ARM-Timing

Generated by Doxygen 1.8.17

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 C:/Users/Paul/Paul/Schule/2019-2020/4BHEL/DIC1/Projekte/ARM-Timing/src/LCD.h File Reference

```
#include <stm32f10x.h>
#include <stdarg.h>
```

Enumerations

- enum {
  LCD_RST = 0x1 << 11, **LCD_CS** = 0x1 << 12, **LCD_SCLK** = 0x1 << 13, **LCD_SOD** = 0x1 << 14,
  LCD_SID = 0x1 << 15 }

  Konstantendefinitionen fuer Portleitungen.
- enum : uint8_t {
  LCD_start = 0x1F, **LCD_read** = 0x1 << 5, **LCD_write** = 0x0 << 5, **LCD_RS_1** = 0x1 << 6,
  LCD_RS_0 = 0x0 << 6, **LCD_instruction_write** = LCD_RS_0 | LCD_write, **LCD_read_busy_flag_and_address_counter** = LCD_RS_0 | LCD_read, **LCD_data_write** = LCD_RS_1 | LCD_write,
  LCD_data_read = LCD_RS_1 | LCD_read }

  Konstantendefinitionen fuer Startbytes.
- enum : uint8_t {
  LCD_Clear_display = 0x1 << 0, **LCD_Return_home** = 0x1 << 1, **LCD_Power_down_mode** = 0x1 << 1, **LCD_power_down_mode_set** = 0x1 << 0,
  LCD_power_down_mode_disable = 0x0 << 0, **LCD_Entry_mode_set** = 0x1 << 2, **LCD_cursor_blink_moves_to_right** = 0x1 << 1, **LCD_cursor_blink_moves_to_left** = 0x0 << 1,
  LCD_Display_OnOff_control = 0x1 << 3, **LCD_display_on** = 0x1 << 2, **LCD_display_off** = 0x0 << 2, **LCD_cursor_on** = 0x1 << 1,
  LCD_cursor_off = 0x0 << 1, **LCD_blink_on** = 0x1 << 0, **LCD_blink_off** = 0x0 << 0, **LCD_Extended_function_set** = 0x1 << 3,
  LCD_6dot_font_width = 0x1 << 2, **LCD_5dot_font_width** = 0x0 << 2, **LCD_black_white_inverting_of_cursor_enable** = 0x1 << 1, **LCD_black_white_inverting_of_cursor_disable** = 0x0 << 1,
  LCD_3line_or_4line_display = 0x1 << 0, **LCD_1line_or_2line_display** = 0x0 << 0, **LCD_Cursor_or_display_shift** = 0x1 << 4, **LCD_display_shift** = 0x1 << 3,
  LCD_cursor_shift = 0x0 << 3, **LCD_shift_to_right** = 0x1 << 2, **LCD_shift_to_left** = 0x0 << 2, **LCD_Double_height_Bias_Displaydot_shift** = 0x1 << 4,
  LCD_3rd_line = 0x0 << 2, **LCD_2nd_line** = 0x1 << 2, **LCD_1st_line** = 0x2 << 2, **LCD_both_lines** = 0x3

$<<$ 2,
LCD_Internal_OSC_frequency = 0x1 $<<$ 4, **LCD_BS0_1** = 0x1 $<<$ 3, **LCD_BS0_0** = 0x0 $<<$ 3, **LCD_↩Shift_enable** = 0x1 $<<$ 4,
LCD_1st_line_display_shift_enable = 0x1 $<<$ 0, **LCD_1st_line_display_shift_disable** = 0x0 $<<$ 0, **L↩CD_2nd_line_display_shift_enable** = 0x1 $<<$ 1, **LCD_2nd_line_display_shift_disable** = 0x0 $<<$ 1,
LCD_3rd_line_display_shift_enable = 0x1 $<<$ 2, **LCD_3rd_line_display_shift_disable** = 0x0 $<<$ 2, **L↩CD_4th_line_display_shift_enable** = 0x1 $<<$ 3, **LCD_4th_line_display_shift_disable** = 0x0 $<<$ 3,
LCD_Scroll_enable = 0x1 $<<$ 4, **LCD_1st_line_dot_scroll_enable** = 0x1 $<<$ 0, **LCD_1st_line_dot_↩scroll_disable** = 0x0 $<<$ 0, **LCD_2nd_line_dot_scroll_enable** = 0x1 $<<$ 1,
LCD_2nd_line_dot_scroll_disable = 0x0 $<<$ 1, **LCD_3rd_line_dot_scroll_enable** = 0x1 $<<$ 2, **LCD_↩3rd_line_dot_scroll_disable** = 0x0 $<<$ 2, **LCD_4th_line_dot_scroll_enable** = 0x1 $<<$ 3,
LCD_4th_line_dot_scroll_disable = 0x0 $<<$ 3, **LCD_Function_set** = 0x1 $<<$ 5, **LCD_8bit** = 0x1 $<<$ 4, **LCD_4bit** = 0x0 $<<$ 4,
LCD_2line_or_4line_display = 0x1 $<<$ 3, **LCD_1line_or_3line_display** = 0x0 $<<$ 3, **LCD_double↩_height_font_control_for_2line_mode_enable** = 0x1 $<<$ 2, **LCD_double_height_font_control_for_↩2line_mode_disable** = 0x1 $<<$ 2,
LCD_RE_0 = 0x0 $<<$ 1, **LCD_instruction_set_1** = 0x1 $<<$ 0, **LCD_instruction_set_0** = 0x0 $<<$ 0, **LC↩D_CGRAM_SERAM_blink_enable** = 0x1 $<<$ 2,
LCD_CGRAM_SERAM_blink_disable = 0x0 $<<$ 2, **LCD_RE_1** = 0x1 $<<$ 1, **LCD_reverse_display** = 0x1 $<<$ 0, **LCD_normal_display** = 0x0 $<<$ 0,
LCD_set_CGRAM_address = 0x1 $<<$ 6, **LCD_set_SEGRAM_address** = 0x1 $<<$ 6, **LCD_Power_Icon↩_control_Contrast_set** = 0x5 $<<$ 4, **LCD_ICON_display_on** = 0x1 $<<$ 3,
LCD_ICON_display_off = 0x0 $<<$ 3, **LCD_set_booster_and_regulator_circuit_on** = 0x1 $<<$ 2, **LCD_↩set_booster_and_regulator_circuit_off** = 0x0 $<<$ 2, **LCD_Follower_control** = 0x3 $<<$ 5,
LCD_set_divider_circuit_on = 0x1 $<<$ 3, **LCD_set_divider_circuit_off** = 0x0 $<<$ 3, **LCD_Contrast_set** = 0x7 $<<$ 4, **LCD_set_DDRAM_address** = 0x1 $<<$ 7,
LCD_set_scroll_quantity = 0x1 $<<$ 7, **LCD_view** = 0x1 $<<$ 2, **LCD_bottom** = 0x1 $<<$ 1, **LCD_top** = 0x1 $<<$ 0 }

Konstantendefinitionen fuer Befehle.

Functions

- void **wait_ms** (int t_ms)

  Wartet fuer t_ms ms.
- void **LCD_init_ports** ()

  Initialisiert Portleitungen.
- void **LCD_reset** ()

  Reset LCD-Display.
- void **LCD_init** ()

  Initialisiert LCD-Display.
- void **LCD_start_byte** (uint8_t start_byte)

  Sendet ein Startbyte zur Anzeige.
- void **LCD_write_byte** (uint8_t data)

  Sendet ein Byte zur Anzeige.
- void **LCD_write_data_byte** (uint8_t cmd)

  Sendet ein Datenbyte zur Anzeige.
- void **LCD_write_data_bytes_0term** (uint8_t cmd[ ])

  Ruft die Funktion LCD_write_data_byte fuer jedes Element im Nullterminierten Array cmd auf.
- void **LCD_write_data_bytes_count** (uint8_t cmd[ ], int count)

  Ruft die Funktion LCD_write_data_byte fuer die ersten count Elemente im Array cmd auf.
- void **LCD_init_normal** ()
- void **LCD_putchar** (int character)

  Gibt ein Zeichen auf der Anzeige aus.
- void **LCD_puts** (char const ∗str)

Gibt eine Zeichenkette auf der Anzeige aus.

- void **LCD_printf** (char const *format,...)

  Siehe `https://www.cplusplus.com/reference/cstdio/printf/.`
- uint8_t **LCD_set_cursor_position** (uint8_t x, uint8_t y)

  Berechnet einen Befehl um den Cursor an die Position (x, y) zu setzen.

2.1.1 Function Documentation

2.1.1.1 LCD_init()

```
void LCD_init ( )
```

Initialisiert LCD-Display.

Die Funktion LCD_init initialisiert die Portleitungen, fuehrt einen Reset aus und initialisiert die restlichen Portleitungen mit einem Defaultwert.

2.1.1.2 LCD_init_ports()

```
void LCD_init_ports ( )
```

Initialisiert Portleitungen.

Die Funktion LCD_init_ports initialisiert die fuenf Portleitungen die zum kommunizieren mit der LCD-Anzeige notwendeg sind. (RST, CS, SCLK, SOD, SID)

2.1.1.3 LCD_printf()

```
void LCD_printf (
            char const * format,
             ...  )
```

Siehe `https://www.cplusplus.com/reference/cstdio/printf/.`

Parameters

in	*format*	Formatstring
in	*...*	weitere Argumente

2.1.1.4 LCD_putchar()

```
void LCD_putchar (
            int character )
```

Gibt ein Zeichen auf der Anzeige aus.

Die Funktion LCD_putchar gibt ein Zeichen auf der LCD-Anzeige aus. ACHTUNG: FUNKTIONIERT NUR WENN VORHER DER BEFEHL LCD_start_byte(LCD_start | LCD_data_write) AUSGWFUEHRT WURDE!

Parameters

in	*character*	Zeichen das ausgegeben werden soll

2.1.1.5 LCD_puts()

```
void LCD_puts (
            char const * str )
```

Gibt eine Zeichenkette auf der Anzeige aus.

Die Funktion LCD_puts gibt alle Zeichen der nullterminierten Zecihenkette str auf der LCD-Anzeige aus. ACHT↩ UNG: FUNKTIONIERT NUR WENN VORHER DER BEFEHL LCD_start_byte(LCD_start | LCD_data_write) AU↩ SGWFUEHRT WURDE!

Parameters

in	*str*	Zeichenkette die ausgegeben werden soll

2.1.1.6 LCD_reset()

```
void LCD_reset ( )
```

Reset LCD-Display.

Fuehrt einen Reset der LCD-Anzeige aus. (Reset-Leitung fuer 1ms auf low setzen)

2.1.1.7 LCD_set_cursor_position()

```
uint8_t LCD_set_cursor_position (
            uint8_t x,
            uint8_t y )
```

Berechnet einen Befehl um den Cursor an die Position (x, y) zu setzen.

Returns

Befehl um den Cursor an die Position (x, y) zu setzen

2.1.1.8 LCD_start_byte()

```
void LCD_start_byte (
            uint8_t start_byte )
```

Sendet ein Startbyte zur Anzeige.

Die Funktion LCD_start_byte unterbricht normalen Datenverkehr mit einer Regelverletzung (SID wird gesetzt wenn Clock = 1) und sendet dann ein Byte mit der Funktion LCD_write_byte.

Parameters

in	*start_byte*	Byte dass nach der Regelverletzung uebertragen wird. Dies ist folgendermassen aufgebaut: 0 RS R/W 1 1 1 1 1

2.1.1.9 LCD_write_byte()

```
void LCD_write_byte (
            uint8_t data )
```

Sendet ein Byte zur Anzeige.

Die Funktion LCD_write_byte sendet ein Byte zur LCD-Anzeige (LSB zuerst).

Parameters

in	*data*	Byte dass uebertragen wird

2.1.1.10 LCD_write_data_byte()

```
void LCD_write_data_byte (
            uint8_t cmd )
```

Sendet ein Datenbyte zur Anzeige.

Die Funktion LCD_write_data_byte bringt zurest das Das Datenbyte in die gewuenschte Form (D7 D6 D5 D4 D3 D2 D1 D0 => 0 0 0 0 D3 D2 D1 D0 0 0 0 0 D7 D6 D5 D4) und sendet die zwei entstandenen Bytes mithilfe der Funktion LCD_write_byte an die Anzeige.

Parameters

in	*cmd*	Datenbyte das uebertragen wird

2.1.1.11 LCD_write_data_bytes_0term()

```
void LCD_write_data_bytes_0term (
            uint8_t cmd[] )
```

Ruft die Funktion LCD_write_data_byte fuer jedes Element im Nullterminierten Array cmd auf.

Die Funktion LCD_write_data_byte_0term ruft die Funktion LCD_write_data_byte fuer jedes Element im Nullterminierten Array cmd auf.

Parameters

in	*cmd*	Datenbytes die uebertragen werden

2.1.1.12 LCD_write_data_bytes_count()

```
void LCD_write_data_bytes_count (
            uint8_t cmd[],
            int count )
```

Ruft die Funktion LCD_write_data_byte fuer die ersten count Elemente im Array cmd auf.

Die Funktion LCD_write_data_byte_count ruft die Funktion LCD_write_data_byte fuer die ersten count Elemente im Array cmd auf.

Parameters

in	*cmd*	Datenbytes die uebertragen werden
in	*count*	Anzahl der Elemente die uebertagen werden

2.1.1.13 wait_ms()

```
void wait_ms (
            int t_ms )
```

Wartet fuer t_ms ms.

Die Funktion wait_ms haelt das Programm fuer t_ms ms an.