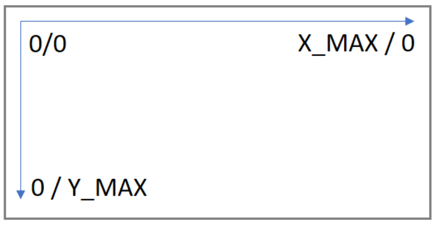
# Smart I2C GLCD – Instructions

Version 1.12, 24-Mar-2018 – Stephan Laage-Witt

## Coordinate System



The origin (coordinate 0/0) is located at the upper left corner.

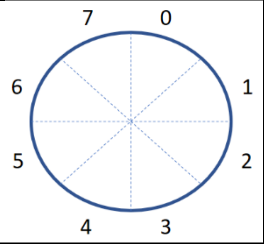
Y_MAX is 63.

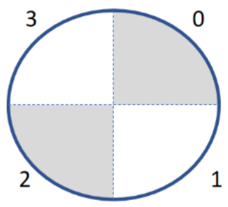
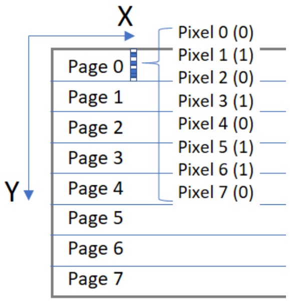
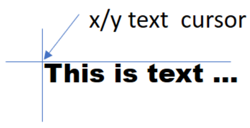
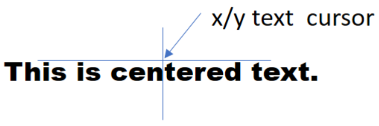
X_MAX is 127 for displays with 2 graphic controller chips (type 1), and 191 for those with 3 chips (type 2)

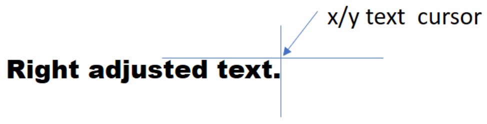
Coordinate specifications outside this range are clipped.

## Instruction Set

| Mnemonic | Code | Parameters (bytes) |
|---|---|---|
| **GLCD_ON** | **01** | None |
| Switches on the display. | | |
| **GLCD_OFF** | **02** | None |
| Switches display off and enters low power mode. The display is blanked; however, display memory will be maintained. | | |
| **GLCD_SET_LIGHT** | **03** | Light level [range: 0…10] |
| Sets the level of the display back light. The range is 0 (completely switched off) to 10 (maximum light level). The new value is saved to the eePROM and will be used as default setting after power on. | | |
| **GLCD_DIM_ON** | **04** | None |
| Shuts off the display back light. The default light level (as stored in the eePROM) remains unchanged. This command is useful for temporarily switching off the display light, e.g. to reduce power consumption. | | |
| **GLCD_DIM_OFF** | **05** | None |
| Restores the display back light to the previous value (prior to GLCD_DIM_ON). | | |
| **GLCD_SET_I2C** | **06** | I2C address [range: 8…127] |
| Changes the current I2C address to a new value. The display will be reset. The new I2C address is captured in the eePROM and will be used as default at power on. | | |
| **GLCD_VERBOSE_ON** | **07** | None |
| Activates verbose mode. In this mode, the display shows the following additional information:<br>1) At start-up, key parameters (firmware version, I2C address, display back light level)<br>2) During use, unknown instruction tokens are displayed preceded by '*'<br>This mode is useful for debugging. Verbose setting is preserved in the eePROM. | | |
| **GLCD_VERBOSE_OFF** | **08** | None |
| Deactivates verbose mode | | |
| **GLCD_AUTOSCROLL_ON** | **09** | None |
| Activates auto scroll for text output via the instructions GLCD_DRAW_STRING, GLCD_DRAW_CHAR, GLCD_DRAW_UDEC, GLCD_DRAW_SDEC. Auto scroll includes line feed/carriage return at the right border and scrolling upwards at the bottom border.<br>Auto scroll setting is preserved in the eePROM. | | |
| **GLCD_AUTOSCROLL_OFF** | **10** | None |
| Deactivates auto scroll. Text output beyond the right or bottom border are being ignored. | | |

| Mnemonic | Code | Parameters (bytes) |
|---|---|---|
| **GLCD_ENTER_SLEEP_MODE** | **11** | None |
| Enters the sleep mode of the controller to further reduce power consumption. This function requires the back light to be switched off (light level 0 or dim function on) to become active. Otherwise it will be ignored. Sleep mode will be disabled by any following instruction. Power consumption will be reduced by approximately 1mA. | | |
| **GLCD_CLEAR_SCREEN** | **15** | mode |
| Erases the display. If mode is 0, then the display is filled with “off”-pixels, otherwise “on”-pixels. | | |
| **GLCD_SET_PIXEL** | **16** | x, y, mode |
| Sets a pixel at the specified coordinate. If *mode* equals 0 then an “off”-pixels is set, otherwise an “on”-pixel. | | |
| **GLCD_DRAW_LINE** | **17** | x_start, y_start, x_end, y_end, mode |
| Plots a straight line. If *mode* equals 0 then “off”-pixels are used, otherwise “on”-pixels. | | |
| **GLCD_DRAW_DOTTED_VER_LINE** | **19** | x_pos, y_start, y_end, spacing, mode |
| Plots a dotted vertical line.<br>*x_pos* specifies the x coordinate for the vertical line. *y_start* and *y_end* define the start and end position of the line.<br>*spacing* defines the number of empty pixels between the dots. The value 0 results in a solid line.<br>*mode* specifies whether “off”-pixels (mode equals 0) and are “on”-pixels should be used. | | |
| **GLCD_DRAW_DOTTED_HOR_LINE** | **20** | y_pos, x_start, x_end, spacing, mode |
| Plots a dotted horizontal line.<br>*y_pos* specifies the y coordinate for the horizontal line. *x_start* and *x_end* define the start and end position of the line.<br>*spacing* defines the number of empty pixels between the dots. The value 0 results in a solid line.<br>*mode* specifies whether “off”-pixels (mode equals 0) and are “on”-pixels should be used. | | |
| **GLCD_DRAW_FUNCTION** | **20** | n, x_start, mode,<br>y data points {y1, y2, … yn} |
| Plots a series of y-coordinates from left to right.<br>*n* specifies the number of data points to be plotted.<br>*x_start* is the first x-coordinate position. The x-coordinate will be auto-incremented by 1 (shifted to the right) for each data item. The sum of *n* and *x_start* must not exceed X_MAX to avoid plotting beyond the right border of the display.<br>*mode* specifies whether “off”-pixels (mode equals 0) and are “on”-pixels should be used.<br>*{y1 … yn}* are the data points to be plotted. The number of data points must match *n*. | | |
| **GLCD_DRAW_SCATTER** | **21** | n, x_start, mode,<br>y_data points {y1, y2, … yn} |
| Same as “GLCD_DRAW_FUNCTION”, except that the y data points are not connected. | | |
| **GLCD_DRAW_RECTANGLE** | **22** | x_start, y_start, x_end, y_end, mode |
| Plots a rectangle of straight lines with *start* being the upper left corner and *end* the lower right coordinate. If *mode* equals 0 then “off”-pixels are used, otherwise “on”-pixels. | | |
| **GLCD_DRAW_FILLED_RECTANGLE** | **23** | x_start, y_start, x_end, y_end, mode |
| Same as “GLCD_DRAW_RECTANGLE”, except that the rectangle will be filled. | | |
| **GLCD_DRAW_CIRCLE** | **25** | x_origin, y_origin, radius, segment, mode |
| Plots a circle at *origin* with *radius* as specified.<br>*Segment* specifies the sector(s) to be plotted. Each bit represents a sector as shown on the right. Any combination is possible. E.g. *0b11111111* plots a full circle, *0b00001111* specifies a right half circle, etc.<br>If *mode* equals 0 then “off”-pixels are used, otherwise “on”-pixels. |  | |

| Mnemonic | Code | Parameters (bytes) |
|---|---|---|
| **GLCD_DRAW_FILLED_CIRCLE** | **26** | x_origin, y_origin, radius, segment, mode |
| Plots a filled circle at *origin* with *radius* as specified. *Segment* specifies the sector to be plotted. Each bit represents a sector as shown, with any combination being possible. Only the 4 lower bits are interpreted. E.g. *0b00001111* plots a full filled circle, etc. If *mode* equals 0 then "off"-pixels are used, otherwise "on"-pixels.  | | |
| **GLCD_LOAD_RAW** | **30** | x_pos, y_page [range: 0 … 7], n, raw data bytes {db1, db2, … dbn} |
| Loads data bytes directly into the display memory. Display memory is organized into 8 rows of bytes with each byte representing a vertical column of 8 pixels. *x_pos* specifies the starting position. *y_page* defines the selected page. *n* is the number of bytes to be loaded, starting a *x_pos* and incrementing to the right. Data exceeding X_MAX will be ignored.   X Y Page 0 Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Pixel 0 (0) Pixel 1 (1) Pixel 2 (0) Pixel 3 (1) Pixel 4 (0) Pixel 5 (1) Pixel 6 (1) Pixel 7 (0) | | |
| **GLCD_DRAW_CHAR** | **32** | char |
| Plots a character using the currently selected font set. The character is plotted at the current text cursor position. The cursor position specified the upper left corner of the character and will be shifted to the right end of the character. | | |
| **GLCD_DRAW_STR** | **33** | zero terminated string {char1, char2, … char n, 0} |
| Plots a string. The current text cursor position specified the upper left corner of the string and will be shifted to the right edge of the string after execution. The system handles 2 special characters: '\n' = ASCII(13) -> new line, advances to the next line below the current line, scrolls the display up if needed, and positions the cursor to the left edge of the display. '\t' = ASCII(8) -> advances the cursor to the next tab position. Tab positions are every 32 pixel.  | | |
| **GLCD_DRAW_CENTER_STRG** | **34** | zero terminated string {char1, char2, … char n, 0} |
| Same as GLCD_DRAW_STR, except that the string is centred at the current x-position of text cursor. The string length must not exceed 64 characters. Longer strings are truncated. Special characters ('/n' or '/t') are not allowed for this function.  | | |
| **GLCD_DRAW_UDEC** | **35** | upper byte, lower byte [of unsigned integer, 16 bits], number of digits [1 … 5], decimal point position [0 … number of digits – 1] |
| Prints a decimal representation of a 16 bit value at the current position to the text cursor (see GLCD_SET_CURSOR) *Upper byte* and *lower byte* represent the 16 bits value *to be* displayed. The routine does not handle negative values. *Number of digits* specifies the expected number of positions. The decimal value will be right adjusted and filled with leading spaces to reach the requested number of digits. It is the user's responsibility to ensure that the number of digits allows the decimal value to be displayed. E.g. 3 digits can handle values up to 999 and will give unpredictable results for greater values. *Decimal point position* specifies the position of a decimal point, counting from the right. 0 omits any decimal point. E.g. a value of 2 inserts a decimal point two positions from the right. This feature is useful for fixed point arithmetic. Example: 123, 4, 2 -> `1.23` 123, 5, 0 -> `123` | | |

| Mnemonic | Code | Parameters (bytes) |
|---|---|---|
| **GLCD_DRAW_SDEC** | **36** | upper byte, lower byte [of signed integer, 16 bits],<br>number of digits [1 … 5],<br>decimal point position [0 … number of digits – 1] |
| Same as GLCD_DRAW_UDEC, however handles signed integers.<br>Example: -123, 4, 2 -> `-1.23`<br>-123, 5, 0 -> `-123` | | |
| **GLCD_DRAW_RADJ_STR** | **37** | zero terminated string {char1, char2, … char n, 0} |
| Same as GLCD_DRAW_STR, except that the string is right adjusted at the current x-position of the text cursor.<br>The string length must not exceed 64 characters. Longer strings are truncated. Special characters ('/n' or '/t') are not allowed for this function.  | | |
| **GLCD_SET_CURSOR** | **50** | x, y |
| Sets the cursor for char and string commands | | |
| **GLCD_SET_FONT** | **51** | font [0 … 5] |
| Sets the font set accordingly. For the list of available fonts, see separate chapter. | | |
| **GLCD_SCROLL_UP** | **52** | Pages [1 … 4] |
| Scrolls up the display from bottom to top by the number of *pages* as specified. A page is a row of 8 pixel lines. This function creates space at the bottom of the display. The upper portion of the display memory will be discarded. | | |
| **Functions retrieving status information from the display**<br>Note: Read access to the display requires synchronous execution and cannot be buffered. The user needs to ensure that the interface module executes the read action prior to retrieving data via the I2C interface. This can be achieved by waiting for the *buffer_empty* line to reach high-level prior to executing data retrieval. Alternatively, the host system may wait for a short amount of time (can be up to 50 msec).<br>Incorrect data may be retrieved in case data retrieval is executed before the interface was able to prepare the requested data. For examples, please refer to the Arduino *glcd_functions* library. | | |
| **GLCD_GET_STR_WIDTH** | **60** | zero terminated string {char1, char2, … char n, 0}<br>Returns: (byte) str_width |
| The function calculates the horizontal width (number of pixel) of the submitted string, based on the current font set. Maximum string length is 64 characters. Special characters (e.g. line feed, tabulator) are ignored. | | |
| **GLCD_GET_CURSOR** | **61** | None<br>Returns: (byte) x_cur, (byte) y_cur |
| Returns the current position of the text cursor. | | |
| **GLCD_GET_FONT** | **62** | None<br>Returns: (byte) font |
| Returns the ID number of the currently selected font. | | |
| **GLCD_GET_MAX_XY** | **63** | None<br>Returns: (byte) max_x, (byte) max_y |
| Returns the display width and height in terms of the maximum x- and y-position.<br>max_x: 127 or 192<br>max_y: 63 | | |
| **GLCD_GET_FONT_HEIGHT** | **65** | None<br>Returns: (byte) font_height |
| Returns the height (number of pixels) of the currently selected font | | |

## Fonts

The interface module provides 8 different fonts.

**Font #0** is a fixed space font with a character size pf 5x8. This font serves as default at system start-up.

Fonts 1 to 6 are fonts of different height and variable character widths:

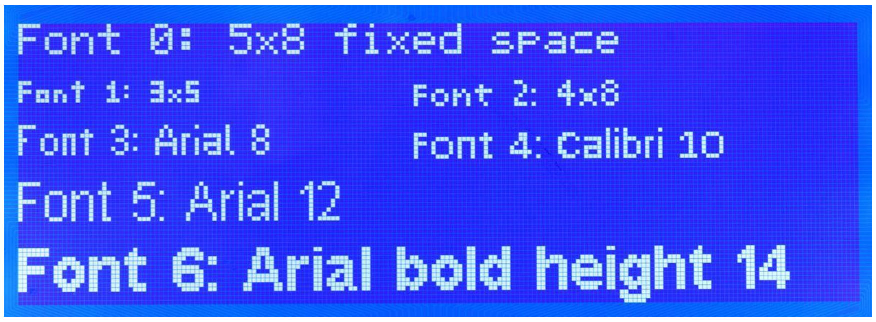
> **Font #1**: height 5 pixels
> **Font #2**: height 8 pixels
> **Font #3**: Arial, height 8
> **Font #4**: Calibri, height 10
> **Font #5**: Arial, height 12
> **Font #6**: Arial bold, height 14

For all fonts, the available ASCII values are limited to the range 32 to 127. ASCII values outside this range are ignored.

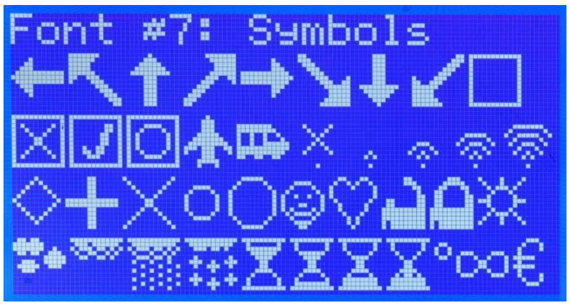
The system accepts control characters:

> '\t' (0x09) advances the cursor to the next horizontal tab stops at 32, 64, 96, 128, 160 pixels
> '\n' (0x0A) initiates a line feed with the pixel height as defined by the current font.

If "auto scroll" is activated, the system inserts line breaks once the text cursor reaches the right border and scrolls up the display at the bottom border.



**Font #7** provides a set of symbols with a height of 12 and variable width. Character values in the range of 32 to 71 are accepted. The following symbols are included as part of the current firmware:



Symbol 32 – 40

Symbol 41 – 50

Symbol 51 – 60

Symbol 61 - 71