DS3231

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 DS3231 Class Reference

This is the main class of the library.

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

Public Member Functions

- DS3231 (bool INTCtr=true)

    By default, no SQW is outputted.
- void begin ()

    Starts the library.
- bool is_12 ()

    Method to check the running state of the clock.
- void set_12 ()

    Changes clock to run in 12 hour mode.
- void set_24 ()

    Changes clock to run in 24 hour mode.
- void setTime (uint8_t number, uint8_t value)

    Method to change one item related to time (hour, minutes or seconds).
- void setTime (const uint8_t hours, const uint8_t minutes, const uint8_t seconds)

    Method to change all items related to time at once.
- void setDate (uint8_t number, uint16_t value)

    Method to change one item related to date (day, date, month or year).
- void setDate (const dayOfWeek day, const Month month, const uint8_t date, const uint16_t year)

    Method to change all items related to date at once.
- RTCdata readTime ()

    Method to read the time keeping registers of the device.
- uint8_t checkAlarmFlag ()

    Method to check whether the alarms were triggered.
- void setAlarm (uint8_t alarmNumber, RTCalarm &alarm)

    Method to change the alarm values.
- void storeAlarmEEPROM (uint8_t alarmNumber)

    Method to store one of the two alarms in memory in case power is lost.

- RTCalarm readAlarmEEPROM (uint8_t alarmNumber)

    Method to read one of the two alarms from memory.
- bool alarmState (uint8_t alarmNumber) const

    Method to check whether one of the two alarms has been triggered.
- RTCalarm readAlarm (uint8_t alarmNumber)

    Method to read alarm information.
- void setAlarmDaily (uint8_t alarmNumber, uint8_t hour, const uint8_t minute)

    Method to set the alarm to trigger every time day at the specified time.
- void setAlarmWeekly (uint8_t alarmNumber, uint8_t hour, const uint8_t minute, const dayOfWeek day)

    Method to set the alarm to trigger at the time and day of the week specified (once a week).
- void toggleAlarm (uint8_t alarmNumber, bool enable)

    Method to toggle the alarm ON or OFF.
- void snoozeAlarm ()

    Method to stop the alarm.
- float readCelcius ()

    Reads temperature registers and returns value in Celcius.
- float readFahrenheit ()

    Uses readCelcius method, converts value to Fahrenheit.
- float readKelvin ()

    Uses readCelcius method, converts value to Kelvin.
- void toggleSQW (bool enable)

    Method to control the state of the SQW pin.
- void setSQW (uint8_t mode)

    Method to modify the frequency of the square wave signal outputted by the SQW pin.
- void toggle32kHz (bool enable)

    Method to enable or disable the 32kHz square wave outputted at pin 32K.
- void enableOSC (bool enable)

    Method to enable or disable the internal oscillator.

Static Public Member Functions

- static const char ∗ dayStr (const dayOfWeek day)

    converts dayOfWeek data to string
- static const char ∗ monthStr (const Month month)

    converts Month data to string

Private Member Functions

- void writeINTCtr (bool enable)

    Method to toggle the INTCN bit (bit 2 of control register).

Static Private Member Functions

- static uint8_t setHigh (uint8_t byte, uint8_t bit)

  Method to set a specific bit of a byte to 1.
- static uint8_t setLow (uint8_t byte, uint8_t bit)

  Method to set a specific bit of a byte to 0.
- static uint8_t BCDtoDEC (const uint8_t bcd)

  Method to convert binary coded decimal to binary.
- static uint8_t DECtoBCD (const uint8_t dec)

  Method to convert binary to binary coded decimal.
- static void readRegister (uint8_t reg, uint8_t byteBuffer[ ], const uint16_t bytes)

  Method to read a specific register of the device.
- static void writeRegister (const uint8_t reg, const uint8_t byteBuffer[ ], const uint8_t bytes)

  Method to write data to a specific register of the device.
- static void readEEPROM (uint8_t address, uint8_t byteBuffer[ ], const uint16_t bytes)

  Method to read data written to a specific address on the EEPROM chip of the device.
- static void writeEEPROM (uint8_t address, uint8_t byteBuffer[ ], const uint16_t bytes)

  Method to write data to a specific address on the EEPROM chip of the device.

Private Attributes

- bool INTCtr

  false -> enables 32KHz SQW; true -> allows A1F & A2F to set INT/SQW pin low in alarm condition.
- RTCdata clockTime

  holds last read clock data.
- RTCalarm alarm1

  holds alarm1 information.
- RTCalarm alarm2

  holds alarm2 information.

2.1.1 Detailed Description

This is the main class of the library.

The class contains the methods which interact with the RTC module to read and write to the time-keeping registers, to control the alarm states, to read temperature and to control the square wave output from SQW pin.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 DS3231()

```
DS3231::DS3231 (
            bool INTCtr = true )
```

By default, no SQW is outputted.

2.1.3 Member Function Documentation

2.1.3.1 alarmState()

```
bool DS3231::alarmState (
            uint8_t alarmNumber ) const
```

Method to check whether one of the two alarms has been triggered.

Alarm flags are set to 1 every time the registers of the alarm match the registers of the clock (based on the alarm mask).

Parameters

alarmNumber	The number of the alarm (1 or 2)

Returns

True means alarm has been triggered, False means alarm has not been triggered

2.1.3.2 BCDtoDEC()

```
uint8_t DS3231::BCDtoDEC (
            const uint8_t bcd )  [static], [private]
```

Method to convert binary coded decimal to binary.

2.1.3.3 begin()

```
initialize the DS3231 RTC void DS3231::begin ( )
```

Starts the library.

Disables alarm flags, sets the INTCtr bit, restores alarms from memory, sets time.

This method reads the alarms stored in memory and sets them accordingly. It also disables the alarm flags in case the power was lost while alarm was triggered.

2.1.3.4 checkAlarmFlag()

```
uint8_t DS3231::checkAlarmFlag ( )
```

Method to check whether the alarms were triggered.

Returns

Returns 0 if both alarm flags are off, 1 if alarm 1 flag is on, 2 if alarm 2 flag is on

2.1.3.5 dayStr()

```
const char * DS3231::dayStr (
             const dayOfWeek day )  [static]
```

converts dayOfWeek data to string

2.1.3.6 DECtoBCD()

```
uint8_t DS3231::DECtoBCD (
             const uint8_t dec )  [static], [private]
```

Method to convert binary to binary coded decimal.

2.1.3.7 enableOSC()

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

Method to enable or disable the internal oscillator.

When powered by VCC, the oscillator stays on no matter the state of the control register. This function only has effect when the device is powered by VBAT. While disabled, all registeres are static!

Parameters

enable	True -> enables the oscillator; False -> disables the oscillator

2.1.3.8 is_12()

```
bool DS3231::is_12 ( )
```

Method to check the running state of the clock.

Returns

True if clock runs in 12 hour mode and false otherwise.

2.1.3.9 monthStr()

```
const char * DS3231::monthStr (
             const Month month )  [static]
```

converts Month data to string

2.1.3.10 readAlarm()

```
RTCalarm DS3231::readAlarm (
             uint8_t alarmNumber )
```

Method to read alarm information.

Parameters

alarmNumber	The number of the alarm (1 or 2)

Returns

Returns an RTCalarm object containing the information about the specified alarm

2.1.3.11 readAlarmEEPROM()

```
RTCalarm DS3231::readAlarmEEPROM (
             uint8_t alarmNumber )
```

Method to read one of the two alarms from memory.

Parameters

alarmNumber	1 -> Reads alarm1; 2 -> Reads alarm2

2.1.3.12 readCelcius()

```
float DS3231::readCelcius ( )
```

Reads temperature registers and returns value in Celcius.

2.1.3.13 readEEPROM()

```
void DS3231::readEEPROM (
             uint8_t address,
             uint8_t byteBuffer[],
             const uint16_t bytes )  [static], [private]
```

Method to read data written to a specific address on the EEPROM chip of the device.

Parameters

address	The address where data is being stored
byteBuffer	A byte buffer that holds the data
bytes	The number of bytes that need to be read.

2.1.3.14 readFahrenheit()

```
float DS3231::readFahrenheit ( )
```

Uses readCelcius method, converts value to Fahrenheit.

2.1.3.15 readKelvin()

```
float DS3231::readKelvin ( )
```

Uses readCelcius method, converts value to Kelvin.

2.1.3.16 readRegister()

```
void DS3231::readRegister (
             uint8_t reg,
             uint8_t byteBuffer[],
             const uint16_t bytes )  [static], [private]
```

Method to read a specific register of the device.

Parameters

reg	The register's address
byteBuffer	A byte buffer that holds the data that's being read
bytes	The number of bytes that need to be read (max 7!)

This method uses the Wire library to communicate with the device via I2C and read the desired register of the device.

2.1.3.17 readTime()

```
RTCdata DS3231::readTime ( )
```

Method to read the time keeping registers of the device.

Returns

Returns an RTCdata object that holds the information from the registers

2.1.3.18 set_12()

```
void DS3231::set_12 ( )
```

Changes clock to run in 12 hour mode.

This method reads the hour register of the clock and modifies bits 6 and 5 so that the clock will run in 12 hour mode.

2.1.3.19 set_24()

```
void DS3231::set_24 ( )
```

Changes clock to run in 24 hour mode.

This method reads the hour register of the clock and modifies bits 6 and 5 so that the clock will run in 24 hour mode.

2.1.3.20 setAlarm()

```
void DS3231::setAlarm (
             uint8_t alarmNumber,
             RTCalarm & alarm )
```

Method to change the alarm values.

This method will only store information in memory. The alarm information will not be communicated to DS3231.

Parameters

alarmNumber	1 -> changes affect alarm1; 2 -> changes affect alarm2
alarm	RTCalarm object which holds information about the new alarm

2.1.3.21 setAlarmDaily()

```
void DS3231::setAlarmDaily (
             uint8_t alarmNumber,
             uint8_t hour,
             const uint8_t minute )
```

Method to set the alarm to trigger every time day at the specified time.

Alarm triggers when the hour and minute registers of the alarm match the hour and minute registers of the clock.

Parameters

alarmNumber	Number of the alarm (1 or 2)

2.1.3.22 setAlarmWeekly()

```
void DS3231::setAlarmWeekly (
             uint8_t alarmNumber,
             uint8_t hour,
             const uint8_t minute,
             const dayOfWeek day )
```

Method to set the alarm to trigger at the time and day of the week specified (once a week).

Parameters

alarmNumber	alarmNumber Number of the alarm (1 or 2)

2.1.3.23 setDate() `[1/2]`

```
void DS3231::setDate (
             const dayOfWeek day,
             const Month month,
             const uint8_t date,
             const uint16_t year )
```

Method to change all items related to date at once.

2.1.3.24 setDate() [2/2]

```
void DS3231::setDate (
            uint8_t number,
            uint16_t value )
```

Method to change one item related to date (day, date, month or year).

Parameters

number	Represents the date item. 0->day, 1->date, 2->month, 3->year
value	New value of the item

2.1.3.25 setHigh()

```
uint8_t DS3231::setHigh (
            uint8_t byte,
            uint8_t bit )  [static], [private]
```

Method to set a specific bit of a byte to 1.

Parameters

byte	The byte that needs to be modified
bit	The bit that needs modified (0-7)

Returns

Returns the modified byte

2.1.3.26 setLow()

```
uint8_t DS3231::setLow (
            uint8_t byte,
            uint8_t bit )  [static], [private]
```

Method to set a specific bit of a byte to 0.

Parameters

byte	The byte that needs to be modified
bit	The bit that needs modified (0-7)

Returns

Returns the modified byte

2.1.3.27 setSQW()

```
void DS3231::setSQW (
             uint8_t mode )
```

Method to modify the frequency of the square wave signal outputted by the SQW pin.

The device can generate 4 frequencies : 1Hz, 1kHz, 4kHz and 8kHz.

Parameters

mode	0 -> 1Hz; 1 -> 1kHz; 2 -> 4kHz; 3 -> 8kHz;

2.1.3.28 setTime() [1/2]

```
void DS3231::setTime (
             const uint8_t hours,
             const uint8_t minutes,
             const uint8_t seconds )
```

Method to change all items related to time at once.

This method converts the parameters to BCD and then writes the values to the device registers.

2.1.3.29 setTime() [2/2]

```
void DS3231::setTime (
             uint8_t number,
             uint8_t value )
```

Method to change one item related to time (hour, minutes or seconds).

Parameters

number	Represents the time item. 0 -> hour, 1 -> minute, 2->seconds
value	New value of the item

As of now, this method only works in 24 hour mode.

2.1.3.30 snoozeAlarm()

```
void DS3231::snoozeAlarm ( )
```

Method to stop the alarm.

This method sets both alarm flags to 0.

2.1.3.31 storeAlarmEEPROM()

```
void DS3231::storeAlarmEEPROM (
            uint8_t alarmNumber )
```

Method to store one of the two alarms in memory in case power is lost.

Parameters

alarmNumber	1 -> Stores alarm1; 2 -> Stores alarm2

2.1.3.32 toggle32kHz()

```
void DS3231::toggle32kHz (
            bool enable )
```

Method to enable or disable the 32kHz square wave outputted at pin 32K.

32K pin goes in high impedance state when disabled.

Parameters

enable	true -> enables the square wave signal; false -> disables the square wave signal

2.1.3.33 toggleAlarm()

```
void DS3231::toggleAlarm (
            uint8_t alarmNumber,
            bool enable )
```

Method to toggle the alarm ON or OFF.

Parameters

alarmNumber	The number of the alarm (1 or 2)
enable	True -> enables the alarm; False -> disables the alarm

2.1.3.34 toggleSQW()

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

Method to control the state of the SQW pin.

When enabled, INTCN bit is set low and a square wave is generated at SQW pin. During this time, no alarm can be triggered, because one of the alarm conditions is INTCN bit to be set to 1.

Parameters

enable	True -> square wave signal is generated; False -> SQW pin stays HIGH and alarms can be triggered

2.1.3.35 writeEEPROM()

```
void DS3231::writeEEPROM (
            uint8_t address,
            uint8_t byteBuffer[],
            const uint16_t bytes ) [static], [private]
```

Method to write data to a specific address on the EEPROM chip of the device.

Parameters

address	address The address where data is being stored
byteBuffer	A byte buffer that holds the data
bytes	The number of bytes that need to be written.

2.1.3.36 writeINTCtr()

```
void DS3231::writeINTCtr (
            bool enable ) [private]
```

Method to toggle the INTCN bit (bit 2 of control register).

This method is private because it is used by other methods (mainly the alarm methods).

2.1.3.37 writeRegister()

```
void DS3231::writeRegister (
            const uint8_t reg,
            const uint8_t byteBuffer[],
            const uint8_t bytes ) [static], [private]
```

Method to write data to a specific register of the device.

Parameters

reg	The register's address
byteBuffer	A byte buffer that holds the data that's being transferred
bytes	The number of bytes that need to be written (max 7!)

This method uses the Wire library to communicate with the device via I2C and write information to the specified register.

2.1.4 Member Data Documentation

2.1.4.1 alarm1

`RTCalarm DS3231::alarm1 [private]`

holds alarm1 information.

2.1.4.2 alarm2

`RTCalarm DS3231::alarm2 [private]`

holds alarm2 information.

2.1.4.3 clockTime

`RTCdata DS3231::clockTime [private]`

holds last read clock data.

2.1.4.4 INTCtr

`bool DS3231::INTCtr [private]`

false -> enables 32KHz SQW; true -> allows A1F & A2F to set INT/SQW pin low in alarm condition.

2.2 RTCalarm Struct Reference

Struct that holds values for alarm time and day registers.

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

Public Member Functions

- RTCalarm **operator=** (const RTCalarm &alarm)

Public Attributes

- uint8_t **seconds**
- uint8_t **minutes**
- uint8_t **hour**
- dayOfWeek **day**
- bool **enabled**
- bool **pm**

2.2.1 Detailed Description

Struct that holds values for alarm time and day registers.

2.3 RTCdata Struct Reference

Struct that holds time & date information of main registers.

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

Public Member Functions

- bool **operator==** (RTCdata clock)
- RTCdata **operator=** (const RTCdata &clock)

Public Attributes

- uint8_t **seconds**
- uint8_t **minutes**
- uint8_t **hour**
- dayOfWeek **day**
- Month **month**
- uint16_t **year**
- uint8_t **date**
- bool **pm**

2.3.1 Detailed Description

Struct that holds time & date information of main registers.