

Erriez DS1302 RTC library for Arduino  
1.0.0

Generated by Doxygen 1.8.11



# 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	DS1302 Class Reference . . . . .	11
4.1.1	Detailed Description . . . . .	12
4.1.2	Constructor & Destructor Documentation . . . . .	12
4.1.2.1	DS1302(uint8_t clkPin, uint8_t ioPin, uint8_t cePin) . . . . .	12
4.1.3	Member Function Documentation . . . . .	13
4.1.3.1	bcdToDec(uint8_t bcd) . . . . .	13
4.1.3.2	begin() . . . . .	13
4.1.3.3	decToBcd(uint8_t dec) . . . . .	13
4.1.3.4	getDateTime(DS1302_DateTime *dateTime) . . . . .	14
4.1.3.5	getTime(uint8_t *hour, uint8_t *minute, uint8_t *second) . . . . .	14
4.1.3.6	halt(bool halt) . . . . .	14
4.1.3.7	isHalted() . . . . .	14
4.1.3.8	isWriteProtected() . . . . .	15
4.1.3.9	readBuffer(void *buf, uint8_t len) . . . . .	15
4.1.3.10	readBufferRAM(uint8_t *buf, uint8_t len) . . . . .	15
4.1.3.11	readByte() . . . . .	15

4.1.3.12	readByteRAM(uint8_t addr)	15
4.1.3.13	readClockRegister(uint8_t reg)	16
4.1.3.14	setDateTime(DS1302_DateTime *dateTime)	16
4.1.3.15	setTime(uint8_t hour, uint8_t minute, uint8_t second)	16
4.1.3.16	writeAddrCmd(uint8_t value)	17
4.1.3.17	writeBufferRAM(uint8_t *buf, uint8_t len)	17
4.1.3.18	writeByte(uint8_t value)	17
4.1.3.19	writeByteRAM(uint8_t addr, uint8_t value)	17
4.1.3.20	writeClockRegister(uint8_t reg, uint8_t value)	17
4.1.3.21	writeProtect(bool enable)	18
4.2	DS1302_DateTime Struct Reference	18
4.2.1	Detailed Description	18
<b>5</b>	<b>File Documentation</b>	<b>19</b>
5.1	ErriezDS1302.h File Reference	19
5.1.1	Detailed Description	21
5.1.2	Macro Definition Documentation	21
5.1.2.1	DS1302_ACB	21
5.1.2.2	DS1302_BIT_CH	21
5.1.2.3	DS1302_REG_SECONDS	21
<b>Index</b>		<b>23</b>

# Chapter 1

## DS1302 RTC (Real Time Clock) library for Arduino

This is an optimized 3-wire [DS1302](#) RTC (Real Time Clock) library for Arduino.

### Library features

- Read / write RTC date and time.
- Read / write 31 Bytes battery backedup RTC RAM.
- Programmable trickle charge to charge super-caps / lithium batteries.
- Optimized IO interface for Atmel AVR platform.
- Tested on platforms:
  - 8-bit Atmel AVR ([Arduino UNO](#) / [Nano](#) / [Mini](#) / [Micro](#) / [Leonardo](#) / [Mega2560](#))
  - 32-bit ESP8266 ([WeMos D1 & R2](#) / [Node MCU ESP12E](#))
  - 32-bit ESP32 ([WeMos LOLIN32 + OLED](#))
- Supported IDE's:
  - [Arduino IDE](#) (v1.8.5)
  - [CLion](#) (2018.1)
  - [Atom](#) / [PlatformIO](#) with CI (Continuous Integration)
  - [Atmel Studio](#) (7.0)

### [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 alarms.
- [Benchmark](#): Benchmark library.
- [GettingStarted](#): Getting started example.
- [PrintDateTime](#): Print date and time with PROGMEM strings.
- [RAM](#): Read/write RTC RAM.
- [SetDateTime](#): Set date time.
- [SetTrickleCharger](#): Program trickle battery/capacitor charger.
- [SquareWave1Hz](#): 1Hz square wave output on DIGITAL pin.
- [Terminal](#) and [Python script](#) to set date time.

## Documentation

- [Online HTML](#)
- [Download PDF](#).
- [DS1302 datasheet](#).

## Usage

### Initialization

```
1 {c++}
2 #include <ErriezDS1302.h>
3
4 // Connect DS1302 data pin to Arduino DIGITAL pin
5 #if defined(ARDUINO_ARCH_AVR)
6 #define DS1302_CLK_PIN      2
7 #define DS1302_IO_PIN      3
8 #define DS1302_CE_PIN      4
9 #elif defined(ARDUINO_ARCH_ESP8266)
10 #define DS1302_CLK_PIN     D4
11 #define DS1302_IO_PIN      D3
12 #define DS1302_CE_PIN      D2
13 #elif defined(ARDUINO_ARCH_ESP32)
14 #define DS1302_CLK_PIN     0
15 #define DS1302_IO_PIN      4
16 #define DS1302_CE_PIN      5
17 #else
18 #error #error "May work, but not tested on this target"
19 #endif
20
21 // Create DS1302 RTC object
22 DS1302 rtc = DS1302(DS1302_CLK_PIN, DS1302_IO_PIN, DS1302_CE_PIN);
23
24 void setup()
25 {
26     bool running;
27
28     // Initialize RTC
29     running = rtc.begin();
30 }
```

### Set date and time

```
1 {C++}
2 DS1302_DateTime dt;
3
4 // Set initial date and time
5 dt.second = 0;
6 dt.minute = 41;
7 dt.hour = 22;
8 dt.dayWeek = 6; // 1 = Monday
9 dt.dayMonth = 21;
10 dt.month = 4;
11 dt.year = 2018;
12 rtc.setDateTime(&dt);
```

## Get date and time

```
1 {c++}
2 DS1302_DateTime dt;
3 char buf[32];
4
5 // Get RTC date and time
6 if (!rtc.getDateTime(&dt)) {
7     Serial.println(F("Error: DS1302 read failed"));
8 } else {
9     sprintf(buf, sizeof(buf), "%d %02d-%02d-%d %d:%02d:%02d",
10         dt.dayWeek, dt.dayMonth, dt.month, dt.year, dt.hour, dt.minute, dt.second);
11     Serial.println(buf);
12 }
```

## Set time

```
1 {c++}
2 // Set time
3 rtc.setTime(12, 0, 0);
```

## Get time

```
1 {c++}
2 uint8_t hour;
3 uint8_t minute;
4 uint8_t second;
5 char buf[10];
6
7 // Read RTC time
8 if (!rtc.getTime(&hour, &minute, &second)) {
9     Serial.println(F("Error: DS1302 read failed"));
10 } else {
11     // Print time
12     sprintf(buf, sizeof(buf), "%d:%02d:%02d", hour, minute, second);
13     Serial.println(buf);
14 }
```

## Write to RTC RAM

```
1 {c++}
2 // Write Byte to RTC RAM
3 rtc.writeByteRAM(0x02, 0xA9);
4
5 // Write buffer to RTC RAM
6 uint8_t buf[NUM_DS1302_RAM_REGS] = { 0x00 };
7 rtc.writeBufferRAM(buf, sizeof(buf));
```

## Read from RTC RAM

```
1 {c++}
2 // Read byte from RTC RAM
3 uint8_t dataByte = rtc.readByteRAM(0x02);
4
5 // Read buffer from RTC RAM
6 uint8_t buf[NUM_DS1302_RAM_REGS];
7 rtc.readBufferRAM(buf, sizeof(buf));
```

## Set Trickle Charger

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

```
1 {c++}
2 // Disable (default)
3 rtc.writeClockRegister(DS1302_REG_TC, DS1302_TCS_DISABLE);
4
5 // Minimum 2 Diodes, 8kOhm
6 rtc.writeClockRegister(DS1302_REG_TC, 0xAB);
7
8 // Maximum 1 Diode, 2kOhm
9 rtc.writeClockRegister(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:

```
1 {c++}
2 // Install Pyserial
3 python3 pip -m pyserial
4
5 // Set RTC date and time
6 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

## Benchmark results

### Arduino UNO (AVR F\_CPU = 16MHz)

```
1 DS1302 RTC benchmark
2
3 rtc.begin(): 160us
4 rtc.writeProtect(false): 148us
5 rtc.halt(false): 144us
6 rtc.setDateTime(&dt): 720us
7 rtc.getDateTime(&dt): 496us
8 rtc.setTime(12, 0, 0): 1224us
9 rtc.getTime(&hour, &minute, &second): 272us
10 rtc.writeRAM(0x00, 0xFF): 144us
11 rtc.writeRAM(buf, sizeof(buf)): 1796us
12 rtc.readRAM(0x00): 140us
13 rtc.readRAM(buf, sizeof(buf)): 1812us
```



### WeMos D1 & R2 (ESP8266 F\_CPU = 80MHz)

```
1 DS1302 RTC benchmark
2
3 rtc.begin(): 180us
4 rtc.writeProtect(false): 112us
5 rtc.halt(false): 149us
6 rtc.setDateTime(&dt): 369us
7 rtc.getDateTime(&dt): 273us
8 rtc.setTime(12, 0, 0): 571us
9 rtc.getTime(&hour, &minute, &second): 154us
10 rtc.writeRAM(0x00, 0xFF): 86us
11 rtc.writeRAM(buf, sizeof(buf)): 852us
12 rtc.readRAM(0x00): 84us
13 rtc.readRAM(buf, sizeof(buf)): 881us
```

### WeMos D1 & R2 (ESP8266 F\_CPU = 160MHz)

```
1 DS1302 RTC benchmark
2
3 rtc.begin(): 152us
4 rtc.writeProtect(false): 73us
5 rtc.halt(false): 108us
6 rtc.setDateTime(&dt): 257us
7 rtc.getDateTime(&dt): 187us
8 rtc.setTime(12, 0, 0): 373us
9 rtc.getTime(&hour, &minute, &second): 105us
10 rtc.writeRAM(0x00, 0xFF): 62us
11 rtc.writeRAM(buf, sizeof(buf)): 553us
12 rtc.readRAM(0x00): 62us
13 rtc.readRAM(buf, sizeof(buf)): 568us
```

## 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:

<a href="#">DS1302</a>		
<a href="#">DS1302</a>	<a href="#">DS1302</a> RTC class . . . . .	<a href="#">11</a>
<a href="#">DS1302_DateTime</a>		
	<a href="#">Date time structure</a> . . . . .	<a href="#">18</a>



## Chapter 3

# File Index

### 3.1 File List

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

<b>ErriezDS1302.cpp</b>	??
<a href="#">ErriezDS1302.h</a>	
<a href="#">DS1302</a> RTC library for Arduino	<a href="#">19</a>



## Chapter 4

# Class Documentation

### 4.1 DS1302 Class Reference

DS1302 RTC class.

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

#### Public Member Functions

- [DS1302](#) (uint8\_t clkPin, uint8\_t ioPin, uint8\_t cePin)  
*Constructor DS1302 RTC.*
- virtual bool [begin](#) ()  
*Initialize DS1302.*
- virtual void [writeProtect](#) (bool enable)  
*Set write protect flag.*
- virtual bool [isWriteProtected](#) ()  
*Get write protect state.*
- virtual void [halt](#) (bool halt)  
*Set RTC clock halted or running.*
- virtual bool [isHalted](#) ()  
*Get RTC halt status.*
- virtual void [setDateTime](#) (DS1302\_DateTime \*dateTime)  
*Set RTC date and time.*
- virtual bool [getDateTime](#) (DS1302\_DateTime \*dateTime)  
*Get RTC date and time.*
- virtual void [setTime](#) (uint8\_t hour, uint8\_t minute, uint8\_t second)  
*Set RTC time.*
- virtual bool [getTime](#) (uint8\_t \*hour, uint8\_t \*minute, uint8\_t \*second)  
*Get RTC time.*
- virtual void [writeClockRegister](#) (uint8\_t reg, uint8\_t value)  
*Write clock register.*
- virtual uint8\_t [readClockRegister](#) (uint8\_t reg)  
*Read clock register.*
- virtual void [writeByteRAM](#) (uint8\_t addr, uint8\_t value)  
*Write a byte to RAM.*

- virtual void [writeBufferRAM](#) (uint8\_t \*buf, uint8\_t len)  
*Write buffer to RAM address 0x00 (burst write)*
- virtual uint8\_t [readByteRAM](#) (uint8\_t addr)  
*Read byte from RAM.*
- virtual void [readBufferRAM](#) (uint8\_t \*buf, uint8\_t len)  
*Read buffer from RAM address 0x00 (burst read)*

## Protected Member Functions

- virtual void [transferBegin](#) ()  
*Start RTC transfer.*
- virtual void [transferEnd](#) ()  
*End RTC transfer.*
- virtual void [writeAddrCmd](#) (uint8\_t value)  
*Write address/command byte.*
- virtual void [writeByte](#) (uint8\_t value)  
*Write byte.*
- virtual uint8\_t [readByte](#) ()  
*Read Byte from RTC.*
- virtual void [readBuffer](#) (void \*buf, uint8\_t len)  
*Read buffer from DS1302.*
- virtual uint8\_t [bcdToDec](#) (uint8\_t bcd)  
*BCD to decimal conversion.*
- virtual uint8\_t [decToBcd](#) (uint8\_t dec)  
*Decimal to BCD conversion.*

## Protected Attributes

- uint8\_t [\\_clkPin](#)  
*Clock pin.*
- uint8\_t [\\_ioPin](#)  
*Data pin.*
- uint8\_t [\\_cePin](#)  
*Chip enable pin.*

### 4.1.1 Detailed Description

[DS1302](#) RTC class.

Definition at line 139 of file ErriezDS1302.h.

### 4.1.2 Constructor & Destructor Documentation

4.1.2.1 [DS1302::DS1302](#) ( uint8\_t *clkPin*, uint8\_t *ioPin*, uint8\_t *cePin* ) [explicit]

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 `uint8_t DS1302::bcdToDec ( uint8_t bcd )` `[protected]`, `[virtual]`

BCD to decimal conversion.

## Parameters

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

## Returns

Decimal value

Definition at line 485 of file ErriezDS1302.cpp.

#### 4.1.3.2 `bool DS1302::begin ( )` `[virtual]`

Initialize [DS1302](#).

Call this function from `setup()`.

## Returns

true: RTC running false: RTC halted or not detected

Definition at line 70 of file ErriezDS1302.cpp.

#### 4.1.3.3 `uint8_t DS1302::decToBcd ( uint8_t dec )` `[protected]`, `[virtual]`

Decimal to BCD conversion.

## Parameters

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

**Returns**

BCD encoded value

Definition at line 497 of file ErriezDS1302.cpp.

**4.1.3.4** `bool DS1302::getDateTime ( DS1302_DateTime * dateTime )` [virtual]

Get RTC date and time.

**Parameters**

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

Definition at line 183 of file ErriezDS1302.cpp.

**4.1.3.5** `bool DS1302::getTime ( uint8_t * hour, uint8_t * minute, uint8_t * second )` [virtual]

Get RTC time.

**Parameters**

<i>hour</i>	Hours
<i>minute</i>	Minutes
<i>second</i>	Seconds

Definition at line 241 of file ErriezDS1302.cpp.

**4.1.3.6** `void DS1302::halt ( bool halt )` [virtual]

Set RTC clock halted or running.

**Parameters**

<i>halt</i>	true: Enable RTC clock false: Halt RTC clock
-------------	--

Definition at line 120 of file ErriezDS1302.cpp.

**4.1.3.7** `bool DS1302::isHalted ( )` [virtual]

Get RTC halt status.

**Returns**

true: RTC clock is halted false: RTC clock is running

Definition at line 143 of file ErriezDS1302.cpp.

#### 4.1.3.8 bool DS1302::isWriteProtected ( ) [virtual]

Get write protect state.

##### Returns

true: RTC registers are read only false: RTC registers are writable

Definition at line 105 of file ErriezDS1302.cpp.

#### 4.1.3.9 void DS1302::readBuffer ( void \* *buf*, uint8\_t *len* ) [protected],[virtual]

Read buffer from [DS1302](#).

##### Parameters

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

Definition at line 471 of file ErriezDS1302.cpp.

#### 4.1.3.10 void DS1302::readBufferRAM ( uint8\_t \* *buf*, uint8\_t *len* ) [virtual]

Read buffer from RAM address 0x00 (burst read)

##### Parameters

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

Definition at line 325 of file ErriezDS1302.cpp.

#### 4.1.3.11 uint8\_t DS1302::readByte ( ) [protected],[virtual]

Read Byte from RTC.

##### Returns

Data Byte

Definition at line 444 of file ErriezDS1302.cpp.

#### 4.1.3.12 uint8\_t DS1302::readByteRAM ( uint8\_t *addr* ) [virtual]

Read byte from RAM.

**Parameters**

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

**Returns**

RAM byte 0..0xFF

Definition at line 306 of file ErriezDS1302.cpp.

**4.1.3.13** `uint8_t DS1302::readClockRegister ( uint8_t reg )` `[virtual]`

Read clock register.

**Parameters**

<i>reg</i>	RTC clock register (See datasheet)
------------	------------------------------------

**Returns**

Register value (See datasheet)

Definition at line 358 of file ErriezDS1302.cpp.

**4.1.3.14** `void DS1302::setDateTime ( DS1302_DateTime * dateTime )` `[virtual]`

Set RTC date and time.

**Parameters**

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

Definition at line 157 of file ErriezDS1302.cpp.

**4.1.3.15** `void DS1302::setTime ( uint8_t hour, uint8_t minute, uint8_t second )` `[virtual]`

Set RTC time.

**Parameters**

<i>hour</i>	Hours
<i>minute</i>	Minutes
<i>second</i>	Seconds

Definition at line 224 of file ErriezDS1302.cpp.

4.1.3.16 `void DS1302::writeAddrCmd ( uint8_t value )` [protected], [virtual]

Write address/command byte.

#### Parameters

<i>value</i>	Address/command byte
--------------	----------------------

Definition at line 397 of file ErriezDS1302.cpp.

4.1.3.17 `void DS1302::writeBufferRAM ( uint8_t* buf, uint8_t len )` [virtual]

Write buffer to RAM address 0x00 (burst write)

#### Parameters

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

Definition at line 289 of file ErriezDS1302.cpp.

4.1.3.18 `void DS1302::writeByte ( uint8_t value )` [protected], [virtual]

Write byte.

#### Parameters

<i>value</i>	Data byte
--------------	-----------

Definition at line 423 of file ErriezDS1302.cpp.

4.1.3.19 `void DS1302::writeByteRAM ( uint8_t addr, uint8_t value )` [virtual]

Write a byte to RAM.

#### Parameters

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

Definition at line 274 of file ErriezDS1302.cpp.

4.1.3.20 `void DS1302::writeClockRegister ( uint8_t reg, uint8_t value )` [virtual]

Write clock register.

## Parameters

<i>reg</i>	RTC clock register (See datasheet)
<i>value</i>	Register value (See datasheet)

Definition at line 343 of file ErriezDS1302.cpp.

#### 4.1.3.21 void DS1302::writeProtect ( bool *enable* ) [virtual]

Set write protect flag.

## Parameters

<i>enable</i>	true: Enable RTC write protect false: Disable RTC write protect
---------------	---

Definition at line 94 of file ErriezDS1302.cpp.

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

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

## 4.2 DS1302\_DateTime Struct Reference

Date time structure.

```
#include <ErriezDS1302.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.*

### 4.2.1 Detailed Description

Date time structure.

Definition at line 127 of file ErriezDS1302.h.

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

- [ErriezDS1302.h](#)

## Chapter 5

# File Documentation

### 5.1 ErriezDS1302.h File Reference

[DS1302](#) RTC library for Arduino.

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

#### Classes

- struct [DS1302\\_DateTime](#)  
*Date time structure.*
- class [DS1302](#)  
*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 NUM\_DS1302\_RAM\_REGS 31
- DS1302 number of RAM registers.*

  - #define DS1302\_BIT\_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.*

### 5.1.1 Detailed Description

`DS1302` RTC library for Arduino.

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

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 #define DS1302\_ACB 0x80

`DS1302` address/command register.

Address command date/time

Definition at line 39 of file ErriezDS1302.h.

#### 5.1.2.2 #define DS1302\_BIT\_CH 7

`DS1302` register bit defines.

Clock halt bit

Definition at line 77 of file ErriezDS1302.h.

#### 5.1.2.3 #define DS1302\_REG\_SECONDS 0x00

`DS1302` registers.

Seconds register

Definition at line 63 of file ErriezDS1302.h.



# Index

bcdToDec  
    DS1302, [13](#)

begin  
    DS1302, [13](#)

DS1302, [11](#)  
    bcdToDec, [13](#)  
    begin, [13](#)  
    DS1302, [12](#)  
    decToBcd, [13](#)  
    getDateTime, [14](#)  
    getTime, [14](#)  
    halt, [14](#)  
    isHalted, [14](#)  
    isWriteProtected, [14](#)  
    readBuffer, [15](#)  
    readBufferRAM, [15](#)  
    readByte, [15](#)  
    readByteRAM, [15](#)  
    readClockRegister, [16](#)  
    setDateTime, [16](#)  
    setTime, [16](#)  
    writeAddrCmd, [16](#)  
    writeBufferRAM, [17](#)  
    writeByte, [17](#)  
    writeByteRAM, [17](#)  
    writeClockRegister, [17](#)  
    writeProtect, [18](#)

DS1302\_ACB  
    ErriezDS1302.h, [21](#)

DS1302\_BIT\_CH  
    ErriezDS1302.h, [21](#)

DS1302\_DateTime, [18](#)

DS1302\_REG\_SECONDS  
    ErriezDS1302.h, [21](#)

decToBcd  
    DS1302, [13](#)

ErriezDS1302.h, [19](#)  
    DS1302\_ACB, [21](#)  
    DS1302\_BIT\_CH, [21](#)  
    DS1302\_REG\_SECONDS, [21](#)

getDateTime  
    DS1302, [14](#)

getTime  
    DS1302, [14](#)

halt  
    DS1302, [14](#)

isHalted  
    DS1302, [14](#)

isWriteProtected  
    DS1302, [14](#)

readBuffer  
    DS1302, [15](#)

readBufferRAM  
    DS1302, [15](#)

readByte  
    DS1302, [15](#)

readByteRAM  
    DS1302, [15](#)

readClockRegister  
    DS1302, [16](#)

setDateTime  
    DS1302, [16](#)

setTime  
    DS1302, [16](#)

writeAddrCmd  
    DS1302, [16](#)

writeBufferRAM  
    DS1302, [17](#)

writeByte  
    DS1302, [17](#)

writeByteRAM  
    DS1302, [17](#)

writeClockRegister  
    DS1302, [17](#)

writeProtect  
    DS1302, [18](#)