**教 案**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 授课章节名 称 | 智能孵蛋控制系统 | | | | 授课教师 | 徐自远 | |
| 开课范围 | 校内 | |
| 授 课  时 间 | 2011年  12月1日 | 授课  班级 | 特色选修 | 授课  类型 | 理论实践一体化 | 课时 | 1 |
| 教学目标 | 1. 任务分析。 2. 硬件电路的设计与搭建 3. 软件编写 4. 任务实施 | | | | | | |
| 教学重点 | 将连续变化的模拟量转换为数字量进行处理。 | | | | | | |
| 教学难点 | 模数转换。 | | | | | | |
| 学情处理 | 1、双向互动、讨论  2、运用多媒体教学手段  3、理论实践一体化教学 | | | | | | |
| 选用教材 | 自编教材 | | | | | | |
| 教材内容处理说明 |  | | | | | | |
| 课外作业 |  | | | | | | |
| 教学后记 |  | | | | | | |

|  |  |  |
| --- | --- | --- |
| 教学  程序 | 教学内容 | 教学手段与方法 |
| 讲授新课 | **总任务书：**  按照要求完成智能孵蛋控制系统制作：  1．智能孵蛋控制系统描述及有关说明  为了保证能更好的孵化鸡蛋，保持鸡蛋的温度，智能孵蛋控制系统可以根据室内的温度来调节灯光的强度，从而达到控温的效果；也可自行调节灯光强度来调节灯光强度。  （1）显示：由8位数码管组成，实现DS18B20测得室温的显示和LED灯光的等级显示。  （2）独立键盘：SB1：实现“启动/停止”功能；SB2：实现自动和手动模式的切换。SB3：实现“调节”功能。  （3）DAC0832：用来控制温度表的孵化灯光的亮度（用LED0模拟孵化灯）。  （4）DS18B20：用来检测室温。  （5）LED0：使用LED0模拟孵化灯。其作用为对室内控制加热，尽量营造一个恒温孵化环境。  2．系统控制要求  系统上电，数码管不显示，按下“启动/停止”按键，数码管靠左两位显示室温XX，第三位显示‘C’，表示摄氏度；靠右显示亮度等级，格式为：“LvX”，开始对室温进行实时检测。灯光等级范围：0-9。  按下“自/手”动按键，可切换自动模式和手动模式。在自动模式下：当温度超过30℃时，灯光等级为0级；当温度超过或等于21度时，灯光等级为9级。当温度为22-30℃时，等级应根据温度的升高而降低，对应的变化范围为8-0。  在手动模式下，按下“调节”按键，可对灯光亮度进行0-9级之间的调节。  若再按下“启动/停止”按键，则数码管不显示，停止测温工作，灯灯熄灭（0级），处于待机状态。  **所需知识点介绍：**   1. 单片机应用系统中，经常需要把连续变化的模拟量转换为数字量进行处理，将模拟信号转换为数字信号的过程叫做模数转换，如图所示。实现模数转换的装置叫做“模数转换器”（简称ADC）。   **（1）**DS18B20单总线数字式温度传感器  DS18B20是Dallas半导体公司的数字化温度传感器。它是世界上第一片支持“一线总线”接口的温度传感器。YL-236实训平台中包含的DS18B20传感器为TO92封装。其实物外形与封装图如所示。  **（2）DS18B20 寄存器与EEPROM解读**  EEPROM用于存放长期需要保存的数据，上下限温度报警值和校验数据，DS18B20共3个字节的EEPROM，并在RAM都存在镜像如图所示，以方便用户操作。此EEPROM可以通过“Copy Scratchpad”指令[48H]，把RAM中对应的数据写入EEPROM中。当系统复位后，DS18B20会自动通过“Recall EEPROM”指令[B8H]把EEPROM中的数据复制到对应RAM中去。  【注】：温度值高位的S为符号位，当S=1时表示当前温度为负温度，此时温度数值以补码形式保存在寄存器中。所以如果检测温度为负温度，得先将读取到的数值由补码转换为原码再计算其对应十进制值（或直接放入带符号变量中）。 | 带入课题  做中教  做中学 |

|  |  |  |
| --- | --- | --- |
| 教学  程序 | 教学内容 | 教学手段与方法 |
| 讲授新课 | **（3）DS18B20控制指令介绍**  控制指令名称 命令字 功能说明：  Read ROM [33H] 读ROM指令：这个命令允许单片机读到DS18B20的64位ROM。  Match ROM [55H] 指定匹配芯片指令：  这个指令后面紧跟着由单片机发出了64位序列号，当总线上有多只 DS18B20时，只有与控制发出的序列号相同的芯片才可以做出反应，其它芯片将等待下一次复位。这条指令适应单芯片和多芯片挂接。  Skip ROM [CCH] 跳过ROM编码指令：  这条指令使芯片不对ROM编码做出反应，在单总线的情况之下，为了节省时间则可以选用此指令。如果在多芯片挂接时使用此指令将会出现数据冲突，导致错误出现。  Search ROM [F0H] 搜索芯片指令：在芯片初始化后，搜索指令允许总线上挂接多芯片时用排除法识别所有器件的64位ROM。  Alarm Search [ECH] 报警芯片搜索指令：  在多芯片挂接的情况下，报警芯片搜索指令只对附合温度高于TH或小于TL报警条件的芯片做出反应。只要芯片不掉电，报警状态将被保持，直到再一次测得温度不达到报警条件为止。  Write Scratchpad [4EH] 向RAM中写数据指令：  这是向RAM中写入数据的指令，随后写入的两个字节的数据将会被存到RAM的第3字节（高温报警TH）和第4字节（低温报警TL）。再次写入的一个字节存入第5字节中（配置寄存器CR）。写入过程中可以用复位信号中止写入。  Read Scratchpad [BEH] 从RAM中读数据指令：  此指令将从RAM中读数据，从RAM的第1字节开始，一直可以读到第9字节，完成整个RAM数据的读出。芯片允许在读过程中用复位信号中止读取，即可以不读后面不需要的字节以减少读取时间。  Copy Scratchpad [48H] 将RAM数据复制到EEPROM 中指令：  此指令将RAM中的数据存入EEPROM中，以使数据掉电不丢失。此后由于芯片忙于EEPROM储存处理，当单片机发一个读时间隙时，总线上输出“0”，当储存工作完成时，总线将输出“1”。在寄生工作方式时必须在发出此指令后立刻超用强上拉并至少保持10MS，来维持芯片工作。  Convert T [44H] 开始温度转换指令：  收到此指令后芯片将进行一次温度转换，将转换的温度值放入RAM的第1、2地址。此后由于芯片忙于温度转换处理，当单片机发一个读时间隙时，总线上输出“0”，当储存工作完成时，总线将输出“1”。在寄生工作方式时必须在发出此指令后立刻超用强上拉并至少保持500mS（12位精度时），来维持芯片工作。  Recall EEPROM [B8H] 将EEPROM 中的报警值复制到RAM指令。  Read Power Supply [B4H] 检测芯片电源状态指令：此指令发出后发出读时间隙，芯片会返回它的电源状态字，“0”为寄生电源状态，“1”为外部电源状态。   1. **DS18B20控制程序编写---复位与应答时序**   每一次通信之前必须对DS18B20进行复位，复位的时间、等待时间、回应时间应严格按时序编程。复位及应答时序如图所示。可以看出，先由主机拉低总线电平，并且保持至少480μs，此信号为主机向DS18B20发送复位信号。而后主机电平恢复高电平，等待15~60μs。这时如果DS18B20正常工作，它会主动把总线电平拉低60~240μs，这个信号就是DS18B20向主机发送的应答信号。   1. **DS18B20控制程序编写---写数据时序**   写时间间隙分为写“0”和写“1”，其时序图如图所示。在写数据时前15μs总线需要是被单片机拉置低电平，而后则将是DS18B20芯片对总线数据的采样时间，采样时间在15~60μs，采样时间内如果单片机将总线拉高则表示写“1”，如果单片机将总线拉低则表示写“0”。每一位的发送都应该有一个至少15μs的低电平起始位，随后的数据“0”或“1”应该在45μs内完成。整个位的发送时间应该保持在60~120μs，否则不能保证通信的正常。【注意】：在通信时是以8位“0”或“1”为一个字节，字节的写是从底位开始的，即从bit0到bit7。   1. **DS18B20控制程序编写---读数据时序**     单片机读取DS18B20的参数时其时序应该更加的精确才行，读数据时必须先由单片机产生至少1μs的低电平，表示读时间的起始。总线被释放15μs后DS18B20会发送内部数据位到总线上，单片机读取总线为高电平表示读出“1”，如果总线为低电平则表示读出数据“0”。此数据会保持45μs，所以单片机读取数据必须在这45μs之内。每一位的读取之前必须由单片机发送一个至少1μs的低电平作为读取起始信号。如图10- 9所示，必须在读间隙开始的 15μs内读取数据位才可以保证通信的正确。【注意】：在通信时是以8位“0”或“1”为一个字节，字节的读是从高位开始的，即从bit7到bit0。  //DS18B20写数据函数如下所示：  sbit DQ = P2^1; //映射温度传送数据IO口为P2.1  void WriteOneChar(unsigned char dat)  {  unsigned char i=0;  for (i=8; i>0; i--)  {  DQ=0; //拉低总线，产生写信号  delay\_μs(15); //延时15μs  DQ=dat&0x01; //把数据最低位输出给总线  delay\_μs(60); //延时60μs  DQ = 1; //释放总线，等待总线恢复  dat>>=1; //准备下一位数据的传送  }  }   1. **DS18B20温度采样范例程序**   有了以上的理论基础，我们就可以试着利用DS18B20来做最简单的检测温度任务。首先构建起DS18B20硬件电路原理图如图10-10所示：  此硬件连接下DS18B20简单检测温度流程如图所示：    //DS18B20温度采样范例程序  sbit DQ = P2^1; //映射温度传送数据IO口为P2.1  /\*\*\*\*\*\*\*\*\*\*\*读取ds18b20当前温度\*\*\*\*\*\*\*\*\*\*\*\*/  void ReadTemp(void)  {  unsigned char a=0;  unsigned char b=0;  if(!RST\_DS18B20()) //判断是否初始化成功  {  WriteOneChar(0xCC); //跳过读序号列号的操作  WriteOneChar(0x44); //启动温度转换  delay\_us(41667); //延时1秒，等待转换完成  if(!RST\_DS18B20()) //判断是否初始化成功  {  WriteOneChar(0xCC); //跳过读序号列号的操作  WriteOneChar(0xBE); //读取温度寄存器等（共可读9个寄存器） 前两个就是温度  delay\_us(41667); //延时1秒，准备读取数据  a=ReadOneChar(); //读取温度值低位  b=ReadOneChar(); //读取温度值高位  temp\_value=b<<4;  temp\_value+=(a&0xf0)>>4;  }  }  }   1. **DAC0832介绍及应用——典型应用电路** |  |

|  |  |  |
| --- | --- | --- |
| 教学  程序 | 教学内容 | 教学手段与方法 |
|  | **硬件电路的设计与搭建**  智能孵蛋控制系统模块接线图(YL-236单片机实训装置)  **软件编程**    //智能孵蛋控制系统参考程序  #include "reg52.h"  #define ON 1  #define OFF 0  bit set\_mark=0; //自动和手动切换标志位 1：自动 0：手动  bit RUN\_or\_STOP=OFF; //"启动/停止"标志位，ON：启动 ，OFF：停止  unsigned char LV=0; //LED灯亮度等级参数 默认为0级  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DAC0832\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char xdata CS\_DAC \_at\_ 0xdfff; //(P2^5)  #define show\_LED CS\_DAC=(9-LV)\*28.33  #define extinguish\_LED CS\_DAC=255  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数码管\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char xdata DM \_at\_ 0x7fff; //断码（P2^7）  unsigned char xdata PX \_at\_ 0xbfff; //片选（P2^6）  unsigned char code M7G[]= //数码管字模  {  0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90, //0~9  0xff, 0xc6,0xc7, 0xe3, //灭，C，L，v  };  unsigned char str[8]= //数码管缓存  {  10,10,10,10,10,10,10,10  };  void Display()  {  static unsigned char L;  PX=255; //消隐  DM=M7G[str[L]]; //段码  PX=~(0x80>>L); //片选  L++;  L&=7;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS18B20\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  sbit DQ = P3^5; //映射温度传送数据IO口为P2.1  unsigned int temp\_value; //温度值  #define show\_temp\_value str[0]=temp\_value/10%10,str[1]=temp\_value%10,str[2]=11,str[5]=12,str[6]=13,str[7]=LV%10 //显示tempC和亮度等级 格式为00C Lv0  #define extinguish\_temp\_value str[0]=str[1]=str[2]=str[5]=str[6]=str[7]=10  //数码管不显示  /\*\*\*\*\*\*\*ds18b20子程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*ds18b20延迟子函数（晶振12MHz ）\*\*\*\*\*\*\*/  void delay\_us(unsigned int i) //1个单位为24us  {  while (--i);  }  /\*\*\*\*ds18b20初始化函数\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char RST\_DS18B20(void)  {  unsigned char x=1; //默认设置为初始化失败  DQ = 1; //DQ复位  delay\_us(1); //稍做延时24μs  DQ = 0; //单片机将DQ拉低  delay\_us(30); //精确延时 480μs-960μs 我们取中间值720μs  DQ = 1; //释放总线  delay\_us(2); //延时15~60μs  delay\_us(5); //DS18B20应答信号60~240μs，取值120  x=DQ; //稍做延时后 如果x=0则初始化成功 x=1则初始化失败  return(x);  }  /\*\*\*\*\*\*\*\*\*\*\*ds18b20读一个字节\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char ReadOneChar(void)  {  unsigned char i=0;  unsigned char dat = 0;  for (i=8;i>0;i--)  {  Dat>>=1; //数据→移1位  DQ = 0; //拉低总线产生读信号号  DQ = 1; //释放总线，准备读数据。  if(DQ) dat|=0x80; //如果DQ为1，dat最高位放入数据。  delay\_us(2); //延时48μs，DS18B20数据会保持45μs  }  return(dat);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*ds18b20写一个字节\*\*\*\*\*\*\*\*\*\*\*\*\*/  void WriteOneChar(unsigned char dat)  {  unsigned char i=0;  for (i=8; i>0; i--)  {  DQ=0; //拉低总线，产生写信号  DQ=dat&0x01; //把数据最低位输出给总线  delay\_us(1); //延时24μs  DQ = 1; //释放总线，等待总线恢复  dat>>=1; //准备下一位数据的传送  }  }  /\*\*\*\*\*\*\*\*\*\*\*读取ds18b20当前温度\*\*\*\*\*\*\*\*\*\*\*\*/  void ReadTemp(void)  {  unsigned char a=0;  unsigned char b=0;  if(!RST\_DS18B20()) //判断是否初始化成功  {  WriteOneChar(0xCC); //跳过读序号列号的操作  WriteOneChar(0x44); //启动温度转换  delay\_us(41667); //延时1秒，等待转换完成  if(!RST\_DS18B20()) //判断是否初始化成功  {  WriteOneChar(0xCC); //跳过读序号列号的操作  WriteOneChar(0xBE); //读取温度寄存器等（共可读9个寄存器） 前两个就是温度  delay\_us(41667); //延时1秒，准备读取数据  a=ReadOneChar(); //读取温度值低位  b=ReadOneChar(); //读取温度值高位  temp\_value=b<<4;  temp\_value+=(a&0xf0)>>4;  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*独立按键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  sbit SB1 = P2^2 ;  sbit SB2 = P2^3 ;  sbit SB3 = P2^4 ;  void KEY() //按键  {  static unsigned int ms;  if(SB1==0||SB2==0||SB3==0) //判断按键有无按下  {  if(++ms==100) //防抖动  {  if(SB1==0)RUN\_or\_STOP=!RUN\_or\_STOP;//启动或停止  if(SB2==0&&RUN\_or\_STOP)set\_mark=!set\_mark;//自动和手动切换  if(SB3==0&&RUN\_or\_STOP&&set\_mark==0)LV++,LV%=10;//等级调节，0-9  if(RUN\_or\_STOP)show\_LED,show\_temp\_value;//当系统启动时，显示数码管和LED  if(!RUN\_or\_STOP)extinguish\_LED,extinguish\_temp\_value;  //当系统停止时，关闭数码管和LED  }  }  else ms=0; //计时清零  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void auto\_mode()//自动模式  {  if(RUN\_or\_STOP)  {  if(temp\_value>30)LV=0; //当温度超过30℃时，等级置为0级  else if(temp\_value<=20)LV=9; //当温度低于或等于20℃时，等级置为9级  else LV=30-temp\_value; //运算  }  }  void TIME0\_ROUTING() interrupt 1  {  TH0=0xfc; //11.0592MHZ  TL0=0x66; //1ms定时器  KEY(); //按键  Display(); //数码管显示  }  void INIT\_TIME0() //定时器0初始化  {  TMOD = 0x01; //设置模式  TH0=0xfc; //11.0592MHZ  TL0=0x66; //1ms定时器  ET0=1; //允许中断标志位  TR0=1; //开始计时标志位  EA=1; //中断总开关  }  void main()  {  INIT\_TIME0();  while (1)  {  f(RUN\_or\_STOP==ON)show\_temp\_value,show\_LED,ReadTemp();  //当启停标志位打开时 开始检测温度  if(set\_mark)auto\_mode(); //自动模式  }  **软件说明**  本程序主要通过DS18B20来读取温度并由按键来控制灯光亮度来达到恒温效果。其中定义了一个LV变量：用来控制LED灯光的亮度等级。位变量RUN\_or\_STOP：用来标志温度的采样启停。位变量set\_mark：用来控制自动和手动的切换。KEY()：用来启停温度采样、调节LED灯光等级亮度和自动手动的切换。Init\_DS18B20()：初始化DS18B20。ReadOneChar()：DS1820读一个字节子程序。WriteOneChar()：DS1820写一个字节子程序。ReadTemp()：读取DS18B20当前温度。auto\_mode()：自动模式下的等级调节。  **项目实施**  1．硬件电路连接：按照硬件电路接线图，选择所需的模块并进行布局，然后将电源模块、主机模块和数码管，LED显示模块、独立键盘和DAC0832等模块用导线进行连接。  2．打开keil软件，通过菜单新建立一个项目文件FDJ，然后再建一个文件名为FDJ.C的源程序文件并添加到项目中，将上面的参考程序输入并保存。  3．对源程序进行编译和链接，产生目标代码并烧录到单片机中。  4．接通电源，让单片机运行，通过对应按键操作检测系统工作是否正常。  5．进行扎线，整理。 |  |