GT23L24T3Y
标准汉字字库芯片

用户手册
DATASHEET

- 字型： 11X12 点阵、15X16 点阵、24X24 点阵
- 汉字字符集：GB2312、GB12345、BIG5
- 兼容 Unicode 内码
- 输入法码本：GT 三维部件输入法（GB2312 字符集）

  GT 快捷拼音输入法（GB2312 字符集）

  GT 三维部件输入法（BIG5 字符集）

  GT 注音输入法（BIG5 字符集）
- 排置方式：竖置横排
- 总线接口：SPI 串行总线

  PLII 精简地址并行总线
- 芯片形式：SO20W 封装

VER 3.6

2010-Q3

版本修订记录

版本号	修改内容	日期	备注
V35	1. 11X12 点 BIG5 字符集汉字字符算法。	2010-7	
	2. 15X16 点 BIG5 字符集汉字字符算法。	2010-7	
	3. 24X24 点 BIG5 基本集汉字字符算法。	2010-7	
	4. 11X12 点 GB12345 字符集汉字字符算法。	2010-7	
	5. 15X16 点 GB12345 字符集汉字字符算法。	2010-7	
	6. 24X24 点 GB12345 基本集汉字字符算法。	2010-7	
V36	7. 24X24 点 unicode 字符集汉字字符算法。	2010-8	
	8.更新 GB2312 1 区字符 (470 字符)。	2010-8	
	9.更新 12 点 GB2312 算法（字符区）。	2010-8	
	10.更新 16 点 GB2312 算法（字符区）。	2010-8	
	11.更新 24 点 GB2312 算法（字符区）。	2010-8	
	12.更新 16 点阵不等宽 ASCII 方头的存储结构图。	2010-8	
	13.附录中增加补丁文件列表。	2010-8	

目　　录

第一部分:硬件部分

第二部分:软件部分

GT23L24T3Y　　多字符集简繁　字库芯片　　标准字库

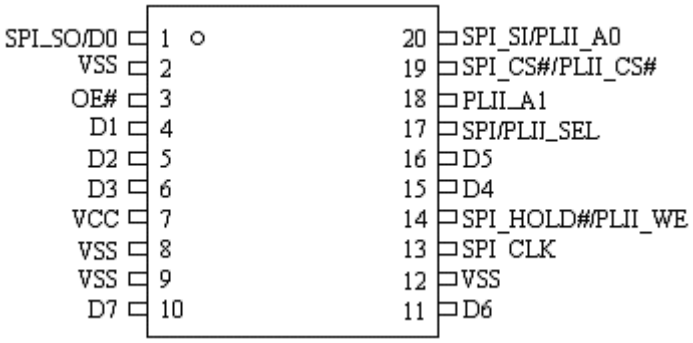
1 概述

GT23L24T3Y是一款内含11X12点阵、15X16点阵、24X24点阵的汉字库芯片，支持GB2312国标汉字（含有国家信标委合法授权）、GB12345国标繁体汉字（含国家信标委合法授权）、BIG5字符集汉字及ASCII字符。并兼容Unicode编码格式。排列格式为竖置横排。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。

本字库芯片内含 GT 快捷拼音输入法、GT 三维部件输入法及 GT 注音输入法的码本，另外配合本公司的输入法程序，可实现数字小键盘 IT 产品的汉字快捷输入。

1.1 芯片特点

- 数据总线： SPI 串行总线接口
  PLII 精简地址并行总线接口
- 点阵排列方式：字节竖置横排
- 访问速度：SPI 时钟频率：20MHz(max.)
  PLII 访问速度：130ns(max.) @3.3V
- 工作电压：2.7V~3.6V
- 电流：
  工作电流：12mA
  待机电流：10uA
- 封装：SO20W
- 尺寸（SO20W）：12.80mmX10.30mm
- 工作温度：-20℃~85℃(SPI 模式下)；-10℃~85℃(PLII 模式下)



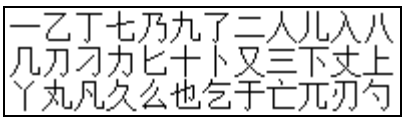
1.2 芯片内容

分类	字库内容	编码体系（字符集）	字符数
汉字及字符	11X12 点阵 GB2312 标准点阵字库	GB2312	6763+470
	11X12 点阵 GB12345 标准点阵字库	GB12345	6866+662
	11X12 点阵 BIG5 点阵字库	BIG5	13060+408
	15X16 点阵 GB2312 标准点阵字库	GB2312	6763+470
	15X16 点阵 GB12345 标准点阵字库	GB12345	6866+662
	15X16 点阵 BIG5 点阵字库	BIG5	13060+408
	24X24 点阵 GB2312 标准点阵字库	GB2312	6763+470
	24X24 点阵 GB12345 标准点阵字库	GB12345	6866+662
	24X24 点阵 BIG5 基本集点阵字库	BIG5 基本集	5401+408
ASCII 字符	5X7 点 ASCII 字符	ASCII	96
	7X8 点 ASCII 字符	ASCII	96
	6X12 点 ASCII 字符	ASCII	96
	8X16 点 ASCII 字符	ASCII	96
	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	24 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
内码索引表	GB12345 汉字内码字符索引表	GB12345	
	BIG5 汉字内码字符索引表	BIG5	
	Unicode 汉字内码字符索引表	Unicode	
输入法码表	GT 快捷拼音输入法	GB2312	
	GT 三维部件输入法	GB2312	
	GT 三维部件输入法	BIG5	
	GT 注音输入法	BIG5	

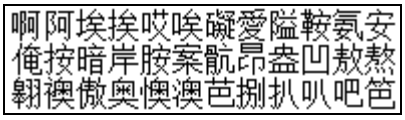
字型样张

GB2312

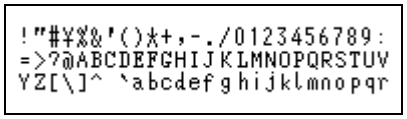
BIG5



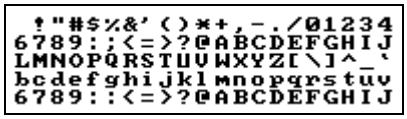
GB12345



5x7 DOT ASCII

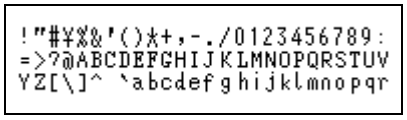


7x8 DOT ASCII

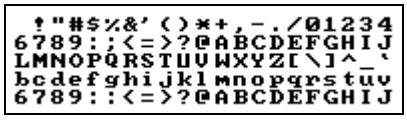


6x12 DOT ASCII

8x16 DOT ASCII



12 DOT ARIAL ASCII

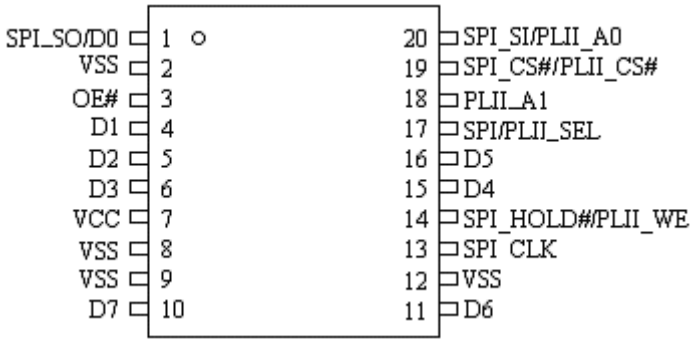


16 DOT ARIAL ASCII

GT23L24T3Y　　多字符集简繁　字库芯片　　标准字库

2 引脚描述与接口连接

2.1 引脚配置



SO20W

SO20W	名称	描述	
		GT23L(SPI 接口)	GT23L(PLII 接口)
1	SPI_SO/D0	Serial data output	Data Outputs
2	VSS	Ground	
3	OE#	No Connection	Output Enable Input
4	D1	No Connection	Data Outputs
5	D2	No Connection	Data Outputs
6	D3	No Connection	Data Outputs
7	VCC	Power Supply(3.3V)	
8	VSS	Ground	
9	VSS	Ground	
10	D7	No Connection	Data Outputs
11	D6	No Connection	Data Outputs
12	VSS	Ground	
13	SPI_CLK	Serial clock input	No Connection
14	SPI_HOLD#/PLII_WE	Hold(to pause the device)	Write Enable Input
15	D4	No Connection	Data Outputs
16	D5	No Connection	Data Outputs
17	SPI/PLII_SEL	SPI/PLII SELECT	
		NC: SPI	GND: PLII
18	PLII_A1	No Connection	Address Inputs
19	SPI_CS#/PLII_CS#	Chip enable input	Chip Enable Input
20	SPI_SI/PLII_A0	Serial data input	Address Inputs

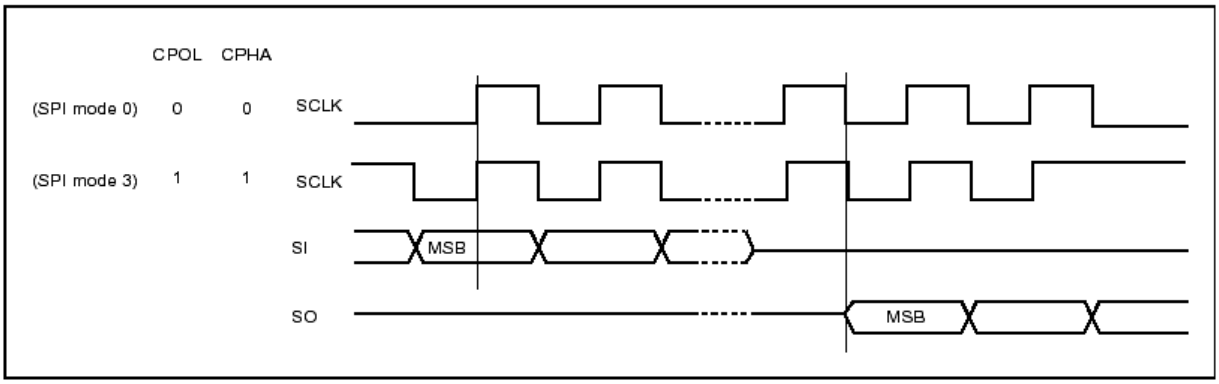
2.2 SPI 接口引脚描述

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。

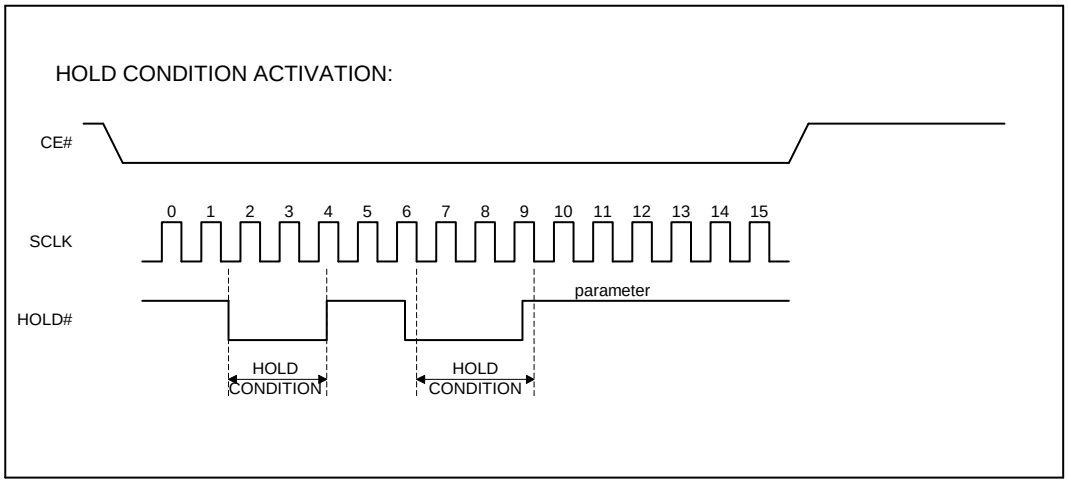


总线挂起输入（HOLD#）：

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。

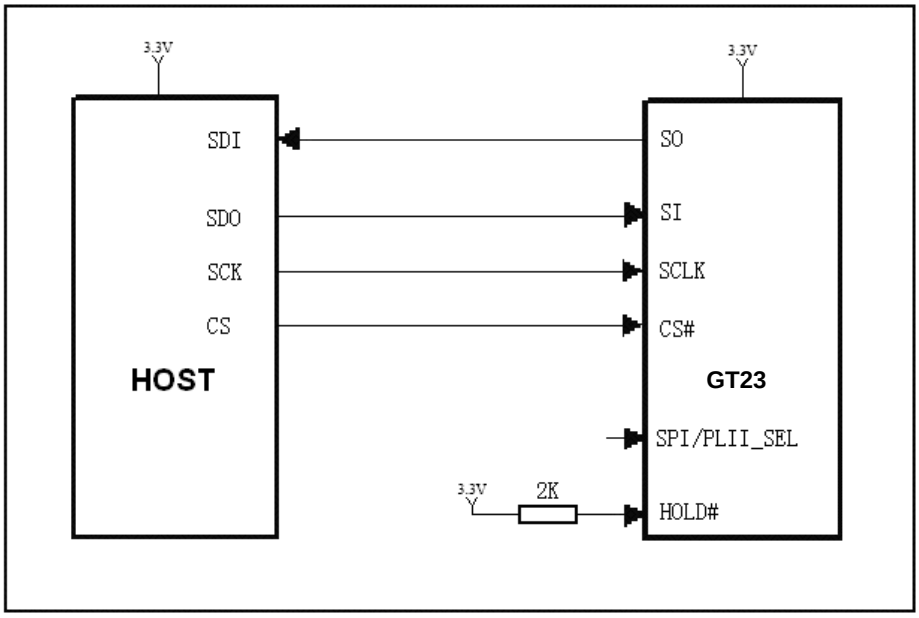
当HOLD#信号变为低并且串行时钟信号（SCLK）处于低电平时，进入总线挂起状态。

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。



2.3 SPI 接口与主机接口电路示意图

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）。



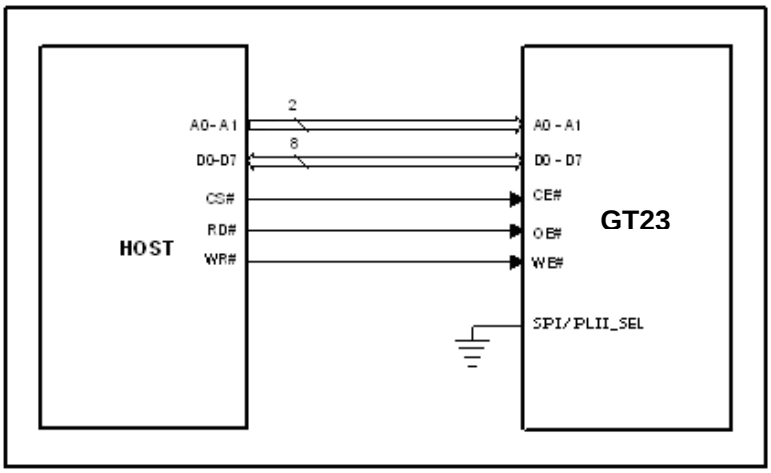
HOST CPU 主机 SPI 接口电路示意图

2.4 PLII 接口引脚描述

Pin name	I/O	描述
A[1..0]	I	地址寄存器寻址
D[7..0]	I/O	地址输入/数据输出
CE#	I	片选信号输入，低有效
OE#	I	“输出使能”信号输入，OE# 为低时输出使能
WE#	I	“写使能”信号输入，WE# 为低时写使能

2.5 PLII 接口与主机接口电路示意图

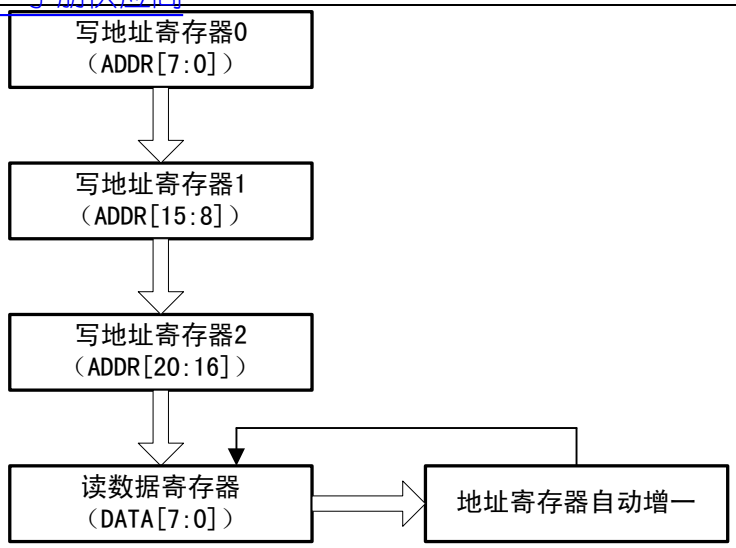
SPI/PLII_SEL（管脚内部有 100K 上拉电阻）接地，字库芯片选择 PLII 接口模式，与主机接口电路连接可以参考下图。



2.6 PLII 总线接口寻址说明

在 PLII 总线模式下，芯片内部有 3 个地址寄存器，主机需要把要读取数据的地址写入这 3 个地址寄存器，然后再从数据寄存器中读出数据。主机每读一次数据寄存器，芯片内部的地址寄存器会自动增一，从而使主机只写一次首地址，就可以连续读取数据。

A1 A0 （地址线）	读写操作	对应地址寄存器
0 0	写	地址寄存器 0 [ADDR7:0]
0 1	写	地址寄存器 1 [ADDR15:8]
1 0	写	地址寄存器 2 [ADDR20:16]
0 0	读	数据寄存器 [DATA7:0]



3 操作指令

3.1 SPI 接口模式下操作

3.1.1 指令参数

Instruction Set

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 ∞

所有对本芯片 SPI 接口的操作只有 2 个，那就是 Read Data Bytes (READ “一般读取”)和 Read Data Bytes at Higher Speed (FAST_READ “快速读取点阵数据”)。.

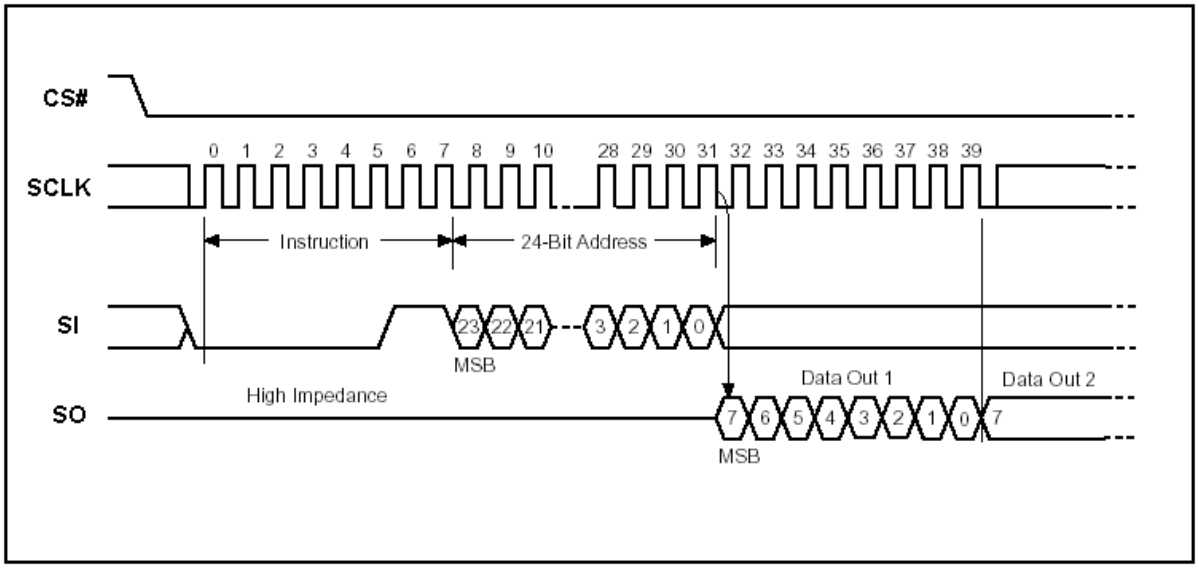
3.1.2 Read Data Bytes（一般读取）

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



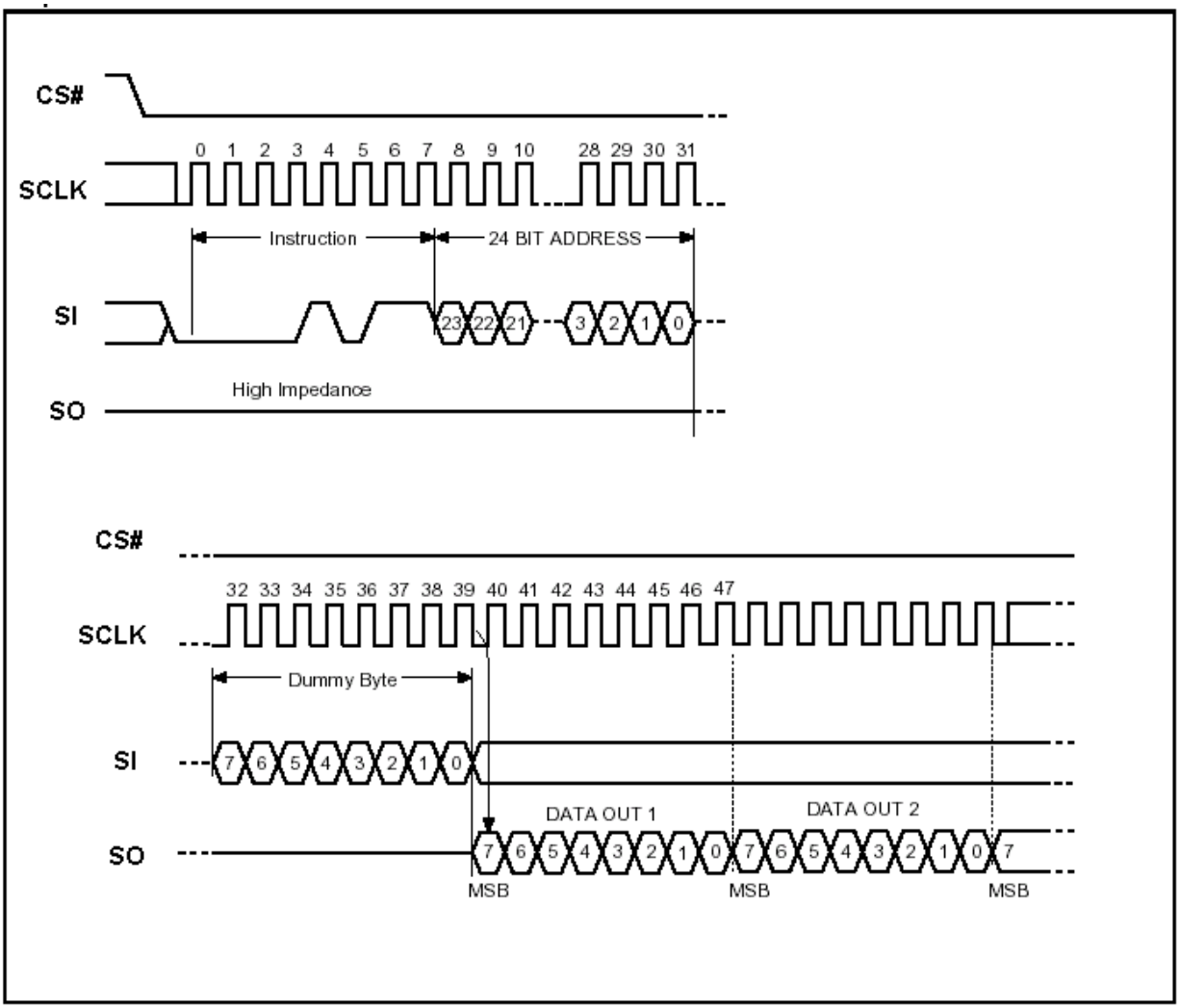
3.1.3 Read Data Bytes at Higher Speed（快速读取点阵数据）

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



3.2 PLII 接口模式下操作

在PLII模式下，字库芯片内部有3个地址保持寄存器，HOST 读字库芯片时，字库芯片把地址寄存器对应字库芯片地址内容送给HOST，并且HOST每读取一个字节后，字库芯片内部会把地址寄存器的值增1。当地址寄存器越过最大地址时自动归零。字库芯片可以对地址寄存器进行写操作。字库芯片上电时，内部硬件把地址寄存器清零。

3.2.1 信号描述

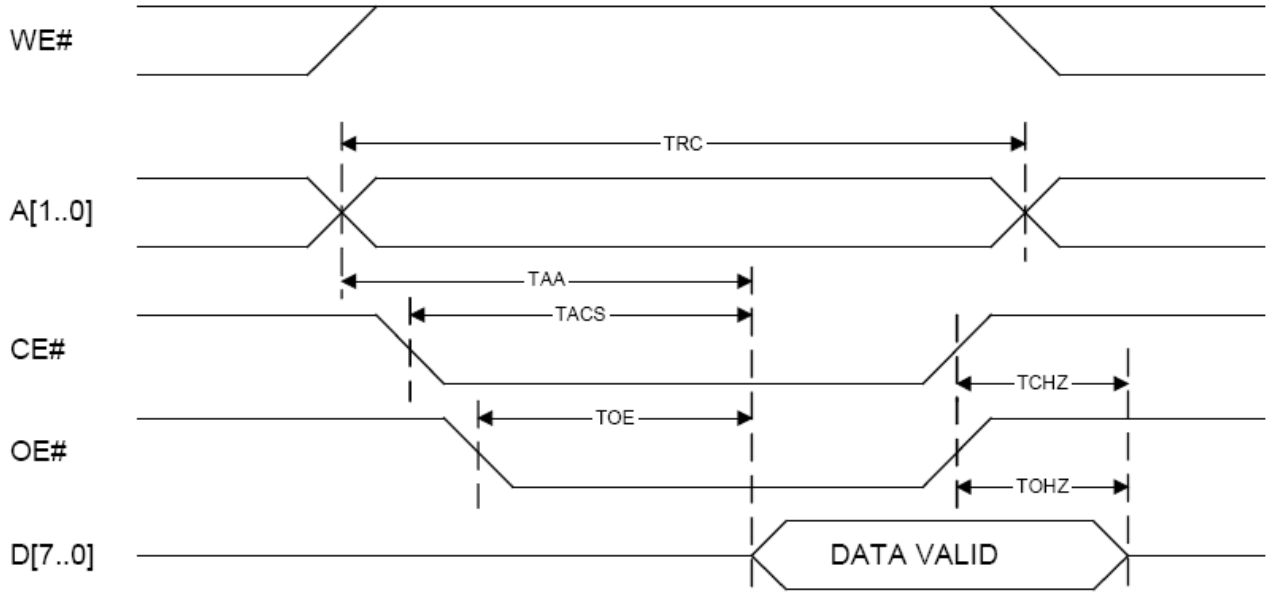
Pin name	I/O	描述
A[1..0]	I	地址寄存器寻址
D[7..0]	I/O	地址输入/数据输出
CE#	I	片选信号输入，低有效
OE#	I	“输出使能”信号输入，OE# 为低时输出使能
WE#	I	“写使能”信号输入，WE# 为低时写使能

真值表

Mode	CE#	OE#	WE#	D[7..0]
Other	H	X	X	High-Z
Read	L	L	H	Data Out
write	L	H	L	Addr In

3.2.2 读操作

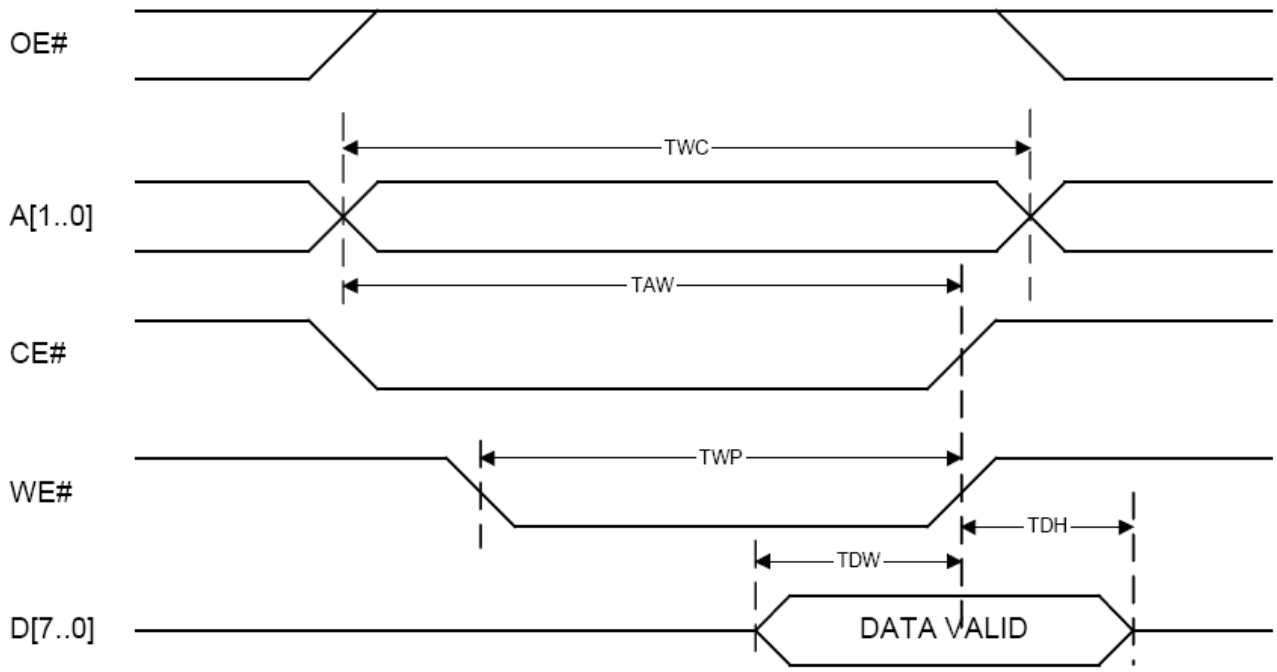
读字库芯片内部的点阵字库数据时，HOST 写字库芯片的 3 个地址寄存器，字库芯片把对应地址的数据送给 HOST。在 OE#和 CE#同时为低电平、WE#为高电平的情况下，HOST 可以从数据总线（D[7..0]）读出字库芯片的个字节的数据。当出现 OE# 或 CE# 中的一个信号变高，则字库芯片内部在此时刻把地址寄存器增 1，并保持地址寄存器的值。



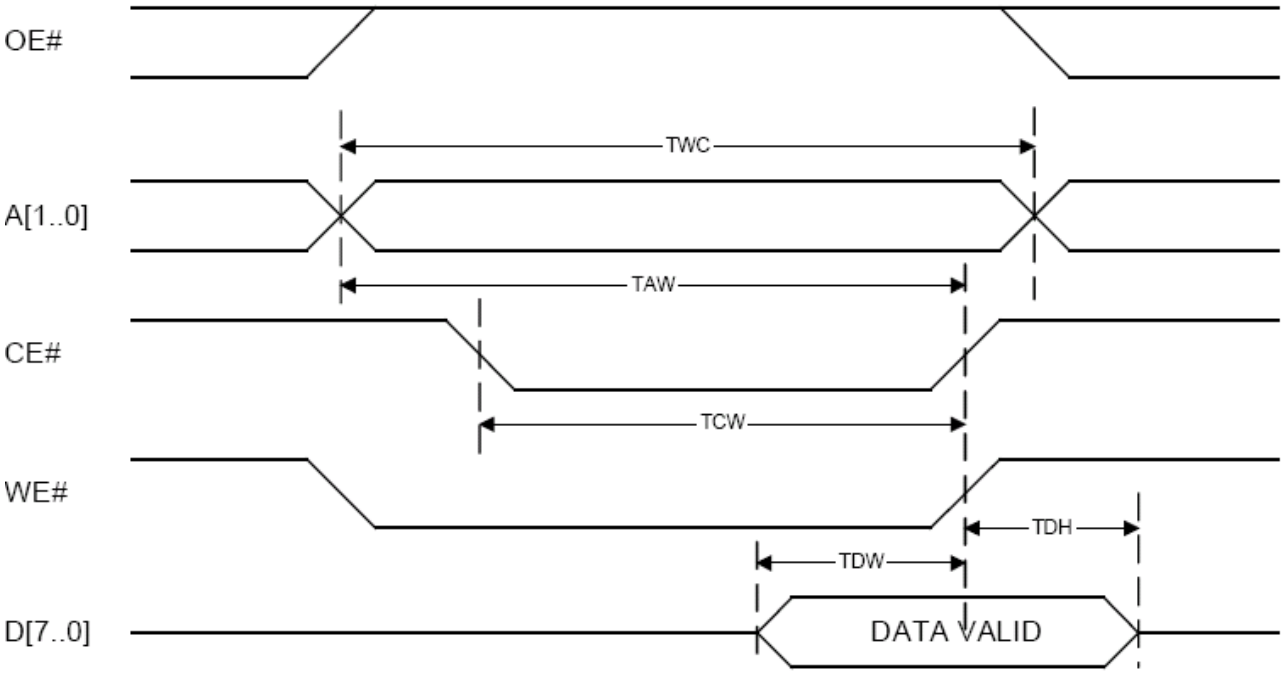
读周期时序波形图

3.2.3 写操作

字库芯片内部有 3 个地址保持寄存器，HOST 读字库芯片时需要把地址写到字库芯片中。在 WE#和 CE#同时为低电平、OE#为高电平的情况下，HOST 可以通过数据总线（D[7..0]）写 1 个字节数据到字库芯片。



写周期时序波形图（WE#控制的时序）



写周期时序波形图（CE#控制的时序）

4 电气特性

4.1 绝对最大额定值

Symbol	Parameter	Min.	Max.	Unit	Condition
T_{OP}	Operating Temperature	-20	85	℃	SPI mode
T_{OP}	Operating Temperature	-10	85	℃	PLII mode
T_{STG}	Storage Temperature	-65	125	℃	
VCC	Supply Voltage	-0.3	3.6	V	
V_{IN}	Input Voltage	-0.5	VCC+0.5	V	
GND	Power Ground	0	0	V	

4.2 DC 特性

Condition：T_{OP} = -20℃ to 85℃，GND=0V in SPI mode； T_{OP} =-10℃ to 85℃，GND=0V in PLII mode

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	0.6	V	VCC=2.7-3.6V
V_{IH}	Input HIGH Voltage	0.7VCC	VCC+0.3	V	
V_{OL}	Output LOW Voltage		0.4 (I_{OL}=1.6mA)	V	
V_{OH}	Output HIGH Voltage	0.8VCC (I_{OH}=-0.4mA)		V	
I_{LI}	Input Leakage Current	0	+10	uA	
I_{LO}	Output Leakage Current	0	+10	uA	

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

4.3 AC 特性

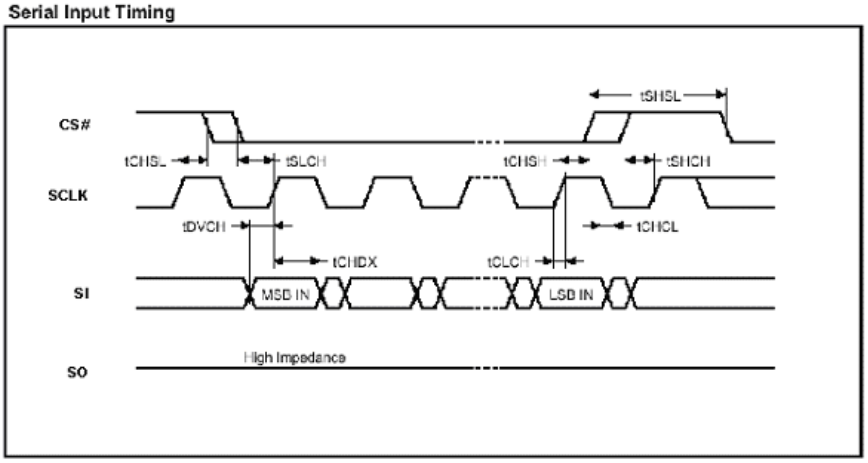
4.3.1 SPI 接口模式下 AC 特性

Condition：T_{OP} = -20℃ to 85℃，VCC= 2.7V to 3.6V

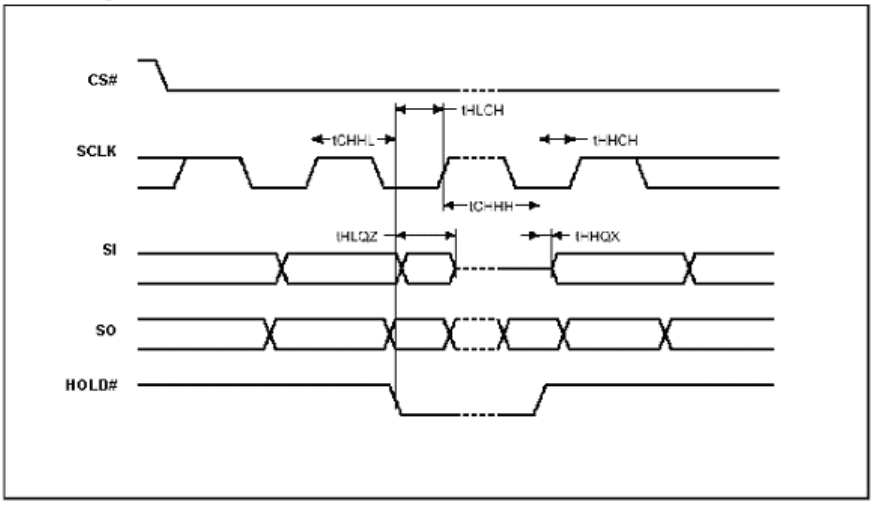
Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	20	MHz
t_{CH}	t_{CLH}	Clock High Time	20		ns
t_{CL}	t_{CLL}	Clock Low Time	20		ns
t_{CLCH}		Clock Rise Time(peak to peak)	0.1		V/ns
t_{CHCL}		Clock Fall Time (peak to peak)	0.1		V/ns
t_{SLCH}	t_{CSS}	CS# Active Setup Time (relative to SCLK)	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	tHO	Output Hold Time	0		ns
t HLCH		HOLD# Setup Time (relative to SCLK)	5		ns
t CHHH		HOLD# Hold Time (relative to SCLK)	5		ns
t HHCH		HOLD Setup Time (relative to SCLK)	5		ns
t CHHL		HOLD Hold Time (relative to SCLK)	5		ns
t HHQX	tLZ	HOLD to Output Low-Z		9	ns
t HLQZ	tHZ	HOLD# to Output High-Z		9	ns

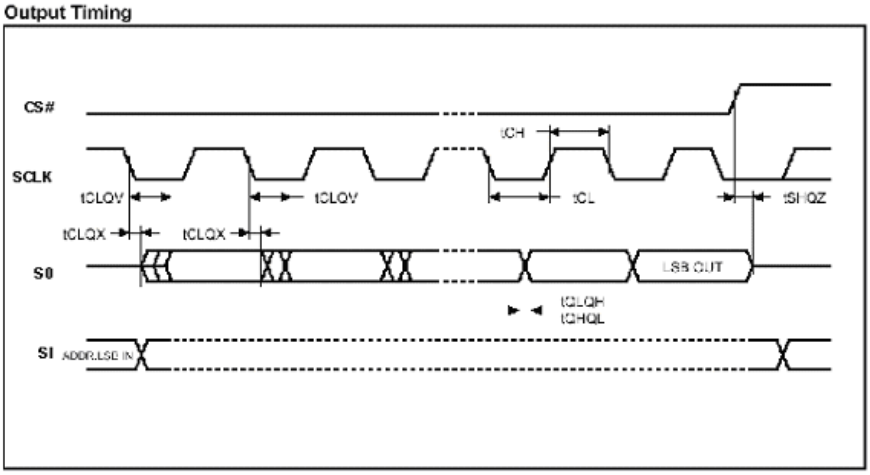
Serial Input Timing



Hold Timing



Output Timing

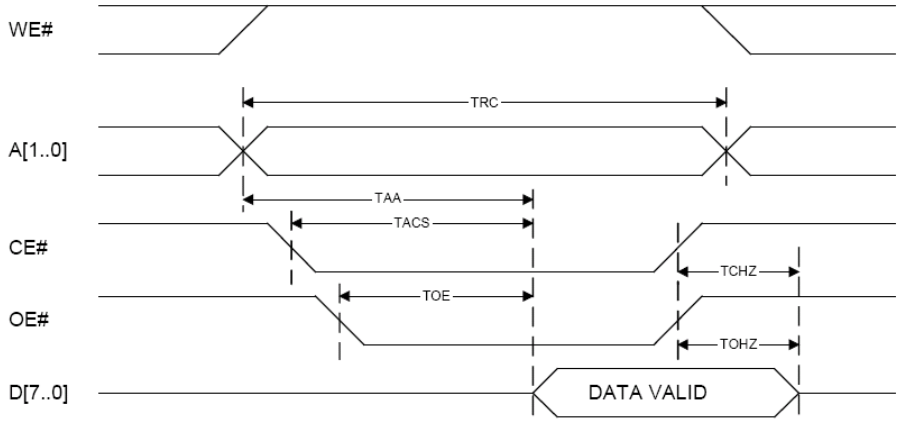


4.3.2 PLII 接口模式下 AC 特性

4.3.2.1 读周期时间特性

Condition：T_{OP} = -10℃ to 85℃，VCC= 2.7V to 3.6V

Symbol	Parameter	Min.	Max.	Unit
TRC	Read Cycle Time	130	-	ns
TAA	Address Access Time	-	110	ns
TACS	Chip Select Access Time	-	110	ns
TOE	Output Enable to Output Valid	-	100	ns
TCHZ	Chip Deselect to Output in High-Z	-	10	ns
TOHZ	Output Disable to Output in High-Z	-	10	ns

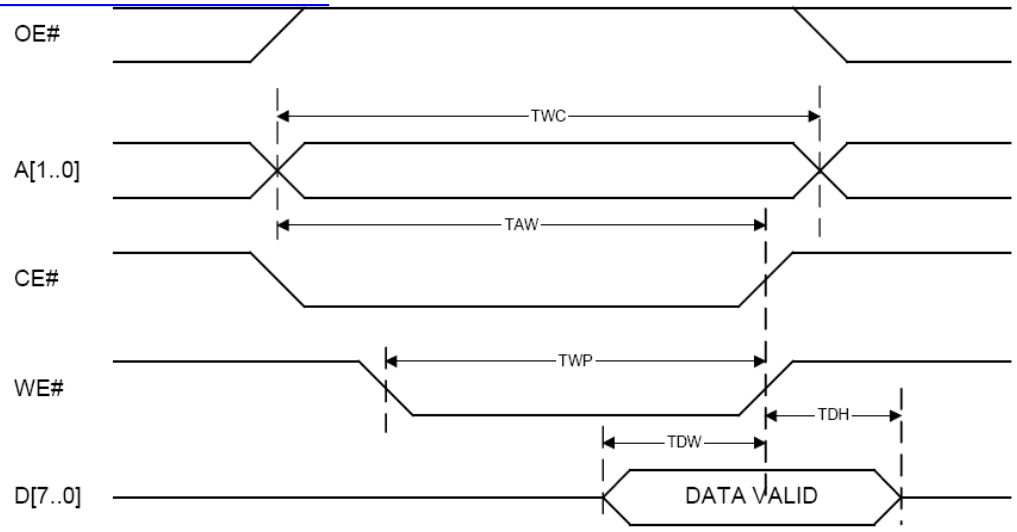


读周期时序波形图

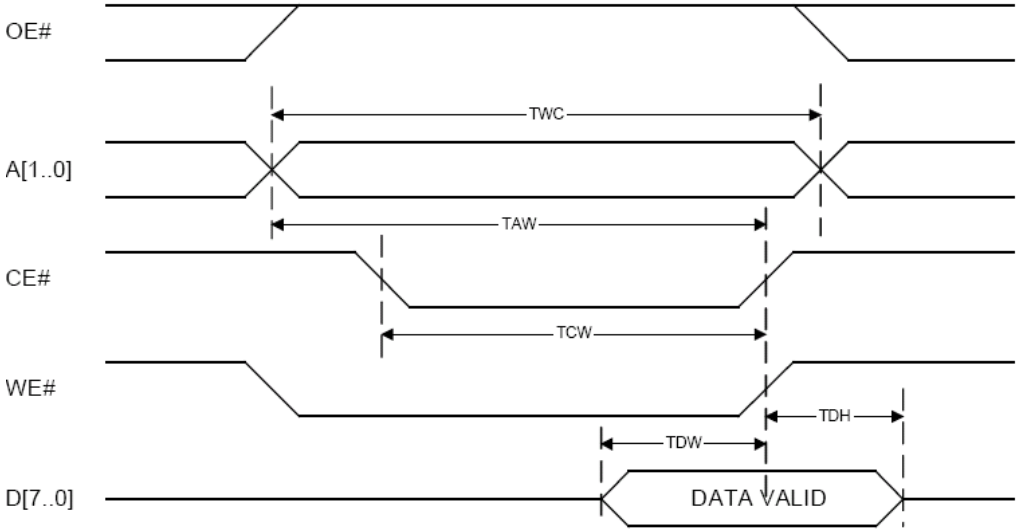
4.3.2.2 写周期时间特性

Condition：T_{OP} = -10℃ to 85℃，VCC= 2.7V to 3.6V

Symbol	Parameter	Min.	Max.	Unit
TWC	Write Cycle Time	130		ns
TAW	Address Valid to End-of-Write	120		ns
TCW	Chip Select to End-of-Write	100		ns
TWP	Write Pulse Width	100		ns
TDW	Dada to Write Time Overlap	30		ns
TDH	Data Hold from Write Time	5		ns



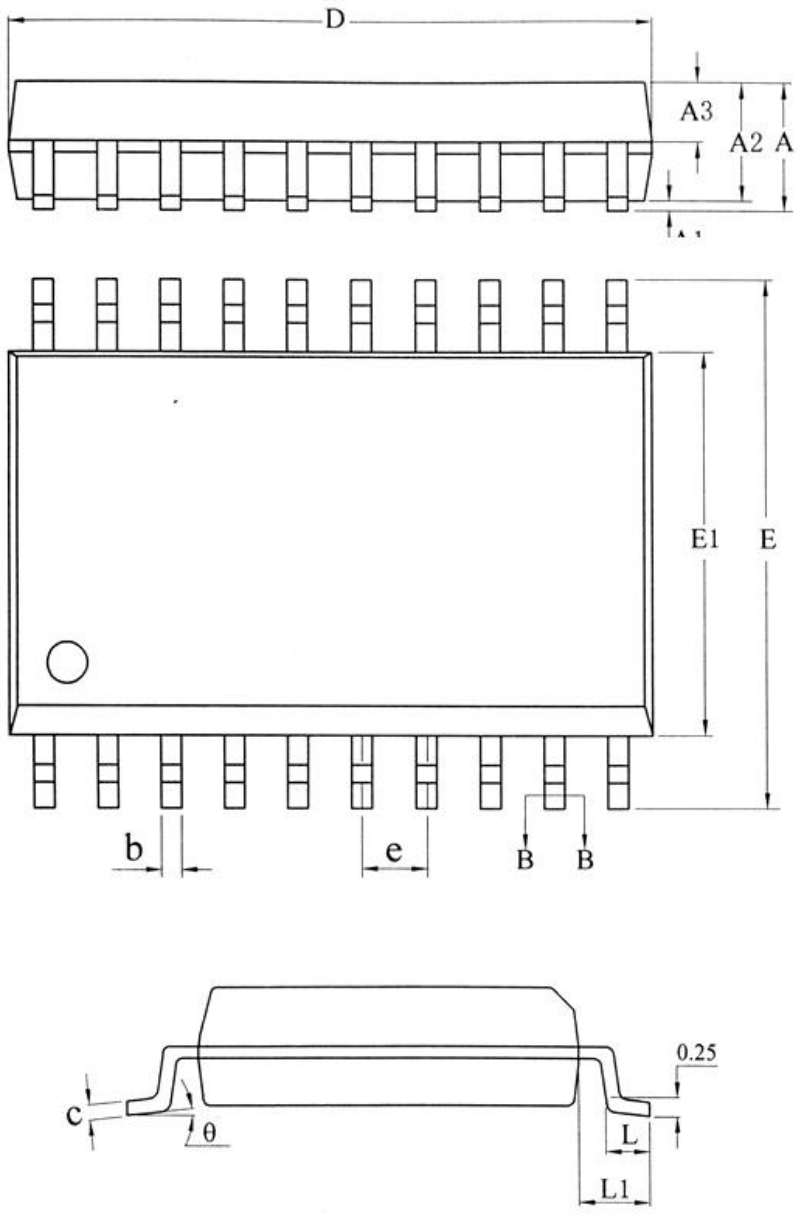
写周期时序波形图（WE#控制的时序）



写周期时序波形图（CE#控制的时序）

5 封装尺寸：SO20W

单位: mm



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.70
A1	0.10	0.20	0.30
A2	2.10	2.30	2.50
A3	0.92	1.02	1.12
b	0.35	—	0.44
b1	0.34	0.37	0.39
c	0.26	—	0.31
c1	0.24	0.25	0.26
D	12.60	12.80	13.00
E	10.10	10.30	10.50
E1	7.30	7.50	7.70
e	1.27BSC		
L	0.70	0.85	1.00
L1	1.40BSC		
θ	0	—	8°

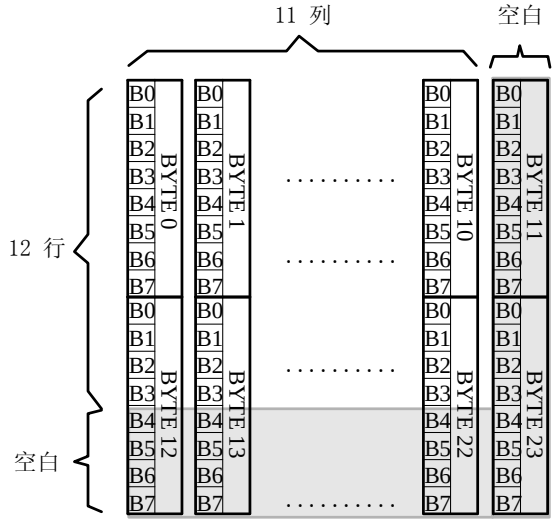
6 字库调用方法

6.1 汉字点阵排列格式

每个汉字在芯片中是以汉字点阵字模的形式存储的，每个点用一个二进制位表示，存 1 的点，当显示时可以在屏幕上显示亮点，存 0 的点，则在屏幕上不显示。点阵排列格式为竖置横排：即一个字节的高位表示下面的点，低位表示上面的点（如果用户按 16bit 总线宽度读取点阵数据，请注意高低字节的顺序），排满一行后再排下一行。这样把点阵信息用来直接在显示器上按上述规则显示，则将出现对应的汉字。

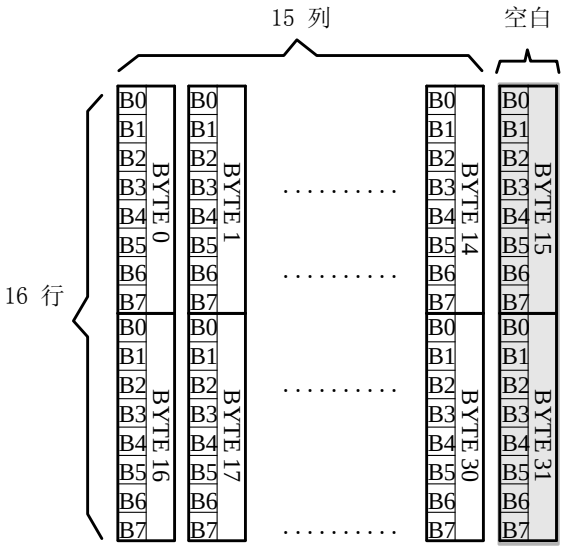
6.1.1 11X12 点汉字排列格式

11X12 点汉字的信息需要 24 个字节（BYTE 0 – BYTE 23）来表示。该 11X12 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：



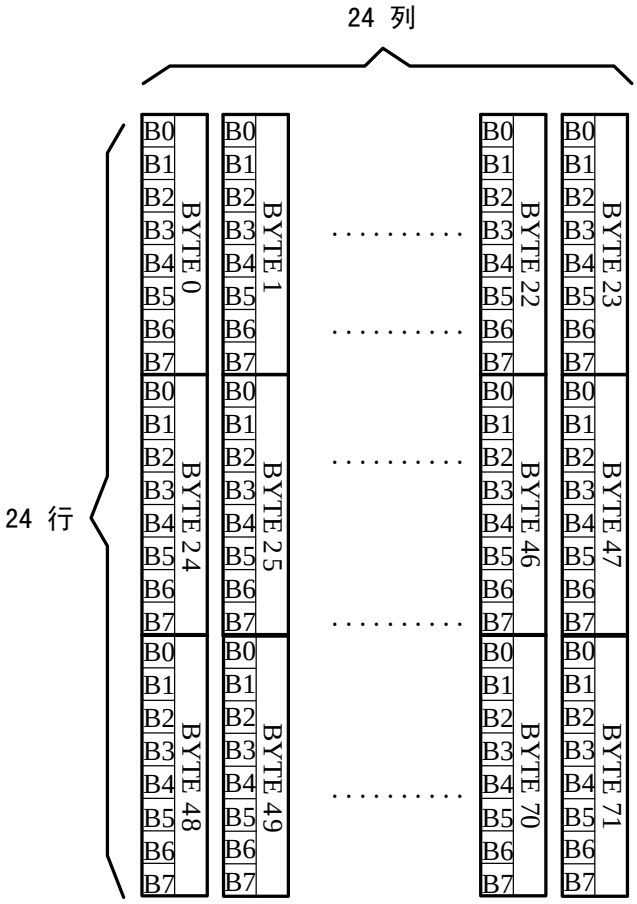
6.1.2 15X16 点汉字排列格式

15X16 点汉字的信息需要 32 个字节（BYTE 0 – BYTE 31）来表示。该 15X16 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：



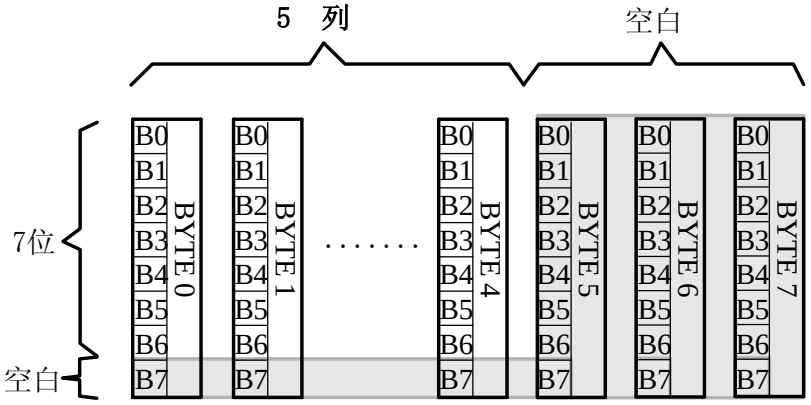
6.1.3 24X24 点汉字排列格式

24X24 点汉字的信息需要 72 个字节（BYTE 0 – BYTE 71）来表示。该 24X24 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：



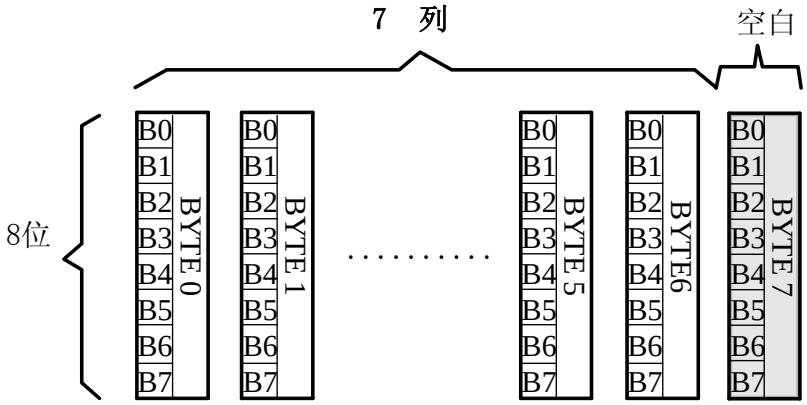
6.1.4 5X7 点 ASCII 字符排列格式

5X7 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



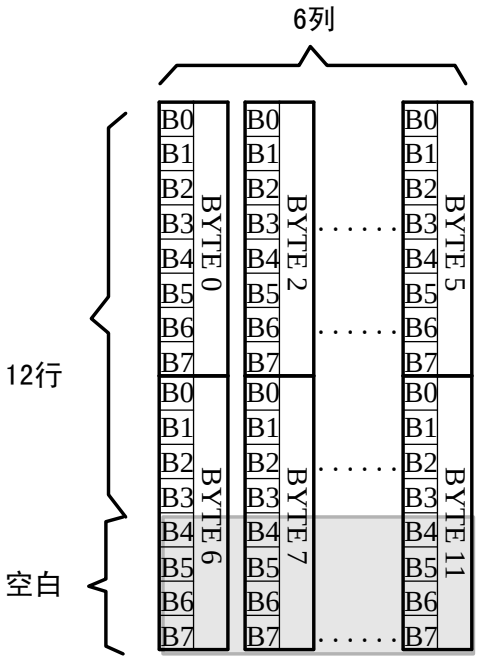
6.1.5 7X8 点 ASCII 字符排列格式

7X8 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



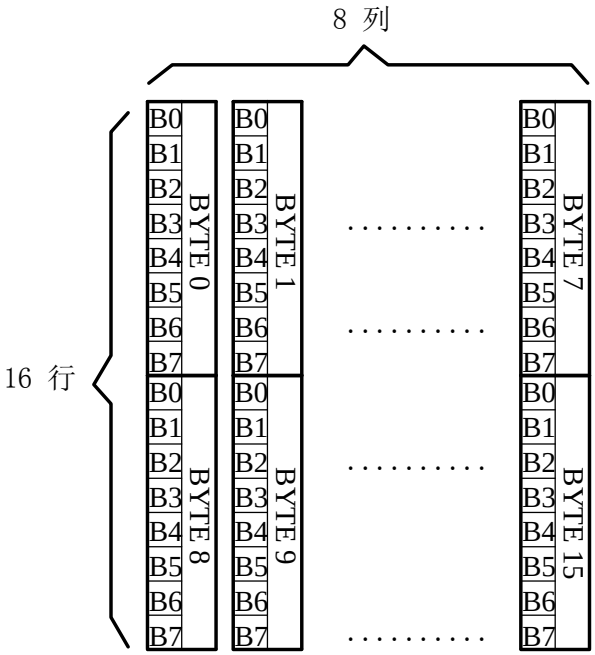
6.1.6 6X12 点 ASCII 字符排列格式

6X12 点字符的信息需要 12 个字节（BYTE 0 – BYTE11）来表示。该点阵数据是竖置横排的，其具体排列结构如下图：



6.1.7 8X16 点 ASCII 字符排列格式

8X16 点字符的信息需要 16 个字节（BYTE 0 – BYTE15）来表示。该点阵数据是竖置横排的，其具体排列结构如下图：

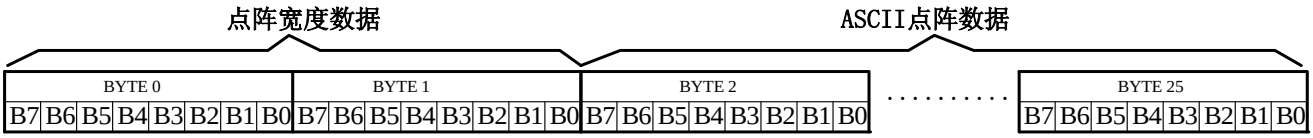


6.1.8 12 点阵不等宽 ASCII 方头（Arial）字符排列格式

12 点阵不等宽字符的信息需要 26 个字节（BYTE 0 – BYTE25）来表示。

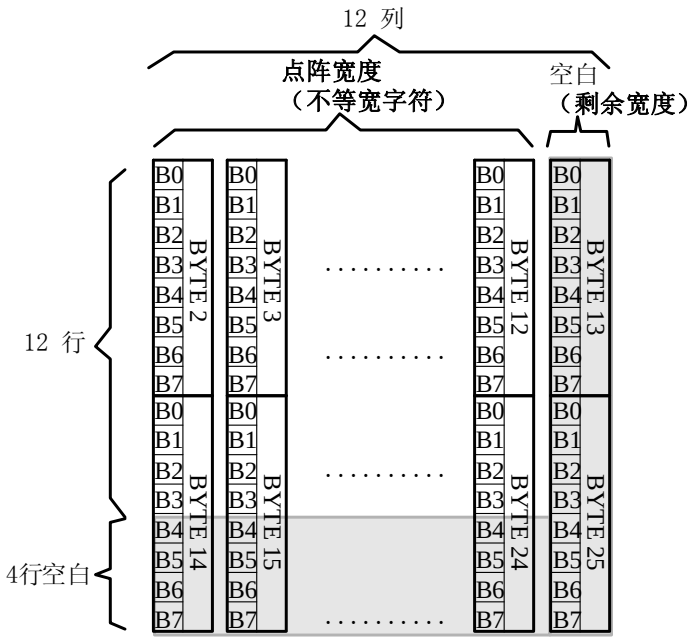
■ 存储格式

由于字符是不等宽的，因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-25 存放竖置横排点阵数据。具体格式见下图：



■ 存储结构

点阵存储宽度固定为 12，根据不同字符，其实际点阵宽度会小于 12，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



例如：ASCII 方头字符 B

0-25BYTE 的点阵数据是: 00 08 F8 F8 88 88 88 F8 70 00 00 00 00 00 0F 0F 08 08 08 0F 07 00 00 00 00 00

其中：

BYTE0~ BYTE1: 00 08

为 ASCII 方头字符 B 的点阵宽度数据 00 08，即：8 位宽度。字符后面有 8 位空白区，可以在排版下一个字时考虑到这一点，将下一个字的起始位置前移。

BYTE2-25: 00 08 F8 F8 88 88 88 F8 70 00 00 00 00 00 0F 0F 08 08 08 0F 07 00 00 00 00 00

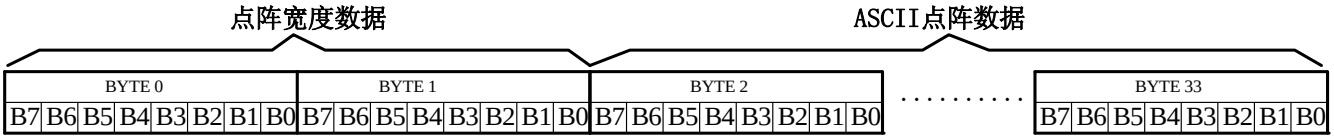
为 ASCII 方头字符 B 的点阵数据。

6.1.9 16 点阵不等宽 ASCII 方头（Arial）字符排列格式

16 点阵不等宽字符的信息需要 34 个字节（BYTE 0 – BYTE33）来表示。

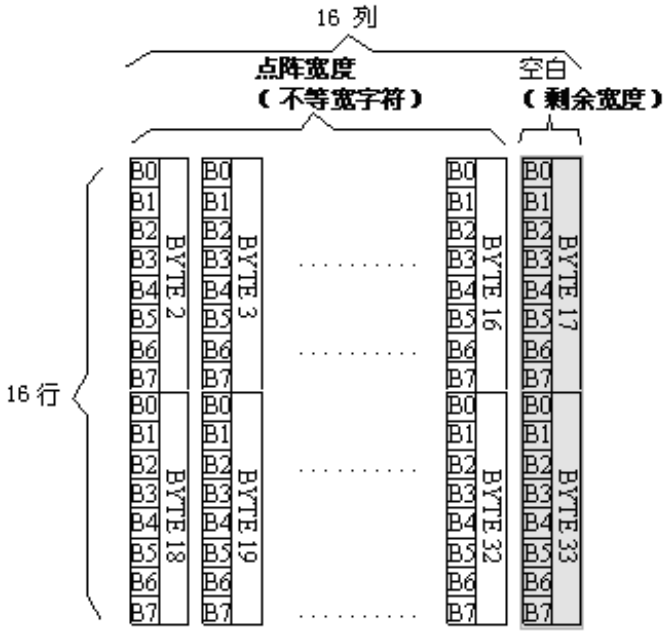
■ 存储格式

由于字符是不等宽的，因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-33 存放竖置横排点阵数据。具体格式见下图：



■ 存储结构

点阵存储宽度固定为 16，根据不同字符，其实际点阵宽度会小于 16，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



例如：ASCII 方头字符 B

0-33BYTE 的点阵数据是: 00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00

其中：

BYTE0~ BYTE1: 00 0C 为 ASCII 方头字符 B 的点阵宽度数据，即：12 位宽度。字符后面有 4 位空白区，可以在排版下一个字时考虑到这一点，将下一个字的起始位置前移。（见下图）

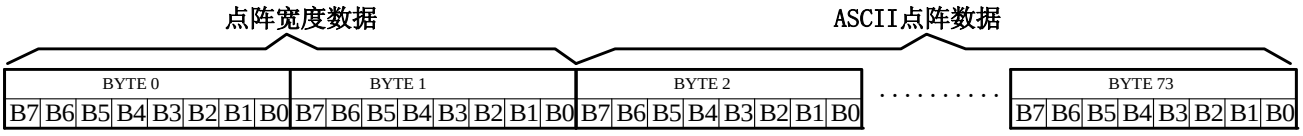
BYTE2-33: 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00 为 ASCII 方头字符 B 的点阵数据。

6.1.10 24 点阵不等宽 ASCII 方头（Arial）字符排列格式

24 点阵不等宽字符的信息需要 74 个字节（BYTE 0 – BYTE73）来表示。

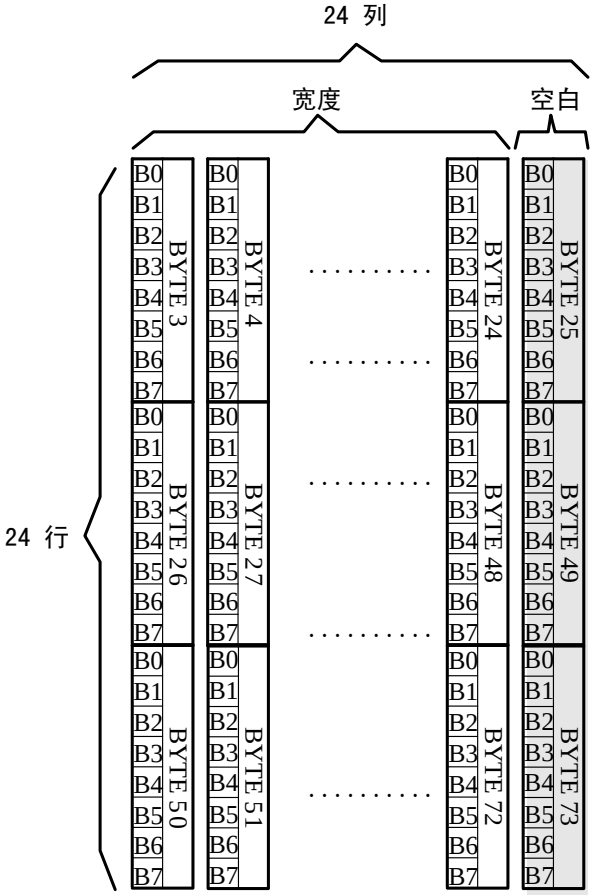
■ 存储格式

由于字符是不等宽的，因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-73 存放竖置横排点阵数据。具体格式见下图：



■ 存储结构

点阵存储宽度固定为 24，根据不同字符，其实际点阵宽度会小于 24，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



6.2 点阵字库地址表

	字库内容	编码体系	码位范围	字符数	起始地址	参考算法
1	11X12 点 GB2312 点阵字库	GB2312	A1A1-F7FE	6763+470	00000	6.3.1.1
2	15X16 点 GB2312 点阵字库	GB2312	A1A1-F7FE	6763+470	5F4C0	6.3.1.2
3	24X24 点 GB2312 点阵字库	GB2312	A1A1-F7FE	6763+470	DE5C0	6.3.1.3
4	GB12345 索引表	GB12345	A1A1-F9A9	6866+662	190958	6.3.1.4-6
5	BIG5 索引表	BIG5	A140-F9DC	5401+408	193F06	6.3.1.7-9
6	Unicode 索引表	Unicode	A0-FF50		19AC30	6.3.1.10-12
7	GB12345 一对多索引表	GB12345		94	1A7020	6.3.1.13
8	5X7 点 ASCII 字符	ASCII	20~7F	96	1A7C33	6.3.2.1
9	7X8 点 ASCII 字符	ASCII	20~7F	96	1A76CC	6.3.2.2
10	6X12 点 ASCII 字符	ASCII	20~7F	96	1A79CC	6.3.2.3
11	8X16 点 ASCII 字符	ASCII	20~7F	96	1A7FCC	6.3.2.4
12	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F	96	1A87CC	6.3.2.6
13	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F	96	1A918C	6.3.2.7
14	24 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F	96	1A9E4C	6.3.2.7
15	输入法码表				1ABA0C	
16	保留区				1FF6A4	

6.3 字符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。

6.3.1 汉字字符地址计算

6.3.1.1 11X12 点 GB2312 标准点阵字库

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=0x0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*24+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*24+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)//12 点阵汉字区
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*24+ BaseAdd;
```

6.3.1.2 15X16 点 GB2312 标准点阵字库

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=0x5F4C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*32+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)//16 点阵汉字区
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*32+ BaseAdd;
```

6.3.1.3 24X24 点 GB2312 标准点阵字库

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=0XDE5C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*72+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*72+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)//24 点阵汉字区
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*72+ BaseAdd;
```

6.3.1.4 11X12 点 GB12345 字符集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

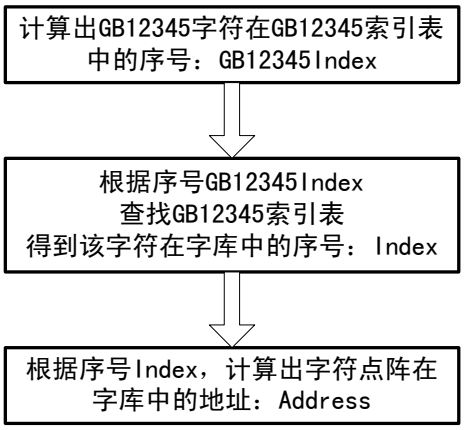
BaseAdd：说明点阵数据在字库芯片中的起始地址。

GB12345Table：表示 GB12345 索引表。表的起始地址为 0x190958

GB12345Index：表示该内码汉字在 GB12345 索引表中的序号。由该序号可查 GB12345 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x00;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*24+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*24+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0XF9 && LSB >=0xA1)
{
    GB12345Index = (MSB - 0xB0) * 94 + (LSB - 0xA1);
    Index = GB12345Table [GB12345Index*2] * 256 + GB12345Table[GB12345Index*2+1];
    Address =Index * 24 + BaseAdd;
}
```

6.3.1.5 15X16 点 GB12345 字符集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

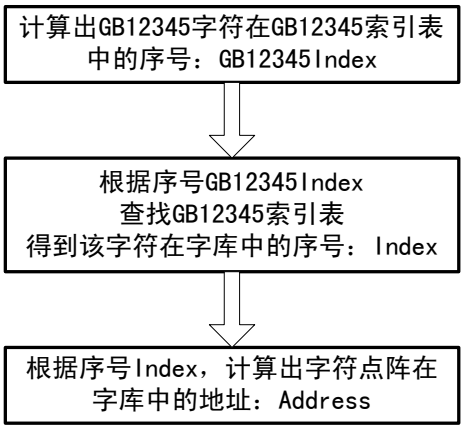
BaseAdd：说明点阵数据在字库芯片中的起始地址。

GB12345Table：表示 GB12345 索引表。表的起始地址为 0x190958

GB12345Index：表示该内码汉字在 GB12345 索引表中的序号。由该序号可查 GB12345 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x5F4C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
      Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
      Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*32+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
            Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0XF9 && LSB >=0xA1)
{
      GB12345Index = (MSB - 0xB0) * 94 + (LSB - 0xA1);
      Index = GB12345Table[GB12345Index*2] * 256 + GB12345Table[GB12345Index*2+1];
      Address =Index * 32 + BaseAdd;
}
```

6.3.1.6 24X24 点 GB12345 基本集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

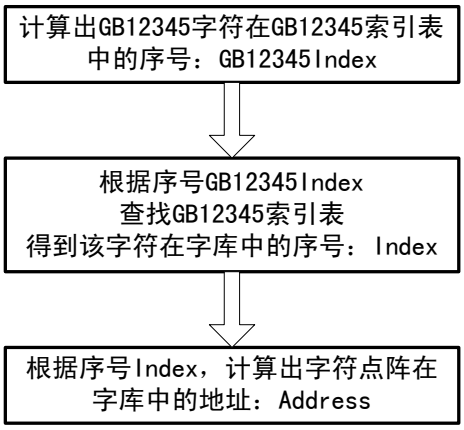
BaseAdd：说明点阵数据在字库芯片中的起始地址。

GB12345Table：表示 GB12345 索引表。表的起始地址为 0x190958

GB12345Index：表示该内码汉字在 GB12345 索引表中的序号。由该序号可查 GB12345 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0XDE5C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)//全角字符 1 区 1 部分
      Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)//全角字符 1 区 2 部分
      Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*72+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)//全角字符 1 区 3 部分
            Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*72+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0XF9 && LSB >=0xA1)
{
      GB12345Index = (MSB - 0xB0) * 94 + (LSB - 0xA1);
      Index = GB12345Table[GB12345Index*2] * 256 + GB12345Table[GB12345Index*2+1];
      Address =Index * 72 + BaseAdd;
}
```

6.3.1.7 11X12 点 BIG5 字符集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

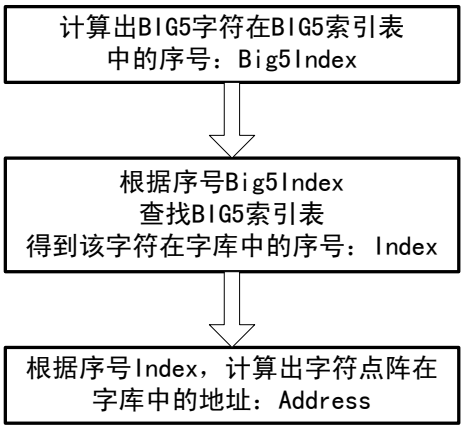
BaseAdd：说明点阵数据在字库芯片中的起始地址。

Big5Table：表示 BIG5 索引表。表的起始地址为 0x193F06

Big5Index：表示该内码汉字在 BIG5 索引表中的序号。由该序号可查 BIG5 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x00;
if(MSB >=0xA1 && MSB <= 0XF9)
{
    if(LSB >=0x40 && LSB <= 0X7E)
        Big5Index =(MSB - 0xA1) * 157 + (LSB - 0x40);
    else if(LSB >=0XA1 && LSB <= 0XFE)
        Big5Index =(MSB - 0xA1) * 157 + 63 + (LSB - 0XA1));
}
Index = Big5Table[Big5Index*2] * 256 + Big5Table[Big5Index*2+1];
Address =Index * 24 + BaseAdd;
```

6.3.1.8 15X16 点 BIG5 字符集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

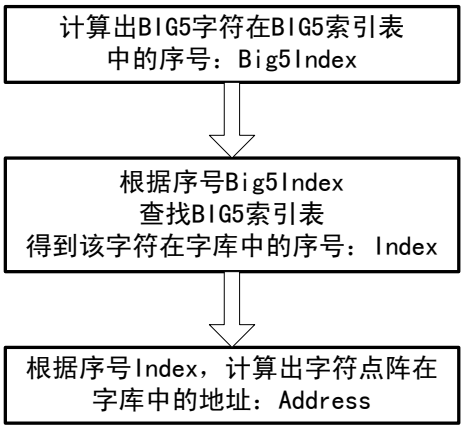
BaseAdd：说明点阵数据在字库芯片中的起始地址。

Big5Table：表示 BIG5 索引表。表的起始地址为 0x193F06

Big5Index：表示该内码汉字在 BIG5 索引表中的序号。由该序号可查 BIG5 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x5F4C0;
```

```
if(MSB >=0xA1 && MSB <= 0XF9)
{
    if(LSB >=0x40 && LSB <= 0X7E)
        Big5Index =(MSB - 0xA1) * 157 + (LSB - 0x40);
    else if(LSB >=0XA1 && LSB <= 0XFE)
        Big5Index =(MSB - 0xA1) * 157 + 63 + (LSB - 0xA1));
}
Index = Big5Table[Big5Index*2] * 256 + Big5Table[Big5Index*2+1];
Address =Index * 32 + BaseAdd;
```

6.3.1.9 24X24 点 BIG5 基本集汉字字符

参数说明：

FontCode表示汉字内码。

MSB 表示汉字内码FontCode的高8bits。

LSB 表示汉字内码FontCode的低8bits。

Address 表示汉字点阵在芯片中的字节地址。

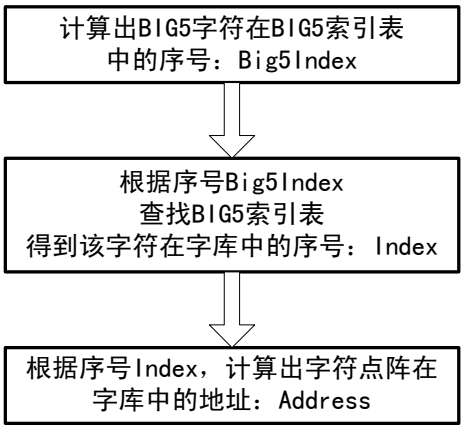
BaseAdd：说明点阵数据在字库芯片中的起始地址。

Big5Table：表示 BIG5 索引表。表的起始地址为 0x193F06

Big5Index：表示该内码汉字在 BIG5 索引表中的序号。由该序号可查 BIG5 索引表得出汉字在字库中的序号

Index：表示该字符在字库中的序号。由于 24 点字库只支持到 BIG5 基本集字符，总共有 10139 个字符。所以当序号 Index 小于 10139 时才有效。

计算方法：




```
BaseAdd=0XDE5C0;
if(MSB >=0xA1 && MSB <= 0XC6)
{
    if(LSB >=0x40 && LSB <= 0X7E)
        Big5Index =(MSB - 0xA1) * 157 + (LSB - 0x40);
    else if(LSB >=0XA1 && LSB <= 0XFE)
        Big5Index =(MSB - 0xA1) * 157 + 63 + (LSB - 0xA1));
}
Index = Big5Table[Big5Index*2] * 256 + Big5Table[Big5Index*2+1];
If(Index<10139)
    Address =Index * 72 + BaseAdd;
```

6.3.1.10 11X12 点 Unicode 中文字符

参数说明：

FontCode表示中文内码。

Address 表示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。

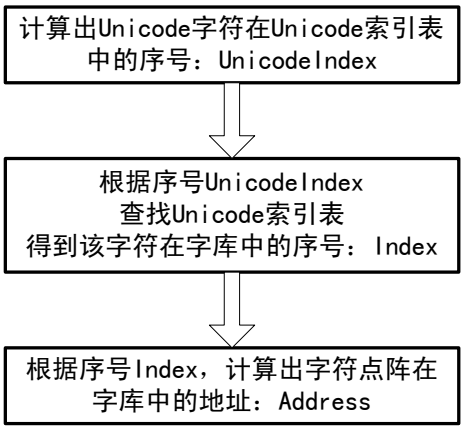
UnicodeTable：表示 Unicode 索引表。表的起始地址为 0x19AC30

UnicodeToIndex()：根据 Unicode 内码计算出该字符在 Unicode 索引表中的序号。

UnicodeIndex：表示该内码字符在 Unicode 索引表中的序号。由该序号可查 Unicode 索引表得出字符在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x00;
UnicodeIndex = UnicodeToIndex(FontCode);
Index = UnicodeTable [UnicodeIndex *2] * 256 + UnicodeTable [UnicodeIndex *2+1] ;
Address =Index * 24 + BaseAdd;

WORD    UnicodeToIndex(WORD code)
{
      BYTE result=0;
      WORD h=0;

      if(code<=0x20) code = 0x3000;
      else if(code<0x7f)
      {
            code=0xfe57-code-0x21;
      }

      if(code<= 0xa0)    result=1;
      else if(code<=0x0451) h=code-160;
      else if(code< 0x2010) result=1;
      else if(code<=0x2642) h=code-160-7102;
      else if(code< 0x3000) result=1;
      else if(code<=0x33d5) h=code-160-7102-2493;
      else if(code< 0x4e00) result=1;
```

```
    else if(code<=0x9fa5) h=code-160-7102-2493-6698;
    else if(code< 0xe76c) result=1;
    else if(code<=0xe864) h=code-160-7102-2493-6698-18374;
    else if(code< 0xf92c) result=1;
    else if(code<=0xfa29) h=code-160-7102-2493-6698-18374-4295;
    else if(code< 0xfe30) result=1;
    else if(code<=0xfe6b) h=code-160-7102-2493-6698-18374-4295-1030;
    else if(code< 0xff01) result=1;
    else if(code<=0xff5e) h=code-160-7102-2493-6698-18374-4295-1030-149;
    else if(code< 0xffe0) result=1;
    else if(code<=0xffe5) h=code-160-7102-2493-6698-18374-4295-1030-149-129;
    else result=1;

    if(result==1)
    {
        h = 0x3000-160-7102-2493;
    }

    return h;
}
```

6.3.1.11 15X16 点 Unicode 中文字符

参数说明：

FontCode表示中文内码。

Address 表示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。

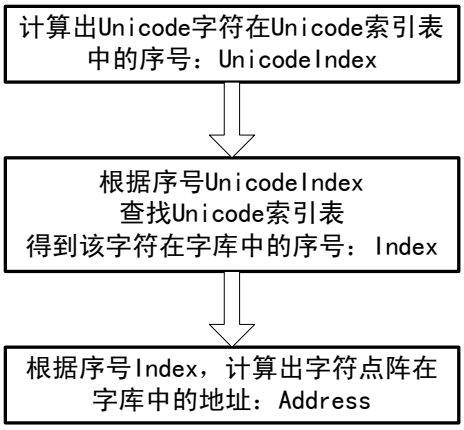
UnicodeTable：表示 Unicode 索引表。表的起始地址为 0x19AC30

UnicodeToIndex()：根据 Unicode 内码计算出该字符在 Unicode 索引表中的序号。

UnicodeIndex：表示该内码字符在 Unicode 索引表中的序号。由该序号可查 Unicode 索引表得出字符在字库中的序号

Index：表示该字符在字库中的序号。

计算方法：




```
BaseAdd=0x5F4C0;
UnicodeIndex = UnicodeToIndex(FontCode);
Index = UnicodeTable [UnicodeIndex *2] * 256 + UnicodeTable [UnicodeIndex *2+1] ;
```

```
Address =Index * 32 + BaseAdd;

WORD    UnicodeToIndex(WORD code)
{
    BYTE result=0;
    WORD h=0;

    if(code<=0x20) code = 0x3000;
    else if(code<0x7f)
    {
        code=0xfe57-code-0x21;
    }

    if(code<= 0xa0)    result=1;
    else if(code<=0x0451) h=code-160;
    else if(code< 0x2010) result=1;
    else if(code<=0x2642) h=code-160-7102;
    else if(code< 0x3000) result=1;
    else if(code<=0x33d5) h=code-160-7102-2493;
    else if(code< 0x4e00) result=1;
    else if(code<=0x9fa5) h=code-160-7102-2493-6698;
    else if(code< 0xe76c) result=1;
    else if(code<=0xe864) h=code-160-7102-2493-6698-18374;
    else if(code< 0xf92c) result=1;
    else if(code<=0xfa29) h=code-160-7102-2493-6698-18374-4295;
    else if(code< 0xfe30) result=1;
    else if(code<=0xfe6b) h=code-160-7102-2493-6698-18374-4295-1030;
    else if(code< 0xff01) result=1;
    else if(code<=0xff5e) h=code-160-7102-2493-6698-18374-4295-1030-149;
    else if(code< 0xffe0) result=1;
    else if(code<=0xffe5) h=code-160-7102-2493-6698-18374-4295-1030-149-129;
    else result=1;

    if(result==1)
    {
        h = 0x3000-160-7102-2493;
    }

    return h;
}
```

6.3.1.12 24X24 点 Unicode 中文字符

参数说明：

FontCode表示中文内码。

Address 表示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。

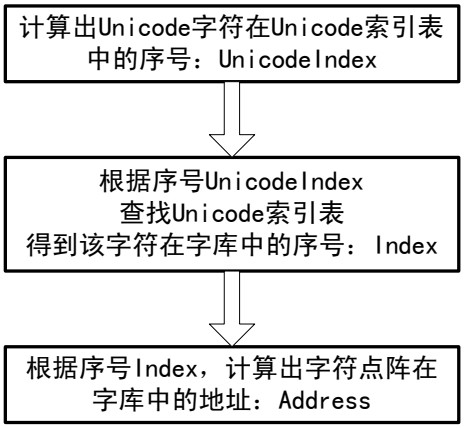
UnicodeTable：表示 Unicode 索引表。表的起始地址为 0x19AC30

UnicodeToIndex()：根据 Unicode 内码计算出该字符在 Unicode 索引表中的序号。

UnicodeIndex：表示该内码字符在 Unicode 索引表中的序号。由该序号可查 Unicode 索引表得出字符在字库中的序号

Index：表示该字符在字库中的序号。由于 24 点字库只支持到 BIG5 基本集字符，总共有 10139 个字符。所以当序号 Index 小于 10139 时才有效。

计算方法：




```
BaseAdd=0XDE5C0;
UnicodeIndex = UnicodeToIndex(FontCode);
Index = UnicodeTable [UnicodeIndex *2] * 256 + UnicodeTable [UnicodeIndex *2+1] ;
If(Index<10139)
    Address =Index * 72 + BaseAdd;

WORD   UnicodeToIndex(WORD code)
{
    BYTE result=0;
    WORD h=0;

    if(code<=0x20) code = 0x3000;
    else if(code<0x7f)
    {
        code=0xfe57-code-0x21;
    }

    if(code<= 0xa0)   result=1;
    else if(code<=0x0451) h=code-160;
    else if(code< 0x2010) result=1;
    else if(code<=0x2642) h=code-160-7102;
    else if(code< 0x3000) result=1;
    else if(code<=0x33d5) h=code-160-7102-2493;
    else if(code< 0x4e00) result=1;
    else if(code<=0x9fa5) h=code-160-7102-2493-6698;
    else if(code< 0xe76c) result=1;
    else if(code<=0xe864) h=code-160-7102-2493-6698-18374;
    else if(code< 0xf92c) result=1;
```

```
    else if(code<=0xfa29) h=code-160-7102-2493-6698-18374-4295;
    else if(code< 0xfe30) result=1;
    else if(code<=0xfe6b) h=code-160-7102-2493-6698-18374-4295-1030;
    else if(code< 0xff01) result=1;
    else if(code<=0xff5e) h=code-160-7102-2493-6698-18374-4295-1030-149;
    else if(code< 0xffe0) result=1;
    else if(code<=0xffe5) h=code-160-7102-2493-6698-18374-4295-1030-149-129;
    else result=1;

    if( result==1 )
    {
        h = 0x3000-160-7102-2493;
    }
    return h;
}
```

6.3.1.13 GB12345 一对多索引表

GB12345 字符集中，有 94 个内码，同一个内码有不同的写法。如下表：

GB2312	GB12345	GB2312	GB12345	GB2312	GB12345	GB2312	GB12345
摆	擺襬	胡	胡鬍	蒙	蒙濛曚	系	系係繫
板	板闆	划	劃划	弥	彌瀰	咸	咸鹹
表	表錶	回	回迴	面	面麵	向	向嚮
别	别彆	汇	匯彙	蔑	蔑衊	须	須鬚
卜	卜蔔	获	獲穫	辟	辟闢	药	藥葯
才	才纔	饥	饑飢	苹	蘋苹	叶	葉叶
厂	廠厂	几	幾几	凭	憑凭	郁	鬱郁
冲	衝冲	家	家傢	扑	撲扑	御	御禦
丑	醜丑	价	價价	仆	僕仆	吁	吁籲
出	出齣	荐	薦荐	朴	樸朴	愿	願愿
当	當噹	姜	姜薑	千	千韆	云	雲云
党	黨党	尽	盡儘	签	簽籤	脏	臟髒
淀	澱淀	据	據据	纤	纖縴	症	癥症
冬	冬鼕	卷	卷捲	秋	秋鞦	只	衹只隻
斗	鬥斗	克	克剋	曲	曲麴	致	致緻
恶	惡噁	夸	誇夸	确	確确	制	制製
发	發髮	困	困睏	舍	捨舍	种	種种
范	範范	蜡	蠟蜡	术	術术	朱	朱硃
丰	豐丰	腊	臘腊	松	鬆松	筑	築筑
复	復複	累	累纍	苏	蘇囌		
干	幹干	里	裏里	台	臺台檯颱		
谷	谷穀	历	歷曆	坛	壇罈		
刮	刮颳	帘	簾帘	涂	塗涂		
广	廣广	卤	滷鹵	团	團糰		
合	合閤	霉	霉黴	万	萬万		

GB12345 一对多索引表的结构：

```
Struct GB12345_MultiTable
{
```

```
    WORD    incode;   //GB12345 汉字内码
    WORD index1;    //该内码的第一个字在字库中的序号
    WORD index2;    //该内码的第二个字在字库中的序号
    WORD index3;    //该内码的第三个字在字库中的序号
    WORD index4;    //该内码的第四个字在字库中的序号
}
```

6.3.2 ASCII 字符的地址计算

6.3.2.1 5X7 点 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=0x1A73CC
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode –0x20 ) * 8+BaseAdd
```

6.3.2.2 7X8 点 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=0x1A76CC
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
        Address = (ASCIICode –0x20 ) * 8+BaseAdd
```

6.3.2.3 6X12 点 ASCII 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=0x1A79CC
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
      Address = (ASCIICode –0x20 ) * 12+BaseAdd
```

6.3.2.4 8X16 点 ASCII 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=0x1A7FCC
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode –0x20 ) * 16+BaseAdd
```

6.3.2.5 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=0x1A87CC

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

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

6.3.2.6 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=0x1A918C

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

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

6.3.2.7 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=0x1A9E4C

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

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

7 附录

7.1 GB2312 1 区字符 (470 字符)

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 846 个字符；

GB2312　1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	ˉ	ˇ	¨	〃	々	—	～	‖	…	‘	’
B	“	”	〔	〕	〈	〉	《	》	「	」	『	』	〖	〗	【	】
C	±	×	÷	∶	∧	∨	∑	∏	∪	∩	∈	∷	√	⊥	∥	∠
D	⌒	⊙	∫	∮	≡	≌	≈	∽	∝	≠	≮	≯	≤	≥	∞	∵
E	∴	♂	♀	°	′	″	℃	$	¤	￠	￡	‰	§	№	☆	★
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	〓	

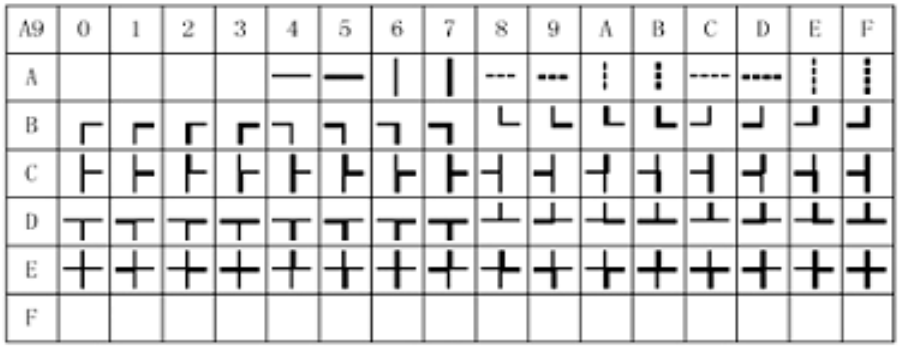
A2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ⅰ	ⅱ	ⅲ	ⅳ	ⅴ	ⅵ	ⅶ	ⅷ	ⅸ	ⅹ					
B		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
C	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
D	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③	④	⑤	⑥	⑦
E	⑧	⑨	⑩	€		㈠	㈡	㈢	㈣	㈤	㈥	㈦	㈧	㈨	㈩	
F		Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	Ⅺ	Ⅻ			

A3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		！	＂	＃	￥	％	＆	＇	（	）	＊	＋	，	－	．	／
B	０	１	２	３	４	５	６	７	８	９	：	；	＜	＝	＞	？
C	＠	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
D	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ	［	＼	］	＾	＿
E	｀	ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ
F	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ	｛	｜	｝	￣	

GT23L24T3Y　　多字符集简繁　字库芯片　　标准字库

GB2312　1 区

A6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
B	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
C		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
D	π	ρ	σ	τ	υ	φ	χ	ψ	ω	'	°	`	:	;	!	?
E	⌒	⌒	⌒	⌒	⌒	⌒	⌒	⌒	⌐	⌐	⌐	⌐	⌐	⌐	⌐	⌐
F	⌒	⌒	⌒	⌒	⌒	⌒										



A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	—	|	|	---	---	¦	¦	----	----	¦	¦
B	┌	┌	┌	┌	┐	┐	┐	┐	└	└	└	└	┘	┘	┘	┘
C	├	├	├	├	├	├	├	├	┤	┤	┤	┤	┤	┤	┤	┤
D	┬	┬	┬	┬	┬	┬	┬	┬	┴	┴	┴	┴	┴	┴	┴	┴
E	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼
F																

7.2 补丁文件

序号	补丁描述	修补方式	对应文档	版本号	发布时间
1	12、16 点 UNICODE 对应字库表。	码本对应表	《GT23L24T3Y-12-16-UNICODE 字库码本》	V1.00	2010-8-20
2	24 点 UNICODE 对应字库表	码本对应表	《GT23L24T3Y-24-UNICODE 字库码本》	V1.00	2010-8-20
3	12、16 点 BIG5 对应字库表	码本对应表	《GT23L24T3Y-12-16-BIG5 字库码本》	V1.00	2010-8-20
4	24 点 BIG5 对应字库表	码本对应表	《GT23L24T3Y-24-BIG5 字库码本》	V1.00	2010-8-20
5	16 点 BIG5 缺损字符码本	码本表	《GT23L24T3Y-16-BIG5 缺损字符码本》	V1.00	2010-8-20
6	24 点 BIG5 缺损字符码本	码本表	《GT23L24T3Y-24-BIG5 缺损字符码本》	V1.00	2010-8-20

GT23L24T3Y　　多字符集简繁　字库芯片　　标准字库