

DS3231 high precision RTC library for Arduino
1.0.0

Generated by Doxygen 1.8.14

Contents

1	DS3231 high precision RTC library for Arduino	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	DS3231 Class Reference	9
5.1.1	Detailed Description	10
5.1.2	Member Function Documentation	11
5.1.2.1	alarmInterruptEnable()	11
5.1.2.2	bcdToDec()	11
5.1.2.3	begin()	11
5.1.2.4	clearAlarmFlag()	12
5.1.2.5	decToBcd()	12
5.1.2.6	getAgingOffset()	13
5.1.2.7	getAlarmFlag()	13
5.1.2.8	getDateTime()	13
5.1.2.9	getEpochTime()	15
5.1.2.10	getTemperature()	15

5.1.2.11	getTime()	16
5.1.2.12	isOscillatorStopped()	16
5.1.2.13	oscillatorEnable()	16
5.1.2.14	outputClockPinEnable()	17
5.1.2.15	readBuffer()	17
5.1.2.16	readControlRegister()	17
5.1.2.17	readRegister()	18
5.1.2.18	readStatusRegister()	18
5.1.2.19	setAgingOffset()	18
5.1.2.20	setAlarm1()	19
5.1.2.21	setAlarm2()	19
5.1.2.22	setDateTime()	20
5.1.2.23	setSquareWave()	20
5.1.2.24	setTime()	21
5.1.2.25	startTemperatureConversion()	21
5.1.2.26	writeBuffer()	22
5.1.2.27	writeControlRegister()	22
5.1.2.28	writeRegister()	22
5.1.2.29	writeStatusRegister()	23
5.2	DS3231_DateTime_s Struct Reference	23
5.2.1	Detailed Description	24
5.3	DS3231Debug Class Reference	24
5.3.1	Detailed Description	24
5.3.2	Member Function Documentation	24
5.3.2.1	dumpRegisters()	24
5.3.2.2	printDiagnostics()	25
5.3.2.3	printRegister()	25
5.3.2.4	printRegisterBitfields()	25

6 File Documentation	29
6.1 DS3231.cpp File Reference	29
6.1.1 Detailed Description	29
6.2 DS3231.h File Reference	29
6.2.1 Detailed Description	32
6.2.2 Macro Definition Documentation	32
6.2.2.1 DS3231_HOUR_12H_24H	32
6.2.2.2 DS3231_NUM_REGS	32
6.2.2.3 DS3231_REG_SECONDS	33
6.2.3 Enumeration Type Documentation	33
6.2.3.1 Alarm1Type	33
6.2.3.2 Alarm2Type	33
6.2.3.3 AlarmId	34
6.2.3.4 SquareWave	34
6.3 DS3231_debug.cpp File Reference	34
6.3.1 Detailed Description	35
6.3.2 Variable Documentation	35
6.3.2.1 PROGMEM	35
6.4 DS3231_debug.h File Reference	35
6.4.1 Detailed Description	35
Index	37

Chapter 1

DS3231 high precision RTC library for Arduino

This is a [DS3231](#) high precision RTC library for Arduino.

Hardware

Any Arduino hardware with a TWI interface.

Examples

- Examples | Erriez [DS3231](#) RTC | [GettingStarted](#)

Documentation

- [Doxygen online HTML](#)

Usage

Initialization

Library configuration

TODO

Library dependencies

- `Wire.h`

Library installation

Please refer to the [Wiki](#) page.

Other Arduino Libraries and Sketches from Erriez

- [Erriez Libraries and Sketches](#)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

DS3231	9
DS3231Debug	24
DS3231_DateTime_s	23

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

DS3231		
DS3231	DS3231 RTC base class	9
DS3231_DateTime_s		
	Date time structure	23
DS3231Debug		
DS3231	DS3231 RTC debug class	24

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

DS3231.cpp	
DS3231 high precision RTC library for Arduino	29
DS3231.h	
DS3231 high precision RTC library for Arduino	29
DS3231_debug.cpp	
DS3231 high precision RTC debug library for Arduino	34
DS3231_debug.h	
DS3231 high precision RTC debug library for Arduino	35

Chapter 5

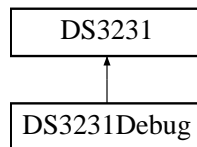
Class Documentation

5.1 DS3231 Class Reference

[DS3231](#) RTC base class.

```
#include <DS3231.h>
```

Inheritance diagram for DS3231:



Public Member Functions

- bool [begin](#) ()
Initialize and detect [DS3231](#) RTC.
- void [oscillatorEnable](#) (bool enable)
Enable or disable oscillator when running on V-BAT.
- bool [isOscillatorStopped](#) ()
Read RTC halt status.
- void [clearOscillatorStopFlag](#) ()
Clear Oscillator Stop Flag (OSF) in status register.
- void [setDateTime](#) (DS3231_DateTime *dateTime)
Write date and time to RTC.
- bool [getDateTime](#) (DS3231_DateTime *dateTime)
Read date and time from RTC.
- void [setTime](#) (uint8_t hour, uint8_t minute, uint8_t second)
Write time to RTC.
- bool [getTime](#) (uint8_t *hour, uint8_t *minute, uint8_t *second)
Read time from RTC.
- uint32_t [getEpochTime](#) (DS3231_DateTime *dateTime)
Get Unix Epoch 32-bit timestamp in current timezone.

- void [setAlarm1](#) ([Alarm1Type](#) alarmType, uint8_t dayDate, uint8_t hours, uint8_t minutes, uint8_t seconds)
Set and enable Alarm 1.
- void [setAlarm2](#) ([Alarm2Type](#) alarmType, uint8_t dayDate, uint8_t hours, uint8_t minutes)
Set and enable Alarm.
- void [alarmInterruptEnable](#) ([AlarmId](#) alarmId, bool enable)
Enable or disable Alarm 1 or 2 interrupt.
- bool [getAlarmFlag](#) ([AlarmId](#) alarmId)
Get Alarm 1 or 2 flag.
- void [clearAlarmFlag](#) ([AlarmId](#) alarmId)
Clear alarm flag.
- void [setSquareWave](#) ([SquareWave](#) squareWave)
Configure SQW (Square Wave) output pin.
- void [outputClockPinEnable](#) (bool enable)
Enable or disable 32kHz output clock pin.
- void [setAgingOffset](#) (int8_t val)
Set aging offset register.
- int8_t [getAgingOffset](#) ()
Get aging offset register.
- void [startTemperatureConversion](#) ()
Start temperature conversion.
- void [getTemperature](#) (int8_t *temperature, uint8_t *fraction)
Read temperature.
- uint8_t [bcdToDec](#) (uint8_t bcd)
BCD to decimal conversion.
- uint8_t [decToBcd](#) (uint8_t dec)
Decimal to BCD conversion.
- uint8_t [readControlRegister](#) ()
Read control register.
- void [writeControlRegister](#) (uint8_t value)
Write control register.
- uint8_t [readStatusRegister](#) ()
Read status register.
- void [writeStatusRegister](#) (uint8_t value)
Write status register.
- uint8_t [readRegister](#) (uint8_t reg)
Read register.
- void [writeRegister](#) (uint8_t reg, uint8_t value)
Write to RTC register.
- void [readBuffer](#) (uint8_t reg, void *buffer, uint8_t len)
Read buffer from RTC.
- void [writeBuffer](#) (uint8_t reg, void *buffer, uint8_t len)
Write buffer to RTC.

5.1.1 Detailed Description

[DS3231](#) RTC base class.

Definition at line 160 of file DS3231.h.

5.1.2 Member Function Documentation

5.1.2.1 alarmInterruptEnable()

```
void DS3231::alarmInterruptEnable (
    AlarmId alarmId,
    bool enable )
```

Enable or disable Alarm 1 or 2 interrupt.

Parameters

<i>alarmId</i>	Alarm1 or Alarm2 enum.
<i>enable</i>	true: Enable alarm interrupt. false: Disable alarm interrupt.

Definition at line 412 of file DS3231.cpp.

5.1.2.2 bcdToDec()

```
uint8_t DS3231::bcdToDec (
    uint8_t bcd )
```

BCD to decimal conversion.

Parameters

<i>bcd</i>	BCD encoded value.
------------	--------------------

Returns

Decimal value.

Definition at line 645 of file DS3231.cpp.

5.1.2.3 begin()

```
bool DS3231::begin ( )
```

Initialize and detect [DS3231](#) RTC.

Call this function from `setup()`.

Return values

<i>Success</i>	RTC detected.
<i>false</i>	RTC not detected.
<i>true</i>	Invalid status register or RTC not detected.

Definition at line 54 of file DS3231.cpp.

5.1.2.4 clearAlarmFlag()

```
void DS3231::clearAlarmFlag (
    AlarmId alarmId )
```

Clear alarm flag.

This function should be called when the alarm flag has been handled in polling and interrupt mode. Clearing the interrupt flag .

Parameters

<i>alarmId</i>	Alarm1 or Alarm2 enum.
----------------	------------------------

Return values

<i>Success</i>	
<i>Failure</i>	Incorrect alarm ID.

Definition at line 466 of file DS3231.cpp.

5.1.2.5 decToBcd()

```
uint8_t DS3231::decToBcd (
    uint8_t dec )
```

Decimal to BCD conversion.

Parameters

<i>dec</i>	Decimal value.
------------	----------------

Returns

BCD encoded value.

Definition at line 657 of file DS3231.cpp.

5.1.2.6 getAgingOffset()

```
int8_t DS3231::getAgingOffset ( )
```

Get aging offset register.

The aging offset register capacitance value is added or subtracted from the capacitance value that the device calculates for each temperature compensation.

Returns

val Aging offset value.

Definition at line 570 of file DS3231.cpp.

5.1.2.7 getAlarmFlag()

```
bool DS3231::getAlarmFlag (
    AlarmId alarmId )
```

Get Alarm 1 or 2 flag.

Call this function to retrieve the alarm status flag. This function can be used in polling as well as with interrupts enabled.

Parameters

<i>alarmId</i>	Alarm1 or Alarm2 enum.
----------------	------------------------

Return values

<i>true</i>	Alarm interrupt flag set.
<i>false</i>	Alarm interrupt flag cleared.

Definition at line 445 of file DS3231.cpp.

5.1.2.8 getDateTime()

```
bool DS3231::getDateTime (
    DS3231_DateTime * dateTime )
```

Read date and time from RTC.

Read all RTC registers at once to prevent a time/date register change in the middle of the register read operation.

Parameters

<i>dateTime</i>	Date and time structure.
-----------------	--------------------------

Return values

<i>false</i>	Success
<i>true</i>	An invalid date/time format was read from the RTC.

Definition at line 180 of file DS3231.cpp.

5.1.2.9 getEpochTime()

```
uint32_t DS3231::getEpochTime (
    DS3231_DateTime * dateTime )
```

Get Unix Epoch 32-bit timestamp in current timezone.

The [DS3231](#) RTC year range is valid between years 2000...2100. The time is in UTC.

Return values

<i>epoch</i>	32-bit unsigned Unix Epoch time
--------------	---------------------------------

Definition at line 288 of file DS3231.cpp.

5.1.2.10 getTemperature()

```
void DS3231::getTemperature (
    int8_t * temperature,
    uint8_t * fraction )
```

Read temperature.

Parameters

<i>temperature</i>	8-bit signed temperature in degree Celsius.
<i>fraction</i>	Temperature fraction in steps of 0.25 degree Celsius. The returned value is a decimal value to prevent floating point usage. The application should divided the fraction by 100.

Definition at line 617 of file DS3231.cpp.

5.1.2.11 getTime()

```
bool DS3231::getTime (
    uint8_t * hour,
    uint8_t * minute,
    uint8_t * second )
```

Read time from RTC.

Read hour, minute and second registers from RTC.

Parameters

<i>hour</i>	Hours 0..23.
<i>minute</i>	Minutes 0..59.
<i>second</i>	Seconds 0..59.

Return values

<i>false</i>	Success
<i>true</i>	Invalid second, minute or hour read from RTC. The time is set to zero.

Definition at line 250 of file DS3231.cpp.

5.1.2.12 isOscillatorStopped()

```
bool DS3231::isOscillatorStopped ( )
```

Read RTC halt status.

The application is responsible for checking the Oscillator Stop Flag (OSF) before reading date/time data. This function may be used to judge the validity of the date/time registers.

Return values

<i>true</i>	RTC oscillator was stopped: The date/time data is invalid. The application should synchronize and program a new date/time.
<i>false</i>	RTC oscillator is running.

Definition at line 103 of file DS3231.cpp.

5.1.2.13 oscillatorEnable()

```
void DS3231::oscillatorEnable (
    bool enable )
```

Enable or disable oscillator when running on V-BAT.

Parameters

<i>enable</i>	true: Enable RTC clock when running on V-BAT. false: Stop RTC clock when running on V-BAT. Oscillator Stop Flag (OSF) bit will be set in status register which can be read on next power-on.
---------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 72 of file DS3231.cpp.

5.1.2.14 outputClockPinEnable()

```
void DS3231::outputClockPinEnable (
    bool enable )
```

Enable or disable 32kHz output clock pin.

Parameters

<i>enable</i>	true: Enable 32kHz output clock pin. false: Disable 32kHz output clock pin.
---------------	--------------------------------------------------------------------------------

Definition at line 515 of file DS3231.cpp.

5.1.2.15 readBuffer()

```
void DS3231::readBuffer (
    uint8_t reg,
    void * buffer,
    uint8_t len )
```

Read buffer from RTC.

Parameters

<i>reg</i>	RTC register number 0x00..0x12.
<i>buffer</i>	Buffer.
<i>len</i>	Buffer length. Reading is only allowed within valid RTC registers.

Definition at line 764 of file DS3231.cpp.

5.1.2.16 readControlRegister()

```
uint8_t DS3231::readControlRegister ( )
```

Read control register.

Returns

8-bit unsigned register value

Definition at line 667 of file DS3231.cpp.

5.1.2.17 readRegister()

```
uint8_t DS3231::readRegister (
    uint8_t reg )
```

Read register.

Please refer to the RTC datasheet.

Parameters

<i>reg</i>	RTC register number 0x00..0x12.
------------	---------------------------------

Returns

value 8-bit unsigned register value.

Definition at line 708 of file DS3231.cpp.

5.1.2.18 readStatusRegister()

```
uint8_t DS3231::readStatusRegister ( )
```

Read status register.

Returns

8-bit unsigned register value

Definition at line 685 of file DS3231.cpp.

5.1.2.19 setAgingOffset()

```
void DS3231::setAgingOffset (
    int8_t val )
```

Set aging offset register.

The aging offset register capacitance value is added or subtracted from the capacitance value that the device calculates for each temperature compensation.

Parameters

<i>val</i>	Aging offset value -127..127, 0.1ppm per LSB (Factory default value: 0). Negative values increases the RTC oscillator frequency.
------------	-------------------------------------------------------------------------------------------------------------------------------------

Definition at line 542 of file DS3231.cpp.

5.1.2.20 setAlarm1()

```
void DS3231::setAlarm1 (
    Alarm1Type alarmType,
    uint8_t dayDate,
    uint8_t hours,
    uint8_t minutes,
    uint8_t seconds )
```

Set and enable Alarm 1.

Alarm 1 contains several alarm modes which can be configured with the `alarmType` parameter. Unused matches can be set to zero. The alarm interrupt must be enabled after setting the alarm, followed by clearing the alarm interrupt flag.

Parameters

<i>alarmType</i>	Alarm 1 types: Alarm1EverySecond Alarm1MatchSeconds Alarm1MatchMinutes Alarm1MatchHours Alarm1MatchDay Alarm1MatchDate
<i>dayDate</i>	Alarm match day of the week or day of the month. This depends on <code>alarmType</code> .
<i>hours</i>	Alarm match hours.
<i>minutes</i>	Alarm match minutes.
<i>seconds</i>	Alarm match seconds.

Definition at line 343 of file DS3231.cpp.

5.1.2.21 setAlarm2()

```
void DS3231::setAlarm2 (
    Alarm2Type alarmType,
    uint8_t dayDate,
    uint8_t hours,
    uint8_t minutes )
```

Set and enable Alarm.

Alarm 2 contains different alarm modes which can be configured with the `alarmType` parameter. Unused matches can be set to zero. The alarm interrupt must be enabled after setting the alarm, followed by clearing the alarm interrupt flag.

Parameters

<i>alarmType</i>	Alarm 2 types: Alarm2EveryMinute Alarm2MatchMinutes Alarm2MatchHours Alarm2MatchDay Alarm2MatchDate
<i>dayDate</i>	Alarm match day of the week or day of the month. This depends on <code>alarmType</code> .
<i>hours</i>	Alarm match hours.
<i>minutes</i>	Alarm match minutes.

Definition at line 385 of file DS3231.cpp.

5.1.2.22 setDateTime()

```
void DS3231::setDateTime (
    DS3231_DateTime * dateTime )
```

Write date and time to RTC.

Write all RTC registers at once to prevent a time/date register change in the middle of the register write operation. This function enables the oscillator and clear the Oscillator Stop Flag (OSF) in the status register.

Parameters

<i>dateTime</i>	Date time structure. Providing invalid date/time data may result in unpredictable behavior.
-----------------	---------------------------------------------------------------------------------------------

Definition at line 145 of file DS3231.cpp.

5.1.2.23 setSquareWave()

```
void DS3231::setSquareWave (
    SquareWave squareWave )
```

Configure SQW (Square Wave) output pin.

This will disable or initialize the SQW clock pin.

Parameters

<i>squareWave</i>	SquareWave configuration: Disable: SquareWaveDisable 1Hz: SquareWave1Hz 1024Hz: SquareWave1024Hz 4096Hz: SquareWave4096Hz 8192Hz: SquareWave8192Hz
-------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Return values

<i>Success</i>	
<i>Failure</i>	Incorrect squareWave.

Definition at line 496 of file DS3231.cpp.

5.1.2.24 setTime()

```
void DS3231::setTime (
    uint8_t hour,
    uint8_t minute,
    uint8_t second )
```

Write time to RTC.

Read all date/time register from RTC, update time registers and write all date/time registers to the RTC with one write operation.

Parameters

<i>hour</i>	Hours 0..23.
<i>minute</i>	Minutes 0..59.
<i>second</i>	Seconds 0..59.

Definition at line 222 of file DS3231.cpp.

5.1.2.25 startTemperatureConversion()

```
void DS3231::startTemperatureConversion ( )
```

Start temperature conversion.

Starting a conversion is only needed when the application requires temperature reads within 64 seconds, or changing the aging offset register.

Definition at line 593 of file DS3231.cpp.

5.1.2.26 writeBuffer()

```
void DS3231::writeBuffer (
    uint8_t reg,
    void * buffer,
    uint8_t len )
```

Write buffer to RTC.

Please refer to the RTC datasheet.

Parameters

<i>reg</i>	RTC register number 0x00..0x12.
<i>buffer</i>	Buffer.
<i>len</i>	Buffer length. Writing is only allowed within valid RTC registers.

Definition at line 744 of file DS3231.cpp.

5.1.2.27 writeControlRegister()

```
void DS3231::writeControlRegister (
    uint8_t value )
```

Write control register.

Parameters

<i>value</i>	8-bit unsigned register value
--------------	-------------------------------

Definition at line 676 of file DS3231.cpp.

5.1.2.28 writeRegister()

```
void DS3231::writeRegister (
    uint8_t reg,
    uint8_t value )
```

Write to RTC register.

Please refer to the RTC datasheet.

Parameters

<i>reg</i>	RTC register number 0x00..0x12.
<i>value</i>	8-bit unsigned register value.

Definition at line 727 of file DS3231.cpp.

5.1.2.29 writeStatusRegister()

```
void DS3231::writeStatusRegister (
    uint8_t value )
```

Write status register.

Parameters

<i>value</i>	8-bit unsigned register value
--------------	-------------------------------

Definition at line 694 of file DS3231.cpp.

The documentation for this class was generated from the following files:

- [DS3231.h](#)
- [DS3231.cpp](#)

5.2 DS3231_DateTime_s Struct Reference

Date time structure.

```
#include <DS3231.h>
```

Public Attributes

- `uint8_t` [second](#)
Second 0..59.
- `uint8_t` [minute](#)
Minute 0..59.
- `uint8_t` [hour](#)
Hour 0..23.
- `uint8_t` [dayWeek](#)
Day of the week (1 = Monday)
- `uint8_t` [dayMonth](#)
Day of the month 1..31.
- `uint8_t` [month](#)
Month 1..12.
- `uint16_t` [year](#)
Year 2000..2099.

5.2.1 Detailed Description

Date time structure.

Definition at line 102 of file DS3231.h.

The documentation for this struct was generated from the following file:

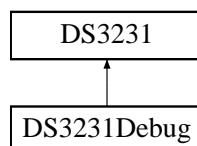
- [DS3231.h](#)

5.3 DS3231Debug Class Reference

[DS3231](#) RTC debug class.

```
#include <DS3231_debug.h>
```

Inheritance diagram for DS3231Debug:



Public Member Functions

- virtual void [dumpRegisters](#) (HardwareSerial *ser, bool printBitFields=true)
Dump all registers to serial port.
- virtual void [printRegister](#) (HardwareSerial *ser, uint8_t reg, bool printBitFields=true)
Print register and value.
- virtual void [printRegisterBitFields](#) (HardwareSerial *ser, uint8_t reg, uint8_t regVal)
Print register bitfields.
- virtual void [printDiagnostics](#) (HardwareSerial *ser)
Print diagnostics.

5.3.1 Detailed Description

[DS3231](#) RTC debug class.

The output will be redirected to a user defined serial port. This class can be used by advanced developers to print internal RTC registers and diagnostics. Keep in mind that all strings used in this class are flash consuming.

Definition at line 47 of file DS3231_debug.h.

5.3.2 Member Function Documentation

5.3.2.1 dumpRegisters()

```
void DS3231Debug::dumpRegisters (
    HardwareSerial * ser,
    bool printBitFields = true ) [virtual]
```

Dump all registers to serial port.

Parameters

<i>ser</i>	Serial port.
<i>printBitFields</i>	true: Print register bitfields.

Definition at line 75 of file DS3231_debug.cpp.

5.3.2.2 printDiagnostics()

```
void DS3231Debug::printDiagnostics (
    HardwareSerial * ser ) [virtual]
```

Print diagnostics.

Parameters

<i>ser</i>	Serial port.
------------	--------------

Definition at line 289 of file DS3231_debug.cpp.

5.3.2.3 printRegister()

```
void DS3231Debug::printRegister (
    HardwareSerial * ser,
    uint8_t reg,
    bool printBitFields = true ) [virtual]
```

Print register and value.

Parameters

<i>ser</i>	Serial port.
<i>reg</i>	Register number.
<i>printBitFields</i>	true: Print register bitfields.

Definition at line 92 of file DS3231_debug.cpp.

5.3.2.4 printRegisterBitFields()

```
void DS3231Debug::printRegisterBitFields (
    HardwareSerial * ser,
```

```
uint8_t reg,  
uint8_t regVal ) [virtual]
```

Print register bitfields.

Parameters

<i>ser</i>	Serial port.
<i>reg</i>	Register number.
<i>regVal</i>	Register value.

Definition at line 119 of file DS3231_debug.cpp.

The documentation for this class was generated from the following files:

- [DS3231_debug.h](#)
- [DS3231_debug.cpp](#)

Chapter 6

File Documentation

6.1 DS3231.cpp File Reference

DS3231 high precision RTC library for Arduino.

```
#include <pgmspace.h>
#include <Wire.h>
#include "DS3231.h"
```

Variables

- `const uint8_t daysMonth [12] PROGMEM = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}`
Define number of days in a month once in flash.

6.1.1 Detailed Description

DS3231 high precision RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231>

6.2 DS3231.h File Reference

DS3231 high precision RTC library for Arduino.

```
#include <stdint.h>
```

Classes

- struct **DS3231_DateTime_s**
Date time structure.
- class **DS3231**
***DS3231** RTC base class.*

Macros

- #define DS3231_REG_SECONDS 0x00
DS3231 registers.
- #define DS3231_REG_MINUTES 0x01
Minutes register.
- #define DS3231_REG_HOURS 0x02
Hours register.
- #define DS3231_REG_DAY_WEEK 0x03
Day of the week register.
- #define DS3231_REG_DAY_MONTH 0x04
Day of the month register.
- #define DS3231_REG_MONTH 0x05
Month register.
- #define DS3231_REG_YEAR 0x06
Year register.
- #define DS3231_REG_ALARM1_SEC 0x07
Alarm 1 seconds register.
- #define DS3231_REG_ALARM1_MIN 0x08
Alarm 1 minutes register.
- #define DS3231_REG_ALARM1_HOUR 0x09
Alarm 1 hour register.
- #define DS3231_REG_ALARM1_DD 0x0A
Alarm 1 day/date register.
- #define DS3231_REG_ALARM2_MIN 0x0B
Alarm 2 seconds register.
- #define DS3231_REG_ALARM2_HOUR 0x0C
Alarm 2 hour register.
- #define DS3231_REG_ALARM2_DD 0x0D
Alarm 2 day/date register.
- #define DS3231_REG_CONTROL 0x0E
Control register.
- #define DS3231_REG_STATUS 0x0F
Status register.
- #define DS3231_REG_AGING_OFFSET 0x10
Aging offset register.
- #define DS3231_REG_TEMP_MSB 0x11
Temperature MSB register.
- #define DS3231_REG_TEMP_LSB 0x12
Temperature LSB register.
- #define DS3231_NUM_REGS 19
DS3231 number of registers.
- #define DS3231_HOUR_12H_24H 6
DS3231 register bit defines.
- #define DS3231_HOUR_AM_PM 5
AM/PM.
- #define DS3231_MONTH_CENTURY 7
Century.
- #define DS3231_CTRL_EOSC 7
Enable oscillator.
- #define DS3231_CTRL_BBSQW 6

- *Battery-Backed Square-Wave Enable.*
- #define DS3231_CTRL_CONV 5
Start temperature conversion.
- #define DS3231_CTRL_RS2 4
Square wave rate-select 2.
- #define DS3231_CTRL_RS1 3
Square wave rate-select 1.
- #define DS3231_CTRL_INTCN 2
Interrupt control.
- #define DS3231_CTRL_A2IE 1
Alarm 2 interrupt enable.
- #define DS3231_CTRL_A1IE 0
Alarm 1 interrupt enable.
- #define DS3231_STAT_OSF 7
Oscillator Stop Flag.
- #define DS3231_STAT_EN32KHZ 3
Enable 32kHz clock output.
- #define DS3231_STAT_BSY 2
Temperature conversion busy flag.
- #define DS3231_STAT_A2F 1
Alarm 2 status flag.
- #define DS3231_STAT_A1F 0
Alarm 1 status flag.
- #define DS3231_A1M1 7
Alarm 1 bit 7 seconds register.
- #define DS3231_A1M2 7
Alarm 1 bit 7 minutes register.
- #define DS3231_A1M3 7
Alarm 1 bit 7 hours register.
- #define DS3231_A1M4 7
Alarm 1 bit 7 day/date register.
- #define DS3231_A2M2 7
Alarm 2 bit 7 minutes register.
- #define DS3231_A2M3 7
Alarm 2 bit 7 hours register.
- #define DS3231_A2M4 7
Alarm 2 bit 7 day/date register.
- #define DS3231_DYDT 6
Alarm 2 bit 6.
- #define DS3231_ADDR (0xD0 >> 1)
DS3231 I2C 7-bit address.
- #define SECONDS_FROM_1970_TO_2000 946684800
Number of seconds between year 1970 and 2000.

Typedefs

- typedef struct DS3231_DateTime_s DS3231_DateTime
Date time structure.

Enumerations

- enum `AlarmId` { `Alarm1` = 1, `Alarm2` = 2 }
- *Alarm ID.*
- enum `Alarm1Type` {
`Alarm1EverySecond` = 0x0F, `Alarm1MatchSeconds` = 0x0E, `Alarm1MatchMinutes` = 0x0C, `Alarm1MatchHours` = 0x08,
`Alarm1MatchDay` = 0x10, `Alarm1MatchDate` = 0x00 }
- *Alarm 1 types enum.*
- enum `Alarm2Type` {
`Alarm2EveryMinute` = 0x0E, `Alarm2MatchMinutes` = 0x0C, `Alarm2MatchHours` = 0x08, `Alarm2MatchDay` = 0x10,
`Alarm2MatchDate` = 0x00 }
- *Alarm 2 types enum.*
- enum `SquareWave` {
`SquareWaveDisable` = (1 << DS3231_CTRL_INTCN), `SquareWave1Hz` = ((0 << DS3231_CTRL_RS2) | (0 << DS3231_CTRL_RS1)), `SquareWave1024Hz` = ((0 << DS3231_CTRL_RS2) | (1 << DS3231_CTRL_RS1)), `SquareWave4096Hz` = ((1 << DS3231_CTRL_RS2) | (0 << DS3231_CTRL_RS1)), `SquareWave8192Hz` = ((1 << DS3231_CTRL_RS2) | (1 << DS3231_CTRL_RS1)) }
- *Squarewave enum.*

6.2.1 Detailed Description

`DS3231` high precision RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231>

6.2.2 Macro Definition Documentation

6.2.2.1 DS3231_HOUR_12H_24H

```
#define DS3231_HOUR_12H_24H 6
```

`DS3231` register bit defines.

12 or 24 hour mode

Definition at line 64 of file DS3231.h.

6.2.2.2 DS3231_NUM_REGS

```
#define DS3231_NUM_REGS 19
```

`DS3231` number of registers.

19 RTC register: 0x00..0x12

Definition at line 61 of file DS3231.h.

6.2.2.3 DS3231_REG_SECONDS

```
#define DS3231_REG_SECONDS 0x00
```

DS3231 registers.

Seconds register

Definition at line 38 of file DS3231.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 Alarm1Type

```
enum Alarm1Type
```

Alarm 1 types enum.

Enumerator

Alarm1EverySecond	Alarm once per second.
Alarm1MatchSeconds	Alarm when seconds match.
Alarm1MatchMinutes	Alarm when minutes and seconds match.
Alarm1MatchHours	Alarm when hours, minutes, and seconds match.
Alarm1MatchDay	Alarm when date, hours, minutes, and seconds match.
Alarm1MatchDate	Alarm when day, hours, minutes, and seconds match.

Definition at line 123 of file DS3231.h.

6.2.3.2 Alarm2Type

```
enum Alarm2Type
```

Alarm 2 types enum.

Enumerator

Alarm2EveryMinute	Alarm once per minute (00 seconds of every minute)
Alarm2MatchMinutes	Alarm when minutes match.
Alarm2MatchHours	Alarm when hours and minutes match.
Alarm2MatchDay	Alarm when date, hours, and minutes match.
Alarm2MatchDate	Alarm when day, hours, and minutes match.

Definition at line 136 of file DS3231.h.

6.2.3.3 AlarmId

enum [AlarmId](#)

Alarm ID.

Enumerator

Alarm1	Alarm ID 1.
Alarm2	Alarm ID 2.

Definition at line 115 of file DS3231.h.

6.2.3.4 SquareWave

enum [SquareWave](#)

Squarewave enum.

Enumerator

SquareWaveDisable	SQW disable.
SquareWave1Hz	SQW 1Hz.
SquareWave1024Hz	SQW 1024Hz.
SquareWave4096Hz	SQW 4096Hz.
SquareWave8192Hz	SQW 8192Hz.

Definition at line 148 of file DS3231.h.

6.3 DS3231_debug.cpp File Reference

[DS3231](#) high precision RTC debug library for Arduino.

```
#include "DS3231_debug.h"
```

Variables

- const char reg_0x00 [] [PROGMEM](#) = "Seconds"
Register names as string in flash.

6.3.1 Detailed Description

[DS3231](#) high precision RTC debug library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231>

6.3.2 Variable Documentation

6.3.2.1 PROGMEM

```
const char* const registerNames [ ] PROGMEM = "Seconds"
```

Register names as string in flash.

Array with all register names in flash.

Definition at line 37 of file DS3231_debug.cpp.

6.4 DS3231_debug.h File Reference

[DS3231](#) high precision RTC debug library for Arduino.

```
#include <Arduino.h>
#include "DS3231.h"
```

Classes

- class [DS3231Debug](#)
DS3231 RTC debug class.

6.4.1 Detailed Description

[DS3231](#) high precision RTC debug library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231>

Index

Alarm1Type
 DS3231.h, [33](#)

Alarm2Type
 DS3231.h, [33](#)

AlarmId
 DS3231.h, [34](#)

alarmInterruptEnable
 DS3231, [11](#)

bcdToDec
 DS3231, [11](#)

begin
 DS3231, [11](#)

clearAlarmFlag
 DS3231, [12](#)

DS3231, [9](#)
 alarmInterruptEnable, [11](#)
 bcdToDec, [11](#)
 begin, [11](#)
 clearAlarmFlag, [12](#)
 decToBcd, [12](#)
 getAgingOffset, [13](#)
 getAlarmFlag, [13](#)
 getDateTime, [13](#)
 getEpochTime, [15](#)
 getTemperature, [15](#)
 getTime, [15](#)
 isOscillatorStopped, [16](#)
 oscillatorEnable, [16](#)
 outputClockPinEnable, [17](#)
 readBuffer, [17](#)
 readControlRegister, [17](#)
 readRegister, [18](#)
 readStatusRegister, [18](#)
 setAgingOffset, [18](#)
 setAlarm1, [19](#)
 setAlarm2, [19](#)
 setDateTime, [20](#)
 setSquareWave, [20](#)
 setTime, [21](#)
 startTemperatureConversion, [21](#)
 writeBuffer, [21](#)
 writeControlRegister, [22](#)
 writeRegister, [22](#)
 writeStatusRegister, [23](#)

DS3231.cpp, [29](#)

DS3231.h, [29](#)
 Alarm1Type, [33](#)
 Alarm2Type, [33](#)
 AlarmId, [34](#)
 DS3231_HOUR_12H_24H, [32](#)
 DS3231_NUM_REGS, [32](#)
 DS3231_REG_SECONDS, [32](#)
 SquareWave, [34](#)
DS3231_DateTime_s, [23](#)
DS3231_HOUR_12H_24H
 DS3231.h, [32](#)
DS3231_NUM_REGS
 DS3231.h, [32](#)
DS3231_REG_SECONDS
 DS3231.h, [32](#)
DS3231_debug.cpp, [34](#)
 PROGMEM, [35](#)
DS3231_debug.h, [35](#)
DS3231Debug, [24](#)
 dumpRegisters, [24](#)
 printDiagnostics, [25](#)
 printRegister, [25](#)
 printRegisterBitfields, [25](#)
decToBcd
 DS3231, [12](#)
dumpRegisters
 DS3231Debug, [24](#)

getAgingOffset
 DS3231, [13](#)

getAlarmFlag
 DS3231, [13](#)

getDateTime
 DS3231, [13](#)

getEpochTime
 DS3231, [15](#)

getTemperature
 DS3231, [15](#)

getTime
 DS3231, [15](#)

isOscillatorStopped
 DS3231, [16](#)

oscillatorEnable
 DS3231, [16](#)

outputClockPinEnable
 DS3231, [17](#)

PROGMEM
 DS3231_debug.cpp, [35](#)

printDiagnostics

- DS3231Debug, [25](#)
- printRegister
 - DS3231Debug, [25](#)
- printRegisterBitFields
 - DS3231Debug, [25](#)
- readBuffer
 - DS3231, [17](#)
- readControlRegister
 - DS3231, [17](#)
- readRegister
 - DS3231, [18](#)
- readStatusRegister
 - DS3231, [18](#)
- setAgingOffset
 - DS3231, [18](#)
- setAlarm1
 - DS3231, [19](#)
- setAlarm2
 - DS3231, [19](#)
- setDateTime
 - DS3231, [20](#)
- setSquareWave
 - DS3231, [20](#)
- setTime
 - DS3231, [21](#)
- SquareWave
 - DS3231.h, [34](#)
- startTemperatureConversion
 - DS3231, [21](#)
- writeBuffer
 - DS3231, [21](#)
- writeControlRegister
 - DS3231, [22](#)
- writeRegister
 - DS3231, [22](#)
- writeStatusRegister
 - DS3231, [23](#)