disable tap interrupts.
- #define ADXL354_REG_ACT_TAP_STATUS 0x2B

  Act/Tap status - indicates witch axis was first involved in an tap or activity event READ ONLY.
- #define ADXL354_REG_CTL_BW_RATE 0x2C

  Set bandwidth and output data rate Default: 0x0A -> 100Hz See ADXL354_REG_BW_RATE_MODES.
- #define ADXL354_REG_CTL_POWER 0x2D

  Power saving features control.
- #define ADXL354_REG_INT_EN 0x2E

  Interrupt enable / disable Refer to site 20 / 26.
- #define ADXL354_REG_INT_MAP 0x2F

  Interrupt and pin mappings.
- #define ADXL354_REG_INT_SRC 0x30

  Interrupt sources - READ ONLY.
- #define ADXL354_REG_DATA_FORMAT 0x31

  Controls the presentation of data to register 0x32 through register 0x37.
- #define ADXL354_REG_DATAX_L 0x32

  Data of X-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAX_H 0x33

  Data of X-Axis High Byte (READ ONLY)
- #define ADXL354_REG_DATAY_L 0x34

  Data of Y-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAY_H 0x35

  Data of Y-Axis High Byte (READ ONLY)
- #define ADXL354_REG_DATAZ_L 0x36

  Data of Z-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAZ_H 0x37

  Data of Z-Axis High Byte (READ ONLY)
- #define ADXL354_REG_CTL_FIFO 0x38

  FIFO Modes.
- #define ADXL354_REG_FIFO_STATUS 0x39

  FIFO Status.

5.2 Possible bitrates for bit rate register (refert to table 8 in datasheet)

Defines

- #define ADXL354_REG_BW_RATE_400 0b1100

  Value for ADXL354_REG_BW_RATE in I2C 400 kHz mode.
- #define ADXL354_REG_BW_RATE_100 0b1010

  Value for ADXL354_REG_BW_RATE in I2C 100 kHz mode.

5.3 Alias names of ADXL354 AXIS

Defines

- #define ADXL354_AXIS_X ADXL354_REG_DATAX_L

  Start register of X-Axis.
- #define ADXL354_AXIS_Y ADXL354_REG_DATAY_L

  Start register of Y-Axis.
- #define ADXL354_AXIS_Z ADXL354_REG_DATAZ_L

  Start register of Z-Axis.

5.4 Macros for simpler use of the ADXL354

Defines

- #define ADXL354_GET_ACCEL_X(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_X, val)

  Get the current acceleration of X-Axis.
- #define ADXL354_GET_ACCEL_Y(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_Y, val)

  Get the current acceleration of Y-Axis.
- #define ADXL354_GET_ACCEL_Z(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_Z, val)

  Get the current acceleration of Z-Axis.

5.5 Settings for AtMega 8 bit micro controllers

Defines

- #define F_CPU 16000000UL

  Clockrate of the CPU.

5.6 Constants for AtMega 8 bit micro controllers

Defines

- #define false 0

  Boolean false.
- #define FALSE false
- #define true 1

  Boolean true.
- #define TRUE true
- #define high 1

  Signal high (a.e. for an IO-Pin)
- #define HIGH high
- #define low 0

  Signal low (a.e. for an IO-Pin)
- #define LOW low
- #define bool uint8_t

  Data type for boolean variables.
- #define BOOL bool

5.6.1 Define Documentation

5.6.1.1 #define BOOL bool

See also

bool

5.6.1.2 #define FALSE false

See also

false

5.6.1.3 #define HIGH high

See also

high

5.6.1.4 #define LOW low

See also

low

5.6.1.5 #define TRUE true

See also

true

5.7 Useful macros for AtMega 8 bit micro controllers

Defines

- #define RAD_TO_DEG(x) ((x) * 57.295779513082320876798154814105)

  Converts an angle from rad to deg.
- #define LOW_BYTE(x) ((uint8_t)((x) & 0xFF))

  Returns low byte as uint8_t
  x should be 2 bytes.
- #define HIGH_BYTE(x) ((uint8_t)(((x) >> 8) & 0xFF))

  Returns high byte as uint8_t
  x should be 2 bytes.

5.8 Macros used to modify the address of the slave for read or write operations

Defines

- #define I2C_READ(addr) (((addr) << 1) | 0x01)
  Modify the address of the slave for read operations.
- #define I2C_WRITE(addr) (((addr) << 1) | 0x00)
  Modify the address of the slave for write operations.

5.8.1 Define Documentation

5.8.1.1 #define I2C_READ(*addr*) (((addr) << 1) | 0x01)

Modify the address of the slave for read operations.

Parameters

addr	Address of the I2C slave

5.8.1.2 #define I2C_WRITE(*addr*) (((addr) << 1) | 0x00)

Modify the address of the slave for write operations.

Parameters

addr	Address of the I2C slave

5.9 Useful macros for I2C

Defines

- #define i2cRepStart(addr) (i2cStart(addr))

  Issues a repeated start condition, sends address and transfer direction This is just an alias of uint8_t i2cStart(uint8_t addr)

5.10 Status codes of I2C functions

Defines

- #define I2C_OK TRUE

  Status Code OK.
- #define I2C_FAIL FALSE

  Status Code Failure.

5.11 Modes for I2C clockrate

Defines

- #define I2C_SCL_NORMAL 10000UL

  Normal Clockrate (100kHz)
- #define I2C_SCL_HIGH 40000UL

  High Clockrate (400kHz)

5.12 Possible device ids of an L3G4200D Gyroscope

Defines

- #define L3G4200D_DEVID_0 0x68

  Possible address 0 of an L3G4200D on I2C.
- #define L3G4200D_DEVID_1 0x69

  Possible address 1 of an L3G4200D on I2C.

5.13 All necessary registers of an L3G4200D

Defines

- #define L3G4200D_REG_WAI 0x0F

  Device identification register (Who Am I)
- #define L3G4200D_REG_CTRL_1 0x20

  Control register 1
  Refer to table 20 of in data sheet.
- #define L3G4200D_REG_CTRL_2 0x21

  Control register 2
  Refer to table 28 in data sheet.
- #define L3G4200D_REG_CTRL_3 0x22

  Control register 3
  Refer to table 28 in data sheet.
- #define L3G4200D_REG_CTRL_4 0x23

  Control register 4
  Refer to table 30 in data sheet.
- #define L3G4200D_REG_CTRL_5 0x24

  Control register 5
  Refer to table 33 in data sheet.
- #define L3G4200D_REG_REF 0x25

  Reference value for interrupt generation
  Refer to table 37 / 38 in data sheet.
- #define L3G4200D_REG_TEMP 0x26

  Temperature data
  Refer to table 39 / 40 in data sheet.
- #define L3G4200D_REG_STATUS 0x27

  Status register
  Refer to table 41 / 42 in data sheet.
- #define L3G4200D_REG_DATAX_L 0x28

  X-Axis angular rate data - Low Byte.
- #define L3G4200D_REG_DATAX_H 0x29

  X-Axis angular rate data - High Byte.
- #define L3G4200D_REG_DATAY_L 0x2A

  Y-Axis angular rate data - Low Byte.
- #define L3G4200D_REG_DATAY_H 0x2B

  Y-Axis angular rate data - High Byte.
- #define L3G4200D_REG_DATAZ_L 0x2C

  Z-Axis angular rate data - Low Byte.
- #define L3G4200D_REG_DATAZ_H 0x2D

  Z-Axis angular rate data - High Byte.
- #define L3G4200D_REG_CTRL_FIFO 0x2E

  Register for FIFO configuration
  Refer to table 43 / 44 / 45.

- #define L3G4200D_REG_SRC_FIFO 0x2F

  Source configuration for FIFO
  Refer to table 46 / 47.
- #define L3G4200D_REG_CFG_INT1 0x30

  Configuration register for interrupt 1
  Refer to table 48 / 49.
- #define L3G4200D_REG_SRC_INT1 0x31

  Configuration register for interrupt source
  Refer to table 50 / 51.
- #define L3G4200D_REG_INT1_TSH_XH 0x32

  Interrupt threshold X-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_XL 0x33

  Interrupt threshold X-Axis - Low Byte.
- #define L3G4200D_REG_INT1_TSH_YH 0x34

  Interrupt threshold Y-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_YL 0x35

  Interrupt threshold Y-Axis - Low Byte.
- #define L3G4200D_REG_INT1_TSH_ZH 0x36

  Interrupt threshold Z-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_ZL 0x37

  Interrupt threshold Z-Axis - Low Byte.
- #define L3G4200D_REG_INT1_DUR 0x38

  Minimum, wait and max duration of an interrupt
  Refer to table 64 / 65 and figure 20 / 21 in data sheet.

5.14 Possible Bit and Data Rates for L3G4200D_REG_CTRL_1 (refer to table 22 in data sheet)

Defines

- #define L3G4200D_BW_RATE_100 0b00110000

  Bit and Data Rate for 100 kHz I2C.
- #define L3G4200D_BW_RATE_400 0b10000000

  Bit and Data Rate for 400 kHz I2C.

5.15 Alias names of L3G4200D Axis

Defines

- #define L3G4200D_AXIS_X L3G4200D_REG_DATAX_L

  Start register of X-Axis.
- #define L3G4200D_AXIS_Y L3G4200D_REG_DATAY_L

  Start register of Y-Axis.
- #define L3G4200D_AXIS_Z L3G4200D_REG_DATAZ_L

  Start register of Z-Axis.

5.16 Macros for simpler use of the L3G4200D

Defines

- #define L3G4200D_GET_AR_X(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAX_L, val)

  Get current angular rate data of X-Axis.
- #define L3G4200D_GET_AR_Y(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAY_L, val)

  Get current angular rate data of Y-Axis.
- #define L3G4200D_GET_AR_Z(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAZ_L, val)

  Get current angular rate data of Z-Axis.

5.17 Pins used for controlling left and right wheel

Defines

- #define MC_PORT PORTD

  AVR PORT used for motor control pins.
- #define MC_DDR DDRD

  AVR DDR used for motor control pins.
- #define MC_LEFT PD5

  Pin for speed left wheel.
- #define MC_LEFT_FW PD2

  Left wheel drives in forward direction if this pin is set to high.
- #define MC_LEFT_BW PD3

  Left wheel drives in backward direction if this pin is set to high.
- #define MC_RIGHT PD6

  Pin for speed right wheel.
- #define MC_RIGHT_FW PD4

  right wheel drives in forward direction if this pin is set to high
- #define MC_RIGHT_BW PD7

  Right wheel drives in forward direction if this pin is set to high.

5.18 Registers for output compare match (0 -> 0% PWM | 255 -> 100% PWM)

Defines

- #define MC_LEFT_OCR OCR0B

  Output compare register for left wheel.
- #define MC_RIGHT_OCR OCR0A

  Output compare register for right wheel.

5.19 Maximum and minimum roll

Defines

- #define ROLL_MIN_LIMIT 55.0

  Minimum value for roll.
- #define ROLL_MAX_LIMIT 135.0

  Maximum value for roll.

5.19.1 Define Documentation

5.19.1.1 #define ROLL_MAX_LIMIT 135.0

Maximum value for roll.

If the measured roll is over this value, RoboWay stops balancing

See also

ROLL_MIN_LIMIT
void roboWayLoop(void)

5.19.1.2 #define ROLL_MIN_LIMIT 55.0

Minimum value for roll.

If the measured roll is under this value, RoboWay stops balancing

See also

ROLL_MAX_LIMIT
void roboWayLoop(void)

5.20 Gain and limits for conservative pid settings (balancing)

Defines

- #define K_B_C 1.0

  Over all gain.
- #define K_P_B_C 1.86

  Gain proportional part.
- #define K_I_B_C 0.030

  Gain integral part.
- #define K_D_B_C 0.006

  Gain differential part.
- #define MIN_OUT_B_C -255.0

  Minimal value for pid output.
- #define MAX_OUT_B_C 255.0

  Maximum value for pid output.

5.21 Possible values for the address register

Defines

- #define SRF08_ADDR_00 0xE0

  Possible address 0.
- #define SRF08_ADDR_01 0xE2

  Possible address 1.
- #define SRF08_ADDR_02 0xE4

  Possible address 2.
- #define SRF08_ADDR_03 0xE6

  Possible address 3.
- #define SRF08_ADDR_04 0xE8

  Possible address 4.
- #define SRF08_ADDR_05 0xEA

  Possible address 5.
- #define SRF08_ADDR_06 0xEC

  Possible address 6.
- #define SRF08_ADDR_07 0xEE

  Possible address 7.
- #define SRF08_ADDR_08 0xF0

  Possible address 8.
- #define SRF08_ADDR_09 0xF2

  Possible address 9.
- #define SRF08_ADDR_10 0xF4

  Possible address 10.
- #define SRF08_ADDR_11 0xF6

  Possible address 11.
- #define SRF08_ADDR_12 0xF8

  Possible address 12.
- #define SRF08_ADDR_13 0xFA

  Possible address 13.
- #define SRF08_ADDR_14 0xFC

  Possible address 14.
- #define SRF08_ADDR_15 0xFE

  Possible address 15.

5.22 Modify an address for read or write operation for the I2C bus

Defines

- #define SRF08_ADDR_READ(x) ((x) + 0x01)

  Modify the address for read operation.
- #define SRF08_ADDR_WRITE(x) ((x))

  Modify the address for write operation.

5.22.1 Define Documentation

5.22.1.1 #define SRF08_ADDR_READ(*x*) ((x) + 0x01)

Modify the address for read operation.

Warning

Do not use I2C_READ(addr) or i2C_WRITE(addr) from i2cmaster.h

5.22.1.2 #define SRF08_ADDR_WRITE(*x*) ((x))

Modify the address for write operation.

Warning

Do not use I2C_READ(addr) or i2C_WRITE(addr) from i2cmaster.h

5.23 Registers of SRF08

Defines

- #define SRF08_REG_VERSION 0x00

  Version of the SRF08 is stored here.
- #define SRF08_REG_COMMAND SRF08_REG_VERSION

  Register for commands - See SRF08_COMMANDS for possible values.
- #define SRF08_REG_LIGHT 0x01

  Value of measured light is stored here.
- #define SRF08_REG_GAIN SRF08_REG_LIGHT

  Register for setting the gain.
- #define SRF08_REG_ECHO_H 0x02

  Value of measured distance - high byte.
- #define SRF08_REG_RANGE SRF08_REG_ECHO_H

  Range in mm (val ∗ 43mm) + 43mm.
- #define SRF08_REG_ECHO_L 0x03

  Value of measured distance - low byte.

5.24 Commands for SRF08

Defines

- #define SRF08_COM_INCH 0x50

  Start measure in inch.
- #define SRF08_COM_CM 0x51

  Start measure in cm.
- #define SRF08_COM_MKS 0x52

  Start measure in microseconds.
- #define SRF08_COM_ADDR0 0xA0

  Sequence one for changing the I2C-Address.
- #define SRF08_COM_ADDR1 0xAA

  Sequence two for changing the I2C-Address.
- #define SRF08_COM_ADDR2 0xA5

  Sequence three for changing the I2C-Address.

5.25 Useful macros for atmega8 usart (RS232)

Defines

- #define USART_BAUD_CALC(x) ((F_CPU / (16UL ∗ (x))) - 1)

  Calculate value for the prescaler of the usart depending on F_CPU and baud rate.
- #define USART_DOUBLE_BAUD_CALC(x) ((F_CPU / (8UL ∗ (x))) - 1)

  Calculate value for the prescaler of the usart
  It is used if the baud rate is out of range.

Chapter 6

Data Structure Documentation

6.1 compFilterData Struct Reference

Data structure for complementary filter.

`#include <complementaryFilter.h>`

Data Fields

- double gG

  Gain for high pass filter of gyroscope.

- double aG

  Gain for low pass filter of accelerometer.

- double aO

  Old angle.

- double dt

  Sampling rate in seconds.

6.1.1 Detailed Description

Data structure for complementary filter.

See also

void compFilterInit(double gyroGain, double accGain, double dt, compFilterData *fd)
double compFilterGetAngle(double gyroData, double accData, compFilterData *fd)

6.1.2 Field Documentation

6.1.2.1 aG

Gain for low pass filter of accelerometer.

Warning

Sum of compFilterData::gG and compFilterData::aG must be 1

6.1.2.2 dt

Sampling rate in seconds.

Warning

This must be as precise as possible

6.1.2.3 gG

Gain for high pass filter of gyroscope.

Warning

Sum of compFilterData::gG and compFilterData::aG must be 1

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

- RoboWay/complementaryFilter/complementaryFilter.h

6.2 pidData Struct Reference

Data structure for PID functions.

`#include <pid.h>`

Data Fields

- double k
  Over all gain.
- double kp
  Gain proportional part.
- double ki
  Gain integral part.

- double kd

  Gain differential part.
- double minOut

  Minimum output value.
- double maxOut

  Maximum output value.
- double eOld

  Error of previous call of double updatePid(double rPos, double cPos, pidData ∗pd)
- double iTerm

  Integral term.

6.2.1 Detailed Description

Data structure for PID functions.

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

- RoboWay/pid/pid.h

Chapter 7

File Documentation

7.1 RoboWay/adxl354/adxl354.c File Reference

This is the source code file for simple use of the digital accelerometer ADXL354.

```
#include "./adxl354.h"
```

Functions

- uint8_t adxl354Init (uint8_t addr, uint8_t bwRate)

  Initial setup of an ADXL354 - need to be called only once.
- uint8_t adxl354SetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Set the value of an register.
- uint8_t adxl354GetAccelAll (uint8_t addr, int16_t *x, int16_t *y, int16_t *z)

  Get current acceleration of all three axis.
- uint8_t adxl354GetAccel (uint8_t addr, int8_t axis, int16_t *val)

  Get current acceleration of one axis.

7.1.1 Detailed Description

This is the source code file for simple use of the digital accelerometer ADXL354.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

31.01.2014

Version

1.4.2

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.1.2 Function Documentation

7.1.2.1 uint8_t adxl354GetAccel (uint8_t *addr*, int8_t *axis*, int16_t * *val*)

Get current acceleration of one axis.

Parameters

addr	Address of the ADXL354
axis	Witch axis (see ADXL354_AXIS)
val	Value of axis

Returns

uint8_t

Return values

true	success
false	failure

7.1.2.2 uint8_t adxl354GetAccelAll (uint8_t *addr*, int16_t * *x*, int16_t * *y*, int16_t * *z*)

Get current acceleration of all three axis.

Parameters

addr	Address of the ADXL354
*x	Value of X-Axis
*y	Value of Y-Axis
*z	Value of Z-Axis

Returns

uint8_t

Return values

true	success
false	failure

7.1.2.3 uint8_t adxl354Init (uint8_t *addr*, uint8_t *bwRate*)

Initial setup of an ADXL354 - need to be called only once.

Parameters

addr	Address of the ADXL354 (refer to ADXL354_STD_DEVID)
bwRate	Bit rate setting (refer to ADXL354_REG_BW_RATE_MODES)

Returns

uint8_t

Return values

true	-> success
false	-> failure

7.1.2.4 uint8_t adxl354SetReg (uint8_t *addr*, uint8_t *reg*, uint8_t *val*)

Set the value of an register.

Parameters

addr	Address of the ADXL354
reg	Address of the register (see ADXL354_REGISTERS)
val	New value for the register

Returns

uint8_t

Return values

true	Success
false	Failure

7.2 RoboWay/adxl354/adxl354.h File Reference

This is the header file for simple use of the digital accelerometer ADXL354.

```
#include <avr/io.h> #include "../i2cmaster/i2cmaster.h"
```

Defines

- #define ADXL354_STD_DEVID 0x53

  Standard address of an ADXL354 on I2C.
- #define ADXL354_REG_DEVID 0x00

  The DEVID holds a fixed device ID (see ADXL354_STD_DEVID)
- #define ADXL354_REG_THRESH_TAP 0x1D

  The THRESH_TAP register holds the threshold value for tap interrupts Data format is unsigned.
- #define ADXL354_REG_OFSX 0x1E

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_OFSY 0x1F

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_OFSZ 0x20

  Offset X-Axis for calibration in twos complement format.
- #define ADXL354_REG_TAP_DUR 0x21

  Maximum time than an event must be above the THRESH_TAP to qualifiy as a tap event (625us/LSB) A value of 0 disables the single / double tap events.
- #define ADXL354_REG_TAP_LATENT 0x22

  Latency from detection of a tap to start the time window (1.25 ms/LSB)
- #define ADXL354_REG_TAP_WINDOW 0x23

  Time window from one to a second valid tap.
- #define ADXL354_REG_THRESH_ACT 0x24

  Threshold value for detecting activity unsigned (62.5mg/LSB)
- #define ADXL354_REG_THRESH_INACT 0x25

  Threshold value for detecting inactivity unsigned (62.5mg/LSB)
- #define ADXL354_REG_TIME_INACT 0x26

  Time value that acceleration must be under the value of ADXL354_REG_THRESH_INACT to generate an inactivity interrupt unsigned (1s/LSB)
- #define ADXL354_REG_ACT_INACT_CTL 0x27

  Control register control activity and inactivity interrupts.
- #define ADXL354_REG_THRESH_FF 0x28

  This register holds the threshold value for free-fall detection unsigned (62.5mg/LSB) a value between 300 and 600mg is recommended.
- #define ADXL354_REG_TIME_FF 0x29

  Minimum time that the value of all axis must be less than ADXL354_REG_THRESH_FF unsigned (5ms/LSB)
- #define ADXL354_REG_TAP_AXES_CTL 0x2A

  Enable/disable tap interrupts.
- #define ADXL354_REG_ACT_TAP_STATUS 0x2B

  Act/Tap status - indicates witch axis was first involved in an tap or activity event READ ONLY.

- #define ADXL354_REG_CTL_BW_RATE 0x2C

  Set bandwidth and output data rate Default: 0x0A -> 100Hz See ADXL354_REG_BW_RATE_MODES.
- #define ADXL354_REG_CTL_POWER 0x2D

  Power saving features control.
- #define ADXL354_REG_INT_EN 0x2E

  Interrupt enable / disable Refer to site 20 / 26.
- #define ADXL354_REG_INT_MAP 0x2F

  Interrupt and pin mappings.
- #define ADXL354_REG_INT_SRC 0x30

  Interrupt sources - READ ONLY.
- #define ADXL354_REG_DATA_FORMAT 0x31

  Controls the presentation of data to register 0x32 through register 0x37.
- #define ADXL354_REG_DATAX_L 0x32

  Data of X-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAX_H 0x33

  Data of X-Axis High Byte (READ ONLY)
- #define ADXL354_REG_DATAY_L 0x34

  Data of Y-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAY_H 0x35

  Data of Y-Axis High Byte (READ ONLY)
- #define ADXL354_REG_DATAZ_L 0x36

  Data of Z-Axis Low Byte (READ ONLY)
- #define ADXL354_REG_DATAZ_H 0x37

  Data of Z-Axis High Byte (READ ONLY)
- #define ADXL354_REG_CTL_FIFO 0x38

  FIFO Modes.
- #define ADXL354_REG_FIFO_STATUS 0x39

  FIFO Status.
- #define ADXL354_REG_BW_RATE_400 0b1100

  Value for ADXL354_REG_BW_RATE in I2C 400 kHz mode.
- #define ADXL354_REG_BW_RATE_100 0b1010

  Value for ADXL354_REG_BW_RATE in I2C 100 kHz mode.
- #define ADXL354_AXIS_X ADXL354_REG_DATAX_L

  Start register of X-Axis.
- #define ADXL354_AXIS_Y ADXL354_REG_DATAY_L

  Start register of Y-Axis.
- #define ADXL354_AXIS_Z ADXL354_REG_DATAZ_L

  Start register of Z-Axis.
- #define ADXL354_GET_ACCEL_X(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_X, val)

  Get the current acceleration of X-Axis.
- #define ADXL354_GET_ACCEL_Y(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_Y, val)

Get the current acceleration of Y-Axis.

- #define ADXL354_GET_ACCEL_Z(addr, val) adxl354GetAccel(addr, ADXL354_AXIS_Z, val)

  Get the current acceleration of Z-Axis.

Functions

- uint8_t adxl354Init (uint8_t addr, uint8_t bwRate)

  Initial setup of an ADXL354 - need to be called only once.
- uint8_t adxl354GetAccel (uint8_t addr, int8_t axis, int16_t *val)

  Get current acceleration of one axis.
- uint8_t adxl354GetAccelAll (uint8_t addr, int16_t *x, int16_t *y, int16_t *z)

  Get current acceleration of all three axis.
- uint8_t adxl354SetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Set the value of an register.

7.2.1 Detailed Description

This is the header file for simple use of the digital accelerometer ADXL354.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

31.01.2014

Version

1.4.2

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.2.2 Function Documentation

7.2.2.1 uint8_t adxl354GetAccel (uint8_t *addr*, int8_t *axis*, int16_t * *val*)

Get current acceleration of one axis.

Parameters

addr	Address of the ADXL354
axis	Witch axis (see ADXL354_AXIS)
val	Value of axis

Returns

uint8_t

Return values

true	success
false	failure

7.2.2.2 uint8_t adxl354GetAccelAll (uint8_t *addr*, int16_t * *x*, int16_t * *y*, int16_t * *z*)

Get current acceleration of all three axis.

Parameters

addr	Address of the ADXL354
*x	Value of X-Axis
*y	Value of Y-Axis
*z	Value of Z-Axis

Returns

uint8_t

Return values

true	success
false	failure

7.2.2.3 uint8_t adxl354Init (uint8_t *addr*, uint8_t *bwRate*)

Initial setup of an ADXL354 - need to be called only once.

Parameters

addr	Address of the ADXL354 (refer to ADXL354_STD_DEVID)
bwRate	Bit rate setting (refer to ADXL354_REG_BW_RATE_MODES)

Returns

uint8_t

Return values

true	-> success
false	-> failure

7.2.2.4 uint8_t adxl354SetReg (uint8_t *addr*, uint8_t *reg*, uint8_t *val*)

Set the value of an register.

Parameters

addr	Address of the ADXL354
reg	Address of the register (see ADXL354_REGISTERS)
val	New value for the register

Returns

uint8_t

Return values

true	Success
false	Failure

7.3 RoboWay/avrConstants.c File Reference

Global constants and settings for AtMega 8 bit micro controllers (source code)

```
#include "avrConstants.h"
```

Functions

- double constrainD (double val, double min, double max)

  Restricts a number to a certain interval.

7.3.1 Detailed Description

Global constants and settings for AtMega 8 bit micro controllers (source code)

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

11.02.2014

Version

2.1.5

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.3.2 Function Documentation

7.3.2.1 double constrainD (double *val,* double *min,* double *max*)

Restricts a number to a certain interval.

Parameters

val	Limiting number
min	lower limit
max	higher limit

Returns

double

Return values

val	if min < val < max
min	if val < min
max	if max < val

7.4 RoboWay/avrConstants.h File Reference

Global constants and settings for AtMega 8 bit micro controllers.

Defines

- #define F_CPU 16000000UL

  Clockrate of the CPU.
- #define false 0

  Boolean false.
- #define FALSE false
- #define true 1

  Boolean true.
- #define TRUE true
- #define bool uint8_t

  Data type for boolean variables.
- #define BOOL bool
- #define high 1

  Signal high (a.e. for an IO-Pin)
- #define HIGH high
- #define low 0

  Signal low (a.e. for an IO-Pin)
- #define LOW low
- #define RAD_TO_DEG(x) ((x) * 57.295779513082320876798154814105)

  Converts an angle from rad to deg.
- #define LOW_BYTE(x) ((uint8_t)((x) & 0xFF))

  Returns low byte as uint8_t
  x should be 2 bytes.
- #define HIGH_BYTE(x) ((uint8_t)(((x) >> 8) & 0xFF))

  Returns high byte as uint8_t
  x should be 2 bytes.

Functions

- double constrainD (double val, double min, double max)

  Restricts a number to a certain interval.

7.4.1 Detailed Description

Global constants and settings for AtMega 8 bit micro controllers.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

11.02.2014

Version

2.1.5

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.4.2 Function Documentation

7.4.2.1 double constrainD (double *val*, double *min*, double *max*)

Restricts a number to a certain interval.

Parameters

val	Limiting number
min	lower limit
max	higher limit

Returns

double

Return values

val	if min $<$ val $<$ max
min	if val $<$ min
max	if max $<$ val

7.5 RoboWay/complementaryFilter/complementaryFilter.c File Reference

This is the source code file for complementary filter.

```
#include "complementaryFilter.h"
```

Functions

- void compFilterInit (double gyroGain, double accGain, double dt, compFilterData *fd)

  Initial setup of an complementary filter.

- double compFilterGetAngle (double gyroData, double accData, compFilterData *fd)

  Get new filtered angle.

7.5.1 Detailed Description

This is the source code file for complementary filter.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

18.03.2014

Version

1.3.1

Copyright

7.5.2 Function Documentation

7.5.2.1 double compFilterGetAngle (double *gyroData*, double *accData*, compFilterData * *fd*)

Get new filtered angle.

It depends on of the angular rate of the gyroscope, the old angle and the angle measured by the accelerometer

The angular rate of the gyroscope is filtered with a high pass filer (CF_STD_GYRO_GAIN).

The angle measured by the acceleromter is filtered with a low pass filter (CF_STD_ACC_GAIN)

Parameters

gyroData	Measured angular rate of the gyroscope in degrees
accData	Meassured angle of the accelerometer in degrees
fd	Pointer to the complementary filter working structure

See also

compFilterData

Returns

double Filtered angle

```
      double compFilterGetAngle( double gyroData, double accData,
compFilterData *fd )
      {
        fd->aO = fd->gG * ( fd->aO + gyroData * fd->dt ) + fd->aG *
accData;
        return fd->aO;
      }
```

7.5.2.2 void compFilterInit (double *gyroGain,* double *accGain,* double *dt,* compFilterData * *fd*)

Initial setup of an complementary filter.

Parameters

gyroGain	Gain for high pass filter of gyroscope
accGain	Gain for low pass filter of accelerometer
dt	Sampling rate in seconds

Warning

dt should be as precise as possible

Parameters

fd	Pointer to a complementary filter working structure

Warning

gyroGain + accGain must be 1

See also

compFilterData

Returns

void

7.6 RoboWay/complementaryFilter/complementaryFilter.h File - Reference

This is the header file for complementary filter.

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

Data Structures

- struct compFilterData

  Data structure for complementary filter.

Defines

- #define CF_STD_GYRO_GAIN 0.98

  Standard gain for the high pass filter (gyroscope)
- #define CF_STD_ACC_GAIN 0.02

  Standart gain for the low pass filter (accelerometer)

Functions

- void compFilterInit (double gyroGain, double accGain, double dt, compFilterData *fd)

  Initial setup of an complementary filter.
- double compFilterGetAngle (double gyroData, double accData, compFilterData *fd)

  Get new filtered angle.

7.6.1 Detailed Description

This is the header file for complementary filter. A complementary filter combines the good characteristics of an accelerometer and a gyroscope.

On the other hand it filters out bad properties like drift or high sensitivity.

A `Kalman Filter` is another solution for this problem.

The result is a little bit better, but it is much more complex.

Because of the limitations of ressources of the used controller and the very time critical application i recommend to use a complementary filter.

The sampling rate is about five times higher with a Kalman Filter.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

18.03.2014

Version

1.3.1

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.6.2 Function Documentation

7.6.2.1 double compFilterGetAngle (double *gyroData*, double *accData*, compFilterData * *fd*)

Get new filtered angle.

It depends on of the angular rate of the gyroscope, the old angle and the angle measured by the accelerometer

The angular rate of the gyroscope is filtered with a high pass filer (CF_STD_GYRO_GAIN).

The angle measured by the acceleromter is filtered with a low pass filter (CF_STD_ACC_GAIN)

Parameters

gyroData	Measured angular rate of the gyroscope in degrees
accData	Meassured angle of the accelerometer in degrees
fd	Pointer to the complementary filter working structure

See also

compFilterData

Returns

double Filtered angle

```
      double compFilterGetAngle( double gyroData, double accData,
compFilterData *fd )
      {
        fd->aO = fd->gG * ( fd->aO + gyroData * fd->dt ) + fd->aG *
accData;
        return fd->aO;
      }
```

7.6.2.2 void compFilterInit (double *gyroGain,* double *accGain,* double *dt,* compFilterData * *fd*)

Initial setup of an complementary filter.

Parameters

gyroGain	Gain for high pass filter of gyroscope
accGain	Gain for low pass filter of accelerometer
dt	Sampling rate in seconds

Warning

dt should be as precise as possible

Parameters

fd	Pointer to a complementary filter working structure

Warning

gyroGain + accGain must be 1

See also

compFilterData

Returns

void

7.7 RoboWay/i2cmaster/i2cmaster.c File Reference

Implementation of the I2C (TWI) bus

Basic source code provided by Peter Fleury

Adapted and documented by Andre Sünnemann.

```
#include "i2cmaster.h"
```

Functions

- void i2cInit (unsigned long sclMode)

  Initialize the I2C master interface. Need to be called only once.
- uint8_t i2cStart (uint8_t addr)

  Issues a start condition, sends address and transfer direction.
- void i2cStartWait (uint8_t addr)

  Issues a start condition, sends address and transfer direction If device is busy, use ack polling to wait until device ready.
- void i2cStop (void)

  Terminates the data transfer and releases the I2C bus.
- uint8_t i2cWrite (uint8_t data)

  Send one byte to I2C device.
- uint8_t i2cReadAck (void)

  Read one byte from the I2C device, request more data from device.
- unsigned char i2cReadNak (void)

  Read one byte from the I2C device, read is followed by a stop condition.

7.7.1 Detailed Description

Implementation of the I2C (TWI) bus

Basic source code provided by Peter Fleury

Adapted and documented by Andre Sünnemann.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Version

0.1.0

Date

29.01.2014

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.7.2 Function Documentation

7.7.2.1 void i2cInit (unsigned long *sclMode*)

Initialize the I2C master interface. Need to be called only once.

Parameters

sclMode	Clock rate mode (See I2C_SCL_MODES)

Returns

void

7.7.2.2 uint8_t i2cReadAck (void)

Read one byte from the I2C device, request more data from device.

Returns

uint8_t

Return values

Data	read from I2C device

7.7.2.3 uint8_t i2cReadNak (void)

Read one byte from the I2C device, read is followed by a stop condition.

Returns

uint8_t

Return values

Data	read from I2C device

7.7.2.4 uint8_t i2cStart (uint8_t *addr*)

Issues a start condition, sends address and transfer direction.

Parameters

addr	Address of I2C Slave

Warning

Modify the address with I2C_READ(addr) or I2C_WRITE depending on the direction

Returns

uint8_t

Return values

I2C_FAIL	Failed to access device
I2C_OK	Device accessible

7.7.2.5 void i2cStartWait (uint8_t *addr*)

Issues a start condition, sends address and transfer direction If device is busy, use ack polling to wait until device ready.

Parameters

addr	Address and transfer direction of I2C device

Returns

none

7.7.2.6 void i2cStop (void)

Terminates the data transfer and releases the I2C bus.

Returns

void

7.7.2.7 uint8_t i2cWrite (uint8_t *data*)

Send one byte to I2C device.

Parameters

data	byte to be transfered

Returns

uint8_t

Return values

I2C_FAIL	Write failed
I2C_OK	Write successful

7.8 RoboWay/i2cmaster/i2cmaster.h File Reference

Implementation of the I2C (TWI) bus

Basic source code provided by Peter Fleury

Adapted and documented by Andre Sünnemann.

```
#include <avr/io.h> #include <stdint.h> #include <inttypes.h> #include <compat/twi.h> #include "../avrConstants.h"
```

Defines

- #define I2C_READ(addr) (((addr) << 1) | 0x01)

  Modify the address of the slave for read operations.
- #define I2C_WRITE(addr) (((addr) << 1) | 0x00)

  Modify the address of the slave for write operations.
- #define i2cRepStart(addr) (i2cStart(addr))

  Issues a repeated start condition, sends address and transfer direction This is just an alias of uint8_t i2cStart(uint8_t addr)
- #define I2C_OK TRUE

  Status Code OK.
- #define I2C_FAIL FALSE

  Status Code Failure.
- #define I2C_SCL_NORMAL 10000UL

  Normal Clockrate (100kHz)
- #define I2C_SCL_HIGH 40000UL

  High Clockrate (400kHz)

Functions

- void i2cInit (unsigned long sclMode)

  Initialize the I2C master interface. Need to be called only once.
- void i2cStop (void)

  Terminates the data transfer and releases the I2C bus.
- uint8_t i2cStart (uint8_t addr)

Issues a start condition, sends address and transfer direction.

- void i2cStartWait (uint8_t addr)

  Issues a start condition, sends address and transfer direction If device is busy, use ack polling to wait until device ready.

- uint8_t i2cWrite (uint8_t data)

  Send one byte to I2C device.

- uint8_t i2cReadAck (void)

  Read one byte from the I2C device, request more data from device.

- uint8_t i2cReadNak (void)

  Read one byte from the I2C device, read is followed by a stop condition.

7.8.1 Detailed Description

Implementation of the I2C (TWI) bus

Basic source code provided by Peter Fleury

Adapted and documented by Andre Sünnemann.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Version

0.1.0

Date

29.01.2014

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.8.2 Function Documentation

7.8.2.1 void i2cInit (unsigned long *sclMode*)

Initialize the I2C master interface. Need to be called only once.

Parameters

sclMode	Clock rate mode (See I2C_SCL_MODES)

Returns

void

7.8.2.2 uint8_t i2cReadAck (void)

Read one byte from the I2C device, request more data from device.

Returns

uint8_t

Return values

Data	read from I2C device

7.8.2.3 uint8_t i2cReadNak (void)

Read one byte from the I2C device, read is followed by a stop condition.

Returns

uint8_t

Return values

Data	read from I2C device

7.8.2.4 uint8_t i2cStart (uint8_t *addr*)

Issues a start condition, sends address and transfer direction.

Parameters

addr	Address of I2C Slave

Warning

Modify the address with I2C_READ(addr) or I2C_WRITE depending on the direction

Returns

uint8_t

Return values

I2C_FAIL	Failed to access device
I2C_OK	Device accessible

7.8.2.5 void i2cStartWait (uint8_t *addr*)

Issues a start condition, sends address and transfer direction If device is busy, use ack polling to wait until device ready.

Parameters

addr	Address and transfer direction of I2C device

Returns

none

7.8.2.6 void i2cStop (void)

Terminates the data transfer and releases the I2C bus.

Returns

void

7.8.2.7 uint8_t i2cWrite (uint8_t *data*)

Send one byte to I2C device.

Parameters

data	byte to be transfered

Returns

uint8_t

Return values

I2C_FAIL	Write failed
I2C_OK	Write successful

7.9 RoboWay/l3g4200d/l3g4200d.c File Reference

This is the header file for the l3g4200d gyro bibliothek.

```
#include "l3g4200d.h"
```

Functions

- uint8_t l3g4200dInit (uint8_t addr, uint8_t bwRate)

  Initial setup of an L3G4200D - need to be called only once.
- uint8_t l3g4200dGetReg (uint8_t addr, uint8_t reg)

  Get the value of a specified register.
- uint8_t l3g4200dSetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Set the value of a specified register.
- uint8_t l3g4200dGetAR (uint8_t addr, uint8_t axis, int16_t *val)

  Get angular rate data of one axis.
- uint8_t l3g4200dGetARAll (uint8_t addr, int16_t *x, int16_t *y, int16_t *z)

  Get current angular rate data of all three axis.

7.9.1 Detailed Description

This is the header file for the l3g4200d gyro bibliothek.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

11.02.2014

Version

0.0.1

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.9.2 Function Documentation

7.9.2.1 uint8_t l3g4200dGetAR (uint8_t *addr*, uint8_t *axis*, int16_t * *val*)

Get angular rate data of one axis.

Parameters

addr	Address of the L3G4200D
axis	Witch axis (see L3G4200D_AXIS)
val	Value of axis

Returns

uint8_t

Return values

true	Success
false	Fail

7.9.2.2 uint8_t l3g4200dGetARAll (uint8_t *addr*, int16_t * *x*, int16_t * *y*, int16_t * *z*)

Get current angular rate data of all three axis.

Parameters

addr	Address of the L3G4200D
x	Value of X-Axis
y	Value of Y-Axis
z	Value of Z-Axis

Returns

uint8_t

Return values

true	Success
false	Fail

7.9.2.3 uint8_t l3g4200dGetReg (uint8_t *addr*, uint8_t *reg*)

Get the value of a specified register.

Parameters

addr	Address of the L3G4200D
reg	Address of the register (see L3G4200D_REGISTERS)

Returns

uint8_t

Return values

Value	of the register

7.9.2.4 uint8_t l3g4200dInit (uint8_t *addr*, uint8_t *bwRate*)

Initial setup of an L3G4200D - need to be called only once.

Parameters

addr	Address of the L3G4200D (refer to L3G4200D_DEVICE_IDS)
bwRate	Bit rate setting (refer to L3G4200D_BW_RATE)

Returns

uint8_t

Return values

true	-> success
false	-> fail

7.9.2.5 uint8_t l3g4200dSetReg (uint8_t *addr*, uint8_t *reg*, uint8_t *val*)

Set the value of a specified register.

Parameters

addr	Address of the L3G4200D
reg	Address of the register (see L3G4200D_REGISTERS)
val	Value for the register

Returns

uint8_t

Return values

true	Success
false	Fail

7.10 RoboWay/l3g4200d/l3g4200d.h File Reference

This is the header file for the l3g4200d gyro bibliothek.

```
#include <avr/io.h> #include "../avrConstants.h" #include
"../i2cmaster/i2cmaster.h"
```

Defines

- #define L3G4200D_DEVID_0 0x68

  Possible address 0 of an L3G4200D on I2C.
- #define L3G4200D_DEVID_1 0x69

  Possible address 1 of an L3G4200D on I2C.
- #define L3G4200D_REG_WAI 0x0F

  Device identification register (Who Am I)
- #define L3G4200D_REG_CTRL_1 0x20

  Control register 1
  Refer to table 20 of in data sheet.
- #define L3G4200D_REG_CTRL_2 0x21

  Control register 2
  Refer to table 28 in data sheet.
- #define L3G4200D_REG_CTRL_3 0x22

  Control register 3
  Refer to table 28 in data sheet.
- #define L3G4200D_REG_CTRL_4 0x23

  Control register 4
  Refer to table 30 in data sheet.
- #define L3G4200D_REG_CTRL_5 0x24

  Control register 5
  Refer to table 33 in data sheet.
- #define L3G4200D_REG_REF 0x25

  Reference value for interrupt generation
  Refer to table 37 / 38 in data sheet.
- #define L3G4200D_REG_TEMP 0x26

  Temperature data
  Refer to table 39 / 40 in data sheet.
- #define L3G4200D_REG_STATUS 0x27

  Status register
  Refer to table 41 / 42 in data sheet.
- #define L3G4200D_REG_DATAX_L 0x28

X-Axis angular rate data - Low Byte.

- #define L3G4200D_REG_DATAX_H 0x29

  X-Axis angular rate data - High Byte.
- #define L3G4200D_REG_DATAY_L 0x2A

  Y-Axis angular rate data - Low Byte.
- #define L3G4200D_REG_DATAY_H 0x2B

  Y-Axis angular rate data - High Byte.
- #define L3G4200D_REG_DATAZ_L 0x2C

  Z-Axis angular rate data - Low Byte.
- #define L3G4200D_REG_DATAZ_H 0x2D

  Z-Axis angular rate data - High Byte.
- #define L3G4200D_REG_CTRL_FIFO 0x2E

  Register for FIFO configuration
  Refer to table 43 / 44 / 45.
- #define L3G4200D_REG_SRC_FIFO 0x2F

  Source configuration for FIFO
  Refer to table 46 / 47.
- #define L3G4200D_REG_CFG_INT1 0x30

  Configuration register for interrupt 1
  Refer to table 48 / 49.
- #define L3G4200D_REG_SRC_INT1 0x31

  Configuration register for interrupt source
  Refer to table 50 / 51.
- #define L3G4200D_REG_INT1_TSH_XH 0x32

  Interrupt threshold X-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_XL 0x33

  Interrupt threshold X-Axis - Low Byte.
- #define L3G4200D_REG_INT1_TSH_YH 0x34

  Interrupt threshold Y-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_YL 0x35

  Interrupt threshold Y-Axis - Low Byte.
- #define L3G4200D_REG_INT1_TSH_ZH 0x36

  Interrupt threshold Z-Axis - High Byte.
- #define L3G4200D_REG_INT1_TSH_ZL 0x37

  Interrupt threshold Z-Axis - Low Byte.
- #define L3G4200D_REG_INT1_DUR 0x38

  Minimum, wait and max duration of an interrupt
  Refer to table 64 / 65 and figure 20 / 21 in data sheet.
- #define L3G4200D_BW_RATE_100 0b00110000

  Bit and Data Rate for 100 kHz I2C.
- #define L3G4200D_BW_RATE_400 0b10000000

  Bit and Data Rate for 400 kHz I2C.
- #define l3g4200dGetTemp(addr) l3g4200dGetReg(addr, L3G4200D_REG_TEMP)

Get measured temperature.

- #define L3G4200D_AXIS_X L3G4200D_REG_DATAX_L

  Start register of X-Axis.
- #define L3G4200D_AXIS_Y L3G4200D_REG_DATAY_L

  Start register of Y-Axis.
- #define L3G4200D_AXIS_Z L3G4200D_REG_DATAZ_L

  Start register of Z-Axis.
- #define L3G4200D_GET_AR_X(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAX_L, val)

  Get current angular rate data of X-Axis.
- #define L3G4200D_GET_AR_Y(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAY_L, val)

  Get current angular rate data of Y-Axis.
- #define L3G4200D_GET_AR_Z(addr, val) l3g4200dGetAR(addr, L3G4200D_REG_DATAZ_L, val)

  Get current angular rate data of Z-Axis.

Functions

- uint8_t l3g4200dInit (uint8_t addr, uint8_t bwRate)

  Initial setup of an L3G4200D - need to be called only once.
- uint8_t l3g4200dGetReg (uint8_t addr, uint8_t reg)

  Get the value of a specified register.
- uint8_t l3g4200dSetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Set the value of a specified register.
- uint8_t l3g4200dGetAR (uint8_t addr, uint8_t axis, int16_t *val)

  Get angular rate data of one axis.
- uint8_t l3g4200dGetARAll (uint8_t addr, int16_t *x, int16_t *y, int16_t *z)

  Get current angular rate data of all three axis.

7.10.1 Detailed Description

This is the header file for the l3g4200d gyro bibliothek.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

11.02.2014

Version

0.0.1

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.10.2 Function Documentation

7.10.2.1 uint8_t l3g4200dGetAR (uint8_t *addr*, uint8_t *axis*, int16_t * *val*)

Get angular rate data of one axis.

Parameters

addr	Address of the L3G4200D
axis	Witch axis (see L3G4200D_AXIS)
val	Value of axis

Returns

uint8_t

Return values

true	Success
false	Fail

7.10.2.2 uint8_t l3g4200dGetARAll (uint8_t *addr*, int16_t * *x*, int16_t * *y*, int16_t * *z*)

Get current angular rate data of all three axis.

Parameters

addr	Address of the L3G4200D
x	Value of X-Axis
y	Value of Y-Axis
z	Value of Z-Axis

Returns

uint8_t

Return values

true	Success
false	Fail

7.10.2.3 uint8_t l3g4200dGetReg (uint8_t *addr*, uint8_t *reg*)

Get the value of a specified register.

Parameters

addr	Address of the L3G4200D
reg	Address of the register (see L3G4200D_REGISTERS)

Returns

uint8_t

Return values

Value	of the register

7.10.2.4 uint8_t l3g4200dInit (uint8_t *addr*, uint8_t *bwRate*)

Initial setup of an L3G4200D - need to be called only once.

Parameters

addr	Address of the L3G4200D (refer to L3G4200D_DEVICE_IDS)
bwRate	Bit rate setting (refer to L3G4200D_BW_RATE)

Returns

uint8_t

Return values

true	-> success
false	-> fail

7.10.2.5 uint8_t l3g4200dSetReg (uint8_t *addr*, uint8_t *reg*, uint8_t *val*)

Set the value of a specified register.

Parameters

addr	Address of the L3G4200D
reg	Address of the register (see L3G4200D_REGISTERS)
val	Value for the register

Returns

uint8_t

Return values

true	Success
false	Fail

7.11 RoboWay/main.c File Reference

Main program file for RoboWay.

`#include <avr/io.h> #include "./RoboWay/roboWay.h"`

Functions

- int main (void)

  Main function for RoboWay.

7.11.1 Detailed Description

Main program file for RoboWay.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

11.02.2014

Version

0.0.1

See also

roboWay.h
roboWay.c

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.12 RoboWay/mainPage.h File Reference

This file is only for documenting purposes.

7.12.1 Detailed Description

This file is only for documenting purposes.

Warning

Insert no program code here

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

20.03.2014

Version

1.3.5

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.13 RoboWay/motorControll/motorControll.c File Reference

Sourcecode for motorcontrol of the RoboWay.

```
#include "motorControll.h"
```

Functions

- void mCInit (void)

  Initialize motor control.
- void mCSetSpeedLeft (uint8_t val)

  Set speed of the left wheel.
- void mCSetSpeedRight (uint8_t val)

  Set speed of the right wheel.
- void mCRightFW (void)

  Set direction of right wheel to forward.
- void mCRightBW (void)

  Set direction of right wheel to backward.
- void mCLeftFW (void)

  Set direction of left wheel to forward.
- void mCLeftBW (void)

  Set direction of right wheel to backward.

7.13.1 Detailed Description

Sourcecode for motorcontrol of the RoboWay.

Version

0.0.1

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

27.01.2014

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.13.2 Function Documentation

7.13.2.1 void mCSetSpeedLeft (uint8_t *val*)

Set speed of the left wheel.

Parameters

val	0 => 0% PWM => 0% Voltage 255 => 100% PWM => 100% Voltage

7.13.2.2 void mCSetSpeedRight (uint8_t *val*)

Set speed of the right wheel.

Parameters

val	0 => 0% PWM => 0% Voltage 255 => 100% PWM => 100% Voltage

7.14 RoboWay/motorControll/motorControll.h File Reference

Header file for motor control of RoboWay.

```
#include <avr/io.h>
```

Defines

- #define MC_PORT PORTD

  AVR PORT used for motor control pins.
- #define MC_DDR DDRD

  AVR DDR used for motor control pins.
- #define MC_LEFT PD5

  Pin for speed left wheel.
- #define MC_LEFT_FW PD2

  Left wheel drives in forward direction if this pin is set to high.
- #define MC_LEFT_BW PD3

  Left wheel drives in backward direction if this pin is set to high.
- #define MC_RIGHT PD6

  Pin for speed right wheel.
- #define MC_RIGHT_FW PD4

  right wheel drives in forward direction if this pin is set to high
- #define MC_RIGHT_BW PD7

  Right wheel drives in forward direction if this pin is set to high.

- #define MC_LEFT_OCR OCR0B

  Output compare register for left wheel.
- #define MC_RIGHT_OCR OCR0A

  Output compare register for right wheel.

Functions

- void mCInit (void)

  Initialize motor control.
- void mCSetSpeedLeft (uint8_t val)

  Set speed of the left wheel.
- void mCSetSpeedRight (uint8_t val)

  Set speed of the right wheel.
- void mCRightFW (void)

  Set direction of right wheel to forward.
- void mCRightBW (void)

  Set direction of right wheel to backward.
- void mCLeftFW (void)

  Set direction of left wheel to forward.
- void mCLeftBW (void)

  Set direction of right wheel to backward.

7.14.1 Detailed Description

Header file for motor control of RoboWay. This implementation of a motor control with PWM is very efficient, because it uses the output compare match interrupt to generate the PWM output. Only when you configure the motor control it needs cpu time.

Version

1.2.5

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

27.01.2014

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.14.2 Function Documentation

7.14.2.1 void mCSetSpeedLeft (uint8_t *val*)

Set speed of the left wheel.

Parameters

val	0 => 0% PWM => 0% Voltage 255 => 100% PWM => 100% Voltage

7.14.2.2 void mCSetSpeedRight (uint8_t *val*)

Set speed of the right wheel.

Parameters

val	0 => 0% PWM => 0% Voltage 255 => 100% PWM => 100% Voltage

7.15 RoboWay/pid/pid.c File Reference

This is the source code file of the pid controller.

```
#include "pid.h"
```

Functions

- void pidInit (double k, double kp, double ki, double kd, double minOut, double maxOut, pidData *pd)

  Initial setup of a pid controller.
- double updatePid (double rPos, double cPos, pidData *pd)

  Calculate new output depending on reference position, current position and the data in the working structure.

7.15.1 Detailed Description

This is the source code file of the pid controller.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

18.03.2014

Version

1.6.2

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.15.2 Function Documentation

7.15.2.1 void pidInit (double *k*, double *kp*, double *ki*, double *kd*, double *minOut*, double *maxOut*, pidData * *pd*)

Initial setup of a pid controller.

Parameters

k	Over all gain
kp	Gain proportional part
ki	Gain integral part
kd	Gain differential part
minOut	Minimum output value
maxOut	Maximum output value
pd	Pointer to a pid working structure

Returns

void

7.15.2.2 double updatePid (double *rPos*, double *cPos*, pidData * *pd*)

Calculate new output depending on reference position, current position and the data in the working structure.

Parameters

rPos	Reference position
cPos	Current position
pd	Pointer to a pid working structure

Returns

double New value for motor speed

See also

motorControll.h

Return values

pidData::minOut	<= x <= pidData::maxOut

```
double updatePid( double rPos, double cPos, pidData *pd
 )
{
        double error = 0.0;
        double pTerm = 0.0;
        double dTerm = 0.0;

        error = rPos - cPos;
        pTerm = pd->kp * error;
        pd->iTerm = constrainD( pd->iTerm + pd
->ki * error, PID_MIN_ITERM, PID_MAX_ITERM );
        dTerm = pd->kd * ( error - pd->eOld );
        return constrainD( pTerm + pd->iTerm +
dTerm, pd->minOut, pd->maxOut );
}
```

7.16 RoboWay/pid/pid.h File Reference

This is the header file for pid closed loop controller for RoboWay.

`#include <stdint.h> #include "../avrConstants.h"`

Data Structures

- struct pidData

  Data structure for PID functions.

Defines

- #define PID_MAX_ITERM 200.0

  Maximum value for pidData::iTerm.
- #define PID_MIN_ITERM -200.0

  Minimum value for pidData::iTerm.

Functions

- void pidInit (double k, double kp, double ki, double kd, double minOut, double maxOut, pidData *pd)

  Initial setup of a pid controller.
- double updatePid (double rPos, double cPos, pidData *pd)

  Calculate new output depending on reference position, current position and the data in the working structure.

7.16.1 Detailed Description

This is the header file for pid closed loop controller for RoboWay.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

18.03.2014

Version

1.6.2

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.16.2 Function Documentation

7.16.2.1 void pidInit (double *k*, double *kp*, double *ki*, double *kd*, double *minOut*, double *maxOut*, pidData * *pd*)

Initial setup of a pid controller.

Parameters

k	Over all gain
kp	Gain proportional part
ki	Gain integral part
kd	Gain differential part
minOut	Minimum output value
maxOut	Maximum output value
pd	Pointer to a pid working structure

Returns

void

7.16.2.2 double updatePid (double *rPos*, double *cPos*, pidData * *pd*)

Calculate new output depending on reference position, current position and the data in the working structure.

Parameters

rPos	Reference position
cPos	Current position
pd	Pointer to a pid working structure

Returns

double New value for motor speed

See also

motorControll.h

Return values

pidData::minOut	<= x <= pidData::maxOut

```
                    double updatePid( double rPos, double cPos, pidData *pd
 )
                    {
                                    double error = 0.0;
                                    double pTerm = 0.0;
                                    double dTerm = 0.0;

                                    error = rPos - cPos;
                                    pTerm = pd->kp * error;
                                    pd->iTerm = constrainD( pd->iTerm + pd
->ki * error, PID_MIN_ITERM, PID_MAX_ITERM );
                                    dTerm = pd->kd * ( error - pd->eOld );
                                    return constrainD( pTerm + pd->iTerm +
dTerm, pd->minOut, pd->maxOut );
                    }
```

7.17 RoboWay/roboWay/roboWay.c File Reference

This is the main source code file for RoboWay - all crazy stuff is placed here.

```
#include "roboWay.h"
```

Functions

- ISR (TIMER2_OVF_vect)

  Interrupt service routine for sampling rate of the motor controller.
- void roboWayInit ()

  Initial setup of RoboWay - need to be called only once.
- void roboWayLoop (void)

  Main function of RoboWay - need to be called only once There is a never ending while loop inside.

Variables

- pidData pidDataC

  Working data structure for pid motor control.
- compFilterData compFilterDataC

  Working data structure for complementary filter motor control.
- volatile bool pidMCEn = true

  Global variable for timing of motor controll.

7.17.1 Detailed Description

This is the main source code file for RoboWay - all crazy stuff is placed here.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

3.02.2014

Version

2.6.5

Copyright

7.17.2 Variable Documentation

7.17.2.1 pidMCEn = true

Global variable for timing of motor controll.

It is used in ISR(TIMER2_OVF_vect) and void roboWayLoop(void)

7.18 RoboWay/roboWay/roboWay.h File Reference

This is the main header file for RoboWay.

```
#include <avr/interrupt.h> #include "../adxl354/adxl354.-
h"  #include "../l3g4200d/l3g4200d.h"  #include "../motor-
Controll/motorControll.h"      #include "../avrConstants.-
h"  #include "../srf08/srf08.h"  #include "../pid/pid.h" ×
#include "../complementaryFilter/complementaryFilter.h"
#include <math.h> #include <util/delay.h>
```

Defines

- #define SRFF SRF08_ADDR_00

  Address of the front srf.
- #define SRFB SRF08_ADDR_01

  Address of the back srf.
- #define ACC_ADDR ADXL354_STD_DEVID

  Address of the Accelerometer.
- #define GYRO_ADDR L3G4200D_DEVID_1

  Address of the gyroscope.
- #define LED_FORWARD PB5

  LED for forward direction.
- #define LED_BACKWARD PB4

  LED for backward direction.
- #define PID_INTERVAL 310

  Sampling rate for PID
  (int)((interval in sec ∗ F_FPU) / 255)
- #define PID_INTERVAL_SEC 0.004940625

  Seconds between two PID stabilissation cicles
  (PID_INTERVAL ∗ 255) / F_CPU.
- #define BALANCE_REF 92.75

  Reference value for roll (balancing)
- #define ROLL_MIN_LIMIT 55.0

  Minimum value for roll.
- #define ROLL_MAX_LIMIT 135.0

  Maximum value for roll.

- #define K_B_C 1.0

  Over all gain.
- #define K_P_B_C 1.86

  Gain proportional part.
- #define K_I_B_C 0.030

  Gain integral part.
- #define K_D_B_C 0.006

  Gain differential part.
- #define MIN_OUT_B_C -255.0

  Minimal value for pid output.
- #define MAX_OUT_B_C 255.0

  Maximum value for pid output.

Functions

- void roboWayInit (void)

  Initial setup of RoboWay - need to be called only once.
- void roboWayLoop (void)

  Main function of RoboWay - need to be called only once There is a never ending while loop inside.

7.18.1 Detailed Description

This is the main header file for RoboWay.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

3.02.2014

Version

2.6.5

Copyright

7.18.2 Define Documentation

7.18.2.1 #define PID_INTERVAL 310

Sampling rate for PID

(int)((interval in sec * F_FPU) / 255)

Measured minimum interval is 305 -> PID_INTERVAL = 310

7.19 RoboWay/srf08/srf08.c File Reference

This is the main source code file for RoboWay - all crazy stuff is placed here.

```
#include "srf08.h"
```

Functions

- void srf08Start (uint8_t addr, uint8_t command)

  Start a measure of distance and light.
- uint8_t srf08Status (uint8_t addr)

  Get status of the srf08.
- uint16_t srf08GetDistance (uint8_t addr)

  Get measured distance.
- uint8_t srf08GetLight (uint8_t addr)

  Get measured light.
- uint8_t srf08SetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Change the value of an specified register.
- uint8_t srf08SetAddr (uint8_t oldAddr, uint8_t newAddr)

  Set a new address for an SRF08.

7.19.1 Detailed Description

This is the main source code file for RoboWay - all crazy stuff is placed here.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

3.02.2014

Version

1.8.4

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.19.2 Function Documentation

7.19.2.1 uint16_t srf08GetDistance (uint8_t *addr*)

Get measured distance.

Parameters

addr	Address of the srf08

Returns

uint16_t

Return values

Distance	

7.19.2.2 uint8_t srf08GetLight (uint8_t *addr*)

Get measured light.

Parameters

addr	Address of the srf08

Returns

uint8_t

Return values

Measured	Light

7.19.2.3 uint8_t srf08SetAddr (uint8_t *oldAddr,* uint8_t *newAddr*)

Set a new address for an SRF08.

Parameters

oldAddr	Old address of the SRF08
newAddr	New address of the SRF08 (see SRF08_ADDRESSES)

Returns

uint8_t

Return values

SRF08_AVAIL	Success
SRF08_UNAVAIL	Error

7.19.2.4 uint8_t srf08SetReg (uint8_t *addr,* uint8_t *reg,* uint8_t *val*)

Change the value of an specified register.

Parameters

addr	Address of the srf08
reg	Address of register to change (see SRF08_REGISTERS)
val	New value for register given by param reg

Returns

uint8_t

Return values

SRF08_AVAIL	Success
SRF08_UNAVAIL	Error

7.19.2.5 void srf08Start (uint8_t *addr,* uint8_t *command*)

Start a measure of distance and light.

Parameters

addr	I2C address of the SRF08
command	Modes for measure (see SRF08_COMMANDS)

Returns

none

7.19.2.6 uint8_t srf08Status (uint8_t *addr*)

Get status of the srf08.

Parameters

addr	Address of the srf08 to check

Returns

uint8_t

Return values

!=	SRF08_UNAVAIL SRF08 is available
SRF08_UNAVAIL	SRF08 is unavailable or measurement is pending

7.20 RoboWay/srf08/srf08.h File Reference

This is the header file for the srf08 bibliothek.

```
#include "../i2cmaster/i2cmaster.h"      #include "../avr-
Constants.h" #include <util/delay.h>
```

Defines

- #define SRF08_ADDR_00 0xE0

  Possible address 0.
- #define SRF08_ADDR_01 0xE2

  Possible address 1.
- #define SRF08_ADDR_02 0xE4

  Possible address 2.
- #define SRF08_ADDR_03 0xE6

  Possible address 3.
- #define SRF08_ADDR_04 0xE8

  Possible address 4.
- #define SRF08_ADDR_05 0xEA

  Possible address 5.
- #define SRF08_ADDR_06 0xEC

  Possible address 6.
- #define SRF08_ADDR_07 0xEE

Possible address 7.

- #define SRF08_ADDR_08 0xF0

  Possible address 8.
- #define SRF08_ADDR_09 0xF2

  Possible address 9.
- #define SRF08_ADDR_10 0xF4

  Possible address 10.
- #define SRF08_ADDR_11 0xF6

  Possible address 11.
- #define SRF08_ADDR_12 0xF8

  Possible address 12.
- #define SRF08_ADDR_13 0xFA

  Possible address 13.
- #define SRF08_ADDR_14 0xFC

  Possible address 14.
- #define SRF08_ADDR_15 0xFE

  Possible address 15.
- #define SRF08_ADDR_READ(x) ((x) + 0x01)

  Modify the address for read operation.
- #define SRF08_ADDR_WRITE(x) ((x))

  Modify the address for write operation.
- #define SRF08_REG_VERSION 0x00

  Version of the SRF08 is stored here.
- #define SRF08_REG_COMMAND SRF08_REG_VERSION

  Register for commands - See SRF08_COMMANDS for possible values.
- #define SRF08_REG_LIGHT 0x01

  Value of measured light is stored here.
- #define SRF08_REG_GAIN SRF08_REG_LIGHT

  Register for setting the gain.
- #define SRF08_REG_ECHO_H 0x02

  Value of measured distance - high byte.
- #define SRF08_REG_RANGE SRF08_REG_ECHO_H

  Range in mm (val ∗ 43mm) + 43mm.
- #define SRF08_REG_ECHO_L 0x03

  Value of measured distance - low byte.
- #define SRF08_COM_INCH 0x50

  Start measure in inch.
- #define SRF08_COM_CM 0x51

  Start measure in cm.
- #define SRF08_COM_MKS 0x52

  Start measure in microseconds.
- #define SRF08_COM_ADDR0 0xA0

  Sequence one for changing the I2C-Address.

- #define SRF08_COM_ADDR1 0xAA

  Sequence two for changing the I2C-Address.
- #define SRF08_COM_ADDR2 0xA5

  Sequence three for changing the I2C-Address.
- #define SRF08_AVAIL 0x00

  SRF08 is not available.
- #define SRF08_UNAVAIL 0xFF

  SRF08 is not available or measure is pending.
- #define srf08StartInch(addr) srf08Start(addr, SRF08_COM_INCH)

  Start measure in inch.
- #define srf08StartCM(addr) srf08Start(addr, SRF08_COM_CM)

  Start measure in cm.
- #define srf08StartMcs(addr) srf08Start(addr, SRF08_COM_MKS)

  Start measure in microseconds.
- #define srf08SetRange(addr, range) srf08SetReg(addr, SRF08_REG_RANGE, ((range) - 43) / 43)

  Set range in mm.

Functions

- void srf08Start (uint8_t addr, uint8_t command)

  Start a measure of distance and light.
- uint8_t srf08Status (uint8_t addr)

  Get status of the srf08.
- uint16_t srf08GetDistance (uint8_t addr)

  Get measured distance.
- uint8_t srf08GetLight (uint8_t addr)

  Get measured light.
- uint8_t srf08SetReg (uint8_t addr, uint8_t reg, uint8_t val)

  Change the value of an specified register.
- uint8_t srf08SetAddr (uint8_t oldAddr, uint8_t newAddr)

  Set a new address for an SRF08.

7.20.1 Detailed Description

This is the header file for the srf08 bibliothek.

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

04.02.2014

Version

1.8.4

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.20.2 Function Documentation

7.20.2.1 uint16_t srf08GetDistance (uint8_t *addr*)

Get measured distance.

Parameters

addr	Address of the srf08

Returns

uint16_t

Return values

Distance	

7.20.2.2 uint8_t srf08GetLight (uint8_t *addr*)

Get measured light.

Parameters

addr	Address of the srf08

Returns

uint8_t

Return values

Measured	Light

7.20.2.3 uint8_t srf08SetAddr (uint8_t *oldAddr,* uint8_t *newAddr*)

Set a new address for an SRF08.

Parameters

oldAddr	Old address of the SRF08
newAddr	New address of the SRF08 (see SRF08_ADDRESSES)

Returns

uint8_t

Return values

SRF08_AVAIL	Success
SRF08_UNAVAIL	Error

7.20.2.4 uint8_t srf08SetReg (uint8_t *addr,* uint8_t *reg,* uint8_t *val*)

Change the value of an specified register.

Parameters

addr	Address of the srf08
reg	Address of register to change (see SRF08_REGISTERS)
val	New value for register given by param reg

Returns

uint8_t

Return values

SRF08_AVAIL	Success
SRF08_UNAVAIL	Error

7.20.2.5 void srf08Start (uint8_t *addr,* uint8_t *command*)

Start a measure of distance and light.

Parameters

addr	I2C address of the SRF08
command	Modes for measure (see SRF08_COMMANDS)

Returns

none

7.20.2.6 uint8_t srf08Status (uint8_t *addr*)

Get status of the srf08.

Parameters

addr	Address of the srf08 to check

Returns

uint8_t

Return values

!=	SRF08_UNAVAIL SRF08 is available
SRF08_UNAVAIL	SRF08 is unavailable or measurement is pending

7.21 RoboWay/usart/usart.c File Reference

Source code file for USART (RS232)

```
#include "usart.h"
```

Functions

- void setUsartBaudRate (uint32_t baudRate)

  Setup baudrate for usart.
- void setupUsart0 (uint32_t baudRate)

  Initial setup of the usart serial interface.
- void usart0Transmit (uint8_t data)

  Send one byte over the usart.
- void usart0PrintStr (char str[])

  Send a zero terminated string over the usart.

7.21.1 Detailed Description

Source code file for USART (RS232)

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

12.08.2013

Version

0.0.1

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.21.2 Function Documentation

7.21.2.1 void setupUsart0 (uint32_t *baudRate*)

Initial setup of the usart serial interface.

Parameters

baudRate	Baudrate in bit/sec Possible values: 2400; 4800; 9600; 19200; 57600; 115200

Returns

void

7.21.2.2 void setUsartBaudRate (uint32_t *baudRate*)

Setup baudrate for usart.

Parameters

baudRate	Baudrate in bit/sec Possible values: 2400; 4800; 9600; 19200; 57600; 115200

Returns

void

7.21.2.3 void usart0PrintStr (char *str[]*)

Send a zero terminated string over the usart.

Parameters

str	String to send

Warning

str must be a zero terminated string

Returns

void

7.21.2.4 void usart0Transmit (uint8_t *data*)

Send one byte over the usart.

Parameters

data	Byte to send

7.22 RoboWay/usart/usart.h File Reference

Header file for USART (RS232)

```
#include <avr/io.h>  #include <avr/interrupt.h>  #include
"../avrConstants.h"
```

Defines

- #define USART_BAUD_CALC(x) ((F_CPU / (16UL * (x))) - 1)

  Calculate value for the prescaler of the usart depending on F_CPU and baud rate.
- #define USART_DOUBLE_BAUD_CALC(x) ((F_CPU / (8UL * (x))) - 1)

  Calculate value for the prescaler of the usart
  It is used if the baud rate is out of range.

Functions

- void setUsartBaudRate (uint32_t baudRate)

  Setup baudrate for usart.
- void setupUsart0 (uint32_t baudRate)

  Initial setup of the usart serial interface.
- void usart0Transmit (uint8_t data)

  Send one byte over the usart.
- void usart0PrintStr (char str[])

  Send a zero terminated string over the usart.

7.22.1 Detailed Description

Header file for USART (RS232)

Author

Andre Sünnemann (a.suennemann@edv-peuker.de | www.edv-peuker.-de)

Date

12.08.2013

Version

0.0.1

Copyright

"THE BEER-WARE LICENSE" (Revision 42):
<a.suennemann@edv-peuker.de> wrote this file.
As long as you retain this notice you can do whatever you want with this stuff.
If we meet some day, and you think this stuff is worth it, you can buy me a beer in return Andre Sünnemann.

7.22.2 Function Documentation

7.22.2.1 void setupUsart0 (uint32_t *baudRate*)

Initial setup of the usart serial interface.

Parameters

baudRate	Baudrate in bit/sec Possible values: 2400; 4800; 9600; 19200; 57600; 115200

Returns

void

7.22.2.2 void setUsartBaudRate (uint32_t *baudRate*)

Setup baudrate for usart.

Parameters

baudRate	Baudrate in bit/sec Possible values: 2400; 4800; 9600; 19200; 57600; 115200

Returns

void

7.22.2.3 void usart0PrintStr (char *str[]*)

Send a zero terminated string over the usart.

Parameters

str	String to send

Warning

str must be a zero terminated string

Returns

void

7.22.2.4 void usart0Transmit (uint8_t *data*)

Send one byte over the usart.

Parameters

data	Byte to send