GT23L24B3Y0-S

标准汉字字库芯片

11X12 点阵

15X16 点阵

24X24 点阵

（BIG5 大五码标准点阵字）

竖置横排

（SPI 接口/PLII 接口）

用户手册

VER 2.5

2007-Q2

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	

目　　录

1 概述 4
2 特点 5
3 引脚配置 8
4 引脚描述 8
4.1 SPI 接口引脚描述 9
4.2 PLII 接口引脚描述 9
5 SPI 接口/PLII 接口与主机接口电路示意图 10
5.1 SPI 接口与主机接口电路示意图 10
5.2 PLII 接口与主机接口电路示意图 11
6 操作指令 12
6.1 SPI 接口模式下操作 12
6.2 PLII 接口模式下操作 14
7 电气特性 16
7.1 绝对最大额定值 16
7.2 DC 特性 16
7.3 AC 特性 16
8 封装尺寸：SOP20 20
9 功能描述 21
9.1 汉字点阵排列格式 21
9.2 汉字点阵字库结构 35
9.3 汉字点阵在芯片中的地址计算方法 37
10 附录 44
10.1 BIG5 (字符区) 44
10.2 扩展字符区字符 46
10.3 5×7 点 ASCII 字符（96 字符） 47
10.4 7×8 点 ASCII 字符（96 字符） 47
10.5 Unicode 字符区字符 48
10.6 8×16 点扩展字符区 ASCII 字符（126 字符） 54
10.7 8×16 自定义特殊字符区 ASCII 字符（64 字符） 55
10.8 8×16 自定义国际音标字符区 ASCII 字符（96 字符） 55

1 概述

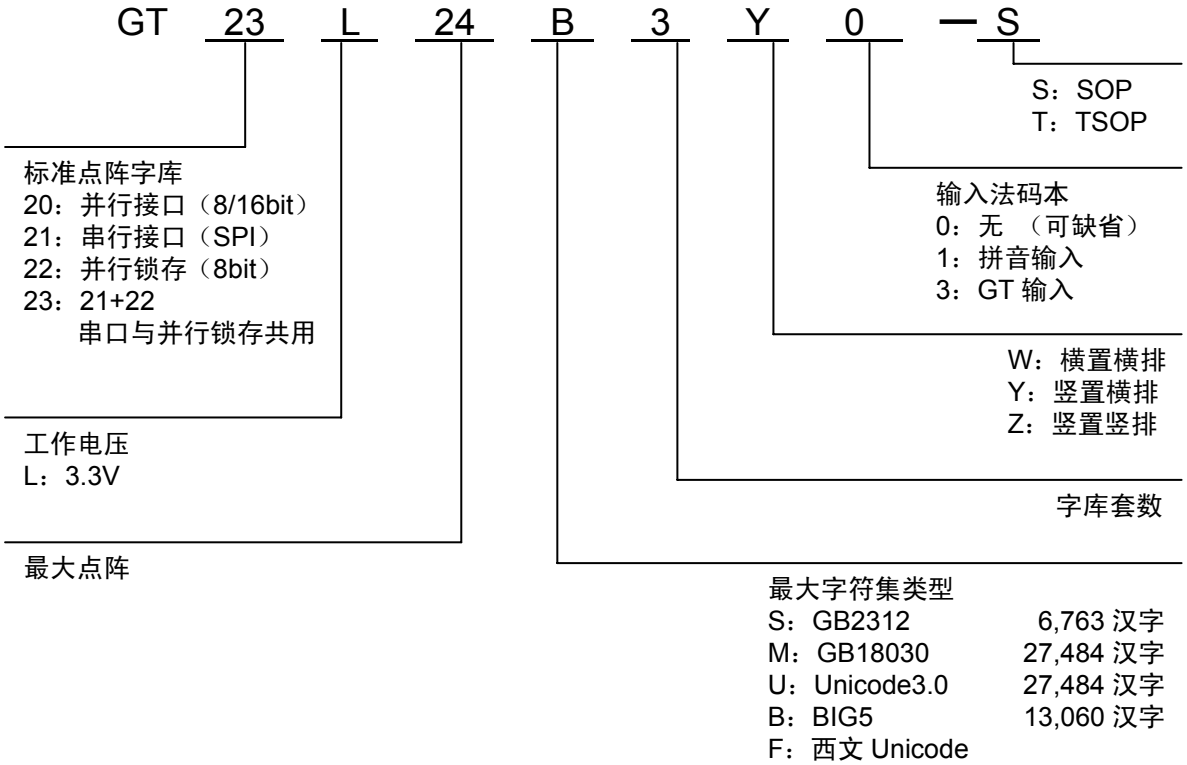
GT23L24B3Y0是一款支持BIG5字符集（包括13060汉字和442字符）11 X 12、15 X 16、24 X 24、点阵字库芯片，另外还含有2套12点高的半角不等宽ASCII字符、2套16点高的半角不等宽ASCII字符、2套24点高的半角不等宽ASCII字符、1套6X12点扩展字符、1套8X16点扩展字符、1套12X24点扩展字符。同时内含1套5X7点ASCII字符、1套7X8点ASCII字符、Unicode字符区拉丁文系（LATIN）、希腊文系（LATIN）、西里尔文系（CYRILLIC）文系字符，基本可以涵盖81个国家及地区。同时还有非常丰富的自定义区的各种图形符号（，以及12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。

GT23L24B3Y0 支持 2 种不同的总线接口：即 SPI 接口、PLII 接口。

命名规则（标准点阵字库芯片）



2 特点

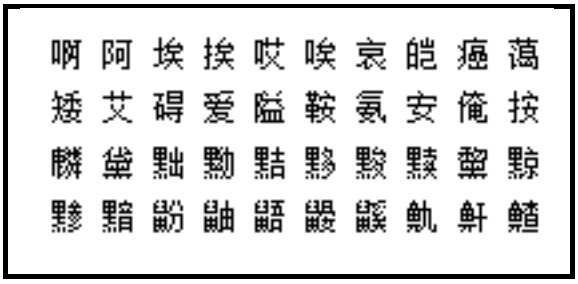
- 数据总线：SPI 总线兼容的串行接口/PLII 接口
- 点阵排列方式：字节竖置横排
- 访问速度：SPI 时钟频率：40MHz(max.)
  PLII 访问速度：70ns(max.) @3.3V
- 工作电压：2.7V – 3.6V
- 电流：
  工作电流：30mA(max.) @3.3V
  待机电流：15uA(max.) @3.3V
- 封装： SOP20
- 字库内容：

汉 字 字 库	
11×12 点	BIG5 （13,060 汉字） （442 字符）
15×16 点	BIG5 （13,060 汉字） （442 字符）
24×24 点	BIG5 （13,060 汉字） （442 字符）

ASCII 字符	
6×12 点	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）
8×16 点	扩展字符（126 字符）
5×7 点	ASCII 字符（96 字符）
7×8 点	ASCII 字符（96 字符）
16 点阵	ASCII 方头（96 字符）
16 点阵	ASCII 白正（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）
12×24 点	扩展字符（126 字符）
24 点阵	ASCII 字符（96 字符）
24 点阵	ASCII 白正（96 字符）

- 字型样张

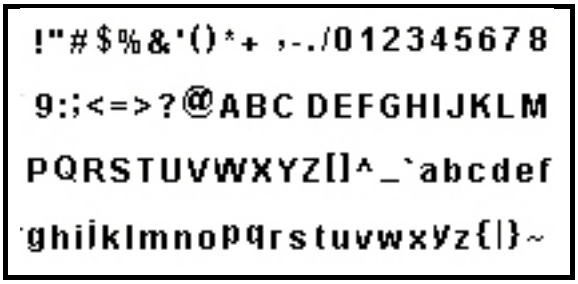
11×12 点 BIG5（13,060 汉字）



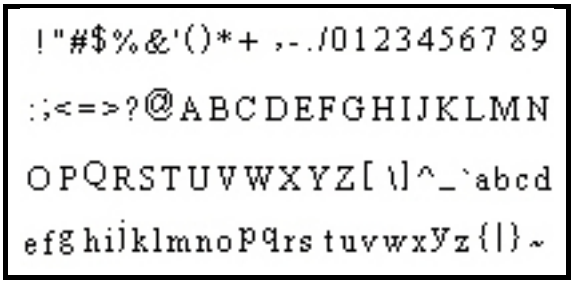
6×12 点扩展字符（126 字符）



12 点阵 ASCII 方头（96 字符）



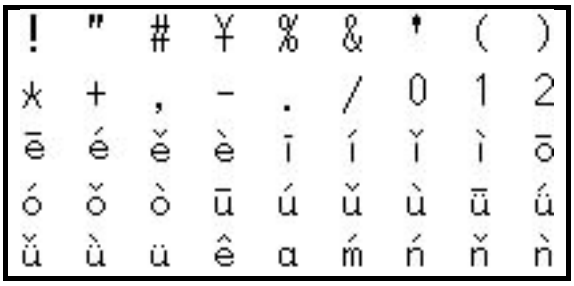
12 点阵 ASCII 白正（96 字符）



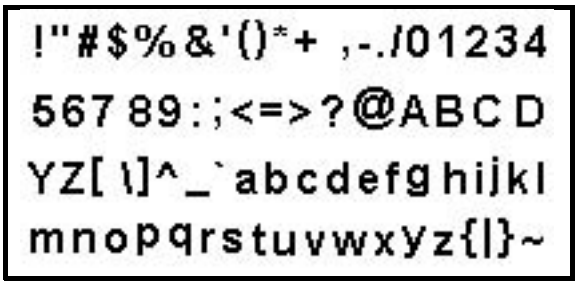
15×16 点 BIG5（13,060 汉字）



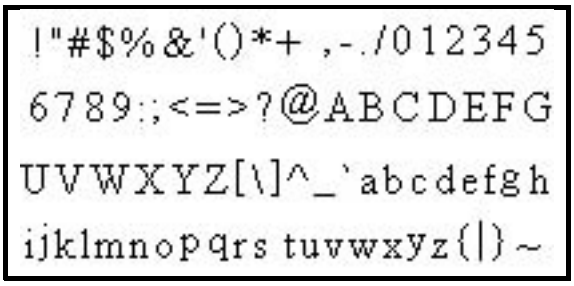
8×16 点扩展字符（126 字符）



16 点阵 ASCII 方头（96 字符）



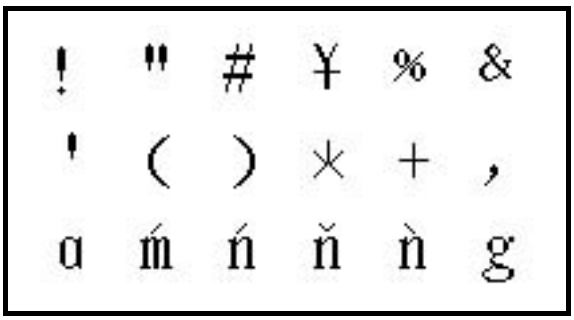
16 点阵 ASCII 白正（96 字符）



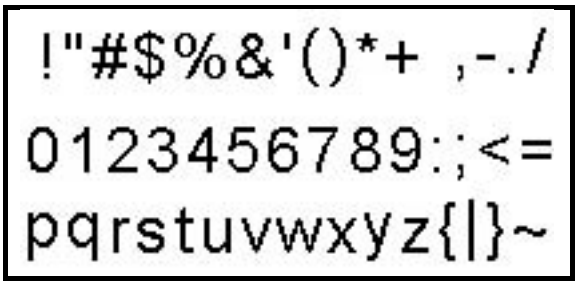
24×24 点 宋体 BIG5（13,060 汉字）



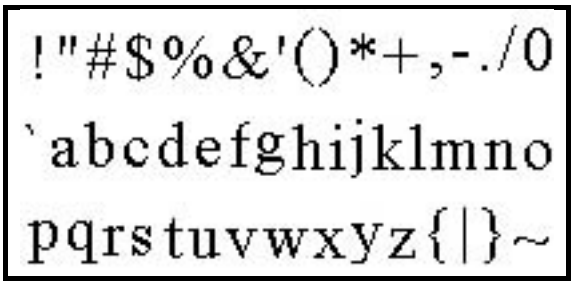
12×24 点扩展字符（126 字符）



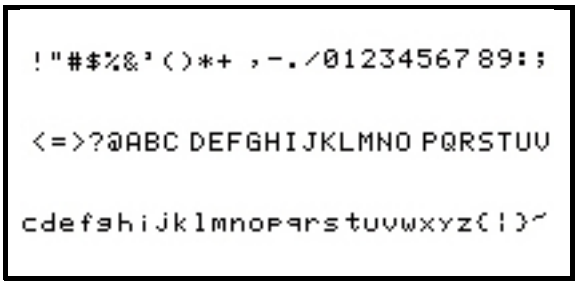
24 点阵 ASCII 方头（96 字符）



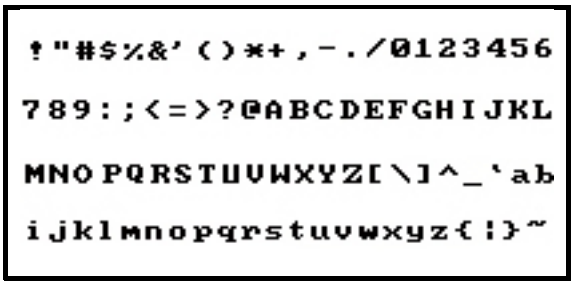
24 点阵 ASCII 白正（96 字符）



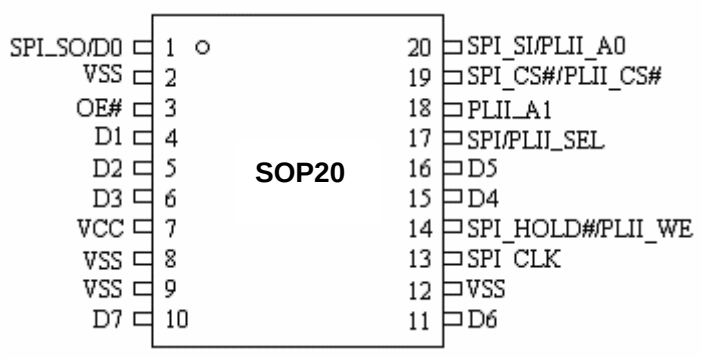
5×7 点 ASCII 字符（96 字符）



7×8 点 ASCII 字符（96 字符）



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

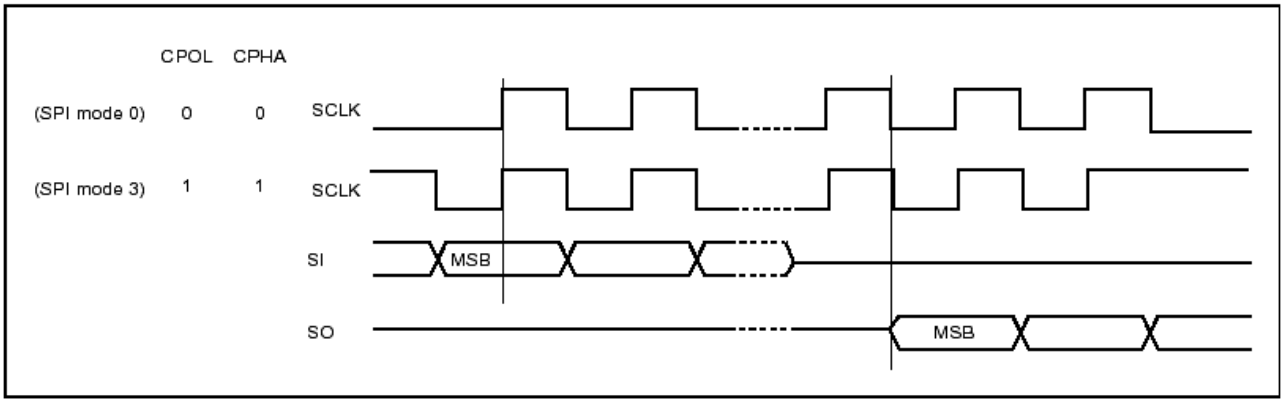
4.1 SPI 接口引脚描述

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。

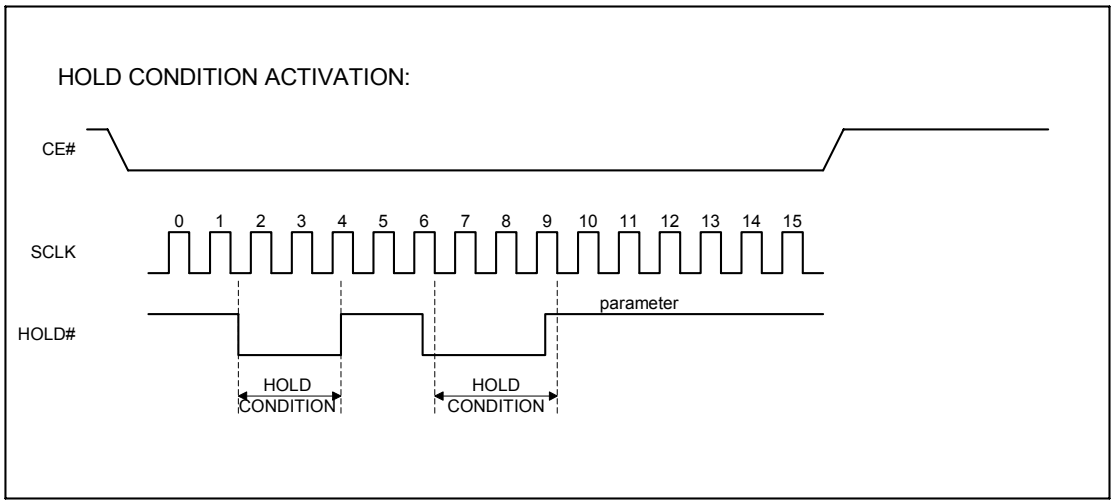


总线挂起输入（HOLD#）：

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。



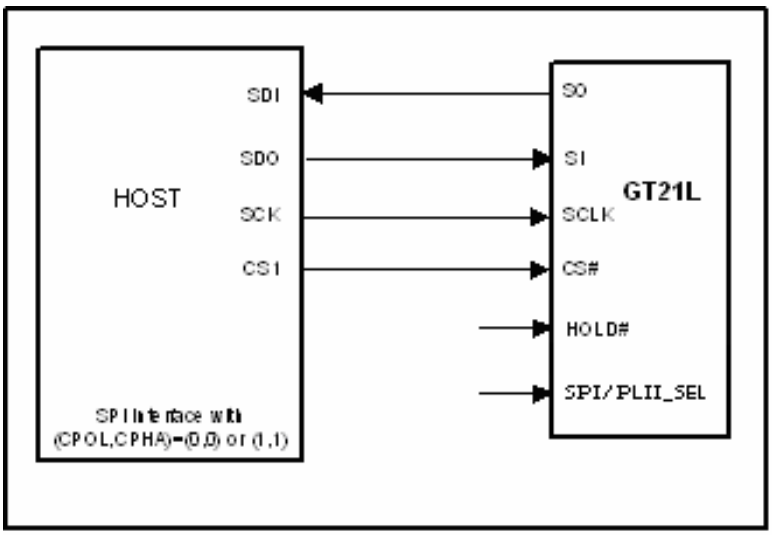
4.2 PLII 接口引脚描述

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

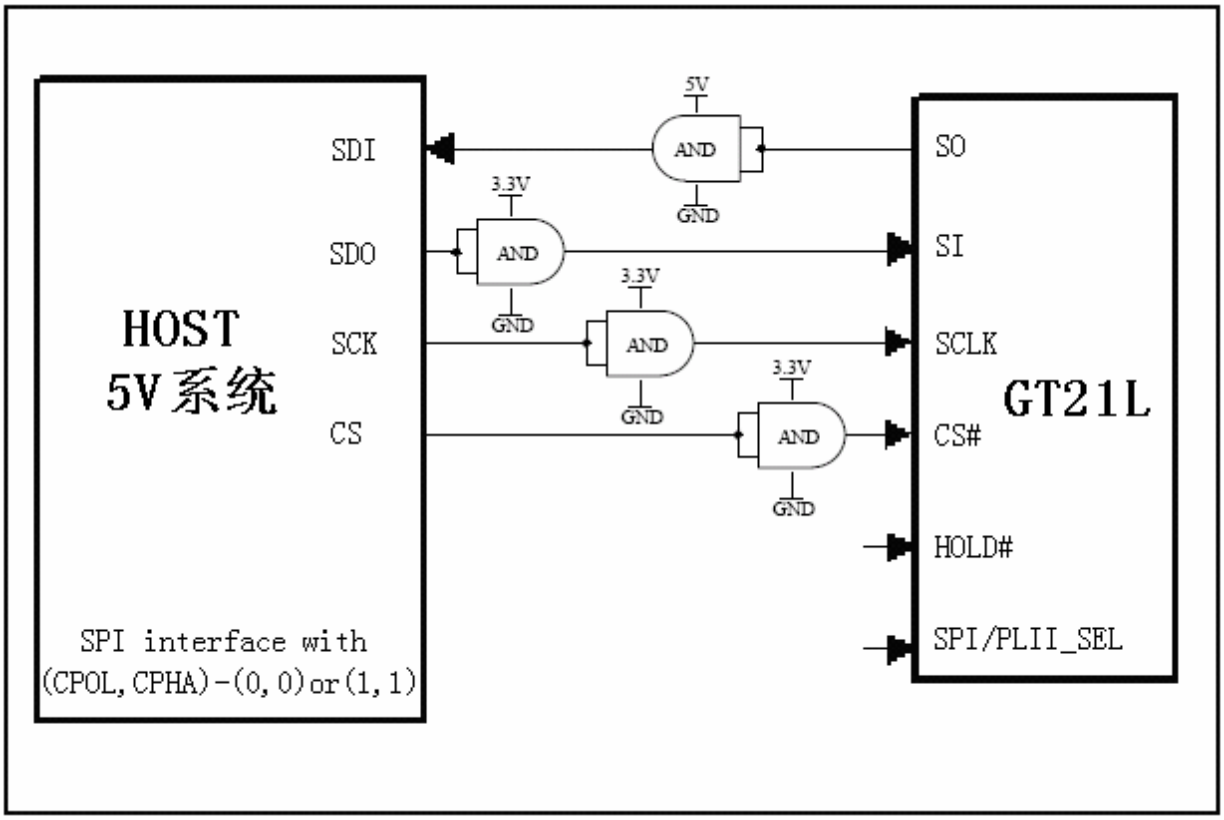
5 SPI 接口/PLII 接口与主机接口电路示意图

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。

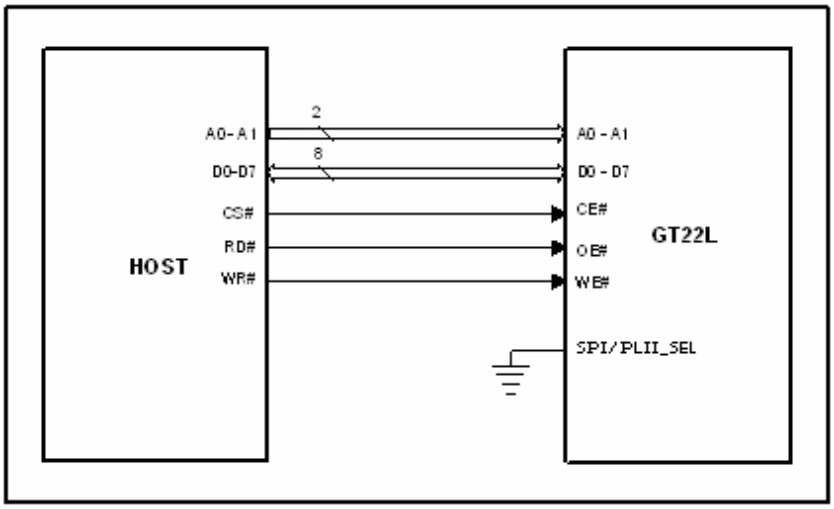


注意：若是采用系统电压为 5V 的，则需要进行电平转换匹配连接 GT21L 库芯片，可以参考下图。

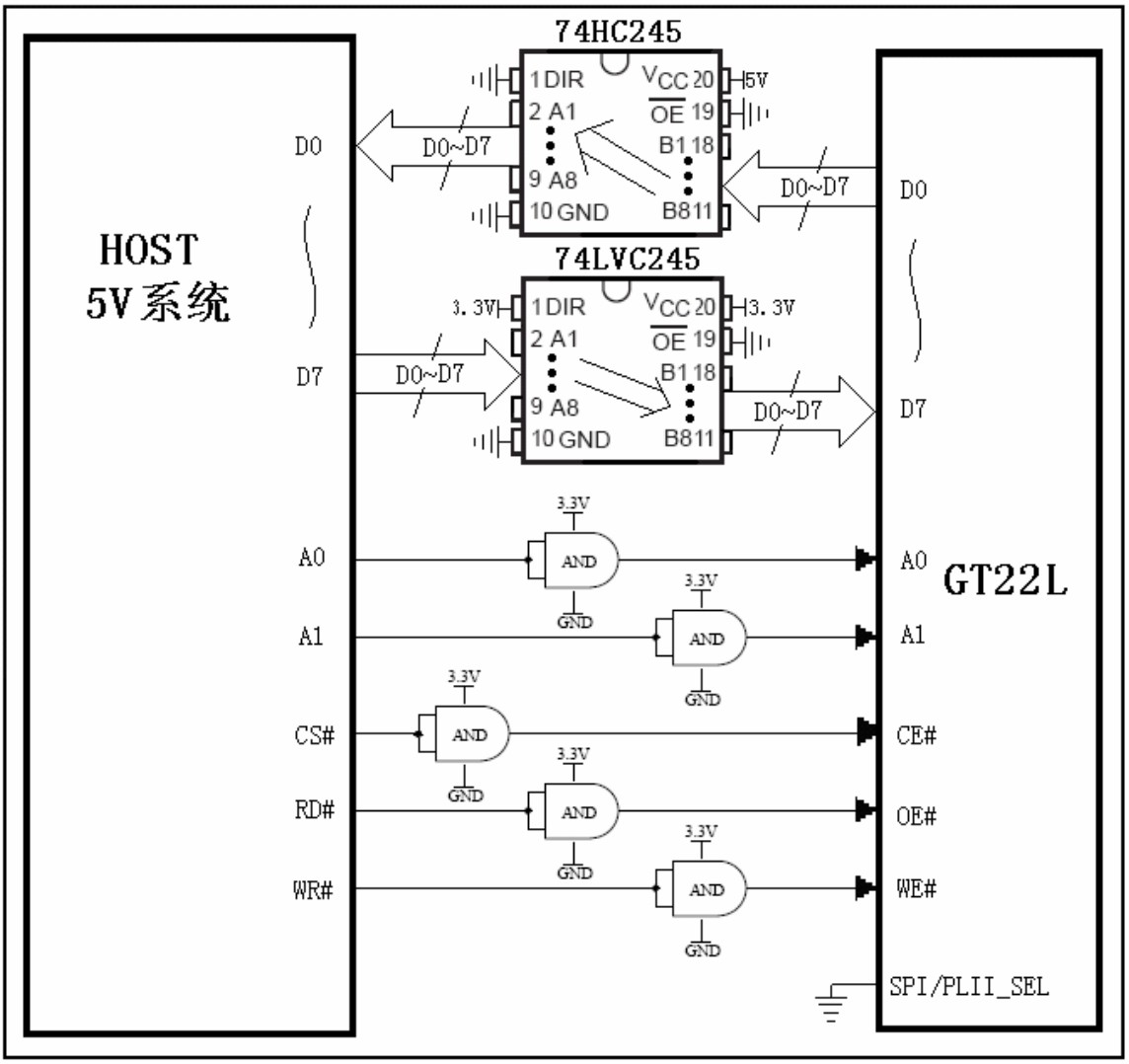


5.2 PLII 接口与主机接口电路示意图

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”)。.

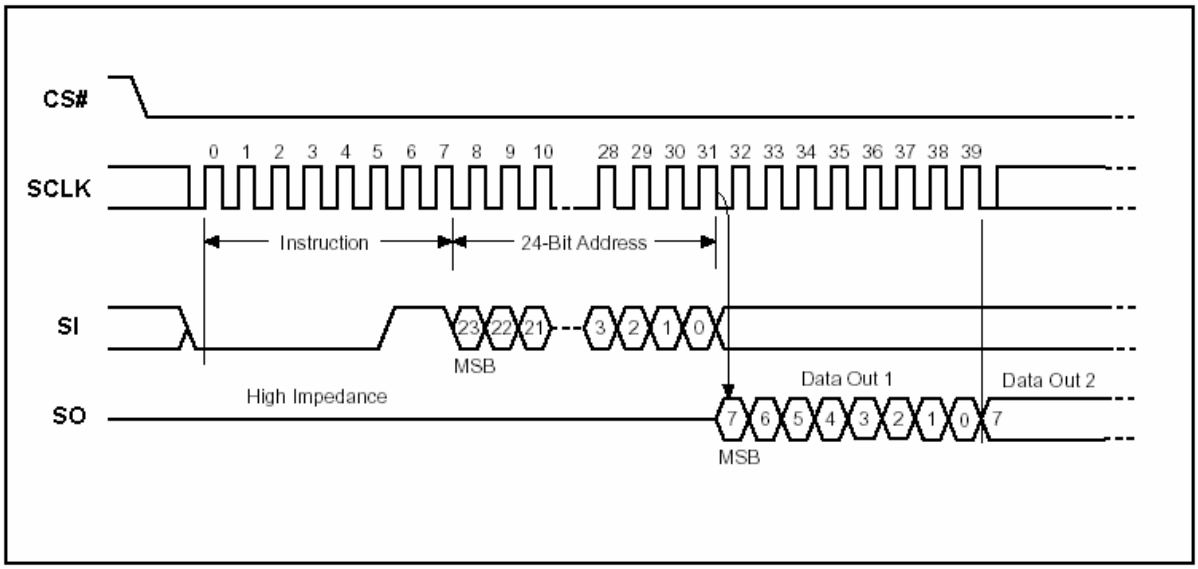
6.1.2 Read Data Bytes（一般读取）

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



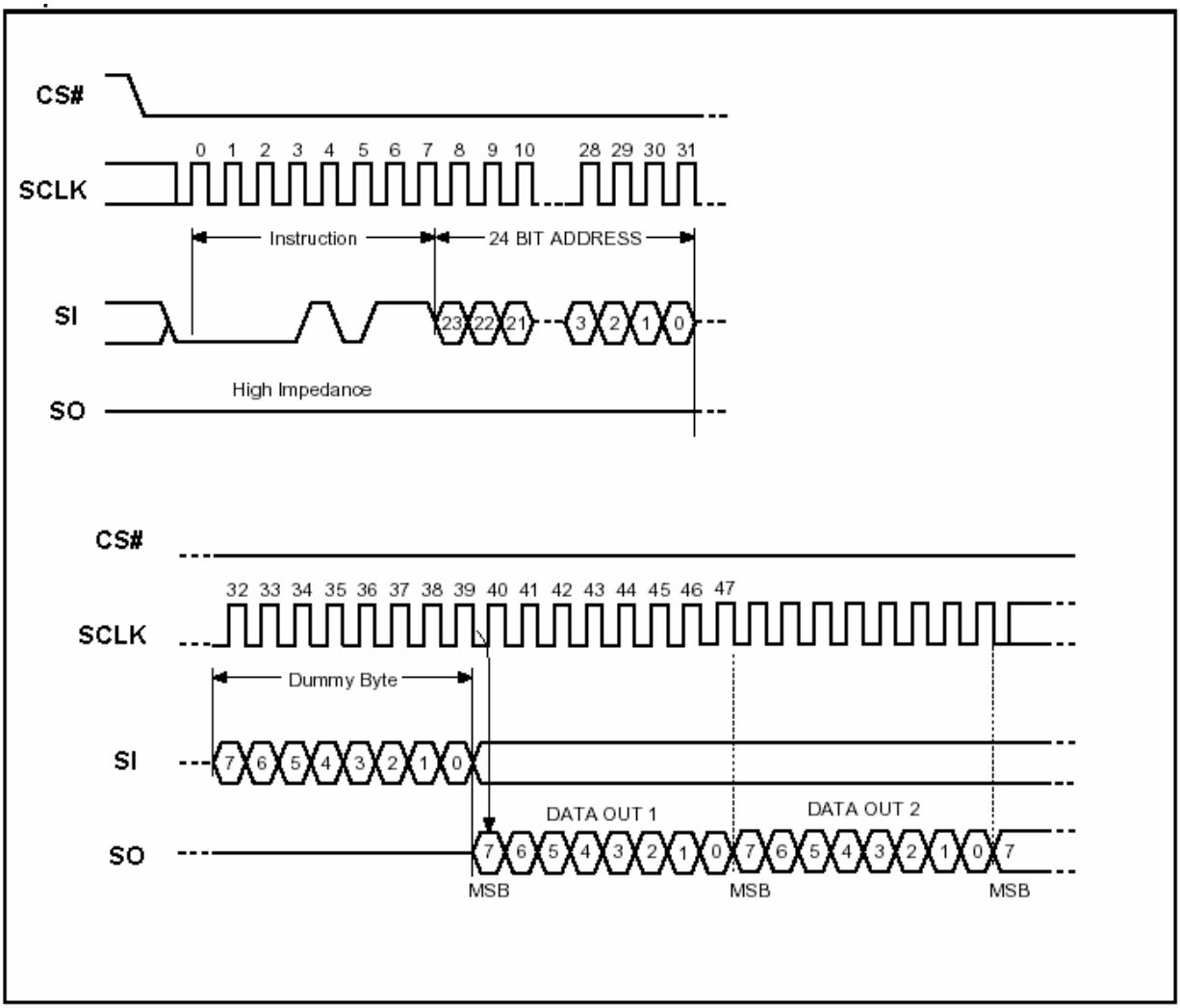
6.1.3 Read Data Bytes at Higher Speed（快速读取点阵数据）

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

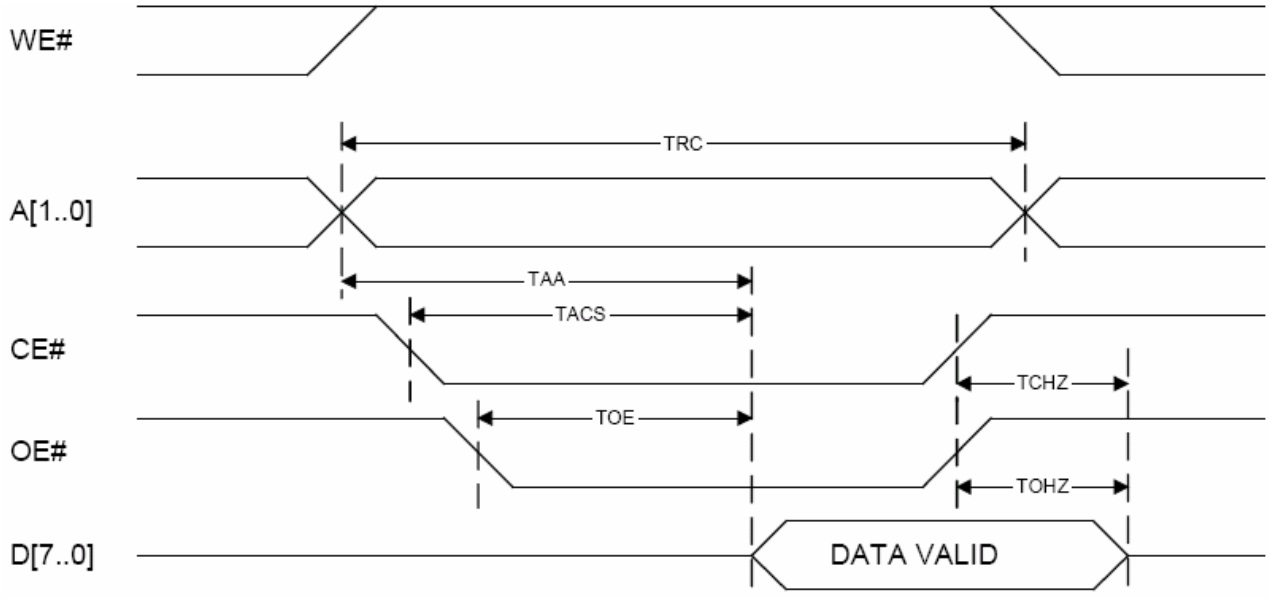
Pin name	I/O	描述
A[1..0]	I	地址寄存器寻址
D[7..0]	I/O	地址输入/数据输出
CE#	I	片选信号输入，低有效
OE#	I	“输出使能”信号输入，OE# 为低时输出使能
WE#	I	“写使能”信号输入，WE# 为低时写使能

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

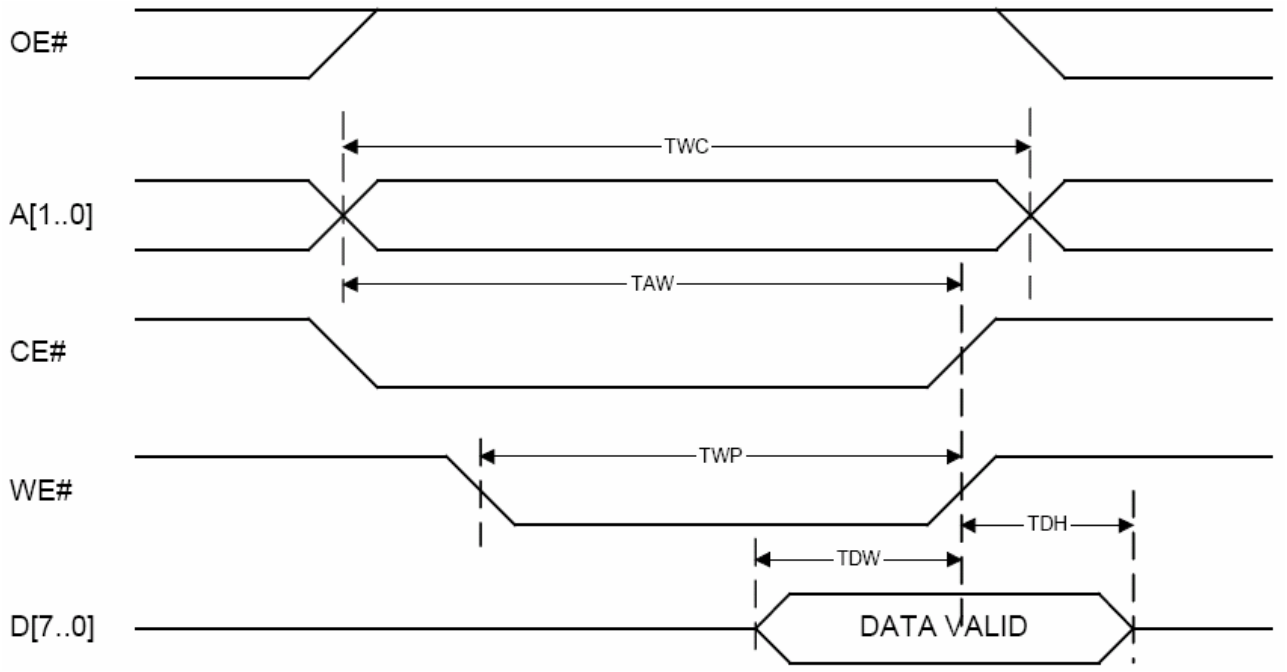
读字库芯片内部的点阵字库数据时，HOST 写字库芯片的 3 个地址寄存器，字库芯片把对应地址的数据送给 HOST。在 OE#和 CE#同时为低电平、WE#为高电平的情况下，HOST 可以从数据总线（D[7..0]）读出字库芯片的个字节的数据。当出现 OE# 或 CE# 中的一个信号变高，则字库芯片内部在此时刻把地址寄存器增 1，并保持地址寄存器的值。



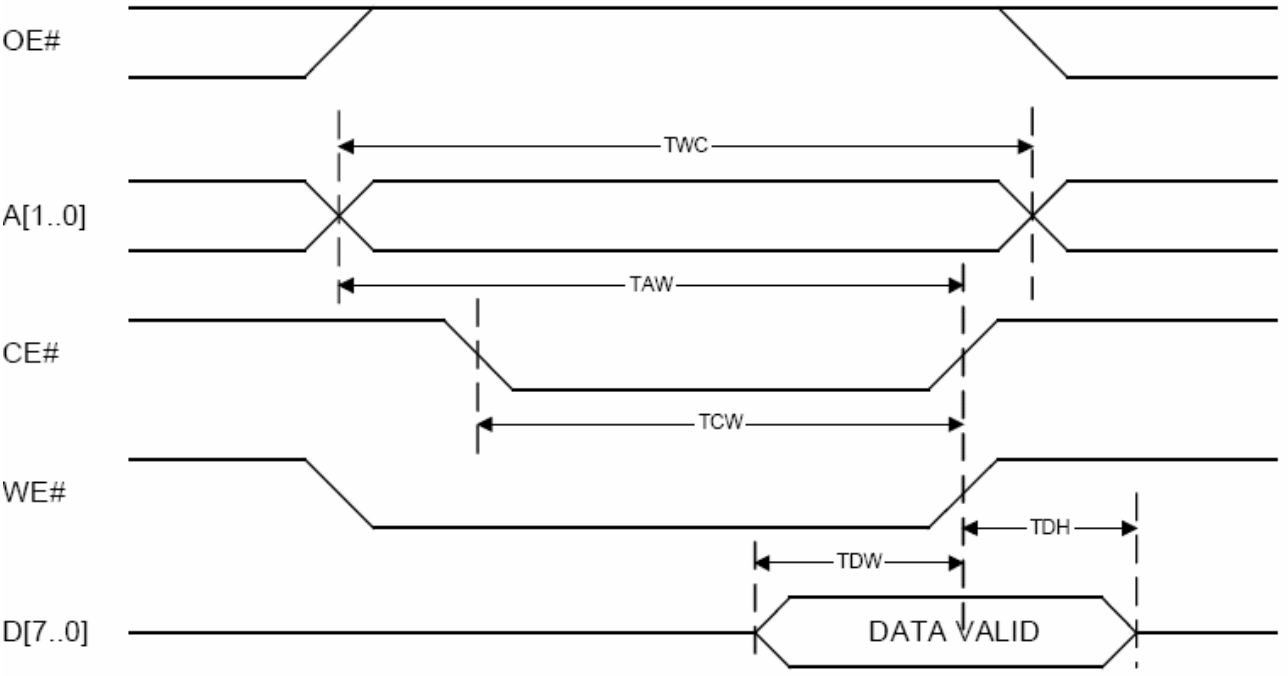
读周期时序波形图

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

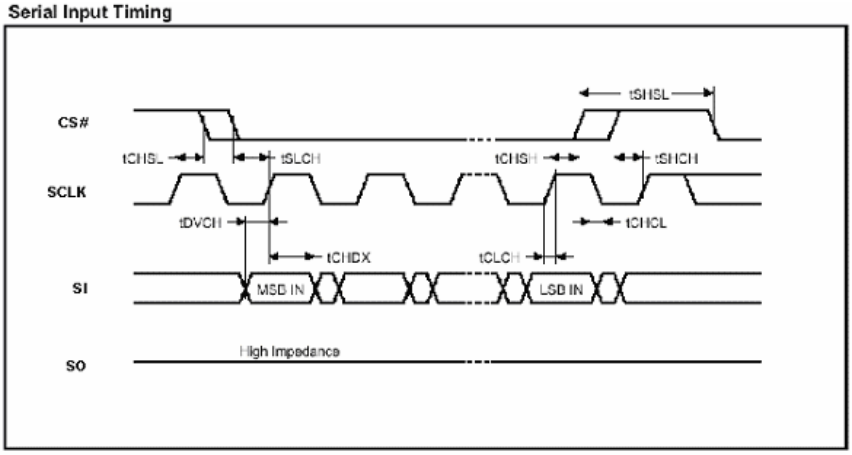
7.3.1 SPI 接口模式下 AC 特性

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

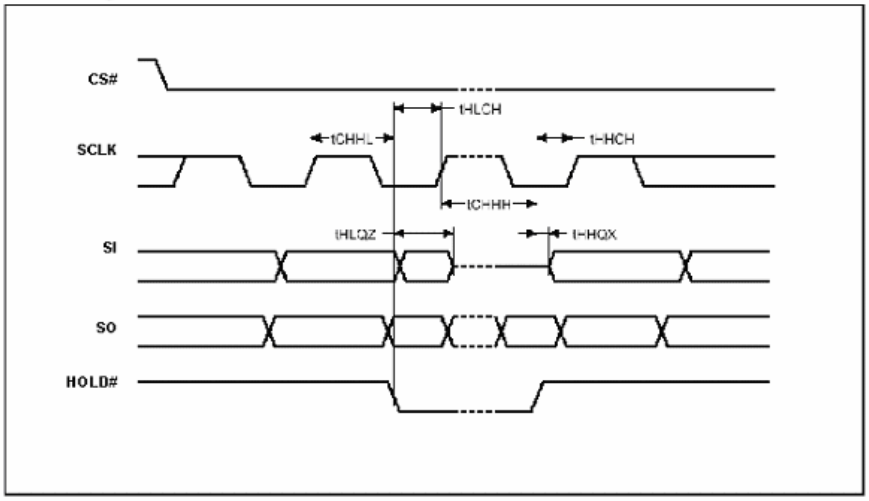
Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	40	MHz
t_{CH}	t_{CLH}	Clock High Time	11		ns
t_{CL}	t_{CLL}	Clock Low Time	11		ns
t_{CLCH}		Clock Rise Time(peak to peak)	0.1		V/ns
t_{CHCL}		Clock Fall Time (peak to peak)	0.1		V/ns
t_{SLCH}	t_{CSS}	CS# Active Setup Time (relative to SCLK)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

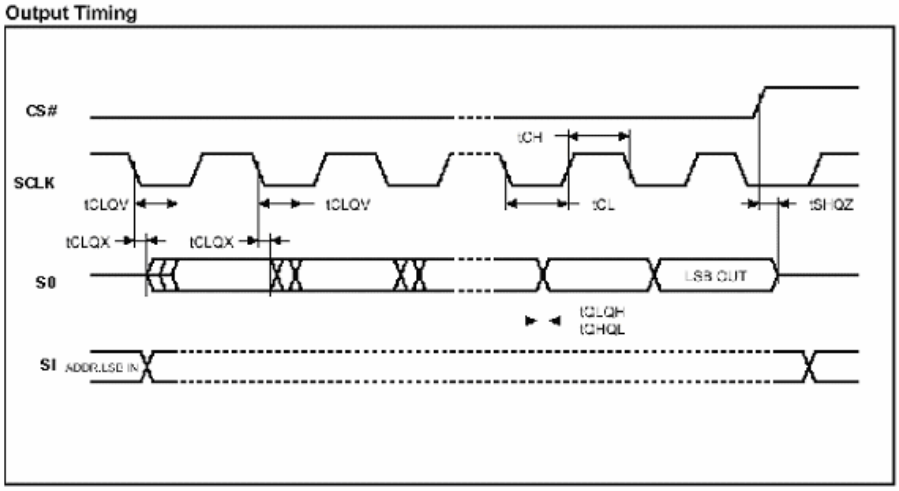
Serial Input Timing



Hold Timing



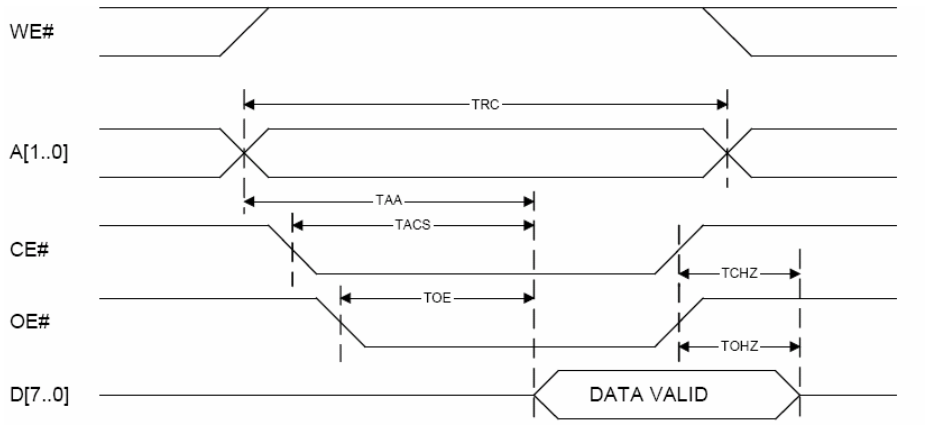
Output Timing



7.3.2 PLII 接口模式下 AC 特性

7.3.2.1 读周期时间特性

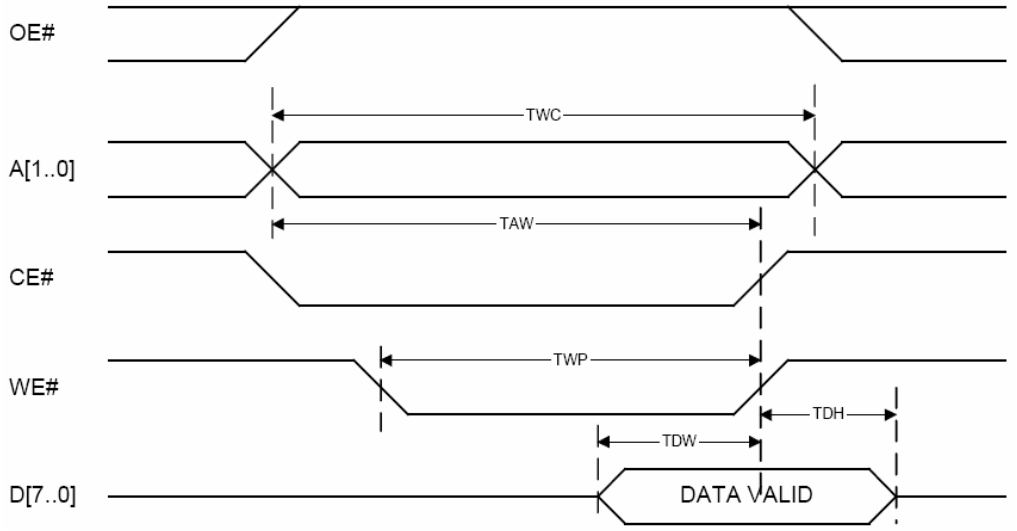
Symbol	Parameter	Min.	Max.	Unit
TRC	Read Cycle Time	70	-	ns
TAA	Address Access Time	-	60	ns
TACS	Chip Select Access Time	-	60	ns
TOE	Output Enable to Output Valid	-	30	ns
TCHZ	Chip Deselect to Output in High-Z	-	10	ns
TOHZ	Output Disable to Output in High-Z	-	10	ns



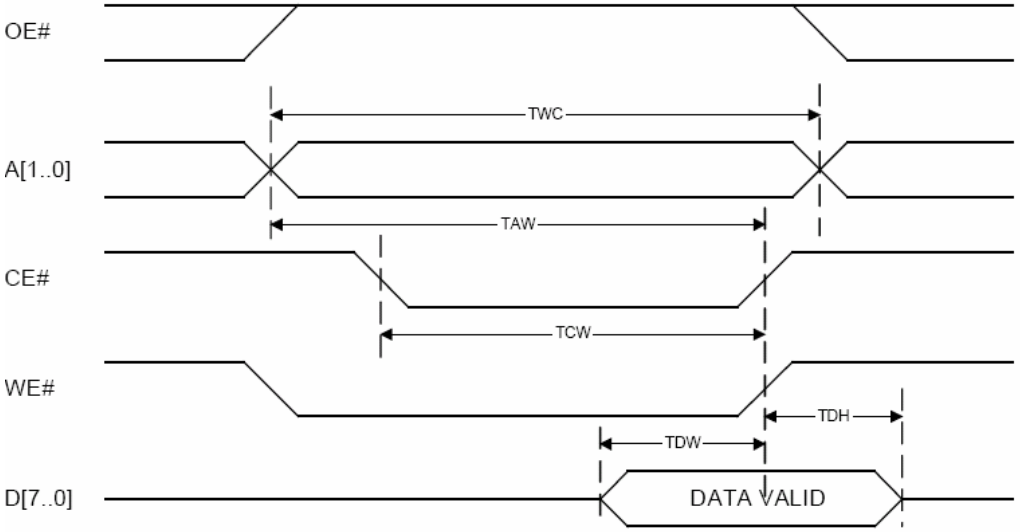
读周期时序波形图

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



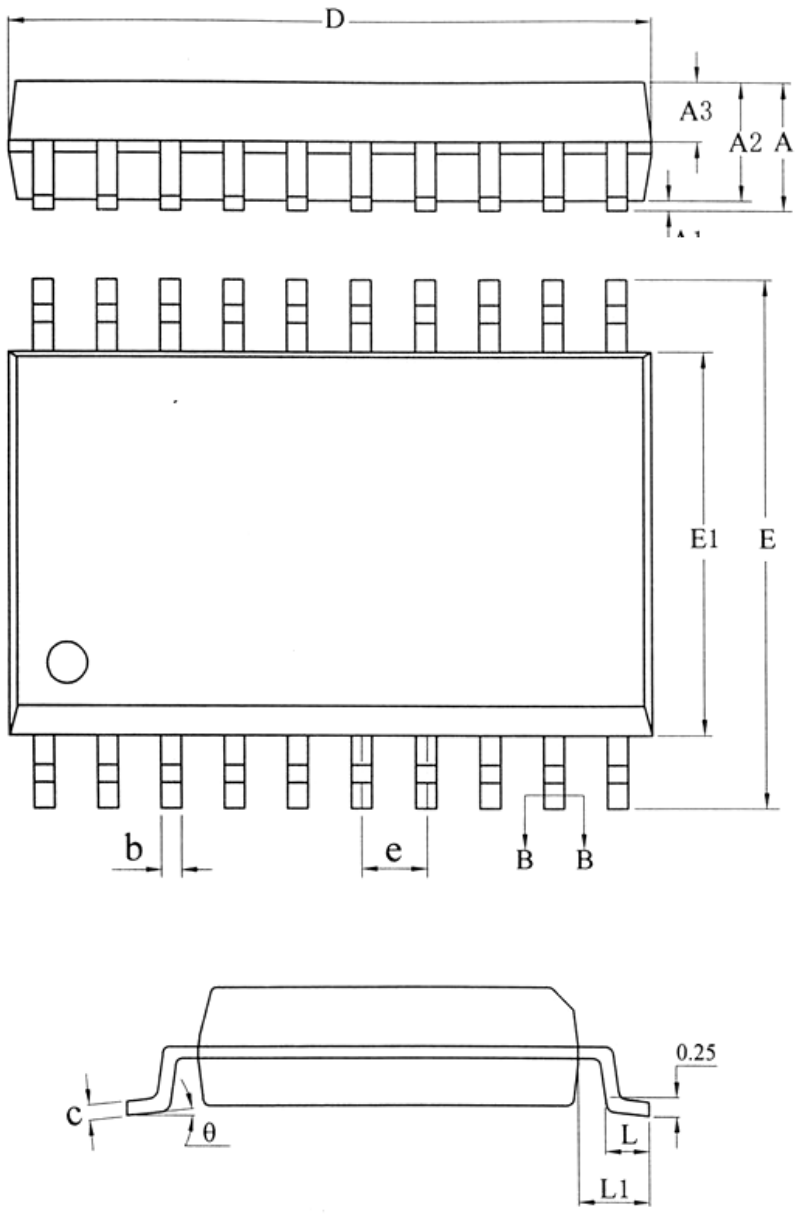
写周期时序波形图（WE#控制的时序）



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°

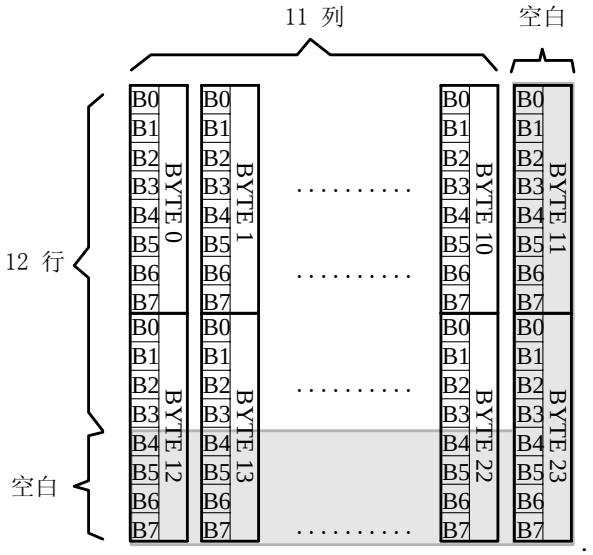
9 功能描述

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：

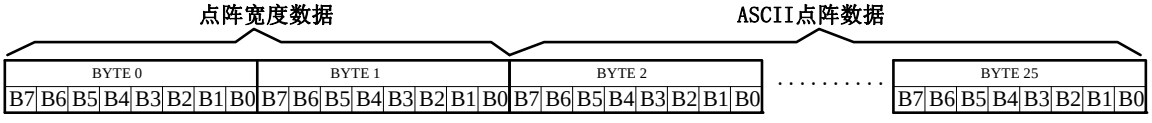


9.1.2 12 点阵 ASCII 方头字符排列格式

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

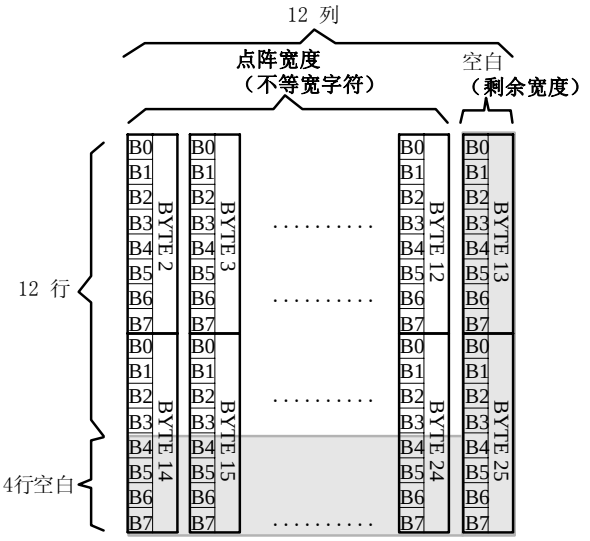
■ 存储格式

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

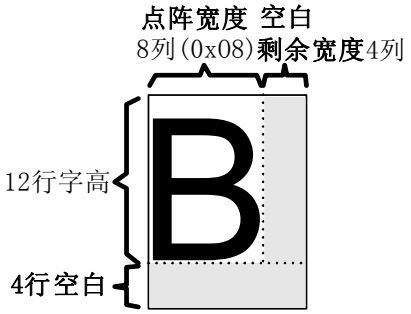
例如：ASCII 方头字符 B

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

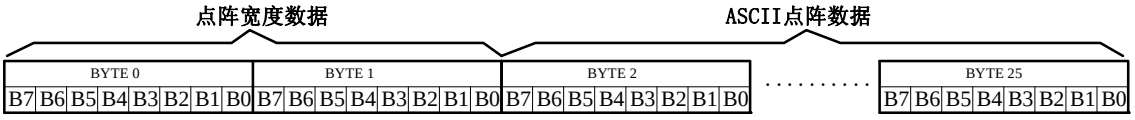
为 ASCII 方头字符 B 的点阵数据。

9.1.3 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。

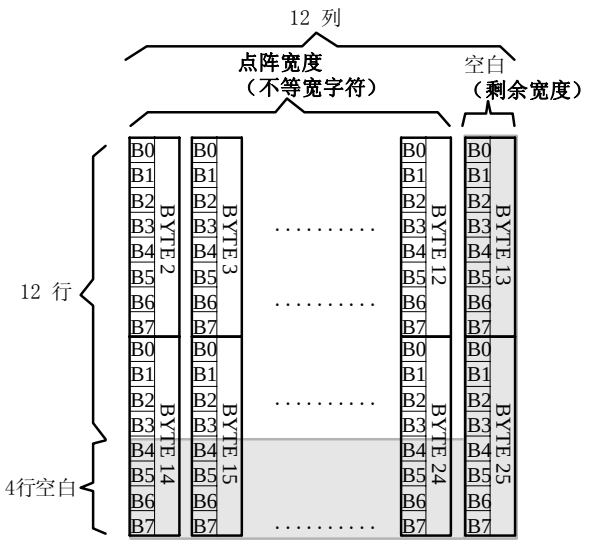
■ 存储格式

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

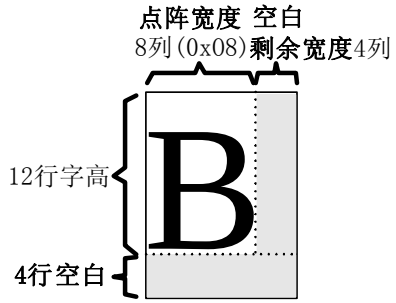
例如：ASCII 白正字符 B

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

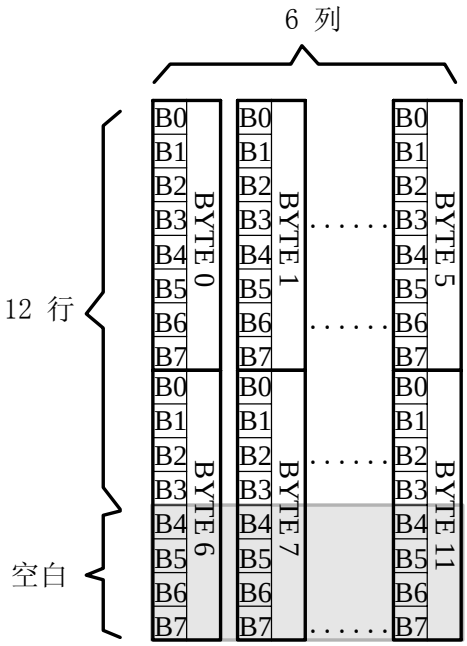
为 ASCII 白正字符 B 的点阵宽度数据，即：8 位宽度。字符后面有 8 位空白区，可以在排版下一个字时考虑到这一点，将下一个字的起始位置前移。（见下图）



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。

9.1.4 6X12 点 ASCII 扩展字符排列格式

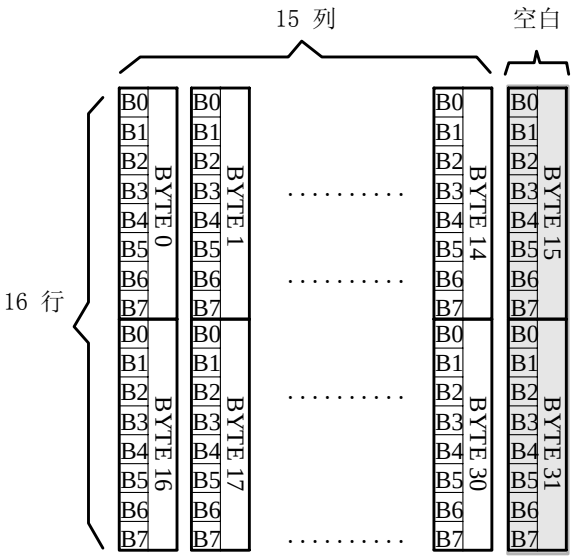
6X12 点 ASCII 的信息需要 12 个字节（BYTE 0 – BYTE11）来表示。



6X12 点 ASCII 字符排列格式图

9.1.5 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：



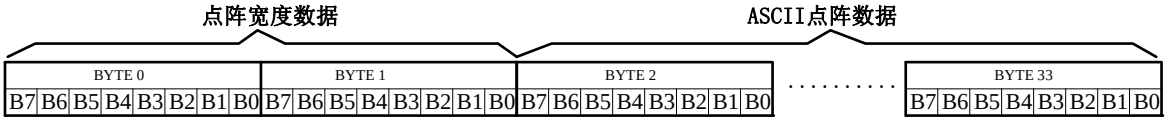
15X16 点汉字排列格式图

9.1.6 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。

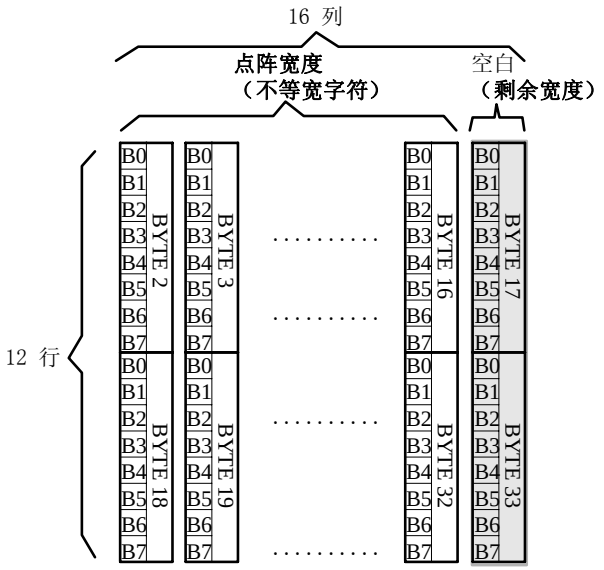
■ 存储格式

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

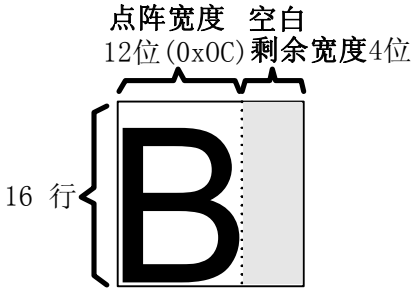
其中：

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。

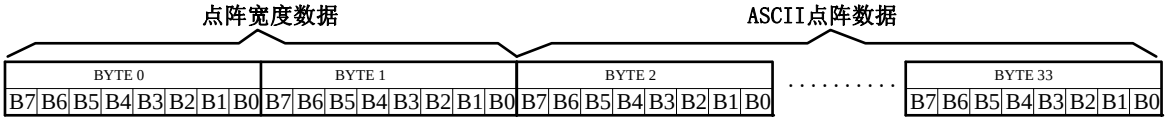


9.1.7 16 点阵 ASCII 白正字符排列格式

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

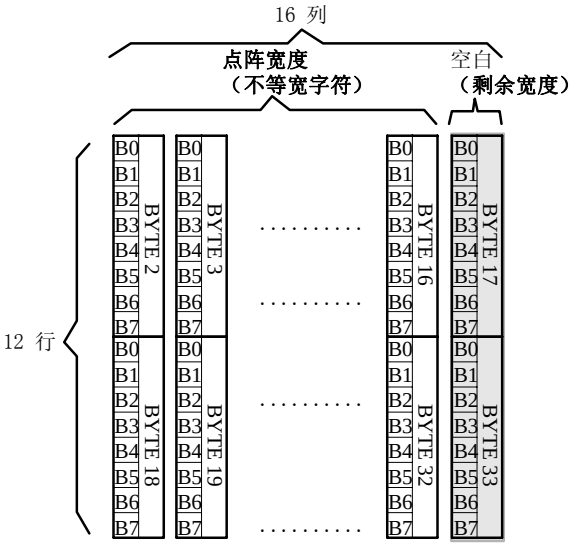
■ 存储格式

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 40 40 7F 42 42 42 43 24 18 00 00 00 00 00 00

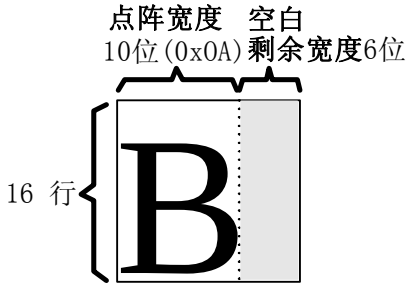
其中：

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）

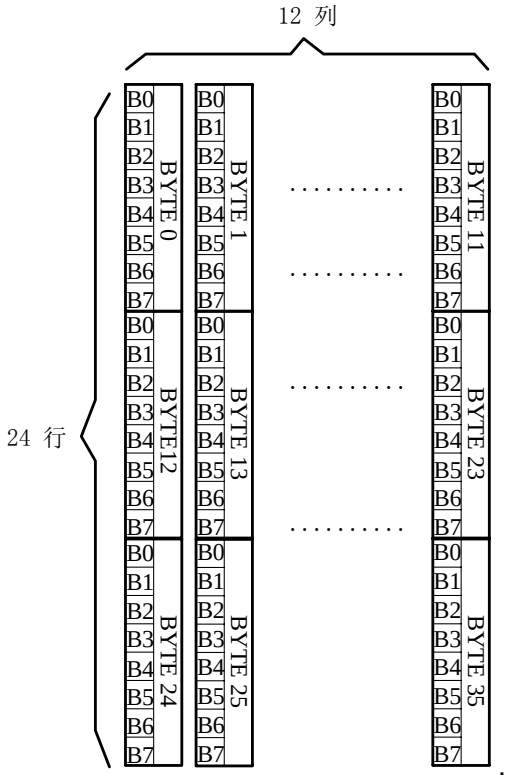
BYTE2-33: 10 10 F0 10 10 10 20 C0 00 00 00 00 00 00 00 00 40 40 7F 42 42 42 43 24 18 00 00 00 00 00 00 00

为 ASCII 白正字符 B 的点阵数据。



9.1.8 24X24 点汉字排列格式

24X24 点汉字的信息需要 36 个字节（BYTE 0 – BYTE 36）来表示。该 12 X24 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：.



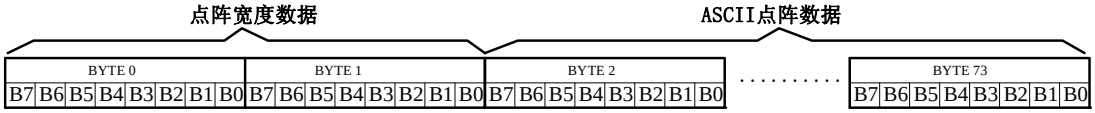
24X24 点汉字排列格式图

9.1.9 24 点阵 ASCII 方头字符排列格式

24 点阵 ASCII 方头字符的信息需要 74 个字节（BYTE 0 – BYTE73）来表示。

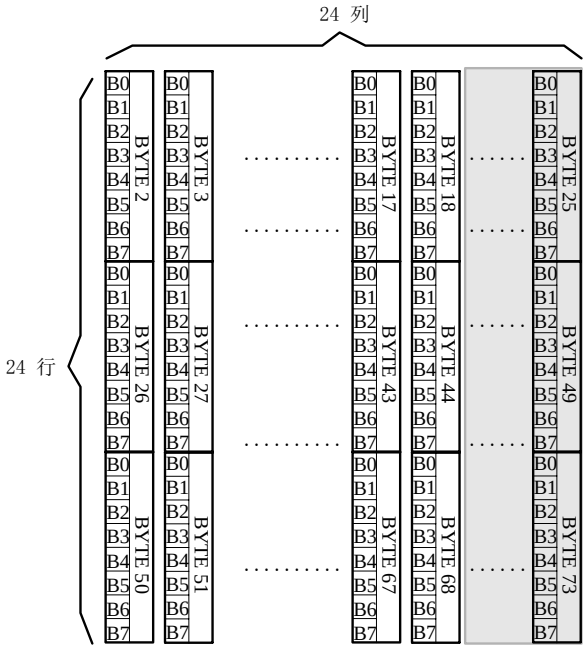
■ 存储格式

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，BYTE2-73 存放点阵数据。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。



24 点阵 ASCII 方头字符排列格式图

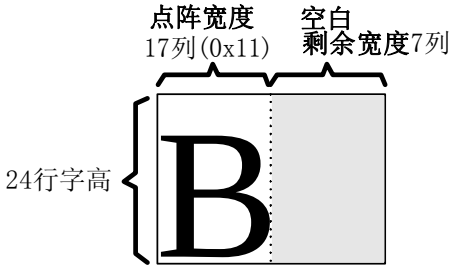
例如：ASCII 方头字符 B

0-73 BYTE 的点阵数据是: 00 11 00 00 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 80 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF E1 E1 E1 E1 E1 E1 F3 FF FF 9E 00 00 00 00 00 00 00 00 00 00 00 00 7F 7F 7F 70 70 70 70 70 70 70 79 3F 3F 0F 00 00 00 00 00 00 00 00

其中：

BYTE0~ BYTE1: 00 11

为 ASCII 方头字符 B 的点阵宽度数据，即：17 位宽度。字符后面有 7 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）



BYTE2-73: 00 00 C0 C0 C0 C0 C0 C0 C0 C0 C0 C0 80 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF E1 E1 E1 E1 E1 E1 F3 FF FF 9E 00 00 00 00 00 00 00 00 00 00 00 00 7F 7F 7F 70 70 70 70 70 70 70 79 3F 3F 0F 00 00 00 00 00 00 00 00

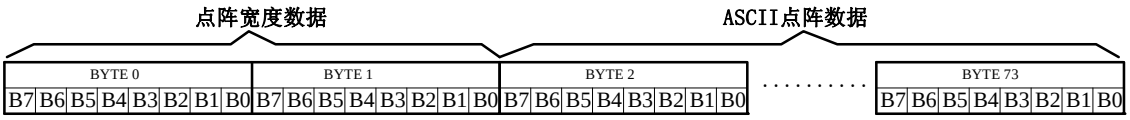
为 ASCII 方头字符 B 的点阵数据。

9.1.10 24 点阵 ASCII 白正字符排列格式

24 点阵 ASCII 白正字符的信息需要 74 个字节（BYTE 0 – BYTE73）来表示。

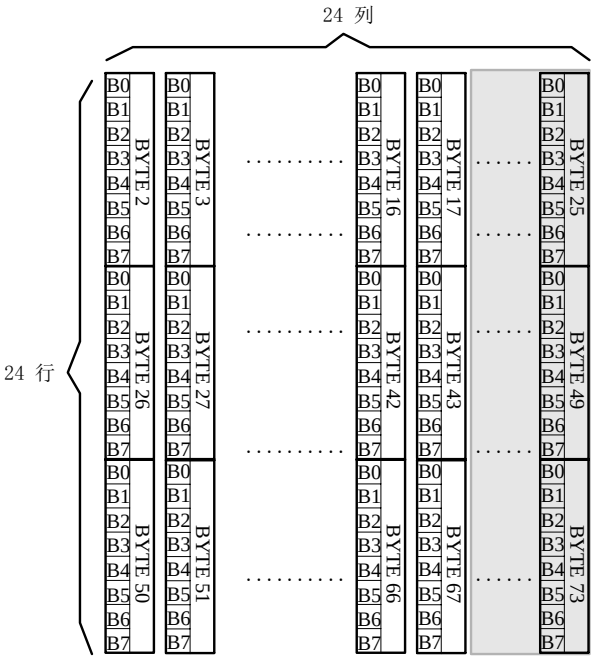
■ 存储格式

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，BYTE2-73 存放点阵数据。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。



24 点阵 ASCII 白正字符排列格式图

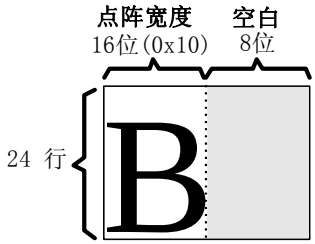
例如：ASCII 白正字符 B

0-73BYTE 的点阵数据是: 00 10 00 20 20 E0 E0 20 20 20 20 20 60 C0 C0 80 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 10 10 10 10 10 30 38 6F E7 80 00 00 00 00 00 00 00 00 00 00 00 10 10 1F 1F 10 10 10 10 10 10 18 0C 0F 03 00 00 00 00 00 00 00 00 00 00

其中：

BYTE0~ BYTE1: 00 10

为 ASCII 白正字符 B 的点阵宽度数据，即：16 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）

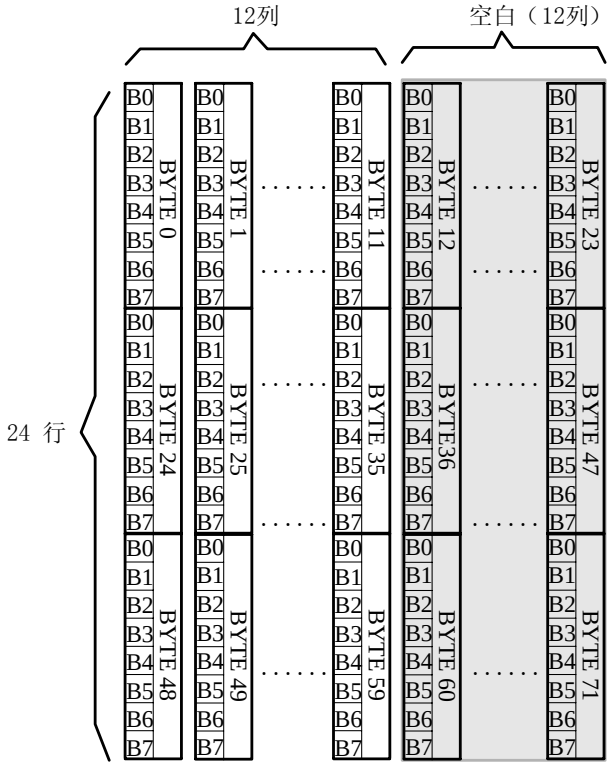


BYTE2-73: 00 20 20 E0 E0 20 20 20 20 20 60 C0 C0 80 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 10 10 10 10 10 30 38 6F E7 80 00 00 00 00 00 00 00 00 00 00 10 10 1F 1F 10 10 10 10 10 10 18 0C 0F 03 00 00 00 00 00 00 00 00 00

为 ASCII 白正字符 B 的点阵数据。

9.1.11 12X24 点扩展字符排列格式

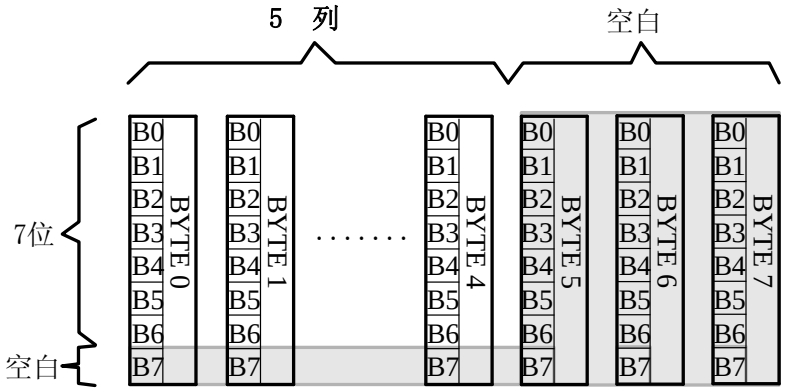
12X24 点扩展字符的信息需要 72 个字节（BYTE 0 – BYTE 71）来表示。但是由于该字符为标准的 24X24 点格式，而存储空间是按照 12X24 点 BYTE 取整进行存储的（即 72 BYTES）,注意在排版时作相应的调整。



12X24 点扩展字符排列格式

9.1.12 5X7 点 ASCII 字符排列格式

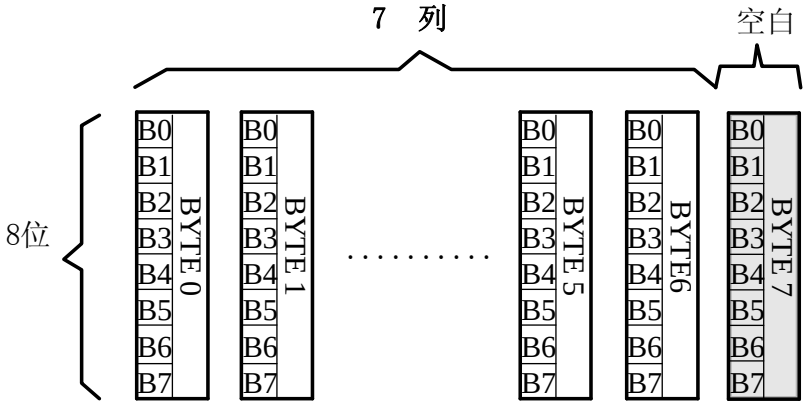
5X7 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



5X7 点 ASCII 字符排列格式图

9.1.13 7X8 点 ASCII 字符排列格式

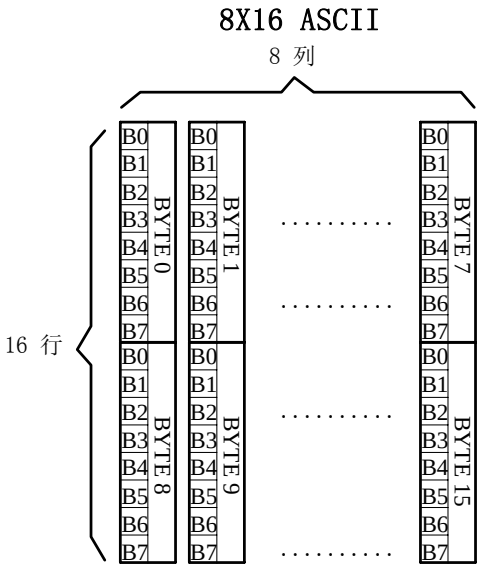
7X8 点 ASCII 的信息需要 8 个字节（BYTE 0 – BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



7X8 点 ASCII 字符排列格式图

9.1.14 8X16 点 ASCII 扩展字符排列格式

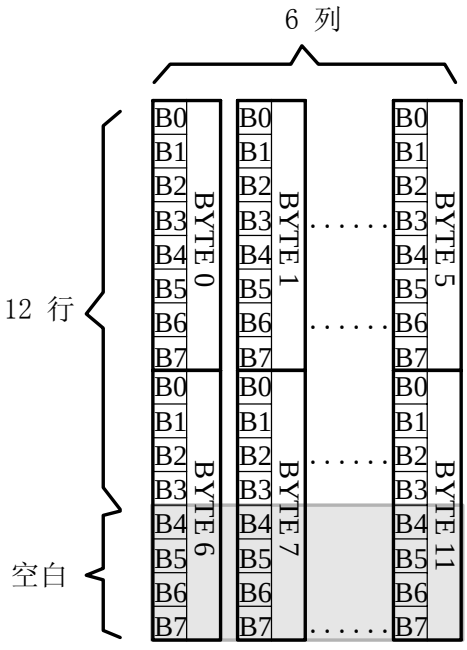
8X16 点 ASCII 的信息需要 16 个字节（BYTE 0 – BYTE15）来表示。



8x16 点 ASCII 字符排列格式图

9.1.15 6X12 点国际音标字符排列格式

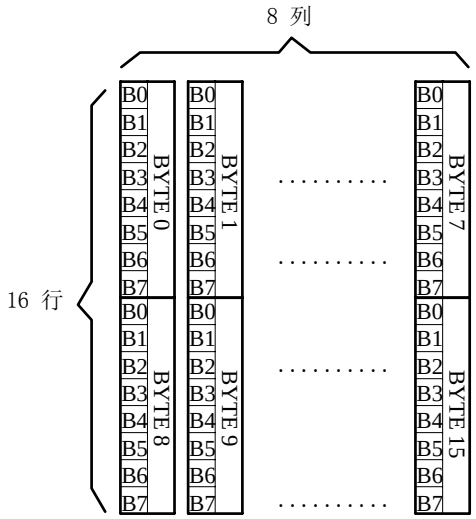
6X12 点国际音标信息需要 12 个字节（BYTE 0 – BYTE12）来表示。



6X12 点国际音标字符排列格式

9.1.16 8X16 点国际音标字符排列格式

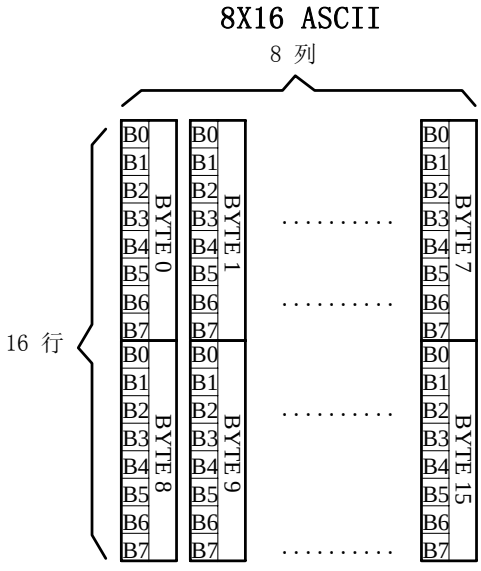
8X16 点国际音标字符信息需要 16 个字节（BYTE 0 – BYTE15）来表示。



8X16 点 ASCII 国际音标字符格式

9.1.17 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

9.2 汉字点阵字库结构

- BIG5 标准点阵字库
  标准点阵字库对应 BIG5 码位 A440~F9DC 共计 13060 个汉字。
- 扩展字符对应 GB2312 码位的 AAA1~ABC0 共计 126 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。
- 扩展字符对应 GB18030 码位的 AAA1~ABC0 共计 126 个字符。
- 5X7 点 ASCII 字符对应 ASCII 码位的 20~7E 共计 96 个字符。
- 7X8 点 ASCII 字符对应 ASCII 码位的 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。(或者对应 ASCII 码位 20~7E 共计 96 个字符,其余忽略)
- 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)
- 1 套 12X24 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)
- 1 套 16X32 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。

GT23L24B3Y0　字库内容说明



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。

11X12 点阵字库地址分配(字节地址)：0000 0000 H ~ 0005 1DF7 H ;

15X16 点阵字库地址分配(字节地址)：0005 1DF8 H ~ 000B F097 H ;

24X24 点阵字库地址分配(字节地址)：000B F098 H ~ 001B 4A7F H .

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 ~ 0005 1DF7 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 GTBIG5(unsigned short code)
功能：计算汉字点阵在芯片中的地址
参数：c1,c2：2 字节汉字内码通过参数 c1,c2 传入
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;
      “一”字的内码为 0xA440,则 byte address = GTBIG5 (0x A440) *24+BaseAdd
                                word address = byte address / 2
*****************************************************************************************************/
unsigned long GTBIG5(unsigned short code)
{
    unsigned long h=0;
    unsigned char c1,c2;
    c1=code>>8; c2=code&0xFF;
    if(c1<0xA1||c1>0xF9) return h;
    if(c2>=0x40 && c2<=0x7E) h=(c1-0xA1)*157+(c2-0x40);
    else if(c2>=0xA1 && c2<=0xFE) h=(c1-0xA1)*157+63+(c2-0xA1);
    else return h;
    return h;
}
```

9.3.2 15X16 汉字点阵地址的计算

15X16 点阵字库地址分配(字节地址)：0005 1DF8 H ~ 000B F097 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 GTBIG5(unsigned short code)
功能：计算汉字点阵在芯片中的地址
参数：c1,c2：2 字节汉字内码通过参数 c1,c2 传入
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＝0x51DF8;
“一”字的内码为 0xA440,则 byte address = GTBIG5 (0x A440) *32+BaseAdd
word address = byte address / 2
**********/

```
unsigned long GTBIG5(unsigned short code)
{
    unsigned long h=0;
    unsigned char c1,c2;
    c1=code>>8; c2=code&0xFF;
    if(c1<0xA1||c1>0xF9) return h;
    if(c2>=0x40 && c2<=0x7E) h=(c1-0xA1)*157+(c2-0x40);
    else if(c2>=0xA1 && c2<=0xFE) h=(c1-0xA1)*157+63+(c2-0xA1);
    else return h;
    return h;
}
```

9.3.3 24X24 汉字点阵地址的计算

15X16 点阵字库地址分配(字节地址)：000B F098 H ~ 001B 4A7F 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 GTBIG5(unsigned short code)
功能：计算汉字点阵在芯片中的地址
参数：c1,c2：2 字节汉字内码通过参数 c1,c2 传入
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＝0x BF098;
“一”字的内码为 0xA440,则 byte address = GTBIG5 (0x A440) *72+BaseAdd
word address = byte address / 2
**********/

```
unsigned long GTBIG5(unsigned short code)
{
    unsigned long h=0;
    unsigned char c1,c2;
    c1=code>>8; c2=code&0xFF;
    if(c1<0xA1||c1>0xF9) return h;
    if(c2>=0x40 && c2<=0x7E) h=(c1-0xA1)*157+(c2-0x40);
    else if(c2>=0xA1 && c2<=0xFE) h=(c1-0xA1)*157+63+(c2-0xA1);
    else return h;
    return h;
}
```

9.3.4 扩展字符地址的计算

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。

(1) 6X12 点扩展字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x1B4A80
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=0x1B6400
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=0x1B6400
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

(3) 12X24 点扩展字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x1B8B80
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) * 48+ 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) * 48+ BaseAdd

9.3.5 ASCII 地址的计算

ASCIICode 表示 ASCII 码（8bits）
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。

(1) 5X7 ASCII 字符点阵在芯片中的字节地址的计算方法如下：

BaseAdd=0x1B6C00
if (ASCIICode >= 0x 20) and (ASCIICode <= 0x 7E) 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=0x1B6F00
if (ASCIICode >=0x 20) and (ASCIICode <= 0x 7E) then
　　Address = (ASCIICode –0x 20) * 8+ BaseAdd

(4) 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=0x1B5080
if (ASCIICode >= 0x 20) and (ASCIICode <= 0x 7E) then
　　Address = (ASCIICode –0x 20) * 26+ BaseAdd

(5) 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=0x1B5A40
if (ASCIICode >= 0x 20) and (ASCIICode <=0x 7E) then
　　Address = (ASCIICode –0x 20) * 26+ BaseAdd

(6) 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=0x1B7200
if (ASCIICode >= 0x 20) and (ASCIICode <=0x 7E) then
　　Address = (ASCIICode –0x 20) * 34+ BaseAdd

(7) 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=0x1B7EC0
if (ASCIICode >= 0x 20) and (ASCIICode <=0x 7E) then
　　Address = (ASCIICode –0x 20) * 34+ BaseAdd

(8) 24 点阵 ASCII 方头字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x1BAF50

if (ASCIICode >= 0x 20) and (ASCIICode <= 0x 7E) then

Address = (ASCIICode –0x 20) * 74+ BaseAdd

(9) 24 点阵 ASCII 白正字符在芯片中的字节地址的计算方法如下：

BaseAdd=0x1BCB10

if (ASCIICode >= 0x 20) and (ASCIICode <=0x 7E) then

Address = (ASCIICode –0x 20) * 74+ BaseAdd

9.3.6 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=0x1C13F0

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

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

9.3.7 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=0x1C17F0

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 1C1C70

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

Address = (CODE –0x 20) * 32+BaseAdd

9.3.8 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 1BE6D0
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 1BFE50
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

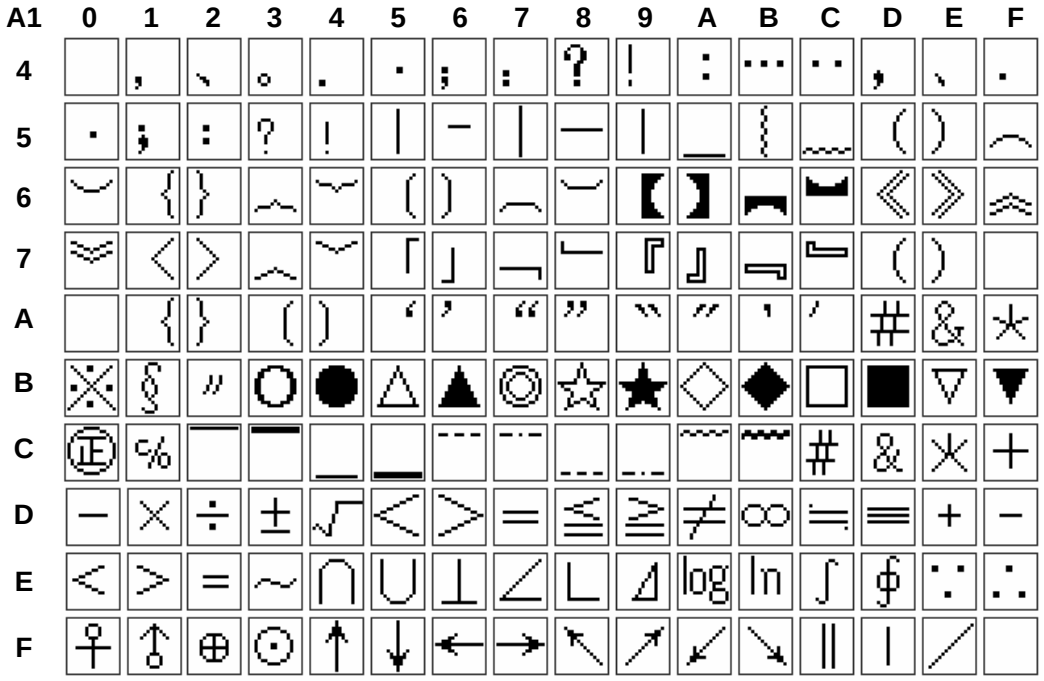
BaseAdd= 0x 1C0450
if (Unicode > = 0x 0400) and (unicode <=0x 04F9) then
Address = (unicode –0x 0400) * 16+BaseAdd

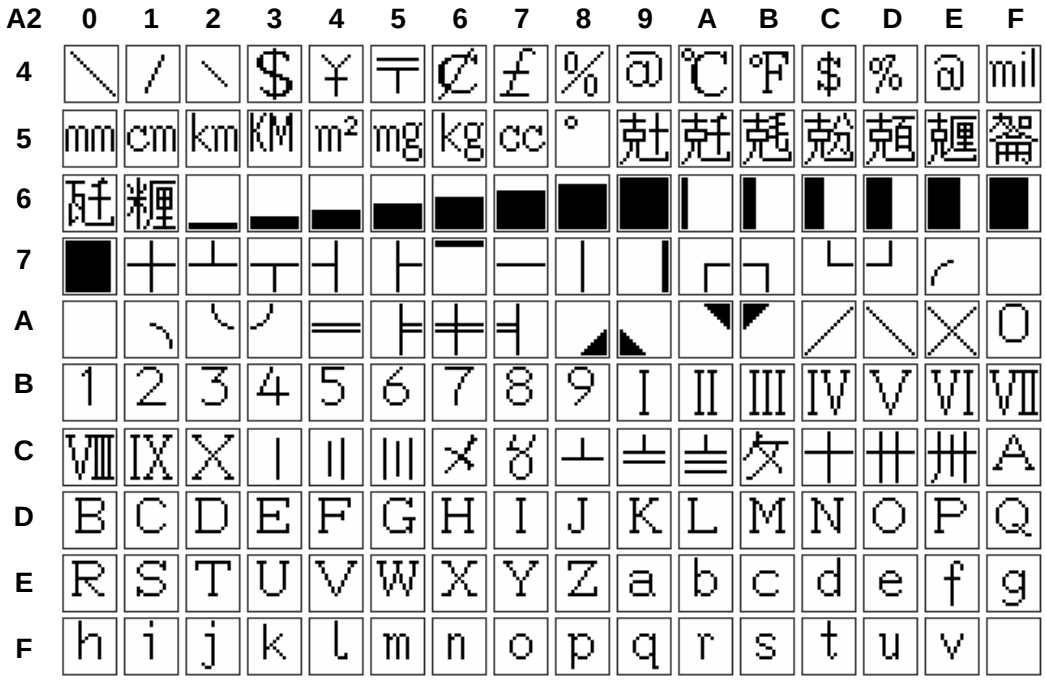
10 附录

10.1 BIG5 (字符区)

BIG5 标准点阵字符区分别对应码位的 A140~A3BF 共计 408 个字符；

BIG5 字符区






A3 0 1 2 3 4 5 6 7 8 9 A B C D E F
4
5
6
7
A
B
C
D
E
F

10.2 扩展字符区字符

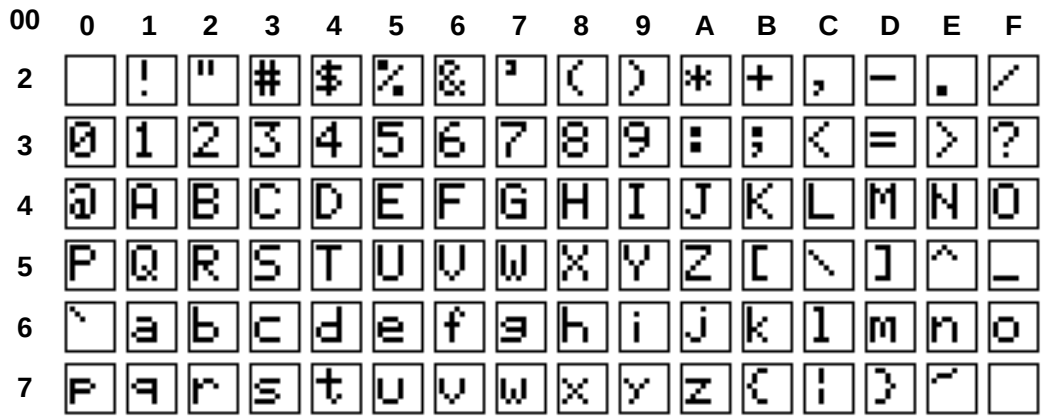
扩展字符区的内码组成为 AAA1~ABC0 共计 126 个字符。

扩展字符区

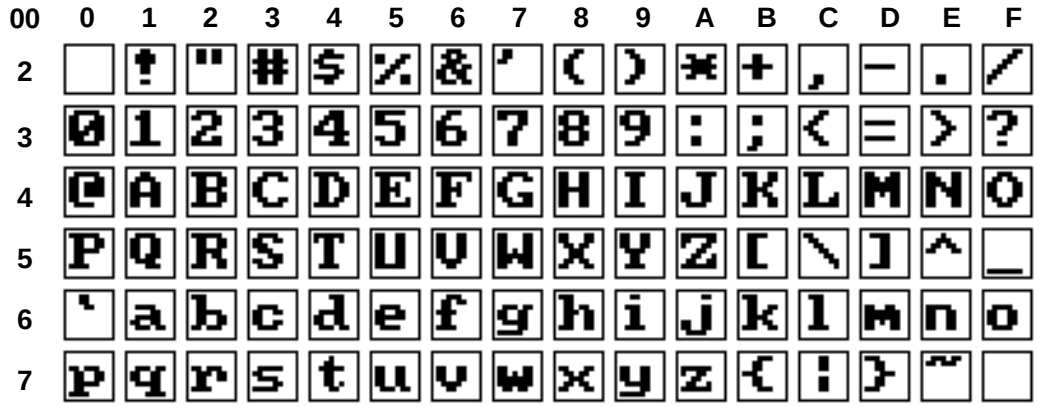
AA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		!	"	#	¥	%	&	'	(	)	*	+	,	-	.	/
B	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
E	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	¯	

AB	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	g															
D																
E																
F																

10.3 5×7 点 ASCII 字符（96 字符）



10.4 7×8 点 ASCII 字符（96 字符）



10.5 Unicode 字符区字符

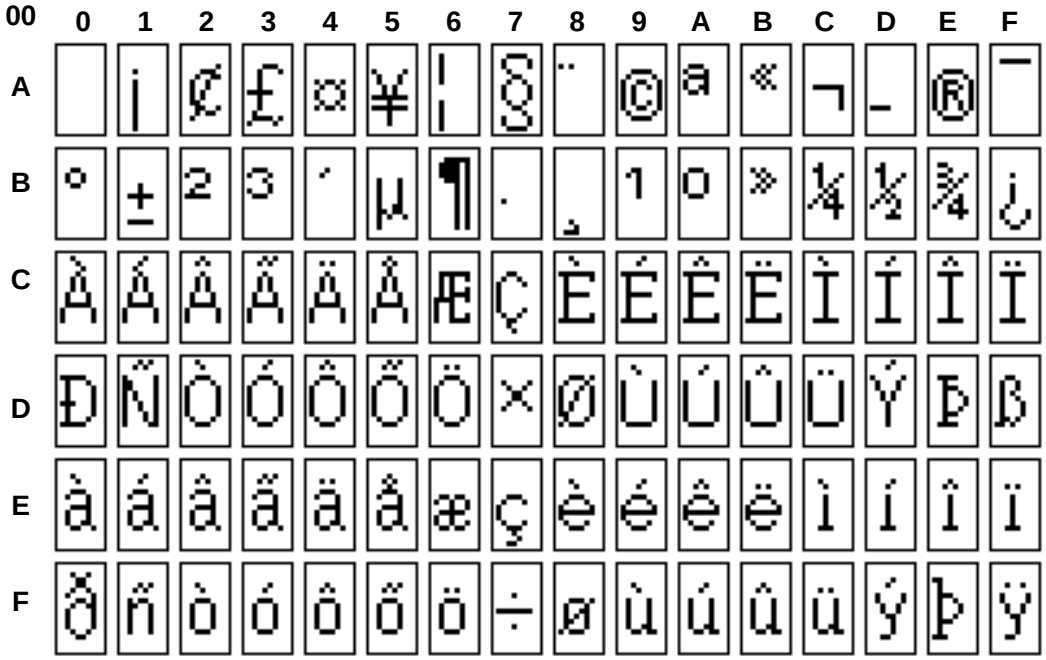
Unicode 字符区共收录　拉丁文系（LATIN）、希腊文系（GREEK）、西里尔文系（CYRILLIC）3 大文系，共计收录字符 456 个

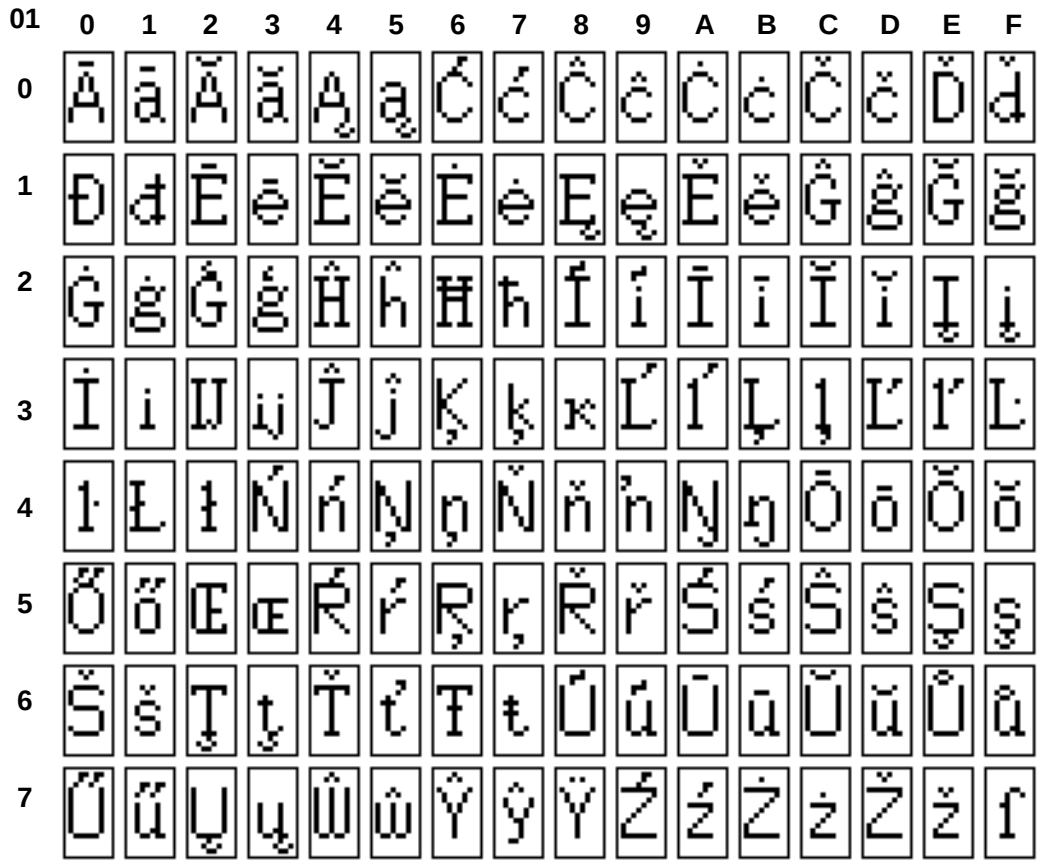
1）拉丁文系（8×16 点 LATIN）

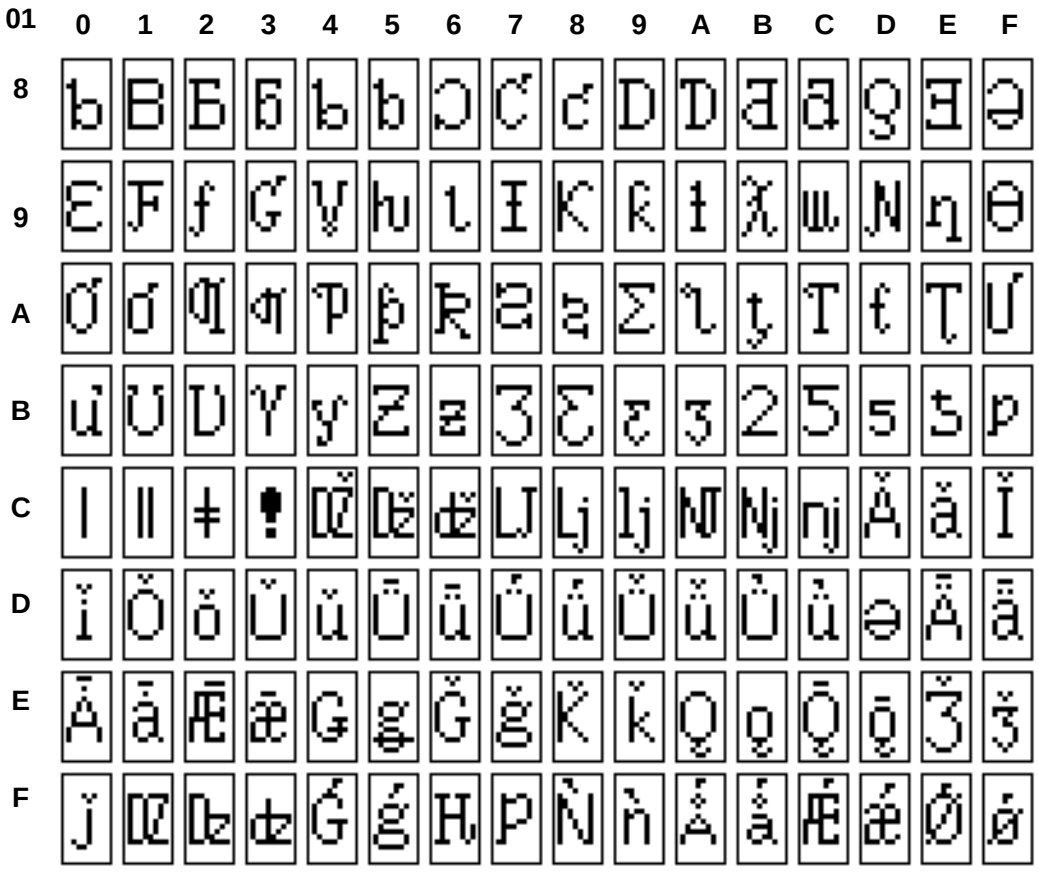
Unicode 字符区-拉丁文系的内码范围为 00A0~0217 共计收录 376 个字符。

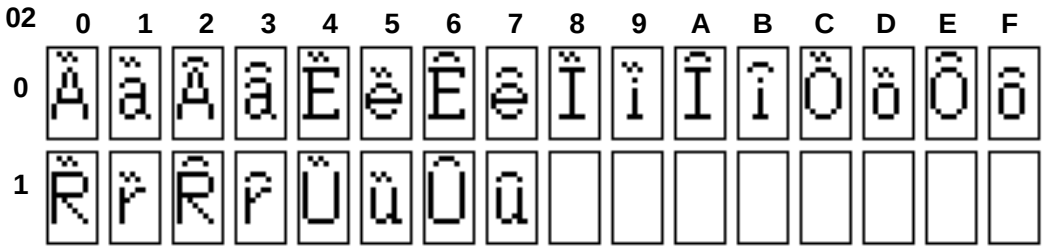
***说明**：由于拉丁文系（BASIC LATIN）的内码范围 0020~007E 与基本 ASCII 编码一致（可以共用），因此未列入 Unicode 字符区-拉丁文系中（见附件：10.2）。

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

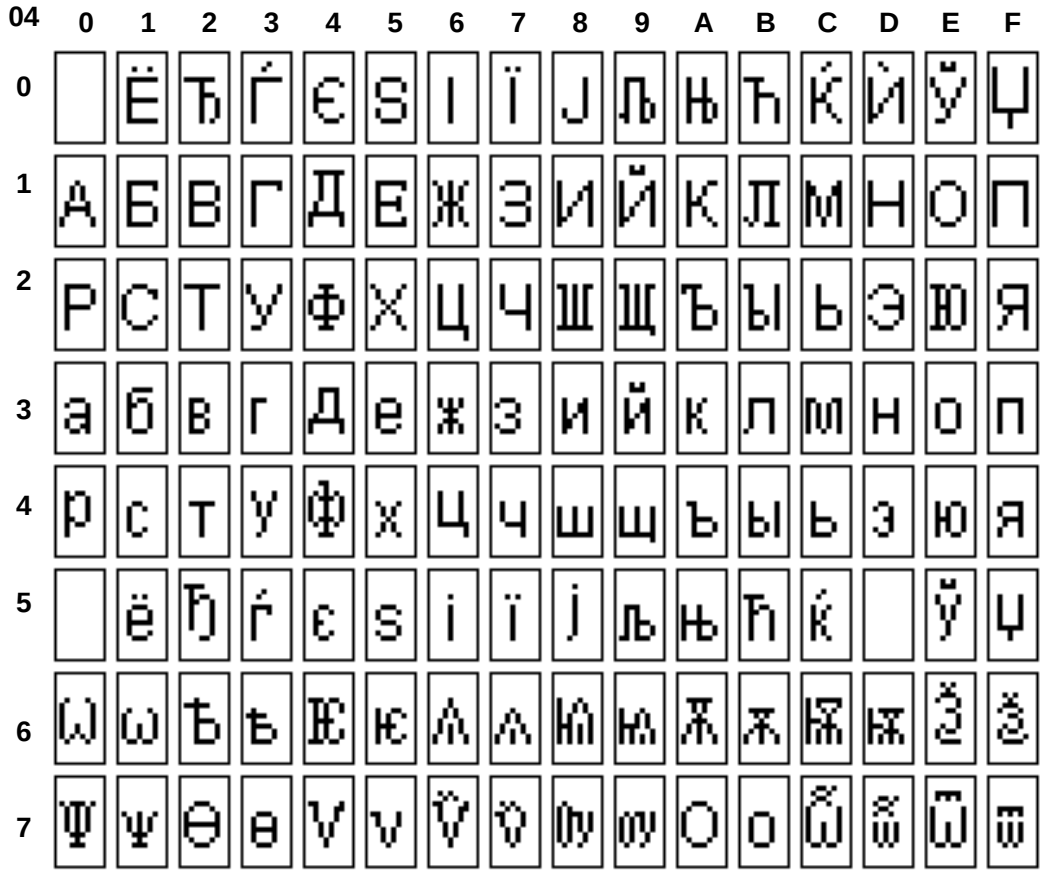

01
0 1 2 3 4 5 6 7 8 9 A B C D E F
8
9
A
B
C
D
E
F

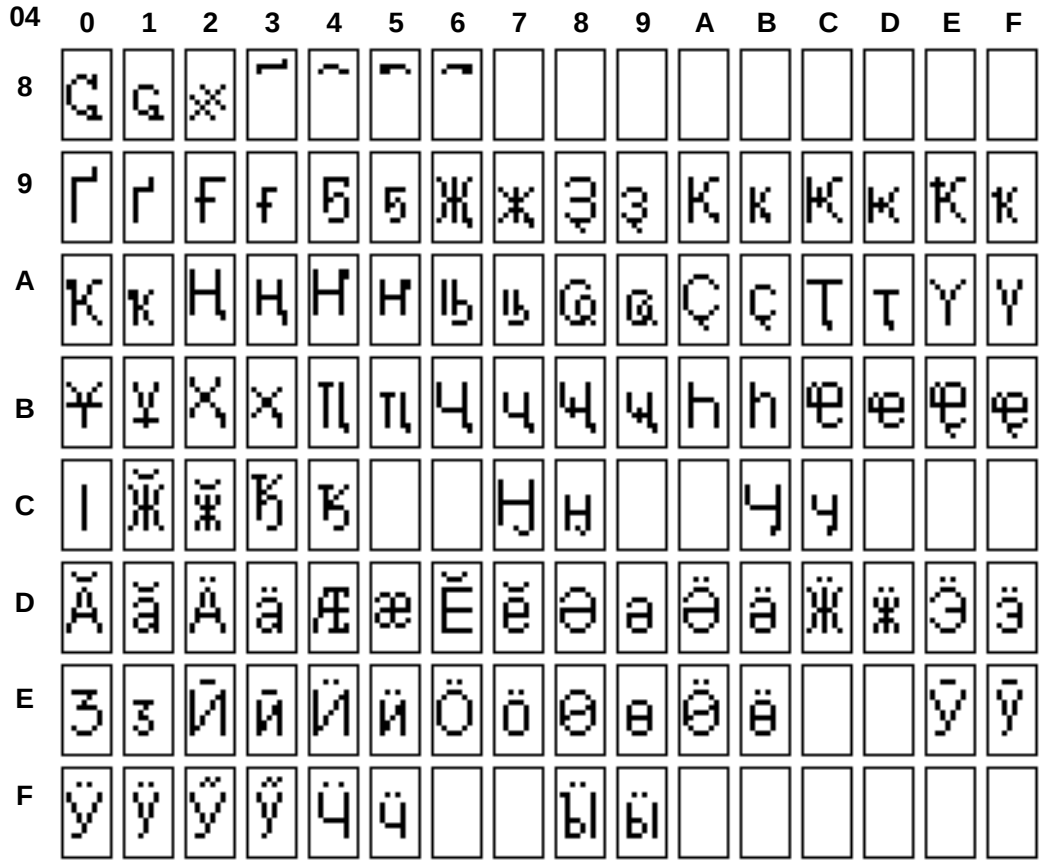

02
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1

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）

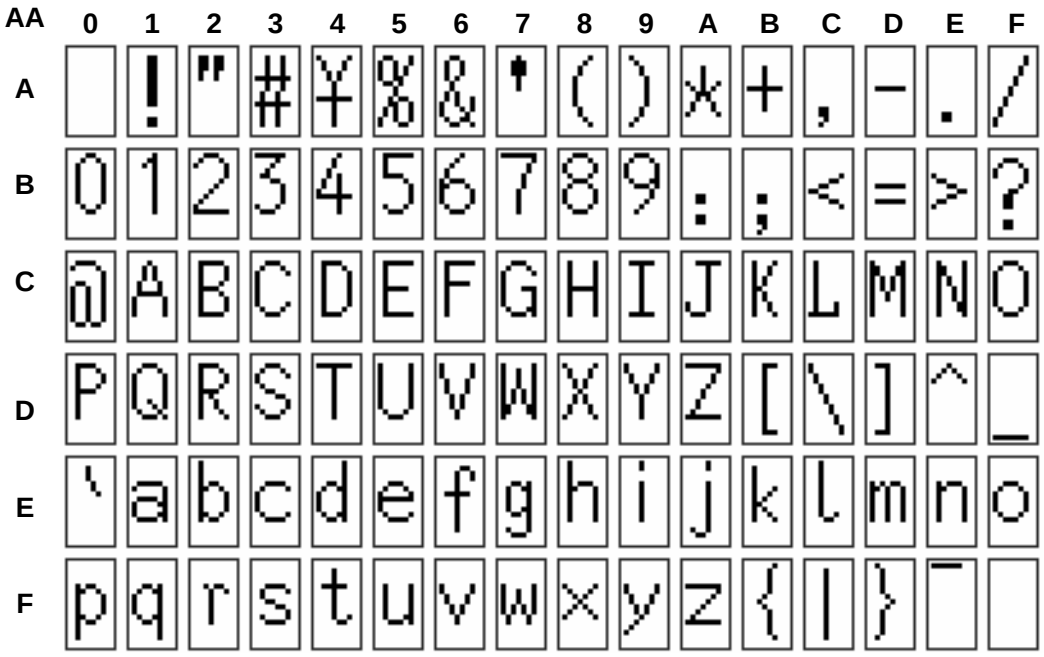
Unicode 字符区-希腊文系的内码范围为 0370~03CF 共计收录 96 个字符。

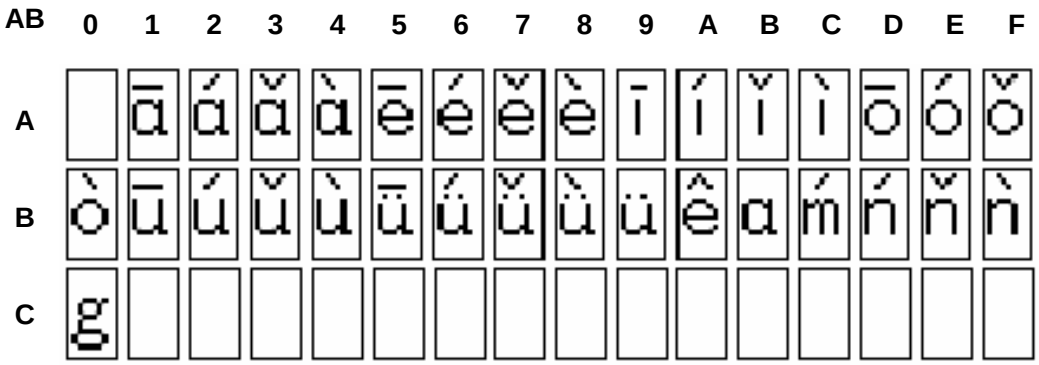
Unicode 字符区-西里尔文系



10.6 8×16 点扩展字符区 ASCII 字符（126 字符）

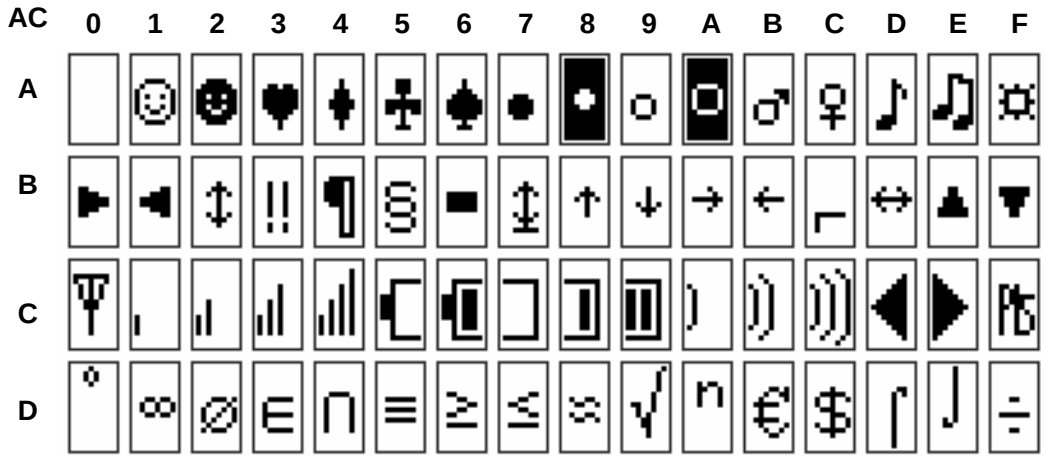
内码组成为 AAA1~ABC0 共计 126 个字符





10.7 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



10.8 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

