

Erriez DS1302 RTC library for Arduino  
2.0.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>DS1302 RTC (Real Time Clock) library for Arduino</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>7</b>
2.1	Class List . . . . .	7
<b>3</b>	<b>File Index</b>	<b>9</b>
3.1	File List . . . . .	9
<b>4</b>	<b>Class Documentation</b>	<b>11</b>
4.1	ErriezDS1302 Class Reference . . . . .	11
4.1.1	Detailed Description . . . . .	12
4.1.2	Constructor & Destructor Documentation . . . . .	12
4.1.2.1	ErriezDS1302() . . . . .	12
4.1.3	Member Function Documentation . . . . .	13
4.1.3.1	bcdToDec() . . . . .	13
4.1.3.2	begin() . . . . .	13
4.1.3.3	clockEnable() . . . . .	13
4.1.3.4	decToBcd() . . . . .	14
4.1.3.5	getDateTime() . . . . .	14
4.1.3.6	getEpoch() . . . . .	15
4.1.3.7	getTime() . . . . .	15
4.1.3.8	isRunning() . . . . .	16
4.1.3.9	read() . . . . .	16
4.1.3.10	readBuffer() . . . . .	17
4.1.3.11	readBufferRAM() . . . . .	17
4.1.3.12	readByteRAM() . . . . .	17
4.1.3.13	readRegister() . . . . .	18
4.1.3.14	setDateTime() . . . . .	18
4.1.3.15	setEpoch() . . . . .	19
4.1.3.16	setTime() . . . . .	19
4.1.3.17	write() . . . . .	20
4.1.3.18	writeBuffer() . . . . .	20
4.1.3.19	writeBufferRAM() . . . . .	21
4.1.3.20	writeByteRAM() . . . . .	21
4.1.3.21	writeRegister() . . . . .	21

<b>5 File Documentation</b>	<b>25</b>
5.1 src/ErriezDS1302.cpp File Reference . . . . .	25
5.1.1 Detailed Description . . . . .	25
5.2 src/ErriezDS1302.h File Reference . . . . .	26
5.2.1 Detailed Description . . . . .	28
5.2.2 Macro Definition Documentation . . . . .	28
5.2.2.1 DS1302_ACB . . . . .	29
5.2.2.2 DS1302_REG_SECONDS . . . . .	29
5.2.2.3 DS1302_SEC_CH . . . . .	29
<b>Index</b>	<b>31</b>

## Chapter 1

# DS1302 RTC (Real Time Clock) library for Arduino

This is a 3-wire DS1302 RTC (Real Time Clock) library for Arduino.

### Library features

- `libc <time.h>` compatible
- Read/write date/time `struct tm`
- Set/get Unix epoch UTC `time_t`
- Set/get time (hours, minutes, seconds)
- Set/get date and time (hour, min, sec, mday, mon, year, wday)
- Read / write 31 Bytes battery backedup RTC RAM.
- Programmable trickle charge to charge super-caps / lithium batteries.
- Optimized IO interface for Atmel AVR platform.

### DS1302 specifications

#### IMPORTANT NOTES:

- The DS1302 RTC time may deviate  $>1$  minute each day, so this device is not recommended for designs with high precision requirements.
- The [high precision DS3231 I2C RTC](#) is recommended for new designs.
- The 3-wire interface is **NOT** compatible with SPI.

## Examples

Arduino IDE | File | Examples | Erriez DS1302 RTC:

- [Alarm](#): Program one or more software alarms
- [Benchmark](#): Benchmark library
- [RAM](#): Read/write RTC RAM.
- [SetBuildDateTime](#): Set build date/time
- [SetGetDateTime](#): Set/get date and time
- [SetGetTime](#): Set/get time
- [SetTrickleCharger](#): Program trickle battery/capacitor charger
- [Terminal](#) and [Python script](#) to set date time
- [Test](#): Regression test
- [WriteRead](#): Regression test

## Documentation

- [Online HTML](#)
- [Doxygen PDF](#)
- [DS1307 datasheet](#)

## Usage

### Initialization

```
{c++}
#include <ErriezDS1302.h>

// Connect DS1302 data pin to Arduino DIGITAL pin
#if defined(ARDUINO_ARCH_AVR)
#define DS1302_CLK_PIN    2
#define DS1302_IO_PIN     3
#define DS1302_CE_PIN     4
#elif defined(ARDUINO_ARCH_ESP8266)
#define DS1302_CLK_PIN    D4
#define DS1302_IO_PIN     D3
#define DS1302_CE_PIN     D2
#elif defined(ARDUINO_ARCH_ESP32)
#define DS1302_CLK_PIN    0
#define DS1302_IO_PIN     4
#define DS1302_CE_PIN     5
#else
#error #error "May work, but not tested on this target"
#endif

// Create RTC object
ErriezDS1302 rtc = ErriezDS1302(DS1302_CLK_PIN, DS1302_IO_PIN, DS1302_CE_PIN);

void setup()
{
    // Initialize RTC
    while (!rtc.begin()) {
        Serial.println(F("RTC not found"));
        delay(3000);
    }
}
```

## Check oscillator status at startup

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

    // Enable oscillator
    rtc.clockEnable(true);
}
```

## Set time

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

## Set date and time

```
{c++}
// Write RTC date/time: 13:45:09 31 December 2019 0=Sunday, 2=Tuesday
if (!rtc.setDateTime(13, 45, 9, 31, 12, 2019, 2)) {
    // Error: RTC write failed
}
```

## Get date and time

```
{c++}
uint8_t hour;
uint8_t min;
uint8_t sec;
uint8_t mday;
uint8_t mon;
uint16_t year;
uint8_t wday;

// Read RTC date/time
if (!rtc.getDateTime(&hour, &min, &sec, &mday, &mon, &year, &wday)) {
    // Error: RTC read failed
}

// hour: 0..23
// min: 0..59
// sec: 0..59
// mday: 1..31
// mon: 1..12
// year: 2000..2099
// wday: 0..6 (0=Sunday .. 6=Saturday)
```

## Write date/time struct tm

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

dt.tm_hour = 12;
dt.tm_min = 34;
dt.tm_sec = 56;
dt.tm_mday = 29;
dt.tm_mon = 1; // 0=January
dt.tm_year = 2020-1900;
dt.tm_wday = 6; // 0=Sunday

if (!rtc.write(&dt)) {
    // Error: RTC Read failed
}
```

### Read date/time struct tm

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

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

### Read Unix Epoch UTC

```
{c++}
time_t t;

// Read Unix epoch UTC from RTC
if (!rtc.getEpoch(&t)) {
    // Error: RTC read failed
}
```

### Write Unix Epoch UTC

```
{c++}
// Write Unix epoch UTC to RTC
if (!rtc.setEpoch(1599416430UL)) {
    // Error: Set epoch failed
}
```

### Write to RTC RAM

```
{c++}
// Write Byte to RTC RAM
rtc.writeByteRAM(0x02, 0xA9);

// Write buffer to RTC RAM
uint8_t buf[NUM_DS1302_RAM_REGS] = { 0x00 };
rtc.writeBufferRAM(buf, sizeof(buf));
```

### Read from RTC RAM

```
{c++}
// Read byte from RTC RAM
uint8_t dataByte = rtc.readByteRAM(0x02);

// Read buffer from RTC RAM
uint8_t buf[NUM_DS1302_RAM_REGS];
rtc.readBufferRAM(buf, sizeof(buf));
```

### Set Trickle Charger

Please refer to the datasheet how to configure the trickle charger.



```
{c++}
// Disable (default)
rtc.writeRegister(DS1302_REG_TC, DS1302_TCS_DISABLE);

// Minimum 2 Diodes, 8kOhm
rtc.writeRegister(DS1302_REG_TC, 0xAB);

// Maximum 1 Diode, 2kOhm
rtc.writeRegister(DS1302_REG_TC, 0xA5);
```

## Set RTC date and time using Python

Flash **Terminal** example.

Set COM port in **examples/Terminal/Terminal.py** Python script.

Run Python script:

```
{c++}
// Install Pyserial
python3 pip -m pyserial

// Set RTC date and time
python3 Terminal.py
```

## Pin configuration

**Note:** ESP8266 pin D4 is high during a power cycle / reset / flashing which may corrupt RTC registers. For this reason, pins D2 and D4 are swapped.

DS1302 Pin	DS1302 IC	Atmel AVR	ESP8266	ESP32
4	GND	GND	GND	GND
8	VCC2	5V (or 3.3V)	3V3	3V3
7	SCLK (CLK)	2 (DIGITAL pin)	D4	0
6	I/O (DAT)	3 (DIGITAL pin)	D2	5
5	CE (RST)	4 (DIGITAL pin)	D2	4

## API changes v1.0.0 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:

```
| v1.0.0 | v2.0.0 | | -----* | -----* | | DS1302_
DateTime | struct tm | | clearOscillatorStopFlag() merged into clockEnable() | | is
Halted() | bool clockEnable(bool enable) | | halt() | void isRunning() | | setDate
Time() | void write(struct tm *dt) | | getDateTime() | bool read(struct tm *dt) | |
getEpochTime() | time_t getEpoch() | | void setEpoch(time_t t) | | writeProtect()
| Removed | | isProtected() | Removed | | void setDateTime(uint8_t hour, uint8_t
min, uint8_t sec, uint8_t mday, uint8_t mon, uint16_t year, uint8_t wday) |
| | void getDateTime(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.

## More Arduino Libraries from Erriez

- [Erriez Libraries](#)

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">ErriezDS1302</a>	
DS1302 RTC class . . . . .	<a href="#">11</a>



## Chapter 3

# File Index

### 3.1 File List

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

src/ <a href="#">ErriezDS1302.cpp</a>	
DS1302 RTC library for Arduino . . . . .	<a href="#">25</a>
src/ <a href="#">ErriezDS1302.h</a>	
DS1302 RTC library for Arduino . . . . .	<a href="#">26</a>



## Chapter 4

# Class Documentation

### 4.1 ErriezDS1302 Class Reference

DS1302 RTC class.

```
#include <ErriezDS1302.h>
```

#### Public Member Functions

- [ErriezDS1302](#) (uint8\_t clkPin, uint8\_t ioPin, uint8\_t cePin)  
*Constructor DS1302 RTC.*
- bool [begin](#) ()  
*Initialize and detect DS1302 RTC.*
- bool [isRunning](#) ()  
*Read RTC CH (Clock Halt) from seconds register.*
- bool [clockEnable](#) (bool enable=true)  
*Enable or disable oscillator.*
- 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 [getDateTime](#) (uint8\_t \*hour, uint8\_t \*min, uint8\_t \*sec, uint8\_t \*mday, uint8\_t \*mon, uint16\_t \*year, uint8\_t \*wday)  
*Get date time.*

- `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 register.*
- `bool readBuffer (uint8_t reg, void *buffer, uint8_t len)`  
*Read buffer from RTC clock registers.*
- `bool writeBuffer (uint8_t reg, void *buffer, uint8_t len)`  
*Write buffer to RTC clock registers.*
- `void writeByteRAM (uint8_t addr, uint8_t value)`  
*Write a byte to RAM.*
- `void writeBufferRAM (uint8_t *buf, uint8_t len)`  
*Write buffer to RAM address 0x00 (burst write)*
- `uint8_t readByteRAM (uint8_t addr)`  
*Read byte from RAM.*
- `void readBufferRAM (uint8_t *buf, uint8_t len)`  
*Read buffer from RAM address 0x00 (burst read)*

### 4.1.1 Detailed Description

DS1302 RTC class.

Definition at line 128 of file ErriezDS1302.h.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 ErriezDS1302()

```
ErriezDS1302::ErriezDS1302 (
    uint8_t clkPin,
    uint8_t ioPin,
    uint8_t cePin )
```

Constructor DS1302 RTC.

#### Parameters

<i>clkPin</i>	Clock pin
<i>ioPin</i>	I/O pin.
<i>cePin</i>	Chip select pin. (In previous versions RST pin which is the same)

Definition at line 44 of file ErriezDS1302.cpp.



### 4.1.3 Member Function Documentation

#### 4.1.3.1 bcdToDec()

```
uint8_t ErriezDS1302::bcdToDec (
    uint8_t bcd )
```

BCD to decimal conversion.

##### Parameters

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

##### Returns

Decimal value.

Definition at line 515 of file ErriezDS1302.cpp.

#### 4.1.3.2 begin()

```
bool ErriezDS1302::begin ( )
```

Initialize and detect DS1302 RTC.

Call this function from setup().

##### Return values

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

Definition at line 70 of file ErriezDS1302.cpp.

#### 4.1.3.3 clockEnable()

```
bool ErriezDS1302::clockEnable (
    bool enable = true )
```

Enable or disable oscillator.

Clear or set CH (Clock Halt) bit to seconds register

**Parameters**

<i>enable</i>	true: Enable RTC clock. false: Stop RTC clock.
---------------	---

**Return values**

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

Definition at line 134 of file ErriezDS1302.cpp.

**4.1.3.4 decToBcd()**

```
uint8_t ErriezDS1302::decToBcd (
    uint8_t dec )
```

Decimal to BCD conversion.

**Parameters**

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

**Returns**

BCD encoded value.

Definition at line 527 of file ErriezDS1302.cpp.

**4.1.3.5 getDateTime()**

```
bool ErriezDS1302::getTime (
    uint8_t * hour,
    uint8_t * min,
    uint8_t * sec,
    uint8_t * mday,
    uint8_t * mon,
    uint16_t * year,
    uint8_t * wday )
```

Get 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>	Get date/time failed.

Definition at line 417 of file ErriezDS1302.cpp.

**4.1.3.6 getEpoch()**

```
time_t ErriezDS1302::getEpoch ( )
```

Read Unix UTC epoch time\_t.

**Returns**

Unix epoch time\_t seconds since 1970.

Definition at line 159 of file ErriezDS1302.cpp.

**4.1.3.7 getTime()**

```
bool ErriezDS1302::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 339 of file ErriezDS1302.cpp.

**4.1.3.8 isRunning()**

```
bool ErriezDS1302::isRunning ( )
```

Read RTC CH (Clock Halt) from seconds register.

The application is responsible for checking the CH (Clock Halt) bit before reading date/time date. This function may be used to judge the validity of the date/time registers.

**Return values**

<i>true</i>	RTC clock is running.
<i>false</i>	The date/time data is invalid when the CH bit is set. The application should enable the oscillator, or program a new date/time.

Definition at line 110 of file ErriezDS1302.cpp.

**4.1.3.9 read()**

```
bool ErriezDS1302::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>	Read failed.

Definition at line 219 of file ErriezDS1302.cpp.

## 4.1.3.10 readBuffer()

```
bool ErriezDS1302::readBuffer (
    uint8_t reg,
    void * buffer,
    uint8_t readLen )
```

Read buffer from RTC clock registers.

## Parameters

<i>reg</i>	RTC register number 0x00..0x07.
<i>buffer</i>	Buffer.
<i>readLen</i>	Buffer length. Reading is only allowed within valid RTC registers.

## Return values

<i>true</i>	Success
<i>false</i>	Read failed.

Definition at line 624 of file ErriezDS1302.cpp.

## 4.1.3.11 readBufferRAM()

```
void ErriezDS1302::readBufferRAM (
    uint8_t * buf,
    uint8_t len )
```

Read buffer from RAM address 0x00 (burst read)

## Parameters

<i>buf</i>	Data buffer
<i>len</i>	Buffer length

Definition at line 498 of file ErriezDS1302.cpp.

## 4.1.3.12 readByteRAM()

```
uint8_t ErriezDS1302::readByteRAM (
    uint8_t addr )
```

Read byte from RAM.

**Parameters**

<i>addr</i>	RAM address 0..0x1E
-------------	---------------------

**Returns**

RAM byte 0..0xFF

Definition at line 479 of file ErriezDS1302.cpp.

**4.1.3.13 readRegister()**

```
uint8_t ErriezDS1302::readRegister (
    uint8_t reg )
```

Read register.

Please refer to the RTC datasheet.

**Parameters**

<i>reg</i>	RTC register number 0x00..0x09.
------------	---------------------------------

**Returns**

value 8-bit unsigned register value.

Definition at line 541 of file ErriezDS1302.cpp.

**4.1.3.14 setDateTime()**

```
bool ErriezDS1302::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
<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 date/time failed.

Definition at line 377 of file ErriezDS1302.cpp.

## 4.1.3.15 setEpoch()

```
bool ErriezDS1302::setEpoch (
    time_t t )
```

Write Unix epoch UTC time to RTC.

## Parameters

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

## Return values

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

Definition at line 191 of file ErriezDS1302.cpp.

## 4.1.3.16 setTime()

```
bool ErriezDS1302::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.
-------------	----------

## Return values

<i>false</i>	Set time failed.
--------------	------------------

Definition at line 308 of file ErriezDS1302.cpp.

## 4.1.3.17 write()

```
bool ErriezDS1302::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.

## Parameters

<i>dt</i>	Date/time struct tm. Providing invalid date/time data may result in unpredictable behavior.
-----------	---

## Return values

<i>true</i>	Success.
<i>false</i>	Write failed.

Definition at line 274 of file ErriezDS1302.cpp.

## 4.1.3.18 writeBuffer()

```
bool ErriezDS1302::writeBuffer (
    uint8_t reg,
    void * buffer,
    uint8_t writeLen )
```

Write buffer to RTC clock registers.

Please refer to the RTC datasheet.

## Parameters

<i>reg</i>	RTC register number 0x00..0x09.
<i>buffer</i>	Buffer.
<i>writeLen</i>	Buffer length. Writing is only allowed within valid RTC registers.



## Return values

<i>true</i>	Success
<i>false</i>	Write failed.

Definition at line 593 of file ErriezDS1302.cpp.

#### 4.1.3.19 writeBufferRAM()

```
void ErriezDS1302::writeBufferRAM (
    uint8_t * buf,
    uint8_t len )
```

Write buffer to RAM address 0x00 (burst write)

## Parameters

<i>buf</i>	Data buffer
<i>len</i>	Buffer length 0x01..0x1E

Definition at line 462 of file ErriezDS1302.cpp.

#### 4.1.3.20 writeByteRAM()

```
void ErriezDS1302::writeByteRAM (
    uint8_t addr,
    uint8_t value )
```

Write a byte to RAM.

## Parameters

<i>addr</i>	RAM address 0..0x1E
<i>value</i>	RAM byte 0..0xFF

Definition at line 447 of file ErriezDS1302.cpp.

#### 4.1.3.21 writeRegister()

```
bool ErriezDS1302::writeRegister (
    uint8_t reg,
    uint8_t value )
```

Write register.

Please refer to the RTC datasheet.

## Parameters

<i>reg</i>	RTC register number 0x00..0x09.
<i>value</i>	8-bit unsigned register value.

## Return values

<i>true</i>	Success
<i>false</i>	Write register failed

Definition at line 567 of file ErriezDS1302.cpp.

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

- src/[ErriezDS1302.h](#)
- src/[ErriezDS1302.cpp](#)



## Chapter 5

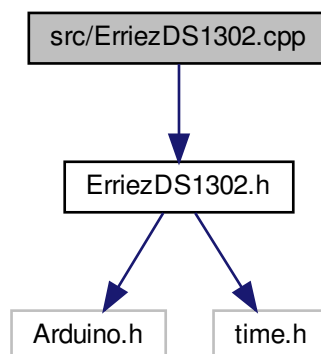
# File Documentation

### 5.1 src/ErriezDS1302.cpp File Reference

DS1302 RTC library for Arduino.

```
#include "ErriezDS1302.h"
```

Include dependency graph for ErriezDS1302.cpp:



#### 5.1.1 Detailed Description

DS1302 RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS1302> Documentation: <https://erriez.github.io/ErriezDS1302>

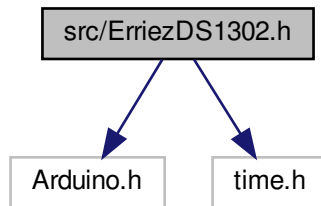
## 5.2 src/ErriezDS1302.h File Reference

DS1302 RTC library for Arduino.

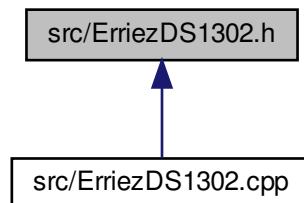
```
#include <Arduino.h>
```

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

Include dependency graph for ErriezDS1302.h:



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



### Classes

- class [ErriezDS1302](#)  
*DS1302 RTC class.*

### Macros

- `#define DS1302_ACB 0x80`  
*DS1302 address/command register.*
- `#define DS1302_ACB_RAM 0x40`  
*Address command RAM.*
- `#define DS1302_ACB_CLOCK 0x00`  
*Address command clock.*

- #define [DS1302\\_ACB\\_READ](#) 0x01  
*Address command read.*
- #define [DS1302\\_ACB\\_WRITE](#) 0x00  
*Address command write.*
- #define [DS1302\\_CMD\\_READ\\_CLOCK\\_REG](#)(reg) (DS1302\_ACB | DS1302\_ACB\_CLOCK | (((reg) & 0x1F) << 1) | DS1302\_ACB\_READ)  
*DS1302 read clock register.*
- #define [DS1302\\_CMD\\_WRITE\\_CLOCK\\_REG](#)(reg) (DS1302\_ACB | DS1302\_ACB\_CLOCK | (((reg) & 0x1F) << 1) | DS1302\_ACB\_WRITE)  
*DS1302 write clock register.*
- #define [DS1302\\_CMD\\_READ\\_CLOCK\\_BURST](#) (DS1302\_ACB | DS1302\_ACB\_CLOCK | 0x3E | DS1302\_ACB\_READ)  
*DS1302 read clock register with burst.*
- #define [DS1302\\_CMD\\_WRITE\\_CLOCK\\_BURST](#) (DS1302\_ACB | DS1302\_ACB\_CLOCK | 0x3E | DS1302\_ACB\_WRITE)  
*DS1302 write clock register with burst.*
- #define [DS1302\\_CMD\\_READ\\_RAM](#)(addr) (DS1302\_ACB | DS1302\_ACB\_RAM | (((addr) & 0x1F) << 1) | DS1302\_ACB\_READ)  
*DS1302 read RAM register.*
- #define [DS1302\\_CMD\\_WRITE\\_RAM](#)(addr) (DS1302\_ACB | DS1302\_ACB\_RAM | (((addr) & 0x1F) << 1) | DS1302\_ACB\_WRITE)  
*DS1302 write RAM register.*
- #define [DS1302\\_CMD\\_READ\\_RAM\\_BURST](#) (DS1302\_ACB | DS1302\_ACB\_RAM | 0x3E | DS1302\_ACB\_READ)  
*DS1302 read RAM register with burst.*
- #define [DS1302\\_CMD\\_WRITE\\_RAM\\_BURST](#) (DS1302\_ACB | DS1302\_ACB\_RAM | 0x3E | DS1302\_ACB\_WRITE)  
*DS1302 write RAM register with burst.*
- #define [DS1302\\_REG\\_SECONDS](#) 0x00  
*DS1302 registers.*
- #define [DS1302\\_REG\\_MINUTES](#) 0x01  
*Minutes register.*
- #define [DS1302\\_REG\\_HOURS](#) 0x02  
*Hours register.*
- #define [DS1302\\_REG\\_DAY\\_MONTH](#) 0x03  
*Day of the month register.*
- #define [DS1302\\_REG\\_MONTH](#) 0x04  
*Month register.*
- #define [DS1302\\_REG\\_DAY\\_WEEK](#) 0x05  
*Day of the week register.*
- #define [DS1302\\_REG\\_YEAR](#) 0x06  
*Year register.*
- #define [DS1302\\_REG\\_WP](#) 0x07  
*Write protect register.*
- #define [DS1302\\_REG\\_TC](#) 0x08  
*Tickle Charger register.*
- #define [DS1302\\_NUM\\_CLOCK\\_REGS](#) 7  
*DS1302 number of RAM registers.*
- #define [DS1302\\_NUM\\_RAM\\_REGS](#) 31
- #define [DS1302\\_SEC\\_CH](#) 7  
*DS1302 register bit defines.*
- #define [DS1302\\_BIT\\_WP](#) 7

- *Write protect bit.*
- `#define DS1302_BIT_READ 0`
- *Bit read.*
- `#define DS1302_TCS_DISABLE 0x5C`
- *Tickle Charger disable value.*
- `#define DS1302_CLK_LOW() { digitalWrite(_clkPin, LOW); }`
- *CLK pin low.*
- `#define DS1302_CLK_HIGH() { digitalWrite(_clkPin, HIGH); }`
- *CLK pin high.*
- `#define DS1302_CLK_INPUT() { pinMode(_clkPin, INPUT); }`
- *CLK pin input.*
- `#define DS1302_CLK_OUTPUT() { pinMode(_clkPin, OUTPUT); }`
- *CLK pin output.*
- `#define DS1302_IO_LOW() { digitalWrite(_ioPin, LOW); }`
- *IO pin low.*
- `#define DS1302_IO_HIGH() { digitalWrite(_ioPin, HIGH); }`
- *IO pin high.*
- `#define DS1302_IO_INPUT() { pinMode(_ioPin, INPUT); }`
- *IO pin input.*
- `#define DS1302_IO_OUTPUT() { pinMode(_ioPin, OUTPUT); }`
- *IO pin output.*
- `#define DS1302_IO_READ() ( digitalRead(_ioPin) )`
- *IO pin read.*
- `#define DS1302_CE_LOW() { digitalWrite(_cePin, LOW); }`
- *CE pin low.*
- `#define DS1302_CE_HIGH() { digitalWrite(_cePin, HIGH); }`
- *CE pin high.*
- `#define DS1302_CE_INPUT() { pinMode(_cePin, INPUT); }`
- *CE pin input.*
- `#define DS1302_CE_OUTPUT() { pinMode(_cePin, OUTPUT); }`
- *CE pin output.*
- `#define DS1302_PIN_DELAY()`
- *Delay between pin changes.*

### 5.2.1 Detailed Description

DS1302 RTC library for Arduino.

Source: <https://github.com/Erriez/ErriezDS1302> Documentation: <https://erriez.github.io/ErriezDS1302>

### 5.2.2 Macro Definition Documentation



### 5.2.2.1 DS1302\_ACB

```
#define DS1302_ACB 0x80
```

DS1302 address/command register.

Address command date/time

Definition at line 40 of file ErriezDS1302.h.

### 5.2.2.2 DS1302\_REG\_SECONDS

```
#define DS1302_REG_SECONDS 0x00
```

DS1302 registers.

Seconds register

Definition at line 64 of file ErriezDS1302.h.

### 5.2.2.3 DS1302\_SEC\_CH

```
#define DS1302_SEC_CH 7
```

DS1302 register bit defines.

Clock halt bit in seconds register

Definition at line 79 of file ErriezDS1302.h.



# Index

- bcdToDec
  - ErriezDS1302, [13](#)
- begin
  - ErriezDS1302, [13](#)
- clockEnable
  - ErriezDS1302, [13](#)
- DS1302\_ACB
  - ErriezDS1302.h, [28](#)
- DS1302\_REG\_SECONDS
  - ErriezDS1302.h, [29](#)
- DS1302\_SEC\_CH
  - ErriezDS1302.h, [29](#)
- decToBcd
  - ErriezDS1302, [14](#)
- ErriezDS1302, [11](#)
  - bcdToDec, [13](#)
  - begin, [13](#)
  - clockEnable, [13](#)
  - decToBcd, [14](#)
  - ErriezDS1302, [12](#)
  - getDateTime, [14](#)
  - getEpoch, [15](#)
  - getTime, [15](#)
  - isRunning, [16](#)
  - read, [16](#)
  - readBuffer, [16](#)
  - readBufferRAM, [17](#)
  - readByteRAM, [17](#)
  - readRegister, [18](#)
  - setDateTime, [18](#)
  - setEpoch, [19](#)
  - setTime, [19](#)
  - write, [20](#)
  - writeBuffer, [20](#)
  - writeBufferRAM, [21](#)
  - writeByteRAM, [21](#)
  - writeRegister, [21](#)
- ErriezDS1302.h
  - DS1302\_ACB, [28](#)
  - DS1302\_REG\_SECONDS, [29](#)
  - DS1302\_SEC\_CH, [29](#)
- getDateTime
  - ErriezDS1302, [14](#)
- getEpoch
  - ErriezDS1302, [15](#)
- getTime
  - ErriezDS1302, [15](#)
- isRunning
  - ErriezDS1302, [16](#)
- read
  - ErriezDS1302, [16](#)
- readBuffer
  - ErriezDS1302, [16](#)
- readBufferRAM
  - ErriezDS1302, [17](#)
- readByteRAM
  - ErriezDS1302, [17](#)
- readRegister
  - ErriezDS1302, [18](#)
- setDateTime
  - ErriezDS1302, [18](#)
- setEpoch
  - ErriezDS1302, [19](#)
- setTime
  - ErriezDS1302, [19](#)
- src/ErriezDS1302.cpp, [25](#)
- src/ErriezDS1302.h, [26](#)
- write
  - ErriezDS1302, [20](#)
- writeBuffer
  - ErriezDS1302, [20](#)
- writeBufferRAM
  - ErriezDS1302, [21](#)
- writeByteRAM
  - ErriezDS1302, [21](#)
- writeRegister
  - ErriezDS1302, [21](#)