GT23L16M2Y3-S

标准汉字字库芯片

11X12 点阵

15X16 点阵

竖置横排

GBK 全拼输入法码表

GT-三维部件输入法

（SPI 接口/PLII 接口）

用户手册

VER 2.7

2007-Q4

版本修订记录

版本号	修改内容	日期	备注
V2.5	生成数据手册	2007-04	
v2.7	增加 GT 输入法，更名为 GT23L16M2Y3-S	2007-10	

目　　录

1 概述 4
2 特点 5
3 引脚配置 7
4 引脚描述 7
4.1 SPI 接口引脚描述 8
4.2 PLII 接口引脚描述 8
5 SPI 接口/PLII 接口与主机接口电路示意图 9
5.1 SPI 接口与主机接口电路示意图 9
5.2 PLII 接口与主机接口电路示意图 10
6 操作指令 11
6.1 SPI 接口模式下操作 11
6.2 PLII 接口模式下操作 13
7 电气特性 15
7.1 绝对最大额定值 15
7.2 DC 特性 15
7.3 AC 特性 15
8 封装尺寸：SOP20 19
9 功能描述 20
9.1 汉字点阵排列格式 20
9.2 汉字点阵字库结构 31
9.3 汉字点阵在芯片中的地址计算方法 33
9.4 全拼拼音－GBK 全拼输入法（21003 字） 40
10 附录 42
10.1 GB18030　1 区和 5 区 (字符区) 42
10.2 5×7 点 ASCII 字符（96 字符） 46
10.3 7×8 点 ASCII 字符（96 字符） 46
10.4 Unicode 字符区字符 47
10.5 8×16 点扩展字符区 ASCII 字符（126 字符） 53
10.6 8×16 自定义特殊字符区 ASCII 字符（64 字符） 54
10.7 8×16 自定义国际音标字符区 ASCII 字符（96 字符） 54
10.8 汉字内码－拼音对照表（部分） 55

1 概述

GT23L16M2Y3是一款支持GB18030字符集（27484汉字）的11X12点阵、15X16点阵字库芯片，另外还含有2套12点高的半角不等宽ASCII字符、1套6X12点扩展字符、2套16点高的半角不等宽ASCII字符、1套8X16点扩展字符、1套5X7点ASCII字符、1套7X8点ASCII字符。同时内含Unicode字符区拉丁文系（LATIN）、希腊文系（LATIN）、西里尔文系（CYRILLIC）文系字符基本可以涵盖81个国家及地区。同时还有非常丰富的自定义区的各种图形符号，以及12点阵和16点阵的国际音标字符。

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，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。

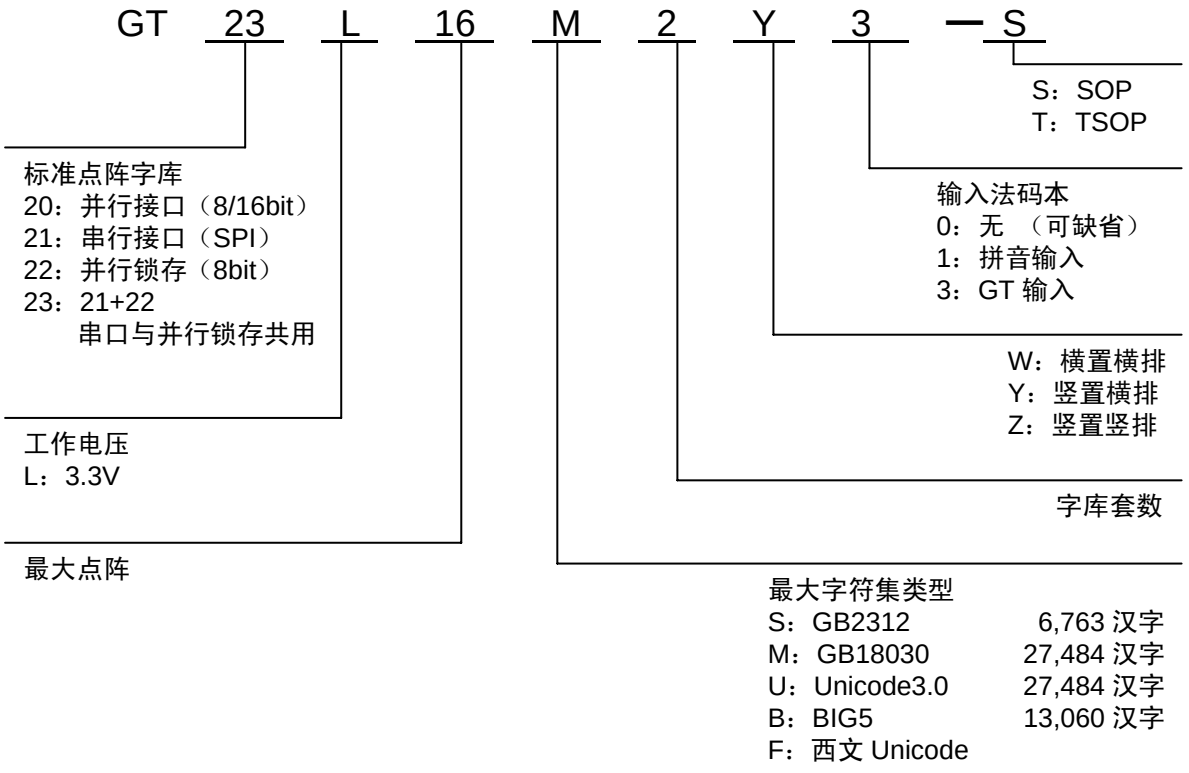
用户只要知道汉字的内码，就可以通过本手册提供的方法计算出该汉字点阵在芯片中的地址，然后就可从该地址连续读出点阵信息。

同时本字库芯片还含有GBK（21003汉字）全拼输入法码表，只要将键值串相应的拼音ASCII字符进行索引，即可得到该键值串所对应的汉字内码群。因此本汉字库芯片根据用户的具体需要即可以输入汉字相应的输入编码获取对应该的汉字内码群，也可以输入汉字的内码获取对应该汉字的点阵字形数据

GT23L16M2Y2 支持2种不同的总线接口：即SPI接口、PLII接口。

本字库芯片内含 GT 输入法码本，该输入法适用于数字小键盘，如手机、遥控器、电话终端、税控收款机等。

命名规则（标准点阵字库芯片）



2 特点

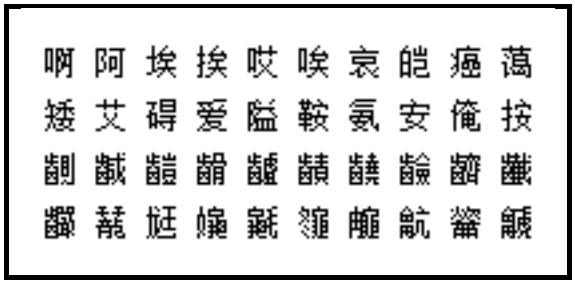
- 数据总线：SPI 总线兼容的串行接口/PLII 接口
- 点阵排列方式：字节竖置横排
- 访问速度：SPI 时钟频率：40MHz(max.)
  PLII 访问速度：70ns(max.) @3.3V
- 工作电压：2.7V – 3.6V
- 电流：
  工作电流：30mA(max.) @3.3V
  待机电流：15uA(max.) @3.3V
- 封装： SOP20
- 字库内容：

汉 字 字 库	
11×12 点	GB18030 （27,484 汉字） （1,038 字符）
15×16 点	GB18030 （27,484 汉字） （1,038 字符）

ASCII 字符	
6×12 点	扩展字符（126 字符） 国际音标（96 字符）
12 点阵	ASCII 方头（96 字符）
12 点阵	ASCII 白正（96 字符）
8×16 点	扩展字符（126 字符） 国际音标（96 字符）
16 点阵	方头（96 字符）
16 点阵	白正（96 字符）
5×7 点	ASCII 字符（96 字符）
7×8 点	ASCII 字符（96 字符）
8×16 点	Unicode 字符区（786 字符） －拉丁文系（376 字符） －西里尔文系（250 字符） －希腊字字符区（96 字符） 自定义字符区（64 字符）

- 字型样张

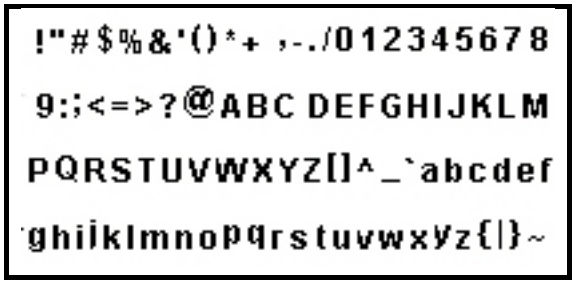
11×12 点 GB18030（27,484 汉字）



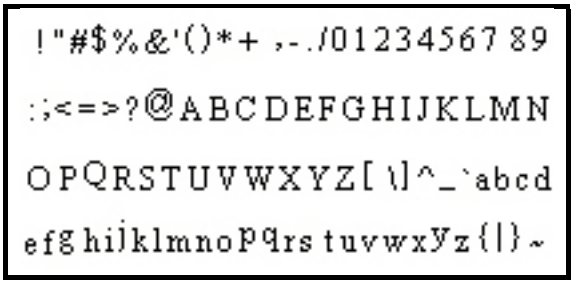
6×12 点 ASCII 扩展字符（126 字符）



12 点阵 ASCII 方头（96 字符）



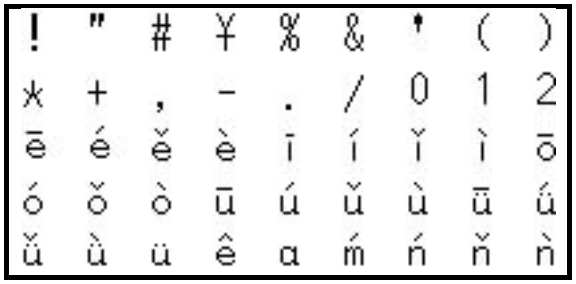
12 点阵 ASCII 白正（96 字符）



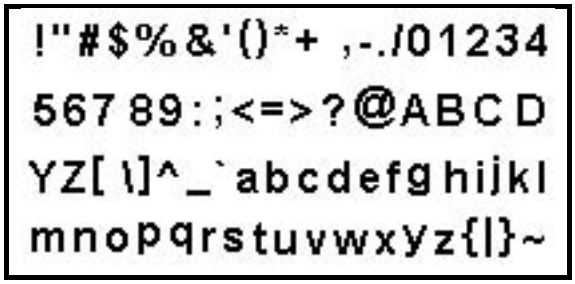
15×16 点 GB18030（27,484 汉字）



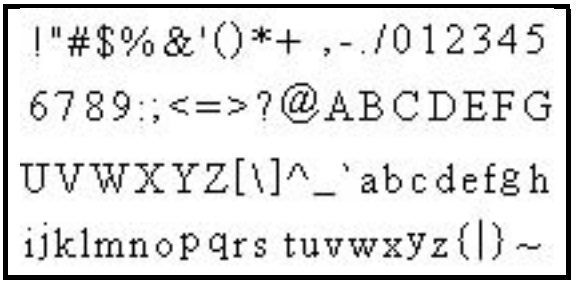
8×16 点 ASCII 扩展字符（126 字符）



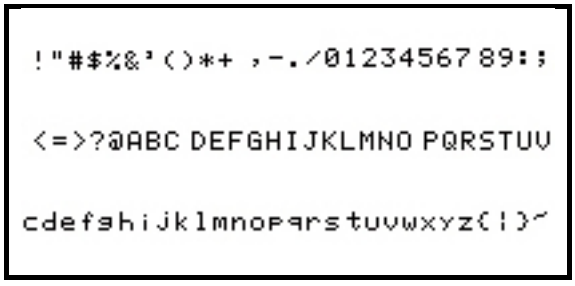
16 点阵 ASCII 方头（96 字符）



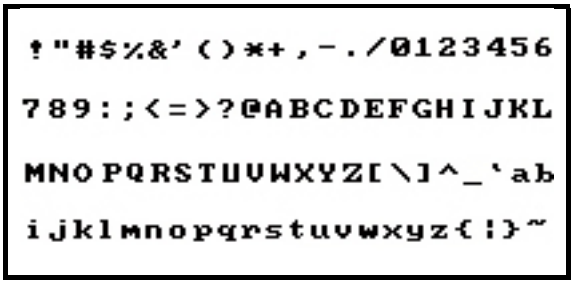
16 点阵 ASCII 白正（96 字符）



5×7 点 ASCII 字符（96 字符）

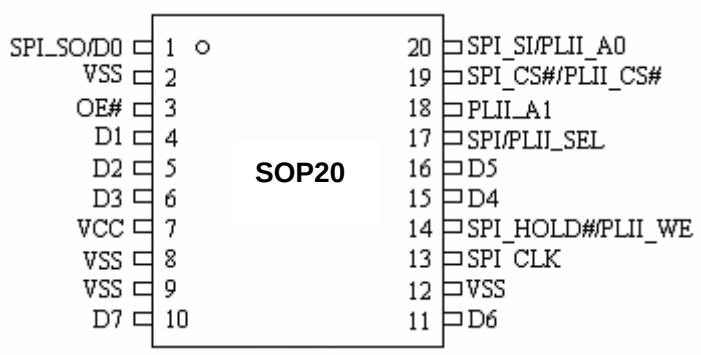


7×8 点 ASCII 字符（96 字符）



其它各种字符样张（如 UNICODE 多国字符等） 详见附件

3 引脚配置



SOP20

4 引脚描述

SOP20	名称	描述	
		GT21L(SPI 接口)	GT22L(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

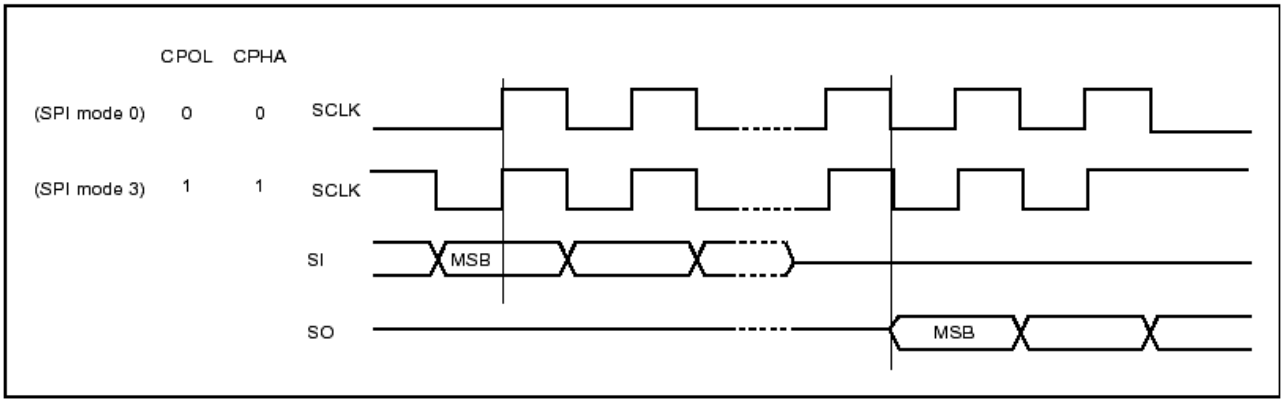
4.1 SPI 接口引脚描述

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CE#下降沿，CE#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。

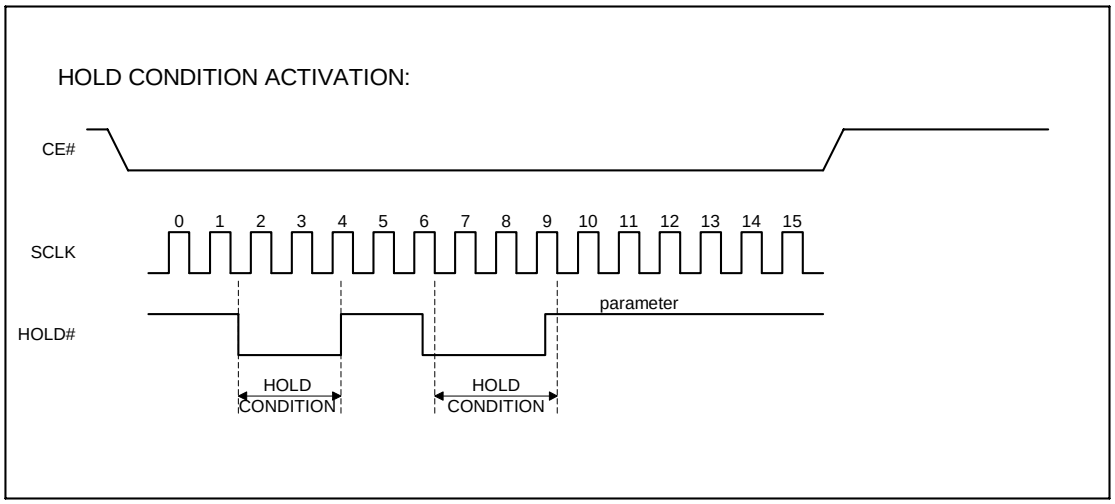


总线挂起输入（HOLD#）：

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。



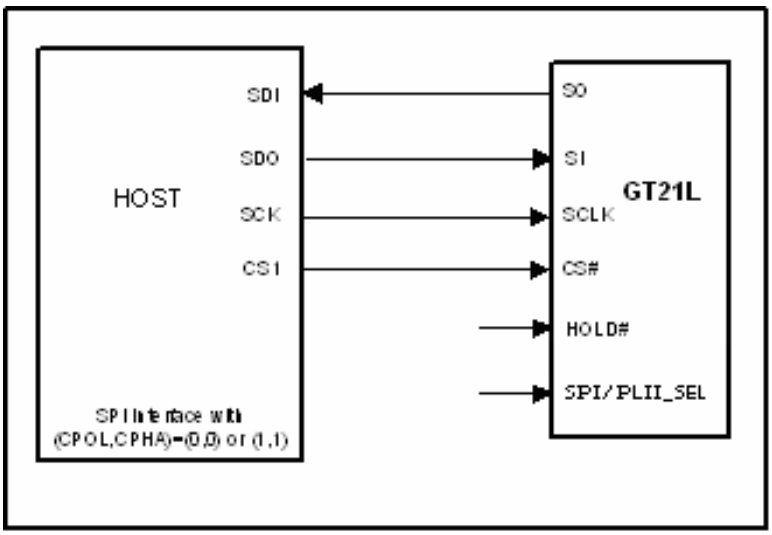
4.2 PLII 接口引脚描述

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

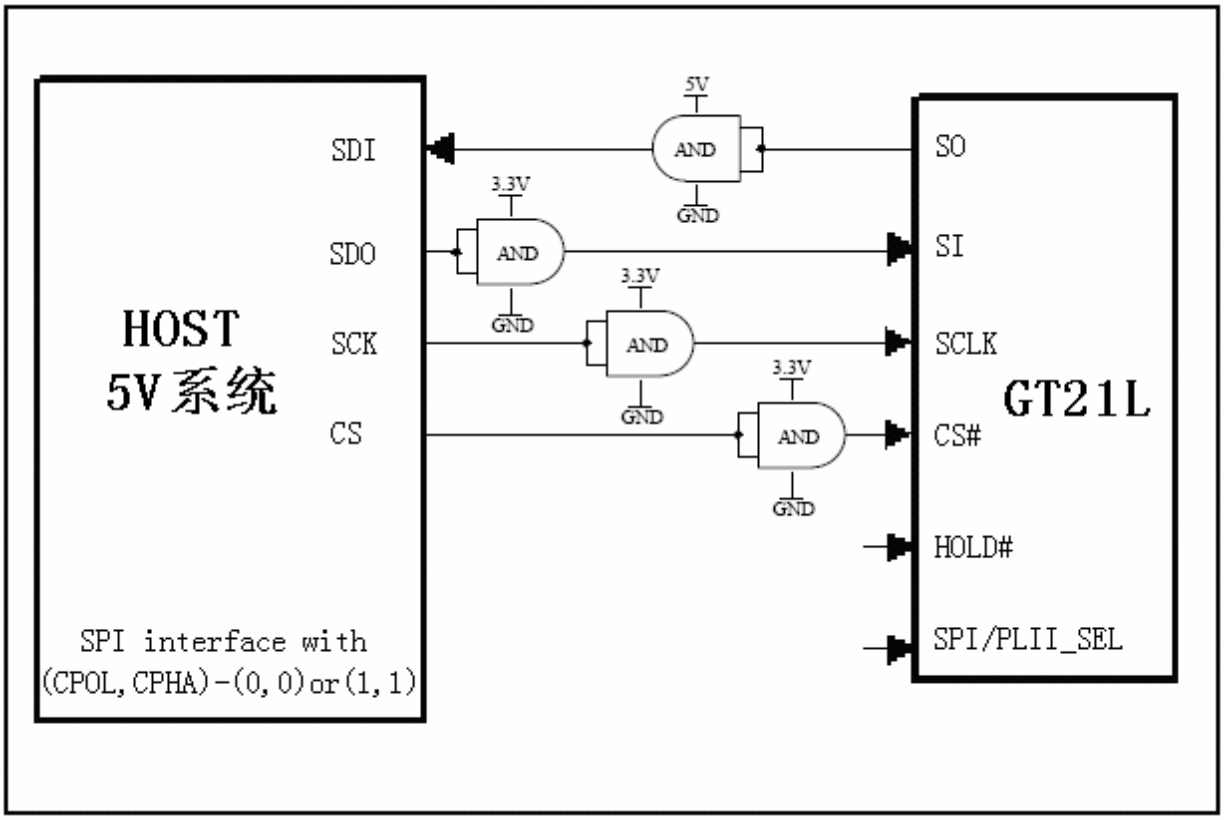
5 SPI 接口/PLII 接口与主机接口电路示意图

5.1 SPI 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）悬空，字库芯片选择 SPI 接口模式，与主机接口电路连接可以参考下图。

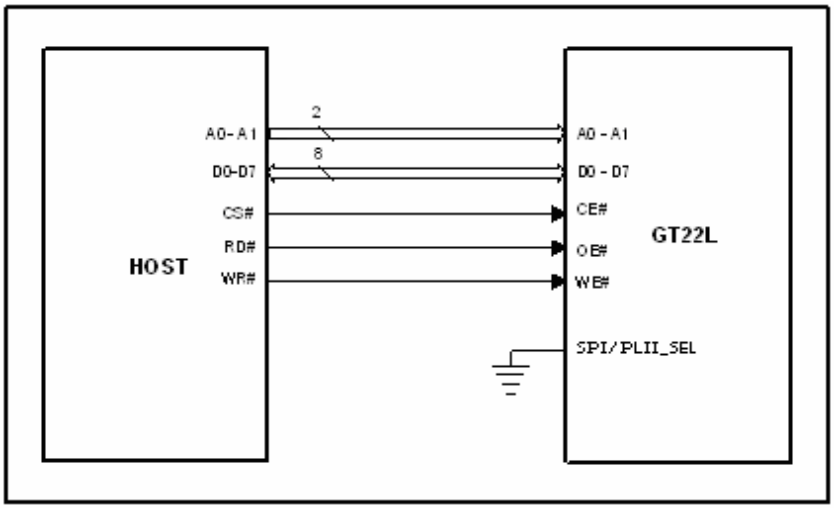


注意：若是采用系统电压为 5V 的，则需要进行电平转换匹配连接 GT21L 库芯片，可以参考下图。

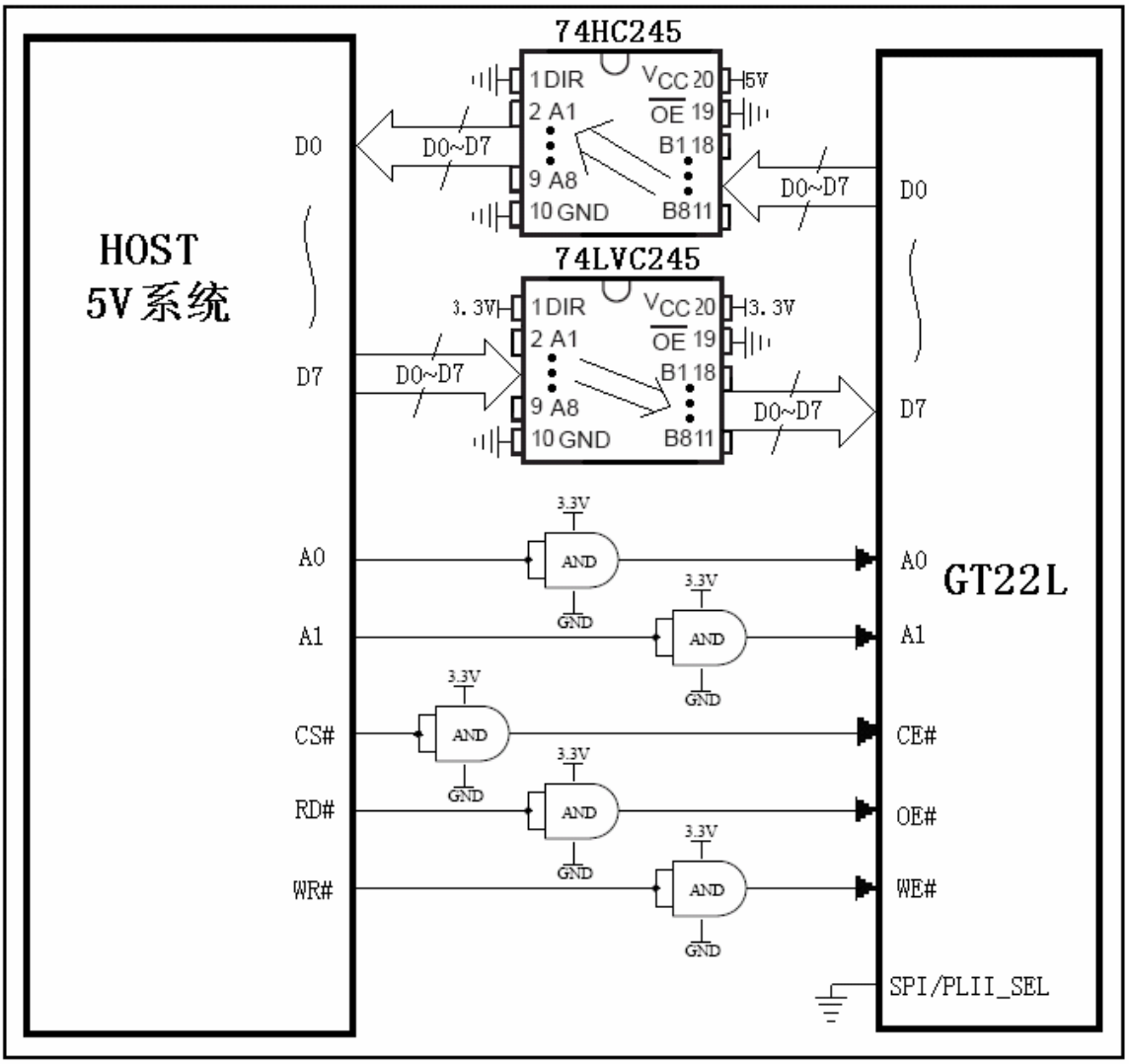


5.2 PLII 接口与主机接口电路示意图

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



注意：若是采用系统电压为 5V 的，则需要进行电平转换匹配(如:通过 74LVC245 进行电平转换)，连接 GT20L 库芯片，可以参考下图。



6 操作指令

6.1 SPI 接口模式下操作

6.1.1 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”)。.

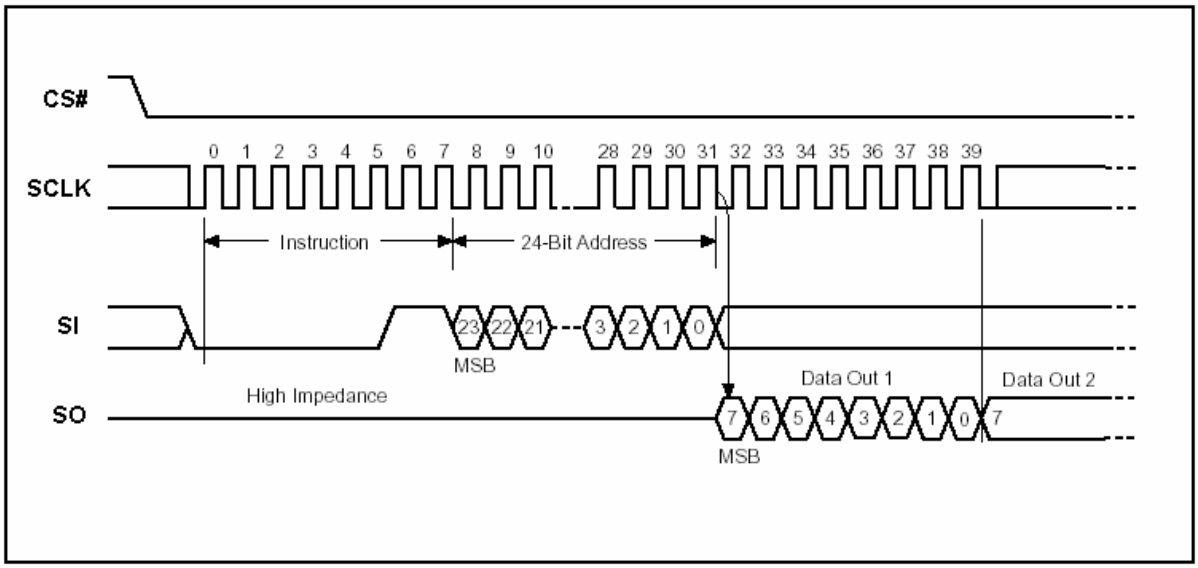
6.1.2 Read Data Bytes（一般读取）

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



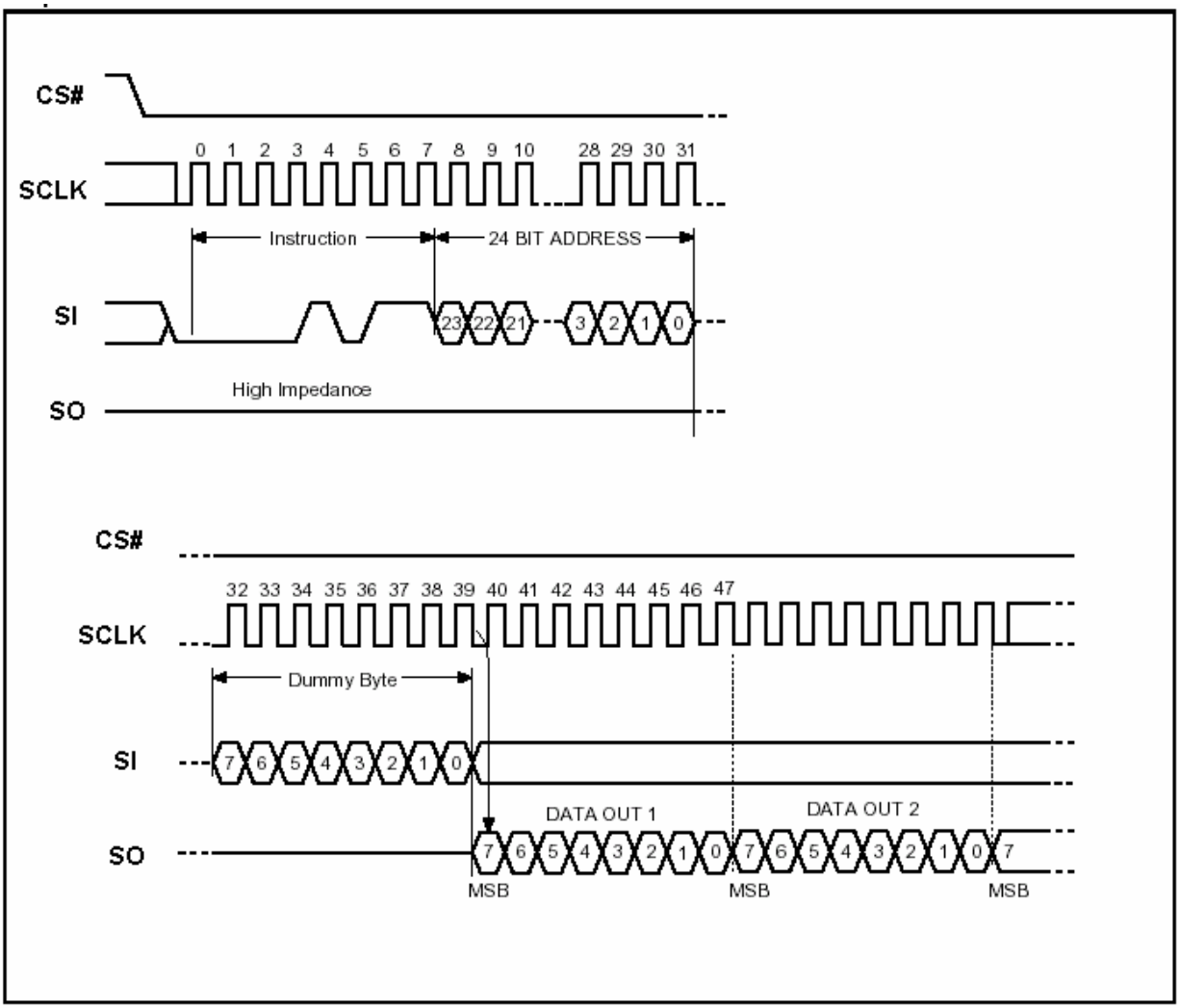
6.1.3 Read Data Bytes at Higher Speed（快速读取点阵数据）

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



6.2 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。

6.2.1 信号描述

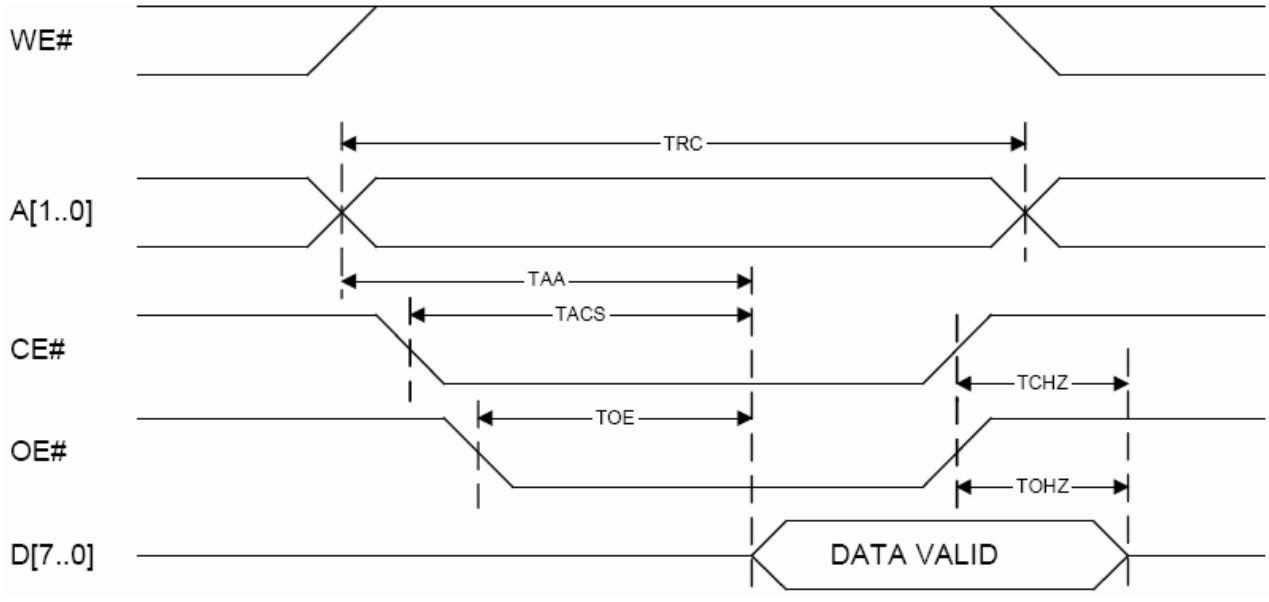
Pin name	I/O	描述
A[1..0]	I	地址寄存器寻址
D[7..0]	I/O	地址输入/数据输出
CE#	I	片选信号输入，低有效
OE#	I	“输出使能”信号输入，OE# 为低时输出使能
WE#	I	“写使能”信号输入，WE# 为低时写使能

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

6.2.2 读操作

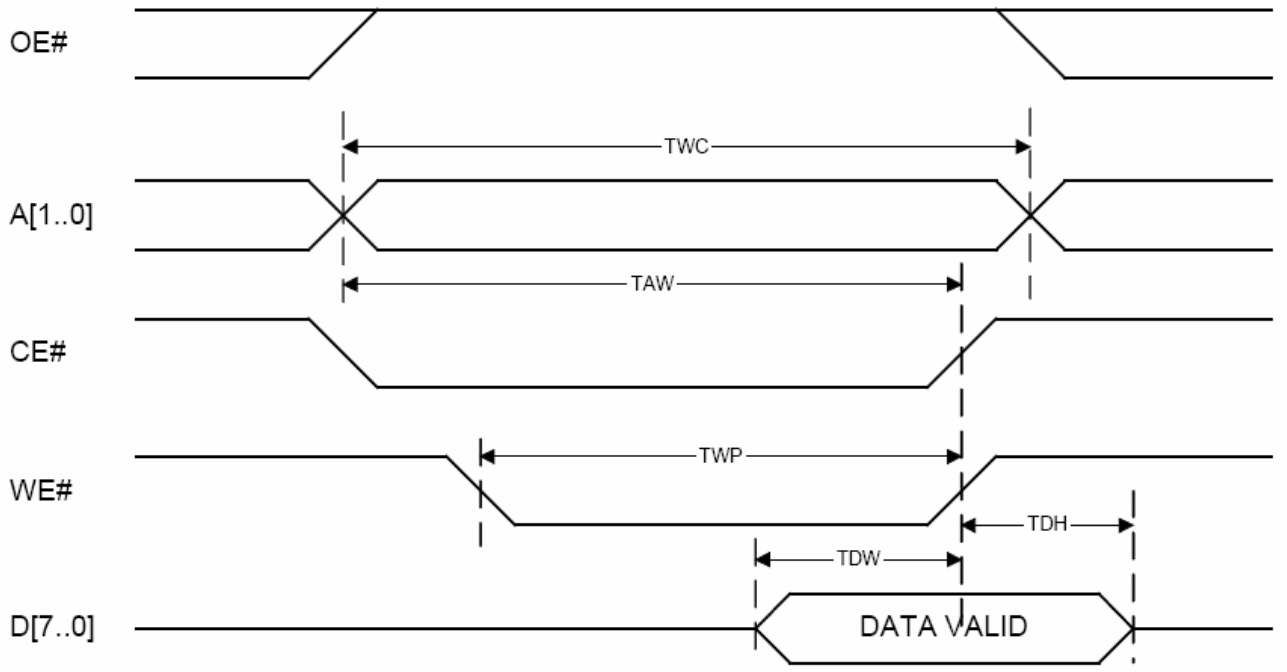
读字库芯片内部的点阵字库数据时，HOST 写字库芯片的 3 个地址寄存器，字库芯片把对应地址的数据送给 HOST。在 OE#和 CE#同时为低电平、WE#为高电平的情况下，HOST 可以从数据总线（D[7..0]）读出字库芯片的个字节的数据。当出现 OE# 或 CE# 中的一个信号变高，则字库芯片内部在此时刻把地址寄存器增 1，并保持地址寄存器的值。



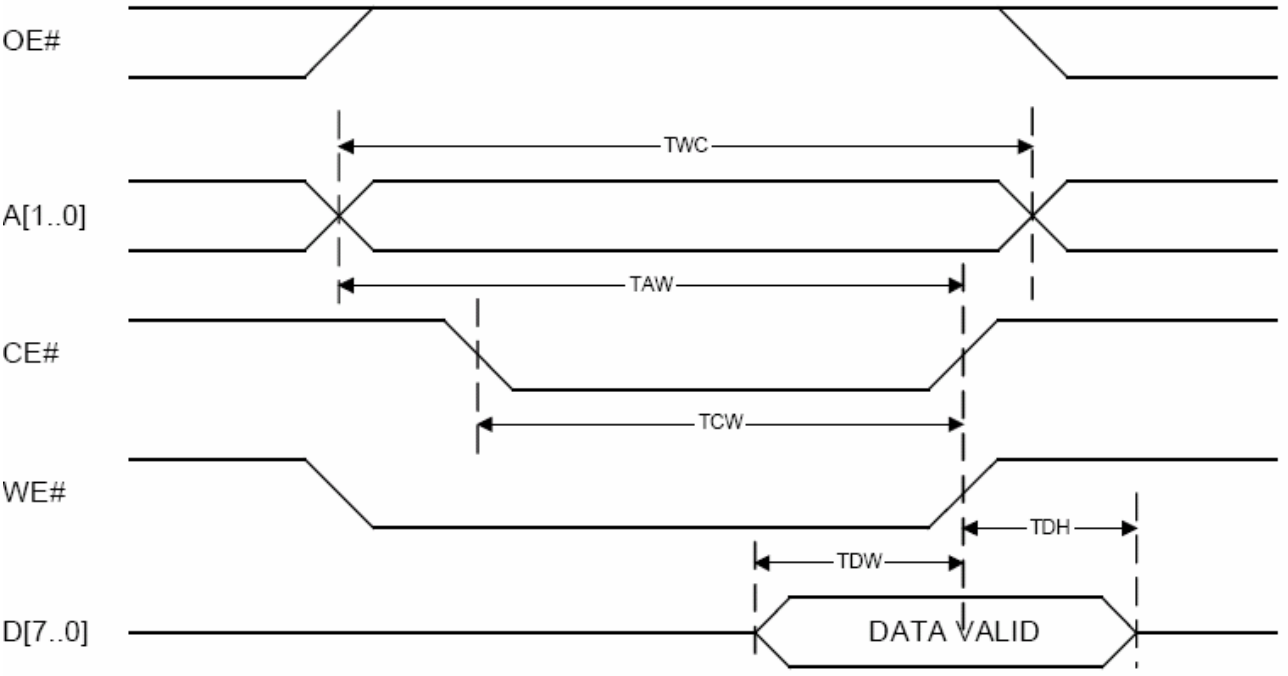
读周期时序波形图

6.2.3 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）

7 电气特性

7.1 绝对最大额定值

Symbol	Parameter	Min.	Max.	Unit	Condition
T_{OP}	Operating Temperature	0	70	℃	
T_{STG}	Storage Temperature	-65	125	℃	
VCC	Supply Voltage	2.7	3.6	V	
V_{IN}	Input Voltage	-0.5	VCC+0.5	V	
GND	Power Ground	0	0	V	

7.2 DC 特性

Condition：T_{OP} = 0℃ to 70℃，GND=0V

Symbol	Parameter	Min.	Max.	Unit	Condition
I_{DD}	VCC Supply Current(active)		40	mA	
I_{SB}	VCC Standby Current		10	uA	
V_{IL}	Input LOW Voltage	-0.3	0.6	V	VCC=3V
V_{IH}	Input HIGH Voltage	2.2	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，

7.3 AC 特性

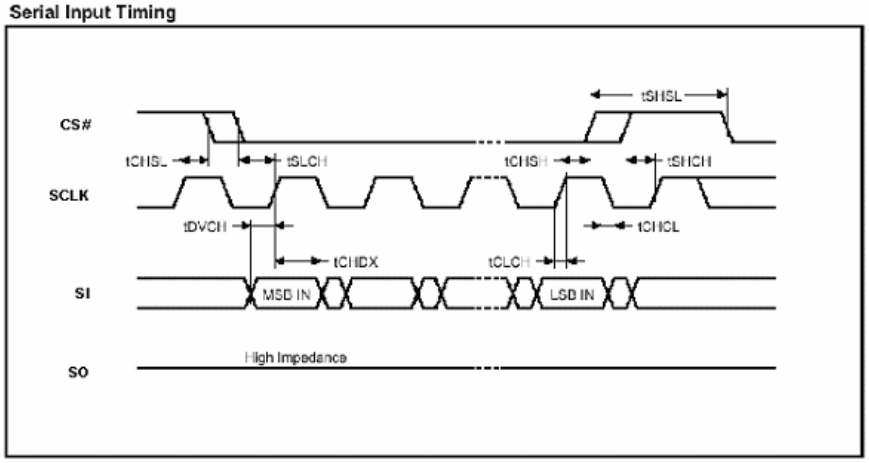
7.3.1 SPI 接口模式下 AC 特性

Condition：T_{OP} = 0℃ to 70℃，VCC= 2.7V to 3.6V

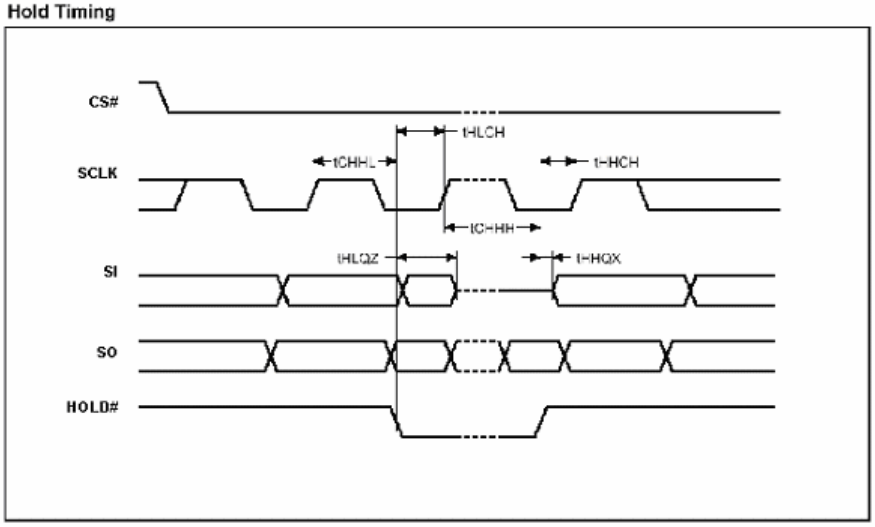
Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	40	MHz
t_{CH}	t_{CLH}	Clock High Time	11		ns
t_{CL}	t_{CLL}	Clock Low Time	11		ns
t_{CLCH}		Clock Rise Time(peak to peak)	0.1		V/ns
t_{CHCL}		Clock Fall Time (peak to peak)	0.1		V/ns
t_{SLCH}	t_{CSS}	CS# Active Setup Time (relative to SCLK)	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

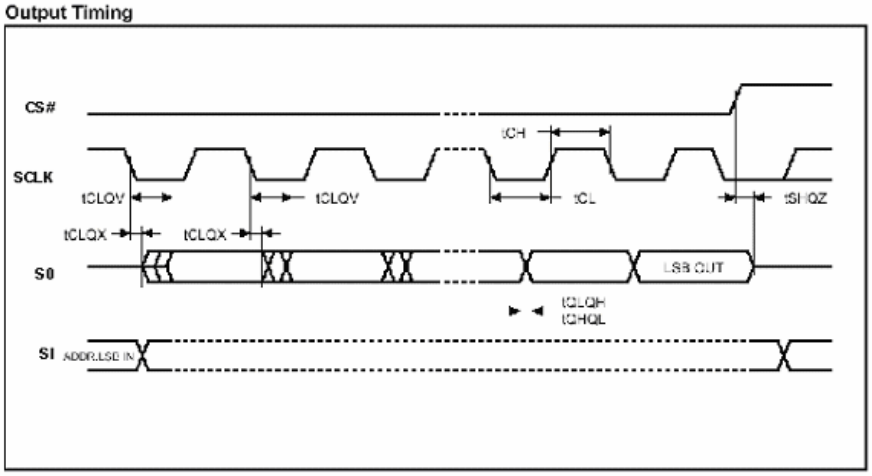
Serial Input Timing



Hold Timing



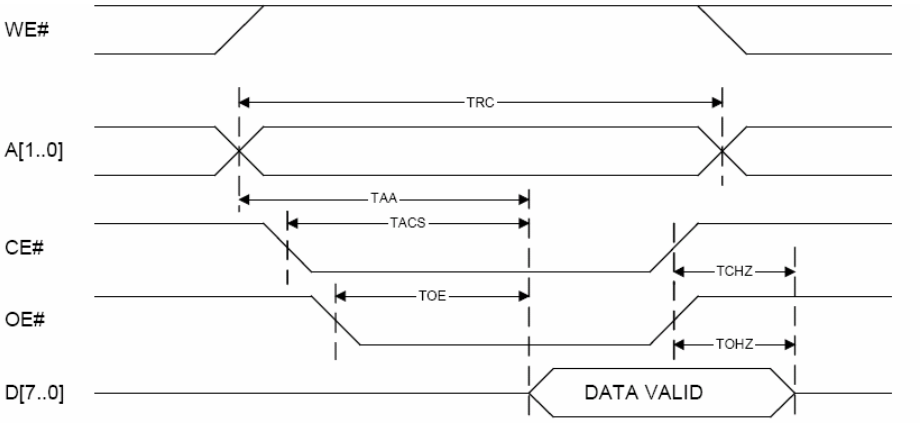
Output Timing



7.3.2 PLII 接口模式下 AC 特性

7.3.2.1 读周期时间特性

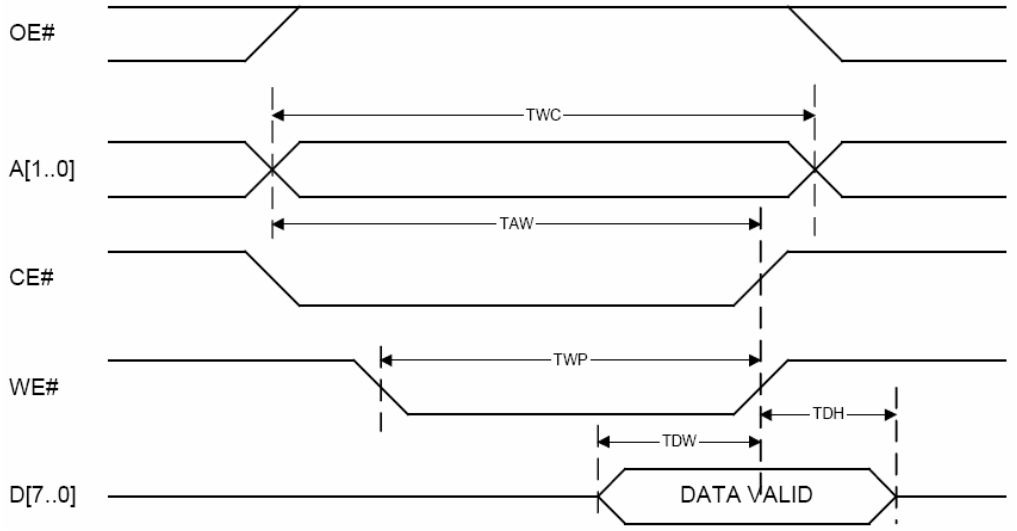
Symbol	Parameter	Min.	Max.	Unit
TRC	Read Cycle Time	70	-	ns
TAA	Address Access Time	-	60	ns
TACS	Chip Select Access Time	-	60	ns
TOE	Output Enable to Output Valid	-	30	ns
TCHZ	Chip Deselect to Output in High-Z	-	10	ns
TOHZ	Output Disable to Output in High-Z	-	10	ns



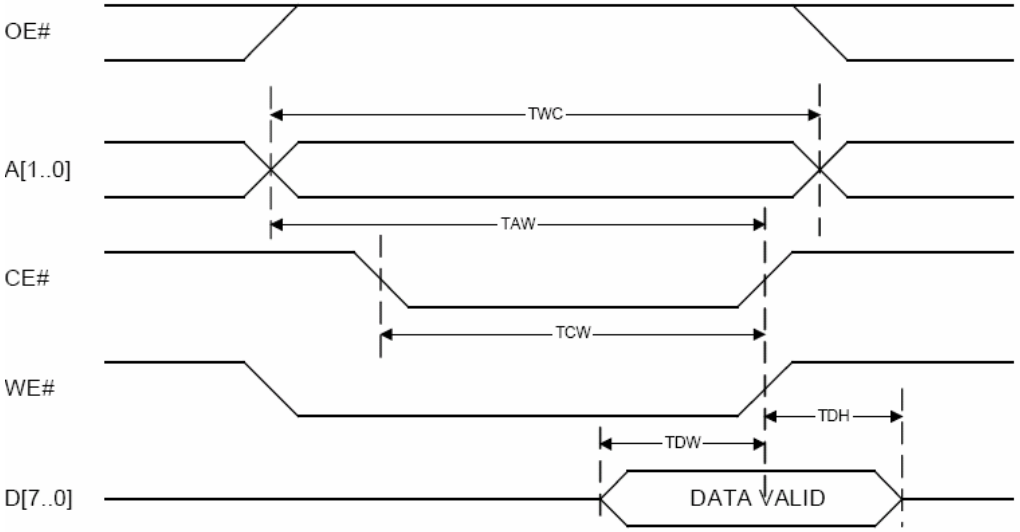
读周期时序波形图

7.3.2.2 写周期时间特性

Symbol	Parameter	Min.	Max.	Unit
TWC	Write Cycle Time	70		ns
TAW	Address Valid to End-of-Write	60		ns
TCW	Chip Select to End-of-Write	35		ns
TWP	Write Pulse Width	35		ns
TDW	Dada to Write Time Overlap	30		ns
TDH	Data Hold from Write Time	5		ns



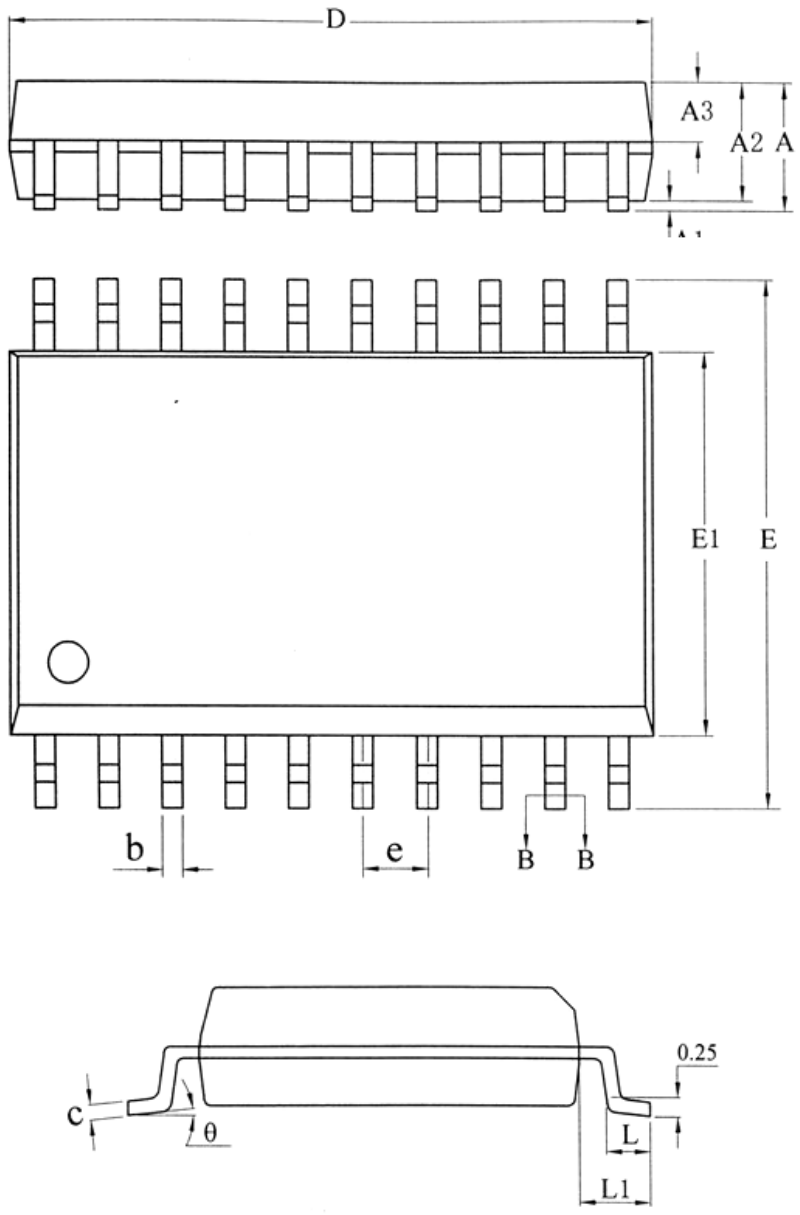
写周期时序波形图（WE#控制的时序）



写周期时序波形图（CE#控制的时序）

8 封装尺寸：SOP20

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°

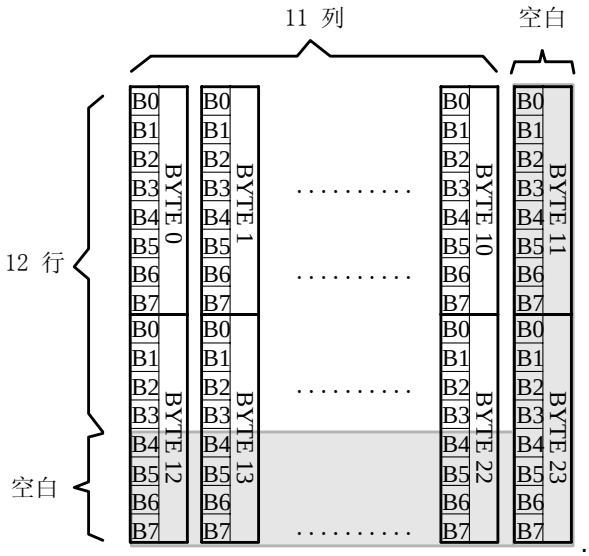
9 功能描述

9.1 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 word mode 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。

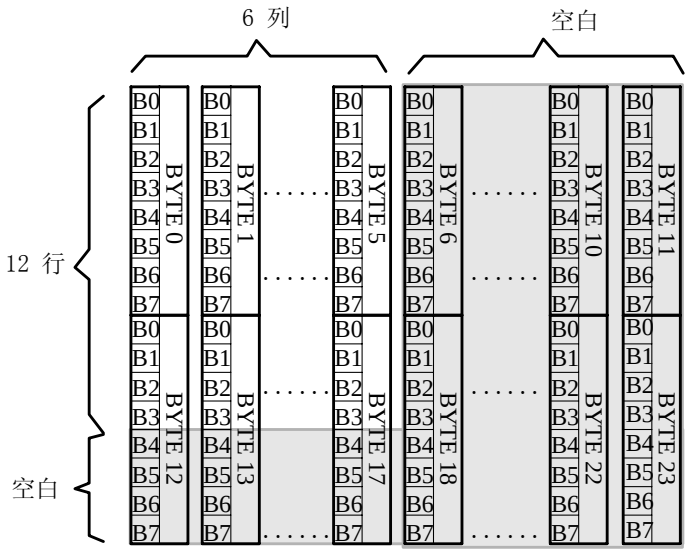
9.1.1 11X12 点汉字排列格式

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：



9.1.2 6X12 点扩展字符排列格式

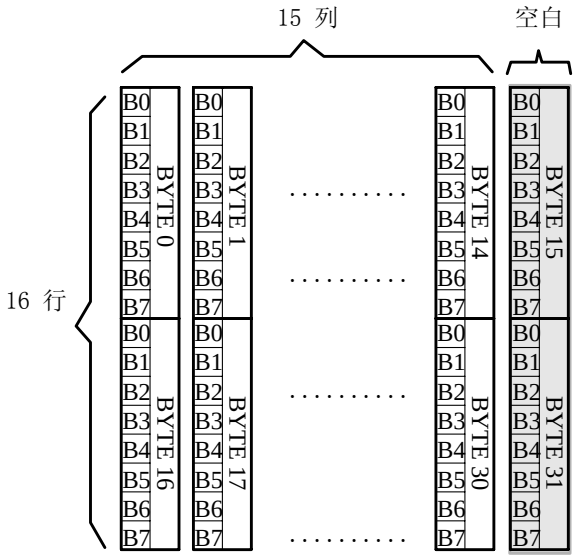
6X12 点扩展字符的信息需要 24 个字节(BYTE 0 – BYTE 23)来表示。但是由于该字符为标准的 6X12 点格式，而存储空间是按照 11X12 点 BYTE 取整进行存储的（即 24 BYTES），需注意在排版时作相应的调整。



6X12 点扩展字符排列格式图

9.1.3 15X16 点汉字排列格式

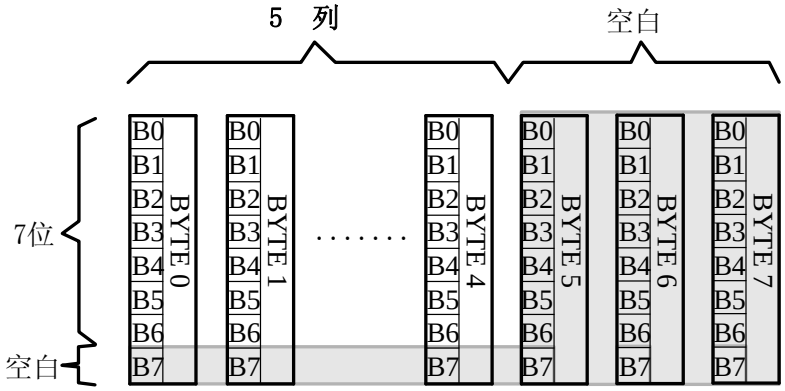
15X16 点汉字的信息需要 32 个字节（BYTE 0 – BYTE 31）来表示。该 15X16 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：



15X16 点汉字排列格式图

9.1.4 5X7 点 ASCII 字符排列格式

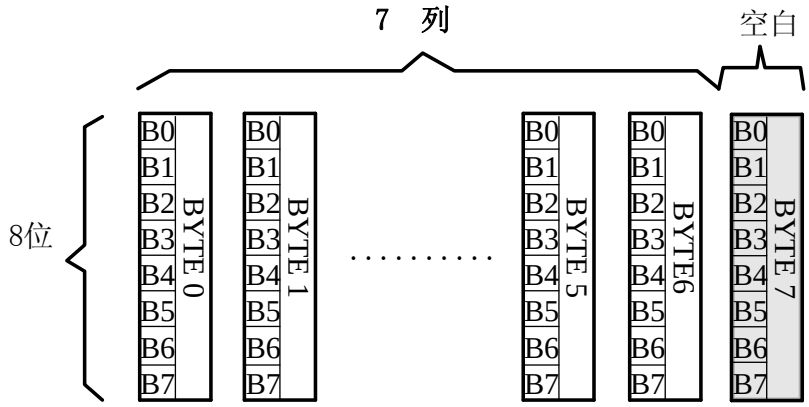
5X7 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



5X7 点 ASCII 字符排列格式图

9.1.5 7X8 点 ASCII 字符排列格式

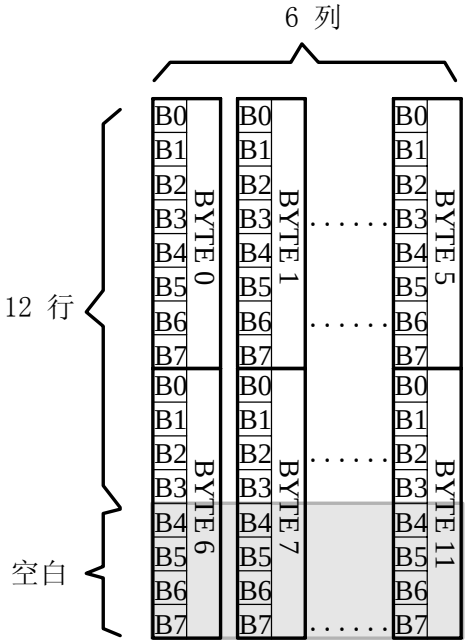
7X8 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



7X8 点 ASCII 字符排列格式图

9.1.6 6X12 点 ASCII 扩展字符排列格式

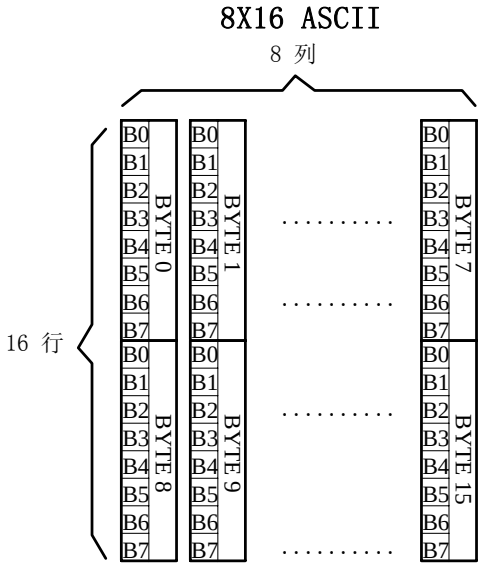
6X12 点 ASCII 的信息需要 12 个字节（BYTE 0 – BYTE11）来表示。



6X12 点 ASCII 字符排列格式图

9.1.7 8X16 点 ASCII 扩展字符排列格式

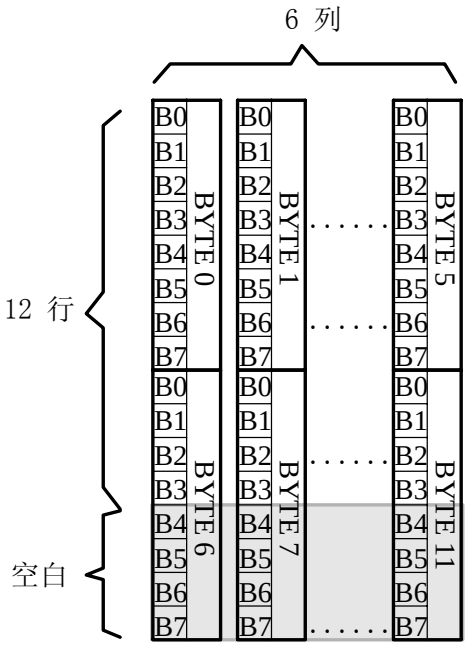
8X16 点 ASCII 的信息需要 16 个字节（BYTE 0 – BYTE15）来表示。



8x16 点 ASCII 字符排列格式图

9.1.8 6X12 点国际音标字符排列格式

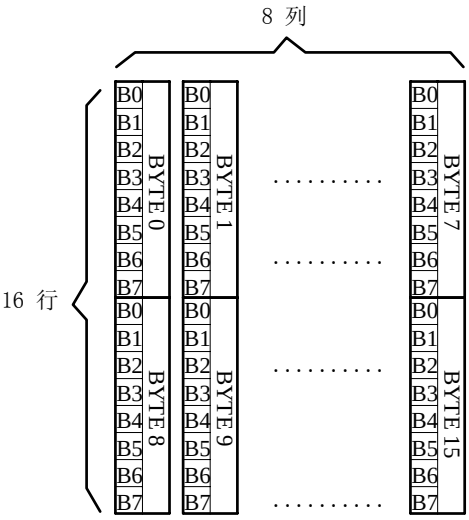
6X12 点国际音标信息需要 12 个字节（BYTE 0 – BYTE12）来表示。



6X12 点国际音标字符排列格式

9.1.9 8X16 点国际音标字符排列格式

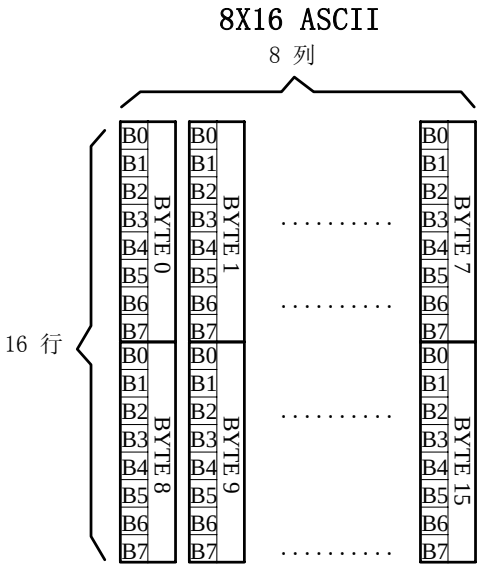
8X16 点国际音标字符信息需要 16 个字节（BYTE 0 – BYTE15）来表示。



8X16 点 ASCII 国际音标字符格式

9.1.10 8X16 点 Unicode 字符（拉丁语、希腊语、西里尔语）及自定义字符区排列格式

8X16 点 ASCII 的信息需要 16 个字节（BYTE 0 – BYTE15）来表示。



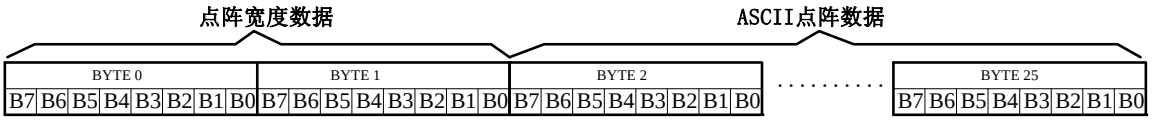
8x16 点排列格式图

9.1.11 12 点阵 ASCII 方头字符排列格式

12 点阵 ASCII 方头字符的信息需要 26 个字节（BYTE 0 – BYTE25）来表示。

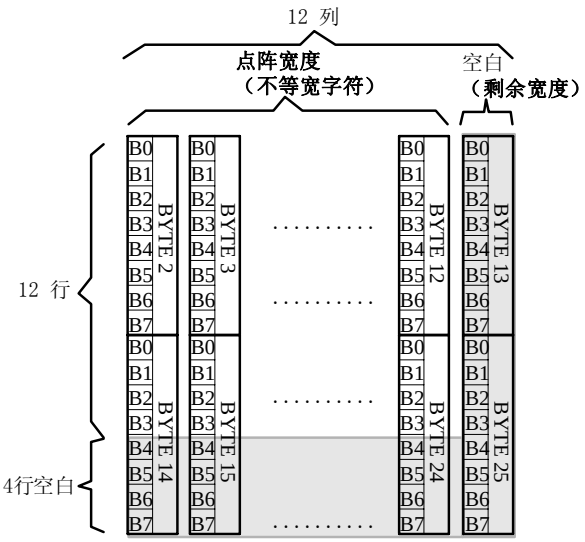
■ 存储格式

由于 ASCII 方头字符是不等宽的，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

不等宽 ASCII 字符的存储结构是以宽度为 BYTE 取整的，根据不同字符宽度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。



12 点阵 ASCII 方头字符排列格式图

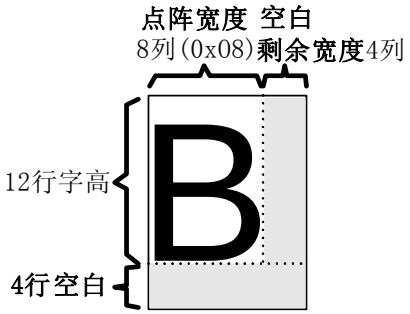
例如：ASCII 方头字符 B

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。（见下图）



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

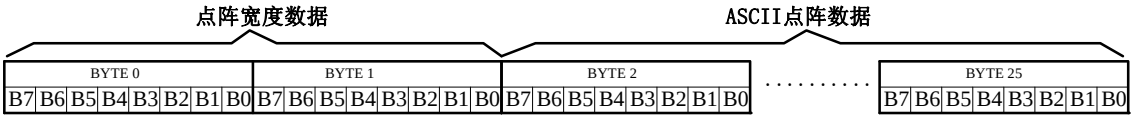
为 ASCII 方头字符 B 的点阵数据。

9.1.12 12 点阵 ASCII 白正字符排列格式

12 点阵 ASCII 白正字符的信息需要 26 个字节（BYTE 0 – BYTE25）来表示。

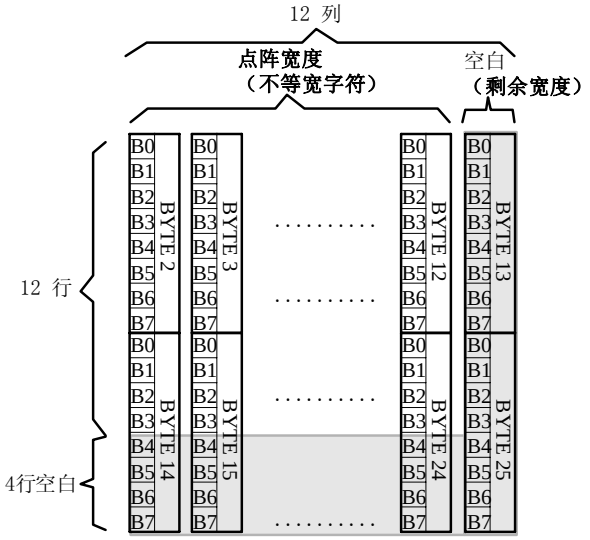
■ 存储格式

由于 ASCII 白正字符是不等宽的，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

不等宽 ASCII 字符的存储结构是以宽度为 BYTE 取整的，根据不同字符宽度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。



16 点阵 ASCII 白正字符排列格式图

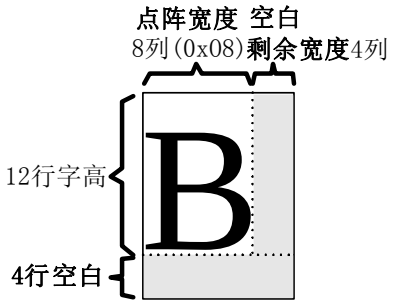
例如：ASCII 白正字符 B

0-25BYTE 的点阵数据是: 00 08 08 F8 88 88 88 F0 00 00 00 00 00 00 08 0F 08 08 08 08 07 00 00 00 00 00

其中：

BYTE0~ BYTE1: 00 08

为 ASCII 白正字符 B 的点阵宽度数据，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。（见下图）



BYTE2-25: 08 F8 88 88 88 F0 00 00 00 00 00 00 08 0F 08 08 08 08 07 00 00 00 00 00

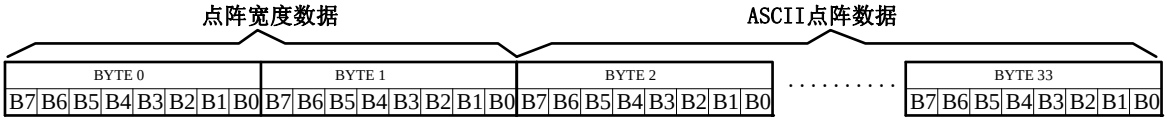
为 ASCII 白正字符 B 的点阵数据。

9.1.13 16 点阵 ASCII 方头字符排列格式

16 点阵 ASCII 方头字符的信息需要 34 个字节（BYTE 0 – BYTE33）来表示。

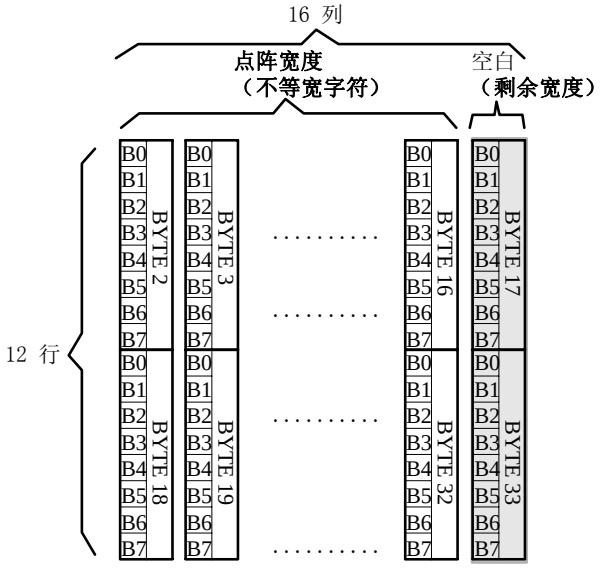
■ 存储格式

由于 ASCII 方头字符是不等宽的，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

不等宽 ASCII 字符的存储结构是以宽度为 BYTE 取整的，根据不同字符宽度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。



16 点阵 ASCII 方头字符排列格式图

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

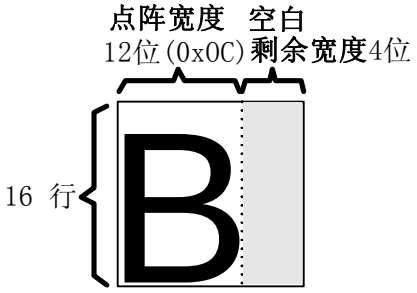
其中：

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。

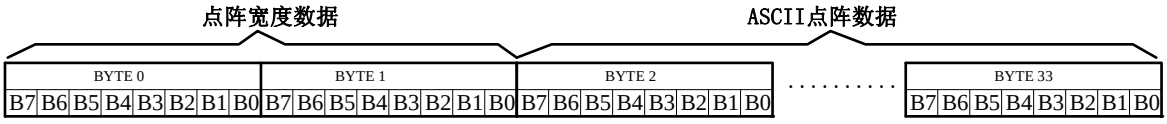


9.1.14 16 点阵 ASCII 白正字符排列格式

16 点阵 ASCII 白正字符的信息需要 34 个字节（BYTE 0 – BYTE33）来表示。

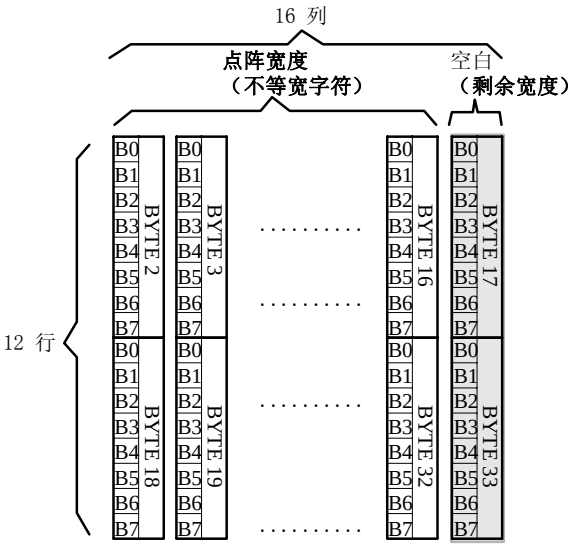
■ 存储格式

由于 ASCII 白正字符是不等宽的，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

不等宽 ASCII 字符的存储结构是以宽度为 BYTE 取整的，根据不同字符宽度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。



16 点阵 ASCII 白正字符排列格式图

例如：ASCII 白正字符 B

0-33BYTE 的点阵数据是: 00 0A 10 10 F0 10 10 10 20 C0 00 00 00 00 00 00 00 00 40 40 7F 42 42 42 43 24 18 00 00 00 00 00 00 00

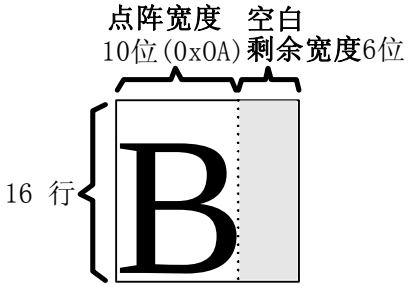
其中：

BYTE0~ BYTE1: 00 0A

为 ASCII 白正字符 B 的点阵宽度数据，即：10 位宽度。字符后面有 6 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: 10 10 F0 10 10 10 20 C0 00 00 00 00 00 00 00 00 40 40 7F 42 42 42 43 24 18 00 00 00 00 00 00 00

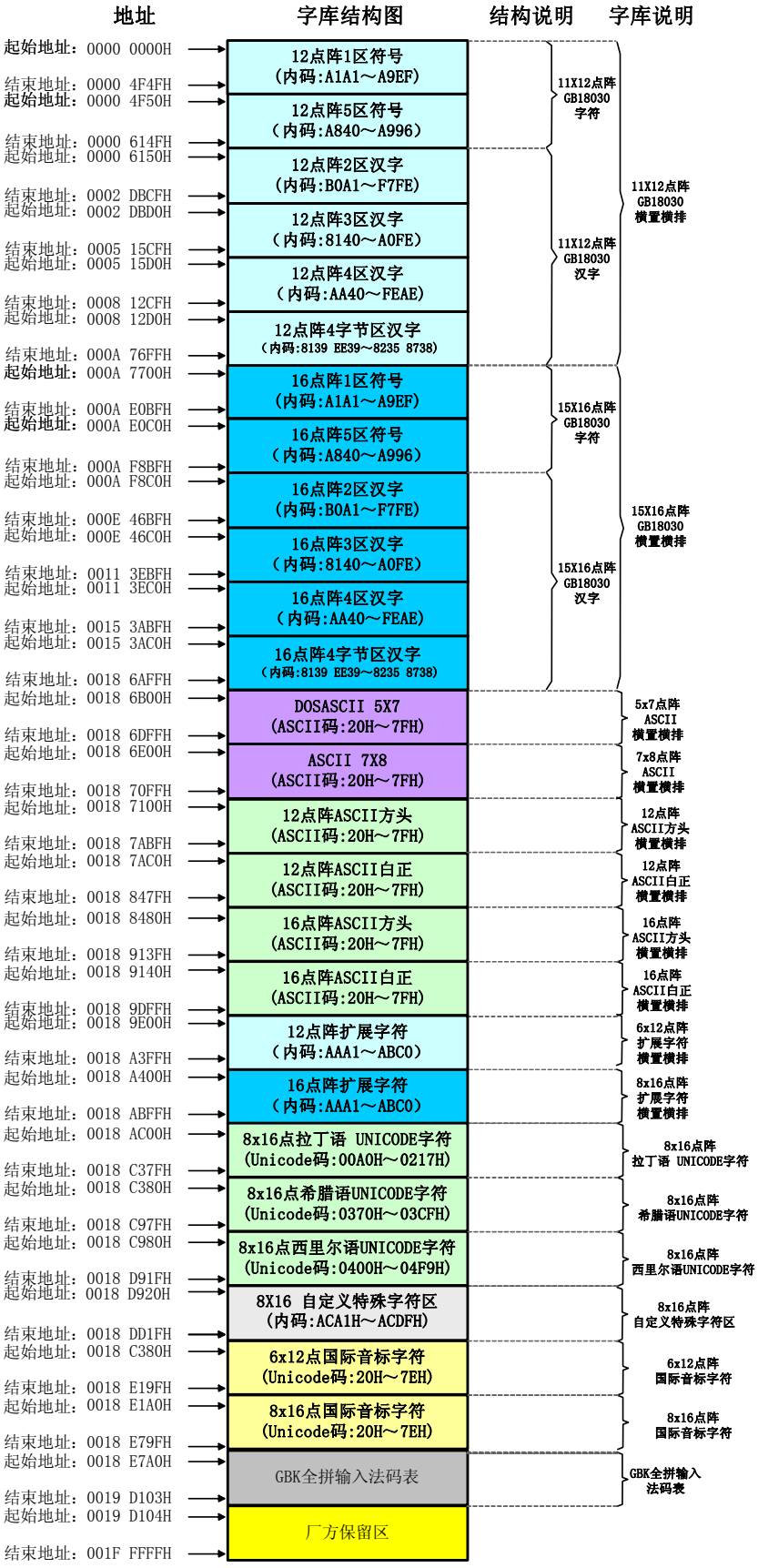
为 ASCII 白正字符 B 的点阵数据。



9.2 汉字点阵字库结构

- GB18030 标准点阵字库
  标准点阵字符 1 区和 5 区分别对应 GB18030 码位的 A1A1~A996 共计 1038 个字符；
  点阵字库中的 2 区、3 区、4 区分别对应 GB18030 双字节码位 B0A1~FEAE 共计 21003 个汉字；
  点阵字库中的四字节区汉字码位范围 8139 EE39~8235 8738 共计 6530 个汉字。
- 1 套 5X7 点 ASCII 字符对应 ASCII 码位的 20~7E 共计 96 个字符。
- 1 套 7X8 点 ASCII 字符对应 ASCII 码位的 20~7E 共计 96 个字符。
- ASCII 方头为不等宽 ASCII，所对应码位 20~7E 共计 96 个字符。
- ASCII 白正为不等宽 ASCII，所对应码位 20~7E 共计 96 个字符。
- 1 套 6X12 点扩展字符对应 GB18030 码位的 AAA1~ABC0 共计 126 个字符。
- 1 套 8X16 点 ASCII 扩展字符集合 对应内码 AAA1~ABC0 共计 126 个字符 (或者对应 ASCII 码位 20~7E 共计 96 个字符,其余忽略) 。
- 8X16 点自定义特殊字符区对应码位的 ACA1~ACDF 共计 64 个字符。
- 8X16 Unicode 字符-拉丁文系:对应拉丁文系 Unicode 码 00A0H~04E9H 共计 376 个字符
- 8X16 Unicode 字符-希腊文系: 对应希腊文系 Unicode 码 0370H~03CFH 共计 96 个字符。
- 8X16 Unicode 字符-西里尔文系:对应西里尔文系 Unicode 码 0400H~04F9H 共计 250 个字符。

GT23L16M2Y3



标准点阵字库数据在汉字芯片中的地址速查表

序号	标准点阵字库数据	在汉字芯片地址
1	11x12 点阵 GB18030	0000 0000H
2	15X16 点阵 GB18030	000A 7700H
3	1 套 5X7 点 ASCII 字符	0018 6B00H
4	1 套 7X8 点 ASCII 字符	0018 6E00H
5	12 点阵 方头（96 字符）	0018 7100H
6	12 点阵 白正（96 字符）	0018 7AC0H
7	16 点阵 方头（96 字符）	0018 8480H
8	16 点阵 白正（96 字符）	0018 9140H
9	6×12 点 扩展字符	0018 9E00H
10	8×16 点 扩展字符	0018 A400H
11	8×16 点 UNICODE 拉丁语字符	0018 AC00H
12	8×16 点 UNICODE 希腊语字符	0018 C380H
13	8×16 点 UNICODE 西里尔语字符	0018 C980H
14	8×16 点自定义字符区	0018 D920H
15	6×12 点 国际音标字符	0018 DD20H
16	8×16 点国际音标字符	0018 E1A0H
17	GBK 拼音输入法码表	0018 E7A0H

9.3 汉字点阵在芯片中的地址计算方法

用户只要知道汉字的内码，就可以计算出该汉字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。

9.3.1 11X12 汉字点阵地址的计算

11X12 点阵字库地址分配(字节地址)：0000 0000 H ~ 000A 76FF H,
地址的计算由下面的函数实现（ANSI C 语言编写）

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(byte address)。如果用户是按 word mode 读取点阵数据，则其地址(word address)为字节地址除以 2，即：word address = byte address / 2 .

例如：BaseAdd: 说明汉字点阵数据在字库芯片中的起始地址，即 BaseAdd＝0x0000;
“啊”字的内码为 0xb0a1,则 byte address = gt(0xb0,0xa1,0x00,0x00) *24+BaseAdd
word address = byte address / 2
“ 上”字的内码为 0x8139ee39,则 byte address = gt(0xee,0x39,0x81,0x39) *24+ BaseAdd
word address = byte address / 2

```
**********************************************************************************************************/
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 <= 0xAB && 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)                    //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;
    }
    if(c2>=0x30 && c2<=0x39)                        //Extended Section (With 4 BYTES InCode)
    {
        if(c4<0x30 || c4>0x39)
            return(h=0);
        h=(c3-0x81)*12600+(c4-0x39)*1260+(c1-0xee)*10+(c2-0x39)+22046;
        if(h>=22046 && h<=22046+6530)
            return(h);
        else
        {   h=0;
            return(h);
        }
    }
    return(h);
}
```

9.3.2 15X16 汉字点阵地址的计算

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(byte address)。如果用户是按 word mode 读取点阵数据，则其地址(word address)为字节地址除以 2，即：word address = byte address / 2 .
例如：BaseAdd: 说明汉字点阵数据在字库芯片中的起始地址，即 BaseAdd＝0xA7700;
      “啊”字的内码为 0xb0a1,则 byte address = gt(0xb0,0xa1,0x00,0x00) *32+BaseAdd
                                word address = byte address / 2
      “ 上”字的内码为 0x8139ee39,则 byte address = gt(0xee,0x39,0x81,0x39) *32+ BaseAdd
                                word address = byte address / 2
*****************************************************************************************************/
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 <= 0xAB && 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)                    //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;
    }
    if(c2>=0x30 && c2<=0x39)                        //Extended Section (With 4 BYTES InCode)
    {
        if(c4<0x30 || c4>0x39)
            return(h=0);
```

```
        h=(c3-0x81)*12600+(c4-0x39)*1260+(c1-0xee)*10+(c2-0x39)+22046;
        if(h>=22046 && h<=22046+6530)
            return(h);
        else
        {   h=0;
            return(h);
        }
    }
    return(h);
}
```

9.3.3 扩展字符地址的计算

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。

(1) 6X12 点扩展字符在芯片中的字节地址的计算方法如下：

```
BaseAdd=0x189E00
if (MSB >= 0x AA && LSB>=0x A1 && LSB<=0x FE) then   //内码为 AAA1~AAFE 的 94 个扩展字符
   Address = (LSB–0x A0 ) * 12+ BaseAdd
if (MSB >= 0x AB && LSB>=0x A1 && LSB<=0x C0) then   //内码为 ABA1~ABC0 的 32 个扩展字符
   Address = (LSB–0x A1 +96) * 12+ BaseAdd
```

(2) 8X16 点扩展字符在芯片中的字节地址的计算方法如下：

```
BaseAdd=0x18A400
if (MSB >= 0x AA && LSB>=0x A1 && LSB<=0x FE) then   //内码为 AAA1~AAFE 的 94 个扩展字符
   Address = (LSB–0x A0 ) * 16+ BaseAdd
if (MSB >= 0x AB && LSB>=0x A1 && LSB<=0x C0) then   //内码为 ABA1~ABC0 的 32 个扩展字符
   Address = (LSB–0x A1 +96) * 16+ BaseAdd
```

若作为标准半角 ASCII （码位 20H~7E 95 个）的调用可以参考以下算法：

```
   LSB 为 ASCII 码
   BaseAdd=0x18A400
   if (LSB>=0x 20 && LSB<=0x 7E) then    //内码为 20H~7E 的 95 个 ASC 字符
        Address = (LSB–0x 20 ) * 16+ BaseAdd
```

9.3.4 ASCII 地址的计算

ASCIICode 表示 ASCII 码（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。

(1) 5X7 ASCII 字符点阵在芯片中的字节地址的计算方法如下：

BaseAdd=0x186B00
if (ASCIICode >= 0x 20) and (ASCIICode <= 0x 7EH) then
　　Address = (ASCIICode –0x 20) * 8+ BaseAdd

(2) 7X8 ASCII 字符点阵在芯片中的字节地址的计算方法如下：

BaseAdd=0x 186E00
if (ASCIICode >= 0x 20) and (ASCIICode <= 0x7E) then
　　Address = (ASCIICode –0x 20) * 8+ BaseAdd

(3) 12 点阵 ASCII 方头字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x187100
if (ASCIICode >= 0x 20) and (ASCIICode <= 0x 7E) then
　　Address = (ASCIICode –0x 20) * 26+ BaseAdd

(4) 12 点阵 ASCII 白正字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x187AC0
if (ASCIICode >=0x20) and (ASCIICode <= 0x7E) then
　　Address = (ASCIICode –0x 20) * 26+ BaseAdd

(5) 16 点阵 ASCII 方头字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x188480
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
　　Address = (ASCIICode –0x 20) * 34+ BaseAdd

(6) 16 点阵 ASCII 白正字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x189140
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
　　Address = (ASCIICode –0x 20) * 34+ BaseAdd

9.3.5 8X16 自定义特殊字符地址的计算

8X16 自定义特殊字符计算方法如下：

GBCODE 表示 GB 内码（16bits）

Address 表示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。

BaseAdd=0x 18D920

if (GBCODE > 0x ACA0) and (GBCODE <=0x ACDF) then

Address = (GBCODE –0x ACA0) * 16+BaseAdd

9.3.6 国际音标字符地址的计算

(1) 6X12 国际音标字符字符计算方法如下：

CODE 表示 国际音标内码（16bits）

Address 表示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。

BaseAdd=0x 18DD20

if (CODE >= 0x20) and (GBCODE <=0x 7E) then

Address = (CODE –0x 20) * 12+BaseAdd

(2) 8X16 国际音标字符字符计算方法如下：

CODE 表示 国际音标内码（16bits）

Address 表示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。

BaseAdd=0x 18E1A0

if (CODE >= 0x20) and (GBCODE <=0x 7E) then

Address = (CODE –0x 20) * 32+BaseAdd

9.3.7 8X16　拉丁语、西里尔语 Unicode 字符在芯片中的字节地址的计算方法

计算方法如下：

unicode表示unicode内码。

Address 表示unicode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。

1.字节地址的计算函数调用方法－拉丁文系

BaseAdd= 0x 18AC00

if (Unicode > = 0x 00A0) and (unicode <=0x 0217) then

Address = (unicode –0x 00A0) * 16+BaseAdd

2.字节地址的计算函数调用方法－希腊文系

BaseAdd= 0x 18C380

if (Unicode > = 0x 0370) and (unicode <=0x 03CF) then

Address = (unicode –0x 0370) * 16+BaseAdd

3.字节地址的计算函数调用方法－西里尔文系

BaseAdd= 0x 18C980

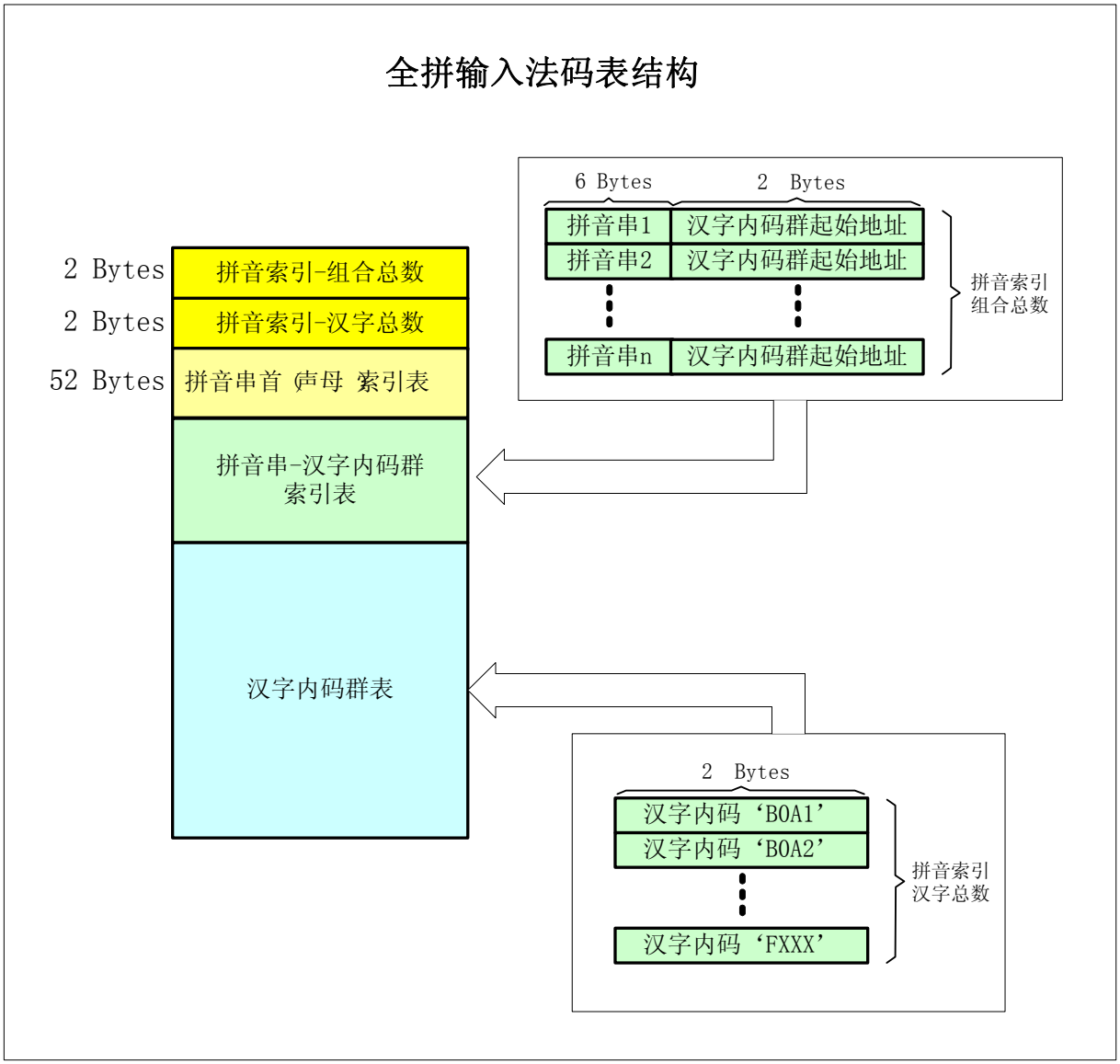
if (Unicode > = 0x 0400) and (unicode <=0x 04F9) then

Address = (unicode –0x 0400) * 16+BaseAdd

9.4 全拼拼音－GBK 全拼输入法（21003 字）

本字库芯片含有 GBK（21003 字）全拼输入法码表，只要将键值串相应的拼音 ASCII 字符进行索引，即可得到该键值串所对应的汉字内码群，产品开发时可以根据用户的需要进行汉字输入的操作。

9.4.1 全拼输入法码表结构



全拼输入法码表结构说明：

1. 拼音索引－组合总数：是全拼输入法码表的所有拼音组合的总数，即声母＋韵母的所有汉字组合。
2. 拼音索引－汉字总数：是本输入法能够支持拼音拼写汉字的总数。
3. 拼音串首（声母）索引表：根据输入拼音串的字首声母来定位该声母在“拼音串－汉字内码群”中该声母的起始地址，主要目的是为了加快查找表速度。
4. 拼音串－汉字内码群：就是根据输入的拼音串进行查找，并找到相应的“汉字内码群“起始地址。
5. 汉字内码群表：该拼音串所对应的汉字内码群。

9.4.2 算法举例

1. 拼音输入法码表的存储格式

为了方便结构的理解结合 8.4.1 图例和以下 ANSI C 语言对结构进行描述。

//存储格式:

short int pycount；//拼音索引－组合总数 2byte

short int hzcount；//拼音索引－汉字总数 2byte

short int pyidx[26]; //拼音串首（声母）索引表，以 26 个英文字母开头的拼音串在拼音表中的起始地址

```
struct {
        char py[6] ;//拼音串
        short int   //汉字码表的起始地址
}
.....................
short int hzcode;       //汉字 1 内码
short int hzcode;       //汉字 2 内码
.....................
```

注: 起始地址已转成绝对地址，不用再计算

2. 拼音输入法的索引

假设拼音串都为小写字母. 输入拼音串 **bang**

1. 用拼音串第一个字母 b-'a'得出拼音串索引下标 1。(检索拼音串首索引表并得到拼音串－汉字内码群地址->idx1)
2. 读出 pyidx[1]的值->变量 idx1。(若 idx1 为 0 说明无此字母开头的拼音串,如 i,u,v 开头，返回)
3. 由 idx1 开始寻址，读出拼音码表,直到找到拼音串 bang。(检索"拼音串－汉字内码群"，并得到汉字内码群表地址。当寻址地址>=pycount*8+56 时无此拼音组合)
   同时读出下一个拼音串。
4. 由找到的拼音串结构,确定汉字码表的地址->idx2，下一个拼音串结构所指汉字码表地址->idx3（下一个拼音串之前的都是该拼音对应的汉字内码区间）
5. 由 idx2 寻址到"汉字内码群表"，查找该拼音串所对应的汉字内码群,同时地址<idx3 的所有汉字均为满足 bang 拼音的汉字 。
6. 当地址>=pycount*8+56+hzcount*2 时，已寻址到码表尾。说明该拼音串不在本输入法码表的范围之内。

10 附录

10.1 GB18030　1 区和 5 区 (字符区)

GB18030 标准点阵字符 1 区和 5 区分别对应码位的 A1A1~A996 共计 1038 个字符；

GB18030　1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	‒	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	｛	｜	｝	￣	

GB18030　1 区

A4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
B	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
C	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
D	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
E	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
F	ゐ	ゑ	を	ん												

A5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
B	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
C	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
D	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
E	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
F	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

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	︿	﹀	｜	⋮	｜	⦚										

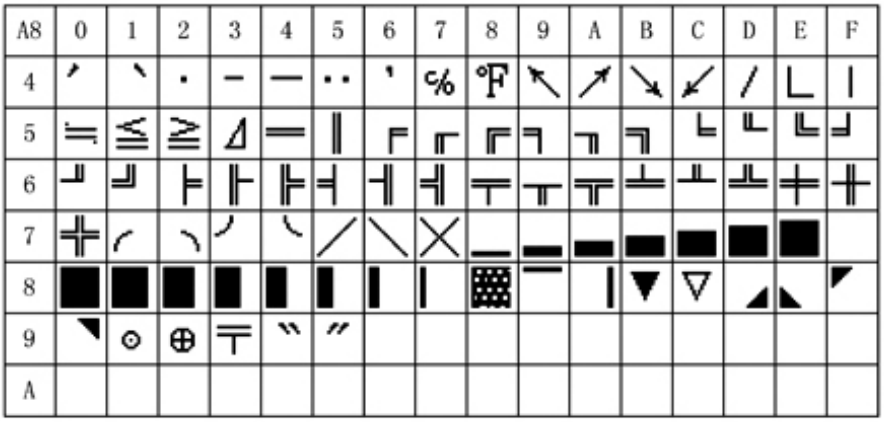
GB18030　1 区

A7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
B	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
C	Ю	Я														
D		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
E	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
F	ю	я														

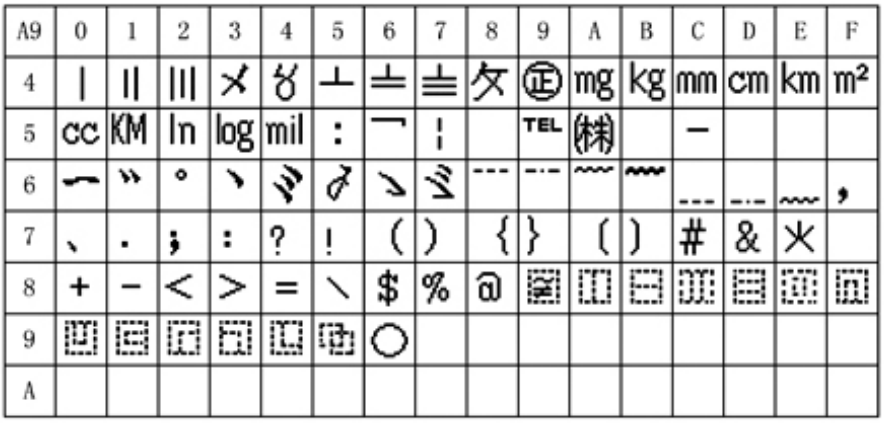
A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǖ	ǘ	ǚ	ǜ	ü	ê	ɑ	ḿ	ń	ň	ǹ
C	ɡ					ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
D	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ
E	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
F																

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	━	|	┃	┄	┅	┆	┇	┈	┉	┊	┋
B	┌	┍	┎	┏	┐	┑	┒	┓	└	┕	┖	┗	┘	┙	┚	┛
C	├	┝	┞	┟	┠	┡	┢	┣	┤	┥	┦	┧	┨	┩	┪	┫
D	┬	┭	┮	┯	┰	┱	┲	┳	┴	┵	┶	┷	┸	┹	┺	┻
E	┼	┽	┾	┿	╀	╁	╂	╃	╄	╅	╆	╇	╈	╉	╊	╋
F																

GB18030　5 区

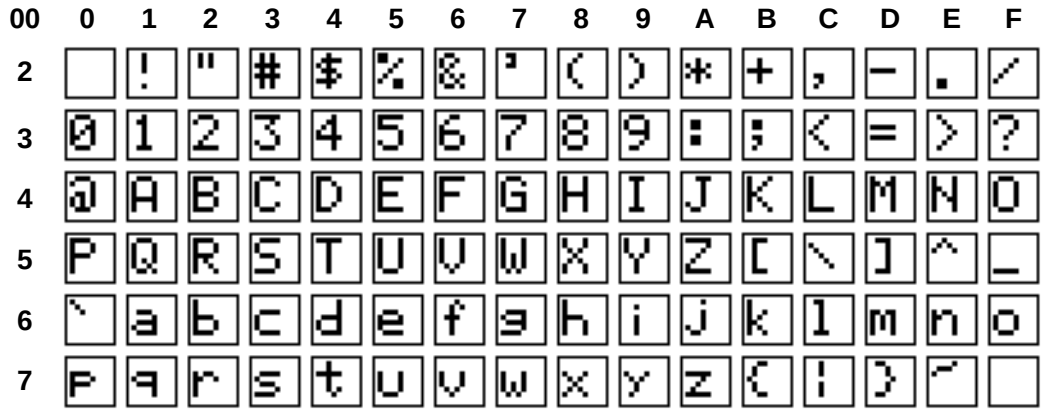


A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	′	ˋ	·	‒	―	‥	'	℅	℉	↖	↗	↘	↙	/	∟	∣
5	≒	≦	≧	⊿	═	‖	╒	╓	╔	╕	╖	╗	╘	╙	╚	╛
6	╜	╝	╞	╟	╠	╡	╢	╣	╤	╥	╦	╧	╨	╩	╪	╫
7	╬	╭	╮	╯	╰	╱	╲	╳	▁	▂	▃	▄	▅	▆	▇	
8	▇	▇	▇	▇	▌	▌	▌	▏	▒	▔	▕	▼	▽	◢	◣	◤
9	◥	⊙	⊕	╤	‶	″										
A																

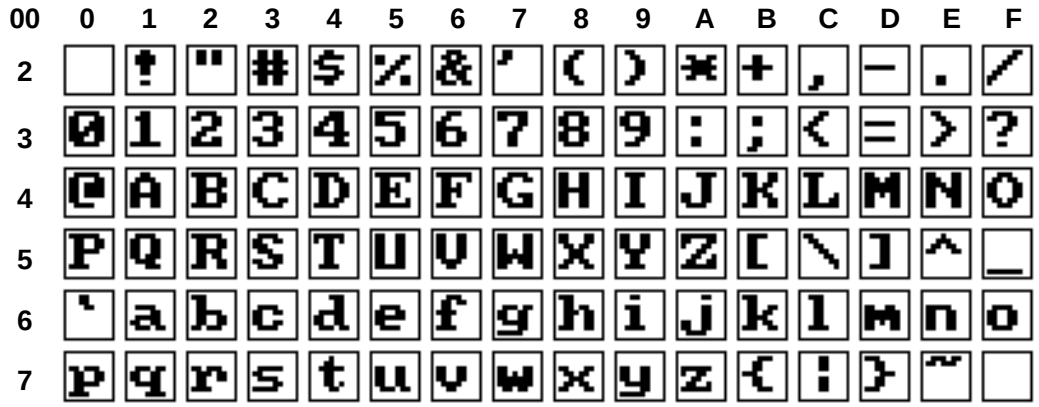


A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	∣	∥	∣∣∣	ㄨ	ㄚ	ㄴ	ㄓ	ㄓ	夂	㊣	mg	kg	mm	cm	km	m²
5	cc	KM	ln	log	mil	:	ㄱ	¦		TEL	㈱		‒			
6	⌒	‶	°	ˋ	ˋ	ˊ	ˋ	ˋ	⋯	⋯	⌒	⌒	⋯	⋯	⌒	，
7	、	.	;	:	?	!	(	)	{	}	〔	〕	#	&	火	
8	+	−	<	>	=	＼	$	%	@	▣	▣	▣	▣	▣	▣	▣
9	▣	▣	▣	▣	▣	▣	○									
A																

10.2 5×7 点 ASCII 字符（96 字符）



10.3 7×8 点 ASCII 字符（96 字符）



10.4 Unicode 字符区字符

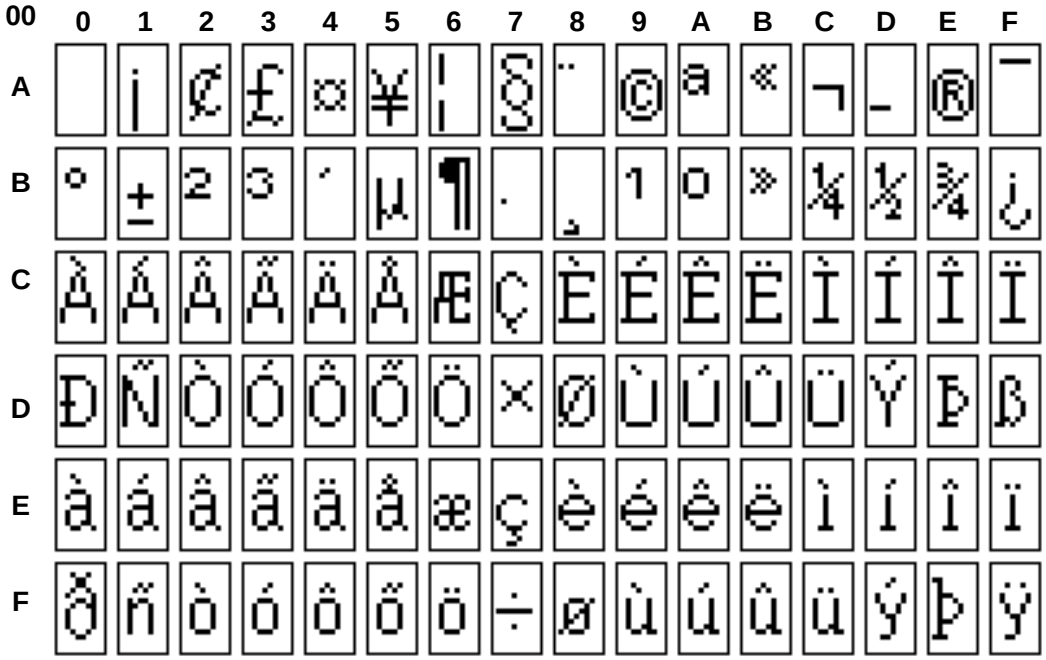
Unicode 字符区共收录　拉丁文系（LATIN）、希腊文系（GREEK）、西里尔文系（CYRILLIC）3 大文系，共计收录字符 456 个

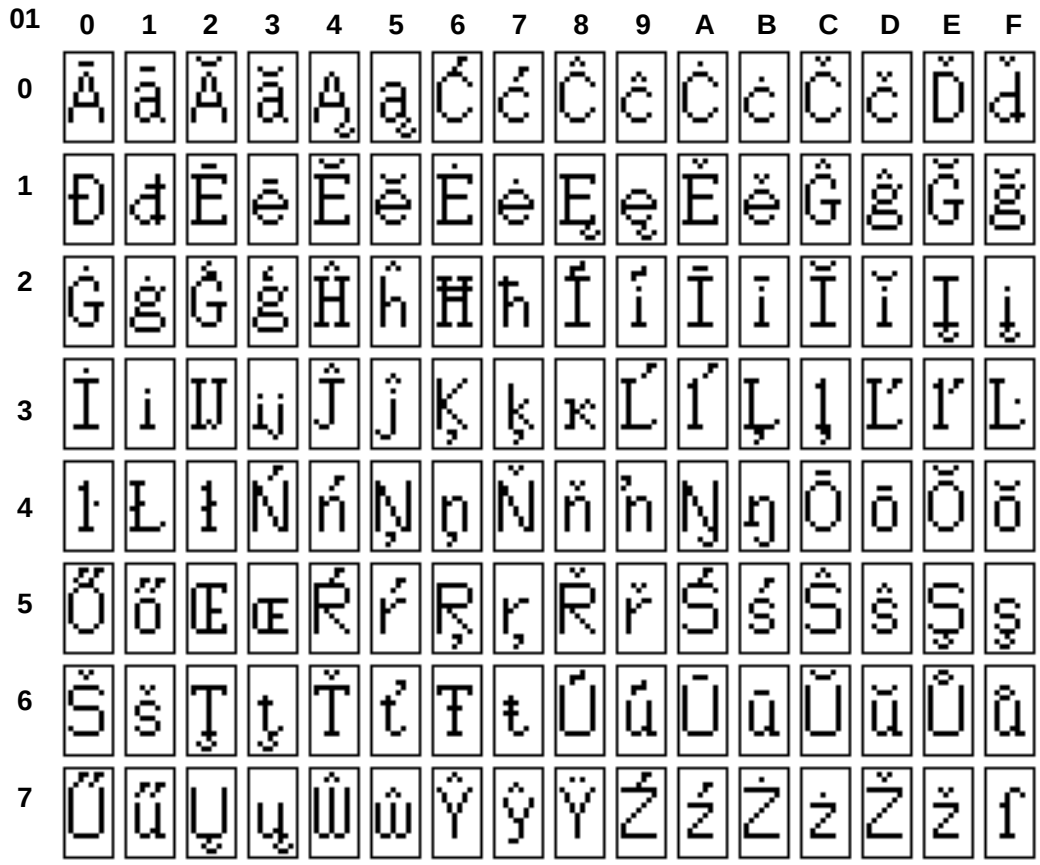
1）拉丁文系（8×16 点 LATIN）

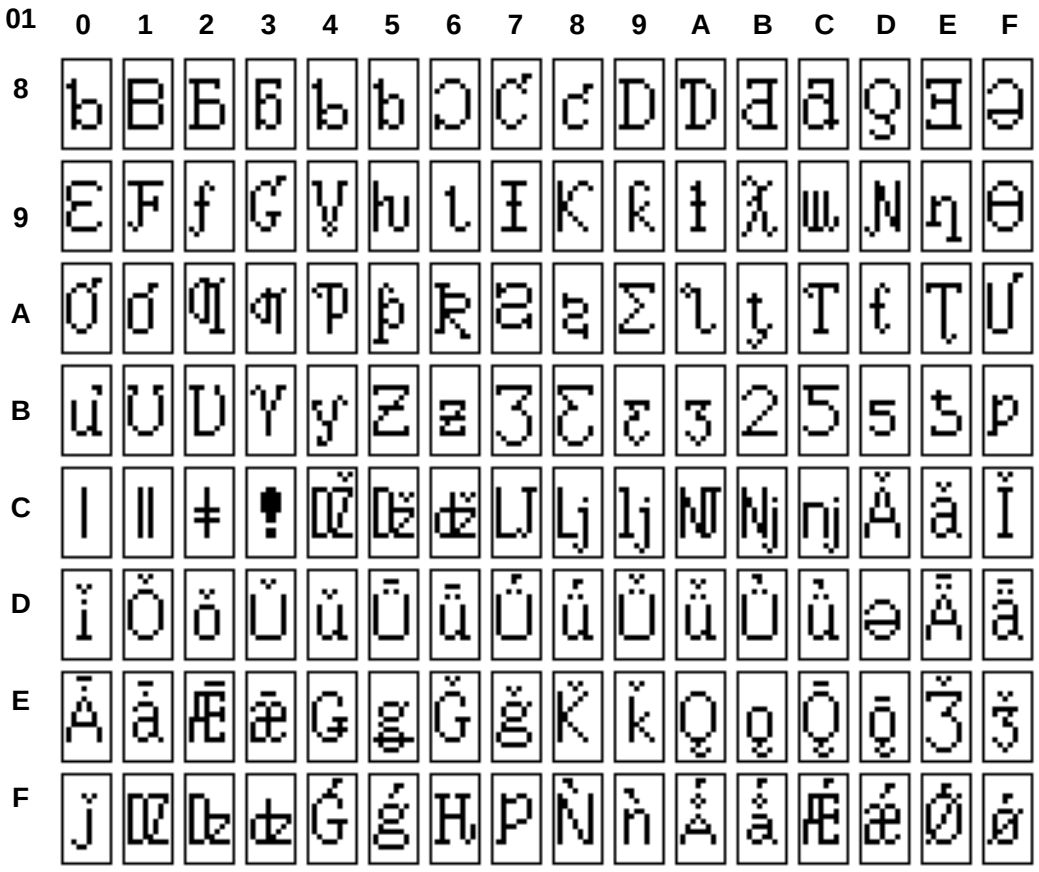
Unicode 字符区-拉丁文系的内码范围为 00A0~0217 共计收录 376 个字符。

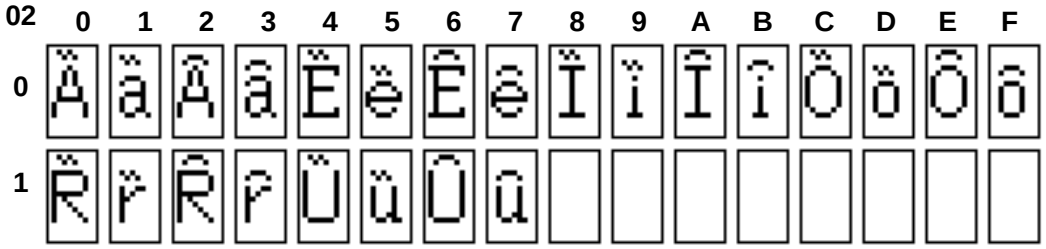
***说明**：由于拉丁文系（BASIC LATIN）的内码范围 0020~007E 与基本 ASCII 编码一致（可以共用），因此未列入 Unicode 字符区-拉丁文系中（见附件：10.2）。

Unicode 字符区-拉丁文系




01
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7


01
0 1 2 3 4 5 6 7 8 9 A B C D E F
8
9
A
B
C
D
E
F

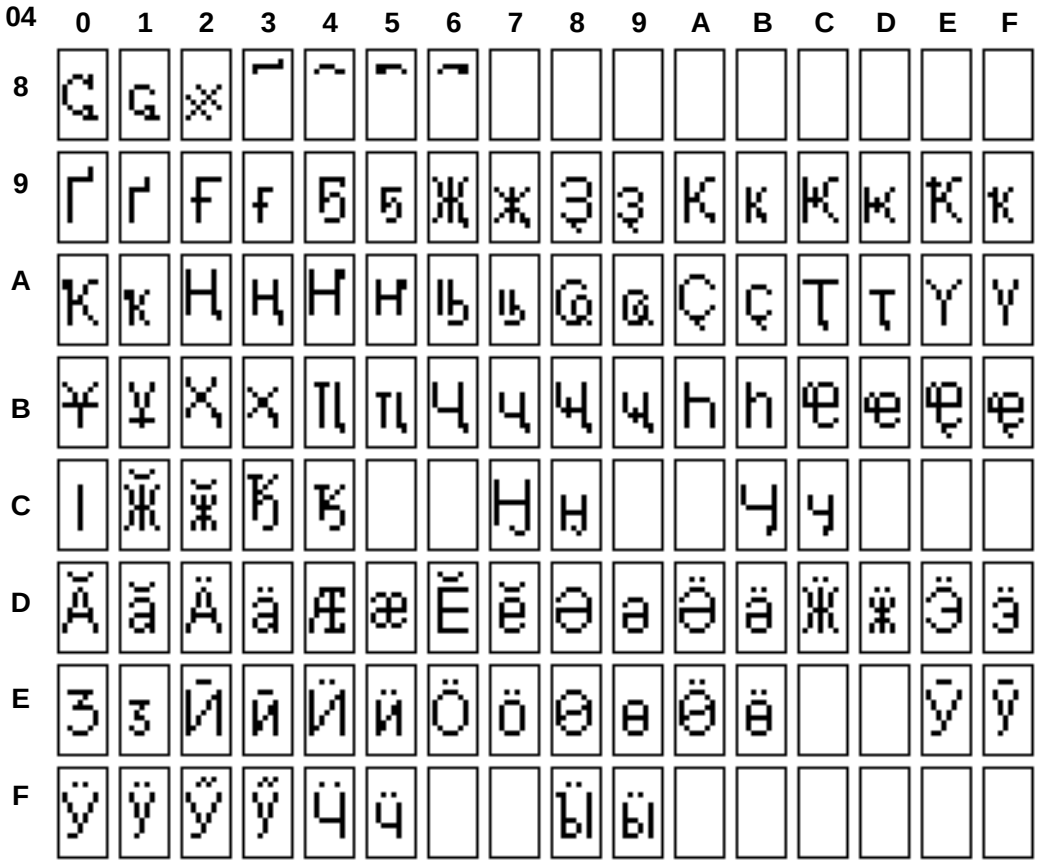

02
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1

2） 西里尔文系（8×16 点 CYRILLIC）

Unicode 字符区-希腊文系的内码范围为 0400~04F9 共计收录 250 个字符。

Unicode 字符区-西里尔文系




04
0 1 2 3 4 5 6 7 8 9 A B C D E F
8
9
A
B
C
D
E
F

2） 希腊文系（8×16 点 BASIC GREEK）

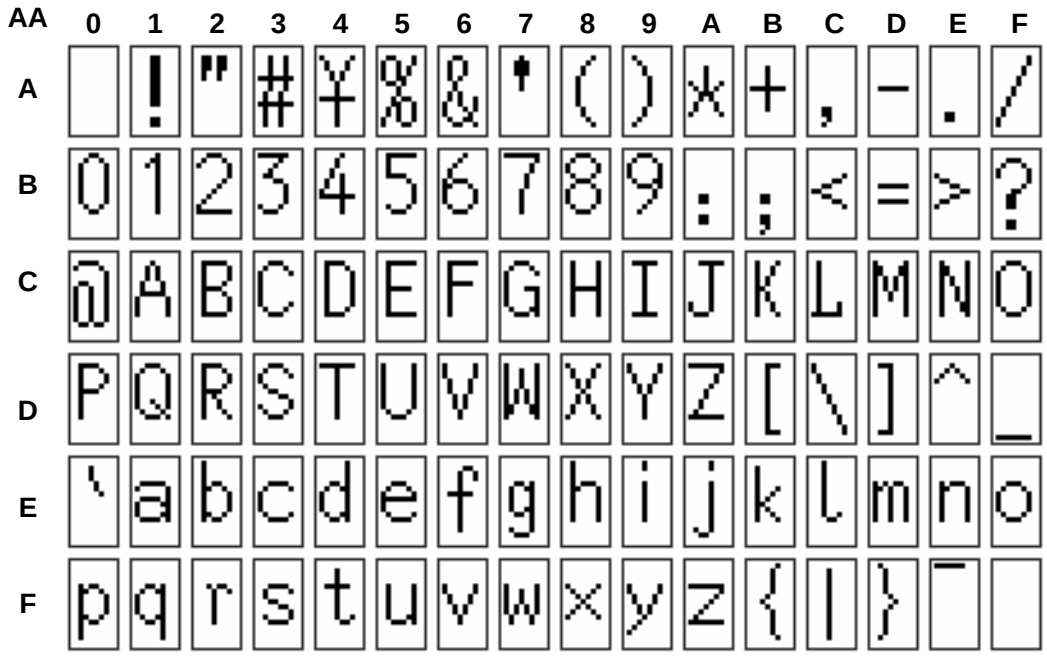
Unicode 字符区-希腊文系的内码范围为 0370~03CF 共计收录 96 个字符。

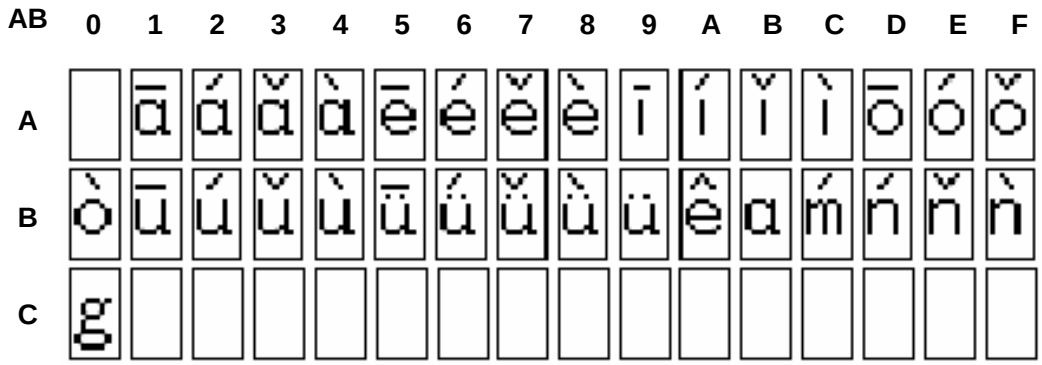
Unicode 字符区-西里尔文系



10.5 8×16 点扩展字符区 ASCII 字符（126 字符）

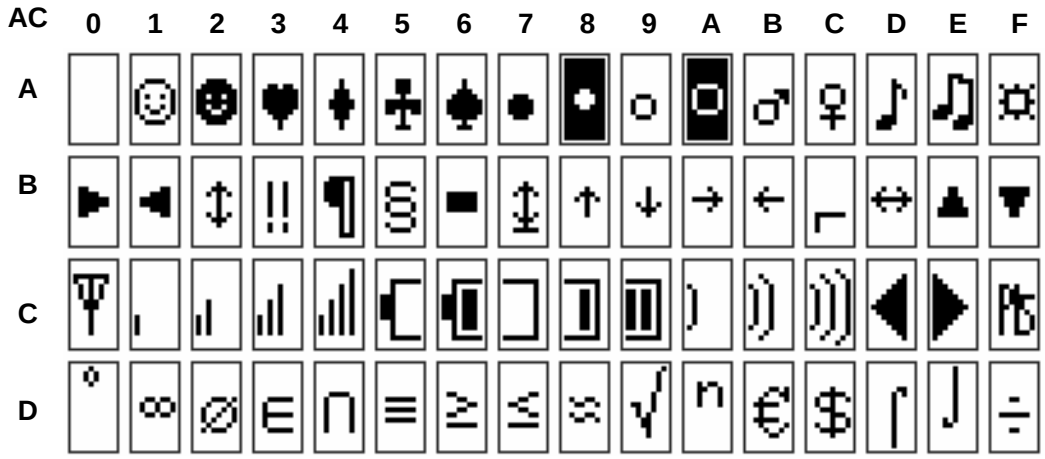
内码组成为 AAA1~ABC0 共计 126 个字符





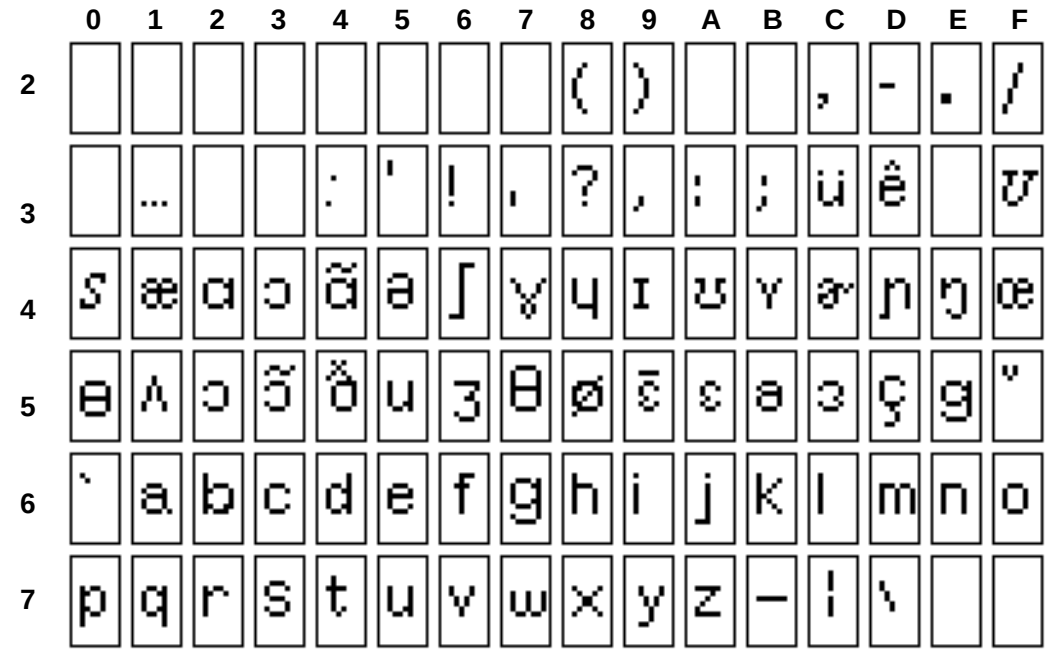
10.6 8×16 自定义特殊字符区 ASCII 字符（64 字符）

内码组成为 AAA1~ABC0 共计 64 个字符



10.7 8×16 自定义国际音标字符区 ASCII 字符（96 字符）

内码组成为 20~7F 共计 96 个字符



10.8 汉字内码－拼音对照表（部分）

由于篇幅限制因此只列举部分对照表，如需要详细《汉字内码－拼音对照表(GBK)》请咨询代理商。

啊	B0A1	a
阿	B0A2	a
阿	B0A2	e
埃	B0A3	ai
挨	B0A4	ai
哎	B0A5	ai
唉	B0A6	ai
哀	B0A7	ai
皑	B0A8	ai
癌	B0A9	ai
蔼	B0AA	ai
矮	B0AB	ai
艾	B0AC	ai
艾	B0AC	yi
碍	B0AD	ai
爱	B0AE	ai
隘	B0AF	ai
鞍	B0B0	an
氨	B0B1	an
安	B0B2	an
俺	B0B3	an
按	B0B4	an
暗	B0B5	an
岸	B0B6	an
胺	B0B7	an
案	B0B8	an
肮	B0B9	ang
昂	B0BA	ang
盎	B0BB	ang
凹	B0BC	ao
凹	B0BC	wa
敖	B0BD	ao
熬	B0BE	ao
翱	B0BF	ao
袄	B0C0	ao
傲	B0C1	ao
奥	B0C2	ao
懊	B0C3	ao
澳	B0C4	ao
芭	B0C5	ba
捌	B0C6	ba
扒	B0C7	ba
扒	B0C7	pa
叭	B0C8	ba
吧	B0C9	ba

笆	B0CA	ba
八	B0CB	ba
疤	B0CC	ba
巴	B0CD	ba
拔	B0CE	ba
跋	B0CF	ba
靶	B0D0	ba
把	B0D1	ba
耙	B0D2	ba
耙	B0D2	pa
坝	B0D3	ba
霸	B0D4	ba
罢	B0D5	ba
爸	B0D6	ba
白	B0D7	bai
柏	B0D8	bai
柏	B0D8	bo
百	B0D9	bai
百	B0D9	bo
摆	B0DA	bai
佰	B0DB	bai
败	B0DC	bai
拜	B0DD	bai
稗	B0DE	bai
斑	B0DF	ban
班	B0E0	ban
搬	B0E1	ban
扳	B0E2	ban
扳	B0E2	pan
般	B0E3	ban
般	B0E3	bo
般	B0E3	pan
颁	B0E4	ban
板	B0E5	ban
版	B0E6	ban
扮	B0E7	ban
拌	B0E8	ban
伴	B0E9	ban
瓣	B0EA	ban
半	B0EB	ban
办	B0EC	ban
绊	B0ED	ban
邦	B0EE	bang
帮	B0EF	bang
梆	B0F0	bang

榜	B0F1	bang
膀	B0F2	bang
膀	B0F2	pang
绑	B0F3	bang
棒	B0F4	bang
磅	B0F5	bang
磅	B0F5	pang
蚌	B0F6	bang
蚌	B0F6	beng
镑	B0F7	bang
傍	B0F8	bang
谤	B0F9	bang
苞	B0FA	bao
胞	B0FB	bao
包	B0FC	bao
褒	B0FD	bao
剥	B0FE	bao
剥	B0FE	bo
薄	B1A1	bao
薄	B1A1	bo
雹	B1A2	bao
保	B1A3	bao
堡	B1A4	bao
堡	B1A4	bu
堡	B1A4	pu
饱	B1A5	bao
宝	B1A6	bao
抱	B1A7	bao
报	B1A8	bao
暴	B1A9	bao
暴	B1A9	pu
豹	B1AA	bao
鲍	B1AB	bao
爆	B1AC	bao
杯	B1AD	bei
碑	B1AE	bei
悲	B1AF	bei
卑	B1B0	bei
北	B1B1	bei
辈	B1B2	bei
背	B1B3	bei
贝	B1B4	bei
钡	B1B5	bei
倍	B1B6	bei
狈	B1B7	bei

备	B1B8	bei
惫	B1B9	bei
焙	B1BA	bei
被	B1BB	bei
奔	B1BC	ben
苯	B1BD	ben
本	B1BE	ben
笨	B1BF	ben
崩	B1C0	beng
绷	B1C1	beng
甭	B1C2	beng
泵	B1C3	beng
蹦	B1C4	beng
迸	B1C5	beng
逼	B1C6	bi
鼻	B1C7	bi
比	B1C8	bi
鄙	B1C9	bi
笔	B1CA	bi
彼	B1CB	bi
碧	B1CC	bi
蓖	B1CD	bi
蔽	B1CE	bi
毕	B1CF	bi
毙	B1D0	bi
毖	B1D1	bi
币	B1D2	bi
庇	B1D3	bi
痹	B1D4	bi
闭	B1D5	bi
敝	B1D6	bi
弊	B1D7	bi
必	B1D8	bi
辟	B1D9	bi
辟	B1D9	pi
壁	B1DA	bi
臂	B1DB	bei
臂	B1DB	bi
避	B1DC	bi
陛	B1DD	bi
鞭	B1DE	bian
边	B1DF	bian
编	B1E0	bian
贬	B1E1	bian
扁	B1E2	bian

扁	B1E2	pian
便	B1E3	bian
便	B1E3	pian
变	B1E4	bian
卞	B1E5	bian
辨	B1E6	bian
辩	B1E7	bian
辫	B1E8	bian
遍	B1E9	bian
标	B1EA	biao
彪	B1EB	biao
膘	B1EC	biao
表	B1ED	biao
鳖	B1EE	bie
憋	B1EF	bie
别	B1F0	bie
瘪	B1F1	bie
彬	B1F2	bin
斌	B1F3	bin
濒	B1F4	bin
滨	B1F5	bin
宾	B1F6	bin
摈	B1F7	bin
兵	B1F8	bing
冰	B1F9	bing
柄	B1FA	bing
丙	B1FB	bing
秉	B1FC	bing
饼	B1FD	bing
炳	B1FE	bing
病	B2A1	bing
并	B2A2	bing
玻	B2A3	bo
菠	B2A4	bo
播	B2A5	bo
拨	B2A6	bo
钵	B2A7	bo
波	B2A8	bo
博	B2A9	bo
勃	B2AA	bo
搏	B2AB	bo
铂	B2AC	bo
箔	B2AD	bo
伯	B2AE	bai
帛	B2AE	bo

帛	B2AF	bo
舶	B2B0	bo
脖	B2B1	bo
膊	B2B2	bo
渤	B2B3	bo
泊	B2B4	bo
泊	B2B4	po
驳	B2B5	bo
捕	B2B6	bu
卜	B2B7	bo
卜	B2B7	bu
哺	B2B8	bu
补	B2B9	bu
埠	B2BA	bu
不	B2BB	bu
布	B2BC	bu
步	B2BD	bu
簿	B2BE	bu
部	B2BF	bu
怖	B2C0	bu
擦	B2C1	ca
猜	B2C2	cai
裁	B2C3	cai
材	B2C4	cai
才	B2C5	cai
财	B2C6	cai
睬	B2C7	cai
踩	B2C8	cai
采	B2C9	cai
彩	B2CA	cai
菜	B2CB	cai
蔡	B2CC	cai
餐	B2CD	can
参	B2CE	can
参	B2CE	cen
参	B2CE	shen
蚕	B2CF	can
残	B2D0	can
惭	B2D1	can
惨	B2D2	can
灿	B2D3	can
苍	B2D4	cang
舱	B2D5	cang
仓	B2D6	cang
沧	B2D7	cang

藏	B2D8	cang
藏	B2D8	zang
操	B2D9	cao
糙	B2DA	cao
槽	B2DB	cao
曹	B2DC	cao
草	B2DD	cao
厕	B2DE	ce
厕	B2DE	si
策	B2DF	ce
侧	B2E0	ce
侧	B2E0	ze
侧	B2E0	zhai
册	B2E1	ce
测	B2E2	ce
层	B2E3	ceng
蹭	B2E4	ceng
插	B2E5	cha
叉	B2E6	cha
茬	B2E7	cha
茶	B2E8	cha
查	B2E9	cha
查	B2E9	zha
碴	B2EA	cha
搽	B2EB	cha
察	B2EC	cha
岔	B2ED	cha
差	B2EE	cha
差	B2EE	chai
差	B2EE	ci
诧	B2EF	cha
拆	B2F0	ca
拆	B2F0	chai
柴	B2F1	chai
豺	B2F2	chai
搀	B2F3	chan
掺	B2F4	can
掺	B2F4	chan
掺	B2F4	shan
蝉	B2F5	chan
馋	B2F6	chan
谗	B2F7	chan
缠	B2F8	chan
铲	B2F9	chan
产	B2FA	chan

闸	B2FB	chan
颤	B2FC	chan
颤	B2FC	zhan
昌	B2FD	chang
猖	B2FE	chang
场	B3A1	chang
尝	B3A2	chang
常	B3A3	chang
长	B3A4	chang
长	B3A4	zhang
偿	B3A5	chang
肠	B3A6	chang
厂	B3A7	an
厂	B3A7	chang
敞	B3A8	chang
畅	B3A9	chang
唱	B3AA	chang
倡	B3AB	chang
超	B3AC	chao
抄	B3AD	chao
钞	B3AE	chao
朝	B3AF	chao
朝	B3AF	zhao
嘲	B3B0	chao
嘲	B3B0	zhao
潮	B3B1	chao
巢	B3B2	chao
吵	B3B3	chao
炒	B3B4	chao
车	B3B5	che
车	B3B5	ju
扯	B3B6	che
撤	B3B7	che
掣	B3B8	che
彻	B3B9	che
澈	B3BA	che
郴	B3BB	chen
臣	B3BC	chen
辰	B3BD	chen
尘	B3BE	chen
晨	B3BF	chen
忱	B3C0	chen
沉	B3C1	chen
陈	B3C2	chen
趁	B3C3	chen