vma209 Documentation

Release latest

Dec 08, 2018

Contents:

This library provides interfaces for the display and all other peripherals present on the Velleman VMA209 "Multi-function shield expansion board for Arduino". By providing this library, we aim to augment the documentation and examples provided by the manufacturer.

Please see ReadTheDocs for the latest documentation.

CHAPTER 1

Introduction

The Velleman VMA209 “Multi-function shield expansion board for Arduino” is an expansion board with the following features:

- 4 digit 7 segment LED display module.
- 4 surface mount LEDs in a parallel configuration.
- 10kΩ adjustable precision potentiometer.
- 3 independent push buttons.
- Piezo buzzer.
- DS18B20 and LM35 interface.
- Infrared receiver interface.
- Serial interface header.

In this project, we aim to augment the documentation and examples provided by the manufacturer.

CHAPTER 2

Installation

In this section we cover retrieval of the latest release or development version of the code and subsequent installation for an Arduino device.

2.1 Prerequisites

This project depends on the arduino-peripherals library:

- peripherals installation instructions.

The demo needs both the device library as well as the host library of simpleRPC:

- simpleRPC device installation instructions.
- simpleRPC host installation instructions.

2.2 Download

2.2.1 Latest release

Navigate to the latest release and either download the `.zip` or the `.tar.gz` file.

Unpack the downloaded archive.

2.2.2 From source

The source is hosted on GitHub, to install the latest development version, use the following commands.

```
git clone https://github.com/jfjlaros/vma209.git
cd vma209
git submodule init
git submodule update
```

2.3 Installation

2.3.1 Arduino IDE

In the Arduino IDE, a library can be added to the list of standard libraries by clicking through the following menu options.

- Sketch
- Import Library
- Add Library

To add the library, navigate to the downloaded folder and select the subfolder named `display`.

- Click OK.

Now the library can be added to any new project by clicking through the following menu options.

- Sketch
- Import Library
- display

2.3.2 Ino

Ino is an easy way of working with Arduino hardware from the command line. Adding libraries is also easy, simply place the library in the `lib` subdirectory.

```
cd lib
git clone https://github.com/jfjlaros/vma209.git
```

CHAPTER 3

Usage

3.1 Simple input and output devices

For more information about the simple peripherals, please see the arduino-peripherals documentation.

3.2 Display

The display constructor takes three parameters, which on this board should have the following values.

Table 1: Constructor parameters.

parameter	value
`clockPin`	7
`dataPin`	8
`latchPin`	4

The display class has the following functions.

Table 2: Functions.

function	description
`clear`	Clear the display.
`delay`	Set the refresh delay time.
`displayInt`	Display an integer value.
`refresh`	Display refresh.

3.2.1 Example

We typically initialise the display as follows.

```
#include <display.h>

Display display(7, 8, 4);
```

To refresh the display, add the following line to the `loop()` body.

```
void loop(void) {
  display.refresh();
}
```

The display can be set to a specific brightness and can be used to show signed integers.

```
display.delay(20);         // Dim the display.
display.displayInt(-123); // Show the string "-123".
```

3.2.2 Demo

A demo is provided to show the full functionality of the display and other peripherals. This demo is written in Python 3.

First make sure all dependencies are installed.

```
pip install -r requirements.txt
```

Run the demo as follows.

```
python demo.py
```

CHAPTER 4

Review

In this document, we review the Velleman VMA209 "Multi-function shield expansion board for Arduino". We cover both the hardware, the provided software and the available documentation.

4.1 Documentation

The manual does not provide much information about the board, its components or how to use it. It mainly consists of code examples that are also present in the examples archive. The examples are instructive, but do not cover all the functionality of the board. Basic things like driving the LED display are covered, but more advanced topics like dimming are not touched upon. The schema is probably the most useful document, as it lists all components and their wiring.

It would have been nice if a library was provided by the manufacturer. Instead, we put all our findings together for others to use.

4.2 Display

The 4-digit 7-segment display is controlled by two MC74HC595AD shift registers, one controlling the digit selection and one controlling the segment selection. Using shift registers instead of a dedicated LED driver IC like the MAX7221 puts the burden of refreshing and dimming the display on the microcontroller.

When the board powers up, some random data is displayed at full brightness because the shift registers are not in a well defined state after startup. This effect stops once the software refresh procedure starts.

4.2.1 Refreshing

As mentioned above, display refreshing must be done by the microcontroller. We tried to do this at first with the hardware timer library MsTimer2. This does work quite well from a performance point of view, but since timer 2 interferes with LED D4, we chose not to use this method.

Eventually, we decided to implement a software timer which needs be polled continuously from the `loop()` body. This timer returns immediately if a refresh is not needed, so other functions can still be performed in the mean time.

During each refresh cycle, each digit is turned on and off as fast as possible.

4.2.2 Dimming

The shift registers have an *output enable* pin. It would have been nice if these pins were wired to a PWM capable pin of the Arduino as this would enable us to set the brightness of the display independently from the refresh rate. Unfortunately these pins are wired to ground.

We therefore dim the display by lowering the refresh rate. After the refresh cycle, the next refresh time is stored and all subsequent refresh calls will return immediately until the refresh time is reached.

This approach has several drawbacks:

- At maximum brightness, all CPU time is spent on refreshing. So an other function that is executed in the mean time will likely influence the brightness of the display.
- If the brightness is set too low, the display will start flickering.

4.3 LEDs

Apart from the fact that the wiring is a bit awkward, the LEDs are working just fine. To enable an LED, the output pin should be LOW, to disable an LED, it should be high. This is opposite to what one might expect.

We used the `invert` parameter for LEDs in our library to make this a bit easier for the end user.

At startup, one of the LEDs (number 13) is lit at full brightness, this is because the Arduino itself has an LED attached to pin 13 which is off during the initialisation cycle. Because of the inverted behaviour of the LEDs on the shield, this LED is on by default.

4.4 Buzzer

The buzzer is driven by a PNP transistor, presumably to use the maximum capacity of the buzzer (it is rather loud). Using a PNP transistor means that a high voltage must be applied to turn off the buzzer, however, the standard `tone()` and `noTone()` functions assume that a buzzer can be turned off by applying a low voltage. Using these functions will therefore result in a loud noise when the buzzer should be silent. An NPN transistor would have been better in this case.

We work around this issue by using the `invert` parameter, like we did for the LEDs.

4.4.1 Advanced features

There are quite some ingenious ways of doing volume control and even generating polyphonic sounds using buzzers. This however requires wiring the buzzer to either two pins, as is the case for ToneAC or to use a dedicated pin, as is the case for the Arduino volume control library.

Unfortunately, none of these features are available in this setting.

4.5 Potmeter

Works fine, no comments.

4.6 Buttons

Work fine, but like most of the other components are wired a bit awkward. They read LOW if the button is pressed, HIGH otherwise. This is also mitigated by the use of the `invert` parameter.

Jumper J2 controls the pull up resistors for the buttons. If this jumper is removed, the `pullUp` parameter can be used to use the internal pull up resistors of the Arduino instead.

4.7 Conclusions

After quite some tinkering, this shield is quite usable for a wide range of projects. We hope that the library that resulted from our endeavours will spare other purchasers of this board some time.

Some obvious candidates for improving the hardware:

- Wiring the shift register OE pins to a PWM pin.
- Connecting the buzzer to pin 5 or 6 for volume control.

CHAPTER 5

Contributors

- Jeroen F.J. Laros <jlaros@fixedpoint.nl> (Original author, maintainer)

Find out who contributed:

```
git shortlog -s -e
```