

Erriez DS3231 high precision I2C RTC library for Arduino
2.0.0

Generated by Doxygen 1.8.13

Contents

1	DS3231 high precision I2C RTC library for Arduino	1
2	Class Index	9
2.1	Class List	9
3	File Index	11
3.1	File List	11
4	Class Documentation	13
4.1	ErriezDS3231 Class Reference	13
4.1.1	Detailed Description	14
4.1.2	Member Function Documentation	14
4.1.2.1	alarmInterruptEnable()	14
4.1.2.2	bcdToDec()	15
4.1.2.3	begin()	15
4.1.2.4	clearAlarmFlag()	16
4.1.2.5	clockEnable()	16
4.1.2.6	decToBcd()	17
4.1.2.7	getAgingOffset()	17
4.1.2.8	getAlarmFlag()	17
4.1.2.9	getEpoch()	18
4.1.2.10	getTemperature()	18
4.1.2.11	getTime()	19
4.1.2.12	isRunning()	19
4.1.2.13	outputClockPinEnable()	20

4.1.2.14	read()	20
4.1.2.15	readBuffer()	20
4.1.2.16	readRegister()	21
4.1.2.17	setAgingOffset()	21
4.1.2.18	setAlarm1()	22
4.1.2.19	setAlarm2()	23
4.1.2.20	setDateTime()	23
4.1.2.21	setEpoch()	24
4.1.2.22	setSquareWave()	24
4.1.2.23	setTime()	25
4.1.2.24	startTemperatureConversion()	25
4.1.2.25	write()	26
4.1.2.26	writeBuffer()	26
4.1.2.27	writeRegister()	27
5	File Documentation	29
5.1	src/ErriezDS3231.cpp File Reference	29
5.1.1	Detailed Description	29
5.2	src/ErriezDS3231.h File Reference	30
5.2.1	Detailed Description	33
5.2.2	Macro Definition Documentation	33
5.2.2.1	DS3231_HOUR_12H_24H	33
5.2.2.2	DS3231_NUM_REGS	33
5.2.2.3	DS3231_REG_SECONDS	33
5.2.3	Enumeration Type Documentation	33
5.2.3.1	Alarm1Type	33
5.2.3.2	Alarm2Type	34
5.2.3.3	AlarmId	34
5.2.3.4	SquareWave	34
	Index	37

Chapter 1

DS3231 high precision I2C RTC library for Arduino

This is a DS3231 high precision I2C RTC library for Arduino.

Library features

- Set / get time
- Set date / time with `struct tm`
- Set / get Unix Epoch UTC 32-bit timestamp `time_t`
- Day of the week calculation
- Read temperature (0.25 degree resolution)
- Alarm 1 (second/minute/hour/day/date match)
- Alarm 2 (minute/hour/day/date match)
- Polling and Alarm `INT`/`SQW` interrupt pin
- Control 32kHz out signal (enable/disable)
- Control `SQW` signal (disable/1/1024/4096/8192Hz)
- Configure aging offset
- Serial terminal interface
- Full RTC register access
- Set date/time over serial with Python script

Hardware

Any Arduino hardware with a TWI interface and `Wire.h` support.

ESP32 notes

ESP32 problem: The Arduino IDE | Board manager installs an old version 1.0.0 from https://dl.espressif.com/dl/package_esp32_index.json which contains a broken I2C repeated start. Generating a repeated start with `Wire.endTransmission(false)`; results in reading zero's from any I2C device and is not a problem of this library.

Solution: Use the Git master branch (or a newer release when available) to solve this problem as described on: <https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/windows.md>.

Pins

Pins board - DS3231	VCC	GND	SDA	SCL	SQW
Arduino UNO (ATMega328 boards)	5V	GND	A4	A5	D2 (INT0)
Arduino Mega2560	5V	GND	D20	D21	D2 (INT4)
Arduino Leonardo	5V	GND	D2	D3	D7 (INT6)
Arduino DUE (ATSAM3X8E)	3V3	GND	20	21	2
ESP8266	3V3	GND	GPIO4 (D2)	GPIO5 (D1)	GPIO0 (D3)
ESP32	3V3	GND	GPIO21	GPIO22	GPIO0

Note: Tested ESP8266 / ESP32 boards:

- **ESP8266 boards:** ESP12E / WeMos D1 & R2 / Node MCU v2 / v3
- **ESP32 boards:** WeMos LOLIN32 / LOLIN D32

Other unlisted MCU's may work, but are not tested.

Examples

Arduino IDE | Examples | Erriez DS3231 RTC:

- [AgingOffset](#) Aging offset programming.
- [AlarmInterrupt](#) Alarm with interrupts.
- [AlarmPolling](#) Alarm polled.
- [DateStrings](#) Date strings in flash example.
- [Read](#) Simple RTC read example.
- [ReadTimeInterrupt](#) Read time with 1Hz SQW interrupt. (Highly recommended)
- [SetDateTime](#) Set date time. (Must be started first)
- [Temperature](#) Temperature.
- [Terminal](#) Advanced terminal interface with [set date/time Python](#) script.
- [Test](#) Regression test.

Documentation

- [Doxygen online HTML](#)
- [Doxygen PDF](#)
- [DS3231 datasheet](#)

Usage

Initialization

```
{c++}
#include <Wire.h>
#include <ErriezDS3231.h>

// Create DS3231 RTC object
DS3231 rtc;

void setup()
{
    // Initialize TWI with a 100kHz (default) or 400kHz clock
    Wire.begin();
    Wire.setClock(400000);

    // Initialize RTC
    while (!ds3231.begin()) {
        // Error: Could not detect DS3231 RTC, retry after some time
        delay(3000);
    }
}
```

Check oscillator status at startup

```
{c++}
// Check oscillator status
if (ds3231.isOscillatorStopped()) {
    // Error: DS3231 RTC oscillator stopped. Date/time cannot be trusted.
    // Set new date/time before reading date/time.

    // Start oscillator with date/time "Sun Jan 1 2000 0:00:00"
    ds3231.oscillatorEnable(true);
}
```

Set time

```
{c++}
// Write time to RTC
if (!ds3231.setTime(12, 0, 0)) {
    // Error: Write time failed
}
```

Get time

```
{c++}
uint8_t hour;
uint8_t minute;
uint8_t second;

// Read time from RTC
if (!ds3231.getTime(&hour, &minute, &second)) {
    // Error: Read time failed
}
```

Set date time

```
{c++}
// Create and initialize date time object
struct tm dt;

dt.tm_sec = 0,      // Seconds 0..59
dt.tm_min = 36,     // Minutes 0..59
dt.tm_hour = 21,    // Hours 0..23
dt.tm_wday = 6;     // Day of the week: 0 = Sunday .. 6=Saturday
dt.tm_mday = 29;    // Day of the month 1..31
dt.tm_mon = 7;      // Month 0..11
dt.tm_year = 2020-1900; // Year starting at 1900

// Write date/time to RTC
if (!ds3231.write(&dt)) {
    // Error: Write date/time failed
}
```

Get date time

```
{c++}
struct tm dt;

// Read date/time from RTC
if (!ds3231.read(&dt)) {
    // Error: Read date/time failed
}
```

Get Epoch Unix UTC time

```
{c++}
// Get Unix epoch UTC time
time_t t = ds3231.getEpoch();
```

Get temperature

```
{c++}
int8_t temperature = 0;
uint8_t fraction = 0;

// Force temperature conversion
// Without this call, it takes 64 seconds before the temperature is updated.
if (!ds3231.startTemperatureConversion()) {
    // Error: Start temperature conversion failed
}

// Read temperature
if (!ds3231.getTemperature(&temperature, &fraction)) {
    // Error: Get temperature failed
}

// Print temperature. The output below is for example: 28.25C
Serial.print(temperature);
Serial.print(F(" "));
Serial.print(fraction);
Serial.println(F("C"));
```

Program Alarm 1

Note: Alarm 1 and Alarm 2 have different behavior. Please refer to the documentation which Alarm1Type and Alarm2Type are supported. Some examples:

```
{c++}
// Generate alarm 1 every second
ds3231.setAlarm1(Alarm1EverySecond, 0, 0, 0, 0);

// Generate alarm 1 every minute and second match
ds3231.setAlarm1(Alarm1EverySecond, 0, 0, 45, 30);

// Generate alarm 1 every day, hour, minute and second match
ds3231.setAlarm1(Alarm1MatchDay,
    1, // Alarm day match (1 = Monday)
    12, // Alarm hour match
    45, // Alarm minute match
    30 // Alarm second match
);
```


Program Alarm 2

```
{c++}
// Generate alarm 2 every minute
ds3231.setAlarm2(Alarm2EveryMinute, 0, 0, 0);

// Generate alarm 2 every hour, minute match
ds3231.setAlarm2(Alarm2MatchHours, 0, 23, 59);

// Generate alarm 2 every date, hour, minute match
ds3231.setAlarm2(Alarm2MatchDate, 28, 7, 0);
```

Alarm polling

Note: The INT pin changes to low when an Alarm 1 or Alarm 2 match occurs and the interrupt is enabled. The pin remains low until both alarm flags are cleared by the application.

```
{c++}
// Poll alarm 1 flag
if (ds3231.getAlarmFlag(Alarm1)) {
    // Handle Alarm 1

    // Clear alarm 1 flag
    ds3231.clearAlarmFlag(Alarm1);
}

// Poll alarm 2 flag
if (ds3231.getAlarmFlag(Alarm2)) {
    // Handle Alarm 2

    // Clear alarm 2 flag
    ds3231.clearAlarmFlag(Alarm2);
}
```

Alarm interrupt

Note: Enabling interrupt will disable the SQW output signal.

```
{c++}
// Uno, Nano, Mini, other 328-based: pin D2 (INT0) or D3 (INT1)
#define INT_PIN 2

// Alarm interrupt flag must be volatile
volatile bool alarmInterrupt = false;

#ifdef ARDUINO_ARCH_ESP8266 || defined(ARDUINO_ARCH_ESP32)
ICACHE_RAM_ATTR
#endif
void alarmHandler()
{
    // Set global interrupt flag
    alarmInterrupt = true;
}

void setup()
{
    ...

    // Attach to INT0 interrupt falling edge
    pinMode(INT_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(INT_PIN), alarmHandler, FALLING);

    // Enable Alarm 1 and 2 interrupts
    ds3231.alarmInterruptEnable(Alarm1, true);
    ds3231.alarmInterruptEnable(Alarm2, true);
}

void loop()
{
    // Check global alarm interrupt flag
    if (alarmInterrupt) {
        if (ds3231.getAlarmFlag(Alarm1)) {
            // Handle alarm 1

            // Clear alarm 1 interrupt
        }
    }
}
```

```

        ds3231.clearAlarmFlag(Alarm1);
    }

    if (ds3231.getAlarmFlag(Alarm2)) {
        // Handle alarm 2

        // Clear alarm 2 interrupt
        ds3231.clearAlarmFlag(Alarm2);
    }
}
}

```

32kHz clock out

Enable or disable 32kHz output pin.

```

{c++}
ds3231.outputClockPinEnable(true); // Enable
ds3231.outputClockPinEnable(false); // Disable

```

Square Wave Out (SQW)

Note: Enabling SQW pin will disable the alarm INT signal.

```

{c++}
ds3231.setSquareWave(SquareWaveDisable); // Disable
ds3231.setSquareWave(SquareWave1Hz); // 1Hz
ds3231.setSquareWave(SquareWave1024Hz); // 1024Hz
ds3231.setSquareWave(SquareWave4096Hz); // 4096Hz
ds3231.setSquareWave(SquareWave8192Hz); // 8192Hz

```

API changes v1.0.1 to v2.0.0

The API has been changed to make RTC libraries compatible with `libc time.h`. This makes it easier to calculate with date/time and port the application to different platforms. See changes below:

v1.0.1	v2.0.0
DS3231_DateTime	struct tm
Function returns true: failure	Function returns false: failure
	clearOscillatorStopFlag() merged into oscillatorEnable()
setDateTime()	bool write(struct tm *dt)
getDateTime()	bool read(struct tm *dt)
getEpochTime()	time_t getEpoch()
	bool setEpoch(time_t t)
	bool setDateTime(uint8_t hour, uint8_t min, uint8_t sec, uint8_t mday, uint8_t mon, uint16_t year, uint8_t wday)
	ErriezDS3231Debug class removed to reduce flash size

Library dependencies

- Wire.h
- Terminal.ino requires ErriezSerialTerminal library.

Library installation

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

Other Arduino Libraries and Sketches from Erriez

- [Erriez Libraries and Sketches](#)

Chapter 2

Class Index

2.1 Class List

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

ErriezDS3231	
DS3231 RTC base class	13

Chapter 3

File Index

3.1 File List

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

src/ ErriezDS3231.cpp	
DS3231 high precision RTC library for Arduino	29
src/ ErriezDS3231.h	
DS3231 high precision RTC library for Arduino	30

Chapter 4

Class Documentation

4.1 ErriezDS3231 Class Reference

DS3231 RTC base class.

```
#include <ErriezDS3231.h>
```

Public Member Functions

- bool [begin](#) ()
Initialize and detect DS3231 RTC.
- bool [isRunning](#) ()
Read RTC OSF (Oscillator Stop Flag) from status register.
- bool [clockEnable](#) (bool enable=true)
Enable or disable oscillator when running on V-BAT.
- time_t [getEpoch](#) ()
Read Unix UTC epoch time_t.
- bool [setEpoch](#) (time_t t)
Write Unix epoch UTC time to RTC.
- bool [read](#) (struct tm *dt)
Read date and time from RTC.
- bool [write](#) (const struct tm *dt)
Write date and time to RTC.
- bool [setTime](#) (uint8_t hour, uint8_t min, uint8_t sec)
Write time to RTC.
- bool [getTime](#) (uint8_t *hour, uint8_t *min, uint8_t *sec)
Read time from RTC.
- bool [setDateTime](#) (uint8_t hour, uint8_t min, uint8_t sec, uint8_t mday, uint8_t mon, uint16_t year, uint8_t wday)
Set date time.
- bool [setAlarm1](#) ([Alarm1Type](#) alarmType, uint8_t dayDate, uint8_t hours, uint8_t minutes, uint8_t seconds)
Set Alarm 1.
- bool [setAlarm2](#) ([Alarm2Type](#) alarmType, uint8_t dayDate, uint8_t hours, uint8_t minutes)
Set Alarm 2.
- bool [alarmInterruptEnable](#) ([AlarmId](#) alarmId, bool enable)

- Enable or disable Alarm 1 or 2 interrupt.*
 - bool [getAlarmFlag](#) ([AlarmId](#) alarmId)
 - Get Alarm 1 or 2 flag.*
 - bool [clearAlarmFlag](#) ([AlarmId](#) alarmId)
 - Clear alarm flag.*
 - bool [setSquareWave](#) ([SquareWave](#) squareWave)
 - Configure SQW (Square Wave) output pin.*
 - bool [outputClockPinEnable](#) (bool enable)
 - Enable or disable 32kHz output clock pin.*
 - bool [setAgingOffset](#) (int8_t val)
 - Set aging offset register.*
 - int8_t [getAgingOffset](#) ()
 - Get aging offset register.*
 - bool [startTemperatureConversion](#) ()
 - Start temperature conversion.*
 - bool [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 [readRegister](#) (uint8_t reg)
 - Read register.*
 - bool [writeRegister](#) (uint8_t reg, uint8_t value)
 - Write to RTC register.*
 - bool [readBuffer](#) (uint8_t reg, void *buffer, uint8_t len)
 - Read buffer from RTC.*
 - bool [writeBuffer](#) (uint8_t reg, void *buffer, uint8_t len)
 - Write buffer to RTC.*

4.1.1 Detailed Description

DS3231 RTC base class.

Definition at line 149 of file ErriezDS3231.h.

4.1.2 Member Function Documentation

4.1.2.1 alarmInterruptEnable()

```
bool ErriezDS3231::alarmInterruptEnable (
    AlarmId alarmId,
    bool enable )
```

Enable or disable Alarm 1 or 2 interrupt.

Enabling the alarm interrupt will disable the Square Wave output on the INT/SQW pin. The INT pin remains high until an alarm match occurs.

Parameters

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

Return values

<i>true</i>	Success
<i>false</i>	Alarm interrupt enable failed.

Definition at line 484 of file ErriezDS3231.cpp.

4.1.2.2 bcdToDec()

```
uint8_t ErriezDS3231::bcdToDec (
    uint8_t bcd )
```

BCD to decimal conversion.

Parameters

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

Returns

Decimal value.

Definition at line 749 of file ErriezDS3231.cpp.

4.1.2.3 begin()

```
bool ErriezDS3231::begin ( )
```

Initialize and detect DS3231 RTC.

Call this function from setup().

Return values

<i>true</i>	RTC detected.
<i>false</i>	RTC not detected.

Definition at line 52 of file ErriezDS3231.cpp.

4.1.2.4 clearAlarmFlag()

```
bool ErriezDS3231::clearAlarmFlag (
    AlarmId alarmId )
```

Clear alarm flag.

This function should be called when the alarm flag has been handled in polling and interrupt mode. The INT pin changes to high when both alarm flags are cleared and alarm interrupts are enabled.

Parameters

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

Return values

<i>true</i>	Success
<i>false</i>	Incorrect alarm ID.

Definition at line 546 of file ErriezDS3231.cpp.

4.1.2.5 clockEnable()

```
bool ErriezDS3231::clockEnable (
    bool enable = true )
```

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.
---------------	---

Return values

<i>true</i>	Success.
<i>false</i>	Oscillator enable failed.

Definition at line 74 of file ErriezDS3231.cpp.

4.1.2.6 decToBcd()

```
uint8_t ErriezDS3231::decToBcd (
    uint8_t dec )
```

Decimal to BCD conversion.

Parameters

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

Returns

BCD encoded value.

Definition at line 761 of file ErriezDS3231.cpp.

4.1.2.7 getAgingOffset()

```
int8_t ErriezDS3231::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 664 of file ErriezDS3231.cpp.

4.1.2.8 getAlarmFlag()

```
bool ErriezDS3231::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.

The INT pin changes to low when an Alarm 1 or Alarm 2 match occurs and the interrupt is enabled. The pin remains low until both alarm flags are cleared by the application.

Parameters

<i>alarm</i> ↔ <i>Id</i>	Alarm1 or Alarm2 enum.
-----------------------------	------------------------

Return values

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

Definition at line 523 of file ErriezDS3231.cpp.

4.1.2.9 getEpoch()

```
time_t ErriezDS3231::getEpoch ( )
```

Read Unix UTC epoch time_t.

Returns

Unix epoch time_t seconds since 1970.

Definition at line 127 of file ErriezDS3231.cpp.

4.1.2.10 getTemperature()

```
bool ErriezDS3231::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.

Return values

<i>true</i>	Success
<i>false</i>	Set get temperature failed.

Definition at line 719 of file ErriezDS3231.cpp.

4.1.2.11 getTime()

```
bool ErriezDS3231::getTime (
    uint8_t * hour,
    uint8_t * min,
    uint8_t * sec )
```

Read time from RTC.

Read hour, minute and second registers from RTC.

Parameters

<i>hour</i>	Hours 0..23.
<i>min</i>	Minutes 0..59.
<i>sec</i>	Seconds 0..59.

Return values

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

Definition at line 300 of file ErriezDS3231.cpp.

4.1.2.12 isRunning()

```
bool ErriezDS3231::isRunning ( )
```

Read RTC OSF (Oscillator Stop Flag) from status register.

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

Definition at line 110 of file ErriezDS3231.cpp.

4.1.2.13 outputClockPinEnable()

```
bool ErriezDS3231::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.
---------------	--

Return values

<i>true</i>	Success
<i>false</i>	Set output clock pin failed.

Definition at line 603 of file ErriezDS3231.cpp.

4.1.2.14 read()

```
bool ErriezDS3231::read (
    struct tm * dt )
```

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>dt</i>	Date and time struct tm.
-----------	--------------------------

Return values

<i>true</i>	Success
<i>false</i>	RTC read failed.

Definition at line 185 of file ErriezDS3231.cpp.

4.1.2.15 readBuffer()

```
bool ErriezDS3231::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.

Return values

<i>true</i>	Success
<i>false</i>	I2C read failed.

Definition at line 843 of file ErriezDS3231.cpp.

4.1.2.16 readRegister()

```
uint8_t ErriezDS3231::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 775 of file ErriezDS3231.cpp.

4.1.2.17 setAgingOffset()

```
bool ErriezDS3231::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.
------------	---

Return values

<i>true</i>	Success
<i>false</i>	Set aging offset failed.

Definition at line 634 of file ErriezDS3231.cpp.

4.1.2.18 setAlarm1()

```
bool ErriezDS3231::setAlarm1 (
    Alarm1Type alarmType,
    uint8_t dayDate,
    uint8_t hours,
    uint8_t minutes,
    uint8_t seconds )
```

Set 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 alarmType.
<i>hours</i>	Alarm match hours.
<i>minutes</i>	Alarm match minutes.
<i>seconds</i>	Alarm match seconds.

Return values

<i>true</i>	Success.
<i>false</i>	Set alarm 1 failed.

Definition at line 394 of file ErriezDS3231.cpp.

4.1.2.19 setAlarm2()

```
bool ErriezDS3231::setAlarm2 (
    Alarm2Type alarmType,
    uint8_t dayDate,
    uint8_t hours,
    uint8_t minutes )
```

Set Alarm 2.

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 alarmType.
<i>hours</i>	Alarm match hours.
<i>minutes</i>	Alarm match minutes.

Return values

<i>true</i>	Success.
<i>false</i>	Set alarm 2 failed.

Definition at line 445 of file ErriezDS3231.cpp.

4.1.2.20 setDateTime()

```
bool ErriezDS3231::setDateTime (
    uint8_t hour,
    uint8_t min,
    uint8_t sec,
    uint8_t mday,
    uint8_t mon,
    uint16_t year,
    uint8_t wday )
```

Set date time.

Parameters

<i>hour</i>	Hours 0..23
<i>min</i>	Minutes 0..59

Parameters

<i>sec</i>	Seconds 0..59
<i>mday</i>	Day of the month 1..31
<i>mon</i>	Month 1..12 (1=January)
<i>year</i>	Year 2000..2099
<i>wday</i>	Day of the week 0..6 (0=Sunday, .. 6=Saturday)

Return values

<i>true</i>	Success.
<i>false</i>	Set time failed.

Definition at line 348 of file ErriezDS3231.cpp.

4.1.2.21 setEpoch()

```
bool ErriezDS3231::setEpoch (
    time_t t )
```

Write Unix epoch UTC time to RTC.

Parameters

<i>t</i>	time_t time
----------	-------------

Returns

See write returns.

Definition at line 157 of file ErriezDS3231.cpp.

4.1.2.22 setSquareWave()

```
bool ErriezDS3231::setSquareWave (
    SquareWave squareWave )
```

Configure SQW (Square Wave) output pin.

This will disable or initialize the SQW clock pin. The alarm interrupt INT pin will be disabled.

Parameters

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

Return values

<i>true</i>	Success
<i>false</i>	Set squareWave failed.

Definition at line 577 of file ErriezDS3231.cpp.

4.1.2.23 setTime()

```
bool ErriezDS3231::setTime (
    uint8_t hour,
    uint8_t min,
    uint8_t sec )
```

Write time to RTC.

Write hour, minute and second registers to RTC.

Parameters

<i>hour</i>	Hours 0..23.
<i>min</i>	Minutes 0..59.
<i>sec</i>	Seconds 0..59.

Return values

<i>true</i>	Success.
<i>false</i>	Set time failed.

Definition at line 274 of file ErriezDS3231.cpp.

4.1.2.24 startTemperatureConversion()

```
bool ErriezDS3231::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.

Return values

<i>true</i>	Success
<i>false</i>	Start temperature conversion failed.

Definition at line 691 of file ErriezDS3231.cpp.

4.1.2.25 write()

```
bool ErriezDS3231::write (
    const struct tm * dt )
```

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 237 of file ErriezDS3231.cpp.

4.1.2.26 writeBuffer()

```
bool ErriezDS3231::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.

Return values

<i>true</i>	Success
<i>false</i>	I2C write failed.

Definition at line 815 of file ErriezDS3231.cpp.

4.1.2.27 writeRegister()

```
bool ErriezDS3231::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 794 of file ErriezDS3231.cpp.

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

- [src/ErriezDS3231.h](#)
- [src/ErriezDS3231.cpp](#)

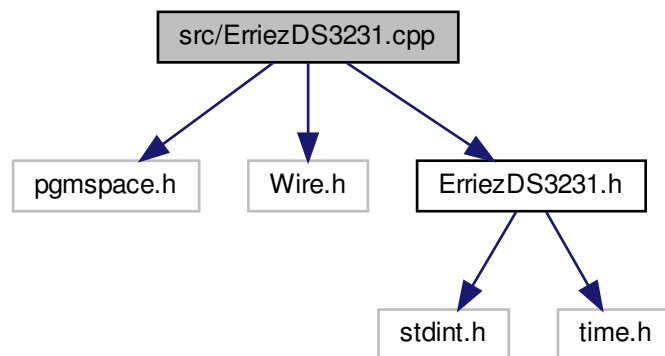
Chapter 5

File Documentation

5.1 src/ErriezDS3231.cpp File Reference

DS3231 high precision RTC library for Arduino.

```
#include <pgmspace.h>
#include <Wire.h>
#include "ErriezDS3231.h"
Include dependency graph for ErriezDS3231.cpp:
```



5.1.1 Detailed Description

DS3231 high precision RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231> Documentation: <https://erriez.github.io/ErriezDS3231>

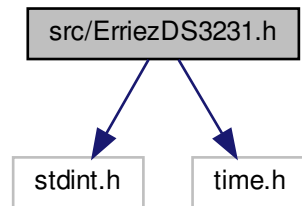
5.2 src/ErriezDS3231.h File Reference

DS3231 high precision RTC library for Arduino.

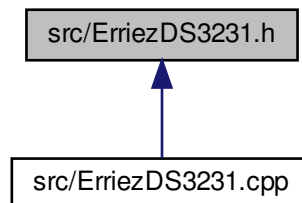
```
#include <stdint.h>
```

```
#include <time.h>
```

Include dependency graph for ErriezDS3231.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ErriezDS3231](#)
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.

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.

5.2.1 Detailed Description

DS3231 high precision RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS3231> Documentation: <https://erriez.github.io/ErriezDS3231>

5.2.2 Macro Definition Documentation

5.2.2.1 DS3231_HOUR_12H_24H

```
#define DS3231_HOUR_12H_24H 6
```

DS3231 register bit defines.

12 or 24 hour mode

Definition at line 66 of file ErriezDS3231.h.

5.2.2.2 DS3231_NUM_REGS

```
#define DS3231_NUM_REGS 19
```

DS3231 number of registers.

19 RTC register: 0x00..0x12

Definition at line 63 of file ErriezDS3231.h.

5.2.2.3 DS3231_REG_SECONDS

```
#define DS3231_REG_SECONDS 0x00
```

DS3231 registers.

Seconds register

Definition at line 40 of file ErriezDS3231.h.

5.2.3 Enumeration Type Documentation

5.2.3.1 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 112 of file ErriezDS3231.h.

5.2.3.2 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 125 of file ErriezDS3231.h.

5.2.3.3 AlarmId

enum [AlarmId](#)

Alarm ID.

Enumerator

Alarm1	Alarm ID 1.
Alarm2	Alarm ID 2.

Definition at line 104 of file ErriezDS3231.h.

5.2.3.4 SquareWave

enum [SquareWave](#)

Squarewave enum.

Enumerator

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

Definition at line 137 of file ErriezDS3231.h.

Index

- Alarm1Type
 - ErriezDS3231.h, [33](#)
- Alarm2Type
 - ErriezDS3231.h, [34](#)
- AlarmId
 - ErriezDS3231.h, [34](#)
- alarmInterruptEnable
 - ErriezDS3231, [14](#)
- bcdToDec
 - ErriezDS3231, [15](#)
- begin
 - ErriezDS3231, [15](#)
- clearAlarmFlag
 - ErriezDS3231, [16](#)
- clockEnable
 - ErriezDS3231, [16](#)
- DS3231_HOUR_12H_24H
 - ErriezDS3231.h, [33](#)
- DS3231_NUM_REGS
 - ErriezDS3231.h, [33](#)
- DS3231_REG_SECONDS
 - ErriezDS3231.h, [33](#)
- decToBcd
 - ErriezDS3231, [16](#)
- ErriezDS3231, [13](#)
 - alarmInterruptEnable, [14](#)
 - bcdToDec, [15](#)
 - begin, [15](#)
 - clearAlarmFlag, [16](#)
 - clockEnable, [16](#)
 - decToBcd, [16](#)
 - getAgingOffset, [17](#)
 - getAlarmFlag, [17](#)
 - getEpoch, [18](#)
 - getTemperature, [18](#)
 - getTime, [19](#)
 - isRunning, [19](#)
 - outputClockPinEnable, [19](#)
 - read, [20](#)
 - readBuffer, [20](#)
 - readRegister, [21](#)
 - setAgingOffset, [21](#)
 - setAlarm1, [22](#)
 - setAlarm2, [22](#)
 - setDateTime, [23](#)
 - setEpoch, [24](#)
 - setSquareWave, [24](#)
 - setTime, [25](#)
 - startTemperatureConversion, [25](#)
 - write, [26](#)
 - writeBuffer, [26](#)
 - writeRegister, [27](#)
- ErriezDS3231.h
 - Alarm1Type, [33](#)
 - Alarm2Type, [34](#)
 - AlarmId, [34](#)
 - DS3231_HOUR_12H_24H, [33](#)
 - DS3231_NUM_REGS, [33](#)
 - DS3231_REG_SECONDS, [33](#)
 - SquareWave, [34](#)
- getAgingOffset
 - ErriezDS3231, [17](#)
- getAlarmFlag
 - ErriezDS3231, [17](#)
- getEpoch
 - ErriezDS3231, [18](#)
- getTemperature
 - ErriezDS3231, [18](#)
- getTime
 - ErriezDS3231, [19](#)
- isRunning
 - ErriezDS3231, [19](#)
- outputClockPinEnable
 - ErriezDS3231, [19](#)
- read
 - ErriezDS3231, [20](#)
- readBuffer
 - ErriezDS3231, [20](#)
- readRegister
 - ErriezDS3231, [21](#)
- setAgingOffset
 - ErriezDS3231, [21](#)
- setAlarm1
 - ErriezDS3231, [22](#)
- setAlarm2
 - ErriezDS3231, [22](#)
- setDateTime
 - ErriezDS3231, [23](#)
- setEpoch
 - ErriezDS3231, [24](#)
- setSquareWave
 - ErriezDS3231, [24](#)

setTime

ErriezDS3231, [25](#)

SquareWave

ErriezDS3231.h, [34](#)

src/ErriezDS3231.cpp, [29](#)

src/ErriezDS3231.h, [30](#)

startTemperatureConversion

ErriezDS3231, [25](#)

write

ErriezDS3231, [26](#)

writeBuffer

ErriezDS3231, [26](#)

writeRegister

ErriezDS3231, [27](#)