max7301 Documentation

Release latest

Jul 31, 2018

Contents:

This library provides an interface for the Arduino to the MAX7301 GPIO port expander.

Please see ReadTheDocs for the latest documentation.

CHAPTER 1

Introduction

> The MAX7301 compact, serial-interfaced I/O expander (or general-purpose I/O (GPIO) peripheral) provides microprocessors with up to 28 ports. Each port is individually user configurable to either a logic input or logic output.
>
> Each port can be configured either as a push-pull logic output capable of sinking 10mA and sourcing 4.5mA, or a Schmitt logic input with optional internal pullup. Seven ports feature configurable transition detection logic, which generates an interrupt upon change of port logic level. The MAX7301 is controlled through an SPI-compatible 4-wire serial interface.
>
> —From the MAX7301 datasheet.

This library provides an API interface to the MAX7301. Additionally, an example sketch is included for serial communication with a host computer and an example host script is included for controlling the MAX7301.

CHAPTER 2

Installation

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

2.1 Download

2.1.1 Latest release

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

Unpack the downloaded archive.

2.1.2 From source

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

```
git clone https://github.com/jfjlaros/max7301.git
```

2.2 Installation

2.2.1 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 `max7301`.

- Click OK.

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

- Sketch
- Import Library
- max7301

2.2.2 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/max7301.git
```

CHAPTER 3

Usage

In this section we describe how to configure and use the API library.

3.1 Configuration

Include the header file and make a global class instance, i.e., put it outside of any function at the top of the sketch. The constructor of this class takes five variables:

position	description	abbreviation
1	clock pin	CLK
2	data in pin	DIN
3	data out pin	DOUT
4	chip select pin	CS
5	chip type	

The chip type should be `true` for the MAX7301AAX, `false` otherwise.

type	pins	value
MAX7301AAX	36	`true`
MAX7301ANI	28	`false`
MAX7301AAI	28	`false`

If, for example, we have the clock, data in, data out and chip select on pins 4, 5, 6 and 7 respectively and our chip is of type MAX7301ANI, initialise the class instance as follows:

```
#include <max7301.h>

MAX7301 max7301(4, 5, 6, 7, false);
```

After initialisation, the MAX7301 is in shutdown mode. Use the `enable()` function to make it enter normal operation mode. This is typically done in the `setup()` function.

```
max7301.enable();
```

Pins can be configured by using the `pinMode()` function which takes the pin number and a mode as arguments.

mode	description
`GPIO_OUTPUT`	logic output
`GPIO_INPUT`	logic input
`GPIO_INPUT_PULLUP`	logic input with internal pullup

For example, configure pin 12 as an input with the internal pullup resistor enabled and pin 22 as an output:

```
max7301.pinMode(12, GPIO_INPUT_PULLUP);
max7301.pinMode(22, GPIO_OUTPUT);
```

The pin configuration can be queried with the `getPinMode()` function.

```
byte result = max7301.getPinMode(12); // Should return GPIO_INPUT_PULLUP.
```

3.2 Communication

The functions `digitalRead()` and `digitalWrite()` can be used to read from a pin, or write to a pin.

```
byte result = max7301.digitalRead(12);
max7301.digitalWrite(22, HIGH);
```

The functions `digitalReadRange()` and `digitalWriteRange()` can be used to read from up to 8 consecutive pin at once. The first parameter indicates the first pin in the range. The pin states are encoded in one byte with the pin with the lowest number in its least significant position.

```
byte result = max7301.digitalReadRange(12); // Read pin 12-19.
max7301.digitalWriteRange(22, 0x01);        // Set pin 22 HIGH and 23-29 LOW.
```

3.3 Shutdown

The MAX7301 can be put in shutdown mode with the `disable()` function. In this mode, all pins are set to input and the pullup resistors are turned off.

```
max7301.disable();
```

3.4 Transition detection

The MAX7301 is capable of transition detection on pins 24 to 30. If a transition is detected, pin 31 will go high.

To set this up, the pins must be configured correctly with the `pinMode()` function and the input pins must be registered for active monitoring with the `configureTransitionDetection()` function.

First make sure pin 31 is configured as an output pin.

```
max7301.pinMode(31, GPIO_OUTPUT);
```

To configure pin 24 as input:

```
max7301.pinMode(24, GPIO_INPUT);
max7301.configureTransitionDetection(24, true);
```

Finally, activate transition detection with the `enableTransitionDetection()` function. This function must be called after every transition event to reenable transition detection.

```
max7301.enableTransitionDetection();
```

Transition detection can be disabled with the `disableTransitionDetection()` function.

3.5 Low level functions

Registers can be read with the `read()` function and written to with the `write()` function. The first parameter is the address of the register.

```
byte result = max7301.read(0x09); // First port configuration register.
max7301.write(0x09, 0x55);        // Set port 4-7 to output.
```

CHAPTER 4

Example sketch

In this sketch, we demonstrate the different capabilities of the MAX7301. We connect two buttons and one LED.

- Button 1 is connected to a normal input pin (12).
- Button 2 is connected to an input pin with transition detection (24).
- The LED is connected to pin 22.

Fig. 1: Schema for the test setup, see the MAX7301 datasheet for full installation instructions.

After compiling and uploading the sketch, connect the Arduino to a USB port and run the host side script.

```
python host.py
```

This script checks the state of button 1 every second. If it is pressed at the moment of checking, a short pulse will be send to the LED. Note that if the button was pressed and released between two consecutive checks, nothing is registered. Furthermore, if the button is pressed for a longer period, multiple pulses will be send to the LED.

For button 2, the script checks whether a transition has occurred every second and sends three short pulses to the LED if this has happened. This transition will be registered even if the button is pressed and released between two consecutive checks. Also, the release of the button is registered as a transition, so the LED will flash upon release of this button as well.

CHAPTER 5

Contributors

- Jeroen F.J. Laros (Original author, maintainer)

Find out who contributed:

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