

HD7279A

串行接口8位LED数码管及64键键盘智能控制芯片 (第四版)

HD7279A是一片具有串行接口的，可同时驱动8位共阴式数码管(或64只独立LED)的智能显示驱动芯片，该芯片同时还可连接多达64键的键盘矩阵，单片即可完成LED显示、键盘接口的全部功能。

HD7279A内部含有译码器，可直接接受BCD码或16进制码，并同时具有2种译码方式（参见后文），此外，还具有多种控制指令，如消隐、闪烁、左移、右移、段寻址等。

HD7279A具有片选信号，可方便地实现多于8位的显示或多于64键的键盘接口。

典型应用:

仪器仪表，工业控制器，条形显示器，控制面板

特点:

- **串行接口，无需外围元件可直接驱动LED**
- 各位独立控制译码/不译码及消隐和闪烁属性
- (循环)左移/(循环)右移指令
- 具有段寻址指令，方便控制独立LED
- 64键键盘控制器，**内含去抖动电路**
- 有DIP和SOIC两种封装形式供选择

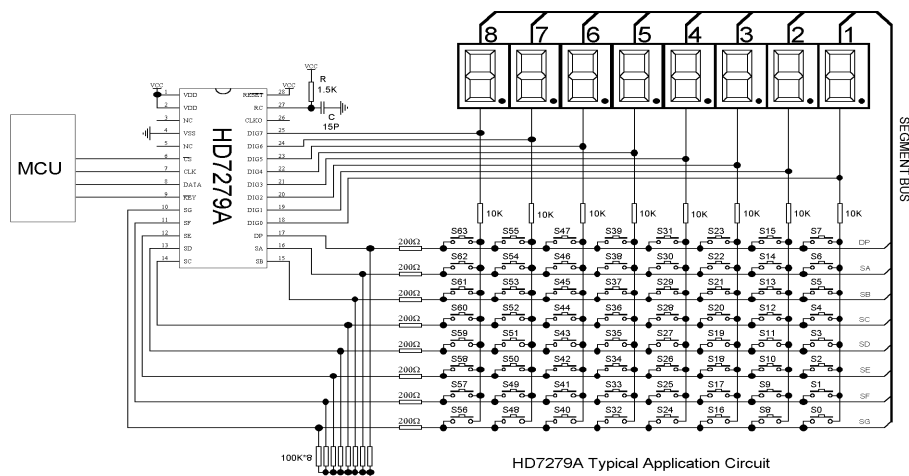
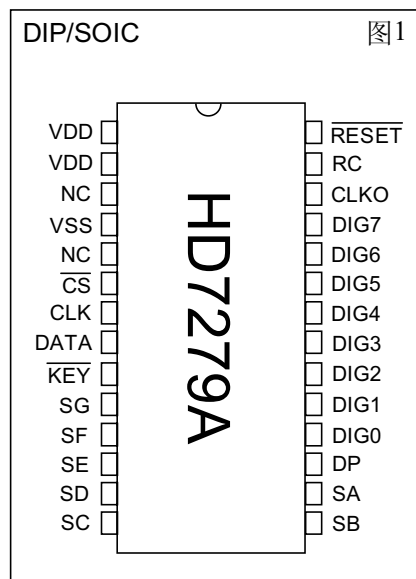


图2

电特性（VCC=5.0V,R=1.5K,C=15pF,TA=25℃）：

符号	参数	测试条件	最小	典型	最大	单位
VCC	电源电压		4.5	5.0	5.5	V
ICC	工作电流	不接LED		3	5	mA
ICC	工作电流	LED全亮，ISEG=10mA		60	100	mA
VIH	逻辑输入高电平		2.0		5.5	V
VIL	逻辑输入低电平		0		0.8	V
TKEY	按键响应时间	含去抖动时间	10	18	40	mS
IKO	KEY引脚输出电流				7	mA
IKI	KEY引脚吸入电流				10	mA
T1	从CS下降沿至CLK脉冲时		25	50	250	μ S
T2	传送指令时CLK脉冲宽度		5	8	250	μ S
T3	字节传送中CLK脉冲时间		5	8	250	μ S
T4	指令与数据时间间隔		15	25	250	μ S
T5	读键盘指令中指令与输出数据时间间隔		15	25	250	μ S
T6	输出键盘数据建立时间		5	8	-	μ S
T7	读键盘数据时CLK脉冲宽		5	8	250	μ S
T8	读键盘数据完成后DATA转为输入状态时间				5	μ S

表1



引脚说明:

引脚	名称	说明
1, 2	VDD	正电源
3, 5	NC	无连接, 必须悬空
4	VSS	接地
6	<u>CS</u>	片选输入端, 此引脚为低电平时, 可向芯片发送指令及读取键盘数据
7	<u>CLK</u>	同步时钟输入端, 向芯片发送数据及读取键盘数据时, 此引脚电平上升沿表示数据有效
8	<u>DATA</u>	串行数据输入/输出端, 当芯片接收指令时, 此引脚为输入端; 当读取键盘数据时, 此引脚在‘读’指令最后一个时钟的下降沿变为输出端
9	<u>KEY</u>	按键有效输出端, 平时为高电平, 当检测到有效按键时, 此引脚变为低电平
10-16	SG—SA	段g——段a驱动输出
17	DP	小数点驱动输出
18-25	DIG0—DIG7	数字0——数字7驱动输出
26	CLKO	振荡输出端
27	RC	RC振荡器连接端
28	RESET	复位端

表2

控制指令:

HD7279A的控制指令分为两大类——纯指令和带有数据的指令。

• 纯指令

不带数据的指令

1、复位(清除)指令A4H

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	1	0	0

当HD7279A收到该指令后, 将所有的显示清除, 所有设置的字符消隐、闪烁等属性也被一起清除。执行该指令后, 芯片所处的状态与系统上电后所处的状态一样。

2、测试指令BFH

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1	1	1	1

该指令使所有的LED全部点亮, 并处于闪烁状态, 主要用于测试。

3、左移指令A1H

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	0	1

使所有的显示自右向左(从第1位向第8位)移动一位(包括处于消隐状态的显示位)，但对各位所设置的消隐及闪烁属性不变。移动后，最右边一位为空(无显示)。例如，原显示为

4	2	5	2	L	P	3	9
---	---	---	---	---	---	---	---

其中第2位 ‘3’ 和第4位 ‘L’ 为闪烁显示，执行了左移指令后，显示变为

2	5	2	L	P	3	9	
---	---	---	---	---	---	---	--

第二位 ‘9’ 和第四位 ‘P’ 为闪烁显示。

4、右移指令A0H

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	0	0

与左移指令类似，但所做移动为自左向右(从第8位向第1位)移动，移动后，最左边一位为空。

5、循环左移指令A3H

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	1	1

与左移指令类似，不同之处在于移动后原最左边一位(第8位)的内容显示于最右位(第1位)。在上例中，执行完循环左移指令后的显示为

2	5	2	L	P	3	9	4
---	---	---	---	---	---	---	---

第二位 ‘9’ 和第四位 ‘P’ 为闪烁显示。

6、循环右移指令A2H

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	1	0

与循环左移指令类似，但移动方向相反。

• 带有数据的指令

1、下载数据且按方式0译码

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	a2	a1	a0		DP	X	X	X	d3	d2	d1	d0

X=无影响

命令由二个字节组成，前半部分为指令，其中a2，a1，a0为位地址，具体分配如下
(显示位编号请参阅典型应用电路图):

a2	a1	a0	显示位
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

固定译码方式
显示固定字符

d0—d3为数据，收到此指令时，HD7279A按以下规则(译码方式0)进行译码，如下表:

d3-d0(十六进制)	d3	d2	d1	d0	7段显示
00H	0	0	0	0	0
01H	0	0	0	1	1
02H	0	0	1	0	2
03H	0	0	1	1	3
04H	0	1	0	0	4
05H	0	1	0	1	5
06H	0	1	1	0	6
07H	0	1	1	1	7
08H	1	0	0	0	8
09H	1	0	0	1	9
0AH	1	0	1	0	-
0BH	1	0	1	1	E
0CH	1	1	0	0	H
0DH	1	1	0	1	L
0EH	1	1	1	0	P
0FH	1	1	1	1	空(无显示)

表3

小数点的显示由DP位控制，DP=1时，小数点显示，DP=0时，小数点不显示。

2、下载数据且按方式1译码

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	1	a2	a1	a0		DP	X	X	X	d3	d2	d1	d0

固定译码方式
显示固定字符

X=无影响

此指令与上一条指令基本相同，所不同的是译码方式，该指令的译码按下表进行：

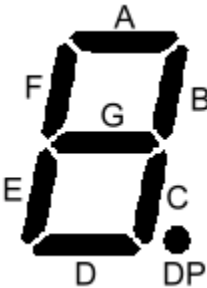
d3-d0(十六进制)	d3	d2	d1	d0	7段显示
00H	0	0	0	0	0
01H	0	0	0	1	1
02H	0	0	1	0	2
03H	0	0	1	1	3
04H	0	1	0	0	4
05H	0	1	0	1	5
06H	0	1	1	0	6
07H	0	1	1	1	7
08H	1	0	0	0	8
09H	1	0	0	1	9
0AH	1	0	1	0	A
0BH	1	0	1	1	b
0CH	1	1	0	0	C
0DH	1	1	0	1	d
0EH	1	1	1	0	E
0FH	1	1	1	1	F

表4

3、下载数据但不译码

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	a2	a1	a0		DP	A	B	C	D	E	F	G

其中，a2，a1，a0为位地址(参见‘下载数据且译码’指令)，A-G和DP为显示数据，分别对应7段LED数码管的各段。数码管各段的定义见下图。当相应的数据位为‘1’时，该段点亮，否则不亮。



软硬件灵活掌控！

硬件：不一定严格要求LED的数码段引脚对应7279的数码段引脚，可以根据硬件连线自由分配！

软件：根据硬件连线构造满足程序要求的个性字形表，通过查表找到数码管译码值，完成任意字符的显示！

4、闪烁控制88H

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	0		d8	d7	d6	d5	d4	d3	d2	d1

此命令控制各个数码管的闪烁属性。d1- d8分别对应数码管1-8，0=闪烁，1=不闪烁。开机后，缺省的状态为各位均不闪烁。

4、消隐控制98H

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	0		d8	d7	d6	d5	d4	d3	d2	d1

此命令控制各个数码管的消隐属性。d1- d8分别对应数码管1-8，1=显示，0=消隐。当某一位被赋予了消隐属性后，HD7279A在扫描时将跳过该位，因此在这种情况下无论对该位写入何值，均不会被显示，但写入的值将被保留，在将该位重新设为显示状态后，最后一次写入的数据将被显示出来。当无需用到全部8个数码管显示的时候，将不用的位设为消隐属性，可以提高显示的亮度。

注意：至少应有一位保持显示状态，如果消隐控制指令中d1- d8全部为0，该指令将不被接受，HD7279A保持原来的消隐状态不变。

5、段点亮指令E0H

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	0	0	0		X	X	d6	d5	d4	d3	d2	d1

此为段寻址指令，作用为点亮数码管中某一指定的段，或LED矩阵中某一指定的LED。指令中，X=无影响；d0- d5为段地址，范围从00H—3FH，具体分配为：第1个数码管的G段地址为00H，F段为01H，……A段为06H，小数点DP为07H，第2个数码管的G段为08H，F段为09H，……，依此类推直至第8个数码管的小数点DP地址为3FH。

6、段关闭指令C0H

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	0	0	0		X	X	d5	d4	d3	d2	d1	d0

段寻址命令，作用为关闭(熄灭)数码管中的某一段，指令结构与‘段点亮指令’相同，请参阅上文。

7、读键盘数据指令15H

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	0	1	0	1		d7	d6	d5	d4	d3	d2	d1	d0

该指令从HD7279A读出当前的按键代码。与其它指令不同，此命令的前一个字节00010101B为微控制器传送到HD7279A的指令，而后一个字节d0- d7则为HD7279A返回的按键代码，其范围是0-3FH(无键按下时为0xFF)，各键键盘代码的定义，请参阅图2及典型应用电路图，图中的键号即键盘代码。

此指令的前半段，HD7279A的DATA引脚处于高阻输入状态，以接受来自微处理器的指令；在指令的后半段，DATA引脚从输入状态转为输出状态，输出键盘代码的值。故微处理器连接到DATA引脚的I/O口应有一从输出态到输入态的转换过程，详情请参阅本文‘串行接口’一节的内容。

当HD7279A检测到有效的按键时，KEY引脚从高电平变为低电平，并一直保持到按键结束。在此期间，如果HD7279A接收到‘读键盘数据指令’，则输出当前按键的键盘代码；如果在收到‘读键盘指令’时没有有效按键，HD7279A将输出FFH (11111111B)。

串行接口

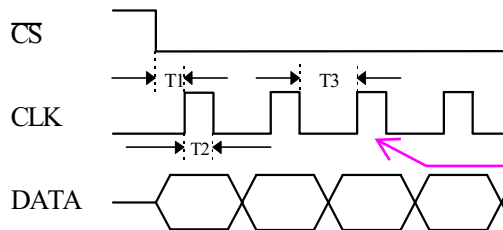
HD7279A采用串行方式与微处理器通讯，串行数据从DATA引脚送入芯片，并由CLK端同步。当片选信号变为低电平后，DATA引脚上的数据在CLK引脚的上升沿被写入HD7279A的缓冲寄存器。

HD7279A的指令结构有三种类型：1、不带数据的纯指令，指令的宽度为8个BIT，即微处理器需发送8个CLK脉冲。2、带有数据的指令，宽度为16个BIT，即微处理器需发送16个CLK脉冲。3、读取键盘数据指令，宽度为16个BIT，前8个为微处理器发送到HD7279A的指令，后8个BIT为HD7279A返回的键盘代码。执行此指令时，HD7279A的DATA端在第9个CLK脉冲的上升沿变为输出状态，并与第16个脉冲的下降沿恢复为输入状态，等待接收下一个指令。

串行接口的时序如下图：

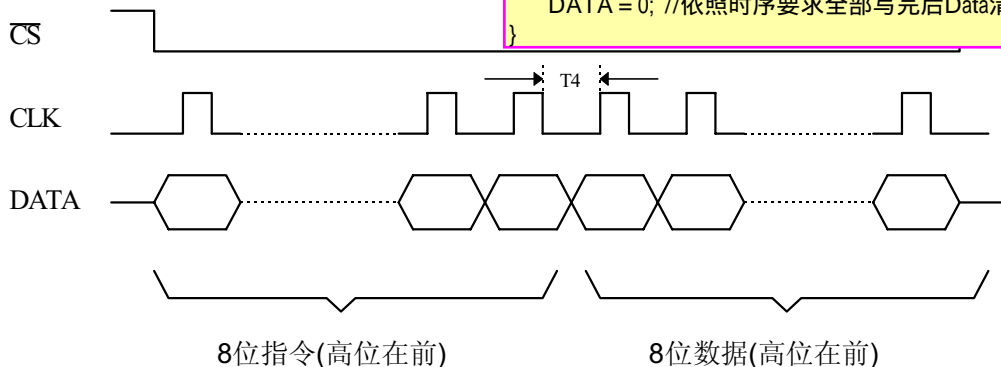
1、纯指令

```
// ***** 向 HD7279 中写数据 *****  
// ***** 输入：cmd 指令, dat 数据 *****  
  
void write_7279(unsigned char cmd, unsigned char dat)  
{  
    CS = 0;  
    Somenop50();Somenop();  
    send_byte(cmd);    // 写指令  
    Somenop25();Somenop();  
    send_byte(dat);    // 写数据  
    Somenop();Somenop();  
    CS = 1;  
}
```

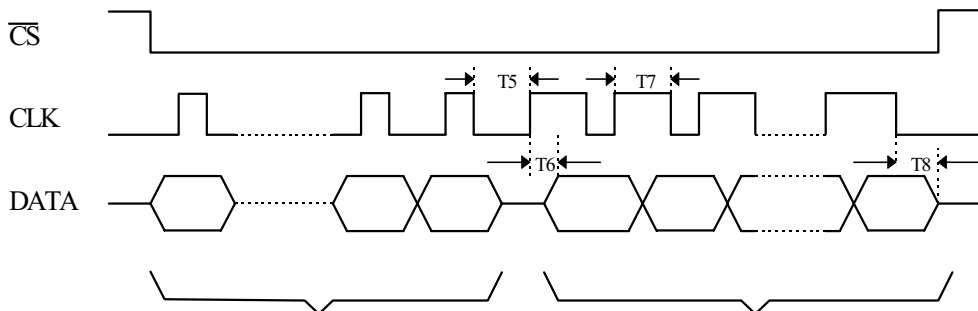



```
void send_byte(unsigned char cmd)
{
    unsigned char i;
    Somenop50();Somenop50(); //时序初始延时
    for(i = 0; i < 8; i++)
    {
        //写cmd的最高位,
        if(cmd & 0x80) //最高位为1则写"1"
            DATA = 1;
        else //最高位为0写"0"
            DATA = 0;
        //CLK形成下降沿,先1后0,注意延时时间要满足时序要求
        CLK = 1; Somenop10();Somenop10();
        CLK = 0; Somenop10();Somenop10();
        //命令行左移,为写下一个位做准备
        cmd = cmd << 1;
    }
    DATA = 0; //依照时序要求全部写完后Data清零
}
```

2、带数据指令



3、读键盘指令

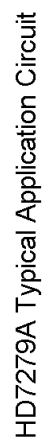


读键盘指令(8位, 高位在前)

```
unsigned char ReadKey(void)
{
    unsigned char readkey;
    CS = 0;
    Somenop50();Somenop50();
    send_byte(0x15); //读键值命令15H
    Somenop25();Somenop25();
    readkey = receive_byte();
    CS = 1;
    return(readkey); //返回键值
}
```

```
unsigned char receive_byte(void)
{
    unsigned char i, in_byte;
    DATA=1;
    Somenop50();Somenop50();
    for (i=0;i<8;i++)
    {
        CLK=1;Somenop10();Somenop10();
        //左移一位, 空出最后一位存放新进来的位DATA
        in_byte=in_byte*2;
        //若新进来的位DATA为1, 则in_byte的末位置1, 否则不需要变
        if (DATA)
            in_byte=in_byte|0x01;
        CLK=0; Somenop10();Somenop10();
    }
    DATA=0;
    return (in_byte); //返回接收值
}
```

HD7279的典型应用图如下:



HD7279A应连接共阴式数码管。应用中，无需用到的键盘和数码管可以不连接，省去数码管或对数码管设置消隐属性均不会影响键盘的使用。

如果不用键盘，则典型电路图中连接到键盘的8只10K电阻和8只100K下拉电阻均可以省去。如果使用了键盘，则电路中的8只100K下拉电阻均不得省略。除非不接入数码管，否则串入DP及SA—SG连线的8只200Ω电阻均不能省去。

实际应用中8只下拉电阻和8只键盘连接位选线DIG0-DIG7的8只电阻（以下简称位选电阻），应遵从一定的比例关系，下拉电阻应大于位选电阻的5倍而小于其50倍，典型值为10倍；下拉电阻的取值范围是10K-100K，位选电阻的取值范围是1K-10K。在不影响显示的前提下，下拉电阻应尽可能地取较小的值，这样可以提高键盘部分的抗干扰能力。

因为采用循环扫描的工作方式，如果采用普通的数码管，亮度有可能不够，采用高亮或超高亮的型号，可以解决这个问题。数码管的尺寸，亦不宜选得过大，一般字符高度不宜超过1英寸，如使用大型的数码管，应使用适当的驱动电路。

HD7279A需要一外接的RC振荡电路以供系统工作，其典型值分别为 $R=1.5K\Omega$ ， $C=15pF$ 。如果芯片无法正常工作，请首先检查此振荡电路。在印刷电路板布线时，所有元件，尤其是振荡电路的元件应尽量靠近HD7279A，并尽量使电路连线最短。

HD7279A的RESET复位端在一般应用情况下，可以直接与正电源连接，在需要较高可靠性的情况下，可以连接一外部的复位电路，或直接由MCU（单片机）控制。在上电或RESET端由低电平变为高电平后，HD7279A大约需要经过18-25MS的时间才会进入正常工作状态。

上电后，所有的显示均为空，所有显示位的显示属性均为‘显示’及‘不闪烁’。当有键按下时，KEY引脚输出变为低电平，此时如果接收到‘读键盘’指令，HD7279A将输出所按下键的代码。键盘代码的定义，请参阅图2及典型应用电路图，图中的键号即键盘代码，图中代码以10进制表示。如果在没有按键的情况下收到‘读键盘’指令，HD7279A将输出FFH(255)。

程序中，尽可能地减少CPU对HD7279A的访问次数，可以使得程序更有效率。

因为芯片直接驱动LED数码管显示，电流较大，且为动态扫描方式，故如果该部分电路电源连线较细较长，可能会引入较大的电源噪声干扰，将HD7279A的正负电源端上并入去耦电容可以提高电路的抗干扰能力。

注意：如果有2个键同时按下，HD7279A将只能给出其中一个键的代码，因此HD7279A不适于应用在需要2个或2个以上键同时按下的场合。

接口程序示例

下面给出ATMEL公司的AT89C2051及MICROCHIP公司的PIC16C54与HD7279A连

接的应用实例，2个程序所完成的功能相同，均为等待键盘输入，然后将所读到的键盘码转换成10进制后，送回HD7279A显示，同时将前面的显示内容左移。

1、AT89C2051接口程序

硬件连接如图，AT89C2051所用时钟频率为12MHz，程序使用Franklin A51编译通过，并经过验证。程序中延时时间以HD7279A外接R=1.5K, C=15pF为准，如使用不同的CPU时钟频率或不同的R/C参数，请注意调整延时时间。

```
$ title (HD7279A Test Program)
$ DB

;*****
; RAM定义
;*****

BIT_COUNT      DATA    07FH
TIMER           DATA    07EH
TIMER1          DATA    07DH
TEN             DATA    07CH
DATA_IN         DATA    020H
DATA_OUT        DATA    021H

;*****
; I/O口定义
;*****

DAT             BIT      P1.2           ;HD7279A的DATA连接于P1.2
KEY             BIT      P1.3           ;HD7279A的KEY连接于P1.3
CS              BIT      P1.4           ;HD7279A的CS连接于P1.4
CLK             BIT      P1.5           ;HD7279A的CLK连接于P1.5

                ORG      000H
                JMP      START

                ORG      100H

START:          MOV      SP,#2FH         ;定义堆栈
                MOV      P1,#11011011B  ;I/O口初始化
                MOV      TIMER,#50       ;延时约25MS
```

```
graph LR
    subgraph AT89C2051
        P1_2[P1.2]
        P1_3[P1.3]
        P1_4[P1.4]
        P1_5[P1.5]
    end
    subgraph HD7279A
        DATA
        KEY
        CS
        CLK
    end
    P1_2 --- DATA
    P1_3 --- KEY
    P1_4 --- CS
    P1_5 --- CLK
```

```

START_DELAY:  MOV     TIMER1,#255
START_DELAY1: DJNZ    TIMER1,START_DELAY1
                DJNZ    TIMER,START_DELAY
                MOV     DATA_OUT,#10100100B ;发复位(清除)指令
                CALL    SEND
                SETB    CS                      ;恢复CS为高电平
MAIN:          JB      KEY,MAIN                  ;检测是否有键按下
                MOV     DATA_OUT,#00010101B ;有键按下,发送读键盘指令
                CALL    SEND
                CALL    RECEIVE
                SETB    CS                      ;设CS为高电平
                MOV     B,#10                   ;16进制——BCD码转换
                MOV     A,DATA_IN
                DIV     AB
                MOV     TEN,A
                MOV     DATA_OUT,#10100001B ;发2次左移指令,使当前显示
                                                ;内容左移,留出空位供显示新
                                                ;数据
                CALL    SEND                    ;发送指令到HD7279A
                MOV     DATA_OUT,#10100001B
                CALL    SEND
                MOV     DATA_OUT,#10000001B ;下载数据且译码指令(第2位)
                CALL    SEND
                MOV     DATA_OUT,TEN          ;发送十位数字到HD7279A

                CALL    SEND
                MOV     DATA_OUT,#10000000B ;下载数据且译码指令(第1
                                                ;位)
                CALL    SEND
                MOV     DATA_OUT,B            ;发送个位数字到HD7279A
                CALL    SEND
                SETB    CS
WAIT:          JNB     KEY,WAIT                ;等待按键放开
                JMP     MAIN

```

;*****

; 发送1个字节到HD7279,高位在前

;*****

```

SEND:          MOV     BIT_COUNT,#8           ;设定位计数器=8
                CLR     CS                     ;设CS为低电平
                CALL    LONG_DELAY            ;长延时

```

```

SEND_LOOP:    MOV     C,DATA_OUT.7           ;输出1位
              MOV     DAT,C
              SETB    CLK                     ;设CLK为高电平
              MOV     A,DATA_OUT             ;待发送数据左移
              RL      A
              MOV     DATA_OUT,A
              CALL    SHORT_DELAY            ;短延时
              CLR     CLK                     ;设CLK为低电平
              CALL    SHORT_DELAY            ;短延时
              DJNZ    BIT_COUNT,SEND_LOOP    ;检查是否8位均发送完毕
              CLR     DAT                     ;发送完毕, 返回
              RET

; *****
; 从HD7279接收一个字节, 高位在前
; *****
RECEIVE:      MOV     BIT_COUNT,#8           ;设定位计数器=8
              SETB    DAT                     ;设P1.2(DATA)口为高电平
                                              ;(输入状态)
              CALL    LONG_DELAY             ;长延时
RECEIVE_LOOP: SETB    CLK                     ;置CLK为高电平
              CALL    SHORT_DELAY            ;短延时
              MOV     A,DATA_IN              ;数据左移
              RL      A
              MOV     DATA_IN,A
              MOV     C,DAT                  ;读取一位数据
              MOV     DATA_IN.0,C
              CLR     CLK                     ;置CLK为低电平
              CALL    SHORT_DELAY            ;短延时
              DJNZ    BIT_COUNT,RECEIVE_LOOP;是否已接收8位数据
              CLR     DAT                     ;重设DAT口为低电平(输出状态)
              RET

; *****
; 延时子程序
; *****
LONG_DELAY:   MOV     TIMER,#25              ;设定延时时间为约50uS
DELAY_LOOP:   DJNZ    TIMER,DELAY_LOOP
              RET
SHORT_DELAY:  MOV     TIMER,#4               ;设定延时时间为约8uS
SHORT_LP:     DJNZ    TIMER,SHORT_LP

```

RET

END

2、PIC16C54接口程序

硬件连接如图，PIC16C54所用时钟频率4MHz。程序使用MICROCHIP公司的MPASM编译程序编译通过，并经过验证。程序中延时时间以HD7279A外接R=1.5K, C=15pF为准，如使用不同的CPU时钟频率或R/C参数，请注意调整延时时间。

```
TITLE          "HD7279A TEST"
LIST           P=16C54
INCLUDE        P16C5X.INC

; *****
; 寄存器定义
; *****

BIT_COUNT      SET          0X07
DATA_OUT        SET          0X08
DATA_IN         SET          0X09
TEN             SET          0X0A
TIMER           SET          0X0B
TIMER1          SET          0X0C

; *****
; I/O口定义
; *****

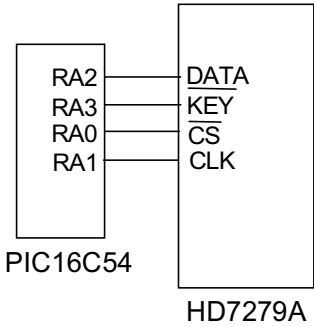
CS              EQU          0           ;CS连接于16C54的RA0
CLK             EQU          1           ;CLK连接于16C65的RA1
DAT             EQU          2           ;DAT连接于16C54的RA2
KEY            EQU          3           ;KEY连接于16C54的RA3

                ORG          0x1FF
                GOTO         START

                ORG          0X00

; *****
; 延时子程序
; *****

LONG_DELAY      MOVLW        D'16'      ;设定延时时间为约50uS
                MOVWF        TIMER
```



The diagram illustrates the hardware connection between a PIC16C54 microcontroller and an HD7279A display driver. The PIC16C54 is represented by a box on the left with pins RA2, RA3, RA0, and RA1. The HD7279A is represented by a box on the right with pins DATA, KEY, CS, and CLK. The connections are as follows: RA2 is connected to DATA, RA3 is connected to KEY, RA0 is connected to CS, and RA1 is connected to CLK.

```

DELAY_LOOP    DECFSZ      TIMER
              GOTO        DELAY_LOOP
              RETLW       0

SHORT_DELAY   MOVLW       D3           ;设定延时时间为约8uS
              MOVWF       TIMER
SHORT_LP      DECFSZ      TIMER
              GOTO        SHORT_LP
              RETLW       0

; *****
; 发送1个字节到HD7279，高位在前
; *****

SEND          MOVWF       DATA_OUT    ;待发送数据存入DATA_OUT
              MOVLW       D'8'
              MOVWF       BIT_COUNT    ;设定位计数器=8
              BCF         PORTA,CS     ;设CS为低电平
              CALL        LONG_DELAY   ;长延时
SEND_LOOP     BCF         STATUS,C
              RLF         DATA_OUT    ;输出1位
              BCF         PORTA,DAT
              BTFSC       STATUS,C
              BSF         PORTA,DAT
              BSF         PORTA,CLK    ;设CLK为高电平
              CALL        SHORT_DELAY  ;短延时
              BCF         PORTA,CLK    ;设CLK为低电平
              CALL        SHORT_DELAY
              DECFSZ      BIT_COUNT    ;检查是否8位均发送完毕
              GOTO        SEND_LOOP    ;未发送完，发送下一位
              BCF         PORTA,DAT
              RETLW       0           ;发送完毕，返回

; *****
; 从HD7279接收一个字节，高位在前
; *****

RECEIVE       MOVLW       D'8'
              MOVWF       BIT_COUNT    ;设定位计数器=8
              MOVLW       B'11111100' ;设RA2(DATA)口为输入状态
              TRIS        PORTA
              CALL        LONG_DELAY   ;长延时
RECEIVE_LOOP  BSF         PORTA,CLK    ;置CLK为高电平
              CALL        SHORT_DELAY  ;短延时

```



```

        BSF          STATUS,C
        BTFSS        PORTA,DAT
        BCF          STATUS,C
        RLF          DATA_IN      ;读取一位数据
        BCF          PORTA,CLK     ;置CLK为低电平
        CALL         SHORT_DELAY
        DECFSZ       BIT_COUNT     ;是否已接收8位数据
        GOTO         RECEIVE_LOOP
        MOVLW        B'11111000'  ;重新设RA2(DATA)口为输出态
        TRIS         PORTA
        RETLW        0

; *****
; 初始化
; *****
START      MOVLW      B'11111000'    ;I/O口初始化
           TRIS       PORTA
           MOVLW      B'11111001'
           MOVWF      PORTA
           MOVLW      0X19           ;延时约25MS
           MOVWF      TIMER
START_DELAY MOVLW      0XFF
           MOVWF      TIMER1
START_DELAY1 DECFSZ    TIMER1
           GOTO       START_DELAY1
           DECFSZ    TIMER
           GOTO       START_DELAY
           MOVLW      B'10100100'    ;发复位(清除)指令
           CALL       SEND
           BSF        PORTA,CS       ;恢复CS为高电平
; *****
; 主程序
; *****
MAIN      BTFSC      PORTA,KEY      ;检测是否有键按下
           GOTO       MAIN
           MOVLW      B'00010101'    ;有键按下,发送读键盘指令
           CALL       SEND           ;发送读键盘指令
           CALL       RECEIVE        ;从HD7279A读键盘代码
           BSF        PORTA,CS       ;设CS为高电平
; *****

```

; 16进制——BCD码转换

; *****

	CLRF	TEN
GET_DEC	MOVLW	D'10'
	SUBWF	DATA_IN,W
	SKPC	
	GOTO	OVER
	MOVWF	DATA_IN
	INCF	TEN
	GOTO	GET_DEC

; *****

;发送按键的BCD码到HD7279

; *****

;发2次左移指令，使当前显示内容左移，

;留出空位供显示新数据

OVER	MOVLW	B'10100001'	;左移指令
	CALL	SEND	;发送指令到HD7279A
	MOVLW	B'10100001'	;左移指令
	CALL	SEND	;发送指令到HD7279A
	MOVLW	B'10000001'	;下载数据且译码指令(第2位)
	CALL	SEND	;发送指令到HD7279A
	MOVFW	TEN	
	CALL	SEND	;发送十位数字到HD7279A
	MOVLW	B'10000000'	;下载数据且译码指令(第1位)
	CALL	SEND	;发送指令到HD7279A
	MOVFW	DATA_IN	
	CALL	SEND	;发送个位数字到HD7279A
	BSF	PORTA,CS	
WAIT	BTFSS	PORTA,KEY	;等待按键放开
	GOTO	WAIT	
	GOTO	MAIN	

END

附录一：用于MCS51的C语言例子程序

这是用于HD7279A评估板的C51程序,使用C语言编程的用户可以参考

```
#include <reg51.h>
```

```
/** 函数定义 **/
```

```
void long_delay(void);           // 长延时
void short_delay(void);          // 短暂延时
void delay10ms(unsigned char);   // 延时10MS
void write7279(unsigned char, unsigned char); // 写入到HD7279
unsigned char read7279(unsigned char); // 从HD7279读出
void send_byte(unsigned char);    // 发送一个字节
unsigned char receive_byte(void); // 接收一个字节
```

```
/** 变量及I/O口定义 **/
```

```
unsigned char digit[5];
unsigned char key_number, j, k;
unsigned int tmr;
unsigned long wait_cnter;
sbit cs=P1^4;           // cs at P1.4
sbit clk=P1^5;          // clk 连接于 P1.5
sbit dat=P1^2;          // dat 连接于 P1.2
sbit key=P1^3;          // key 连接于 P1.3
```

```
/** HD7279A 指令 **/
```

```
#define CMD_RESET 0xa4
#define CMD_TEST 0xbf
#define DECODE0 0x80
#define DECODE1 0xc8
#define CMD_READ 0x15
#define UNDECODE 0x90
#define RTL_CYCLE 0xa3
#define RTR_CYCLE 0xa2
#define RTL_UNCYL 0xa1
#define RTR_UNCYL 0xa0
#define ACTCTL 0x98
#define SEGON 0xe0
#define SEGOFF 0xc0
#define BLINKCTL 0x88
```

```
/** 主程序 **/
```

```
main()
```

```

{
    while (1)
    {
        for (tmr=0;tmr<0x2000;tmr++);        // 上电延时
        send_byte(CMD_RESET);                // 复位HD7279A
//*****
//          测试指令演示
//*****
        send_byte(CMD_TEST);                // 测试指令
        for (j=0;j<3;j++)                    // 延时约3秒
        {
            delay10ms(100);
        }
        send_byte(CMD_RESET);                // 清除显示

//*****
//          闪烁指令及键盘接口测试
// 将用户按键的键码显示出来，如果10秒内无按键
// 或按s0键即进入下一步演示
//*****
        wait_cnter=0;
        key_number=0xff;
        write7279(BLINKCTL,0xfc); // 第1、2两位设为闪烁显示
        write7279(UNDECODE,0x08); // 在第1位显示下划线 '_'
        write7279(UNDECODE+1,0x08); // 在第2位显示下划线 '_'
        do
        {
            if (!key)                    // 如果有键按下
            {
                key_number=read7279(CMD_READ);
// 读出键码
                write7279(DECODE1+1,key_number/16);
// 在第2位显示键码高8位
                write7279(DECODE1,key_number&0x0f);
// 在第1位显示键码低8位
                while (!key);            // 等待按键放开
                wait_cnter=0;
            }
            wait_cnter++;
        } while (key_number!=0 && wait_cnter<0x30000);
// 如果按键为 '0' 和超时则进入下一步演示

```

```

write7279(BLINKCTL,0xff); // 清除闪烁设置

//*****
//          快速计数演示
//*****
for (j=0;j<5;j++)          // 计数初始值为00000
{
    digit[j]=0;
    write7279(DECODE0+j,digit[j]);
}
while (digit[4]<2)          // 如果计数达到20000就停止
{
    digit[0]++;
    if (digit[0]>9)
    {
        digit[0]=0;
        digit[1]++;
        if (digit[1]>9)
        {
            digit[1]=0;
            digit[2]++;
            if (digit[2]>9)
            {
                digit[2]=0;
                digit[3]++;
                if (digit[3]>9)
                {
                    digit[3]=0;
                    digit[4]++;
                    if (digit[4]>9)
                    {
                        digit[4]=0;
                    }
                }
            }
        }
    }
}
write7279(DECODE0,digit[0]);
if (digit[0]==0)
{
    write7279(DECODE0+1,digit[1]);

```

```

        if (digit[1]==0)
        {
            write7279(DECODE0+2,digit[2]);
            if (digit[2]==0)
            {
                write7279(DECODE0+3,digit[3]);
                if (digit[3]==0)
                {
                    write7279(DECODE0+4,digit[4]);
                }
            }
        }
    }
    delay10ms(150);
    send_byte(CMD_RESET);          // 清除显示

//*****
//          下载数据但不译码指令测试
//*****
        write7279(UNDECODE+7,0x49);
    // 在第8位按不译码方式显示一字符'三'
        delay10ms(80);

//*****
//          循环左/右移测试
//          "三"字向右运动3次,再向左运动3次
//*****
        for (j=0;j<23;j++)
        {
            send_byte(RTR_CYCLE);          // 循环右移23次
            delay10ms(12);
        }
        for (j=0;j<23;j++)
        {
            send_byte(RTL_CYCLE);          // 循环左移23次
            delay10ms(12);
        }

//*****
//          译码方式0及左移指令测试

```

```

//*****
    for (j=0;j<16;j++)
    {
        send_byte(RTL_UNCYL);          // 不循环左移指令
        write7279(DECODE0,j);
// 译码方式0指令, 显示在第1位
        delay10ms(50);
    }
    delay10ms(150);
    send_byte(CMD_RESET);

//*****
//      译码方式1及右移指令测试
//*****
    for (j=0;j<16;j++)
    {
        send_byte(RTR_UNCYL);          // 不循环左移指令
        write7279(DECODE1+7,j);
// 译码方式0指令, 显示在第8位
        delay10ms(50);
    }
    delay10ms(150);

//*****
//      消隐指令测试
//*****
    k=0xff;
    for (j=0;j<6;j++)
    {
        k=k/2;
        write7279(ACTCTL,k); // 每隔一秒钟增加一个消隐位
        delay10ms(100);
    }
    write7279(ACTCTL,0xff); // 恢复8位显示
    delay10ms(100);
    send_byte(CMD_RESET); // 清除显示

//*****
//      段点亮指令和段关闭指令
//*****
    for (j=0;j<64;j++)

```

```

        {
            write7279(SEGON,j);        // 将64个显示段逐个点亮
            write7279(SEGOFF,j-1);    // 同时将前一个显示段关闭
            delay10ms(20);
        }
    }

void write7279(unsigned char cmd, unsigned char dta)
{
    send_byte (cmd);
    send_byte (dta);
}

unsigned char read7279(unsigned char command)
{
    send_byte(command);
    return(receive_byte());
}

void send_byte(    unsigned char out_byte)
{
    unsigned char i;
    cs=0;
    long_delay();
    for (i=0;i<8;i++)
    {
        if (out_byte&0x80)
        {
            dat=1;
        }
        else
        {
            dat=0;
        }
        clk=1;
        short_delay();
        clk=0;
        short_delay();
        out_byte=out_byte*2;
    }
}

```



```

        dat=0;
    }

unsigned char receive_byte(void)
{
    unsigned char i, in_byte;
    dat=1;                                // 设为输入状态
    long_delay();
    for (i=0;i<8;i++)
    {
        clk=1;
        short_delay();
        in_byte=in_byte*2;
        if (dat)
        {
            in_byte=in_byte|0x01;
        }
        clk=0;
        short_delay();
    }
    dat=0;
    return (in_byte);
}

void long_delay(void)
{
    unsigned char i;
    for (i=0;i<0x30;i++);
}

void short_delay(void)
{
    unsigned char i;
    for (i=0;i<8;i++);
}

// ***** 延时 n*10ms *****
void delay10ms(unsigned char time)
{
    unsigned char i;
    unsigned int j;

```

```
for (i=0;i<time;i++)
{
    for(j=0;j<0x390;j++)
    {
        if (!key)
        {
            key_int();
        }
    }
}
}
```

附录二：

HD7279A评估板

专为评估HD7279(A)的性能和辅助调试用户电路、程序而设计，即可向您全面演示HD7279(A)的各项功能，同时也是您在使用HD7279(A)过程中调试电路及程序的得力助手。主要特点包括：

- ★ 带有标准电路及演示程序，通电即可全面演示HD7279A的各项功能
- ★ 带有外部CPU接口，可借板上的电路调试用户程序，也可用板上的程序测试用户电路
- ★ 自带64键键盘(板上仅预装16只按键)
- ★ CPU采用AT89C2051,用户可随时更改演示程序
- ★ 附带软盘，内含多种演示程序源程序及编译软件
- ★ 可采用5V或9V(AC/DC)两种电源

评估板标准配置如下：

HD7279A	1只
AT89C2051	1只
数码管	8只
按键	16只
其它辅助电路	
3.5” 软盘	1张
使用说明书	1份

评估板电路图

