Parking assistant

Generated by Doxygen 1.8.20

1 Main Page	1
2 Module Index	3
2.1 Modules	3
3 File Index	5
3.1 File List	5
4 Module Documentation	7
4.1 GPIO <gpio.h></gpio.h>	7
4.1.1 Detailed Description	7
4.1.2 Function Documentation	8
4.1.2.1 GPIO_config_input_nopull()	8
4.1.2.2 GPIO_config_input_pullup()	8
4.1.2.3 GPIO_config_output()	8
4.1.2.4 GPIO_read()	9
4.1.2.5 GPIO_toggle()	9
4.1.2.6 GPIO_write_high()	9
4.1.2.7 GPIO_write_low()	0
4.2 LCD library <lcd.h></lcd.h>	1
4.2.1 Detailed Description	3
4.2.2 Macro Definition Documentation	4
4.2.2.1 LCD_CONTROLLER_KS0073	4
4.2.2.2 LCD_DATA0_PORT	4
4.2.2.3 LCD_DATA1_PORT	4
4.2.2.4 LCD_DATA2_PORT	4
4.2.2.5 LCD_DATA3_PORT	4
4.2.2.6 LCD_DELAY_BOOTUP	4
4.2.2.7 LCD_DELAY_BUSY_FLAG	4
4.2.2.8 LCD_DELAY_ENABLE_PULSE	5
4.2.2.9 LCD_DELAY_INIT	5
4.2.2.10 LCD_DELAY_INIT_4BIT	5
4.2.2.11 LCD_DELAY_INIT_REP	5
4.2.2.12 LCD_DISP_LENGTH	5
4.2.2.13 LCD_IO_MODE	5
4.2.2.14 LCD_LINE_LENGTH	5
4.2.2.15 LCD_LINES	6
4.2.2.16 LCD_RW_PIN	6
4.2.2.17 LCD_RW_PORT	6
4.2.2.18 LCD_START_LINE1	6
4.2.2.19 LCD_START_LINE2	6
4.2.2.20 LCD_START_LINE3	6
4.2.2.21 LCD_START_LINE4	6

4.2.2.22 LCD_WRAP_LINES	17
4.2.3 Function Documentation	17
4.2.3.1 lcd_clrscr()	17
4.2.3.2 lcd_command()	17
4.2.3.3 lcd_data()	17
4.2.3.4 lcd_gotoxy()	18
4.2.3.5 lcd_home()	18
4.2.3.6 lcd_init()	18
4.2.3.7 lcd_putc()	19
4.2.3.8 lcd_puts()	19
4.2.3.9 lcd_puts_p()	20
4.3 Bargraphs controller < serial_controller_for_diodes.h>	21
4.3.1 Detailed Description	21
4.3.2 Function Documentation	21
4.3.2.1 DIODE_FRONT()	21
4.3.2.2 DIODE_init()	22
4.3.2.3 DIODE_REAR()	22
4.4 UART Library <uart.h></uart.h>	23
4.4.1 Detailed Description	24
4.4.2 Macro Definition Documentation	24
4.4.2.1 UART_BAUD_SELECT	24
4.4.2.2 UART_BAUD_SELECT_DOUBLE_SPEED	25
4.4.2.3 UART_RX_BUFFER_SIZE	25
4.4.2.4 UART_TX_BUFFER_SIZE	25
4.4.3 Function Documentation	25
4.4.3.1 uart1_getc()	25
4.4.3.2 uart1_init()	26
4.4.3.3 uart1_putc()	26
4.4.3.4 uart1_puts()	26
4.4.3.5 uart1_puts_p()	26
4.4.3.6 uart_getc()	27
4.4.3.7 uart_init()	27
4.4.3.8 uart_putc()	27
4.4.3.9 uart_puts()	28
4.4.3.10 uart_puts_p()	28
5 File Documentation	31
5.1 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/gpio.h File Reference	31
5.1.1 Detailed Description	31
5.2 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/← ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/lcd.h File Reference	32
5.2.1 Detailed Description	34

	5.3	C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/← ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/lcd_definitions.h File Reference	34
			35
	- 4		30
	5.4	C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/ ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/serial_controller_for_diodes.h File Reference	35
		5.4.1 Detailed Description	35
	5.5	C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/← ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/timer.h File Reference	35
		5.5.1 Detailed Description	36
	5.6	C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/← ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/uart.h File Reference	36
		5.6.1 Detailed Description	38
Inc	dex		39

Chapter 1

Main Page

Author

Filip Dusek Marek Svoboda 2 Main Page

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

GPIO <gpio.h></gpio.h>	7
LCD library <lcd.h></lcd.h>	11
Bargraphs controller < serial_controller_for_diodes.h >	2
UART Library <uart.h></uart.h>	23

4 Module Index

Chapter 3

File Index

3.1 File List

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

C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/ gpio.h	
Library for basic pin operation	31
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/ Icd.h	
Basic routines for LCD with driver HD44780U	32
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/Icd_definitions.h	
Defining connection of LCD to AVR	34
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/ serial_controller_for_diodes.h	
Bargraphs controller	35
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/timer.h	
Timer library for AVR-GCC	35
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/Digital-electronics-2/Labs/project/←	
ParkingAssistant_HC-SR04/ParkingAssistant_HC-SR04/ uart.h	
Basic routines for UART communication	36

6 File Index

Chapter 4

Module Documentation

4.1 GPIO < gpio.h >

Library for basic pin operation as set pin as OUT/IN...

Functions

• void **GPIO_config_output** (volatile uint8_t *reg_name, uint8_t pin_num)

Configure one output pin in Data Direction Register.

• void **GPIO_config_input_nopull** (volatile uint8_t *reg_name, uint8_t pin_num)

Configure one input pin in Data Direction Register without pull-up resistor.

• void **GPIO_config_input_pullup** (volatile uint8_t *reg_name, uint8_t pin_num)

Configure one input pin in Data Direction Register with pull-up resistor.

• void GPIO_write_low (volatile uint8_t *reg_name, uint8_t pin_num)

Set output to LOW.

• void GPIO_write_high (volatile uint8_t *reg_name, uint8_t pin_num)

Set output to HIGH.

void GPIO_toggle (volatile uint8_t *reg_name, uint8_t pin_num)

Change state of output (LOW to HIGH or HIGH to LOW).

uint8_t GPIO_read (volatile uint8_t *reg_name, uint8_t pin_num)

Read value of one input pin.

4.1.1 Detailed Description

Library for basic pin operation as set pin as OUT/IN... # include < gpio.h >

The library contains functions for controlling AVRs' gpio pin(s).

Note

Based on AVR Libc Reference Manual. Tested on ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2.

Copyright

(c) 2019-2020 Tomas Fryza Dept. of Radio Electronics, Brno University of Technology, Czechia This work is licensed under the terms of the MIT license.

4.1.2 Function Documentation

4.1.2.1 GPIO_config_input_nopull()

Configure one input pin in Data Direction Register without pull-up resistor.

Parameters

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.1.2.2 GPIO_config_input_pullup()

Configure one input pin in Data Direction Register with pull-up resistor.

Parameters

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.1.2.3 GPIO_config_output()

Configure one output pin in Data Direction Register.

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.1 GPIO <gpio.h> 9

4.1.2.4 GPIO_read()

Read value of one input pin.

Parameters

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.1.2.5 GPIO_toggle()

Change state of output (LOW to HIGH or HIGH to LOW).

Parameters

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.1.2.6 GPIO_write_high()

Set output to HIGH.

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,	
pin_num	- Pin designation in the interval 0 to 7	

4.1.2.7 GPIO_write_low()

Set output to LOW.

reg_name	- Address of Data Direction Register, such as &DDRA, &DDRB,
pin_num	- Pin designation in the interval 0 to 7

4.2 LCD library < lcd.h>

Basic routines for interfacing a HD44780U-based character LCD display.

Definition for LCD controller type

Use 0 for HD44780 controller, change to 1 for displays with KS0073 controller.

• #define LCD_CONTROLLER_KS0073 0

Definitions for Display Size

Change these definitions to adapt setting to your display

These definitions can be defined in a separate include file $lcd_definitions.h$ (p. 34) instead modifying this file by adding -D_LCD_DEFINITIONS_FILE to the CDEFS section in the Makefile. All definitions added to the file lcd_{\leftarrow} definitions.h (p. 34) will override the default definitions from lcd.h (p. 32)

- #define LCD_LINES 2
- #define LCD DISP LENGTH 16
- #define LCD LINE LENGTH 0x40
- #define LCD_START_LINE1 0x00
- #define LCD_START_LINE2 0x40
- #define LCD START LINE3 0x14
- #define LCD START LINE4 0x54
- #define LCD_WRAP_LINES 0

Definitions for 4-bit IO mode

The four LCD data lines and the three control lines RS, RW, E can be on the same port or on different ports. Change LCD_RS_PORT, LCD_RW_PORT, LCD_E_PORT if you want the control lines on different ports.

Normally the four data lines should be mapped to bit 0..3 on one port, but it is possible to connect these data lines in different order or even on different ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.

Adjust these definitions to your target.

These definitions can be defined in a separate include file **lcd_definitions.h** (p. **34**) instead modifying this file by adding **-D_LCD_DEFINITIONS_FILE** to the **CDEFS** section in the Makefile. All definitions added to the file **lcd_**← **definitions.h** (p. 34) will override the default definitions from **lcd.h** (p. 32)

- #define LCD_IO_MODE 1
- #define LCD_DATA0_PORT LCD_PORT
- #define LCD_DATA1_PORT LCD_PORT
- #define LCD DATA2 PORT LCD PORT
- #define LCD DATA3 PORT LCD PORT
- #define LCD RW PORT LCD PORT
- #define LCD_RW_PIN 5

Definitions of delays

Used to calculate delay timers. Adapt the F_CPU define in the Makefile to the clock frequency in Hz of your target

These delay times can be adjusted, if some displays require different delays.

These definitions can be defined in a separate include file **lcd_definitions.h** (p. **34**) instead modifying this file by adding **-D_LCD_DEFINITIONS_FILE** to the **CDEFS** section in the Makefile. All definitions added to the file **lcd_**← **definitions.h** (p. 34) will override the default definitions from **lcd.h** (p. 32)

- #define LCD_DELAY_BOOTUP 16000
- #define LCD_DELAY_INIT 5000
- #define LCD_DELAY_INIT_REP 64
- #define LCD DELAY INIT 4BIT 64
- #define LCD_DELAY_BUSY_FLAG 4
- #define LCD DELAY ENABLE PULSE 1

Definitions for LCD command instructions

The constants define the various LCD controller instructions which can be passed to the function **lcd_command()** (p. 17), see HD44780 data sheet for a complete description.

- #define LCD_CLR 0 /* DB0: clear display */
- #define LCD_HOME 1 /* DB1: return to home position */
- #define LCD_ENTRY_MODE 2 /* DB2: set entry mode */
- #define LCD ENTRY INC 1 /* DB1: 1=increment, 0=decrement */
- #define LCD_ENTRY_SHIFT 0 /* DB2: 1=display shift on */
- #define LCD_ON 3 /* DB3: turn lcd/cursor on */
- #define LCD_ON_DISPLAY 2 /* DB2: turn display on */
- #define LCD_ON_CURSOR 1 /* DB1: turn cursor on */
- #define LCD ON BLINK 0 /* DB0: blinking cursor ? */
- #define LCD_MOVE 4 /* DB4: move cursor/display */
- #define LCD_MOVE_DISP 3 /* DB3: move display (0-> cursor) ? */
- #define LCD_MOVE_RIGHT 2 /* DB2: move right (0-> left) ? */
- #define LCD_FUNCTION 5 /* DB5: function set */
- #define LCD FUNCTION_8BIT 4 /* DB4: set 8BIT mode (0->4BIT mode) */
- #define LCD FUNCTION 2LINES 3 /* DB3: two lines (0->one line) */
- #define LCD FUNCTION 10DOTS 2 /* DB2: 5x10 font (0->5x7 font) */
- #define LCD_CGRAM 6 /* DB6: set CG RAM address */
- #define LCD_DDRAM 7 /* DB7: set DD RAM address */
- #define LCD_BUSY 7 /* DB7: LCD is busy */
- #define LCD_ENTRY_DEC 0x04 /* display shift off, dec cursor move dir */
- #define LCD_ENTRY_DEC_SHIFT 0x05 /* display shift on, dec cursor move dir */
- #define LCD ENTRY INC 0x06 /* display shift off, inc cursor move dir */
- #define LCD_ENTRY_INC_SHIFT 0x07 /* display shift on, inc cursor move dir */
- #define LCD_DISP_OFF 0x08 /* display off */
- #define LCD_DISP_ON 0x0C /* display on, cursor off */
- #define LCD_DISP_ON_BLINK 0x0D /* display on, cursor off, blink char */
- #define LCD_DISP_ON_CURSOR 0x0E /* display on, cursor on */
- #define LCD DISP ON CURSOR BLINK 0x0F /* display on, cursor on, blink char */
- #define LCD MOVE CURSOR LEFT 0x10 /* move cursor left (decrement) */
- #define LCD_MOVE_CURSOR_RIGHT 0x14 /* move cursor right (increment) */
- #define LCD_MOVE_DISP_LEFT 0x18 /* shift display left */
- #define LCD MOVE DISP_RIGHT 0x1C /* shift display right */
- #define LCD_FUNCTION_4BIT_1LINE 0x20 /* 4-bit interface, single line, 5x7 dots */
- #define $LCD_FUNCTION_4BIT_2LINES$ 0x28 /* 4-bit interface, dual line, 5x7 dots */
- #define LCD FUNCTION 8BIT 1LINE 0x30 /* 8-bit interface, single line, 5x7 dots */
- #define LCD FUNCTION 8BIT 2LINES 0x38 /* 8-bit interface, dual line, 5x7 dots */
- #define LCD_MODE_DEFAULT ((1 << LCD_ENTRY_MODE) | (1 << LCD_ENTRY_INC))

Functions

• void **lcd init** (uint8 t dispAttr)

Initialize display and select type of cursor.

void lcd_clrscr (void)

Clear display and set cursor to home position.

void Icd_home (void)

Set cursor to home position.

• void **lcd_gotoxy** (uint8_t x, uint8_t y)

Set cursor to specified position.

void lcd_putc (char c)

Display character at current cursor position.

• void **lcd_puts** (const char *s)

Display string without auto linefeed.

void lcd_puts_p (const char *progmem_s)

Display string from program memory without auto linefeed.

void lcd_command (uint8_t cmd)

Send LCD controller instruction command.

• void Icd_data (uint8_t data)

Send data byte to LCD controller.

#define lcd_puts_P(_s) lcd_puts_p(PSTR(_s))

macros for automatically storing string constant in program memory

4.2.1 Detailed Description

Basic routines for interfacing a HD44780U-based character LCD display.

LCD character displays can be found in many devices, like espresso machines, laser printers. The Hitachi HD44780 controller and its compatible controllers like Samsung KS0066U have become an industry standard for these types of displays.

This library allows easy interfacing with a HD44780 compatible display and can be operated in memory mapped mode (LCD_IO_MODE defined as 0 in the include file **lcd.h** (p. 32).) or in 4-bit IO port mode (LCD_IO_MODE defined as 1). 8-bit IO port mode is not supported.

Memory mapped mode is compatible with old Kanda STK200 starter kit, but also supports generation of R/W signal through A8 address line.

See also

The chapter Interfacing a HD44780 Based LCD to an AVR on my home page, which shows example circuits how to connect an LCD to an AVR controller.

Author

Peter Fleury pfleury@gmx.ch http://tinyurl.com/peterfleury

Version

2.0

Copyright

(C) 2015 Peter Fleury, GNU General Public License Version 3

4.2.2 Macro Definition Documentation

4.2.2.1 LCD_CONTROLLER_KS0073

#define LCD_CONTROLLER_KS0073 0

Use 0 for HD44780 controller, 1 for KS0073 controller

4.2.2.2 LCD_DATA0_PORT

#define LCD_DATAO_PORT LCD_PORT

port for 4bit data bit 0

4.2.2.3 LCD_DATA1_PORT

#define LCD_DATA1_PORT LCD_PORT

port for 4bit data bit 1

4.2.2.4 LCD_DATA2_PORT

#define LCD_DATA2_PORT LCD_PORT

port for 4bit data bit 2

4.2.2.5 LCD_DATA3_PORT

#define LCD_DATA3_PORT LCD_PORT

port for 4bit data bit 3

4.2.2.6 LCD_DELAY_BOOTUP

#define LCD_DELAY_BOOTUP 16000

delay in micro seconds after power-on

4.2.2.7 LCD_DELAY_BUSY_FLAG

#define LCD_DELAY_BUSY_FLAG 4

time in micro seconds the address counter is updated after busy flag is cleared

4.2.2.8 LCD_DELAY_ENABLE_PULSE

```
#define LCD_DELAY_ENABLE_PULSE 1
```

enable signal pulse width in micro seconds

4.2.2.9 LCD_DELAY_INIT

```
#define LCD_DELAY_INIT 5000
```

delay in micro seconds after initialization command sent

4.2.2.10 LCD_DELAY_INIT_4BIT

```
#define LCD_DELAY_INIT_4BIT 64
```

delay in micro seconds after setting 4-bit mode

4.2.2.11 LCD_DELAY_INIT_REP

```
#define LCD_DELAY_INIT_REP 64
```

delay in micro seconds after initialization command repeated

4.2.2.12 LCD_DISP_LENGTH

```
#define LCD_DISP_LENGTH 16
```

visibles characters per line of the display

4.2.2.13 LCD_IO_MODE

```
#define LCD_IO_MODE 1
```

0: memory mapped mode, 1: IO port mode

4.2.2.14 LCD_LINE_LENGTH

```
#define LCD_LINE_LENGTH 0x40
```

internal line length of the display

4.2.2.15 LCD_LINES

#define LCD_LINES 2

number of visible lines of the display

4.2.2.16 LCD_RW_PIN

#define LCD_RW_PIN 5

pin for RW line

4.2.2.17 LCD_RW_PORT

#define LCD_RW_PORT LCD_PORT

port for RW line

4.2.2.18 LCD_START_LINE1

#define LCD_START_LINE1 0x00

DDRAM address of first char of line 1

4.2.2.19 LCD_START_LINE2

#define LCD_START_LINE2 0x40

DDRAM address of first char of line 2

4.2.2.20 LCD_START_LINE3

#define LCD_START_LINE3 0x14

DDRAM address of first char of line 3

4.2.2.21 LCD_START_LINE4

#define LCD_START_LINE4 0x54

DDRAM address of first char of line 4

4.2.2.22 LCD_WRAP_LINES

```
#define LCD_WRAP_LINES 0
```

0: no wrap, 1: wrap at end of visibile line

4.2.3 Function Documentation

4.2.3.1 lcd_clrscr()

```
void lcd_clrscr (
     void )
```

Clear display and set cursor to home position.

Returns

none

4.2.3.2 lcd_command()

Send LCD controller instruction command.

Parameters

cmd instruction to send to LCD controller, see HD44780 data sheet

Returns

none

4.2.3.3 lcd_data()

Send data byte to LCD controller.

Similar to Icd_putc() (p. 19), but without interpreting LF

Parameters

Returns

none

4.2.3.4 lcd_gotoxy()

Set cursor to specified position.

Parameters

X	horizontal position (0: left most position)	
У	vertical position (0: first line)	

Returns

none

4.2.3.5 lcd_home()

```
void lcd_home (
     void )
```

Set cursor to home position.

Returns

none

4.2.3.6 lcd_init()

Initialize display and select type of cursor.

Parameters

	dispAttr	LCD_DISP_OFF display off	
		LCD_DISP_ON display on, cursor off	
LCD_DISP_ON_CURSOR display on, cursor on		LCD_DISP_ON_CURSOR display on, cursor on	
		LCD_DISP_ON_CURSOR_BLINK display on, cursor on flashing	

Returns

none

4.2.3.7 lcd_putc()

```
void lcd_putc ( \mbox{char } c \mbox{ )}
```

Display character at current cursor position.

Parameters

c character to be displayed

Returns

none

4.2.3.8 lcd_puts()

```
void lcd_puts ( {\tt const\ char\ *\ s\ )}
```

Display string without auto linefeed.

Parameters

s string to be displayed

Returns

none

4.2.3.9 lcd_puts_p()

Display string from program memory without auto linefeed.

Parameters

progmem←	string from program memory be be displayed
S	

Returns

none

See also

 $\textbf{lcd_puts_P} \; (p. \; 13)$

4.3 Bargraphs controller < serial_controller_for_diodes.h>

Basic routines for interfacing two bargraphs.

Connection of hardware.

- #define **DATA_INPUT_FRONT** PC0
- #define DATA INPUT REAR PC3
- #define SEGMENT_LATCH_BAR_1 PC1
- #define SEGMENT_LATCH_BAR_2 PC4
- #define **SEGMENT_CLK** PC2

Functions.

void **DIODE_init** (void)

Initialization of bargraphs driver.

• void **DIODE_FRONT** (uint8_t number_of_diodes_bar_1)

Displaying front distance by bargraph.

• void **DIODE_REAR** (uint8_t number_of_diodes_bar_2)

Displaying rear distance by bargraph.

4.3.1 Detailed Description

Basic routines for interfacing two bargraphs.

```
#include <serial_controller_for_diodes.h>
#include <lcd_definitions.h>
```

Author

Filip Dusek, Marek Svoboda

4.3.2 Function Documentation

4.3.2.1 DIODE FRONT()

Displaying front distance by bargraph.

number_of_diodes_bar↔	number of LEDs that should be displayed	
1		

Returns

none

4.3.2.2 DIODE_init()

```
void DIODE_init (
     void )
```

Initialization of bargraphs driver.

Returns

none

4.3.2.3 DIODE_REAR()

Displaying rear distance by bargraph.

Parameters

number_of_diodes_bar↔	number of LEDs that should be displayed
2	

Returns

none

4.4 UART Library <uart.h>

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

Macros

#define UART_BAUD_SELECT(baudRate, xtalCpu) (((xtalCpu) + 8UL * (baudRate)) / (16UL * (baudRate))
 - 1UL)

UART Baudrate Expression.

#define UART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((((xtalCpu) + 4UL * (baudRate)) / (8UL * (baudRate)) - 1UL)) | 0x8000)

UART Baudrate Expression for ATmega double speed mode.

• #define UART_RX_BUFFER_SIZE 32

Size of the circular receive buffer, must be power of 2.

• #define UART_TX_BUFFER_SIZE 32

Size of the circular transmit buffer, must be power of 2.

#define UART FRAME ERROR 0x1000

Framing Error by UART

#define UART OVERRUN ERROR 0x0800

Overrun condition by UART

#define UART PARITY ERROR 0x0400

Parity Error by UART

• #define UART_BUFFER_OVERFLOW 0x0200

receive ringbuffer overflow

• #define **UART_NO_DATA** 0x0100

no receive data available

#define uart_puts_P(__s) uart_puts_p(PSTR(__s))

Macro to automatically put a string constant into program memory.

• #define $uart1_puts_P(_s)$ $uart1_puts_p(PSTR(_s))$

Macro to automatically put a string constant into program memory.

Functions

· void uart_init (unsigned int baudrate)

Initialize UART and set baudrate.

• unsigned int uart_getc (void)

Get received byte from ringbuffer.

void uart_putc (unsigned char data)

Put byte to ringbuffer for transmitting via UART.

• void $uart_puts$ (const char *s)

Put string to ringbuffer for transmitting via UART.

void uart_puts_p (const char *s)

Put string from program memory to ringbuffer for transmitting via UART.

void uart1_init (unsigned int baudrate)

Initialize USART1 (only available on selected ATmegas)

unsigned int uart1_getc (void)

Get received byte of USART1 from ringbuffer. (only available on selected ATmega)

• void **uart1_putc** (unsigned char data)

Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)

• void uart1_puts (const char *s)

Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)

• void **uart1_puts_p** (const char *s)

Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

4.4.1 Detailed Description

Interrupt UART library using the built-in UART with transmit and receive circular buffers. #include <uart.h>

This library can be used to transmit and receive data through the built in UART.

An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

The UART_RX_BUFFER_SIZE and UART_TX_BUFFER_SIZE constants define the size of the circular buffers in bytes. Note that these constants must be a power of 2. You may need to adapt these constants to your target and your application by adding CDEFS += -DUART_RX_BUFFER_SIZE=nn -DUART_TX_BUFFER_SIZE=nn to your Makefile.

Note

Based on Atmel Application Note AVR306

Author

```
Peter Fleury pfleury@gmx.ch http://tinyurl.com/peterfleury
```

Copyright

(C) 2015 Peter Fleury, GNU General Public License Version 3

4.4.2 Macro Definition Documentation

4.4.2.1 UART_BAUD_SELECT

UART Baudrate Expression.

xtalCpu	system clock in Mhz, e.g. 4000000UL for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

4.4.2.2 UART BAUD SELECT DOUBLE SPEED

UART Baudrate Expression for ATmega double speed mode.

Parameters

xtalCpu	system clock in Mhz, e.g. 4000000UL for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

4.4.2.3 UART_RX_BUFFER_SIZE

```
#define UART_RX_BUFFER_SIZE 32
```

Size of the circular receive buffer, must be power of 2.

You may need to adapt this constant to your target and your application by adding CDEFS += -DUART_RX_BUF← FER_SIZE=nn to your Makefile.

4.4.2.4 UART_TX_BUFFER_SIZE

```
#define UART_TX_BUFFER_SIZE 32
```

Size of the circular transmit buffer, must be power of 2.

You may need to adapt this constant to your target and your application by adding CDEFS += -DUART_TX_BUF \leftarrow FER_SIZE=nn to your Makefile.

4.4.3 Function Documentation

4.4.3.1 uart1_getc()

Get received byte of USART1 from ringbuffer. (only available on selected ATmega)

See also

uart_getc (p. 26)

```
4.4.3.2 uart1_init()
```

```
void uart1_init (
          unsigned int baudrate )
```

Initialize USART1 (only available on selected ATmegas)

See also

```
uart_init (p. 27)
```

4.4.3.3 uart1_putc()

```
void uart1_putc (
          unsigned char data )
```

Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

```
uart_putc (p. 27)
```

4.4.3.4 uart1_puts()

```
void uart1_puts ( {\rm const~char~*}~s~)
```

Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

```
uart_puts (p. 28)
```

4.4.3.5 uart1_puts_p()

Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

```
uart_puts_p (p. 28)
```

4.4.3.6 uart_getc()

Get received byte from ringbuffer.

Returns in the lower byte the received character and in the higher byte the last receive error. UART_NO_DATA is returned when no data is available.

Returns

lower byte: received byte from ringbuffer

higher byte: last receive status

- 0 successfully received data from UART
- · UART NO DATA

no receive data available

UART_BUFFER_OVERFLOW

Receive ringbuffer overflow. We are not reading the receive buffer fast enough, one or more received character have been dropped

UART OVERRUN ERROR

Overrun condition by UART. A character already present in the UART UDR register was not read by the interrupt handler before the next character arrived, one or more received characters have been dropped.

UART FRAME ERROR

Framing Error by UART

4.4.3.7 uart init()

Initialize UART and set baudrate.

Parameters

baudrate Specify baudrate using macro UART_BAUD_SELECT() (p. 24)

Returns

none

4.4.3.8 uart putc()

Put byte to ringbuffer for transmitting via UART.

Parameters

data	byte to be transmitted
------	------------------------

Returns

none

4.4.3.9 uart_puts()

```
void uart_puts ( {\tt const\ char\ *\ s\ )}
```

Put string to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

```
s string to be transmitted
```

Returns

none

4.4.3.10 uart_puts_p()

Put string from program memory to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

s program memory string to be transmitted

Returns

none

See also

uart_puts_P (p. 23)

Chapter 5

File Documentation

5.1 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/
Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/
ParkingAssistant_HC-SR04/gpio.h File
Reference

Library for basic pin operation.

#include <avr/io.h>

Functions

- void **GPIO_config_output** (volatile uint8_t *reg_name, uint8_t pin_num)
- Configure one output pin in Data Direction Register.

 void GPIO_config_input_nopull (volatile uint8_t *reg_name, uint8_t pin_num)

Configure one input pin in Data Direction Register without pull-up resistor.

- void **GPIO_config_input_pullup** (volatile uint8_t *reg_name, uint8_t pin_num)

 Configure one input pin in Data Direction Register with pull-up resistor.
- void **GPIO_write_low** (volatile uint8_t *reg_name, uint8_t pin_num)

Set output to LOW.

• void **GPIO_write_high** (volatile uint8_t *reg_name, uint8_t pin_num)

Set output to HIGH.

• void **GPIO_toggle** (volatile uint8_t *reg_name, uint8_t pin_num)

Change state of output (LOW to HIGH or HIGH to LOW).

• uint8_t **GPIO_read** (volatile uint8_t *reg_name, uint8_t pin_num)

Read value of one input pin.

5.1.1 Detailed Description

Library for basic pin operation.

32 File Documentation

5.2 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/ Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/ ParkingAssistant_HC-SR04/Icd.h File Reference

Basic routines for LCD with driver HD44780U.

```
#include <inttypes.h>
#include <avr/pgmspace.h>
#include "lcd_definitions.h"
```

Macros

Definition for LCD controller type

Use 0 for HD44780 controller, change to 1 for displays with KS0073 controller.

• #define LCD CONTROLLER KS0073 0

Definitions for Display Size

Change these definitions to adapt setting to your display

These definitions can be defined in a separate include file **Icd_definitions.h** (p. **34**) instead modifying this file by adding -D_LCD_DEFINITIONS_FILE to the CDEFS section in the Makefile. All definitions added to the file **Icd_definitions.h** (p. 34) will override the default definitions from **Icd.h** (p. 32)

- #define LCD LINES 2
- #define LCD DISP LENGTH 16
- #define LCD_LINE_LENGTH 0x40
- #define LCD START LINE1 0x00
- #define LCD START LINE2 0x40
- #define LCD_START_LINE3 0x14
- #define LCD_START_LINE4 0x54
- #define LCD_WRAP_LINES 0

Definitions for 4-bit IO mode

The four LCD data lines and the three control lines RS, RW, E can be on the same port or on different ports. Change LCD RS PORT, LCD RW PORT, LCD E PORT if you want the control lines on different ports.

Normally the four data lines should be mapped to bit 0..3 on one port, but it is possible to connect these data lines in different order or even on different ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.

Adjust these definitions to your target.

These definitions can be defined in a separate include file **Icd_definitions.h** (p. **34**) instead modifying this file by adding **-D_LCD_DEFINITIONS_FILE** to the **CDEFS** section in the Makefile. All definitions added to the file **Icd_definitions.h** (p. 34) will override the default definitions from **Icd.h** (p. 32)

- #define LCD IO MODE 1
- #define LCD_DATA0_PORT_LCD_PORT
- #define LCD_DATA1_PORT LCD_PORT
- #define LCD_DATA2_PORT LCD_PORT
- #define LCD_DATA3_PORT LCD_PORT
- #define LCD RW PORT LCD PORT
- #define LCD_RW_PIN 5

Reference 33

Definitions of delays

Used to calculate delay timers. Adapt the F_CPU define in the Makefile to the clock frequency in Hz of your target

These delay times can be adjusted, if some displays require different delays.

These definitions can be defined in a separate include file **Icd_definitions.h** (p. **34**) instead modifying this file by adding **-D_LCD_DEFINITIONS_FILE** to the **CDEFS** section in the Makefile. All definitions added to the file **Icd_definitions.h** (p. 34) will override the default definitions from **Icd.h** (p. 32)

- #define LCD DELAY BOOTUP 16000
- #define LCD DELAY INIT 5000
- #define LCD DELAY INIT_REP 64
- #define LCD_DELAY_INIT_4BIT 64
- #define LCD DELAY BUSY FLAG 4
- #define LCD_DELAY_ENABLE_PULSE 1

Definitions for LCD command instructions

The constants define the various LCD controller instructions which can be passed to the function **Icd_**← **command()** (p. 17), see HD44780 data sheet for a complete description.

- #define LCD_CLR 0 /* DB0: clear display */
- #define LCD_HOME 1 /* DB1: return to home position */
- #define LCD_ENTRY_MODE 2 /* DB2: set entry mode */
- #define LCD_ENTRY_INC 1 /* DB1: 1=increment, 0=decrement */
- #define LCD_ENTRY_SHIFT 0 /* DB2: 1=display shift on */
- #define LCD ON 3 /* DB3: turn lcd/cursor on */
- #define LCD ON DISPLAY 2 /* DB2: turn display on */
- #define LCD ON CURSOR 1 /* DB1: turn cursor on */
- #define LCD ON BLINK 0 /* DB0: blinking cursor ? */
- #define **LCD MOVE** 4 /* DB4: move cursor/display */
- #define LCD_MOVE_DISP 3 /* DB3: move display (0-> cursor) ? */
- #define LCD_MOVE_RIGHT 2 /* DB2: move right (0-> left) ? */
- #define LCD_FUNCTION 5 /* DB5: function set */
- #define LCD_FUNCTION_8BIT 4 /* DB4: set 8BIT mode (0->4BIT mode) */
- #define LCD FUNCTION 2LINES 3 /* DB3: two lines (0->one line) */
- #define LCD_FUNCTION_10DOTS 2 /* DB2: 5x10 font (0->5x7 font) */
- #define LCD CGRAM 6 /* DB6: set CG RAM address */
- #define LCD_DDRAM 7 /* DB7: set DD RAM address */
- #define LCD_BUSY 7 /* DB7: LCD is busy */
- #define LCD ENTRY DEC 0x04 /* display shift off, dec cursor move dir */
- #define LCD_ENTRY_DEC_SHIFT 0x05 /* display shift on, dec cursor move dir */
- #define LCD_ENTRY_INC_ 0x06 /* display shift off, inc cursor move dir */
- #define LCD ENTRY INC SHIFT 0x07 /* display shift on, inc cursor move dir */
- #define LCD_DISP_OFF 0x08 /* display off */
- #define LCD DISP ON 0x0C /* display on, cursor off */
- #define LCD_DISP_ON_BLINK 0x0D /* display on, cursor off, blink char */
- #define LCD_DISP_ON_CURSOR 0x0E /* display on, cursor on */
- #define LCD_DISP_ON_CURSOR_BLINK 0x0F /* display on, cursor on, blink char */
- #define LCD_MOVE_CURSOR_LEFT 0x10 /* move cursor left (decrement) */
- #define LCD MOVE CURSOR RIGHT 0x14 /* move cursor right (increment) */
- #define LCD_MOVE_DISP_LEFT 0x18 /* shift display left */
- #define LCD_MOVE_DISP_RIGHT 0x1C /* shift display right */
- #define LCD_FUNCTION_4BIT_1LINE 0x20 /* 4-bit interface, single line, 5x7 dots */
- #define LCD_FUNCTION_4BIT_2LINES 0x28 /* 4-bit interface, dual line, 5x7 dots */
- #define LCD FUNCTION 8BIT 1LINE 0x30 /* 8-bit interface, single line, 5x7 dots */
- #define LCD_FUNCTION_8BIT_2LINES 0x38 /* 8-bit interface, dual line, 5x7 dots */
- #define LCD_MODE_DEFAULT ((1 << LCD_ENTRY_MODE) | (1 << LCD_ENTRY_INC))

34 File Documentation

Functions

#define lcd_puts_P(_s) lcd_puts_p(PSTR(_s))

macros for automatically storing string constant in program memory

void lcd_init (uint8_t dispAttr)

Initialize display and select type of cursor.

void lcd_clrscr (void)

Clear display and set cursor to home position.

void Icd_home (void)

Set cursor to home position.

• void **lcd_gotoxy** (uint8_t x, uint8_t y)

Set cursor to specified position.

• void **lcd_putc** (char c)

Display character at current cursor position.

void Icd_puts (const char *s)

Display string without auto linefeed.

void lcd_puts_p (const char *progmem_s)

Display string from program memory without auto linefeed.

void Icd command (uint8 t cmd)

Send LCD controller instruction command.

· void Icd data (uint8 t data)

Send data byte to LCD controller.

5.2.1 Detailed Description

Basic routines for LCD with driver HD44780U.

5.3 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/
Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/
ParkingAssistant_HC-SR04/Icd_definitions.h File
Reference

Defining connection of LCD to AVR.

Macros

- #define LCD_PORT PORTD
- #define LCD DATA0 PIN PD4
- #define LCD_DATA1_PIN PD5
- #define LCD DATA2 PIN PD6
- #define LCD_DATA3_PIN PD7
- #define LCD_RS_PORT PORTB
- #define LCD RS PIN PB0
- #define LCD E PORT PORTB
- #define LCD_E_PIN PB1

5.3.1 Detailed Description

Defining connection of LCD to AVR.

Copyright

(c) 2019-2020 Tomas Fryza Dept. of Radio Electronics, Brno University of Technology, Czechia This work is licensed under the terms of the MIT license.

5.4 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/
Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/
ParkingAssistant_HC-SR04/serial_controller_for_diodes.h File
Reference

Bargraphs controller.

Macros

Connection of hardware.

- #define **DATA_INPUT_FRONT** PC0
- #define **DATA_INPUT_REAR** PC3
- #define **SEGMENT_LATCH_BAR_1** PC1
- #define SEGMENT_LATCH_BAR_2 PC4
- #define **SEGMENT_CLK** PC2

Functions

Functions.

• void **DIODE_init** (void)

Initialization of bargraphs driver.

• void **DIODE_FRONT** (uint8_t number_of_diodes_bar_1)

Displaying front distance by bargraph.

void DIODE_REAR (uint8_t number_of_diodes_bar_2)

Displaying rear distance by bargraph.

5.4.1 Detailed Description

Bargraphs controller.

5.5 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/
Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/
ParkingAssistant_HC-SR04/timer.h File
Reference

Timer library for AVR-GCC.

#include <avr/io.h>

36 File Documentation

Macros

```
    #define TIM0_stop() TCCR0B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));</li>

• #define TIMO overflow 16us() TCCR0B &= \sim((1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00);
• #define TIM0_overflow_128us() TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS01);
• #define TIMO_overflow_1ms() TCCR0B &= ~(1<<CS02); TCCR0B |= ((1<<CS01) | (1<<CS00));
• #define TIM0_overflow_4ms() TCCR0B &= ~((1<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS02);
• #define TIM0_overflow_16ms() TCCR0B &= \sim(1<<CS01); TCCR0B |= ((1<<CS02) | (1<<CS00));
• #define TIMO overflow interrupt enable() TIMSK0 |= (1<<TOIE0);

    #define TIMO overflow interrupt disable() TIMSK0 &= ~(1<<TOIE0);</li>

• #define TIM1 stop() TCCR1B &= \sim((1<<CS12) | (1<<CS11) | (1<<CS10));
• #define TIM1_overflow_4ms() TCCR1B &= ~((1<<CS12) | (1<<CS11)); TCCR1B |= (1<<CS10);
• #define TIM1_overflow_33ms() TCCR1B &= ~((1<<CS12) | (1<<CS10)); TCCR1B |= (1<<CS11);
• #define TIM1_overflow_262ms() TCCR1B &= ~(1<<CS12); TCCR1B |= (1<<CS11) | (1<<CS10);
• #define TIM1_overflow_1s() TCCR1B &= ~((1<<CS11) | (1<<CS10)); TCCR1B |= (1<<CS12);
• #define TIM1 overflow 4s() TCCR1B &= ~(1<<CS11); TCCR1B |= (1<<CS12) | (1<<CS10);

    #define TIM1_overflow_interrupt_enable() TIMSK1 |= (1<<TOIE1);</li>

• #define TIM1 overflow interrupt disable() TIMSK1 &= \sim(1<<TOIE1);

    #define TIM2_stop() TCCR2B &= ~((1<<CS22) | (1<<CS21) | (1<<CS20));</li>

• #define TIM2_overflow_16us() TCCR2B &= ~((1<<CS22) | (1<<CS21)); TCCR2B |= (1<<CS20);
• #define TIM2 overflow 128us() TCCR2B &= \sim((1 << CS20) | (1 << CS20)); TCCR2B |= (1 << CS21);
• #define TIM2_overflow_512us() TCCR2B &= ~(1<<CS21); TCCR2B |= ((1<<CS21) | (1<<CS20));
• #define TIM2 overflow 1ms() TCCR2B &= ~((1<<CS21) | (1<<CS20)); TCCR2B |= (1<<CS22);
• #define TIM2_overflow_2ms() TCCR2B &= ~(1 << CS21); TCCR2B |= ((1 << CS22) | (1 < CS20));
• #define TIM2 overflow 4ms() TCCR2B \&= (1 << CS20); TCCR2B |= ((1 << CS2) | (1 << CS21));
• #define TIM2 overflow 16ms() TCCR2B |= ((1 << CS22) | (1 << CS21) | (1 << CS20));

    #define TIM2 overflow interrupt enable() TIMSK2 |= (1<<TOIE2);</li>

• #define TIM2 overflow interrupt disable() TIMSK2 &= \sim(1<<TOIE2);
```

5.5.1 Detailed Description

Timer library for AVR-GCC.

The library contains macros for controlling the timer modules. Every timer has some possibility to set overflow time (it depends on pre-scalers). For interrupt from timer overflow vector is necessary to enable it.

Note

Based on Microchip Atmel ATmega328P manual and no source file is needed for the library.

Copyright

(c) 2019-2020 Tomas Fryza Dept. of Radio Electronics, Brno University of Technology, Czechia This work is licensed under the terms of the MIT license.

5.6 C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektronika_2/ Digital-electronics-2/Labs/project/ParkingAssistant_HC-SR04/ ParkingAssistant_HC-SR04/uart.h File Reference

Basic routines for UART communication.

#include <avr/pgmspace.h>

Reference 37

Macros

#define UART_BAUD_SELECT(baudRate, xtalCpu) (((xtalCpu) + 8UL * (baudRate)) / (16UL * (baudRate))
 - 1UL)

UART Baudrate Expression.

• #define **UART_BAUD_SELECT_DOUBLE_SPEED**(baudRate, xtalCpu) (((((xtalCpu) + 4UL * (baudRate)) / (8UL * (baudRate)) - 1UL)) | 0x8000)

UART Baudrate Expression for ATmega double speed mode.

#define UART_RX BUFFER SIZE 32

Size of the circular receive buffer, must be power of 2.

• #define UART TX BUFFER SIZE 32

Size of the circular transmit buffer, must be power of 2.

• #define UART_FRAME_ERROR 0x1000

Framing Error by UART

• #define UART OVERRUN ERROR 0x0800

Overrun condition by UART

• #define UART PARITY ERROR 0x0400

Parity Error by UART

• #define UART BUFFER OVERFLOW 0x0200

receive ringbuffer overflow

• #define UART_NO_DATA 0x0100

no receive data available

#define uart_puts_P(_s) uart_puts_p(PSTR(_s))

Macro to automatically put a string constant into program memory.

• #define $uart1_puts_P(_s)$ $uart1_puts_p(PSTR(_s))$

Macro to automatically put a string constant into program memory.

Functions

void uart_init (unsigned int baudrate)

Initialize UART and set baudrate.

unsigned int uart_getc (void)

Get received byte from ringbuffer.

• void **uart_putc** (unsigned char data)

Put byte to ringbuffer for transmitting via UART.

• void **uart_puts** (const char *s)

Put string to ringbuffer for transmitting via UART.

void uart_puts_p (const char *s)

Put string from program memory to ringbuffer for transmitting via UART.

void uart1_init (unsigned int baudrate)

Initialize USART1 (only available on selected ATmegas)

unsigned int uart1_getc (void)

Get received byte of USART1 from ringbuffer. (only available on selected ATmega)

void uart1_putc (unsigned char data)

Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)

void uart1_puts (const char *s)

Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)

void uart1_puts_p (const char *s)

Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

38 File Documentation

5.6.1 Detailed Description

Basic routines for UART communication.

Index

Bargraphs controller <serial_controller_for_diodes.h>,</serial_controller_for_diodes.h>	GPIO <gpio.h>, 8</gpio.h>
21	GPIO_read
DIODE_FRONT, 21	GPIO <gpio.h>, 9</gpio.h>
DIODE_init, 22	GPIO_toggle
DIODE_REAR, 22	GPIO <gpio.h>, 9</gpio.h>
0.41 440/D	GPIO_write_high
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	
SR04/ParkingAssistant_HC-SR04/gpio.h, 31	GPIO <gpio.h>, 9</gpio.h>
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	
SR04/ParkingAssistant_HC-SR04/lcd.h, 32	lcd_clrscr, 17
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	
SR04/ParkingAssistant_HC-SR04/lcd_definitio	
34	LCD_DATA0_PORT, 14
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	
SR04/ParkingAssistant_HC-SR04/serial_contro	LCD_DELAY_BOOTUP, 14
35	
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	LCD DELAY INIT, 15
SR04/ParkingAssistant_HC-SR04/timer.h, 35	:
C:/Users/MS/Documents/VUT/Cislicovka/Digitalni_elektro	
electronics-2/Labs/project/ParkingAssistant_H0	LCD_DISP_LENGTH, 15
SR04/ParkingAssistant_HC-SR04/uart.h, 36	lcd_gotoxy, 18
DIODE_FRONT	lcd_home, 18
Bargraphs controller < serial_controller_for_diodes.h	
21	LCD_IO_MODE, 15
DIODE init	LCD_LINE_LENGTH, 15
Bargraphs controller < serial_controller_for_diodes.h	
22	lcd_putc, 19
DIODE REAR	lcd_puts, 19
Bargraphs controller < serial_controller_for_diodes.h	
22	LCD_RW_PIN, 16
	LCD_RW_PORT, 16
GPIO <gpio.h>, 7</gpio.h>	LCD_START_LINE1, 16
GPIO_config_input_nopull, 8	LCD START LINE2, 16
GPIO_config_input_pullup, 8	LCD START LINE3, 16
GPIO config output, 8	LCD START LINE4, 16
GPIO_read, 9	LCD WRAP LINES, 16
GPIO_toggle, 9	lcd_clrscr
GPIO write high, 9	LCD library <lcd.h>, 17</lcd.h>
GPIO write low, 9	lcd_command
GPIO_config_input_nopull	LCD library <lcd.h>, 17</lcd.h>
GPIO <gpio.h>, 8</gpio.h>	LCD_CONTROLLER_KS0073
GPIO_config_input_pullup	LCD library <lcd.h>, 14</lcd.h>
GPIO <gpio.h>, 8</gpio.h>	lcd_data
GPIO_config_output	LCD library <lcd.h>, 17</lcd.h>
1	,,

40 INDEX

LCD_DATA0_PORT	uart1_puts, 26
LCD library <lcd.h>, 14</lcd.h>	uart1_puts_p, 26
LCD_DATA1_PORT	UART_BAUD_SELECT, 24
LCD library <lcd.h>, 14</lcd.h>	UART_BAUD_SELECT_DOUBLE_SPEED, 25
LCD DATA2 PORT	uart_getc, 26
LCD library <lcd.h>, 14</lcd.h>	uart_init, 27
LCD_DATA3_PORT	uart_putc, 27
LCD library <lcd.h>, 14</lcd.h>	uart_puts, 28
LCD_DELAY_BOOTUP	uart_puts_p, 28
LCD library <lcd.h>, 14</lcd.h>	UART_RX_BUFFER_SIZE, 25
LCD_DELAY_BUSY_FLAG	UART_TX_BUFFER_SIZE, 25
LCD library <lcd.h>, 14</lcd.h>	uart1_getc
LCD_DELAY_ENABLE_PULSE	UART Library <uart.h>, 25</uart.h>
LCD library <lcd.h>, 14</lcd.h>	uart1_init
LCD_DELAY_INIT	UART Library <uart.h>, 25</uart.h>
	•
LCD library <lcd.h>, 15</lcd.h>	uart1_putc
LCD_DELAY_INIT_4BIT	UART Library <uart.h>, 26</uart.h>
LCD library <lcd.h>, 15</lcd.h>	uart1_puts
LCD_DELAY_INIT_REP	UART Library $<$ uart.h $>$, 26
LCD library <lcd.h>, 15</lcd.h>	uart1_puts_p
LCD_DISP_LENGTH	UART Library <uart.h>, 26</uart.h>
LCD library <lcd.h>, 15</lcd.h>	UART_BAUD_SELECT
lcd_gotoxy	UART Library <uart.h>, 24</uart.h>
LCD library <lcd.h>, 18</lcd.h>	UART_BAUD_SELECT_DOUBLE_SPEED
•	
lcd_home	UART Library <uart.h>, 25</uart.h>
LCD library <lcd.h>, 18</lcd.h>	uart_getc
lcd_init	UART Library <uart.h>, 26</uart.h>
LCD library <lcd.h>, 18</lcd.h>	uart_init
LCD_IO_MODE	UART Library <uart.h>, 27</uart.h>
LCD library <lcd.h>, 15</lcd.h>	uart_putc
LCD_LINE_LENGTH	UART Library <uart.h>, 27</uart.h>
 LCD library <lcd.h>, 15</lcd.h>	uart_puts
LCD_LINES	UART Library <uart.h>, 28</uart.h>
LCD library <lcd.h>, 15</lcd.h>	uart_puts_p
lcd_putc	UART Library <uart.h>, 28</uart.h>
LCD library <lcd.h>, 19</lcd.h>	UART_RX_BUFFER_SIZE
lcd_puts	UART Library <uart.h>, 25</uart.h>
LCD library <lcd.h>, 19</lcd.h>	UART_TX_BUFFER_SIZE
lcd_puts_p	UART Library <uart.h>, 25</uart.h>
LCD library <lcd.h>, 19</lcd.h>	
LCD_RW_PIN	
LCD library <lcd.h>, 16</lcd.h>	
LCD_RW_PORT	
LCD library <lcd.h>, 16</lcd.h>	
LCD_START_LINE1	
LCD library <lcd.h>, 16</lcd.h>	
•	
LCD_START_LINE2	
LCD library <lcd.h>, 16</lcd.h>	
LCD_START_LINE3	
LCD library <lcd.h>, 16</lcd.h>	
LCD_START_LINE4	
LCD library <lcd.h>, 16</lcd.h>	
LCD_WRAP_LINES	
LCD library <lcd.h>, 16</lcd.h>	
UART Library <uart.h>, 23</uart.h>	
uart1_getc, 25	
uart1_init, 25	
uart1_nuto_06	

uart1_putc, 26