特别说明

目　录

摘　要

系统由监测终端和若干环境探测节点组成，监测终端由 C8051F120 单片机、液晶显示器、无线收发电路构成，环境探测节点由 C8051F340 单片机、光照探测电路、温度采集电路、探测节点编码预置电路、无线收发电路构成。整个系统由监测终端作为主控制端，环境探测节点作为分支节点，监测终端和环境探测节点采用 27.145MHz 的调频方式进行无线通信。每个环境探测节点在监测终端的控制下分时向监测终端提供环境探测节点的预置编号、温度、光照情况等参数信息，而监测终端则实时显示相应环境参数信息。系统成本低，无线通信稳定，传输距离大于 50 厘米，探测节点功耗小于 0.3 瓦，环境信息更新周期小于 5 秒，满足设计要求。

关键词：环境监测、传感器、无线通信、调频、液晶显示、C8051F120、C8051F340

Abstract

The system consists of the monitoring terminal and a number of environmental detection notes, the monitoring terminal are mainly made up of the C8051F120 MCU, LCD display, wireless transceiver circuits and the environment detection notes are made up of C8051F340 MCU light detection circuit, the temperature acquisition circuit, detecting node pre-coding circuit and wireless transceiver circuit. The entire system is mainly controlled by the monitoring terminal while environmental detection notes are wireless branch nodes and they communicate by the 27.145MHz FM wireless way. Each environmental detection node transmit their own preset number, temperature, light conditions and other parameter information time-shared under the control of the monitoring terminal while the monitoring terminal display the node information real-time received from the environmental detection notes ,so the system realizes multi-point monitoring of the surrounding environment. Wireless communication system works stably, transmission distance is more than 50 cm, power consumption of the detection node is less than 0.3 watts, environmental information update cycle is less than 5 seconds, the entire system meets the design requirements.

Keywords: environmental monitoring, sensors, wireless communications, frequency modulation, ,liquid crystal display， C8051F120, C8051F340

1. 系统方案

1.1 题目解析

根据命题要求，设计并制作一个无线环境监测模拟装置，实现对周边温度和光照信息的探测。该装置由 1 个监测终端和不多于 255 个探测节点组成（实际制作 2 个）。监测终端和探测节点均含一套无线收发电路，要求具有无线传输数据功能，收发共用一个天线。

主要性能指标有：（1）制作 2 个探测节点。探测节点有编号预置功能，编码预置范围为 00000001B～11111111B。探测节点能够探测其环境温度和光照信息。温度测量范围为 0℃～100℃，绝对误差小于 2℃；光照信息仅要求测量光的有无。探测节点采用两节 1.5V 干电池串联，单电源供电。（2）制作 1 个监测终端，用外接单电源供电。监测终端可以分别与各探测节点直接通信，并能显示当前能够通信的探测节点编号及其探测到的环境温度和光照信息。（3）无线环境监测模拟装置的探测时延不大于 5s，监测终端天线与探测节点天线的距离 D 不小于 10cm。在 0～10cm 距离内，各探测节点与监测终端应能正常通信。（4）发挥部分要求每个探测节点增加信息的自动转发功能，在监测终端电源供给功率≤1W，无线环境监测模拟装置探测时延不大于 5s 的条件下，使探测距离 $D+D_1$ 达到 50cm。尽量降低各探测节点的功耗，以延长干电池的供电时间。各探测节点应预留干电池供电电流的测试端子。

1.2 系统总体方案设计

根据试题要求，设计需要有三部分组成：一是监测终端模块，二是探测节点模块，三通信协议规则设计。下面分别从这三方面阐述方案的设计思想。

1.2.1 监测终端模块设计

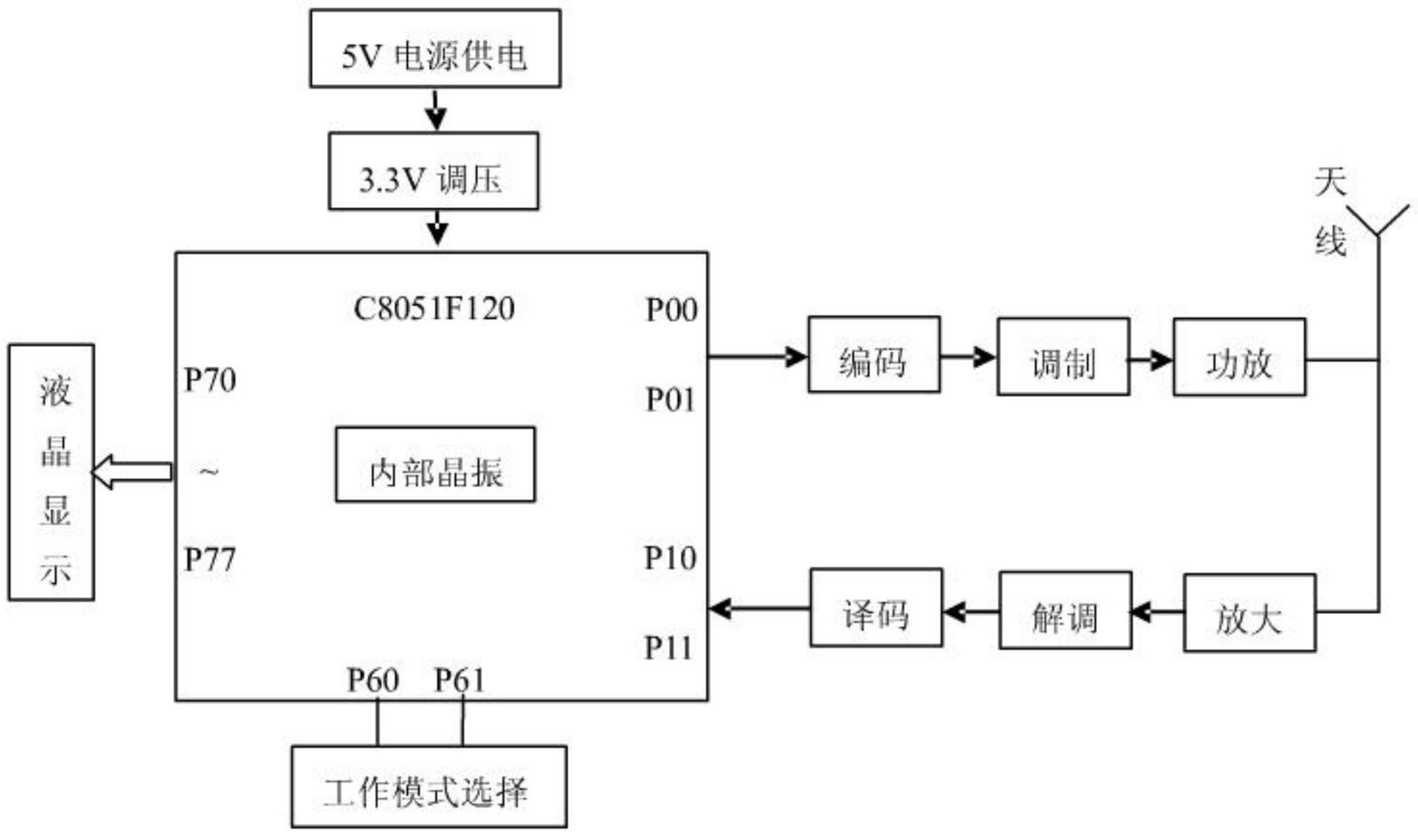

图 1　监测终端模块设计方案

监测终端模块设计方案如图 1 所示，监测终端模块选用 C8051F120 单片机作为主控制器，控制通信模式的选择，控制探测节点发送数据，控制液晶显示探测节点的编号、温度、光照等参数。采用两位二进制码表示通信模式。采用八位二

进制码表示各监测节点的编号。监测终端模块通过向探测节点发送八位编号命令选择探测节点向其发送数据。为了便于码元的传输，通过编码芯片 TX-2B 对码元进行二次编码，以提高数据传输的可靠性，通过调制电路对信号进行 FSK 调制，调制频率为 27.145MHz，信号经放大电路通过天线发送给探测节点。

1.2.2 探测节点模块设计

探测节点模块设计方案如图 2 所示，探测节点模块选用 C8051F340 单片机作为主控器，在低功耗的控制通信方式，控制光电传感器、温度传感器的数据采集，可以读取可预置的探测节点编号。用十六位二进制码串表示探测节点编号、环境温度值、光照信息。根据通信协议规则，在接到监测终端的命令后开始向其发送十六位二进制码串，通过编码芯片 TX-2B 二次编码，通过调制电路对信号进行 FSK 调制，调制频率为 27MHz，码串经过放大电路通过天线发送给监测终端。

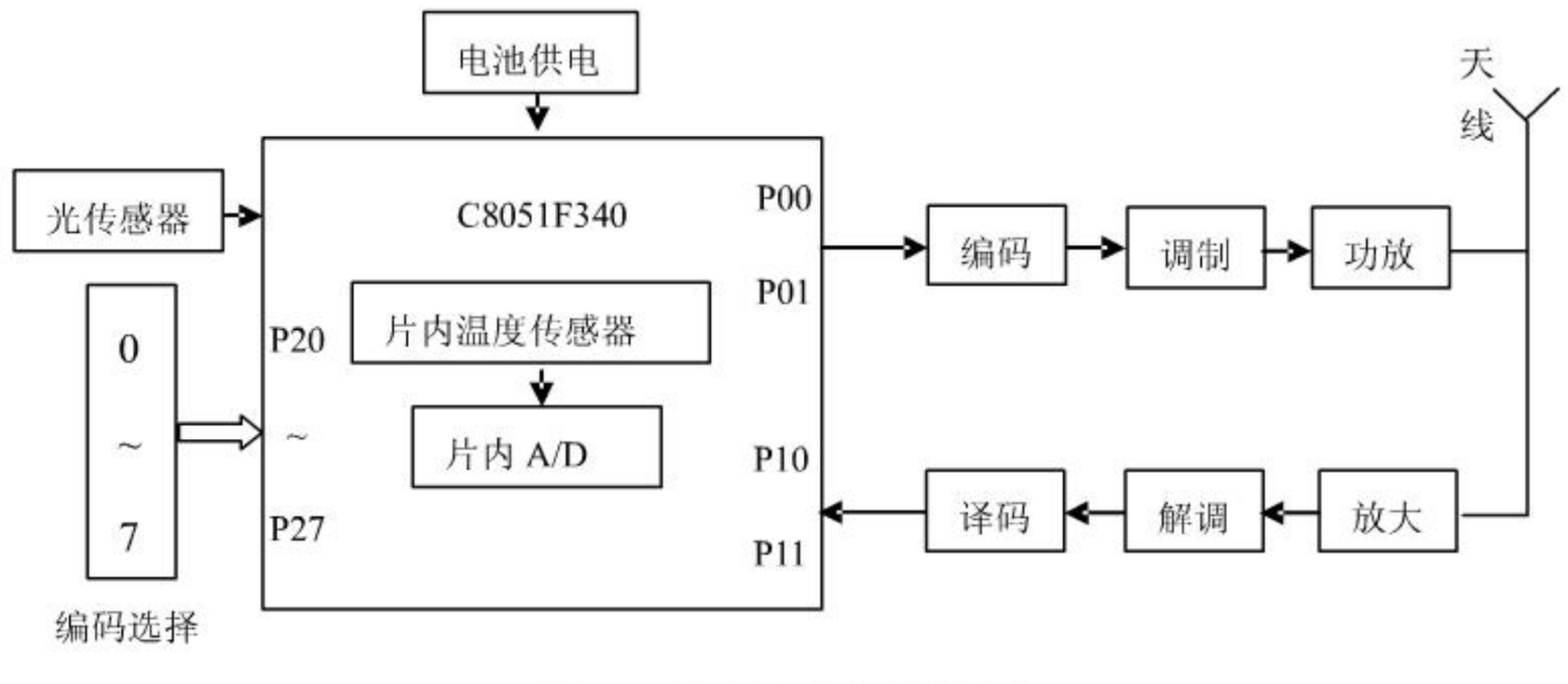

图 2　探测节点模块设计方案

1.2.3 通信协议规则设计

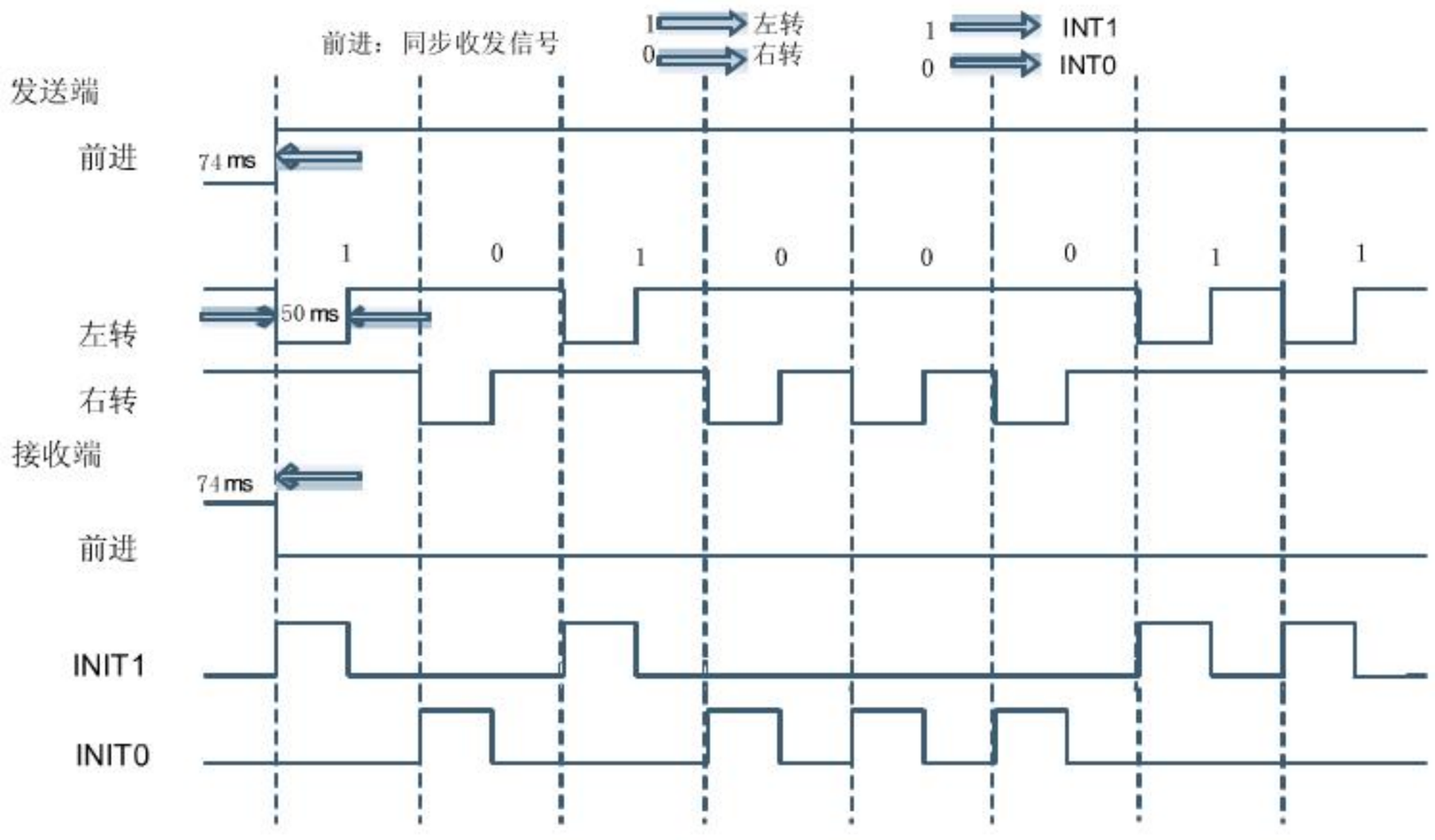

图3 通信协议图

无线通信的时序如图3所示。具体无线通信协议规则如下：

1、发送协议：发送端单片机先由IO引脚发送低电平有效的同步脉冲，然后由单片机发1引脚和发0引脚分别接到编码器TX-2B的“左转”和“右转”引脚，

通过使能编码器TX-2B“左转”和“右转”引脚来实现1、0信号的发送编码，其中所谓的使能是指给相应引脚发送脉冲来实现，该脉冲先低电平后高电平，高低电平各持续50ms。

2、接收协议：接收端译码器RX-2B的“左转”引脚和“右转”引脚分别接到单片机INT1和INT0，通过使能单片机INT1和INT0来实现1、0信号的发送译码，其中使能是指给相应引脚发送脉冲实现，该脉冲先高电平后低电平，高低电平各持续50ms，而接收同步信号由译码器RX-2B的“前进”引脚高电平实现。

2. 电路分析与计算

2.1 监测终端和环境探测节点单片机控制模块电路

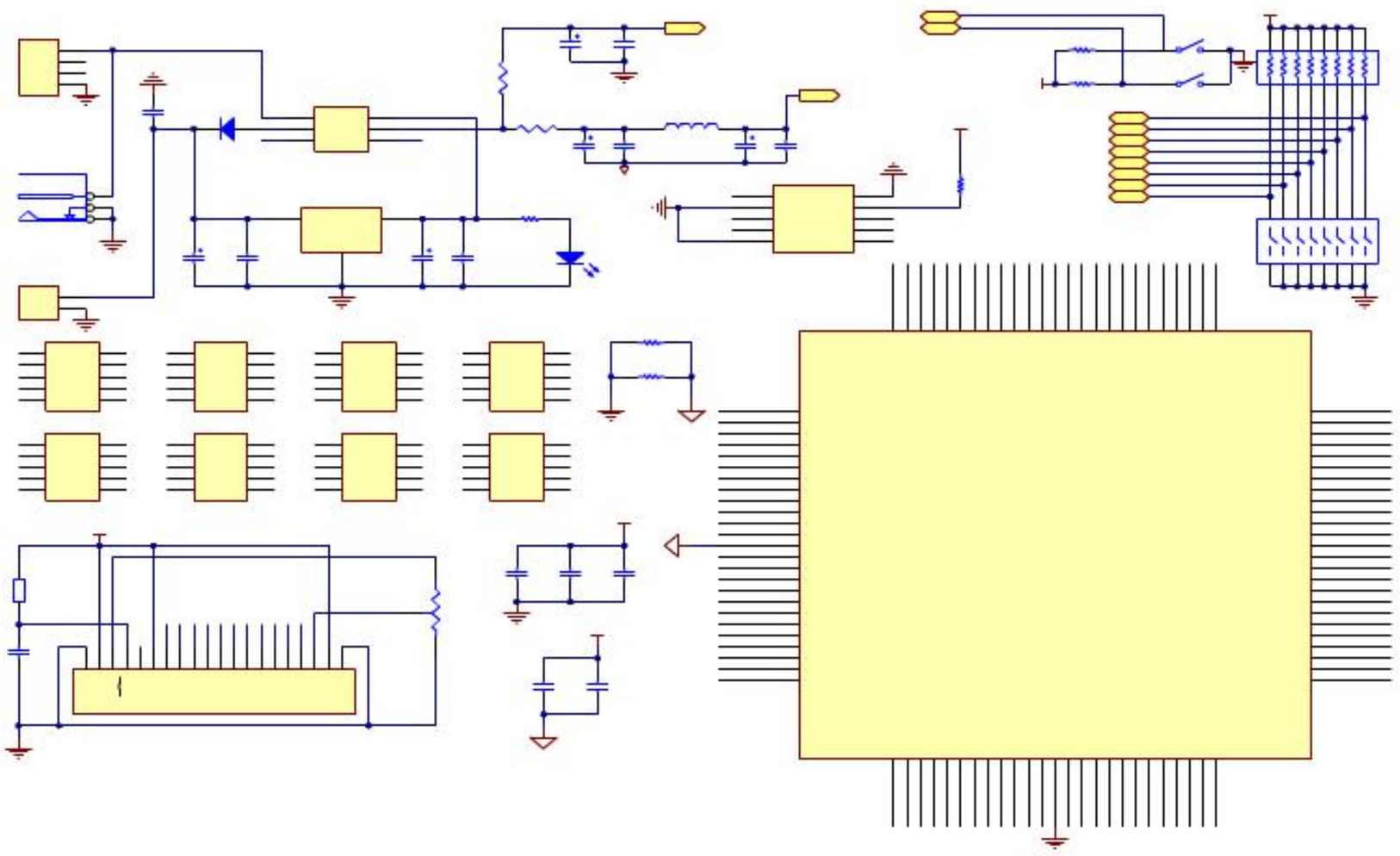

图 4 C8051F120 单片机控制电路

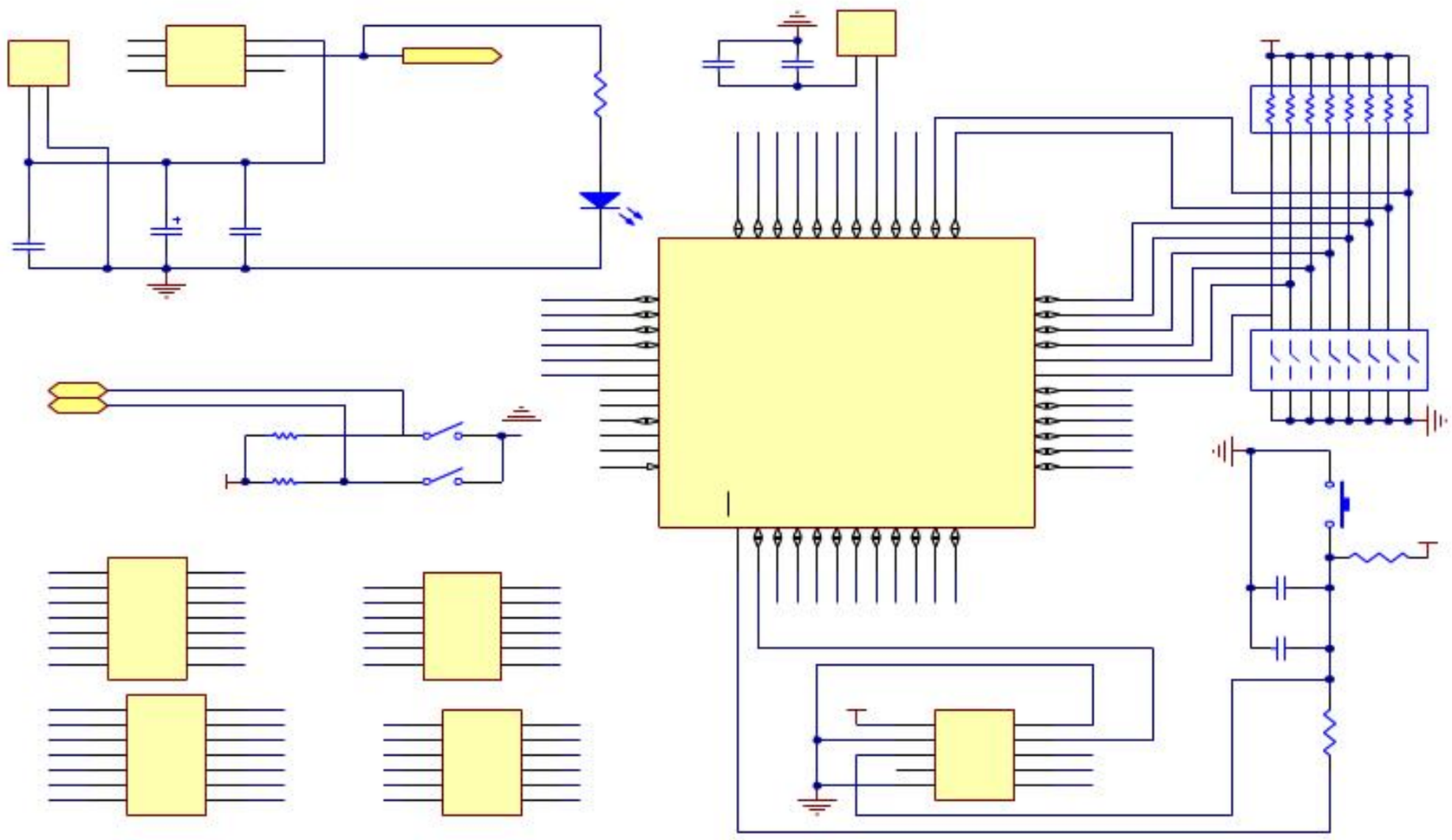

图5 C8051F340单片机控制电路

监测终端的控制模块主要由C8051F120单片机最小系统、液晶显示和模式选则电路构成，如图4所示；环境探测节点的控制模块主要由C8051F340单片机最小系统、预置码输入电路和模式选择电路构成，如图5所示。

2.2 光与温度信号采集电路

光信号采集电路如图6所示。本设计采用光敏二极管对光照信息进行数据采集，根据光敏二极管感光电流可变的原理，设计了如图6所示的光信号检测电路。当有光照射时，光敏二极管电流变大，相应地加载到电阻上的电压就会变大，此电压经比较器LM393P输出高电平,反之输出低电平。470k电阻上的电压变化范围为0.01V到0.48V，所以比较器选择的参考电压为0.2V，此电路可以精确、高速测试到光照的变化。温度采集利用C8051F340片内温度传感器与A/D转换器完成。

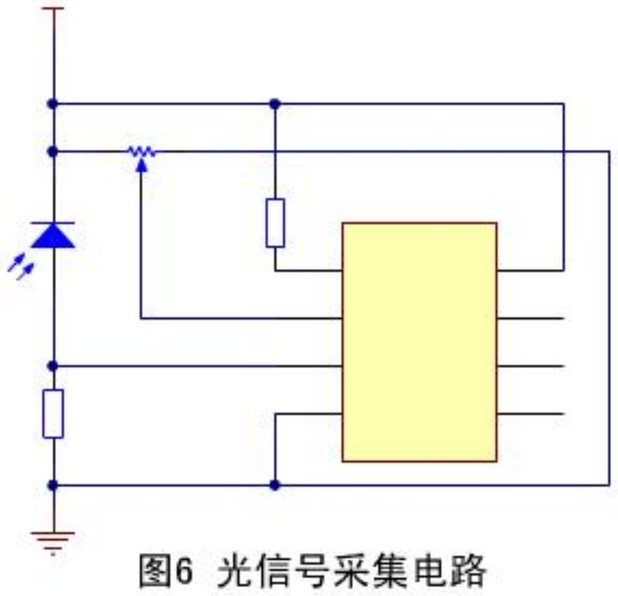

图6 光信号采集电路

2.3 发射电路分析

发射电路设计如图7所示。利用TX-2B对单片机输出的数据编码进行二次编码，由27.145MHZ的无源晶振与自激电路产生载波信号，码串经调制之后放大，通过天线发射，且当发射信号时LED灯亮。

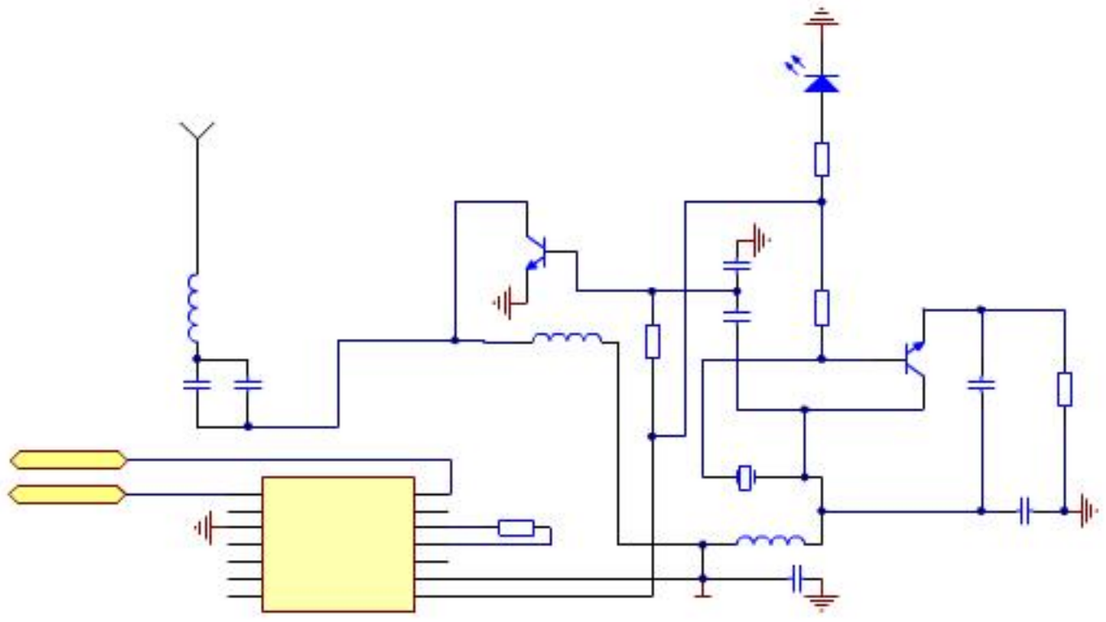

图7 发射电路

2.4 接收电路分析

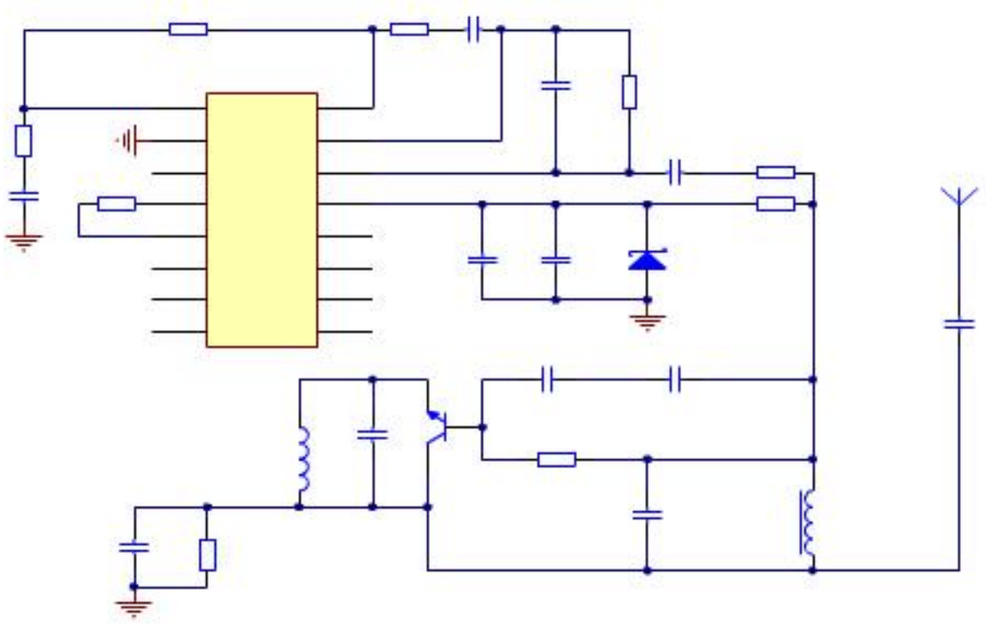

图8 接收电路

如图8所示，本设计采用RX-2B无线接收芯片，接收携带命令或者数据信息的

27.145M射频信号，经过放大电路，经射频接收电路解调还原成相应的命令或者数据二进制码串。

3. 系统程序设计

3.1 监测终端程序设计

监测终端的程序流程图如图9所示，其中基本模式（00）是指终端先给节点A发射发送数据命令，然后接收节点A所发送的数据并显示该数据信息，之后再向节点B发射发送数据命令，然后接收节点A所发送的数据并显示该数据信息。中继模式是指终端发送中继命令给最近节点，然后接收最近节点发送的32位二进制数据信息，将其拆分为两个16位二进制并分别显示对应的温度与光照信息。

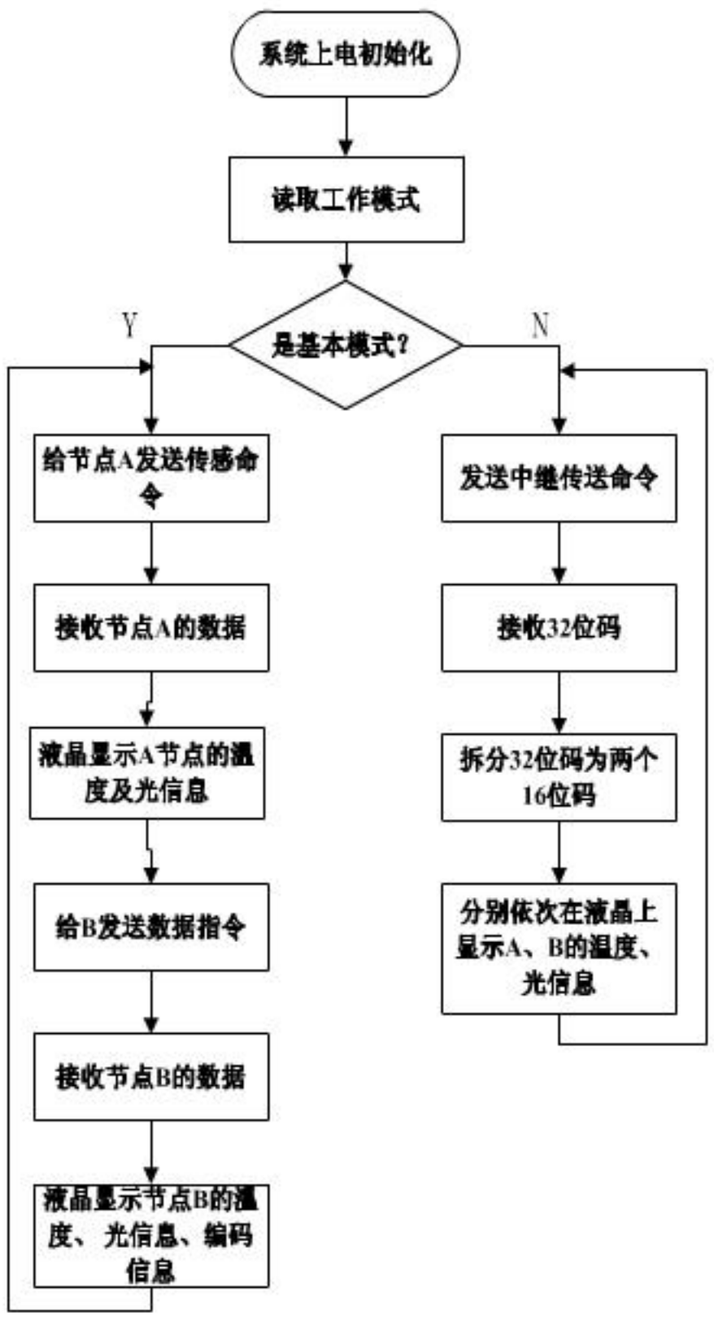

图 9 监测终端的 C8051F120 程序流程图

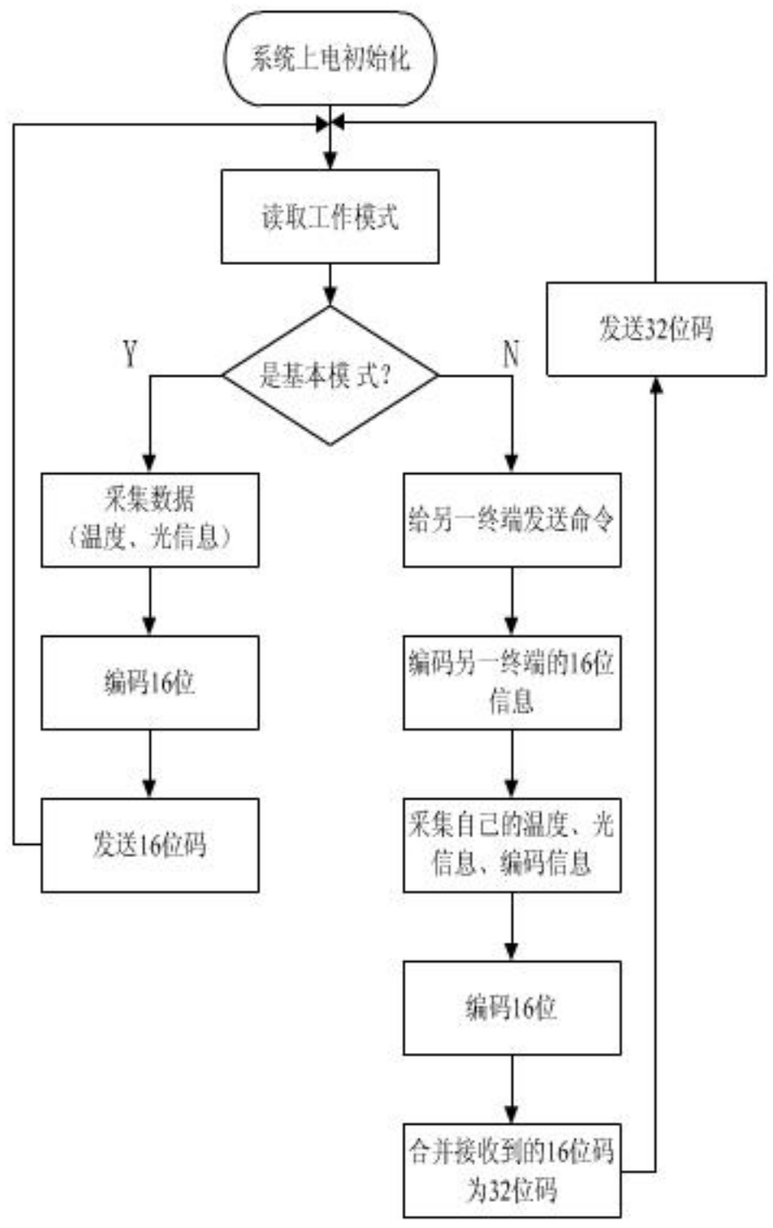

图 10 探测节点的 C8051F340 程序流程图

3.2 探测节点程序设计

探测节点的程序流程图如图 10 所示，其中基本模式是指该探测节点采集温度与光照信息，对该信息编码并发送给终端。中继模式是指该探测节点给另一节点发送基本命令，然后接收另一节点所发送的数据信息，之后与自己所采集的温度与光照信息组合为 32 位二进制码并发送给终端。

4. 测试方案与测试结果

4.1 调试方法与仪器

测试仪器：KIGOL DS5062MA 示波器、EE1640C 函数信号发生器、C8051F 单片机 EC-3 仿真器等。

调试方法：首先测试收发的幅频特性，用信号源产生一定频率，通过发射电路将电磁波发射出去，然后通过接收电路接收，测试传输距离以及准确的发射频

率。其次是所需的数据传输的测试，通过控制模块模拟输入信号，采用示波器观察记录波形数据。最终完成系统的搭建，进入最终的联合测试。

4.2 测试数据完整性

表 1　数据记录表

	测试次数	1 次	2 次	3 次	4 次	5 次
节点 1	测试温度	15	23	36	39	47
	实际温度	15	23	36	38	46
节点 2	测试温度	17	21	34	39	42
	实际温度	17	21	34	39	41
	测试次数	1 次	2 次	3 次	4 次	5 次
节点 1	测试光照	无	有	无	有	有
	实际光照	无	有	无	有	有
节点 2	测试光照	有	无	无	有	有
	实际光照	有	无	无	有	有
	测试次数	1 次	2 次	3 次	4 次	5 次
节点 1	测试编码	00011001	01011000	10100011	11001001	00010000
	实际编码	00011001	01011000	10100011	11001001	00010000
节点 2	测试编码	01011010	11010111	00000001	11000111	11000010
	实际编码	01011010	11010111	00000001	11000111	11000010
	功耗	距离	环境信息更新周期			
节点 1	0.234 W	56 cm	合计周期			
节点 2	0.236 W	60 cm	4.2 s			

4.3 测试结果分析

本系统以 C8051F 单片机芯片为控制核心部件，利用软件编程，实现了对环境温度、光照信息以及环境监测节点预置编号的采集、短距离无线通信以及液晶显示等功能。光源检测与编码部分都完全正确，温度检测出现了误差，测试温度比实际的温度高，这是由于使用单片机自带温度传感器，单片机工作时功率消耗产生热量，从而使测试温度比实际温度高，但通过对单片机采集温度的实验校正，有效克服了上述问题。另外，监测节点由于采用了 3.3V 的低功耗 C8051F340 单片机以及对无线收发电路功率的控制，使得实际测得的监测节点电流小于 75 毫安,故而功率小于 0.245 瓦。两个监测节点实际有效测试距离均大于 50 厘米，而两个监测节点环境参数信息更新周期合计小于 4.2 秒。

通过最终的测试结果可以看出题目中要求的各项参数指标均能实现，而且有些指标远高于题目的要求，性能可靠稳定。

参考文献

[1] 高吉祥.全国大学生电子设计竞赛培训系列教程,北京：电子工业出版社，2007.5.
[2] 黄智伟.全国大学生电子设计竞赛制作实训，北京：北京航空航天大学出版社，2007.2.
[3] 周晓光.王晓华.射频识别（RFID）技术原理与应用实例,北京：人民邮电出版社,2006.12.
[4] 童诗白.华成英.模拟电子技术基础,北京： 高等教育出版社,2004.10.
[5] 戴仙金.51 单片机及其 C 语言程序开发实例,北京:清华大学出版社,2008.2.
[6] 李朝青.单片机原理及接口技术，北京：北京航空航天大学出版社，2008.7.

附录：C 语言程序

C8051F340 发送程序：

```
#include<c8051f340.h>
#include<math.h>
sbit TRAN0=P0^0;
sbit TRAN1=P0^1;
sbit    TRAN_tb=P0^2;
sbit guangceshi = P0^3;
void Sendchar (unsigned int x);
void Timer_Init(void);
void ADC_Init(void);
void Port_IO_Init(void);
void Oscillator_Init(void);
void Interrupts_Init(void);
void Voltage_Reference_Init(void);
void Init_Device(void);
unsigned int caiji(void);
unsigned int caiji_value;
void delay32(void);
void delay24(void);
unsigned char ADC_FLAG=0;
unsigned char flag_int=0;
void main(void)
{
    PCA0MD &= ~0x40;    // Disable Watchdog timer
    Init_Device();
    while(1)
    {
        caiji_value=caiji();
        Sendchar(caiji_value);
       delay32();
    }
}
void delay32()
{
 int panduan = 64;
while(panduan > 0)
{
        TL0          = 0x57;
         TH0          = 0x9E;
        TR0=1;
        flag_int = 1;
```

```
            while(  flag_int == 1)
            {
            };
             panduan--;
      };
}
void delay24()
{
       int panduan = 48;
      while(panduan > 0)
      {
            TL0             = 0x57;
             TH0             = 0x9E;
            TR0=1;
            flag_int = 1;
            while(  flag_int == 1)
            {
            };
             panduan--;
      };
}

unsigned int caiji()
{
       unsigned int caiji_data=0,jiedian_bianhao=0,wendu=0,guang=0;
      int wendu_temp1=0,wendu_temp2=0;
       jiedian_bianhao = P2;
       guang = guangceshi;
      ADC_FLAG = 1;
       AD0BUSY=1;
      while(ADC_FLAG == 1)
      {
      };
       wendu_temp1 = ADC0H;
      wendu_temp2 = ((wendu_temp1<<8)|ADC0L);
      wendu = 0.82*wendu_temp2-271.33;
       caiji_data =    ((wendu << 1) | guang);
       caiji_data = (jiedian_bianhao<<8)|caiji_data;
      return(caiji_data);
}

void Sendchar (unsigned int x)
{
```

```
 unsigned int v,temp[16];
 int i=15;
v=x;
if( (v & 0x01)!=0 )
        temp[0]=1;
else
        temp[0]=0;
if( (v & 0x02)!=0 )
        temp[1]=1;
else
        temp[1]=0;
if( (v & 0x04)!=0 )
        temp[2]=1;
else
        temp[2]=0;
if( (v & 0x08)!=0 )
        temp[3]=1;
else
        temp[3]=0;
if( (v & 0x10)!=0 )
        temp[4]=1;
else
        temp[4]=0;
if( (v & 0x20)!=0 )
        temp[5]=1;
else
        temp[5]=0;
if( (v & 0x40)!=0 )
        temp[6]=1;
else
        temp[6]=0;
if( (v & 0x80)!=0 )
        temp[7]=1;
else
        temp[7]=0;
if( (v & 0x100)!=0 )
        temp[8]=1;
else
        temp[8]=0;
if( (v & 0x200)!=0 )
        temp[9]=1;
else
        temp[9]=0;
```

```
if( (v & 0x0400)!=0 )
      temp[10]=1;
else
      temp[10]=0;
if( (v & 0x0800)!=0 )
      temp[11]=1;
else
      temp[11]=0;
if( (v & 0x1000)!=0 )
      temp[12]=1;
else
      temp[12]=0;
if( (v & 0x2000)!=0 )
      temp[13]=1;
else
      temp[13]=0;
if( (v & 0x4000)!=0)
      temp[14]=1;
else
      temp[14]=0;
if( (v & 0x8000)!=0)
      temp[15]=1;
else
      temp[15]=0;
      TRAN_tb = 0;
      TL0             = 0x57;
       TH0             = 0x6e;
      TR0=1;
      flag_int=1;
      while(flag_int==1)
      {
       }
    TRAN_tb = 1;
while(i>=0)
{
if(temp[i]==0)
   TRAN0 = 0;
else
   TRAN1 = 0;
TL0             = 0x57;
 TH0             = 0x9e;
TR0=1;
flag_int=1;
```

```
while(flag_int==1)
{
 }
if(temp[i]==0)
   TRAN0 = 1;
else
   TRAN1 = 1;
TL0             = 0x57;
 TH0             = 0x9e;
TR0=1;
 flag_int=1;
while(flag_int==1)
{
}
i--;
    }
}

// Peripheral specific initialization functions,
// Called from the Init_Device() function
void ADC_Init()
{
    AMX0P        = 0x1E;
    AMX0N        = 0x1F;
    ADC0CN       = 0x80;
}
void Voltage_Reference_Init()
{
    REF0CN       = 0x05;
}
void Port_IO_Init()
{
    P1MDIN       = 0xDF;
    P0MDOUT      = 0x07;
    P1MDOUT      = 0x20;
    P1SKIP       = 0x20;
    XBR1         = 0x40;
}
void Oscillator_Init()
{
    int i = 0;
    OSCICN       = 0x83;
    CLKMUL       = 0x80;
```

```
    for (i = 0; i < 20; i++);       // Wait 5us for initialization
    CLKMUL       |= 0xC0;
    while ((CLKMUL & 0x20) == 0);
    CLKSEL       = 0x02;
}

void Interrupts_Init()
{
    EIE1         = 0x08;
    IE           = 0x82;
}
void Timer_Init()
{
     TMOD         = 0x01;
    CKCON        = 0x02;
    TL0          = 0x57;
    TH0          = 0x9E;
}
// Initialization function for device,
// Call Init_Device() from your main program
void Init_Device(void)
{
    Oscillator_Init();
    Port_IO_Init();
    Interrupts_Init();
    Timer_Init();
    ADC_Init();
    Voltage_Reference_Init();
}

void Timer0_ISR(void)    interrupt 1
{
     TF0=0;
    flag_int=0;
}
void ADC0_ISR(void) interrupt 10
{
    AD0INT=0;                          // 清 ADC 中断标志位
    ADC_FLAG = 0;
}
```

接收程序：

```
#include<c8051f120.h>
#include"HS12864.h"
void Timer_Init(void);
void Port_IO_Init(void);
void Oscillator_Init(void);
void Interrupts_Init(void);
void Init_Device(void);
void Disp(unsigned int result);
unsigned int Receievechar(void);
unsigned int RV,RI;
unsigned int result1;
sbit fore_lead=P1^2;
unsigned int flag_int0,result_A,result_B;

void main(void)
{
    SFRPGCN=1;
     WDTCN        = 0xDE;
     WDTCN        = 0xAD;    // Disable Watchdog timer
    Init_Device();
    SFRPAGE = CONFIG_PAGE;
    LCD_initial_operator();
     while(1)
     {
         result_A=Receievechar();
          result_A=~result_A;
         Disp(result_A);
         result_B=Receievechar();
          result_B=~result_B;
         Disp(result_B);
     }
}
void Disp(unsigned int result)
{
      unsigned char
bianhao_sz[8]={0,0,0,0,0,0,0,0},wendu_sz[3]={0,0,0},guang_sz[1]={0};
     unsigned int bianhao,flag_tempe;
      bianhao = result & 0xff00;
     bianhao = bianhao >> 8;
     bianhao = bianhao & 0x00ff;
    LCD_write_chars("探测节点信息显示",0x0,0x0);
     if((bianhao&0x80)==0)
```

```
    bianhao_sz[0] =48;
else
    bianhao_sz[0] =49;
if((bianhao&0x40)==0)
    bianhao_sz[1] =48;
else
    bianhao_sz[1] =49;
if((bianhao&0x20)==0)
    bianhao_sz[2] =48;
else
    bianhao_sz[2] =49;
if((bianhao&0x10)==0)
    bianhao_sz[3] =48;
else
    bianhao_sz[3] =49;
if((bianhao&0x08)==0)
    bianhao_sz[4] =48;
else
    bianhao_sz[4] =49;
if((bianhao&0x04)==0)
    bianhao_sz[5] =48;
else
    bianhao_sz[5] =49;
if((bianhao&0x02)==0)
    bianhao_sz[6] =48;
else
    bianhao_sz[6] =49;
if((bianhao&0x01)==0)
    bianhao_sz[7] =48;
else
    bianhao_sz[7] =49;
LCD_write_chars("编号: ",0x01,0x0);
LCD_write_chars(bianhao_sz,0x01,0x03);
 flag_tempe=result&0x00fe;
flag_tempe=flag_tempe>>1;
if(flag_tempe>=100)
   {
        wendu_sz[0]=49;
        flag_tempe=flag_tempe-100;
   }
else
    wendu_sz[0]=32;
wendu_sz[1]=48+flag_tempe/10;
```

```
    wendu_sz[2]=48+flag_tempe%10;
    LCD_write_chars("温度:        摄氏度",0x02,0x0);
    LCD_write_chars(wendu_sz,0x02,0x03);
    if((result&0x0001)==1)
        LCD_write_chars("光照情况: 有光    ",0x03,0x0);
    else
        LCD_write_chars("光照情况: 无光    ",0x03,0x0);
}
unsigned int Receievechar(void)
{
     unsigned int j,receive=0,flag_lead=1;
    SFRPAGE     = CONFIG_PAGE;
     while(flag_lead==1)
     {
         if(fore_lead==1)
         {
              for(j=0;j<5;j++)
            {
            }
             if(fore_lead==1)
                 flag_lead=0;
         }
     }
     EX0=1;
    EX1=1;
    RV = 0;
    RI = 0;
    while(RI<16)
    {
    }
     EX0=0;
    EX1=0;
     receive=RV;
     return(receive);
}
// Peripheral specific initialization functions,
// Called from the Init_Device() function
void Port_IO_Init()
{
    char SFRPAGE_SAVE = SFRPAGE;          // Save Current SFR page
    SFRPAGE     = CONFIG_PAGE;
    XBR0        = 0x07;
    XBR1        = 0x14;
```

```
    XBR2          = 0x40;
    SFRPAGE = SFRPAGE_SAVE;                    // Restore SFRPAGE
}
void Oscillator_Init()
{
    char SFRPAGE_SAVE = SFRPAGE;               // Save Current SFR page
    SFRPAGE     = CONFIG_PAGE;
    OSCICN      = 0x83;
    SFRPAGE = SFRPAGE_SAVE;                    // Restore SFRPAGE
}
void Interrupts_Init()
{
    char SFRPAGE_SAVE = SFRPAGE;               // Save Current SFR page
    IE              = 0x85;
    SFRPAGE = SFRPAGE_SAVE;                    // Restore SFRPAGE
}
void Timer_Init()
{
    char SFRPAGE_SAVE = SFRPAGE;               // Save Current SFR page
    SFRPAGE     = TIMER01_PAGE;
    TCON          = 0x05;
    SFRPAGE = SFRPAGE_SAVE;                    // Restore SFRPAGE
}
// Initialization function for device,
// Call Init_Device() from your main program
void Init_Device(void)
{
     Oscillator_Init();
    Port_IO_Init();
     Timer_Init();
    Interrupts_Init();
}
void INT0_ISR(void) interrupt 0
{
    RV =    (RV<<1);
     RI++;
}
void INT1_ISR(void) interrupt 2
{
    RV =    ((RV<<1)|0x01);
     RI++;
}
```

液晶显示程序：

```
#include "HS12864.H"
#include "c8051f120.h"

//液晶 HS12864-15 的接口引脚定义
#define HS12864DataBus P7 //数据端口选择
sbit HS12864_RS = P4^2;      //数据/指令选择
sbit HS12864_E = P4^0;       //使能控制
sbit HS12864_RW = P4^1;      //读/写控制



void HS12864_Delay(unsigned long int delx)
{
      unsigned long int i=0;
      while( i < delx )
      {
            i++;
      }
}

//*****************LCD 读指令操作函数
//读取忙标志和地址 a。
unsigned char LCD_read_instructor()
{
       unsigned char Value;
      HS12864_Delay(10);
      HS12864_RS = 0;
      HS12864_Delay(10);
      HS12864_RW = 1;
      HS12864_Delay(10);
      HS12864_E = 1;
      HS12864_Delay(10);
      Value = HS12864DataBus;
      HS12864_Delay(10);
      HS12864_E = 0;
      return(Value);
}


//*****************LCD 读显示数据操作函数
```

```
//读取显示 RAM 数据，其中第一次读为 Dummy Read，第二次读为正确数据。
unsigned char LCD_read_ramdata()
{
      unsigned char Value;
//第一次读
     HS12864_Delay(10);
     HS12864_RS = 1;
     HS12864_Delay(10);
     HS12864_RW = 1;
     HS12864_Delay(10);
     HS12864_E = 1;
     HS12864_Delay(10);
     Value = HS12864DataBus;
     HS12864_Delay(10);
     HS12864_E = 0;

//第二次读
      HS12864_Delay(10);
     HS12864_RS = 1;
     HS12864_Delay(10);
     HS12864_RW = 1;
     HS12864_Delay(10);
     HS12864_E = 1;
     HS12864_Delay(10);
     Value = HS12864DataBus;
     HS12864_Delay(10);
     HS12864_E = 0;
     return(Value);
}


//******************LCD 写指令操作函数
void LCD_write_instructor(unsigned char Value)
{
      HS12864_Delay(10);
 HS12864_RS = 0;
HS12864_Delay(10);
HS12864_RW = 0;
HS12864_Delay(10);
HS12864DataBus = Value;
HS12864_Delay(10);
HS12864_E = 1;
HS12864_Delay(10);
```

```
    HS12864_E = 0;
}

//*****************LCD 写数据操作函数
void LCD_write_data(unsigned char Value)
{
     HS12864_Delay(10);
     HS12864_RS = 1;
    HS12864_Delay(10);
    HS12864_RW = 0;
    HS12864_Delay(10);
    HS12864DataBus = Value;
    HS12864_Delay(10);
    HS12864_E = 1;
    HS12864_Delay(10);
    HS12864_E = 0;
}

//*****************LCD 忙状态检测函数
void LCD_busy_check(void)
{
     unsigned char Value;
    do
    {
        HS12864_Delay(10);
        Value = LCD_read_instructor() & 0x80;
    }
    while(Value);//检测 LCD 是否空闲
}

//*****************LCD 写命令函数
void LCD_write_command(unsigned char command_number,unsigned char
command_data)
{
    LCD_busy_check();//检测 LCD 是否空闲
    switch(command_number & 0x0f)
    {
        case 0x00:{LCD_write_instructor(0x01); break;}
        //指令 0，清除显示，不需要 command_data。
       case 0x01:{LCD_write_instructor(0x02); break;}
```

```
        //指令 1，地址归零，且不影响 DDRAM 中的内容，不需要 command_data。
        case 0x02:{LCD_write_instructor(command_data); break;}
        //指令 2，进入设定点，需要 command_data。 指令形式：0 0 0 0 1 ID S
        case 0x03:{LCD_write_instructor(command_data); break;}
        //指令 3，显示开关设置，需要 command_data。 指令形式：0 0 0 0 1 D C
B
        case 0x04:{LCD_write_instructor(command_data); break;}
        //指令 4，游标或显示移位控制，需要 command_data。 指令形式：0 0 0
1 SC RL X X
        case 0x05:{LCD_write_instructor(command_data); break;}
        //指令 5，功能设定，需要 command_data。 指令形式：0 0 1 DL X RE X
X
        case 0x06:{LCD_write_instructor(command_data); break;}
        //指令 6，设定 CGRAM 地址，需要 command_data。指令形式：0 1 A5 A4
A3 A2 A1 A0
        case 0x07:{LCD_write_instructor(command_data); break;}
        //指令 7，设定 DDRAM 地址，需要 command_data。指令形式：1 0 A5 A4
A3 A2 A1 A0
        case 0x08:{LCD_write_data(command_data); break;}
        //指令 8，写显示数据到 RAM，需要 command_data。 指令形式：D7 D6
D5 D4 D3 D2 D1 D0
        default: break;
    }
}


//*****************LCD 初始化操作函数
void LCD_initial_operator(void)
{
    HS12864_Delay(20000);//延时>40ms
    LCD_busy_check();//检测 LCD 是否空闲
    LCD_write_command(0x05,0x30);//设置功能设定控制字，设置为基本指令模
式
    HS12864_Delay(20000);//延时>100us
    LCD_busy_check();//检测 LCD 是否空闲
    LCD_write_command(0x05,0x30);//设置功能设定控制字，设置为 8 位并口、
基本指令模式
    HS12864_Delay(20000);//延时>37us
    LCD_busy_check();//检测 LCD 是否空闲
    LCD_write_command(0x03,0x0e);//设置显示设定控制字，设置整体显示开、
游标显示开模式
    HS12864_Delay(20000);//延时>100us
    LCD_busy_check();//检测 LCD 是否空闲
```

```
    LCD_write_command(0x00,0x00);//设置清除屏幕控制字，清除屏幕
    HS12864_Delay(20000);//延时>10ms
    LCD_busy_check();//检测 LCD 是否空闲
    LCD_write_command(0x02,0x06);//设置设定点控制字，设置游标右移模式
    HS12864_Delay(20000);//延时>100us
    LCD_busy_check();//检测 LCD 是否空闲
}

//*****************LCD 写一个字符(16 ×16 点)操作函数
void LCD_write_one_char(unsigned char zifu, unsigned char hang_addr,unsigned
char lie_addr)
{
     unsigned char addr;
    switch(hang_addr)
    {
          case 0:{addr = 0x80 + lie_addr;break;}
        case 1:{addr = 0x90 + lie_addr;break;}
        case 2:{addr = 0x88 + lie_addr;break;}
        case 3:{addr = 0x98 + lie_addr;break;}
        default: break;
    }
     LCD_write_command(0x07,addr);
    HS12864_Delay(20);
    LCD_write_command(0x08,zifu);
}

//*****************LCD 读一个字符(16 ×16 点)操作函数
unsigned char LCD_read_one_char(unsigned char hang_addr,unsigned char lie_addr)
{
     unsigned char addr,value;
    switch(hang_addr)
    {
          case 0:{addr = 0x80 + lie_addr;break;}
        case 1:{addr = 0x90 + lie_addr;break;}
        case 2:{addr = 0x88 + lie_addr;break;}
        case 3:{addr = 0x98 + lie_addr;break;}
        default: break;
}
 LCD_write_command(0x07,addr);
HS12864_Delay(20);
value = LCD_read_ramdata();
return(value);
```

```
}


//*****************LCD 写一串字符操作函数
void LCD_write_chars(unsigned char *zifu, unsigned char hang_addr, unsigned char
lie_addr)
{
      unsigned char addr;
switch(hang_addr)
{
       case 0:{addr = 0x80 + lie_addr;break;}
     case 1:{addr = 0x90 + lie_addr;break;}
     case 2:{addr = 0x88 + lie_addr;break;}
     case 3:{addr = 0x98 + lie_addr;break;}
     default: break;
}
 LCD_write_command(0x07,addr);
HS12864_Delay(20);
while(*zifu > 0)
{
      LCD_write_command(0x08,*zifu++);
}
}
```