GT31L16M1Y80
标准汉字字库芯片

规格书
DATASHEET

- **字符集：GB18030**
- **字号：16x16 点阵**
- **排置方式：竖置横排**
- **总线接口：SPI 串行总线**
- **封装类型：SOP8-B**

VER1.0I_A

2012-08

版本修订记录

版本号	修改内容	日期	备注
VER1.0I_A	字库说明书的制定	2012-08	字库定制

目　　录

1 概述

GT31L16M1Y80是一款16x16点阵字库芯片，支持GB18030国标汉字（同时支持UNICODE编码）。排列格式为竖置横排。用户通过字符内码，利用用户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。

GT31L16M1Y80除含有上述字库以外，还提供客户1M字节的可自由读写空间，包括256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：0x000000~0xFFFFF。

1.1 芯片特点

- 数据总线：SPI 串行总线接口
- 点阵排列方式：竖置横排
- 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

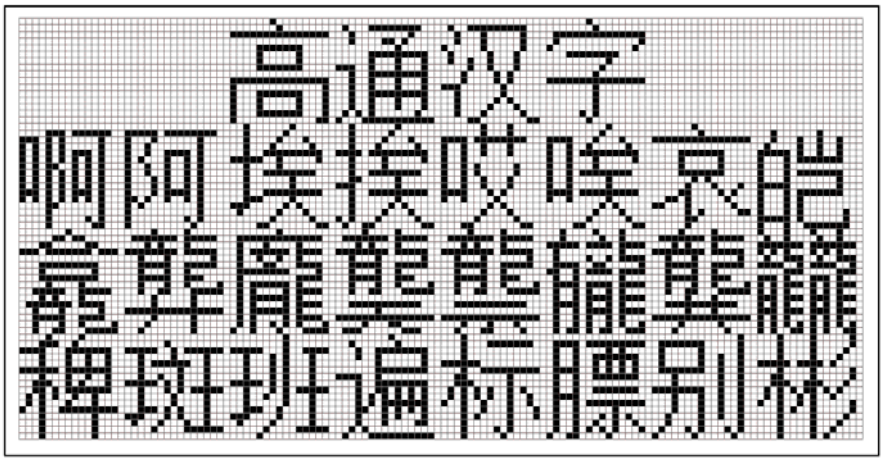
1.2 芯片内容

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	Y-竖置横排	
	ASCII	7x8	96	标准	Y-竖置横排	
	ASCII	7x10	96	打印机字体	Y-竖置横排	
	ASCII	7x12	96	打印机字体	Y-竖置横排	
	ASCII	8x16	96	标准	Y-竖置横排	
	ASCII	8x16	96	粗体	Y-竖置横排	
	ASCII	16x32	96	标准	Y-竖置横排	
	ASCII	16x32	96	粗体	Y-竖置横排	
	ASCII	16 点阵不等宽	96	方头（Arial）	Y-竖置横排	
GB18030 字符集	GB18030 汉字	16x16	27484	宋体	Y-竖置横排	
	GB18030 字符	16x16	1038	宋体	Y-竖置横排	
Unicode -> GBK 转码表	20902 + 1038					
BIG5-> GBK 转码表	13468					

1.3 字型样张

1.3.1 汉字字符

16x16 点阵 GB18030 汉字

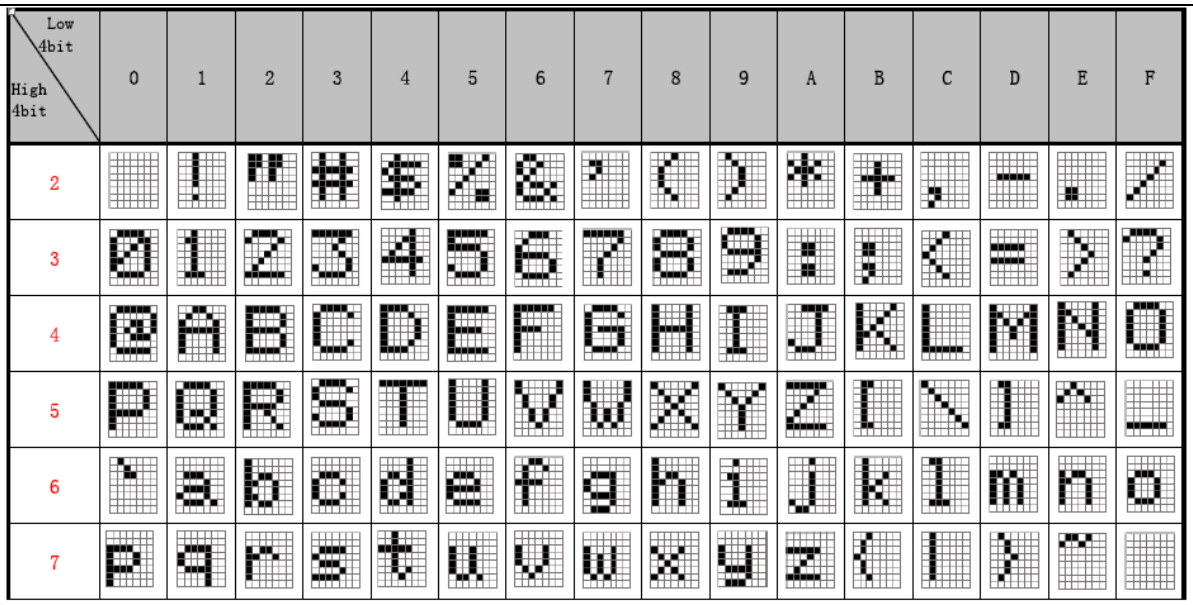


1.3.2 其它点阵字符

5x7 点阵 ASCII 标准字符

Low 4bit / High 4bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

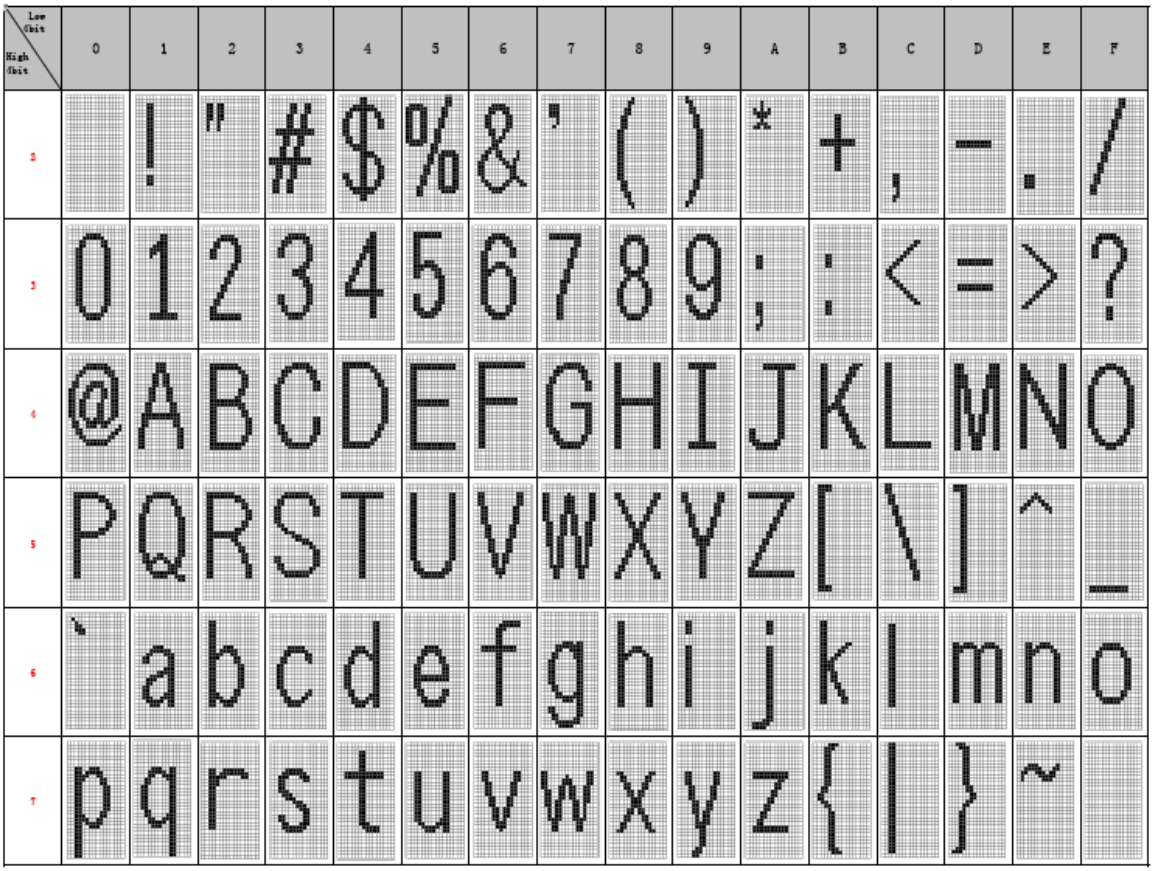
7x8 点阵 ASCII 标准字符



8x16 点阵 ASCII 标准字符

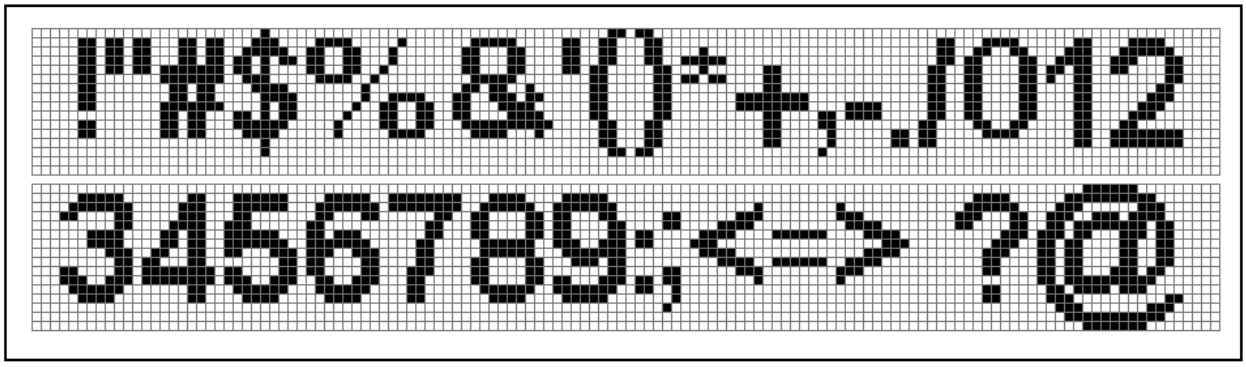
8x16 点阵 ASCII 粗体字符

16x32 点阵 ASCII 标准字符



16x32 点阵 ASCII 粗体字符

16 点阵不等宽 ASCII 方头（Arial）



2 操作指令

2.1 Instruction Parameter(指令参数)

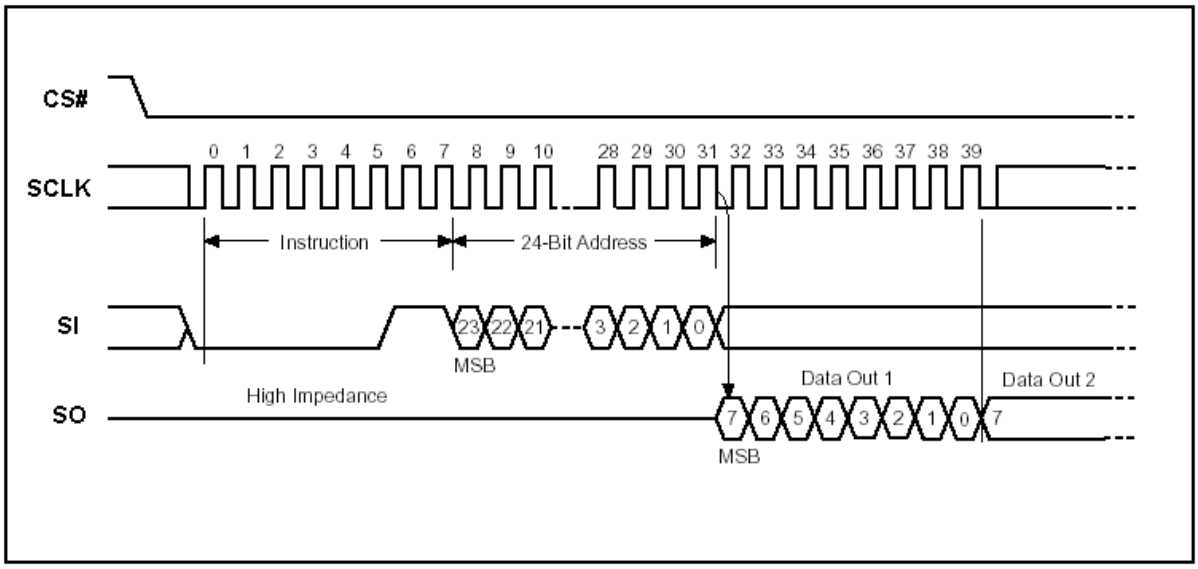
Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
Read	Read Data Bytes	0000 0011	03 h	3	—	1 to ∞
Fast Read	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞
WREN	Write Enalbe	0000 0110	06 h	—	—	—
WRDI	Write Disable	0000 0100	04 h			
PP	Page Program	0000 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:



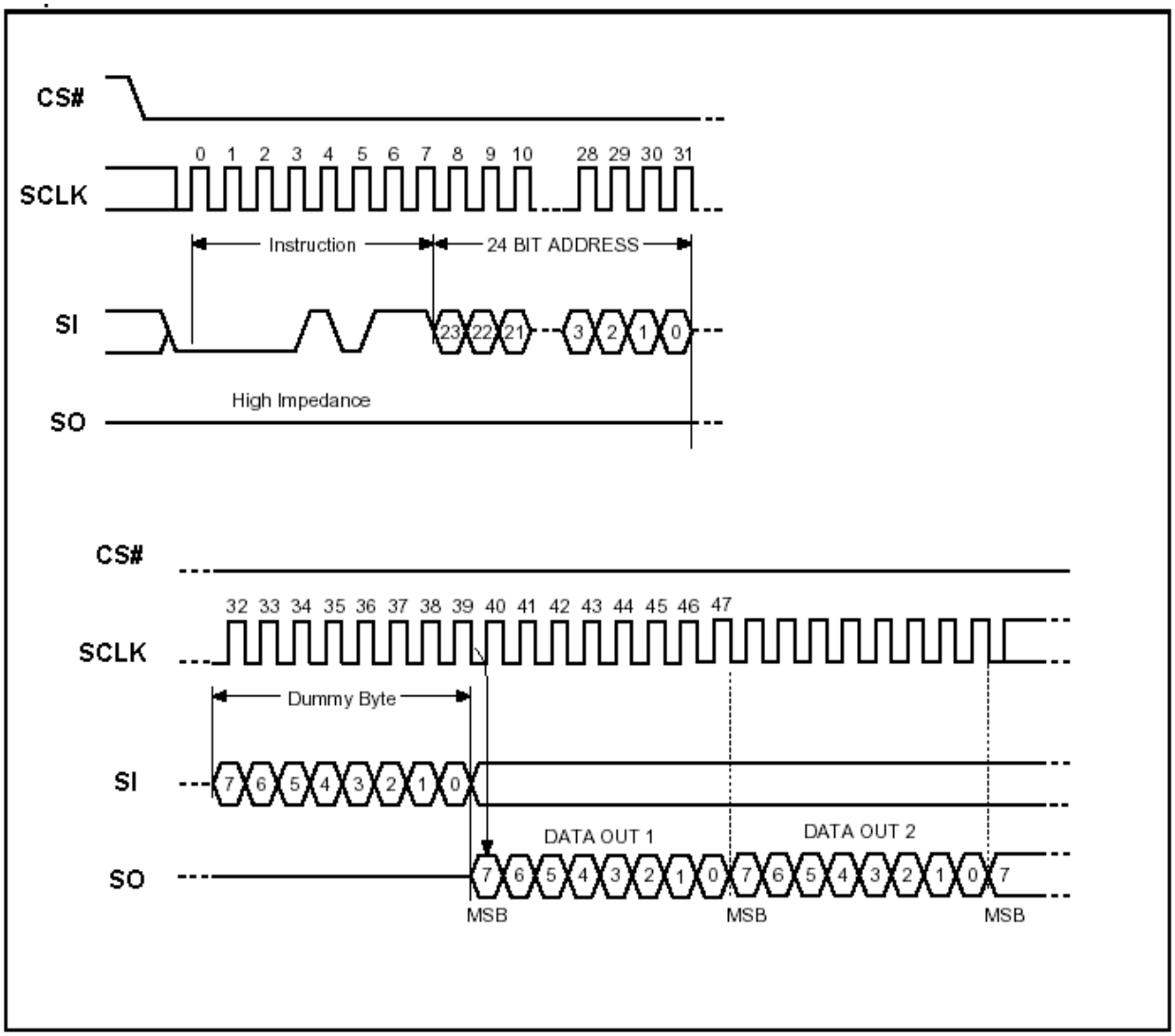
2.3 Read Data Bytes at Higher Speed（快速读取点阵数据）

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。

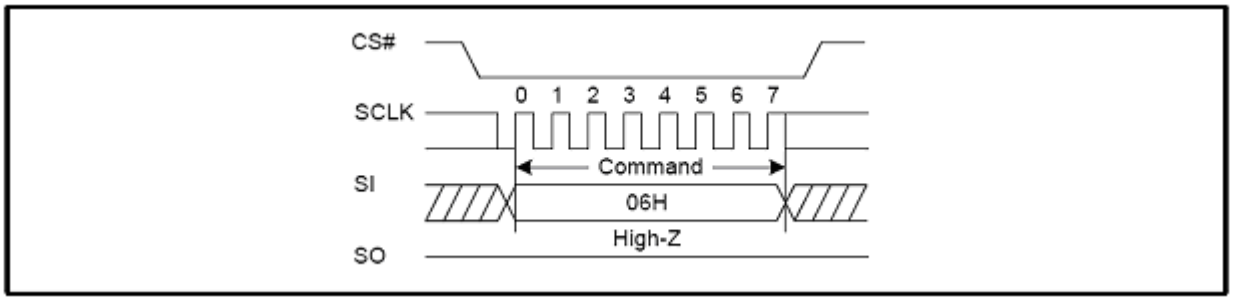
图：Read Data Bytes at Higher Speed (READ_FAST) Instruction Sequence and Data-out sequence:



2.4 Write Enable（写使能）

Write Enable 指令的时序如下(图)：

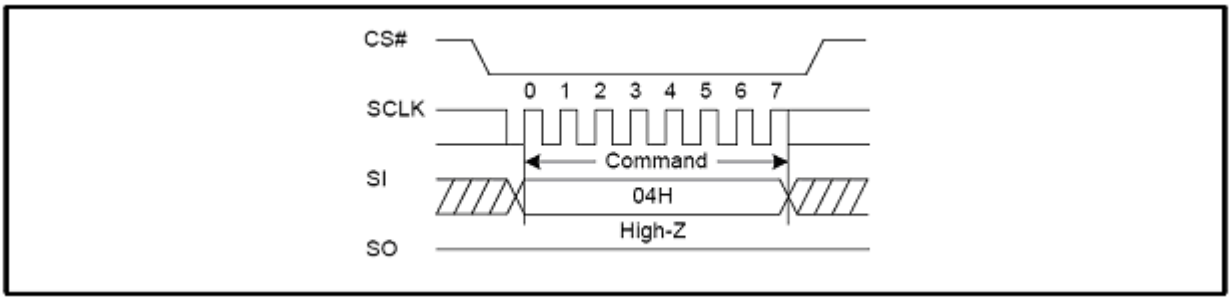
CS#变低-→发送 Write Enable 命令—>CS#变高



2.5 Write Disable（写非能）

Write Enable 指令的时序如下(图)：

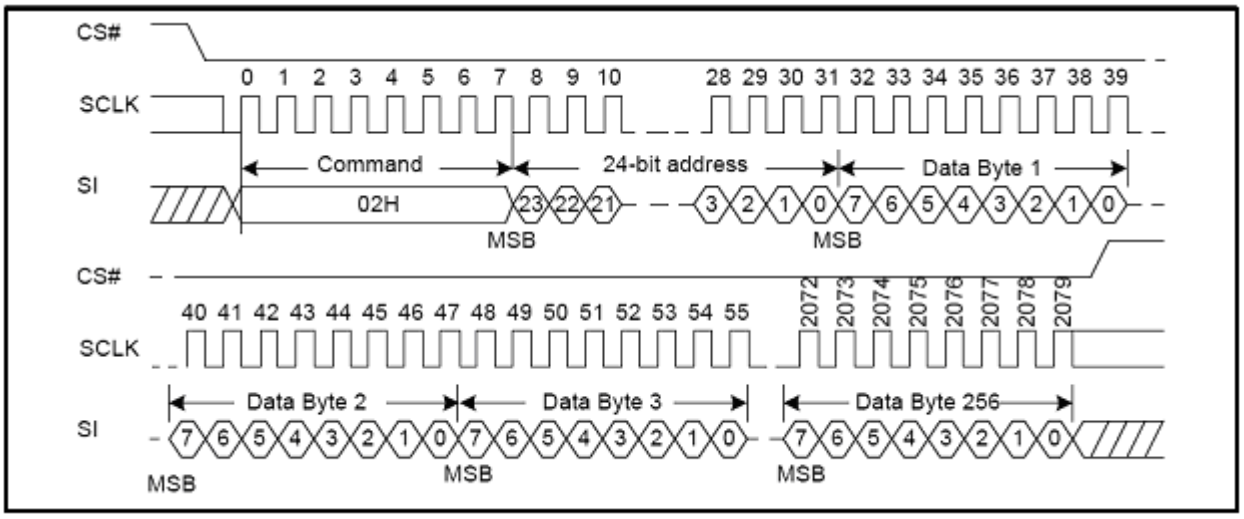
CS#变低-→发送 Write Disable 命令—>CS#变高



2.6 Page Program（页写入）

Page Program 指令的时序如下(图)：

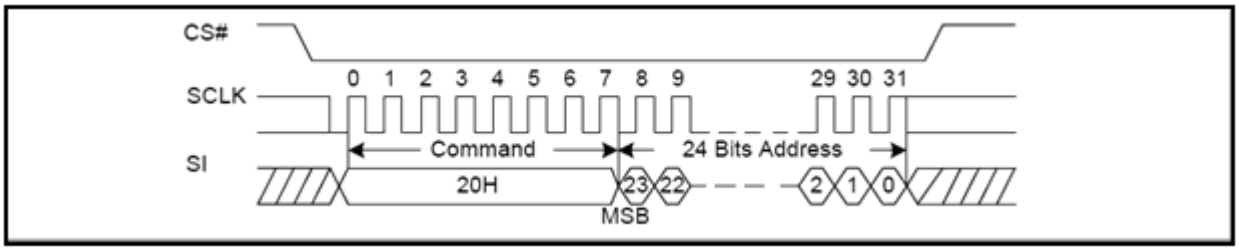
CS#变低-→发送 Page Program 命令→发送 3 字节地址—>发送数据—>CS#变高



2.7 Sector Erase（扇区擦除）

Sector Erase 指令的时序如下(图)：

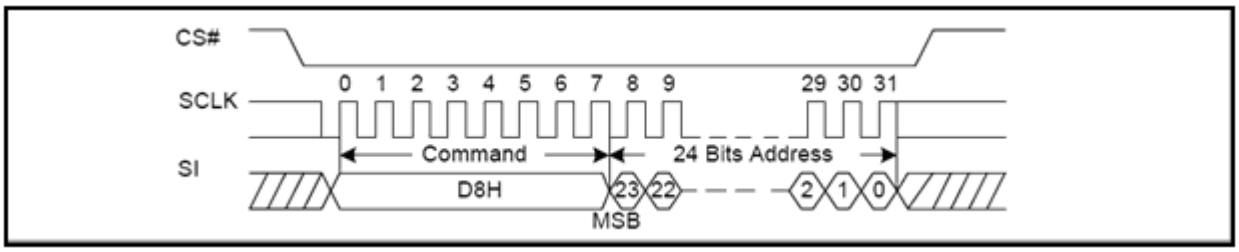
CS#变低->发送 Sector Erase 命令->发送 3 字节地址—>CS#变高



2.8 Block Erase(64K)（块擦除）

Block Erase 指令的时序如下(图)：

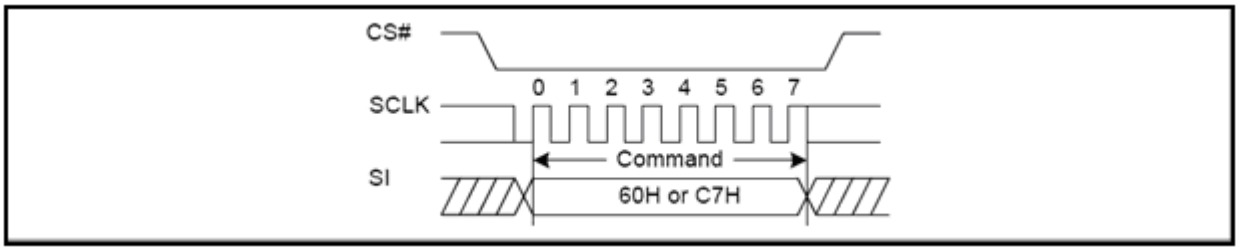
CS#变低->发送 64K Block Erase 命令->发送 3 字节地址—>CS#变高



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.	内容	编码体系	字符数	起始地址	参考算法
1	5x7 点阵 ASCII 标准字符	ASCII	96	0x100000	4.2.1
2	7x8 点阵 ASCII 标准字符		96	0x100300	4.2.2
3	8x16 点阵 ASCII 标准字符		96	0x100600	4.2.3
4	16 点阵不等宽（Arial）字符		96	0x100C00	4.2.7
5	16x32 点阵 ASCII 标准字符		96	0x1F64FC	4.2.8
6	16x32 点阵 ASCII 粗体字符		96	0X1F8320	4.1.9
7	8x16 点阵 ASCII 粗体字符		96	0x1F7CFC	4.2.6
8	7x10 点阵 ASCII 打印机字符		96	0x1EF384	4.2.4
9	7x12 点阵 ASCII 打印机字符		96	0x1EF744	4.2.5
10	16x16 点阵 GB18030 汉字&字符	GB18030	27484+ 1038	0x1018C0	4.1.1
11	Unicode→GBK 转码表		20902+ 1038	0x1E0CC0	4.3.1
12	BIG5 转 GB 转码表		13468	0x1EFBC4	4.2.10
13	保留区			0x1F9B20	

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 16x16 点阵 GB18030 字库

16X16 点阵字库地址分配(字节地址)：0-914400（10进制）
地址的计算由下面的函数实现（ANSI C 语言编写）
函数：unsigned long gt(unsigned char c1, unsigned char c2, unsigned char c3, unsigned char c4)
功能：计算汉字点阵在芯片中的地址
参数：c1,c2,c3,c4：4 字节汉字内码通过参数c1,c2,c3,c4 传入，双字节内码通过参数c1,c2 传入，c3=0,c4=0
返回：汉字点阵的字节地址。
例如：BaseAdd: 说明汉字点阵数据在字库芯片中的起始地址，即BaseAdd＝0x1018C0;
“啊”字的内码为0xb0a1,则byte address = gt(0xb0,0xa1,0x00,0x00) ;

“丄”字的内码为0x8139ee39,则byte address = gt(0x81,0x39,0xee,0x39);

Section X：表示第X区字符。

```
unsigned long gt (unsigned char c1, unsigned char c2, unsigned char c3, unsigned char c4)
{
   unsigned long h=0;
   if(c2==0x7f) return (h);

   if(c1>=0xA1 && c1 <= 0Xa9 && c2>=0xa1) //Section 1
      h= (c1 - 0xA1) * 94 + (c2 - 0xA1);

   else if(c1>=0xa8 && c1 <= 0xa9 && c2<0xa1) //Section 5
   {
      if(c2>0x7f)
      c2--;
      h=(c1-0xa8)*96 + (c2-0x40)+846;
   }

if(c1>=0xb0 && c1 <= 0xf7 && c2>=0xa1) //Section 2
      h= (c1 - 0xB0) * 94 + (c2 - 0xA1)+1038;

else if(c1<0xa1 && c1>=0x81 && c2>=0x40 ) //Section 3
{
   if(c2>0x7f)
   c2--;
   h=(c1-0x81)*190 + (c2-0x40) + 1038 +6768;
}
```

```
else if(c1>=0xaa && c2<0xa1) //Section 4
{
   if(c2>0x7f)
        c2--;
   h=(c1-0xaa)*96 + (c2-0x40) + 1038 +12848;
}

else if(c1==0x81 && c2>=0x39) //四字节区1
{
     h =1038 + 21008+(c3-0xEE)*10+c4-0x39;
}

else if(c1==0x82)//四字节区2
{
     h =1038 + 21008+161+(c2-0x30)*1260+(c3-0x81)*10+c4-0x30;
}

return(h*32+BaseAdd);
}
```

4.2 ASCII 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）

addr：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd= 0x100000
if(ASCIICODE >=0x20 && ASCIICODE <=0x7F)
     addr =(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）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x100300
          if(ASCIICODE >=0x20 && ASCIICODE <=0x7F)
          Address =(ASCIICODE-0x20)*8 + BaseAdd
```

4.2.3 8x16 点阵 ASCII 标准字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x100600

```
if((ASCIICode >=0x20)&&( ASCIICode <=0x7F))
addr = (ASCIICode -0x20)*16+ BaseAdd
```

4.2.4 10x8 点阵 ASCII 打印机字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x1EF384

```
if((unicode>=0x20)&&(unicode<=0x7F))
addr = (ASCIICode -0x20)*10+ BaseAdd
```

4.2.5 12x8 点阵 ASCII 打印机字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x1EF744

```
if((ASCIICode >=0x20)&&( ASCIICode <=0x7F))
addr = (ASCIICode -0x20)*12+ BaseAdd
```

4.2.6 8x16 点阵 ASCII 粗体字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x1F7CFC;

```
if((ASCIICode >=0x20)&&( ASCIICode <=0x7F))
addr = (ASCIICode -0x20)*16+ BaseAdd
```

4.2.7 16 点阵不等宽 ASCII 方头（Arial）字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

Address：ASCII 字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x100C00

```
if((ASCIICode >=0x20)&&( ASCIICode <=0x7F))
addr = (ASCIICode -0x20)*34+ BaseAdd
```

4.2.8 16x32 点阵 ASCII 标准字符

参数说明：

ASCIICode：表示 ASCII码（8bits）

Address：ASCII字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x1F64FC

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7F)

addr = (ASCIICode –0x20) * 64 + BaseAdd

4.2.9 16X32 点 ASCII 字符（粗体）

参数说明：

ASCIICode：表示 ASCII码（8bits）

Address：ASCII字符点阵在芯片中的字节地址。

BaseAdd：表示点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0x1F8320

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7F)

addr = (ASCIICode –0x20) * 64 + BaseAdd

4.2.10 8x16 点阵字符保留区

参数说明：

uncode：表示unicode码（8bits）

Address：unicode字符点阵在芯片中的字节地址。

计算方法：

if (unicode == 0xa5) then

address= 0x1F82FC;

4.3 内码转换程序

4.3.1 Unicode 转 GBK 转换算法

地址的计算由下面的函数实现（ANSI C 语言编写）

函数：DWORD U2GB(WORD UCODE)

功能：计算UNICODE对应GB码存放的起始地址

参数：WORD UCODE 输入unicode码双字节

返回：对应GB码存放地址,以大端模式存放。

```
DWORD U2GB(WORD UCODE)
{
    DWORD part_addr;
```

```
DWORD base_addr=0x1E0CC0;
DWORD addr;
WORD UCODE;
if(UCODE<=0x0451&&UCODE>=0x00a0)
{
    part_addr=0;
    addr=part_addr+(UCODE-0x00a0)*2;

}
if(UCODE<=0x2642&&UCODE>=0x2010)
{
    part_addr=1892;
    addr=part_addr+(UCODE-0x2010)*2;

}
if(UCODE<=0x33d5&&UCODE>=0x3000)
{
    part_addr=5066;
    addr=part_addr+(UCODE-0x3000)*2;
}
if(UCODE<=0x9fa5&&UCODE>=0x4e00)
{
    part_addr=7030;
    addr=part_addr+(UCODE-0x4e00)*2;
}
if(UCODE<=0xfe6b&&UCODE>=0xfe30)
{
    part_addr=48834;
    addr=part_addr+(UCODE-0xfe30)*2;
}
if(UCODE<=0xff5e&&UCODE>=0xff01)
{
    part_addr=48954;
    addr=part_addr+(UCODE-0xff01)*2;
}
if(UCODE<=0xffe5&&UCODE>=0xffe0)
{
    part_addr=49142;
    addr=part_addr+(UCODE-0xffe0)*2;
}
//新增部分


if(UCODE<=0xFA29&&UCODE>=0XF92C)
{
    part_addr=49312;
```

```
        addr=part_addr+(UCODE-0XF92C)*2;
    }
    if(UCODE<=0XE864&&UCODE>=0XE816)
    {
        part_addr=49820;
        addr=part_addr+(UCODE-0XE816)*2;
    }
    if(UCODE<=0X2ECA&&UCODE>=0X2E81)
    {
        part_addr=49978;
        addr=part_addr+(UCODE-0X2E81)*2;
    }
    if(UCODE<=0X49B7&&UCODE>=0X4947)
    {
        part_addr=50126;
        addr=part_addr+(UCODE-0X4947)*2;
    }
    if(UCODE<=0X4DAE&&UCODE>=0X4C77)
    {
        part_addr=50352;
        addr=part_addr+(UCODE-0X4C77)*2;
    }
    if(UCODE<=0X3CE0&&UCODE>=0X3447)
    {
        part_addr=50976;
        addr=part_addr+(UCODE-0X3447)*2;
    }
    if(UCODE<=0X478D&&UCODE>=0X4056)
    {
        part_addr=55380;
        addr=part_addr+(UCODE-0X4056)*2;
    }
return addr+base_addr;
}
```

4.3.2 BIG5 转 GB18030 转换算法

地址的计算由下面的函数实现（ANSI C 语言编写）

函数：unsigned long BIG5_G(unsigned short GBcode)

功能：计算BIG5码对应GB码存放的起始地址

参数：WORD UCODE 输入unicode码双字节

返回：对应GB码存放地址，对应GB码高位在前，低位在后。

```
unsigned long BIG5_G(unsigned short B5code)
{

    unsigned char TMP,B5_MSB,B5_LSB;
    B5_MSB= B5code >>8;
```

```
    B5_LSB= B5code &0x00ff;
    unsigned long part_addr,BaseAddr=0x1EFBC4;//转码表基地址;
    if(B5_MSB>=0xa1&&B5_MSB<=0xa3)
    {
        part_addr=0;
        TMP=0xa1;
    }

    else if(B5_MSB>=0xa4&&B5_MSB<=0xc6)
    {
        part_addr=816;
        TMP=0xa4;
    }

    else if(B5_MSB>=0xc9&&B5_MSB<=0xf9)
    {
        part_addr=11618;
        TMP=0xc9;
    }

    if(B5_LSB<=0x7e&&B5_LSB>=0x40)
    {
        return(part_addr+((B5_MSB-TMP)*157+B5_LSB-0x40)*2+BaseAddr);
    }

    else if(B5_LSB<=0xfe&&B5_LSB>=0xa1)
    {
        return(part_addr+((B5_MSB-TMP)*157+B5_LSB-0xa1+63)*2+BaseAddr);
    }
}
```

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	
1M	64K	4K	256	字节
4K	256	16		页
256	16			扇区
16				块

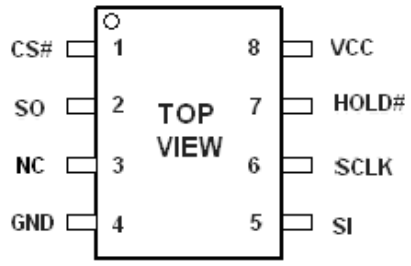
5.2 存储块、扇区结构

块	扇区	地址范围	
15	255	0x0FF000	0x0FFFFF
	.....	.....	.....
	240	0x0F0000	0x0F0FFF
14	239	0x0EF000	0x0EFFFF
	.....	.....	.....
	224	0x0E0000	0x0E0FFF
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
.....	.....	.....	.....
	.....	.....	.....
	.....	.....	.....
2	47	0x02F000	0x02FFFF
	.....	.....	.....
	32	0x020000	0x020FFF
1	31	0x01F000	0x01FFFF
	.....	.....	.....
	16	0x010000	0x010FFF
0	15	0x00F000	0x00FFFF
	.....	.....	.....
	0	0x000000	0x000FFF

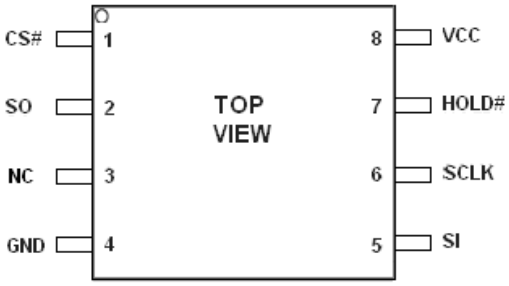
6 引脚描述与电路连接

6.1 引脚配置

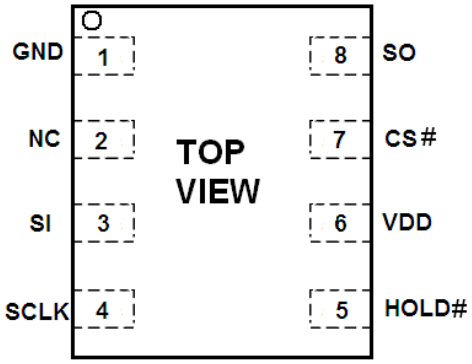
SOP8-A



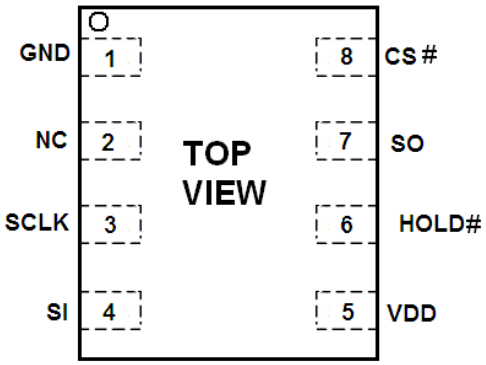
SOP8-B



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)

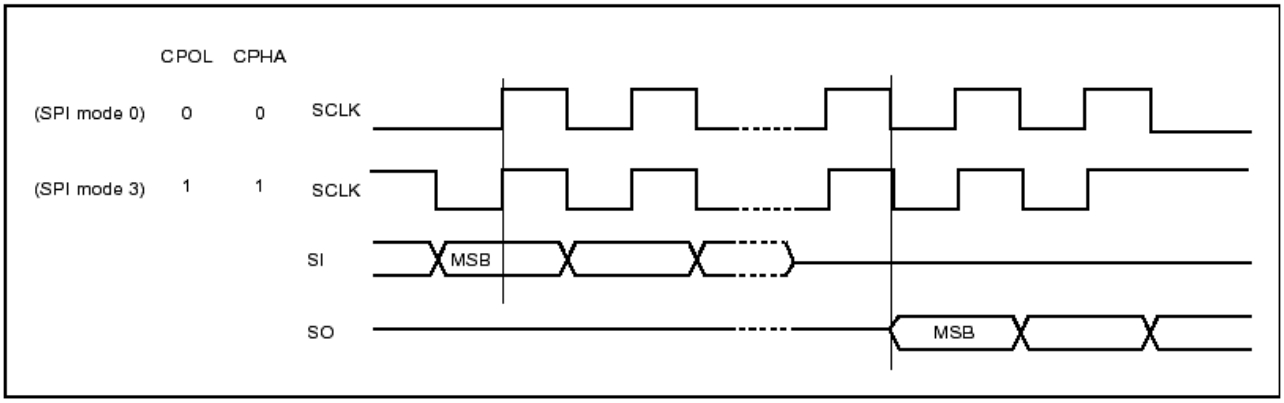
DFN8-A

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。

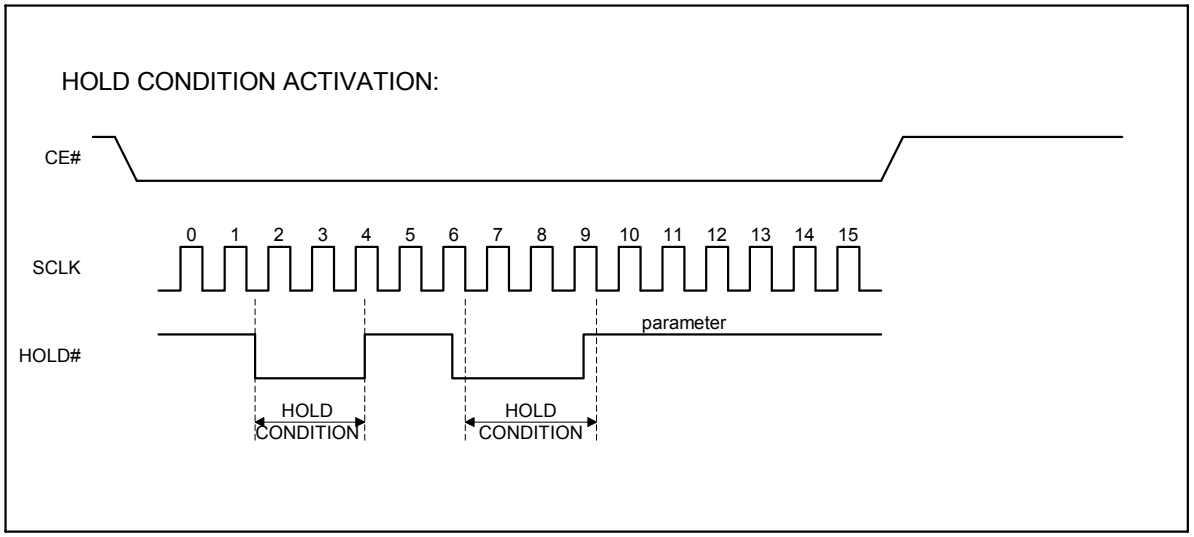


总线挂起输入（HOLD#）：

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。

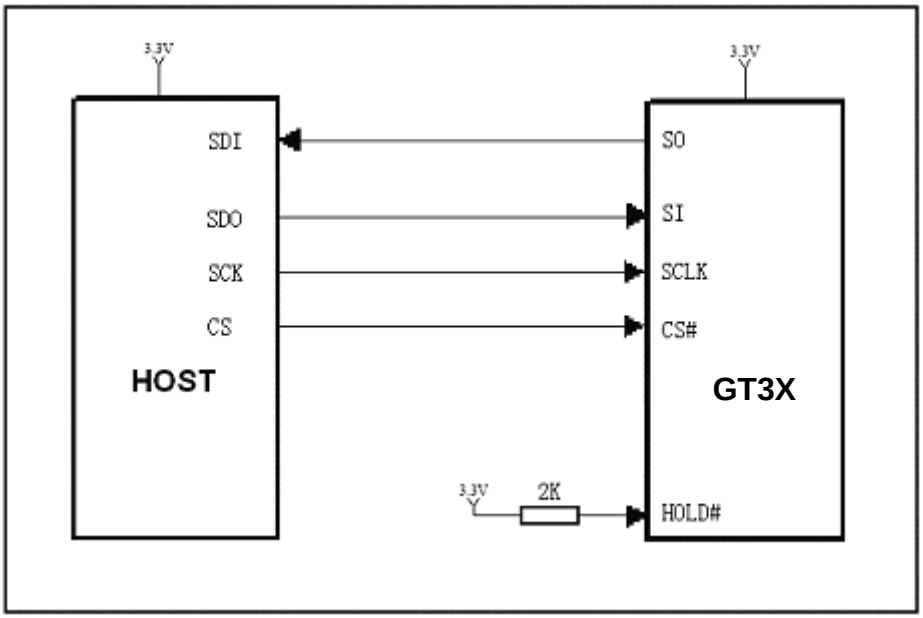
当HOLD#信号变为低并且串行时钟信号（SCLK）处于低电平时，进入总线挂起状态。

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

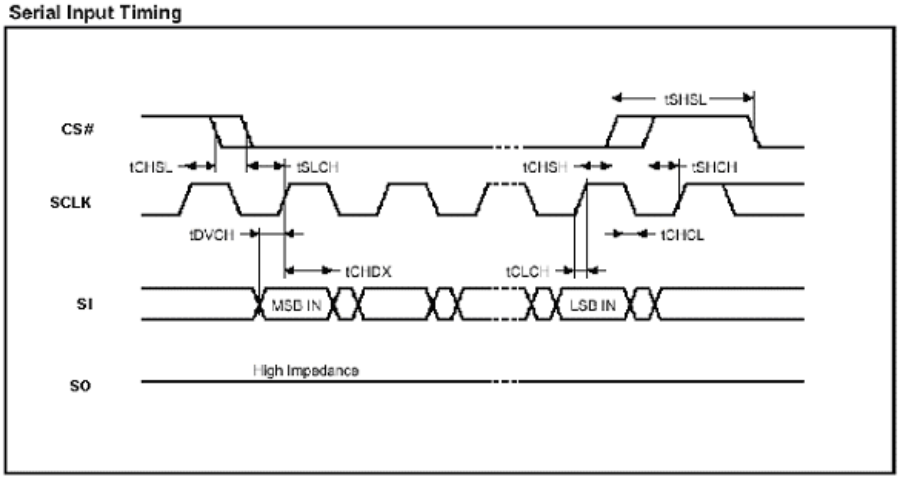
Symbol	Parameter	Min.	Max.	Unit	Condition
I_{DD}	VCC Supply Current(active)		12	mA	
I_{SB}	VCC Standby Current		10	uA	
V_{IL}	Input LOW Voltage	-0.3	0.3VCC	V	VCC=2.2~3.6V
V_{IH}	Input HIGH Voltage	0.7VCC	VCC+0.4	V	
V_{OL}	Output LOW Voltage		0.4 (I_{OL}=1.6mA)	V	
V_{OH}	Output HIGH Voltage	0.8VCC (I_{OH}=-100uA)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

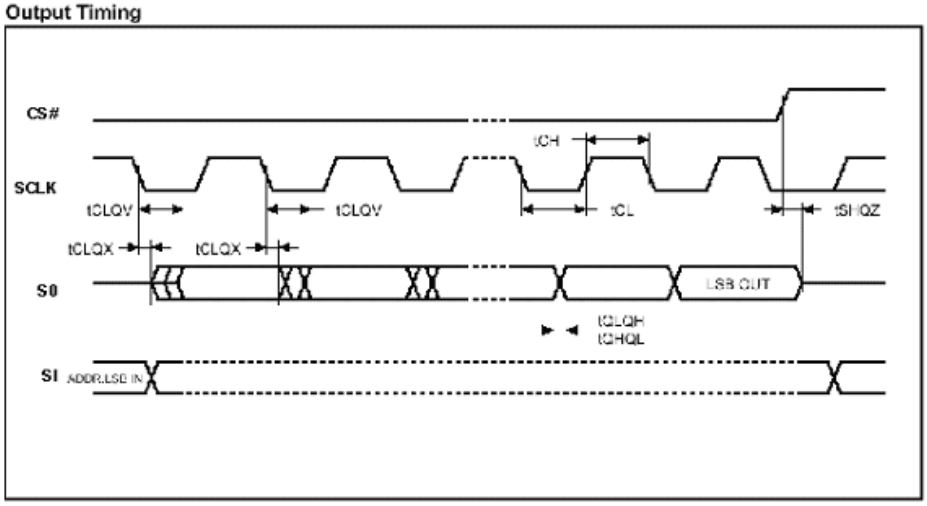
7.3 AC 特性

Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	90	MHz
t_{CH}	t_{CLH}	Clock High Time	15		ns
t_{CL}	t_{CLL}	Clock Low Time	15		ns
t_{CLCH}		Clock Rise Time(peak to peak)	0.1		V/ns
t_{CHCL}		Clock Fall Time (peak to peak)	0.1		V/ns
t_{SLCH}	t_{CSS}	CS# Active Setup Time (relative to SCLK)	5		ns
t_{CHSL}		CS# Not Active Hold Time (relative to SCLK)	5		ns
t_{DVCH}	t_{DSU}	Data In Setup Time	2		ns
t_{CHDX}	t_{DH}	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}	t_{CSH}	CS# Deselect Time	100		ns
t_{SHQZ}	t_{DIS}	Output Disable Time		9	ns
t_{CLQV}	t_V	Clock Low to Output Valid		9	ns
t_{CLQX}	t_{HO}	Output Hold Time	0		ns

Serial Input Timing



Output Timing

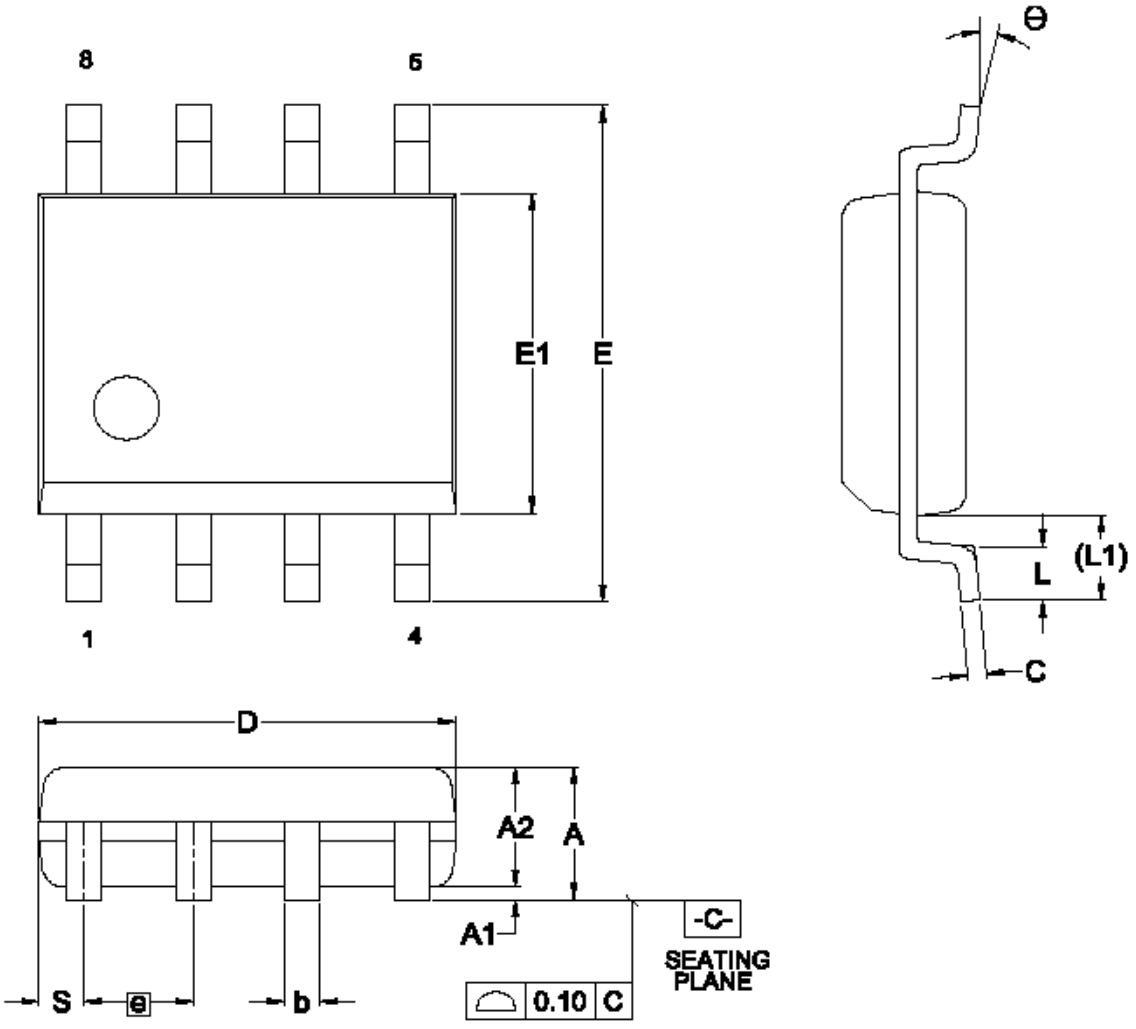


8 封装尺寸

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

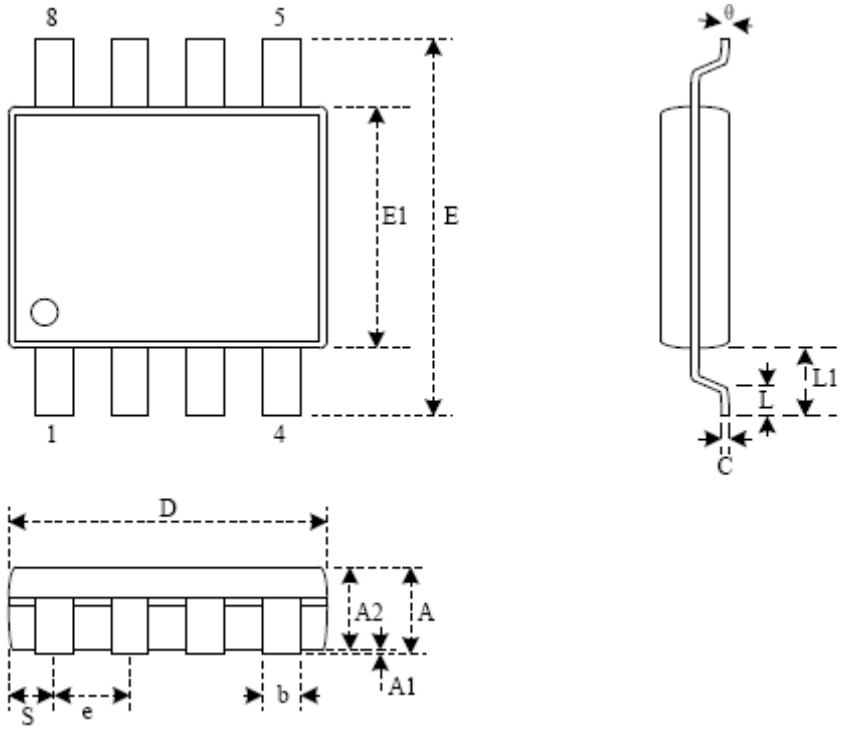


Dimensions(inch dimensions are derived from the original mm dimensions)

		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

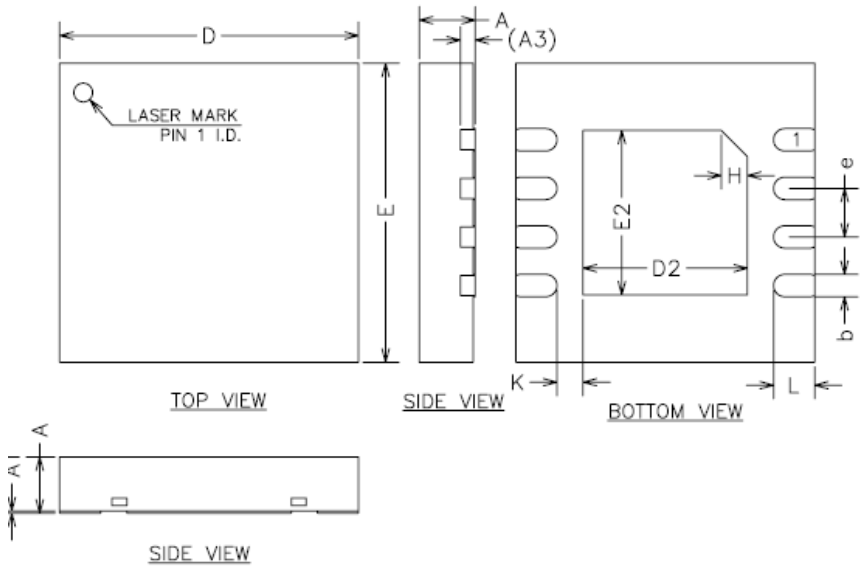
SOP8-B Unit :mm



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	–	–