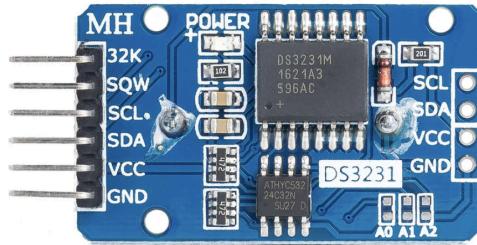


RTC



Source: <https://www.analog.com/media/en/technical-documentation/data-sheets/ds3231.pdf>

Library Overview

The RTC Library provides a compact driver for the DS3231 real-time clock over I₂C for the RFID Security Access System. It supports setting and reading calendar time (24-hour format), reading day-of-week, and retrieving the on-chip temperature sensor. The library also clears the Oscillator Stop Flag (OSF) to ensure valid time after power events.

Key Features

- **Precise RTC Control:** DS3231 timekeeping, 24-hour format, day/date/month/year
- **Temperature Sensing:** Built-in sensor with 0.25°C resolution
- **I₂C Interface:** Simple single-register read/write helpers
- **Robust Startup:** Clears OSF and disables SQW output by default
- **Simple API:** Minimal, easy-to-use functions and a single time struct

Use Cases

- Timestamping PIN entries and RFID scans
- Displaying current time/date on LCD
- Environmental readout using the RTC's internal temperature sensor

Function Summary

Function Name	Description
rtc_init(void)	<p>Initializes the DS3231 RTC device. Clears oscillator stop flag and configures control settings</p> <p>Input: none</p> <p>Return: none</p> <p>Usage:</p> <pre>int main(void) { rtc_init(); // Call once during system initialization while(1) { // Main program loop } }</pre>
rtc_set_time(rtc_time_t *time)	<p>Sets the current time and date in the DS3231.</p> <p>Input: time - pointer to structure containing time data</p> <p>Return: none</p> <p>Usage:</p> <pre>rtc_time_t time = {...}; rtc_set_time(&time);</pre>
rtc_get_time(rtc_time_t *time)	<p>Retrieves the current time and date from the DS3231. BCD values are converted to decimal, and mask clear control bits.</p> <p>Input: time - pointer to structure for holding time data</p> <p>Return: none</p> <p>Usage:</p> <pre>rtc_time_t time = {0}; rtc_get_time(&time); // time now contains valid values</pre>
rtc_get_temperature(void)	<p>Reads temperature from DS3231 sensor in 0.25°C increments.</p> <p>Input: none</p> <p>Return: temp - floating point value of IC temperature in celsius.</p> <p>Usage:</p> <pre>chip_temp tmp = rtc_get_temperature(); //</pre>

Wiring Diagram

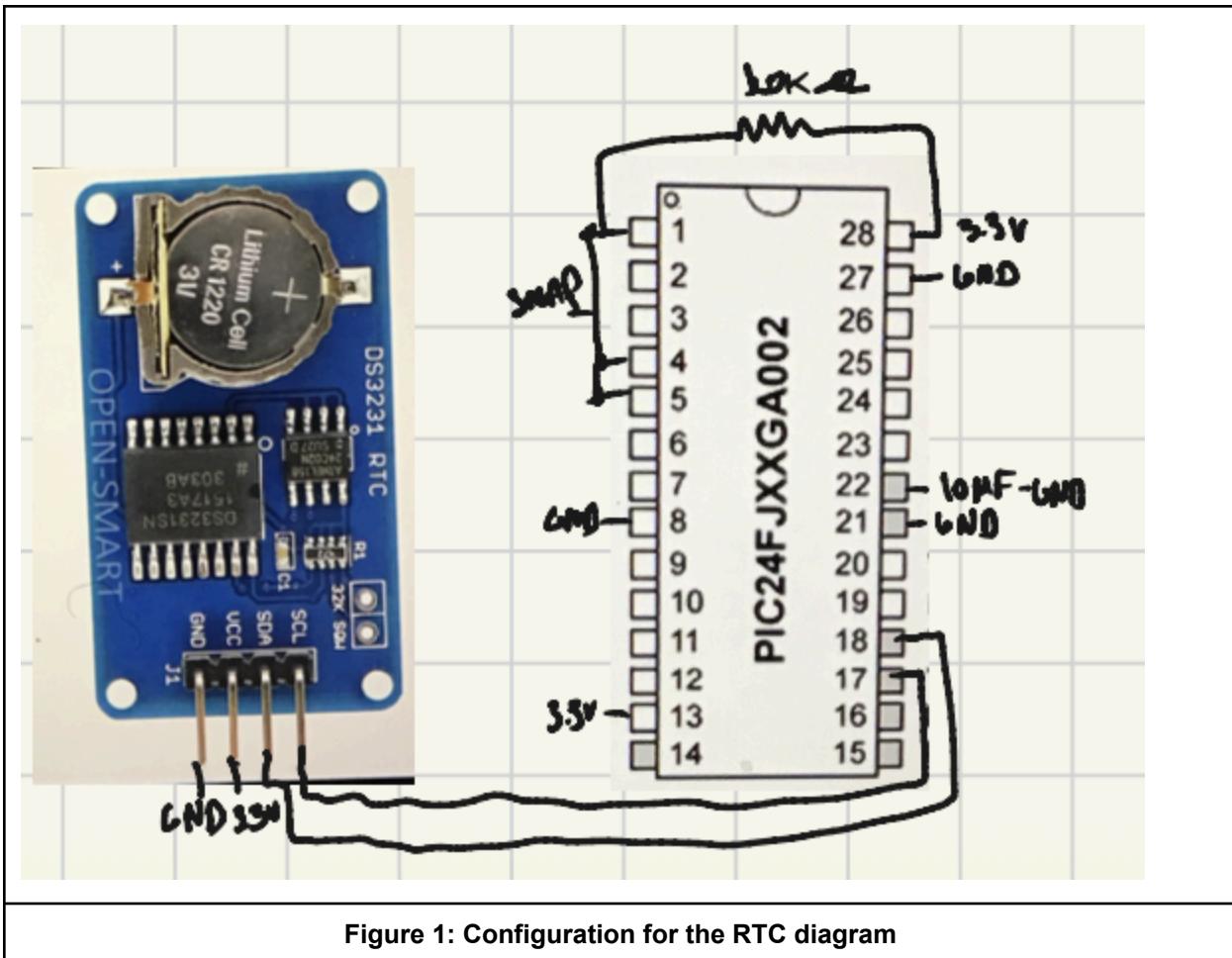


Figure 1: Configuration for the RTC diagram

Notes: Two 4.7k Ω resistors used as pullup resistors, and the actual DS3231 module. Both pullup resistors are connected to the SDA/SCL lines and VCC, and the SDA/SCL lines themselves are connected to pin 18 and 17 respectively. Finally, VCC and GND are connected to the PIC24's appropriate pins.

Code

Source file

```
/*
 * File: rtc.c
 * Author: Alamin Suliman
 * Project: RFID Security Access System
 * Description: Implementation of the DS3231 RTC clock driver, over I2C
 */

#include "xc.h"
```

```

#include "pic24.h"
#include "rtc.h"

#define FCY 16000000UL      // Define instruction cycle frequency
#include <libpic30.h>      // Include for __delay_ms() function

#define RTC_ADDR_I2C 0x68    // DS3231 7-bit I2C address

// DS3231 Register addresses
#define RTC_REG_SECONDS 0x00
#define RTC_REG_MINUTES 0x01
#define RTC_REG_HOURS 0x02
#define RTC_REG_DAY 0x03
#define RTC_REG_DATE 0x04
#define RTC_REG_MONTH 0x05
#define RTC_REG_YEAR 0x06
#define RTC_REG_CONTROL 0x0E
#define RTC_REG_STATUS 0x0F
#define RTC_REG_TEMP_MSB 0x11
#define RTC_REG_TEMP_LSB 0x12

// Function: rtc_write_register
// Description: Writes a single register to the DS3231 RTC over I2C
// Input: reg - register address to write
//         data - data byte to write to the register
// Return: none
static void rtc_write_register(uint8_t reg, uint8_t data)
{
    uint8_t buf[2] = {reg, data};
    pic24_write_i2c(RTC_ADDR_I2C, buf, 2);
}

// Function: rtc_read_register
// Description: Reads a single register from the DS3231 RTC over I2C
// Input: reg - register address to read
// Return: 8-bit value read from the specified register
static uint8_t rtc_read_register(uint8_t reg)
{
    uint8_t buf = reg;
    pic24_write_i2c(RTC_ADDR_I2C, &buf, 1);
    pic24_read_i2c(RTC_ADDR_I2C, &buf, 1);
    return buf;
}

// Function: dec_to_bcd
// Description: Converts an 8-bit decimal value to BCD format

```

```

// Input: val - decimal value (0-99)
// Return: BCD-encoded representation of the value
static uint8_t dec_to_bcd(uint8_t val)
{
    return ((val / 10) << 4) | (val % 10);
}

// Function: bcd_to_dec
// Description: Converts an 8-bit BCD value to decimal
// Input: val - BCD-encoded value
// Return: Decimal representation of the BCD value
static uint8_t bcd_to_dec(uint8_t val)
{
    return ((val >> 4) * 10) + (val & 0x0F);
}

// Function: rtc_is_oscillator_stopped
// Description: Checks the oscillator stop flag (OSF) in the status register
// Return: 1 if OSF is set (oscillator stopped), 0 if clear
static uint8_t rtc_is_oscillator_stopped(void)
{
    uint8_t status_reg = rtc_read_register(RTC_REG_STATUS);
    return (status_reg & 0x80) ? 1 : 0; // Bit 7 = OSF
}

// Function: rtc_clear_oscillator_flag
// Description: Clears the oscillator stop flag (OSF) in the status register
// Return: none
static void rtc_clear_oscillator_flag(void)
{
    uint8_t status_reg = rtc_read_register(RTC_REG_STATUS);
    rtc_write_register(RTC_REG_STATUS, status_reg & ~0x80);
}

// Function: rtc_init
// Description: Initializes the DS3231 RTC device
// Clears oscillator stop flag and configures control settings
// Return: none
void rtc_init(void)
{
    __delay_ms(100);           // Allow device to power up

    // Clear the oscillator stop flag and ensure 24-hour mode is usable
    uint8_t status_reg = rtc_read_register(RTC_REG_STATUS);
    rtc_write_register(RTC_REG_STATUS, status_reg & ~0x80);
}

```

```

// Enable oscillator; disable square-wave output (INTCN=1, BBSQW=0, RS=00)
rtc_write_register(RTC_REG_CONTROL, 0x04);

    __delay_ms(10);           // Small delay for settings to take effect
}

// Function: rtc_set_time
// Description: Sets the current time and date in the DS3231
// Input: time - pointer to rtc_time_t structure with fields populated
// Return: none
void rtc_set_time(rtc_time_t *time)
{
    rtc_write_register(RTC_REG_SECONDS, dec_to_bcd(time->seconds));
    rtc_write_register(RTC_REG_MINUTES, dec_to_bcd(time->minutes));
    rtc_write_register(RTC_REG_HOURS,
                       dec_to_bcd(time->hours)); // 24-hour format
    rtc_write_register(RTC_REG_DAY, time->day); // 1-7 (Day of week)
    rtc_write_register(RTC_REG_DATE, dec_to_bcd(time->date));
    rtc_write_register(RTC_REG_MONTH, dec_to_bcd(time->month));
    rtc_write_register(RTC_REG_YEAR, dec_to_bcd(time->year));
}

// Function: rtc_get_time
// Description: Reads the current time and date from the DS3231
// Input: time - pointer to rtc_time_t structure to populate
// Notes: BCD values are converted to decimal; masks clear control bits
// Return: none
void rtc_get_time(rtc_time_t *time)
{
    time->seconds =
        bcd_to_dec(rtc_read_register(RTC_REG_SECONDS) & 0x7F); // bit7=CH
    time->minutes =
        bcd_to_dec(rtc_read_register(RTC_REG_MINUTES) & 0x7F);
    time->hours =
        bcd_to_dec(rtc_read_register(RTC_REG_HOURS) & 0x3F); // 24-hour mask
    time->day = rtc_read_register(RTC_REG_DAY) & 0x07; // 1-7
    time->date = bcd_to_dec(rtc_read_register(RTC_REG_DATE) & 0x3F);
    time->month = bcd_to_dec(rtc_read_register(RTC_REG_MONTH) & 0x1F);
    time->year = bcd_to_dec(rtc_read_register(RTC_REG_YEAR));
}

// Function: rtc_get_temperature
// Description: Reads temperature from DS3231 sensor in 0.25°C increments
// Return: Temperature in degrees Celsius as a float
float rtc_get_temperature(void)
{

```

```

    uint8_t temp_msb = rtc_read_register(RTC_REG_TEMP_MSB); // Signed MSB
    uint8_t temp_lsb = rtc_read_register(RTC_REG_TEMP_LSB); // Fraction bits

    int16_t temp = (int16_t)((temp_msb << 8) | temp_lsb); // Combine
    temp >>= 6; // Align to 0.25°C

    return temp * 0.25f; // Convert to °C
}

```

Header file

```

/*
 * File: rtc.h
 * Author: Alamin Suliman
 * Project: RFID Security Access System
 * Description: Header file for the DS3231 RTC clock module
 */

#ifndef RTC_H
#define RTC_H

#include <stdint.h>

typedef struct {
    uint8_t seconds; // 0-59
    uint8_t minutes; // 0-59
    uint8_t hours; // 0-23 (24-hour format)
    uint8_t day; // 1-7 (day of week)
    uint8_t date; // 1-31
    uint8_t month; // 1-12
    uint8_t year; // 0-99 (years since 2000)
} rtc_time_t;

void rtc_init(void);
void rtc_set_time(rtc_time_t *time);
void rtc_get_time(rtc_time_t *time);
float rtc_get_temperature(void);

#endif /* RTC_H */

```