Real Time Clock Module

DESCRIPTION:

The DS3231 is a extremely accurate I2C real-time clock (RTC) with an integrated temperature compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package. The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-ofday alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I2C bidirectional bus. A precision temperature-compensated voltage reference and comparator circuit monitors the status of VCC to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the RST pin is monitored as a pushbutton input for generating a reset externally.

Specification：

● Highly Accurate RTC Completely Manages All Timekeeping Functions

• Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100

• Accuracy ±2ppm from 0°C to +40°C

• Accuracy ±3.5ppm from -40°C to +85°C

• Digital Temp Sensor Output: ±3°C Accuracy

• Register for Aging Trim

• RST Output/Pushbutton Reset Debounce Input

• Two Time-of-Day Alarms

• Programmable Square-Wave Output Signal

● Simple Serial Interface Connects to Most Microcontrollers

• Fast (400kHz) I2C Interface

● Battery-Backup Input for Continuous Timekeeping

• Low Power Operation Extends Battery-Backup Run Time

• 3.3V Operation

● Operating Temperature Ranges: Commercial (0° C to +70° C) and Industrial (-40° C to +85° C)

PIN CONFIGURATION:

1、"GND": Ground

2、"VCC": +5VDC

3、"SDA": Serial Data Input/Output. This pin is the data input/output for the I2C serial interface. This open-drain pin requires an external pullup resistor.

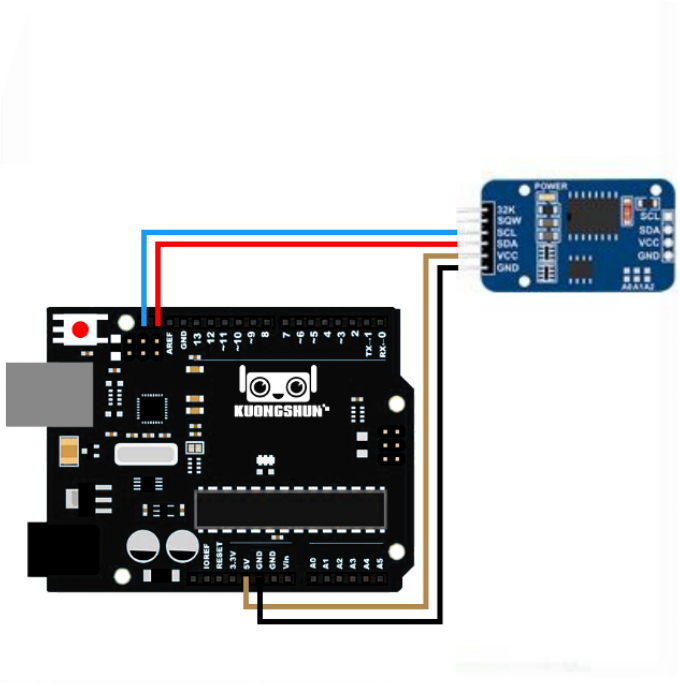
4、"SCL": Serial Clock Input. This pin is the clock input for the I2C serial interface and is used to synchronize data movement on the serial interface.

5、"SQW":(Active-Low Interrupt or Square-Wave Output. This open-drain pin requires an external pull-up resistor connected to a supply at 5.5V or less. It may be left open if not used. This multifunction pin is determined by the state of the INTCN bit in the

Control Register (0Eh). When INTCN is set to logic 0, this pin outputs a square wave and its frequency is determined by RS2 and RS1 bits. When INTCN is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled.)

6、"32K": 32kHz Output. This open-drain pin requires an external pullup resistor. When enabled, the output operates on either power supply. It may be left open if not used.

Example:



Code:

```
#include <Wire.h>
#include <DS3231.h>

DS3231 clock;
RTCDateTime dt;

void setup()
```

```
{
   Serial.begin(9600);

   // Initialize DS3231
   Serial.println("Initialize DS3231");;
   clock.begin();

   // Set sketch compiling time
   clock.setDateTime(__DATE__, __TIME__);
}

void loop()
{
   dt = clock.getDateTime();

   // For leading zero look to DS3231_dateformat example

   Serial.print("Raw data: ");
   Serial.print(dt.year);      Serial.print("-");
   Serial.print(dt.month);    Serial.print("-");
   Serial.print(dt.day);        Serial.print(" ");
   Serial.print(dt.hour);      Serial.print(":");
   Serial.print(dt.minute); Serial.print(":");
   Serial.print(dt.second); Serial.println("");

   delay(1000);
}
```