DS1302 RTC library for Arduino

1.0.0

Generated by Doxygen 1.8.11

Contents

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 backupped 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

```
{c++}
#include <DS1302.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
DS1302 rtc = DS1302(DS1302_CLK_PIN, DS1302_IO_PIN, DS1302_CE_PIN);

void setup()
{
    bool running;

    // Initialize RTC
    running = rtc.begin();
}
```

Set date and time

```
{C++}
DS1302_DateTime dt;

// Set initial date and time
dt.second = 0;
dt.minute = 41;
dt.hour = 22;
dt.dayWeek = 6; // 1 = Monday
dt.dayMonth = 21;
dt.month = 4;
dt.year = 2018;
rtc.setDateTime(&dt);
```

Get date and time

```
{c++}
DS1302_DateTime dt;
char buf[32];

// Get RTC date and time
if (!rtc.getDateTime(&dt)) {
    Serial.println(F("Error: DS1302 read failed"));
} else {
    snprintf(buf, sizeof(buf), "%d %02d-%02d-%d %d:%02d:%02d",
             dt.dayWeek, dt.dayMonth, dt.month, dt.year, dt.hour, dt.minute, dt.second);
     Serial.println(buf);
}
```

Set time

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

Get time

```
{c++}
uint8_t hour;
uint8_t minute;
uint8_t second;
char buf[10];

// Read RTC time
if (!rtc.getTime(&hour, &minute, &second)) {
    Serial.println(F("Error: DS1302 read failed"));
} else {
     // Print time
     snprintf(buf, sizeof(buf), "%d:%02d:%02d", hour, minute, second);
     Serial.println(buf);
}
```

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.writeClockRegister(DS1302_REG_TC, DS1302_TCS_DISABLE);

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

// Maximum 1 Diode, 2kOhm
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:

```
{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

Benchmark results

Arduino UNO (AVR F_CPU = 16MHz)

```
DS1302 RTC benchmark

rtc.begin(): 160us
rtc.writeProtect(false): 148us
rtc.halt(false): 144us
rtc.setDateTime(&dt): 720us
rtc.getDateTime(&dt): 496us
rtc.setTime(12, 0, 0): 1224us
rtc.getTime(&hour, &minute, &second): 272us
rtc.writeRAM(0x00, 0xFF): 144us
rtc.writeRAM(buf, sizeof(buf): 1796us
rtc.readRAM(0x00): 140us
rtc.readRAM(buf, sizeof(buf)): 1812us
```

WeMos D1 & R2 (ESP8266 F_CPU = 80MHz)

```
DS1302 RTC benchmark

rtc.begin(): 180us
rtc.writeProtect(false): 112us
rtc.halt(false): 149us
rtc.setDateTime(&dt): 369us
rtc.getDateTime(&dt): 273us
rtc.setTime(12, 0, 0): 571us
rtc.getTime(&hour, &minute, &second): 154us
rtc.writeRAM(0x00, 0xFF): 86us
rtc.writeRAM(buf, sizeof(buf): 852us
rtc.readRAM(0x00): 84us
rtc.readRAM(buf, sizeof(buf)): 881us
```

WeMos D1 & R2 (ESP8266 F_CPU = 160MHz)

```
DS1302 RTC benchmark

rtc.begin(): 152us
rtc.writeProtect(false): 73us
rtc.halt(false): 108us
rtc.setDateTime(&dt): 257us
rtc.getDateTime(&dt): 187us
rtc.setTime(12, 0, 0): 373us
rtc.getTime(&hour, &minute, &second): 105us
rtc.writeRAM(0x00, 0xFF): 62us
rtc.writeRAM(buf, sizeof(buf): 553us
rtc.readRAM(0x00): 62us
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:

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 DS1302 Class Reference

DS1302 RTC class.

`#include <DS1302.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 DS1302.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

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 DS1302.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

bcd	BCD encoded value

Returns

Decimal value

Definition at line 485 of file DS1302.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 DS1302.cpp.

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

Decimal to BCD conversion.

Parameters

dec	Decimal value

Returns

BCD encoded value

Definition at line 497 of file DS1302.cpp.

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

Get RTC date and time.

Parameters

dateTime	Date and time structure

Definition at line 183 of file DS1302.cpp.

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

Get RTC time.

Parameters

hour	Hours
minute	Minutes
second	Seconds

Definition at line 241 of file DS1302.cpp.

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

Set RTC clock halted or running.

Parameters

halt	true: Enable RTC clock false: Halt RTC clock

Definition at line 120 of file DS1302.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 DS1302.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 DS1302.cpp.

4.1.3.9 void DS1302::readBuffer (void ∗ *buf*, uint8_t *len*) `[protected]`, `[virtual]`

Read buffer from DS1302.

Parameters

buf	Buffer
len	Buffer length

Definition at line 471 of file DS1302.cpp.

4.1.3.10 void DS1302::readBufferRAM (uint8_t ∗ *buf*, uint8_t *len*) `[virtual]`

Read buffer from RAM address 0x00 (burst read)

Parameters

buf	Data buffer
len	Buffer length

Definition at line 325 of file DS1302.cpp.

4.1.3.11 uint8_t DS1302::readByte ( ) `[protected]`, `[virtual]`

Read Byte from RTC.

Returns

Data Byte

Definition at line 444 of file DS1302.cpp.

4.1.3.12 uint8_t DS1302::readByteRAM (uint8_t *addr*) `[virtual]`

Read byte from RAM.

Parameters

addr	RAM address 0..0x1E

Returns

RAM byte 0..0xFF

Definition at line 306 of file DS1302.cpp.

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

Read clock register.

Parameters

reg	RTC clock register (See datasheet)

Returns

Register value (See datasheet)

Definition at line 358 of file DS1302.cpp.

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

Set RTC date and time.

Parameters

dateTime	Date time structure

Definition at line 157 of file DS1302.cpp.

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

Set RTC time.

Parameters

hour	Hours
minute	Minutes
second	Seconds

Definition at line 224 of file DS1302.cpp.

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

Write address/command byte.

Parameters

value	Address/command byte

Definition at line 397 of file DS1302.cpp.

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

Write buffer to RAM address 0x00 (burst write)

Parameters

buf	Data buffer
len	Buffer length 0x01..0x1E

Definition at line 289 of file DS1302.cpp.

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

Write byte.

Parameters

value	Data byte

Definition at line 423 of file DS1302.cpp.

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

Write a byte to RAM.

Parameters

addr	RAM address 0..0x1E
value	RAM byte 0..0xFF

Definition at line 274 of file DS1302.cpp.

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

Write clock register.

Parameters

reg	RTC clock register (See datasheet)
value	Register value (See datasheet)

Definition at line 343 of file DS1302.cpp.

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

Set write protect flag.

Parameters

enable	true: Enable RTC write protect false: Disable RTC write protect

Definition at line 94 of file DS1302.cpp.

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

- DS1302.h
- DS1302.cpp

4.2 DS1302_DateTime Struct Reference

Date time structure.

`#include <DS1302.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 DS1302.h.

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

- DS1302.h

Chapter 5

File Documentation

5.1 DS1302.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 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_ACB←_READ)

    DS1302 read RAM register with burst.
- #define DS1302_CMD_WRITE_RAM_BURST (DS1302_ACB | DS1302_ACB_RAM | 0x3E | DS1302_AC←B_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 DS1302.h.

5.1.2.2 #define DS1302_BIT_CH 7

DS1302 register bit defines.

Clock halt bit

Definition at line 77 of file DS1302.h.

5.1.2.3 #define DS1302_REG_SECONDS 0x00

DS1302 registers.

Seconds register

Definition at line 63 of file DS1302.h.

Index