ER3303-1
DATASHEET

- Font size: 11X12dots、15X16dots、24X24dots
- Chinese character set: GB2312、GB12345、BIG5
- Compatible with Unicode
- Data arrangement: vertical byte, horizontal string
- Bus interface: SPI serial bus
- Chip package: SOP-8

VER 1.0

2013-Q7

INDEX

Section 1:Hardware

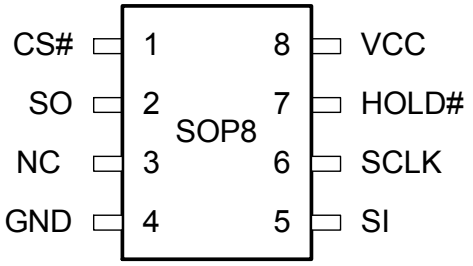
Section 2:Software

1 General

ER3303-1 is a Chinese font chip(with official license). It includes 11X12 dots, 15X16 dots, 24X24 dots. It supports GB2312 national standard character set, GB12345 national standard traditional character set, BIG5 traditional character set, and ASCII character, also compatible with Unicode. The data arrangement is byte vertical, string horizontal. The user can find the address of certain character dot matrix in the chip via the method provided by this datasheet.

1.1 Chip Feature

- Bus interface: SPI serial bus
- Data arrangement: byte vertical, string horizontal
- SPI access time: 20MHz(max.) @3.3V
- Supply voltage: 2.7V~3.6V
- Current:
  Operating: 12mA
  Standby: 10uA
- Package: SOP-8
- Package Size(SOP-8)：4.90mmX3.90mm(193milX154mil)
- Operating Temperature：-20℃~85℃(in SPI mode)；



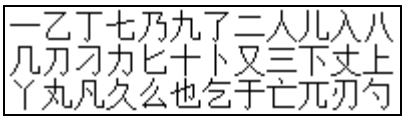
1.2 Chip Content

Type	Content	Character set	Characters
Chinese Font	11X12 dots GB2312 font	GB2312	6763+846
	11X12 dots GB12345 font	GB12345	6866+846
	11X12 dots BIG5 font	BIG5	13060+408
	15X16 dots GB2312 font	GB2312	6763+846
	15X16 dots GB12345 font	GB12345	6866+846
	15X16 dots BIG5 font	BIG5	13060+408
	24X24 dots GB2312 font	GB2312	6763+846
	24X24 dots GB12345 font	GB12345	6866+846
	24X24 dots BIG5 basic set font	BIG5 basic set	5401+408
ASCII font	5X7 dots ASCII font	ASCII	96
	7X8 dots ASCII font	ASCII	96
	6X12 dots ASCII font	ASCII	96
	8X16 dots ASCII font	ASCII	96
	12 dots Arial font	ASCII	96
	16 dots Arial font	ASCII	96
	24 dots Arial font	ASCII	96
Character set index table	GB12345 index table	GB12345	
	BIG5 index table	BIG5	
	Unicode index table	Unicode	

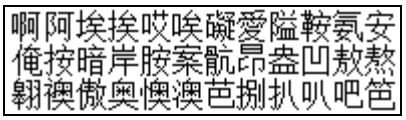
Font sample

GB2312

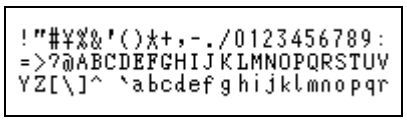
BIG5



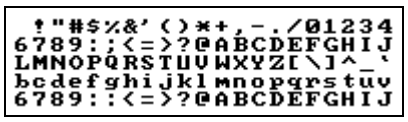
GB12345



5x7 DOT ASCII

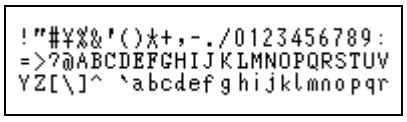


7x8 DOT ASCII

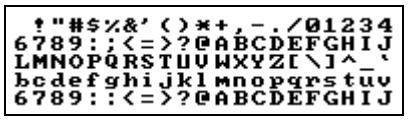


6x12 DOT ASCII

8x16 DOT ASCII



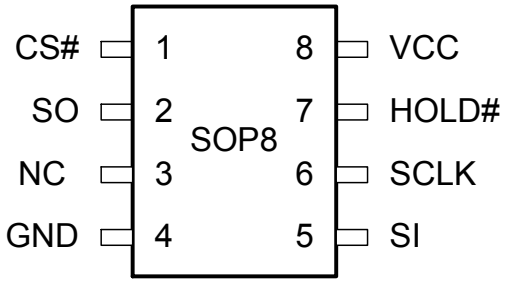
12 DOT ARIAL ASCII



16 DOT ARIAL ASCII

2 Pin description and interface connection

2.1 Pin configuration



SOP-8

SOP8	name	I/O	description
1	CS#	I	Chip enable input
2	SO	O	Serial data output
3	NC		No Connected
4	GND		Ground
5	SI	I	Serial data input
6	SCLK	I	Serial clock input
7	HOLD#	I	Hold, to pause the device without
8	VCC		+ 3.3V Power Supply

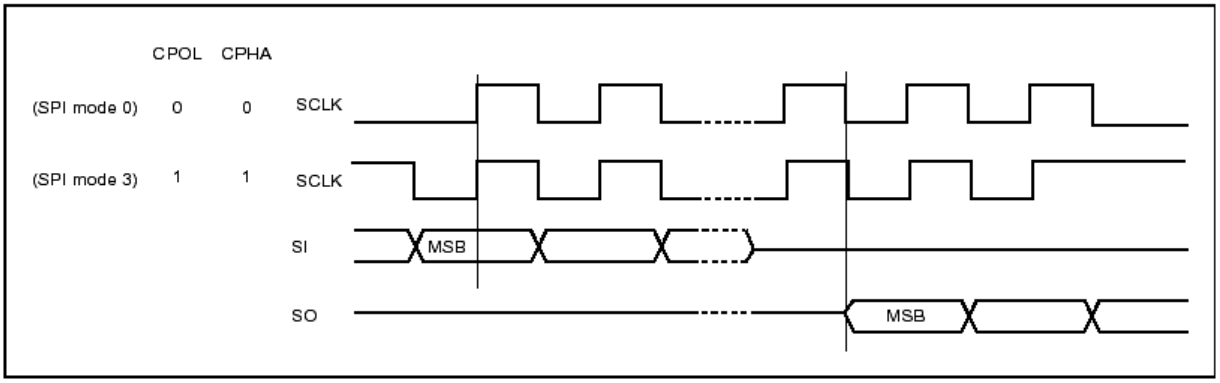
2.2 SPI interface description

Serial data output(SO): Data is shifted out on the falling edge of the serial clock.

Serial data input(SI) : Inputs are latched on the rising edge of the serial clock.

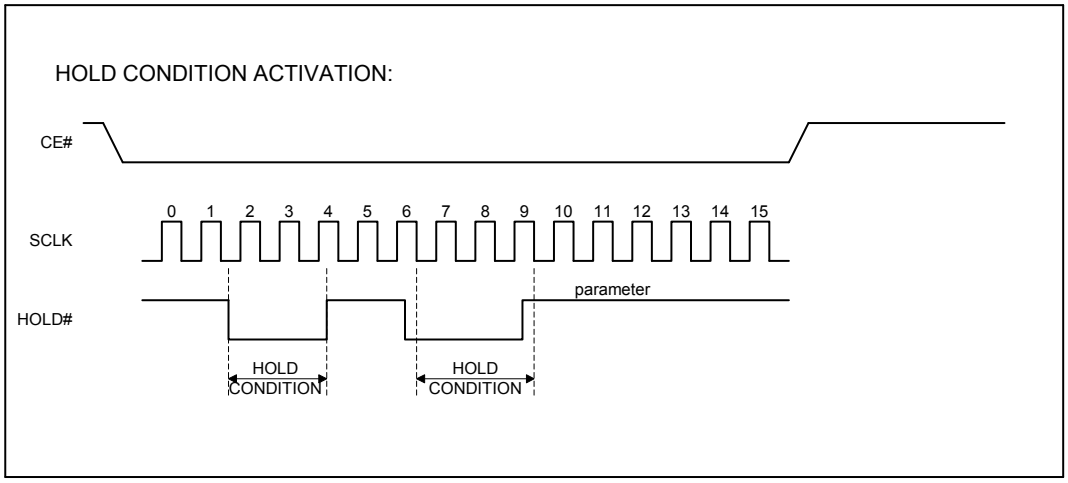
Serial clock input(SCLK): To provide the timing of the serial interface.

Chip enable input(CS#): The device is enabled by a high to low transition on CS#. CS# must remain low for the duration of any command sequence.



HOLD#: To temporarily stop serial communication with SPI flash memory without resetting the device.

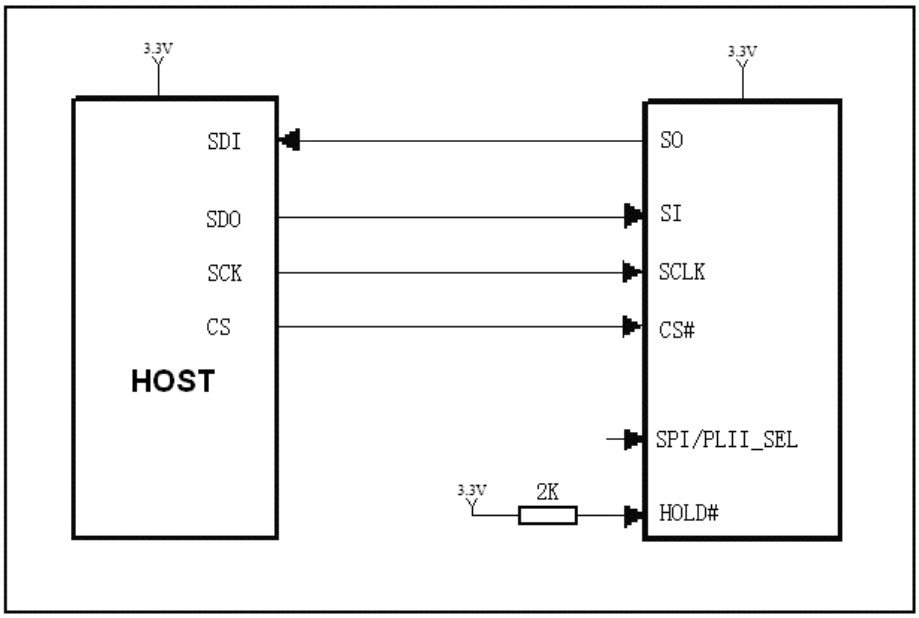
The HOLD# mode begins when the SCK active low state coincides with the falling edge of the HOLD# signal. The HOLD mode ends when the HOLD# signal's rising edge coincides with the SCK active low state.



2.3 SPI connection block diagram

When SPI/PLII_SEL is not connected,the chip is at SPI bus mode.

HOLD# PIN should pulled to 3.3V through 2K resister



3 Operating instruction

3.1 SPI bus operating instruction

Instruction Set

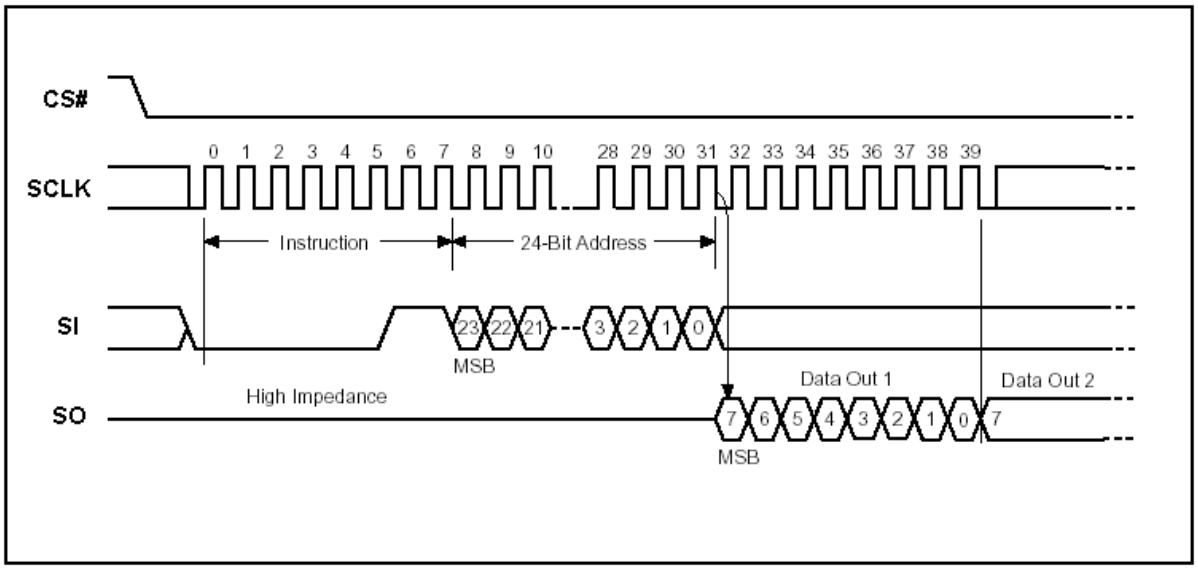
Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
READ	Read Data Bytes	0000 0011	03 h	3	—	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞

3.2 Read Data Bytes

The Read instruction supports up to 20 MHz, It outputs the data starting from the specified address location. The data output stream is continuous through all addresses until terminated by a low to high transition on CE#. The internal address pointer will automatically increment.

The Read instruction is initiated by executing an 8-bit command,03H, followed by address bits [A23-A0]. CE# must remain active low for the duration of the Read cycle.

Figure：Read Data Bytes (READ) Instruction Sequence and Data-outsequence:

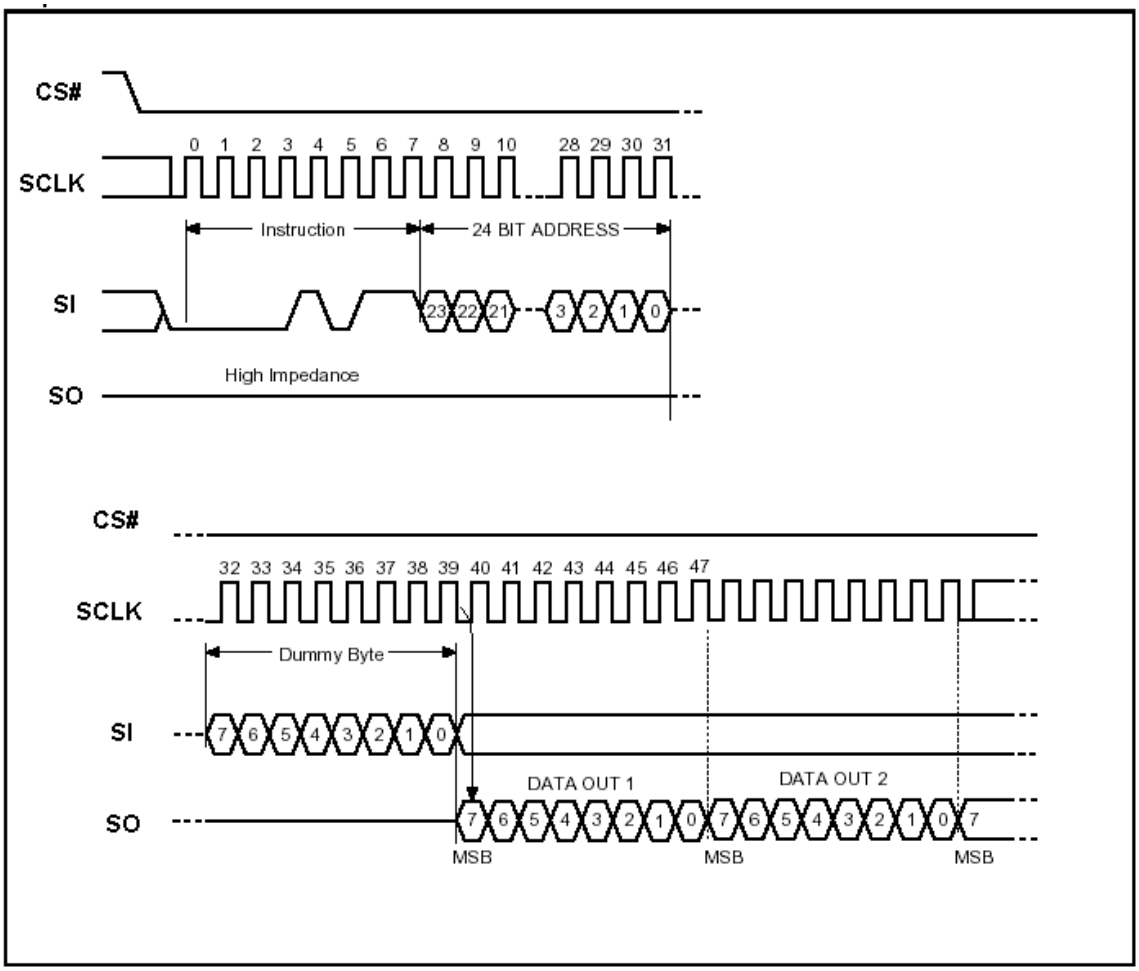


3.3 Read Data Bytes at Higher Speed

The High-Speed-Read instruction supporting up to 30 MHz is initiated by executing an 8-bit command, 0BH, followed by address bits [A23-A0] and a dummy byte. CE# must remain active low for the duration of the High-Speed-Read cycle.

Following a dummy byte (8 clocks input dummy cycle), the High-Speed-Read instruction outputs the data starting from the specified address location. The data output stream is continuous through all addresses until terminated by a low to high transition on CE#. The internal address pointer will automatically increment.

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



4 Electric characteristic

4.1 Absolute maximum rating

Symbol	Parameter	Min.	Max.	Unit	Condition
T_{OP}	Operating Temperature	-20	85	℃	In SPI mode
T_{OP}	Operating Temperature	-10	85	℃	In 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 Feature

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 characteristic

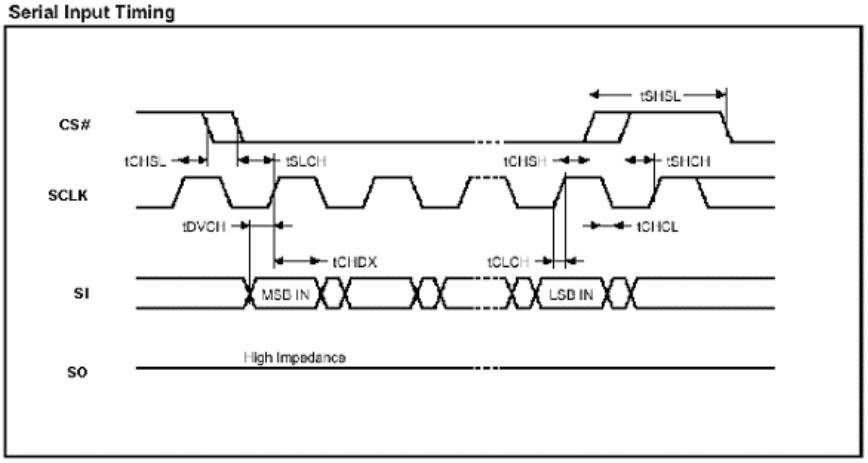
4.3.1 SPI bus AC characteristic

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

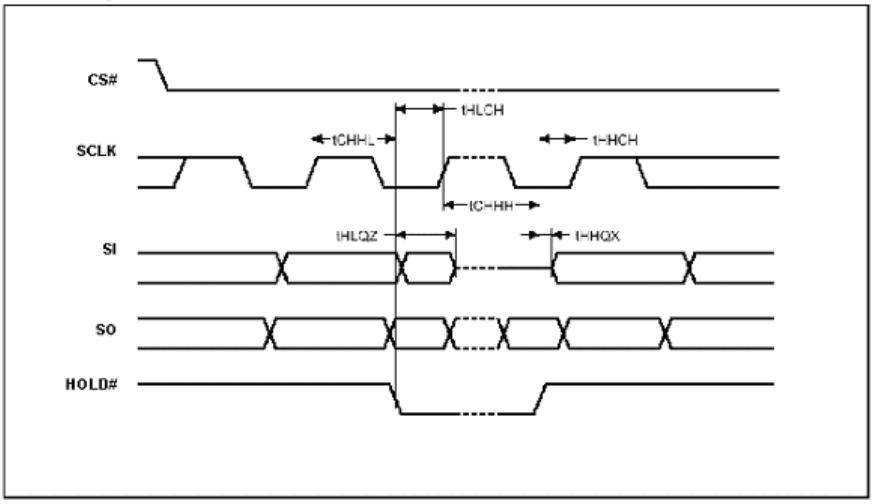
Symbol	Alt.	Parameter	Min.	Max.	Unit
Fc	Fc	Clock Frequency	D.C.	20	MHz
t_{CH}	t_{CLH}	Clock High Time	20		ns
t_{CL}	t_{CLL}	Clock Low Time	20		ns
t_{CLCH}		Clock Rise Time(peak to peak)	0.1		V/ns
t_{CHCL}		Clock Fall Time (peak to peak)	0.1		V/ns
t_{SLCH}	t_{CSS}	CS# Active Setup Time (relative to SCLK)	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}	t_{HO}	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}	t_{LZ}	HOLD to Output Low-Z		9	ns
t_{HLQZ}	t_{HZ}	HOLD# to Output High-Z		9	ns

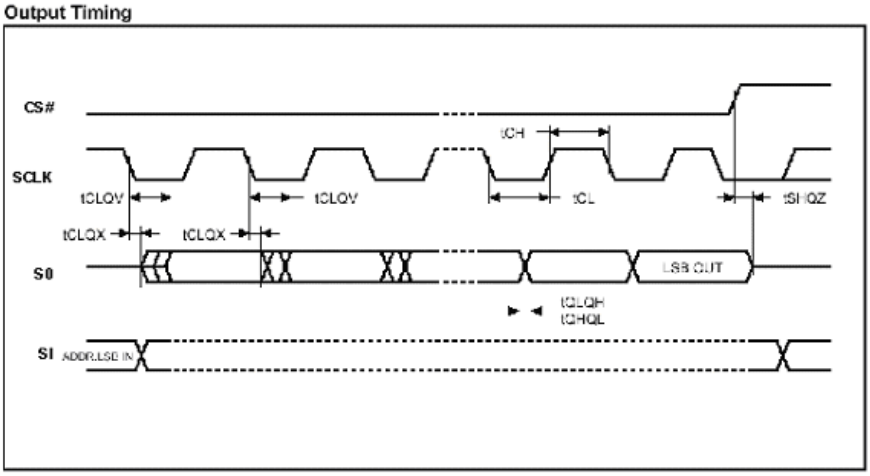
Serial Input Timing



Hold Timing

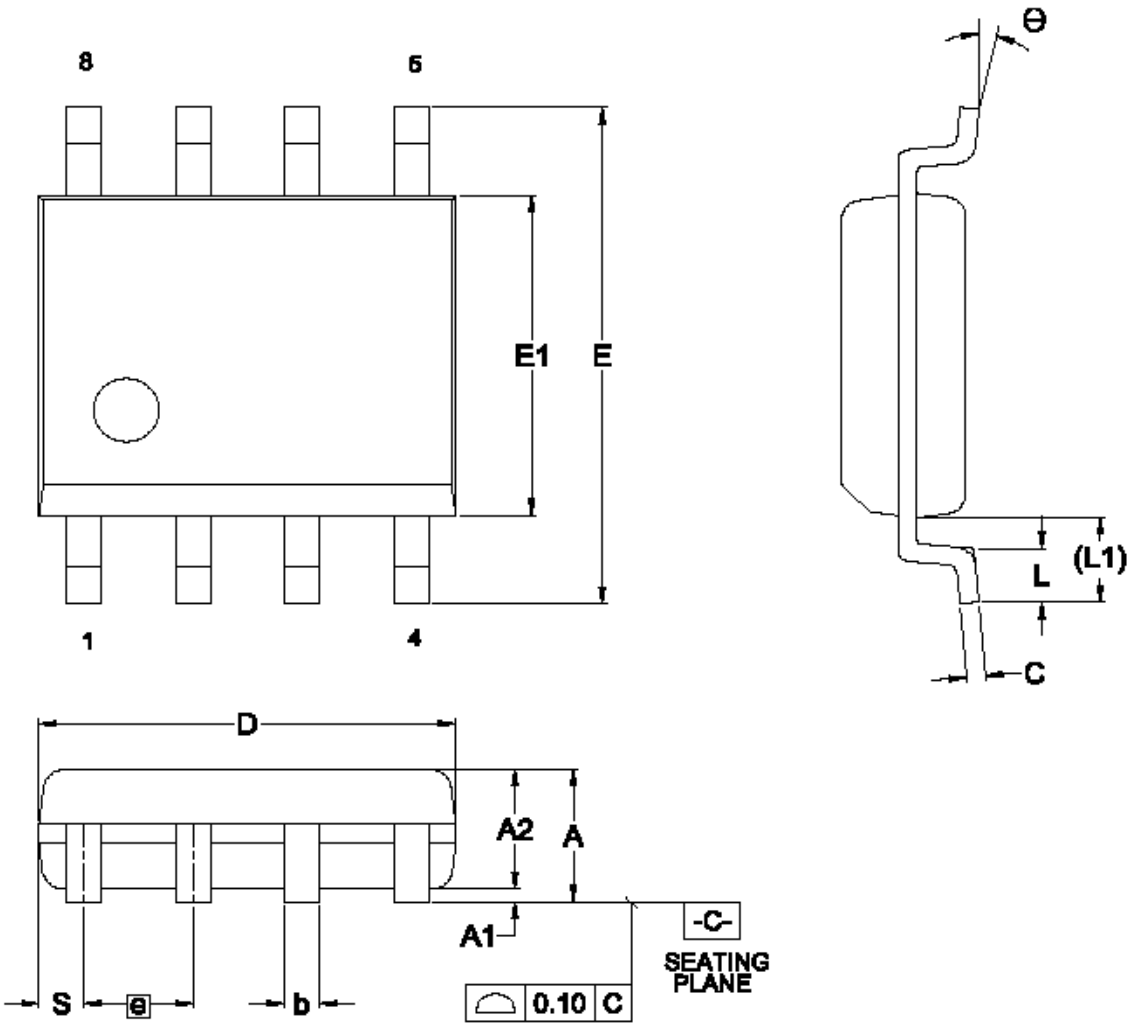


Output Timing



5 Package size： SOP-8

Unit: mm



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

		A	A1	A2	b	C	D	E	E1	⊖	L	L1	S	θ
Mm	Min.	-	0.10	1.35	0.36	0.15	4.77	5.80	3.60		0.46	0.65	0.41	0
	Norm.	-	0.15	1.45	0.41	0.20	4.90	5.99	3.90	1.27	0.66	1.05	0.54	5
	Max.	1.75	0.20	1.55	0.51	0.25	5.03	6.20	4.00		0.86	1.25	0.67	8
inch	Min.	-	0.004	0.053	0.014	0.006	0.188	0.228	0.150		0.018	0.033	0.016	0
	Norm.	-	0.006	0.057	0.016	0.008	0.193	0.236	0.154	0.050	0.026	0.041	0.021	5
	Max.	0.069	0.008	0.061	0.020	0.010	0.198	0.244	0.156		0.034	0.049	0.026	8

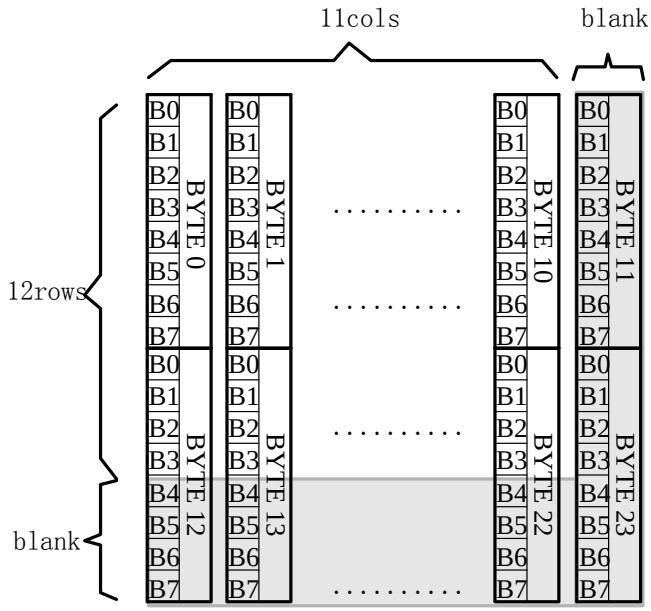
6 Font Read Method

6.1 Character dots arrangement

The data arrangement is byte horizontal, string horizontal. The highest Bit of BYTE represent left point, the lowest Bit of BYTE represent right point.

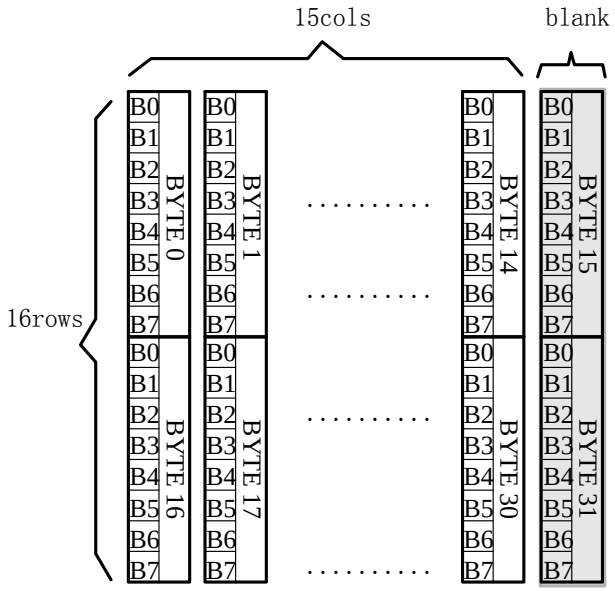
6.1.1 11X12 dots font

11X12 dots font has 24 bytes (BYTE 0 – BYTE 23)data。



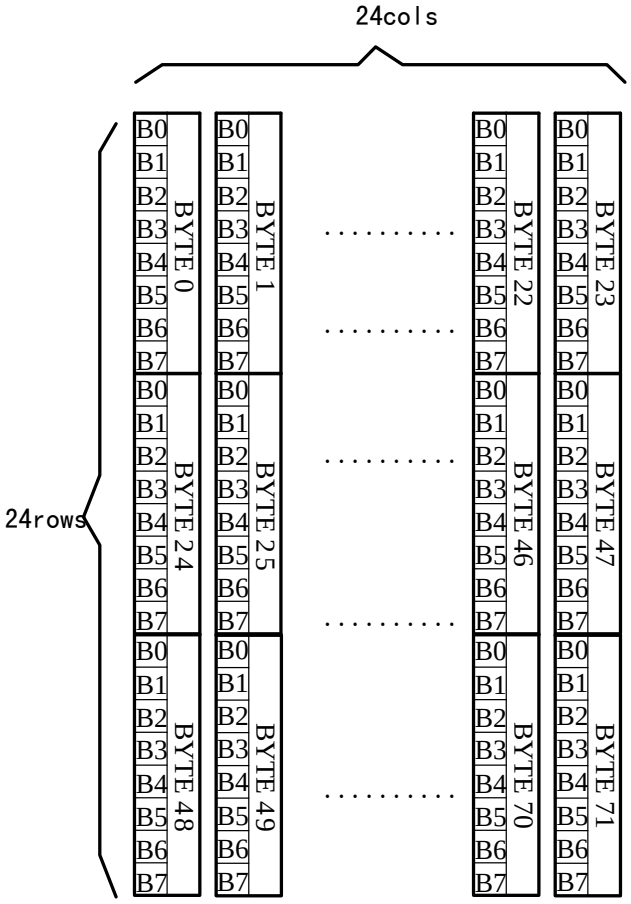
6.1.2 15X16 dots font

15X16 dots font has 32 bytes (BYTE 0 – BYTE 31)data。



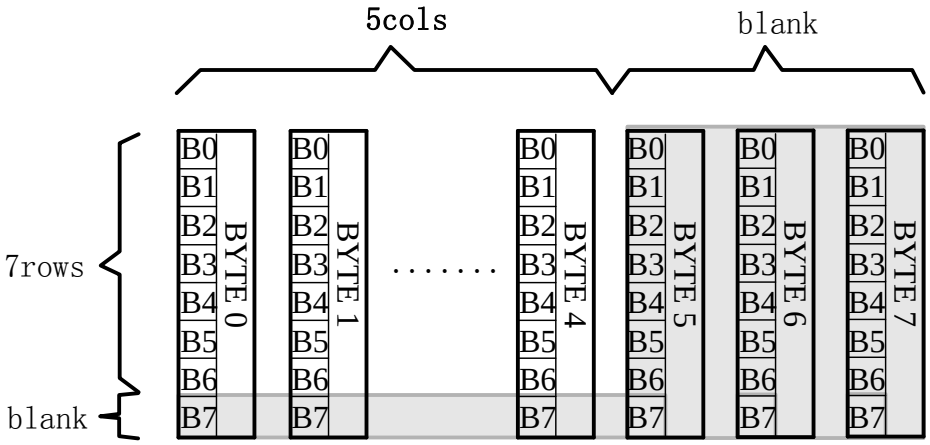
6.1.3 24X24 dots font

24X24 dots font has 72 bytes (BYTE 0 – BYTE 71)data。



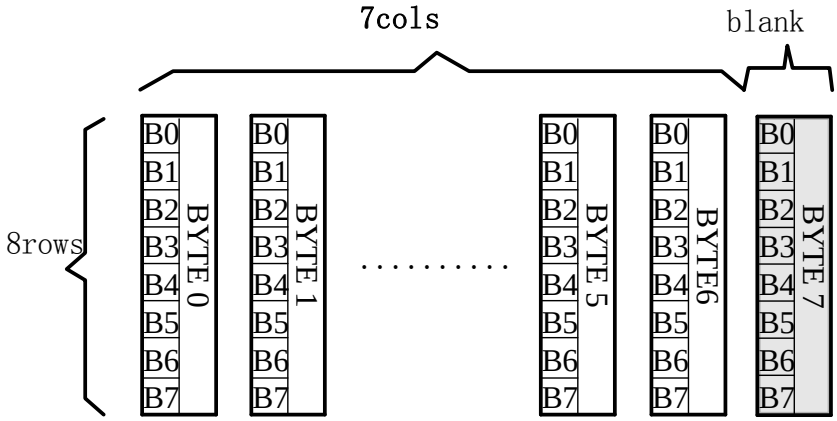
6.1.4 5X7 dots ASCII font

5X7 dots ASCII font has 8 bytes（BYTE 0 – BYTE7）data。



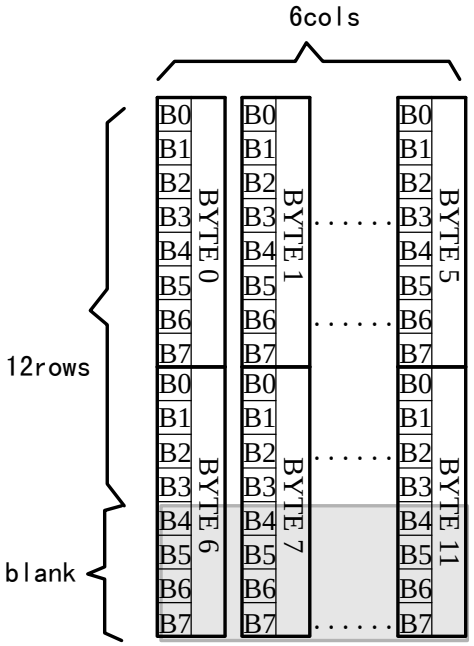
6.1.5 7X8 dots ASCII font

7X8 dots ASCII font has 8 bytes（BYTE 0 – BYTE7）data.



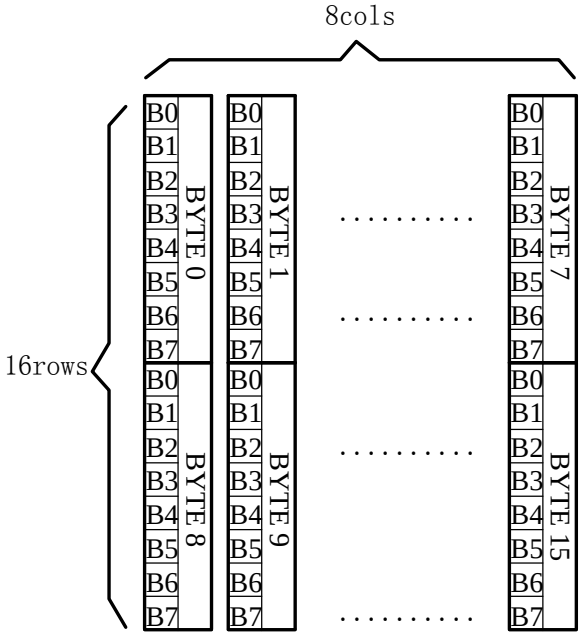
6.1.6 6X12 dots ASCII font

6X12 dots ASCII font has12 bytes（BYTE 0 – BYTE11）data.



6.1.7 8X16 dots font

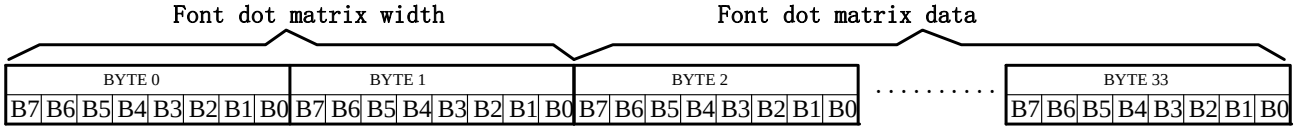
8X16 dots font has 16 bytes（BYTE 0 – BYTE15）data:



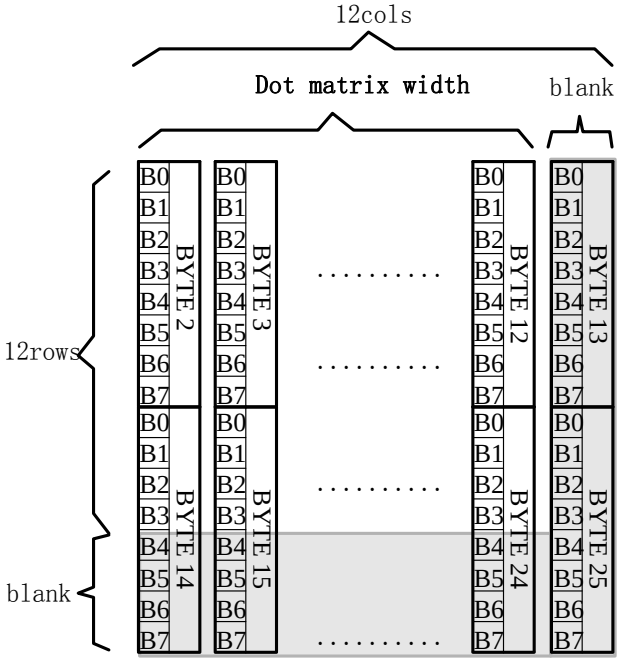
6.1.8 12 dots proportional arial font

12 dots proportional font has 26 bytes（BYTE 0 – BYTE25）data。

For the font is variant width, BYTE0~ BYTE1 are stored font width data,.BYTE2-25 are stored dots matrix data.



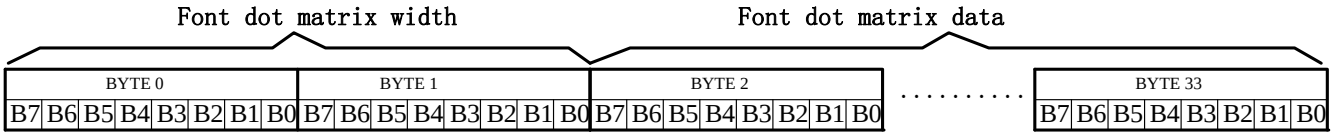
The dots matrix width of proportional font use BYTE as its unit. Different font width will reveal corresponding blanks. With the font's actual width data stored in BYTE0~BYTE 1, it can be used as reference for the position of the next word.



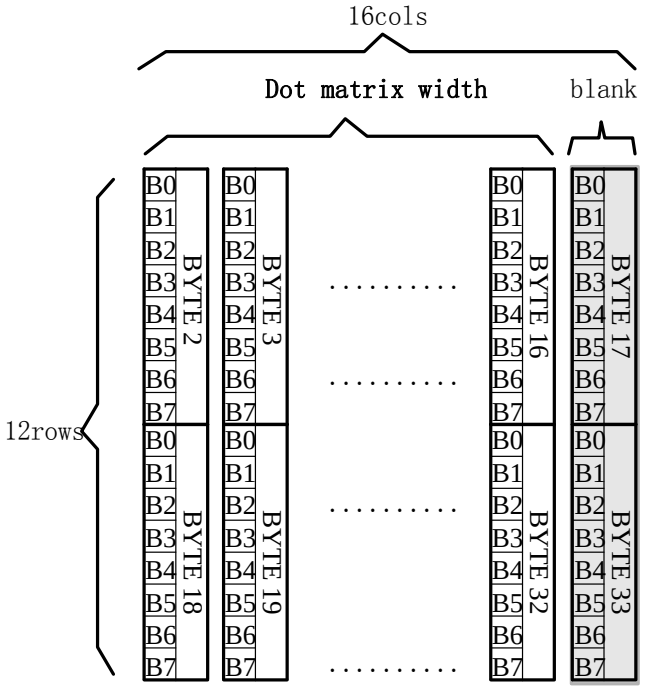
6.1.9 16 dots proportional arial font

16 dots proportional font has 34 bytes（BYTE 0 – BYTE33）data。

For the font is variant width, BYTE0~ BYTE1 are stored font width data,.BYTE2-33 are stored dots matrix data.



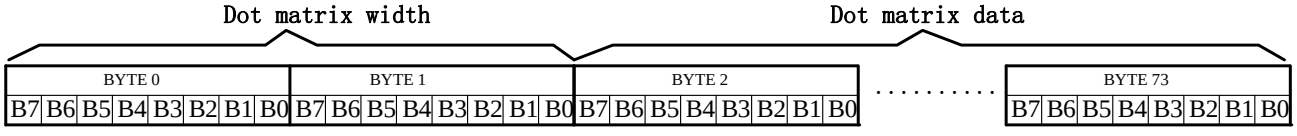
The dot matrix width of proportional font use BYTE as its unit. Different font width will reveal corresponding blanks. With the font's actual width data stored in BYTE0~BYTE 1, it can be used as reference for the position of the next word.



6.1.10 24 dots proportional font

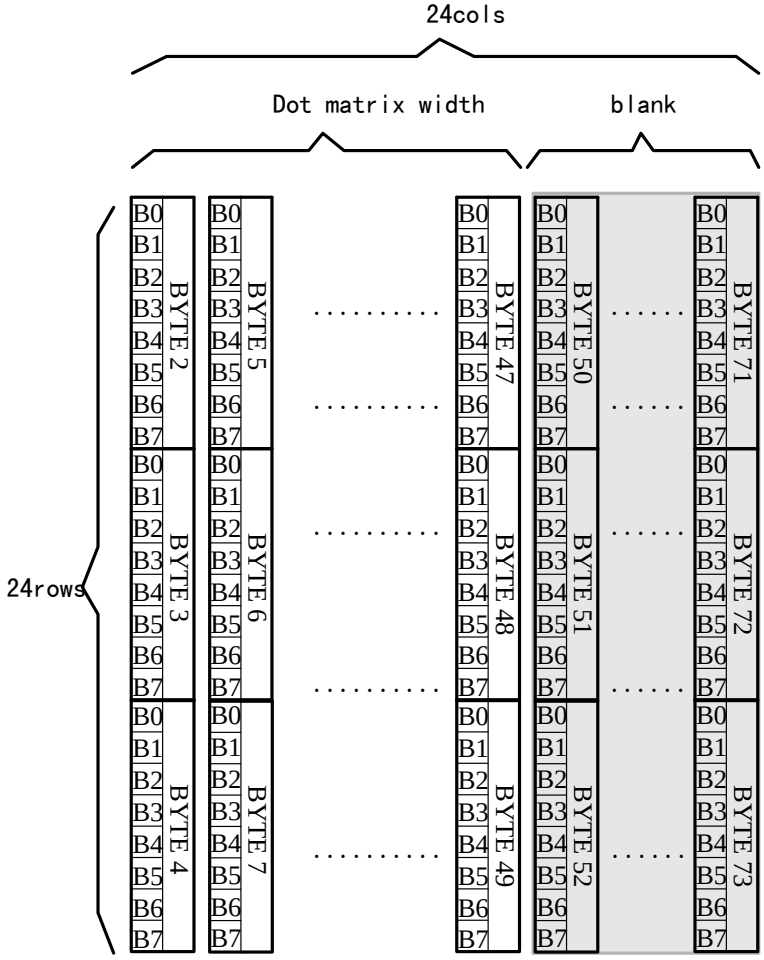
24 dots proportional font has 74 bytes（BYTE 0 – BYTE73）data。

For the font is variant width, BYTE0~ BYTE1 are stored font width data,.BYTE2-73 are stored dots matrix data.



The dot matrix width of proportional font use BYTE as its unit. Different font width will reveal

corresponding blanks. With the font s actual width data stored in BYTE0~BYTE 1, it can be used as reference for the position of the next word.



6.2 Dots font address table

	content	charset	Code scope	Characters	Address	Reference method
1	11X12 dots GB2312 font	GB2312	A1A1-F7FE	6763+846	00000	6.3.1.1
2	15X16 dots GB2312 font	GB2312	A1A1-F7FE	6763+846	5F4C0	6.3.1.2
3	24X24 dots GB2312 font	GB2312	A1A1-F7FE	6763+846	DE5C0	6.3.1.3
4	GB12345 index table	GB12345	A1A1-F9A9	6866+846	190958	6.3.1.4-6
5	BIG5 index table	BIG5	A140-F9DC	5401+408	193F06	6.3.1.7-9
6	Unicode index table	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 dots ASCII font	ASCII	20~7F	96	1A7C33	6.3.2.1
9	7X8 dots ASCII font	ASCII	20~7F	96	1A76CC	6.3.2.2
10	6X12 dots ASCII font	ASCII	20~7F	96	1A79CC	6.3.2.3
11	8X16 dots ASCII font	ASCII	20~7F	96	1A7FCC	6.3.2.4
12	12 dots Arial font	ASCII	20~7F	96	1A87CC	6.3.2.6
13	16 dots Arial font	ASCII	20~7F	96	1A918C	6.3.2.7
14	24 dots Arial font	ASCII	20~7F	96	1A9E4C	6.3.2.7
15	Input method code list				1ABA0C	
16	reserved				1FF6A4	

6.3 Calculation of character address

With certain calculation method, the user may obtain certain character dots address using character code.

6.3.1 Chinese font and Japanese font

6.3.1.1 11X12 dots GB2312 font

Parameters:

GBCode: character code.

MSB: high byte of GBCode.

LSB: low byte of GBCode.

Address: address of character data in chip.

BaseAdd: the base address of the font in chip

Calculation of character address:

```
BaseAdd=0x0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*24+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*24+ BaseAdd;
else if(MSB >= 0xa8 && MSB <= 0xa9 && LSB <= 0xa1)
    {
        if(LSB>0x7f)LSB--;
```

```
            Address = ((MSB-0xa8)*96 + (LSB-0x40)+94*5)*24+ BaseAdd;
        }
    else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
        Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*24+ BaseAdd;
```

6.3.1.2 15X16 dots GB2312 font

Parameters:

GBCode: character code.
MSB: high byte of GBCode.
LSB: low byte of GBCode.
Address: address of character data in chip.
BaseAdd: the base address of the font in chip

Calculation of character address:

```
    BaseAdd=0x5F4C0;
    if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
        Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
    else if(MSB ==0xA6 && LSB >=0xA1)
        Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*32+ BaseAdd;
    else if(MSB ==0xA9 && LSB >=0xA1)
        Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*32+ BaseAdd;
    else if(MSB >= 0xa8 && MSB <= 0xa9 && LSB <= 0xa1)
        {
            if(LSB>0x7f)LSB--;
            Address = ((MSB-0xa8)*96 + (LSB-0x40)+94*5)*32+ BaseAdd;
        }
    else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
        Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*32+ BaseAdd;
```

6.3.1.3 24X24 dots GB2312 font

Parameters:

GBCode: character code.
MSB: high byte of GBCode.
LSB: low byte of GBCode.
Address: address of character data in chip.
BaseAdd: the base address of the font in chip

Calculation of character address:

```
BaseAdd=0XDE5C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*72+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*4 )*72+ BaseAdd;
else if(MSB >= 0xa8 && MSB <= 0xa9 && LSB <= 0xa1)
    {
        if(LSB>0x7f)LSB--;
```

```
        Address = ((MSB-0xa8)*96 + (LSB-0x40)+94*5)*72+ BaseAdd;
    }
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 662)*72+ BaseAdd;
```

6.3.1.4 11X12 dots GB12345 font

Parameters:

FontCode: character code.

MSB: high byte of FontCode.

LSB: low byte fo FontCode.

Address: address of character data in chip.

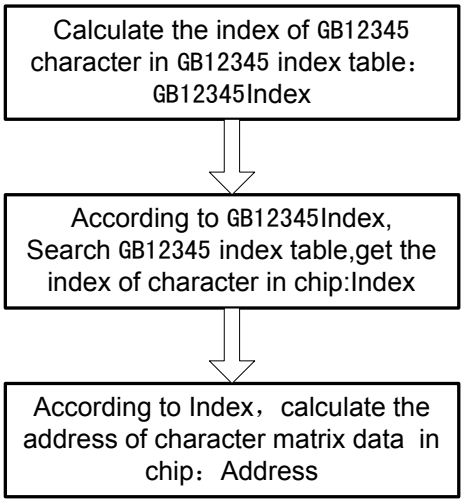
BaseAdd: the base address of the font in chip

GB12345Table: GB12345 index table. The table address is 0x190958

GB12345Index: index of the character in GB12345 index table.

Index: index of the character in font sets。

Calculation of character address:



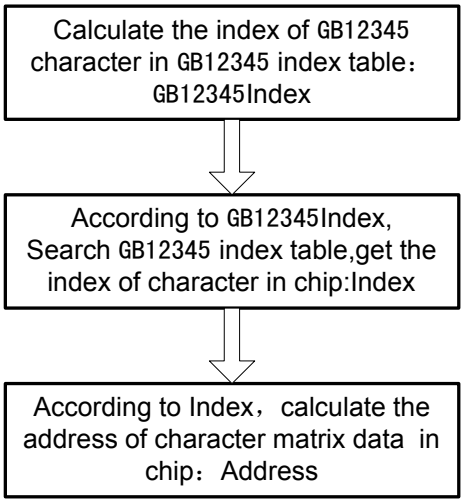

```
BaseAdd=0x00;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*24+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 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 dots GB12345 font

Parameters:

FontCode: character code.

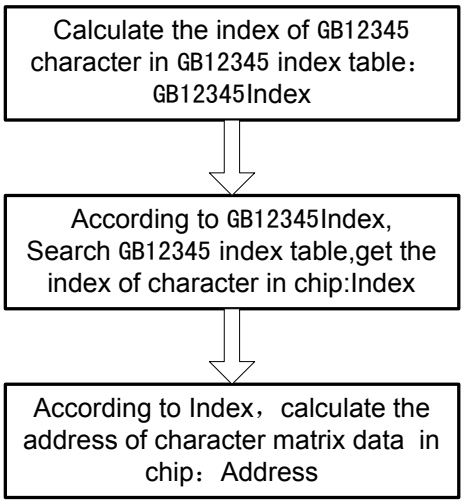
MSB: high byte of FontCode.
LSB: low byte fo FontCode.
Address: address of character data in chip.
BaseAdd: the base address of the font in chip
GB12345Table: GB12345 index table. The table address is 0x190958
GB12345Index: index of the character in GB12345 index table.
Index: index of the character in font sets。
Calculation of character address:




```
BaseAdd=0x5F4C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*32+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)
    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 dots GB12345 font

Parameters:
FontCode: character code.
MSB: high byte of FontCode.
LSB: low byte fo FontCode.
Address: address of character data in chip.
BaseAdd: the base address of the font in chip
GB12345Table: GB12345 index table. The table address is 0x190958
GB12345Index: index of the character in GB12345 index table.
Index: index of the character in font sets。
Calculation of character address:




```
BaseAdd=0XDE5C0;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*72+ BaseAdd;
else if(MSB ==0xA6 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1)-94*2 )*72+ BaseAdd;
else if(MSB ==0xA9 && LSB >=0xA1)
    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 dots BIG5 font

Parameters:

FontCode: character code.

MSB: high byte of FontCode.

LSB: low byte fo FontCode.

Address: address of character data in chip.

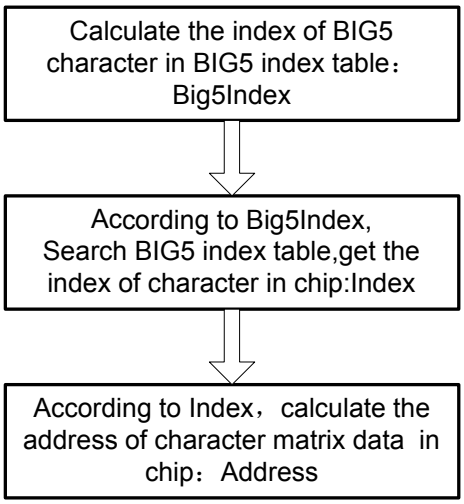
BaseAdd: the base address of the font in chip

Big5Table: BIG5index table. The table address is 0x193F06

Big5Index: index of the character in BIG5index table.

Index: index of the character in font sets。

Calculation of character address:




```
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 dots BIG5 font

Parameters:

FontCode: character code.

MSB: high byte of FontCode.

LSB: low byte fo FontCode.

Address: address of character data in chip.

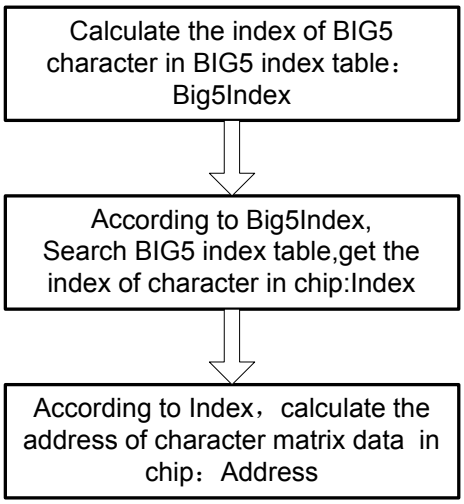
BaseAdd: the base address of the font in chip

Big5Table: BIG5index table. The table address is 0x193F06

Big5Index: index of the character in BIG5index table.

Index: index of the character in font sets。

Calculation of character address:




```
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 dots BIG5 font

Parameters:

FontCode: character code.

MSB: high byte of FontCode.

LSB: low byte fo FontCode.

Address: address of character data in chip.

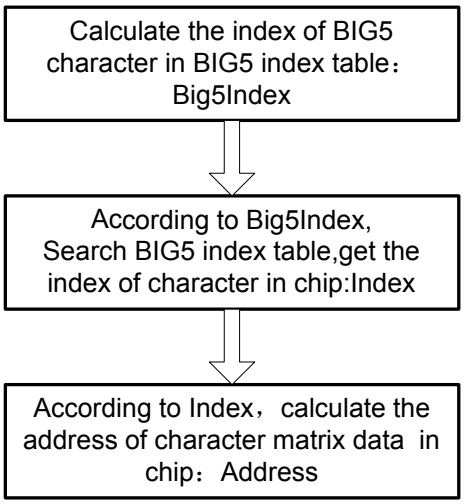
BaseAdd: the base address of the font in chip

Big5Table: BIG5index table. The table address is 0x193F06

Big5Index: index of the character in BIG5index table.

Index: index of the character in font sets。

Calculation of character address:




```
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 dots Unicode font

Parameters:

FontCode: character code.

Address: address of character data in chip.

BaseAdd: the base address of the font in chip

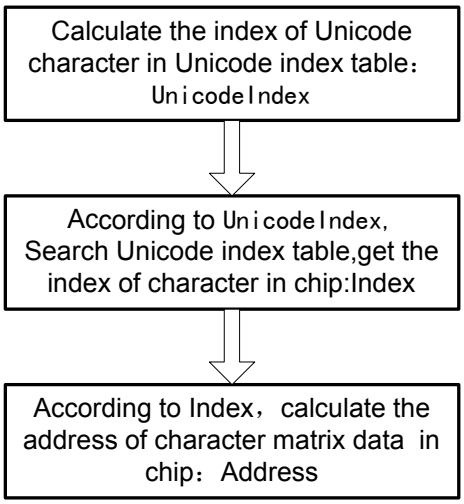
UnicodeTable: Unicode index table. The table address is 0x19AC30

UnicodeToIndex(): calculate index of the character in Unocode index table.

UnicodeIndex: index of the character in JIS index table.

Index：index of the character in font sets

Calculation of character address:




```
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;
```

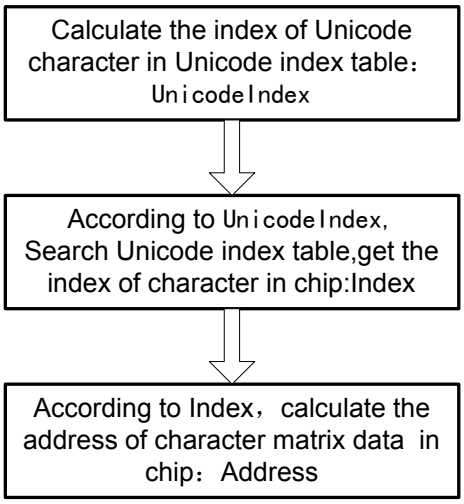
```
    else if(code<=0xffe5) h=code-160-7102-2493-6698-18374-4295-1030-149-129;
    else result=1;

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

    return h;
}
```

6.3.1.11 15X16 dots Unicode font

Parameters:

FontCode: character code.
Address: address of character data in chip.
BaseAdd: the base address of the font in chip
UnicodeTable: Unicode index table. The table address is 0x19AC30
UnicodeToIndex(): calculate index of the character in Unocode index table.
UnicodeIndex: index of the character in JIS index table.
Index：index of the character in font sets

Calculation of character address:




```
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 dots Unicode font

Parameters:

FontCode: character code.

Address: address of character data in chip.

BaseAdd: the base address of the font in chip

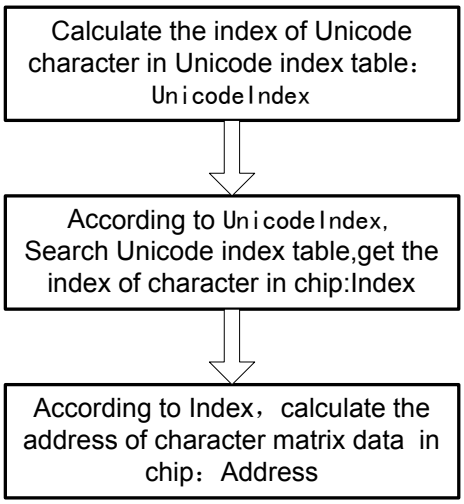
UnicodeTable: Unicode index table. The table address is 0x19AC30

UnicodeToIndex(): calculate index of the character in Unocode index table.

UnicodeIndex: index of the character in JIS index table.

Index：index of the character in font sets

Calculation of character address:




```
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>10138))
    {
        h = 0x3000-160-7102-2493;
    }
    return h;

}
```

6.3.1.13 GB12345 one-to-many Table

There are 94 codes in GB12345, each code has multi write methods.

GB2312	GB12345	GB2312	GB12345	GB2312	GB12345	GB2312	GB12345
摆	擺襬	胡	胡鬍	蒙	蒙濛矇	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	曲麯	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 one-to-many table struct：

```
Struct GB12345_MultiTable
{
    WORD    incode;
    WORD index1;
    WORD index2;
    WORD index3;
```

```
    WORD index4;
}
```

6.3.2 ASCII font

6.3.2.1 5X7 5X7 dots ASCII font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A73CC

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode –0x20 ) * 8+BaseAdd
```

6.3.2.2 7X8 dots ASCII font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A76CC

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
        Address = (ASCIICode –0x20 ) * 8+BaseAdd
```

6.3.2.3 6X12 dots ASCII font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A79CC

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
        Address = (ASCIICode –0x20 ) * 12+BaseAdd
```

6.3.2.4 8X16 dots ASCII font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A7FCC

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode –0x20 ) * 16+BaseAdd
```

6.3.2.5 12 dots arial font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A87CC

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

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

6.3.2.6 16 dots arial font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A918C

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

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

6.3.2.7 24 dots arial font

Parameters:

ASCIICode: ASCII code(8 bits)

BaseAdd: the base address of the font in chip

Address: address of character data in chip.

Calculation of character address:

BaseAdd=0x1A9E4C

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

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

7 Appendix

7.1 Character of GB2312 (846 Non-Chinese characters)

Corresponding codes: A1A1~A9EF

GB2312 1 section

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	｛	｜	｝	￣	

GB2312　1 section

A4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		あ	あ	い	い	う	う	え	え	お	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	ウ	ウ	エ	エ	オ	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	'	°	ˋ	:	;	!	?
E	ˉ	ˇ	ˊ	ˇ	ˋ	ˇ	ˆ	ˆ	¬	¬	¬	¬	¬	¬	¬	¬
F	ˉ	ˇ	‖	¦	‖	⁞										

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																