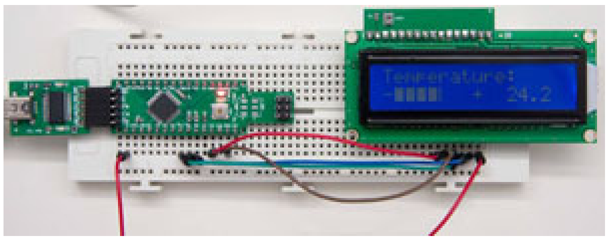
LCD Library

Index

Introduction



Welcome to the *LCD Library* for **Arduino** and **Chipkit**. It is a derivate of the original LiquidCrystal Library as sourced in the Arduino SDK. It has been developed to be compatible with the current LiquidCrystal library, its performance is almost 5 times faster and fully extendable if need be.

Being faster, gives your applications more time to do more things than just controlling the LCD. So, its cool, you can do more stuff.

It supports most Hitachi HD44780 based LCDs, or compatible, connected to any project using: 4, 8 wire parallel interface, I2C IO port expander (native I2C and bit bang) and Shift Regiter.

It currently supports 4 types of connections:

- 4 bit parallel LCD interface
- 8 bit parallel LCD interface
- I2C IO bus expansion board with the PCF8574* I2C IO expander ASIC such as I2C LCD extra IO.
- ShiftRegister adaptor board as described Shift Register project home or in the HW configuration described below, 2 and 3 wire configurations supported.
- ShiftRegister 3 wire latch adaptor board as described ShiftRegister 3 Wire Home
- Support for 1 wire shift register ShiftRegister 1 Wire
- I2C bus expansion using general purpose IO lines.

Library Overview

This library provides the same interface to applications as the LiquidCrystal library sourced by the Arduino SDK. The main changes to the LiquidCrystal Library is that it has been changed to be a pure abstract class from which particular implementations derive from.

All the basic functions to control an LCD are implemented in the base class, while the particular way to "talk" to the LCD is done by a class that simply knows how to write to the LCD.

Therefore, it is possible to create new drivers to the library by simply inhering from the base class and develop the functions that are specific to "talk" to the LCD.

The library currently supports 3 types of connections:

- 4 bit parallel LCD interface
- 8 bit parallel LCD interface
- I2C IO bus expansion board with the PCF8574* I2C IO expander ASIC.
- ShiftRegister adaptor board (please visit the HW schematics and configuration for details).

The *LCD library* started as a base support for the PCF8574* I2C IO expander ASIC in the**LCDI2CextraIO** board but it has rapidly grown to support other LCD driving mechanism due to the simplicity of adding new drivers to it.

Since this library is a full class hierarchy, new interfacing mechanisms can be added without having to re-write the entire driver. Drivers for the MCP2300, SPI and Serial are very easy to develop since you would only have to worry about how to write to the particular new device.

Usage

The library is used just like the current stock LiquidCrystal LCD library. You only have to tell it what type of LCD and how you have connected to your project and you are set. The main difference is that the *LCD library* is a "collection" of libraries with the same common interface (a class hierarchy with a base abstract class - in the technical jargon).

Here is how to use the library for a 4 bit LCD interface:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define   CONTRAST_PIN   9
#define   BACKLIGHT_PIN  7
#define   CONTRAST       125

void setup()
{
  // Switch on the backlight and LCD contrast levels
  pinMode(CONTRAST_PIN, OUTPUT);
  pinMode(BACKLIGHT_PIN, OUTPUT);

  digitalWrite(BACKLIGHT_PIN, HIGH);
  analogWrite (CONTRAST_PIN, CONTRAST);

  lcd.begin(16,2);               // initialize the lcd

  lcd.home ();                   // go home
  lcd.print("Hello, ARDUINO ");
  lcd.setCursor ( 0, 1 );        // go to the next line
  lcd.print (" WORLD!");
}

void loop()
{

}
```

Here's of how to use the library for a I2C LCD interface using the I2CLCDextraIO board or similar:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x38);  // Set the LCD I2C address

#define BACKLIGHT_PIN     13

void setup()
{
  // Switch on the backlight
  pinMode ( BACKLIGHT_PIN, OUTPUT );
  digitalWrite ( BACKLIGHT_PIN, HIGH );

  lcd.begin(16,2);               // initialize the lcd

  lcd.home ();                   // go home
  lcd.print("Hello, ARDUINO ");
  lcd.setCursor ( 0, 1 );        // go to the next line
  lcd.print (" WORLD!  ");
}

void loop()
{

}
```

Being a "collection" of libraries with a common interface, you can develop a complete project just by using a reference to the base LCD class (a pointer to an "LCD variable") and then in the project (sketch) and during the initialization (setup routine) you would just have to pass the "concrete" LCD that you are using.

This is particularly useful for projects that have a MMI built-in where you would like to use it with multiple LCDs maintaining the user interface hidden from the particular LCD you are using. It is also very useful if you have a project and you don't know how you are going to connect the LCD or your change your mind halfway through the project.

Using the virtual LCD class in projects:

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

#define BACKLIGHT_PIN     13

LiquidCrystal_I2C lcd(0x38);  // Set the LCD I2C address
LCD *myLCD = &lcd;

void setup()
{
  // Switch on the backlight
  pinMode ( BACKLIGHT_PIN, OUTPUT );
  digitalWrite ( BACKLIGHT_PIN, HIGH );

  myLcd->begin(16,2);               // initialize the lcd

  myLcd->home ();                   // go home
  myLcd->print("Hello, ARDUINO ");
  myLcd->setCursor ( 0, 1 );        // go to the next line
  myLcd->print (" WORLD!  ");
}

void loop()
{

}
```

The only difference is the "include" and how you initialize the LCD. Not a bit difference, right!

When you install the library it may come out with a compilation error. This is due to the fact that it can't find the Wire library header files. Please include Wire.h at the beginning of your sketch. Why? Because of the peculiarities of the Arduino compilation environment. I am working on a way to work around this.

The library is configured by default to run fast. Should you have problems with slow LCDs, you will have to disable the *FAST_MODE* in the LCD.h header file of the library by simply commenting the *FAST_MODE*define.

Tests

All the HW configurations described in the HW section have been tested, the last one was the ShiftRegister adaptor by piccaso and the 1 wire interface by Stephen Erisman, thanks for sharing and testing.

The code has been fully tested with a 4 bit interface and with the **I2CLCDextraIO** (companion board to this library) using the PCF8574* ASIC.

Performance and Benchmakrs

Each supported and tested library class has gone through a performance benchmark and it is compared with respect to the original LiquidCrystal library.

AVR bench marks:

- AVR @ 16MHz
- I2C @ 100KHz

ChipKit Uno32:

- PIC32 @ 80MHz

ShiftRegister class benched marked by piccaso - flo, thanks for testing it and sharing.

Benchmark 1: write every position of the LCD one by one, 10 cycles.

Benchmark 2: write a bar graph using the entire LCD to draw the bars. This has 5 writes per LCD position, 10 cycles.

Benchmark 3: write a string to each row of the LCD from a string in RAM, 10 cycles.

Benchmark 4: write a string to each row of the LCD from a string stored in FLASH, 10 cycles.

Benchmark Results

The execution time is based on the sample benchmark file using FAST_MODE enabled.

Execution time

```
            LiquidCrystal    4BIT       ShiftRegister    ShiftRegister SR3W       I2C
benchmark1  11055 us         2966 us    2576 us          3519 us                  32062
benchmark2  125770 us        34660 us   29224 us         39839 us                 36295
benchmark3  13660 us         3616 us    3198 us          4370 us                  39613
benchmark4  13643 us         3597 us    3180 us          4352 us                  39610
```

Average LCD write operation time

```
            LiquidCrystal      4BIT       ShiftRegister      ShiftRegister SR3W     I2C
write op.   325 us             87 us      75.92 us              103.72 us           943.6
```

Performance ratio vs LiquidCrystal library

```
                LiquidCrystal      4BIT        ShiftRegister      ShiftRegister SR3W     I2C
Performance     1                  3.72        4.28               3.13                   0.34
```

Benchmark II

16MHz AVR

```
Interface     ByteXfer     16x2FPS      Ftime
----------------------------------------------
4BIT           338us        86.92     11.51ms (orignal Arduino IDE Liquid Crystal)
4BIT            98us       298.58      3.35ms
SR2W            76us       388.62      2.57ms
SR_2W           72us       406.90      2.46ms
SR_3W           61us       480.03      2.08ms
SR3W           102us       287.92      3.47ms
I2C            957us        30.74     32.25ms
I2C (GPIO)     839us        35.07     28.51ms
SR1W_HC         94us       311.41      3.21ms
SR1W_SC        116us       252.83      3.96ms
```

80Mhz Pic32 (ChipKit Uno32)

```
4BIT           232uS       126.73      7.89ms (orignal mpide Liquid Crystal)
4BIT            57uS       517.41      1.93ms
SR2W            53uS       557.35      1.79ms
SR_2W           53uS       554.66      1.80ms
SR_3W           50uS       591.40      1.69ms
SR3W            56uS       524.91      1.91ms
```

HW configuration, schematics and initialization of different boards and configuration

Please visit the HW configuration schematics of all the tested configurations of the library.

HW schematics

- Hardware configurations and initialization
  - 4 bit parallel connection
    - Board layout
  - I2C connection
    - Board layout
    - Configurable I2C address modules
  - ShiftRegister connection
    - Non Latching Shift Register
    - Latch Shift Register
  - Other configurations
    - DFRobot
      - Pin configuration
    - mjkdz

Downloading and Installation

Source code for the library and documentation can be downloaded from the download section of this repository: here

The library comes in source and with examples that will get you started. Additionally you have a full description of the library in the docs folder in HTML format that you can browse.

To install the library:

- Download the most recent version of the library.
- Extract the library. It should be in a subfolder of its own (something like /myLCDSketch/libraries/LCD) in your sketch or in the library folder of the Arduino distribution.
- When you start the Arduino IDE, it should contain a new menu "LCD" with examples. (File > Examples > LCD...). It should also appear in the menu Sketch > import library.
- Alternatively you can install the library in the Arduino SDK library pool. To install and learn about libraries please follow the instructions in Library Tutorial, there is a Section in the Tutorial indicating how to install a library.

The library has been developed to replace the current Arduino library, therefore you will need to remove/backup the LiquidCrystal folder from the Arduino library folder the original LiquidCrystal library and replace it for this one. You will also potentially need to remove other LCD libraries like LiquidCrystal_I2C as that will also conflict with this library.

Also in the download section you can find the I2CIO driver library for the PCF8574* I2C IO expander ASIC.

Version

Current New LiquidCrystal is the latest zip file in the download section. You may also take the integration branch which contains the latest contributions and addtions. The integration branch is more up to date and contains the most updated resolution of defects found.

The New LiquidCrystal library has been tested and is compatible with Arduino SDK 1.0.1, 1.0 and Arduino SDK 0022.

This library has also been tested with the chipKit.

Minor changes would have to be done to be compatible with previous versions of the SDK.

Licensing

Lets keep this section very short.

New LiquidCrystal Library by F. Malpartida is licensed under of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version: GPL v3.0

This work was inspired on the Arduino SKD "LiquidCrystal" library and the "LiquidCrystal_I2C" library from Mario_H.

Have fun!

Updated 2018-04-16