DAC

1.0

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 LCD library <lcd.h> . . . 7

4.1.1 Detailed Description . . . 10

4.1.2 Macro Definition Documentation . . . 10

4.1.2.1 LCD_CONTROLLER_KS0073 . . . 10

4.1.2.2 LCD_DATA0_PORT . . . 11

4.1.2.3 LCD_DATA1_PORT . . . 11

4.1.2.4 LCD_DATA2_PORT . . . 11

4.1.2.5 LCD_DATA3_PORT . . . 11

4.1.2.6 LCD_DELAY_BOOTUP . . . 11

4.1.2.7 LCD_DELAY_BUSY_FLAG . . . 12

4.1.2.8 LCD_DELAY_ENABLE_PULSE . . . 12

4.1.2.9 LCD_DELAY_INIT . . . 12

4.1.2.10 LCD_DELAY_INIT_4BIT . . . 12

4.1.2.11 LCD_DELAY_INIT_REP . . . 12

4.1.2.12 LCD_DISP_LENGTH . . . 13

4.1.2.13 LCD_IO_MODE . . . 13

4.1.2.14 LCD_LINE_LENGTH . . . 13

4.1.2.15 LCD_LINES . . . 13

4.1.2.16 LCD_RW_PIN . . . 13

4.1.2.17 LCD_RW_PORT . . . 14

4.1.2.18 LCD_START_LINE1 . . . 14

4.1.2.19 LCD_START_LINE2 . . . 14

4.1.2.20 LCD_START_LINE3 . . . 14

4.1.2.21 LCD_START_LINE4 . . . 14

4.1.2.22 LCD_WRAP_LINES . . . 15

4.1.3 Function Documentation . . . 15

4.1.3.1 lcd_clrscr() . . . 15

4.1.3.2 lcd_command() . . . 15

4.1.3.3 lcd_data() . . . 16

4.1.3.4 lcd_gotoxy() . . . 16

4.1.3.5 lcd_home() . . . 16

4.1.3.6 lcd_init() . . . 17

4.1.3.7 lcd_putc() . . . 17

4.1.3.8 lcd_puts() . . . 18

4.1.3.9 lcd_puts_p() . . . 18
4.2 Main file <main.c> . . . 19
4.2.1 Detailed Description . . . 19
4.3 DAC Library <Sig_gen.h> . . . 20
4.3.1 Detailed Description . . . 20
4.3.2 Function Documentation . . . 20
4.3.2.1 generate_signal() . . . 20
4.3.2.2 return_wvftype() . . . 21
4.3.2.3 send_uart() . . . 21
4.3.2.4 update_disp() . . . 22
4.4 UART Library <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() . . . 26
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() . . . 27
4.4.3.6 uart_getc() . . . 27
4.4.3.7 uart_init() . . . 27
4.4.3.8 uart_putc() . . . 28
4.4.3.9 uart_puts() . . . 28
4.4.3.10 uart_puts_p() . . . 29

5 File Documentation **31**
5.1 lcd.h File Reference . . . 31
5.2 main.c File Reference . . . 33
5.2.1 Macro Definition Documentation . . . 34
5.2.1.1 F_CPU . . . 34
5.3 Sig_gen.h File Reference . . . 34
5.4 timer.h File Reference . . . 34
5.4.1 Detailed Description . . . 35
5.4.2 Macro Definition Documentation . . . 35
5.4.2.1 TIM1_stop . . . 36
5.4.2.2 TIM2_stop . . . 36
5.5 uart.h File Reference . . . 36

Index **39**

Chapter 1

Main Page

DAC for project in Digital Electronics 2, using Peter Fleurys libraries for LCD and UART communication.

Author

Jiri Vitous

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

LCD library <lcd.h> 7
Main file <main.c> 19
DAC Library <Sig_gen.h> 20
UART Library <uart.h> 23

Chapter 3

File Index

3.1 File List

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

lcd.c . . . . . **??**
lcd.h 31
lcd_definitions.h . . . . . **??**
main.c 33
Sig_gen.c . . . . . **??**
Sig_gen.h 34
timer.h
Timer library for AVR-GCC 34
uart.c . . . . . **??**
uart.h 36

Chapter 4

Module Documentation

4.1 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** 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 will override the default definitions from lcd.h

- #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** 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 will override the default definitions from lcd.h

- #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** 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 will override the default definitions from lcd.h

- #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(), 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 lcd_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 lcd_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

4.1.1 Detailed Description

Basic routines for interfacing a HD44780U-based character LCD display.
`#include <lcd.h>`

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.) 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

4.1.2 Macro Definition Documentation

4.1.2.1 LCD_CONTROLLER_KS0073

```
#define LCD_CONTROLLER_KS0073 0
```

Use 0 for HD44780 controller, 1 for KS0073 controller

Definition at line 64 of file lcd.h.

4.1.2.2 LCD_DATA0_PORT

```
#define LCD_DATA0_PORT LCD_PORT
```

port for 4bit data bit 0

Definition at line 128 of file lcd.h.

4.1.2.3 LCD_DATA1_PORT

```
#define LCD_DATA1_PORT LCD_PORT
```

port for 4bit data bit 1

Definition at line 131 of file lcd.h.

4.1.2.4 LCD_DATA2_PORT

```
#define LCD_DATA2_PORT LCD_PORT
```

port for 4bit data bit 2

Definition at line 134 of file lcd.h.

4.1.2.5 LCD_DATA3_PORT

```
#define LCD_DATA3_PORT LCD_PORT
```

port for 4bit data bit 3

Definition at line 137 of file lcd.h.

4.1.2.6 LCD_DELAY_BOOTUP

```
#define LCD_DELAY_BOOTUP 16000
```

delay in micro seconds after power-on

Definition at line 198 of file lcd.h.

4.1.2.7 LCD_DELAY_BUSY_FLAG

```
#define LCD_DELAY_BUSY_FLAG 4
```

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

Definition at line 210 of file lcd.h.

4.1.2.8 LCD_DELAY_ENABLE_PULSE

```
#define LCD_DELAY_ENABLE_PULSE 1
```

enable signal pulse width in micro seconds

Definition at line 213 of file lcd.h.

4.1.2.9 LCD_DELAY_INIT

```
#define LCD_DELAY_INIT 5000
```

delay in micro seconds after initialization command sent

Definition at line 201 of file lcd.h.

4.1.2.10 LCD_DELAY_INIT_4BIT

```
#define LCD_DELAY_INIT_4BIT 64
```

delay in micro seconds after setting 4-bit mode

Definition at line 207 of file lcd.h.

4.1.2.11 LCD_DELAY_INIT_REP

```
#define LCD_DELAY_INIT_REP 64
```

delay in micro seconds after initialization command repeated

Definition at line 204 of file lcd.h.

4.1.2.12 LCD_DISP_LENGTH

```
#define LCD_DISP_LENGTH 16
```

visibles characters per line of the display

Definition at line 80 of file lcd.h.

4.1.2.13 LCD_IO_MODE

```
#define LCD_IO_MODE 1
```

0: memory mapped mode, 1: IO port mode

Definition at line 120 of file lcd.h.

4.1.2.14 LCD_LINE_LENGTH

```
#define LCD_LINE_LENGTH 0x40
```

internal line length of the display

Definition at line 83 of file lcd.h.

4.1.2.15 LCD_LINES

```
#define LCD_LINES 2
```

number of visible lines of the display

Definition at line 77 of file lcd.h.

4.1.2.16 LCD_RW_PIN

```
#define LCD_RW_PIN 5
```

pin for RW line

Definition at line 161 of file lcd.h.

4.1.2.17 LCD_RW_PORT

```
#define LCD_RW_PORT LCD_PORT
```

port for RW line

Definition at line 158 of file lcd.h.

4.1.2.18 LCD_START_LINE1

```
#define LCD_START_LINE1 0x00
```

DDRAM address of first char of line 1

Definition at line 86 of file lcd.h.

4.1.2.19 LCD_START_LINE2

```
#define LCD_START_LINE2 0x40
```

DDRAM address of first char of line 2

Definition at line 89 of file lcd.h.

4.1.2.20 LCD_START_LINE3

```
#define LCD_START_LINE3 0x14
```

DDRAM address of first char of line 3

Definition at line 92 of file lcd.h.

4.1.2.21 LCD_START_LINE4

```
#define LCD_START_LINE4 0x54
```

DDRAM address of first char of line 4

Definition at line 95 of file lcd.h.

4.1.2.22 LCD_WRAP_LINES

```
#define LCD_WRAP_LINES 0
```

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

Definition at line 98 of file lcd.h.

4.1.3 Function Documentation

4.1.3.1 lcd_clrscr()

```
void lcd_clrscr (
            void  )
```

Clear display and set cursor to home position.

Returns

none

Definition at line 407 of file lcd.c.

4.1.3.2 lcd_command()

```
void lcd_command (
            uint8_t cmd )
```

Send LCD controller instruction command.

Parameters

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

Returns

none

Definition at line 348 of file lcd.c.

4.1.3.3 lcd_data()

```
void lcd_data (
            uint8_t data )
```

Send data byte to LCD controller.

Similar to lcd_putc(), but without interpreting LF

Parameters

data	byte to send to LCD controller, see HD44780 data sheet

Returns

none

Definition at line 360 of file lcd.c.

4.1.3.4 lcd_gotoxy()

```
void lcd_gotoxy (
            uint8_t x,
            uint8_t y )
```

Set cursor to specified position.

Parameters

x	horizontal position (0: left most position)
y	vertical position (0: first line)

Returns

none

Definition at line 373 of file lcd.c.

4.1.3.5 lcd_home()

```
void lcd_home (
            void  )
```

Set cursor to home position.

Returns

none

Definition at line 415 of file lcd.c.

4.1.3.6 lcd_init()

```
void lcd_init (
             uint8_t dispAttr )
```

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_BLINK** display on, cursor on flashing

Returns

none

Definition at line 507 of file lcd.c.

4.1.3.7 lcd_putc()

```
void lcd_putc (
             char c )
```

Display character at current cursor position.

Parameters

c	character to be displayed

Returns

none

Definition at line 425 of file lcd.c.

4.1.3.8 lcd_puts()

```
void lcd_puts (
            const char * s )
```

Display string without auto linefeed.

Parameters

s	string to be displayed

Returns

none

Definition at line 472 of file lcd.c.

4.1.3.9 lcd_puts_p()

```
void lcd_puts_p (
            const char * progmem_s )
```

Display string from program memory without auto linefeed.

Parameters

progmem_s	string from program memory be be displayed

Returns

none

See also

lcd_puts_P

Definition at line 488 of file lcd.c.

4.2 Main file <main.c>

DAC signal generating, using AVR MCu and R2R DAC with output on 16x2 LCD and UART.

Functions

- void decode_button ()

  Decodes the selected button from butt variable.
- void change_size ()

  Changes the size of an array appropriatel to period, can switch timer ovf period if required (if the period does not fit within the interval, not used currently)

4.2.1 Detailed Description

DAC signal generating, using AVR MCu and R2R DAC with output on 16x2 LCD and UART.

This project uses simple R2R DAC to generate some usefull waveforms in frequency range from 100 Hz to 2kHz. It uses two main timers, first to sample the output signal and second to check pressed buttons and update signal parameters and display them onto LCD or send throught UART.

Note

Uses Peter Fleurys libraries for LCD and UART communication

Author

Jiri Vitous

4.3 DAC Library <Sig_gen.h>

Generate signal based on entered type frequency and other parameters.

Functions

- void generate_signal (uint8_t *frame_buffer, uint8_t type, uint16_t freq, float tim_set, uint8_t multiplier)

  Signal Buffer filling function.
- void update_disp (uint8_t type, uint16_t frequency, uint8_t multiplier)

  Display updating function.
- void send_uart (uint8_t type, uint16_t frequency)

  UART send current state function.
- void return_wvftype (char text[ ], uint8_t type)

  Return generated function name function.

4.3.1 Detailed Description

Generate signal based on entered type frequency and other parameters.
`#include <Sig_gen.h>`

This library generates signals from LUTs or using formulas. The available waveforms are: Sine, positive ramp, negative ramp, triangle, square, ECG or DTMF with parameter. The framebuffer is filled with corresponding data which can then be output to the output PORTX. The default output precision is of data type uint8_t

Note

Uses Peter Fleurys libraries for LCD and UART communication

Author

Jiri Vitous

4.3.2 Function Documentation

4.3.2.1 generate_signal()

```
void generate_signal (
            uint8_t * frame_buffer,
            uint8_t type,
            uint16_t freq,
            float tim_set,
            uint8_t multiplier )
```

Signal Buffer filling function.

Parameters

frame_buffer	Pointer to frame buffer vector
type	button pressed (later decoded by type_map)
freq	frequency of output signal in Hz should be in range 100-2000 Hz for default setting
tim_set	Timer set ovf period, to calculate active frame buffer size
multiplier	Extra parameter for DTMF describes the ratio of two generated sine waves

Returns

none

Definition at line 107 of file Sig_gen.c.

4.3.2.2 return_wvftype()

```
void return_wvftype (
            char text[],
            uint8_t type )
```

Return generated function name function.

Parameters

text	empty char array where to store the wvf. name
type	button pressed

Returns

none

Definition at line 255 of file Sig_gen.c.

4.3.2.3 send_uart()

```
void send_uart (
            uint8_t type,
            uint16_t frequency )
```

UART send current state function.

Parameters

type	Button pressed
frequency	Frequency of output signal in Hz should be in range 100-2000 Hz for default setting

Returns

none

Definition at line 240 of file Sig_gen.c.

4.3.2.4 update_disp()

```
void update_disp (
             uint8_t type,
             uint16_t frequency,
             uint8_t multiplier )
```

Display updating function.

Parameters

type	Button pressed
frequency	Frequency of output signal in Hz should be in range 100-2000 Hz for default setting
multiplier	Extra parameter for DTMF describes the ratio of two generated sine waves

Returns

none

Definition at line 309 of file Sig_gen.c.

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

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

UART Baudrate Expression.

Parameters

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

Definition at line 70 of file uart.h.

4.4.2.2 UART_BAUD_SELECT_DOUBLE_SPEED

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

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

Definition at line 76 of file uart.h.

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.

Definition at line 84 of file uart.h.

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↩FER_SIZE=nn to your Makefile.

Definition at line 93 of file uart.h.

4.4.3 Function Documentation

4.4.3.1 uart1_getc()

```
unsigned int uart1_getc (
            void  )
```

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

See also

uart_getc

4.4.3.2 uart1_init()

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

Initialize USART1 (only available on selected ATmegas)

See also

uart_init

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

4.4.3.4 uart1_puts()

```
void uart1_puts (
            const char * s )
```

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

See also

uart_puts

4.4.3.5 uart1_puts_p()

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

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

See also

uart_puts_p

4.4.3.6 uart_getc()

```
unsigned int uart_getc (
             void  )
```

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

Definition at line 494 of file uart.c.

4.4.3.7 uart_init()

```
void uart_init (
             unsigned int baudrate )
```

Initialize UART and set baudrate.

Parameters

baudrate	Specify baudrate using macro UART_BAUD_SELECT()

Returns

none

Definition at line 439 of file uart.c.

4.4.3.8 uart_putc()

```
void uart_putc (
            unsigned char data )
```

Put byte to ringbuffer for transmitting via UART.

Parameters

data	byte to be transmitted

Returns

none

Definition at line 526 of file uart.c.

4.4.3.9 uart_puts()

```
void uart_puts (
            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

Definition at line 551 of file uart.c.

4.4.3.10 uart_puts_p()

```
void uart_puts_p (
             const char * s )
```

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

Definition at line 563 of file uart.c.

Chapter 5

File Documentation

5.1 lcd.h File Reference

```
#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 **lcd_definitions.h** 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 will override the default definitions from lcd.h*

- #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** 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 will override the default definitions from lcd.h*

- #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** 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 will override the default definitions from lcd.h*

- #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(), 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

- #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 lcd_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 lcd_data (uint8_t data)

  Send data byte to LCD controller.

5.2 main.c File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "timer.h"
#include "Sig_gen.h"
#include "lcd.h"
#include <stdlib.h>
#include <string.h>
#include "uart.h"
```

Macros

- #define F_CPU 16000000

  F_CPU.

Functions

- void decode_button ()

  Decodes the selected button from butt variable.
- void change_size ()

  Changes the size of an array appropriatel to period, can switch timer ovf period if required (if the period does not fit within the interval, not used currently)
- int **main** (void)
- ISR (TIMER0_OVF_vect)

  Frame buffer update timer (main sampling clock generator) moves in frame buffer.
- **ISR** (TIMER2_OVF_vect)

Variables

- uint8_t **frame_buffer** [1000]
- volatile uint16_t **frequency** = 161
- volatile uint8_t **multiplier** =1
- uint16_t **butt** = 0
- uint16_t **arr_size** = 0
- float **tim_set** = 16e-6
- const float **timer_values** [ ] = { 16e-6,128e-6,1e-3 }

5.2.1 Macro Definition Documentation

5.2.1.1 F_CPU

```
#define F_CPU 16000000
```

F_CPU.

Warning

Should Correspod with CPU frequency

Definition at line 32 of file main.c.

5.3 Sig_gen.h File Reference

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

Functions

- void generate_signal (uint8_t *frame_buffer, uint8_t type, uint16_t freq, float tim_set, uint8_t multiplier)
  Signal Buffer filling function.
- void update_disp (uint8_t type, uint16_t frequency, uint8_t multiplier)
  Display updating function.
- void send_uart (uint8_t type, uint16_t frequency)
  UART send current state function.
- void return_wvftype (char text[ ], uint8_t type)
  Return generated function name function.

5.4 timer.h File Reference

Timer library for AVR-GCC.

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

Macros

- #define **TIM0_stop**() TCCR0B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));
- #define **TIM0_overflow_16u**() TCCR0B &= ~((1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00);
- #define **TIM0_overflow_128u**() TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS01);
- #define **TIM0_overflow_1m**() TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) | (1<<CS00);
- #define **TIM0_overflow_4m**() TCCR0B &= ~((1<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS02);
- #define **TIM0_overflow_16m**() TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00);
- #define TIM1_stop() TCCR1B &= ~((1<<CS12) | (1<<CS11) | (1<<CS10));

  Defines prescaler CPU frequency values for Timer/Counter1.
- #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 TIM2_stop() TCCR2B &= ~((1<<CS22) | (1<<CS21) | (1<<CS20));

  Defines prescaler CPU frequency values for Timer/Counter2.
- #define **TIM2_overflow_16u**() TCCR2B &= ~((1<<CS22) | (1<<CS21)); TCCR2B |= (1<<CS20);
- #define **TIM2_overflow_128u**() TCCR2B &= ~((1<<CS22) | (1<<CS20)); TCCR2B |= (1<<CS21);
- #define **TIM2_overflow_512u**() TCCR2B &= ~(1<<CS22); TCCR2B |= (1<<CS21) | (1<<CS20);
- #define **TIM2_overflow_1m**() TCCR2B &= ~((1<<CS21) | (1<<CS20)); TCCR2B |= (1<<CS22);
- #define **TIM2_overflow_2m**() TCCR2B &= ~(1<<CS21); TCCR2B |= (1<<CS22) | (1<<CS20);
- #define **TIM2_overflow_4m**() TCCR2B &= ~(1<<CS20); TCCR2B |= (1<<CS22) | (1<<CS21);
- #define **TIM2_overflow_16m**() TCCR2B |= (1<<CS21) | (1<<CS20) | (1<<CS22);
- #define TIM1_overflow_interrupt_enable() TIMSK1 |= (1<<TOIE1);

  Defines interrupt enable/disable modes for Timer/Counter1.
- #define **TIM1_overflow_interrupt_disable**() TIMSK1 &= ~(1<<TOIE1);
- #define **TIM0_overflow_interrupt_enable**() TIMSK0 |= (1<<TOIE0);
- #define **TIM0_overflow_interrupt_disable**() TIMSK0 &= ~(1<<TOIE0);
- #define **TIM2_overflow_interrupt_enable**() TIMSK2 |= (1<<TOIE2);
- #define **TIM2_overflow_interrupt_disable**() TIMSK2 &= ~(1<<TOIE2);

5.4.1 Detailed Description

Timer library for AVR-GCC.

The library contains macros for controlling the timer modules.

Note

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

Copyright

5.4.2 Macro Definition Documentation

5.4.2.1 TIM1_stop

```
#define TIM1_stop( ) TCCR1B &= ~((1<<CS12) | (1<<CS11) | (1<<CS10));
```

Defines prescaler CPU frequency values for Timer/Counter1.

Note

F_CPU = 16 MHz

Definition at line 49 of file timer.h.

5.4.2.2 TIM2_stop

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

Defines prescaler CPU frequency values for Timer/Counter2.

Note

F_CPU = 16 MHz

Definition at line 61 of file timer.h.

5.5 uart.h File Reference

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

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)

Index

DAC Library <Sig_gen.h>, 20
    generate_signal, 20
    return_wvftype, 21
    send_uart, 21
    update_disp, 22

F_CPU
    main.c, 34

generate_signal
    DAC Library <Sig_gen.h>, 20

LCD library <lcd.h>, 7
    lcd_clrscr, 15
    lcd_command, 15
    LCD_CONTROLLER_KS0073, 10
    lcd_data, 15
    LCD_DATA0_PORT, 10
    LCD_DATA1_PORT, 11
    LCD_DATA2_PORT, 11
    LCD_DATA3_PORT, 11
    LCD_DELAY_BOOTUP, 11
    LCD_DELAY_BUSY_FLAG, 11
    LCD_DELAY_ENABLE_PULSE, 12
    LCD_DELAY_INIT, 12
    LCD_DELAY_INIT_4BIT, 12
    LCD_DELAY_INIT_REP, 12
    LCD_DISP_LENGTH, 12
    lcd_gotoxy, 16
    lcd_home, 16
    lcd_init, 17
    LCD_IO_MODE, 13
    LCD_LINE_LENGTH, 13
    LCD_LINES, 13
    lcd_putc, 17
    lcd_puts, 17
    lcd_puts_p, 18
    LCD_RW_PIN, 13
    LCD_RW_PORT, 13
    LCD_START_LINE1, 14
    LCD_START_LINE2, 14
    LCD_START_LINE3, 14
    LCD_START_LINE4, 14
    LCD_WRAP_LINES, 14
lcd.h, 31
lcd_clrscr
    LCD library <lcd.h>, 15
lcd_command
    LCD library <lcd.h>, 15
LCD_CONTROLLER_KS0073
    LCD library <lcd.h>, 10
lcd_data
    LCD library <lcd.h>, 15
LCD_DATA0_PORT
    LCD library <lcd.h>, 10
LCD_DATA1_PORT
    LCD library <lcd.h>, 11
LCD_DATA2_PORT
    LCD library <lcd.h>, 11
LCD_DATA3_PORT
    LCD library <lcd.h>, 11
LCD_DELAY_BOOTUP
    LCD library <lcd.h>, 11
LCD_DELAY_BUSY_FLAG
    LCD library <lcd.h>, 11
LCD_DELAY_ENABLE_PULSE
    LCD library <lcd.h>, 12
LCD_DELAY_INIT
    LCD library <lcd.h>, 12
LCD_DELAY_INIT_4BIT
    LCD library <lcd.h>, 12
LCD_DELAY_INIT_REP
    LCD library <lcd.h>, 12
LCD_DISP_LENGTH
    LCD library <lcd.h>, 12
lcd_gotoxy
    LCD library <lcd.h>, 16
lcd_home
    LCD library <lcd.h>, 16
lcd_init
    LCD library <lcd.h>, 17
LCD_IO_MODE
    LCD library <lcd.h>, 13
LCD_LINE_LENGTH
    LCD library <lcd.h>, 13
LCD_LINES
    LCD library <lcd.h>, 13
lcd_putc
    LCD library <lcd.h>, 17
lcd_puts
    LCD library <lcd.h>, 17
lcd_puts_p
    LCD library <lcd.h>, 18
LCD_RW_PIN
    LCD library <lcd.h>, 13
LCD_RW_PORT
    LCD library <lcd.h>, 13
LCD_START_LINE1
    LCD library <lcd.h>, 14

LCD_START_LINE2
LCD library <lcd.h>, 14
LCD_START_LINE3
LCD library <lcd.h>, 14
LCD_START_LINE4
LCD library <lcd.h>, 14
LCD_WRAP_LINES
LCD library <lcd.h>, 14

Main file <main.c>, 19
main.c, 33
F_CPU, 34

return_wvftype
DAC Library <Sig_gen.h>, 21

send_uart
DAC Library <Sig_gen.h>, 21
Sig_gen.h, 34

TIM1_stop
timer.h, 35
TIM2_stop
timer.h, 36
timer.h, 34
TIM1_stop, 35
TIM2_stop, 36

UART Library <uart.h>, 23
uart1_getc, 25
uart1_init, 26
uart1_putc, 26
uart1_puts, 26
uart1_puts_p, 26
UART_BAUD_SELECT, 24
UART_BAUD_SELECT_DOUBLE_SPEED, 25
uart_getc, 27
uart_init, 27
uart_putc, 28
uart_puts, 28
uart_puts_p, 29
UART_RX_BUFFER_SIZE, 25
UART_TX_BUFFER_SIZE, 25
uart.h, 36
uart1_getc
UART Library <uart.h>, 25
uart1_init
UART Library <uart.h>, 26
uart1_putc
UART Library <uart.h>, 26
uart1_puts
UART Library <uart.h>, 26
uart1_puts_p
UART Library <uart.h>, 26
UART_BAUD_SELECT
UART Library <uart.h>, 24
UART_BAUD_SELECT_DOUBLE_SPEED
UART Library <uart.h>, 25
uart_getc
UART Library <uart.h>, 27
uart_init
UART Library <uart.h>, 27
uart_putc
UART Library <uart.h>, 28
uart_puts
UART Library <uart.h>, 28
uart_puts_p
UART Library <uart.h>, 29
UART_RX_BUFFER_SIZE
UART Library <uart.h>, 25
UART_TX_BUFFER_SIZE
UART Library <uart.h>, 25
update_disp
DAC Library <Sig_gen.h>, 22