SCD55100A Arduino Library by CANADUINO®

Arduino library for the Osram/Siemens SCD5510xA series 10-digit 5×5 LED dot matrix displays.

Overview

This library provides complete control over the SCD5510xA vintage LED display, including text output, custom patterns, brightness control, and scrolling text. Compatible with any Arduino board.

Compatible Displays

- **SCD55100A** - Red
- **SCD55101A** - Yellow
- **SCD55102A** - High Efficiency Red (HER)
- **SCD55103A** - Green
- **SCD55104A** - High Efficiency Green (HEG)

Features

- Simple text display with built-in 5×5 font
- Custom character and pattern support
- Scrolling text with adjustable speed
- Seven brightness levels (6% to 100%)
- Direct pixel control for custom graphics
- Low power consumption for battery operation
- Hardware reset support
- Lamp test and power-down modes
- Software “display clear” function

Hardware Requirements

Components

- Arduino-compatible board
- SCD55100A series LED display
- Power supply 4.5-5.5V
- Connecting wires

Pin Configuration

The display requires 4 control signals from your Arduino:

Display Pin	Function	Description
1	SDCLK	Serial Data Clock
2	/LOAD	Load/Latch Signal
13	/RST	Reset
27	DATA	Serial Data Input
16	CLKSEL	Clock Select - Connect to VCC
14, 28	GND	Ground connections
19	VCC	Power supply

You can connect these to any available digital pins on your Arduino.

Installation

Arduino IDE

1. Download the library as a ZIP file
2. In Arduino IDE, go to **Sketch → Include Library → Add .ZIP Library**
3. Select the downloaded ZIP file
4. Restart Arduino IDE

Manual Installation

1. Download the library
2. Extract to your Arduino libraries folder:
    - **Windows:** Documents\Arduino\libraries\
    - **macOS:** ~/Documents/Arduino/libraries/
    - **Linux:** ~/Arduino/libraries/
3. Restart Arduino IDE

Wiring Diagram

Arduino Board		SCD55100A Display
Digital Pin (SDCLK)	→	Pin 1 (SDCLK)
Digital Pin (LOAD)	→	Pin 2 (LOAD)
Digital Pin (RESET)	→	Pin 13 (RST)
Digital Pin (DATA)	→	Pin 27 (DATA)
VCC	→	Pin 16 (CLKSEL)
VCC	→	Pin 19 (VCC)
GND	→	Pin 14,28 (GND)

Quick Start

Basic Example

```
#include <SCD55100A.h>

// Define your pin connections
#define SDCLK_PIN 4   // Choose any digital pin
#define LOAD_PIN  5   // Choose any digital pin
#define DATA_PIN  3   // Choose any digital pin
#define RST_PIN   13  // Choose any digital pin

// Create display object
SCD55100A display(SDCLK_PIN, LOAD_PIN, DATA_PIN, RST_PIN);

void setup() {
  display.begin();
  display.setBrightness(SCD_BRIGHTNESS_40);
  display.print("HELLO");
}

void loop() {
  // Your code here
}
```

Display Text

```
void setup() {
  display.begin();
```

```
  // Display text (left-aligned)
  display.print("HELLO");

  // Display text at specific position
  display.printAt(2, "WORLD");

  // Display single character
  display.printChar(5, 'A');
}
```

Scrolling Text

```
void loop() {
  // Scroll text across display
  // Parameters: text, delay in milliseconds
  display.scrollText("WELCOME TO ARDUINO", 100);
  delay(1000);
}
```

Custom Patterns

```
void setup() {
  display.begin();

  // Define custom 5×5 pattern (smiley face)
  uint8_t smiley[5] = {
    0b01010,
    0b01010,
    0b00000,
    0b10001,
    0b01110
  };

  // Display on digit 4
  display.writeDigit(4, smiley);
}
```

Brightness Control

```
void setup() {
  display.begin();

  // Set brightness (7 levels available)
  display.setBrightness(SCD_BRIGHTNESS_100);  // 100%
  display.setBrightness(SCD_BRIGHTNESS_53);   // 53%
  display.setBrightness(SCD_BRIGHTNESS_40);   // 40%
  display.setBrightness(SCD_BRIGHTNESS_27);   // 27%
  display.setBrightness(SCD_BRIGHTNESS_20);   // 20%
  display.setBrightness(SCD_BRIGHTNESS_13);   // 13%
  display.setBrightness(SCD_BRIGHTNESS_6);    // 6%
}
```

API Reference

Constructor

SCD55100A(uint8_t sdclkPin, uint8_t loadPin, uint8_t dataPin, uint8_t rstPin)

Creates a new SCD55100A display object with specified pins.

Initialization

void begin()

Initializes the display. Must be called in setup().

void reset()

Performs hardware reset of the display.

Display Control

void clear()

Clears all content from the display.

void setBrightness(uint8_t level)

Sets display brightness. Use predefined constants:

- SCD_BRIGHTNESS_100 - 100% brightness
- SCD_BRIGHTNESS_53 - 53% brightness
- SCD_BRIGHTNESS_40 - 40% brightness
- SCD_BRIGHTNESS_27 - 27% brightness
- SCD_BRIGHTNESS_20 - 20% brightness
- SCD_BRIGHTNESS_13 - 13% brightness
- SCD_BRIGHTNESS_6 - 6% brightness

void powerDown()

Puts display into low-power mode.

void lampTest()

Activates lamp test mode (all LEDs on).

Text Functions

void print(const char* text)

Displays text starting from the leftmost digit.

void printAt(uint8_t digit, const char* text)

Displays text starting at specified digit position (0-9).

void printChar(uint8_t digit, char c)

Displays a single character at specified digit position.

void scrollText(const char* text, uint16_t delayMs = 100)

Scrolls text across the display with specified delay between steps.

Custom Graphics

void writeDigit(uint8_t digit, const uint8_t pattern[5])

Writes a 5×5 pattern to a single digit (0-9).

void writeDigits(const uint8_t patterns[][5], uint8_t count)

Writes multiple 5×5 patterns starting from digit 0.

void setPixel(uint8_t digit, uint8_t row, uint8_t col, bool on)

Sets individual pixel state. Requires `updateDisplay()` to take effect.

void updateDisplay()

Updates the display with buffered pixel changes.

Low-Level Functions

void writeCommand(uint8_t cmd)

Sends a command byte to the display.

void writeByte(uint8_t data)

Sends a data byte to the display.

Examples

The library includes an example sketch that show:

- **BasicText** - Simple text display
- **ScrollingText** - Scrolling message demo
- **CustomPatterns** - Custom 5×5 graphics
- **BrightnessControl** - Brightness adjustment
- **Counter** - Numeric counter display
- **PixelControl** - Individual pixel manipulation

Power Consumption

Typical current consumption (varies by display model and color):

- **Active (100% brightness):** ~150mA
- **Active (20% brightness):** ~40mA
- **Power-down mode:** <1mA

For battery-powered applications, use lower brightness settings and implement power-down during inactive periods to extend battery life.

Clock Speed Compatibility

This library works with **any Arduino clock speed** (1MHz, 8MHz, 16MHz, etc.) since it uses software-controlled serial communication. The timing is independent of the microcontroller's clock frequency.

The example configuration uses ATmega328P-PU at 8MHz internal clock for low-power battery operation, but this is optional.

Troubleshooting

Display not responding

- Verify all pin connections
- Check power supply voltage (4.5V nominal)
- Ensure CLKSEL pin is connected to VCC
- Try hardware reset with `display.reset()`

Dim or flickering display

- Check power supply voltage (ensure it meets display specs)
- Adjust brightness: `display.setBrightness(SCD_BRIGHTNESS_40)`
- Verify ground connections are solid

Garbled characters

- Check DATA and SDCLK connections
- Verify LOAD pin connection
- Ensure stable power supply

Some digits not working

- Check for faulty display chip
- Test with lamp test: `display.lampTest()`
- Try different brightness levels

Technical Specifications

- **Display Type:** 5×5 LED dot matrix
- **Number of Digits:** 10
- **Supply Voltage:** Refer to display datasheet (typically 4.5-5.5V)
- **Logic Voltage:** 5V TTL compatible
- **MCU Clock Speed:** Any (1MHz to 16MHz+)
- **Interface:** Serial (3-wire + reset)
- **Brightness Levels:** 7 (6% to 100%)
- **Operating Temperature:** 0°C to +70°C

License

This library is released under the MIT License. See LICENSE file for details.

Credits

Library developed for vintage Osram/Siemens SCD55100A displays. Based on original datasheet specifications.

Contributing/Support

Contributions and inquiries are welcome! Just send an email to the author.

Version History

v1.0.0 (2025)

- Initial release
- Basic text display functions
- Custom pattern support
- Scrolling text
- Brightness control
- Power management
- Hardware reset support