GT32L32S0140
标准汉字字库芯片

规格书
DATASHEET

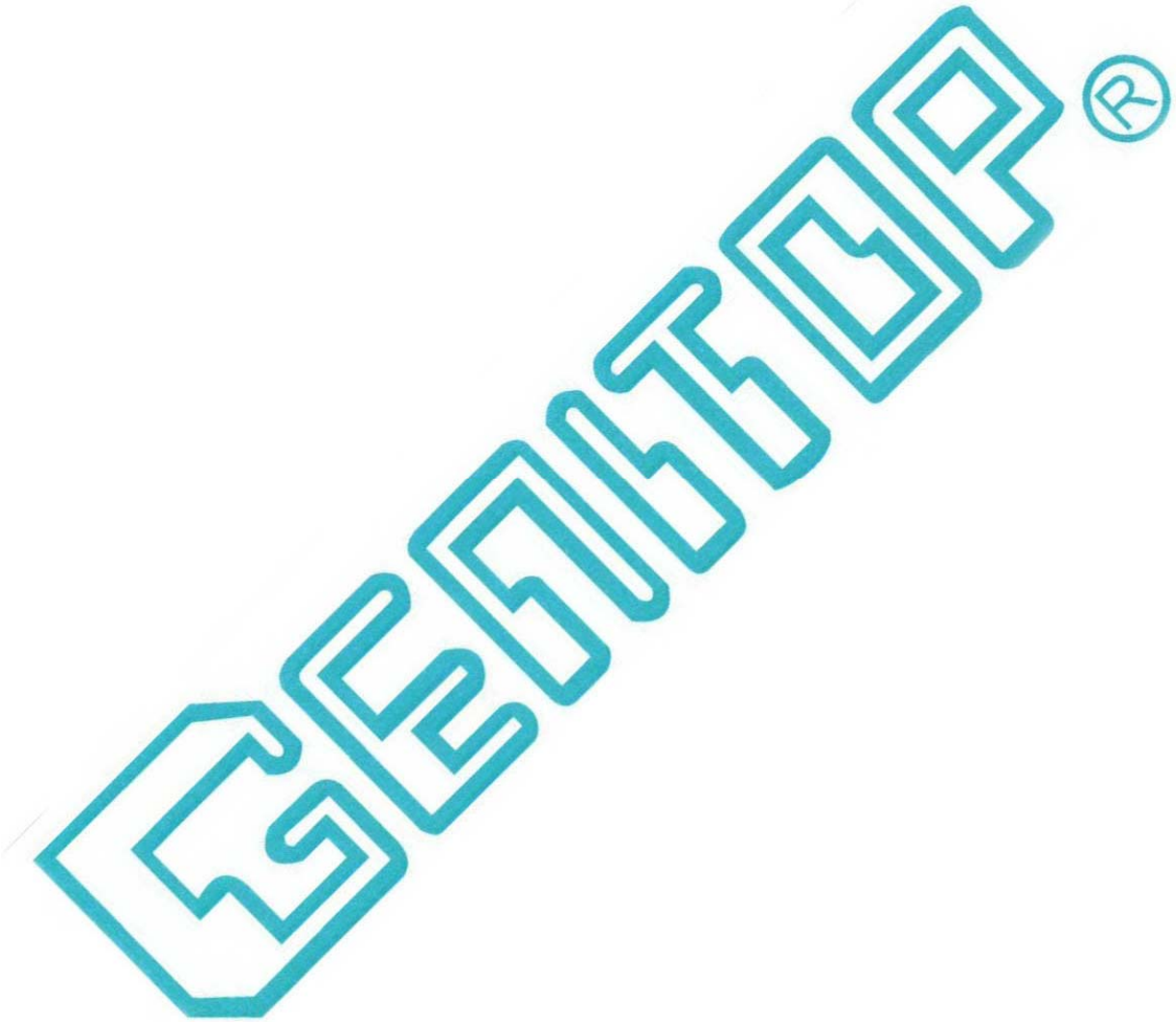
- 字符集：GB2312
- 兼容 Unicode
- 字号：12x12、16x16、24x24、32x32 点阵
- 排置方式：横置横排
- 总线接口：SPI 串行总线
- 封装类型：SOP8-B

VER 1.0 I_A

2012-07

版本修订记录

版本号	修改内容	日期	备注
V1.0	字库芯片说明书的制定	2012-07	字库定制



目　　录

1. 概述 4

1.1 芯片特点 4

1.2 芯片内容 5

1.3 字型样张 6

2. 操作指令 16

2.1 Instruction Parameter(指令参数) 16

2.2 Read Data Bytes（一般读取） 16

2.3 Read Data Bytes at Higher Speed（快速读取点阵数据） 17

2.4 Write Enable（写使能） 18

2.5 Write Disable（写非能） 18

2.6 Page Program（页写入） 18

2.7 Sector Erase（扇区擦除） 19

2.8 Block Erase(64K)（块擦除） 19

2.9 Chip Erase（芯片擦除） 19

3. 字符点阵字库地址表 20

4. 字符点阵数据在芯片中的地址计算方法 22

4.1 汉字字符点阵数据的地址计算 22

4.2 其它字符点阵数据的地址计算 24

4.3 内码转换程序 35

5. 自由可读写空间描述 36

5.1 存储组织 36

5.2 存储块、扇区结构 36

6. 引脚描述与电路连接 37

6.1 引脚配置 37

6.2 引脚描述 38

6.3 SPI接口与主机接口参考电路示意图 40

7. 电气特性 41

7.1 绝对最大额定值 41

7.2 DC特性 41

7.3 AC特性 41

8. 封装尺寸 43

1. 概述

GT32L32S0140是一款内含12x12、16x16、24x24、32x32点阵的汉字字库芯片，支持GB2312国标简体汉字（含有国家信标委合法授权）及ASCII字符。排列格式为横置横排，用户通过字符内码，利用本手册提供的方法计算出该字符点阵在芯片中的地址，可从该地址连续读出字符点阵信息。

GT32L32S0140除含有上述字库以外，还提供客户512K字节的可自由读写空间，包括128个扇区，每个扇区4K字节或16页，每页256字节，可自由读写空间地址范围为：0x000000~0x07FFFF。

1.1 芯片特点

- 数据总线：SPI 串行总线接口
- 点阵排列方式：横置横排
- 时钟频率：60MHz @3.3V
- 工作电压：2.7V~3.6V
- 电流：
  - 工作电流：12mA
  - 待机电流：10uA
- 工作温度：-40℃~85℃
- 封装：SOP8-B

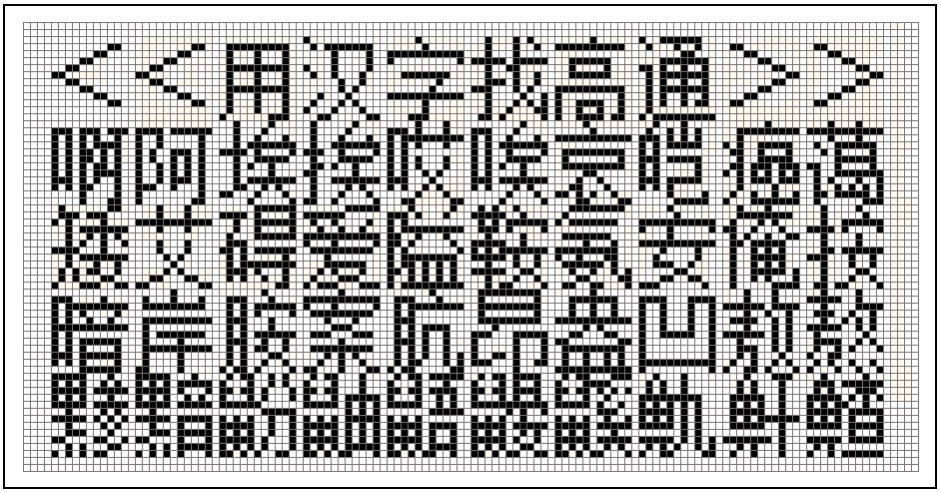
1.2 芯片内容

分类	字库	字号	字符数	字体	排列方式	备注
ASCII 字符集	ASCII	5x7	96	标准	W-横置横排	
	ASCII	7x8	96	标准	W-横置横排	
	ASCII	7x8	96	粗体	W-横置横排	
	ASCII	6x12	96	标准	W-横置横排	
	ASCII	8x16	128	标准	W-横置横排	
	ASCII	8x16	96	粗体	W-横置横排	
	ASCII	12x24	96	标准	W-横置横排	
	ASCII	12x24	224	打印机标准	W-横置横排	
	ASCII	12 点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	16 点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	24 点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	32 点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	12 点阵不等宽	96	Times New Roman（白正）	W-横置横排	
	ASCII	16 点阵不等宽	96	Times New Roman（白正）	W-横置横排	
	ASCII	24 点阵不等宽	96	Times New Roman（白正）	W-横置横排	
	ASCII	32 点阵不等宽	96	Times New Roman（白正）	W-横置横排	
	ASCII	16x32	96	标准	W-横置横排	
	ASCII	16x32	96	粗体	W-横置横排	
数字符号字符集	数字符号字符	14x28	15	黑体半角	W-横置横排	
	数字符号字符	20x40	12	黑体半角	W-横置横排	
	数字符号字符	28 点阵不等宽	15	Arial	W-横置横排	
	数字符号字符	40 点阵不等宽	12	Arial	W-横置横排	
GB2312 字符集	GB2312 汉字	12x12	6763	宋体	W-横置横排	
		16x16	6763	宋体	W-横置横排	
		24x24	6763	宋体	W-横置横排	
		24x24	6763	黑体	W-横置横排	
		32x32	6763	宋体	W-横置横排	
		32x32	6763	黑体	W-横置横排	
	GB2312 字符	12x12	846	宋体	W-横置横排	
		16x16	846	宋体	W-横置横排	
		24x24	846	宋体	W-横置横排	
		24x24	846	黑体	W-横置横排	
		32x32	846	宋体	W-横置横排	
		32x32	846	黑体	W-横置横排	
Unicode -> GB2312 转码表						
其它图符集	条形码字符 EAN13	12x27	60	标准	W-横置横排	
	条形码字符 CODE128	16x20	107	标准	W-横置横排	
	天线符号	12x12	5		W-横置横排	
	电池符号	12x12	4		W-横置横排	

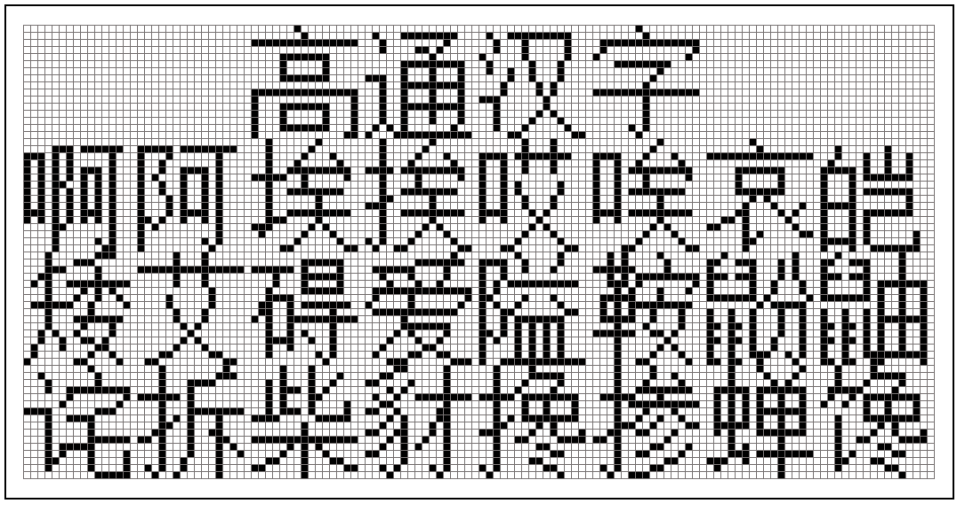
1.3 字型样张

1.3.1 汉字字符

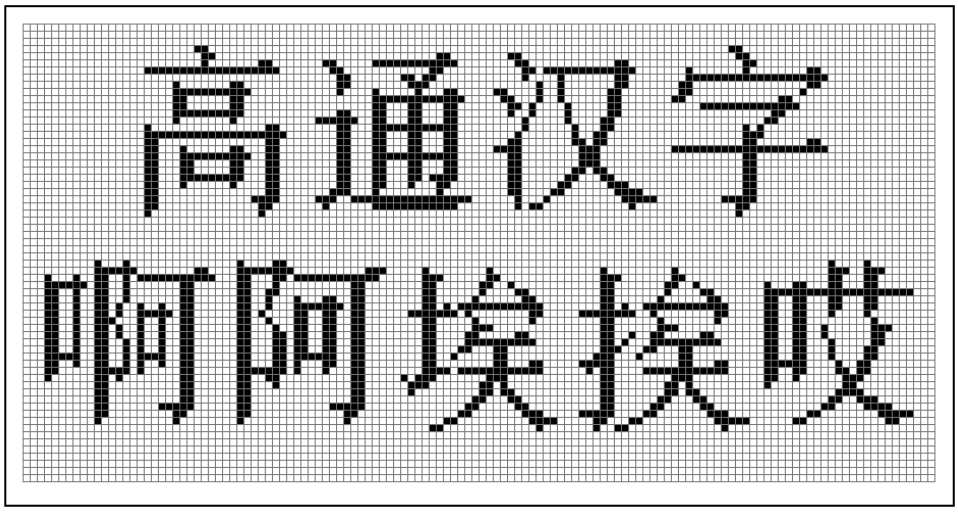
12x12 点阵 GB2312 汉字（宋体）



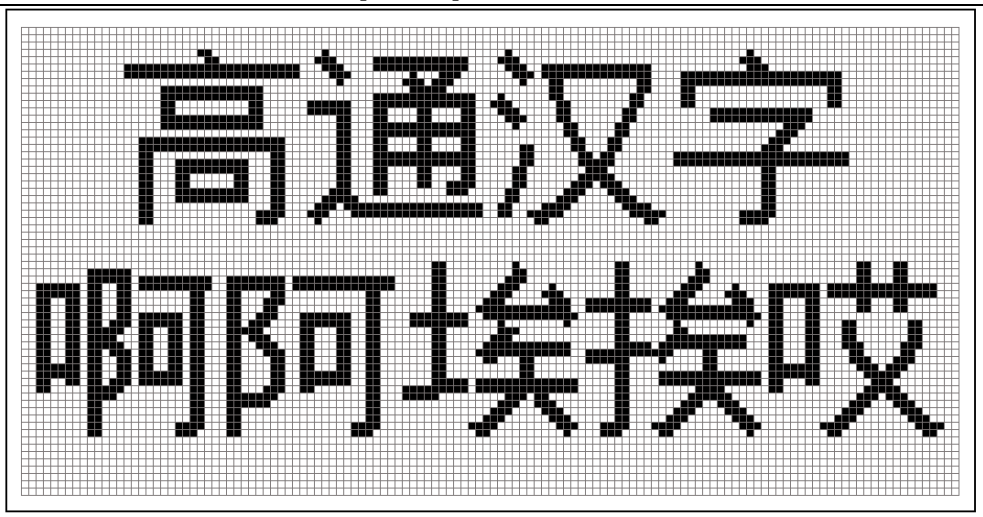
16x16 点阵 GB2312 汉字（宋体）



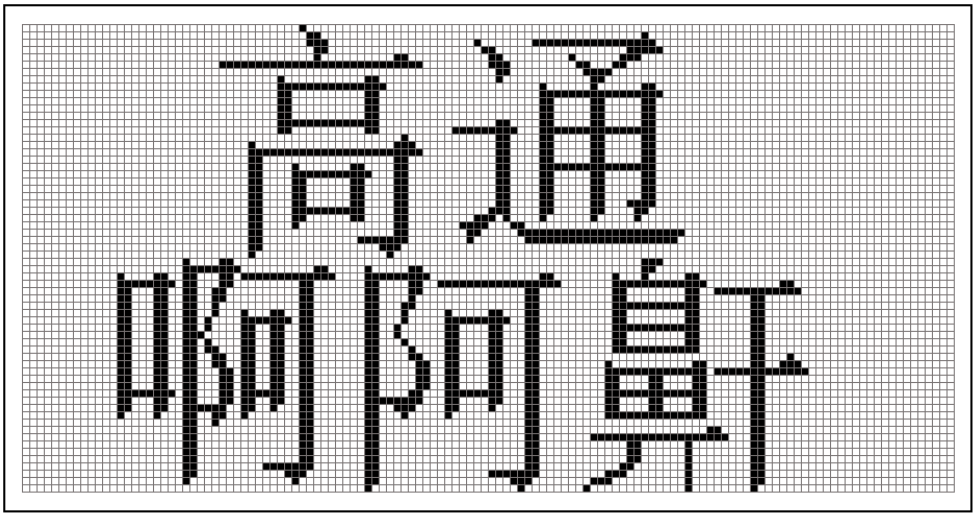
24x24 点阵 GB2312 汉字（宋体）



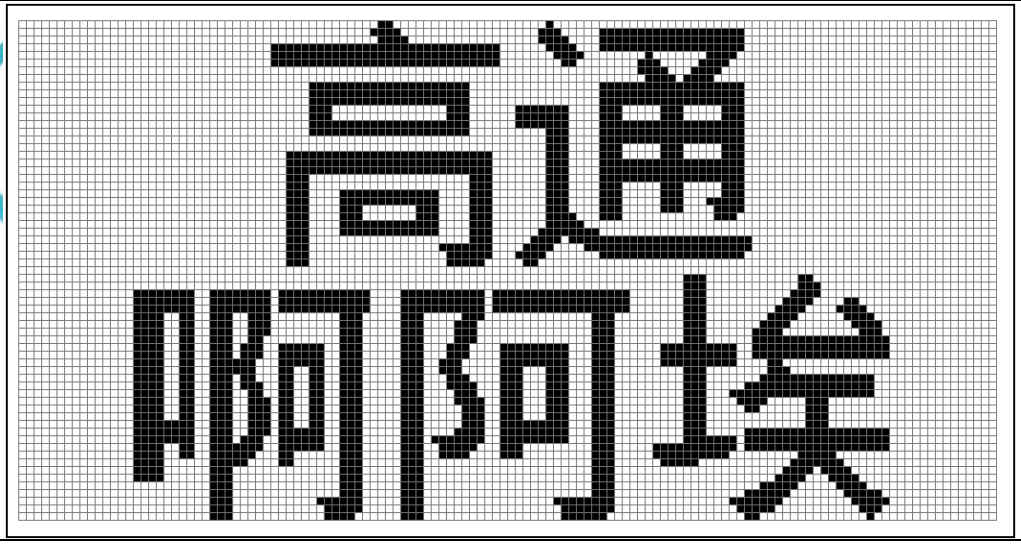
24x24 点阵 GB2312 汉字(黑体)



32x32 点阵 GB2312 汉字 （宋体）

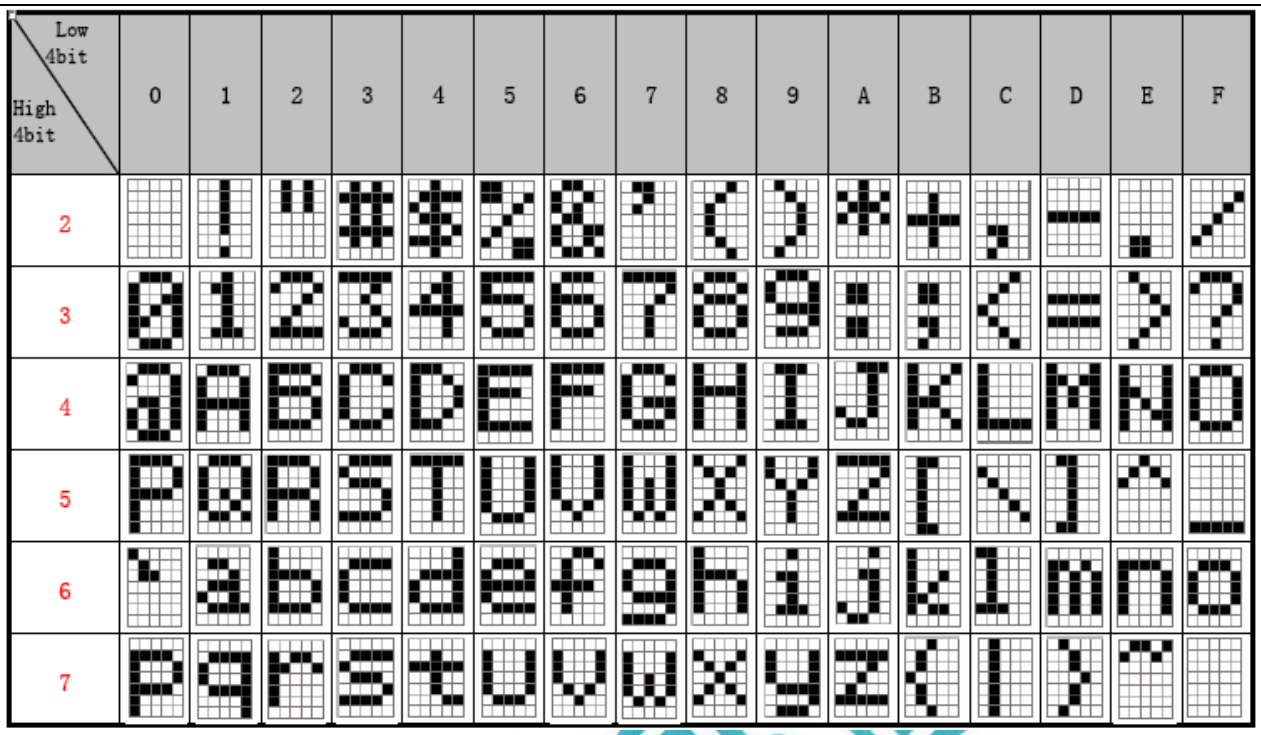


32x32 点阵 GB2312 汉字 （黑体）



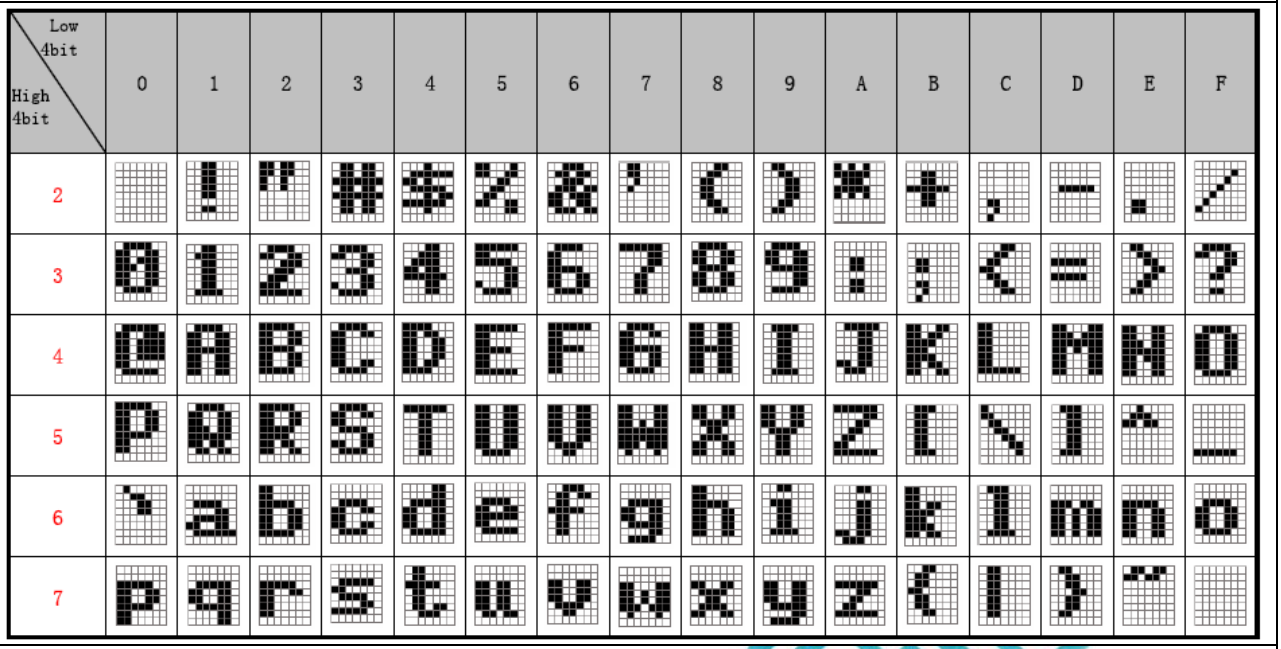
1.3.2 其它点阵字符

5x7 点阵 ASCII 标准字符

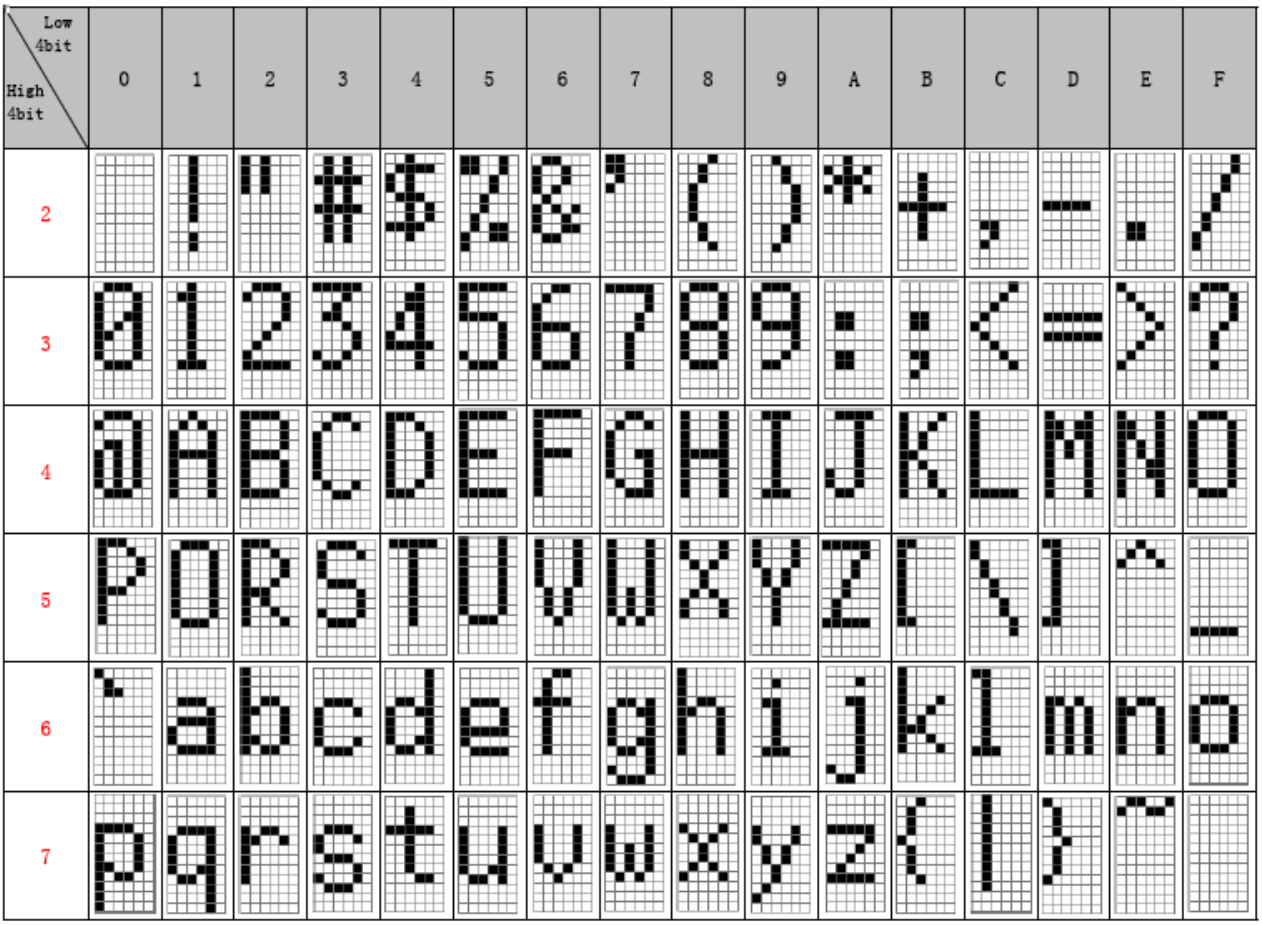


7x8 点阵 ASCII 标准字符

7x8 点阵 ASCII 粗体字符



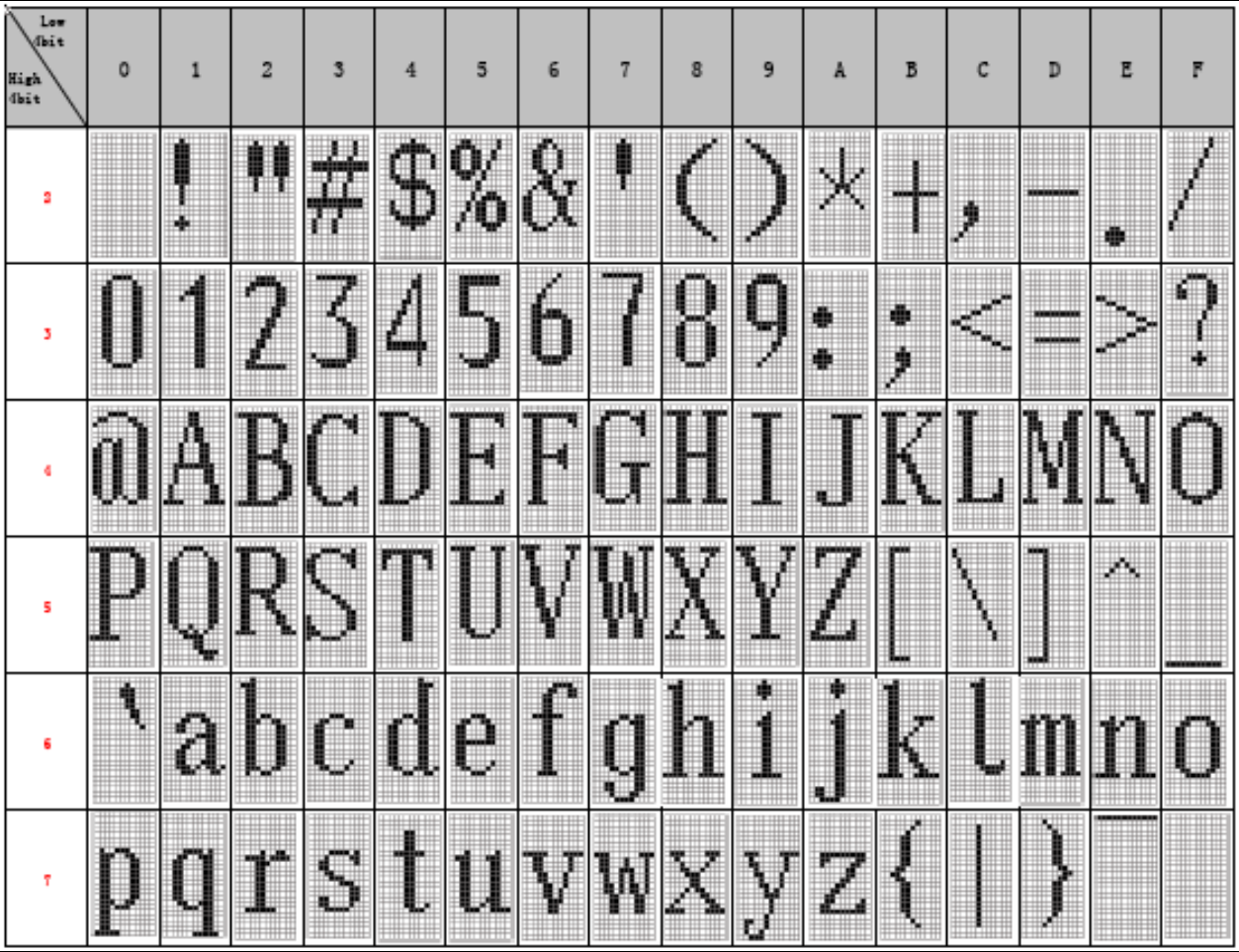
6x12 点阵 ASCII 标准字符



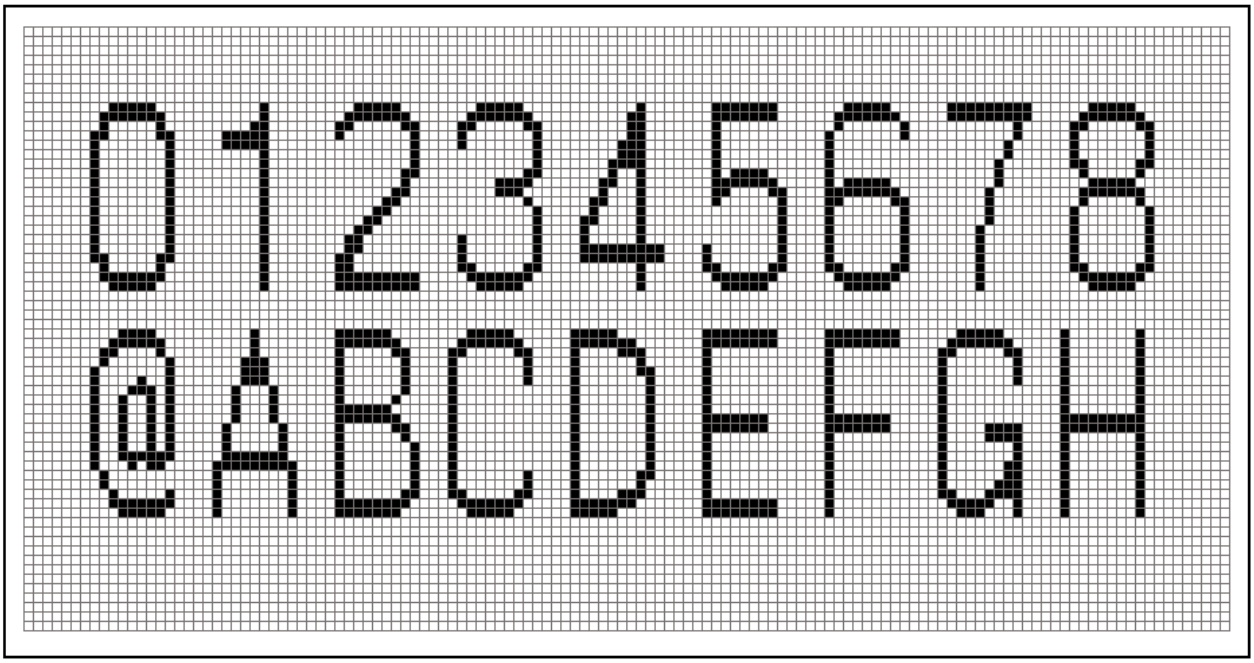
8x16 点阵 ASCII 标准字符

8x16 点阵 ASCII 粗体字符

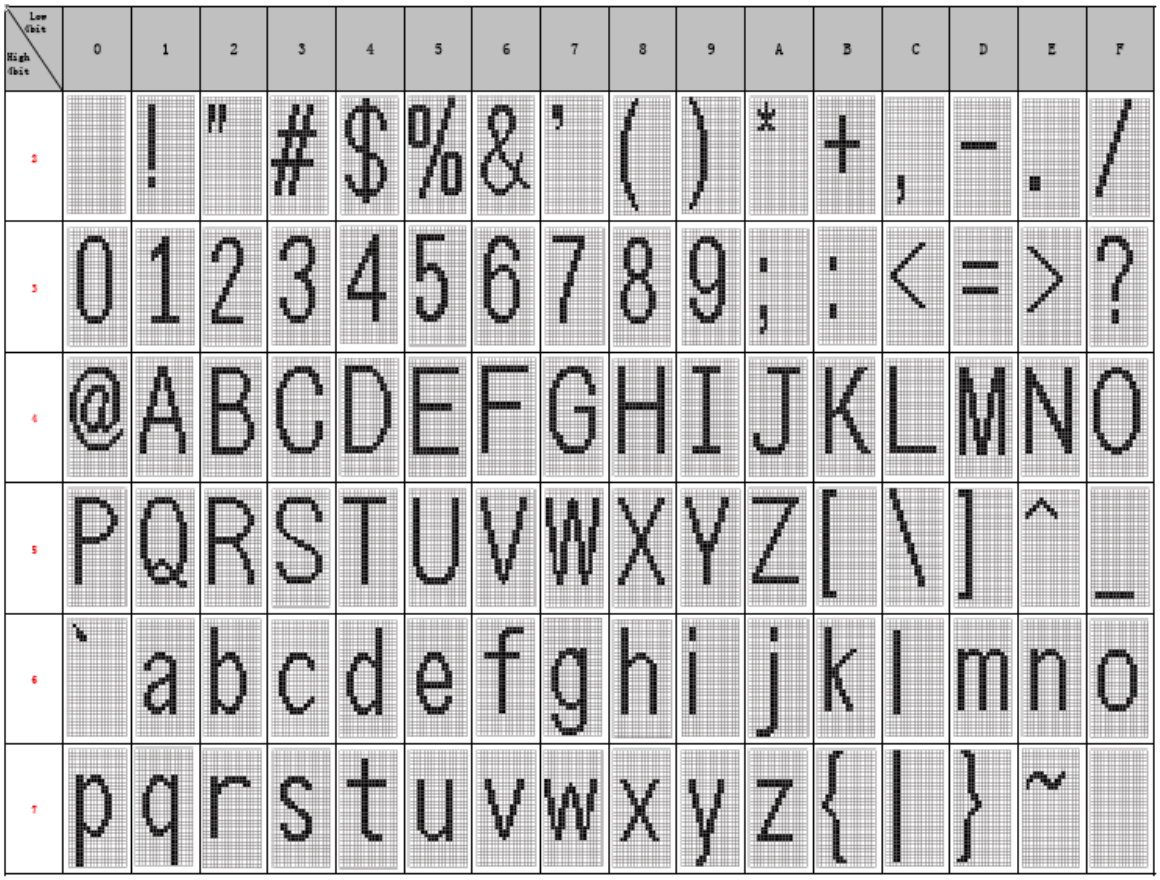
12x24 点阵 ASCII 标准字符



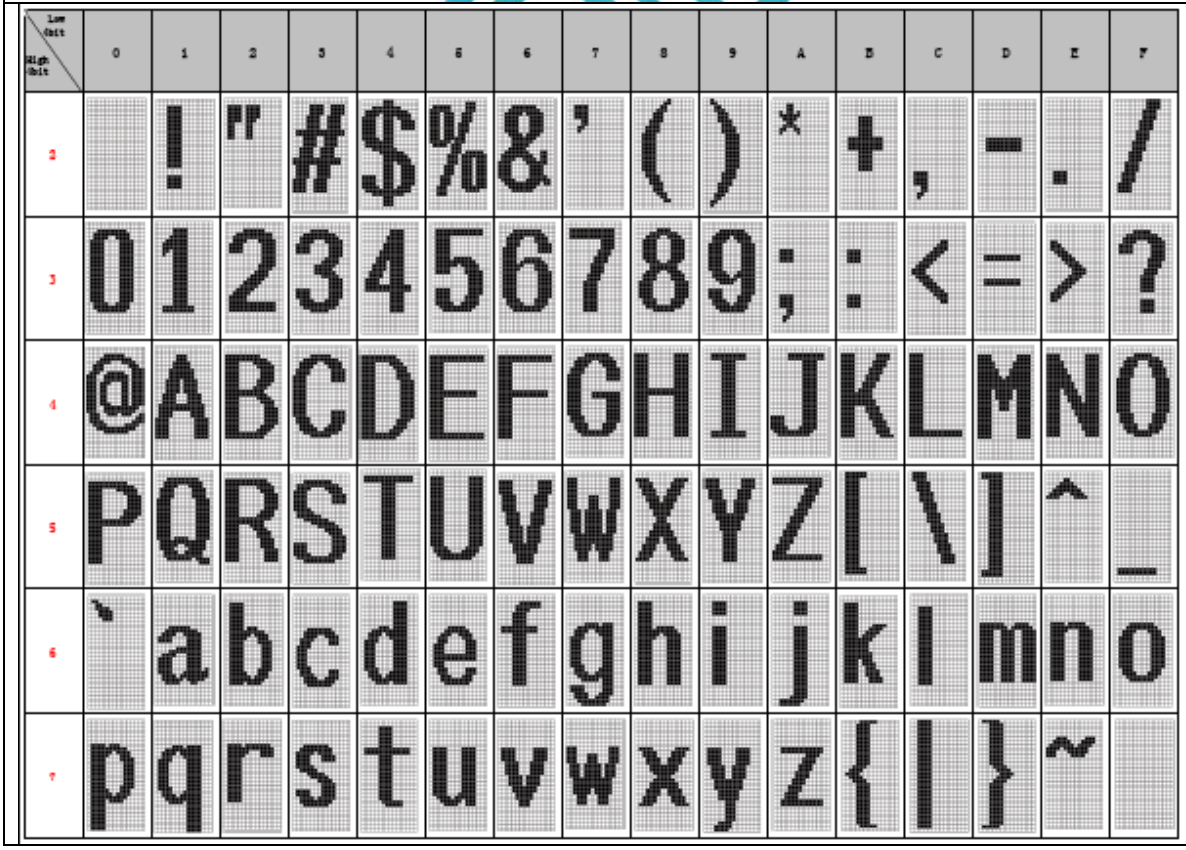
12x24 点阵打印机字符



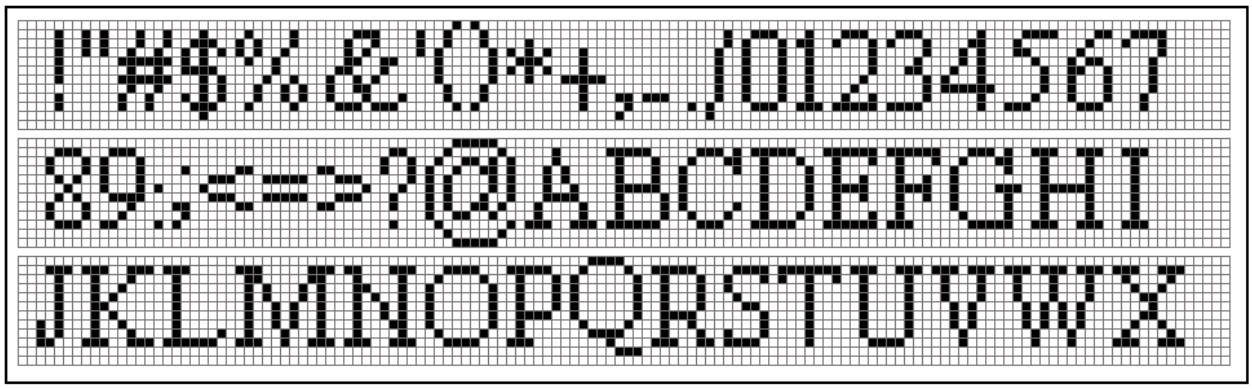
16x32 点阵 ASCII 标准字符



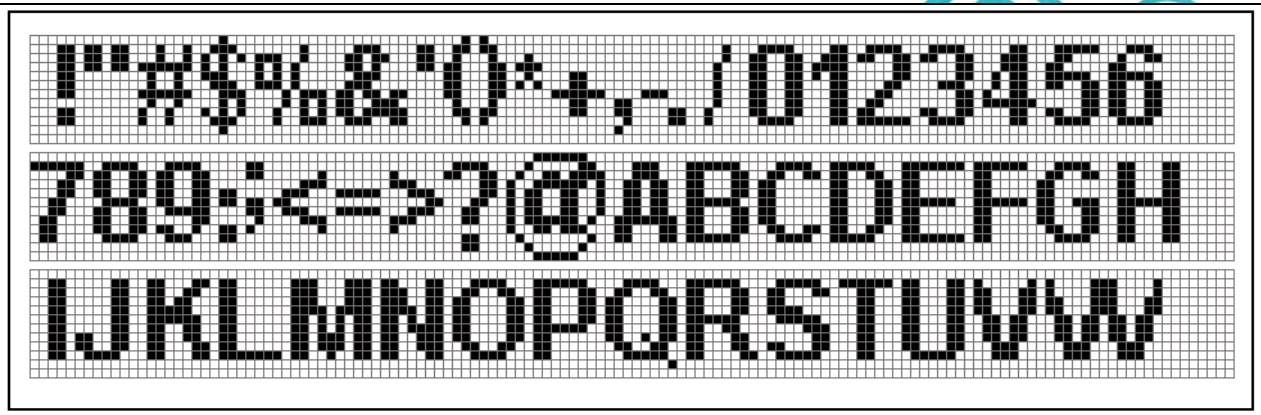
16x32 点阵 ASCII 粗体字符



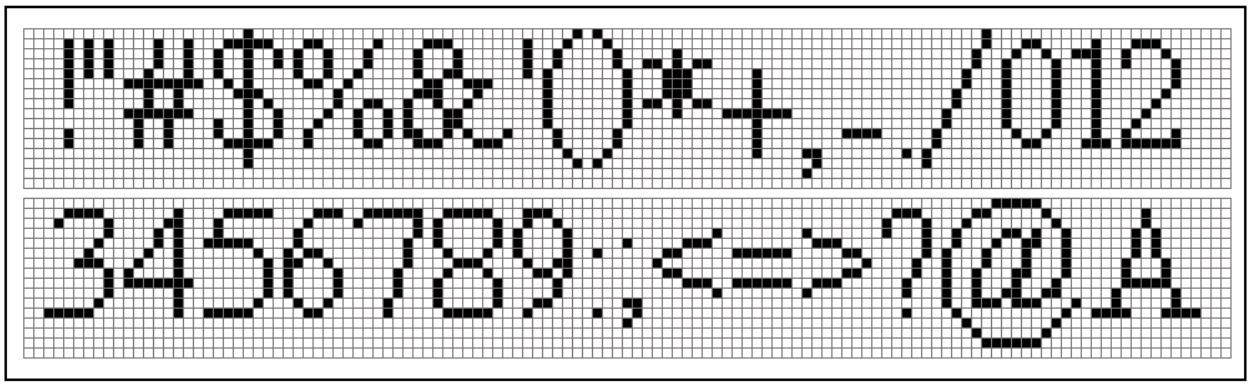
12 点阵不等宽 ASCII 白正(Times New Roman)



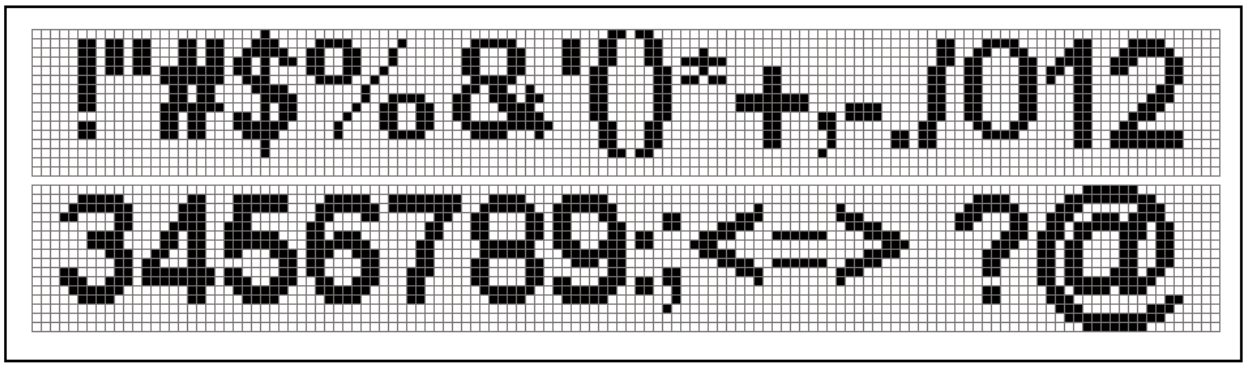
12 点阵不等宽 ASCII 方头(Arail)



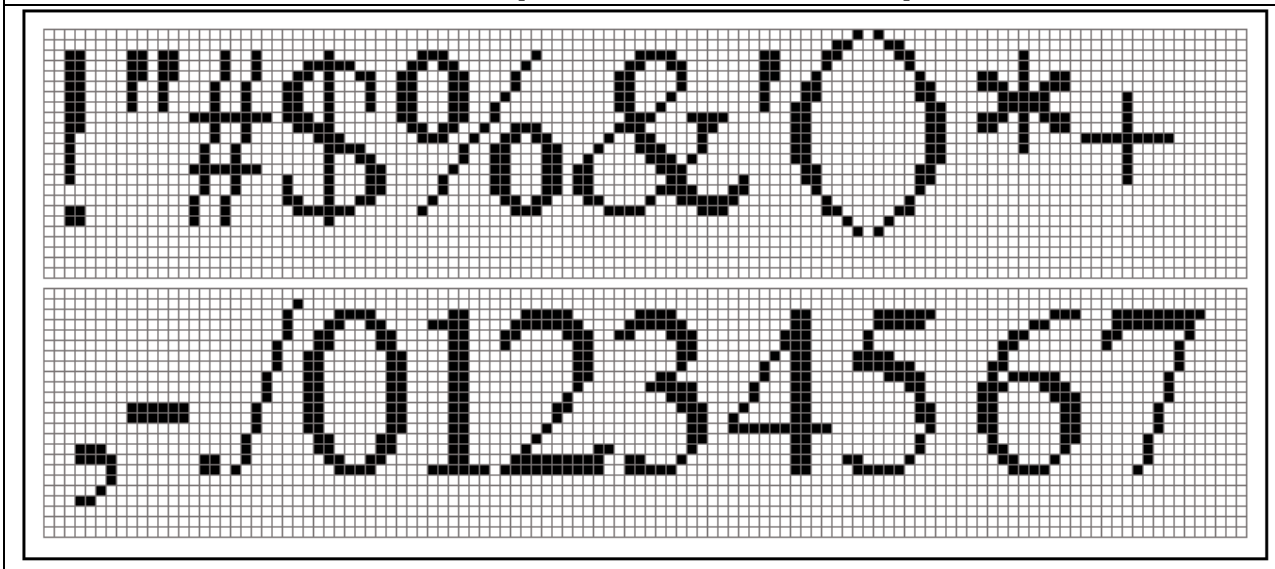
16 点阵不等宽 ASCII 白正(Times New Roman)



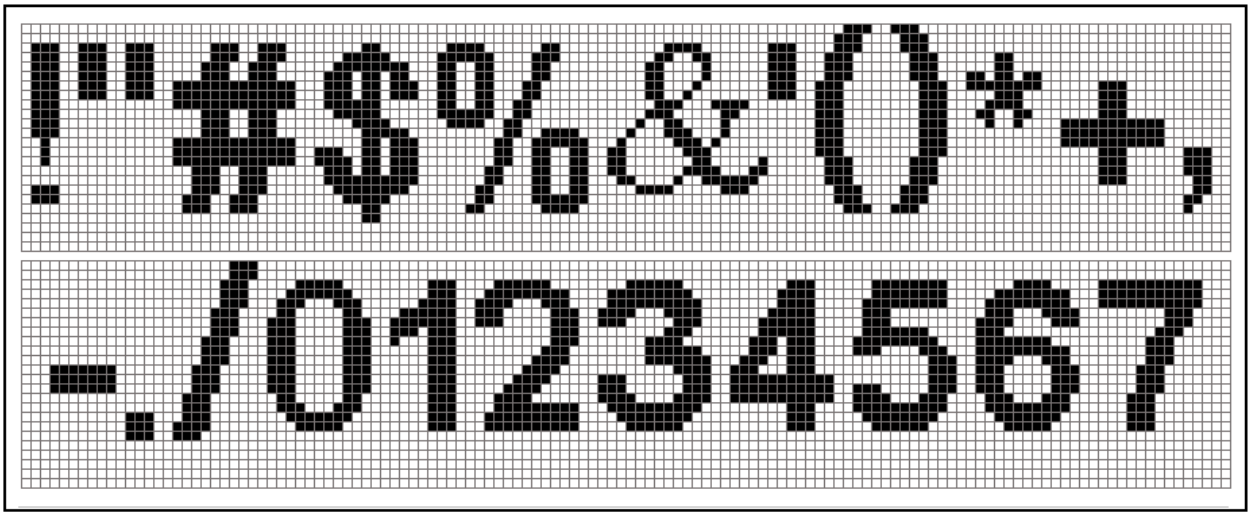
16 点阵不等宽 ASCII 方头(Arial)



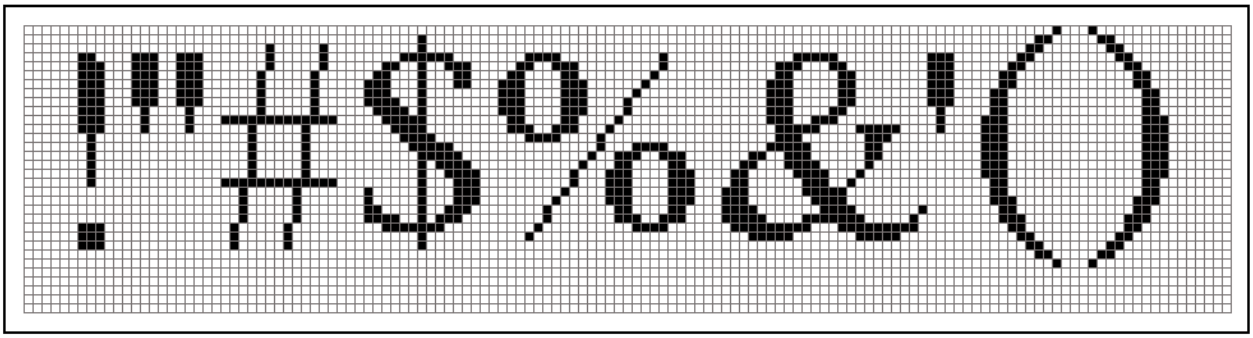
24 点阵不等宽 ASCII 白正(Times New Roman)



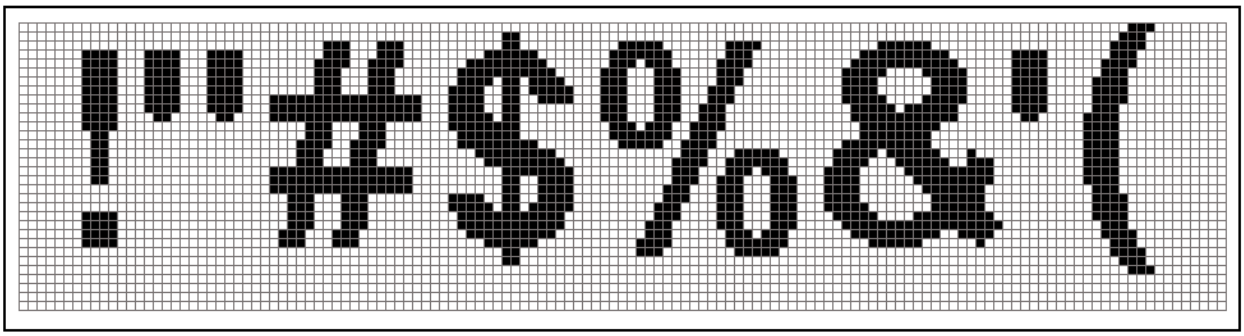
24 点阵不等宽 ASCII 方头(Arial)



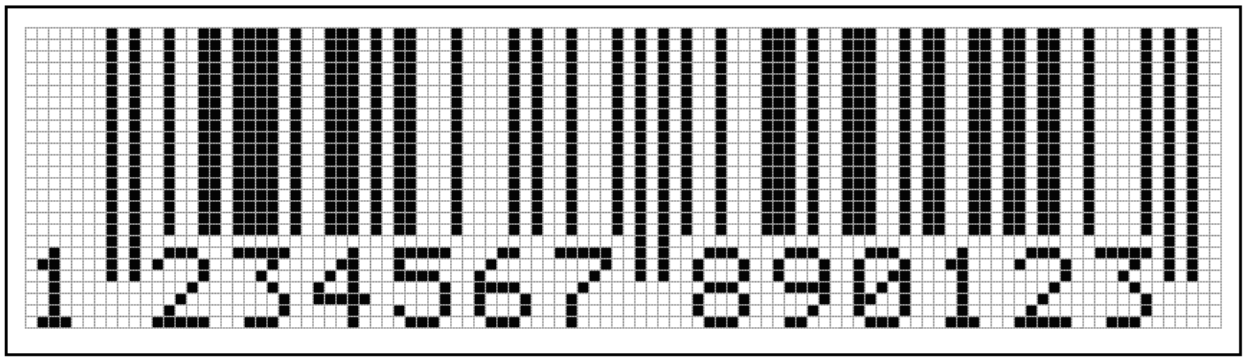
32 点阵不等宽 ASCII 白正(Times New Roman)



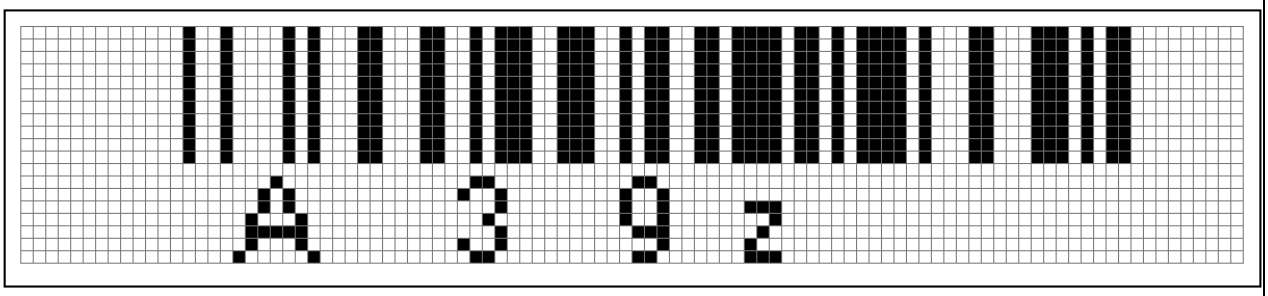
32 点阵不等宽 ASCII 方头(Arial)



条形码字符 EAN13



条形码字符 CODE128



2. 操作指令

2.1 Instruction Parameter(指令参数)

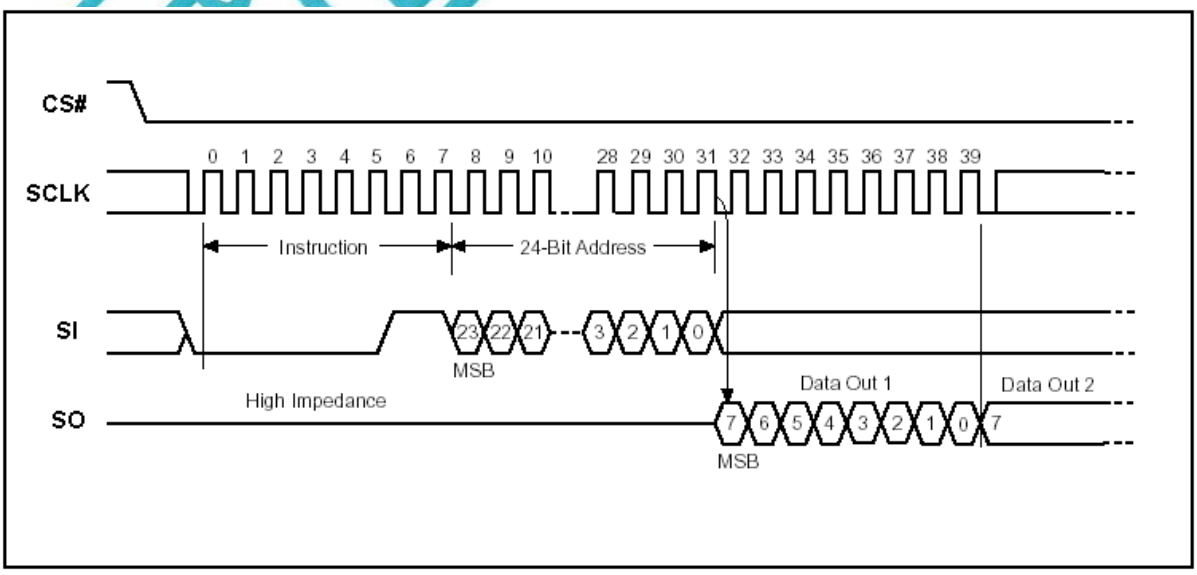
Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
Read	Read Data Bytes	0000 0011	03 h	3	—	1 to ∞
Fast Read	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞
WREN	Write Enalbe	0000 0110	06 h	—	—	—
WRDI	Write Disable	0000 0100	04 h			
PP	Page Program	0000 0010	02 h	3	—	1 to256
SE	Sector Erase	0010 0000	20 h	3	—	—
BE	Block Erase(64K)	1101 1000	D8 h	3	—	—
CE	Chip Erase	0110 0000/ 1100 0111	60 H/ C7 H	—	—	—

2.2 Read Data Bytes（一般读取）

Read Data Bytes 需要用指令码来执行每一次操作。READ 指令的时序如下(图)：

- 首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（03 h）和 3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。
- 然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。
- 读取字节数据后，则把片选信号（CS#）变为高，结束本次操作。
  如果片选信号（CS#）继续保持为底，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。

图：Read Data Bytes (READ) Instruction Sequence and Data-out sequence:

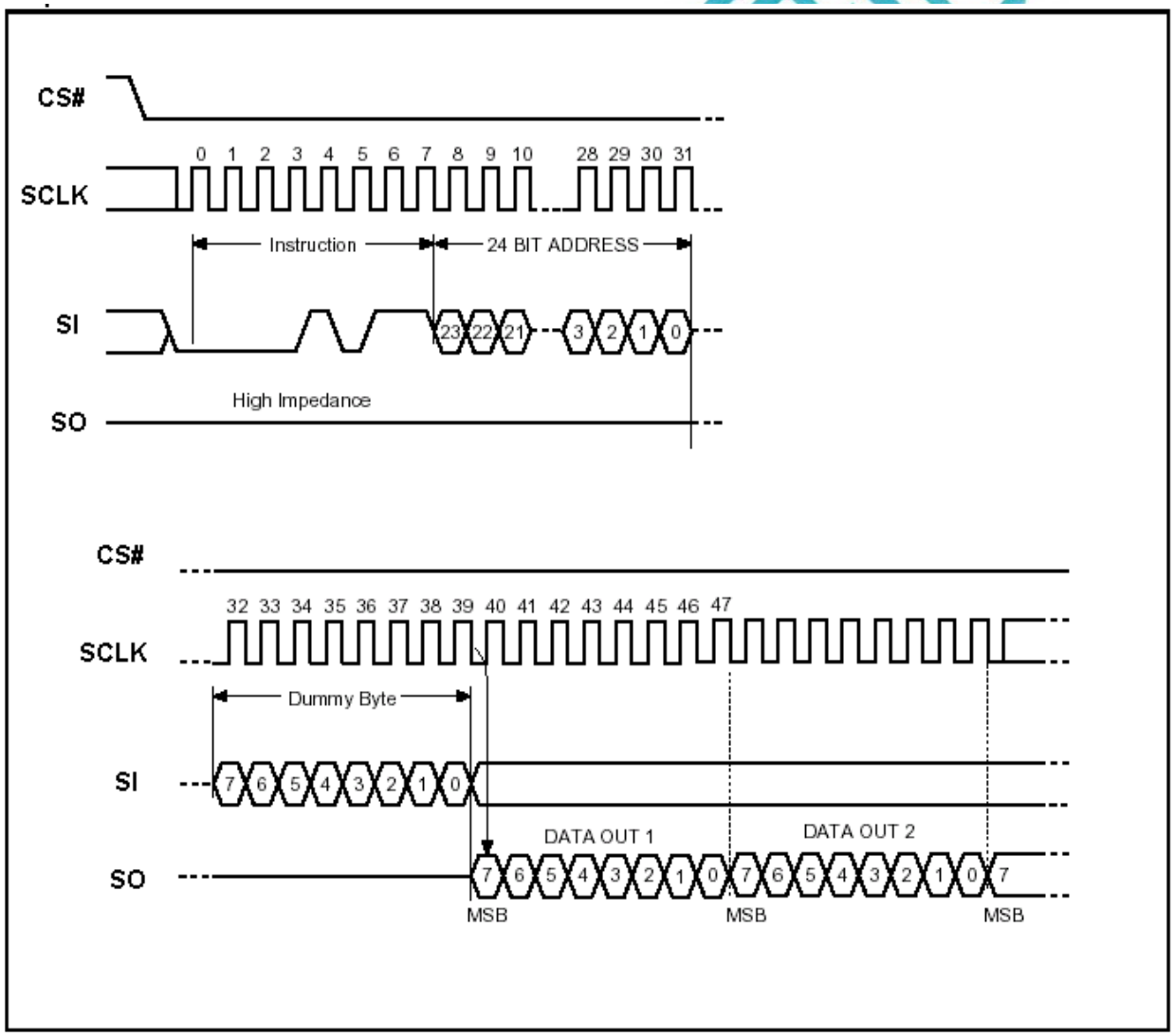


2.3 Read Data Bytes at Higher Speed（快速读取点阵数据）

Read Data Bytes at Higher Speed 需要用指令码来执行操作。READ_FAST 指令的时序如下(图)：

- 首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（0B h）和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。
- 然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。
- 如果片选信号（CS#）继续保持为底，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。例：读取一个 15x16 点阵汉字需要 32Byte，则连续 32 个字节读取后结束一个汉字的点阵数据读取操作。
  如果不需要继续读取数据，则把片选信号（CS#）变为高，结束本次操作。

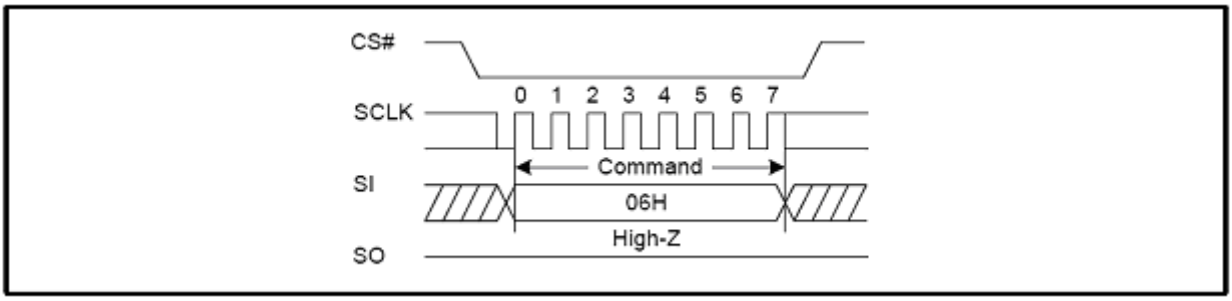
图：Read Data Bytes at Higher Speed (READ_FAST) Instruction Sequence and Data-out sequence:



2.4 Write Enable（写使能）

Write Enable 指令的时序如下(图)：

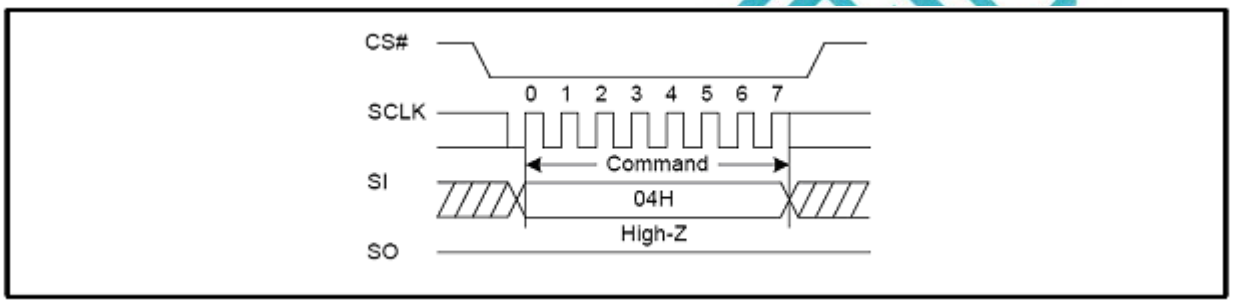
CS#变低-→发送 Write Enable 命令—>CS#变高



2.5 Write Disable（写非能）

Write Enable 指令的时序如下(图)：

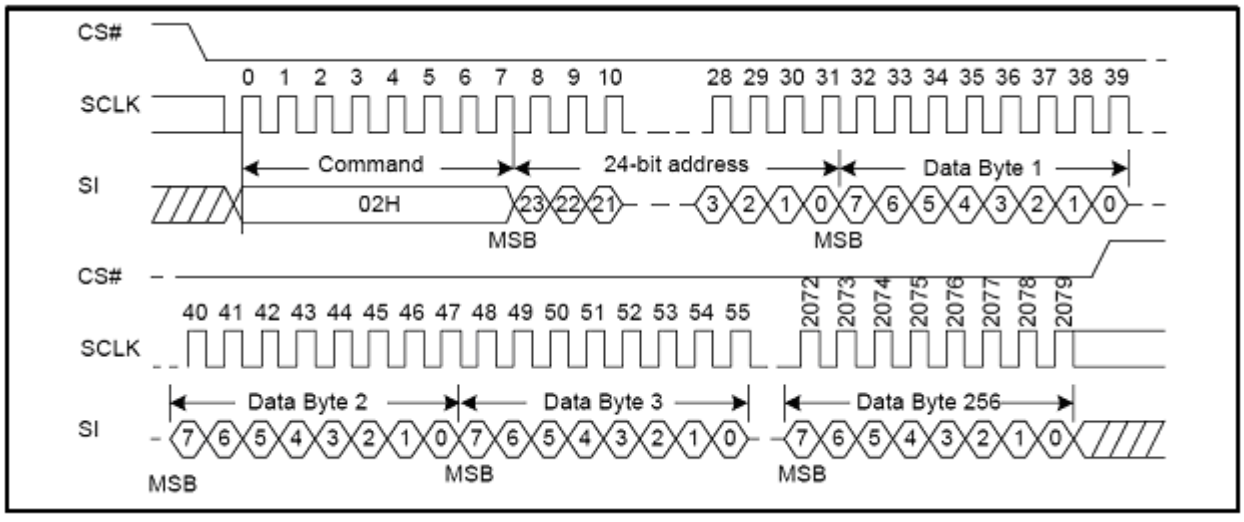
CS#变低-→发送 Write Disable 命令—>CS#变高



2.6 Page Program（页写入）

Page Program 指令的时序如下(图)：

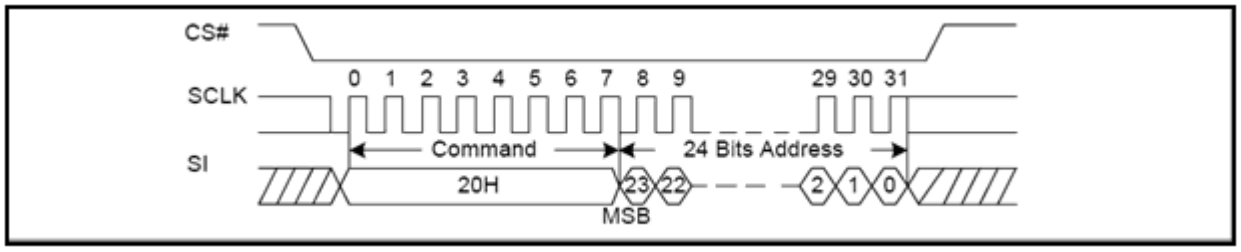
CS#变低-→发送 Page Program 命令→发送 3 字节地址—>发送数据—>CS#变高



2.7 Sector Erase（扇区擦除）

Sector Erase 指令的时序如下(图)：

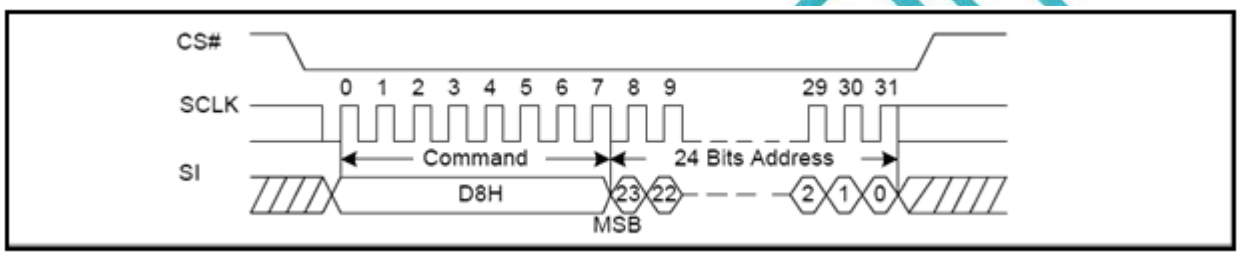
CS#变低->发送 Sector Erase 命令->发送 3 字节地址—>CS#变高



2.8 Block Erase(64K)（块擦除）

Block Erase 指令的时序如下(图)：

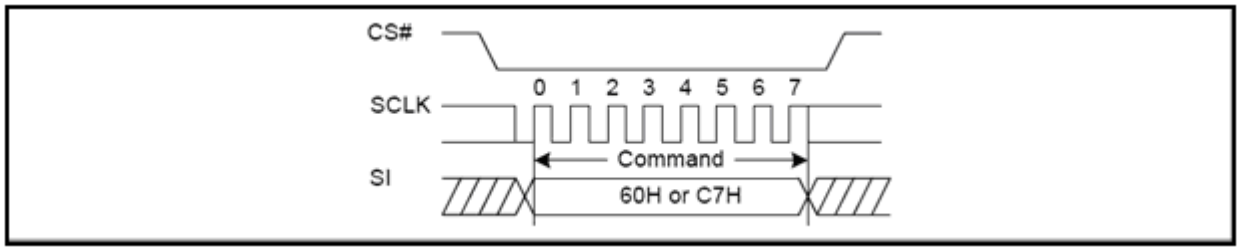
CS#变低->发送 64K Block Erase 命令->发送 3 字节地址—>CS#变高



2.9 Chip Erase（芯片擦除）

Chip Erase 指令的时序如下(图)：

CS#变低->发送 Chip Erase 命令—>CS#变高



3. 字符点阵字库地址表

NO.	字库内容	编码体系（字符集）	字符数	起始地址	参考算法
1	5x7 点阵 ASCII 标准字符	ASCII	96	0x080000	4.2.1
2	7x8 点阵 ASCII 粗体字符	ASCII	96	0x080300	4.2.2
3	7x8 点阵 ASCII 粗体字符	ASCII	96	0x080600	4.2.3
4	6x12 点阵 ASCII 字符	ASCII	96	0x080900	4.2.4
5	8x16 点阵 ASCII 标准字符	ASCII	128	0x080D80	4.2.5
6	8x16 点 ASCII 粗体字符	ASCII	96	0x081580	4.2.6
7	12x24 点阵 ASCII 字符	ASCII	96	0x081B80	4.2.7
8	12x24 点阵 ASCII 打印机字符	ASCII	224	0x082D80	4.2.8
9	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96	0x085780	4.2.9
10	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96	0x086140	4.2.10
11	24 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96	0x086E00	4.2.11
12	32 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96	0x0889C0	4.2.12
13	12 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96	0x08BA80	4.2.13
14	16 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96	0x08C450	4.2.14
15	24 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96	0x08D140	4.2.15
16	32 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96	0x08ED40	4.2.16
17	16x32 点阵 ASCII 标准字符	ASCII	96	0x091E00	4.2.17
18	16x32 点阵 ASCII 粗体字符	ASCII	96	0x093600	4.2.18
19	14x28 点阵数字符号字符	数字符号字符	15	0x094E00	4.2.19
20	20x40 点阵数字符号字符	数字符号字符	12	0x095148	4.2.20
21	28 点阵不等宽数字符号字符	数字符号字符	15	0x0956E8	4.2.21
22	40 点阵不等宽数字符号字符	数字符号字符	12	0x095D96	4.2.22
23	12x12 点阵 GB2312 汉字（宋体）	GB2312	6763	0x09670E	4.1.1
24	16x16 点阵 GB2312 汉字（宋体）	GB2312	6763	0x0C30DE	4.1.2
25	24x24 点阵 GB2312 汉字（宋体）	GB2312	6763	0x0FE89E	4.1.3
26	24x24 点阵 GB2312 汉字（黑体）	GB2312	6763	0x18460E	4.1.4
27	32x32 点阵 GB2312 汉字（宋体）	GB2312	6763	0x20A37E	4.1.5
28	32x32 点阵 GB2312 汉字（黑体）	GB2312	6763	0x2F828B	4.1.6
29	12x12 点阵 GB2312 字符（宋体）	GB2312	846	0x09670E	4.1.1
30	16x16 点阵 GB2312 字符（宋体）	GB2312	846	0x0C30DE	4.1.2
31	24x24 点阵 GB2312 字符（宋体）	GB2312	846	0x0FE89E	4.1.3
32	24x24 点阵 GB2312 字符（黑体）	GB2312	846	0x18460E	4.1.4
33	32x32 点阵 GB2312 字符（宋体）	GB2312	846	0x20A37E	4.1.5
34	32x32 点阵 GB2312 字符（黑体）	GB2312	846	0x2F828B	4.1.6

35	Unicode->GB2312 转码表		6763+846	0x3E618B	4.3.1
36	条形码字符 EAN13		60	0x3F222B	4.2.23
37	条形码字符 CODE128		107	0x3F2ED3	4.2.24
38	12x12 天线符号		5	0x3F3F8B	4.2.25
39	12x12 电池符号		4	0x3F4003	4.2.26
40	保留区	0x3F40C2~0x3FFFFF			

4. 字符点阵数据在芯片中的地址计算方法

用户只要知道字符的内码，就可以计算出该字符点阵在芯片中的地址，然后就可从该地址连续读出点阵信息用于显示。

4.1 汉字字符点阵数据的地址计算

4.1.1 12x12 点阵 GB2312 汉字&字符 （宋体）

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x09670E;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*24+ BaseAdd;
```

4.1.2 16x16 点阵 GB2312 汉字&字符（宋体）

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x0C30DE;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;
```

4.1.3 24x24 点阵 GB2312 汉字&字符（宋体）

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x0FE89E;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*72+ BaseAdd;
```

4.1.4 24x24 点阵 GB2312 汉字&字符 (黑体)

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x18460E;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*72+ BaseAdd;
```

4.1.5 32x32 点阵 GB2312 标准点阵字库&字符（宋体）

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x20A37E;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*128+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
```

Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*128+ BaseAdd;

4.1.6 32x32 点阵 GB2312 标准点阵字库&字符（黑体）

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x2F828B;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*128+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*128+ BaseAdd;
```

4.2 其它字符点阵数据的地址计算

4.2.1 5x7 点阵 ASCII 标准字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

```
BaseAdd=0x080000;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
    Address = (ASCIICode –0x20 ) * 8+BaseAdd;
```

4.2.2 7x8 点阵 ASCII 标准字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

```
BaseAdd=0x080300;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
    Address = (ASCIICode –0x20 ) * 8+BaseAdd;
```

4.2.3 7x8 点阵 ASCII 粗体字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：

```
BaseAdd=0x080600;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
        Address = (ASCIICode –0x20 ) * 8+BaseAdd;
```

4.2.4 6x12 点阵 ASCII 字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：

```
BaseAdd=0x080900;
if (ASCIICode >= 0x20) && (ASCIICode <= 0x7E)
     Address = (ASCIICode –0x20 ) * 12+BaseAdd;
```

4.2.5 8x16 点阵 ASCII 标准字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：

```
BaseAdd=0x080D80;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
     Address = (ASCIICode –0x20 ) * 16+BaseAdd;
```

4.2.6 8x16 点阵 ASCII 粗体字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：

```
BaseAdd=0x081580;
if((ASCIICode>=0x20)&&(ASCIICode<=0x7F))
```

Address = (ASCIICode-0x20)*16+BaseAdd;

4.2.7 12x24 点阵 ASCII 标准字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x081B80
if (ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 48+BaseAdd

4.2.8 12x24 点阵 ASCII 打印机字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x082D80;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 48+BaseAdd;

4.2.9 16x32 点阵 ASCII 标准字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x091E00;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 64+BaseAdd;

4.2.10 16x32 点阵 ASCII 粗体字符

参数说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：

BaseAdd=0x093600;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 64+BaseAdd;

4.2.11 12 点阵不等宽 ASCII 方头（Arial）字符

说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x085780;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 26 + BaseAdd;

4.2.12 16 点阵不等宽 ASCII 方头（Arial）字符

说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x086140;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 34 + BaseAdd;

4.2.13 24 点阵不等宽 ASCII 方头（Arial）字符

说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x086E00;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 74 + BaseAdd;

4.2.14 32 点阵不等宽 ASCII 方头（Arial）字符

说明：
ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x0889C0;

if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))

Address = (ASCIICode –0x20) * 130 + BaseAdd;

4.2.15 12 点阵不等宽 ASCII 白正（Times New Roman）字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x08BA80;

if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))

Address = (ASCIICode –0x20) *26 + BaseAdd;

4.2.16 16 点阵不等宽 ASCII 白正（Times New Roman）字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x08C450;

if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))

Address = (ASCIICode –0x20) * 34 + BaseAdd;

4.2.17 24 点阵不等宽 ASCII 白正（Times New Roman）字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x08D140;

if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))

Address = (ASCIICode –0x20) * 74+ BaseAdd;

4.2.18 32 点阵不等宽 ASCII 白正（Times New Roman）字符

说明：

ASCIICode：表示 ASCII 码（8bits）
BaseAdd：说明该套字库在芯片中的起始地址。
Address：ASCII 字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x08ED40;
if ((ASCIICode >= 0x20) && (ASCIICode <= 0x7E))
Address = (ASCIICode –0x20) * 130 + BaseAdd;

4.2.19 14x28 点阵数字符号字符

说明：
此部分内容为 0 1 2 3 4 5 6 7 8 9 , . ￥ $ £
Squence：表示 字符顺序，从 0 开始计数。
BaseAdd：说明该套字库在芯片中的起始地址。
Address：对应字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x094E00;
Address = Squence * 56+ BaseAdd;

4.2.20 20x40 点阵数字符号字符

说明：
此部分内容为 0 1 2 3 4 5 6 7 8 9 . ,
Squence：表示 字符顺序，从 0 开始计数。
BaseAdd：说明该套字库在芯片中的起始地址。
Address：对应字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x095148;
Address = (Squence) * 120+BaseAdd;

4.2.21 28 点阵不等宽数字号字符

说明：
此部分内容为 0 1 2 3 4 5 6 7 8 9 , . ￥ $ £
Squence：表示 字符顺序，从 0 开始计数。
BaseAdd：说明该套字库在芯片中的起始地址。
Address：对应字符点阵在芯片中的字节地址。
注：前两个字节为宽度信息。
计算方法：
BaseAdd=0x0956E8;
Address = Squence * 114+ BaseAdd;

4.2.22 40 点阵不等宽数字符号字符

说明：
此部分内容为 0 1 2 3 4 5 6 7 8 9 . ,
Squence：表示 字符顺序，从 0 开始计数。
BaseAdd：说明该套字库在芯片中的起始地址。
Address：对应字符点阵在芯片中的字节地址。
计算方法：
BaseAdd=0x095D96;
Address = Squence * 202+ BaseAdd;

4.2.23 EAN13 条形码调用程序

函数：DWORD* BAR_CODE(int* BAR_NUM)
功能：将数组条形码转为对应条形码图形地址。
参数：int* BAR_NUM 条形码数字数组指针，BAR_NUM[13]数组包含 13 个数字。
返回：定义 DWORD BAR_PIC_ADDR[13];用于存放对应地址，返回此数组指针。

```
DWORD BAR_PIC_ADDR[13];
DWORD* BAR_CODE(int* BAR_NUM)
{
    DWORD i,BaseAddr=0x3F222B;
BAR_PIC_ADDR[0]=BAR_NUM[0]*54+540*0+ BaseAddr;
BAR_PIC_ADDR[1]=BAR_NUM[1]*54+540*1+ BaseAddr;

    switch(BAR_NUM[0])
    {
    case 0:
        for(i=2;i<=6;i++)
        {
            BAR_PIC_ADDR[i]=BAR_NUM[i]*54+540*1+ BaseAddr;
        }
        break;

    case 1:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*2+ BaseAddr;
        break;
    case 2:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*2+ BaseAddr;
```

```
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*2+ BaseAddr;
        break;
    case 3:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*1+ BaseAddr;
        break;

case 4:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*2+ BaseAddr;
        break;
    case 5:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*2+ BaseAddr;
        break;
    case 6:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*1+ BaseAddr;
        break;
    case 7:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*2+ BaseAddr;
        break;
    case 8:
        BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*1+ BaseAddr;
        BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*2+ BaseAddr;
        BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*1+ BaseAddr;
```

```
            break;
        case 9:
            BAR_PIC_ADDR[2]=BAR_NUM[2]*54+540*2+ BaseAddr;
            BAR_PIC_ADDR[3]=BAR_NUM[3]*54+540*2+ BaseAddr;
            BAR_PIC_ADDR[4]=BAR_NUM[4]*54+540*1+ BaseAddr;
            BAR_PIC_ADDR[5]=BAR_NUM[5]*54+540*2+ BaseAddr;
            BAR_PIC_ADDR[6]=BAR_NUM[6]*54+540*1+ BaseAddr;
            break;

        }

BAR_PIC_ADDR[7]=BAR_NUM[7]*54+540*3+ BaseAddr;
        for(i=8;i<=11;i++)
        {
            BAR_PIC_ADDR[i]=BAR_NUM[i]*54+540*4+ BaseAddr;
        }
        BAR_PIC_ADDR[12]=BAR_NUM[12]*54+540*5+ BaseAddr;
return BAR_PIC_ADDR;
}
```

4.2.24 GB/T 18347-2001(CODE128)条形码的调用程序

函数：DWORD* BAR_CODE(int* BAR_NUM)
功能：将数组条形码转为对应条形码图形地址
参数：int* BAR_NUM 条形码数字数组指针,BAR_NUM[4]数组包含 4 个条形码 ASCII 符(数组取值

为 0~94)。
返回：定义 DWORD BAR_PIC_ADDR[7] ；用于存放对应地址，返回数组指针。
设基地址：BaseAddr=0x3F2ED3;
起始符有 3 种模式
当 flag=1 时为 Code-128-A;
当 flag=2 时为 Code-128-B;
当 flag=3 时为 Code-128-C;

```
DWORD flag;
DWORD BAR_PIC_ADDR[7];
DWORD* BAR_CODE(int* BAR_NUM)
{
  int i;
  for(i=0;i<7;i++)
  {
      switch(flag)
      case 1 :
```

```
    if(i==0)
    {
      BAR_PIC_ADDR[i]=103*40+BaseAddr;
    }
    else if(i==1||i==2||i=3||i==4)
    {
      BAR_PIC_ADDR[i]=BAR_NUM[i-1]*40+BaseAddr;
    }
    else if(i==5)
    {
      BAR_PIC_ADDR[i]=95*40+BaseAddr;
    }
    else if(i==6)
    {
      BAR_PIC_ADDR[i]=106*40+BaseAddr;
    }
    break;

    case 2 :
    if(i==0)
    {
      BAR_PIC_ADDR[i]=104*40+BaseAddr;
    }
    else if(i==1||i==2||i=3||i==4)
    {
      BAR_PIC_ADDR[i]=BAR_NUM[i-1]*40+BaseAddr;
    }
    else if(i==5)
    {
      BAR_PIC_ADDR[i]=95*40+BaseAddr;
    }
    else if(i==6)
    {
      BAR_PIC_ADDR[i]=106*40+BaseAddr;
    }
    break;


  case 3 :
    if(i==0)
    {
      BAR_PIC_ADDR[i]=105*40+BaseAddr;
    }
    else if(i==1||i==2||i=3||i==4)
    {
```

```
        BAR_PIC_ADDR[i]=BAR_NUM[i-1]*40+BaseAddr;
      }
      else if(i==5)
      {
        BAR_PIC_ADDR[i]=95*40+BaseAddr;
      }
      else if(i==6)
      {
        BAR_PIC_ADDR[i]=106*40+BaseAddr;
      }
      break;

      default:
      break;
  }
   return BAR_PIC_ADDR;

}
```

注：在屏上打点时要按照国家规范进行拼凑。

4.2.25 天线调用程序

函数：DWORD* Antenna_CODE(int* NUM)
功能：获取 12X12 天线 调用地址。
参数：NUM 0123 带表天线信号强度。
返回：数据地址

```
DWORD* Antenna_CODE(int* NUM)
{
    DWORD BaseAdd=0x3F3F8B;
    return(BaseAdd+NUM*24);
}
```

4.2.26 电池调用程序

函数：DWORD* Battery_CODE(int* NUM)
功能：获取 12X12 电池 调用地址。
参数：NUM 0123 带表电池电量。
返回：数据地址

```
DWORD* Battery_CODE(int* NUM)
{
    DWORD BaseAdd=0x3F4003;
    return(BaseAdd+NUM*24);
}
```

4.3 内码转换程序

16X16 unicode 简体中文调用程序

BYTE readbyte(DWORD address);
函数功能：从字库芯片任意地址读取1BYTE数据。
参数：数据在字库存放的地址。
返回值：地址对应字库中的内容。

参数说明：
U_Addr:表示需要查询的汉字或者字符在unicode码表中的索引序号
Address:表示需要查询Unicode码的汉字或者字符对应的GB2312编码存放的地址
BaseAddr: 表示unicode码表的起始位置。
unicodenum : 表示address在unicode码表中的序号
FontAddr：对应字符点阵数据的存放起始地址。
MSB：中文字符序号的高八位。
LSB：中文字符序号的低八位。
计算方法：

```
BaseAddr=0x3E618B;
u16 UnicodeToGB2312(u16 code)
{
u16 U_Addr;
if(code>=0x00a0 && code<=0x0451) U_Addr =2*(code-0xA0);
else if(code>=0x2010 && code<=0x2642) U_Addr =2*(code-0x2010)+1892;
else if(code>=0x3000 && code<=0x33d5) U_Addr =2*(code-0x3000)+5066;
else if(code>=0x4E00 && code<=0x9FA5) U_Addr =2*(code-0x4E00)+7030;
else if(code>=0xfe30&& code<=0xfe6b) U_Addr =2*(code-0xfe30)+48834;
else if(code>=0xff01 && code<=0xFF5e) U_Addr =2*(code-0xff01)+48954;
else if(code>=0xffe0 && code<=0xFFe5) U_Addr =2*(code-0xff01)+49142;
else U_Addr =0;
return U_Addr;
}
Address= UnicodeToGB2312 (Unicode)+ BaseAddr;
MSB=ZK_Read_1_byte(Address);  //从字库中读取 GB 编码的高字节（1 个字节）
LSB=ZK_Read_1_byte(Address+1);  //从字库中读取 GB 编码的低字节（1 个字节）
```

ZK_Read_1_byte(unsigned long addr)函数功能为从字库的地址 addr 读取一个字节的数据。需客户自己根据实际情况编写读写函数，或者参考集通科技显示例程编写。本规格书未提供。
验证数据：啊字的 UNICODE 编码：0x554A, GB 编码为 0xB0A1,
读取后转入 GB2312 编码计算公式计算 UNICODE 编码对应的汉字的实际地址。

5. 自由可读写空间描述

5.1 存储组织

每设备	每块	每扇区	每页	
512K	64K	4K	256	字节
2K	256	16		页
128	16			扇区
8				块

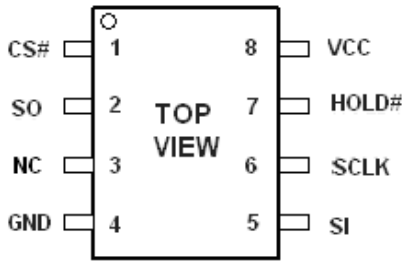
5.2 存储块、扇区结构

块	扇区	地址范围	
7	127	0x07F000	0x07FFFF
	.....	.....	.....
	112	0x070000	0x070FFF
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
2	47	0x02F000	0x02FFFF
	.....	.....	.....
	32	0x020000	0x020FFF
1	31	0x01F000	0x01FFFF
	.....	.....	.....
	16	0x010000	0x010FFF
0	15	0x00F000	0x00FFFF
	.....	.....	.....
	0	0x000000	0x000FFF

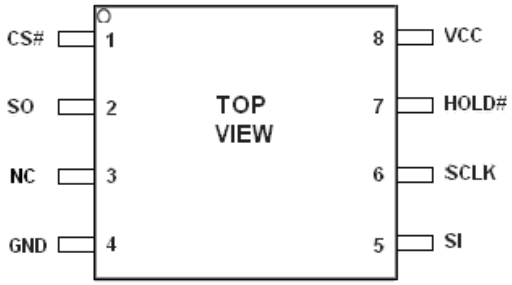
6. 引脚描述与电路连接

6.1 引脚配置

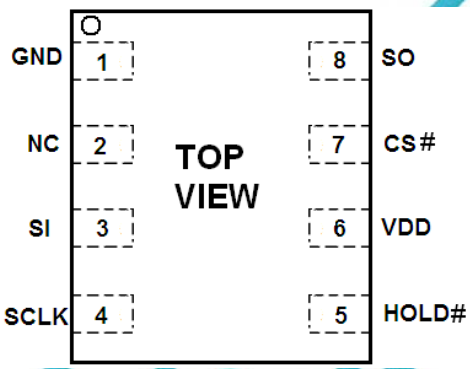
SOP8-A



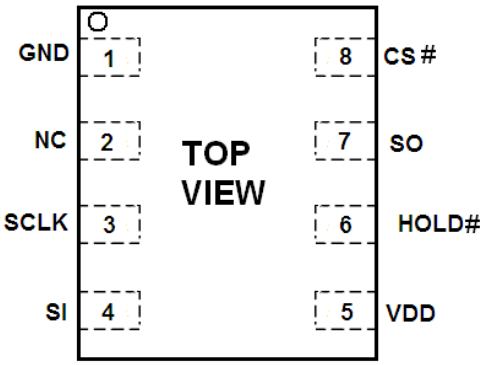
SOP8-B



DFN8-A



DFN8-B



6.2 引脚描述

SOP8-A/SOP8-B

NO.	名称	I/O	描述
1	CS#	I	片选输入（Chip enable input）
2	SO	O	串行数据输出 (Serial data output)
3	NC		悬空
4	GND		地(Ground)
5	SI	I	串行数据输入 (Serial data input)
6	SCLK	I	串行时钟输入（Serial clock input）
7	HOLD#	I	总线挂起（Hold, to pause the device without）
8	VCC		电源(+ 3.3V Power Supply)

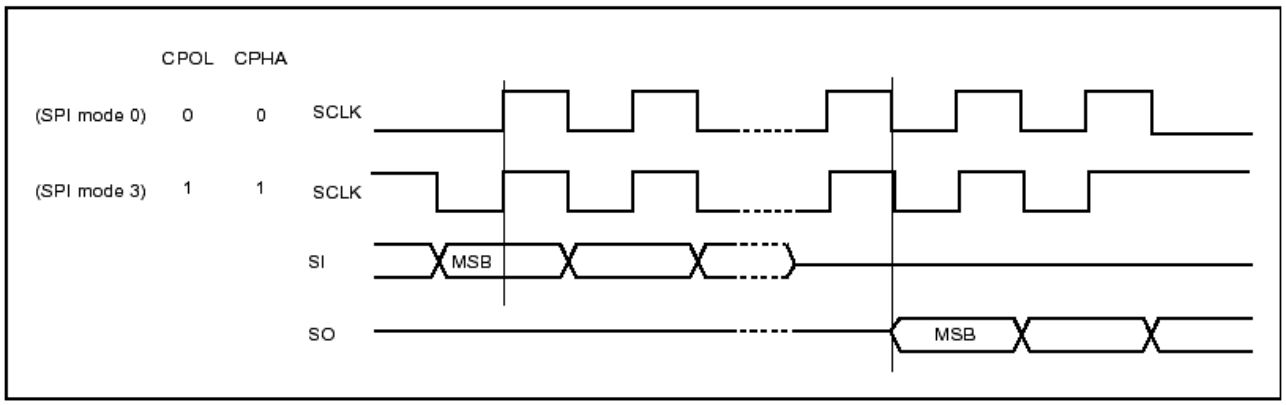
DFN8-A

NO.	名称	I/O	描述
1	GND		地(Ground)
2	NC		悬空
3	SI	I	串行数据输入 (Serial data input)
4	SCLK	I	串行时钟输入（Serial clock input）
5	HOLD#	I	总线挂起（Hold, to pause the device without）
6	VCC		电源(+ 3.3V Power Supply)
7	CS#	I	片选输入（Chip enable input）
8	SO	O	串行数据输出 (Serial data output)

DFN8-B

NO.	名称	I/O	描述
1	GND		地(Ground)
2	NC		悬空
3	SCLK	I	串行时钟输入（Serial clock input）
4	SI	I	串行数据输入 (Serial data input)
5	VCC		电源(+ 3.3V Power Supply)
6	HOLD#	I	总线挂起（Hold, to pause the device without）
7	SO	O	串行数据输出 (Serial data output)
8	CS#	I	片选输入（Chip enable input）

串行数据输出（SO）：该信号用来把数据从芯片串行输出，数据在时钟的下降沿移出。
串行数据输入（SI）：该信号用来把数据从串行输入芯片，数据在时钟的上升沿移入。
串行时钟输入（SCLK）：数据在时钟上升沿移入，在下降沿移出。
片选输入（CS#）：所有串行数据传输开始于CS#下降沿，CS#在传输期间必须保持为低电平，在两条指令之间保持为高电平。

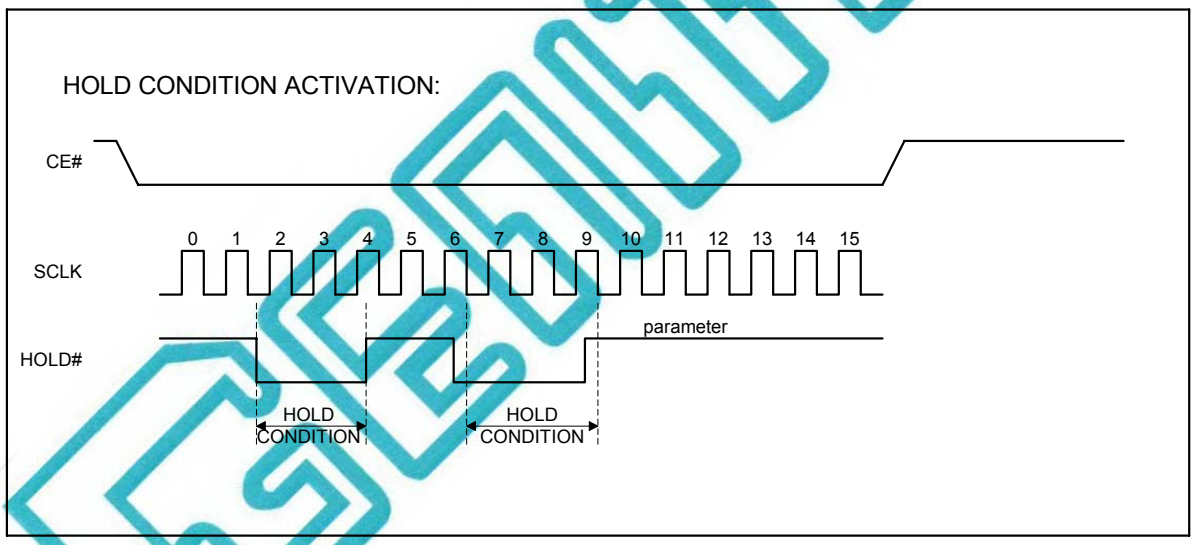


总线挂起输入（HOLD#）：

该信号用于片选信号有效期间暂停数据传输，在总线挂起期间，串行数据输出信号处于高阻态，芯片不对串行数据输入信号和串行时钟信号进行响应。

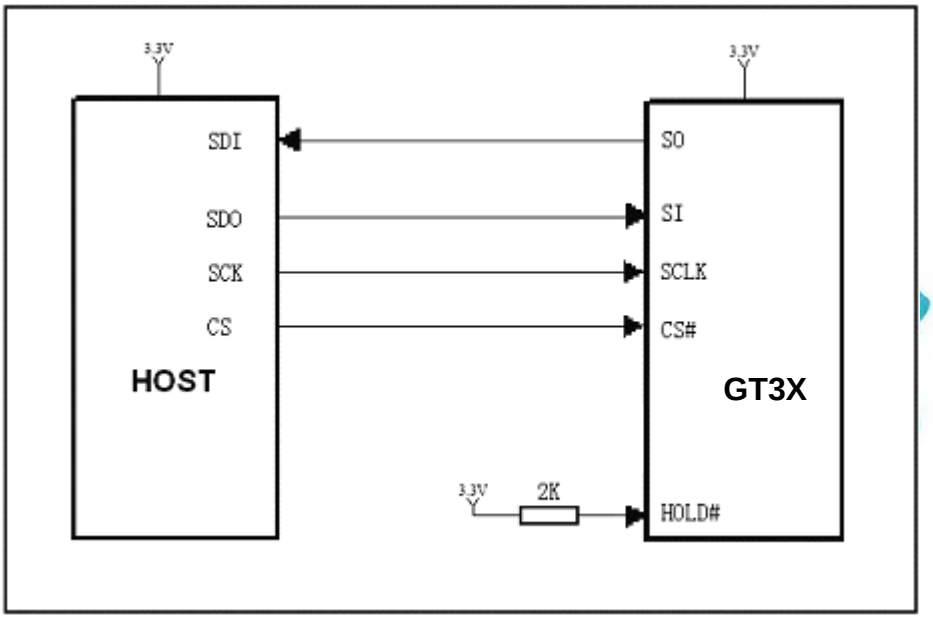
当HOLD#信号变为低并且串行时钟信号（SCLK）处于低电平时，进入总线挂起状态。

当HOLD#信号变为高并时串行时钟信号（SCLK）处于低电平时，结束总线挂起状态。



6.3 SPI 接口与主机接口参考电路示意图

SPI 与主机接口电路连接可以参考下图（#HOLD 管脚建议接 2K 电阻 3.3V 拉高）。



SPI 接口与主机接口参考电路示意图

7. 电气特性

7.1 绝对最大额定值

Symbol	Parameter	Min.	Max.	Unit	Condition
T_{OP}	Operating Temperature	-40	85	℃	
T_{STG}	Storage Temperature	-65	150	℃	
VCC	Supply Voltage	-0.3	3.6	V	
V_{IN}	Input Voltage	-0.3	VCC+0.3	V	
GND	Power Ground	-0.3	0.3	V	

7.2 DC 特性

Condition：T_{OP} =-20℃ to 70℃，GND=0V

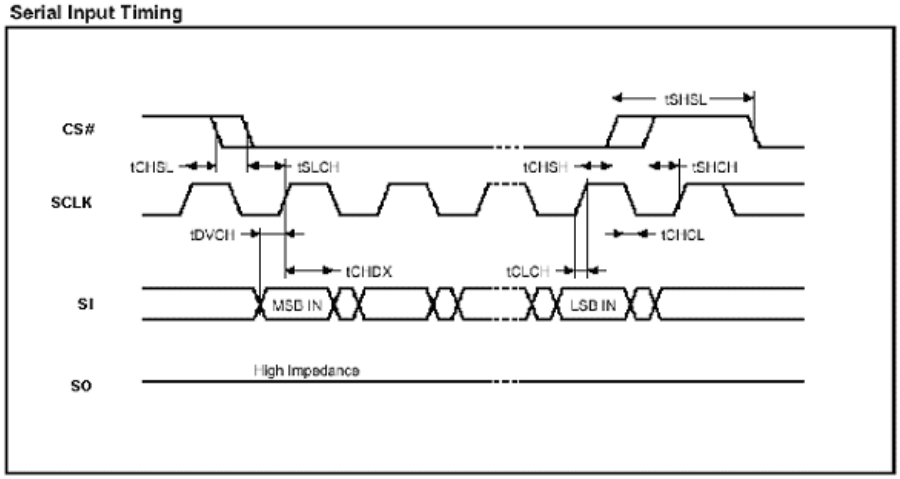
Symbol	Parameter	Min.	Max.	Unit	Condition
I_{DD}	VCC Supply Current(active)		12	mA	
I_{SB}	VCC Standby Current		10	uA	
V_{IL}	Input LOW Voltage	-0.3	0.3VCC	V	VCC=2.2~3.6V
V_{IH}	Input HIGH Voltage	0.7VCC	VCC+0.4	V	
V_{OL}	Output LOW Voltage		0.4 (I_{OL}=1.6mA)	V	
V_{OH}	Output HIGH Voltage	0.8VCC (I_{OH}=-100uA)		V	
I_{LI}	Input Leakage Current	0	2	uA	
I_{LO}	Output Leakage Current	0	2	uA	

Note：I_{IL}：Input LOW Current，I_{IH}：Input HIGH Current，
I_{OL}：Output LOW Current，I_{OH}：Output HIGH Current，

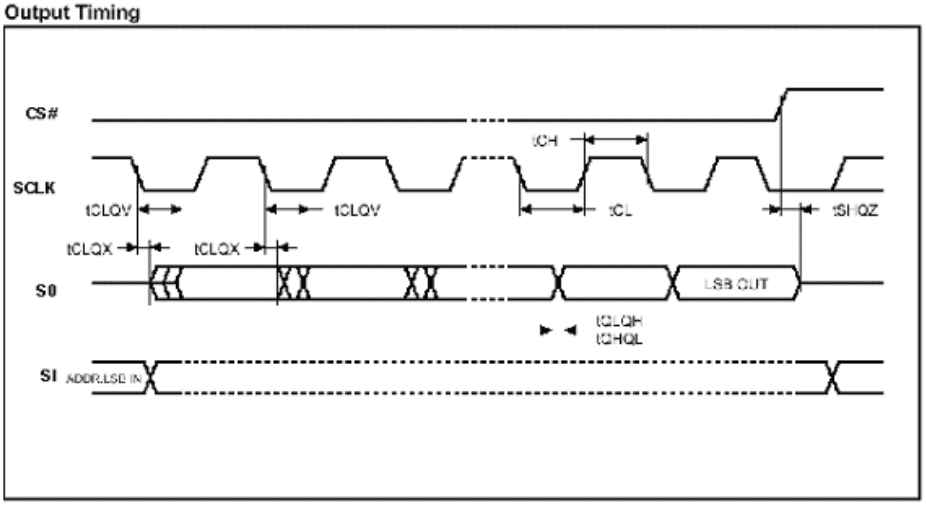
7.3 AC 特性

Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	90	MHz
tCH	tCLH	Clock High Time	15		ns
tCL	tCLL	Clock Low Time	15		ns
tCLCH		Clock Rise Time(peak to peak)	0.1		V/ns
tCHCL		Clock Fall Time (peak to peak)	0.1		V/ns
tSLCH	tCSS	CS# Active Setup Time (relative to SCLK)	5		ns
tCHSL		CS# Not Active Hold Time (relative to SCLK)	5		ns
tDVCH	tDSU	Data In Setup Time	2		ns
tCHDX	tDH	Data In Hold Time	5		ns
t CHSH		CS# Active Hold Time (relative to SCLK)	5		ns
t SHCH		CS# Not Active Setup Time (relative to SCLK)	5		ns
t SHSL	tCSH	CS# Deselect Time	100		ns
t SHQZ	tDIS	Output Disable Time		9	ns
t CLQV	tV	Clock Low to Output Valid		9	ns
t CLQX	tHO	Output Hold Time	0		ns

Serial Input Timing



Output Timing

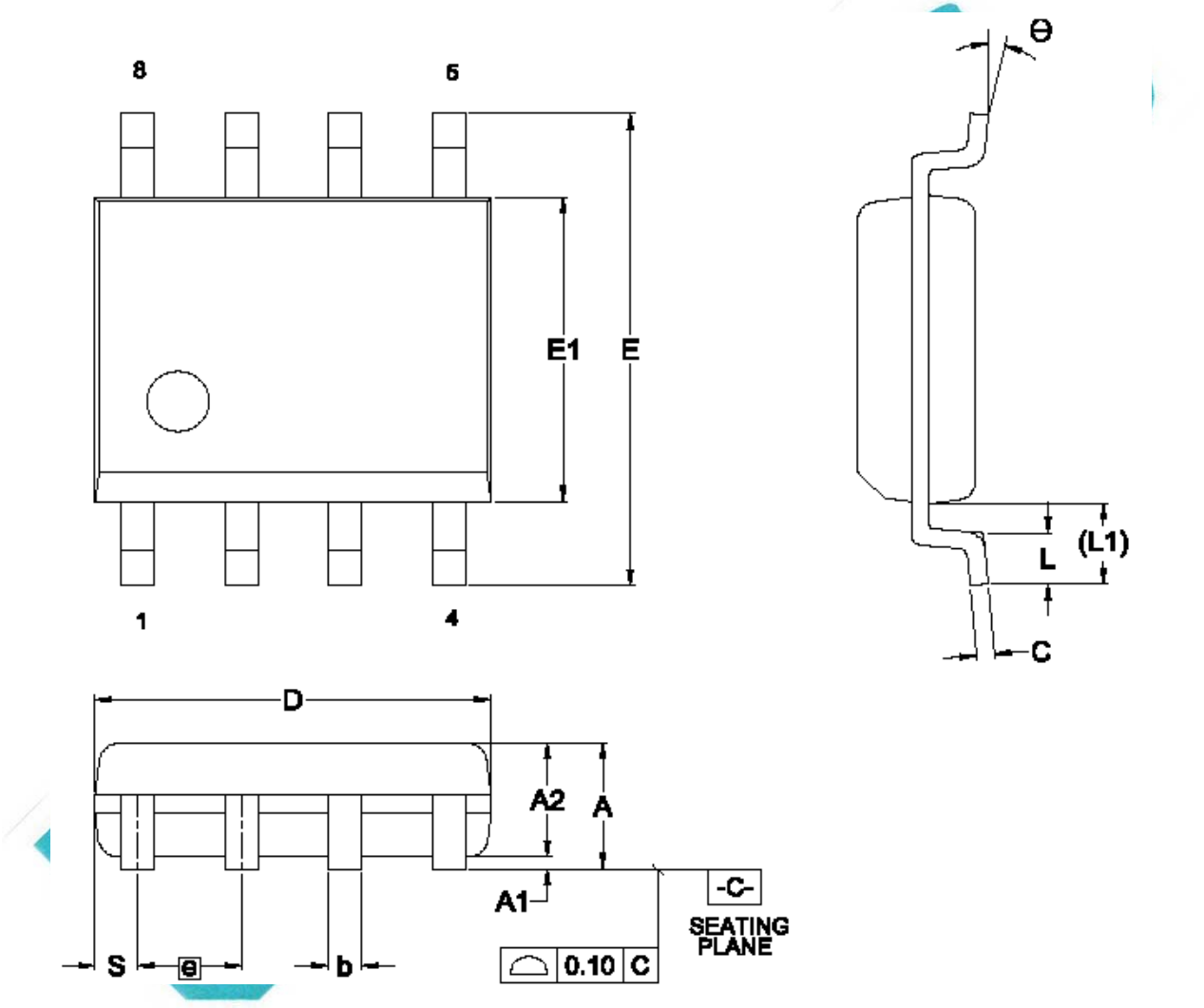


8. 封装尺寸

封 装 类 型	封装尺寸
SOP8-A	4.90mmX3.90mm（193milX154mil）
SOP8-B	5.28mmX7.90mm（206milX311mil）
DFN8-A	4.0mmx 4.0mm (158milX158mil)
DFN8-B	4.0mmx 4.0mm (158milX158mil)

Package

SOP8-A Unit :mm

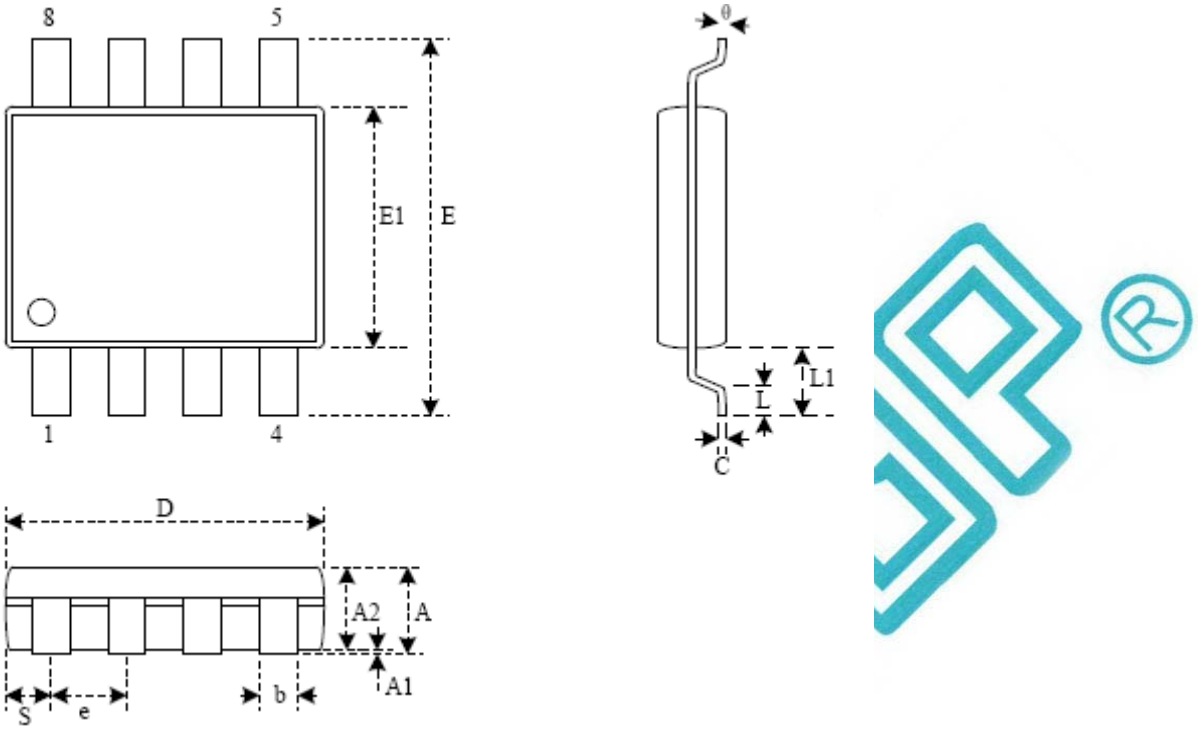


Dimensions(inch dimensions are derived from the original mm dimensions)

		A	A1	A2	b	C	D	E	E1	⊖	L	L1	S	θ
Mm	Min.	-	0.10	1.35	0.36	0.15	4.77	5.80	3.60		0.46	0.65	0.41	0
	Norm.	-	0.15	1.45	0.41	0.20	4.90	5.99	3.90	1.27	0.66	1.05	0.54	5
	Max.	1.75	0.20	1.55	0.51	0.25	5.03	6.20	4.00		0.86	1.25	0.67	8
inch	Min.	-	0.004	0.053	0.014	0.006	0.188	0.228	0.150		0.018	0.033	0.016	0
	Norm.	-	0.006	0.057	0.016	0.008	0.193	0.236	0.154	0.050	0.026	0.041	0.021	5

	Max.	0.069	0.008	0.061	0.020	0.010	0.198	0.244	0.156		0.034	0.049	0.026	8

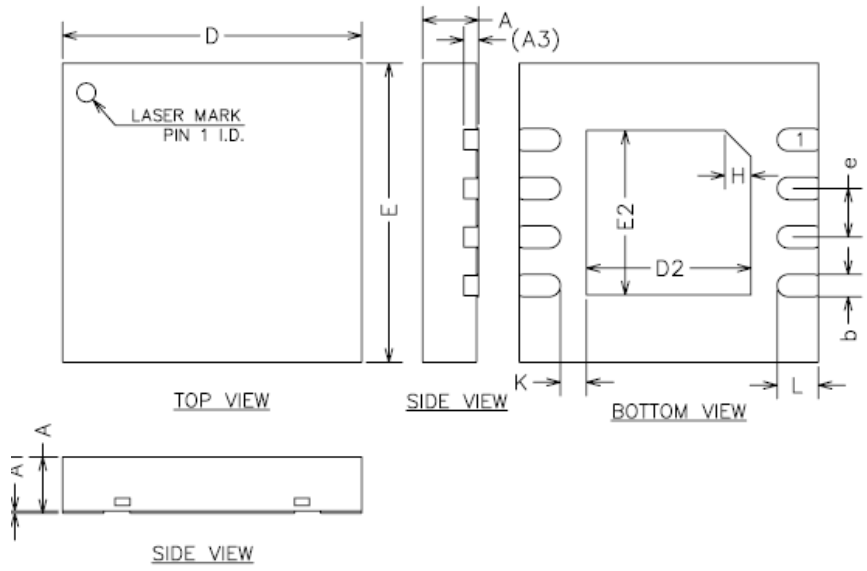
SOP8-B Unit :mm



Dimensions(inch dimensions are derived from the original mm dimensions)

		A	A1	A2	b	C	D	E	E1	⊖	L	S	θ
Mm	Min.	-	0.05	0.75	0.35	0.15	5.18	7.70	5.18		0.50	0.41	0
	Norm.	-	0.10	0.80	0.42	0.20	5.28	7.90	5.28	1.27	0.65	0.54	5
	Max.	1.0	0.15	0.85	0.48	0.25	5.38	8.10	5.38		0.80	0.67	10
inch	Min.	-	0.002	0.030	0.014	0.006	0.204	0.303	0.204		0.020	0.016	0
	Norm.	-	0.004	0.032	0.016	0.008	0.206	0.311	0.206	0.050	0.026	0.021	5
	Max.	0.04	0.006	0.034	0.020	0.010	0.210	0.319	0.210		0.031	0.026	10

DFN8-A/DFN8-B Unit :mm



COMMON DIMENSIONS
(UNITS OF MEASURE=MILLIMETER)

SYMBOL	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A3		0.20REF	
b	0.25	0.30	0.35
D	3.90	4.00	4.10
E	3.90	4.00	4.10
D2	2.10	2.20	2.30
E2	2.10	2.20	2.30
e	0.55	0.65	0.75
H		0.35REF	
K		0.35REF	
L	0.45	0.55	0.65
R	0.13	–	–