
μ OS LCD

Instrukcja Obsługi

Viresco μ CS / Micro Control Systems

20 listopada 2012

Spis treści

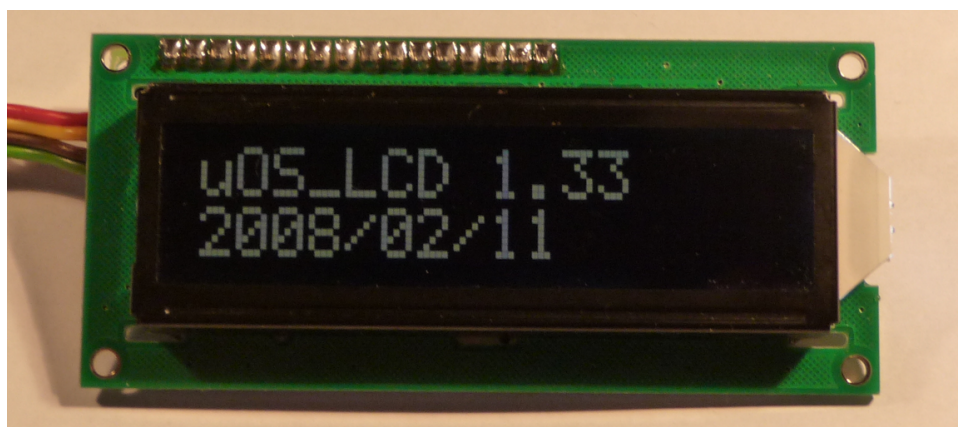
1	Ogólna charakterystyka	3
2	Interfejsy komunikacyjne	3
2.1	Buforowanie	4
2.2	UART	4
2.3	I ² C	4
2.4	SPI	4
3	Instalacja	4
3.1	Podłączenie	4
3.2	Integracja z istniejącym oprogramowaniem	4
4	Dokumentacja pliku uOS_lcd_host.c	7
4.1	Opis szczegółowy	9
4.2	Historia zmian	9
4.3	Dokumentacja funkcji	9
4.3.1	uOS_lcd_host_init	9
4.3.2	uOS_lcd_send_str	10
4.3.3	uOS_lcd_send_raw_command	10
4.3.4	uOS_lcd_print	10
4.3.5	uOS_lcd_goto	10
4.3.6	uOS_lcd_terminal	11
4.3.7	uOS_lcd_write_at	11
4.3.8	uOS_lcd_set_contrast	11
4.3.9	uOS_lcd_set_backlight	11
4.3.10	uOS_lcd_cls	12

4.3.11	uOS_lcd_define_char	12
4.3.12	uOS_lcd_set_char	12
4.3.13	uOS_lcd_disp_user_char	12
4.3.14	uOS_lcd_save_defaults	13
4.3.15	uOS_lcd_load_defaults	13
4.3.16	uOS_lcd_reset	13
4.3.17	uOS_lcd_cursor_off	14
4.3.18	uOS_lcd_cursor_block	14
4.3.19	uOS_lcd_cursor_underscore	14
4.3.20	uOS_lcd_scroll_left	14
4.3.21	uOS_lcd_normal	14
4.3.22	uOS_lcd_scroll_right	15
4.3.23	uOS_lcd_roll_left	15
4.3.24	uOS_lcd_roll_right	15
4.3.25	uOS_lcd_disp_user_char_ex	15
4.3.26	uOS_lcd_clear_row	16
4.3.27	uOS_lcd_sleep	16
4.3.28	uOS_lcd_enter_power_mode	16

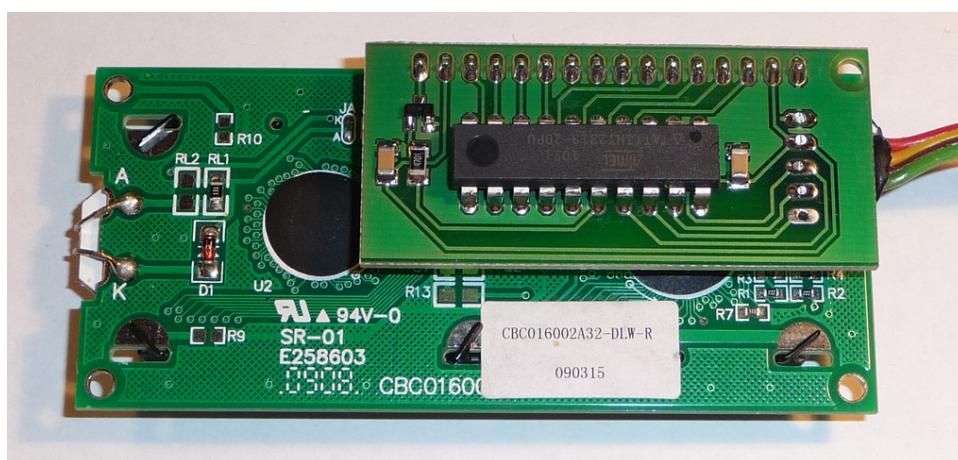
1 Ogólna charakterystyka

μ OS LCD to moduł sterownika alfanumerycznego wyświetlacza LCD. Jego przeznaczenie to zwolnienie konstruktorów z potrzeby oprogramowania wyświetlacza, konieczności projektowania skomplikowanego obwodu nawet dla prostego projektu oraz oszczędność linii sterujących.

Rysunek 1: Uruchomiony μ OS LCD



Rysunek 2: Widok sterownika



Podstawowe cechy:

1. Miniaturowa płytką drukowaną montowana bezpośrednio na wyświetlaczu;
2. Prostota podłączenia: w minimalnej wersji tylko 3 przewody;
3. Obsługiwane wszystkie rozmiary i typy standardowych wyświetlaczy alfanumerycznych (w tym nowe wyświetlacze OLED).
4. Gotowa biblioteka programowa obsługi wyświetlacza.

2 Interfejsy komunikacyjne

Urządzenie dostępne jest w trzech wykonaniach komunikacyjnych. We wszystkich wariantach komunikacja odbywa się za pomocą logiki 5 V (dotyczy również podłączenia UART).

We wszystkich wariantach prędkość transmisji danych do wyświetlacza jest ograniczona przez szybkość działania samego modułu LCD. Użytkownik musi dostosować prędkość transmisji do konkretnego typu wyświetlacza. Odbywa się to poprzez dodanie przerw pomiędzy przesyłanymi bajtami (UART, I²C, SPI) lub poprzez obniżenie prędkości bitowej (I²C, SPI).

! → Większość modułów LCD nie wymaga ograniczania prędkości transmisji poniżej 19200 kbps.

2.1 Buforowanie

Jak wspomniano powyżej, prędkość transmisji do urządzenia nie może przekraczać możliwości modułu LCD. Dotyczy to jednak prędkości średnich. Chwilowe prędkości mogą być znacznie wyższe.

Moduł wyposażony jest w bufor wejściowy umożliwiający chwilowy przyrost prędkości transmisji. Rozmiar bufora pozwala na zapisanie całej zawartości typowego wyświetlacza 2x16 z prędkością 1 Mbps¹.

2.2 UART

Umożliwia podłączenie minimalną liczbą przewodów - 3. Prędkość transmisji to 19200 kbps. Wadą tego rozwiązania jest potrzeba zapewnienia dużej stałości parametrów czasowych sygnału. Przy pracy w warunkach laboratoryjnych nie ma to znaczenia. Nie zalecany do pracy w warunkach przemysłowych (zwłaszcza przy dużej zmienności temperatur).

2.3 I²C

Standardowy protokół komunikacyjny wykorzystywany wewnątrz urządzeń. Domyślnie adres urządzenia jest ustawiony na 15 (0xF).

2.4 SPI

Najprostszy a jednocześnie najpewniejszy sposób komunikacji. Nie ma dolnego ograniczenia prędkości. Urządzenie dokonuje automatycznej synchronizacji więc zakłócenia lub nawet chwilowe odłączenie przewodów sygnałowych nie powoduje przerwania pracy. Rekomendowany do używania w trudnych warunkach (również przy dużej długości kabli).

3 Instalacja

3.1 Podłączenie

Tablica 1: Wyprowadzenia złącza

Konfiguracja	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5
UART	GND	VCC	RXD	N.C.	N.C.
I ² C			SCL	SDA	N.C.
SPI			SCL	DI	N.C.

3.2 Integracja z istniejącym oprogramowaniem

Biblioteka obsługi μ OS LCD jest dostarczona w postaci kodu źródłowego w języku C: `uOS_lcd_host.c`. Odpowiadający mu plik nagłówkowy to `uOS_lcd_host.h`. Dodatkowo załączony jest również plik nagłówkowy `bin_defs.h` ułatwiający definiowanie

¹ Niedostępne w wersji UART

znaków użytkownika (zawiera definicje wartości binarnych, w niektórych językach dostępnych przez przedrostek 0b).

Biblioteka jest napisana zgodnie ze standardem ANSI C i powinna się kompilować na dowolnym kompilatorze C. Oprócz tych trzech plików użytkownik musi jedynie zdefiniować funkcję wysyłającą jeden bajt do wyświetlacza. Treść tej funkcji zależy od rodzaju interfejsu komunikacyjnego oraz platformy sprzętowej. Sygnatura funkcji jest następująca:

```
1     typedef void (*uOS_lcd_send_byte_type)(char c);
```

Przykładowa funkcja wysyłania bajtu może wyglądać tak jak na listingu 1.

Listing 1: Przykład funkcji wysyłającej bajt

```
1     /* ***** */
2     * @param[in]    c - byte to be transmitted.
3     * @param[out]  -
4     * @param[in, out] -
5     *
6     * @return      N/A.
7     *
8     * @brief       Transmit byte thru SPI.
9     *
10    * @note        The function uses fixed baudrate equal to 25 kbps.
11    ***** */
12    void spi_soft_transmit_byte(char c)
13    {
14        unsigned char idx;
15
16        SCK_LO();
17
18        for (idx = 0; idx < 8; idx++)
19        {
20            if ((c & (1 << 7)) != 0) DATA_HI();
21            else                      DATA_LO();
22
23            delay_us(_20us);
24
25            SCK_HI();
26
27            delay_us(_20us);
28
29            SCK_LO();
30
31            c <<= 1;
32        }
33    }
```

Po dodaniu takiej funkcji należy jeszcze zdefiniować globalnie symbole odpowiadające za konfigurację biblioteki:

- LCD_CHAR_PER_ROW (domyślnie ustawiony na 16)
- LCD_NUM_ROWS (domyślnie ustawiony na 2)

Aby to zrobić można po prostu zmienić wartości w pliku `uOS_lcd_host.h` lub użyć opcji kompilatora (poniżej przykład dla GCC):

```
1     gcc -c -DLCD_CHAR_PER_ROW=20 -DLCD_NUM_ROWS=4 uOS_lcd_host.c
```

Po tych przygotowaniach pozostaje tylko wywołanie funkcji instalującej procedurę wysyłania bajtu (wykorzystujemy przykładową funkcję z listingu 1):

```
1     uOS_lcd_host_init(spi_soft_transmit_byte);
```

Po wykonaniu tej operacji możemy zacząć korzystać z wyświetlacza.

Dokumentacja wszystkich funkcji z biblioteki μ OS LCD znajduje się w kolejnym rozdziale.

4 Dokumentacja pliku uOS_lcd_host.c

Provide host routines for uOS_LCD driver.

Funkcje

- void `uOS_lcd_host_init` (uOS_lcd_send_byte_type send)
Set user defined/default LCD routine.
- void `uOS_lcd_send_str` (const char *str)
Send NULL - terminated control string to LCD.
- void `uOS_lcd_send_raw_command` (unsigned char command)
Send control command to LCD.
- void `uOS_lcd_print` (char *str)
Print NULL - terminated string onto LCD.
- void `uOS_lcd_goto` (unsigned char row, unsigned char col)
Move cursor to specified position.
- void `uOS_lcd_terminal` (char *newline)
Add a new line to a terminal emulated on LCD.
- void `uOS_lcd_write_at` (char *str, unsigned char row, unsigned char col)
Print NULL - terminated string at specified position.
- void `uOS_lcd_set_contrast` (unsigned char contrast)
Set LCD contrast.
- void `uOS_lcd_set_backlight` (unsigned char backlight)
Set LCD backlight intensity.
- void `uOS_lcd_cls` (void)
Clear the LCD.
- void `uOS_lcd_define_char` (unsigned char row0, unsigned char row1, unsigned char row2, unsigned char row3, unsigned char row4, unsigned char row5, unsigned char row6, unsigned char row7, unsigned char number)
Configure user - defined LCD character.
- void `uOS_lcd_set_char` (unsigned char *buf, unsigned char number)
Configure user - defined character using specified bitmap buffer.
- void `uOS_lcd_disp_user_char` (unsigned char number)
Display one of the user - defined characters.

- void `uOS_lcd_save_defaults` (void)
Save settings (contrast and backlight at a time) to non - volatile memory.
- void `uOS_lcd_load_defaults` (void)
Load settings (contrast and backlight at a time) from non - volatile memory.
- void `uOS_lcd_reset` (void)
Resets the entire LCD controller.
- void `uOS_lcd_cursor_off` (void)
No description.
- void `uOS_lcd_cursor_block` (void)
Show a block cursor.
- void `uOS_lcd_cursor_underscore` (void)
Show an underscore cursor.
- void `uOS_lcd_scroll_left` (void)
Scroll the display left.
- void `uOS_lcd_normal` (void)
Go back to normal LCD display mode.
- void `uOS_lcd_scroll_right` (void)
No description.
- void `uOS_lcd_roll_left` (void)
No description.
- void `uOS_lcd_roll_right` (void)
No description.
- void `uOS_lcd_disp_user_char_ex` (unsigned char number)
Display user defined character (extended).
- void `uOS_lcd_clear_row` (unsigned char row)
Clear single row.
- void `uOS_lcd_sleep` (void)
Enter idle mode for low power consumption.
- void `uOS_lcd_enter_power_mode` (LCD_PWR_MODE_TYPE mode)
Enter the specified power mode.

4.1 Opis szczegółowy

Provide host routines for uOS_LCD driver. [uOS_lcd_host.c](#)

```
Copyright (c) 2006, 2007, 2010, 2012 Viresco
Lukasz Matecki (lukasz.matecki(at)viresco.pl)
All rights reserved
Project:          uOS_LCD
Platform:         N/A.
Processor:        Portable
```

Wersja

\$Revision: 1.6 \$

Data

Created: 2006/12/27 18:21:41

Last modified: \$Date: 2012/03/19 15:46:07 \$

Autor

Created by: Lukasz Matecki

Last modified by: \$Author: Lukasz \$

This file contains the API functions for the uOS lcd controller. There is little to do during initialization. Usually you will only need to call [uOS_lcd_host_init\(\)](#). Below is an example of initialization.

```
#include <uOS_lcd_host.h>

// Initialize your transmission interface (SPI in this example).
spi_soft_init();

// Install your transmission routine into the LCD library.
uOS_lcd_host_init(spi_soft_transmit_byte);

// Now the LCD is ready to use. Lets clear the display.
uOS_lcd_cls();

// If this is an OLED then make sure it is powered up.
uOS_lcd_enter_power_mode(LCD_PWR_OLED_UP);

// And, as always, the Hello World example.
uOS_lcd_write_at("Hello World", 0, 3);
```

4.2 Historia zmian

Wersja	Data	Opis zmiany
1.6	2012-03-19	Added coding examples.
1.5	2012-03-17	Documentation.
1.4	2008-08-30	Corrected char signedness.
1.3	2008-08-30	Merged with another file.
1.2	2008-04-10	Committed.
1.1	2008-04-10	Initial revision.

4.3 Dokumentacja funkcji

4.3.1 void uOS_lcd_host_init (uOS_lcd_send_byte_type send)

Set user defined/default LCD routine.

Parametry

← *send* - byte transmit routine to be used for LCD control.

→ -

↔ -

Zwraca N/A.4.3.2 void uOS_lcd_send_str (const char * *str*)

Send NULL - terminated control string to LCD.

Parametry← ***str*** - the string to send.

→ -

↔ -

Zwraca N/A.4.3.3 void uOS_lcd_send_raw_command (unsigned char *command*)

Send control command to LCD.

Parametry← ***command*** - raw LCD command to send.

→ -

↔ -

Zwraca N/A.**Nota** This function can be used to directly access the LCD module.4.3.4 void uOS_lcd_print (char * *str*)

Print NULL - terminated string onto LCD.

Parametry← ***str*** - string to be printed.

→ -

↔ -

Zwraca N/A.4.3.5 void uOS_lcd_goto (unsigned char *row*, unsigned char *col*)

Move cursor to specified position.

uOS_lcd_goto

Parametry← ***row*** - zero based row number to go to.← ***col*** - zero based column number to go to.

→ -

↔ -

Zwraca N/A.

4.3.6 void uOS_lcd_terminal (char * *newline*)

Add a new line to a terminal emulated on LCD.

Parametry

← ***newline*** - string to be added.

→ -

↔ -

Zwraca

N/A.

4.3.7 void uOS_lcd_write_at (char * *str*, unsigned char *row*, unsigned char *col*)

Print NULL - terminated string at specified position.

Parametry

← ***str*** - string to print.

← ***row*** - row number where to start printing;

← ***col*** - column number where to start printing.

Zwraca

N/A.

Nota

For performance, nothing is done if the arguments are identical as in the previous invocation.

4.3.8 void uOS_lcd_set_contrast (unsigned char *contrast*)

Set LCD contrast.

Parametry

← ***contrast*** - 0:255 - contrast voltage regulated in range 0:5 VDC.

→ -

↔ -

Zwraca

N/A.

The controller doesn't save this setting automatically. If you want to save it, use [uOS_lcd_save_defaults\(\)](#).

4.3.9 void uOS_lcd_set_backlight (unsigned char *backlight*)

Set LCD backlight intensity.

Parametry

← ***backlight*** - 0:255 - backlight voltage regulated in range 0:5 VDC.

→ -

↔ -

Zwraca

N/A.

The controller doesn't save this setting automatically. If you want to save it, use [uOS_lcd_save_defaults\(\)](#).

4.3.10 void uOS_lcd_cls (void)

Clear the LCD.

Parametry

← -

→ -

↔ -

Zwraca

N/A.

4.3.11 void uOS_lcd_define_char (unsigned char *row0*, unsigned char *row1*, unsigned char *row2*, unsigned char *row3*, unsigned char *row4*, unsigned char *row5*, unsigned char *row6*, unsigned char *row7*, unsigned char *number*)

Configure user - defined LCD character.

Parametry

← ***row0:row7*** - bitmap of the user - defined character.

← ***number*** - character index to be specified.

→ -

↔ -

Zwraca

N/A.

row0 contains first (top most) row pixels, *row1* - second etc. 3 most significant bits of *row0:row7* are not visible. You can use definitions from `bin_defs.h` for simplicity:

```
// This will define a glyph with number 10 written vertically.
uOS_lcd_define_char(
    _b____X____,
    _b____XX____,
    _b____X____,
    _b____XXX____,
    _b____XXX____,
    _b____X____X,
    _b____X____X,
    _b____XXX____,
    0);
```

4.3.12 void uOS_lcd_set_char (unsigned char * *buf*, unsigned char *number*)

Configure user - defined character using specified bitmap buffer.

Parametry

← ***buf*** - the bitmap buffer, 8 bytes long.

← ***number*** - character index to be specified.

→ -

↔ -

Zwraca

N/A.

Zobacz również [uOS_lcd_define_char\(\)](#)

4.3.13 void uOS_lcd_disp_user_char (unsigned char *number*)

Display one of the user - defined characters.

Parametry

← *number* - 0:7 - the index of character to be displayed.

→ -

↔ -

Zwraca N/A.

4.3.14 void uOS_lcd_save_defaults (void)

Save settings (contrast and backlight at a time) to non - volatile memory.

Parametry

← -

→ -

↔ -

Zwraca N/A.

Zobacz również [uOS_lcd_load_defaults\(\)](#), [uOS_lcd_set_contrast\(\)](#), [uOS_lcd_set_backlight\(\)](#)

Nota Depending on the LCD module type, the controller may refuse setting (and saving) of a certain values of backlight setting. This is done to prevent LED backlight source from overheating.

```
// Set some values for contrast and backlight.

// Medium contrast.
uOS_lcd_set_contrast(120);

// Backlight almost off.
uOS_lcd_set_backlight(50);

// Save the new settings. They will be stored in EEPROM.
uOS_lcd_save_defaults();
```

4.3.15 void uOS_lcd_load_defaults (void)

Load settings (contrast and backlight at a time) from non - volatile memory.

Parametry

← -

→ -

↔ -

Zwraca N/A.

Nota The load operation is also automatically performed at power - up.

4.3.16 void uOS_lcd_reset (void)

Resets the entire LCD controller.

Parametry

← -

Zwraca N/A.

Nota Initializing the controller may take as long as about 1 second.

4.3.17 void uOS_lcd_cursor_off (void)

No description.

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.18 void uOS_lcd_cursor_block (void)

Show a block cursor.

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.19 void uOS_lcd_cursor_underscore (void)

Show an underscore cursor.

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.20 void uOS_lcd_scroll_left (void)

Scroll the display left.

Parametry

← -

→ -

↔ -

Zwraca N/A.

Nota This operation uses the internal LCD scroll operation. Based on the LCD size, this will effect in wrapping the character around. This only happens when the internal LCD memory block has the same size as the actual LCD (the size of the block is always 80 characters).

Ostrzeżenie Scrolling (shifting) alters the relationship between addresses and their positions on the screen.

4.3.21 void uOS_lcd_normal (void)

Go back to normal LCD display mode.

uOS_lcd_normal

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.22 void uOS_lcd_scroll_right (void)

No description.

Parametry

← -

→ -

↔ -

Zwraca N/A.

Nota This operation uses the internal LCD scroll operation. Based on the LCD size, this will effect in wrapping the character around. This only happens when the internal LCD memory block has the same size as the actual LCD (the size of the block is always 80 characters).

Ostrzeżenie Scrolling (shifting) alters the relationship between addresses and their positions on the screen.

4.3.23 void uOS_lcd_roll_left (void)

No description.

uOS_lcd_roll_left

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.24 void uOS_lcd_roll_right (void)

No description.

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.25 void uOS_lcd_disp_user_char_ex (unsigned char *number*)

Display user defined character (extended).

Parametry

← **number** - the user defined glyph number.

→ -

↔ -

Zwraca N/A.

This function can display more than 8 different user defined characters. The already defined glyphs are stored in `defined_glyphs` for performance (they are not redefined if they are already in the LCD memory).

Nota Defining new glyphs does not consume RAM memory.

4.3.26 void `uOS_lcd_clear_row` (unsigned char *row*)

Clear single row.

Parametry

← **row** - number of row (0-based) to clear.

→ -

↔ -

Zwraca N/A.

Nota The cursor position is not restored after operation.

Zobacz również [uOS_lcd_cls\(\)](#)

4.3.27 void `uOS_lcd_sleep` (void)

Enter idle mode for low power consumption.

Parametry

← -

→ -

↔ -

Zwraca N/A.

4.3.28 void `uOS_lcd_enter_power_mode` (LCD_PWR_MODE_TYPE *mode*)

Enter the specified power mode.

Parametry

← **mode** - the power mode to enter:

- LCD_PWR_OLED_DOWN - enter low power mode for an OLED display;
- LCD_PWR_OLED_UP - enter normal power mode for an OLED display.

Zwraca N/A.

Nota The behavior depends on the LCD type.

Zobacz również [uOS_lcd_sleep\(\)](#)

Spis tablic

1	Wyprowadzenia złącza	4
---	--------------------------------	---

Spis rysunków

1	Uruchomiony μ OS LCD	3
2	Widok sterownika	3