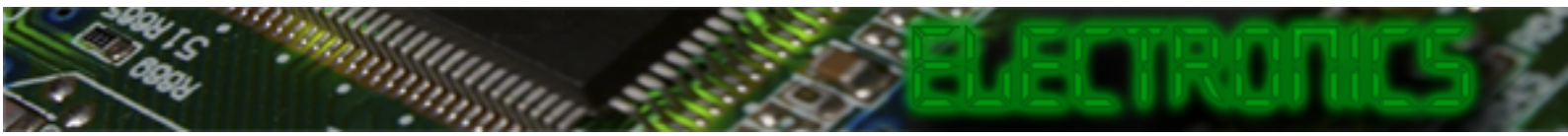
DS3231

DS3231 I^{2}C Real-Time Clock Arduino and chipKit library

Manual



INTRODUCTION:

This library has been made to easily interface and use the DS3231 RTC with an Arduino or chipKit. The library will also work with the DS3232 RTC chip but you will not be able to use the internal SRAM.

This library will default to I^2C Fast Mode (400 KHz) when using the hardware I^2C interface.

The library has not been tested in combination with the Wire library and I have no idea if they can share pins. Do not send me any questions about this. If you experience problems with pin-sharing you can move the DS3231/DS3232 SDA and SCL pins to any available pins on your development board. This library will in this case fall back to a software-based, TWI-/I^2C-like protocol which *will* require exclusive access to the pins used.

I highly recommend using the DS3231 (or DS3232) instead of the DS1307. While the DS3231/DS3232 may be slightly more expensive than the DS1307 it is much more accurate due to the internal TCXO (temperature-compensated crystal oscillator) and crystal. This also means that you don't have to use an external crystal like you have to with the DS1307.

If you are using a chipKit Uno32 or uC32 and you want to use the hardware I^2C interface you must remember to set the JP6 and JP8 jumpers to the I^2C position (closest to the analog pins).

From the DS3231 datasheet:

> The DS3231 is a low-cost, extremely accurate I^2C realtime clock (RTC) with an integrated temperaturecompensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package.
>
> The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I^2C bidirectional bus.
>
> A precision temperature-compensated voltage reference and comparator circuit monitors the status of V_{CC} to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the RST pin is monitored as a pushbutton input for generating a µP reset.

Please note that this library only makes use of the 24-hour format, and that alarms are not implemented.

You can always find the latest version of the library at
http://electronics.henningkarlsen.com/

If you make any modifications or improvements to the code, I would appreciate that you share the code with me so that I might include it in the next release. I can be contacted through **http://electronics.henningkarlsen.com/contact.php**.

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

Structures:

Time;

Structure to manipulate time- and date-data.

```
Variables:      hour, min, sec:  For holding time-data
                date, mon, year: For holding date-data
                dow:             Day-of-the-week with Monday being the first day
Usage:          Time t; // Define a structure named t of the Time-class
```

Defined Literals:

Weekdays

For use with setDOW() and Time.dow

MONDAY:	1
TUESDAY:	2
WEDNESDAY:	3
THURSDAY:	4
FRIDAY:	5
SATURDAY:	6
SUNDAY:	7

Select length

For use with getTimeStr(), getDateStr(), getDOWStr() and getMonthStr()

FORMAT_SHORT:	1
FORMAT_LONG:	2

Select date format

For use with getDateStr()

FORMAT_LITTLEENDIAN:	1
FORMAT_BIGENDIAN:	2
FORMAT_MIDDLEENDIAN:	3

Select Square Wave Output rate

For use with setSQWRate()

SQW_RATE_1:	0
SQW_RATE_1K:	1
SQW_RATE_4K:	2
SQW_RATE_8K:	3

Select Output signal for the INT/SQW pin

For use with setOutput()

OUTPUT_SQW:	0
OUTPUT_INT:	1

Functions:

DS3231(SDA, SCL);

The main class constructor.

Parameters: SDA: Pin connected to the SDA-pin of the DS3231
SCL: Pin connected to the SCL-pin of the DS3231

Usage: DS3231 rtc(SDA, SCL); // Start an instance of the DS3231 class using the hardware I^2C interface

Notes: You can connect the DS3231 to any available pin but if you use any other than hardware I^2C pin the library will fall back to a software-based, TWI-like protocol which will require exclusive access to the pins used, and you will also have to use appropriate, external pull-up resistors on the data and clock signals. External pull-up resistors are *always* needed on chipKit boards.

getTime();

Get current data from the DS3231.

Parameters: None

Returns: Time-structure

Usage: t = rtc.getTime(); // Read current time and date.

getTimeStr([format]);

Get current time as a string.

Parameters: format: **<Optional>**
FORMAT_LONG "hh:mm:ss" (default)
FORMAT_SHORT "hh:mm"

Returns: String containing the current time with or without seconds.

Usage: Serial.print(rtc.getTimeStr()); // Send the current time over a serial connection

getDateStr([slformat[, eformat[, divider]]]);

Get current date as a string.

Parameters: slformat: **<Optional>** *1
FORMAT_LONG Year with 4 digits (yyyy) (default)
FORMAT_SHORT Year with 2 digits (yy)
eformat: **<Optional>** *2
FORMAT_LITTLEENDIAN "dd.mm.yyyy" (default)
FORMAT_BIGENDIAN "yyyy.mm.dd"
FORMAT_MIDDLEENDIAN "mm.dd.yyyy"
divider: **<Optional>**
Single character to use as divider. Default is '.'

Returns: String containing the current date in the specified format.

Usage: Serial.print(rtc.getDateStr()); // Send the current date over a serial

Notes: *1: Required if you need eformat or divider.
*2: Required if you need divider. More information on Wikipedia (http://en.wikipedia.org/wiki/Date_format#Date_format).

getDOWStr([format]);

Get current day-of-the-week as a string.

Parameters: format: **<Optional>**
FORMAT_LONG Day-of-the-week in English (default)
FORMAT_SHORT Abbreviated Day-of-the-week in English (3 letters)

Returns: String containing the current day-of-the-week in full or abbreviated format.

Usage: Serial.print(rtc.getDOWStr(FORMAT_SHORT)); // Send the current day in abbreviated format over a serial connection

getMonthStr([format]);

Get current month as a string.

Parameters: format: **<Optional>**
FORMAT_LONG Month in English (default)
FORMAT_SHORT Abbreviated month in English (3 letters)

Returns: String containing the current month in full or abbreviated format.

Usage: Serial.print(rtc.getMonthStr()); // Send the current month over a serial connection

getUnixTime(time);

Convert the supplied time to the Unixtime format.

Parameters: time: A Time structure containing the time and date to convert

Returns: (long) Unixtime of the supplied Time structure

Usage: `Serial.print(rtc.getUnixTime(rtc.getTime())); // Send the current Unixtime over a serial connection`

getTemp();

Get the current temperature from the DS3231 internal thermometer.

Parameters: None

Returns: (float) Current temperature of the DS3231 chip in ° Celsius

Usage: `Serial.print(rtc.getTemp()); // Send the temperature over a serial connection`

Notes: The internal temperature is measured and updated every 64 seconds.
The temperature has a resolution of 0.25°C.

setTime(hour, min, sec);

Set the time.

Parameters: hour: Hour to store in the DS3231 (0-23)
min: Minute to store in the DS3231 (0-59)
sec: Second to store in the DS3231 (0-59)

Returns: Nothing

Usage: `rtc.setTime(23, 59, 59); // Set the time to 23:59:59`

setDate(date, mon, year);

Set the date.

Parameters: date: Date of the month to store in the DS3231 (1-31) *1
mon: Month to store in the DS3231 (1-12)
year: Year to store in the DS3231 (2000-2099)

Returns: Nothing

Usage: `rtc.setDate(4, 7, 2014); // Set the date to July 4th 2014.`

Notes: *1: No checking for illegal dates so Feb 31th is possible to input. *The effect of doing this is unknown.*

setDOW([dow]);

Set the day-of-the-week.

Parameters: dow: **<Optional>**
Day of the week to store in the DS3231 (1-7) *1

If no parameter is given the dow will be calculated from the date currently stored in the DS3231.

Returns: Nothing

Usage: `rtc.setDOW(FRIDAY); // Set the day-of-the-week to be Friday`

Notes: *1: Monday is 1, and through to Sunday being 7.

enable32KHz(enable);

Enable or disable Square Wave output on the 32kHz pin.

Parameters: `enable: TRUE enables Square Wave output, and FALSE disables it.`

Returns: `Nothing`

Usage: `rtc.enable32KHz(true); // Enable 32KHz Square Wave`

setOutput(mode);

Select what signal will be output on the INT/SQW pin.

Parameters: `mode: OUTPUT_SQW or OUTPUT_INT`

Returns: `Nothing`

Usage: `rtc.setOutput(OUTPUT_SQW); // Enable SQW output on the INT/SQW pin`

setSQWRate(rate);

Set the Square Wave output rate if the INT/SQW pin is set to output SQW.

Parameters:
```
rate: SQW_RATE_1   sets a 1Hz rate
      SQW_RATE_1K  sets a 1.024KHz rate
      SQW_RATE_4K  sets a 4.096KHz rate
      SQW_RATE_8K  sets a 8.192KHz rate
```

Returns: `Nothing`

Usage: `rtc.setSQWRate(SQW_RATE_1); // Sets the rate for SQW to 1 Hz`