Erriez DS3231 high precision I2C RTC library for Arduino

1.0.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

DS3231 high precision I2C RTC library for Arduino

This is an advanced DS3231 high precision I2C RTC library for Arduino.

Library features

- Read time
- Set time
- Read date and time
- Set date and time
- Read Unix Epoch UTC 32-bit timestamp
- 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
- Easy debug functionality
- Basic and advanced examples
- Set date/time over serial with Python script
- Low RAM footprint:
  - `sizeof(DS3231)`: 1 Byte RTC object.
  - `sizeof(DS3231_DateTime)`: 8 Bytes date/time object.
- Full Doxygen documentation

Hardware

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

Pins

DS3231	Arduino UNO / Nano / Micro / Pro Micro	Mega2560	Leonardo	ESP8266 / WeMos D1 & R2 / Node MCU
VCC	5V	5V	5V	3V3
GND	GND	GND	GND	GND
SDA	A4	D20	D2	D2 (GPIO4)
CLK	A5	D21	D3	D1 (GPIO5)
SQW	D2 (INT0)	D2 (INT4)	D7 (INT6)	D3 (GPIO0)

Examples

Arduino IDE | Examples | Erriez DS3231 RTC:

- `AgingOffset` Aging offset programming.
- `AlarmInterrupt` Alarm with interrupts.
- `AlarmPolling` Alarm polled.
- `GettingStarted` Getting started example which contains most date/time/temperature features.
- `Minimum` Minimum example to read time.
- `PrintDiagnostics` Print diagnostics and registers.
- `ReadTimeInterrupt` Read time with 1Hz SQW interrupt. (Highly recommended)
- `ReadTimePolled` Read time polled.
- `SetDateTime` Set date time. (Must be started first)
- `Temperature` Temperature.
- `Terminal` Advanced terminal interface with `set date/time Python` script.

Documentation

- `Doxygen online HTML`
- `Doxygen PDF`
- `DS3231 datasheet`

Usage

Initialization

```
{c++}
#include <Wire.h>
#include <DS3231.h>

// Create DS3231 RTC object
static DS3231 rtc;


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

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

Check oscillator status at startup

```
{c++}
// Check oscillator status
if (rtc.isOscillatorStopped()) {
    // Error: DS3231 RTC oscillator stopped. Date/time cannot be trusted.
    // Set new date/time before reading date/time.
    while (1) {
        ;
    }
}
```

Set time

```
{c++}
// Write time to RTC
if (rtc.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 (rtc.getTime(&hour, &minute, &second)) {
    // Error: Read time failed
}
```

Set date time

```
{c++}
// Create and initialize date time object
static DS3231_DateTime dt = {
    .second = 0,
    .minute = 36,
    .hour = 21,
    .dayWeek = 7, // 1 = Monday
    .dayMonth = 29,
    .month = 7,
     .year = 2018
};

// Set new RTC date/time
rtc.setDateTime(&dt);
```

Get date time

```
1 {c++}
2 DS3231_DateTime dt;
3
4 // Read RTC date and time from RTC
5 if (rtc.getDateTime(&dt)) {
6     // Error: Read date time failed
7 }
```

Get Epoch Unix UTC time

```
1 {c++}
2 uint32_t epoch;
3
4 // Read date/time from RTC
5 if (rtc.getDateTime(&dt)) {
6     // Error: Read date/time failed
7     return;
8 }
9
10 // Convert date/time to 32-bit epoch time
11 epoch = rtc.getEpochTime(&dt));
```

Get temperature

```
1 {c++}
2 int8_t temperature;
3 uint8_t fraction;
4
5 // Force temperature conversion
6 // Without this call, it takes 64 seconds before the temperature is updated.
7 rtc.startTemperatureConversion();
8
9 // Read temperature
10 rtc.getTemperature(&temperature, &fraction);
11
12 // Print temperature. The output below is for example: 28.25C
13 Serial.print(temperature);
14 Serial.print(F("."));
15 Serial.print(fraction);
16 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:

```
1 {c++}
2 // Generate alarm 1 every second
3 rtc.setAlarm1(Alarm1EverySecond, 0, 0, 0, 0);
4
5 // Generate alarm 1 every minute and second match
6 rtc.setAlarm1(Alarm1EverySecond, 0, 0, 45, 30);
7
8 // Generate alarm 1 every day, hour, minute and second match
9 rtc.setAlarm1(Alarm1MatchDay,
10               1,  // Alarm day match (1 = Monday)
11               12, // Alarm hour match
12               45, // Alarm minute match
13               30  // Alarm second match
14 );
```

Program Alarm 2

```
1 {c++}
2 // Generate alarm 2 every minute
3 rtc.setAlarm2(Alarm2EveryMinute, 0, 0, 0);
4
5 // Generate alarm 2 every hour, minute match
6 rtc.setAlarm2(Alarm2MatchHours, 0, 23, 59);
7
8 // Generate alarm 2 every date, hour, minute match
9 rtc.setAlarm2(Alarm2MatchDate, 28, 7, 0);
```

Alarm polling

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

```
1 {c++}
2 // Poll alarm 1 flag
3 if (rtc.getAlarmFlag(Alarm1)) {
4     // Handle Alarm 1
5
6     // Clear alarm 1 flag
7     rtc.clearAlarmFlag(Alarm1);
8 }
9
10 // Poll alarm 2 flag
11 if (rtc.getAlarmFlag(Alarm2)) {
12     // Handle Alarm 2
13
14     // Clear alarm 2 flag
15     rtc.clearAlarmFlag(Alarm2);
16 }
```

Alarm interrupt

Note: Enabling interrupt will disable the `SQW` output signal.

```
1 {c++}
2 // Uno, Nano, Mini, other 328-based: pin D2 (INT0) or D3 (INT1)
3 #define INT_PIN     2
4
5 // Alarm interrupt flag must be volatile
6 static volatile bool alarmInterrupt = false;
7
8
9 static void alarmHandler()
10 {
11     // Set global interrupt flag
12     alarmInterrupt = true;
13 }
14
15 void setup()
16 {
17     ...
18
19     // Attach to INT0 interrupt falling edge
20     pinMode(INT_PIN, INPUT_PULLUP);
21     attachInterrupt(digitalPinToInterrupt(INT_PIN), alarmHandler, FALLING);
22
23     // Enable Alarm 1 and 2 interrupts
24     rtc.alarmInterruptEnable(Alarm1, true);
25     rtc.alarmInterruptEnable(Alarm2, true);
26 }
27
28 void loop()
29 {
30     // Check global alarm interrupt flag
31     if (alarmInterrupt) {
32         if (rtc.getAlarmFlag(Alarm1)) {
33             // Handle alarm 1
34
35             // Clear alarm 1 interrupt
36             rtc.clearAlarmFlag(Alarm1);
37         }
38
39         if (rtc.getAlarmFlag(Alarm2)) {
40             // Handle alarm 2
41
42             // Clear alarm 2 interrupt
43             rtc.clearAlarmFlag(Alarm2);
44         }
45     }
46 }
```

32kHz clock out

Enable or disable `32kHz` output pin.

```
1 {c++}
2 rtc.outputClockPinEnable(true);     // Enable
3 rtc.outputClockPinEnable(false);    // Disable
```

Square Wave Out (SQW)

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

```
1 {c++}
2 rtc.setSquareWave(SquareWaveDisable);   // Disable
3 rtc.setSquareWave(SquareWave1Hz);       // 1Hz
4 rtc.setSquareWave(SquareWave1024Hz);    // 1024Hz
5 rtc.setSquareWave(SquareWave4096Hz);    // 4096Hz
6 rtc.setSquareWave(SquareWave8192Hz);    // 8192Hz
```

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

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

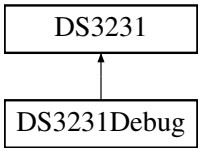
Class Documentation

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 OSF (Oscillator Stop Flag) from status register.
- 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 Alarm 1.
- void setAlarm2 (Alarm2Type alarmType, uint8_t dayDate, uint8_t hours, uint8_t minutes)

  Set Alarm 2.
- 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 161 of file DS3231.h.

5.1.2 Member Function Documentation

5.1.2.1 void DS3231::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

alarmId	Alarm1 or Alarm2 enum.
enable	true: Enable alarm interrupt. false: Disable alarm interrupt.

Definition at line 417 of file DS3231.cpp.

5.1.2.2 uint8_t DS3231::bcdToDec (uint8_t *bcd*)

BCD to decimal conversion.

Parameters

bcd	BCD encoded value.

Returns

Decimal value.

Definition at line 658 of file DS3231.cpp.

5.1.2.3 bool DS3231::begin ( )

Initialize and detect DS3231 RTC.

Call this function from setup().

Return values

Success	RTC detected.
false	RTC not detected.
true	Invalid status register or RTC not detected.

Definition at line 55 of file DS3231.cpp.

5.1.2.4 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. The INT pin changes to high when both alarm flags are cleared and alarm interrupts are enabled.

Parameters

alarmId	Alarm1 or Alarm2 enum.

Return values

Success	
Failure	Incorrect alarm ID.

Definition at line 478 of file DS3231.cpp.

5.1.2.5 uint8_t DS3231::decToBcd (uint8_t *dec*)

Decimal to BCD conversion.

Parameters

dec	Decimal value.

Returns

BCD encoded value.

Definition at line 670 of file DS3231.cpp.

5.1.2.6 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 583 of file DS3231.cpp.

5.1.2.7 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.

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

alarm↩ Id	Alarm1 or Alarm2 enum.

Return values

true	Alarm interrupt flag set.
false	Alarm interrupt flag cleared.

Definition at line 456 of file DS3231.cpp.

5.1.2.8 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

dateTime	Date and time structure.

Return values

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

Definition at line 176 of file DS3231.cpp.

5.1.2.9 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

epoch	32-bit unsigned Unix Epoch time

Definition at line 284 of file DS3231.cpp.

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

Read temperature.

Parameters

temperature	8-bit signed temperature in degree Celsius.
fraction	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 630 of file DS3231.cpp.

5.1.2.11 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

hour	Hours 0..23.
minute	Minutes 0..59.
second	Seconds 0..59.

Return values

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

Definition at line 246 of file DS3231.cpp.

5.1.2.12 bool DS3231::isOscillatorStopped ( )

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

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

Return values

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

Definition at line 104 of file DS3231.cpp.

5.1.2.13 void DS3231::oscillatorEnable (bool *enable*)

Enable or disable oscillator when running on V-BAT.

Parameters

enable	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 73 of file DS3231.cpp.

5.1.2.14 void DS3231::outputClockPinEnable (bool *enable*)

Enable or disable 32kHz output clock pin.

Parameters

enable	true: Enable 32kHz output clock pin. false: Disable 32kHz output clock pin.

Definition at line 528 of file DS3231.cpp.

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

Read buffer from RTC.

Parameters

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

Definition at line 777 of file DS3231.cpp.

5.1.2.16 uint8_t DS3231::readControlRegister ( )

Read control register.

Returns

8-bit unsigned register value

Definition at line 680 of file DS3231.cpp.

5.1.2.17 uint8_t DS3231::readRegister (uint8_t *reg*)

Read register.

Please refer to the RTC datasheet.

Parameters

reg	RTC register number 0x00..0x12.

Returns

value 8-bit unsigned register value.

Definition at line 721 of file DS3231.cpp.

5.1.2.18 uint8_t DS3231::readStatusRegister ( )

Read status register.

Returns

8-bit unsigned register value

Definition at line 698 of file DS3231.cpp.

5.1.2.19 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

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

Definition at line 555 of file DS3231.cpp.

5.1.2.20 void DS3231::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

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

Definition at line 339 of file DS3231.cpp.

5.1.2.21 void DS3231::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

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

Definition at line 384 of file DS3231.cpp.

5.1.2.22 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

dateTime	Date time structure. Providing invalid date/time data may result in unpredictable behavior.

Definition at line 141 of file DS3231.cpp.

5.1.2.23 void DS3231::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

squareWave	SquareWave configuration: Disable: SquareWaveDisable 1Hz: SquareWave1Hz 1024Hz: SquareWave1024Hz 4096Hz: SquareWave4096Hz 8192Hz: SquareWave8192Hz

Return values

Success	
Failure	Incorrect squareWave.

Definition at line 509 of file DS3231.cpp.

5.1.2.24 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

hour	Hours 0..23.
minute	Minutes 0..59.
second	Seconds 0..59.

Definition at line 218 of file DS3231.cpp.

5.1.2.25 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 606 of file DS3231.cpp.

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

Write buffer to RTC.

Please refer to the RTC datasheet.

Parameters

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

Definition at line 757 of file DS3231.cpp.

5.1.2.27 void DS3231::writeControlRegister (uint8_t *value*)

Write control register.

Parameters

value	8-bit unsigned register value

Definition at line 689 of file DS3231.cpp.

5.1.2.28 void DS3231::writeRegister (uint8_t *reg*, uint8_t *value*)

Write to RTC register.

Please refer to the RTC datasheet.

Parameters

reg	RTC register number 0x00..0x12.
value	8-bit unsigned register value.

Definition at line 740 of file DS3231.cpp.

5.1.2.29 void DS3231::writeStatusRegister (uint8_t *value*)

Write status register.

Parameters

value	8-bit unsigned register value

Definition at line 707 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 103 of file DS3231.h.

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

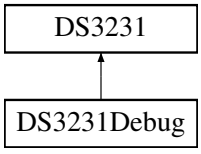
- DS3231.h

5.3 DS3231Debug Class Reference

DS3231 RTC debug class.

`#include <DS3231_debug.h>`

Inheritance diagram for DS3231Debug:



Public Member Functions

- virtual void dumpRegisters (Stream ∗ser, bool printBitfields=true)

  Dump all registers to serial port.
- virtual void printRegister (Stream ∗ser, uint8_t reg, bool printBitfields=true)

  Print register and value.
- virtual void printRegisterBitfields (Stream ∗ser, uint8_t reg, uint8_t regVal)

  Print register bitfields.
- virtual void printDiagnostics (Stream ∗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 48 of file DS3231_debug.h.

5.3.2 Member Function Documentation

5.3.2.1 void DS3231Debug::dumpRegisters (Stream ∗ *ser*, bool *printBitfields* = `true`) [virtual]

Dump all registers to serial port.

Parameters

ser	Serial port.
printBitfields	true: Print register bitfields.

Definition at line 76 of file DS3231_debug.cpp.

5.3.2.2 void DS3231Debug::printDiagnostics (Stream ∗ *ser*) [virtual]

Print diagnostics.

Parameters

ser	Serial port.

Definition at line 290 of file DS3231_debug.cpp.

5.3.2.3 void DS3231Debug::printRegister (Stream ∗ *ser*, uint8_t *reg*, bool *printBitfields* = `true`) [virtual]

Print register and value.

Parameters

ser	Serial port.
reg	Register number.
printBitfields	true: Print register bitfields.

Definition at line 93 of file DS3231_debug.cpp.

5.3.2.4 void DS3231Debug::printRegisterBitfields (Stream * *ser*, uint8_t *reg*, uint8_t *regVal*) [virtual]

Print register bitfields.

Parameters

ser	Serial port.
reg	Register number.
regVal	Register value.

Definition at line 120 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 Documentation: https://erriez.↩
github.io/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, Alarm1Match↩
  Hours = 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 Documentation: https://erriez.↩
github.io/ErriezDS3231

6.2.2 Macro Definition Documentation

6.2.2.1 #define DS3231_HOUR_12H_24H 6

DS3231 register bit defines.

12 or 24 hour mode

Definition at line 65 of file DS3231.h.

6.2.2.2 #define DS3231_NUM_REGS 19

DS3231 number of registers.

19 RTC register: 0x00..0x12

Definition at line 62 of file DS3231.h.

6.2.2.3 #define DS3231_REG_SECONDS 0x00

DS3231 registers.

Seconds register

Definition at line 39 of file DS3231.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 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 124 of file DS3231.h.

6.2.3.2 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 137 of file DS3231.h.

6.2.3.3 enum AlarmId

Alarm ID.

Enumerator

Alarm1 Alarm ID 1.

Alarm2 Alarm ID 2.

Definition at line 116 of file DS3231.h.

6.2.3.4 enum SquareWave

Squarewave enum.

Enumerator

SquareWaveDisable SQW disable.

SquareWave1Hz SQW 1Hz.

SquareWave1024Hz SQW 1024Hz.

SquareWave4096Hz SQW 4096Hz.

SquareWave8192Hz SQW 8192Hz.

Definition at line 149 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 Documentation: https://erriez.↩
github.io/ErriezDS3231

6.3.2 Variable Documentation

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

Register names as string in flash.

Array with all register names in flash.

Definition at line 38 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 Documentation: https://erriez.↩
github.io/ErriezDS3231

Index