12.1 DS3231 Based Real Time Clock

12.1.1 Introduction

(1) The physical pin diagram of DS3231 RTC chip is depicted in Fig-12.1. The chip can be found in the DS3231 RTC Module of Fig-12.2, and it communicates with MCU using I2C Bus at 7-bit address of 0b1101000. The RTC Module contains 2x4.7k pull-up resistors for I2C Bus. The clock has a V_{BAT} pin to connect a 3.3V Battery Cell to retain the time of the day when main supply fails. The oscillator is always on when the chip is powered by the Vcc-pin.

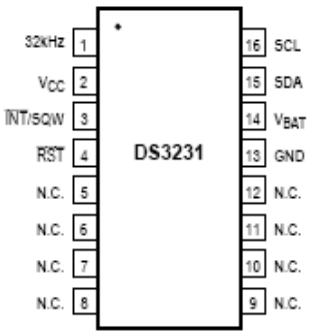

Figure-12.1: *DS3231 RTC chip*

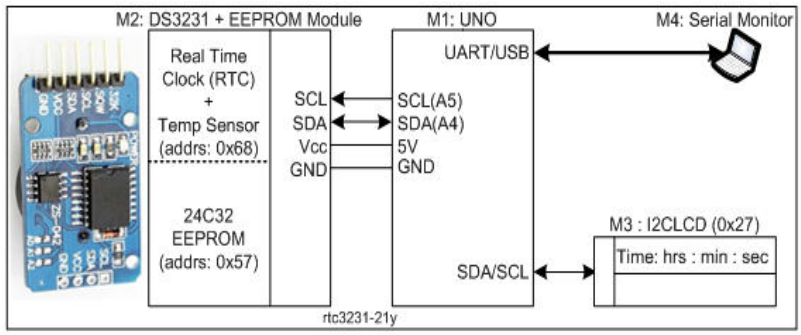

Figure-12.2: *RTC Module and it connection with UNO and I2CLCD*

(2) The time-drift (slow/fast) of the DS3231 based Clock is +/- 2 minutes in 2-year (1 sec in 5 days). The RTC chip maintains such an accurate timing by virtue of a built-in 'temperature compensated crystal oscillator subsystem (Fig-12.3)' which provides 1 Hz timing signal for Clock, Calendar, and Alarms. The time of the clock is advanced by 1-sec period.

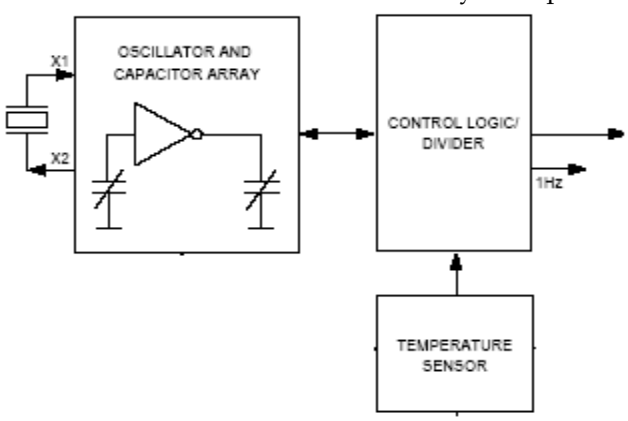

Figure-12.3: *Temperature compensated internal crystal oscillator subsystem of DS3231 RTC chip*

(3) There are Seconds Register, Minutes Register, and Hours Register inside the RTC chip to hold the current time of the day in BCD formats. In Fig-12.4, we have depicted the 'bit layout' of the 'Seconds Register' and 'Minutes Register'.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59

Figure-12.4: Bit layout of 'Seconds Register' and 'Minutes Register' of DS3231 RTC chip

(4) The Seconds Register holds the seconds of the clock (range: 00 - 59) in BCD (Binary Coded Decimal) format; where, a decimal digit is coded into 4-bit binary value. Thus, 00 seconds are coded as 0000000; 01 seconds are coded as 00000001; ...; 19 seconds are coded as 00011001; 20 seconds are coded as 00100000; ...; 59 seconds are coded as 01011001. Here, we observe that only 7-bit are good enough to hold the range (00 - 59) of the seconds of the clock; accordingly, the size of the Seconds Register is 7-bit (Fig-12.3); the lower 4-bit accommodate the value 0-9 (0000 - 1001) with positional weight of 1 (one) and the upper 3-bit accommodate the value 0-5 (000 - 101) with positional weight of 10 (ten). For example: assume that the current content of the Seconds Register is 0101001. What is the value of seconds? It is: $(010)_2$*10 + $(1001)_2$*1 = 2*10 + 9*1 = 29.

(5) Explanations of the Seconds Register can be extended for the description of the Minutes Register.

(6) Bit layout of Hours Register of DS3231 RTC Chip

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23

Figure-12.5: *Bit layout of Hours Register*

(7) In 12-hour Clock Time format, there is AM and PM. To maintain 12-Hr Time (12:00:00 AM ---> 12:30:00 PM ---> 01:0:00 PM ---> 12:30:00 PM ---> 01:00:00 AM ---> ...), we need to put HIGH at BIT-6 of Hours Register of Fig-12.6 during initialization. The RTC automatically maintains LOW or HIGH at BIT-5 of Hours Register to indicate AM (PM). As a result, the bit layout of Hours Register becomes as:

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
02h	0	12/$\overline{24}$	$\overline{AM}$/PM	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM

Figure-12.6: *Bit layout of Hours Register in 12-hour clock time format*

(8) In 24-hour Clock Time format (Fig-12.7), there is no AM and PM. The conventional 11 PM (11 o'clock by the night) is indicated by the figure 23:00:00. The 12 PM is indicated by 00:00:00. To operate DS3231 in 24-hour clock time mode, we put LOW at BIT-6 (Fig-12.7) of the Hours Register during initialization. As a result, the bit layout of Hours Register becomes as:

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
02h	0	12/$\overline{24}$	20 Hour	10 Hour	Hour				Hours	00–23

Figure-12.7: *Bit layout of Hours Register in 24-hour clock time format*

The Hours Register holds the hours of the clock (range: 00 - 23) in BCD format. Thus, 00 hours are coded as 00000000; ...; 17 hours are coded as 00010111;...; 23 hours are coded as 00100011. Here, we observe that only 6-bit is good enough to hold the range (00 - 23) of the hours of the clock; accordingly, the size of the hours Register is 6-bit (Fig-12.7); the lower 4-bit accommodate the value 0-9 (0000 - 1001) with positional weight of 1 (one); Bit 4 accommodates the value 0-1 (0 - 1) with positional weight of 10 (ten); BIT-5 accommodates the value 0-1 (0 - 1) with positional weight of 20 (twenty) . For example: assume that the current content of the hours Register is 100001. What is the value of hours? It is: 1*20 + 0*10 + (0001)*1 = 20 + 0 + 1 = 21. The bits will remain confined to a pattern such that the sum of the positional values of the 'Hour Field', '10 Hour Field' and '20 Hour Field' will never exceed 23.

12.1.2 Basic Operation of RTC using Register Level Codes

(1) Connect the DS3231 RTC Module with UNO as per diagram of Fig-12.2.

(2) **[Check the presence of RTC]** Create sketch to check the presence of the RTC chip on the I2C Bus at address 0b1101000. Save the sketch as P12122.ino. Upload the sketch and observe the message "RTC is found" has appeared on Serial Monitor.

```
#include<Wire.h>

void setup()
{
    Serial.begin(9600);
    Wire.begin();           //I2c Bus is formed

    Wire.beginTransmission(0b1101000);
    byte busStatus = Wire.endTransmission();
    if(busStatus !=0)
    {
       Serial.print("RTC is not found...!");
       While(1);    //wait for ever
    }
    Serial.println("RTC is found.");
}

void loop()
{

}
```

(3) **[Set current time in 12-hr clock format]** Consult Fig-12.6 and create sketch to set the current time into the RTC in 12-hr clock format and then show the running time on LCD and Serial Monitor (Fig-12.2) at 1-sec interval. Save sketch as P12123.ino.

12.1.3 Basic Operation of RTC using Library Functions

(1) The **RTClib.h** Library supports the following RTC Chips:
PCF8523, DS1307, and DS3231.

(2) The RTClib.h allows creating the following object from class RTC_DS3231:

```
RTC_DS3231 rtc;
```

(3) The following functions/methods are provided by the library to perform various operations on DS3231 chip.

(a) **begin():** To check if the DS3231 chip is present in the I2C Bus at address 0b1101000 (0x68). Example:

```
bool flag1 = rtc.begin();    //flag1 = true = HIGH indicates that the chip is present
```

(b) **adjust():** To set the intial date and time. Example:

(i)
```
rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));  //takes from PC; 12-hr or 24-hr?
```

(ii)
```
rtc.adjust(2019, 8, 31, 12, 35, 17); //manual:yrs,mon,day,hrs,min,sec;12-hr or 24-hr?
```

(c) **now():** It transfers the current date (year, month, day) and time (hrs, min, sec) in user declared data structure (the nowData). Example:

```
DateTime nowData = rtc.now();
```

(d) **day():** It returns the 2-digit BCD day of the month (00 – 31). Example:

```
int myDay = nowData.day();
```

(e) **month():** It returns the 2-digit BCD month of the year (01 – 12). Example:

```
int myMonth = nowData.month();
```

(f) **year():** It returns the 4-digit BCD year. Example:

```
int myYear = nowData.year();
```

(g) **second():** It returns the 2-digit BCD second of the time (00 – 59). Example:

```
int mySecond = nowData.second();
```

(h) **minute():** It returns the 2-digit BCD minute of the time (00 – 59). Example:

```
int myMinute = nowData.minute();
```

(i) hour(): It returns the 2-digit BCD hour of the time. Example:

```
int myHour = nowData.hour();
```

(4) [Check the presence of RTC] Create sketch to check the presence of the RTC chip on the I2C Bus at address 0b1101000. Save the sketch as P12132.ino. Upload the sketch and observe the message "RTC is found" has appeared on Serial Monitor.

```
#include<Wire.h>
#include<RTClib.h>
RTC_DS3231 rtc;     //the object rtc is created from the class RTC_DS3231

Void setup()
{
    Serial.begin(9600);
    Wire.begin();           //I2c Bus is formed

    Bool flag1 = rtc.begin();       //if the DS3231 chip is present, then flag1 = true = HIGH
    if(n flag1 != true)
    {
       Serial.print("RTC is not found...!");
       While(1);    //wait for ever
    }
    Serial.println("RTC is found.");
}

void loop()
{

}
```

12.1.4 Exercises