目录

第 1 章 ZLG72128 简介……1
1.1 主要特性……1
1.2 描述……1
1.3 引脚……2
第 2 章 引脚功能说明……3
第 3 章 典型应用电路图……4
3.1 电路原理图……4
3.2 电路简析……5
3.3 驱动大型数码管的方法……6
第 4 章 功能详解……9
4.1 功能概述……9
4.2 寄存器详解……9
4.2.1 系统寄存器 SystemReg（地址：00H）……9
4.2.2 键值寄存器 Key（地址：01H）……9
4.2.3 连击计数器 RepeatCnt（地址：02H）……9
4.2.4 功能键寄存器 FunctionKey（地址：03H）……9
4.2.5 命令缓冲区 CmdBuf0 和 CmdBuf1（地址：07H 和 08H）……10
4.2.6 闪烁控制寄存器 FlashOnOff（地址：0BH）……10
4.2.7 消隐寄存器 DispCtrl0(地址：0CH)和 DispCtrl1(地址：0DH)……10
4.2.8 闪烁寄存器 Flash0(地址：0EH)和 Flash1(地址：0FH)……10
4.2.9 显示缓冲区 DispBuf0～DispBuf11（地址：10H～1BH）……10
4.3 控制命令详解……10
4.3.1 段寻址（SegOnOff）……11
4.3.2 下载数据并译码（Download）……11
4.3.3 复位命令(Reset)……12
4.3.4 测试命令(Test)……12
4.3.5 左移命令(ShiftLeft)……12
4.3.6 循环左移命令(CyclicShiftLeft)……12
4.3.7 右移命令(ShiftRight)……13
4.3.8 循环右移命令(CyclicShiftRight)……13
第 5 章 I2C 接口……14
5.1 I2C 数据传输时序……14
5.1.1 起动（START）和停止（STOP）条件……14
5.1.2 位传输……14
5.1.3 应答位……14
5.2 ZLG72128 I2C 数据传输……15
第 6 章 ZLG72128 驱动程序软件包……17
6.1 软件包说明……17
6.1.1 I2C 驱动文件……17
6.1.2 I/O 中断定义与处理文件……18
6.1.3 ZLG72128 功能实现文件……19
第 7 章 ZLG72128 演示程序……23

附录 A ZLG72128 封装说明…………………………………………………………………… 31

第 1 章 ZLG72128 简介

1.1 主要特性

- 直接驱动 12 位共阴式数码管（1 英寸以下）或 96 只独立的 LED；
- 能够管理多达 32 只按键，自动消除抖动，其中有 8 只可以作为功能键使用；
- 利用功率电路可以方便地驱动 1 英寸以上的大型数码管；
- 具有位闪烁、位消隐、段点亮、段熄灭、功能键、连击键计数等强大功能；
- 提供有 10 种数字和 21 种字母的译码显示功能，或者直接向显示缓存写入显示数据；
- 与微控制器之间采用 I2C 串行总线接口，只需两根信号线，节省 I/O 资源；
- 工作电压范围：3.0～5.5V；
- 工作温度范围：－40～＋85℃；
- 封装： 标准 TSSOP28。

1.2 描述

ZLG72128 是广州周立功单片机科技有限公司自行设计的数码管显示驱动及键盘扫描管理芯片。能够直接驱动 12 位共阴式数码管（或 96 只独立的 LED），同时还可以扫描管理多达 32 只按键。其中有 8 只按键还可以作为功能键使用，就像电脑键盘上的 Ctrl、Shift、Alt 键一样。另外 ZLG72128 内部还设置有连击计数器，能够使某键按下后不松手而连续有效。采用 I2C 总线方式，与微控制器的接口仅需两根信号线。该芯片为工业级芯片，抗干扰能力强，在工业测控中已有大量应用。

电气特性（Vcc=5.0V，TA =25℃）

符号	参数	测试条件	最小	典型	最大	单位
V_{CC}	电源电压		3.0	5	5.5	V
I_{CC}	工作电流	LED 都不亮		2	5	mA
I_{CC}	工作电流	LED 全点亮		40	100	mA
I_{IO}	引脚输出电流				20	mA
V_{IH}	逻辑输入高电平		2.1		5.8	V
V_{IL}	逻辑输入低电平		-0.3		1.65	V
V_{OH}	逻辑输出高电平	Vcc = 5V Icc = 10mA	2.8			V
V_{OL}	逻辑输出低	Vcc = 5V Icc = 10mA			2.0	V
T_{KEY}	按键响应时间	含去抖时间		12		ms
T_{SCLL}	SCL 低电平时间		4.7			μs
T_{SCLH}	SCL 高电平时间		4.0			μs

T_{SUSDA}	SDA 建立时间		250			ns
T_{HSDA}	SDA 数据保持时间		0			ns
T_{RSDA}/T_{RSCL}	SDA 与 SCL 上升沿时间				1000	ns
T_{FSDA}/T_{FSCL}	SDA 与 SCL 下降沿时间				300	ns
T_{STA}	启动时间		4.0			μs
T_{RSTA}	重复启动时间		4.7			μs

1.3 引脚

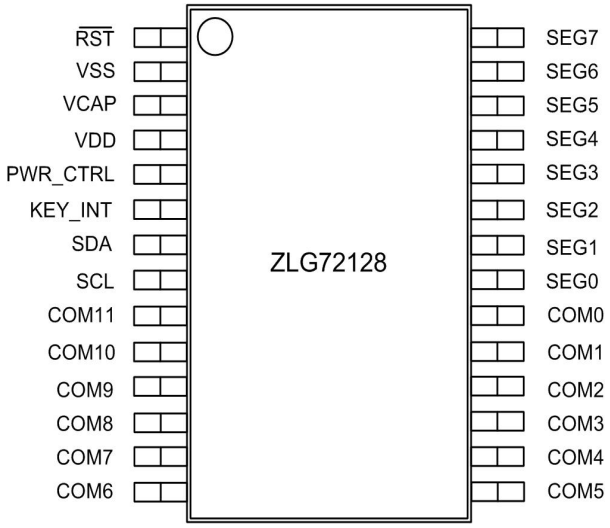


图 1.1 ZLG72128 引脚图

第 2 章 引脚功能说明

表 2.1 引脚功能表

引脚序号_28	引脚名称	功能描述
1	RST	复位信号，低电平有效
2	GND	接地
3	VCAP	外置电容端口
4	VDD	电源 3.0~5.0V
5	PWR_CTRL	未定义
6	KEY_INT	键盘中断输出，低电平有效
7	SDA	I2C 总线数据信号
8	SCL	I2C 总线时钟信号
9	COM11/KR3	数码管位选信号 11/键盘行信号 3
10	COM10/KR2	数码管位选信号 10/键盘行信号 2
11	COM9/KR1	数码管位选信号 9/键盘行信号 1
12	COM8/KR0	数码管位选信号 8/键盘行信号 0
13	COM7/KC7	数码管位选信号 7/键盘列信号 7
14	COM6/KC6	数码管位选信号 6/键盘列信号 6
15	COM5/KC5	数码管位选信号 5/键盘列信号 5
16	COM4/KC4	数码管位选信号 4/键盘列信号 4
17	COM3/KC3	数码管位选信号 3/键盘列信号 3
18	COM2/KC2	数码管位选信号 2/键盘列信号 2
19	COM1/KC1	数码管位选信号 1/键盘列信号 1
20	COM0/KC0	数码管位选信号 0/键盘列信号 0
21	SEG0	数码管 a 段
22	SEG1	数码管 b 段
23	SEG2	数码管 c 段
24	SEG3	数码管 d 段
25	SEG4	数码管 e 段
26	SEG5	数码管 f 段
27	SEG6	数码管 g 段
28	SEG7	数码管 dp 段

第 3 章 典型应用电路图

3.1 电路原理图

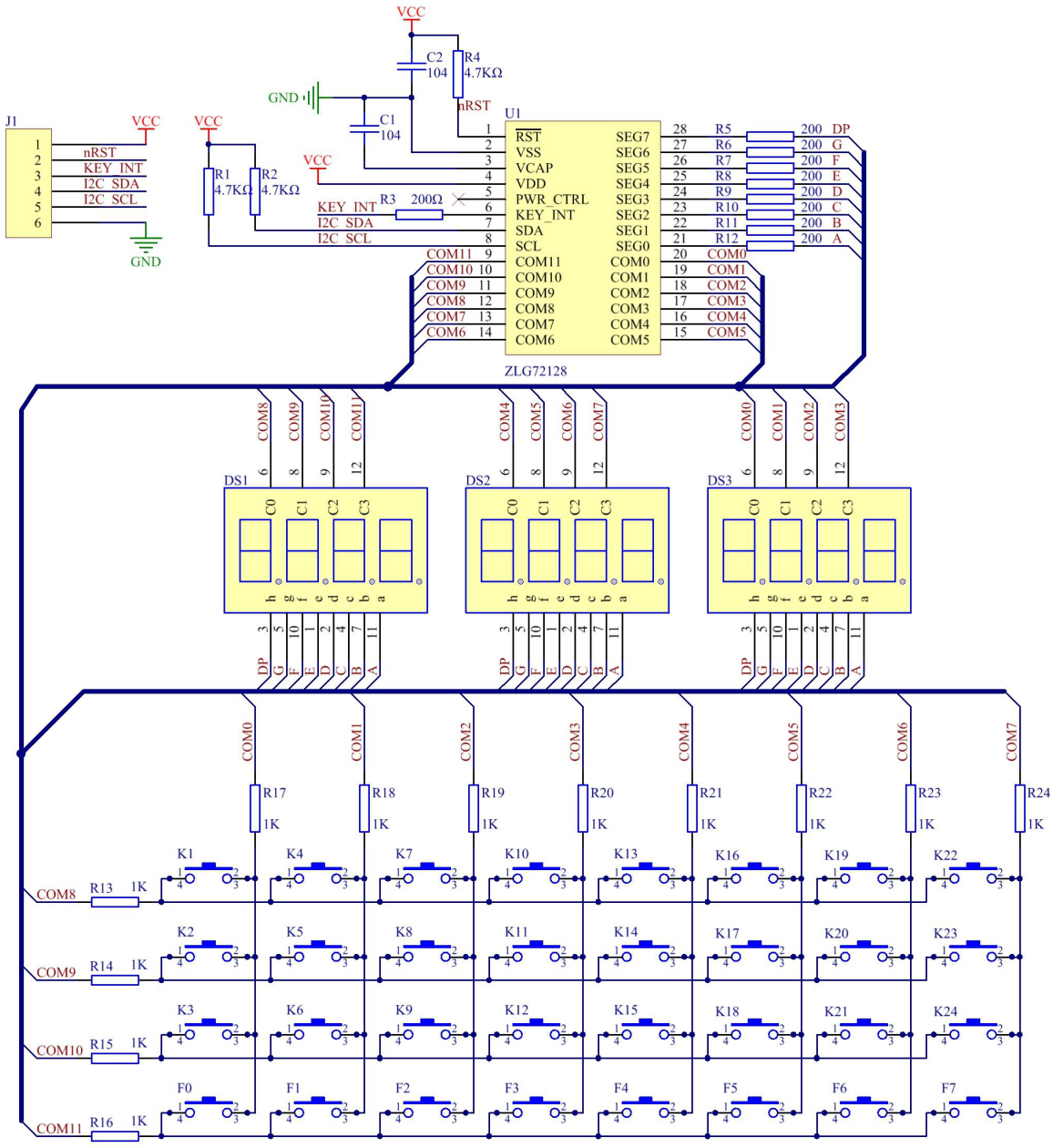


图 3.1 典型应用原理图

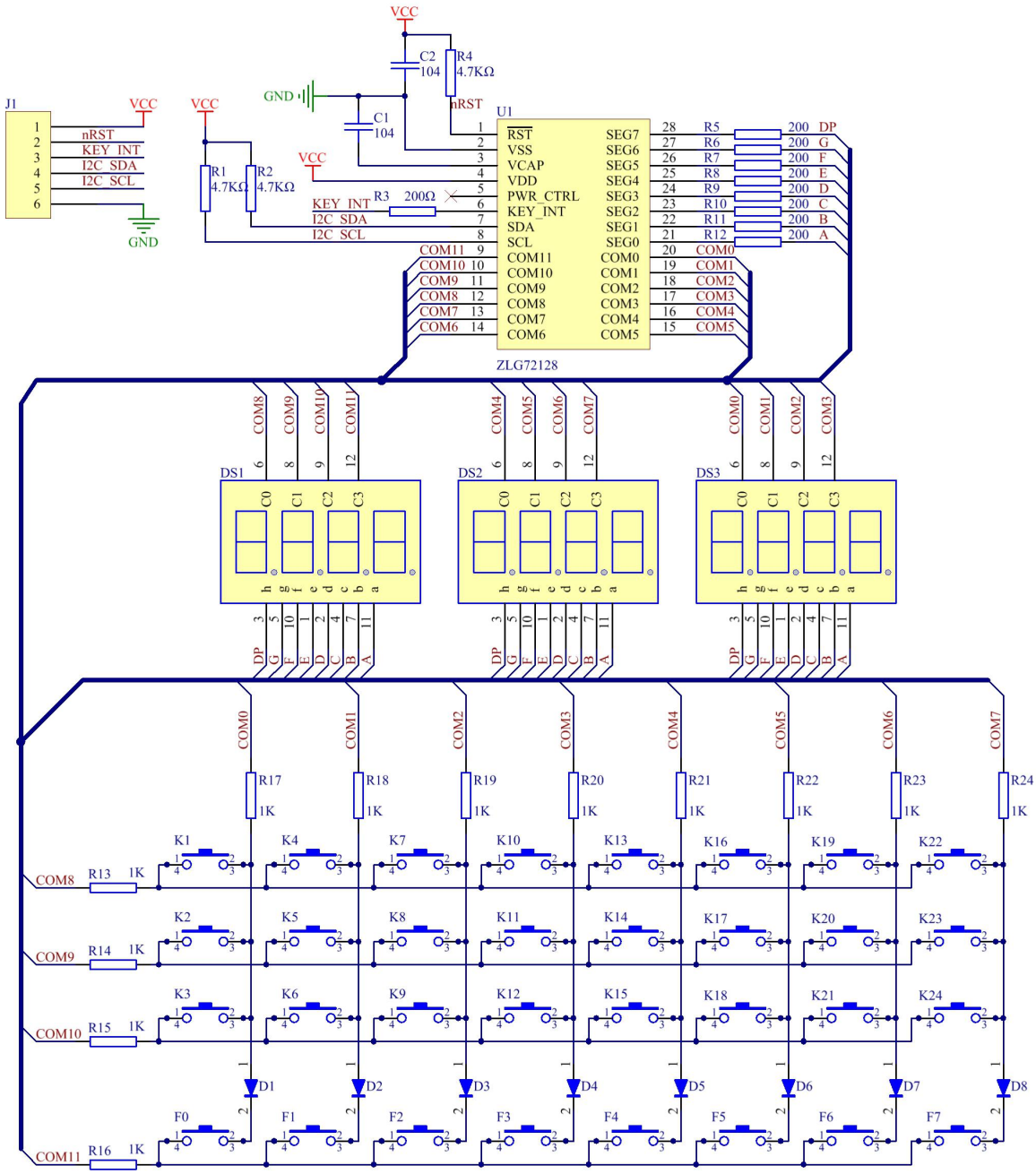


图 3.2　多个功能键使用时推荐应用原理图

3.2　电路简析

图 3.1 中是 ZLG72128 典型应用电路，U1 就是 ZLG72128 芯片。为了使电源更加稳定，一般要在 Vcc 到 GND 之间接入 47～470uF 的电解电容。J1 是 ZLG72128 与微控制器的接口以及电源输入，按照 I2C 总线协议的要求，信号线 SCL 和 SDA 上必须要分别接上拉电阻，其典型值是 4.7KΩ。当通信速率大于 100kbps 时，建议减小上拉电阻的值。复位信号是低电平有效，一般只需外接简单的 RC 复位电路，也可以通过直接拉低 RST 引脚的方法进行复位。不用悬空即可。

数码管必须是共阴式的，不能直接使用共阳式的。DS1、DS2 和 DS3 是 4 位联体式数码管，共同组成完整的 12 位。当然还可以采用其它的组合方式，如 6 只双联体式数码管。数码管在工作时要消耗较大的电流，R5～R12 是限流电阻，典型值是 270Ω。如果要增大数码管的亮度，

可以适当减小电阻值。

32 只按键中，前 24 个按键是普通按键 K1～K24，最后 8 个为功能键 F0～F7。键盘电阻 R13～R24 的典型值是 1KΩ。在多数应用当中可能不需要这么多的按键，这时可以按行或按列裁减键盘。如果不使用数码管，限流电阻 R5～R12 也都可以省略。

图 3.1 的典型应用原理图对于两个或两个按键按下的所有情况都是没有问题的，当如果需要使用两个或两个以上功能键与单个普通键组合使用的话，就需要在功能键与普通键之间加上一列二极管，如图 3.2 所示。二极管的选择主要在于导通压降，导通压降越小越好。

3.3 驱动大型数码管的方法

ZLG72128 的驱动能力毕竟有限，当使用大型数码管时，则可能显示亮度不够，这时可以适当减小数码管的限流电阻值以增加亮度，阻值最小为 200Ω，如果亮度依旧不够，就必须加入功率驱动电路。

图 3.3 是是使用分立功率驱动器件驱动大型数码管的电路图，以其中一路为例，ZLG72128 的段选信号 SA 是高电平有效，位选信号 COM 是低电平有效。SA 为高电平时，使 Q2 导通，Q2 的导通使得 Q1 导通；COM0 为低电平时，使 Q25 导通，Q25 导通使得 Q26 导通；这样就会有电流从 Vs 经限流电阻 R1 和数码管流到 GND，于是相应的数码管字段就被点亮。如果 SA 和 COM0 信号不是一高一低的组合，则相应的数码管字段就不会被点亮。这种接法符合 ZLG72128 的动态扫描工作方式，并且不会影响键盘扫描管理功能。

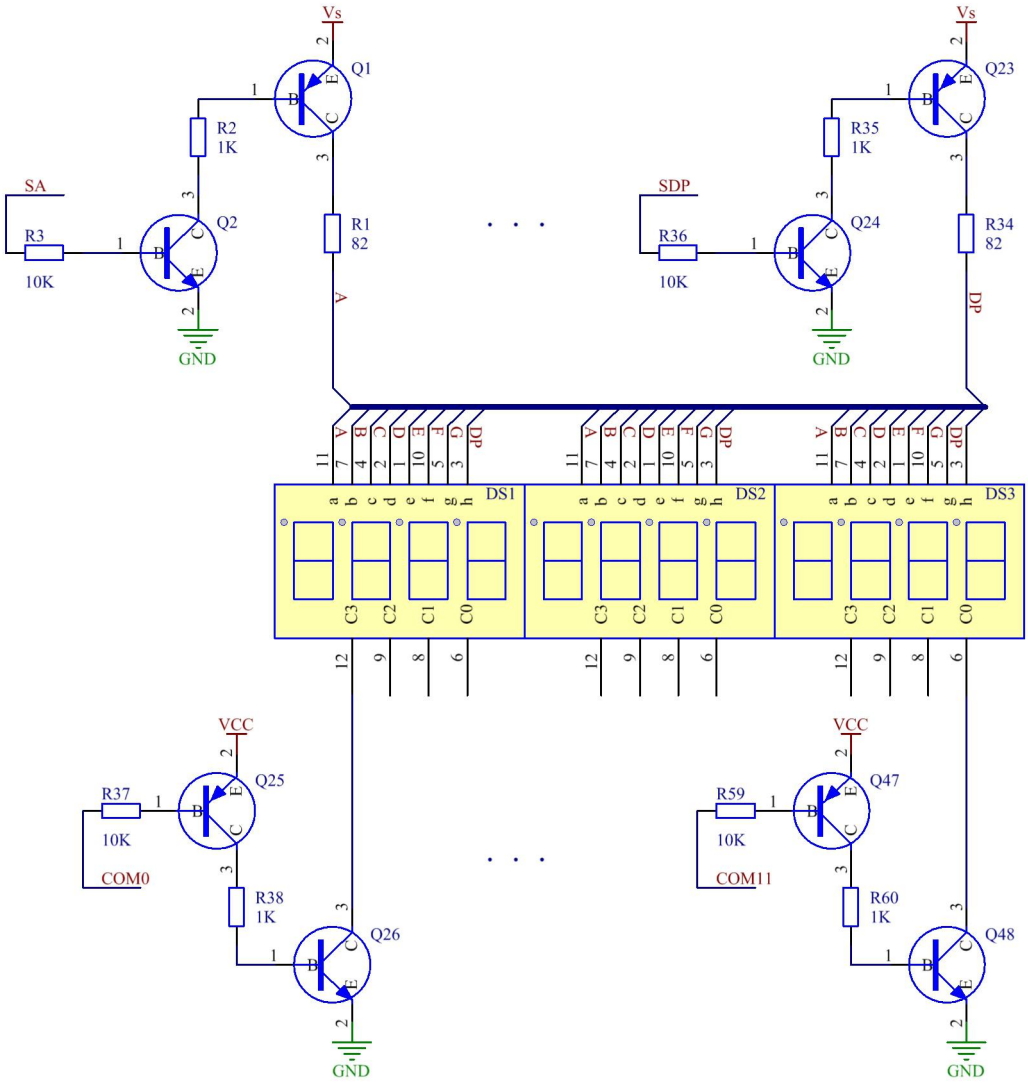


图 3.3　分立晶体管驱动大型数码管

用分立功率晶体管作为驱动电路显然太麻烦，不过我们可以选择使用功率集成电路来代替，这里就推荐一种方案，可以使用 Allegro 公司的两款达林顿芯片 UDN2981A 和 UDN2596A 进行组合，他们的驱动电流高达 1A。其中 UDN2981A 相当于中的 Q1 和 Q2 的组合，UDN2596A 相当于中的 Q25 和 Q26 的组合。ZLG72128 与它们相配合一起驱动大型数码管的完整电路如图所示。这个比分立功率晶体管的电路简单的多，图中 UDN2981A 的电源 Vs 可以是单独的高压电源（20V 以内），R5~R12 是限流电阻，阻值视具体情况而定。

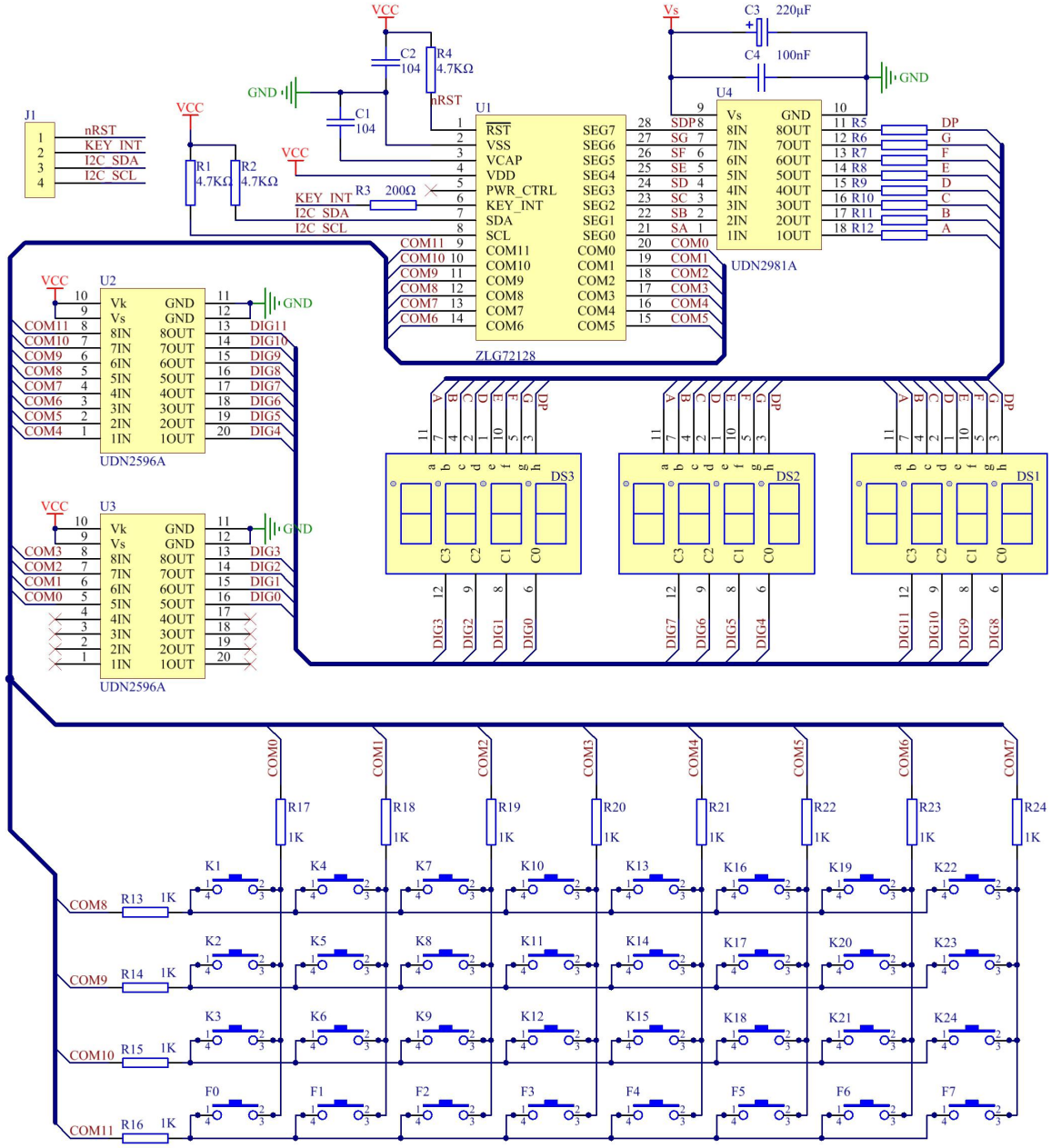


图 3.4　利用达林顿阵列驱动大型数码管

第 4 章 功能详解

4.1 功能概述

如图 3.1 所示，ZLG72128 可以扫描管理多达 32 个按键，K1～K24 为普通按键，F0～F7 为功能键。普通按键有连击检测功能。

ZLG72128 内部有 12 个显示缓冲寄存器 DispBuf0～DispBuf11，它们直接决定数码管显示的内容。ZLG72128 提供有两种显示控制方式，一种是直接向显存写入字型数据，另一种是通过向命令缓冲寄存器写入控制指令实现自动译码显示。

访问这些寄存器需要通过 I2C 总线接口来实现。ZLG72128 的 I2C 总线器件地址是 60H（写操作）和 61H（读操作）。访问内部寄存器要通过“子地址”来实现。

4.2 寄存器详解

4.2.1 系统寄存器 SystemReg（地址：00H）

系统寄存器的第 0 位（LSB）称作 KeyAvi，标志着按键是否有效，0－没有按键被按下，1－有某个按键被按下。SystemReg 寄存器的其它位暂时没有定义。当按下某个键时，ZLG72128 的 INT 引脚会产生一个低电平的中断请求信号。当读走键值后，中断信号就会自动撤销（变为高电平）。而 KeyAvi 也同时予以反映。正常情况下，微控制器只需要判断 INT 引脚就可以了。通过不断查询 KeyAvi 位也能判断是否有键按下，这样就可以节省微控制器的一根 I/O 口线，但是 I2C 总线处于频繁的活动状态，多消耗电流并且不利于抗干扰。

4.2.2 键值寄存器 Key（地址：01H）

如果某个普通键（图 3.1 中的 K1～K24）被按下，则微控制器可以从键值寄存器 Key 中读取相应的键值 1～24。如果微控制器发现 ZLG72128 的 INT 引脚产生了中断请求，而从 Key 中读到的键值是 0，则表示按下的可能是功能键。键值寄存器 Key 的值在被读走后自动变成 0。

4.2.3 连击计数器 RepeatCnt（地址：02H）

ZLG72128 为普通键（图 3.1 中的 K1～K24）提供了连击计数功能。所谓连击是指按住某个普通键不松手，经过两秒钟的延迟后，开始连续有效，连续有效间隔时间约两百毫秒。这一特性跟电脑上的键盘很类似。在微控制器能够及时响应按键中断并及时读取键值的前提下，当按住某个普通键一直不松手时：首先会产生一次中断信号，这时连击计数器 RepeatCnt 的值仍然是 0；经过一秒延迟后，会连续产生中断信号，每中断一次 RepeatCnt 就自动加 1；当 RepeatCnt 计数到 255 时就不再增加，而中断信号继续有效。在此期间，键值寄存器的值每次都会产生。

4.2.4 功能键寄存器 FunctionKey（地址：03H）

ZLG72128 提供有 8 个功能键（图 3.1 中的 F0～F7）。功能键常常是配合普通键一起使用的，就像电脑键盘上的 Shift、Ctrl 和 Alt 键。当然功能键也可以单独去使用，就像电脑键盘上的 F1～F12。当按下某个功能键时，在 INT 引脚也会像按普通键那样产生中断信号。功能键的键值是被保存在 FunctionKey 寄存器中的。功能键寄存器 FunctionKey 的初始值是 FFH，每一个位对应一个功能键，第 0 位（LSB）对应 F0，第 1 位对应 F1，依次类推，第 7 位（MSB）对应 F7。某一功能键被按下时，相应的 FunctionKey 位就清零。功能键还有一个特性就是“二次中断”，按下时产生一次中断信号，抬起时又会产生一次中断信号；而普通键只会在被按下时产生一次中断。

4.2.5 命令缓冲区 CmdBuf0 和 CmdBuf1（地址：07H 和 08H）

通过向命令缓冲区写入相关的控制命令可以实现段寻址、下载显示数据功能。详见第 4.3 节。

4.2.6 闪烁控制寄存器 FlashOnOff（地址：0BH）

FlashOnOff 寄存器决定闪烁频率和占空比。复位值为 0111,0111B。高 4 位表示闪烁时亮的持续时间，低 4 位表示闪烁时灭的持续时间。改变 FlashOnOff 的值，可以同时改变闪烁频率和占空比。FlashOnOff 取值 00H 时可获得最快的闪烁速度。

亮和灭的时间计算公式如下：

T = N×50+150ms

T 为闪烁时亮或灭的持续时间，N 为寄存器的高 4 位或者低 4 位的值，取值 0~15。最快闪烁频率为 3.33Hz（周期 300mS），最慢闪烁频率为 0.55Hz（周期 1.8S）。

特别说明：单独设置 FlashOnOff 寄存器的值，并不会看到显示闪烁，而应该配合闪烁控制命令一起使用。

4.2.7 消隐寄存器 DispCtrl0(地址：0CH)和 DispCtrl1(地址：0DH)

DispCtrl0(0CH)									DispCtrl1(0DH)							
D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	B11	B10	B9	B8		B7	B6	B5	B4	B3	B2	B1	B0

DispCtrl0、DispCtrl1 寄存器决定哪些位是否显示，对应数码管的 1～12 位。寄存器位为 1 时，对应数码管位不显示。复位值都是 0x00，即数码管的 12 个位都扫描显示。实际应用中可能需要显示的位数不足 12 位，例如只显示 8 位，这时可以把 DispCtrl0 的值设置为 0x0F，把 DispCtrl1 的值设置为 0x00，则数码管的第 0~7 位被扫描显示，而第 8～12 位不会显示。

4.2.8 闪烁寄存器 Flash0(地址：0EH)和 Flash1(地址：0FH)

Flash0(0EH)									Flash1(0FH)							
D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	B11	B10	B9	B8		B7	B6	B5	B4	B3	B2	B1	B0

Flash0、Flash 1 寄存器决定哪些位是否闪烁，对应数码管的 1～12 位。寄存器位为 1 时，对应数码管位闪烁。复位值都是 0x00，即数码管的 12 个位都不闪烁。实际应用中可能需要某些位闪烁，例如最后 2 位闪烁，这时可以把 Flash0 的值设置为 0x00，把 Flash1 的值设置为 0x03，数码管的第 1、2 位闪烁，而第 3～12 位不会闪烁。

4.2.9 显示缓冲区 DispBuf0～DispBuf11（地址：10H～1BH）

DispBuf0～DispBuf11 这 12 个寄存器的取值直接决定了数码管的显示内容。每个寄存器的 8 个位分别对应数码管的 a,b,c,d,e,f,g,dp 段。

DispBuf_n(10H~1BH)							
D7	D6	D5	D4	D3	D2	D1	D0
Dp	g	f	e	d	c	b	a

例如大写字母 H 的字型数据为 76H（不带小数点）或 F6H（带小数点）。

4.3 控制命令详解

寄存器 CmdBuf0（地址：07H）和 CmdBuf1（地址：08H）共同组成命令缓冲区。通过向

命令缓冲区写入相关的控制命令可以实现段寻址、下载显示数据、控制闪烁等功能。

4.3.1 段寻址（SegOnOff）

CmdBuf0(07H)									CmdBuf1(08H)							
D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	0	0	0	on		B3	B2	B1	B0	S3	S2	S1	S0

在段寻址命令中，12 位数码管(0~11)，每个数码管 8 个段，每一个段实际上就是一只独立的 LED。

双字节命令，在指令格式中，CmdBuf0 的高 4 位的“0001”是命令码；CmdBuf0 的最低位 on 位表示该段是否点亮，0－灭，1－亮。CmdBuf1 的 B3B2B1B0 是位地址，取值 0~11；S3S2S1S0 是 4 位段地址，取值 0～7，对应数码管的 a,b,c,d,e,f,g,dp 段。

4.3.2 下载数据并译码（Download）

CmdBuf0(07H)									CmdBuf1(08H)							
D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	A3	A2	A1	A0		dp	flash	0	d4	d3	d2	d1	d0

双字节命令，在指令格式中，CmdBuf0 的高 4 位的“0010”是命令码；A3A2A1A0 是数码管显示数据的位地址，位地址编号按从左到右的顺序依次为 0,1,2,3,……, 11；dp 控制小数点是否点亮，1 为点亮，0 为熄灭；flash 表示是否要闪烁，0－正常显示，1－闪烁；d4d3d2d1d0 是要显示的数据，包括 10 种数字和 21 种字母。显示数据按照表 4.1 中的规则进行译码：

表格 4-1 下载数据并译码命令的数据表

d4d3d2d1d0（二进制）					d4d3d2d1d0（十六进制）	显示结果
0	0	0	0	0	00H	0
0	0	0	0	1	01H	1
0	0	0	1	0	02H	2
0	0	0	1	1	03H	3
0	0	1	0	0	04H	4
0	0	1	0	1	05H	5
0	0	1	1	0	06H	6
0	0	1	1	1	07H	7
0	1	0	0	0	08H	8
0	1	0	0	1	09H	9
0	1	0	1	0	0AH	A
0	1	0	1	1	0BH	b
0	1	1	0	0	0CH	C
0	1	1	0	1	0DH	d
0	1	1	1	0	0EH	E
0	1	1	1	1	0FH	F
1	0	0	0	0	10H	G
1	0	0	0	1	11H	H
1	0	0	1	0	12H	i

1	0	0	1	1	13H	J
1	0	1	0	0	14H	L
1	0	1	0	1	15H	o
1	0	1	1	0	16H	p
1	0	1	1	1	17H	q
1	1	0	0	0	18H	r
1	1	0	0	1	19H	t
1	1	0	1	0	1AH	U
1	1	0	1	1	1BH	y
1	1	1	0	0	1CH	c
1	1	1	0	1	1DH	h
1	1	1	1	0	1EH	T
1	1	1	1	1	1FH	（无显示）

4.3.3 复位命令(Reset)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	1	0	0	0	0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“0011”是命令码。功能是将所有 LED 熄灭。

4.3.4 测试命令(Test)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	0	0	0	0	0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“0100”是命令码。功能是将所有 LED 按照 0.5S 的速率闪烁。

4.3.5 左移命令(ShiftLeft)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	1	b3	b2	b1	b0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“0101”是命令码。按照数码管位为单位，左移 n 位。左移后右边空出的位不显示任何内容，即全部 LED 熄灭状态。n 的取值范围 1~11，大于 11 的值无效，n 的值由 CmdBuf0 的低 4 位决定，按下列公式计算。

$$n = (b3 \times 8) + (b2 \times 4) + (b1 \times 2) + b0$$

4.3.6 循环左移命令(CyclicShiftLeft)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	1	0	b3	b2	b1	b0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“0110”是命令码。功能是按照数码管位为单位，循环左移 n 位。左移后右边显示从最左边移出的内容。n 的取值范围 1~11，大于 11 的值无效，n 的值由 CmdBuf0 的低 4 位决定，按下列公式计算。

$$n = (b3\times 8)+(b2\times 4)+ (b1\times 2)+ b0$$

4.3.7 右移命令(ShiftRight)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	1	1	b3	b2	b1	b0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“0111”是命令码。功能是按照数码管位为单位，右移 n 位。右移后左边空出的位不显示任何内容，即全部 LED 熄灭状态。n 的取值范围 1~11，大于 11 的值无效，n 的值由 CmdBuf0 的低 4 位决定，按下列公式计算。

$$n = (b3\times 8)+(b2\times 4)+ (b1\times 2)+ b0$$

4.3.8 循环右移命令(CyclicShiftRight)

CmdBuf0(07H)							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	b3	b2	b1	b0

单字节命令，在指令格式中，CmdBuf0 的高 4 位的“1000”是命令码。功能是按照数码管位为单位，循环右移 n 位。右移后左边显示从最右边移出的内容。n 的取值范围 1~11，大于 11 的值无效，n 的值由 CmdBuf0 的低 4 位决定，按下列公式计算。

$$n = (b3\times 8)+(b2\times 4)+ (b1\times 2)+ b0$$

第 5 章 I2C 接口

ZLG72128 的串行接口是 I2C 总线，I2C 总线用两条线（SDA 和 SCL）使芯片与芯片间传递信息。SDA 为串行数据线，SCL 为串行时钟线。这两根线引脚都是漏极开路输出结构，在实际使用中两条线都必须通过上拉电阻（R_P，Pull-Up Resistor）与电源相连。上拉电阻一般取值为 3~10KΩ，开漏结构的好处是：当总线空闲时，这两条信号线都保持高电平，几乎不消耗电流；电气兼容性好，上拉电阻接 5V 电源就能与 5V 逻辑器件接口，上拉电阻接 3V 电源又能与 3V 逻辑器件接口；因为是开漏结构，所以不同器件的 SDA 与 SDA 之间、SCL 与 SCL 之间可以直接相连，不需要额外的转换电路。

5.1 I2C 数据传输时序

5.1.1 起动（START）和停止（STOP）条件

总线空闲时，数据线和时钟线保持高电平状态。当数据线在下降沿而时钟线为高电平时为起动信号，数据线在上升沿而时钟线为高电平时为停止信号。

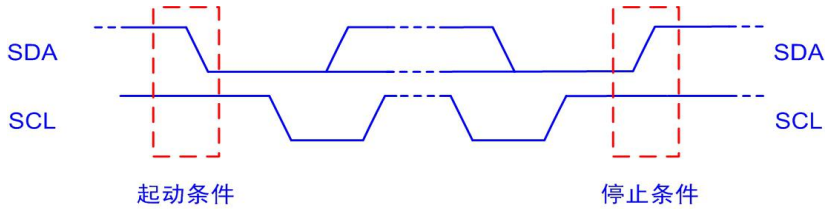


图 4.1 I²C 起动和停止条件示意图

5.1.2 位传输

每个时钟脉冲传输一个数据位，SDA 线上的数据在时钟脉冲为高电平时应保持稳定，也就是这时候的数据是有效的，否则 SDA 线上的数据将变成起动或者停止控制信号了。在时钟脉冲为低电平时允许 SDA 线上的数据变换。

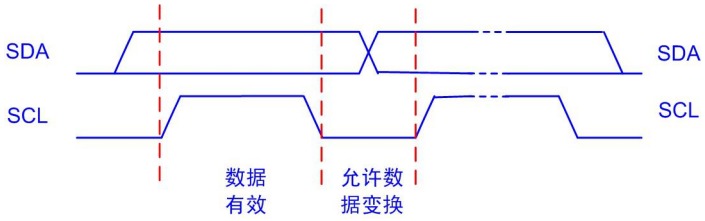


图 4.2 I²C 总线数据有效性示意图

5.1.3 应答位

在起动和停止条件之间数据传输的数量是没有限制的，每 8 位字节后加一个应答位，当然在应答位期间主设备需要附加一个时钟脉冲，当这个应答位时钟脉冲出现时，接收数据方使 SDA 数据线保持低电平，表示应答成功，这时主设备可产生停止条件。

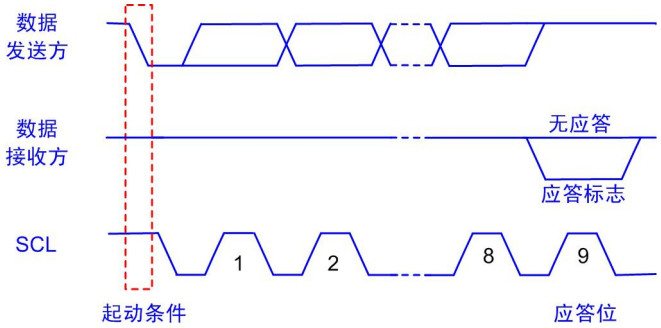


图 4.3 I²C 总线数据接收方应答标识示意图

5.2 ZLG72128 I2C 数据传输

在使用 I2C 总线传递数据时，ZLGZ72128 将作为一个从设备，这时时钟信号线 SCL 只能是输入信号线，数据信号线 SDA 是一条双向信号线。ZLGZ72128 的设备地址是 0x30，是一个 7 位地址。在起动 I2C 通信之后，设备地址与读写命令位组成一个字节进行传输，设备地址占用第一个字节的高 7 位，读写命令占最低位。当读写位为 0，表示主机将向从机写入数据，读写位为 1，表示主机将从从机读取数据。图 4.4 是从机地址与读写命令组合示意图。

图 4.4 从机地址与读写命令组合示意图

I2C 总线以字节为单位收发数据，传输到 SDA 数据线上的数据必须是 8 位，每次传输的字节数量是不受限制的。数据在传输过程中，高位在前，低位在后。每个字节之后都会跟上一个应答响应位。不过注意一下，在接收数据方接收完最后一个字节数据，或者不能再接收更多数据时，应当产生非应答信号来通知发送数据方。发送数据方发现接收数据方产生的非应答信号，则应当终止发送。

图 4.5 是主机向从机发送数据的示意图，I2C 起动后，主机首先向从机发送从机地址和写标志位，从机与自身地址匹配一致，在第 9 个时钟信号到来时，发送一个应答信号。然后主机再发送一个需要访问的从机设备子地址（I2C 设备子地址都是 8 位）的起始地址，从机应答之后，就可以发送数据，没发送完一字节数据，都需要从机进行应答。当主机向从机发送最后一个数据时，从机可能应答或者不应答，主机都可以停止数据传输。

图 4.5 主机向从机发送数据示意图

图 4.6 是主机从从机接收数据的示意图，I2C 起动后，主机首先向从机发送从机地址和写标志位，从机与自身地址匹配一致，在第 9 个时钟信号到来时，发送一个应答信号，然后主机再发送一个需要访问的从机设备子地址起始地址，从机应答之后，需要重复启起始条件，再发送一字节带有从机地址的读命令，从机应答后，就可以从从机读取数据，每次主机读取到一字节数据，就发出应答信号。在主机读取到最后一个数据时，不需要从机应答，也表明数据不再传输，主机发送停止命令停止数据传输。

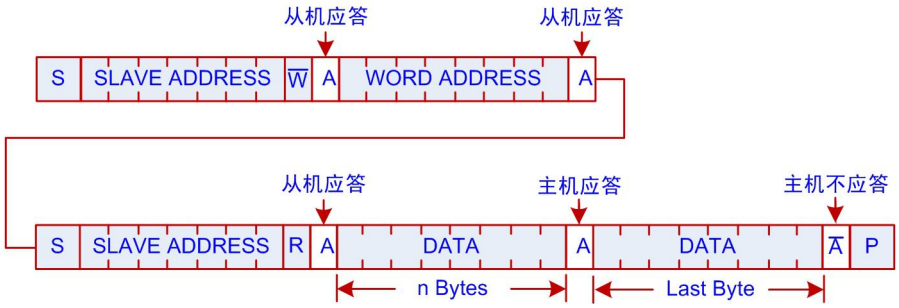


图 4.6　主机从从机接收数据示意图

第 6 章 ZLG72128 驱动程序软件包

6.1 软件包说明

ZLG72128 的驱动程序软件包是基于 LPC824 单片机来编写的，数据通信是通过 LPC824 单片机硬件 I2C 接口来实现的。软件包主要由三部分组成：I2C 底层驱动文件、I/O 中断处理文件以及对 ZLG72128 进行操作的函数实现文件。

6.1.1 I2C 驱动文件

I2C 驱动文件包括两部分：“i2c.h”和“i2c.c”，程序清单 6.1 是“i2c.h”文件中的相关内容，其主要是对从机地址、读写命令的宏定义以及 I2C 接口函数的声明。“i2c.c 源文件主要是 I2C 接口函数的具体实现。这个对于使用不同的 MCU 芯片，配置是不一样的，所以这里就不做详解。想了解该工程中 I2C 接口函数的实现的可以参考 LPC824 相关手册以及 I2C 相关协议标准。

程序清单 6.1 I2C 驱动程序头文件

```
/*********************************************************************************************************
  宏定义
*********************************************************************************************************/
#define READ                1                                   //读标志
#define WRITE               0                                   //写标志
#define SLAVE_ADDR          0x30                                //ZLG72128 I2C 7 位设备地址，不带读写标志
#define SLAVE_SIZE          0x01                                //ZLG72128 I2C 设备地址长度
#define I2C_SLV_WRITE       ((SLAVE_ADDR << 1) | WRITE)         //I2C 写命令，设备地址+写标志位
#define I2C_SLV_READ        ((SLAVE_ADDR << 1) | READ)          //I2C 读命令，设备地址+读标志位

/*********************************************************************************************************
** Function name:           I2CMSTInit
** Descriptions:            I2C 主机初始化
** input parameters:        CurrentRate  当前设置的波特率
** output parameters:       无
** Returned value:          无
*********************************************************************************************************/
extern void I2CMSTInit (uint16_t CurrentRate);

/*********************************************************************************************************
**********
** Function name:           I2CWriteByte
** Descriptions:            I2C 向从机发送单字节数据，函数内部完成 I2C 起动、停止等过程
** input parameters:        uiSubAddr    器件子地址
**                          ucData       写入数据
** output parameters:       无
** Returned value:          1            数据发送成功
**                          0            数据发送失败
*********************************************************************************************************/
extern uint8_t I2CWriteByte (uint32_t uiSubAddr, uint8_t ucData);
```

```
/*********************************************************************************************
** Function name:          I2CReadByte
** Descriptions:           I2C 从从机读取单字节数据，函数内部完成 I2C 起动、停止等过程
** input parameters:       uiSubAddr    器件子地址
**                         pucData      保存读取数据指针
** output parameters:      无
** Returned value:         1            数据读取成功
**                         0            数据读取失败
*********************************************************************************************
**********/
extern uint8_t I2CReadByte (uint32_t uiSubAddr, uint8_t *pucData);

/*********************************************************************************************
**********
** Function name:          I2CWriteNByte
** Descriptions:           I2C 向从机发送 N 字节数据，函数内部完成 I2C 起动、停止等过程
** input parameters:       uiSubAddr：  器件子地址
**                         ucData：     写入数据
**                         Length：     数据长度
** output parameters:      无
** Returned value:         1            数据发送成功
**                         0            数据发送失败
*********************************************************************************************
**********/
extern uint8_t I2CWriteNByte (uint32_t uiSubAddr, uint8_t *ucData, uint8_t Length);

/*********************************************************************************************
** Function name:          I2CReadNByte
** Descriptions:           I2C 从从机读取 N 字节数据，函数内部完成 I2C 起动、停止等过程
** input parameters:       uiSubAddr：  器件子地址
**                         pucData：    保存读出数据的指针
**                         Length       数据长度
** output parameters:      无
** Returned value:         1            数据读取成功
**                         0            数据读取失败
*********************************************************************************************/
extern uint8_t I2CReadNByte (uint32_t uiSubAddr, uint8_t *pucData, uint8_t Length);
```

6.1.2 I/O 中断定义与处理文件

中断定义与处理文件包括两部分：“gpio_int.c”和“gpio_int.h”。“gpio_int.c”主要是将外部 I/O 设置成中断功能的初始化函数以及 IO 中断函数的处理函数。“gpio_int.h”是初始化函数的声明。在初始化中，将与 ZLG72128 芯片 KEY_INT 引脚相连的单片机引脚设置成外部 I/O

下降沿中断，当有按键按下，就会产生中断，在中断处理函数中通过 I2C 读取当前键值寄存器的值。

6.1.3 ZLG72128 功能实现文件

I2C 功能实现文件包括两部分：“zlg72128.h”和“zlg72128.c”，程序清单 6.2 是“zlg72128.h”文件关于 ZLG72128 相关寄存器地址以及命令的宏定义，方便后面进行调用。程序清单 6.3 是通过调用 I2C 接口函数对 ZLG72128 进行操作的部分函数的定义，

程序清单 6.2 “zlg72128.h”文件内容

```
/*********************************************************************************************
  宏定义
*********************************************************************************************/
#define ZLG72128_SystemReg              0x00            //系统寄存器
#define ZLG72128_Key                    0x01            //键值寄存器
#define ZLG72128_RepeatCnt              0x02            //连击计数器
#define ZLG72128_FunctionKey            0x03            //功能键寄存器
#define ZLG72128_CmdBuf                 0x07            //命令缓冲区
#define ZLG72128_CmdBuf0                0x07            //命令缓冲区 0
#define ZLG72128_CmdBuf1                0x08            //命令缓冲区 1
#define ZLG72128_FlashOnOff             0x0b            //闪烁控制寄存器
#define ZLG72128_DispCtrl               0x0c            //消隐寄存器
#define ZLG72128_DispCtrl0              0x0c            //消隐寄存器 0
#define ZLG72128_DispCtrl1              0x0d            //消隐寄存器 1
#define ZLG72128_Flash                  0x0e            //闪烁寄存器
#define ZLG72128_Flash0                 0x0e            //闪烁寄存器 0
#define ZLG72128_Flash1                 0x0f            //闪烁寄存器 1
#define ZLG72128_DispBuff(n)            ((0x10) | (n))  //显示缓冲区

#define ZLG72128_SegOnOff               0x10            //段寻址命令
#define ZLG72128_Download               0x20            //下载数据并译码命令
#define ZLG72128_Rseset                 0x30            //复位命令
#define ZLG72128_Test                   0x40            //测试命令
#define ZLG72128_ShiftLeft              0x50            //左移命令
#define ZLG72128_CycShiftLeft           0x60            //循环左移命令
#define ZLG72128_ShiftRight             0x70            //右移命令
#define ZLG72128_CycShiftRight          0x80            //循环右移命令

#define ZLG72128_ShiftBytes(n)          (n)             //移位位数
```

程序清单 6.3 “zlg72128.c”文件内容

```
/*********************************************************************************************
** Function name:       ZLG72128SegOnOff
** Descriptions:        ZLG72128 段寻址操作函数，单独点亮或者熄灭数码管某一位的某一段
** input parameters:    addr：    数码管位选端  (0~11)
**                      seg：     数码管段选端  (0~7)
```

```
**                          bit：       0 熄灭
**                                      1 点亮
** output parameters:       无
** Returned value:          0   操作 ZLG72128 异常
**                          1   正常
*********************************************************************************************/
uint8_t ZLG72128SegOnOff(uint8_t addr , uint8_t seg , uint8_t bit)
{
    uint8_t cmdbuf[2];
    cmdbuf[0] = ZLG72128_SegOnOff;
    if (bit) {
        cmdbuf[0] |= 0x01;
    }
    cmdbuf[1] = (addr<<4) | seg;
    return I2CWriteNByte(ZLG72128_CmdBuf , cmdbuf , 0x02);
}

/*********************************************************************************************
** Function name:           ZLG72128Downlad
** Descriptions:            下载数据并译码
** input parameters:        addr：      数码管位选端  (0~11)
**                          dp：        0 dp 小数点熄灭
**                                      1 dp 小数点点亮
**                          flash：     0  正常显示
**                                      1  闪烁
**                          data：      显示的数据 0,1,...,9,A,...,T(0~0x1E)
** output parameters:       无
** Returned value:          0   操作 ZLG72128 异常
**                          1   正常
*********************************************************************************************/
uint8_t ZLG72128Downlad(uint8_t addr    , uint8_t dp,
                        uint8_t flash , uint8_t data)
{
    uint8_t cmdbuf[2];
    cmdbuf[0] = addr & 0x0f;
    cmdbuf[0] |= ZLG72128_Download;
    cmdbuf[1] = data & 0x1f;
    if (dp) {
        cmdbuf[1] |= 0x80;
    }
    if (flash) {
        cmdbuf[1] |= 0x40;
    }
    return I2CWriteNByte(ZLG72128_CmdBuf , cmdbuf , 0x02);
```

```
}

/*********************************************************************************************
** Function name:           ZLG72128ResetCmd
** Descriptions:            ZLG72128 复位
** input parameters:        无
**
** output parameters:       无
** Returned value:          0   操作 ZLG72128 异常
**                          1   正常
*********************************************************************************************/
uint8_t ZLG72128ResetCmd(void)
{
    return I2CWriteByte(ZLG72128_CmdBuf , ZLG72128_Rseset);
}

/*********************************************************************************************
** Function name:           ZLG72128TestCmd
** Descriptions:            ZLG72128 进入测试模式
** input parameters:        无

** output parameters:       无
** Returned value:          0   操作 ZLG72128 异常
**                          1   正常
*********************************************************************************************/
uint8_t ZLG72128TestCmd(void)
{
    return I2CWriteByte(ZLG72128_CmdBuf , ZLG72128_Test);
}

/*********************************************************************************************
** Function name:           ZLG72128ShiftCmd
** Descriptions:            数码管移位显示处理
** input parameters:        cmd：      移位命令，可以是循左移、循环左移、右移、循环右移
**                          shiftbits：  每次移位位数(1~11,大于 11 无效)
** output parameters:       无
** Returned value:          0   操作 ZLG72128 异常
**                          1   正常
*********************************************************************************************/
uint8_t ZLG72128ShiftCmd(uint8_t cmd , uint8_t shiftbits)
{
    uint8_t cmdbuf;
    cmdbuf = cmd | shiftbits;
    return I2CWriteByte(ZLG72128_CmdBuf , cmdbuf);
```

```
}
```

第 7 章 ZLG72128 演示程序

ZLG72128 演示程序“ZLG72128Demo.c”是在图 3.1 ZLG72128 典型应用电路基础上编写的，程序清单 7.1 是演示程序各个功能的具体实现。演示程序涉及数码管显示清除、通过显示缓冲区显示数据、通过下载数据并译码命令显示数据、段寻址显示数据、循环移位功能测试、消隐和闪烁功能测试以及按键与键值显示测试。该演示程序已经在 ZLG72128 测试板演示通过。当然对于更详细的 ZLG72128 测试功能和案例可参考 ZLG72128Test.c 文件以及对应的说明文档“ZLG72128 测试案例说明”。

程序清单 7.1　ZLG72128G 功能演示程序清单

```
/*********************************************************************************************
** Function name:          ClearAll
** Descriptions:           通过下载数据并译码命令清除数码管显示
** input parameters:       无
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void ClearAll(void)
{
    uint8_t x;
    for(x=0; x<12; x++)
    {
        ZLG72128Downlad(x, 0, 0 ,0x1f);
    }
}

/*********************************************************************************************
** Function name:          Test_DisBuff
** Descriptions:           通过显示缓冲区显示数码管数据
** input parameters:       无
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void Test_DispBuff(void)
{
    uint8_t DispDat[10] =
    {//0~9 字形显示
        0x3f, 0x06, 0x5b, 0x4f, 0x66,
        0x6d, 0x7d, 0x07, 0x7f, 0x6f
    };
    uint8_t x , y;
    for (x=0; x<10; x++)
    {
        //向 12 个 ZLG72128_DispBuff 寄存器写入数据
```

```
        for (y=0; y<12; y++)
        {
            I2CWriteByte(ZLG72128_DispBuff(y), DispDat[x]);
        }
        myDelay(200);
    }
}

/*********************************************************************************************
** Function name:          Test_Download
** Descriptions:           通过下载数据并译码命令显示数据
** input parameters:       无
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void Test_Download(void)
{
    uint8_t x, dp, flash, dat;
    //所有数码管显点亮
    dp = 1;
    flash = 0;
    dat = 8;
    for (x=0; x<12; x++)
    {
        ZLG72128Downlad(x, dp, flash, dat);
    }
    myDelay(500);

    //依次显示所有字形
    dp = 0;
    flash = 0;
    for (dat=0; dat<0x1f; dat++)
    {
        for (x=0; x<12; x++)
        {
            ZLG72128Downlad(x, dp, flash, dat);
        }
        myDelay(100);
    }
}

/*********************************************************************************************
** Function name:          Test_SegOnOff
** Descriptions:           通过段寻址命令进行操作
```

```
** input parameters:        无
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
void Test_SegOnOff(void)
{
    uint8_t addr, seg;
    ClearAll();
    myDelay(100);
    //点亮所有数码管
    for (addr=0; addr<12; addr++)
    {
        for (seg=0; seg<8; seg++)
        {
            ZLG72128SegOnOff(addr, seg, 1);
            myDelay(50);
        }
    }
    myDelay(500);
    //熄灭所有数码管
    for (addr=0; addr<12; addr++)
    {
        for (seg=0; seg<8; seg++)
        {
            ZLG72128SegOnOff(addr, seg, 0);
            myDelay(50);
        }
    }
}

/*********************************************************************************************
** Function name:           Test_Shift
** Descriptions:            通过移位命令进行移位测试
** input parameters:        无
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
void Test_Shift(void)
{
    uint8_t Disdat[12] =
    { //0~9、A、b 字形显示
        0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
        0x7d, 0x07, 0x7f, 0x6f, 0x77, 0x7c
    };
```

```
    uint8_t x;
    //对应位显示对应数字字形
    for (x=0; x<12; x++)
    {
        I2CWriteByte(ZLG72128_DispBuff(x), Disdat[x]);
    }
    myDelay(500);
    //循环左移测试，每次移位 1 位
    for (x=0; x<12; x++)
    {
        ZLG72128ShiftCmd(ZLG72128_CycShiftLeft, 1);
        myDelay(500);
    }
    myDelay(500);
    //循环右移测试，每次移位 1 位
    for (x=0; x<12; x++)
    {
        ZLG72128ShiftCmd(ZLG72128_CycShiftRight, 1);
        myDelay(500);
    }
}

/*********************************************************************************************
** Function name:          Test_Download
** Descriptions:           通过消隐寄存器进行消隐功能测试
** input parameters:       无
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void Test_Disp(void)
{
    uint8_t   x;
    uint8_t   dat[2];
    uint16_t addr;
    addr = 0x1;
    //数码管依次消隐
    for (x=0; x<12; x++)
    {
        addr     |= (addr<<x);
        dat[0]   = (uint8_t)((addr>>8)&0xf);
        dat[1]   = (uint8_t)(addr&0xff);
        I2CWriteNByte(ZLG72128_DispCtrl, dat, 2);
        myDelay(100);
    }
```

```
    myDelay(100);
    //数码管依次恢复
    addr = 0xfff;
    for (x=0; x<12; x++)
    {
        addr    = (addr<<1);
        dat[0]  = (uint8_t)((addr>>8)&0xf);
        dat[1]  = (uint8_t)(addr&0xff);
        I2CWriteNByte(ZLG72128_DispCtrl, dat, 2);
        myDelay(100);
    }
    myDelay(500);
}

/*********************************************************************************************
** Function name:          Test_Download
** Descriptions:           通过闪烁寄存器进行闪烁功能测试
** input parameters:       无
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void Test_Flash(void)
{
    uint8_t  x;
    uint8_t  dat[2];
    uint16_t addr;
    //数码管依次闪烁
    addr = 0x1;
    for (x=0; x<12; x++)
    {
        addr    |= (addr<<x);
        dat[0]  = (uint8_t)((addr>>8)&0xf);
        dat[1]  = (uint8_t)(addr&0xff);
        I2CWriteNByte(ZLG72128_Flash, dat, 2);
    }
    myDelay(5000);
    //数码管依次恢复
    addr = 0xfff;
    for (x=0; x<12; x++)
    {
        addr    = (addr<<1);
        dat[0]  = (uint8_t)((addr>>8)&0xf);
        dat[1]  = (uint8_t)(addr&0xff);
        I2CWriteNByte(ZLG72128_Flash, dat, 2);
```

```
    }
    myDelay(1000);
}

/*********************************************************************************************
** Function name:           Test_Key
** Descriptions:            按键按下在数码管相应位置显示键值，连击计数值
** input parameters:        无
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
void Test_Key(void)
{
    uint8_t x, y = 0, z = 10;
    //初始化 IO 中断
    GPIOIntInit();
    for (;;)
    {
        if (pQue->u32Count == (MAXSIZE)) { //IO 中断读取键值数据有效
            u8QueuePop(pQue, &KeyStatic[1]);
            u8QueuePop(pQue, &KeyStatic[2]);
            u8QueuePop(pQue, &KeyStatic[3]);
            ClearAll();
            //普通键按下显示键值
            if (KeyStatic[1]&0xff)
            {
                ZLG72128Downlad(1, 0 ,0 ,(KeyStatic[1]/10));
                ZLG72128Downlad(0, 0 ,0 ,(KeyStatic[1]%10));
            }

            //连击计数有效
            if (KeyStatic[2]&0xff)
            {
                ZLG72128Downlad(5, 0 ,0 ,(KeyStatic[2]/100));
                ZLG72128Downlad(4, 0 ,0 ,(KeyStatic[2]%100/10));
                ZLG72128Downlad(3, 0 ,0 ,(KeyStatic[2]%10));
            }

            //功能键按下显示键值砰
            if (KeyStatic[3] != 0xff)
            {
                for (x=0; x< 8; x++)
                {
                    if (!((KeyStatic[3]>>x) & 0x1))
```

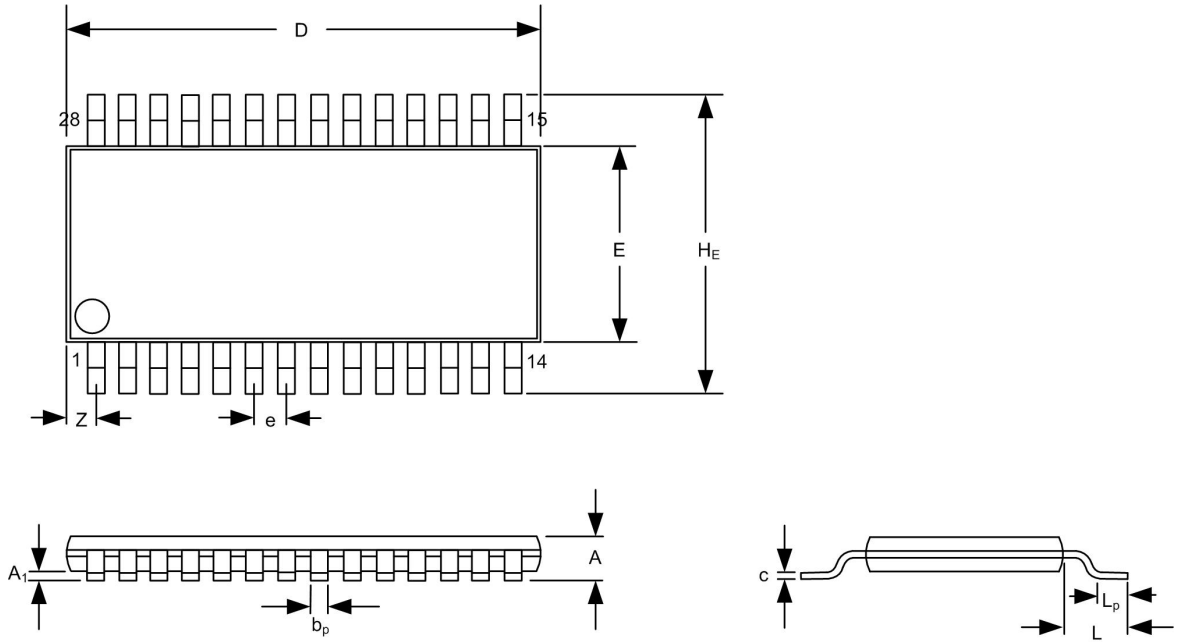
```
                        {
                            ZLG72128Downlad(y+7, 0, 0, 7-x);
                            y++;
                            if (x == z)
                            {
                                y = 0;
                            }
                            if (y == 5)
                            {
                                y = 0;
                            }
                            z = x;
                        }
                    }
                    y = 0;
                    z = 10;
                }
            }
        }
}

/*********************************************************************************************
** Function name:           main
** Descriptions:            ZLG72128 按键和数码管显示测试例程
** input parameters:        无
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
int main (void)
{
    //ZLG72128 系统初始化
    ZLG72128_Init();
    //清除数码管显示
    ClearAll();
    //通过显示缓冲区显示数码管数据
    Test_DispBuff();
    //清除数码管显示
    ClearAll();
    //通过下载数据并译码命令显示数据
    Test_Download();
    //清除数码管显示
    ClearAll();
    //通过段寻址命令进行操作
    Test_SegOnOff();
```

```
    //清除数码管显示
    ClearAll();
    //通过移位命令进行移位测试
    Test_Shift();
    //通过消隐寄存器进行消隐功能测试
    Test_Disp();
    //通过闪烁寄存器进行闪烁功能测试
    Test_Flash();
    //清除数码管显示
    ClearAll();
    //按键键值测试
    Test_Key();
    while(1);
}
```

附录 A ZLG72128 封装说明

ZLG72128 采用的是标准的 TSSOP28 封装，下图是其参数说明：



单位	D	H_E	E	Z	e	A	A_1	b_p	c	L_p	L
mm	9.8 9.6	6.6 6.2	4.5 4.3	0.8 0.5	0.65	1.1	0.15 0.05	0.30 0.19	0.2 0.1	0.75 0.50	1