Project Multi-Functional Clock

Date: Dec. 5th, 2018

Author: 梁家伟

Contents

I. Abstract

- This design uses STC89C52 microcontroller as the core and is composed of DS1302 chip clock control module, 7-segment nixie tube display module, on time alarm module, infrared remote control module, DS18B20 temperature sensor module, AT24C02 chip(EEPROM) stored time and temperature module, UART sent EEPROM data module.

II. Task

1. Basic requirement:

- a) Realizing a digital clock, which can count hour, minute, second, and include the second dots.

2. Required functions:

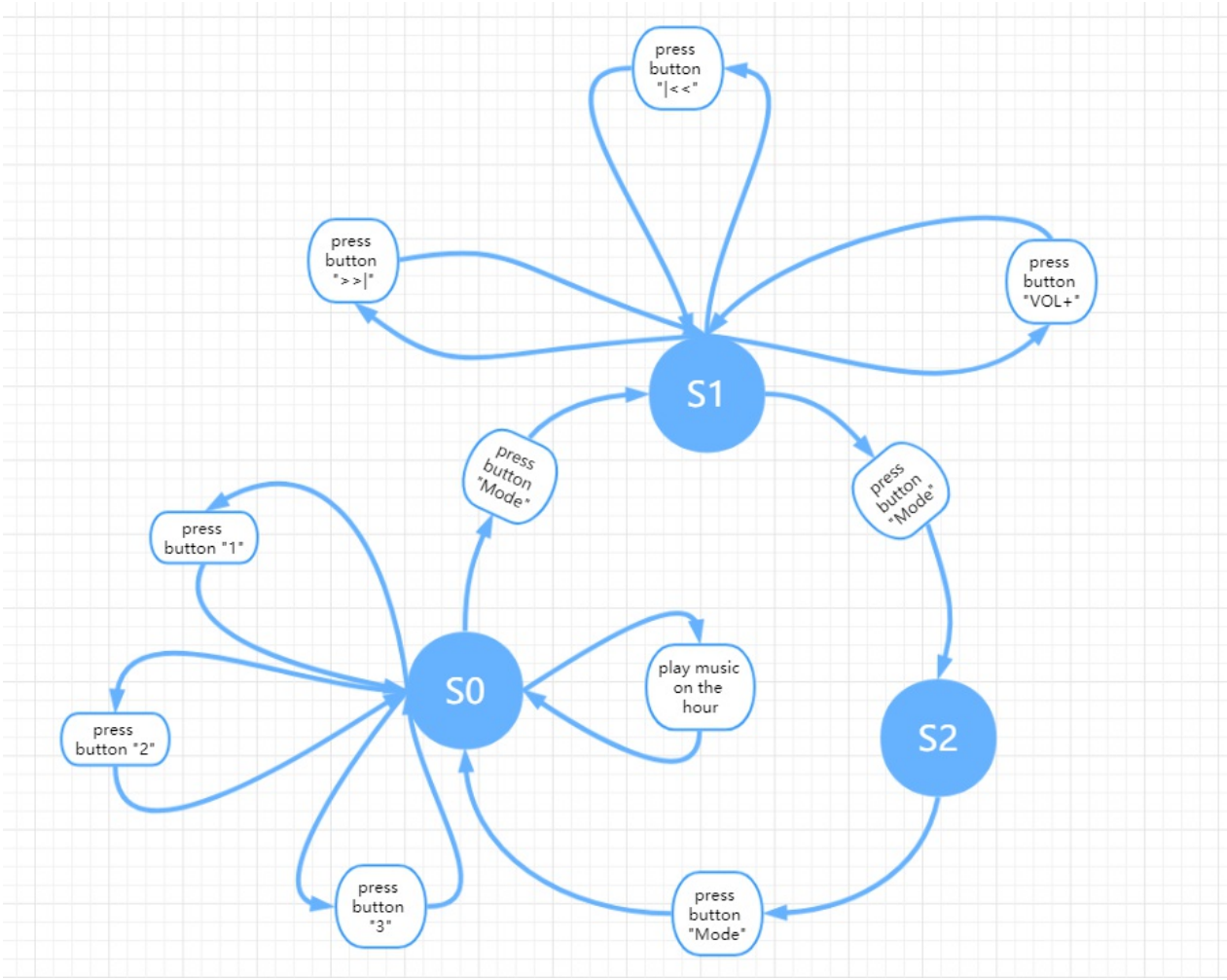
- a) Display can choose LCD or LED 7-segment. Indicate the current state (timer, setting, or temperate) in the display as well as the values. (15’)
- b) Realizing the time setting by using the keyboard. The timing and display should not be stop during setting (15’)
- c) Play some music on every hour (整点唱歌报时). The timing and display should not be stop during music (30’)
- d) Use the ADC & thermistor to measure the temperature every second, display it and set some alarm above a specific level (10’)
- e) Send the instant time and temperature data to the PC through the UART (10’)

3. Suggested functions for extra scores.

- a) Replace the keyboard in (2.b) with infrared remote control for realizing the time setting (+10’)
- b) Replace the ADC + thermistor with the DS18B20 1-wire temperature sensor for the measurement in (2.d)(+10’)
- c) Use the EEPROM to store the offline time and temperature history and later send through UART in (2.e) (+10’)
- d) Accelerating digit setting at key holding (键按下不放加速数字跳动) (+10)

III. Total design scheme

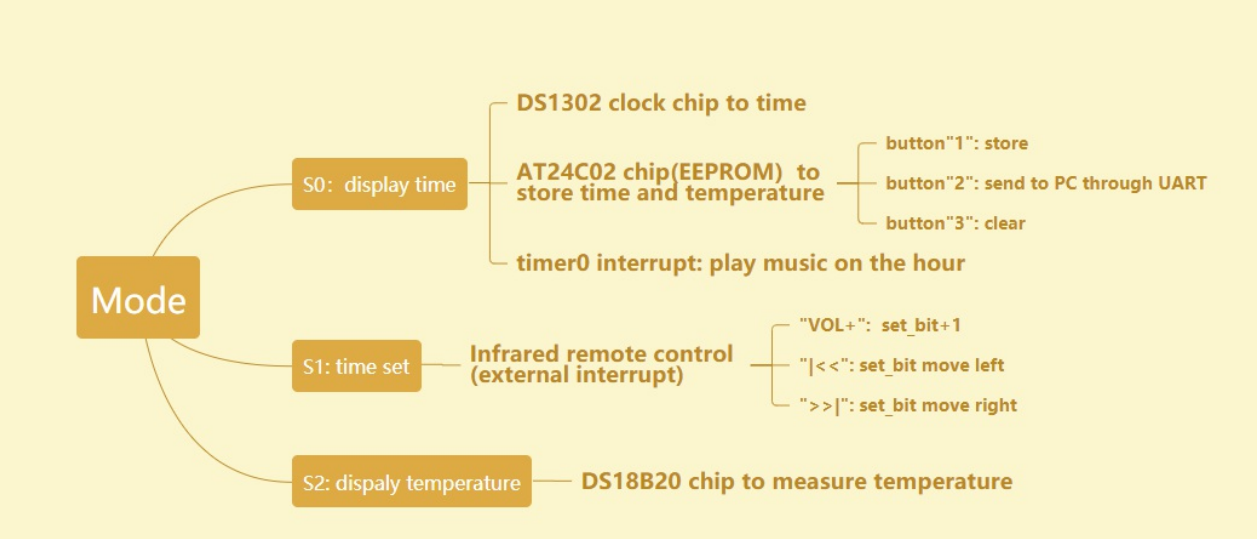
1. Finite State Machine



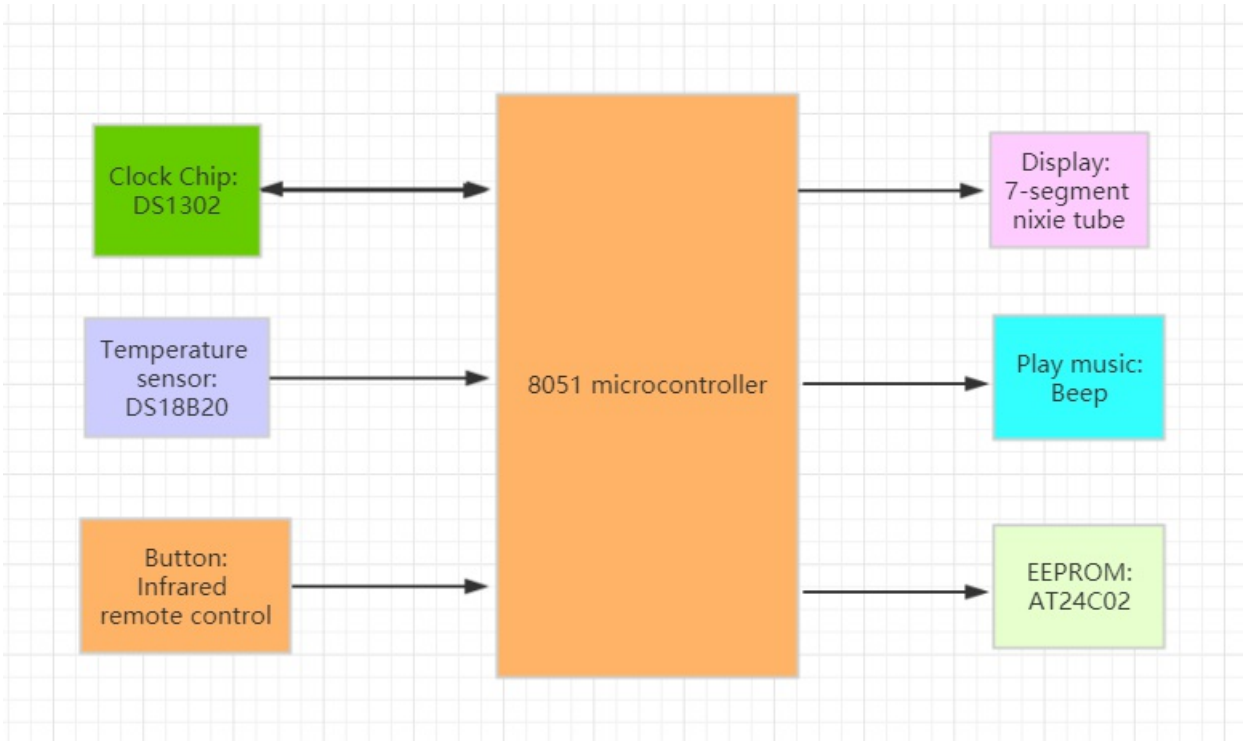
Current state	S0: display time	S1: set time	S2: display temperature
press button "Mode"	S1	S2	S0
press button "|<<"	null	Set_bit move left	null
press button ">>|"	null	Set_bit move right	null
press button "VOL+"	null	Set_bit+1	null
press button "1"	store current time and temperature	null	null
press button "2"	send current time and temperature to PC through UART	null	null
press button "3"	clear the data stored in EEPROM	null	null

On the hour	play music on the hour	null	null

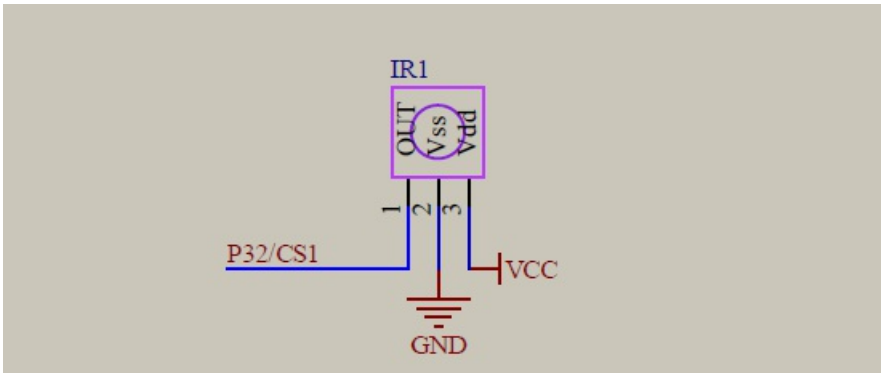
2. Flow chart



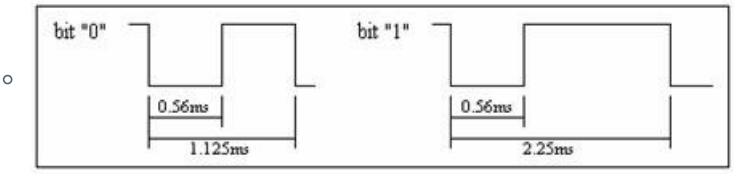
IV. Hardware design

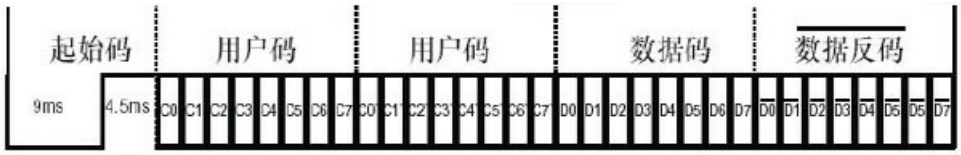


1. The principle of infrared communication



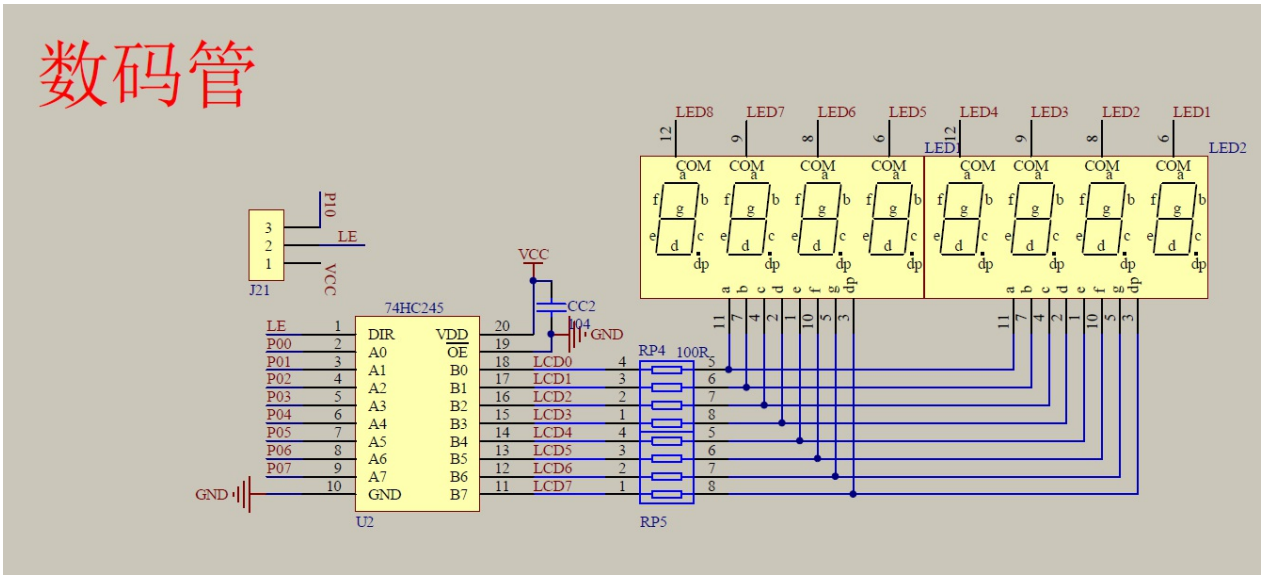
- The infrared remote control system generally consists of two parts: an infrared emitting device and an infrared receiving device.
  - The infrared emitting device can be composed of a keyboard circuit, an infrared encoding chip, a power source and an infrared transmitting circuit.
  - The infrared receiving device can be composed of an infrared receiving circuit, an infrared decoding chip, a power source and an application circuit.
- Usually, in order to make the signal better transmitted to the transmitting end, the baseband binary signal is modulated into a burst signal, which is transmitted through the infrared transmitting tube.
- There are two ways to realize:
  - Pulse width modulation (PWM) for signal modulation by pulse width
  - Pulse time modulation (PPM) for signal modulation by time interval between two pulse
- The carrier wave:
  - Using 455KHz crystal, the signal is modulated at 37.91KHz and the duty ratio is 1/3 through the internal frequency circuit
  - Modulation frequency (when the crystal oscillator is used at 455KHz)
    - $f_{CAR} = \frac{1}{T_C} = \frac{f_{OSC}}{12} \approx 38KHz$)
    - (f_{OSC}) is crystal frequency
    - duty = ($\frac{T_1}{T_c} = \frac{1}{3}$)
- Data Format：
  - The data format includes the start code, user code, data code and data inverse code, and the total code has 32 bits
  - The data inverse code is the complement code of data code and can be used for error correction of the data when encoding
  - Start code: 9ms low level, 4.5ms high level
- bit difinition：
  - Each bit in the user code or data code can be either bit '1' or bit '0'.The distinction between '0' and '1' is distinguished by the time interval of the pulse. This encoding method is called the pulse position modulation method, and the English shorthand PPM.
    - Low level time：0.56ms
    - Hige level time：0: 0.56ms<1ms ; 1:1.69ms>1ms



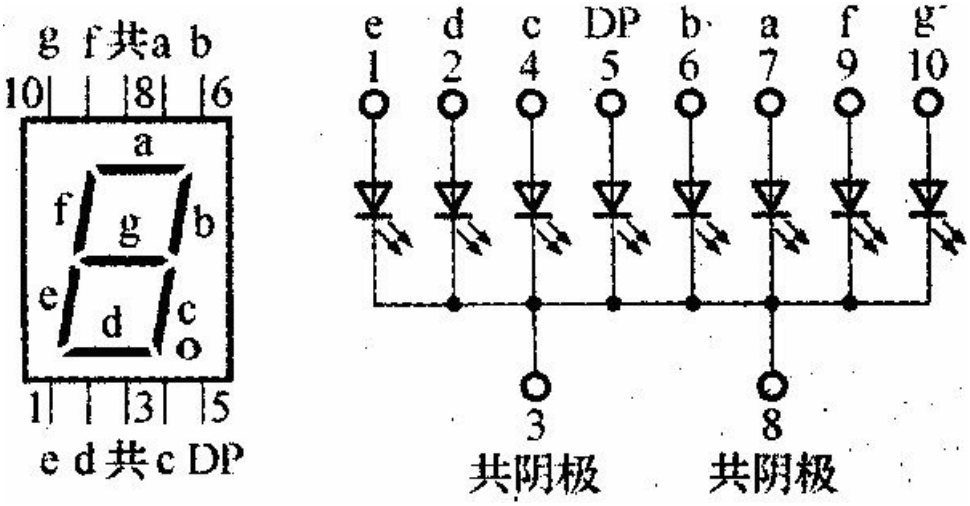


Button	Data code	button	Data code	Buttoon	Data code
OFF	0x45	Mode	0x46	Mute	0x47
Pause	0x44	|<<	0x40	>>|	0x43
EQ	0x07	VOL-	0x15	VOL+	0x09
0	0x16	RPT	0x19	U/SD	0x0D
1	0x0C	2	0x18	3	0x5E
4	0x08	5	0x1C	6	0x5A
7	0x42	8	0x52	9	0x4A

2. Nixie Tube

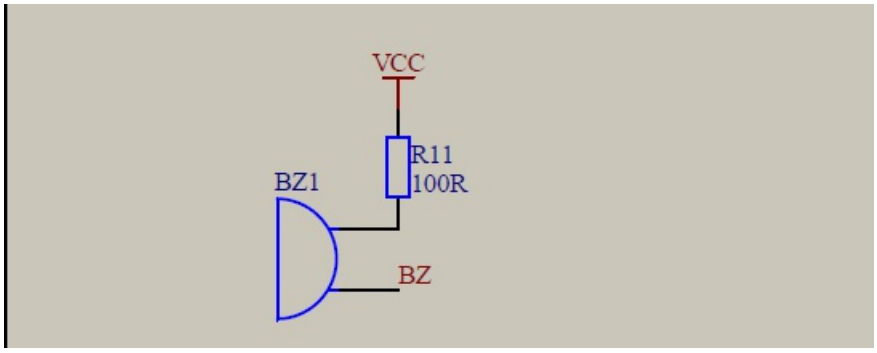


- Common cathode



Number	Pin	Decimal code	Hexadecimal code	Binary code
0	abcdef	63	3F	0011 1111
1	bc	6	6	0000 0110
2	abdeg	91	5B	0101 1011
3	abcdg	79	4F	0100 1111
4	bcfg	102	66	0110 0110
5	acdfg	109	6D	0110 1101
6	acdefg	125	7D	0111 1101
7	abc	7	7	0000 0111
8	abcdefg	127	7F	0111 1111
9	abcdfg	111	6F	0110 1111

3. Buzzer



- The function of the buzzer alarm module:
  - When the timer on the hour, the buzzer makes a preset sound.

- The BZ is connected to the pin P1.5 of the single-chip STC89C52.

- 定时器每10ms重置一次，整点调用
- Fn[0][1]函数每次播放一个note

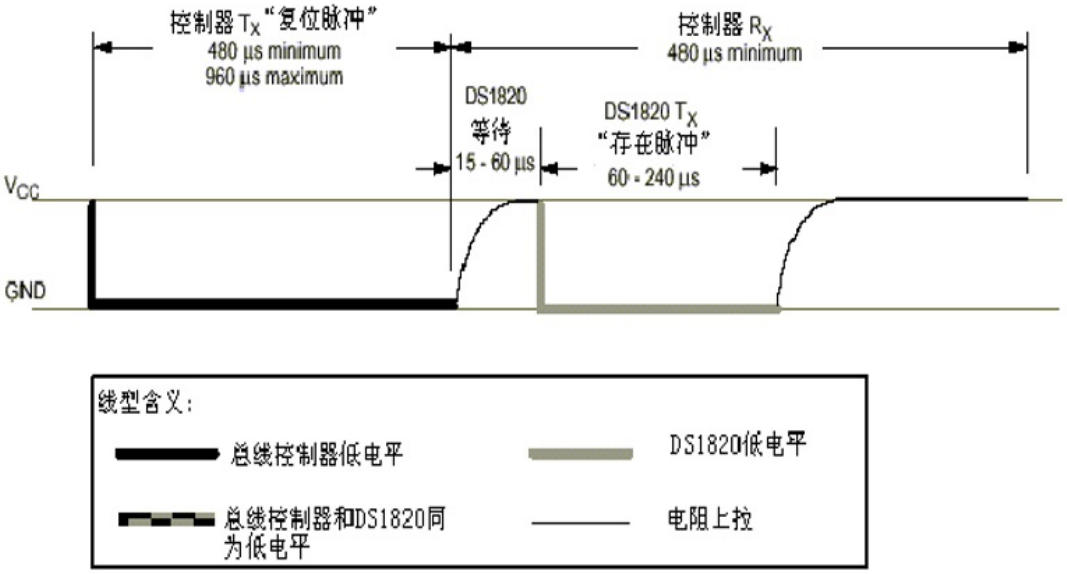
4. Temperature sensor DS18B20

A. Introduction of DS18B20

-
- DS18B20: single-line digital temperature sensors, or "one-line devices," has unique advantages:
  - The single-bus interface is used to connect the microprocessor with only one port line to achieve two-way communication between the microprocessor and the DS18B20.
  - Wide measurement temperature range and high measurement accuracy. The DS18B20 has a measurement range of -55 °C to +125 °C and an accuracy of ±0.5 °C in the range of -10 to +85 °C.

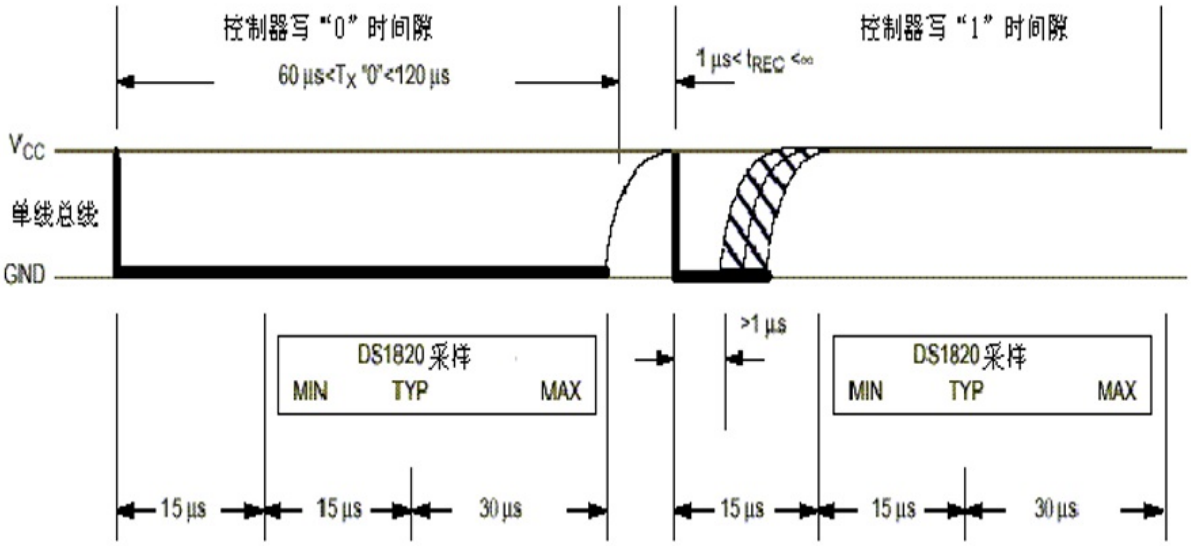
B. DS18B20 single bus working mode

Timing Sequence

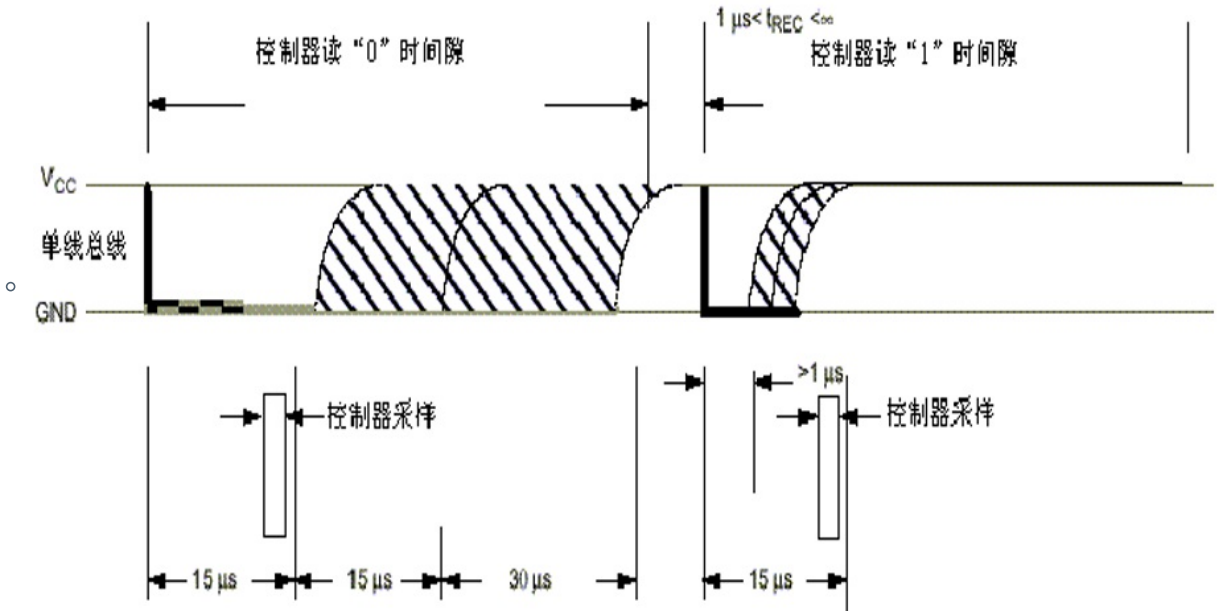
- Initializatoin
  - The host first sends a low-level pulse of 480-960 ms, then releases the bus to a high level, and detects the bus for the next 480 ms. If there is a low level, the device on the bus has been responded. If there is no low level, it is always high to indicate no device response on the bus.
  - As a slave device, the DS18B20 always detects if there is a low level of 480-960 us on the bus after power-on. If it is, wait for 15-60 us after the bus goes high. The level is pulled down 60-240 us to respond to the presence of a pulse, telling the host that the device is ready. If it is not detected, it is always waiting for detection.
  - 初始化过程“复位和存在脉冲”
  - 

- Write sequence
  - Data line set to 0 at first
  - delay 15us

- send data from LSB to MSB(one by ont)
- delay 60us
- Data line set to 1
- Repeat 8 times to send the whole byte
- Data line pulls up



- Read sequence
  - Data line set to 0 at first
  - delay 1us
  - Data line pulls up to 1 and ready to read data
  - delay 10us
  - Data line set to 1
  - get one bit from data line and save it
  - delay 45us
  - Repeat 8 times to save the whole byte
  - return the whole byte



Read temperature

- The simple steps to read the temperature value are as follows
  - Skip ROM operation
  - Send transform temperature command
  - Skip ROM operation
  - Send read temperature command
  - Read temperature value
- The first 5 bits of temperature value are sign bits, 0 represents positive; 1 represents negative
  - 0(positive): multiply by 0.0625 to get the actual temperature
  - 1(negative): minus 1, take the complement and then multiply by 0.0625 to get the actual temperature

5. DS1302 Clock Chip

A. Introduction of DS1302

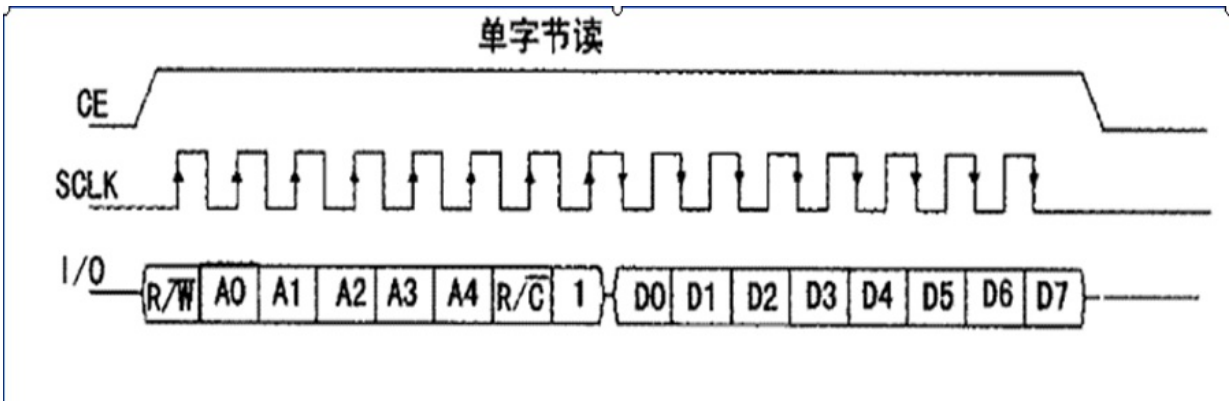
- The DS1302 is a high-performance, low-power real-time clock chip with 31 bytes of static RAM. It communicates with the CPU using the SPI three-wire interface.
- The real-time clock provides seconds, minutes, hours, days, weeks, months, and years, and can be automatically adjusted in one month and 31 days, with leap year compensation.
- The working voltage is as wide as 2.5 to 5.5V.

B. working mode

Single byte read

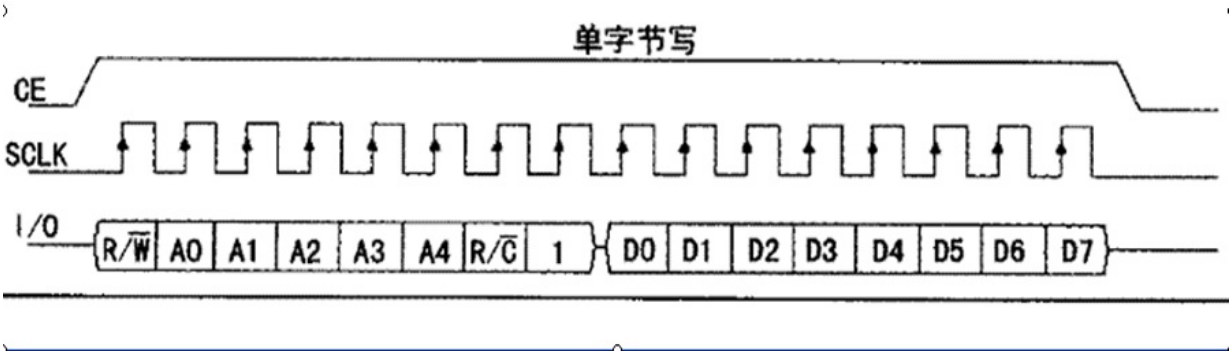
- The DS1302 communicates with the microcontroller through the SPI serial bus. When performing a read and write operation, it must read and write two bytes at a minimum. The first byte is the control byte, which is a command that tells the DS1302 whether to read or write. Is the RAM or the CLOK register operation. The second byte is the data to be read or written.

- Single Byte Read: CE can be asserted high only when SCLK is low. So set SCLK low before performing the operation, then set CE high, then start to put the level signal to be transmitted on IO, then jump SCLK. When the data is on the rising edge of SCLK, the DS1302 reads the data. On the falling edge of SCLK, the DS1302 places the data on the IO.



Single byte wirte

- RST can be asserted high only when SCLK is low. Say to set SCLK low before performing the operation, then set RST high, start to put the level signal to be transmitted on the IO, and then jump SCLK. When the data is on the rising edge of SCLK, the DS1302 reads the data. On the falling edge of SCLK, the DS1302 places the data on the IO.



6. EEPROM((I^2C)Bus) and UART

A. Introduction of AT24C02

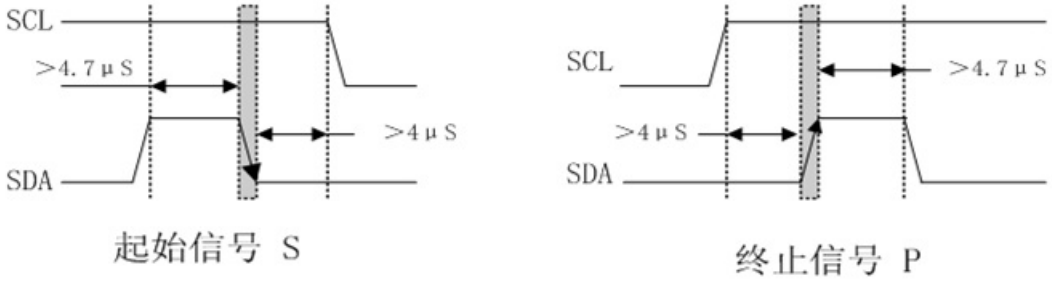
- The 24C02 is a 2K-bits serial CMOS EEPROM with 256 8-bit bytes internally.
- Compatible with 400KHz (I^2C) bus
- 1.8 to 6.0 V operating voltage range
- Data can be saved for 100 years

B. AT24C02 Working mode

Start and stop signal

- Start signal: SCL=1, SDA failling edge
  - SDA=1 keep more than 4.7us，then SDA=0 keep more than 4us
- Stop signal: SCL=1, SDA rising edge

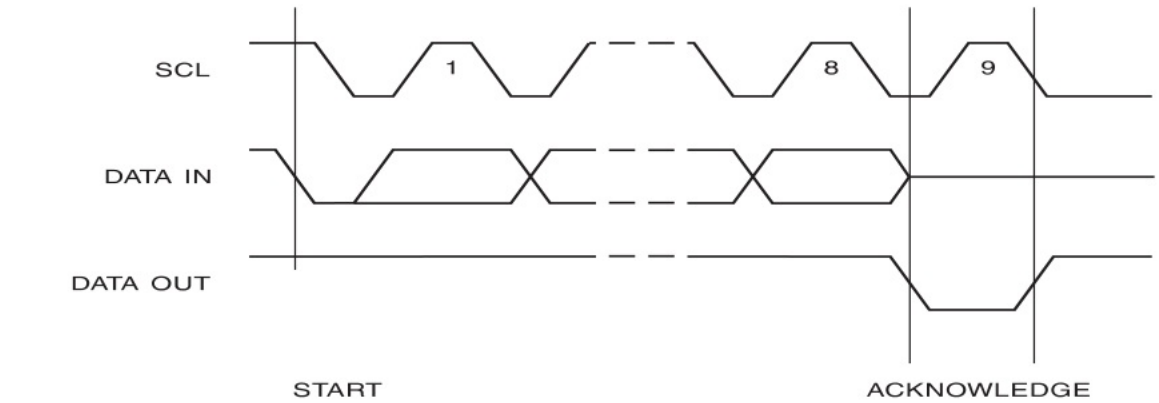
- SDA=0 keep more than 4us，then SDA=1 keep more than 4.7us



Answer signal

- Each byte transmitted must be guaranteed to be 8 bits long. When data is transmitted, the most significant bit (MSB) is transmitted first, and each transmitted byte must be followed by a response signal (that is, one frame has 9 bits in total).
- For example, after the host sends one byte of data to the bus, the bus is released, and the bus is pulled low (ie, the answer signal) to indicate that the byte is successfully transmitted.
- Similarly, after the host reads a byte of data from the bus, the host will pull the bus low, "telling" the slave to receive the data in this byte.

Figure 6. Output Acknowledge



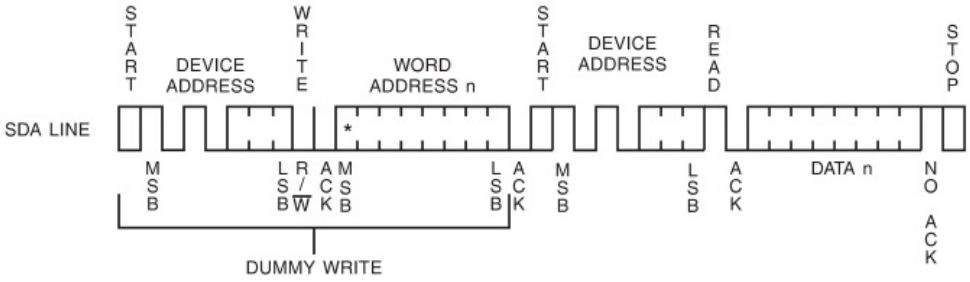
Read-Write Operation

- Write one byte of data to the specified address of the AT24C02 in the following order:
  - Start signal→write device address(0xA0, write)→answer signal→write memory address→answer signal→write

data→answer signal→stop signal

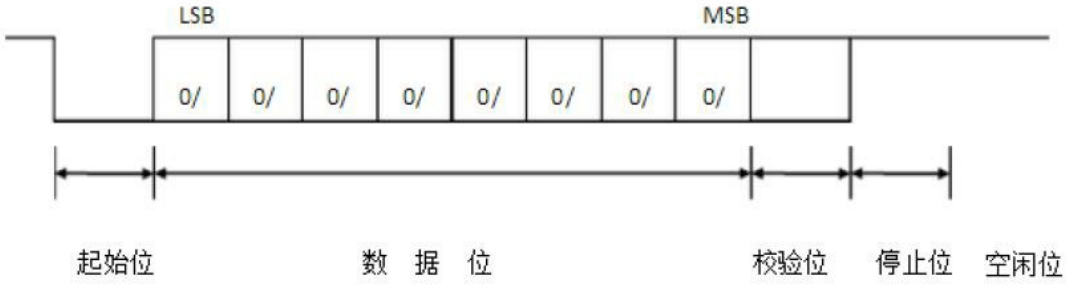
- Read one byte of data from the specified address of the AT24C02 in the following order:
  - Start signal→write device address(0xA0, write)→answer signal→write memory address→answer signal→start signal→write device address(0xA1, read)→answer signal→read data→stop signal→return the data

Figure 11. Random Read



C. UART

- The UART is a kind of asynchronous serial communication protocol. The working principle is to transmit each character of the transmitted data one by one. When two devices use serial communication, they must first agree on a data transmission rate, and the two devices The respective clock frequencies must be kept close to this rate, and a large difference in the clock frequency of one side will cause data transmission confusion.
- UART Pin：
  - TX：CPU sends data to the device, corresponding to the RX of the device.
  - RX：CPU receives the data from the device, and corresponds to the TX of the device.
  - GND：ground line
- UART definition：
  - Start bit a logic "0" signal indicating the beginning of the transmitted character
  - Data bit：The number can be 5-8, forming a character, starting from the lowest bit
  - Parity: The data bit plus this bit, so that the number of bits in "1" should be even (even parity) or odd (odd parity) to verify the correctness of data transmission.
  - Stop bit: the end of the character data
  - Baud Rate: A measure of the data transfer rate. Indicates the number of bits of binary code transmitted per second



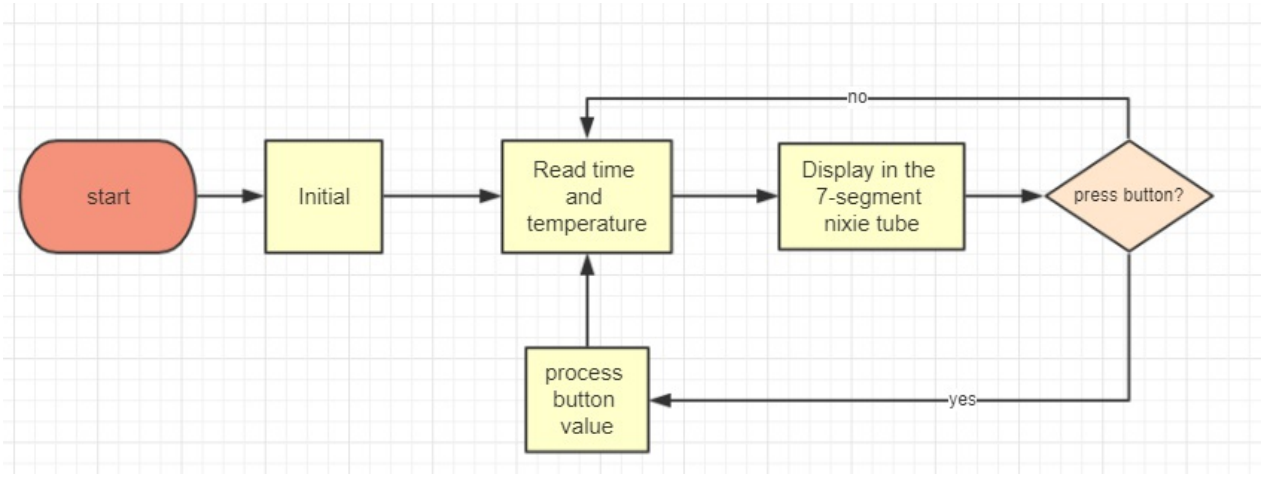
Notes:

- (I^2C) bus data validity provisions：
  - When SCL=1, SDA must remain stable. When SCL=0, SDA is allowed to change.

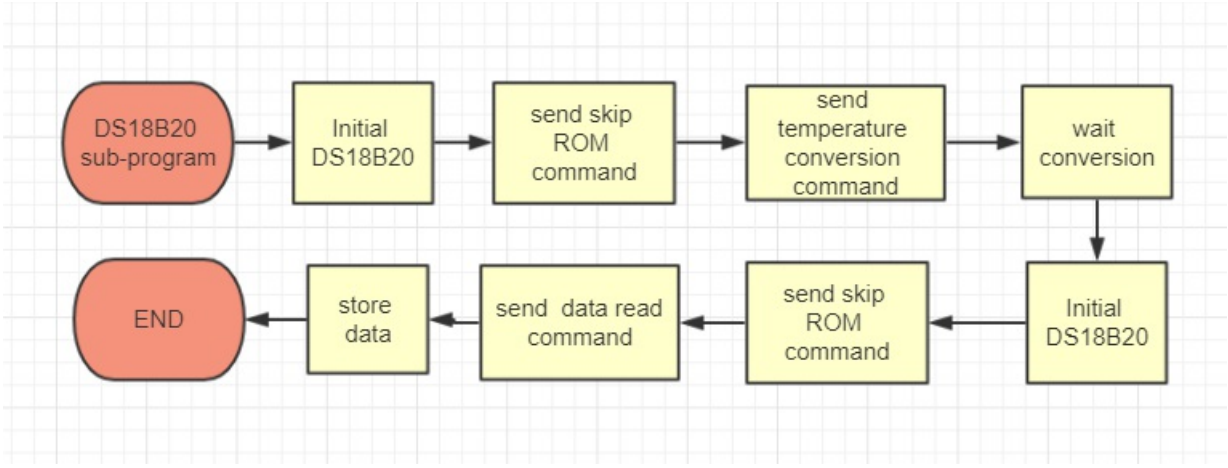
- Reference:
  - 基于51单片机IIC通信的AT24C02学习笔记: https://www.cnblogs.com/whik/p/6650092.html

V. Software Design

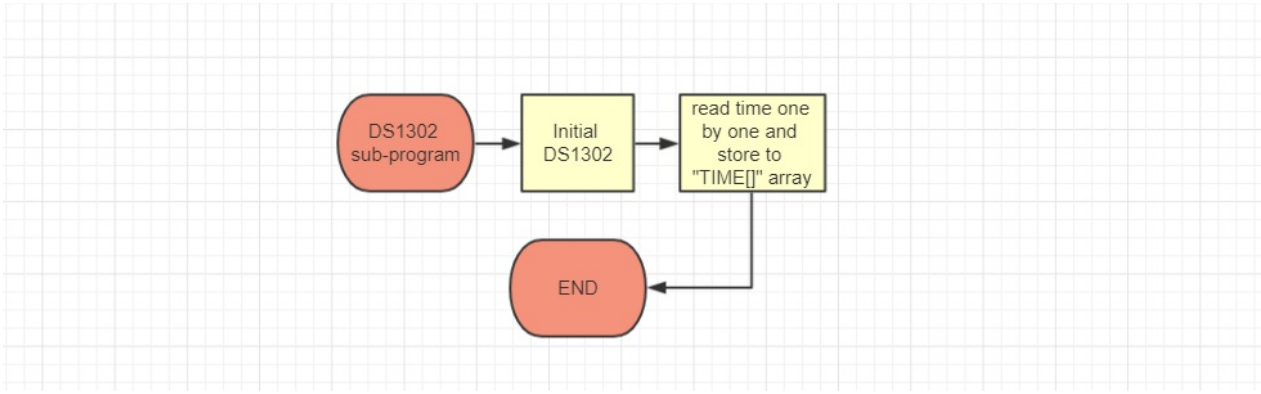
1. main program



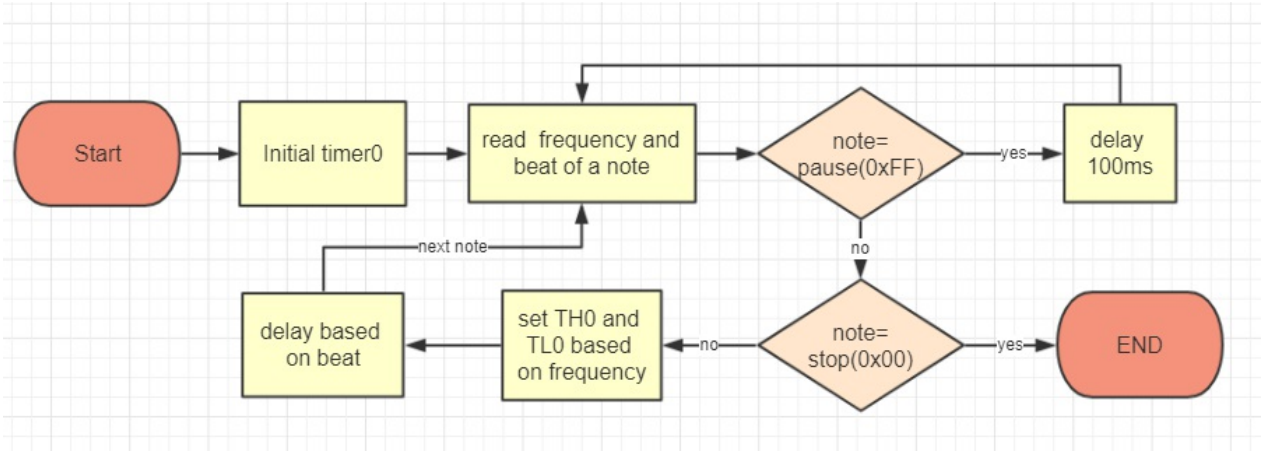
2. DS18B20 Program



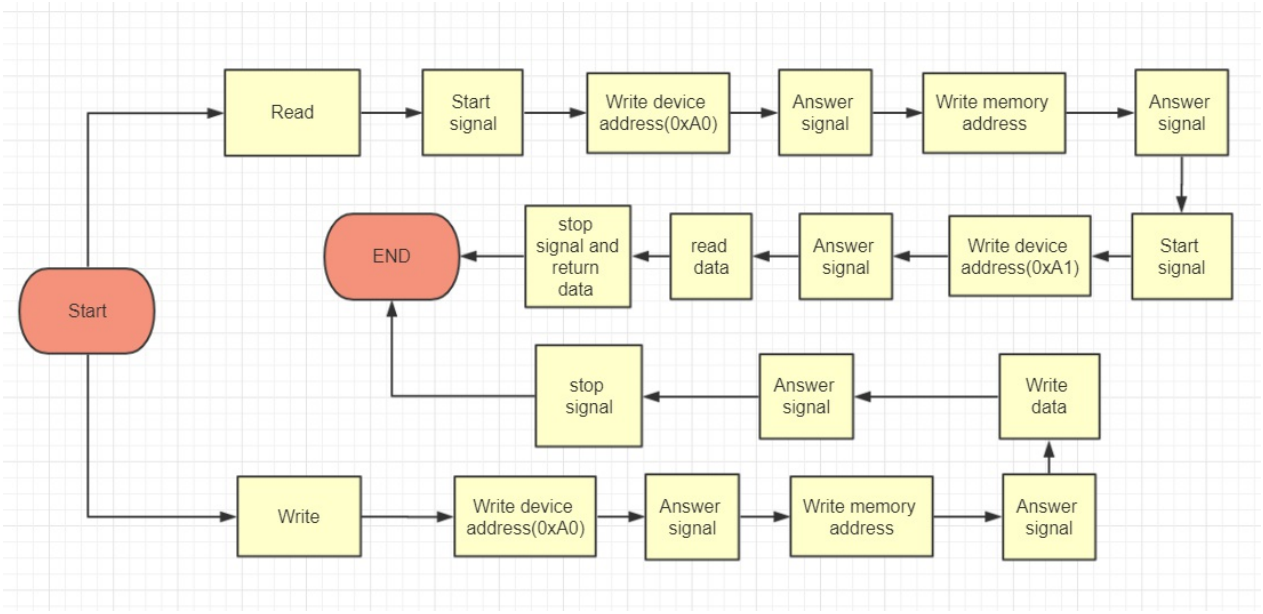
3. DS1302 clock program



4. music program(play on the hour)

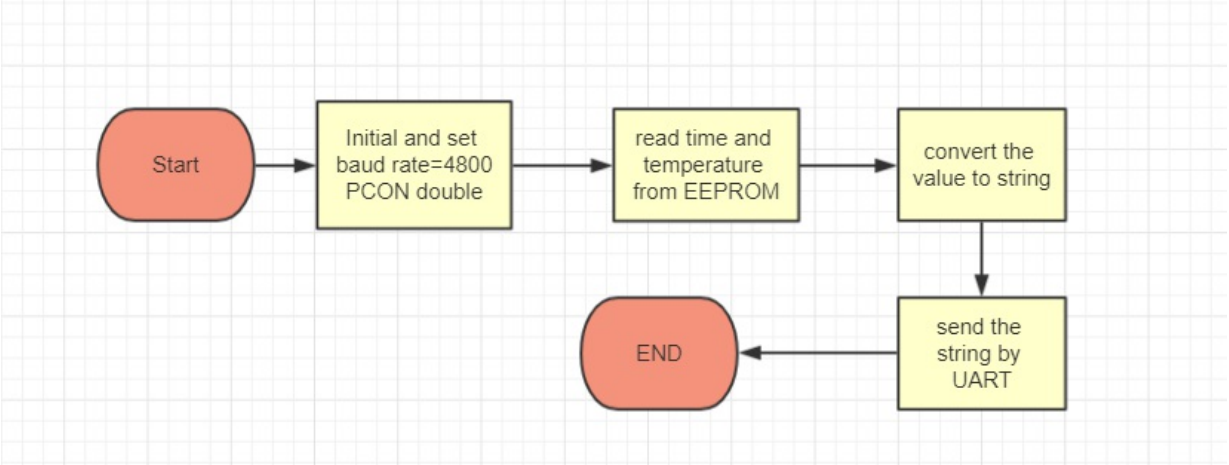


5. AT24C02(EEPROM) program



6. UART program

- baud rate should be set to 4800 and use PCON register to double since the error of the way is minimum using the PZ 8051 single-chip



VI. Code

1. Multifunction_clock.c(Main program)

```
#include <Multifunction_clock.h>

#define BCD(n) n/10*16+n%10

code unsigned char SegmentData[16]={
0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71
}; //common cathode

code void(code *Fn[3][2])()={{S0_timer,S0_music},{S1_set,S1_add},
{S2_tem,null}};// the state-input transition table using the function pointer

uchar State=0; // state variable
uchar Key=0; // input variable

uchar FlashLED=0; // use variables to communicate among modularized functions
uchar set_bit=0;
uchar debouncing=0;
uint time_count=0;
int temp=0;
float tp0;
int tp1=0;
uchar s=0;
uchar DispSegment[8];
uchar Infrared_value[4];

uchar str0[]=":";
uchar str1[]="-";
uchar str2[]=".";
uchar str3[]="*C
";

void main(void)
{
 Init();
 while(1){
   temp = DS18B20ReadTemp();
   tp0 = temp;
   tp1 = tp0*0.0625*100+0.5;
   Ds1302ReadTime();
   Fn[State][Key]();
   Fn_disp_buffer();
   Fn_refresh_disp();
 }
}

void Init()
{
 // only do initializatoin in main function
 TMOD=0x21; // timer0 mode1
 TH0=0xD8; // T0 with highest priority for tasks which needs rigid timing
 TL0=0xF0; //10ms
 TH1=0xF3;
  TL1=0xF3; //bund rate=4800
 //PCON register
 PCON=0x80;//bund rate double
 //SCON register
  SM0=0;
  SM1=1;    //serial mode1
  REN=1;    //enable
 //TCON register
 TR1=1;    //open timer1
 IT0=1;    //open external interrupt0, drop trigger
 //IE register
 ET0=1;    //enable timer0 interrupt 1
 EX0=1;    //enable external interrupt 0
 EA=1;     //enable all interrupt
 Key=0;
 Beep=0;
 Infrared=1;
 if(TIME[0]==0x00){
  TIME[0]=AT24C02Read(3);
  TIME[1]=AT24C02Read(2);
  TIME[2]=AT24C02Read(1);
 }
 Ds1302Init();
}

void delay (uint m)   //控制频率延时
{
 while(m--);
}

void Fn_refresh_disp(void)
```

```
{
 // function for refreshing the LED every 10 ms
 uchar i;
 for(i=0;i<8;i++){
  switch(i){
   case(0):
    LSA=0;LSB=0;LSC=0;break;
   case(1):
    LSA=1;LSB=0;LSC=0;break;
   case(2):
    LSA=0;LSB=1;LSC=0;break;
   case(3):
    LSA=1;LSB=1;LSC=0;break;
   case(4):
    LSA=0;LSB=0;LSC=1;break;
   case(5):
    LSA=1;LSB=0;LSC=1;break;
   case(6):
    LSA=0;LSB=1;LSC=1;break;
   case(7):
    LSA=1;LSB=1;LSC=1;break;
  }
    if(i==set_bit&FlashLED>=15&State==1){
      P0 = 0x00;
    }else{
      P0 = DispSegment[7-i];
    }
  delay(80);  //0.8ms
  P0 = 0x00;   //blanking
 }
}

void Fn_disp_buffer(void)
{
 // function for buffering the display variables
 float tp;
  if(State!=2){
    DispSegment[0] = SegmentData[TIME[2]/16];
    DispSegment[1] = SegmentData[TIME[2]%16];
    DispSegment[2] = 0x40;
    DispSegment[3] = SegmentData[TIME[1]/16];
    DispSegment[4] = SegmentData[TIME[1]%16];
    DispSegment[5] = 0x40;
    DispSegment[6] = SegmentData[TIME[0]/16];
    DispSegment[7] = SegmentData[TIME[0]%16];
  }else{
  //Temperature (Infrared_value)
  if(temp==0){}
  else{
   DispSegment[0] = 0x00;
   tp = temp;
   temp = tp*0.0625*100+0.5;
  }
    DispSegment[1] = SegmentData[temp/10000];
    DispSegment[2] = SegmentData[temp%10000/1000];
    DispSegment[3] = SegmentData[temp%10000%1000/100]|0x80;
    DispSegment[4] = SegmentData[temp%100/10];
    DispSegment[5] = SegmentData[temp%100%10];

  DispSegment[6] = 0x63;
  DispSegment[7] = 0x39;
 }
}

void null(void){}
void S0_timer(void)
{
 if(TIME[0]==0&TIME[1]==0&TIME[2]!=0&State==0) Key=1;
}
void S0_music(void)
{
 // input your own code
 TR0 = 1;
 music();
}
void S1_set(void)
{
 // input your own code
 Ds1302ReadTime();
  if(++FlashLED==30) FlashLED=0;   //500ms
}
void S1_add(void)
{
 // input your own code
  switch(set_bit){
    case(0):
      if(TIME[0]%16<9){TIME[0]++;}else{TIME[0]-=9;}break;
    case(1):
      TIME[0]+=0x10;if(TIME[0]>=0x60){TIME[0]-=0x60;}break;
    case(2):
      break;
    case(3):
```

```
177.       if(TIME[1]%16<9){TIME[1]++;}else{TIME[1]-=9;}break;
178.     case(4):
179.      TIME[1]+=0x10;if(TIME[1]>=0x60){TIME[1]-=0x60;}break;
180.     case(5):
181.       break;
182.     case(6):
183.       if(TIME[2]<0x20){if(TIME[2]%16<9){TIME[2]++;}else{TIME[2]-=9;}}
184.       else{if(TIME[2]%16<3){TIME[2]++;}else{TIME[2]-=3;}}
185.       break;
186.     case(7):
187.       TIME[2]+=0x10;if(TIME[2]>=0x24){TIME[2]&=0x0F;}break;
188.   }
189.  Ds1302Init();
190.   Key = 0;
191. }
192. void S2_tem(void)
193. {
194.  if(tp1%10000%1000/100>=8){TR0=1;T0RH=0xFE;T0RL=0x05;}
195.  else {TR0=0;Beep=0;}
196. }
197.
198. //infrared interrupt
199. void IR_ISR(void) interrupt 0
200. {
201.   uchar i,j;
202.   uchar Htime=0;    //high_level time, decide bit whether 0 or 1
203.   uint timeout;   //prevent a timeout
204.   Fn_refresh_disp();
205.   if(Infrared==0){
206.    timeout = 500;    //wait start_code from 0 to 1
207.    while(Infrared==0&timeout>0){delay(1);timeout--;}
208.    if(Infrared==1){  //stop start_code first 9ms 0 and enter 4.5ms 1
209.     timeout = 500;
210.     while(Infrared==1&timeout>0){delay(1);timeout--;}
211.     for(i=0;i<4;i++){     //4 data form
212.      for(j=0;j<8;j++){   //8 bits of one code
213.       timeout = 60;     //0.56ms low_level
214.       while(Infrared==0&timeout>0){delay(1);timeout--;}
215.       while(Infrared==1){
216.        delay(10);
217.        Htime++;
218.        if(Htime>30) return;  //beyond 3ms, wrong
219.       }
220.       Infrared_value[i]>>=1;
221.       if(Htime>=10) Infrared_value[i]|=0x80;
222.       Htime = 0;
223.      }
224.     }
225.    }
226.   }
227.   Fn_refresh_disp();
228.   //data code != ~data complement code, wrong
229.   if(Infrared_value[0]!=0x00|Infrared_value[1]!=0xFF) return;
230.   if(Infrared_value[2]!=~Infrared_value[3]) return;
231.   if(++debouncing==3){
232.      if(Infrared_value[2]==0x46) {
233.        if(State++>=2){State=0;set_bit=0;}    //set mode
234.      }else if(Infrared_value[2]==0x40){    //set_bit move left
235.        if(set_bit==1|set_bit==4) set_bit++;
236.        if(++set_bit>7) set_bit=0;
237.      }else if(Infrared_value[2]==0x43){    //set_bit move right
238.        if(set_bit==3|set_bit==6) set_bit--;
239.        if(set_bit--==0) set_bit=7;
240.      }else if(Infrared_value[2]==0x09&State==1) {
241.     Key=1;
242.    }
243.    else if(Infrared_value[2]==0x0C){
244.     AT24C02Write(1,TIME[2]);
245.
246.     AT24C02Write(2,0);
247.     AT24C02Write(2,TIME[1]);
248.
249.     AT24C02Write(3,0);
250.     AT24C02Write(3,TIME[0]);
251.
252.     AT24C02Write(4,0);
253.     AT24C02Write(4,BCD(tp1/100));
254.
255.     AT24C02Write(5,0);
256.     AT24C02Write(5,BCD(tp1%100));
257.    }else if(Infrared_value[2]==0x5E){
258.     for(i=1;i<6;i++) AT24C02Write(i,0);
259.     AT24C02Write(2,0);
260.     AT24C02Write(2,0);
261.     AT24C02Write(4,0);
262.     AT24C02Write(4,0);
263.    }else if(Infrared_value[2]==0x18){
264.     Uchar2String(AT24C02Read(1));
265.     SendString(str0);
266.     Uchar2String(AT24C02Read(2));
267.     SendString(str0);
268.     Uchar2String(AT24C02Read(3));
```

```
269.     SendString(str1);
270.     Uchar2String(AT24C02Read(4));
271.     SendString(str2);
272.     Uchar2String(AT24C02Read(5));
273.     SendString(str3);
274.    }
275.      debouncing=0;
276.    }
277.  }
```

2. Multifunction_clock.h

```
#ifndef __MULTIFUNCTION_CLOCK_H__
#define __MULTIFUNCTION_CLOCK_H__

#include <reg52.H>
#include <DS18B20.h>
#include <DS1302.h>
#include <MUSIC.h>
#include <AT24C02.h>
#include <UART.h>

#define uchar unsigned char
#define uint unsigned int

sbit LSA=P2^2;  //select LED 7-segment
sbit LSB=P2^3;
sbit LSC=P2^4;
sbit Infrared=P3^2;  //infrared interrupt, INT0

void Init();
void S0_timer(void); // declaring the state-input transition functions (can be put in .h file)
void S0_music(void);
void S1_set(void);
void S1_add(void);
void S2_tem(void);
void null(void);
void delay(uint x);
void Fn_refresh_disp(void);
void Fn_disp_buffer(void);

extern uchar Key;
#endif
```

3. AT24C02.c

```
1. /*********************************************************************
2. [功能]    EEPROM储存时间和温度
3.
4. *********************************************************************/
5. #include <AT24C02.h>
6.
7. void Delay5us()
8. {
9.  uchar a;
10.  for(a=1;a>0;a--);
11. }
12.
13. void Start()
14. {
15.  SDA = 1;
16.  Delay5us();
17.  SCL = 1;
18.  Delay5us();
19.  SDA = 0;       //failing
20.  Delay5us();
21.  SCL=0;
22.  Delay5us();
23. }
24.
25. void Stop()
26. {
27.  SDA = 0;
28.  Delay5us();
29.  SCL = 1;
30.  Delay5us();
31.  SDA = 1;      //rising
32.  Delay5us();
33. }
34.
35. void Answer()
36. {
37.  uchar i=0;
38.  SCL = 1;
```

```
 while(SDA&i<250) i++;
 SDA = 0;
 Delay5us();
 SCL=0;
 Delay5us();
}

void AT24C02WriteByte(uchar dat)
{
 uchar i;
 for(i=0;i<8;i++){
  SCL = 0;       //send dat to SDA when SCL=0
  Delay5us();
  SDA = dat>>7;
  Delay5us();
  dat<<=1;
  SCL = 1;
  Delay5us();
 }
 SCL = 0;
 Delay5us();
 SDA = 1;        //release SDA
 Delay5us();
}

uchar AT24C02ReadByte()
{
 uchar i;
 uchar dat=0;
 SDA=1;   //起始和发送一个字节之后SCL都是0
 Delay5us();
 for(i=0;i<8;i++){
  SCL = 1;       //send dat to SDA when SCL=0
  Delay5us();
  dat = (dat<<1)|SDA;
  Delay5us();
  SCL = 0;
  Delay5us();
 }
 return dat;
}

void AT24C02Write(uchar addr,uchar dat)
{
 Start();
 AT24C02WriteByte(0xA0);
 Answer();
 AT24C02WriteByte(addr);
 Answer();
 AT24C02WriteByte(dat);
 Answer();
 Stop();
}

uchar AT24C02Read(uchar addr)
{
 uchar dat;
 Start();
 AT24C02WriteByte(0xA0);
 Answer();
 AT24C02WriteByte(addr);
 Answer();
 Start();
 AT24C02WriteByte(0xA1);
 Answer();
 dat = AT24C02ReadByte();
 Stop();
 return dat;
}
```

4. AT24C02.h

```
#ifndef __AT24C02_H__
#define __AT24C02_H__

#include <reg52.h>

#define uchar unsigned char
#define uint unsigned int

sbit SCL=P2^1;
sbit SDA=P2^0;

void AT24C02Write(uchar addr,uchar dat);
uchar AT24C02Read(uchar addr);

#endif
```

5. DS18B20.c

```
/*****************************************************************
[功能]    温度读取

*****************************************************************/
#include <DS18B20.h>

void delay1ms(uchar i)
{
   uchar j;
   for(;i>0;i--){
     for(j=110;j>0;j--);
   }
}
uchar DS18B20Init()    //success=1,fail=0
{
   uint i;
   TemPort = 0;
   for(i=0;i<80;i++);   //8*80+3=643us
   TemPort = 1;         //high
   while(TemPort){      //wait for low in 480us
     delay1ms(1); //wait 1ms
     i++;
     if(i>5) return 0;          //Initialization failed
   }
   return 1;            //Initialization successful
}

void DS18B20Write(uchar dat)
{
   uchar i,j;
   for(i=0;i<8;i++){
     TemPort = 0;
     j++;
     TemPort = dat&0x01;
     for(j=0;j<8;j++);  //delay≈64us+3us>60us
     TemPort = 1;
     dat>>=1;           //next bit
   }
}

uchar DS18B20Read()
{
   uchar i,j,k;
   uchar b,byte;
   for(i=0;i<8;i++){
     TemPort = 0;
     k++;               //delay >1us
     TemPort = 1;
     for(j=0;j<1;j++);  //delay >10us
     b = TemPort;
     byte = (byte>>1)|(b<<7);
     for(j=0;j<6;j++);  //delay>45us
   }
   return byte;
}

//transform temperature command
void DS18B20TransTempCom()
{
   DS18B20Init();
   delay1ms(1);
   DS18B20Write(0xCC);    //jump ROM
   DS18B20Write(0x44);    //start to transform Temperature
}

//read temperature command
void DS18B20ReadTempCom()
{
   DS18B20Init();
   delay1ms(1);
   DS18B20Write(0xCC);    //jump ROM
   DS18B20Write(0xBE);    //start to read temperature
}

//read temperature value
int DS18B20ReadTemp()
{
   int temp0 = 0;
   uchar TemH,TemL;
   DS18B20TransTempCom();
   DS18B20ReadTempCom();
   TemL = DS18B20Read();   //read low 8 bits
   TemH = DS18B20Read();   //read high 8 bits
   temp0 = TemH;            //joint 16bits high-low temperature
   temp0<<=8;
   temp0|=TemL;
   return temp0;
}
```

6. DS18B20.h

```
#ifndef _DS18B20_H
#define _DS18B20_H

#include <reg52.h>

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

int DS18B20ReadTemp();

sbit TemPort = P3^7;

#endif
```

7. DS1302.c

```
/**************************************************************************
[功能]    时钟计时

**************************************************************************/
#include <DS1302.h>

uchar code READ_RTC_ADDR[7] = {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};
uchar code WRITE_RTC_ADDR[7] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};

//second,minute,hour,day,month,week,year,BCD code---//
uchar TIME[7] = {0, 0, 0x00, 0x07, 0x05, 0x06, 0x16};

void Ds1302Write(uchar addr, uchar dat)
{
 uchar n;
 RST = 0;
 _nop_();
 SCLK = 0;
 _nop_();
 RST = 1;
 _nop_();
 for (n=0; n<8; n++)//send 8bits address
 {
  DSIO = addr & 0x01;//from LSB to MSB
  addr >>= 1;
  SCLK = 1;//DS1302 read when SCLK rising
  _nop_();
  SCLK = 0;
  _nop_();
 }
 for (n=0; n<8; n++)//write 8bits data
 {
  DSIO = dat & 0x01;
  dat >>= 1;
  SCLK = 1;        //DS1302 read when SCLK rising
  _nop_();
  SCLK = 0;
  _nop_();
 }

 _nop_();
}

uchar Ds1302Read(uchar addr)
{
 uchar n,dat,dat1;
 RST = 0;
 _nop_();
 SCLK = 0;
 _nop_();
 RST = 1;
 _nop_();
 for(n=0; n<8; n++)//send 8bits address
 {
  DSIO = addr & 0x01;
  addr >>= 1;
  SCLK = 1;
  _nop_();
  SCLK = 0;
  _nop_();
```

```
 }
 _nop_();
 for(n=0; n<8; n++)//read 8bits data
 {
  dat1 = DSIO;//LSB-MSB
  dat = (dat>>1) | (dat1<<7);
  SCLK = 1;
  _nop_();
  SCLK = 0;//get data when SCLK failling
  _nop_();
 }
 //RST = 0;
 _nop_(); //wait the DS1302 reset
 SCLK = 1;
 _nop_();
 DSIO = 0;
 _nop_();
 DSIO = 1;
 _nop_();
 return dat;
}

void Ds1302Init()
{
 uchar n;
 Ds1302Write(0x8E,0X00);   //close wirte protection
 for (n=0; n<7; n++)//write initial time
 {
  Ds1302Write(WRITE_RTC_ADDR[n],TIME[n]);
 }
 Ds1302Write(0x8E,0x80);   //open wirte protection
}

void Ds1302ReadTime()
{
 uchar n;
 for (n=0; n<7; n++)//read current time
 {
  TIME[n] = Ds1302Read(READ_RTC_ADDR[n]);
 }
}
```

8. DS1302.h

```
#ifndef __DS1302_H_
#define __DS1302_H_

#include<reg52.h>
#include<intrins.h>

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

sbit DSIO=P3^4;
sbit RST=P3^5;
sbit SCLK=P3^6;

void Ds1302Write(uchar addr, uchar dat);
uchar Ds1302Read(uchar addr);
void Ds1302Init();
void Ds1302ReadTime();

extern uchar TIME[7]; //加入全局变量

#endif
```

9. music.c

```
/*****************************************************************
[功能]    整点报时

*****************************************************************/
#include <MUSIC.h>


uchar T0RH=0xFF,T0RL=0x00;
uchar i=0;
uchar n=0;
uint code noteFreq[] = {    //低音，中音 1-7 和高音 1-7对应的频率列表
```

```
 0xF88B,0xF95B,0xFA14,0xFA66,0xFB03,0xFB8F,0xFC0B,
 0xFC43,0xFCAB,0xFD08,0xFD33,0xFD81,0xFDC7,0xFE05,
 0xFB21,0xFE55,0xFE84,0xFE99,0xFEC0,0xFEE3,0xFF02,0xFF,0x00
};

uchar code music_tab[] = {      //简谱+节拍
    mid(1),40, mid(2),40, mid(3),40, mid(1),40,
    mid(1),40, mid(2),40, mid(3),40, mid(1),40,
    mid(3),40, mid(4),40, mid(5),40,mid(5),20,
    mid(3),40, mid(4),40, mid(5),40,mid(5),20,
    mid(5),20, mid(6),20, mid(5),20,mid(4),20, mid(3),40, mid(1),40,
    mid(5),20, mid(6),20, mid(5),20,mid(4),20, mid(3),40, mid(1),40,
    mid(1),40, 5,40, mid(1),40,mid(1),40,
  mid(1),40, 5,40, mid(1),40,mid(1),40,22
};

void delayms(uint x)
{
    uint y;
    for(;x>0;x--)
        for(y=110;y>0;y--);
}

void music(void)
{
 uint p,m;   //m为频率常数变量
 if(n==0){
  p=noteFreq[music_tab[i]];
  if(p==0x00)       { i=0;TR0=0;Key=0;Beep=0;return;}     //如果碰到结束符,结束
  else if(p==0xff)  { i=i+1;delayms(100);return;}  //若碰到休止符,延时100ms,继续取下一音符
  else {m=noteFreq[music_tab[i++]], n=music_tab[i++]/2;}  //取频率常数 和 节拍常数
  T0RH = m/256;
  T0RL = m%256;
 }
 else{n--;}
}

void InterruptTmr0() interrupt 1 {
    TH0 = T0RH;
    TL0 = T0RL;
    Beep = ~Beep;
}
```

10. MUSIC.h

```
#ifndef __MUSIC_H__
#define __MUSIC_H__

#include <reg52.H>
#include <Multifunction_clock.h>

#define uchar unsigned char
#define uint unsigned int
#define mid(n) (n+7)
#define high(n) (n+14)

sbit Beep=P1^5;

void music(void);

extern uchar T0RH,T0RL;
#endif
```

11. UART.c

```
/*****************************************************************
[功能]    发送时间和温度给PC

*****************************************************************/
#include <UART.h>

void trans(uchar x) //trans one character
{
  SBUF=x;
  while(!TI);
  TI=0;
}

void SendString(uchar *dat){             //trans initial string
 while((*dat != ' ')&(*dat!='s'))
 {
  trans(*dat);
  dat++;
 }
```

```
}

void Uchar2String(uchar x)
{
  uchar string0[10]={"0123456789"};
  uchar string[3];
  string[0] = string0[x/16];
  string[1] = string0[x%16];
  string[2] = 's';
  SendString(string);
}
```

12. UART.h

```
#ifndef __UART_H__
#define __UART_H__

#include <reg52.h>

#define uchar unsigned char
#define uint unsigned int

void SendString(uchar *dat);
void Uchar2String(uchar x);
#endif
```

VII. Reference

- 基于51单片机IIC通信的AT24C02学习笔记: https://www.cnblogs.com/whik/p/6650092.html
- 【单片机】普中单片机视频教程完整版: https://www.bilibili.com/video/av19430241?from=search&seid=10712074114264622703
- [美]穆罕默德·阿里·马兹迪（Muhammad Ali[M]. 北京： 机械工业出版社，2016.1.