Erriez DS1302 RTC library for Arduino

2.0.0

Generated by Doxygen 1.8.13

Contents

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 backupped 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 DS1302 RTC object
ErriezDS1302 ds1302 = ErriezDS1302(DS1302_CLK_PIN, DS1302_IO_PIN, DS1302_CE_PIN);

void setup()
{
    // Initialize RTC
    while (!ds1302.begin()) {
        Serial.println(F("RTC not found"));
        delay(3000);
    }
}
```

Check oscillator status at startup

```
{c++}
// Check oscillator status
if (!ds1302.isRunning()) {
    // Error: DS1302 RTC oscillator stopped. Date/time cannot be trusted.
    // Set new date/time before reading date/time.

    // Enable oscillator
    ds1302.clockEnable(true);
}
```

Set time

```
{c++}
// Write time to RTC
ds1302.setTime(12, 0, 0);
```

Get time

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

// Read time from RTC
if (!ds1302.getTime(&hour, &minute, &second)) {
    // Error: RTC read failed
}
```

Set date and time

```
{c++}
// Write RTC date/time: 13:45:09  31 December 2019  2=Tuesday
if (!ds1302.setDateTime(13, 45, 9,  31, 12, 2019,  2) {
    // Error: RTC write failed
}
```

Get date/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 (!ds1307.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

ds1302.write(&dt);
```

Read date/time struct tm

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

// Read RTC date/time
if (!ds1307.read(&dt)) {
    // Error: RTC read failed
}
```

Read Unix Epoch UTC

```
{c++}
time_t t;

// Read Unix epoch UTC from RTC
if (!ds1307.getEpoch(&t)) {
    // Error: RTC read failed
}
```

Write Unix Epoch UTC

```
{c++}
// Write Unix epoch UTC to RTC
if (!ds1307.setEpoch(1599416430UL)) {
    // Error: Set epoch failed
}
```

Write to RTC RAM

```
{c++}
// Write Byte to RTC RAM
ds1302.writeByteRAM(0x02, 0xA9);

// Write buffer to RTC RAM
uint8_t buf[NUM_DS1302_RAM_REGS] = { 0x00 };
ds1302.writeBufferRAM(buf, sizeof(buf));
```

Read from RTC RAM

```
{c++}
// Read byte from RTC RAM
uint8_t dataByte = ds1302.readByteRAM(0x02);

// Read buffer from RTC RAM
uint8_t buf[NUM_DS1302_RAM_REGS];
ds1302.readBufferRAM(buf, sizeof(buf));
```

Set Trickle Charger

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

```
{c++}
// Disable (default)
ds1302.writeRegister(DS1302_REG_TC, DS1302_TCS_DISABLE);

// Minimum 2 Diodes, 8kOhm
ds1302.writeRegister(DS1302_REG_TC, 0xAB);

// Maximum 1 Diode, 2kOhm
ds1302.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()` | | `isHalted()` | `bool clockEnable(bool enable)` | | `halt()` | `void isRunning()` | | `setDateTime()` | `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)` |

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:

Chapter 3

File Index

3.1 File List

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

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

clkPin	Clock pin
ioPin	I/O pin.
cePin	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

bcd	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

true	RTC detected.
false	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

enable	true: Enable RTC clock. false: Stop RTC clock.

Return values

true	Success.
false	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

dec	Decimal value.

Returns

BCD encoded value.

Definition at line 527 of file ErriezDS1302.cpp.

4.1.3.5 getDateTime()

```
bool ErriezDS1302::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.

Parameters

hour	Hours 0..23
min	Minutes 0..59

Parameters

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

Return values

true	Success.
false	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

hour	Hours 0..23.
min	Minutes 0..59.
sec	Seconds 0..59.

Return values

true	Success.
false	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

true	RTC clock is running.
false	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

dt	Date and time struct tm.

Return values

true	Success
false	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

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

Return values

true	Success
false	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

buf	Data buffer
len	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

addr	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

reg	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

hour	Hours 0..23
min	Minutes 0..59
sec	Seconds 0..59
mday	Day of the month 1..31
mon	Month 1..12 (1=January)
year	Year 2000..2099
wday	Day of the week 0..6 (0=Sunday, .. 6=Saturday)

Return values

true	Success.
false	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

t	time_t time

Return values

true	Success.
false	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

hour	Hours 0..23.
min	Minutes 0..59.
sec	Seconds 0..59.

Return values

true	Success.

Return values

false	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

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

Return values

true	Success.
false	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

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

Return values

true	Success
false	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

buf	Data buffer
len	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

addr	RAM address 0..0x1E
value	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

reg	RTC register number 0x00..0x09.
value	8-bit unsigned register value.

Return values

true	Success
false	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

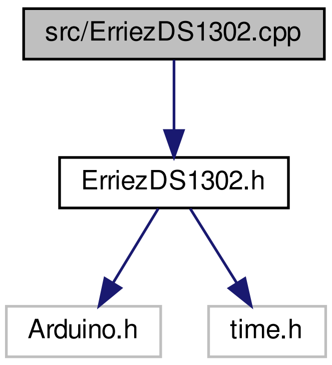
Chapter 5

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.

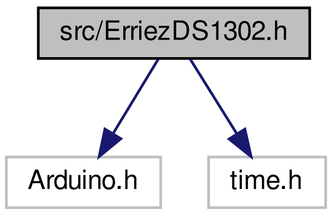
Source: https://github.com/Erriez/ErriezDS1302 Documentation: https://erriez.↵
github.io/ErriezDS1302

5.2 src/ErriezDS1302.h File Reference

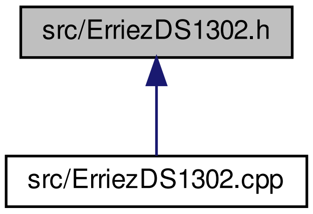
DS1302 RTC library for Arduino.

```
#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 | D←S1302_ACB_WRITE)

  DS1302 writeclock 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_AC←B_READ)

  DS1302 read RAM register with burst.
- #define DS1302_CMD_WRITE_RAM_BURST (DS1302_ACB | DS1302_ACB_RAM | 0x3E | DS1302_A←CB_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