

# OLED\_StatusIcons

Add-on Library for OLED\_I2C to display a status bar with icons

## Manual

The logo for Rinky-Dink Electronics features the company name in a stylized, glowing cyan font with a 3D effect. The text is set against a dark background that includes a close-up of a green printed circuit board (PCB) with various electronic components and traces visible.

## Introduction:

























This library is an add-on to OLED\_I2C and will not work on its own.

This library will allow you to display an interrupt-driven status bar with icons on the display. The status bar can be positioned either at the top or the bottom of the display, and the icons can be left or right aligned.

























The library supports 8x8 and 16x16 pixels icons but you cannot mix the two sizes. Only one size can be used at a time.

Icons can be made with the online monochrome Image Converter. Make sure that the images are exactly 8x8 or 16x16 pixels. No other sizes are supported.

### Supplied icons:

8x8					
 arrow_down_8	 arrow_left_8	 arrow_right_8	 arrow_up_8	 ethernet_8	 fan_8
 mail_8	 power_8	 wireless0_8	 Wireless1_8	 Wireless2_8	 Wireless3_8
 batteryh0_8	 batteryh1_8	 batteryh2_8	 batteryh3_8	 batteryh4_8	 batteryhc_8
 batteryv0_8	 batteryv1_8	 batteryv2_8	 batteryv3_8	 batteryv4_8	 batteryvc_8

16x16					
 arrow_down_16	 arrow_left_16	 arrow_right_16	 arrow_up_16	 ethernet_16	 fan_16
 mail_16	 power_16	 wireless0_16	 Wireless1_16	 Wireless2_16	 Wireless3_16
 batteryh0_16	 batteryh1_16	 batteryh2_16	 batteryh3_16	 batteryh4_16	 batteryhc_16
 batteryv0_16	 batteryv1_16	 batteryv2_16	 batteryv3_16	 batteryv4_16	 batteryvc_16

**IMPORTANT:** This library uses Timer2 (for AVR-based boards like Arduino Uno and Mega2560)/Timer1 (for PIC32-based boards like the chipKits)/Timer Counter 1, Channel 0 (for Arduino Due) for the status bar interrupt. This may cause conflicts with other libraries. The timer interrupt is only used while the status bar is running/active.

You can always find the latest version of the library at <http://www.RinkyDinkElectronics.com/>

For version information, please refer to `version.txt`.

This library is licensed under a [CC BY-NC-SA 3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/) (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported) License.

For more information see: <http://creativecommons.org/licenses/by-nc-sa/3.0/>

## Defined Literals:

Icon sizes
For use with the class constructor
ICONSIZE_8X8: 1
ICONSIZE_16X16: 2

Icon alignment
For use with the class constructor
ICONPOSITION_TOP_RIGHT: 0
ICONPOSITION_TOP_LEFT: 1
ICONPOSITION_BOTTOM_RIGHT: 2
ICONPOSITION_BOTTOM_LEFT: 3

## Functions:

### **OLED\_StatusIcons(OLED, size, position);**

The main class constructor.

Parameters:    OLED:        a reference to an already created OLED\_I2C (OLED) object  
                  size:        **ICONSIZE\_8X8** or **ICONSIZE\_16X16**  
                  position: **ICONPOSITION\_TOP\_RIGHT**, **ICONPOSITION\_TOP\_LEFT**, **ICONPOSITION\_BOTTOM\_RIGHT** or  
                                  **ICONPOSITION\_BOTTOM\_LEFT**

Usage:            **OLED\_StatusIcons myIcons(&myOLED, ICONSIZE\_8X8, ICONPOSITION\_BOTTOM\_LEFT); // Start an instance**

Notes:            Remember the '&' in front of the OLED object name

### **begin();**

Initialize the status bar.

Parameters:       None

Usage:            **myIcons.begin(); // Initialize the status bar**

### **updateSpeed(ms);**

Set the rate of which the status bar will update.

Parameters:       ms:    Number of milliseconds between each update (100-10000)

Usage:            **myIcons.updateSpeed(1000); // Set the update rate to the default once every 1000ms**

### **enableStatusbar();**

Enable the display of the interrupt-driven status bar.

Parameters:       None

Usage:            **myIcons.enableStatusbar(); // Enable the display of the interrupt-driven status bar**

### **disableStatusbar();**

Disable the display of the interrupt-driven status bar.

Parameters:       None

Usage:            **myIcons.disableStatusbar(); // Disable the display of the interrupt-driven status bar**

### **refreshStatusbar();**

Do an immediate refresh of the status bar.

Parameters:       None

Usage:            **myIcons.refreshStatusbar(); // Do an immediate refresh of the status bar**

Notes:            When clearing the display the status bar will also be cleared. It will not reappear until the next refresh cycle which, depending on the refresh rate, may be a while. Calling **refreshStatusbar()** will refresh/redraw the status bar immediately if the status bar is active/running.

#### **createIcon(position, bitmap [, enabled]);**

Create a new icon for the status bar

Parameters:      position: position of the new icon (0-13 for 8x8 icons and 0-6 for 16x16 icons)  
                  bitmap:    the array holding the bitmap for this icon  
                  enabled:    <optional>  
                              state when created: **true** or **false** (default)

Returns:            (boolean) **true** if the icon was successfully created, otherwise **false**

Usage:             myIcons.createIcon(0, power\_8, true); // Create a new icon in the position closest to the chosen edge and enable it

Notes:             Position 0 is always closest to your chosen edge of alignment.  
                  Creating a new icon in a position where there already is an icon will fail.

#### **deleteIcon(position);**

Delete an icon.

Parameters:      position: position of the icon to delete

Returns:            (boolean) **true** if the icon was successfully deleted, otherwise **false**

Usage:             myIcons.deleteIcon(0); // Delete the icon in position 0

#### **enableIcon(position);**

Enable/show an icon.

Parameters:      position: position of the icon to enable/show

Usage:             myIcons.enableIcon(0); // Enable/show the icon in position 0

#### **disableIcon(position);**

Disable/hide an icon.

Parameters:      position: position of the icon to disable/hide

Usage:             myIcons.disableIcon(0); // Disable/show the icon in position 0

#### **changeIcon(position, bitmap);**

Change the bitmap for an existing icon.

Parameters:      position: position of the icon that will have its bitmap replaced  
                  bitmap:    the array holding the new bitmap for this icon

Usage:             myIcons.changeIcon(0, mail\_8); // Change the bitmap for the icon in position 0