

# SPECIFICATION FOR APPROVAL

## 产 品 说 明 书

T16032A (带中文字库)

---

**KE FEI YAN DISPLAY CO.,LTD**

## 目 录

- (一) 概述
- (二) 外形尺寸
- (三) 模块主要硬件构成说明
- (四) 模块的外部接口
- (五) 指令说明
- (六) 读写操作时序
- (七) 应用举例
- (八) 附录

### 一、概述

1. 液晶显示模块是 160×32 点阵的汉字图形型液晶显示模块，可显示汉字及图形  
内置 8192 个中文汉字（16X16 点阵）、128 个字符（8X16 点阵）及 64X256 点  
显示 RAM（GDRAM）。可与 CPU 直接接口，提供两种界面来连接微处理机：8-位  
行及串行两种连接方式。具有多种功能：光标显示、画面移位、睡眠模式等。
2. 外观尺寸：
3. 116×37×13.0mm
4. 视域尺寸：83X18.6mm

## 二、外形尺寸图

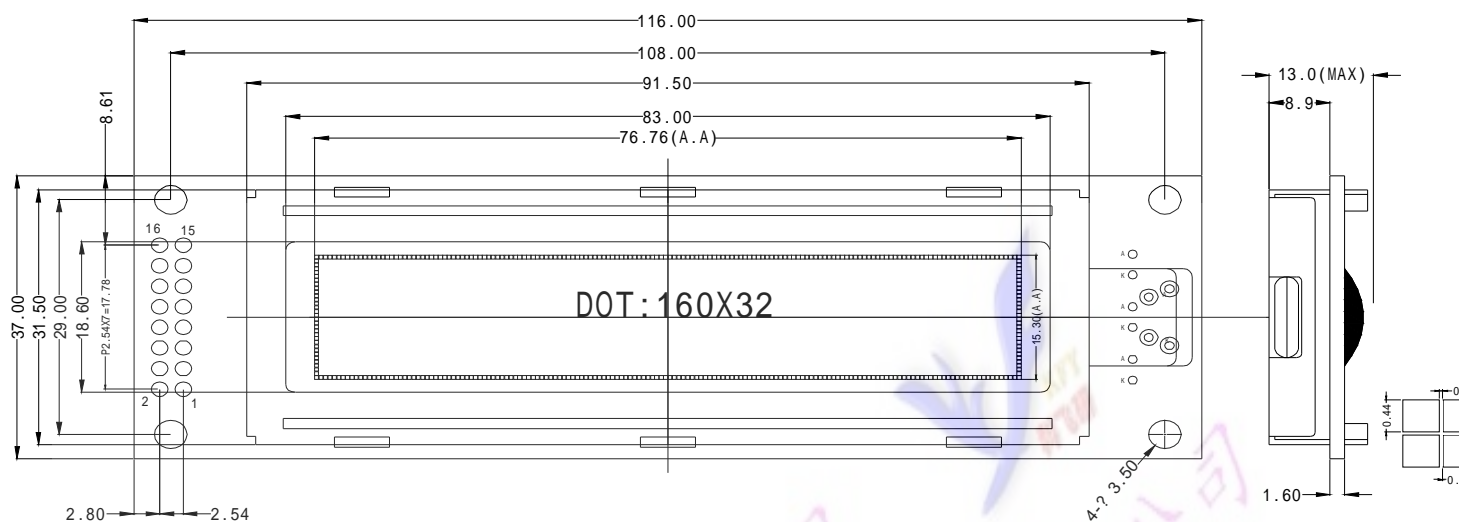
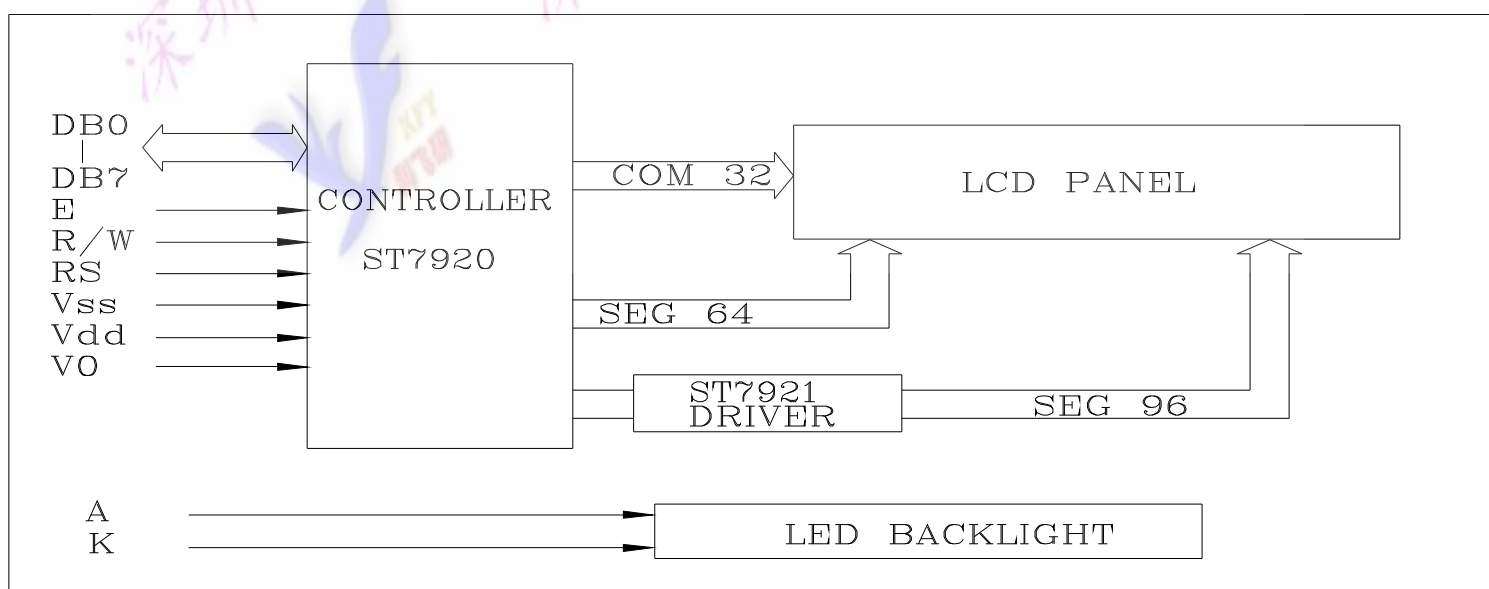


图 1

## 外形尺寸

ITEM	NOMINAL DIMEN	UNIT
模块体积	116×35×13.0	mm
视域	83X18.6	mm
行列点阵数	160X32	dots
点距离	0.04×0.04	mm
点大小	0.44X0.44	mm

## 三. 模块主要硬件构成说明

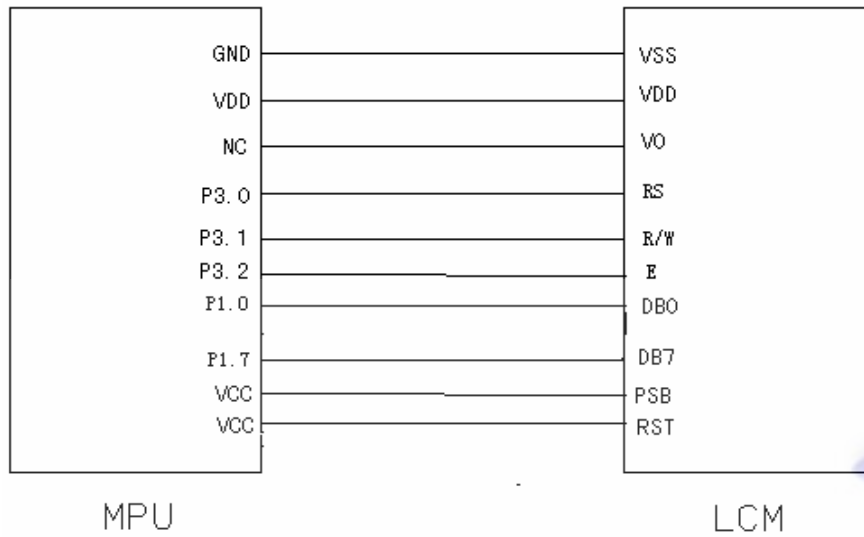


## T16032A 引脚说明

引脚	名称	方向	说明	引脚	名称	方向	说明
1	VSS	-	GND (0V)	11	DB4	I	数据 4
2	VDD	-	Supply Voltage For Logic (+5V)	12	DB5	I	数据 5
3	V0	-	Supply Voltage For LCD	13	DB6	I	数据 6
4	RS (CS)	0	H: Data L: Instruction Code	14	DB7	I	数据 7
5	R/W (SID)	0	H: Read L: Write	15	BLA	I	背光源正极 (LED+5V)
6	E (SCLK)	0	Enable Signal	16	VEE	I	NEGATIVE
7	DB0	I	数据 0				
8	DB1	I	数据 1			-	
9	DB2	I	数据 2			-	
10	DB3	I	数据 3				

## 四、模块的外部接口

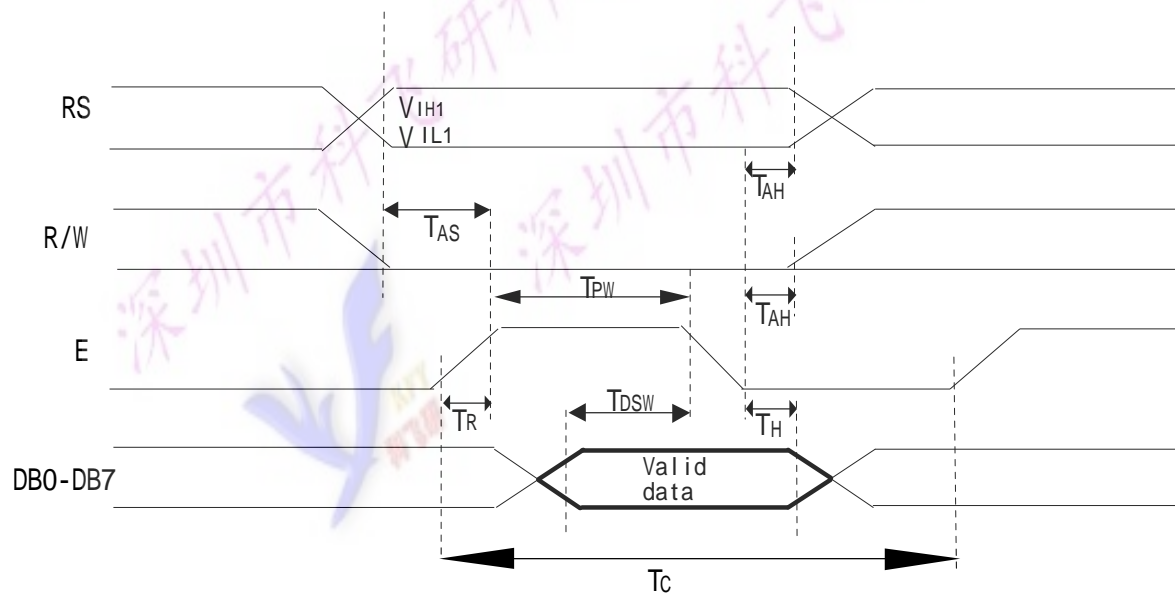
- 1、逻辑工作电压(VDD): 4.5~5.5V
- 2、电源地(GND): 0V
- 3、工作温度(Ta): -20~75℃ (宽温)
- 4、电气特性见附图 1 外部连接图 (参考附图 2)



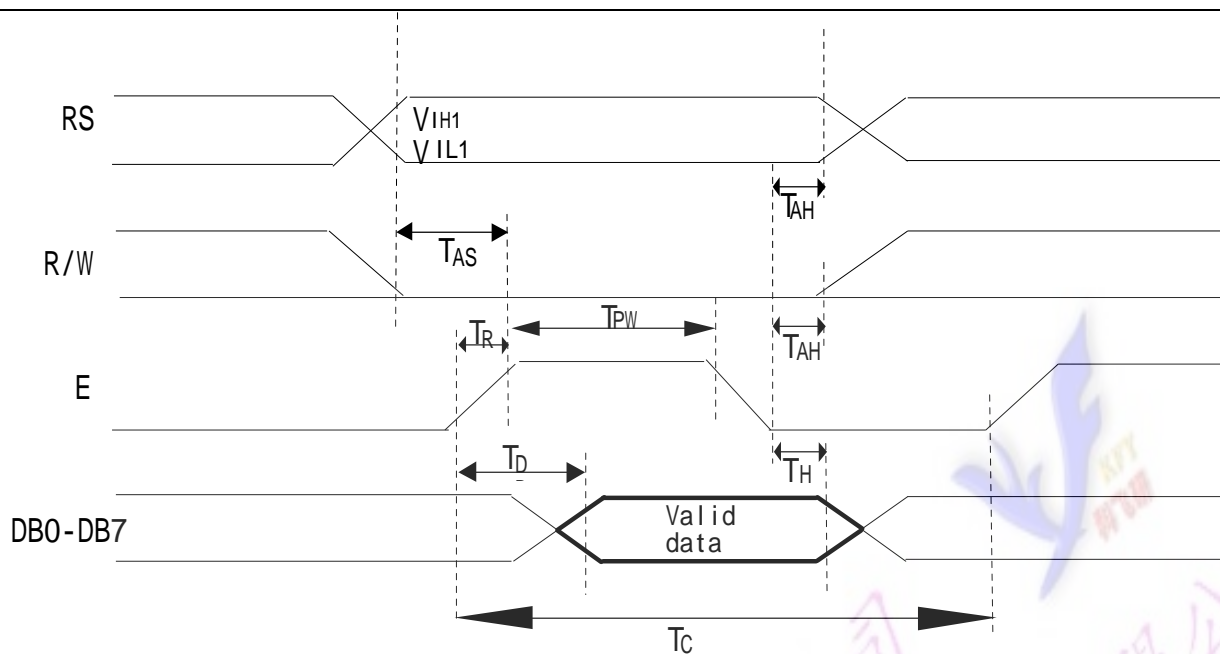
模块有并行和串行两种连接方法（时序如下）：

### 1、8 位并行连接时序图

MPU 写资料到模块



MPU 从模块读出资料



## 2、串行连接时序图

### 五、指令说明

模块控制芯片提供两套控制命令，基本指令和扩充指令如下：

指令表 1：(RE=0：基本指令)

指令	指令码										功能
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	
清除显示	0	0	0	0	0	0	0	0	0	1	将 DDRAM 填满"20H",并且设定 DDRAM 的地址计数器 (AC) 到 "00H"
地址归位	0	0	0	0	0	0	0	0	1	X	设定 DDRAM 的地址计数器(AC)到 "00H",并且将游标移到开头原点位置;这个指令不改变 DDRAM 的内容
显示状态开/关	0	0	0	0	0	0	1	D	C	B	D=1: 整体显示 ON C=1: 游标 ON B=1: 游标位置反白允许
进入点设定	0	0	0	0	0	0	0	1	I/D	S	指定在数据的读取与写入时,设定游标的移动方向及指定显示的移位
游标或示位制	0	0	0	0	0	1	S/C	R/L	X	X	设定游标的移动与显示的移位控制位;这个指令不改变 DDRAM 的内容

功能设定	0	0	0	0	1	DL	X	RE	X	X	DL=0/1: 4/8 位数据 RE=1: 扩充指令操作 RE=0: 基本指令操作
设定 CGRAM 地址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	设定 CGRAM 地址
设定 DDRAM 地址	0	0	1	0	AC5	AC4	AC3	AC2	AC1	AC0	设定 DDRAM 地址 (显示位址) 第一行: 80H—87H 第二行: 90H—97H
读取忙标志和地址	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	读取忙标志 (BF) 可以确认内部动作是否完成, 同时可以读出地址计数器 (AC) 的值
写数据到 RAM	1	0	数据								将数据 D7~D0 写入到内部的 RAM (DDRAM/CGRAM/IRAM/GRAM)
读出 RAM 的值	1	1	数据								从内部 RAM 读取数据 D7~D0 (DDRAM/CGRAM/IRAM/GRAM)

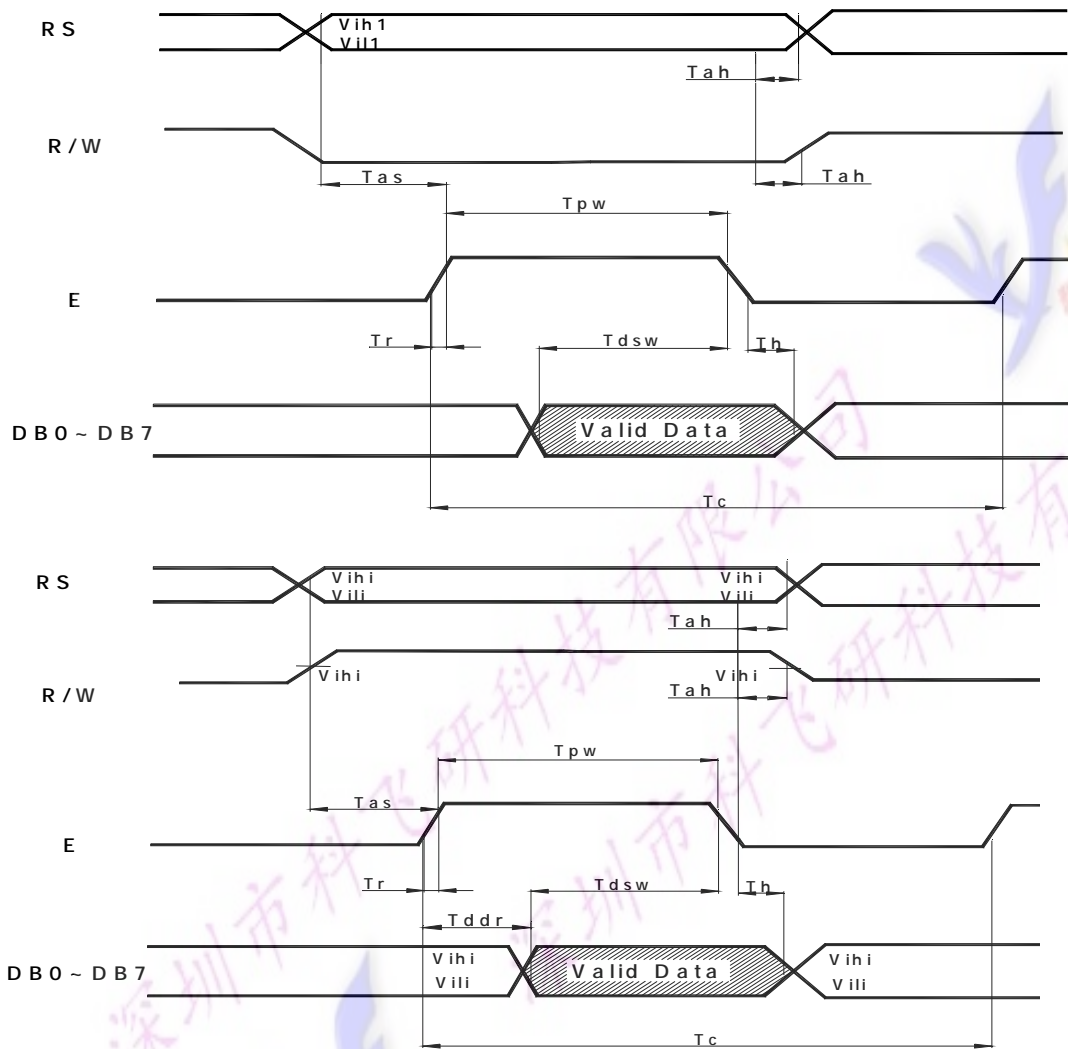
指令表 2: (RE=1: 扩充指令)

指令	指令码										功能
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	
待命模式	0	0	0	0	0	0	0	0	0	1	进入待命模式, 执行其他指令都裸终止待命模式
卷动地址开关开启	0	0	0	0	0	0	0	0	1	SR	SR=1: 允许输入垂直卷动地址 SR=0: 允许输入 IRAM 和 CGRAM 地址
反白选择	0	0	0	0	0	0	0	1	R1	R0	选择 2 行中的任一行作反白显示, 并可决定反白与否。初始值 R1R0=00, 第一次设定为反白显示, 再次设定变回正常
睡眠模式	0	0	0	0	0	0	1	SL	X	X	SL=0: 进入睡眠模式 SL=1: 脱离睡眠模式
扩充功能设定	0	0	0	0	1	CL	X	RE	G	0	CL=0/1: 4/8 位数据 RE=1: 扩充指令操作 RE=0: 基本指令操作 G=1/0: 绘图开关
设定绘图 RAM 地址	0	0	1	0	0	0	AC3	AC2	AC1	AC0	设定绘图 RAM 先设定垂直 (列) 地址 AC6AC5...AC0 再设定水平 (行) 地址 AC3AC2AC1AC0 将以上 16 位地址连续写入即可

备注:当 IC1 在接受指令前,微处理器必须先确认其内部处于非忙碌状态,即读取 BF 标志时,需为零,方可接受新的指令;如果在送出一个指令前并不检查 BF 标志,那么在前一个指令和这指令中间必须延长一段较长的时间,即是等待前一个指令确实执行完成。

## 六. 时序图

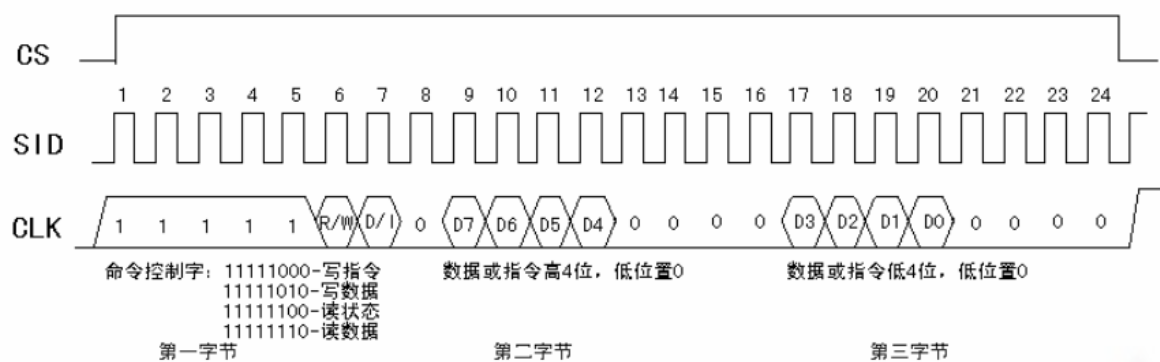
并口读写时序图:



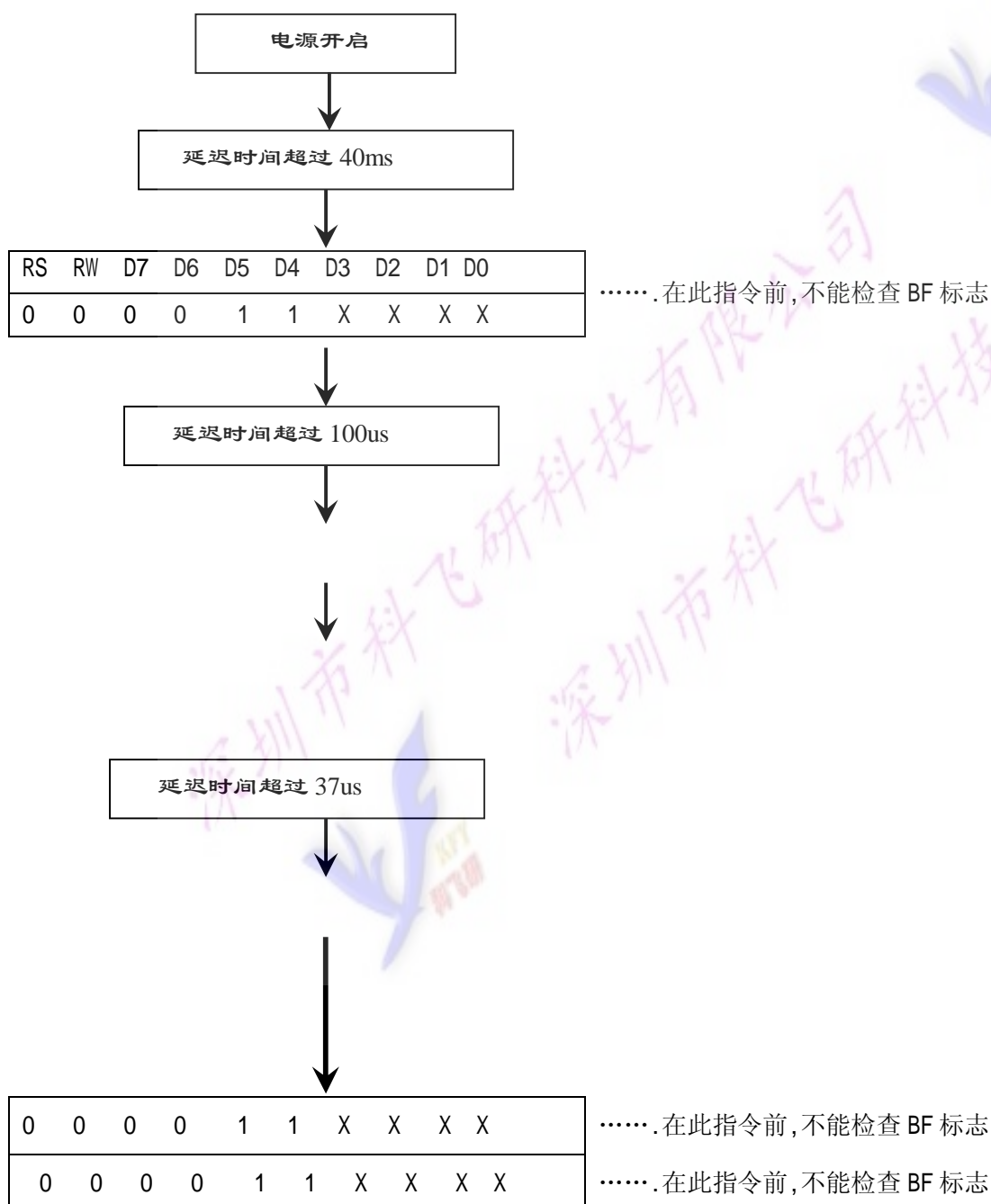
MPU 写数据  
MPU 读数据



## 串口读写时序:



## 七、软件初始化:



0	0	0	0	1	1	1	1	x	x	..... 在此指令前,不能检查 BF 标志
0	0	0	0	1	1	1	0	0	0	.... 设为基本指令
0	0	0	0	0	0	0	0	0	1	.... 荧屏显示清除
0	0	0	0	0	0	0	1	I/D	S	.... 进入点设定

↓

初始设定结束

#### 八. 应用举例:

16032 与单片机 8031 的一种接口如图 5.所示

;This program is for 16032

```
; RS-----P3.3
; R/W-----P3.1
; E-----P3.0
```

```
; DB0~7-----P1
```

```
DI      EQU P3.3
RW      EQU P3.1
E       EQU P3.0
```

```
ORG 0000H
AJMP  START
ORG 0003H
LCALL PAUSE
```

START:

```
MOV  IE,#81H      ;EXT. INTO PERMIT
MOV  IP,#01H     ;INTO IS FIRST INT. LEVEL
MOV  TCON,#00H   ;TIMER/COUNTER CONTROLER INIT.
```

```
mov  SP, #67h
```

```
LCALL DELAY
```

```
LCALL DELAY
```

```
LCALL SETUP
```

```
LCALL DEF_CHAR
```

```
MOV  A,#80H
```

```
LCALL WRITE_COM
```

```
MOV  R3,#8
```

TEST11:

```
MOV  DPTR,#CGRAM1 ;CGRAM TEST
```

```
LCALL WRITE_CGRAM
```

```
DJNZ R3,TEST11
```

```
MOV  A,#90H
```

```
LCALL WRITE_COM
```

```
MOV  R3,#8
```

TEST12:

```
MOV  DPTR,#CGRAM1
```

```
LCALL WRITE_CGRAM
DJNZ R3,TEST12
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
MOV A,#80H
LCALL WRITE_COM
MOV R3,#8
TEST21:
MOV DPTR,#CGRAM2
LCALL WRITE_CGRAM
DJNZ R3,TEST21
MOV A,#90H
LCALL WRITE_COM
MOV R3,#8
TEST22:
MOV DPTR,#CGRAM2
LCALL WRITE_CGRAM
DJNZ R3,TEST22
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
MOV A,#80H
LCALL WRITE_COM
MOV R3,#8
TEST31:
MOV DPTR,#CGRAM3
LCALL WRITE_CGRAM
DJNZ R3,TEST31
MOV A,#90H
LCALL WRITE_COM
MOV R3,#8
TEST32:
MOV DPTR,#CGRAM3
LCALL WRITE_CGRAM
DJNZ R3,TEST32
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
MOV A,#80H
LCALL WRITE_COM
MOV R3,#8
TEST41:
MOV DPTR,#CGRAM4
LCALL WRITE_CGRAM
DJNZ R3,TEST41
MOV A,#90H
```



深圳市科飞研科技有限公司  
深圳市科飞研科技有限公司

```
LCALL WRITE_COM
MOV R3,#8
TEST42:
MOV DPTR,#CGRAM4
LCALL WRITE_CGRAM
DJNZ R3,TEST42
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY

MOV A#80H ;WORD TEST
LCALL WRITE_COM
MOV DPTR,#CHINESE
LCALL WRITE_HZ
MOV A,#90H
LCALL WRITE_COM
MOV DPTR,#TABLE1
LCALL WRITE_ASCII
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
MOV A#80H
LCALL WRITE_COM
MOV DPTR,#table1
LCALL WRITE_ascii
MOV A,#90H
LCALL WRITE_COM
MOV DPTR,#chinese
LCALL WRITE_hz
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
AAA: LJMPL START

SETUP:
LCALL DELAY
LCALL DELAY
LCALL DELAY
MOV A,#01H ;CLEAR DISPLAY
LCALL WRITE_COM
MOV A,#00110000B ;FUNCTION SETTING
LCALL WRITE_COM
MOV A,#00000010B ;DDRAM SET TO '00H'
LCALL WRITE_COM
MOV A,#00000100B ;
LCALL WRITE_COM
MOV A,#00001100B ;DISPLAY ON
```

```
LCALL WRITE_COM
MOV A,#0000001B ;CLEARING SCREEN
LCALL WRITE_COM
MOV A,#1000000B ;SET DDRAM ADDRESS
LCALL WRITE_COM
RET
```

```
WRITE_COM: ;WRIT///cv
;WRITE COMMANDS TO ST7920
;INSTEAD OF CHECKING BF STATE
```

```
LCALL DELAY1
CLR RS
CLR RS
CLR RW
CLR RW
MOV P1,A
MOV P1,A
SETB E
SETB E
NOP
NOP
CLR E
CLR E
;LCALL DELAY1
RET
```

```
WRITE_DAT: ;WRITE DISPLAY DATAS TO ST79220
```

```
LCALL DELAY1
SETB RS
SETB RS
CLR RW
CLR RW
MOV P1,A
MOV P1,A
SETB E
SETB E
NOP
NOP
CLR E
CLR E
RET
```

```
DELAY1:
MOV R7,#010H
D11: MOV R6,#010H
DJNZ R6,$
DJNZ R7,D11
RET
```

```
DELAY:
MOV R1,#00H
D2: MOV R2,#00H
DJNZ R2,$
DJNZ R1,D2
RET
```

```
DEF_CHAR:
    ;WRITE TO CGRAM
    MOV A,#0100000B    ;SET CGRAM ADDRESS
    LCALL WRITE_COM
    MOV R3,#8
DEF1:
    MOV A,#000H
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    MOV A,#0FFH
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    DJNZ R3,DEF1
    MOV R3,#8
DEF2:
    MOV A,#0AAH
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    MOV A,#0AAH
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    DJNZ R3,DEF2
    MOV R3,#8
DEF3:
    MOV A,#055H
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    MOV A,#0AAH
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    DJNZ R3,DEF3
    mov R3,#8
DEF4:
    MOV A,#0FFH
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    LCALL WRITE_DAT
    DJNZ R3,DEF4
    RET
WRITE_ASCII:
    MOV R4,#16
DDDD:  CLR A
        MOVC A,@A+DPTR
        LCALL WRITE_DAT
        INC DPTR
        DJNZ R4,DDDD
        RET
WRITE_HZ:    ;WRITE 8 CHINESE TO LCD
    MOV R4,#8
DD:  CLR A
        MOVC A,@A+DPTR
```

```

INC DPTR
LCALL WRITE_DAT
CLR A
MOVC A,@A+DPTR
INC DPTR
LCALL WRITE_DAT
DJNZ R4,DD
RET

```

```

WRITE_CGRAM:          ;CGRAM TESTING
CLR A
MOVC A,@A+DPTR
LCALL WRITE_DAT
INC DPTR
CLR A
MOVC A,@A+DPTR
LCALL WRITE_DAT
RET
PAUSE: SETB P3.2          ;PAUSE KEY PROCESS
SETB P3.2
LCALL DELAY1
MOV C,P3.2
MOV C,P3.2
JNC PAUSE              ;CHECK KEY WAS PRESSED
PAUSE1: MOV C,P3.2
MOV C,P3.2
LCALL DELAY1
JC PAUSE1              ;CHECK KEY OPEN AFTER PRESSED
PAUSE2: SETB P3.2
SETB P3.2
LCALL DELAY1
MOV C,P3.2
MOV C,P3.2
JNC PAUSE2            ;CHECK KEY WAS PRESSED AGAIN
RETI

```

```

TABLE1:
; “这里是 16*8 点阵的字符代码”
CGRAM1: DB 000H,000H      ;这里是自造字符地址表
CGRAM2: DB 000H,002H
CGRAM3: DB 000H,004H
CGRAM4: DB 000H,006H
CHINESE:
; “这里是 16*16 点阵的汉字代码表”
END

```

以下为串口写指令和数据的子程序:

```

WRITE_COM:
LCALL DELAY1          ; INSTEAD OF CHECKING BF STATE
SETB CS
PUSH ACC
MOV R0,#8
MOV A,#11111000B

```

COMM1:

```
CLR C
RLC A
MOV SID,C
CLR CLK
SETB CLK
DJNZ R0,COMM1
POP ACC
MOV R5,A
ANL A,#0FOH
MOV R0,#8
```

COMM2: CLR C

```
RLC A
MOV SID,C
CLR CLK
SETB CLK
DJNZ R0,COMM2
MOV A,R5
SWAP A
ANL A,#0FOH
MOV R0,#8
```

COMM3: CLR C

```
RLC A
MOV SID,C
CLR CLK
SETB CLK
DJNZ R0,COMM3
CLR CS
```

RET

WRITE\_DAT:

```
LCALL DELAY1
SETB CS
PUSH ACC
MOV R0,#8
MOV A,#11111010B
```

DATA1: CLR C

```
RLC A
MOV SID,C
CLR CLK
SETB CLK
DJNZ R0,DATA1
POP ACC
MOV R5,A
ANL A,#0FOH
MOV R0,#8
```

DATA2: CLR C

```
RLC A
MOV SID,C
CLR CLK
SETB CLK
DJNZ R0,DATA2
MOV A,R5
SWAP A
```



```
ANL  A,#0F0H
MOV  RO,#8
DATA3:CLR C
      RLC  A
      MOV  SID,C
      CLR  CLK
      SETB CLK
      DJNZ RO,DATA3
      CLR  CS
RET
```

## 八、附录部分

### 附录 1: ASCII 码表

☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	♂	♀	♫	♫	✂
▶	◀	‡	!!	¶	§	—	‡	†	↓	→	←	⊥	⇄	▲	▼
□	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	:	<	=	>	?
Q	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	△

16x8 半寬字型符號表

## 附录 2: 汉字码址表







1160 律呂  
 1161 後  
 1162 律呂  
 1163 律呂  
 1164 律呂  
 1165 律呂  
 1166 律呂  
 1167 律呂  
 1168 律呂  
 1169 律呂  
 1170 律呂  
 1171 律呂  
 1172 律呂  
 1173 律呂  
 1174 律呂  
 1175 律呂  
 1176 律呂  
 1177 律呂  
 1178 律呂  
 1179 律呂  
 1180 律呂  
 1181 律呂  
 1182 律呂  
 1183 律呂  
 1184 律呂  
 1185 律呂  
 1186 律呂  
 1187 律呂  
 1188 律呂  
 1189 律呂  
 1190 律呂  
 1191 律呂  
 1192 律呂  
 1193 律呂  
 1194 律呂  
 1195 律呂  
 1196 律呂  
 1197 律呂  
 1198 律呂  
 1199 律呂  
 1200 律呂

1201 律呂  
 1202 律呂  
 1203 律呂  
 1204 律呂  
 1205 律呂  
 1206 律呂  
 1207 律呂  
 1208 律呂  
 1209 律呂  
 1210 律呂  
 1211 律呂  
 1212 律呂  
 1213 律呂  
 1214 律呂  
 1215 律呂  
 1216 律呂  
 1217 律呂  
 1218 律呂  
 1219 律呂  
 1220 律呂  
 1221 律呂  
 1222 律呂  
 1223 律呂  
 1224 律呂  
 1225 律呂  
 1226 律呂  
 1227 律呂  
 1228 律呂  
 1229 律呂  
 1230 律呂  
 1231 律呂  
 1232 律呂  
 1233 律呂  
 1234 律呂  
 1235 律呂  
 1236 律呂  
 1237 律呂  
 1238 律呂  
 1239 律呂  
 1240 律呂  
 1241 律呂  
 1242 律呂  
 1243 律呂  
 1244 律呂  
 1245 律呂  
 1246 律呂  
 1247 律呂  
 1248 律呂  
 1249 律呂  
 1250 律呂  
 1251 律呂  
 1252 律呂  
 1253 律呂  
 1254 律呂  
 1255 律呂  
 1256 律呂  
 1257 律呂  
 1258 律呂  
 1259 律呂  
 1260 律呂

