LD3320

单芯片/非特定人/动态编辑识别列表

语音识别芯片

并行/串行读写辅助说明

ICRoute 用声音去沟通
VUI (Voice User Interface)

Web : www.icroute.com
Tel : 021-68546025
Mail: info@icroute.com

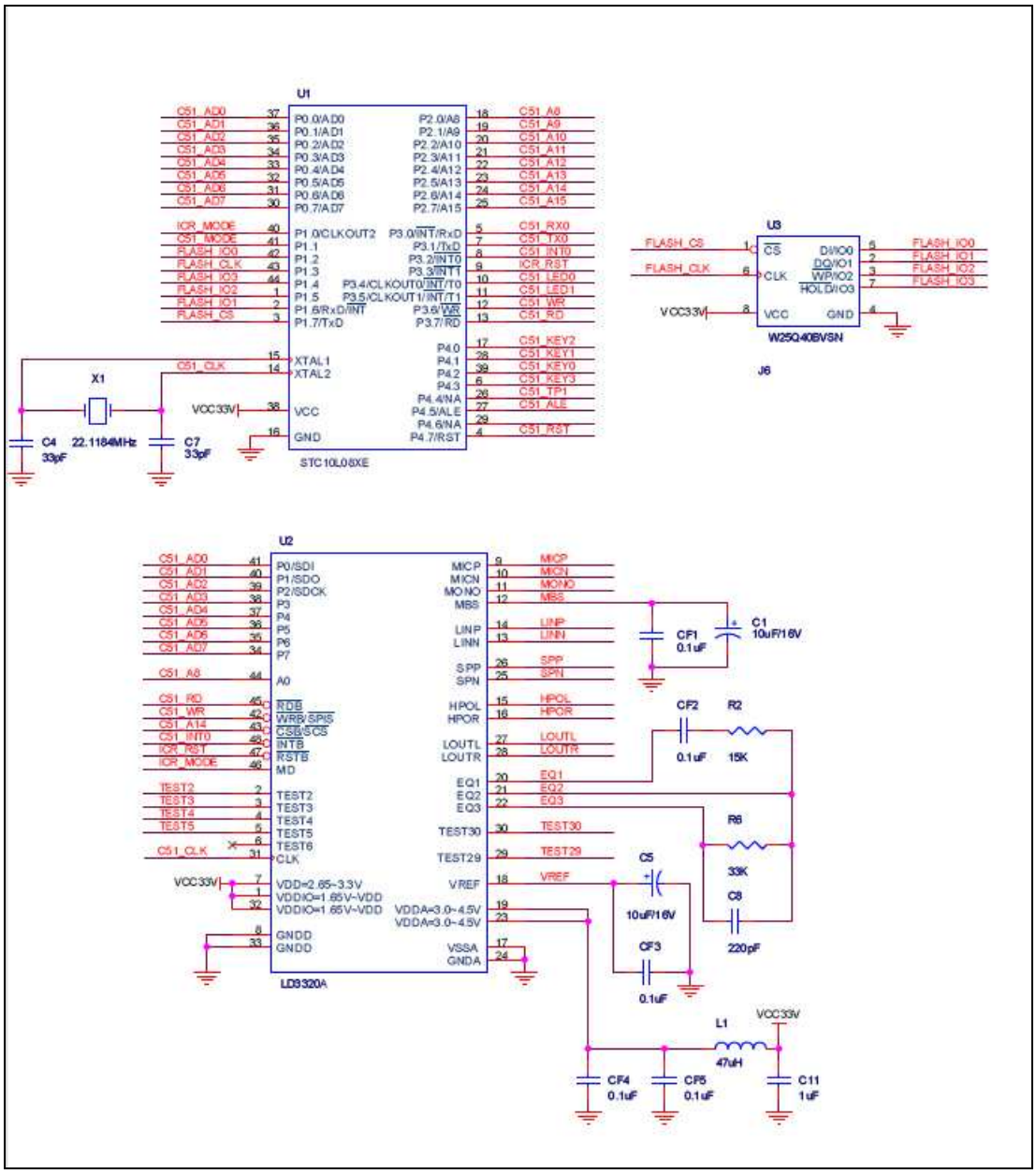
目录

一、系统介绍和说明

本文介绍四种对 LD 芯片的读写方式，分别是串行 SPI 的软、硬方式和并行 8 位总线的软、硬方式。所列出的代码都是在评估板上可以正常工作的代码。为了方便理解代码，首先介绍一下硬件的连接。（全部内容见《LD3320 测试版原理图》）

本文介绍的读写程序源代码，全部在参考程序 LD_Demo_Source 中“Reg_RW.c ”文件中。

评估板上的 MCU 型号为 STC10L08XE。



请注意以下连接：

1. 控制串行/并行的管脚：
   - ICR_MODE (P1.0)　连接 LD3320 芯片的 MD
     高电平为 SPI 方式，低电平为并行方式。
     用户选择好一种合适自己的方式后，最好以后不要变来变去。
     所以实际系统里这个管脚可以固定接高或者低。

2. 和串行 SPI 方式关联的管脚：
   - C51_WR（P3.6）连接 LD3320 芯片的 SPIS，
     低电平为 SPI 有效。
   - C51_AD0（P0.0）连接 LD3320 芯片的 SDI。
   - C51_AD1（P0.1）连接 LD3320 芯片的 SDO。
   - C51_AD2（P2.0）连接 LD3320 芯片的 SDCK。

3. 和并行总线方式关联的管脚：
   - C51_AD0 ~ C51_AD7 连接 LD3320 芯片的 P0~P7。
   - C51_A8（P2.0）连接 LD3320 芯片的 A0。
   - C51_A14（P2.6）连接 LD3320 芯片的 CSB。
   - C51_WR（P3.6）连接 LD3320 芯片的 WRB。
   - C51_RD（P3.7）连接 LD3320 芯片的 RDB。

二．并行方式

1. 并行方式--直接读写 （硬件实现并行读写方式）

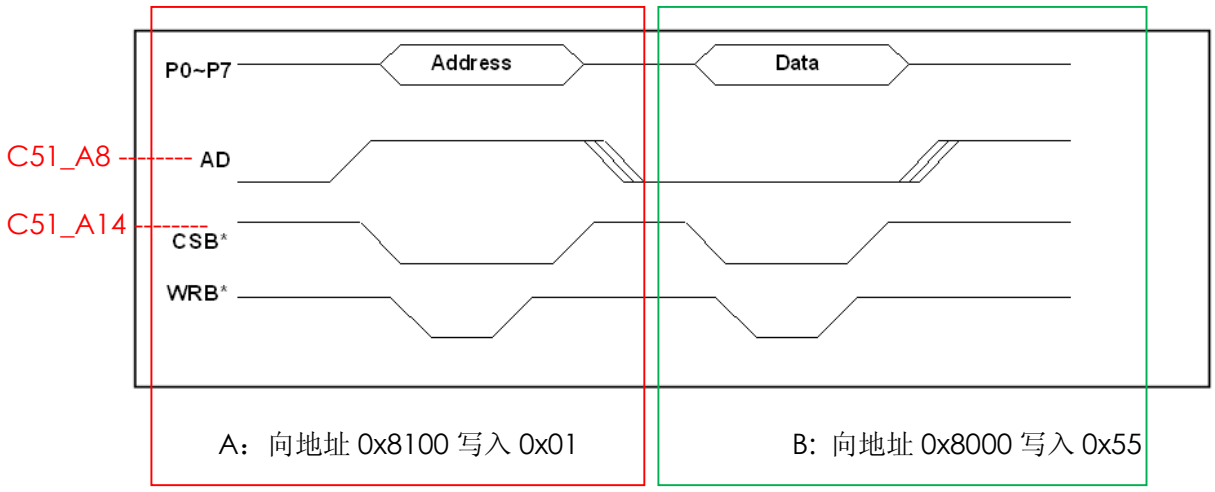
由于设计硬件电路板时，考虑到了双方芯片读写的时序特征，那么在合理连接的基础上，通过 2 条语句就可实现对 LD 芯片的操作。这种方式代码简练，执行速度最快。
这是因为 STC 的单片机 STC10L08XE 自身带有硬件的并口方式，STC10L08XE 有单独的 WR 和 RD 端口，可以在读写并行总线时，自动产生 WR 和 RD 信号。

例程代码如下：

```
#define LD_INDEX_PORT (*((volatile uint8 xdata*)(0x8100)))
#define LD_DATA_PORT  (*((volatile uint8 xdata*)(0x8000)))
void LD_WriteReg( uint8 ulAddr, uint8 ucVal )
{
  LD_INDEX_PORT  = ulAddr;
  LD_DATA_PORT = ucVal;
}
uint8 LD_ReadReg( uint8 ulAddr )
{
  LD_INDEX_PORT = ulAddr;
  return (uint8)LD_DATA_PORT;
}
```

并行写：

例如，向寄存器 0x01 写 0x55，时序图如下。

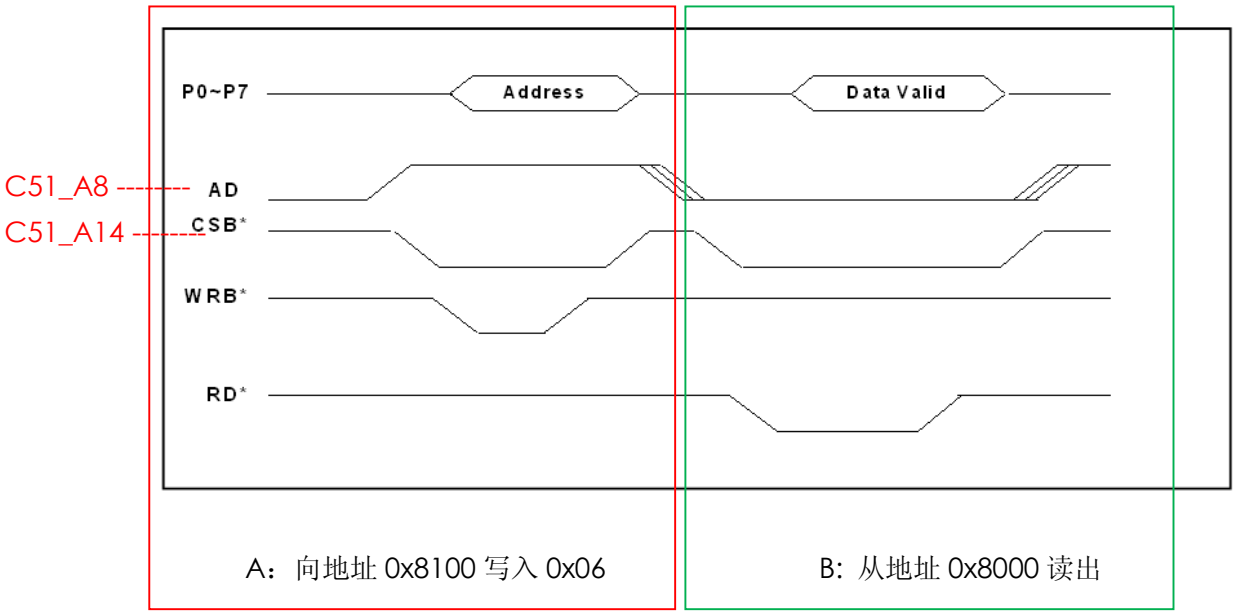


A: 0x8100 的二进制是 1**0**000000**1** 00000000　　A14=0　A8=1
根据连接可知，此时 P0-P7 是 51 发送的数据（0x01）
同时 CSB=0　A0=1　MCU 的 WR 也自动会送出一个低有效。

B: 0x8000 的二进制是 1**0**000000**0** 00000000　　A14=0　A8=0
根据连接可知，此时 P0-P7 是 51 发送的数据（0x55）
同时 CSB=0　A0=0　MCU 的 WR 也会自动送出一个低有效。

并行读：

例如，从寄存器 0x06 读取 8 字节数据，时序图如下。



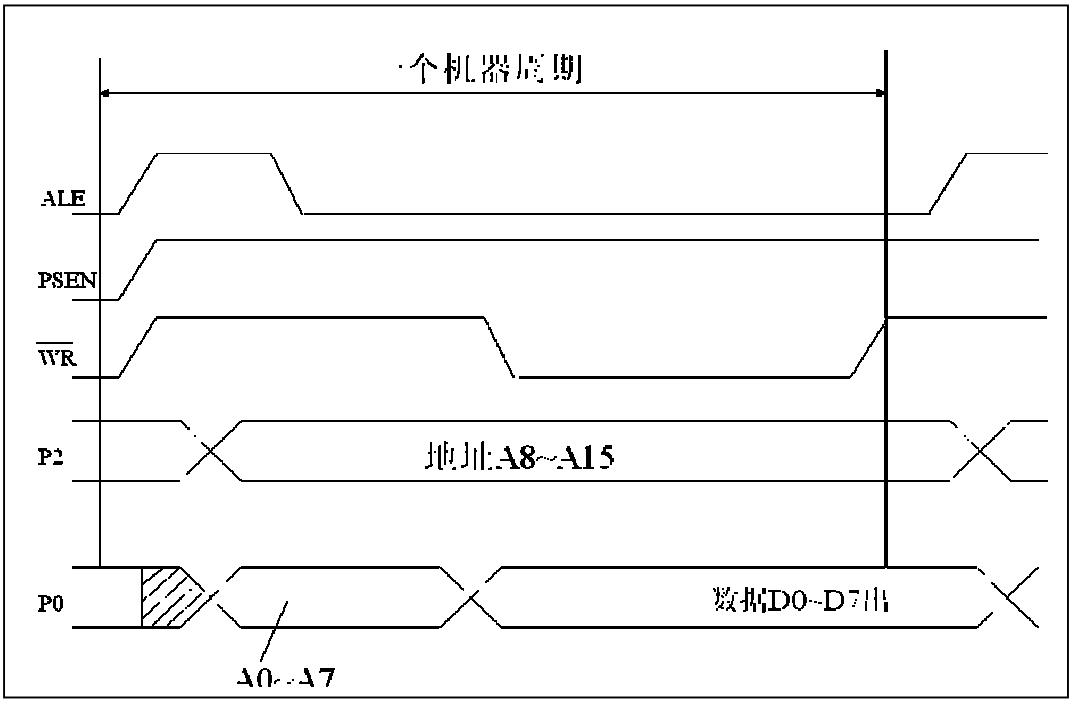
A: 0x8100 的二进制是 1**0**000000**1** 00000000　　A14=0　A8=1
根据连接可知，此时 P0-P7 是 51 发送的数据（0x06）
同时 CSB=0　A0=1　MCU 的 WR 也会送出一个低有效。

B: 0x8000 的二进制是 1**0**000000**0** 00000000　　A14=0　A8=0
根据连接可知，CSB=0　A0=0　MCU 的 RD 会送出一个低有效。
此时 LD3320 芯片会准备好要读出的数据放在 P0-P7。
于是 MCU 可以读取想要的数据。

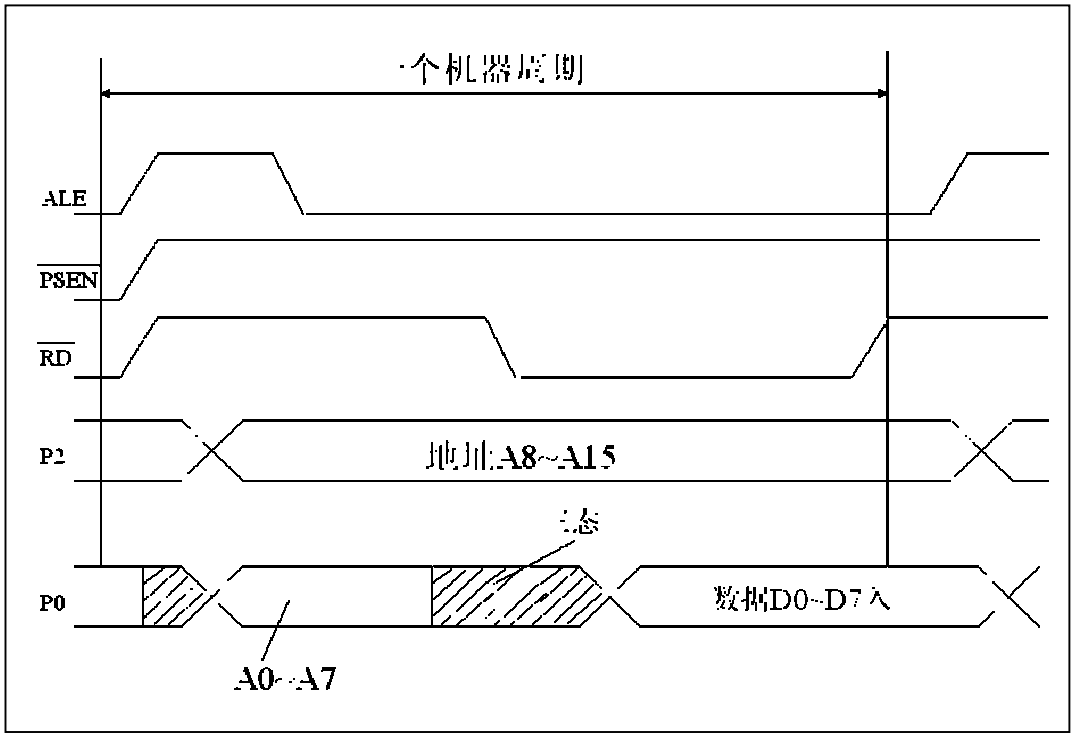
补充说明：
1. 地址 A15=1 是为了避免和低端地址空间冲突。
2. A14 和 A8 以外其他地址线和读写没有关系。
3. WR 和 RD 的有效低信号是 MCU 读写外存（xdata）地址时自动发出。
4. 可参考下面 MCS51 的外存（xdata）读写时序，以帮助理解。
   在评估板的线路中，ALE 和 PSEN 没有使用，而用了 A8 和 A14 作为控制信号。
   A0-A7 的数据虽然会有时出现在 P0-P7 上，但是对读写没有影响。

MCS-51 对外存(xdata)的读写时序。

写时序（地址可达 16 位，数据 8 位）



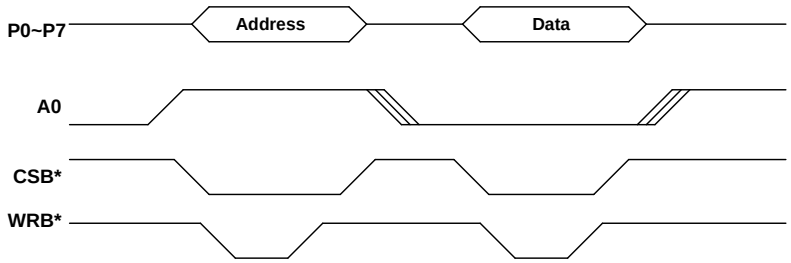
读时序（地址可达 16 位，数据 8 位）



2. 并行方式—软件模拟时序 （软件模拟并行读写）

并行写的时序

如下：



代码就是完全按照时序来写： 准备好地址后，A0 设为 1，通过对 CSB 和 WRB 拉低拉高来写入地址；准备好数据后，A0 设为 0，通过对 CSB 和 WRB 拉低拉高来写入数据。

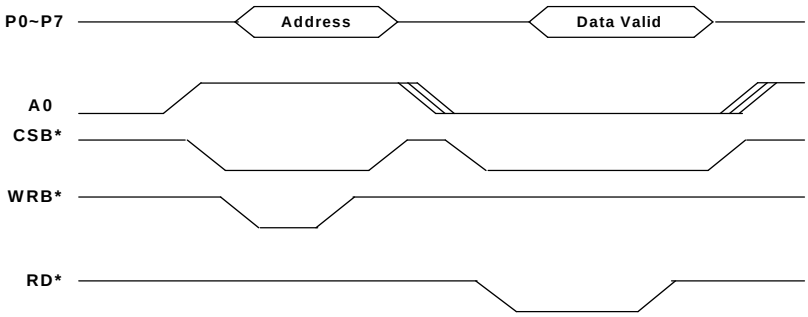
后面还有对延时长度的详细讨论。

```
#define DELAY_NOP _nop_();_nop_();_nop_();
sbit LD_WR   = P3^6;
sbit LD_RD   = P3^7;
sbit LD_CS   = P2^6;
sbit LD_A0   = P2^0;
void LD_WriteReg( unsigned char address, unsigned char dataout )
{
    P0 = address;
    LD_A0= 1;
    LD_CS = 0;
    LD_WR = 0;
    DELAY_NOP;

    LD_WR = 1;
    LD_CS = 1;
    DELAY_NOP;

    P0 = dataout;
    LD_A0 = 0;
    LD_CS = 0;
    LD_WR = 0;
    DELAY_NOP;

    LD_WR = 1;
    LD_CS = 1;
```

```
    DELAY_NOP;
}
```

并行读的时序

如下：



代码就是完全按照时序来写： 准备好地址后，A0 设为 1，通过对 CSB 和 WRB 拉低拉高来写入地址；A0 设为 0，通过对 CSB 和 RD 拉低拉高来读入数据。

（相关定义在写函数前）

```
unsigned char LD_ReadReg( unsigned char address )
{
    unsigned char datain;

    P0 = address;
    LD_A0 = 1;
    LD_CS = 0;
    LD_WR = 0;
    DELAY_NOP;

    LD_WR = 1;
    LD_CS = 1;
    DELAY_NOP;

    LD_A0= 0;
    LD_CS = 0;
    LD_RD = 0;
    DELAY_NOP;

    datain = P0;
    LD_RD = 1;
    LD_CS = 1;
    DELAY_NOP;
```

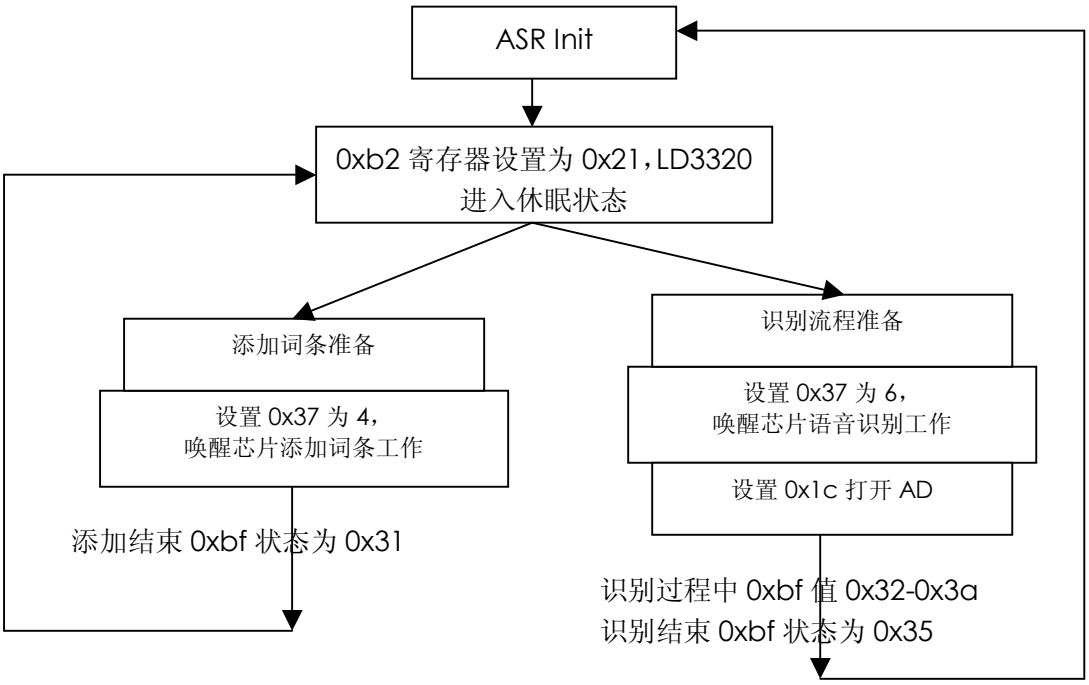
```
    return datain;
}
```

（重要说明！）

关于**并行方式—软件模拟时序**方式下，延时的使用说明。

在这种读写方式下，如果 DELAY_NOP 相应部分的**延迟过长**，会导致写 0x37 寄存器时数据无法正常写入，芯片也可能会出现各种各样的工作异常。

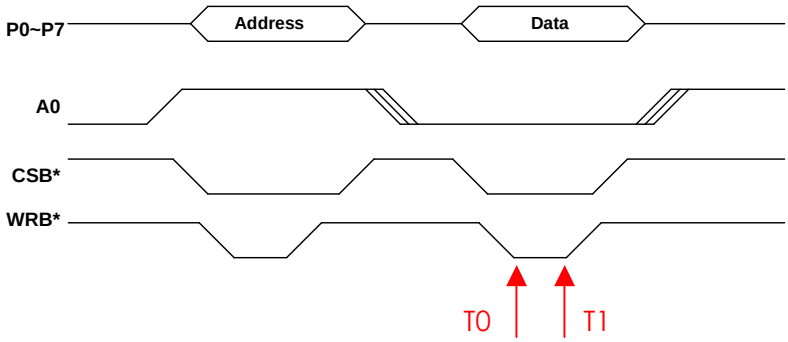
在语音识别流程中，0x37 寄存器是控制命令入口，用来启动芯片内部的语音识别运算模块。0xb2 寄存器是内部忙闲的状态寄存器，一些内部状态将由 0xbf 寄存器报告出来



在这个流程中，写 0x37 寄存器实际上有二重功能， 1).唤醒芯片内部工作，2).将命令字送进去选择哪项工作。

当调用 LD_WriteReg(0x37, 0x06);时，先把 0x37 这个寄存器地址写入了芯片，此时已经唤醒了芯片内部的语音识别模块，需要立即把命令字(0x06)传入芯片，否则就会造成语音识别模块接受不到该命令字，造成该语句设置失效。一般的表现就是在 LD_WriteReg(0x37, 0x06);后读取 0xBF 的数值没有按照预期进行变化。

重温一下 LD3320 的并口写时序图：



在图中，标出了“T0” 和“T1”二个时间点。对于写 0x37 寄存器，在 T0 时间点的时候，由于已经确定了是写 0x37 寄存器，所以就将芯片内部唤醒并开始语音识别流程；而在 T1 时间点，数据才会随同上跳沿拿进来。对其他寄存器，只是为了设置寄存器的值，这个长度没有什么关系。而对于 0x37 寄存器的双重功能来说，这个时间长度太长就会带来时序上的错误。

请注意：SPI 方式是异步写寄存器的，不会有这个时序错误的风险。

在软件模拟并口的时序中，此时就需要尽可能地避免 delay，在给出的参考程序中，在把 CS 拉低后，到把 CS 再拉高完成一次写，最多只放置了 3 个 nop。

用户如果出现 LD_WriteReg(0x37, 0x06);Delay(100);延时一段时间后 0xBF 寄存器没有按照预期进行变化，就需要在并口读写的时序中把这里的写时序尽可能地拉快。

芯片内部为 0x37 寄存器准备了一个 byte 的缓冲用来接受命令字,但是当外部提供的时序过于缓慢时，就是指[T0 ， T1] 脉冲太宽时，该缓冲无法正常接受到命令字。

所以用户也可以连续调用两次该设置语句，看是否可以起作用。

（以上的解释也包括对 LD_WriteReg(0x37, 0x04);的说明）

综上，用户如果发现前面的流程正常（包括声音播放），仅仅是最后的识别无法正常进行，LD_WriteReg(0x37, 0x06);后 0xBF 寄存器始终没有变化，芯片不停地送出中断，0x2B 寄存器的 bit[3]始终为 1。可能是因为没有正确地设置 0x37 寄存器导致语音识别模块没有启动而造成。

解决办法：

1. 调整并口的读写时序，保证写寄存器的速度足够快。

2. 权益之计：尝试连续调用二次 LD_WriteReg(0x37, 0x04)； 以及连续调用二次 LD_WriteReg(0x37, 0x06);看是否可以解决。如果可以解决，还是希望用户可以通过调整并口的写时序，来从根本上解决。

三．串行方式

LD3320 芯片接受 SPI 规则，但是要注意：

1） LD3320 接受的命令为写(0x04) 和读(0x05)。
2） SDCK 的下降沿有效。
3） 读数据的时候，每当遇到上升沿，LD3320 芯片的 SDO 会发生数据变化。
4） LD 芯片的 SPI 接口可以接受的最大 SDCK 时钟频率是 1.5MHZ。
5） SDO 在不使用的时候也会输出低电平，（包括 SCS=高 的时候），所以如果 MCU 需要连接多个 SPI 设备的时候，应在硬件设计时做好隔离。

1． 串行方式—直接读写 （硬件实现 SPI 读写）

有些 MCU 有硬件的 SPI 接口，通过合适的链接，可以直接用 SPI 读写命令操作 LD3320 芯片。注意此时要将 LD3320 的 MD 管脚设为高电平，而 SPIS 管脚设为低电平。有高低电平编号的管脚为：SCS，SDCK，SDI 和 SDO。（后面有详细的时序介绍）

STC 单片机（带 SPI 口的种类）兼顾读写的函数例程

```
unsigend char SPI_TR( unsigned char x )
{
    SPSTAT=0xC0;
    SPDAT=x;
    while(!(SPSTAT&0x80));
    return SPDAT;
}
```

AVR 单片机（带 SPI 口的种类）兼顾读写的函数例程

```
unsigend char SPI_TR( unsigned char x )
{
    SPDR=x;
    while(!(SPSR & (1<<SPIF)));
    return SPDR;
}
```

写和读的代码如下。

```
void LD_WriteReg( unsigned char address, unsigned char dataout )
{
    // 这里添加硬件 SPI 口的操作代码：
    SPI_TR(0x04); // 发送 0x04
    SPI_TR(address); // 发送 address
```

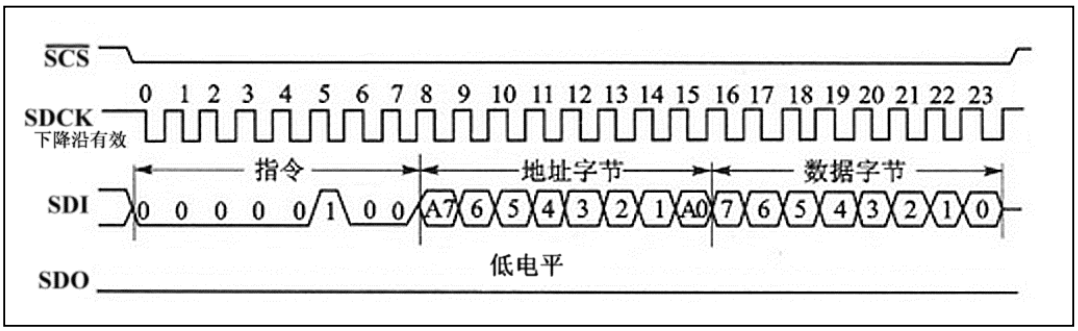
```
    SPI_TR(dataout); // 发送 dataout
}

unsigned char LD_ReadReg( unsigned char address )
{
    // 这里添加硬件 SPI 口的操作代码：
    SPI_TR(0x05); // 发送 0x05
    SPI_TR(address); // 发送 address
    return (SPI_TR(0)); // 读出数据，并返回
}
```

2. 串行方式—软件模拟时 （软件模拟 SPI 读写）

这种方式下，需要用程序来模拟 SPI 的时序。

写寄存器

时序图：



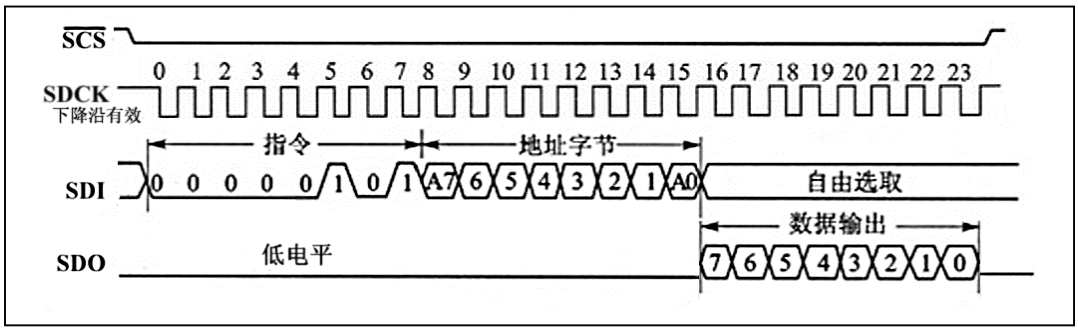
写操作的源代码：

```
#define DELAY_NOP _nop_();_nop_();_nop_();

sbit SCS=P2^6;      //芯片片选信号
sbit SDCK=P0^2;     //SPI 时钟信号
sbit SDI=P0^0;      //SPI 数据输入
sbit SDO=P0^1;      //SPI 数据输出
sbit SPIS=P3^6;     //SPI 模式设置：低有效。

void LD_WriteReg(unsigned char address,unsigned char dataout)
{
    unsigned char i = 0;
    unsigned char command=0x04;
    SPIS =0;
    SCS = 0;
    DELAY_NOP;
```

```
        //write command
        for (i=0;i < 8; i++)
        {
            if ((command & 0x80) == 0x80)
                SDI = 1;
            else
                SDI = 0;

            DELAY_NOP;
            SDCK = 0;
            command = (command << 1);
            DELAY_NOP;
            SDCK = 1;
        }
        //write address
        for (i=0;i < 8; i++)
        {
            if ((address & 0x80) == 0x80)
                SDI = 1;
            else
                SDI = 0;
            DELAY_NOP;
            SDCK = 0;
            address = (address << 1);
            DELAY_NOP;
            SDCK = 1;
        }
        //write data
        for (i=0;i < 8; i++)
        {
            if ((dataout & 0x80) == 0x80)
                SDI = 1;
            else
                SDI = 0;
            DELAY_NOP;
            SDCK = 0;
            dataout = (dataout << 1);
            DELAY_NOP;
            SDCK = 1;
        }
        DELAY_NOP;
        SCS = 1;
    }
```

读寄存器

时序图：



读操作的源代码：（相关定义在写函数前）

```
unsigned char LD_ReadReg(unsigned char address)
{
    unsigned char i = 0;
    unsigned char datain =0 ;
    unsigned char temp = 0;
    unsigned char command=0x05;
    SPIS =0;
    SCS = 0;
    DELAY_NOP;

    //write command
    for (i=0;i < 8; i++)
    {
        if ((command & 0x80) == 0x80)
            SDI = 1;
        else
            SDI = 0;
        DELAY_NOP;
        SDCK = 0;
        command = (command << 1);
        DELAY_NOP;
        SDCK = 1;
    }

    //write address
    for (i=0;i < 8; i++)
    {
        if ((address & 0x80) == 0x80)
```

```
            SDI = 1;
        else
            SDI = 0;
        DELAY_NOP;
        SDCK = 0;
        address = (address << 1);
        DELAY_NOP;
        SDCK = 1;
    }
    DELAY_NOP;

    //Read data
    for (i=0;i < 8; i++)
    {
        datain = (datain << 1);
        temp = SDO;
        DELAY_NOP;
        SDCK = 0;
        if (temp == 1)
            datain |= 0x01;
        DELAY_NOP;
        SDCK = 1;
    }

    DELAY_NOP;
    SCS = 1;
    return datain;
}
```

在 SPI-软件模拟方式下，读写寄存器操作比较麻烦，速度也较慢。在评估板上，这种方式下语音识别没有问题，但是声音播放可能会不流畅。 这是由于采用的低端 51 处理器作系统主控 MCU，系统运行频率很低，导致通过 SPI 传送的 MP3 数据的速度跟不上播放的速度。如果用户的系统中采用高主频的主控 MCU，就会解决这一问题。

在这个例程里，使用的延时是 3 个 nop(); 但是不同的 MCU，不同的系统，也可能遇到无法正常读写寄存器的情况，此时调整延时的长短，比如增加一些 nop() 可能会解决问题。

完。