Osama's Lab

# KS0108 Graphic LCD Library For AVR Microcontrollers GCC & CodeVisionAVR

Ver. 2.00

Osama's Lab

## Contents

Contents ........ 2
Introduction ........ 3
Features ........ 3
Function summary ........ 4
Before you begin ........ 14
Using the library ........ 15
  Download the library ........ 15
  Initialize the library ........ 16
  Use the library ........ 17
How to ........ 17
  Reading the contents of a certain location on the display ........ 17
  Drawing solid, dotted, and dashed lines and shapes ........ 17
  Specifying the color of lines and shapes ........ 18
  Moving the whole contents of the display up and down without repainting ........ 18
  Writing Arabic and English texts ........ 18
Hardware connections ........ 19
Examples ........ 19
License ........ 19
FAQ ........ 20
  How to turn on/off the backlight? ........ 20
  How to reset the GLCD module? ........ 21
  Can I assign control pins that belong to different ports? ........ 21
  The GLCD displays hazy or messy paints and texts, what is the problem? ........ 21
  The backlight shines, but nothing appears on the display, what is the problem? ........ 22
  Only one half of the display work, what is the problem? ........ 22
  How can I adjust the contrast of the display? ........ 22
  How can I change the font? ........ 22
  How can I ask a question not listed here? ........ 22
Version History ........ 23
References ........ 23

## Introduction

Osama's Lab GLCD Library is used for controlling KS0108 dependent graphic LCD modules, it provides a number of easy – to – use functions for reading and writing from and to the RAM of the GLCD.

## Features

- Compatible with CodeVisionAVR and GCC (two versions)
- Arabic, Farsi and English texts at 7 different sizes
- Character spacing
- BMP images display
- Turning the display on and off
- Changing the start line of the display
- Reading, writing, and clearing any line
- Drawing single points without changing the neighboring points
- Drawing the following geometric shapes, each can be drawn solid, dotted, or dashed, even the length of dashes can be controlled!
    - Horizontal lines
    - Vertical lines
    - Inclined lines
    - Rectangles (Note: Squares are rectangles)
    - Cuboids (Note: Cubes are cuboids)
    - Horizontal parallelograms
    - Vertical parallelograms
    - Horizontal parallelepipeds
    - Vertical parallelepipeds
    - Circles

## Function summary

The following functions are provided by the library:

- bmp_disp
- circle
- cuboid
- glcd_clear
- glcd_clrln
- glcd_off
- glcd_on
- glcd_putchar
- glcd_puts
- glcd_read
- glcd_write
- goto_col
- goto_row
- goto_xy
- h_line
- h_parallelepiped
- h_parallelogram
- line
- point_at
- rectangle
- set_start_line
- v_line
- v_parallelepiped
- v_parallelogram

| Method | Description | Parameters |
|---|---|---|
| glcd_on() | Turn on the GLCD | No |
| glcd_off() | Turn off the GLCD | No |
| set_start_line(unsigned char line) | Changes the top line on the display [1] | line: line number to be set at the top (Range: 0 → 63) |
| goto_col(unsigned int x) | Go to the specified column | x: Column number (Range: 0 → 127) |
| goto_row(unsigned int y) | Go to the specified row [1] | y: Row address (Range: 0 → 7) |
| goto_xy(unsigned int x, unsigned ,int y) | Go to the specified column and row | x: Column number<br>y: Row address |
| glcd_write(unsigned char b) | Writes 1 byte data at the current location | b: 1 – byte data to be written at the current location |
| glcd_clrln(unsigned char ln) | Clears the specified row [1] | ln: the number of the row to be cleared (Range: 0 → 7) |
| glcd_clear() | Clears the display | No |
| glcd_read(unsigned char column) | Reads the byte at the current position | column: Current column number [2] |
| point_at(<br>unsigned int x,<br>unsigned int y,<br>byte color) | Adds a point at the specified location | x: column number<br>y: row number<br>color: 0 → Light spot<br>1→ Dark spot |

[1] Each row consists of 8 lines, hence, the GLCD contains 8 rows and 64 lines
[2] The column number is just used for enabling the appropriate GLCD half

| | | |
|---|---|---|
| h_line(<br>unsigned int x,unsigned int y, byte l,byte s,byte c) | Draws a horizontal line | x: The column number at which the line starts<br>y: The row number at which the line starts<br>l: Line length<br>s: The space between line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |
| v_line(<br>unsigned int x,unsigned int y, signed int l,byte s,byte c) | Draws a vertical line | x: The column number at which the line starts<br>y: The row number at which the line starts<br>l: Line length<br>s: The space between line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |

| | | |
|---|---|---|
| line(unsigned int x1,unsigned int y1, unsigned int x2,unsigned int y2, byte s,byte c) | Draws a general (inclined) line | x1: The column number at which the line starts<br>y1: The row number at which the line starts<br>x2: The column number at which the line ends<br>y2: The row number at which the line ends<br>s: The space between line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |
| rectangle(<br>unsigned int x1,unsigned int y1,<br>unsigned int x2,unsigned int y2,<br>byte s,byte c) | Draws a rectangle | x1: The x of the upper left point<br>y1: The y of the upper left point<br>x2: The x of the lower right point<br>y2: The y of the lower right point<br>s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |

| | | |
|---|---|---|
| cuboid(<br>unsigned int x11,unsigned int y11,<br>unsigned int x12,unsigned int y12,<br>unsigned int x21,unsigned int y21,<br>unsigned int x22,unsigned int y22,<br>byte s,byte c) | Draws a cuboid by defining two surfaces | x11: The x of the upper left point of the first surface<br>y11: The y of the upper left point of the first surface<br>x12: The x of the lower right point of the first surface<br>y12: The y of the lower right point of the first surface |
| | | X21: The x of the upper left point of the second surface<br>Y21: The y of the upper left point of the second surface<br>x22: The x of the lower right point of the second surface<br>y22: The y of the lower right point of the second surface |
| | | s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |

| | | |
|---|---|---|
| h_parallelogram(<br>unsigned int x1,unsigned int y1,<br>unsigned int x2,unsigned int y2,<br>byte l,byte s,byte c) | Draws a parallelogram its upper and lower sides are horizontal | x1: The x of the upper left point<br>y1: The y of the upper left point<br>x2: The x of the lower right point<br>y2: The y of the lower right point<br>l: The length of the horizontal side (upper or lower)<br>s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |
| v_parallelogram(<br>unsigned int x1,unsigned int y1,<br>unsigned int x2,unsigned int y2,<br>byte l,byte s,byte c) | Draws a parallelogram its right and left sides are vertical | x1: The x of the upper left point<br>y1: The y of the upper left point<br>x2: The x of the lower right point<br>y2: The y of the lower right point<br>l: The length of the vertical side (right or left)<br>s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |

| | | |
|---|---|---|
| h_parallelepiped(<br>unsigned int x11,unsigned int y11,<br>unsigned int x12,unsigned int y12,<br>byte l1,<br>unsigned int x21,unsigned int y21,<br>unsigned int x22,unsigned int y22,<br>byte l2,<br>byte s,byte c) | Draws a parallelepiped its bases are two horizontal parallelograms (See h_parallelogram) | x11: The x of the upper left point of the first surface<br>y11: The y of the upper left point of the first surface<br>x12: The x of the lower right point of the first surface<br>y12: The y of the lower right point of the first surface |
| | | X21: The x of the upper left point of the second surface<br>Y21: The y of the upper left point of the second surface<br>x22: The x of the lower right point of the second surface<br>y22: The y of the lower right point of the second surface |
| | | s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots<br>l1: The length of the horizontal side of the first surface<br>l2: The length of the horizontal side of the second surface |

| | | |
|---|---|---|
| v_parallelepiped(<br>unsigned int x11,unsigned int y11,<br>unsigned int x12,unsigned int y12,<br>byte l1,<br>unsigned int x21,unsigned int y21,<br>unsigned int x22,unsigned int y22,<br>byte l2,<br>byte s,byte c) | Draws a parallelepiped its bases are two vertical parallelograms (See v_parallelogram) | x11: The x of the upper left point of the first surface<br>y11: The y of the upper left point of the first surface<br>x12: The x of the lower right point of the first surface<br>y12: The y of the lower right point of the first surface |
| | | X21: The x of the upper left point of the second surface<br>Y21: The y of the upper left point of the second surface<br>x22: The x of the lower right point of the second surface<br>y22: The y of the lower right point of the second surface |
| | | s: The space between each line points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots<br>l1: The length of the vertical side of the first surface<br>l2: The length of the vertical side of the second surface |

| | | |
|---|---|---|
| circle(<br>unsigned int x0,unsigned int y0, unsigned int r,byte s,byte c) | Draws a circle | x0: The x point of the center of the circle<br>y0: The y point of the center of the circle<br>r: Circle's radius<br>s: The space between perimeter points:<br>0 → solid line<br>1 → dotted line<br>>1 → dashed line<br>c: 0 → Light spots<br>1→ Dark spots |
| glcd_putchar(byte c,int x,int y,byte l, byte sz) | Writes a character at the specified position, with size sz | c: The character to be typed<br>x: The column number to start typing the character at (One character occupies 8 columns)<br>y: The row number to type the character at<br>l: The language of the character<br>0 → English<br>1 → Arabic or Farsi<br>sz: Font size (from 1 to 7) |

| | | |
|---|---|---|
| glcd_puts(<br>byte *c,int x,int y,unsigned char l, byte sz,signed char space) | Writes a string ,stored in the flash memory ,on the display | c: A pointer to the string to be written on the display<br>x: The column number to start typing the character at (One character occupies 8 columns)<br>y: The row number to type the character at<br>l: The language of the character<br>0 → English<br>1 → Arabic or Farsi<br>sz: Font size (from 1 to 7)<br>space:<br>English: Character spacing<br>Arabic and Farsi:Word spacing |
| bmp_disp(flash byte *bmp, unsigned int x1,unsigned int y1, unsigned int x2,unsigned int y2) | Displays a bmp image array stored in the flash memory | bmp: A pointer to the array where the bmp image is stored<br>x1: The x of the upper left point of the image on the glcd<br>y1: The y of the upper left point of the image on the glcd<br>x2: The x of the lower right point of the image on the glcd<br>y2: The y of the lower right point of the image on the glcd |

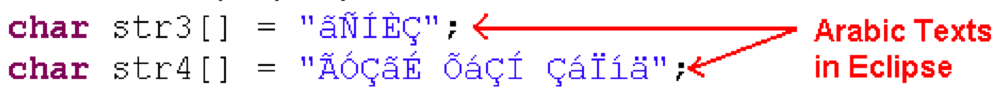
## Before you begin

Please read the following notes carefully before you start using the library:

- For flexibility, ports used for interfacing the graphic LCD aren't initialized by the library, you **must** initialize them at the start of your program:

```
DDRA = 0xFF;          //Data port direction
DDRB = 0x3F;          //Control port direction
```

- Some compilers don't support Arabic texts, you can type any Arabic string you want in the code, and it will be displayed properly on the GLCD even if it appears in the code editor improperly:



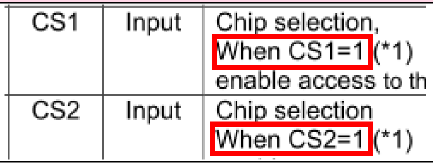
```
char str3[] = "ãÑÍÈÇ";
char str4[] = "ÃÓçãÉ õáçÍ çáÏíä";
```

Don't change character encoding in Eclipse, but instead you can edit the code using Notepad ++

- There are two versions of the library, "CodeVisionAVR" version and "gcc" version, if you use "gcc" version, only one port can be used as a control port, but if you use "CodeVisionAVR" version, each control pin can be specified separately.

| CodeVisionAVR | gcc |
|---|---|
| #define CS1 PORTC.2<br>#define CS2 PORTC.3<br>#define RS PORTD.7<br>#define RW PORTC.0<br>#define EN PORTC.1 | #define CONTROLPORT PORTC<br>#define CS1 (1<<2)<br>#define CS2 (1<<3)<br>#define DI (1<<4)<br>#define RW (1<<0)<br>#define EN (1<<1) |
| You can specify control pins individually and in different ports | You can select the order of control pins, but they must be in the same port |

- Some graphic LCD modules use active – low enable pins, while other modules use active – high enable pins, refer to the datasheet before you use the library to know to which type your module belongs, then it is easy to tell the library which type it uses by un/commenting the "#define CS_ACTIVE_LOW" in the library.

| | Active High | Active Low |
|---|---|---|
| From the datasheet |  CS1 Input Chip selection, When CS1=1 (*1) enable access to th CS2 Input Chip selection When CS2=1 (*1) | NA |
| In the library | // #define CS_ACTIVE_LOW 0 | #define CS_ACTIVE_LOW 0 |

- It is strongly recommended that you read precautions at the end of the datasheet of the GLCD module you use.

## Using the library

### Download the library

There are two versions of the library; CodeVisionAVR version and gcc version, Select any of them:

- CodeVisionAVR
- GCC

- Extract the compressed file, and include "glcd.h" into your project, "font.h" is included implicitly by "glcd.h".

## Initialize the library

| | CodeVisionAVR | GCC | Notes |
|---|---|---|---|
| Modify the following lines at the start of "glcd.h" file | | | |
| Trigger pulse delay | #define E_DELAY d | #define E_DELAY d | d: delay time in us (5 us is suitable) |
| Data port | #define DATAPORT PORTX<br>#define DATADDR DDRX<br>#define DATAPIN PINX | #define DATAPORT PORTX<br>#define DATADDR DDRX<br>#define DATAPIN PINX | X: port name (A, B, …) |
| Control pins/port | #define CS1 PORTX.n<br>#define CS2 PORTX.n<br>#define RS PORTX.n<br>#define RW PORTX.n<br>#define EN PORTX.n | #define CS1 (1<<n)<br>#define CS2 (1<<n)<br>#define DI (1<<n)<br>#define RW (1<<n)<br>#define EN (1<<n) | X: port name (A, B, …)<br>n: pin number (0, 1, 2, …. ) |
| Active High/Low | #define CS_ACTIVE_LOW 0 | #define CS_ACTIVE_LOW 0 | - Uncomment this line for active LOW<br>- '0' can be any other number |
| Add the following lines at the start of your code (inside the main() body) | | | |
| Data direction of data port | DATADDR = 0xff; | DATADDR = 0xff; | |
| Data direction of control port | DDRX = pattern; | DDRX = pattern; | X: data port name (A, B, ….)<br>- pattern depends on "Control pins/port" settings |

## Use the library

Now, you are ready to use the library, each function is described in "Function summary" section

## How to

- Read the contents of a certain location on the display
- Draw solid, dotted, and dashed lines and shapes
- Specify the color of lines and shapes
- Move the whole contents of the display up and down without repainting
- Write Arabic and English texts

## Reading the contents of a certain location on the display

- Use "goto_xy" function to go to the required location
- Use "glcd_read" function to read the byte at the current row

## Drawing solid, dotted, and dashed lines and shapes

For each drawing function, there is an argument named "s", this specifies the separation between the points of the shape
Example:

h_line(1,1,20,2,0)

The third parameter is the separation, in this example it is "2", this means that the horizontal line consists of a point followed by two blank spaces, actually these two spaces are not always blank, but I mean here that they aren't touched, so if they contained any data previously, this data will remain untouched. For drawing solid shapes, simply set this parameter to 0

## Specifying the color of lines and shapes

For each drawing function, there is an argument named "c", this argument defines the color of the shape, 0 for light shapes, and 1 for light shapes

Example:

h_line(1,1,20,0,0) → Draws bright horizontal line
h_line(1,1,20,0,1) → Draws dark horizontal line

## Moving the whole contents of the display up and down without repainting

The function "set_start_line" changes the line to be displayed at the top of the display, thus you can make the any line top one.

Example:

The following example scrolls the display vertically, once a line disappears at the top, it appears at the bottom, as if the display is mounted on a pulley.

## Writing Arabic and English texts

"glcd_puts" and "glcd_putchar" functions, have an argument named "l", this argument defines the language of the written string; 0 for English strings, and 1 for Arabic strings

Example:

```
char ar_string[] = "بسم الله الرحمن الرحيم";
char *en_string[] = "Hello";
glcd_puts(ar_string,5,5,1);            //Writes an Arabic string
glcd_puts(en_string,5,5,0);            //Writes an English(Latin) string
```

## Hardware connections

The order of the pins differs from a module to another, so before you connect your hardware please double – check your connections not to connect any wire improperly, this may damage your module, the order and the name of each pin can be found in the datasheet under a section named "Terminal Functions" or a similar name.

## Examples

There are four illustrative examples:

- CodeVisionAVR with active HIGH GLCD
- CodeVisionAVR with active LOW GLCD
- GCC with active HIGH GLCD
- GCC with active LOW GLCD

- For compiling GCC projects, I used eclipse with AVR – eclipse plug-in.
- The latest version of eclipse is available here.
- AVR – Eclipse download
- Download CodeVisionAVR
- Download Proteus Simulator

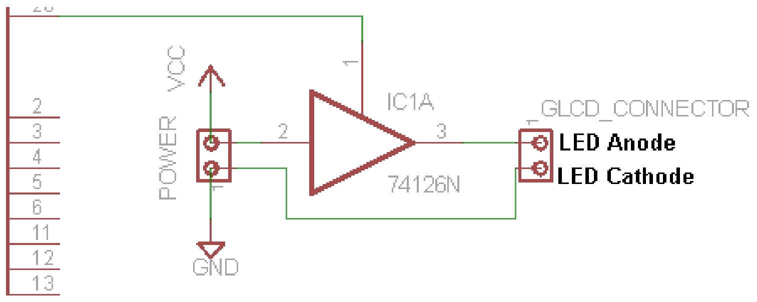
## License

## FAQ

- How to turn on/off the backlight?
- How to reset the GLCD module?
- Can I assign control pins that belong to different ports?
- The GLCD displays hazy or messy paints and texts, what is the problem?
- The backlight shines, but nothing appears on the display, what is the problem?
- Only one half of the display work, what is the problem?
- How can I adjust the contrast of the display?
- How can I change the font?
- How can I ask a question not listed here?

### How to turn on/off the backlight?

The backlight is a part of the LCD itself, not the controller, so, it is out of the GLCD controller's control, but you can assign a pin in your microcontroller to enable/disable a buffer that acts as a switch for the backlight.



## How to reset the GLCD module?

Simply, connect one of the free pins of the microcontroller to the "RST" pin of the GLCD module, and then you can reset the GLCD module by pulling this pin low for a short period, this period can be found in the datasheet of the GLCD module, it is about 200 nsec.

## Can I assign control pins that belong to different ports?

Only in CodeVisionAVR version, this is not available in gcc version

## The GLCD displays hazy or messy paints and texts, what is the problem?

- Make sure that the data port and control pins are defined properly at the library header file
- Make sure that you set the data direction of the data port and the control port properly, remember, the data direction registers are not set by the library, you must set them manually.
- This problem is mostly caused by the delay time, try increasing the defined delay time at the start of the library header file:

  ```
  #define E_DELAY 3
  ```

  If it didn't work, try increasing the other delays at the different functions of the library

- Bad connections can also cause such problems, double – check your connections
- Poor power supplies may also be the reason, ensure that your power supply is sufficient

## The backlight shines, but nothing appears on the display, what is the problem?

The previous answer.

## Only one half of the display work, what is the problem?

- Check your connections
- Make sure that the data direction registers, the control pins, and the data port are initialized properly (Refer to “Before you begin” section).
- Refer to this answer

## How can I adjust the contrast of the display?

This must be illustrated in the datasheet under a section entitled “Adjusting the LCD display contrast” or a similar title, but in general, this is achieved by using a 25 – 50k ohms potentiometer, the middle terminal is connected to “Vo” pin, and the other two terminals are connected to “Vout” and the power source positive terminal (Vdd)

## How can I change the font?

1. Download arabic.zip and extract it.
2. Download LCD font generator.
3. Open "arabic.lff" and modify each character as you like.
4. Generate the modified array and replace the array “font” in "font.h" with the newly generated array, don't change array’s name or type, leave it as it is.

## How can I ask a question not listed here?

Use the feedback form at Osama’s Lab website

Or contact me by e-mail:

os.dexterity@gmail.com

## Version History

Ver. 2.00

- Farsi language support
- Different text sizes support (for all languages)
- "delay.h"/ "util/delay.h" have been included into "glcd.h", you no longer have to include them

## References

- KS0108 datasheet
- LM12864LFW LCD module manual

If you have any question, suggestion, note, found a bug, or like to say anything, please don't hesitate to contact me.

Thanks,

*Osama Salah Eldin*

- Feedback
- Subscribe to Osama's Lab newsletter