ZLG7290 I²C 接口键盘及 LED 驱动器

一、特点

1. I²C 串行接口，提供键盘中断信号，方便与处理器接口；
2. 可驱动 8 位共阴数码管或 64 只独立 LED 和 64 个按键；
3. 可控扫描位数，可控任一数码管闪烁；
4. 提供数据译码和循环，移位，段寻址等控制；
5. 8 个功能键，可检测任一键的连击次数；
6. 无需外接元件即直接驱 LED，可扩展驱动电流和驱动电压；
7. 提供工业级器件，多种封装形式 PDIP24，SO24。

二、引脚及说明

采用 24 引脚封装，引脚图如图 1 所示。其引脚功能分述如下：

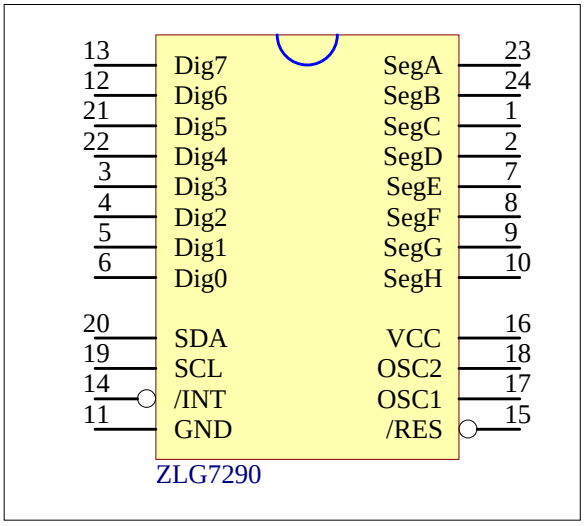


图 1　引脚图

三、功能描述

1．键盘部分

ZLG7290 可采样 64 个按键或传感器，可检测每个按键的连击次数。其基本功能如下：

（1）键盘去抖动处理

当键被按下和放开时，可能会出现电平状态反复变化，称作键盘抖动。若不作处理会引起按键盘命令错误，所以要进行去抖动处理，以读取稳定的键盘状态为准。

（2）双键互锁处理

当有两个以上按键被同时按下时，ZLG7290 只采样优先级高的按键（优先顺序为 S1>S2>…>S64，如同时按下 S2 和 S18 时采样到 S2）。

（3）连击键处理

当某个按键按下时，输出一次键值后，如果该按键还未释放，该键值连续有效，就像连续压按该键一样，这种功能称为连击。连击次数计数器（RepeatCnt）可区别出单击（某些功能不允许连击，如开/关）或连击。判断连击次数可以检测被按时间，以防止某些功能误操作（如连续按 5 秒经入参数设置状态）。

（4）功能键处理

功能键能实现 2 个以上按键同时按下来扩展按键数目或实现特殊功能。如 PC 机上的“Shift”、“Ctrl”、“Alt”键。典型应用图中的 S57~S64 为功能键。

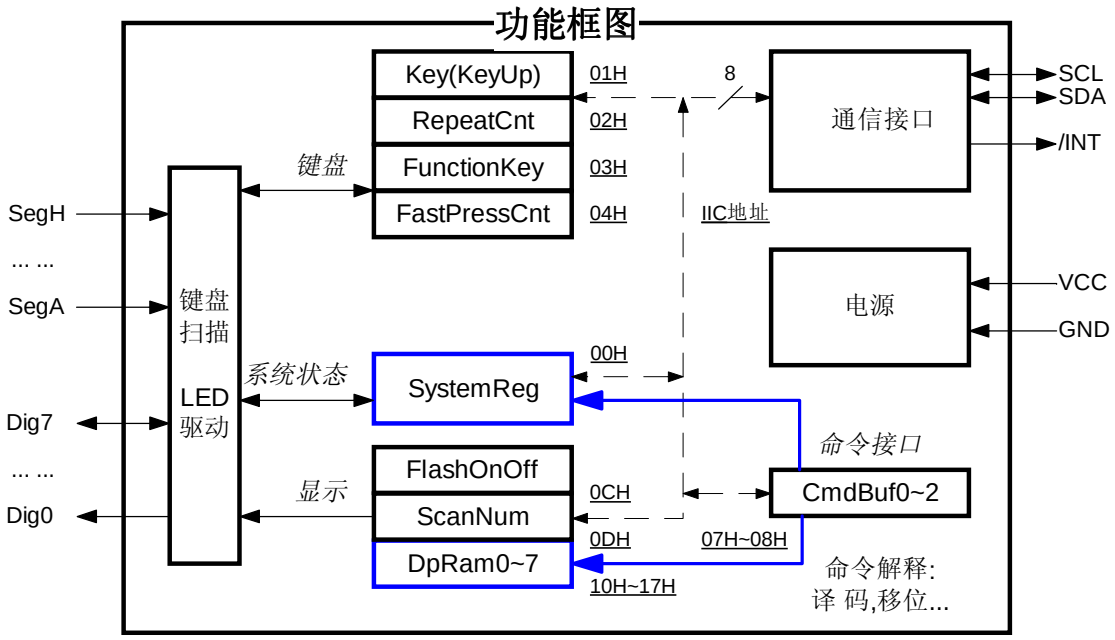


图 2 系统功能框图即寄存器映象图

2．显示部分

在每个显示刷新周期，ZLG7290 按照扫描位数寄存器（ScanNum）指定的显示位数 N，把显示缓存 DpRam0～DpRamN 的内容按先后循序送入 LED 驱动器实现动态显示，减少 N 值可提高每位显示扫描时间的占空比，以提高 LED 亮度，显示缓存中的内容不受影响。修改闪烁控制寄存器（FlashOnOff）可改变闪烁频率和占空比（亮和灭的时间）。

ZLG7290 提供两种控制方式：寄存器映象控制和命令解释控制，如上述对显示部分的控制，寄存器映象控制是指直接访问底层寄存器，实现基本控制功能，这些寄存器须字节操作。

命令解释控制是指通过解释命令缓冲区（CmdBuf0～CmdBuf1）中的指令，间接访问底层寄存器实现扩展控制功能。如实现寄存器的位操作；对显示缓存循环，移位；对操作数译码等操作。请参考指令详解部分。

四、寄存器详解

系统状态部分

1．系统寄存器（SystemReg）：地址 00H，复位值 11110000B。系统寄存器保存 ZLG7290 系统状态，并可对系统运行状态进行配置，其功能分位描述如下：

- KeyAvi（SystemReg.0）： 置 1 时表示有效的按键动作（普通键的单击，连击，和功能键状态变化），/INT 引脚信号有效（变为低电平）；清 0 表示无按键动作，/INT 引脚信号无效（变为高阻态）。有效的按键动作消失后或读 Key 后，KeyAvi 位自动清 0。

键盘部分

2．键值寄存器（Key）：地址 01H，复位值 00H。Key 表示被压按键的键值。当 Key=0 时，表示没有键被压按。

3．连击次数计数器（RepeatCnt）：地址 02H，复位值 00H。RepeatCnt=0 时，表示单击键。RepeatCnt 大于 0 时，表示键的连击次数。用于区别出单击键或连击键，判断连击次数可以检测被按时间。

4．功能键寄存器（FunctionKey）：地址 03H，复位值 0FFH。 FunctionKey 对应位的值=0 表示对应功能键被压按（FunctionKey.7～FunctionKey.0 对应 S64～S57）。

命令接口部分

5．命令缓冲区（CmdBuf0～CmdBuf1）：地址 07H～08H，复位值 00H～00H。用于传输指令。

显示部分

6．闪烁控制寄存器（FlashOnOff）：地址 0CH，复位值 0111B/0111B。高 4 位表示闪烁时亮的时间，低 4 位表示闪烁时灭的时间，改变其值同时也改变了闪烁频率，也能改变亮和灭的占空比。FlashOnOff 的 1 个单位相当于 150～250ms（亮和灭的时间范围为：1～16，0000B 相当 1 个时间单位），所有象素的闪烁频率和占空比相同。

7．扫描位数寄存器（ScanNum）：地址 0DH，复位值 7。用于控制最大的扫描显示位数（有效范围为：0～7，对应的显示位数为：1～8），减少扫描位数可提高每位显示扫描时间的占空比，以提高 LED 亮度。不扫描显示的显示缓存寄存器则保持不变。如 ScanNum＝3 时，只显示 DpRam0～DpRam3 的内容。

8．显示缓存寄存器（DpRam0～DpRam7）：地址 10H～17H，复位值 00H～00H。缓存中一位置 1 表示该像素亮，DpRam7～DpRam0 的显示内容对应 Dig7～Dig0 引脚。

五、通信接口

ZLG7290 的 I^2C 接口传输速率可达 32kbit/s，容易与处理器接口。并提供键盘中断信号，提高主处理器时间效率。ZLG7290 的从地址 （slave address）为 70H(01110000B)。

有效的按键动作（普通键的单击，连击和功能键状态变化），都会令系统寄存器（SystemReg）的 KeyAvi 位置 1，/INT 引脚信号有效（变为低电平）。用户的键盘处理程序可由/INT 引脚低电平中断触发，以提高程序效率；也可以不采样/INT 引脚信号节省系统的 I/O 数，而轮询系统寄存器的 KeyAvi 位。要注意读键值寄存器会令 KeyAvi 位清 0，并会令/INT 引脚信号无效。为确保某个有效的按键动作所有参数寄存器的同步性，建议利用 I^2C 通信的自动增址功能连续读 RepeatCnt，FunctionKey 和 Key 寄存器，但用户无需太担心寄存器的同步性问题，应为键参数寄存器变化速度较缓慢（典型 250ms，最快 9ms）。

ZLG7290 内可通过 I^2C 总线访问的寄存器地址范围为：00H～17H，任一寄存器都可按字节直接读写，也可以通过命令接口间接读写或按位读写，请参考指令详解部分。支持自动增址功能（访问一寄存器后，寄存器子地址（sub address）自动加一）和地址翻转功能（访问最后一寄存器（子地址＝17H）后，寄存器子地址翻转为 00H）。ZLG7290 的控制和状态查询全部都是通过读/写寄存器实现的，用户只需象读写 24C02 内的单元一样，即可实现对 ZLG7290 的控制，关于 I^2C 总线访问的细节请参考 I^2C 总线规范。。

六、指令详解

ZLG7290 提供两种控制方式：寄存器映象控制和命令解释控制，寄存器映象控制是指直接访问底层寄存器（除通信缓冲区外的寄存器），实现基本控制功能，请参考寄存器详解部分。命令解释控制是指通过解释命令缓冲区（CmdBuf0～CmdBuf1）中的指令，间接访问底层寄存器实现扩展控制功能。如实现寄存器的位操作；对显示缓存循环，移位；对操作数译码等操作。

一个有效的指令由一字节操作码和数个操作数组成，只有操作码的指令称为纯指令，带操作数的指令称为复合指令。一个完整的指令须在一个 I^2C 帧中（起始信号和结束信号间）连续传输到命令缓冲区（CmdBuf0～CmdBuf1）中，否则会引起错误。

1. 纯指令

(1) 左移指令

命令缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CmdBuf0:	0	0	0	1	N3	N2	N1	N0

该指令使与 ScanNum 相对应的显示数据和显示属性（闪烁）自右向左移动 N 位（(N3~N0）+1)。移动后，右边 N 位无显示，与 ScanNum 不相关的显示数据和显示属性则不受影响。

例： DpRamB～DpRam0=“87654321”其中“4”闪烁，ScanNum＝5（“87”不显示)。

执行指令 00010001B 后，DpRamB～DpRam0=“ 4321 ”。“4”闪烁，高两位和低两位无显示。

(2) 右移指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	0	1	0	N3	N2	N1	N0

与左移指令类似，只是移动方向为自左向右，移动后，左边 N 位（(N3~N0）+1）无显示。

例： DpRamB～DpRam0=“87654321”其中“3”闪烁，ScanNum＝5（“87”不显示)。

执行指令 00100001B 后，DpRamB～DpRam0=“ 6543”。“3”闪烁，高四位无显示。

(3) 循环左移指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	0	1	1	N3	N2	N1	N0

与左移指令类似，不同的是在每移动一位后，原最左位的显示数据和属性转移到最右位。

例： DpRamB～DpRam0=“87654321”其中“4”闪烁，ScanNum＝5（“87”不显示)。

执行指令 00110001B 后，DpRamB～DpRam0=“ 432165”。“4”闪烁，高两位无显示。

(4) 循环右移指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	1	0	0	N3	N2	N1	N0

与循环左移指令类似，只是移动方向相反。

例： DpRamB～DpRam0=“87654321”其中“3”闪烁，ScanNum＝5（“87”不显示)。

执行指令 01000001B 后，DpRamB～DpRam0=“ 216543”。“3”闪烁。

(5) SystemReg 寄存器位寻址指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	1	0	1	On	S2	S1	S0

当 On＝1 时，第 S（S2~S0）位置 1；当 On＝0 时，第 S 位清 0。

2. 复合指令

(1) 显示像素寻址指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	0	0	0	0	0	0	1
ComBuf1:	On	0	S5	S4	S3	S2	S1	S0

当 On＝1 时，第 S（S5~S0）点像素亮（置 1)；当 On＝0 时，第 S 点像素灭（清 0)。该指令用于点亮 / 关闭数码管中某一段，或 LED 矩阵中某一特定的 LED；该指令受 ScanNum 的内容影响。S6~S0 为像素地址，有效范围从 00H—3FH，无效的地址不会产生任何作用。像素位地址映象如下：

像素地址	Sa	Sb	Sc	Sd	Se	Sf	Sg	Sh
DpRam0	00H	01H	02H	03H	04H	05H	06H	07H
DpRam1	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH

...								
DpRam7	38H	39H	3AH	3BH	3CH	3DH	3EH	3FH

(2) 按位下载数据且译码指令

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ComBuf0:	0	1	1	0	A3	A2	A1	A0
ComBuf1:	DP	Flash	0	D4	D3	D2	D1	D0

其中 A3～A0 为显示缓存编号（范围为：0000B～0111B，对应 DpRam0～DpRam7，无效的编号不会产生任何作用），DP=1 时点亮该位小数点，Flash=1 时该位闪烁显示，Flash=0 时该位正常显示，D4～D0 为要显示的数据，按以下表规则进行译码：

D5	D4	D3	D2	D1	D0	十六进制	显示内容	D5	D4	D3	D2	D1	D0	十六进制	显示内容
0	0	0	0	0	0	00H	0	0	1	0	0	0	0	10H	G
0	0	0	0	0	1	01H	1	0	1	0	0	0	1	11H	H
0	0	0	0	1	0	02H	2	0	1	0	0	1	0	12H	i
0	0	0	0	1	1	03H	3	0	1	0	0	1	1	13H	J
0	0	0	1	0	0	04H	4	0	1	0	1	0	0	14H	L
0	0	0	1	0	1	05H	5	0	1	0	1	0	1	15H	o
0	0	0	1	1	0	06H	6	0	1	0	1	1	0	16H	P
0	0	0	1	1	1	07H	7	0	1	0	1	1	1	17H	q
0	0	1	0	0	0	08H	8	0	1	1	0	0	0	18H	r
0	0	1	0	0	1	09H	9	0	1	1	0	0	1	19H	t
0	0	1	0	1	0	0AH	A	0	1	1	0	1	0	1AH	U
0	0	1	0	1	1	0BH	b	0	1	1	0	1	1	1BH	y
0	0	1	1	0	0	0CH	C	0	1	1	1	0	0	1CH	c
0	0	1	1	0	1	0DH	d	0	1	1	1	0	1	1DH	h
0	0	1	1	1	0	0EH	E	0	1	1	1	1	0	1EH	T
0	0	1	1	1	1	0FH	F	0	1	1	1	1	1	1FH	无显示

(3) 闪烁控制指令：

通信缓冲区	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CmdBuf0:	0	1	1	1	X	X	X	X
CmdBuf1:	F7	F6	F5	F4	F3	F2	F1	F0

当 Fn=1 时，该位闪烁（n 的范围为：0~7，对应 0~7 位）；当 Fn=0 时，该位不闪烁。该指令会改变所有像素的闪烁属性！

例： 执行指令 01110000B，00000000B 后，所有数码管不闪烁。

七、附录

A. 电气特性 T_A=25℃

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
VCC	工作电压	—	—	3.3	5	5.5	V
IDD1	工作电流	3.3	LED 全灭	—	1	2	mA
		5	无键按下	—	3	5	
VIL1	SDA,SCL 输入低电平	—	—	0	-	0.3VCC	V
VIH1	SDA,SCL 输高低电平	—	—	0.7VCC	-	VCC	V
VIL2	/RST 输入低电平	—	—	0	-	0.4VCC	V
VIH2	/RST 口输入高电平	—	—	0.9VCC	-	VCC	V
IOL	INT 输出灌电流	3.3V	VOL=0.1VCC	4	8	-	mA
		5V	VOL=0.1VCC	10	20	-	
IDL	Dig0~Dig7 灌电流	3.3V	VOL=0.1VCC	4	8	-	mA
		5V	VOL=0.1VCC	10	20	-	
IDH	SegA~SegH 源电流	3.3V	VOH=0.9VCC	-2	-4	-	mA
		5V	VOH=0.9VCC	-5	-10	-	
f_{I2C}	I2C 接口速度	—	上拉电阻 3K3	20	-	32	KHz

极限参数:

电源供应电压..... GND-0.3V ~ GND+6.0V　　储存温度..... -50 ~ 125℃

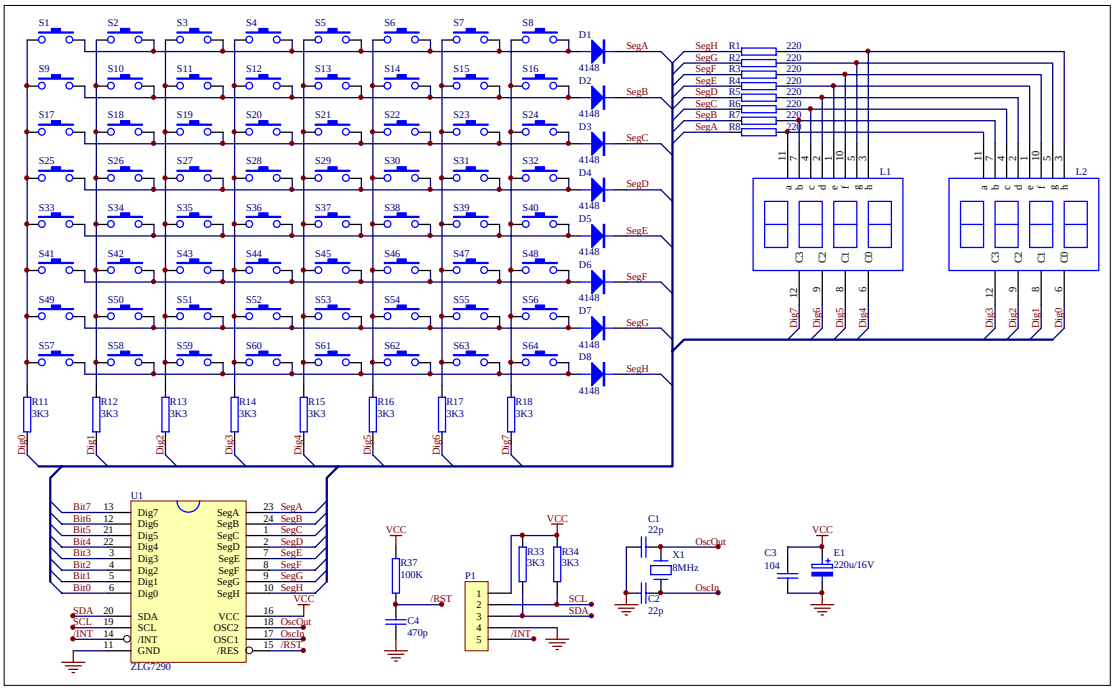
端口输入电压..... GND-0.3V ~ VCC+0.3V　　工作温度..... -40 ~ 85℃

注.这里只强调：额定功率超过极限参数所规定的范围将对芯片造成损害无法预期！芯片在上述标示范围外的工作状态而且若长期在标示范围外的条件下工作可能影响芯片的可靠性。

B. 引脚说明

引脚号	引脚名称	引脚属性	引脚描述
13,12,21,22,3~6	Dig7~ Dig0	输入/输出	LED 显示位驱动及键盘扫描线
10~7,2,1,24,23	SegH~SegA	输入/输出	LED 显示段驱动及键盘扫描线
20	SDA	输入/输出	I2C 总线接口数据/地址线
19	SCL	输入/输出	I2C 总线接口时钟线
14	/INT	输出	中断输出端,低电平有效
15	/RES	输入	复位输入端,低电平有效
17	OSC1	输入	连接晶体以产生内部时钟
18	OSC2	输出	
16	VCC	电源	电源正(3.3~5.5V)
11	GND	电源	电源地

C. 应用举例



D. 范例程序

该范例实现数字的输入与修改功能，K0~K9(S10，S1~S9)为数字键，对应数字 0，1~9，用于输入和修改数字。以上键都可连击实现快速输入和修改； KRight(S11)为右移键， KLeft(S12)为左移键，在修改模式下，右移键或左移键用于选择要修改的位，可连击；KMode(S13)模式键，实现进入 / 退出修改模式。不允许连击。

使用该程序前要根据系统接线和速度配置 Zlg90Demo.c（或 Zlg90Demo.a51）文件内的 KeyInt 和 IicSMasU.inc 文件。其中项目 DemoC 是用 C 语言实现的范例，项目 DemoAsm 是用汇编语言实现

的范例。两者实现同样的功能，并且项目中都必须包含 IicSMasU.a51 文件，C 编译器使用 Keil C51 V6.0 及以上。

```
/*****
;FileName:        Zlg90Demo.c
;Describe:        ZLG7290 应用范例：80c51 系列模拟 I2C 总线主控器，ZLG7290 为被控器
;                 此程序在 ZLG7290 test board 上运行，
;              * K0~K9(S10,S1~S9)为数字键,对应数字 0,1~9
;                   用于输入和修改数字,以上键都可连击实现快速输入和修改;
;              * KRight(S11)为右移键; KLeft(S12)为左移键
;               在修改模式下，右移键或左移键用于选择要修改的位;
;              * KMode(S13)模式键
;                   KMode(S13)键实现进入 / 退出修改模式。不允许连击。
;Author:      广州周立功单片机发展有限公司 伍仕峰 Blueyes Wu
;Date:            2003/04/25
*****/
#include<reg51.h>
#define      uchar    unsigned char
#define      uint     unsigned int

//hardware 与硬件有关的定义
    //按键定义
    #define K0            10                  //S10
    #define K1            1                   //S1
    #define K2            2                   //S2
    #define KRight        11                  //S11
    #define KLeft         12                  //S12
    #define KMode             13              //S13
    #define SlvZlg7290    0x70                //ZLG7290 从地址
    #define SubKey        0x1                 //键码值子地址
    #define SubCmdBuf     0x7                 //命令缓冲区子地址
    #define SubDpRam      0x10                //显存子地址
    //根据硬件配置 IicSM.inc
    //根据硬件配置 ZLG7290 的中断信号引脚 INT
    sbit KeyInt=P3^3;                         //中断信号
//hardware

//80c51 系列模拟 I2C 总线主控器驱动程序接口说明
/*****
;Name:        _IicTxdRxd(TxdByte,RxdByte,&IicDataBuf)
;Describe:    发送数据给被控器/接收来自被控器的数据
;Input:       TxdByte(R7)=要发送数据的字节数
;         RxdByte(R5)=要接收数据的字节数
;         IicDataBuf(R1)=发送/接收缓冲区首字节的地址
;         (SlvAddr(被控器地址),SubAddr(单元地址))
```

```
;Output: C(Retry=1):操作失败标志
;         IicDataBuf=接收到的数据首字节地址(主接收时有效)
;调用说明:     A.现行地址写: SlvAddr(写)   !!!(带 SubAddr 的器件不能使用该子程序)
;                   TxdByte=(发送数据字节数(SubAddr 为第一个要发送的数据))
;                   RxdByte=0
;         B.指定地址写: SlvAddr(写),SubAddr
;                   TxdByte=(1+发送数据字节数)
;                   RxdByte=0
;         C.现行地址读: SlvAddr(读)
;                   TxdByte=0
;                   RxdByte=要接收数据的字节数
;         D.指定地址读: SlvAddr(读),SubAddr
;                   TxdByte=1
;                   RxdByte=要接收数据的字节数
;Nesting level:    1
;Change:     A,C,R1, R4~R7
;****/
extern      bit IicTxdRxd(uchar TxdByte,uchar RxdByte, uchar *IicDataBuf);
                                                    //函数定义(程序入口地址)
extern      data uchar SlvAddr;                     //被控器从地址
extern      data uchar SubAddr;                     //单元地址（子地址）

uchar       DpBuf[8];                               //显示缓冲区
uchar       i;                                      //显示缓冲区指针
uchar       IicWriteBuf[3];                         //IIC 写缓冲区
uchar       IicReadBuf[3];                          //IIC 读缓冲区
    #define key             IicReadBuf[0]
    #define key_repeat      IicReadBuf[1]
    #define FunctionKey     IicReadBuf[2]
bit         EditMode;                               //修改模式

void main(void){
    uchar   KeyNum;                                 //数字键键码（0～9）
    uchar   Temp;

    KeyInt=1;                                       //置 KeyInt 引脚为输入状态

    for (i=0;i<8;i++){
        DpBuf[i]=0;                                 //显示缓冲区初始化值
    }
    EditMode=0;                                     //非修改模式（输入模式）
    i=0;                                            //显示缓冲区指针初始化

    while(1){
```

```
while(KeyInt==0){                                   //等待按键
    //读 key,key_repeat,FunctionKey 的内容到 IicReadBuf0~2
    SlvAddr=SlvZlg7290;
    SubAddr=SubKey;
    while(IicTxdRxd(1,3,&IicReadBuf)==1);           //读出错重试

    if(key<=K0){                                    //有效的数字键
        KeyNum=key;
        if(key==K0){
            KeyNum=0;
        };
        if(EditMode==0){
            //输入模式下，左移一位。
            IicWriteBuf[0]=0x10;                    //左移一位指令
            SlvAddr=SlvZlg7290;
            SubAddr=SubCmdBuf;
            IicTxdRxd(1+1,0,&IicWriteBuf);
        };
        //输出一位，控制闪烁
        DpBuf[i]=KeyNum;
        IicWriteBuf[1]=DpBuf[i];
        IicWriteBuf[0]=0x60+i;                  //在第 i 位数码管译码并显示 DpBuf[i]
        if(EditMode){
            IicWriteBuf[1]|=0x40;               //修改模式下闪烁
        };
        SlvAddr=SlvZlg7290;
        SubAddr=SubCmdBuf;
        IicTxdRxd(1+2,0,&IicWriteBuf);
    }else{                                          //控制键
        if((key==KMode)&(key_repeat==0)){           //不允许连击
            EditMode=~EditMode;
            i=0;                                    //显示缓冲区指针初始化
        };
        IicWriteBuf[1]=0x00;                        //当前位不闪烁
        if (EditMode==1){
            //修改模式,选择要修改的位
            if(key==KLeft){
                if(i<7){
                    i++;
                };
            };
            if(key==KRight){
                if(i>0){
                    i--;
```

```
                        };
                    };
                    Temp=i+1;
                    IicWriteBuf[1]=1;
                    for(;Temp>1;Temp--){
                        //当前位(新选择的)闪烁
                    IicWriteBuf[1]=IicWriteBuf[1]+IicWriteBuf[1];
                    }
                }
                IicWriteBuf[0]=0x70;                    //闪烁控制指令
                SlvAddr=SlvZlg7290;
                SubAddr=SubCmdBuf;
                IicTxdRxd(1+2,0,&IicWriteBuf);
            }
        }
    }
}

/*****
; FileName:       Zlg90Demo.a51
; Describe:       ZLG7290 应用范例：80c51 系列模拟 I2C 总线主控器，ZLG7290 为被控器
;                 此程序在 ZLG7290 test board 上运行，
;             * K0~K9(S10,S1~S9)为数字键,对应数字 0,1~9
;                  用于输入和修改数字,以上键都可连击实现快速输入和修改；
;             * KRight(S11)为右移键; KLeft(S12)为左移键
;                  在修改模式下，右移键或左移键用于选择要修改的位；
;             * KMode(S13)模式键
;                  KMode(S13)键实现进入 / 退出修改模式。不允许连击。
; Author:     广州周立功单片机发展有限公司 伍仕峰 Blueyes Wu
; Date:           2003/04/25
*****/
$include (reg51.inc)

;hardware 与硬件有关的定义
;按键定义
K0          equ 10      ;S10
K1          equ 1       ;S1
K2          equ 2       ;S2
KRight      equ 11      ;S11
KLeft       equ 12      ;S12
KMode       equ 13      ;S13

SlvZlg7290  equ 0x70                    ;ZLG7290 从地址
SubKey      equ 0x1                     ;键码值子地址
```

```
SubCmdBuf   equ 0x7                         ;命令缓冲区子地址
SubDpRam    equ 0x10                        ;显存子地址

;根据硬件配置 IicSM.inc

;根据硬件配置 ZLG7290 的中断信号引脚 INT
KeyInt      equ P3.3                        ;中断信号
;hardware

extrn       code(_IicTxdRxd)                ;程序入口地址
extrn       data(SlvAddr)                   ;被控器从地址
extrn       data(SubAddr)                   ;单元地址（子地址）

DataSegZlg90Demo SEGMENT DATA
RSEG DataSegZlg90Demo
DpBuf:      ds  8                           ;显示缓冲区
i:      ds  1                               ;显示缓冲区指针
IicWriteBuf: ds  3                          ;IIC 写缓冲区
IicReadBuf: ds  3                           ;IIC 读缓冲区
    key     equ IicReadBuf
    key_repeat  equ IicReadBuf+1
    FunctionKey equ IicReadBuf+2

EditModebit 1                               ;修改模式
KeyNum:         ds  1                       ;数字键键码（0～9）
Temp:       DS  1

?STACK      SEGMENT   IDATA
        RSEG    ?STACK
        DS  1

CodeSegZlg90Demo SEGMENT CODE
RSEG CodeSegZlg90Demo
    org     0
main:
    MOV     sp,#?STACK-1                    ;系统堆栈初始化
    SETB    KeyInt                          ;置 KeyInt 引脚为输入状态

    CLR     A                               ;显示缓冲区初始化值
    MOV     R0,#DpBuf
    MOV     R1,#8
DpBufIni:
    MOV     @R0,a
    INC     R0
```

```
    DJNZ    R1,DpBufIni

    CLR     EditMode                        ;非修改模式（输入模式）
    MOV     i,A                             ;显示缓冲区指针初始化

Loop:
    JB      KeyInt,Loop
    ;读 key,key_repeat,FunctionKey 的内容到 IicReadBuf0~2
    MOV     SlvAddr,#SlvZlg7290
    MOV     SubAddr,#SubKey
Loop1:
    MOV     R1,#IicReadBuf
    MOV     R5,#3
    MOV     R7,#1
    LCALL   _IicTxdRxd
    JC      Loop1                           ;读出错重试
    MOV     A,key
    SETB    C
    SUBB    A,#K0
    JNC     CtrlKey
    ;有效的数字键
    MOV     KeyNum,key
    MOV     A,key
    CJNE    A,#K0,ForKey0End
    MOV     KeyNum,#0
ForKey0End:
    JB      EditMode, ForEditEnd
    ;输入模式下，左移一位。
    MOV     IicWriteBuf,#010H               ;左移一位指令
    MOV     SlvAddr,#SlvZlg7290
    MOV     SubAddr,#SubCmdBuf
    MOV     R1,#IicWriteBuf
    MOV     R5,#0
    MOV     R7,#1+1
    LCALL   _IicTxdRxd
ForEditEnd:
    ;输出一位，控制闪烁
    MOV     A,#LOW (DpBuf)
    ADD     A,i
    MOV     R0,A
    MOV     @R0,KeyNum

    MOV     A,@R0
    JNB     EditMode, ForNoEditEnd
```

```
    ORL     A,#040H                     ;修改模式下闪烁
ForNoEditEnd:
    MOV     IicWriteBuf+01H,A
    MOV     A,i
    ADD     A,#060H
    MOV     IicWriteBuf,A               ;在第 i 位数码管译码并显示 DpBuf+i
    MOV     SlvAddr,#SlvZlg7290
    MOV     SubAddr,#SubCmdBuf
    MOV     R1,#IicWriteBuf
    MOV     R5,#0
    MOV     R7,#1+2
    LCALL   _IicTxdRxd
    SJMP    ForNext
CtrlKey:                                ;控制键
    MOV     A,key_repeat
    JNZ     ForKModeEnd
    MOV     A,key
    CJNE    A,#KMode, ForKModeEnd
    ;单击 KMode
    CPL     EditMode
    CLR     A
    MOV     i,A                         ;显示缓冲区指针初始化
ForKModeEnd:
    MOV     IicWriteBuf+1,#0H           ;当前位不闪烁
    JNB     EditMode, InInputMode
    ;修改模式,选择要修改的位
    MOV     A,key
    CJNE    A,#KLeft, ForKLeftEnd
    INC     i
ForKLeftEnd:
    MOV     A,key
    CJNE    A,#KRight, ForKRightEnd
    DEC     i
ForKRightEnd:
    ANL     i,#111B

    MOV     DPTR,#Tab8Sel1
    MOV     A,i
    MOVC    A,@A+DPTR
    MOV     IicWriteBuf+1,A             ;当前位(新选择的)闪烁
InInputMode:
    MOV     IicWriteBuf,#70H            ;闪烁控制指令
    MOV     SlvAddr,#SlvZlg7290
    MOV     SubAddr,#SubCmdBuf
```

```
    MOV     R1,#IicWriteBuf
    MOV     R5,#0
    MOV     R7,#1+2
    LCALL   _IicTxdRxd
ForNext:
    LJMP    Loop

Tab8Sel1:
    DB  00000001B,00000010B,00000100B,00001000B
    DB  00010000B,00100000B,01000000B,10000000B

END
;/*****
;FileName:      IicSMasU.a51
;Describe:      80c51 系列模拟 I2C 总线主控器驱动程序
;Author:        伍仕峰 Blueyes Wu
;Date:          2003/01/12
;*****/

$include (IicSMasU.inc)

public _IicTxdRxd
;RAM
public SlvAddr
public SubAddr

;/*****
;Name:      bit _IicTxdRxd(TxdByte,RxdByte,&IicDataBuf)
;Describe:  发送数据给被控器/接收来自被控器的数据
;Input:     TxdByte=要发送数据的字节数
;           RxdByte=要接收数据的字节数
;           IicDataBuf=发送/接收缓冲区的首字节
;           (SlvAddr(被控器地址),SubAddr(单元地址))
;Output:    C(Retry=1):操作失败标志
;           IicDataBuf=接收到的数据的首字节(主接收时有效)
;调用说明:
;       A.现行地址写: SlvAddr(写)   !!!(带 SubAddr 的器件不能使用该子程序)
;                   TxdByte=(发送数据字节数(SubAddr 为第一个要发送的数据))
;                   RxdByte=0
;       B.指定地址写: SlvAddr(写),SubAddr
;                   TxdByte=(1+发送数据字节数)
;                   RxdByte=0
;       C.现行地址读: SlvAddr(读)
;                   TxdByte=0
```

```
;                   RxdByte=要接收数据的字节数
;         D.指定地址读: SlvAddr(读),SubAddr
;                   TxdByte=1
;                   RxdByte=要接收数据的字节数
;Nesting level:    1
;Change:     A,C,R1, R4~R7
;****/

BitSeglicSM     SEGMENT BIT overlayable
RSEG BitSeglicSM
Retry:          dbit      1     ;指明 I2C 最后的数据传送失败应该重复操作

DataSeglicSM    SEGMENT DATA overlayable
RSEG DataSeglicSM
SlvAddr:        ds   1          ;被控器地址
SubAddr:        ds   1          ;单元地址

TxdByte         equ  r7         ;要发送数据的字节数(第一传递参数)
RxdByte         equ  r5         ;要接收数据的字节数(第二传递参数)

WaitXTm macro   X               ;延时 X 个机器周期
        if X=0
            exitm
        endif
        if X=1
            nop
        endif
        if X=2
            nop
            nop
        endif
        if X=3
            nop
            nop
            nop
        endif

        if X>255
            error "the number of X is too much"
        else
            mov r6,#X/2
            DJNZ    r6,$
        endif
    endm
```

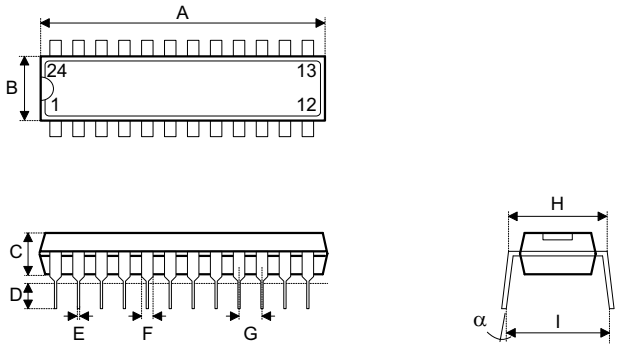
```
CodeSegIicSM    SEGMENT CODE
RSEG CodeSegIicSM

_IicTxdRxd:
    SETB    Retry                   ;设置错误标志位
;/*****
;发送起动条件
;*****/
SendStart:
    SETB    SDA
    SETB    SCL
    WaitXTm IicDelay
    CLR     SDA                     ;产生起始信号
    WaitXTm IicDelay
    CLR     SCL                     ;结束起动条件
;/*****
;送被控器地址地址,数据
;*****/
SendSlaAdr:
    MOV     A,SlvAddr
    CJNE    TxdByte,#0,SendSlaAdr1
    SETB    ACC.0                   ;TxdByte=0 时进行读操作
SendSlaAdr1:
    SETB    C                       ;检测应答位时释放 SDA 线
    CALL    XmByte

    JC      IicErr                  ;无应答出错
    JB      ACC.0,ReceiveData       ;SlaAdr.0=1 时进行读操作
    ;写操作
    MOVA,SubAddr
SendData:
    SETB    C                       ;检测应答位时释放 SDA 线
    CALL    XmByte

    JC      IicErr                  ;无应答出错
    MOV     A,@R1
    INC     R1
    DJNZ    TxdByte,SendData
    DEC     R1
    MOV     A,RxdByte
    JNZ     SendStart               ;RxdByte>0 时进行读操作
    JMP     SendStop
```

```
;/*****
;Name:       RcvByte
;Describe:   接收 1 字节数据(接收 8 位,发送+1 位(非)应答位)
;Input:      C=1:发送非应答位(通知 slave 停止通信)
;            C=0:发送应答位(通知 slave 继续发送后续字节)
;Output:     A=接收到的数据, C=(非)应答位
;Use:        A,R4,C,R6(WaitXTm)
;*****/
RcvByte:
    MOV     A,#0FFH             ;释放 SDA 线允许输入

;/*****
;Name:       XmByte
;Describe:   发送 1 字节数据(发送 8 位,接收 1 位(非)应答位)
;Input:      A=待发送的数据, C=1(检测应答位时释放 SDA 线)
;Output:     C=1:slave 非应答(slave 不响应)
;            C=0:slave 应答(slave 接收成功)
;Use:        A,R4,C,R6(WaitXTm)
;*****/
XmByte:
    MOV     R4,#9               ;设置数据格式为 8 位+1 位(非)应答位
RXBit:
    RLC     A                   ;左移数据
    MOV     SDA,C               ;output data
    SETB    SCL
    MOV     C,SDA               ;input data
    WaitXTm IicDelay
    CLR     SCL
    WaitXTm IicDelay
    DJNZ    R4,RXBit            ;重复操作直到处理完所有数据位
    RET

ReceiveData:
    MOV     A,RxdByte
    CJNE    A,#2,ReceiveData1   ;RxdByte=1(最后一个字节)时,发送非应答位(C=1)
                                ;否则发送应答位(C=0)
ReceiveData1:
    CALL    RcvByte
    MOV     @R1,A
    INC     R1
    DJNZ    RxdByte,ReceiveData

;/*****
;产生 I2C 停止条件程序
```

```
;*****/
SendStop:
    CLR     Retry                   ;清除错误标志位
IicErr:                             ;出错返回
    CLR     SDA
    SETB    SCL
    WaitXTm IicDelay
    SETB    SDA
    MOVC,Retry                      ;RETURN ERROR FLAG(C=Retry)
    RET

    END
;/*****
;FileName:      IicSMasU.inc
;Describe:      80c51 系列模拟 I2C 总线主控器驱动程序包含文件
;Author:        伍仕峰 Blueyes Wu
;Date:          2003/01/12
;*****/

$include (reg51.inc)

;hardware
IicDelay    EQU     15  ;与 I2C 信号延时(100KHz 时为 4.7uS)相对应的机器周期个数
                        ;如你的系统的 1 机器周期=0.5uS, I2C 总线速度为 50KHz(I2C 信号延
                        ;时约为 10uS) ,所以 IicDelay=(10uS/0.5uS)=20

SCL         EQU     p3.4
SDA         EQU     p3.5
;hardware
```

Plastic SKDIP Outline Dimensions

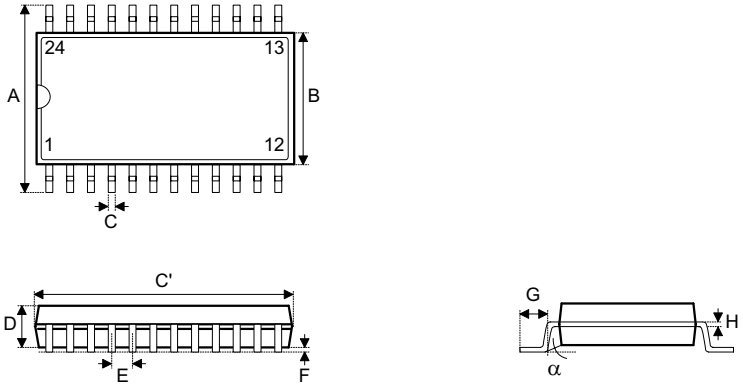
24-pin SKDIP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1235	—	1265
B	255	—	265
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	345	—	360
α	0°	—	15°

SOP Outline Dimensions

24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	590	—	614
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°