Microduino mCookie-RTC

USER GUIDE

https://eckstein-shop.de/

Content

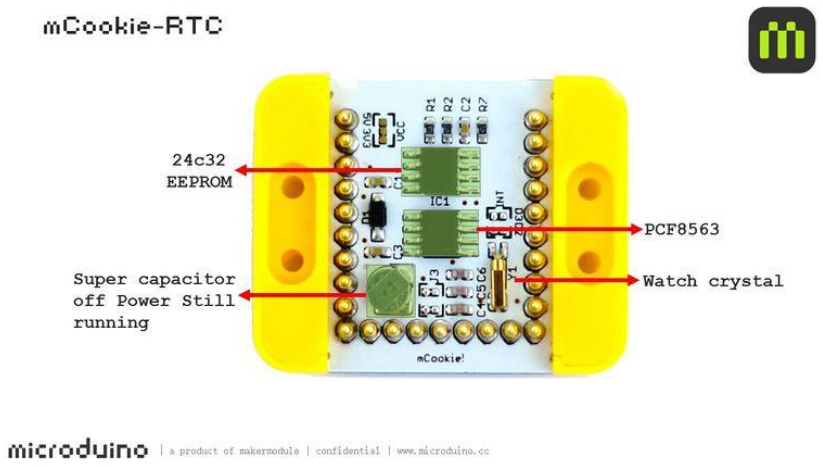
mCookie-RTC is a clock module adopting IIC interface communication, which can acquire time. With an onboard capacitor, RTC module can keep timing for several times after power disconnected.

Features

- Marked with century flag, second, minute, hour, day, week, month and year;
- Adopt I2C communication with the core modules;
- Own EEPROM memory chip with I2C interface.
- Low-power clock chip with typical 0.25μA current value and a super capacitor, capable of keeping the module running even after power disconnected;

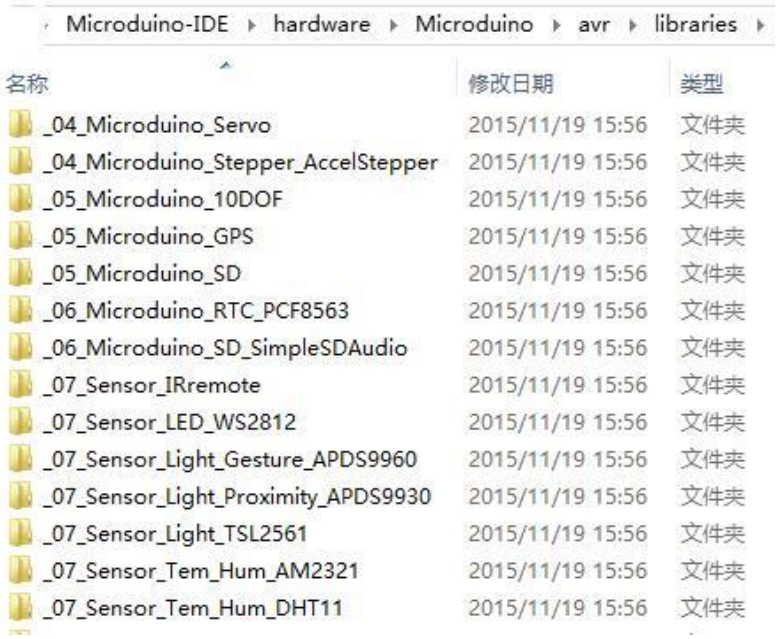
Specification

- Clock chip: PCF8563
  - Low-power CMOS real-time clock/calendar chip;
  - Offer a programmable clock output, an interruption output and a power-fail detector. All addresses and data can have a serial transmission through I2C bus interface;
  - The maximum bus speed is 400Kbits/s. The embedded word address register will generate increment after reading and writing data every time.
- EEPROM chip：AT24C32
  - Provide 32,768 EPROM serial power, which can be organized with a length of 4096 words × 8-bit;
  - Cascade feature allows AT24C32 to articulate eight devices on the same I2C bus, to have the replication cycle of millions times and save the data to 100 years with a write-protect function.
- Super capacitor: XH414
  - Provide a power-down timing function in a time slot.



Development

For the development, it need to be supported by _06_Microduino_RTC_PCF8563 library and check the library file here: (Installation Address)\Microduino-IDE\hardware\Microduino\avr\libraries



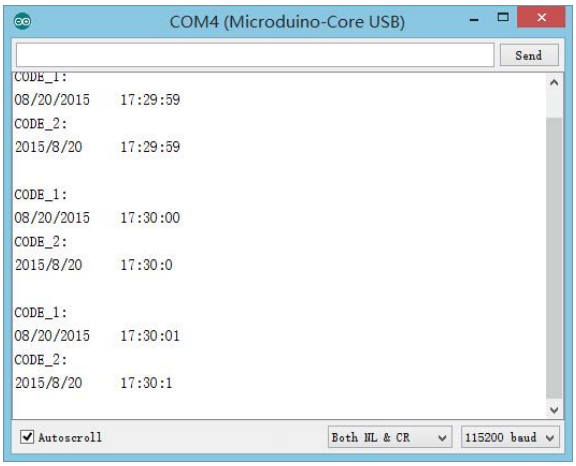
Application

Detect Power-down Time Duration

```
#include <Wire.h>
#include <Rtc_Pcf8563.h>

//init the real time clock
Rtc_Pcf8563 rtc;

void setup()
{
  Serial.begin(9600);
  settime(15, 8, 10, 4, 15, 57, 36);//Year, month, day, week, hour, minute and second
}

void loop()
```

```
{
  //both format functions call the internal getTime() so that the
  //formatted strings are at the current time/date.
  Serial.println("CODE_1:");
  Serial.print(rtc.formatDate());//Acquire data
  Serial.print("    ");
  Serial.println(rtc.formatTime());//Acquire time

  Serial.println("CODE_2:");
  Serial.print("20");
  Serial.print(rtc.getYear());//Acquire year
  Serial.print("/");
  Serial.print(rtc.getMonth());//Acquire month
  Serial.print("/");
  Serial.print(rtc.getDay());//Acquire day
  Serial.print("     ");
  Serial.print(rtc.getHour());//Acquire hour
  Serial.print(":");
  Serial.print(rtc.getMinute());//Acquire minute
  Serial.print(":");
  Serial.println(rtc.getSecond());//Acquire second

  delay(1000);
  Serial.print("\r\n");
}
void settime(int _year, int _month, int _day, int _week, int _hour, int _min
, int _sec)
{
  //clear out the registers
  rtc.initClock();
  //set a time to start with.
  //day, weekday, month, century(1=1900, 0=2000), year(0-99)
  rtc.setDate(_day, _week, _month, 0, _year);
  //hr, min, sec
  rtc.setTime(_hour, _min, _sec);
}
```

- Download program
  - Stack mCookie-BT and mCookie-CoreUSB, plug USB cable into mCookie-CoreUSB, then connect to PC with the other end;
  - Start Arduino IDE, copy the program above to IDE and set time in " settime(15, 8, 10, 4, 15, 57, 36);";
  - Select Microduino-CoreUSB in (Tools) -> (Board) and the corresponding serial number in (Tools) ->(Serial);
  - Click the icon (√) on top left of IDE and compile the program. After that, please click Download (->) and burn the program to the RTC board;
- Result
  - Open serial monitor after download and you can see the time.



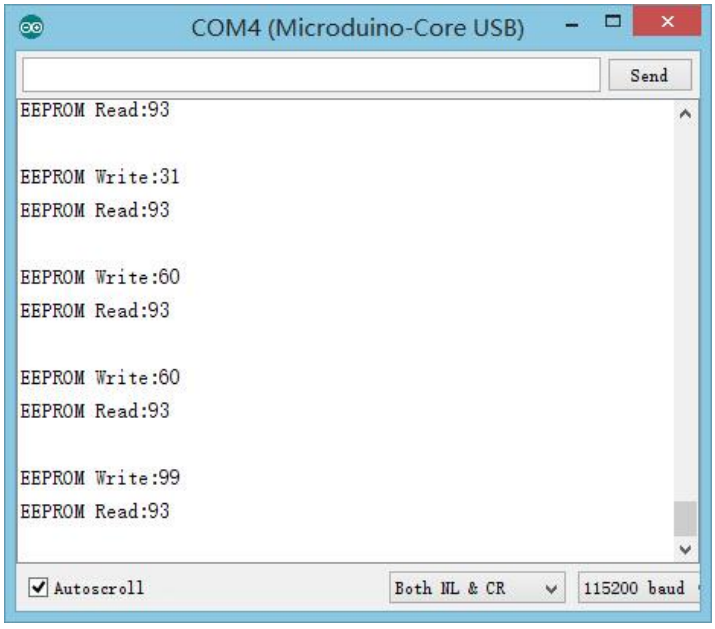
- Add "//" before “settime(15, 8, 10, 4, 15, 57, 36);" to comment out the function and download program again. Open serial monitor and you'll see the time.
- Cut off the power and connect to PC after several minutes, open serial monitor and you'll the running time after blackout rather than initialized time.
- (Note: RTC is programmable clock output, which can be set by settime() function. By setting time through settime() function, you need to comment out settime() so that you can take blackout timing next time. Otherwise, you have to reset the value after power-on and restart.)

Test EEPROM Read/Write

```
1.  #include <EEPROM.h>
2.
3.  long randNumber, data; //Define random number using the name of data
4.
5.  //EEPROM configuration
6.  #define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p);for(; i
    < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}
7.  #define EEPROM_read(address, p)  {int i = 0; byte *pp = (byte*)&(p);for(; i
    < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}
8.
9.  //Define EEPROMdata
10. struct config_type
11. {
12.   int EEPROMdata;
13. };
14.
15. void setup()
16. {
17.   Serial.begin(115200);
18.   /*EEPROM read evaluation*/
19.   config_type config_readback;
20.   EEPROM_read(0, config_readback);
21.   data = config_readback.EEPROMdata;
22. }
23. void loop()
24. {
25.   randNumber = random(10, 100);//Random function: The values of randNumber c
    hange from 10 to 99.
26.   delay(1000);//Rrefresh one time every second
```

```
27.   if (randNumber != data) //Judge if EEPROM data is changed, if yes, then wr
    ite in.
28.     eeprom_write();//EEPROM write function
29.   Serial.print("EEPROM Write:");
30.   Serial.println(randNumber);
31.   Serial.print("EEPROM Read:");
32.   Serial.println(data);
33.   delay(1000);
34.   Serial.println("");
35. }
36.
37. //======================EEPROM Write Function=========================//

38. void eeprom_write()
39. {
40.   config_type config;                  // Define config and its content
41.   config.EEPROMdata = randNumber;
42.   EEPROM_write(0, config);          //Save "config" to EEPROM and write addre
    ss 0
43. }
```

- Copy the program above to IDE, download program, open serial monitor and you'll see EEPROM writing and reading data.



- By unplugging power and then plugging in during testing, you can see that "EEPROM Read" is the value of "EEPROM Write" from serial monitor.

FAQ

- Some of the example codes that use the RTC module doesn't compile or gives an error!
  - There is an issue with the incorrect and outdated software library included in the Microduino software package. To fix it, please follow the guide below:
  - Download the correct library file from here: https://wiki.microduino.cc/images/9/90/Rtc_pcf8563.zip
  - In the Microduino IDE go to Sketch->Add .ZIP library...
  - Navigate to the downloaded file and select the "Rtc_pcf8563.zip". Then click Choose on the bottom right.
  - Now the IDE should use the correct library version and should be able to compile the program.