

# MLX90614 系列红外测温模块的原理及应用

南京航空航天大学 曾德志

**摘要：** MLX90614 系列模块是一组通用的红外测温模块。在出厂前该模块已进行校验及线性化，具有非接触、体积小、精度高，成本低等优点。被测目标温度和环境温度能通过单通道输出，并有两种输出接口，适合于汽车空调、室内暖气、家用电器、手持设备以及医疗设备应用等。本文以MLX90614AAA为例介绍其原理和应用。

## 1 引言

一般来说，测温方式可分为接触式和非接触式，接触式测温只能测量被测物体与测温传感器达到热平衡后的温度，所以响应时间长，且极易受环境温度的影响；而红外测温是根据被测物体的红外辐射能量来确定物体的温度，不与被测物体接触，具有影响动被测物体温度分布场，温度分辨率高、响应速度快、测温范围广、不受测温上限的限制、稳定性好等特点，近年来在家庭自动化、汽车电子、航空和军事上得到越来越广泛的应用。

## 2 测温原理概述

物体红外辐射能量的大小和波长的分布与其表面温度关系密切。因此，通过对物体自身红外辐射的测量，能准确地确定其表面温度，红外测温就是利用这一原理测量温度的。红外测温器由光学系统、光电探测器、信号放大器和信号处理及输出等部分组成。光学系统汇聚其视场内的目标红外辐射能量，视场的大小由测温仪的光学零件及其位置确定。红外能量聚焦在光电探测器上并转变为相应的电信号。该信号经过放大器和信号处理电路，并按照仪器内的算法和目标发射率校正后转变为被测目标的温度值。

PWM 的全称是 Pulse Width Modulation（脉冲宽度调制）即通过调节脉冲的周期、宽度，以达到变压、变频的目的，数字式脉宽调制方式中，数字是控制信号，通过改变高低电平数的比值达到改变占空比的目的，PWM 控制电路在开关稳压电源、不间断电源（UPS）以及直流电机调速，交流电机变频调速等控制电路中有着广泛应用。

SMBus(System Management Bus,)是 1995 年由 intel 公司提出的一种高效同步串行总线，SMBus 只有两根信号线:双向数据线和时钟信号线，容许 CPU 与各种外围接口器件以串行方式进行通信、交换信息，即可以提高传输速度也可以减小器件的资源占用，另外即使在没有 SMBus 接口的单片机上也可利用软件进行模拟。

Melexis 公司生产的 MLX90614 系列测温模块是应用非常方便的红外测温装置，其所有的模块都在出厂前进行了校验，并且可以直接输出线性或准线性信号，具有很好的互换性，免去了复杂的校正过程。

该模块以 81101 热电元件作为红外感应部分。输出是被测物体温度（TO）与传感器自身温度（Ta）共同作用的结果，理想情况下热电元件的输出电压为：

$$V_{ir} = A(T_o^4 - T_a^4)$$

其中温度单位均为 Kelvin，A 为元件的灵敏度常数。

目标温度和环境温度由 81101 内置的热电偶测定测量，从 81101 中输出的两路温度信号分别经内部 MLX90302 器件上高性能、低噪声的斩波稳态放大器放大再经一个 17-bit 的模数转换器（ADC）和强大的数字信号处理(DSP)单元后输出。

该系列模块的温度解析度可达 0.01℃，体积小巧，被测目标和环境温度能通过单通道（由 MLX90302 内的状态机控制）输出，有两种输出方式：PWM 输出、可编程 SMBus 输出，适于多种应用环境，下面以 MLX90614 为例，重点介绍其特性和使用方法。

通过 SMBus 编程可以更改模块 EEPROM 内的预设值并按照应用要求进行配置，并可以读出 EEPROM 内的配置信息；还可以读出模块 RAM 内温度等数据。

MLX90614 有适用于 3 伏和 5 伏电源操作的两种类型。由于 3 伏型其小于 2 毫安的电流消耗，它非常适用于手提装置和电池动力装置。为此，传感器也具有一个节能“休眠”模式，此时电流消耗可低于 2 毫安。对于 12 伏汽车电池直接供电的情况，5 伏型包含的电子部件可与几个外部元件一起在较高电压下运行。

3 MLX90614 简介

3.1 MLX90614 管脚

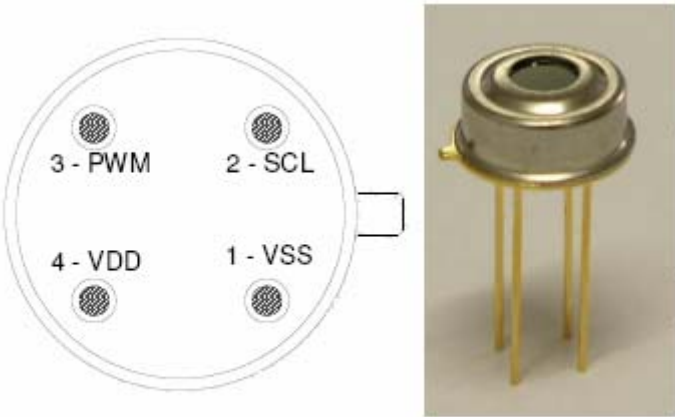


图1 MLX90614的管脚分布

表 1 MLX90614 的管脚功能

名称	功能描述
VSS	电源地，金属外壳和该管脚相连
SCL/Vz	SMBus 接口的时钟信号，或 8-16V 电源供电时接三极管基极
PWM/SDA	PWM 或 SMBus 接口的数据信号，通常模式下从该管脚通过 PWM 输出物体温度
VDD	电源

3.2 MLX90614 存储器

3.2.1 EEPROM

只有某些存储单元用户能够写入，但是可以读出全部存储单元。

EEPROM (32X16)		
Name	Address	Write acces
To <sub>max</sub>	000h	Yes
To <sub>min</sub>	001h	Yes
PWMCTRL	002h	Yes
Ta range	003h	Yes
Ke	004h	Yes
Config Register1	005h	Yes
Melexis reserved	006h	No
...	...	...
Melexis reserved	01Ah	No
ID number	01Bh	No
ID number	01Ch	No
ID number	01Dh	No
ID number	01Eh	No
ID number	01Fh	No

MLX90614 的 EEPROM 有 32 个 16 位存储单元，其中存储单元 Tomax,Tomin,Ta 分别是用户物体温度上下限和环境温度范围，PWMCTRL 是 PWM 配置寄存器。

3.2.2 RAM

用户不能向 RAM 写入数据，但是可以读一些存储单元。

RAM (32x17)		
Name	Address	Read access
Melexis reserved	000h	Yes
...	...	...
Melexis reserved	005h	Yes
T <sub>A</sub>	006h	Yes
T <sub>OBJ1</sub>	007h	Yes
T <sub>OBJ2</sub>	008h	Yes
Melexis reserved	009h	Yes
...	...	...
Melexis reserved	01Fh	Yes
StackOverflow	020h	

MLX90614 的RAM有 32 个 17 位存储单元，其中T<sub>A</sub>,T<sub>OBJ1</sub>,T<sub>OBJ2</sub>是环境温度和物体温度，在SMBus方式下，可以从这几个存储单元读出环境和被测物体的温度。

4 应用设计

4.1 MLX90614 测温特性

4.1.1 MLX90614 的 SMBus 协议

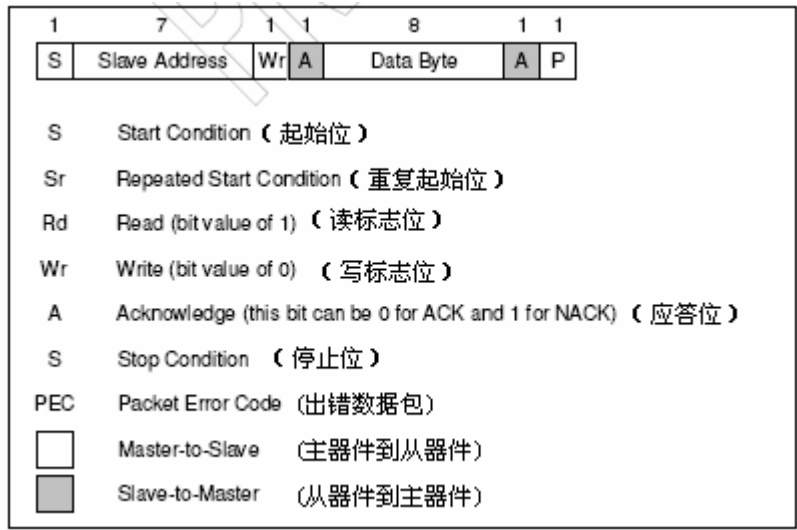
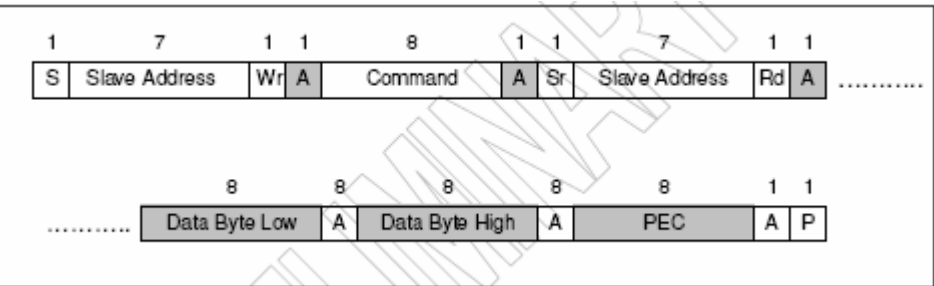
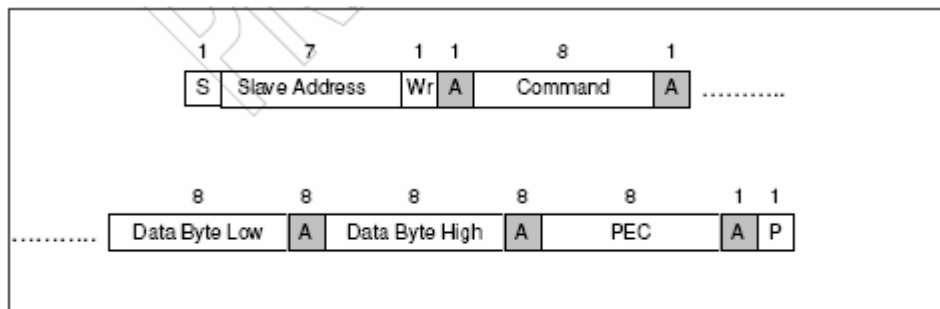


图 2 SMBus 的数据包组成

4.1.2 读器件（命令决定是读 RAM 或 EEPROM）数据格式



### 4.1.3 写器件（命令决定是写 RAM 或 EEPROM）数据格式



### 4.1.4 数据传输时序

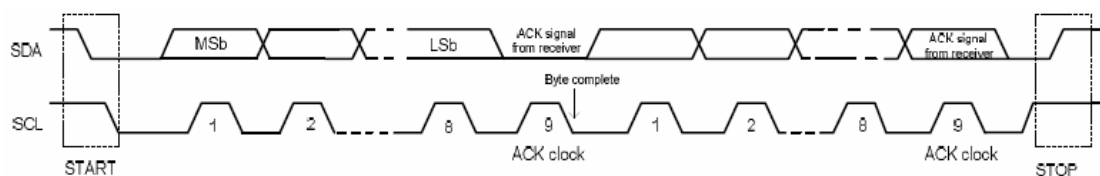


图 3 MLX90614 的数据传输时序

PWM/SDA 上的数据在 SCL 变为低电平 300n 后即可改变，数据在 SCL 的上升沿被捕获。16 位数据分两次传输，每次传一个字节。每个字节都是按照高位(MSB)在前，低位(LSB)在后的格式传输，两个字节中间的第九个时钟是应答时钟。

### 4.2 单片机接口电路

MLX90614AAA 与单片机连接的硬件电路如图 4 所示。SCL、PWM/SDA 管脚直接连接 MCU 的普通 I/O 口即可，由于 MLX90614AAA 的输入输出接口是漏级开路（OD）结构，需要加上拉电阻。多个 MLX90614 可以用于一个系统中，每个 MLX90614 对应一个不同地址，通过地址的不同而访问不同的 MLX90614，最多可以达到 127 个。

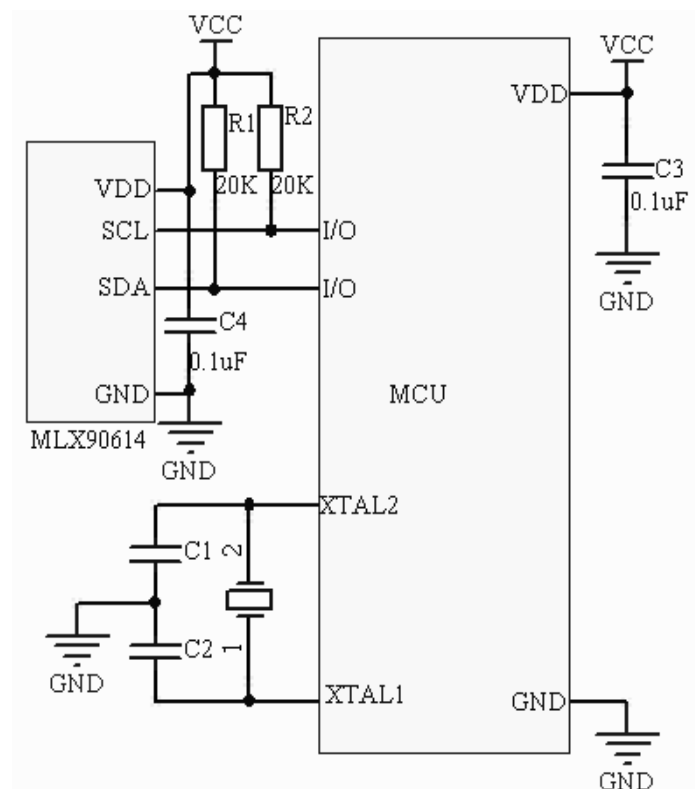
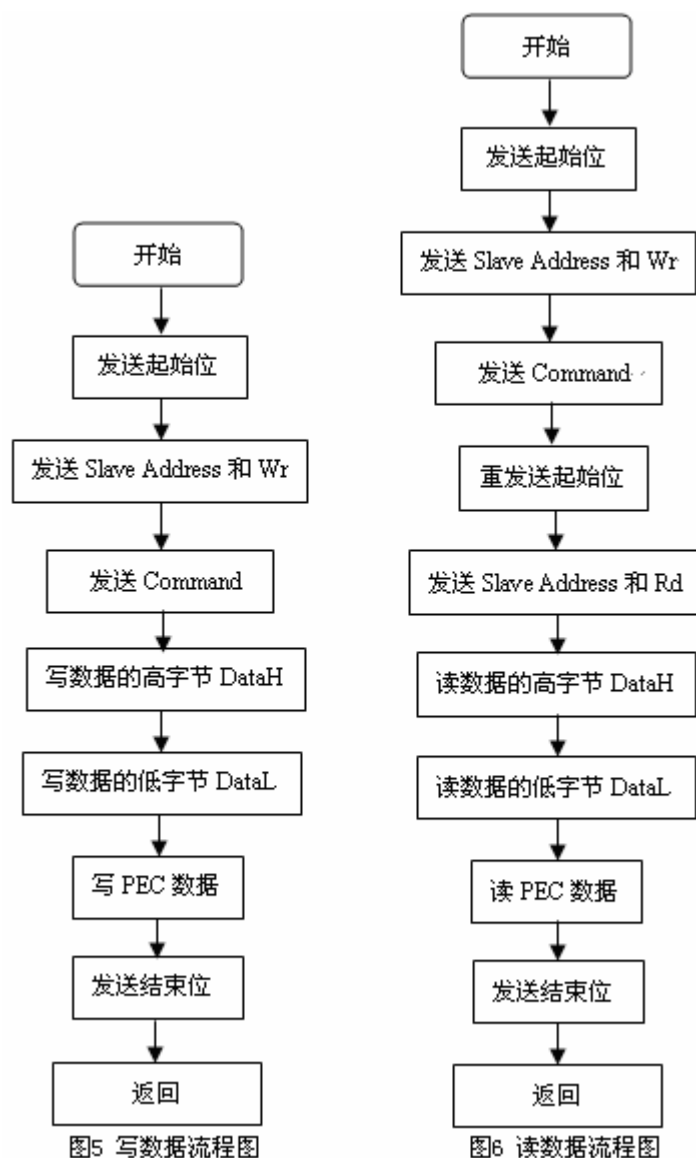


图 4 MLX90614AAA 与单片机连接电路

### 4.3 软件流程图

多个 MLX90614 可以用于一个系统中,通过地址不同区分器件,器件默认的地址为 5AH,因此在多 MLX90614 系统中,需要给每个 MLX90614 分配一个不同的地址,在只有一个 MLX90614 的系统中, MLX90614 识别地址 00h,即在单个 MLX90614 系统中,可以使用该地址访问它。



发送和接收数据是以字节为单位进行的,程序流程如图 7、图 8 所示。每次发送一个字节(按位发送,发送 8 个位就是一个字节),然后就判断对方是否有应答,如果有应答,就接着发送下一个字节;如果没有应答,多次重发该字节,直到有应答,就接着发送下一个字节,如果多次重发后,仍然没有应答,就结束。接收数据时,每次接收一个字节(按位接收,接收 8 个位就是一个字节),然后向对方发送一个应答信号,然后就可以继续接收下一个字节。

从 MLX90614 种读出的数据是 16 位的,由高 8 位(DataH)和低 8 位(DataL)两部分组成,其中 RAM 地址 07H 单元存储的是  $T_{OBJ}$  数据,数据范围从 0x27AD 到 0x7FFF,表示的温度范围是 -70.01℃ 到 +382.19℃。

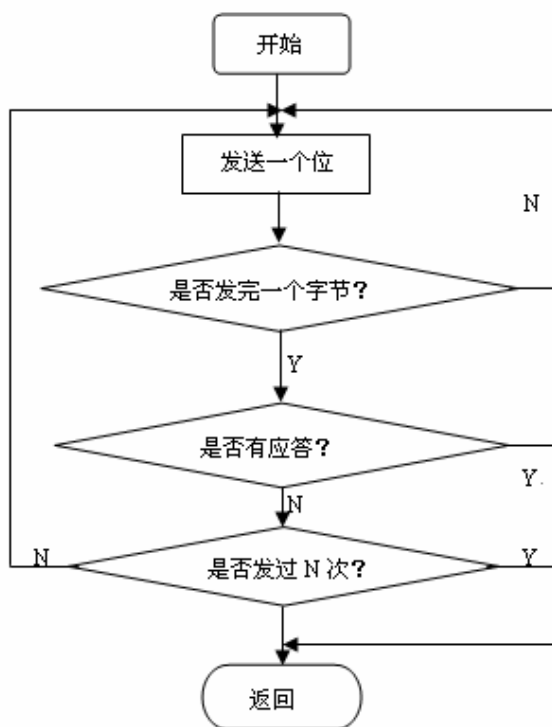


图 7 发送字节流程图

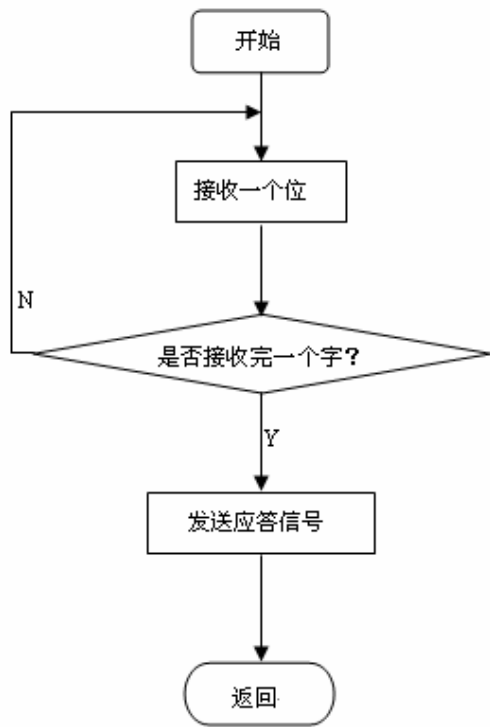


图 8 接收字节流程图

从 MLX90614 中读出的数据(DataH:DataL)换算为温度数据(T,单位为℃)如下所示:

$$T = (\text{DataH:DataL}) * 0.02 - 273.15 \quad \text{式 (1)}$$

例如: DataH:DataL=0x27AD, 代入式 (1) 中  $T = -70.01^{\circ}\text{C}$

C 语言程序清单:

单片机 89S52、MLX90614、LCD1602

晶振: 12M

```
#include "at89x52.h"
```

```
#include "intrins.h"
```

```
/**/
```

```
#define uint unsigned int
```

```
#define uchar unsigned char
```

```
#define Nack_counter 10
```

```
/**/端口定义
```

```
//LCD 控制线接口
```

```
uchar flag1;
```

```
sbit RS=P2^7;
```

```
sbit RW=P2^6;
```

```
sbit LCDE=P2^5;
```

```
//mlx90614 端口定义
```

```

sbit  SCL=P2^3;//时钟线
sbit  SDA=P2^2;//数据线
//*****数据定义*****

bdata uchar flag;//可位寻址数据
sbit bit_out=flag^7;
sbit bit_in=flag^0;
uchar DataH,DataL,Pecreg;
//*****函数声明*****

void  start_bit();           //MLX90614 发起始位子程序
void  stop_bit();           //MLX90614 发结束位子程序
uchar rx_byte(void);        //MLX90614 接收字节子程序
void  send_bit(void);       //MLX90614 发送位子程序
void  tx_byte(uchar dat_byte); //MLX90614 接收字节子程序
void  receive_bit(void);    //MLX90614 接收位子程序
void  delay(uint N);        //延时程序
uint  memread(void);        //读温度数据
void  init1602(void);       //LCD 初始化子程序
void  chk_busy_flg(void);   //LCD 判断忙子程序
void  dis_cmd_wrt(uchar cmd); //LCD 写命令子程序
void  dis_dat_wrt(uchar dat); //LCD 写数据子程序
void  display(uint Tem);    //显示子程序
//*****主函数*****

void main()
{
    uint Tem;
    //函数部分
    SCL=1;SDA=1;_nop_();
    _nop_();_nop_();_nop_();
    SCL=0;
    delay(1000);
    SCL=1;
    init1602();
    while(1)
    {
        Tem=memread();
        display(Tem);
        delay(20);
    }
}
//*****输入转换并显示*****

void display(uint Tem)
{
    uint T,a,b;
    T=Tem*2;

```

```

dis_cmd_wrt(0x01);//清屏
if(T>=27315)
{
    T=T-27315;
    a=T/100;
    b=T-a*100;
//-----
    if(a>=100)
    {
        dis_dat_wrt(0x30+a/100);
        a=a%100;
        dis_dat_wrt(0x30+a/10);
        a=a%10;
        dis_dat_wrt(0x30+a);
    }
    else if(a>=10)
    {
        dis_dat_wrt(0x30+a/10);
        a=a%10;
        dis_dat_wrt(0x30+a);
    }
    else
    {
        dis_dat_wrt(0x30+a);
    }
    dis_dat_wrt(0x2e);//显示点
//-----
    if(b>=10)
    {
        dis_dat_wrt(0x30+b/10);
//    b=b%10;
//    dis_dat_wrt(0x30+b);
    }
    else
    {
        dis_dat_wrt(0x30);
//    dis_dat_wrt(0x30+b);
    }
}
//=====
else
{
    T=27315-T;
    a=T/100;

```



```

b=T-a*100;
dis_dat_wrt(0x2d);
//-----
if(a>=10)
{
    dis_dat_wrt(0x30+a/10);
    a=a%10;
    dis_dat_wrt(0x30+a);
}
else
{
    dis_dat_wrt(0x30+a);
}
dis_dat_wrt(0x2e);//显示点
//-----
if(b>=10)
{
    dis_dat_wrt(0x30+b/10);
    b=b%10;
    dis_dat_wrt(0x30+b);
}
else
{
    dis_dat_wrt(0x30);
    dis_dat_wrt(0x30+b);
}
}
}
//*****
void    start_bit(void)
{
    SDA=1;
    _nop();_nop();_nop();_nop();_nop();
    SCL=1;
    _nop();_nop();_nop();_nop();_nop();
    SDA=0;
    _nop();_nop();_nop();_nop();_nop();
    SCL=0;
    _nop();_nop();_nop();_nop();_nop();

}
//-----
void    stop_bit(void)
{

```

```

    SCL=0;
    _nop_();_nop_();_nop_();_nop_();_nop_();
    SDA=0;
    _nop_();_nop_();_nop_();_nop_();_nop_();
    SCL=1;
    _nop_();_nop_();_nop_();_nop_();_nop_();
    SDA=1;
}
//-----发送一个字节-----
void tx_byte(uchar dat_byte)
{
    char i,n,dat;
    n=Nack_counter;
TX_again:
    dat=dat_byte;
    for(i=0;i<8;i++)
    {
        if(dat&0x80)
            bit_out=1;
        else
            bit_out=0;
        send_bit();
        dat=dat<<1;
    }
    receive_bit();
    if(bit_in==1)
    {
        stop_bit();
        if(n!=0)
        {n--;goto Repeat;}
        else
            goto exit;
    }
    else
        goto exit;
Repeat:
    start_bit();
    goto TX_again;
exit: ;
}
//-----发送一个位-----
void send_bit(void)
{
    if(bit_out==0)

```

```

        SDA=0;
else
        SDA=1;
    _nop_();
    SCL=1;
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    SCL=0;
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
}
//-----接收一个字节-----
uchar rx_byte(void)
{
    uchar i,dat;
    dat=0;
    for(i=0;i<8;i++)
    {
        dat=dat<<1;
        receive_bit();
        if(bit_in==1)
            dat=dat+1;
    }
    send_bit();
    return dat;
}
//-----接收一个位-----
void receive_bit(void)
{
    SDA=1;bit_in=1;
    SCL=1;
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    bit_in=SDA;
    _nop_();
    SCL=0;
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
}
//-----延时-----
void delay(uint N)
{
    uint i;
    for(i=0;i<N;i++)

```

```

        _nop_();
    }
    //-----
uint memread(void)
{
    start_bit();
    tx_byte(0x00); //Send SlaveAddress
    tx_byte(0x07); //Send Command
    //-----
    start_bit();
    tx_byte(0x01);
    bit_out=0;
    DataL=rx_byte();
    bit_out=0;
    DataH=rx_byte();
    bit_out=1;
    Pecreg=rx_byte();
    stop_bit();
    return(DataH*256+DataL);
}
//*****LCD 显示子函数*****
void init1602(void) //初始化 LCD
{
    dis_cmd_wrt(0x01);
    dis_cmd_wrt(0x0c);
    dis_cmd_wrt(0x06);
    dis_cmd_wrt(0x38);
}

void chk_busy_flg(void) //LCD 忙标志判断
{
    flag1=0x80;
    while(flag1&0x80)
    {
        P0=0xff;
        RS=0;
        RW=1;
        LCDE=1;
        flag1=P0;
        LCDE=0;
    }
}

void dis_cmd_wrt(uchar cmd) //写命令子函数

```

```
{  
    chk_busy_flg();  
    P0=cmd;  
    RS=0;  
    RW=0;  
    LCDE=1;  
    LCDE=0;  
}
```

```
void dis_dat_wrt(uchar dat) //写数据子函数
```

```
{  
    chk_busy_flg();  
    if(flag1==16)  
    {  
        P0=0XC0;  
        RS=0;  
        RW=0;  
        LCDE=1;  
        LCDE=0;  
    }  
    P0=dat;  
    RS=1;  
    RW=0;  
    LCDE=1;  
    LCDE=0;  
}
```