RAiO

RA8803
RA8822

双图层中文文字/图形

LCD 控制器

应 用 手 册

Version 2.6

July 7, 2007

RAiO Technology Inc.

改 版 说 明		
版 本	日 期	说 明
1.0	January 26, 2004	First Release Version
1.1	March 9, 2004	增加 9-23 灰阶显示
1.2	April, 12, 2004	增加 附录 B-3 Power 应用电路 增加 9-24 扩展模式显示
1.3	May 3, 2004	增加 4-1、4-2、8-2、附录 C、D 节 修改 2-3、5、7、8、8-1、9-24、附录 B 节
1.4	October 1, 2004	增加表 3-1：RA8803/8822 与驱动器 IC 的接口名称对照表 修改图 5-2：用 DAC 控制 LCD 亮度的应用电路及文字说明 增加图 8-2A：重置脚位 RST# 的时序 修改图 8-3：一般 RA8803/8822 电源开启或重置的流程图 修改表 8-2：基本的缓存器设定范例 增加图 9-21B : Key Scan 流程图 修改图 B-3, B-4 修改附录 G. 范例程序 – C51 增加附录 H. 字型与字码表(BIG-5)
2.0	January 20, 2005	修改附录 B-2 电源应用电路
2.1	March 4, 2005	修改附录 B-2 电源应用电路的图 B-4, B-5 增加附录 B-2-4 电路板的电源布局建议
2.2	March 11, 2005	增加附录 B-3 Frame 信号、图 B-7
2.3	April 22, 2005	修改表 8-2：基本的缓存器设定范例 移除附录 B-3 Frame 信号
2.4	August 4, 2005	修改第 5 章液晶显示器的亮度调整 增加图 5-4 REG[D0h]与 Iout 输出的对应曲线图 修改附录 B-2-1 电源结构与图 B-3
2.5	January 10, 2006	修改第 25 页: REG [D0h] LCD Contrast Control Register (LCCR)
2.6	July 7, 2007	修改第 2-3 节 4Bit/8Bit 的 MPU 界面 增加第 2-5 节 MPU 接口的范例 修改第 9-14 节中断(Interrupt) 与忙碌(Busy) 设定 修改第 9-19 节 Key Scan 应用 增加附录 B-3 用 8051 的片选模式应用电路

章 节 内 容 页 数

1. 简介

RA8803/8822 是一个双图层(Two Pages)中英文文字与绘图模式的点矩阵液晶显示(LCD)控制器，可支持最大 320x240 / 240x160 点的 LCD 面板。内建 512K Byte 的字型码，可以显示中文字型、数字符号、英日欧文等字母，使用者只要透过 MPU 对 RA8803/8822 写入中/英文字型码，就可以直接在 LCD 面板上显示中英文字型，而不需要透过 MPU 以绘图方式来处理中英文的显示。为了让使用者更加了解 RA8803/8822 的使用与其附加的许多软硬件功能，因此建立此应用手册供客户参考。

图 1-1 是 RA8803/8822 的系统接口图，我们将依据此系统接口图，在以下的几章分别做完整的界面介绍，同时会在每一个应用上举出图示与例题，让使用者了解硬件的连接状态。在第九章我们将对 RA8803/8822 所提供的功能做详细的说明与介绍，同样配合许多图示与例题让使用者在实际设计时能轻易上手。最后在目录内我们附了很多参考数据，如完整电路图、Demo Program 等等，为了配合此应用手册，希望使用者能与 RA8803/8822 的规格书一同参考，以其达到最快的学习效果。

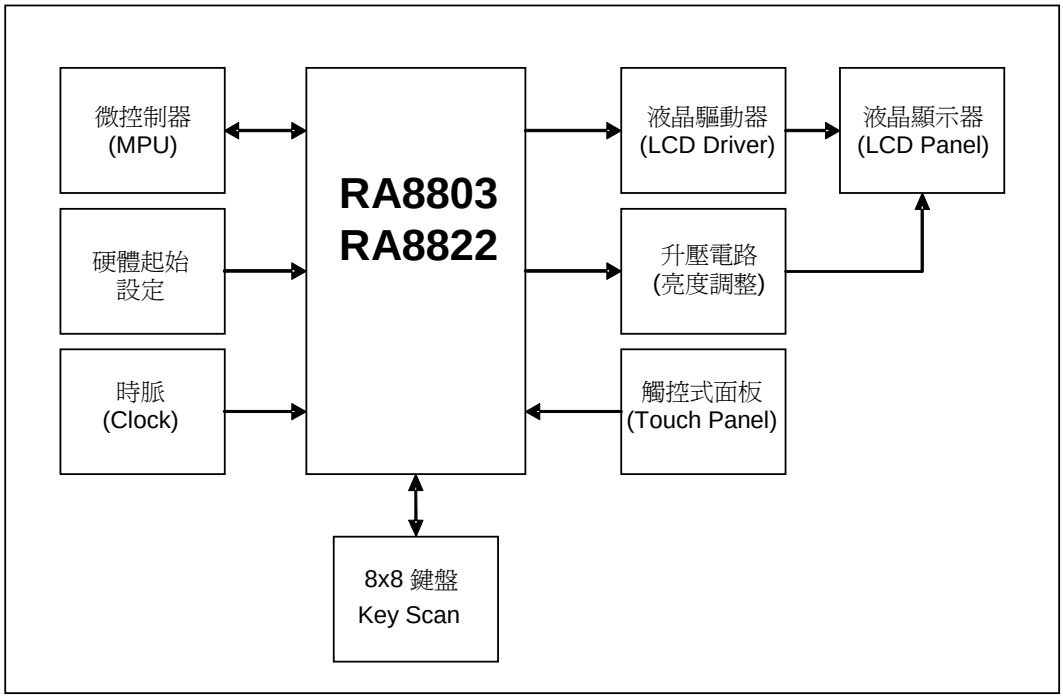


图 1-1：RA8803/8822 系统接口图

2. 微控制器(MPU)的接口

RA8803/8822 文字/图形 LCD 控制器与一般的 LCD 控制器相类似，都有支持 8080 和 6800 两大系列属性的 MPU 接口。使用者可以透过 SYS_MI 这根脚位去选择 RA8803/8822 的 MPU 接口是 8080 或者是 6800 的兼容系统，如果 SYS_MI 外接一 Pull Low 电阻，则 RA8803/8822 的 MPU 将定义成与 8080 兼容的接口。反之，如果 SYS_MI 外接一 Pull High 电阻，则 RA8803/8822 的 MPU 接口将定义成与 6800 兼容的接口。

2-1 8080 系列的 MPU 接口

图 2-1 是 RA8803/8822 与 8080 兼容系列的 MPU 接口示意图，此时 RA8803/8822 将只接受与 8080 系列兼容的 MPU 所传送出来的控制信号。

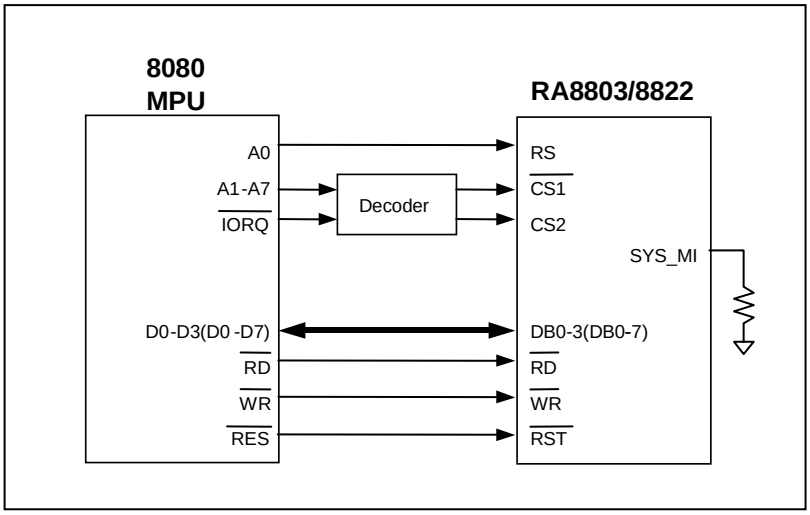


图 2-1：8080 (4/8-bit) MPU 与 RA8803/8822 的界面图

图 2-2 是 8080 系列 MPU 与 RA8803/8822 间的系统时序图，在 RA8803/8822 的定义中，RS 为 “L” 时是表示对缓存器下命令，也就是对 RA8803/8822 的缓存器进行读写的动作(Register Access Cycle)，而 RS 为 “H” 时是表示对 Display RAM 进行 Data 读写的动作(Data Access Cycle)。不论是 8080 或 6800， “RS” Pin 通常接到 MPU 的 Address Pin “A0”，8080 系列 MPU 与 6800 最大的不同是 Read、Write 的控制信号是分开的，RD 为 Low 时是进行读取动作，WR 为 Low 时是进行写入动作，至于读写的目的地则由 RS 决定。

下面图 2-2 表示如果是对缓存器进行读取动作，MPU 必须透过数据总线先送出缓存器的地址，然后才能在数据总线上读取缓存器的数据，如果是对缓存器进行写入动作，MPU 必须透过数据总线先送出缓存器的地址，然后再送出要写入的数据。当 8088 MPU 对 RA8803/8822 Display RAM 进行数据的读取动作，MPU 能直接在数据总线上读取 Display RAM 的数据，如果 8088 MPU 对 Display RAM 进行数据的

写入动作，MPU 则直接在数据总线上送出要写入的数据。

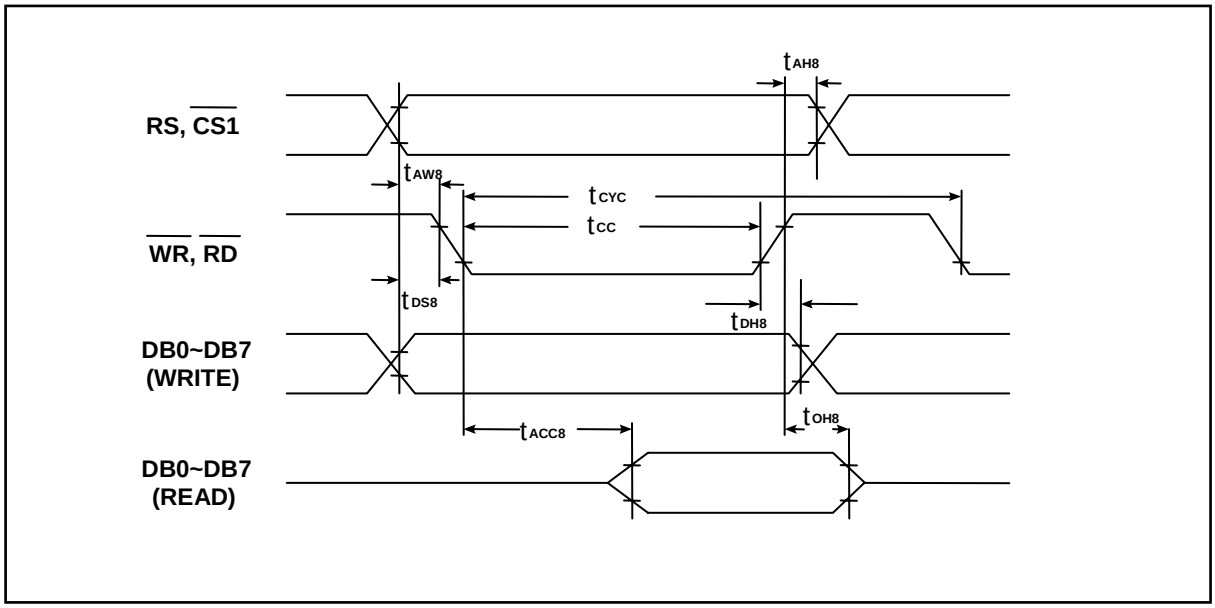


图 2-2：8-Bit 8080 MPU 对 RA8803/8822 缓存器/Data 进行读取/写入动作

表 2-1

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
RS, CS1#	t_{AH8}	Address hold time	10	--	ns	System Clock: 8MHz Voltage: 3.3V
	t_{Aw8}	Address setup time	63	--	ns	
WR#, RD#	t_{CYC}	System cycle time	800	--	ns	
	t_{CC}	Strobe pulse width	400	--	ns	
DB0 to DB7	t_{DS8}	Data setup time	63	--	ns	
	t_{DH8}	Data hold time	10	--	ns	
	t_{ACC8}	RD access time	--	330	ns	
	t_{OH8}	Output disable time	10	--	ns	

2-2 6800 系列的 MPU 接口

图 2-3 是 RA8803/8822 与 6800 兼容系列的 MPU 接口示意图，此时 RA8803/8822 将只接受 6800 系列兼容的 MPU 所传送出来的控制时序。6800 系列 MPU Read、Write 的控制信号是同一根 Pin，R/W# 为 High 时是进行读取动作，R/W# 为 Low 时是进行写入动作，而 EN 则是确定读写的动作是否有效(Enable)，至于读写的目的地仍由 RS 决定。

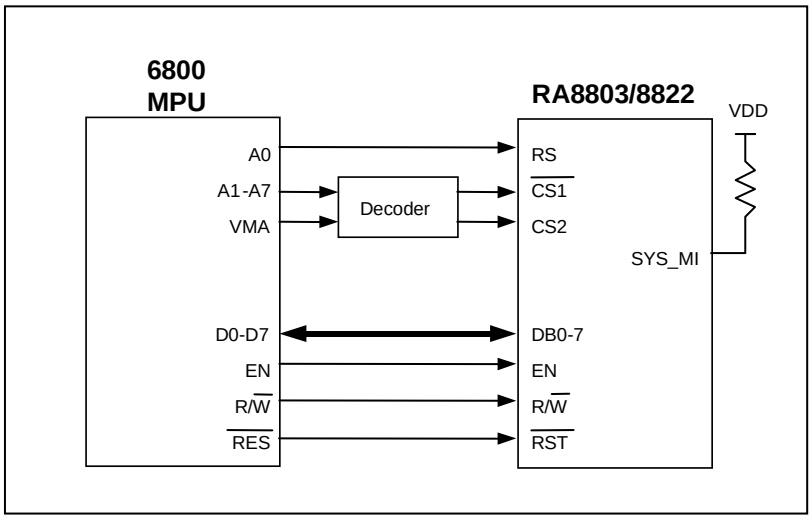


图 2-3：6800 (8-bit) MPU 与 RA8803/8822 的界面图

RA8803/8822 无法同时接受 6800 及 8080 的控制信号，因此在 MPU 的接口上，某些脚位上会因为使用者选择不同的 MPU 而有不同的定义，例如脚位 RD#(EN)，当使用者选择的 MPU 接口为 8080 时是定义成 RD#，而选择 6800 MPU 时是定义为 EN。而脚位 WR#(R/W#)，当使用者选择的 MPU 接口为 8080 时是定义成 WR#，而选择 6800 MPU 时是定义为 R/W#，对于 MPU 接口的脚位定义，使用者可以参考 RA8803/8822 规格书第 4-1 节的说明。

下面图 2-4 表示如果是 6800 MPU 对 RA8803/8822 缓存器进行读取动作，MPU 必须透过数据总线先送出缓存器的地址，然后才能在数据总线上读取缓存器的数据，如果是对缓存器进行写入动作，MPU 必须透过数据总线先送出缓存器的地址，然后再送出要写入的数据。当 6800 对 RA8803/8822 Display RAM 进行数据的读取动作，MPU 能直接在数据总线上读取 Display RAM 的数据，如果 6800 MPU 对 Display RAM 进行数据的写入动作，则 MPU 直接在数据总线上送出要写入的数据。

对于 6800 MPU 的接口，RA8803/8822 只提供 8Bit 的传输功能，而对于 8080 MPU 的接口，RA8803/8822 提供 4Bit 或 8Bit 的传输功能。

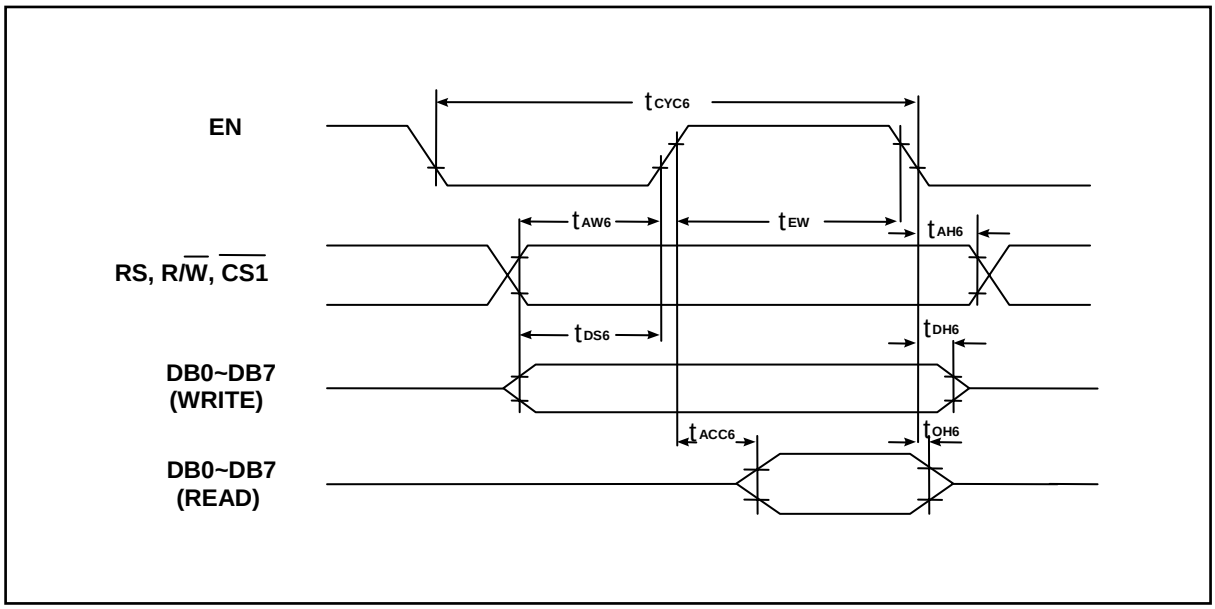


图 2-4：8-bit 6800 MPU 对 RA8803/8822 缓存器/Data 进行读取/写入动作

表 2-2

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
A0, R/W#, CS1#	t_{AH6}	Address hold time	10	--	ns	System Clock: 8MHz Voltage: 3.3V
	t_{Aw6}	Address setup time	63	--	ns	
	t_{CYC6}	System cycle time	800	--	ns	
DB0 to DB7	t_{DS6}	Data setup time	63	--	ns	
	t_{DH6}	Data hold time	10	--	ns	
	t_{ACC6}	Access time	--	330	ns	
	t_{OH6}	Output disable time	10	--	ns	
EN	t_{EW}	Enable pulse width	400	--	ns	

2-3 4Bit/8Bit 的 MPU 界面

对于 8080 MPU 的接口，RA8803/8822 提供 4Bit 或 8Bit 的传输功能。使用者可以透过 SYS_DB 这根脚位去选择 8080 MPU 的数据总线(Data Bus)接口，如果 SYS_DB 外接一 Pull Low 电阻，则 RA8803/8822 的 MPU 数据总线接口将定义成 4-Bit。反之，如果 SYS_DB 外接一 Pull High 电阻，则 RA8803/8822 的 MPU 数据总线接口将定为 8-Bit。因为 RA8803/8822 内部的缓存器大多是 8-Bit 的架构，因此如果使用 4-Bit 的数据总线接口，MPU 将会花较多的周期(Cycle)去存取 RA8803/8822 内部的缓存器。

当选择 4-bit MPU 作传输模式时，RA8803/8822 的 MPU 接口只有用到数据总线的 D3~D0，而没有用到的 D7~D4 必须接到 VDD 保持 High 准位，同时每一个八位的指令或数据将被分为两个 Nibble(4-Bit)依序透过数据总线的 D3~D0 进行传送，第一次先透过总线(DB3~DB0)传送数据的较高位 Bit[7..4]，第二次再透过总线(D3~D0)传送数据的较低位 Bit[3..0]，使用者可以参考 2-4 节中的例题 5~8。

不过对于 6800 MPU 的接口，RA8803/8822 只提供 8Bit 的传输功能，由于大部份使用者使用 8051 系统产品做系统开发，因此建议使用 8080 的 MPU 接口。

2-4 MPU 接口的程序范例

下面将列出一些简单的程序说明 MPU 与 RA8803/8822 存取缓存器的方式，这些程序及以后的范例都是以 8051 的汇编语言撰写，非常浅显易懂，也相当容易转成其它的语言格式。

表 2-3

No.	RS	6800	8080		DB0-DB7	Function
		R/W#	RD#	WR#		
①	1	1	0	1	xxh	Read Display Data
②	1	0	1	0	High Byte → Low Byte	Write Display Data (Character Mode--中文) 步骤②必须作两次，第一次写入中文字型码高字节数据，第二次再写入中文字型码低字节数据
③	1	0	1	0	xxh	Write Display Data (Character Mode--英文，ASCII) 步骤③只须作一次，直接写入英文字型码
④	1	0	1	0	xxh	Write Display Data (Graphic Mode)
⑤	0	0	1	0	Address	当要**读取状态**(Read Status)必须要完成两项步骤: 先步骤⑤ → 步骤⑥，才可以读取状态
⑥	0	1	0	1	Status	
⑦	0	0	1	0	Address	当要**写入控制命令到缓存器**必须要完成两项步骤: 先步骤⑦ → 步骤⑧，才可以写入控制命令到缓存器
⑧	0	0	1	0	Command	

例 题 1：8-Bit MPU 写入 Data 到 RA8803/8822 的缓存器

范例 1-1: REG [00h] = #CDH

```
MOV     A,#00h          ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#CDh          ; 写入“CD” 到 WLCR 缓存器
CALL    RegAddr_WRITE
```

范例 1-2: REG [E0h] = #5AH

```
MOV     A,#E0h          ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#5Ah          ; 写入“5A” 到 PNTR 缓存器
CALL    RegAddr_WRITE
```

例 题 2：8-Bit MPU 读取 RA8803/8822 的缓存器的 Data

范例 2-1: 读取 REG [00h]

```
MOV     A,#00h          ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read    ; 读取 WLCR 缓存器的值
```

范例 2-2: 读取 REG [E0h]

```
MOV     A,#E0h          ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read    ; 读取 PNTR 缓存器的值
```

例 题 3：8-Bit MPU 写入一中文到光标所在的位置

范例 3-1: LCD 显示”网”

```
MOV     A,#BAH          ; 加载 “网” 的中文码高位 “BA”
CALL    RegData_Write
MOV     A,#F4H          ; 加载 “网” 的中文码低位 “F4”
CALL    RegData_Write   ; 光标所在的位置将显示 “网” 的中文字
```

范例 3-2: LCD 显示”页”

```
MOV     A,#ADH          ; 加载 “页” 的中文码高位 “AD”
CALL    RegData_Write
MOV     A,#B6H          ; 加载 “页” 的中文码低位 “B6”
CALL    RegData_Write   ; 光标所在的位置将显示 “页” 的中文字
```

例 题 4：8-Bit MPU 读取 Display RAM 的 Data

```
CALL    RegData_Read    ; 读取光标所在的位置的 Display RAM Data
```

上面的例题 1~4 是 8-Bit 的 MPU 存取方式，如果是使用 4-Bit 的数据总线接口，MPU 将会花较多的 Cycle Time 去存取 RA8803/8822 内部的缓存器及 Display RAM Data，使用者可以比较一下例题 5~8 与例题 1~4 的差异性。

例 题 5：4-Bit MPU 写入 Data 到 RA8803/8822 的缓存器

范例 5-1: REG [00h] = #CDH

```
MOV     A,#00h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#0Ch              ; 写入“C” 到 WLCR 缓存器
CALL    RegAddr_WRITE
MOV     A,#0Dh              ; 写入“D” 到 WLCR 缓存器
CALL    RegAddr_WRITE
```

范例 5-2: REG [E0h] = #5AH

```
MOV     A,#0Eh              ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#00h              ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#05h              ; 写入“5” 到 PNTR 缓存器
CALL    RegAddr_WRITE
MOV     A,#0Ah              ; 写入“A” 到 PNTR 缓存器
CALL    RegAddr_WRITE
```

例 题 6：4-Bit MPU 读取 RA8803/8822 的缓存器的 Data

范例 6-1: 读取 REG [00h]

```
MOV     A,#00h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read        ; 读取 WLCR 缓存器的值(High Nibble)
  :
  :
CALL    RegAddr_Read        ; 读取 WLCR 缓存器的值(Low Nibble)
```

范例 6-2: 读取 REG [E0h]

```
MOV     A,#0Eh              ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#00h              ; 选择 Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read        ; 读取 PNTR 缓存器的值(High Nibble)
  :
  :
CALL    RegAddr_Read        ; 读取 PNTR 缓存器的值(Low Nibble)
```

例 题 7：4-Bit MPU 写入一中文到光标所在的位置

范例 7-1: LCD 显示"网"

```
MOV     A,#0BH          ; 加载 "网" 的中文码高位 "B"
CALL    RegData_Write
MOV     A,#0AH          ; 加载 "网" 的中文码高位 "A"
CALL    RegData_Write
MOV     A,#0FH          ; 加载 "网" 的中文码低位 "F"
CALL    RegData_Write
MOV     A,#04H          ; 加载 "网" 的中文码低位 "4"
CALL    RegData_Write   ; 光标所在的位置将显示 "网" 的中文字
```

范例 7-2: LCD 显示"页"

```
MOV     A,#0AH          ; 加载 "页" 的中文码高位 "A"
CALL    RegData_Write
MOV     A,#0DH          ; 加载 "页" 的中文码高位 "D"
CALL    RegData_Write
MOV     A,#0BH          ; 加载 "页" 的中文码低位 "B"
CALL    RegData_Write
MOV     A,#06H          ; 加载 "页" 的中文码低位 "6"
CALL    RegData_Write   ; 光标所在的位置将显示 "页" 的中文字
```

例 题 8：4-Bit MPU 读取 Display RAM 的 Data

```
CALL    RegData_Read    ; 读取光标所在的位置的 Display RAM Data(High Nibble)
  :
  :
CALL    RegData_Read    ; 读取光标所在的位置的 Display RAM Data(Low Nibble)
```

2-5 MPU 接口的范例

MPU 连接到 RA8803 或 RA8822 的方式有两种，本节以工业上常用的 8051 MPU(8080 接口)为例介绍两种方式，一种是用 8051 的 I/O 连接到 RA8803/8822，如下图 2-5 所示，设计人员再用软件控制 8051 的 I/O 以产生 8080 的接口信号，这种方式较常使用在液晶显示上，而完成一个对 RA8803/8822 读或写的动作需要好几个 8051 指令。

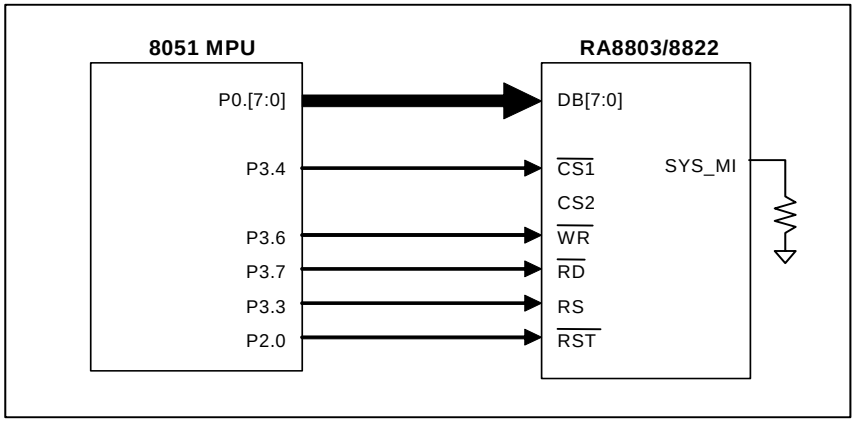


图 2-5：用 8051 的 I/O 连接图

另一种是用 8051 的片选模式(Fully Decode)，配合外部译码电路直接产生 Real 8080 的信号连接到 RA8803/8822，Fully Decode 是运用 8051 对外部内存存取的方法，当使用者的系统上有多个组件(Device)，可藉由外部内存间接寻址存取方式，来区别控制哪一个组件该动作，并被读取或写入数据，如下图 2-6。

图 2-6 中 RA8803/8822 的 CS# 与控制信号 RD#(Read)、WR#(Write)，透过 74LS32 作 OR 运算，目的是为了确保 RA8803/8822 读取或写入动作正常。

对外部数据存储器存取时会利用到 Port0 和 Port2 作为 16-bit 的 Data/Address Bus，并利用 ALE、WR、RD 作为控制讯号，进行外部 Code Memory 读取或写入，如下图 2-7。P0 端口首先送出待读写内存地址之低字节地址码(A0~A7)，随即配合地址闩锁效能(ALE)信号将此低字节地址码闩锁于外加之闩锁器(74LS373)上，然后 P0 端口马上又当成数据总线以便传递待读入/写出之数据。当 ALE 为高电位期间，地址闩锁器 74LS373 被致能，此时来自 P0 端口的信号 A0~A7 将会呈现在 74LS373 的输出端上。紧接着当 ALE 从高电位变成低电位时，74LS373 被禁能，此时会使得先前出现的地址信号 A0~A7 被锁住于 74LS373 输出端上，直到下一个 ALE 的高电位脉冲信号来临时才会改变。 74LS373 只有在 ALE 从高电位变成低电位，才能锁住地址信号于其输出端上。而将地址信号 A0~A7 锁住的理由是因为他们在整个读取周期中并非一直保持有效的缘故，而 A8~A15 由于在整个读取周期中一直保持有效，所以就不

必闩锁，当送出指定的地址确定后，MPU 会令 $\overline{WR}$ (P16)脚或 $\overline{RD}$ (P17)脚送出相对应的控制信号。

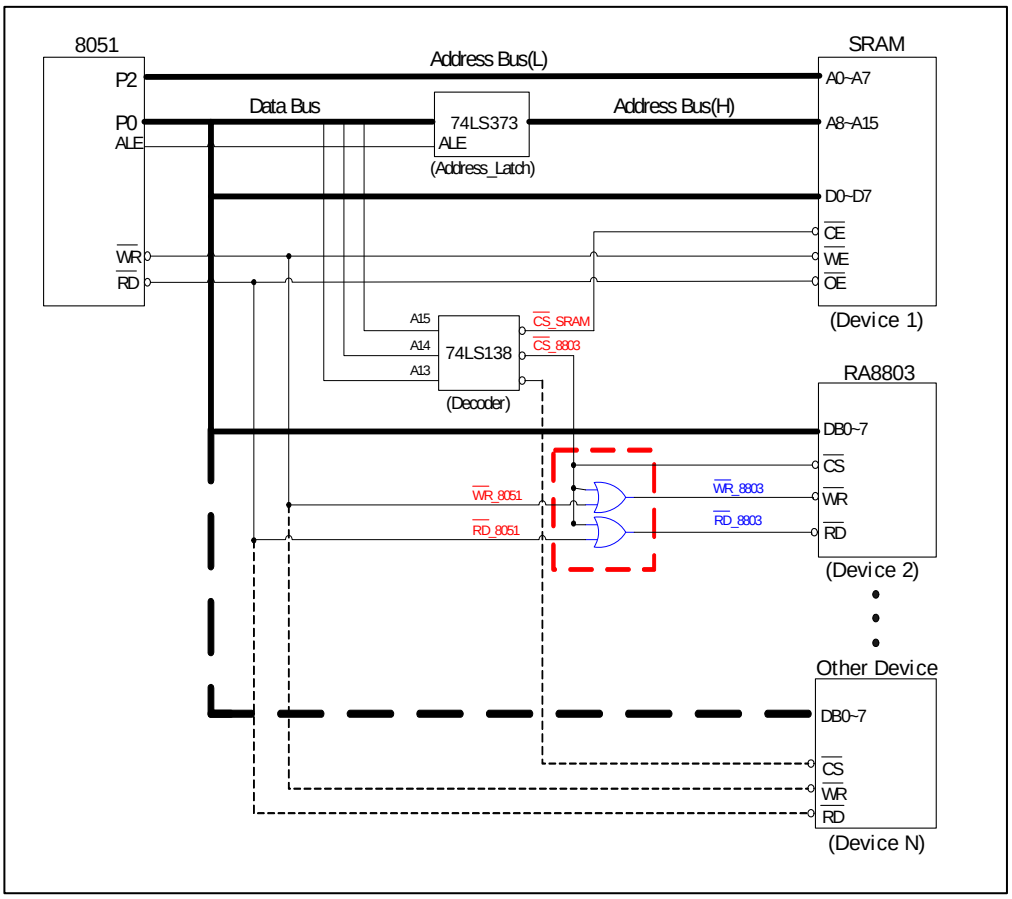


图 2-6：用 8051 的 Fully Decode 连接图

74LS138：User在多个Device之下可使用74LS138译码器透过间接寻址的方式，去选择哪一个Device动作(Chip Select)。

74LS373：在此当作地址栓锁器。

74LS32：Postive OR Gate 确保 RA8803/8822 读或写正常动作。

Port 0：为多功能型态，在扩充模式下，P0 既提供地址信号也提供数据信号，经由ALE 线及闩锁芯片如 74LS373 可将地址信号和数据信号分开

Port 2：当外接扩充内存或接口电路时，提供扩充 电路高八位地址(A8~A15)。

P3.6($\overline{WR}$)：将数据存入外部内存或接口芯片之控制信号输出脚。

P3.7($\overline{RD}$)：从外部内存或接口芯片读取数据之控制信号输出脚。

ALE(Address Latch Enable)：地址栓锁致能脚。

如果使用 8051 的片选模式(Fully Decode)，设计人员只需要一个 8051 指令就能完成一个对 RA8803/8822 读或写的动作。

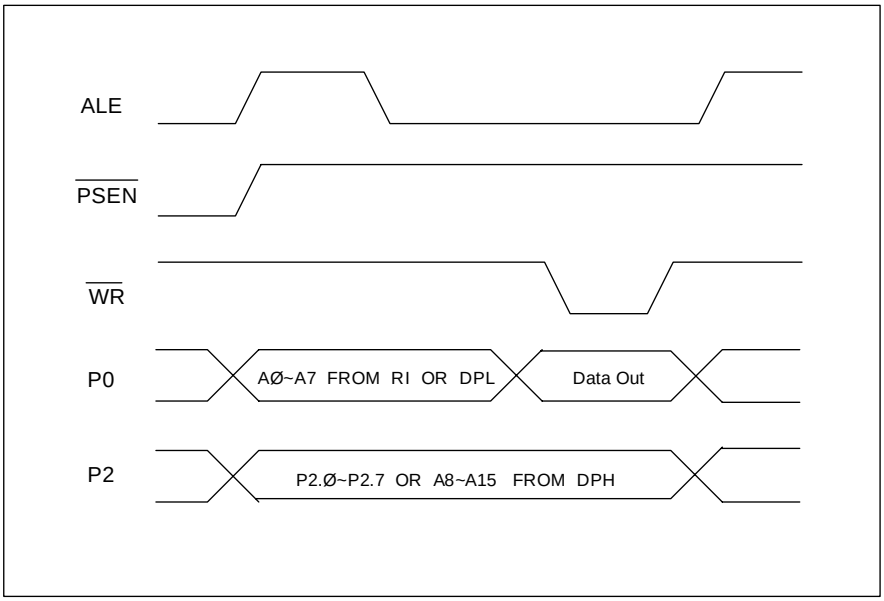


图 2-7：8051 对外部内存写入的时序图

3. 液晶显示驱动器(LCD Driver)的接口

本章将介绍 RA8803/8822 与液晶显示驱动器(LCD Driver)之间的接口，RA8803 在双图层模式下最大可支持 320x240 点的液晶显示器(LCD Panel)，而 RA8822 最大可支持 240x160 点，使用者可以依据在此范围内想设计的显示器 Panel 大小来选择适当的 LCD 驱动器。图 3-1 是 RA8803/8822 与 ST8016 LCD Driver 连接的示意图，用来驱动 160x160 的液晶显示器面板。

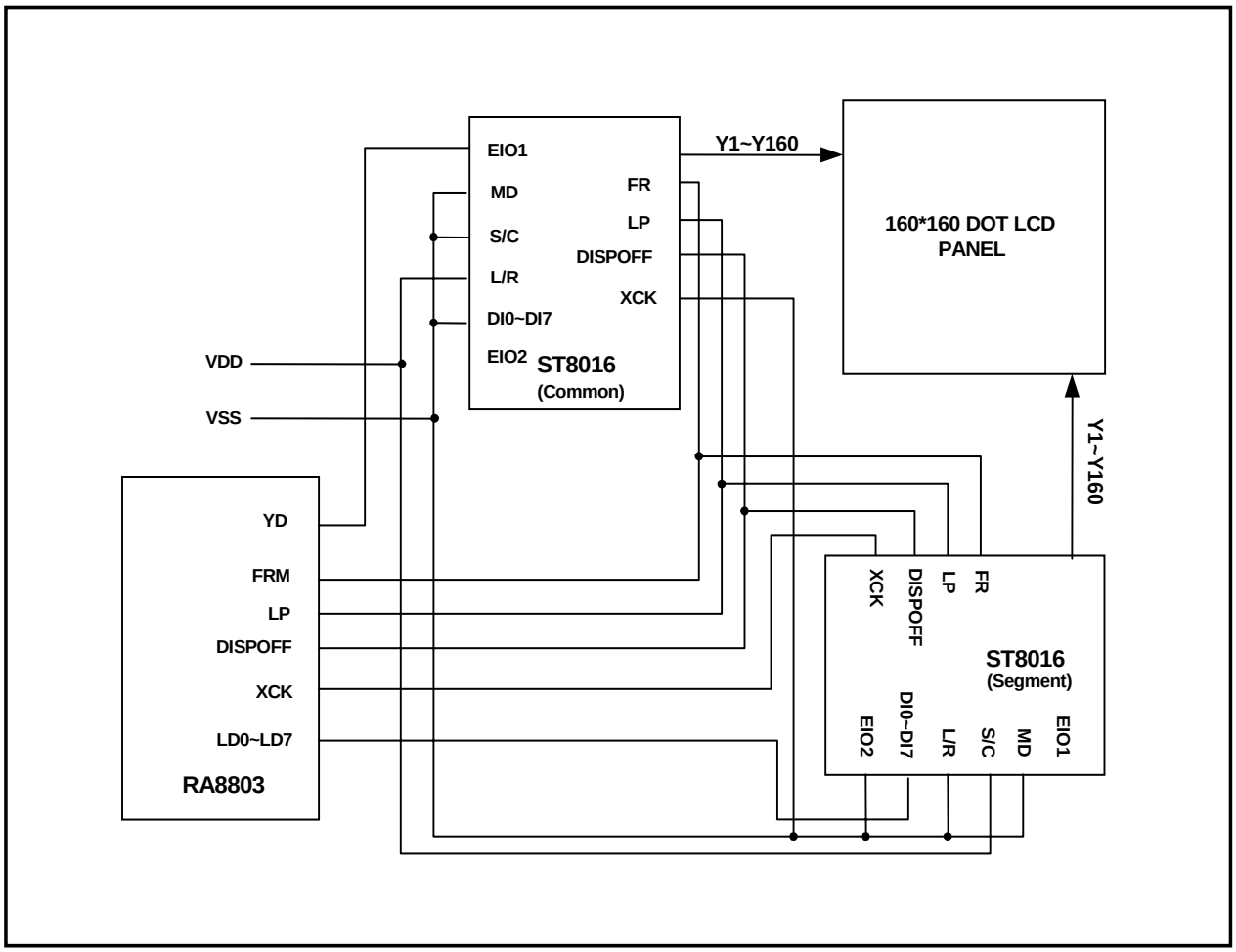


图 3-1：RA8803/8822 与 LCD Driver(ST8016)的接线图

在图 3-1 中使用两个 ST8016 的 LCD 驱动器，分别处理 160x160 LCD Panel 的 Common 及 Segment，而 RA8803/8822 则送出 Frame(FRM)、Latch Pulse(LP)、YD 及 Data Bus 等信号给 ST8016，图 3-2 是 RA8803/8822 与 LCD 驱动器之间的接口讯号波形图，对于 LCD 驱动器接口的脚位定义，使用者可以参考 RA8803/8822 规格书第 4-2 节的说明。

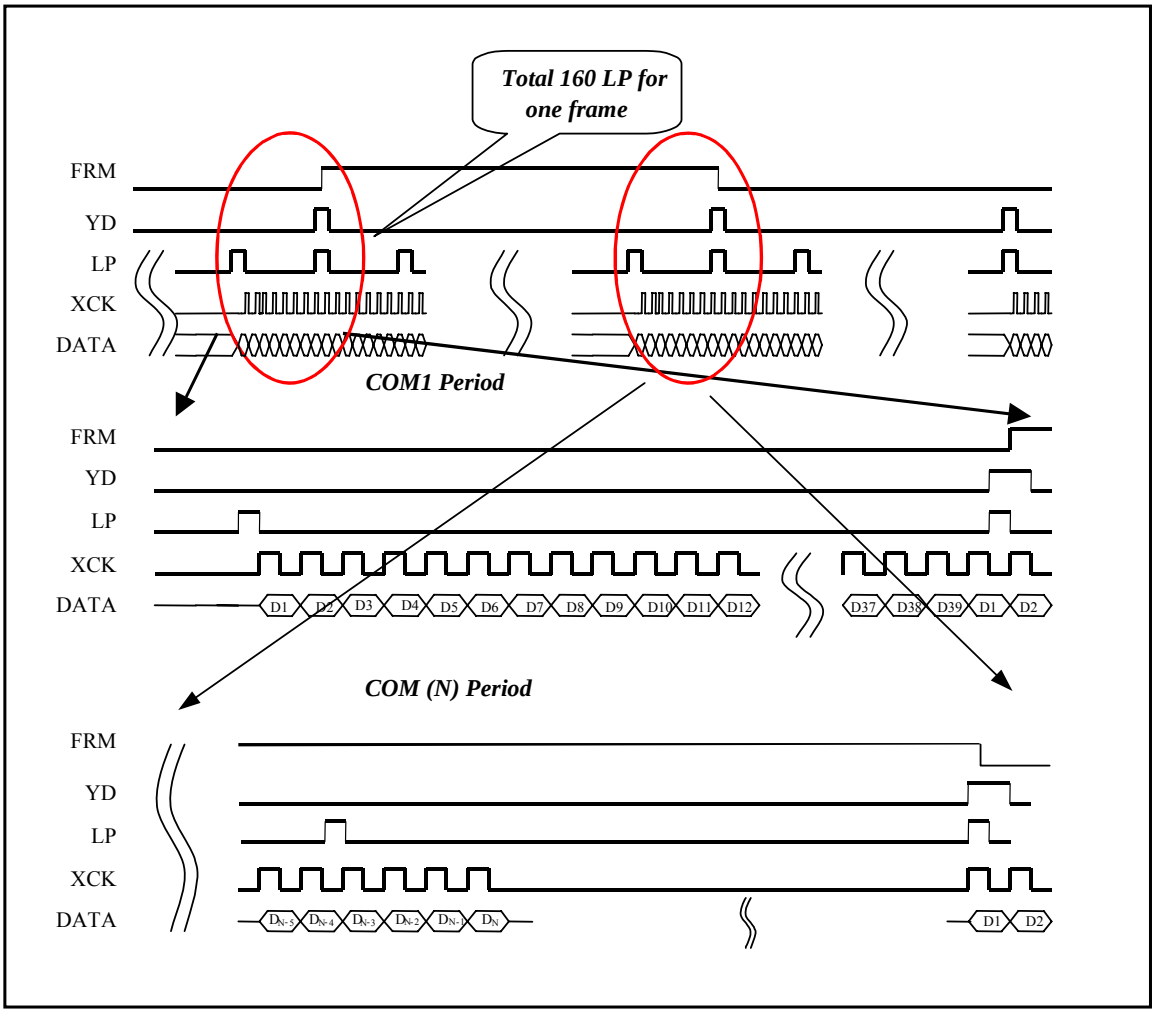


图 3-2：RA8803/8822 与驱动器的接口讯号波形图

RA8803/8822 也可以设定连接至 LCD 驱动器上的数据总线接口是 4-Bit 或是 8-Bit，使用者可以透过 SYS_DW 这根脚位去选择，如果 SYS_DW 外接一 Pull Low 电阻，则 RA8803/8822 的 LCD 驱动器数据总线接口将定义成 4-Bit。反之，如果 SYS_DW 外接一 Pull High 电阻，则 RA8803/8822 的 LCD 驱动器数据总线接口将定为 8-Bit。图 3-1 中 RA8803/8822 与 ST8016 LCD 驱动器连接的就是 8-Bit 的总线接口。

RA8803 可以支持 320x240 点尺寸的液晶显示器(LCD Panel)，也就是 20 列 x 15 行的中文字(RA8803/8822 内定每一中文字型为 16x16 点)，RA8822 可以支持 240x160 点尺寸的液晶显示器(LCD Panel)，也就是 15 列 x 10 行的中文字，针对不同尺寸的液晶显示器，可以用软件的方式来设定，使用者透过设定缓存器的方式，来更改对应的显示器大小。利用显示窗口(Display Window) REG[21h, 31h, 41h, 51h]和工作窗口(Active Window)REG[20h, 30h, 40h, 50h]，来改变 RA8803/8822 对显示器大小的设定。例如 RA8803，使用者选用的是 320x240 LCD 面板，所使用到的范围也是 320x240 点的大小，此时的显示窗口与工作窗口的缓存器设定值是相同的。

例如使用 RA8803 用在 320x240 的 LCD Panel，那么显示窗口相关缓存器设法如下:

DWRR = (320 / 8) - 1 = 39 = 27h
DWBR = 240 – 1 = 239 = EFh
DWLR = 0
DWTR = 0

工作窗口一般比显示窗口还小，相关缓存器须依照下列规则:

1. DWRR≥ AWRR≥ CPXR≥ AWLR≥ DWLR
2. DWBR≥ AWBR≥ CPYR≥ AWTR≥ DWTR

表 3-1：RA8803/8822 与驱动器 IC 的接口名称对照表

RA8803/8822 Driver 界面名称	Driver IC 界面名称	Driver IC 界面名称之定义
LP	LP	Data Latch Clock Latch Pulse in one line
	LOAD	Latch pulse of display data
	CL1	Data Latch Pulse
XCK	CP	Data Shift Clock Clock pulse for segment shift register
	SCP	Shift Clock Pulse for X-Drivers
	CL2	Data Shift Pulse
	HSCP	Shift Clock Pulse
YD	FLM	Scan Start-up Signal First Line Marker
	FR	Frame Pulse
	FRAME	Frame start signal(First line mark of common signal)
	CDATA	Synchronous Data
FRM	DF(M)	Switch signal to convert LCD drive waveform into AC
LD[7:0]	D[7:0]	LCD Data Bus
DISPOFF	/DISPOFF	Display OFF
	/D.OFF	Display OFF
	DISP	Display OFF

4. 中文字型 ROM

4-1 中文字型 ROM 的使用

RA8803/8822 内建有 512KByte 的 16x16 中文显示字型 ROM(Font ROM) 与 8x16 的 ASCII 半型字型。除了内建的 8x16 和 16x16 的字号外，还提供字型放大的功能，可利用 REG[F1h]的设定，将显示字号放大到 32x32、48x48 或 64x64。其中 RA8803/8822-T 储存标准繁体中文 BIG5 码，包含 13,094 个常用与次常用字型、408 个特殊字与两组 ASCII CODE，RA8803/8822-S 储存 7602 个标准 GB 码的简体中文。

缓存器[F0h]是用来设定与字型 ROM 相关的功能，当使用者选择 RA8803/8822-T 时，必须将 Bit[5..4]设成 “01” 才能正确显示繁体字型，选择 RA8803/8822-S 时，必须将 Bit[5..4]设成 “10” 才能正确显示简体字型。

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
7	**字型 ROM 的转换电路控制** 1：致能 0：Bypass (客户建立字型 ROM 时使用)	--	1h	R/W
6	**字型 ROM 的地址空间选择** 当 bit5~4 设定 ”00”➔ ROM Mode0，该位可以用来选择上或下的 256KB ROM 的地址空间。 1：选择下部 256KB 字型 ROM 0：选择上部 256KB 字型 ROM	--	0h	R/W
5-4	**字型 ROM 的语系选择** 0 0：选择简体（GB）字型 (256KB, Mode0) 0 1：选择繁体（BIG5）字型 (512KB, Mode1) 1 0：选择简体（GB）字型 (512KB, Mode2)	--	00h	R/W
2	**强制为 ASCII 解碼 (注 1)** 1：所有输入的 Data，都以 ASCII 解碼(00~FFh) 0：RA8803/22 会先检视输入 Data 的第一个字节介于： 00~9Fh，视为 ASCII (半角字) A0~FFh，视为 GB/BIG5 (全角字)	Text	0h	R/W

下面的例题在第一章有提到，只要先设定光标位置，然后将要显示的中文其中文码(Big-5 或 GB 码)共两个 Byte，透过 MPU 写入 Data Address 既可：

例 题 ： 8-Bit MPU 写入一中文字 “网” 到光标所在的位置

```
MOV     A,#F0h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#90h              ; 选择繁体字型
CALL    RegAddr_WRITE
MOV     A,#BAH              ; 加载 “网” 的中文码高位 “BA”
CALL    RegData_Write
MOV     A,#F4H              ; 加载 “网” 的中文码低位 “F4”
CALL    RegData_Write       ; 光标所在的位置将显示 “网” 的中文字
```

注 1：中文内码不论是 GB 或 BIG5 码都是由两个 Byte 组成，但是英文及一些符号 ASCII 码只由一个 Byte 组成(00h~FFh)，通常 RA8803/8822 将送到 Display RAM 的 Data(00h~9Fh)视为 ASCII 码，也就半角文字(8x16)，大于等于 “A0h” 的视为全角码(如繁简中文)的高位，必须再送一次低位内码，才能显示全角字型。如果使用者有用到 A0h~FFh 的 ASCII 码，则 MPU 在送 Data(ASCII 码)到 Display RAM 之前必须将缓存器[F0h]的 Bit2 设成 “1” 。

4-2 自建字型 ROM

RA8803/8822 内建有 512KByte 字型 ROM(Font ROM)，也可以开放给客户下 Mask 使用，客户可以自行编码自建字库，每个字都是 16x16 的字型，因为 16x16 的字形型需要 32Byte 的内存空间，512Kbyte 的 ROM 共可以储存 16K 个字型(16Kx32=512K)，512Kbyte 的 ROM 共有 19 条地址线(Address Line) → A[18:0]，00000h~0001Fh 的 32Byte 储存第一个字形，00020h~0003Fh 的 32Byte 储存第二个字形，依此类推，如下表 4-1:

表 4-1

Addr[18:5]	Addr[4:0]	字型 No.
000,0000,0000,000	XXXXX	1
000,0000,0000,001	XXXXX	2
:	XXXXX	:
:	XXXXX	:
111,1111,1111,110	XXXXX	16383
111,1111,1111,111	XXXXX	16384

至于 32Byte 的字型储存顺序如图 4-1 所示，假设您想将图 4-2 的字当成 Font ROM 的第一个字形，那么 ROM 00000h~0001Fh 的储存数据如下表 4-2:

表 4-2

Addr[18:5]	Addr[4:0]	Data
000,0000,0000,000	00000	08h
	00001	1Ch
	00010	1Ch
	00011	FFh
	00100	7Fh
	00101	1Ch
	00110	3Eh
	00111	3Eh
	01000	77h
	01001	41h
	01010	00h
	01011	00h
	01100	83h
	01101	7Fh
	01110	3Fh
	01111	0Fh
	10000	20h
	10001	10h
	10010	1Ch
	10011	9Eh
	10100	1Eh
	10101	1Fh
	10110	1Fh
	10111	1Fh
	11000	1Fh
	11001	3Fh
	11010	7Eh
	11011	FEh
	11100	FCh
	11101	F8h
	11110	F0h
	11111	C0h

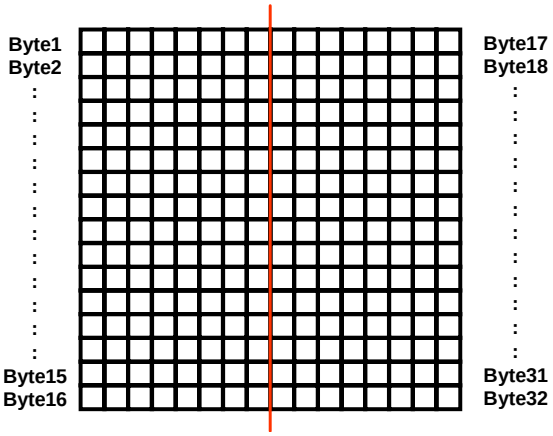


图 4-1：32Byte 的字型储存顺序

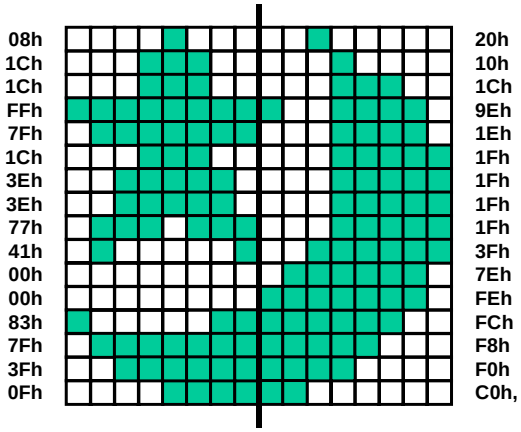


图 4-2：32Byte 的字型 Data

因为 512Kbyte 的 ROM 共可以储存 16K 的字型，所以我们用 2 个 Byte 的字型码来选择显示的字型，事实上字型码与 ROM Address 的对应如图 4-3 所示。字型码的 High Byte 与 Low Byte 各取 Bit[6:0]组合成 Font ROM 的 Address A[18:5]，也就是 A[18] 对应 High Byte 的 Bit6，A[17] 对应 High Byte 的 Bit5，依此类推，A[11] 对应 Low Byte 的 Bit6，A[10] 对应 Low Byte 的 Bit5，直到 A[5] 对应 High Byte 的 Bit0，至于 High Byte 与 Low Byte 的 Bit7 为 0 或 1 皆不影响选择显示的字型。

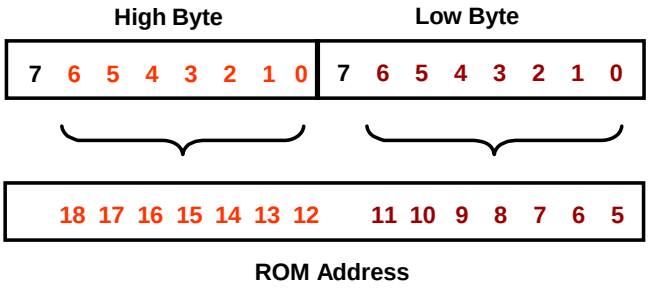


图 4-3：字型码与 ROM Address 的对应

如果 Font ROM 的 00000h~0001Fh 储存表 4-2 的 Data，那么在文字模式下连续写入”00h”(或 80h)两次(High Byte and Lowe Byte)，光标所在位置会秀出图 4-2 所示的字型，请参考图 4-4。图 4-4 中 High Byte 与 Low Byte 的 Bit7 为 X，代表 Don’t Care，也就是 0 或 1 皆不影响选择显示的字型。

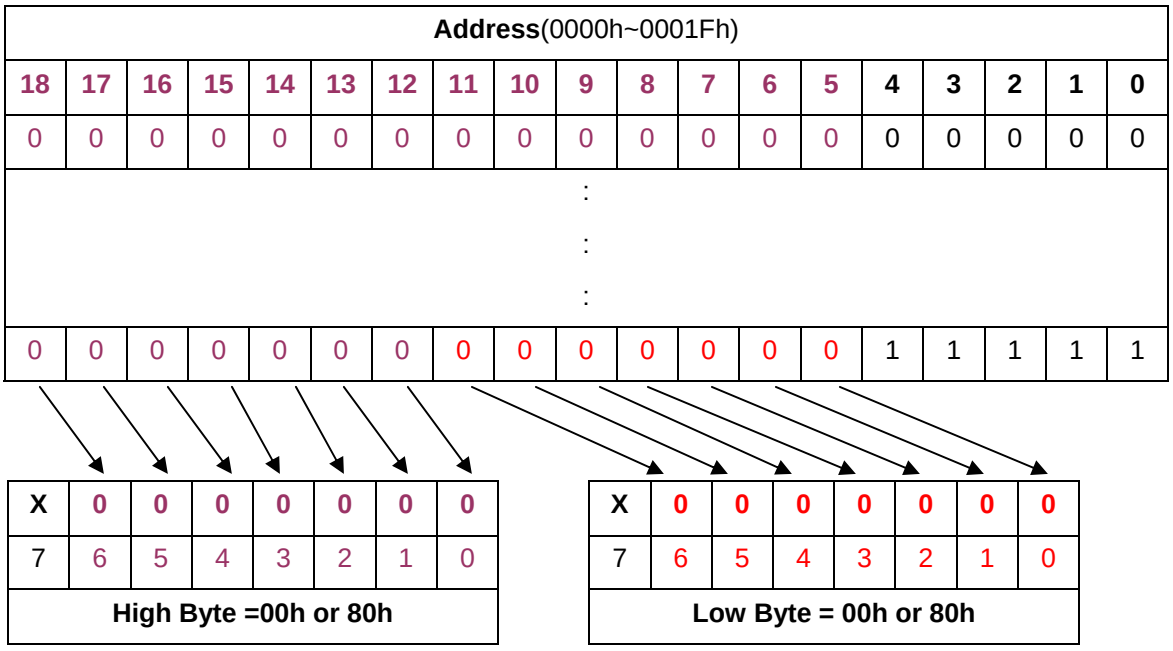


图 4-4：字型码与 ROM Address 对应的例题(1)

再假设 Font ROM 的 5C8A0h~5C8BFh 储存表 4-2 的 Data，那么在文字模式下连续写入”5Ch”(High Byte) 与”45h”(Lowe Byte)，光标所在位置会秀出图 4-4 所示的字型，请参考图 4-5。同样的在图 4-5 中 High Byte 与 Low Byte 的 Bit7 为 X，代表 Don’t Care，也就是 0 或 1 皆不影响选择显示的字型。

因此客户可以依据上述法则建立自己的 ROM Code 与字型码，经由下 Mask 产生专有的字型用于产品上。而客户拿到自己建立字型 ROM 的 RA8803/8822 后，在使用上必须将缓存器[F0h]的 Bit[7]设成 “0” 才能正确显示相对应于字型码的字型。至于外挂字型 ROM 也是同样的根据这个规则，只是使用时要将缓存器[F0h]的 Bit6 要设成 1。

附带一提的是您也可以将字型当成一 16x16 的 Bitmap，每个 Bitmap 由 2 个 Byte 码来定义，利用不同的 Bitmap 组合成一图形，相对于 512Kbyte 的 ROM，可以储存许多图形在里面!

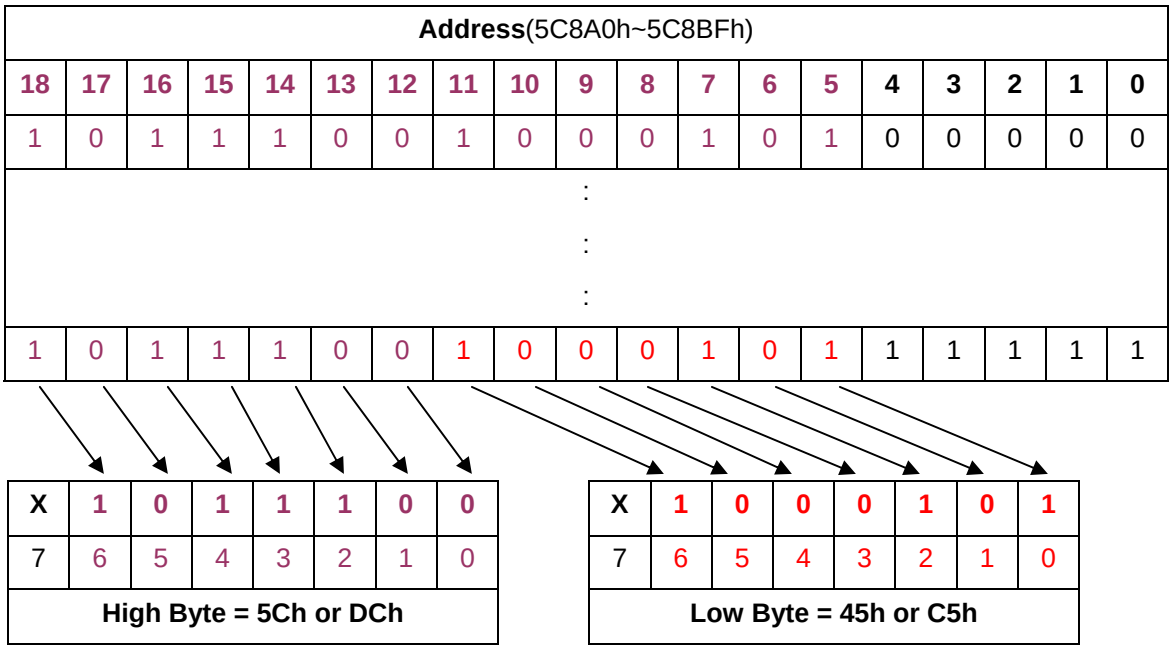


图 4-5：字型码与 ROM Address 对应的例题(2)

5. 液晶显示器的亮度调整

传统的 LCD 亮度调整方式大都是以可调式电阻为主，藉由电阻值的改变去控制供给 LCD 面板的升压电路，来达到调整 LCD 亮度的目的，使用上非常不方便。因此 RA8803/8822 内建了一个定电流输出的 5-bit 数字-模拟转换器(Digital to Analog Converter, DAC)，使用者可以利用这个 DAC 产生不同的电流输出，进而控制外部的升压电路，使得供给 LCD Panel 高压的电压准位随着 DAC 的设定值而改变，这样透过 MPU 就可以达到用程序的方法去控制 LCD 的亮度。

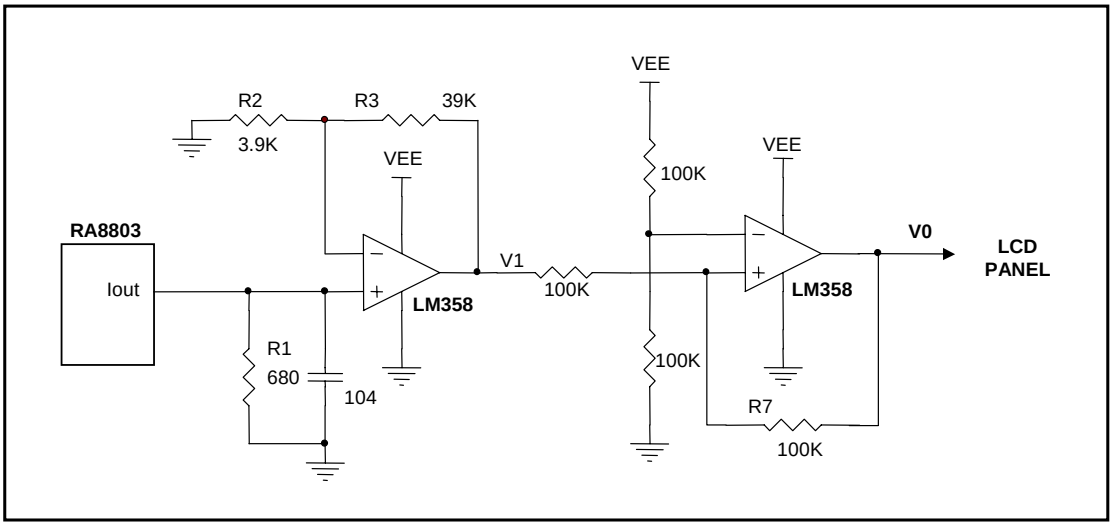


图 5-1：用 DAC 控制 LCD 亮度的应用电路(I)

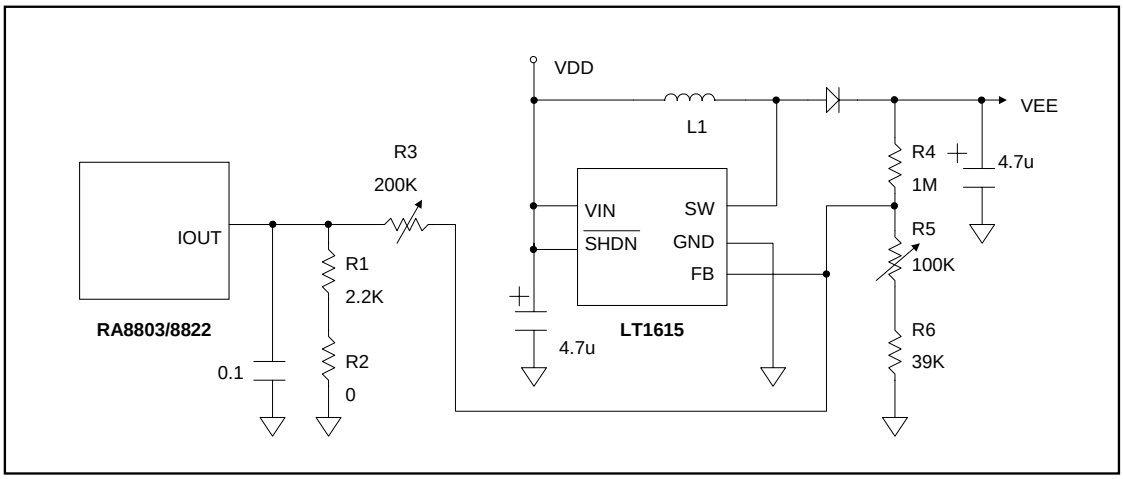


图 5-2：用 DAC 控制 LCD 亮度的应用电路(II)

图 5-1 是用 RA8803/8822 的 DAC 控制 LCD 亮度的应用电路，在此图中 RA8803/8822 是利用外部减法器电路和控制 DAC 电流输出范围，去改变供给 LCD 面板的输出电压 “V0” 的变动范围，进而得到想要的亮度控制。图 5-2 是藉由升压组件(LT1615)来做 VEE 电压的输出调整，以供给 LCD 面板的应用电路，R5 用来调

整 VEE(通常为 VLCD 电压)，让 VEE 调整为适合所使用的 Panel 与驱动 IC 之电压准位，若要经由 RA8803/8822 进行亮度控制，可由 RA8803/8822 的 DAC 输出电流 IOUT 进行(5Bit 的 DAC 可以有 32 阶的变化)，由 R3 来调整控制的范围，使缓存器 LCCR[4:0] 为 00000b~11111b 时，VEE 的+/- 变化在 1V~2V 左右(视 Panel 的特性) 。RA8803/8822 DAC 输出电流与 VEE 为反比关系，此电路可节省较多的组件，降低成本。事实上控制 LCD 亮度的方法很简单，只要透过 MPU 去设定缓存器 LCCR 就可以控制整个 DAC 的功能，下面的程序例题是说明控制 LCD 的亮度为最亮及最暗的方法。

REG [D0h] LCD Contrast Control Register (LCCR)

Bit	Description	Default	Access
7	**LCD 亮度控制(DAC 功能)** 1：禁能 0：致能	1h	R/W
4-0	**设定 DAC 输出电流 Iout 的值(LCD 亮度控制)** 0 0 0 0 0b → 0μA±0.2 uA (Min. Current) ： ： 1 1 1 1 1b → 540μA±140 μA (Max. Current)	0h	R/W

例 题：DAC - LCD 亮度调整

```
MOV     A,#D0h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00011111b        ; 设定 LCD 的亮度为最暗
CALL    RegAddr_WRITE       ; 存入 Data 到缓存器[D0h]LCCR

MOV     A,#D0h              ; 选择 LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00000000b        ; 设定 LCD 的亮度为最亮
CALL    RegAddr_WRITE       ; 存入 Data 到缓存器[D0h]LCCR
```

图 5-3 是 RA8803/8822 DAC 的输出电流 “Iout” 和图 5-2 LCD 面板的输出电压 “VEE”，两者之间的对应曲线图。每种 LCD 面板所需要的电压与明亮控制范围不同，因此使用者在开发时应配合 LCD 面板规格、升压电路与控制 DAC 电流输出范围，才能得到想要的亮度控制，就如同图 5-1 的外部减法器电路，使用者可能需要改变 R1、R2 与 R3，以及用软件的方式和控制 DAC 电流输出范围，才能得到适当的亮度控制。

RA8803/8822 有 5-bit 的 DAC，因此在 Iout 的输出电流会有 2^5＝32 组的电流，当不同的电流输出到减法电路，可以得到 32 组不同的 V0 输出电压供给 LCD 面板，来达到亮度调整的功能。当使用者搭配不同的 LCD 面板，所需要的电压与明亮控制范围会不同，所以在整个减法电路的 R1，R2 和 R3 也可以依照需求来做调整，得到适合的 V0 输出电压供给 LCD 面板(VLCD)。电流输出脚位 IOUT 在 Disable 时为 Tri-state。

虽然 DAC 可用于控制升压电路，进行对比显示(Contrast)设定，但仍须要注意的是升压电路本身的精确度，即使是同一批号的生压 IC，产生的 VLCD 电压准位也会不同，而且 LCD Panel 对相同 VLCD 电压产生的对比显示效果也不一样，因此如果使用 RA8803/8822 的 DAC 进行对比显示(Contrast)设定，建议仍要加上可调电阻做为出厂设定，可参考图 B-1。图 5-4 为 REG[D0h]与 Iout 输出的对应曲线图范例。

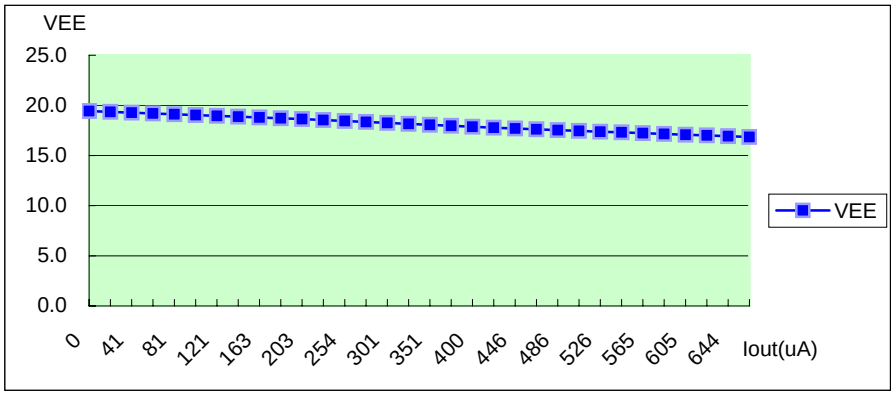


图 5-3：Iout 输出透过亮度调整电路与 VEE 的对应曲线图

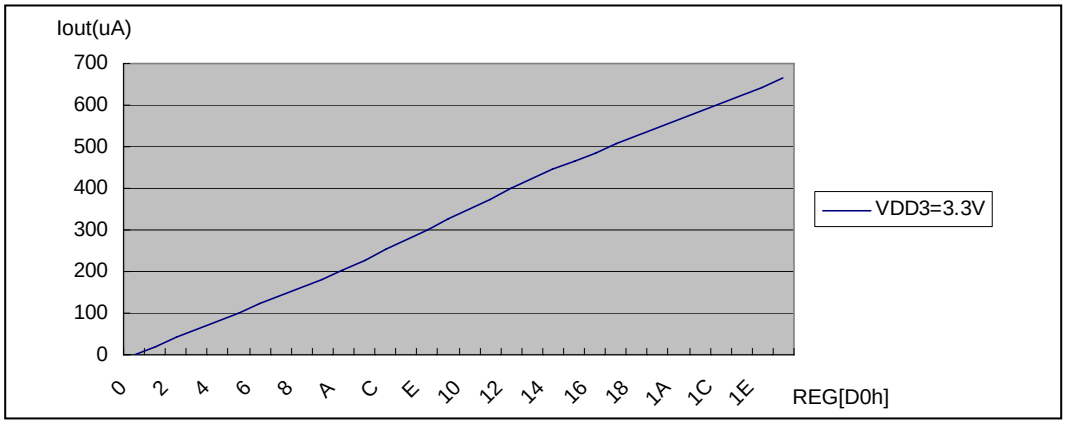


图 5-4：REG[D0h]与 Iout 输出的对应曲线图

REG[D0h]	Iout(uA)
0	0.1
1	20.1
2	40.6
3	60.5
4	80.5
5	101.4
6	121.4
7	141.6

REG[D0h]	Iout(uA)
8	162.7
9	182.4
A	203.3
B	227.2
C	254.4
D	276
E	301
F	325.5

REG[D0h]	Iout(uA)
10	350.9
11	375
12	400.1
13	425
14	446
15	466
16	486
17	506

REG[D0h]	Iout(uA)
18	526
19	545
1A	565
1B	585
1C	605
1D	624
1E	644
1F	664

6. 触摸式面板(Touch Panel)的界面

目前触摸式面板(Touch Panel)的应用愈来愈多，然而目前市面上的 LCD Controller 或驱动器大都无法直接提供触摸式面板的解决方案，因此使用者必须外加许多电路与零件，造成成本上的增加，而 RA8803/8822 内建了一个 10-bit 模拟-数字转换器(Analog to Digital Converter, ADC)及数个模拟开关(Analog Switch)，使用者可以将四线电阻式触摸式面板的 XL, XR, YU, YD 接到 RA8803/8822，然后利用模拟开关切换让 ADC 读取电阻上的电压值，再由 MPU 读取 ADC 的转换值，而得到触摸面板 Touch 的相对位置。

6-1 电阻式触摸面板

电阻式触摸面板是由两层极薄的电阻面板组成，如图 6-1 所示，两层面板之间有一个很小的间距，当有外力在面板上的某一点压下去时，会在施力点造成两层电阻接触，也就是短路(Short)，而两层电阻面板的端点都各有电极，如图 6-2 所示 YU, YD, XL, XR，因此配合一些开关就可侦测出面板上哪一相对位置被 Touch。

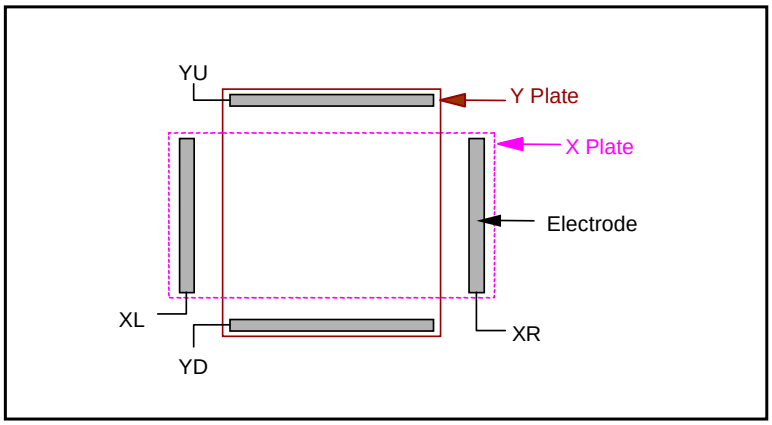


图 6-1：触摸面板(Touch Panel)

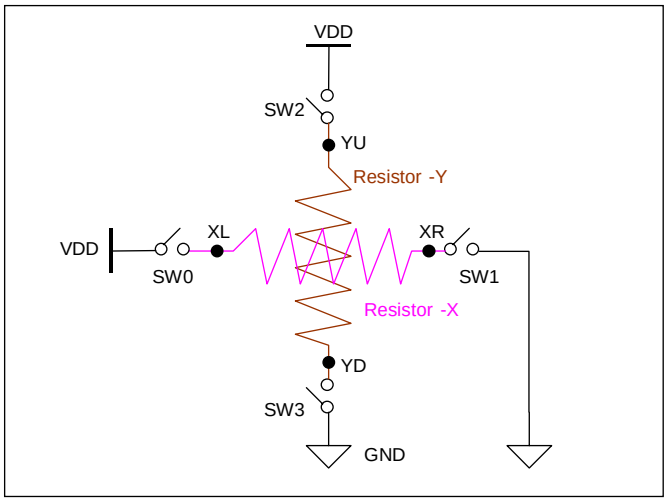


图 6-2：触摸面板与侦测开关

在图 6-3 中，设定开关 SW2 与 SW3 是 OFF(Open)，SW0 与 SW1 是 ON(Close)，当有外力在面板上的某一点压下去时，由 YU 点取得电压接到 ADC(Analog to Digital Converter)，就可以得到被 Touch 点的 X 坐标相对位置。

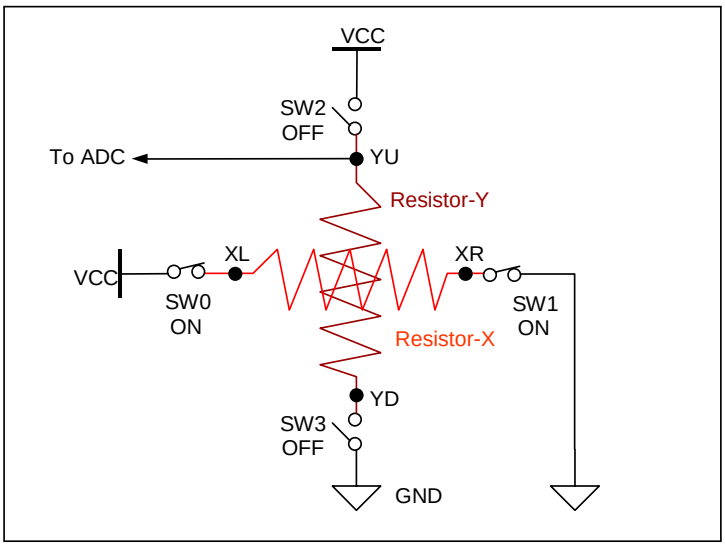


图 6-3：读取 X 坐标

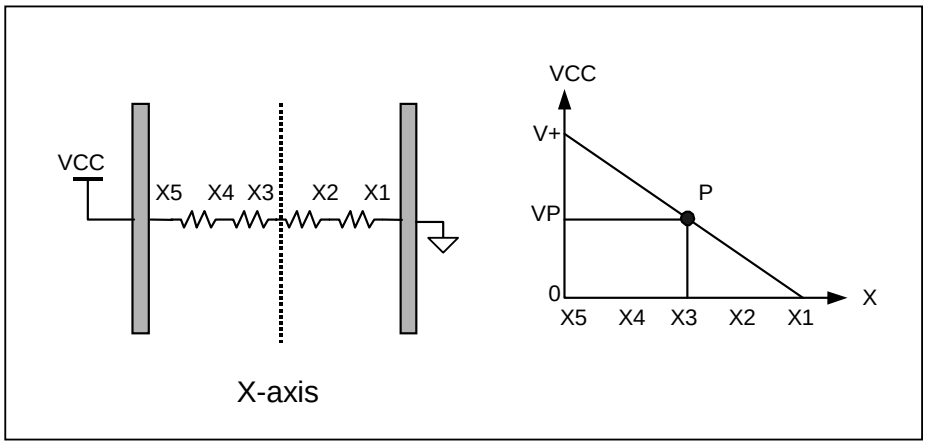


图 6-4：Resistor-X 的分压

在图 6-3 中，因为开关 SW2 与 SW3 是 OFF，因此 YD 点是 Floating，所以当有外力在面板上的某一点压下去时，YU 上的电压事实上就是 X 的 Panel(也就是电阻)上的分压结果，压在面板上的不同一点，就会得到不同的分压值，如图 6-4 所示。

同理，在图 6-5 中，设定开关 SW0 与 SW1 是 OFF(Open)，SW2 与 SW3 是 ON(Close)，当有外力在面板上的某一点压下去时，由 XL 点取得电压接到 ADC(Analog to Digital Converter)，就可以得到被 Touch 点的 Y 坐标相对位置。一般说来许多触摸面板都是贴在 LCD 面板上面，因此在程序设计上如果

重复图 6-3 与 6-5 的读取步骤就可以顺利得知被 Touch 的点是在屏幕上的哪一位置。

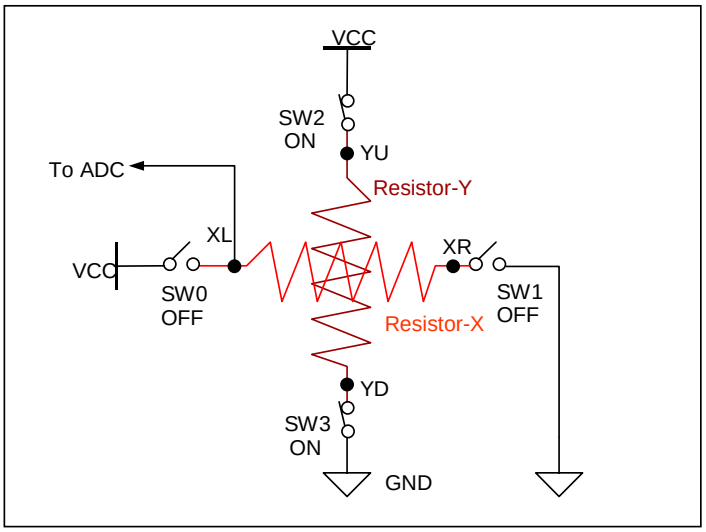


图 6-5：读取 Y 坐标

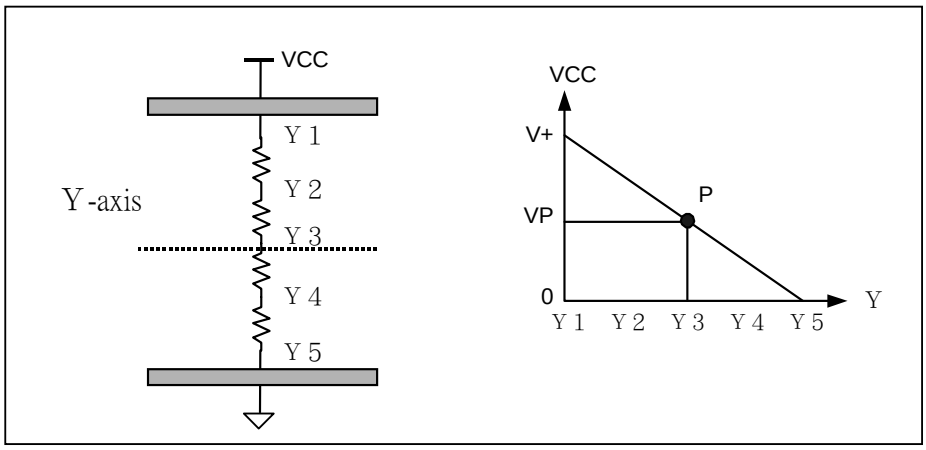


图 6-6：Resistor-Y 的分压

在图 6-5 中，因为开关 SW0 与 SW1 是 OFF，因此 XR 点是 Floating，所以当有外力在面板上的某一点压下去时，XL 上的电压事实上就是 Y 的 Panel(也就是电阻)上的分压结果，压在面板上的不同一点，就会得到不同的分压值，如图 6-6 所示。

注: 如果应用上要求使用高精度触摸屏(如 12-Bit）功能的话，建议自行外加 ADS7843 或其它类似功能的触摸屏控制器。

6-2 触摸面板的应用

图 6-7 是用 RA8803/8822 的触摸式面板应用电路，电路图上的电容可以少噪声，图 6-8 触摸式面板侦测的示意图与 6-9 的流程图则是说明 RA8803/8822 触摸式面板读取的控制方式，与触摸式面板有关的缓存器为 TPCR、TPXR、TPYR 与 TPSR(ADCS)，在使用触摸式面板时必须先将触摸式面板功能开启，缓存器 TPCR 的 Bit-7 与 Bit-6 设为 “1”，同时 TPCR 的 Bit[3..0] 设为 “1000”，也就是 Switch SW3 为 On 的状态，然后程序可以侦测缓存器 TPSR 的 Bit-6 是否为 “1”，如果缓存器 TPSR 的 Bit-6 为 “1”，则表示触摸式面板目前被 Touch，请参考图 6-8。

在侦测阶段时，缓存器 TPCR 的 Bit-7 与 Bit-6 可以先为 “0” (ADC Disable)，如果程序侦测到缓存器 TPSR 的 Bit-6 为 “1”，表示触摸式面板目前被 Touch，然后再将 ADC Enable -- 缓存器 TPCR 的 Bit-7 与 Bit-6 设为 “1” 也可以，如此可避免触摸式面板未被 Touch 而让 ADC 动作产生不必要的耗电。

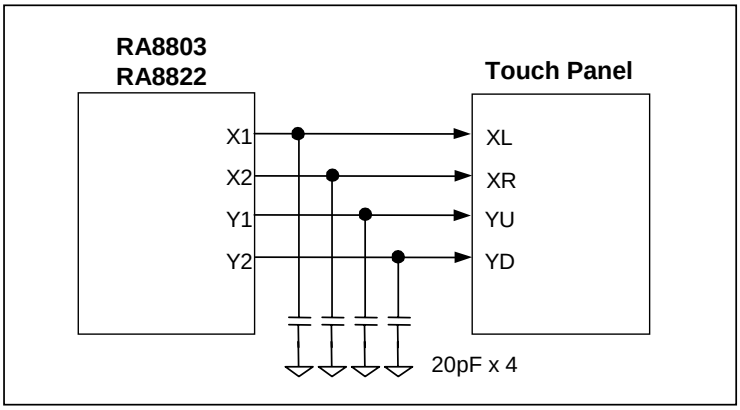


图 6-7：RA8803/8822 的触摸式面板应用电路

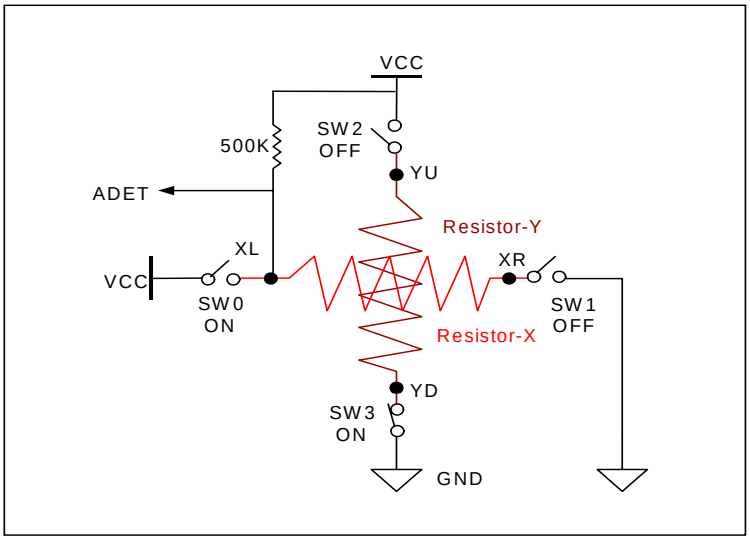


图 6-8：RA8803/8822 的触摸式面板的侦测

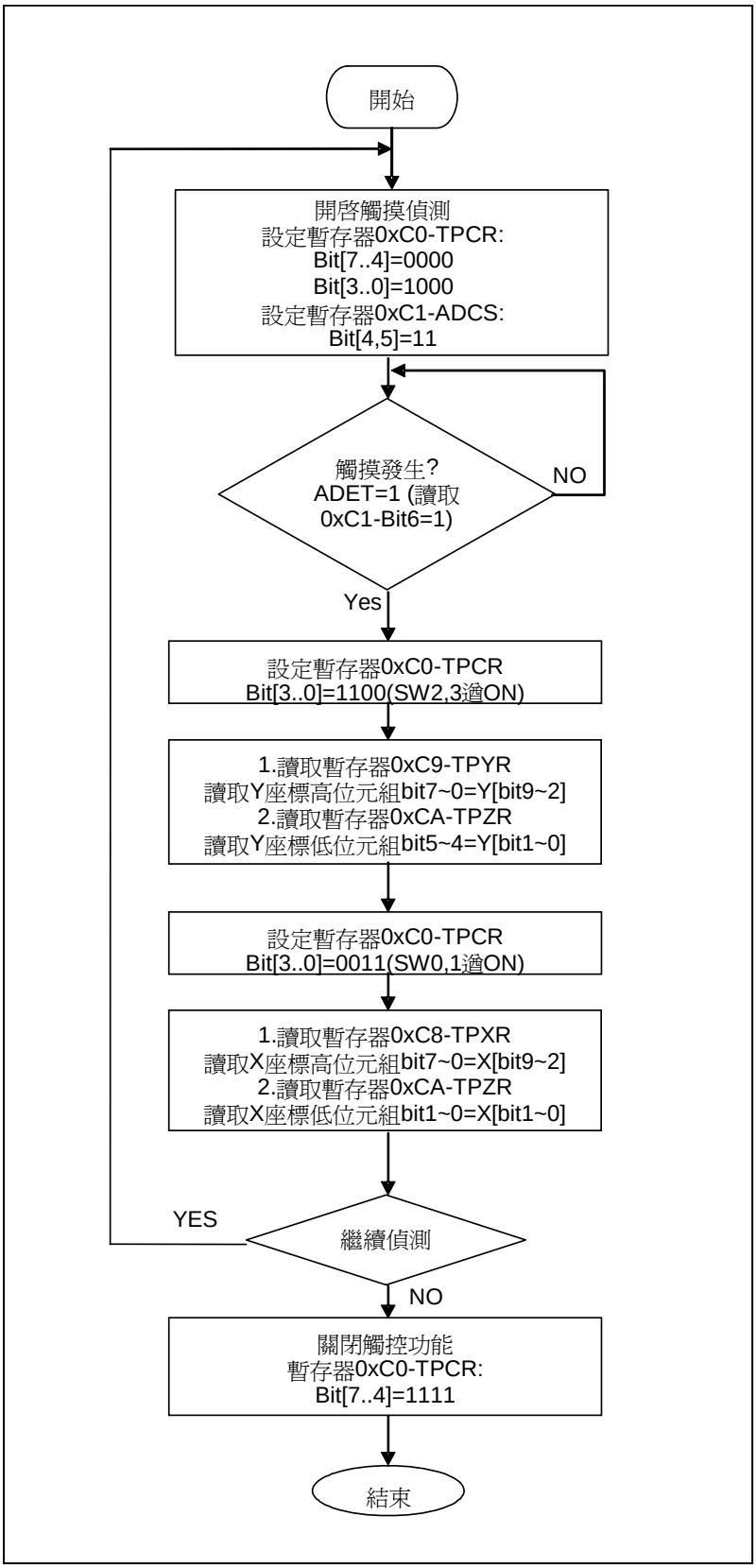


图 6-9：触摸式面板读取的控制流程图

REG [C0h] Touch Panel Control Register (TPCR)

Bit	Description	Default	Access
7	**触控屏幕功能启动** 1：致能 0：禁能	1h	R/W
6	**触控屏幕数据输出** 1：致能触控屏幕的数据输出 0：禁能触控屏幕的数据输出	1h	R/W
4	**触控屏幕自动/手动扫描** 1：自动 0：手动	1h	R
3-0	**触控屏幕控制位** Bit3 = 0 → Switch SW3 OFF, Bit3 = 1 → Switch SW3 ON Bit2 = 0 → Switch SW2 OFF, Bit2 = 1 → Switch SW2 ON Bit1 = 0 → Switch SW1 OFF, Bit1 = 1 → Switch SW1 ON Bit0 = 0 → Switch SW0 OFF, Bit0 = 1 → Switch SW0 ON	图 6-2	R/W

REG [C1h] ADC Status Register (TPSR/ADCS)

Bit	Description	Default	Access
7	**ADC 数据转换完成指示** 1：ADC 数据转换已完成 0：ADC 数据转换未完成	0h	R/W
6	**触摸事件的侦测指示** 1：有被触摸 0：没被触摸	0h	R/W
5	此位必须为 “1 “	1h	R/W
3-2	**设定 ADC 的时脉转换速度** 0 0：SCLK/32 0 1：SCLK//64 1 0：SCLK/128 1 1：SCLK/256	2h	R/W

REG [C8h] Touch Panel Segment High Byte Data Register (TPXR)

Bit	Description	Default	Access
7-0	**储存触控屏幕行的高字节(bit9~2)的相对位置数据**	80h	R

REG [C9h] Touch Panel Common High Byte Data Register (TPYR)

Bit	Description	Default	Access
7-0	**储存触控屏幕列的高字节(bit9~2)的相对位置数据**	80h	R

REG [CAh] Touch Panel Segment/Common Low Byte Data Register (TPZR)

Bit	Description	Default	Access
7-6	**储存触控屏幕行的低字节(bit1~0)数据**	0h	R
3-2	**储存触控屏幕列的低字节(bit1~0)数据**	0h	R

下面的程序(*.C)例题是说明如何判断触摸式面板被 “Touch” 及如何读取 ADC 的 Data 值。

例 题：

```
while(Read_TP_status() == 0x40)          // 侦测 Touch Panel 是否被 “Touch”
{                                        // 如果 0xC1-bit6=1 表示有被 “Touch”

   LCD_CmdWrite(0xC0,0xCC);              // 切换模拟开关，准备读取垂直数据
   y_temp = LCD_CmdRead(0xC9);           // 读取垂直相对位置(Get Column Data)
     :
     :
   LCD_CmdWrite(0xC0,0xC3);              // 切换模拟开关，准备读取水平数据
   x_temp = LCD_CmdRead(0xC8);           // 读取水平相对位置(Get Row Data)
     :
     :
}


// 侦测 Touch Panel 是否被按下 子程序  ……..... //
unsigned char Read_TP_status(void)
{
   LCD_CmdWrite(0xC0,0x08);              // 平常侦测触控屏幕状态只要将 SW3-ON 即可
   LCD_CmdWrite(0xC1,0x35);              // ADC 初始值设定
   delay(1000);                          // delay()只是防止触控时的噪声抑制
   return  LCD_CmdRead(0xC1) & 0x40;     // 0xC1 地址之 bit6 若为”1”，表示有触控发生
}
```

7. 系统时序(System Clock)

RA8803/8822 内部的系统时序(System Clock)可以由一外部的 32768Hz 石英晶体(X'tal)配合内部的一锁相回路(PLL)所产生。图 7-1 与 7-2 是 RA8803/8822 的系统时序接线应用电路图。

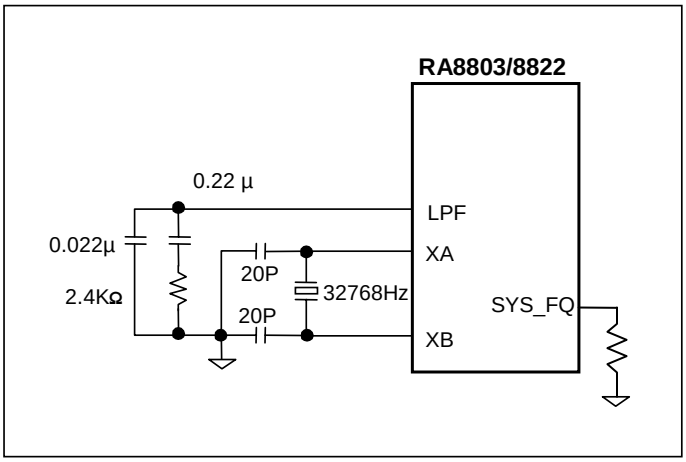


图 7-1：系统时序产生方式为 X'Tal 与 内部 PLL

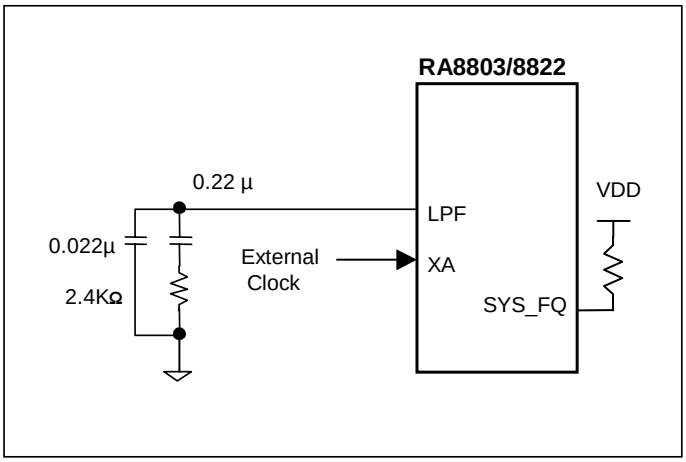


图 7-2：系统时序产生方式为外部 Clock

如果是使用外部 X'tal(32.678KHz)与锁相回路(PLL)所产生的系统时脉(System Clock)，则不同电压(VDD)对系统时脉所产生的影响将非常小。RA8803/8822 的锁相回路(PLL)所产生的系统时脉(System Clock)可透过缓存器[01]设定选择不同的系统频率，设定方式为如下:

REG [01h] Misc. Register (MISC)

Bit	Description	Default	Access
6	CLK_OUT 致能控制 1：致能 0：禁能	1h	R/W
1-0	系统时脉选择 0 0：3MHz 0 1：4MHz 1 0：8MHz 1 1：12MHz	0h	R/W

当选择不同的 System Clock 时，RA8803/8822 提供 CLK_OUT 输出脚位,可直接量测实际输出频率，此输出频率可作为验证 MPU 与 RA8803/8822 的接口连接是否有错，或是 RA8803/8822 是否正常工作。

8. 软硬件的启始设定

8-1 重置(Reset)与系统设定

RA8803/8822 驱动器可以在系统开机(Power-On)或进行硬件重置(Hardware Reset)时做启始化的设定，如 SYS_MI 这根脚位，可以由 Pull High 或 Pull Low 电阻去选择 RA8803/8822 的 MPU 接口是 8080 或者是 6800，表 8-1 将这些硬件的启始设定列出供使用者参考。

表 8-1：硬件的启始设定

PIN	脚 位 名 称	简　述	“1” mean (Pull High)	“0” mean (Pull Low)
99	SYS_MI	MPU Type Select	M6800	8080
SYS_MI 是用来做 MPU 形式之选择。 如果 SYS_MI 外接一 Pull Low 电阻，则 RA8803/8822 的 MPU 接口将定义成 8080，反之，如果 SYS_MI 外接一 Pull High 电阻，则 RA8803/8822 的 MPU 接口将定义成 6800。				
98	SYS_DB	MPU Data Bus Select	8-bit	4-bit
SYS_DB 是选择 8080 MPU 的数据总线为 4 位或 8 位。 如果 SYS_DB 外接一 Pull Low 电阻，则 RA8803/8822 的 8080 MPU Data Bus 接口将定义成 4-Bit，反之，如果 SYS_DB 外接一 Pull High 电阻，则 RA8803/8822 的 8080 MPU Data Bus 界面将定为 8-Bit。				
3	SYS_FQ	Clock Select	PLL_CLK	EX-CLOCK
SYS_FQ 是选择产生系统时脉为 PLL 或是 外接 CLOCK。 如果 SYS_FQ 外接一 Pull High 电阻，则 RA8803/8822 系统时序产生将是外接 CLOCK 的方式，反之，如果 SYS_FQ 外接一 Pull Low 电阻，则 RA8803/8822 的系统时序产生将是 X’tal 与 PLL。				
100	SYS_DW	LCD Data Bus Select	8-bit	4-bit
如果 SYS_DW 外接一 Pull Low 电阻，则 RA8803/8822 的 LCD Driver Data Bus 接口将定义成 4-Bit。反之，如果 SYS_DW 外接一 Pull High 电阻，则 RA8803/8822 的 LCD Driver Data Bus 界面将定为 8-Bit。				
4	SYS_NM	Test Mode	Set → “1”	
系统测试脚位，此脚位必需强制接到 High 准位				
50	OPM1	Test Mode	Set → “1”	
49	OPM0			
测试模式，一般使用者将 OPM0 与 OPM1 设成 NC Pin 既可。				

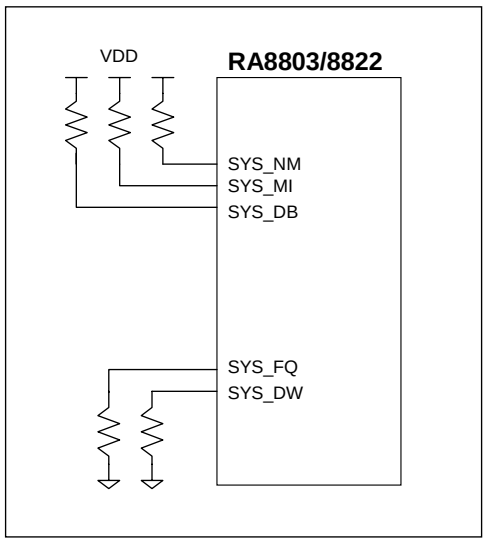


图 8-1：硬件的设定范例

RST# 动作时会读取 SYS_MI、SYS_DB、SYS_FQ 等输入值作为系统设定，如果相对应的 Pin 外接 Pull-High 电阻，则输入值将被视为 ”1”，如果相对应的 Pin 外接 Pull-Low 电阻则输入值将被视为 “0”，Pull 的电阻为 10Kohm 即可。

图 8-1 是一硬件设定范例，表示 RA8803/8822 选择 6800 series MPU 界面，MPU Data Bus 接口将定义成 8-Bit，系统时序的产生将是 X’tal 与 PLL， LCD Driver Data Bus 接口定义成 4-Bit，请参考表 8-1。

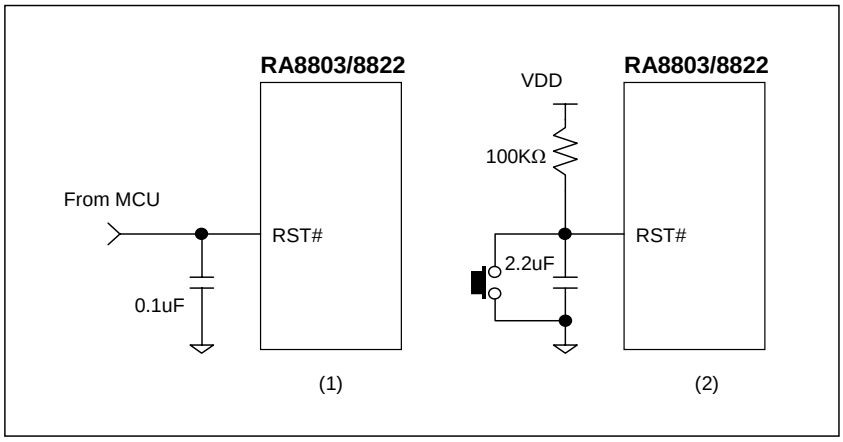


图 8-2：重置脚位 RST# 的接法范例

图 8-2 是一 Reset Pin – RST# 的应用范例，它可以由 MPU 来控制如图 8-2 的(1)，或是由一 RC 电路来产生，如图 8-2 的(2)。RA8803/8822 没有完成 Reset 的动作是无法接受 MPU 的任何指令，甚至会造成系统设定错误。

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
5	**软件重置** 所有缓存器回到初始值，但是 RAM 的内容不会被清除。 1：重置所有缓存器 0：正常模式，平常应保持为 ”0”	--	0h	R/W

RA8803/8822 也可以由缓存器[00h]的 Bit5 来进行软件重置，缓存器[00h]的 Bit5 由 MPU 写入”1”后 RA8803/8822 会被重置，之后缓存器[00h]的 Bit5 会自动回复到”0”。

8-2 电源开启或重置(Power On/Reset)的程序

图 8-2A 为一般 RA8803/8822 重置(Reset)的时序，以 RA8803 使用 320x240 pixel 的 LCD Panel 例，t_{RST} 必须大于 250ms，t_{RSTH} 必须大于 50ms，足够的重置时间才能确定 RA8803/8822 Reset 完成。

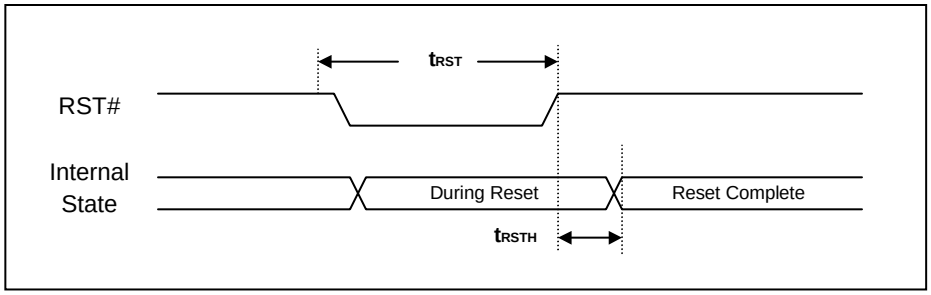


图 8-2A：重置脚位 RST# 的时序

下图 8-3 为一般 RA8803/8822 电源开启或重置(Power On/Reset)的程序说明，此范例也是以 RA8803 使用 320x240 pixel 的 LCD Panel 例子，说明设定方式的流程。

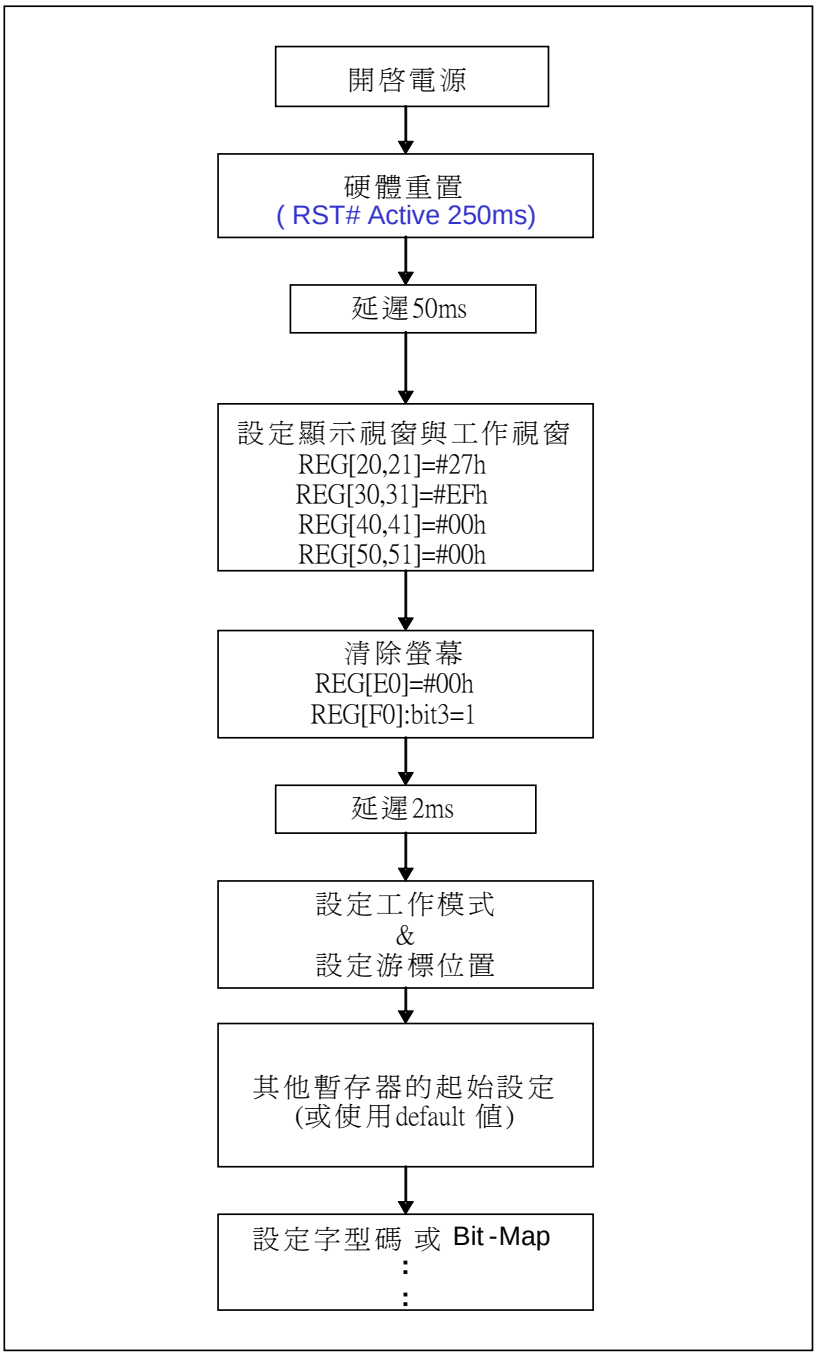


图 8-3：一般 RA8803/8822 电源开启或重置的流程图

8-3 缓存器的起始设定

RA8803/8822 的缓存器在系统电源开启或重置(Power On/Reset)时大都会产生 Default 的值，但是随系统应用不同等因素，若干缓存器最好在电源开启或重置后进行设定，表 8-2 提供基本的缓存器设定范例。

表 8-2：基本的缓存器设定范例

```
LCD_CmdWrite(0x00, 0Xc5) ;      // 设定显示工作模式
LCD_CmdWrite(0x01, 0xF1) ;      // 设定 System Clock, 此例为 4MHz

LCD_CmdWrite(0x10, 0x6B);       // 游标相关信息
LCD_CmdWrite(0x11, 0x22);       // 设定光标高度与行距
LCD_CmdWrite(0x20, 0x27);       // 设定工作窗口(Active window)右边位置 → Segment-Right
                                // 此例为 320x240 , 工作窗口与显示窗口设成一样
LCD_CmdWrite(0x30, 0xEF);       // 设定工作窗口(Active window) 底边位置→ Common-Bottom
LCD_CmdWrite(0x40, 0x00);       // 设定工作窗口(Active window)左边位置 → Segment-Left
LCD_CmdWrite(0x50, 0x00);       // 设定工作窗口(Active window) 顶边位置 → Common-Top
LCD_CmdWrite(0x21, 0x27);       // 设定显示窗口(Display Window)右边位置 → Segment-Right
                                // 此例为 320x240 , 工作窗口与显示窗口设成一样
LCD_CmdWrite(0x31, 0xEF);       // 设定显示窗口(Display Window) 底边位置 → Common-Bottom
LCD_CmdWrite(0x41, 0x00);       // 设定显示窗口(Display Window) 左边位置 → Segment-Le
LCD_CmdWrite(0x51, 0x00);       // 设定显示窗口(Display Window) 顶边位置 → Common-Top
LCD_CmdWrite(0x60, 0x00);       // 设定光标 Segment 地址
LCD_CmdWrite(0x70, 0x00);       // 设定光标 Common 地址
LCD_CmdWrite(0x80, 0x06);       // 光标闪烁时间设定
LCD_CmdWrite(0x90, 0x06);       // 设定 XCK 讯号周期
LCD_CmdWrite(0x81, 0x00);       // 缓存器[81h]设成 00h
LCD_CmdWrite(0xC0, 0xC0);       // 触控屏幕功能, 没用到可不设定
LCD_CmdWrite(0xC1, 0x35);       // 触控屏幕相关信息, Bit[5:4] 必须设成 11b
LCD_CmdWrite(0xD0, 0x11);       // LCD 亮度控制(DAC 功能) , 没用到可不设定
```

注: LCD_CmdWrite 是一写入 Data 到缓存器的子程序，请参考附录 G-3 子程序范例。

8-4 Wakeup 的程序

当 REG[00]之 bit7-6 为"00"，则进入关闭模式(OFF MODE),若是要做唤醒的动作(Wake-UP)，此时可使用三种方式，将 RA8803/8822 唤醒。

1. 利用 MPU 将 缓存器[00]的 bit7-6 再设定为"11"，就可回到正常模式(Normal Mode) 。

2. Touch Panel 中断功能:

设定缓存器[A0]的 bit2 为"1"及缓存器[C0]的 bit3="1"，当整个系统进入 OFF-mode 之后，若此时有任何触控屏幕的动作，则 RA8803/8822 将产生中断讯号 INT 由"0"到"1"的变化，此控制讯号可连接至微处理机做其它动作的延续。

缓存器设定方法可参考下面程序:

```
unsigned char intr=LCD_CmdRead(0xA0) | 0x04 ;
unsigned char tpcr=LCD_CmdRead(0xC0) & 0xf8 ;
tpcr |= 0x80;

LCD_CmdWrite(0xA0,intr);          // REG[A0]:bit2=1
LCD_CmdWrite(0xC0,tpcr);          // REG[C0]:bit[3..0]=1000
   :
   :
```

3. Key SCAN 中断功能:

应用方式与触控屏幕相同，也是产生中断 INT 输出讯号。

缓存器设定方法可参考下面程序:

```
unsigned char kscr=LCD_CmdRead(0xA1) | 0x80 ;
unsigned char intr=LCD_CmdRead(0xA0) | 0x08 ;

LCD_CmdWrite(0xA1,kscr);          // REG[A1]:bit7=1(Key Scan 致能)
LCD_CmdWrite(0xA0,intr);          // REG[A0]:bit3=1
   :
   :
```

9. RA8803/8822 功能应用介绍

9-1 文字模式设定

9-1-1 文字显示

RA8803/8822 的文字模式可以支持全角(中文或英文)及半角(英文)的显示，全角文字是以 16x16 的点矩阵组成，半角文字是 8x16 的点矩阵组成，如图 9-1 所示，而图 9-2 是全角(中文)及半角(英文)文字的混和显示：

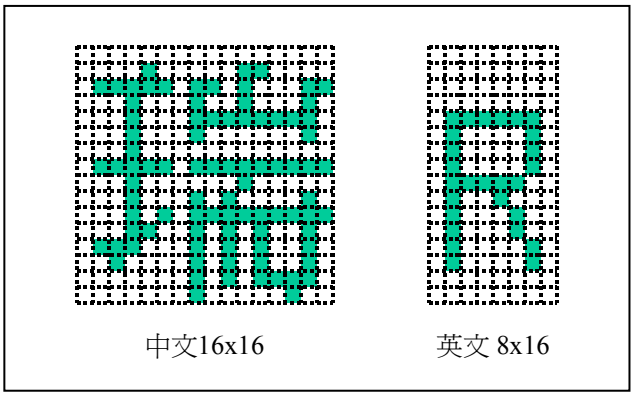


图 9-1：全角与半角文字

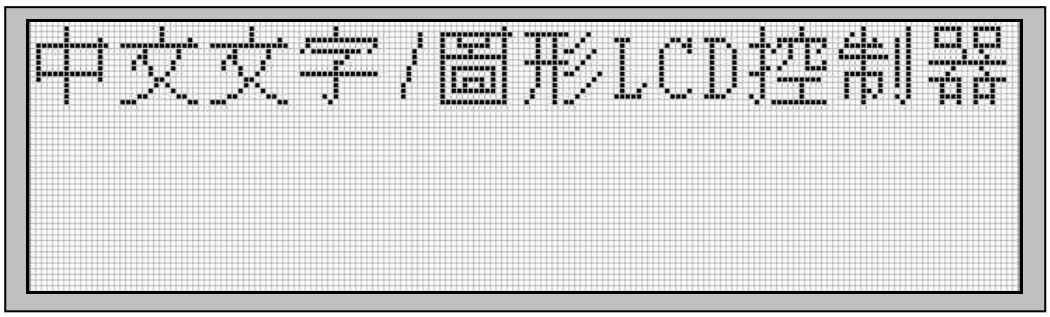


图 9-2：全角与半角文字的混和显示

RA8803/8822 的中文显示方式与传统的 LCD Controller 不同，传统的 LCD Controller 是在绘图模式下，以 Bit-Map 的方式去绘出中文，RA8803/8822 的中文显示方式则是在文字模式，直接输入中文字码(GB 或 BIG5 码)，就可以在光标所在位置显示中文。因为中文字码占两个 Byte，所以如果 MPU 接口是 8-Bit，则 MPU 必须分两次将中文字码(High Byte & Low Byte)写入 RA8803/8822，而英文或数字码只占一个 Byte，因此只要将内码一次写入 RA8803/8822 既可。RA8803 支持之最大显示像素范围为 320x240 点，若以显示文字为例，全角字型（16x16）即是 20 行 x15 列，半角字型（8x16）则可以显示到 40 行 x15 列。RA8822 支持之最大显示像素范围为 240x160 点，若以显示文字为例，全角字型即是 15 行 x10 列，半角字型则可以显示到 30 行 x10 列。表 9-1 为图 9-2 所示

之全角(中文)与半角文字的字型码，下面例题程序就是说明如何显示图 9-2 的画面。

表 9-1：文字码的对照表（BIG5）

显示字型	字型码
中	A4A4
文	A4E5
文	A4E5
字	A672
/	2F
图	B9CF
形	2F
L	4C
C	43
D	44
控	B1B1
制	A8EE
器	BEB9

例 题：

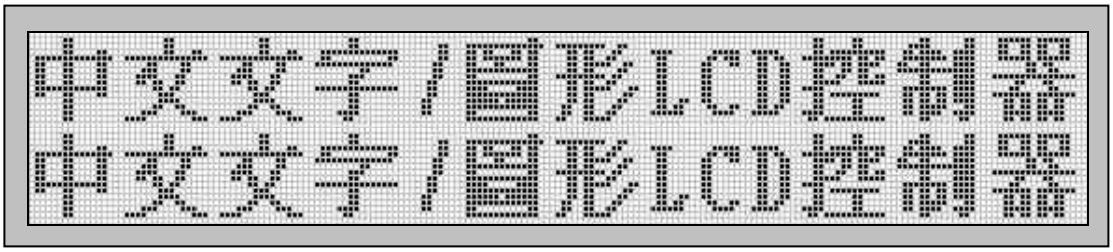
```
MOV     A,#A4H              ; 写入 “中” 的字型码 High Byte
CALL    RegData_Write
MOV     A,#A4H              ; 写入 “中” 的字型码 Low Byte
CALL    RegData_Write       ; 在光标所在位置会显示 “中”

MOV     A,#A4H              ; 写入 “文” 的字型码 High Byte
CALL    RegData_Write
MOV     A,#E5H              ; 写入 “文” 的字型码 Low Byte
CALL    RegData_Write       ; 在光标所在位置会显示 “文”
  :
  :
```

9-1-2 粗体字之显示功能

RA8803/8822 的中英文显示都可以秀出粗体字的显示效果，图 9-3 说明欲表现粗体字的显示效果时，缓存器应如何设定。

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
4	**设定粗体字型(仅文字模式适用)** 0：正常字型 1：粗体字型	Text	1h	R/W



1. 设定缓存器[10h] bit4=1
2. 写入中文字型码

图 9-3：粗体字的显示

9-2 绘图模式设定

RA8803/8822 的绘图模式是以字符映像(bit map)方式填入图形数据在 Display RAM 上，图 9-4 说明进入绘图模式时，缓存器要如何设定：

1. 设定 REG[00h] bit3=0
2. 使用字符映像(bit map)方式填入图形数据

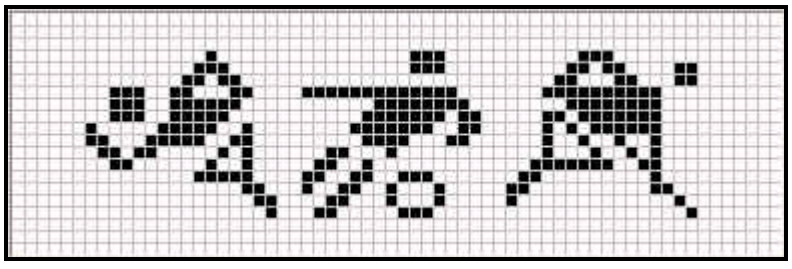


图 9-4：绘图模式的显示

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
3	**选择显示工作模式** 1：文字模式，写入的数据会被视为是 GB/BIG/ASCII 等字码。 0：绘图模式，写入的数据会被视为是 Bit-Map 的模式。	--	1h	R/W

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
7	**图形模式时，光标自动移位的方向选择** 1：先水平移动再垂直移动 0：先垂直移动再水平移动	1h	R/W

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
7	**设定当数据读出 DDRAM 时，光标是否自动移位。** 1：致能(自动移位) 0：禁能(不自动移位)	Text/Graph	0h	R/W
3	**此位用来设定当数据写入 DDRAM 时，光标是否自动移位** 1：致能(自动移位) 0：禁能(不自动移位)	Text/Graph	1h	R/W

RA8803 支持之最大显示像素范围为 320 点 x240 点，因此需要约 9.6K Byte 的 Display Data RAM (DDRAM)来储存欲显示的每个像素点，而 RA8822 支持之最大显示像素范围为 240 点 x 160 点，因此需要约 4.8K Byte 的 Display Data RAM 来储存欲显示的每个像素点，在 DDRAM 里，只有在显示范围内的对应资料会被显示于 LCD 面板上，不在显示范围内的则会被忽略掉。当 RA8803/8822 在显示图形的时候，是以字符映像(Bit Map)的方式写入 DDRAM，若 DDRAM 的某个位置被填满为 '1' 时，相对于 LCD 面板的位置会被显示出亮点，由图 9-5 可看出，在 DDRAM 上所储存之像素资料，会对应到显示屏幕(LCD)上，而构成文字、符号或图形之显示效果。

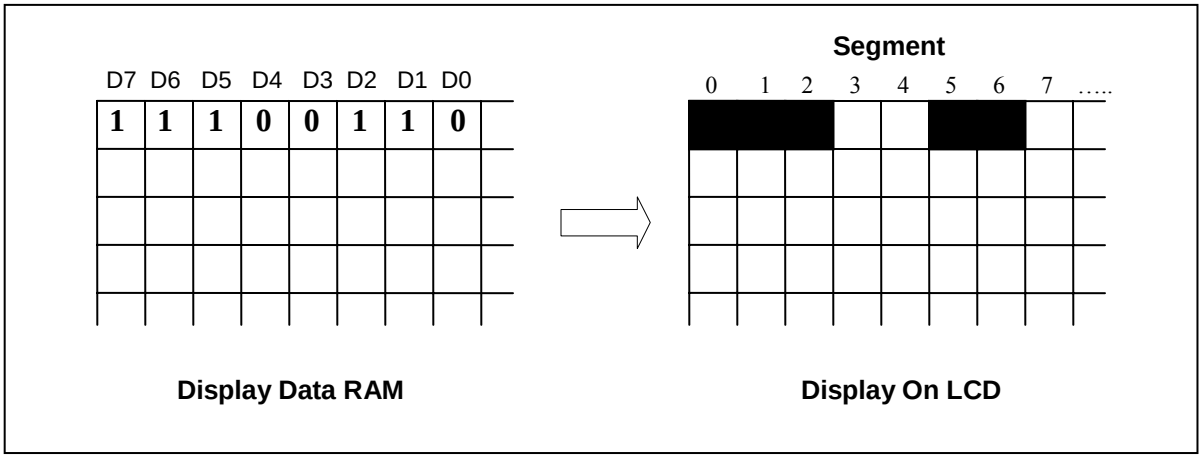


图 9-5：Display Data 到 LCD 显示的映射

以下程序就是以图 9-5 做例子，用绘图模式在 LCD Panel 的左上角秀出 Pattern：

例 题：(8051-ASM)

```
MOV     A,#60h                  ; 选择光标设定缓存器 (CPXR)
CALL    RegAddr_WRITE

MOV     A,#00h                  ; 设定坐标 X=0
CALL    RegAddr_WRITE

MOV     A,#70h                  ; 选择光标设定缓存器 (CPYR)
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定坐标 Y=0
CALL    RegAddr_WRITE           ; 设定光标位置为(0,0)

MOV     A,#E6H                  ;在 LCD Panel 的左上角秀出 “E6” 的图形 Pattern
CALL    RegData_Write
```

例 题：(8051-C)

```
LCD_CmdWrite(0x60,0x00);        // 设定坐标 X=0
LCD_CmdWrite(0x70,0x00);        // 设定坐标 Y=0

LCD_DataWrite(0xE6);            //在 LCD Panel 的左上角秀出 “E6” 的图形 Pattern
```

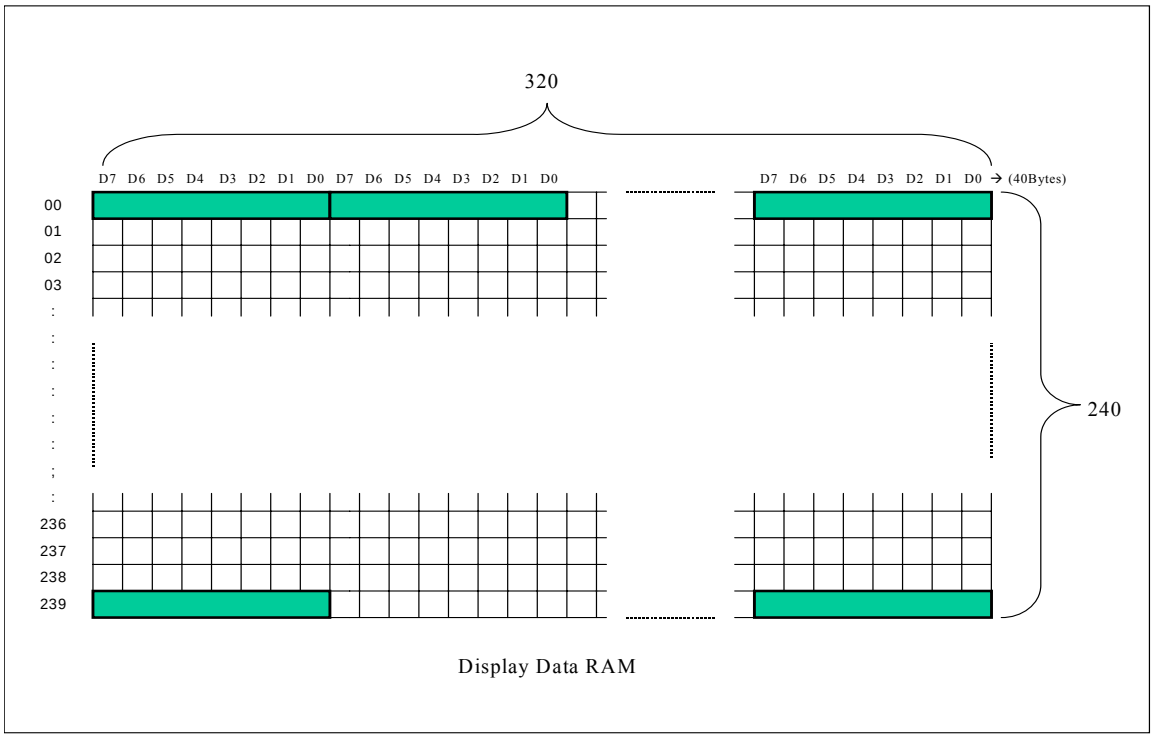


图 9-6：Display Data RAM 的格式(320 x 240)

在绘图模式下，缓存器[12h]的 Bit7 用来选择光标的移动是先水平移动再垂直移动或是先垂直移动再水平移动，如图 9-7A。

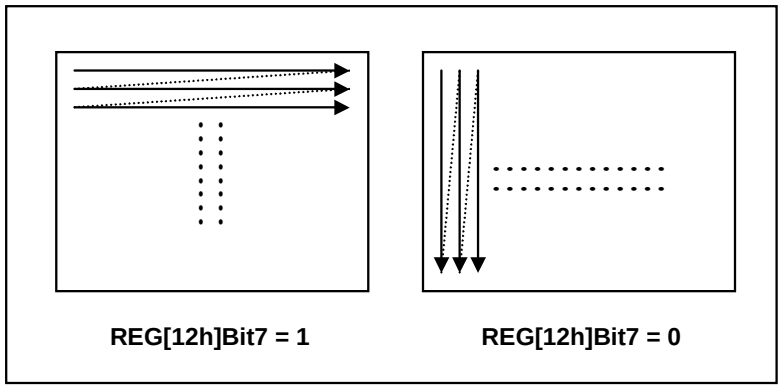


图 9-7A：选择光标的移动

图 9-7B 范例:

```
MOV     A,#12h                  ; 选择缓存器 [12h] (MAMR)
CALL    RegAddr_WRITE
MOV     A,#91h                  ; Bit7=1, 先水平移动再垂直移动
CALL    RegAddr_WRITE
MOV     A,#11H                  ;在 LCD Panel 的左上角秀出 “11” 的图形 Pattern
CALL    RegData_Write
MOV     A,#22H                  ;在 LCD Panel 的左上角秀出 “22” 的图形 Pattern
CALL    RegData_Write
MOV     A,#33H                  ;在 LCD Panel 的左上角秀出 “33” 的图形 Pattern
CALL    RegData_Write
MOV     A,#44H                  ;在 LCD Panel 的左上角秀出 “44” 的图形 Pattern
CALL    RegData_Write
```

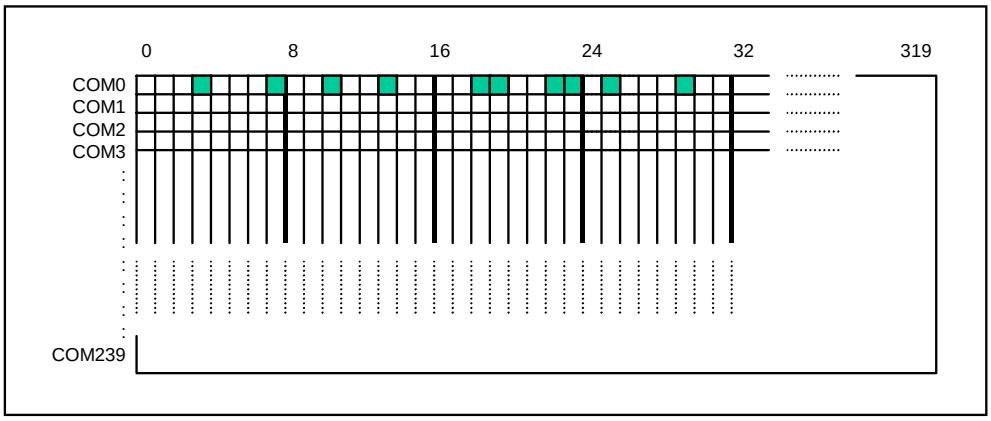


图 9-7B：光标先水平移动再垂直移动

图 9-7C 范例:

```
MOV     A,#12h                  ; 选择缓存器 [12h] (MAMR)
CALL    RegAddr_WRITE
MOV     A,#11h                  ; Bit7=0, 先垂直移动再水平移动
CALL    RegAddr_WRITE
MOV     A,#11H                  ;在 LCD Panel 的左上角秀出 “11” 的图形 Pattern
CALL    RegData_Write
MOV     A,#22H                  ;在 LCD Panel 的左上角秀出 “22” 的图形 Pattern
CALL    RegData_Write
MOV     A,#33H                  ;在 LCD Panel 的左上角秀出 “33” 的图形 Pattern
CALL    RegData_Write
MOV     A,#44H                  ;在 LCD Panel 的左上角秀出 “44” 的图形 Pattern
CALL    RegData_Write
```

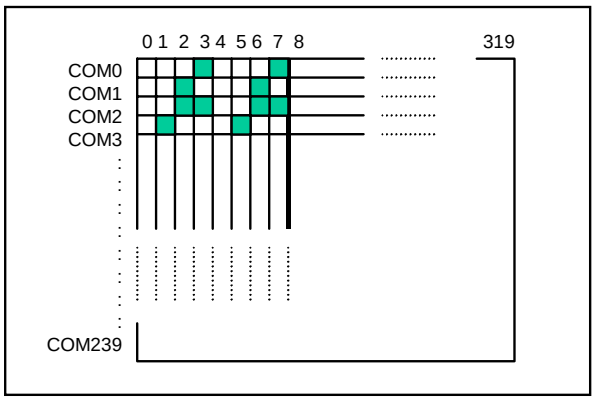


图 9-7C：光标先垂直移动再水平移动

在绘图模式下，若要读取 Display RAM 的数据时，也是由缓存器[12h]的 Bit7 用来选择光标的移动是先水平移动再垂直移动或是先垂直移动再水平移动，如图 9-7A。不论写入或读取 Display RAM 的数据都必须注意光标的设定是否有自动加一的功能，也就是缓存器[10h]的 Bit7 与 Bit3。如图 9-7D 是代表缓存器[12h]Bit7=1(先水平移动再垂直移动)时 Display RAM 数据的读取方向(以 320x240 显示器为例)。

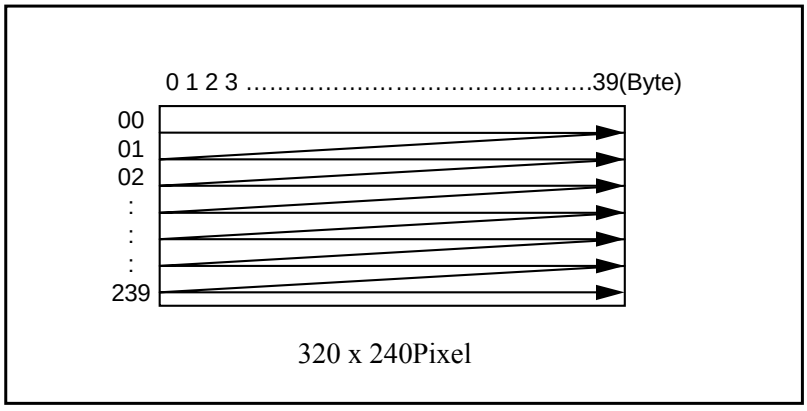


图 9-7D：图形模式时数据读取方向

9-3 闪烁与反白显示

9-3-1 闪烁显示

图 9-8 说明要闪烁显示时，缓存器要如何设定：

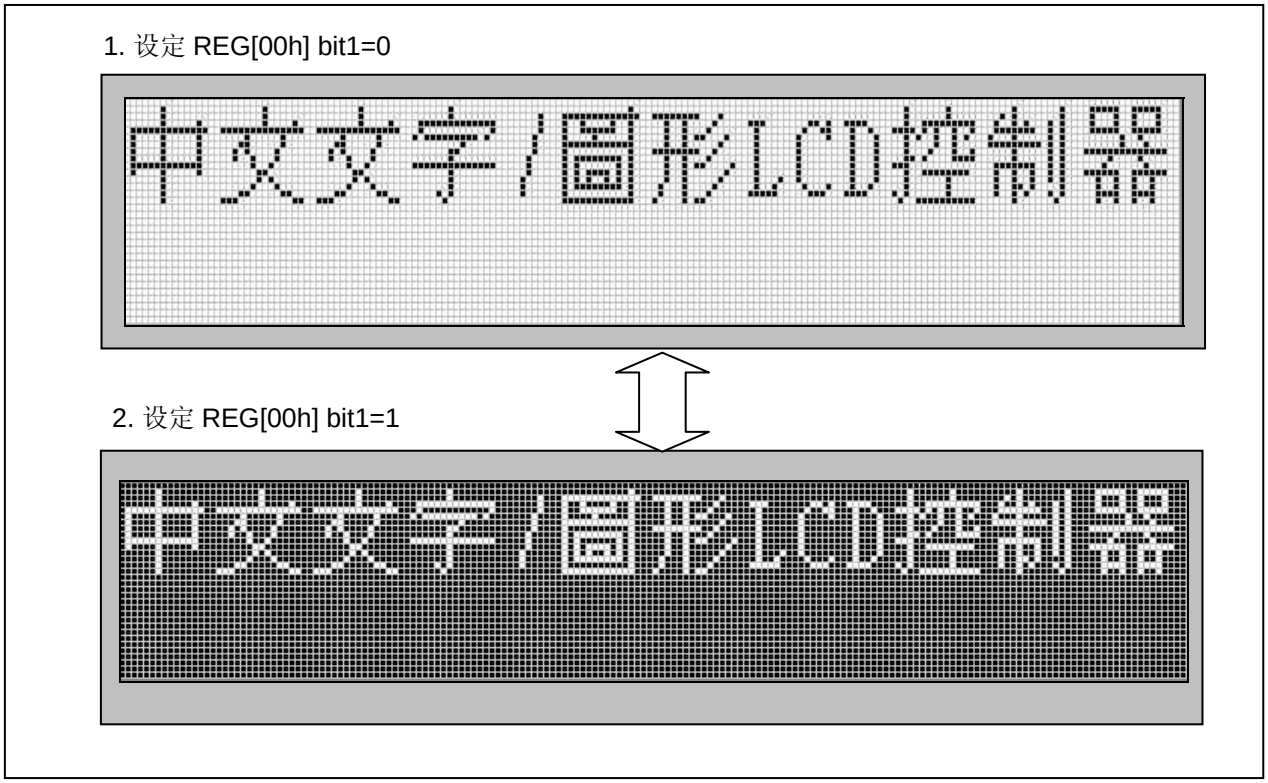


图 9-8：屏幕闪烁

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
1	**闪烁模式选择** 0：正常显示 1：整个屏幕闪烁，闪烁时间由缓存器[80h]BTR 来设定	Text/Graph	0h	R/W

9-3-2 屏幕反白

如果要将 LCD 画面全部反白只要设定缓存器[00]的 Bit0 既可，图 9-9 说明要反白显示时，缓存器要如何设定：

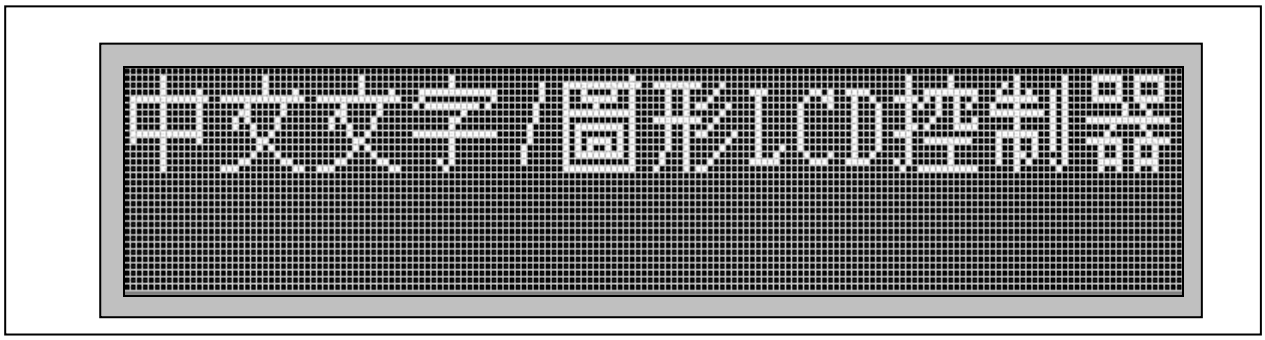


图 9-9：屏幕反白

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
0	屏幕反白模式选择 1：正常显示 0：全屏幕反白显示，DDRAM 内的资料会被全部反相。	Text/Graph	1h	R/W

9-3-3 文字反白

如果要将 LCD 画面秀出反白的字体只要设定缓存器[10]的 Bit5 既可，图 9-10 说明要反白显示时，缓存器要如何设定：

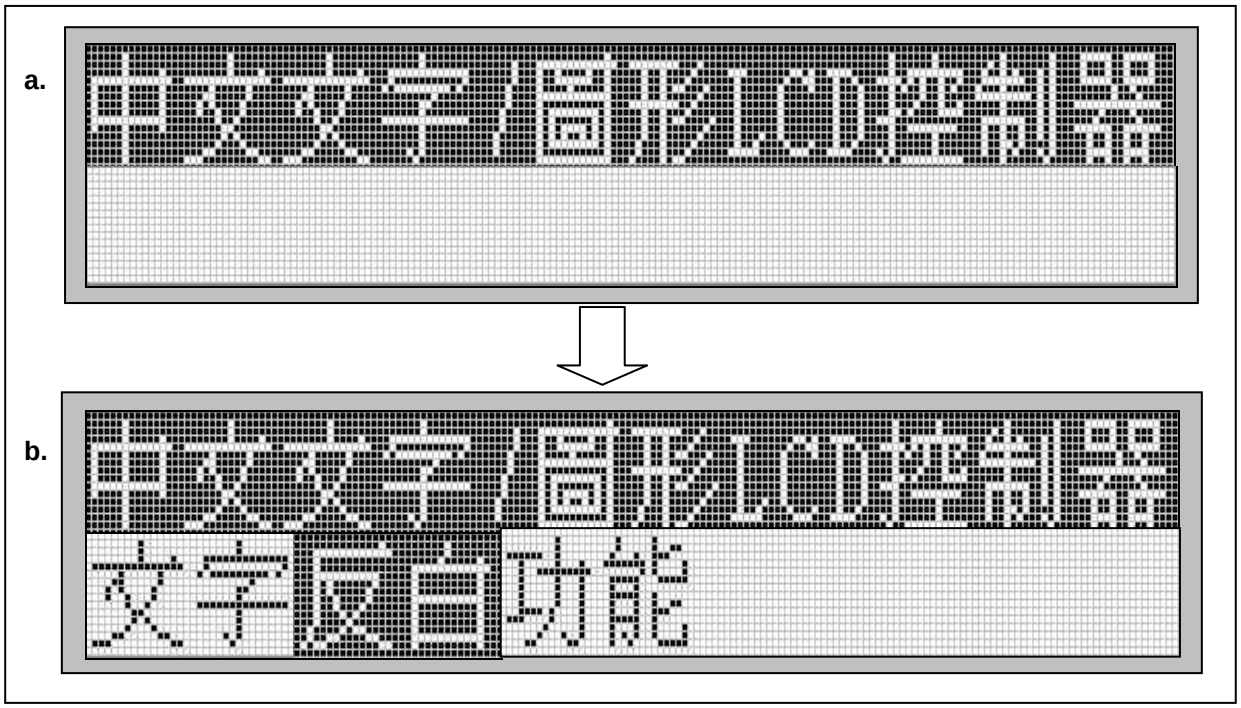


图 9-10：反白显示

(a)

1. 设定缓存器[10h] bit5=0
2. 写入"中文文字/图形 LCD 控制器"的 BIG5 码，然后可显示出"中文文字/图形 LCD 控制器"

(b)

3. Hold on (a)
4. 设定缓存器[10h] bit5=1
5. 写入"文字"的 BIG5 码，LCD 就可显示出"文字"
6. Hold on
7. 设定缓存器[10h] bit5=0
8. 写入"反白"的 BIG5 码，LCD 就可显示出"反白"字样
9. Hold on
10. 设定缓存器[10h] bit5=1
11. 写入"功能"的 BIG5 码，LCD 就可显示出 "功能"

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
5	**储存 MPU 进来数据(正相/反相)于 DDRAM** 1：直接储存数据于 DDRAM 中 0：存入相反的数据于 DDRAM 中	Text	1h	R/W

9-4 中/英文文字对齐

由于英文字体与中文字体所占的宽度不一样，因此在显示中文英文都有的画面时必须考虑整体显示效果，RA8803/8822 可以设定中文英文显示时不同行的显示效果以决定文字是否对齐，图 9-11 说明要表现出中英文文字 “对齐” 之情形时，缓存器要如何设定：

1. 设定 缓存器 WCCR,ALG=1
2. 写入“中文文字/图形 LCD 控制器” 两次，则屏幕会秀出”中文文字/图形 LCD 控制器” ←上下两行文字对齐

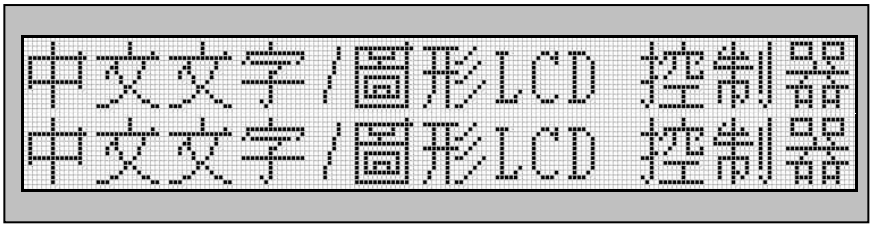


图 9-11：文字对齐的显示范例

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
6	**中/英文字对齐** 1：致能 0：禁能 此功能仅在文字模式时有效，可以将全角与半角混合显示时作对齐调整。	Text	1h	R/W

图 9-12 说明要表现出中英文文字 “不对齐” 之情形时，缓存器要如何设定：

1. 设定 缓存器 WCCR,ALG=1
2. 写入“中文文字/图形 LCD 控制器”
3. 设定 缓存器 WCCR,ALG=0
4. 写入“中文文字/图形 LCD 控制器” ←上下两行文字没对齐

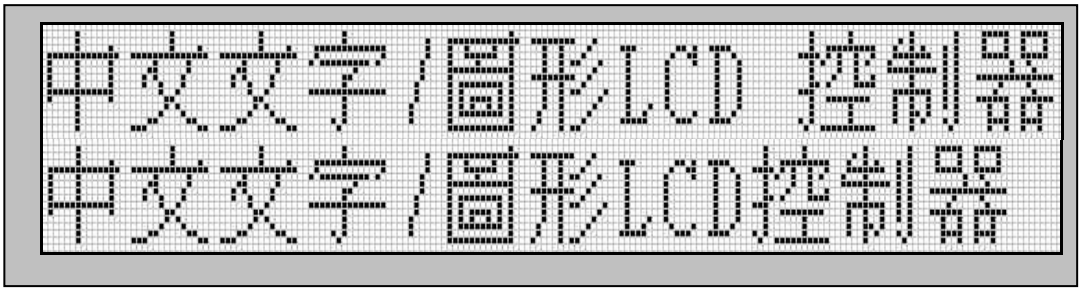


图 9-12：文字不对齐的显示范例

9-5 LCD 屏幕显示 On/Off 设定

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
2	**设定屏幕显示为开启或关闭**，此位用来控制连接到 LCD 驱动 IC 接口的 “DISPOFF#” 信号 1： “DISPOFF” 讯号输出 High(屏幕开启) 0： “DISPOFF” 讯号输出 Low(屏幕关闭)	Text/Graph	0h	R/W

9-6 光标 On/Off 设定

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
2	**光标显示 On/Off 设定** 1：设定光标显示 On 0：设定光标显示 Off	Text/Graph	0h	R/W

9-7 光标位置与移位设定

9-7-1 光标位置

缓存器[60h]CPXR 的 Bit[5..0]用来设定光标的 Segment 地址，RA8803/8822 可以分别支持 320 (Segment) x 240 (Common) 或 240x160 的 Panel Size，但是光标的 Segment 地址是以每 8-Bit 为单位，例如，想在 Panel 的左上角秀出 “控”，则必须设定光标缓存器 CPXR = 00h，CPYR = 00h，又例如想在 Panel 的左上角第三个全角位置秀出 “制”，则必须设定光标缓存器 CPXR = 04h，CPYR = 00h，同理，想在 Panel 的左上角第二行第一个全角位置秀出 “器”，则必须设定光标缓存器 CPXR = 00h，CPYR = 10h，请参考图 9-13。

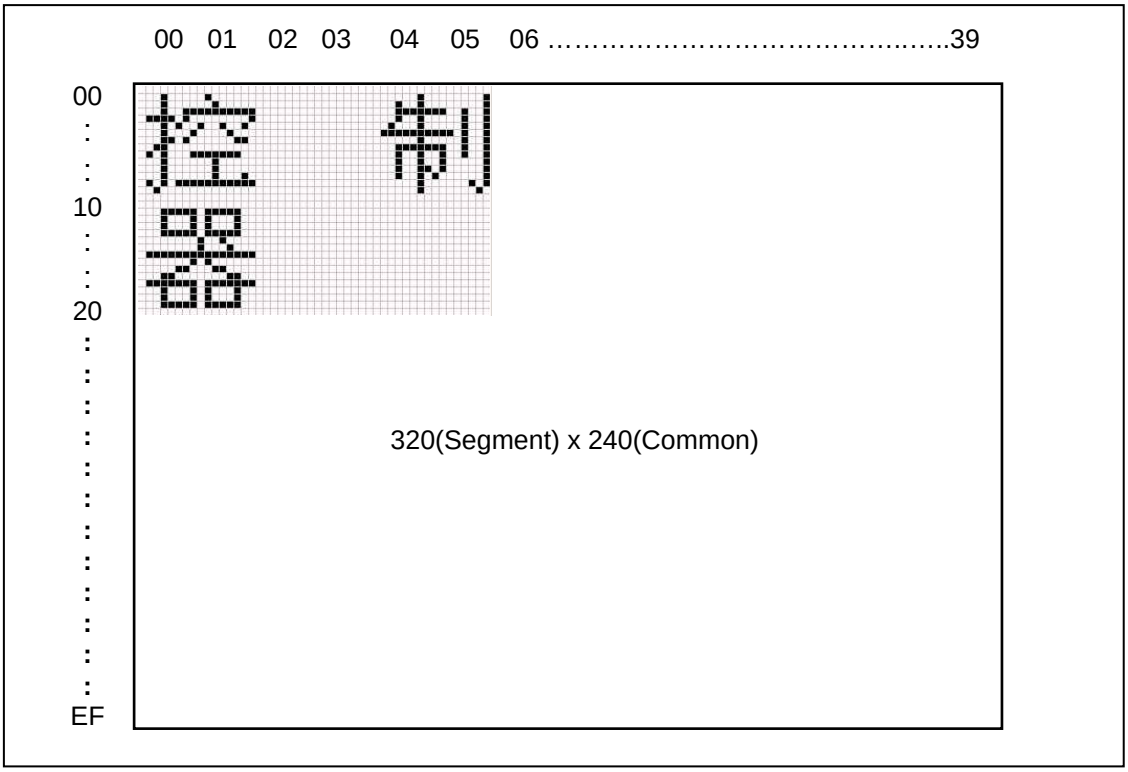


图 9-13：光标位置设定的显示范例

REG [60h] Cursor Position X Register (CPXR)

Bit	Description	Default	Access
5-0	设定光标 **Segment** 地址	0h	R/W

REG [70h] Cursor Position Y Register (CPYR)

Bit	Description	Default	Access
7-0	设定光标 **Common** 地址	0h	R/W

不论文字或是绘图模式，都是使用缓存器[60h]CPXR 与[70h]CPYR 来设定光标的地址，如下图 9-14 所示，在绘图模式下设定光标缓存器 CPXR = 00h，CPYR = 10h，则由 DDRAM 读到的的数值会是 “00h”，如果缓存器 CPXR = 00h，CPYR = 12h，DDRAM 读到的数值会是 “78h”，又如缓存器 CPXR = 00h，CPYR = 14h，DDRAM 读到的数值会是 “0Ah”。请参考图 9-15。

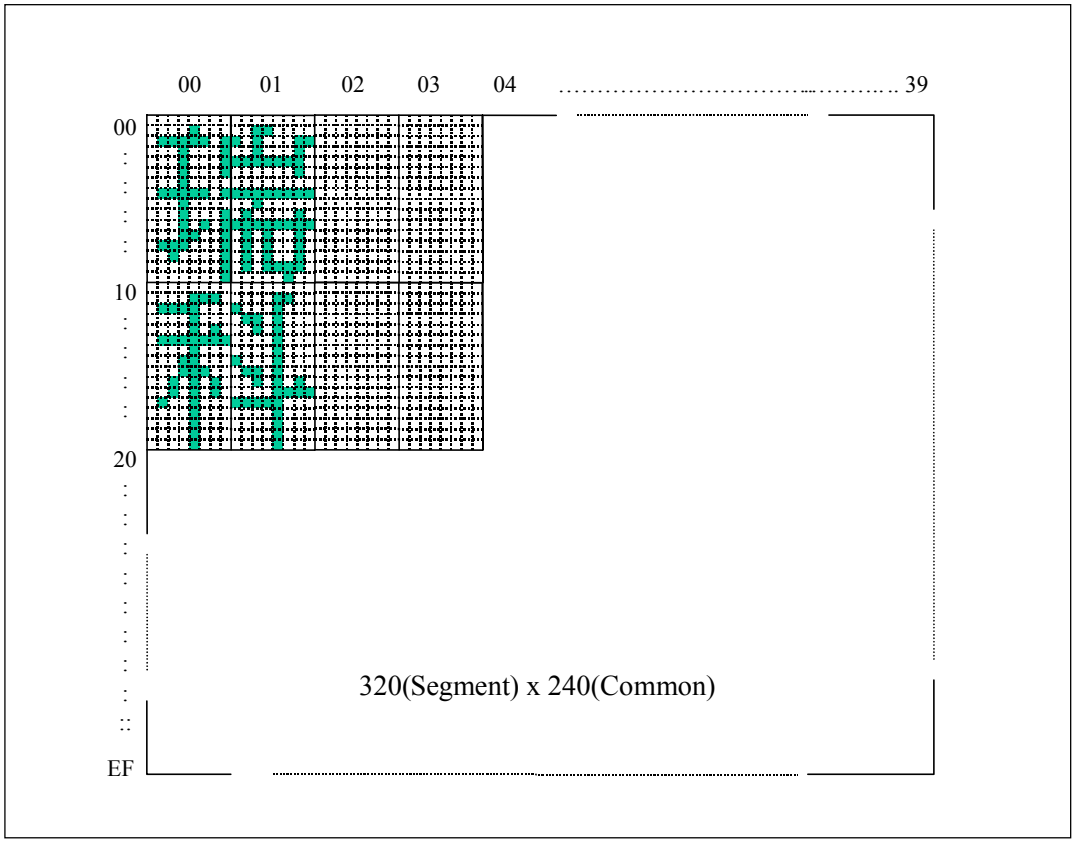


图 9-14：光标位置设定与 DDRAM 的读取范例

缓存器[60h]CPXR 与[70h]CPYR 的光标地址是属于绝对地址，不会因工作窗口大小而改变，也就是(0 ,0)始终是在屏幕的左上角，请参考 9-10 节的说明。

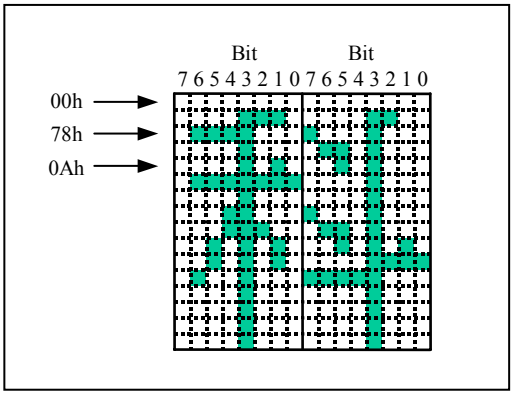


图 9-15：将图 9-14 的 “科”字放大

9-7-2 游标移位

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
7	**设定当数据读出 DDRAM 时，光标是否自动移位。** 1：致能 (自动移位) 0：禁能(不自动移位)	Text/Graph	0h	R/W
3	**此位用来设定当数据写入 DDRAM 时，光标是否自动移位** 1：致能 (自动移位) 0：禁能(不自动移位)	Text/Graph	1h	R/W

请参考 9-2 节绘图模式设定的说明。

9-8 光标闪烁设定

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
1	**光标闪烁控制** 1：光标闪烁，闪烁时间由缓存器[80h] BTR 决定。 0：游标不闪烁	Text/Graph	0h	R/W

REG [80h] Blink Time Register (BTR)

Bit	Description	Text/Graph	Default	Access
7-0	**光标/屏幕闪烁时间设定** 闪烁时间 = Bit[7..0] x (1/Frame_Rate)	Text/Graph	23h	R/W

如果 Frame Rate = 60Hz，则 1/Frame_Rate = 1/60Hz = 1.67ms，光标闪烁时间 = REG[80h] x 1.67ms，例如设定 REG[80h] = 35h = 53(十进制)，因此光标闪烁时间 = 53 x 16.7ms = 885ms。

9-9 光标高度与宽度设定

9-9-1 游标高度

RA8803/8822 在做文字显示时，有提供光标高度的设定，在正常显示文字时，光标的高度为一个 Pixel 的高度，但依不同使用者的需要，提供了 Pixel 的高度的设定，Pixel 的高度设定范围为(1~16)Pixel，使用者可依需求来决定光标的高度大小。

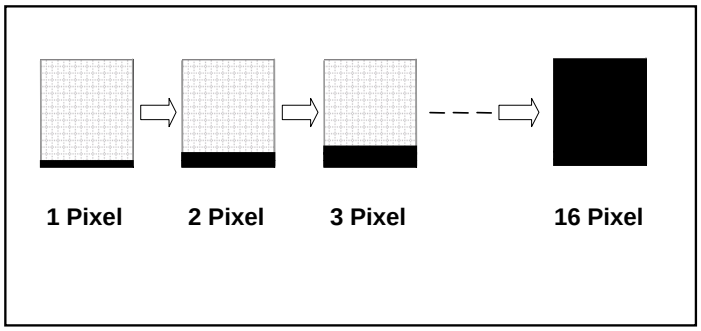


图 9-16：光标高度之设定

REG [11h] Cursor Height and Lines Distance Register (CHLD)

Bit	Description	Text/Graph	Default	Access
7-4	**设定光标高度**	Text	0010h	R/W

9-9-2 游标宽度

RA8803/8822 在做文字显示时，有提供两种光标宽度的设定。第一种为 REG[10h] bit0=0 时，光标的宽度将会固定为 1 个 Byte 的宽度(也就是 8 个 Pixel)。第二种为 REG[10h] bit0=1 时，光标的宽度会随着所输入文字来做变化，例如当输入一个全角字时，文字后面的光标宽度会自动变为 2 个 Byte(也就是 16 个 Pixel)。当输入一个半角字时，文字后面的光标宽度会自动变为 1 个 Byte。

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
0	**设定光标宽度** 1：会随着输入的数据而变动光标宽度，当数据为半型时，光标为一个字节宽度(8 个 Pixel)，当数据为全型时，光标为二个字节宽度(16 个 Pixel)。 0：光标固定为一个字节的宽度(8 个 Pixel)	Text	0h	R/W

9-10 工作及显示窗口大小设定

RA8803/8822 应用在面板的显示上，供使用者有两种窗口选择。一个是显示窗口(Display Window)，一个是工作窗口(Active Window)。显示窗口(Display Window)是实际 LCD 面板的大小，而工作窗口(Active Window)是在实际的显示窗口(Display Window)内设定比显示窗口小的子窗口。

例如面板大小为 320x240，而它的显示窗口就为 320x240。在显示窗口(320x240)内可依使用者需要，来设定工作窗口的大小，也就是子窗口的大小，子窗口也可在显示窗口内任意调整所要放置的地方。以下是相关的缓存器说明：

REG [21h] Display Window Right Register (DWRR)

Bit	Description	Default	Access
5-0	**设定显示窗口(Display Window)右边位置 → Segment-Right (注 1)** **Segment_ Right = (Segment Number / 8) – 1** RA8803: 如果 LCD Panel 为 320x240，则此缓存器的值为： (320 / 8) – 1 = 39 = 27h RA8822: 如果 LCD Panel 为 240x160，则此缓存器的值为： (240 / 8) – 1 = 29 = 1Dh	xxh	R/W

REG [31] Display Window Bottom Register (DWBR)

Bit	Description	Default	Access
7-0	**设定显示窗口(Display Window) 底边位置 → Common_Bottom** **Common_Bottom = LCD Common Number –1** RA8803: 如果 LCD Panel 为 320x240，则此缓存器的值为： 240 – 1 = 239 = EFh RA8822: 如果 LCD Panel 为 240x160，则此缓存器的值为： 160 – 1 = 159 = 9Fh	xxh	R/W

REG [41] Display Window Left Register (DWLR)

Bit	Description	Default	Access
7-0	**设定显示窗口(Display Window) 左边位置 → Segment-Left (注 1)** 通常将此缓存器的值设定为 “00h”。	xxh	R/W

REG [51] Display Window Top Register (DWTR)

Bit	Description	Default	Access
7-0	**设定显示窗口(Display Window) 顶边位置 → Common-Top (注 1)** 通常将此缓存器的值设定为 “00h”。	xxh	R/W

注 1：光标地址应设定在显示窗口的范围内，因此缓存器[60h, 70h]、[B0h, B1h]与[21h, 31h, 41h, 51h]的设定必须遵照以下的规范：

1. DWRR≥ AWRR≥ CPXR≥ AWLR≥ DWLR
2. DWBR≥ AWBR≥ CPYR≥ AWTR≥ DWTR

REG [20h] Active Window Right Register (AWRR)

Bit	Description	Default	Access
5-0	**设定工作窗口(Active window)右边位置 → Segment-Right (注 2)**	xxh	R/W

REG [30h] Active Window Bottom Register (AWBR)

Bit	Description	Default	Access
7-0	**设定工作窗口(Active window) 底边位置→ Common-Bottom (注 2)**	xxh	R/W

REG [40h] Active Window Left Register (AWLR)

Bit	Description	Default	Access
5-0	**设定工作窗口(Active window)左边位置 → Segment-Left (注 2)**	xxh	R/W

REG [50h] Active Window Top Register (AWTR)

Bit	Description	Default	Access
7-0	**设定工作窗口(Active window) 顶边位置 → Common-Top (注 2)**	xxh	R/W

注 2：REG [20h, 30h, 40h, 50h] 可作为换行/换页的功能，可让使用者利用这 4 个 Register 自行设定一个区块为工作窗口(Active Window)。当数据超过窗口的右边界 REG [20h, 30h, 40h, 50h]所设定的值，光标会自动换行(也就是光标移到工作窗口的左边界 REG[40h]所设定的值)，继续将数据写入。当数据写入到工作窗口的右下角时（REG[20h]与[30h]所设定的值），会自动把光标移到工作窗口的的左上角(REG[40h, 50h]所设定的值)，继续的将数据填入窗口。

设定完工作窗口后，光标地址不会自动移到工作窗口的范围内，因为缓存器[60h]CPXR 与[70h]CPYR 的光标地址是属于绝对地址，不会因工作窗口大小而改变，也就是(0 ,0)始终是在屏幕

的左上角，因此设定完工作窗口后想要进行秀字，必须先将光标地址设定在工作窗口的范围内，之后光标地址就只会在工作窗口的范围内移动。

下面的例题 1 与 2 是以 RA8803 为例，设定 LCD Panel 的显示窗口为 320x240，工作窗口为 160x160 且位于显示窗口的左上角，如图 9-17A 所示。

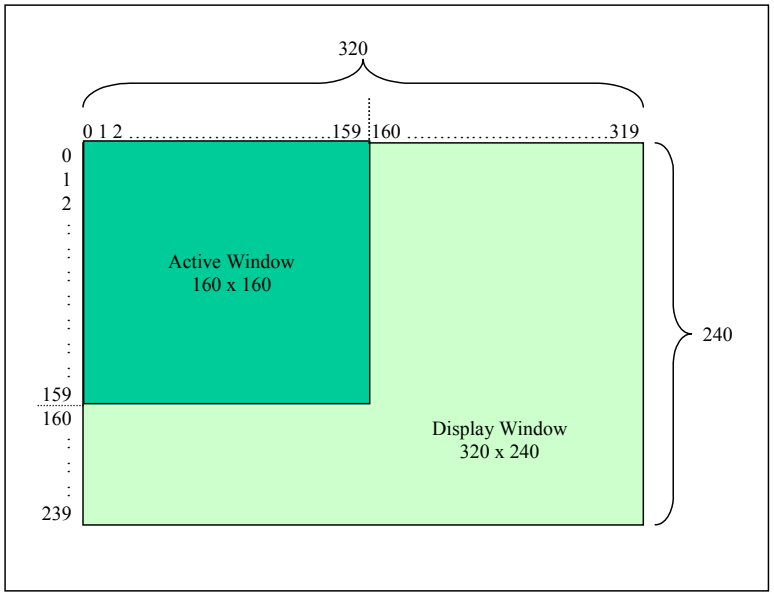


图 9-17A：例题 1 与例题 2 的显示窗口与工作窗口

例 题 1：(8051-ASM)

```
MOV     A,#21h                  ; 设定 Display Window is 320x240 pixel
CALL    RegAddr_WRITE
MOV     A,#27h                  ; 设定 DWRR = (320/8) –1 = 39 = 27h
CALL    RegAddr_WRITE
MOV     A,#31h
CALL    RegAddr_WRITE
MOV     A,#EFh                  ; 设定 DWBR = 240 –1 = 239 = EFh

MOV     A,#41h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#51h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h

MOV     A,#20h                  ; 设定 Active Window is 160x160 pixel
CALL    RegAddr_WRITE
MOV     A,#13h                  ; 设定 AWRR = (160)/8 –1 = 19 = 13h
CALL    RegAddr_WRITE
MOV     A,#30h
```

```
CALL    RegAddr_WRITE
MOV     A,#9Fh                  ; 设定 AWBR = 160 –1 = 159 = 9Fh

MOV     A,#40h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#50h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup the AWLR, AWTR = 00h
```

例 题 2：(8051-C)

```
LCD_CmdWrite(0x21,0x27);        // Display Window Right Register(DWRR)
LCD_CmdWrite(0x31,0xEF);        // Display Window Bottom Register(DWBR)
LCD_CmdWrite(0x41,0x00);        // Display Window Left Register(DWLR)
LCD_CmdWrite(0x51,0x00);        // Display Window Top Register(DWTR)

LCD_CmdWrite(0x20,0x13);        // Active Window Right Register(AWRR)
LCD_CmdWrite(0x30,0x9F);        // Active Window Bottom Register(AWBR)
LCD_CmdWrite(0x40,0x00);        // Active Window Left Register(AWLR)
LCD_CmdWrite(0x50,0x00);        // Active Window Top Register(AWTR)
```

下面的例题 3 与 4 是以 RA8822 为例，设定 LCD Panel 的显示窗口为 240x160，工作窗口为 120x120 且位于显示窗口的左上角，如图 9-17B 所示。

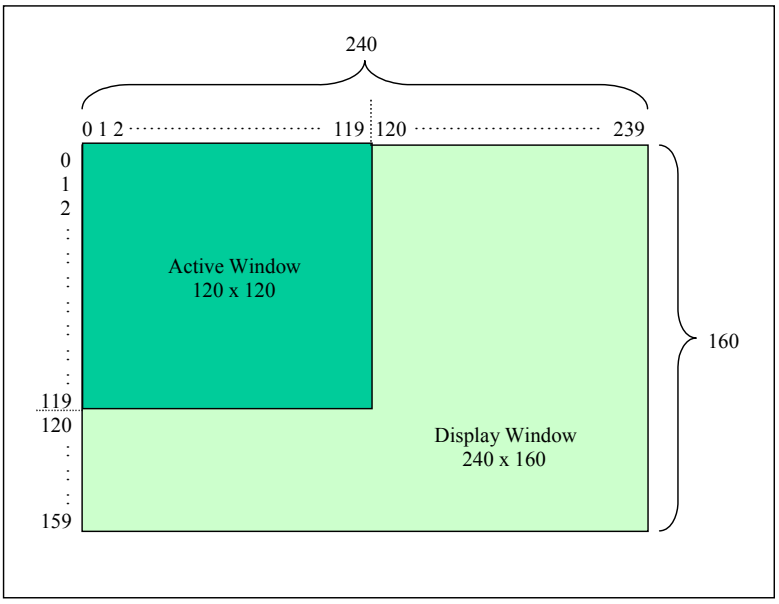


图 9-17B：例题 2 的显示窗口与工作窗口

例 题 3：(8051-ASM)

```
MOV     A,#21h                  ; 设定 Display Window is 240x160 pixel
CALL    RegAddr_WRITE
MOV     A,#1Dh                  ; 设定 DWRR = (240/8) –1 = 29 = 1Dh
CALL    RegAddr_WRITE
MOV     A,#31h
CALL    RegAddr_WRITE
MOV     A,#9Fh                  ; 设定 DWBR = 160 –1 = 159 = 9Fh

MOV     A,#41h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#51h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h

MOV     A,#20h                  ; 设定 Active Window is 120x120 pixel
CALL    RegAddr_WRITE
MOV     A,#0Eh                  ; 设定 AWRR = (120)/8 –1 = 14 = 0Eh
CALL    RegAddr_WRITE
MOV     A,#30h
CALL    RegAddr_WRITE
MOV     A,#77h                  ; 设定 AWBR = 120 –1 = 119 = 77h

MOV     A,#40h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; 设定 DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#50h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup the AWLR, AWTR = 00h
```

例 题 4：(8051-C)

```
LCD_CmdWrite(0x21,0x1D);        // Display Window Right Register(DWRR)
LCD_CmdWrite(0x31,0x9F);        // Display Window Bottom Register(DWBR)
LCD_CmdWrite(0x41,0x00);        // Display Window Left Register(DWLR)
LCD_CmdWrite(0x51,0x00);        // Display Window Top Register(DWTR)

LCD_CmdWrite(0x20,0x0E);        // Active Window Right Register(AWRR)
LCD_CmdWrite(0x30,0x77);        // Active Window Bottom Register(AWBR)
LCD_CmdWrite(0x40,0x00);        // Active Window Left Register(AWLR)
LCD_CmdWrite(0x50,0x00);        // Active Window Top Register(AWTR)
```

9-11 行距设定

RA8803/8822 在做文字显示时，提供了行距设定的功能，尤其是做中文显示时，每一行如果有适当的间隔，LCD 的显示画面看起来会比较舒适。RA8803/8822 行与行相隔的间距设定范围为 1~16 Pixel 的高度，使用者可依需求来决定行与行间距的大小，一旦设定后，当每填完一行的中文字，跳到下一行时，其行距会依照先前所设定的间距来显示。

REG [11h] Cursor Height and Lines Distance Register (CHLD)

Bit	Description	Text/Graph	Default	Access
3-0	**行距设定**	Text	0010h	R/W

9-12 自动填入数据到 DDRAM

REG [E0h] Pattern Data Register (PNTR)

Bit	Description	Text/Graph	Default	Access
7-0	**设定写入到 DDRAM 的数据** 当缓存器[F0h]的 bit3 为 '1'，RA8803/8822 内部将自动读取本缓存器[E0h] 的 Data，然后全部填写到 DDRAM 内，之后缓存器[F0h]的 bit3 被清除为 '0'。	Graph	0h	R/W

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
3	**重复写入 PNTR -- REG [E0h] 的数据到 DDRAM** 1：开始写入 0：未动作 当 FDA 为 '1'，RA8803/8822 内部将自动读取 PNTR 的 Data，填写到 DDRAM 内(Range:[AWLR, AWTR] ~ [AWRR, AWBR])，之后此位会被自动清除为 '0'。	Graph	0h	R/W

例 题：

```
void LCD_Clear(void) small                  //清除屏幕子程序
{
    unsigned char READ_REG;
    LCD_CmdWrite(0xE0,0x00);                //设定 REG[E0] = #00h
    READ_REG = LCD_CmdRead(0xF0);
    READ_REG &= 0xF7;
    READ_REG |= 0x08;
    LCD_CmdWrite(0xF0,READ_REG);            //设定 REG[F0] :bit3=1
}                                           //此时 DDRAM 就全部自动被写入”00”,
                                            //而达到快速清除屏幕的动作。
```

9-13 屏幕更新频率设定

REG [90h] Shift Clock Control Register (SCCR)

Bit	Description	Default	Access
7-0	**设定 XCK 讯号周期** **SCCR = (SCLK x DBW) / (Column x Row x FRS)** SCLK：系统频率(System Clock) (单位：Hz) DBW：LCD 驱动器 的 Data Bus 宽度(单位：Bit) Column：LCD 面板的 Segment 大小(单位：Pixel) Row：LCD 面板的 Common 大小 (单位：Pixel) FRS：LCD 面板的 Frame Rate(单位：Hz) 限制条件 SYS_DW=0，LCD 的 Data Bus 为 4it，SCCR ≥ 4 SYS_DW=1，LCD 的 Data Bus 为 8it，SCCR ≥ 2	--	R/W

例 题：

1. 如果使用 X'tal + PLL 的方式，系统频率(SCLK) = 8MHz
2. LCD 驱动器 的 Data Bus 宽度(DBW) = 8Bit
3. 使用 320 x 240 Pixel 的 LCD 面板，Column = 320，Row = 240
4. LCD 面板的 Frame Rate 为 70Hz

则 SCCR = (8MHz x 8) / (320 x 240 x 70) = 11.9

所以建议设定 SCCR = 12 = 0Ch

如果设定数值太大，会造成 LCD 屏幕闪动，有时会伴随水波纹产生，除降低数值外也可以提高系统频率来解决→ 调整 REG[01h] 设定数值。为达到良好 LCD 显示品质，使用者必须根据 Panel、Driver 之特性与 VLCD 电压、系统频率等等进行调整。

9-14 中断(Interrupt)与忙碌(Busy)设定

RA8803/8822 提供一中断信号线(INT)用来表示有四种中断讯息可能发生：

1. 假如光标 Segment 地址缓存器(CPXR)与 Segment 中断地址缓存器(INTX)值相同，发生中断。如不使用此功能请将此缓存器设成 FFh，避免发生无用的中断浪费 MCU 的处理时间。
2. 假如光标 Common 地址缓存器(CPYR)与 Common 中断地址缓存器(INTY)值相同，发生中断。如不使用此功能请将此缓存器设成 FFh，避免发生无用的中断浪费 MCU 的处理时间。
3. 触控屏幕侦测到被 Touch，发生中断。
4. KeyScan 侦测到被按下，发生中断。.

这四种中断都可以单独被致能或禁能，而中断的设定与中断讯息可有由缓存器[A0h] INTR 来控制与读取。

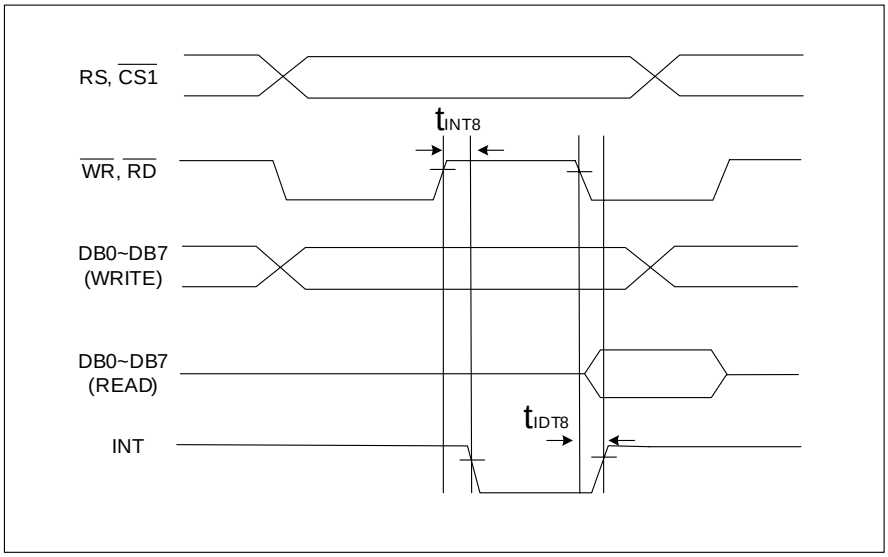


图 9-17C：中断脚位 INT 的时序

如图 9-17C 当上述四种中断事件发生时，约经过 445ns 后(T_{INT8})，RA8803/8822 会产生 INT 信号通知 MPU，MPU 可藉由此项信号执行相对应的中断子程序，当中断发生时 INT 脚位将会维持您所设定的触发准位，使用者若要结束中断必须去 Read 缓存器[A0h]，约经过 60ns 后(T_{IDT8})，中断请求就会结束，详见下表。

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
BUSY	t_{INT8}	Interrupt Setup Time	445	--	ns	System Clock: 8MHz Voltage: 3.3V
	t_{IDT8}	Interrupt Disable Ttime	60	--	ns	

此外 RA8803/8822 提供一忙碌(Busy)信号线，用来表示 RA8803/8822 内部 DDRAM 与 ROM 的存取状态是否因 Busy 而暂时无法接收 MPU 来的 Command。此 BUSY Pin 通常与 MPU 的 I/O 端连接，MPU 在对 RA8803/8822 做存取前可以先判断 RA8803/8822 是否可以接受存取动作(Available)。

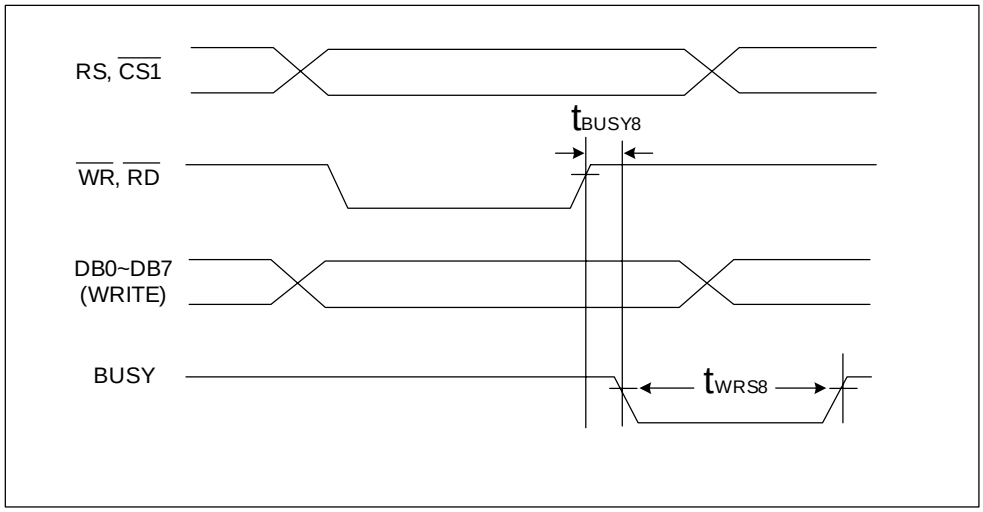


图 9-17D：忙碌脚位 BUSY 的时序

上图 9-17D 为 8bit 8080 MPU 对 RA8803/8822 进行数据写入完后所产生的 BUSY 信号，当数据写入完毕 t_{BUSY8} 约经过 30ns 后产生 BUSY 信号，而 BUSY 的信号脉波宽度 t_{WRS8} 会随着写入的资料量而有所不同，详见下表。

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
BUSY	t_{BUSY8}	The Period of Busy	30	--	ns	System Clock: 8MHz Voltage: 3.3V T=1/System Clock
	t_{WRS8}	Busy Setup Time for Write(半角字)	--	20T	ns	
		Busy Setup Time for Write(全角字)	--	40T	ns	

以下是相关的缓存器说明：

REG [01h] Misc. Register (MISC)

Bit	Description	Default	Access
4	**设定输出脚 -- 中断讯号(INT)与忙碌讯号 BUSY 的触发准位** 1：设定高电位触发动作 0：设定低电位触发动作	1h	R/W

REG [A0h] Interrupt Setup & Status Register (INTR)

Bit	Description	Default	Access
7	**Key Scan 中断旗标** 1：Key Scan 有侦测到按键输入 0：Key Scan 无侦测到按键输入	0h	R
6	**触控屏幕侦测** 1：触控屏幕有侦测到触摸(Touch) 0：触控屏幕未侦测到触摸	0h	R
5	**光标 Column 状态** 1：光标的 Column 等于缓存器[B0h]INTX 0：光标的 Column 不等于缓存器[B0h]INTX	0h	R
4	**光标 Row 状态** 1：光标的 Row 等于缓存器[B1h]INTY 0：光标的 Row 不等于缓存器[B1h]INTY	0h	R
3	**Key Scan 中断屏蔽控制** 1：致能 Key Scan 中断 0：禁能 Key Scan 中断	0h	R/W
2	**触控屏幕中断屏蔽** 1：如果触控屏幕被侦测到，则产生中断输出 0：如果触控屏幕被侦测到，则不产生中断输出	0h	R/W
1	**设定缓存器[B0h]INTX 是否发生中断** 1：致能 INTX 中断 0：禁能 INTX 中断	0h	R/W
0	**设定缓存器[B1h]INTY 是否发生中断** 1：致能 INTY 中断 0：禁能 INTY 中断	0h	R/W

注：Bit3~Bit0 的任一 Bit 被设为 ”1” 将使得中断讯号功能 (INT) 和忙碌讯号功能(BUSY) 被致能 (Enable)，而 INT、BUSY 的触发准位由缓存器 [01h] 的 Bit4 决定。

REG [B0h] Interrupt Column Setup Register (INTX)

Bit	Description	Default	Access
5-0	**设定行(Column)中断地址** 假如光标位置 X 缓存器(CPXR)=INTX，发生中断。	27h	R/W

REG [B1h] Interrupt Row Setup Register (INTY)

Bit	Description	Default	Access
7-0	设定列**(Row)**中断地址 假如光标位置 Y 缓存器(CPYR)=INTY，发生中断。	EFh	R/W

9-15 省电模式

RA8803/8822 的电源工作模式分两种：正常模式(Normal Mode)，关闭模式(Off Mode)，请参考下面缓存器及例题。

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
7-6	**电源模式(Power Mode)** 1 1：正常模式(Normal Mode) RA8803/8822 的所有功能都可以使用(Available)。 0 0：关闭模式(Off Mode) 除了唤醒(Wake-Up)电路工作外，其它功能都被禁止。当 Wake-Up 电路被触发，RA8803/8822 将进入正常模式。	--	3h	R/W

例 题：

```
Normal_Mode:
    CALL    Read_R00
    MOV     A,REG00_READ
    OR      A,#00000011b
    CALL    Write_R00
    RTS

Sleep_Mode:
    CALL    Read_R00
    MOV     A,REG00_READ
    AND     A,#11111100b
    CALL    Write_R00
    RTS
```

9-16 如何读取 Font ROM 字型

RA8803/8822 允许 MPU 读取 Font ROM 的 Data，只要将缓存器[02h]的 Bit3 设为 1，然后写入两个 Byte 的中文码，之后连续读取的 32Byte Data 就是该中文码相对映的 Font Data，如下图 9-18A 的流程图。

REG [02h] Advance Power Setup Register (APSR)

Bit	Description	Default	Access
3	**字型 ROM 的直接读取** 1：致能 0：禁能	0h	R/W

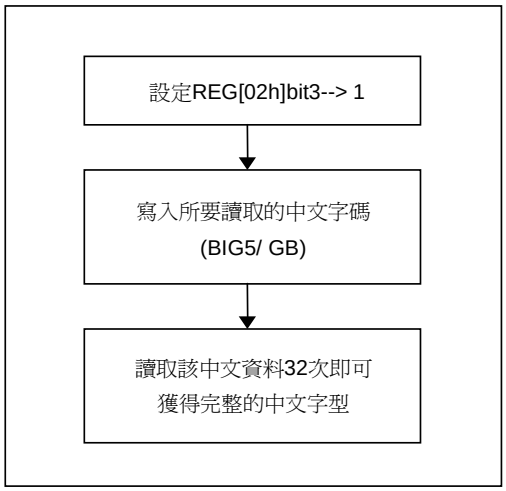
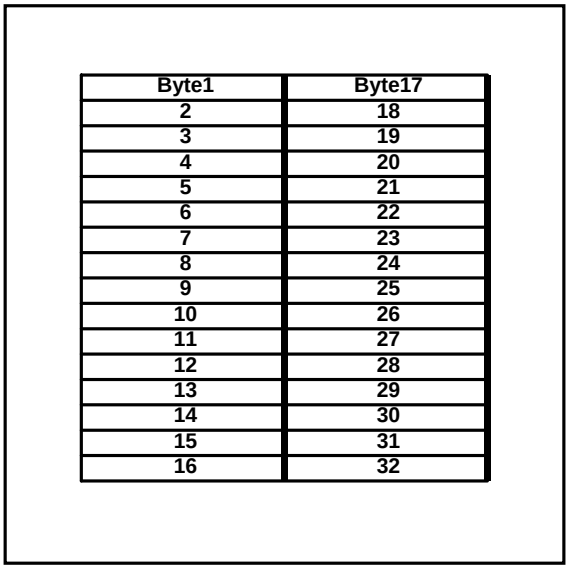


图 9-18A：读取 Font ROM 字型流程

之前提过 RA8803/8822 的全型字型为 16x16 的 Bitmap 所组成，每个全型字型占用 Font ROM 32Byte，在 MPU 读取 Font ROM 的 Data 时其顺序如下图 9-18B 所示。



Byte1	Byte17
2	18
3	19
4	20
5	21
6	22
7	23
8	24
9	25
10	26
11	27
12	28
13	29
14	30
15	31
16	32

图 9-18B：读取 Font ROM 字型 Data 的顺序

9-17 字号放大设定

RA8803/8822 内建有 512KByte 的中文显示字型 ROM(Font ROM)，全角 16x16 中文与 8x16 的 ASCII 半型字型。除了内建的 8x16 和 16x16 的字号外，还提供字型放大的功能，可利用 REG[F1h]bit7~4 的设定，将显示字号放大到 32x32 或 48x48，64x64。下图 9-19 是表示 16x16 的字型放大到 32x32。

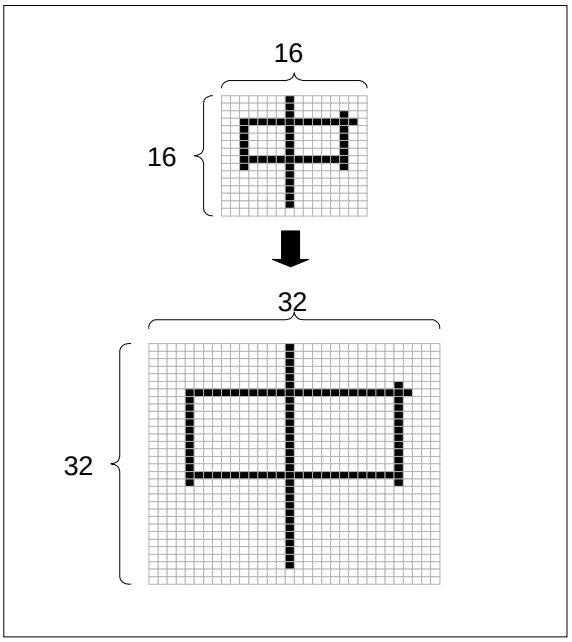


图 9-19 : 字号放大

REG [F1h] Font Size Control Register (FVHT)

Bit	Description	Default	Access
7-6	**设定字型水平的大小** 0 0：一倍 0 1：二倍 1 0：三倍 1 1：四倍	0h	R/W
5-4	**设定字型垂直的大小** 0 0：一倍 0 1：二倍 1 0：三倍 1 1：四倍	0h	R/W

例 题：

```
Font_Tab:     db   "中",0Dh
Test_Font_Size:
   MOV      A, #5Fh          ; Set Font Size = 32 x 32
   CALL     Font_Size
   Printf   Font_Tab         ; 显示 32x32 的 “中”
   MOV      A, #AFh          ; Set Font Size = 48 x 48
   CALL     Font_Size
   Printf   Font_Tab         ; 显示 48x48 的 “中”
      :
      :
      :
Font_Size:
   CALL     Write_RF1        ; Write A to Register $F1
   RTS
```

9-18 图层显示功能设定

RA8803/8822 提供了双图层的功能，可经由缓存器 REG[12h]来做设定，并提供 4 种(OR，NOR，XOR 和 AND)图层显示模式，供使用者设定选用。实际的显示效果，请参考图 9-20。

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
6-4	**设定选择 Display Data RAM 的图层显示模式** 0 0 1：只有显示 Page1 的图层 (单一上层显示模式) 0 1 0：只有显示 Page2 的图层 (单一下层显示模式) 0 1 1：同时显示 Page1 和 Page2 的图层 (双层模式) 0 0 0：灰阶显示(Gray Mode)，此模式下每一个点的灰度决定于 DDRAM Page1 与 Page2 相对映的值。 Page1 Page2 灰度 0 0 Level1 1 0 Level2 0 1 Level3 1 1 Level4 1 1 0：扩展模式(1)，同时显示 Page1 和 Page2 的图层，让 RA8803 可用于 640x240，RA8822 可用于 480x160 的 Panel。 1 1 1：扩展模式(2)，同时显示 Page1 和 Page2 的图层，让 RA8803 可用于 320x480，RA8822 可用于 240x320 的 Panel。	1h	R/W
3-2	**在双层模式下图层逻辑关系** 0 0：Page1 RAM “OR” Page2 RAM 0 1：Page1 RAM “XOR” Page2 RAM 1 0：Page1 RAM “NOR” Page2 RAM 1 1：Page1 RAM “AND” Page2 RAM 请参考 ”图 9-20” 的图形说明	0h	R/W
1-0	**设定 Read/ Write 要在哪一个图层运行** 0 0：存取 Page0 (512B SRAM)的 Display Data RAM 0 1：存取 Page1 (9.6KB SRAM)的 Display Data RAM 1 0：存取 Page2 (9.6KB SRAM)的 Display Data RAM 1 1：同时存取 Page1 和 Page2 的 Display Data RAM	1h	R/W

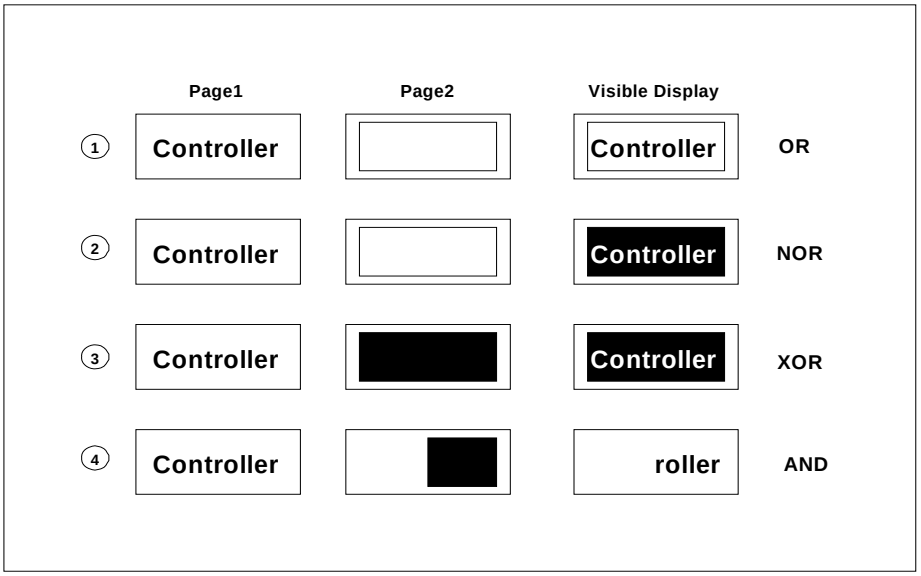


图 9-20：图层显示效果

例 题：

```
        CALL    Display_Off         ; 先将显示器 OFF
        CALL    Graphic_Mode        ; 进入绘图模式
        MOV     A,#91h              ; 设定数据写入 Page#1
        CALL    Write_R12
        CALL    Show_Page1          ; 填入 Page#1 的图

        CALL    A,#a2h              ; 设定数据写入 Page#2
        CALL    Write_R12
        CALL    Show_Page2          ; 填入 Page#2 的图

        MOV     A,#bdh              ; 设定显示 Page#1 AND Page#2
        CALL    Write_R12
        CALL    Display_On          ; 将显示器 ON
          :
          :
          :
Display_On:
        CALL    Read_R00
        MOV     A,REG00_READ
        OR      A,#00000100b
        CALL    Write_R00
        RTS

 Display_Off:
        CALL    Read_R00
        MOV     A,REG00_READ
        AND     A,#11111011b
        CALL    Write_R00
        RTS
```

9-19 Key Scan 应用

RA8803/8822 内建有 4x8/8x8 的 Key Scan 电路，可用来作为 Keyboard 的功能，帮助系统发展者可轻易整合开发含有 Keyboard 的周边电路，其相关设定缓存器为 KSCR，KSDR，KSER。

图 9-21A 为 Key Scan 的应用电路图，事实上只要将 Key PAD 直接接到 RA8803/8822 就可以了。设定完缓存器 KSCR 后，当您按下按键时，透过程序的 programming 会将被按到的那一个键的 Key Code 分别存到缓存器 KSDR 及 KSER，之后只要将 KSDR 及 KSER 这两个缓存器里面的按键数据读出作比对，就可判断哪一个按键被按下。

REG [A1h] Key Scan Controller Register (KSCR)

Bit	Description	Default	Access
7	**Key Scan 的致能控制位** 1：致能 0：禁能	0h	R/W
6	**Key San 的数组选择** 1：Key Scan 为 8x8 数组 0：Key Scan 为 4x8 数组	0h	R/W
5-4	**KeyScan 的扫描周期** 0 0：2 倍的 Key Scan 扫描周期 0 1：4 倍的 Key Scan 扫描周期 1 0：8 倍的 Key Scan 扫描周期 1 1：16 倍的 Key Scan 扫描周期	0h	R/W
2-0	**Key Scan 的扫描周期选择** 0 0 0：2 倍(LP peak to peak period) 0 0 1：4 倍(LP peak to peak period) 0 1 0：8 倍(LP peak to peak period) 0 1 1：16 倍(LP peak to peak period) 1 0 0：32 倍(LP peak to peak period) 1 0 1：64 倍(LP peak to peak period) 1 1 0：128 倍(LP peak to peak period) 1 1 1：256 倍(LP peak to peak period)	0h	R/W

缓存器 [A1h]中的 bit[5-4]为 KeyScan 的扫描周期,例如：若是您将 bit[5-4]的值设为 00 → 2 倍的 Key Scan 扫描周期，也就是当您按下按键时，它会完整的扫描键盘两次，若这两次扫描出来的值都是一样才会输出那一个按键被按下，确保按键输出的准确性，那依此类推，当 bit[5-4]为 4、8、16，分别就代表扫描 4、8、16 次后判断哪一个按键被按下后，再将结果值输出，相对的比对的越多次，误判的机率就

较低!!

缓存器 [A1h] 中的 bit[2-0] 为 Key Scan 的扫描周期选择，目的是要设定您需要花多少时间去完整的扫描所有按键，例如：当 bit[2-0] 被设为 00 → 2 倍(LP peak to peak period)，Frame Rate 为 80 ，panel 大小 320*240，所以您完整扫描键盘所需的时间为：

2*(1/80)*(1/240) = 约 33ms

REG [A2h] Key Scan Data Register (KSDR)

Bit	Description	Default	Access
7-0	**Key Scan KC[7~0] 的输出值**	0h	R

REG [A3h] Key Scan Data Expand Register (KSER)

Bit	Description	Default	Access
7-0	**Key Scan KR[7~0] 的输入值**	0h	R

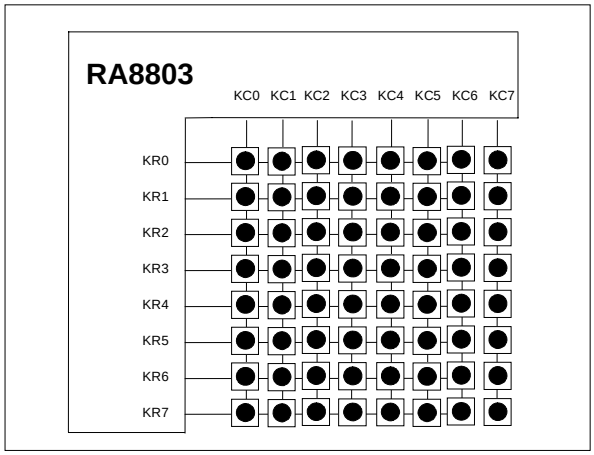


图 9-21A：Key Scan 示意图

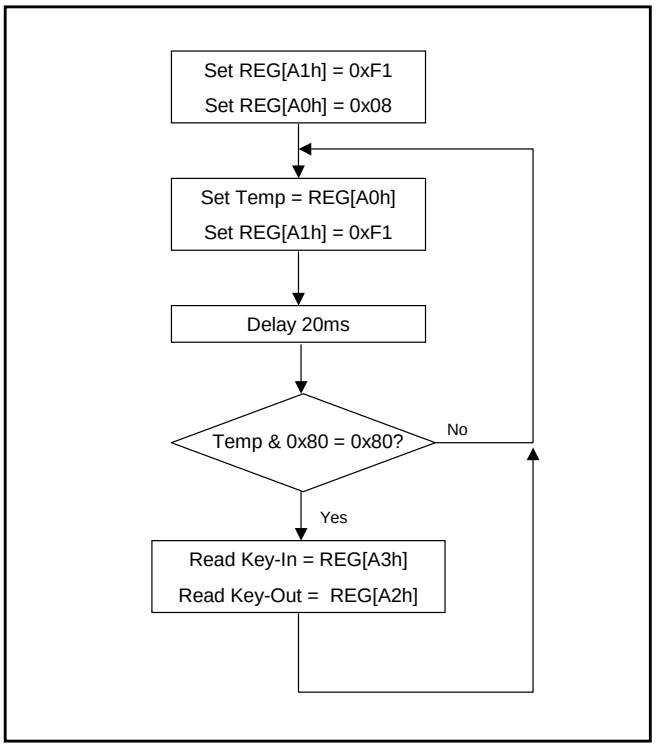


图 9-21B：Key Scan 流程图

注 1：Delay 的反应时间与 REG[A1h] bit[6]、bit[5:4]、bit[2:0] 的设定有关，当 bit(6) 设为 0，bit[5:4]设为 11 ,bit[2:0]设为 001，Frame Rate 为 80，Panel 大小 320*240，反应时间 8*16*4*(1/80)*(1/240) = 大约 26ms，所以当 Key 被按下去大约 Delay 26ms 就可以去读 Register 了。

注 2：在 Key Scan 的应用上，如果客户要用到组合键、长按键等功能，建议自行用单片机的 I/O 口或其它方法实现。

9-20 屏幕水平卷动及垂直卷动设定

在屏幕所显示的画面可以作水平卷动，须由缓存器[03h]来做设定。该项功能可达到左右的水平卷动，每次移动的刻度为 1 个 Byte。另外，还可透过缓存器[71h, 72h]来设定屏幕的区块水平卷动，如图 9-22B。

REG [03h] Advance Display Setup Register (ADSR)

Bit	Description	Default	Access
2	**设定 Common 的自动卷动** 1： 致能 0： 禁能	0h	R/W
1	**设定 Segment 的自动平移** 1： 致能 0： 禁能	0h	R/W
0	**设定选择 Common 的卷动或是 Segment 的平移模式** 1：Segment 的平移 0：Common 的卷动	0h	R/W

REG [71h] Shift action range, Begin Common Register (BGCM)

Bit	Description	Default	Access
7-0	**在水平移动模式下，设定区块移动的启始 Common 位置**	0h	R/W

REG [72h] Shift action range END Common Register (EDCM)

Bit	Description	Default	Access
7-0	**在水平移动模式下，设定区块移动的结束 Common 位置**	EFh	R/W

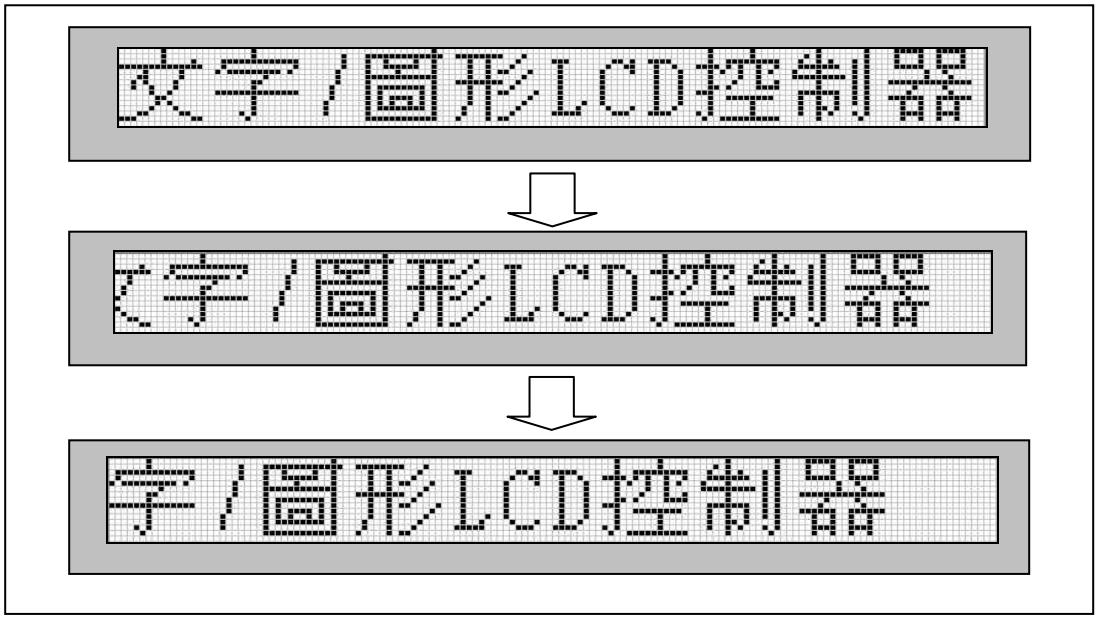


图 9-22A：水平卷动的效果

```
LCD_CmdWrite(0x80,0x05);        //设定水平或垂直卷动速度
LCD_CmdWrite(0x71,0x00);        //设定 REG[71]区块 Y1 坐标
LCD_CmdWrite(0x72,0x00);        //设定 REG[72]区块 Y2 坐标
LCD_CmdWrite(0x03,0x83);        //设定 REG[03]:bit[1,0]="11"
                                //此时屏幕将以 Y1/Y2 设定的区块做水平卷动
```

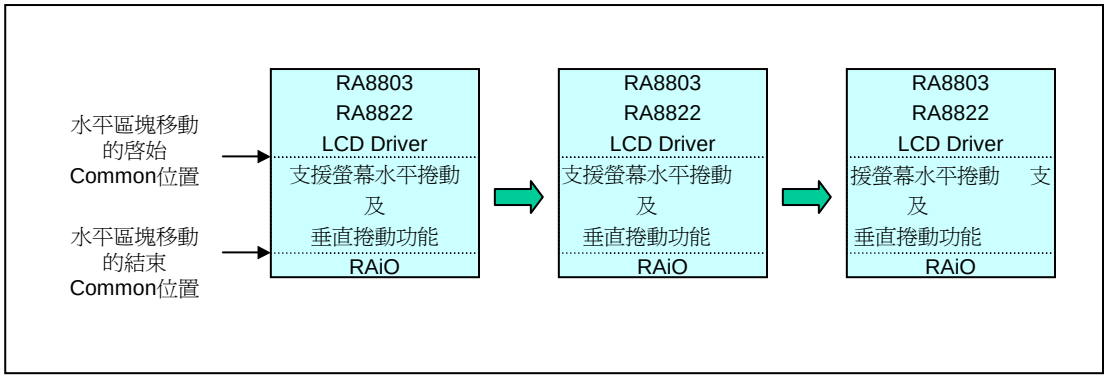


图 9-22B：水平卷动的效果

图 9-22B，透过缓存器[71h, 72h]来设定水平卷动范围，屏幕的画面是由 Common 的哪一段区域进行卷动。

在屏幕所显示的画面可以作垂直卷动，须由缓存器[03h]来做设定。该项功能可达到上下的垂直卷动，每次移动的刻度为 1 个像素(Pixel)。如图 9-23 所示，可作卷动的效果。

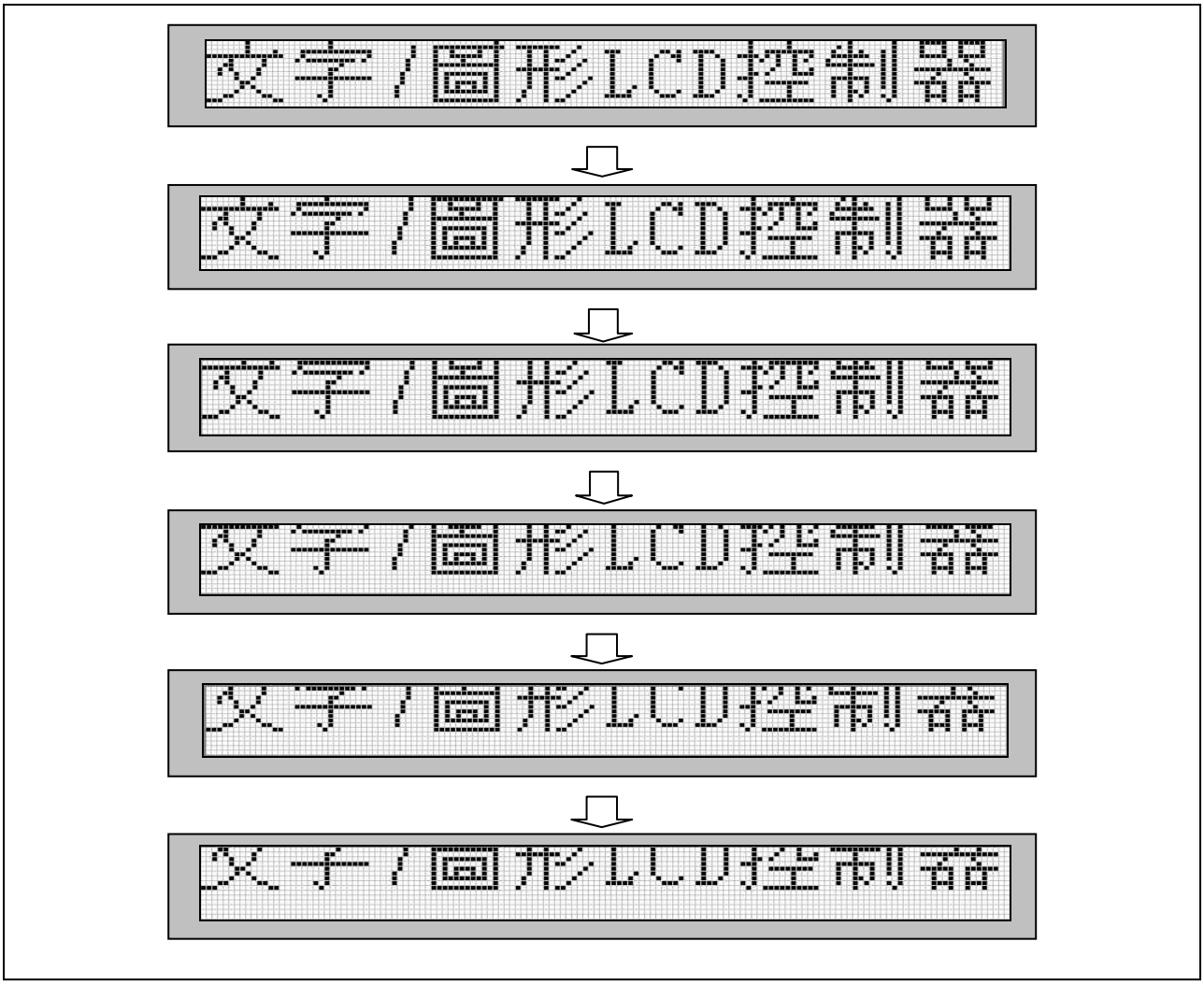


图 9-23：垂直卷动的效果

```
LCD_CmdWrite(0x80,0x05);          //设定水平或垂直卷动速度
LCD_CmdWrite(0x03,0x86);          //设定 REG[03]:bit[2,1]=”11”， 此时整个屏幕将
                                  做垂直卷动
```

9-21 ASCII 区块选择设定

RA8803/8822 内建四个 ASCII 区块，包含许多文字、特殊符号或图形等，可供使用者直接取用，此功能可以由缓存器[F0h]的 bit[1..0]来设定。如果使用者需要特殊符号或图形，亦可经由调整 ROM Code 的方式来建立。下面我们将介绍这四个区块的 Pattern(如图 9-24~9-27)、选择方式及使用范例。

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
1-0	**4 种 ASCII 区块选择** 0 0：ASCII 选择区块 0，Latin_1 0 1：ASCII 选择区块 1，Latin_2 1 0：ASCII 选择区块 2，Latin_3 1 1：ASCII 选择区块 3， Latin_4	--	2h	R/W

9-21-1 ASCII 字形区块 0

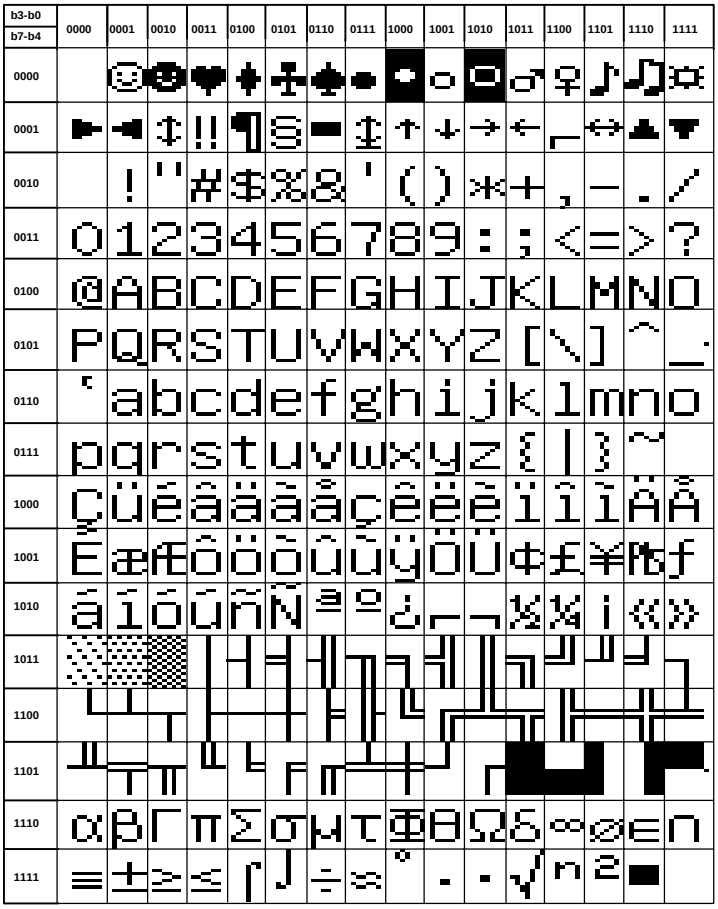


图 9-24：内建 ASCII 区块 Bank0

例 题：

```
MOV     A,#xxxxxx00b        ; 设定 选择 ASCII Code 表为 Block 0
CALL    Write_RF0
MOV     A,#00000100b        ; 选择 Block0 里的“@”
MOV     DATA_ADDR,A         ; 光标所指的位置就会显示“@”
MOV     A,#10010011b        ; 选择 Block0 里的“9”
MOV     DATA_ADDR,A         ;“@”之后，光标所指的位置就会显示“9”
```

9-21-2 ASCII 字形区块 1

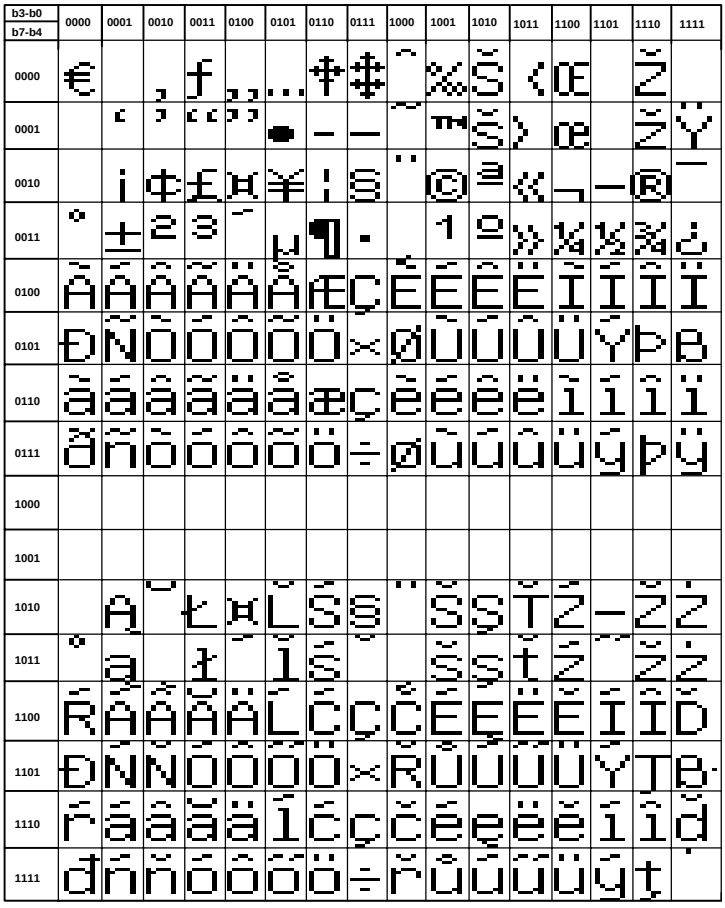


图 9-25：内建 ASCII 区块 Bank1

9-21-3 ASCII 字形区块 2

区块 2 的选择方式与上面相同，只要设定缓存器[F0h]的 bit[1..0]，再将选择的 Pattern 写入光标所在的位置既可。

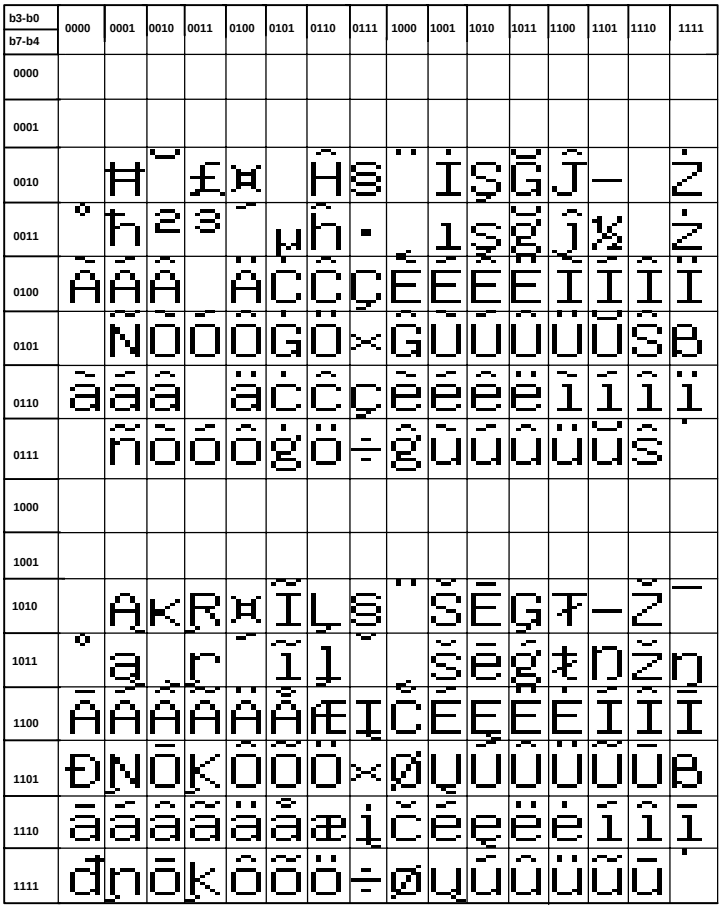


图 9-26：内建 ASCII 区块 Bank2

9-21-4 ASCII 字形区块 3

区块 3 的选择方式与上面相同，也只要设定缓存器[F0h]的 bit[1..0]，再将选择的 Pattern 写入光标所在的位置既可。在区块 3 有许多空的 Pattern，如果使用者需要少量的特殊符号或图形，可经由调整 ROM Code 的方式填入 Pattern 在此区块。

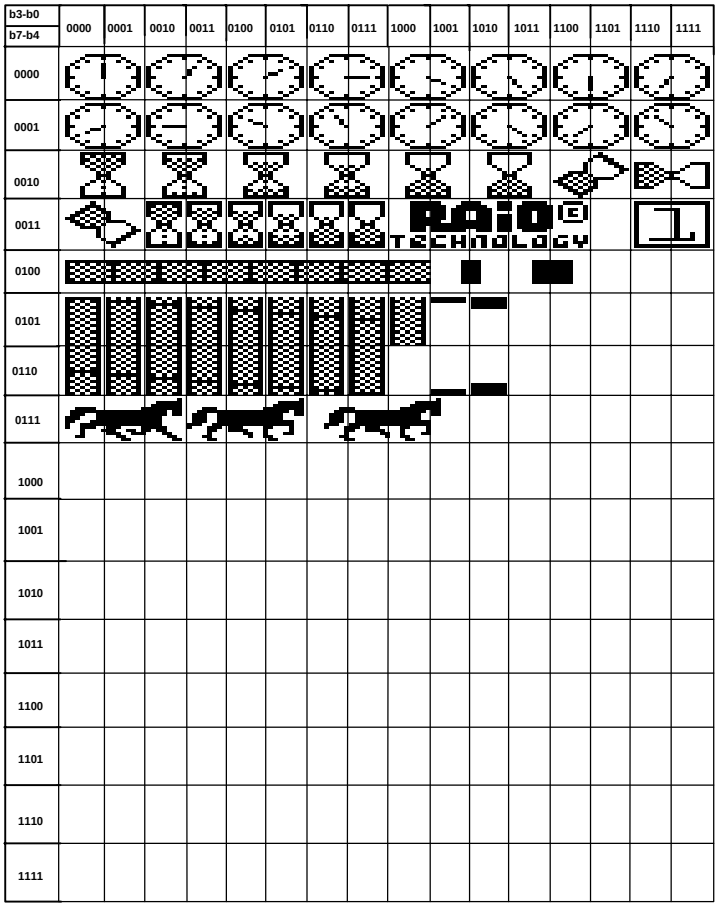


图 9-27：内建 ASCII 区块 Bank3

9-22 自行造字

RA8803/8822 内建 512Byte SRAM 可支持自行造字功能，最大字数为 16 个全角中文字(16x16)。若用到特殊字，是字库内没有的字型，可利用该项功能，增加内建字库的内容，来提升 MPU 的存取效率。下面是造字会用到的缓存器及范例:

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
1-0	**设定 Read/ Write 要在哪一个图层运行** 0 0：存取 Page0 (512B SRAM)的 Display Data RAM 0 1：存取 Page1 (9.6KB SRAM)的 Display Data RAM 1 0：存取 Page2 (9.6KB SRAM)的 Display Data RAM 1 1：同时存取 Page1 和 Page2 的 Display Data RAM	1h	R/W

REG [60h] Cursor Position X Register (CPXR)

Bit	Description	Default	Access
5-0	**设定光标 Segment 地址**	0h	R/W

例 题：

```
Create_Font_Tab0: db    08h,1ch,1ch,ffh,7fh,1ch,3eh,3eh,
                        77h,41h,00h,00h,83h,7fh,3fh,0fh, 0Dh
Create_Font_Tab1: db    20h,10h,1ch,9eh,1eh,1fh,1fh,1fh,
                        1fh,3fh,7eh,feh,fch,f8h,f0h,c0h, 0Dh
Create_Font_Tab2: db    FFh, F0h, 0Dh

Test_Create_Font:
   CALL    Graphic_Mode        ; 设定成绘图模式
   MOV     A,#10h              ; Write to Page0 → 512Byte SRAM
   CALL    Write_R12
   MOV     A,#0h               ; 对中文码 "FFF0" 进行造字
   CALL    Write_R60           ; 设定光标 Segment 地址
   Printf  Create_Font_Tab0    ; 前 16Byte
   MOV     A,#01h
   CALL    Write_R60           ; 设定光标 Segment 地址(每 16Byte 要加 1)
   Printf  Create_Font_Tab1    ; 后 16Byte

   CALL    Text_Mode           ; 设定成文字模式
   MOV     A,#91h              ; Page1
   CALL    Write_R12
   Printf  Create_Font_Tab2    ; 显示码为 "FFF0"的字样 → 图 9-28
```

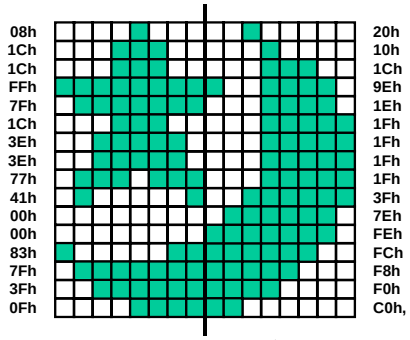


图 9-28：造字

每个全角 16X16 中文字占 32Byte，因此内建 512Byte SRAM 可造 16 个字，中文码内订为”FFF0~FFFF” 。上例为自建中文码为”FFF0”的字样，若是”FFF1”则写入 Data 到 Page0 之前的前 16Byte 要先将缓存器[60h]设成”02h”，写入 Data 到 Page0 之前的后 16Byte 要将缓存器[60h]设成”03h”，依此类推。

注: 在可造字时须要先将 Line Distance 设为 0，也就是缓存器[11h]的 Bit[3:0]设成 0，造完字后就无此限定，请参考 9-11 节。

9-23 灰阶显示

RA8803/8822 可利用分时显示的原理达到灰阶显示的效果，灰阶模式需要同时使用 Page1 和 Page2 的图层，在此模式下 LCD 每一个点的灰阶效果决定于 Display RAM Page1 与 Page2 的值。对 LCD 的同一点来说，[Page1, Page2] 可以为[0,0]、[1,0]、[0,1]、或 [1,1]，如果它们的显示不同将会产生不同的灰度效果，由于是利用分时显示的原理，为了达到良好的显示品质及避免闪烁必须将 Frame Rate 或系统频率提高。下面是灰阶显示会用到的缓存器及范例:

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
6-4	**设定选择 Display Data RAM 的图层显示模式** 0 0 1：只有显示 Page1 的图层 (单一上层显示模式) 0 1 0：只有显示 Page2 的图层 (单一下层显示模式) 0 1 1：同时显示 Page1 和 Page2 的图层 (双层模式) 0 0 0：灰阶模式(同时显示 Page1 和 Page2 的图层)，在此模式下 LCD 每一个点的灰阶效果决定于 Display RAM Page1 与 Page2 的值。 Page1 Page2 Gray --- 0 0 Level1 1 0 Level2 0 1 Level3 1 1 Level4 1 1 0：扩展模式(1)，同时显示 Page1 和 Page2 的图层，让 RA8803 可用于 640x240，RA8822 可用于 480x160 的 Panel。 1 1 1：扩展模式(2)，同时显示 Page1 和 Page2 的图层，让 RA8803 可用于 320x480，RA8822 可用于 240x320 的 Panel。	1h	R/W

REG [E0h] Pattern Data Register (PNTR)

Bit	Description	Default	Access
7-0	**(1) Data Written to DDRAM** **(2) Display Times of Gray Mode** 在灰阶模式下(Register MAMR bit[6..4] = 000) ，此缓存器用来控制显示时间，如果 Frame Rate 固定，此缓存器 “1” 和 “0” 的数目代表显示 1 和 0 的比率。	0h	R/W

PNTR = 55h, AAh, 0Fh, F0h, CCh, 33h 或 99h 皆表示缓存器 Data 中 “1” 和 “0” 的数目一样，那么灰阶 Level2 与 Level3 的显示效果是一样的，如果设成这些值只能有 3 阶的显示效果，必须让 ”1” 的数目多于 “0” 的数目才能有 4 灰阶显示的效果。

图 9-29 是在屏幕上秀出四灰阶的基本概念，如果 Display RAM 的 Page1 上半部全部填”00”，下半部全部填”FF”，且 Page2 的左半部全部填”00”，右半部全部填”FF”，那么启动灰阶功能后可以在屏幕上秀出四个不同灰阶的方块。

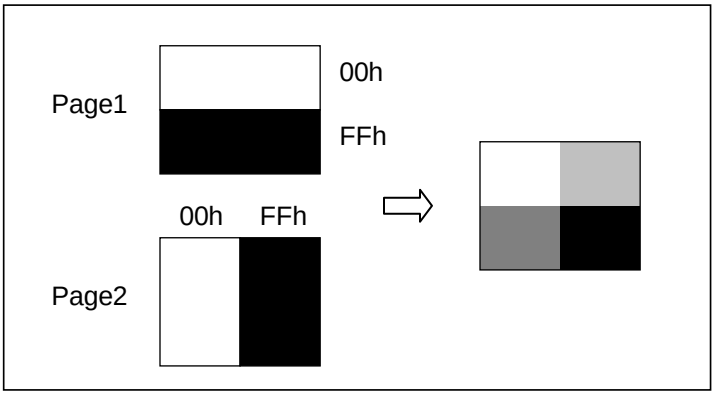


图 9-29：灰阶

例 题：

```
CALL     Graphic_Mode
LCD_CmdWrite(0x12, 0X91);          //Write Page#1
CALL     Show_Gray2
LCD_CmdWrite(0x12,0x92);           //Write Page#2
CALL     Show_Gray1
LCD_CmdWrite(0x01,0XF2);           //提高 System Clock
LCD_CmdWrite(0xD0,0x01);           // 提高亮度

LCD_CmdWrite(0xE0,0x3F);           //灰阶对比
LCD_CmdWrite(0x90,0x04);           //提高 Frame Rate
LCD_CmdWrite(0x12,0x00);           //显示灰阶
CALL     Display_Off
CALL     Display_On
CALL     Delay1s
```

9-24 扩展模式显示

RA8803/8822 提供了扩展模式，让 DDRAM 的双图层(Page1 & Page2)同时显示在更大的 LCD 面板上，可经由缓存器 MAMR 的 Bit[6:4]来做设定，当 MAMR 的 Bit[6:4]=110b 时，RA8803 可显示到 640x240 的 LCD 面板上，RA8822 可显示到 480x160 的 LCD 面板，Screen 的左半部会显示 DDRAM Page1 的内容，Screen 的右半部会显示 DDRAM Page2 的内容，实际的显示效果，如图 9-30。缓存器 MAMR 的 Bit[6:4] 说明请参考上一节 9-23 或规格书的说明。

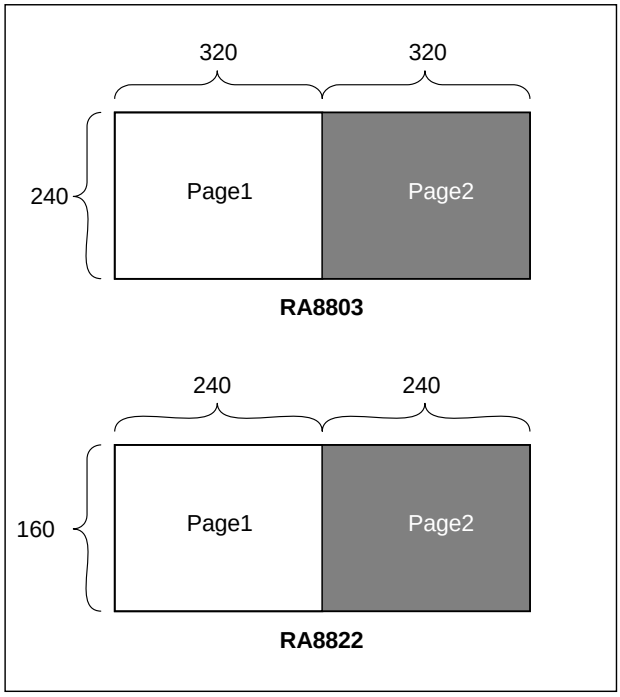


图 9-30：扩展模式(1) 缓存器 MAMR bit[6:4] = 110h

当 MAMR 的 Bit[6:4]=111b 时，RA8803 可显示到 320x480 的 LCD 面板上，RA8822 可显示到 240x320 的 LCD 面板，Screen 的上半部会显示 DDRAM Page1 的内容，Screen 的下半部会显示 DDRAM Page2 的内容，实际的显示效果，如图 9-31。

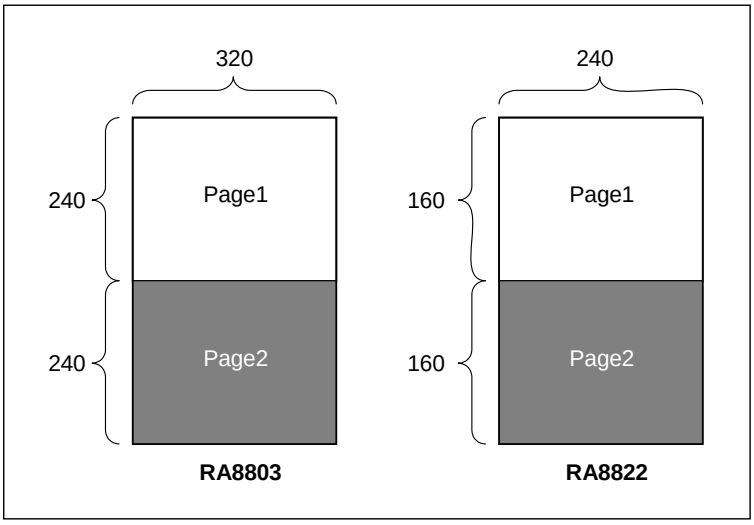


图 9-31：扩展模式(2) 缓存器 MAMR bit[6:4] = 111h

在扩展模式下使用有些地方会受限制，如图 9-30 的 RA8803 设成扩展模式(1)，为显示 600x240 的 Panel，必须将 Data 分别写入 Page1 与 Page2，游标的移位会以 Page 为单位，而非延续由 0 到 599(Common)，如图 9-32 所示。使用者必须将显示画面分为两个，再分别写入 Page1 与 Page2，才能构成一完整的 640x240 画面。

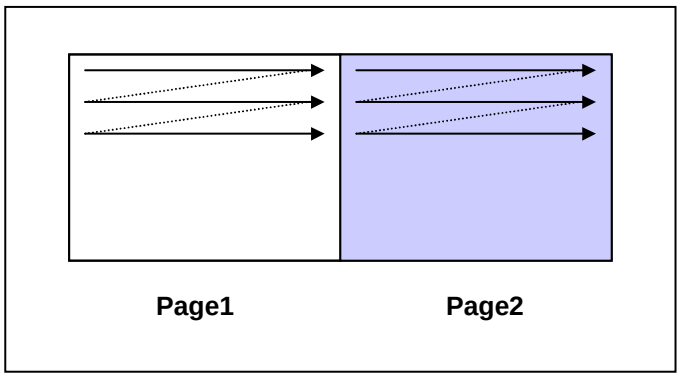


图 9-32：扩展模式(1)的光标移位

以图 9-30 的 RA8803 为例，做水平卷动时会如图 9-33 所示。

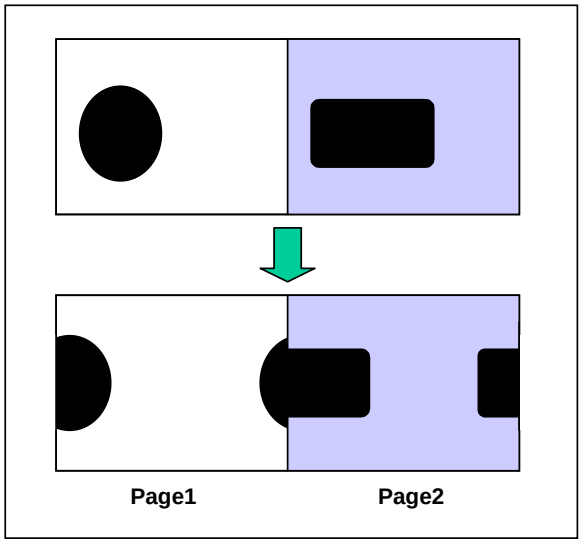


图 9-33：扩展模式(1)的水平卷动

相反的，如图 9-31 的 RA8803 设成扩展模式(2)，为显示 320x480 的 Panel，Data 也必须分别写入 Page1 与 Page2，但游标的移位仍由 0 到 319(Common)，如图 9-34 所示。而做水平卷动时会如图 9-35 所示。

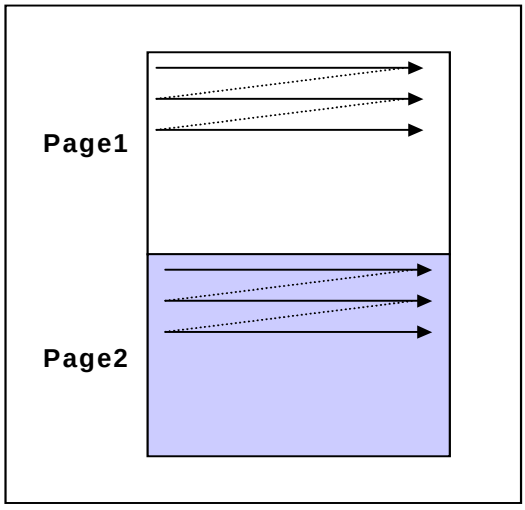


图 9-34：扩展模式(2)的光标移位

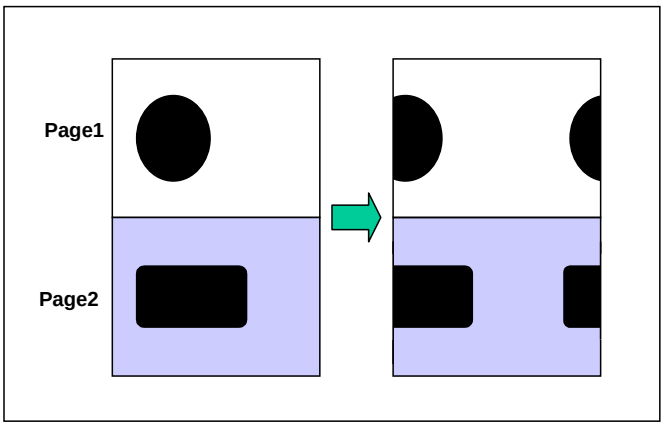


图 9-35：扩展模式(2)的水平卷动

在扩展模式下不提供垂直卷动功能，此外在扩展模式下因为 Page1 与 Page2 都被使用到，因此像灰阶显示、双图层的 4 种(OR，NOR，XOR 和 AND)图层显示模式等将无法使用。

附录 A. 液晶显示驱动器(LCD Driver)的时序图

附录 A 是 RA8803/8822 搭配驱动器(Driver)ST8016/NT7701 在 Segment 和 Common 模式下的时序特性波形图，及参数表。

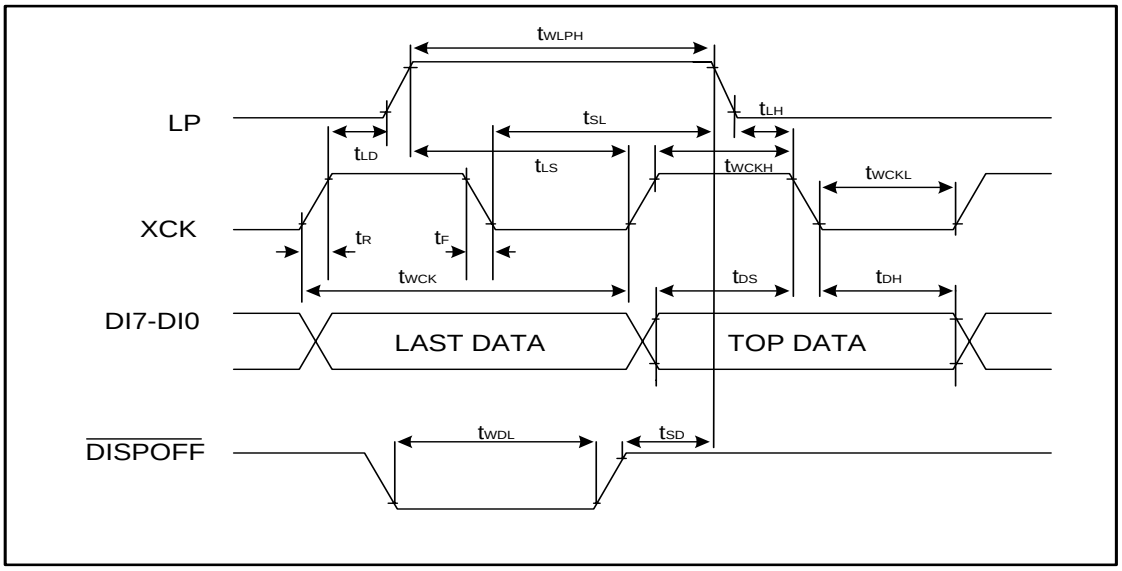


图 A-1：Segment 模式的时序特性波形图

表 A-1：Segment 操作的 Timing 参数

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit	Note
Shift Clock Period	t_{WCK}	t_R, t_F≦11ns	125			ns	1
Shift Clock “H” Pulse Width	t_{WCKH}		51			ns	
Shift Clock “L” Pulse Width	t_{WCKL}		51			ns	
Data Setup Time	t_{DS}		30			ns	
Data Hold Time	t_{DH}		40			ns	
Latch Pulse “H” Pulse Width	t_{WLPH}		51			ns	
Shift Clock Rise to Latch Pulse Rise Time	t_{LD}		0			ns	
Shift Clock Fall to Latch Pulse Fall Time	t_{SL}		21			ns	
Latch Pulse Rise to Shift Clock Rise Time	t_{LS}		51			ns	
Latch Pulse Fall to Shift Clock Fall Time	t_{LH}		51			ns	
Enable Setup Time	t_S		36			ns	
Input Signal Rise Time	t_R				50	ns	2
Input Signal Fall Time	t_F				50	ns	2
DISPOFF Removal Time	t_{SD}		100			ns	

DISPOFF “L” Pulse Width	t_{WDL}		1.2			ns	
Output Delay Time(1)	t_D	CL=15pF			78	ns	
Output Delay Time(2)	t_{PD1}, t_{PD2}	CL=15pF			1.2	us	
Output Delay Time(3)	t_{PD3}	CL=15pF			1.2	us	

Note：

1. Takes the cascade connection into consideration.
2. $(t_{WCK}-t_{WCKH}-t_{WCKL})/2$ is maximum in the case of high-speed operation.

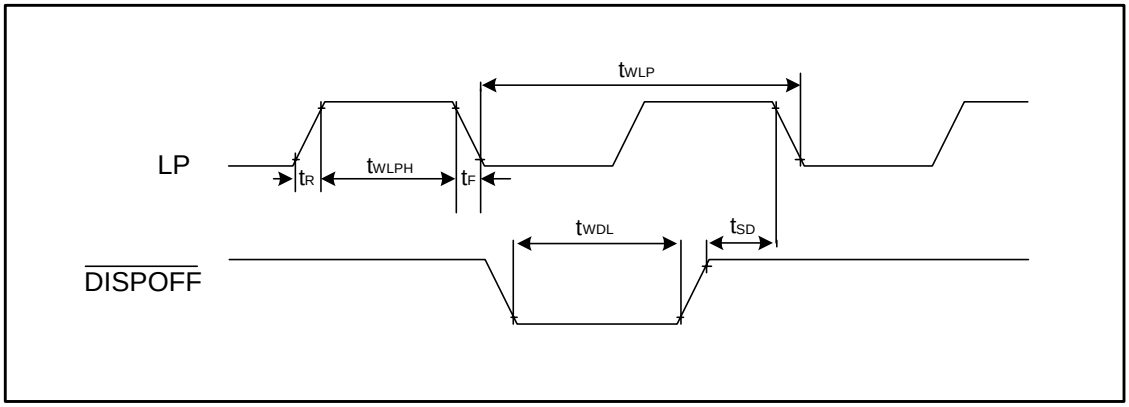


图 A-2：Common 模式的时序特性波形图

表 A-2：Common 操作的 Timing 参数

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit	Note
Shift Clock Period	t_{WLP}	t_R, t_F≦20ns	125			ns	1
Shift Clock “H” Pulse Width	t_{WLPH}	VDD=5	51			ns	
Input Signal Rise Time	t_R				50	ns	2
Input Signal Fall Time	t_F				50	ns	2
DISPOFF Removal Time	t_{SD}		100			ns	
DISPOFF “L” Pulse Width	t_{WDL}		1.2			ns	
Output Delay Time(1)	t_D	CL=10pF			78	ns	
Output Delay Time(2)	t_{PD1}, t_{PD2}	CL=10pF			1.2	us	
Output Delay Time(3)	t_{PD3}	CL=10pF			1.2	us	

附录 B. 应用电路图

B-1 应用电路

附录 B 是 RA8803/8822 应用在 LCM 模块或系统端的控制电路图，也将 MPU 及 Driver 的总线接口脚位拉出，供使用者参考。

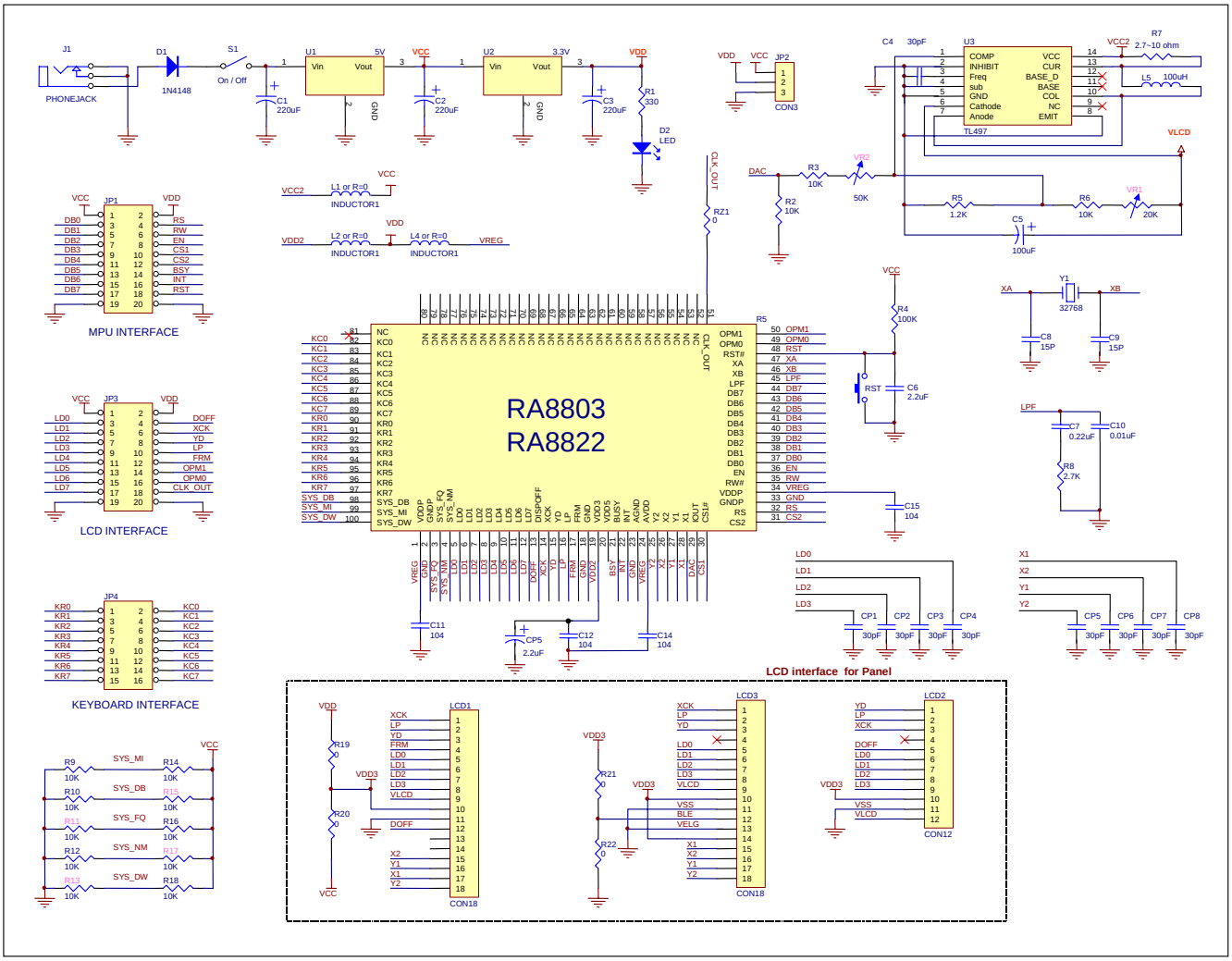


图 B-1：应用电路图

注:

1. Reset Pin – RST#，它可以由 MPU 来控制，或是由一 RC(R4 与 C6)电路来产生，请参考 8-1 节与图 8-2 的说明。请注意，RA8803/8822 没有完成 Reset 的动作是无法接受 MPU 的任何指令，甚至会造成起振不正确或系统设定错误。

2. VLCD 的电压是正的，由 U3 的 TL497 产生，电压范围由 VR1 来控制(≒12V~30V)，输出电流的限制由 R7 决定，且 R7 不能短路，如果接上 LCD Panel 让 VLCD 电压下降，表示 LCD Panel 上的

Droving 负载(Loading)较大，此时可将 R7 调小。

3. 图 B-1 的 VLCD 电压是正的，如果须要的 VLCD 电压是负的，则必须使用别的升压电路。如图 B-2 就是一负压的升压电路。
4. VR2 用来调整 DAC 对 VLCD 的变动范围，虽然 DAC 可用于控制升压电路，进行对比显示(Contrast)设定，但仍须要注意的是升压电路本身的精确度，即使是同一批号的生压 IC，产生的 VLCD 电压准位也会不同，而且不同批的 LCD Panel 对相同 VLCD 电压产生的对比显示效果也不一样，因此如果使用 RA8803/8822 的 DAC 进行对比显示(Contrast)设定，建议仍要加上 VR2 可调电阻做为出厂设定。
5. R19 与 R20 用来选择接到 LCD Panel 的电压是 5V 或 3.3V。
6. R9~R18 用来选择系统设定，请参考 8-1 节的说明。
7. 图 B-1 RA8803/8822 的电源使用的是 3.3V(VREG 与 VDD2 经 L2 与 L4 到 VDD)，而 VDD 是 3.3V。
8. 如果系统时序(System Clock)产生方式为外部 Clock(也就是 R16 焊上，R11 不焊)，外部 Clock 须由 XA 输入，此时 Y1、C8、C9 可不用接。
9. LD[3:0]各并接一电容(CP1~CP4)，可以减少噪声产生。
10. 如果使用触摸式面板，电路图上的 CP5~CP9 电容可以少噪声增加稳定度。

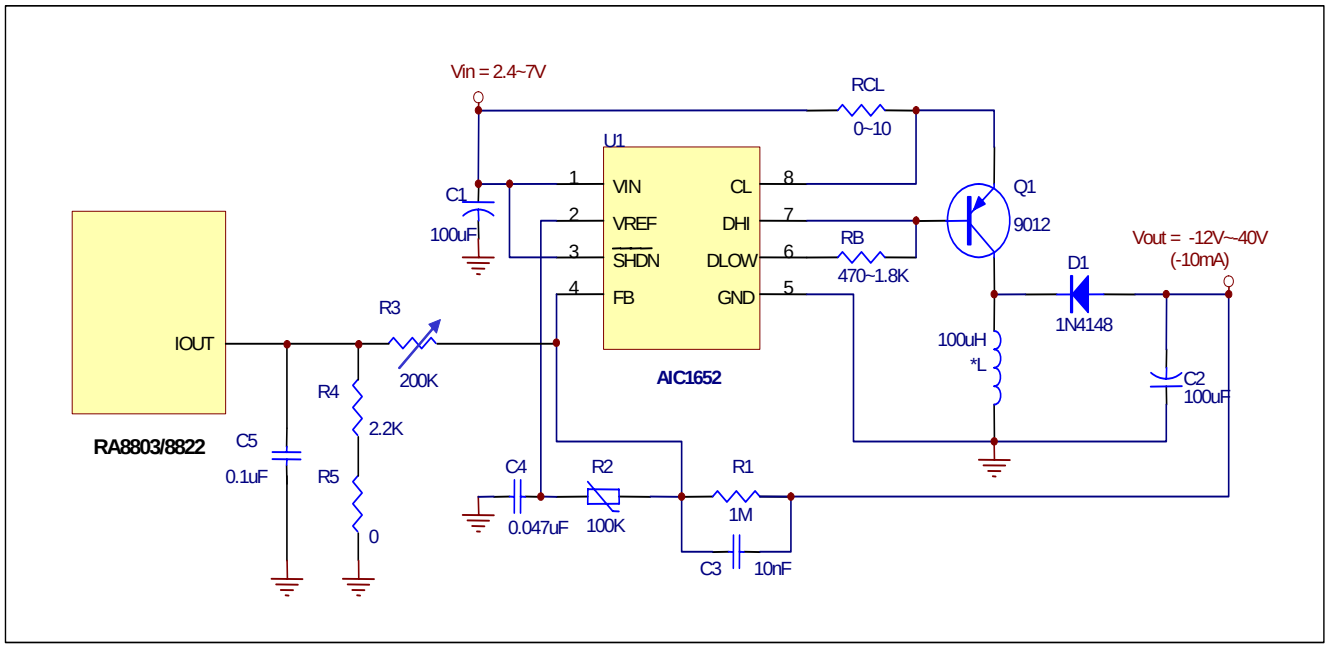


图 B-2：应用电路图-负压的升压电路

B-2 电源(Power)应用电路

B-2-1 电源结构

RA8803/8822 的电源结构如于图 B-3 所示，VDDP 为芯片的 I/O 接脚电源，其对映的地线为 GNDP。RA8803/8822 内部有一 5V 转 3V 的 DC-to-DC 电路，此电路的电源为 VDD5，而 VDD3 为此电路的输出电源，供给芯片内部的组件(Core)与 DAC 使用，也可以由外部再接给 AVDD 使用，DC-to-DC 电路其对映的地线为 GND。RA8803/8822 内部还有一 ADC 电路，用于触摸式面板的控制器，AVDD 为此 ADC 的电源，其对映的地线为 AGND。

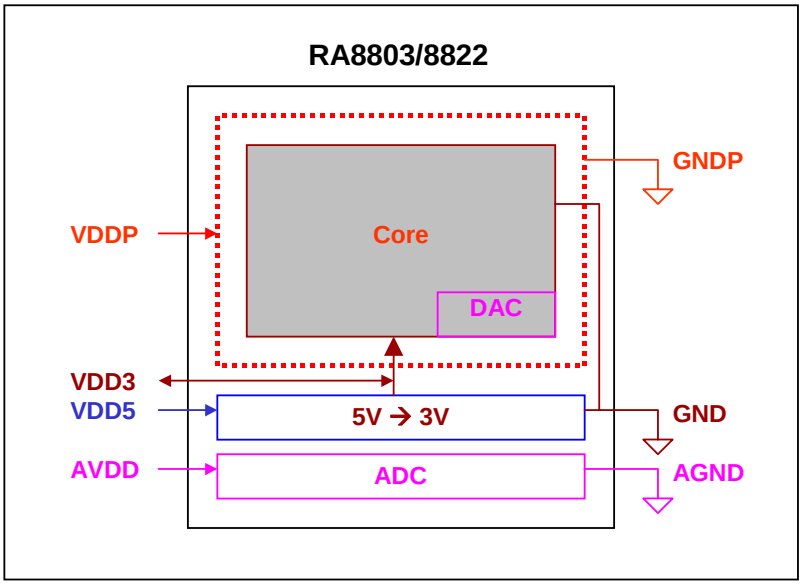


图 B-3：电源结构

如果系统应用上没有使用 RA8803/8822 内部的 ADC 电路，则建议将 AVDD 接到 VDD3。

B-2-2 3V电源应用电路

RA8803/8822 工作于 3V 可减少芯片及系统的电源功率消耗，其电源接法如图 B-4，由于此时内部 5V 转 3V 的 DC-to-DC 电路不需被使用，因此 VDD5 保持浮接即可。图 B-4 中的 0.47uH 电感与 0.1u 电容可以增加系统稳定度及避免 ADC 受到严重干扰影响精确度，如果系统工作环境较严格及有使用内部触摸控制器(Touch Panel Controller)，建议加上这些组件。

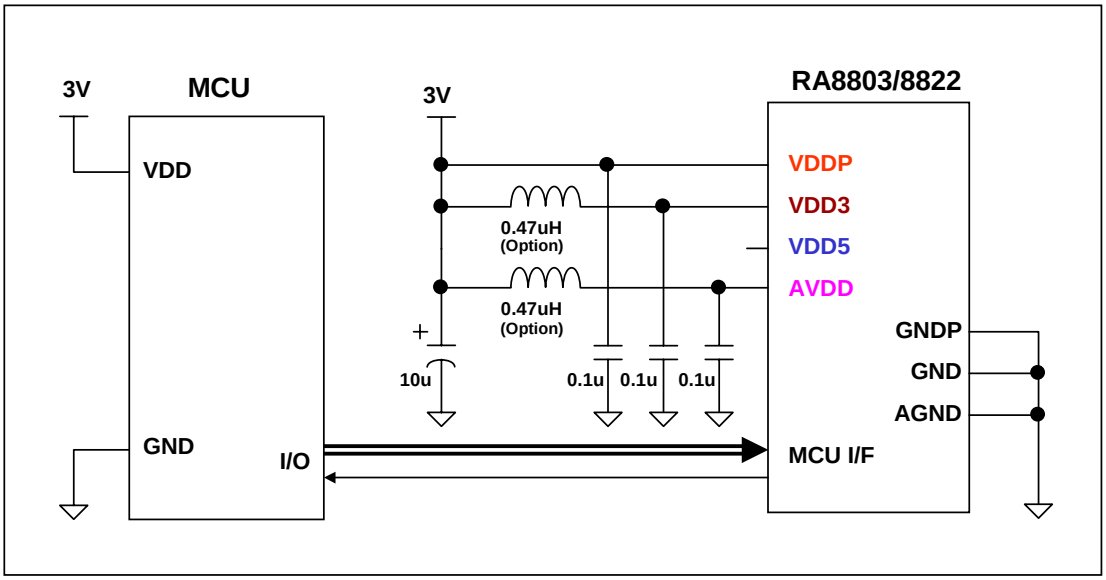


图 B-4：3V 电源应用电路

B-2-3 5V电源应用电路

当 RA8803/8822 工作于 5V 时，其电源接法建议如图 B-5，由于此时内部 5V 转 3V 的 DC-to-DC 电路必须启动以供给芯片内部的组件使用，因此 VDD5 接到 5V 的电源，同时为了增加 VDD3 的稳定性必须外加一 1uF 及 0.1uF 的电容。另外图 B-5 中的 0.47uH 电感与 0.1u 电容可以增加系统稳定度及避免 ADC 受到严重干扰影响精确度，如果系统工作环境较严格及有使用内部触摸控制器(Touch Panel Controller)，建议加上这些组件。

值得注意的是此情况下内部 DC-to-DC 电路会产生些许电流消耗，即使在 RA8803/8822 进入 Sleep 模式时仍有约 20uA 的静态电流耗损。

注: 原先 1.4 版之前的应用手册(含 1.4 版)并未提到 5V 转 3V 的 Dc to DC 电路，并且在 5V 的系统是将 5V 直接接到 VDDP、VDD3(VDD)及 AVDD，但在考虑系统的稳定度情况下，我们建议客户如果应用在 5V 的系统，可考虑使用图 B-5 的方式，同时也可以减少工作电流的消耗。

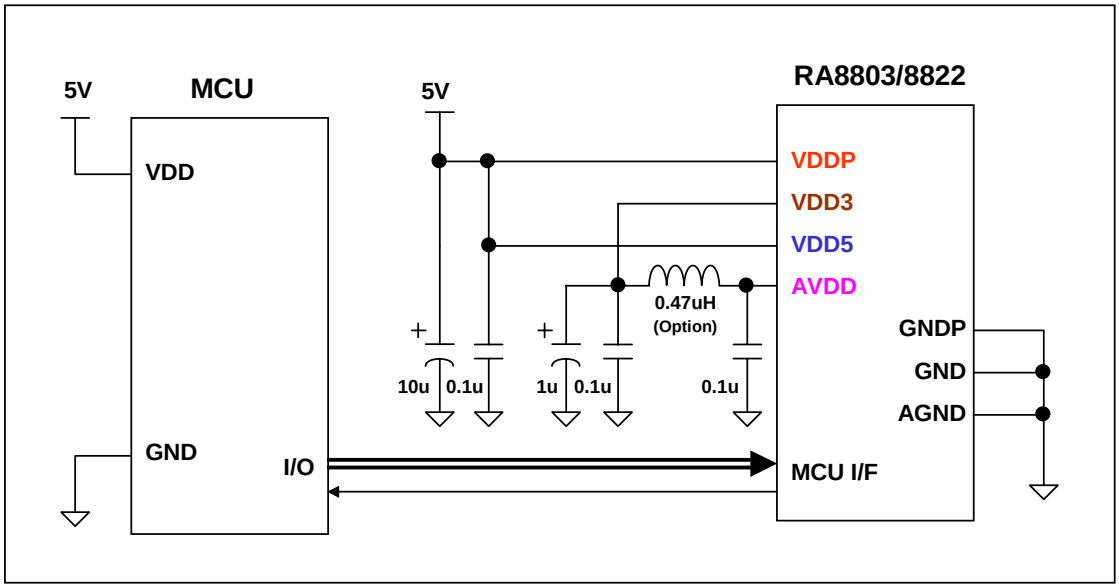


图 B-5：5V 电源应用电路

B-2-4 电路板的电源布局建议

图 B-6 是 PCB 在电源布局时的建议方式，可以减少芯片 ESD 受损的机会，并增加抗干扰能力，供使用者参考。

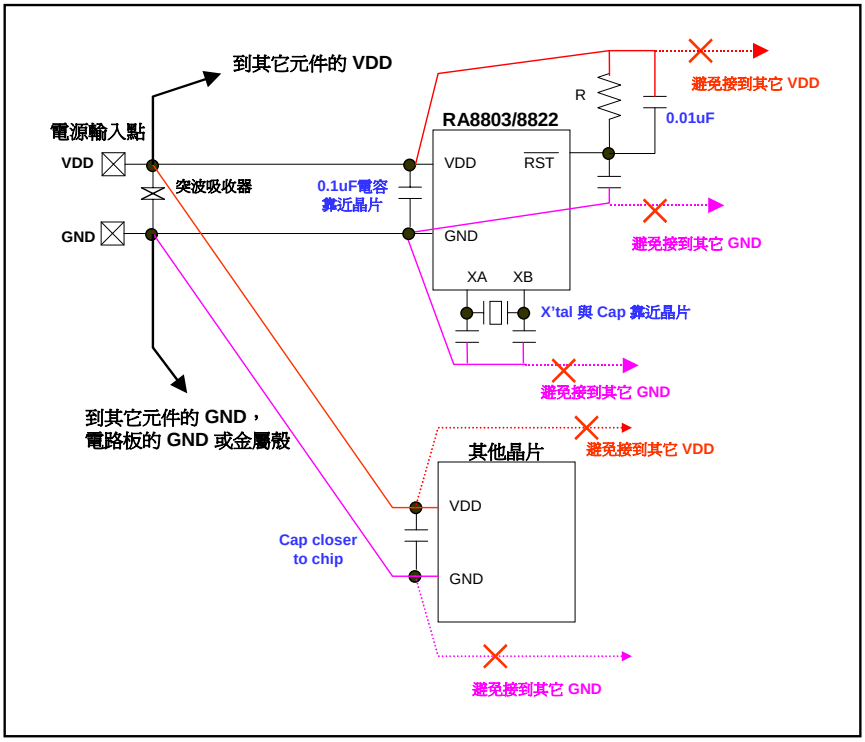
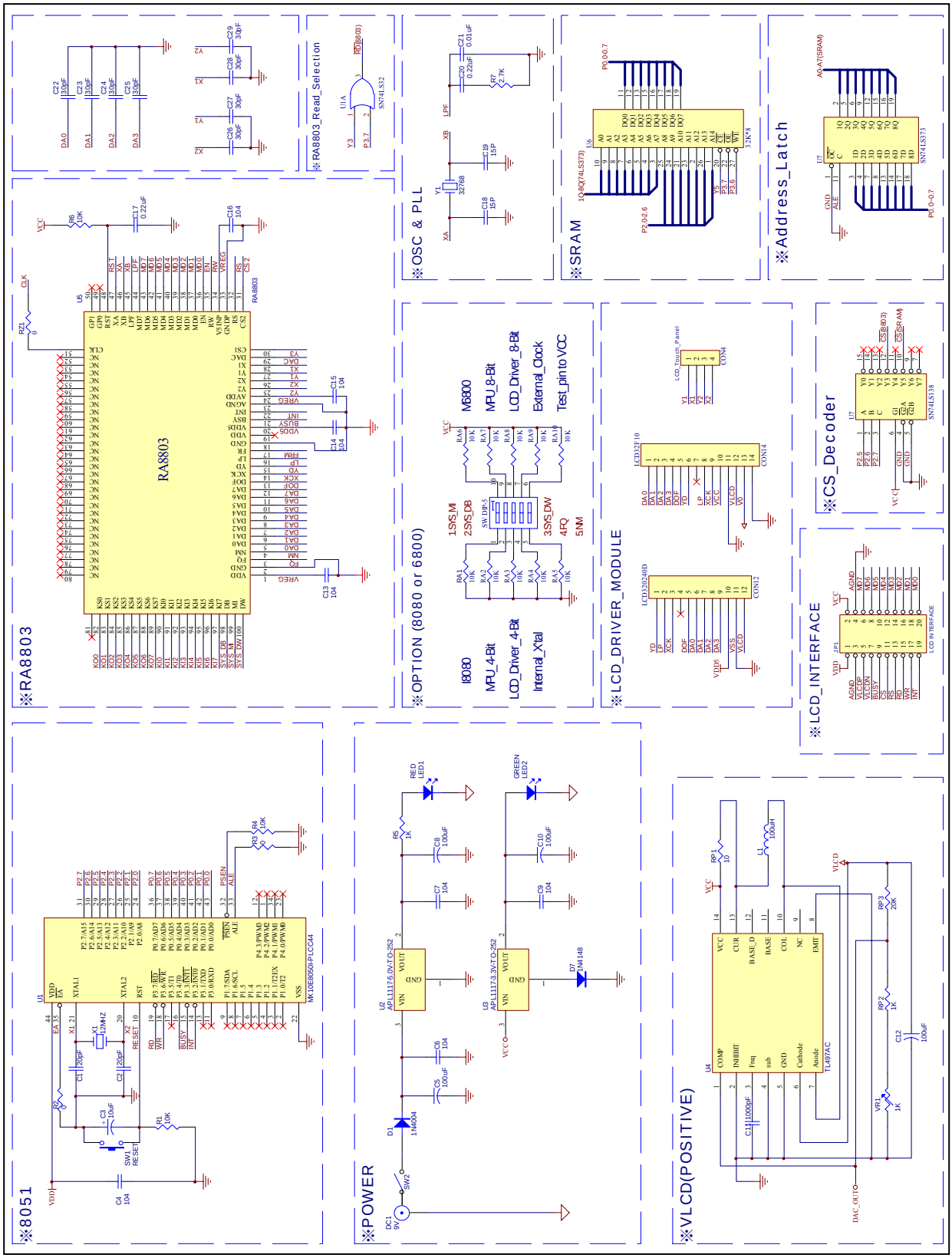


图 B-6：电源 PCB 布局建议

B-3 用 8051 的片选模式应用电路



附录 C. RA8803/8822 控制板

RAiO 提供 RA8803/8822 的控制板、Gerber File 与电路图，供客户以现有 Panel 或模块进行验证，图 C-1 为 RA8803/8822 的控制板，其电路图请参考图 B-1，客户可以利用 JP1 与使用的 MPU 相连接，现有 Panel 可以接到 JP3、LCD1、LCD2 或 LCD3 的 Driver 界面，Key-Scan 可以用 JP4 来实验，如图 C-2。

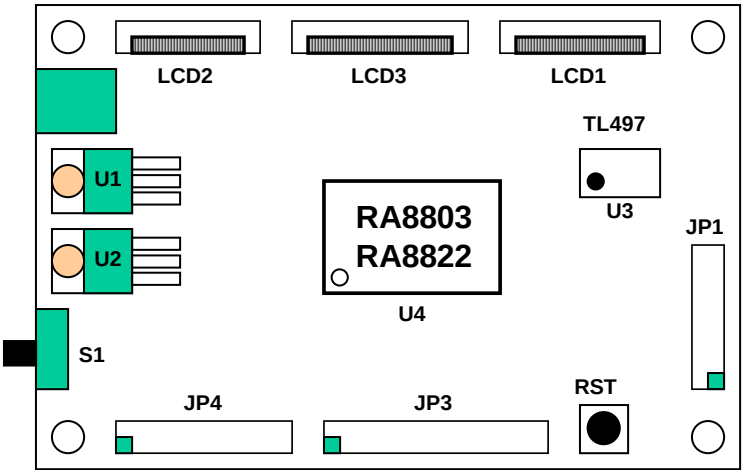


图 C-1：RA8803/8822 的控制板

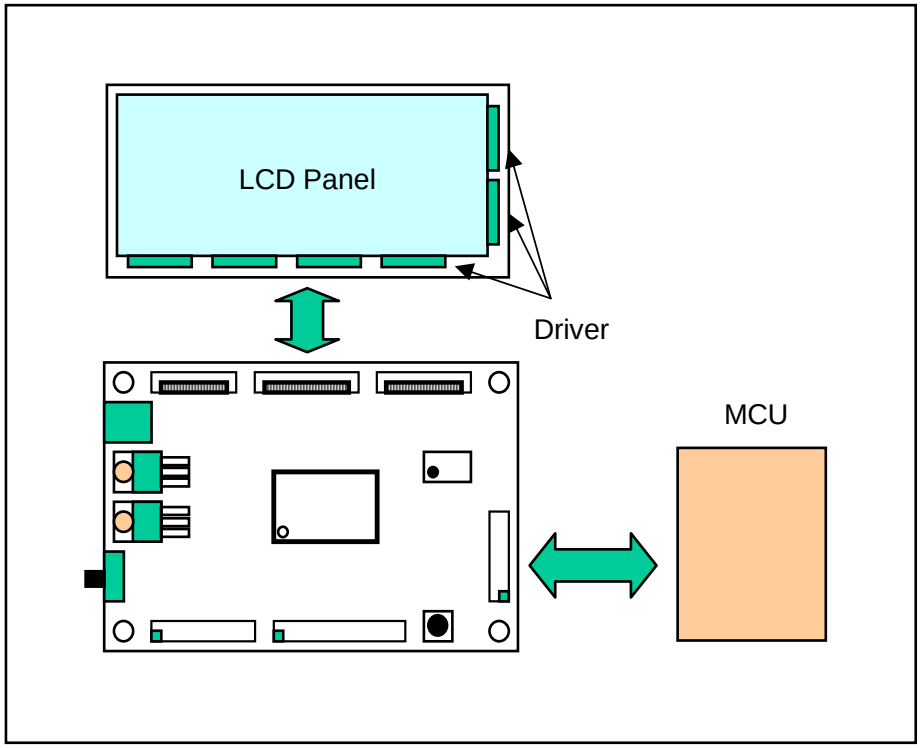


图 C-2：控制板应用图

附录 D. 除错与分析流程

附录 D 是在说明 RA8803/8822 在组合成 LCM 或系统时若遇到困难时所采取的几个步骤，例如 Demo Circuit 完成或 Demo PCB 焊接完成时，必须先核对的重要项目，在接上电源后检查 Clock、Reset 等信号，一但 MPU 可以透过软件进行缓存器的读写，表示 MPU 与 RA8803/8822 间的硬件设定基本上没有问题，MPU 透过软件进行缓存器的读写动作与 LCD Driver、Panel、或升压电路无关联。

若缓存器的读写没问题，可以透过文字或绘图模式进行显示部份的测试。若遇到困难此时就必须检查与 LCD Driver、Panel 或升压部份的电路了。

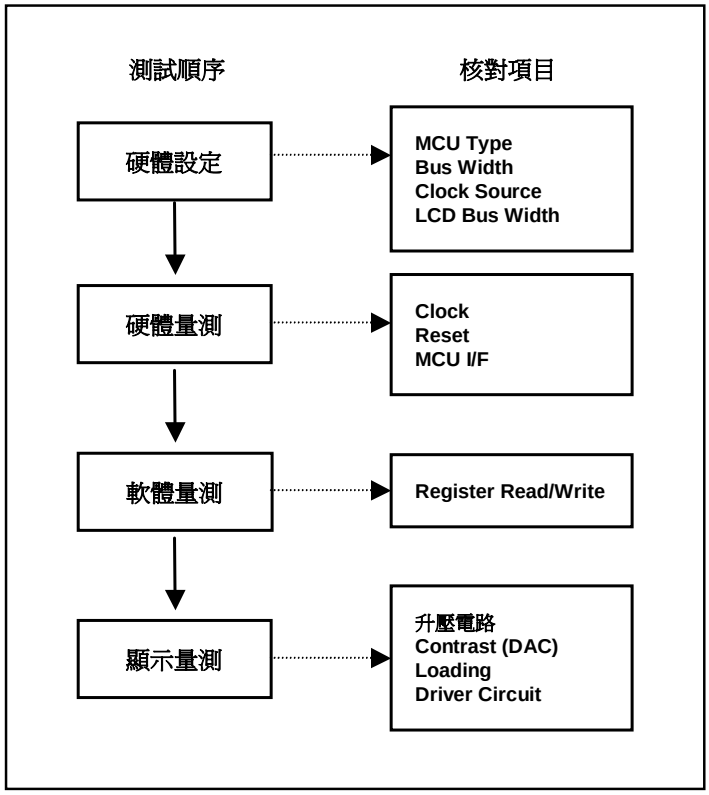


图 D-1：除错与分析流程

附录 E. RA8803/8822 支持的驱动器型号

Company	Driver Part.	Driver capacity	Support
HITACHI	HD66130	320-channel segment driver	▲：表示支持
SAMSUNG	S6A2067	80-dot segment driver	▲
	S6B0794	160-dot seg/com driver	▲
	S6B0086	80-dot seg/com driver	▲
	S6B2104	80-dot segment driver	▲
Novatek	NT3883	80-ch driver	--
	NT7701	160-dot seg/com driver	▲
	NT7702	240-dot seg/com driver	▲
	NT7703	160-dot seg/com driver	▲
	NT7704	240-dot seg/com driver	▲
Sitronix	ST7063	80-dot segment driver	--
	ST7065	40-dot seg/com driver	--
	ST8016	160-dot seg/com driver	▲
	ST8012	120-dot seg/com driver	▲
Elan	EM65160	160-dot seg/com driver	▲
	EM65240	240-dot seg/com driver	▲
	EM65H134	240-channel segment driver	▲
	EM65H137	240-channel common driver	▲
Toshiba	T6A92	80-channel segment driver	--
	T6B07	80-channel segment driver	▲
	T6B08	68-dot common driver	--
	T6B23	80-channel segment driver	--
	T6B36	80-dot common driver	▲
	T6C03	160-dot seg/com driver	▲
	T6C13B	240-dot seg/com driver	▲
	T6C25	160-dot seg/com driver	▲
	T6C61	160-channel segment driver	▲
	T6C63	240-channel segment driver	▲
	T6C72A	120-dot common driver	▲
	T6J05	128-dot common driver	▲
	T6J06	120-dot common driver	▲
Epson	S1D16501	100-dot common driver	▲
	S1D16700	100-dot common driver	▲
	S1D16702	68-dot common driver	▲
	S1D17403	160-dot common driver	▲
	S1D16006	80-channel segment driver	▲
	S1D16400	80-channel segment driver	▲
	S1D17503	120-dot common driver	▲
	S1D17508	160-dot segment driver	▲
Eureka	EK7010	240-dot seg/com driver	▲
	EK7011	160-dot seg/com driver	▲
Hynix	HM11S210	160-dot seg/com driver	▲
	HM11S220	240-dot seg/com driver	▲
Sanyo	LC79430	80-dot common driver	▲
	LC79401D	80-dot segment driver	▲

注: 未列于表内的其它 Driver，可以将编号提供给 RAiO 判断是否有支持。

附录 F. 指令时间

附录 D 是在说明 RA8803/8822 在做读/写或是各种模式下写到内存所需的时间。可依使用者所设定的不同系统频率(SYS_CLK)，来决定各个指令所需要的时间。例如，SYS_CLK=8MHz，每个 Clock 的时间=1/SYS_CLK=125ns，而写入缓存器所需的 Clock 为 3 个机械周期，所以对缓存器做读取或是写入时所需的时间约为 125ns X 3 lock=375ns，用以此方式来计算指令所需的时间。

下列是说明各个指令动作所需的机械周期时间：

- 写入缓存器的时间为 3 个机械周期
- 读取缓存器的时间为 3 个机械周期
- 写入内存的时间为 3 个机械周期
- 在绘图模式下写入内存的时间为 3 个机械周期
- 在中文字型下写入一个字到内存的时间为 35 个机械周期
- 在 ASCII 字型下写入一个字到内存的时间为 19 个机械周期
- 硬件清除屏幕所需的机械周期时间，公式： 3 + (Com x Seg) / 8

附录 G. 范例程序 – C51

G-1 范例程序(1) – 显示一中文字

```
//*****************************************************************************//
// ***  Demo1.c  ***************************************************************//
//*****************************************************************************//
#include <stdio.h>
#include <absacc.h>
#include "command.h"

void main (void)
{
  LCD_Reset();
  LCD_Initial();
  LCD_Clear();

  LCD_CmdWrite(0x00,0xCD);                    // 中文模式 + DISPLAY ON
  GotoXY(0,0);                                // 设定光标位置

    LCD_DataWrite(0xA4);                      // 显示单一中文字“中” (BIG-5 码为 A4A4)
    LCD_DataWrite(0xA4);
       .
       .
       .

}
```

G-2 范例程序(2) – 显示一中文字符串

```
//*****************************************************************************//
// ***  Demo2.c  ***************************************************************//
//*****************************************************************************//
#include <stdio.h>
#include <absacc.h>
#include "command.h"

void main (void)
{
  LCD_Reset();
  LCD_Initial();
  LCD_Clear();

  LCD_CmdWrite(0x00,0xCD);                    // 中文模式 + DISPLAY ON
  GotoXY(0,0);                                // 设定光标位置

    PrintStr("中文字号可变",1);               // 显示字符串“中文字号可变”
       .
       .
       .
}
```

G-3 范例程序(3) – 8x8(Key_Scan)扫描

```
//*****************************************************************************//
// ***  这个范例程序是做 8x8(Key_scan)扫描的程序  ***********************************//
// ***  程序说明: 8x8 按键,每按一个按键出现一个值,总共 1-64 个数字 **********************//
//*****************************************************************************//
#include <stdio.h>
#include <reg52.h>                          // 定义 8051 对 RA8803 输入/输出 控制信号
#include <absacc.h>
#include "command.h"                        // RA8803 读写命令 控制子程序
bit KeyPress;                               // 参数的宣告
void putHEX10(unsigned char var) small;
                                            // 子程序-LCD 显示”10 进制的”宣告
unsigned char FindKey(void);                // 子程序-取得按键输入值的宣告
bit Key_TP_ststus(void) small;              // 子程序-判断 key_scan 的宣告
void main (void)
{

   bit KeyPress;                            // 参数的宣告
   unsigned char Key_Number;
   unsigned char x_temp=0,y_temp=0;
   unsigned char test;
  LCD_Reset();                              // 重置 RA8803 “pin-48 RESET”维持
                                            // 低电位 250ms
  LCD_Initial();                            // 所有缓存器的初始值设定
  LCD_Clear();                              // 屏幕清除
  LCD_CmdWrite(0xA1,0xF1);                  // 设定缓存器(Kscr)
  LCD_CmdWrite(0xA0,0x08);                  // 设定缓存器(Intr)
  LCD_CmdWrite(0x00,0xCD);                  // 设定写入 LCD 数据为文字模式且将 LCD
                                            // 显示开启,
  GotoXY(0,0);                              // 设定显示坐标
  PrintStr("Please pull any key",00);
                                            // LCD 显示字符串"Please pull any key"
  GotoXY(0,40);                             // 设定显示坐标
  PrintStr("Key Number : ",0);              // LCD 显示字符串” Key Number :”
  KeyPress = 0;

  while(1)                                  // 侦测按键输入主程序
  {
    if(Key_TP_ststus())                     // 判断 key_scan 是否有中断
    {

      delay(500);                           // 延迟时间
      GotoXY(13,40);                        // 设定显示坐标
      Key_Number = FindKey();               // 取得按键输入的值
      putHEX10(Key_Number);                 // 把值转换成 10 进制并显示于 LCD
    }
  }
while(1);
}
```

```
//*****************************************************************************//
//  ***  Key_scan 的判断子程序 ***********************************************//
//*****************************************************************************//
bit Key_TP_ststus(void) small

{
  unsigned char temp;

  temp = LCD_CmdRead(0xA0);                    // 读取缓存器[0xA0],并判断 Bit7 是否为"1"
  temp &= 0x80;
  LCD_CmdWrite(0xA1,0xF1);                     //设定缓存器(Kscr)
  delay(5000);                                 //延迟时间
  if(temp)
     return 1;                                 // 若缓存器[0xA0]-Bit7 = 1,表示有按键输入
  else
     return 0;                                 // 若缓存器[0xA0]-Bit7 = 0,表示无按键输入
}


//*****************************************************************************//
// ***  取得按键输入的值子程序  *****************************************//
//*****************************************************************************//
unsigned char FindKey(void)
{
  unsigned char Key_Out;
  unsigned char Key_In;
  unsigned char Key_Number;

  Key_Out = LCD_CmdRead(0xa2);                 //读取按键输出缓存器
  Key_In = LCD_CmdRead(0xa3);                  //读取按键输入缓存器
  switch(Key_Out)                              //扫描判断方式,并设定相对应的按键数值
  {
    case 0xfe:
          Key_Number = 1;
          break;
    case 0xfd:
          Key_Number = 9;
          break;
    case 0xfb:
          Key_Number = 17;
          break;
    case 0xf7:
          Key_Number = 25;
          break;
    case 0xef:
          Key_Number = 33;
          break;
    case 0xdf:
          Key_Number = 41;
          break;
    case 0xbf:
          Key_Number = 49;
          break;
    case 0x7f:
          Key_Number = 57;
```

```
            break;
    default:
            break;
}
   switch(Key_In)                                    //扫描判断方式,并设定相对应的按键数值
   {
     case 0xfe:
            Key_Number += 0;
            break;
     case 0xfd:
            Key_Number += 1;
            break;
     case 0xfb:
            Key_Number += 2;
            break;
     case 0xf7:
            Key_Number += 3;
            break;
     case 0xef:
            Key_Number += 4;
            break;
     case 0xdf:
            Key_Number += 5;
            break;
     case 0xbf:
            Key_Number += 6;
            break;
     case 0x7f:
            Key_Number += 7;
            break;
     default:
            break;
   }
   return Key_Number;                                // 回传输入按键的数值

}


//*****************************************************************************//
// ***  LCD 显示"10 进制转换数值"  ***************************************//
//*****************************************************************************//
void putHEX10(unsigned char var) small
{
  unsigned char div_val,base;
  base = 10;
  div_val = 10;
    do
    {
      LCD_DataWrite(ASCIITable1[var / div_val]);
                                                     // 抓出商显示于 LCD
      var %= div_val;
      div_val /= base;
    } while (div_val);
}
```

G-4 子程序范例

```
//*****************************************************************************//
// ***  COMMAND.H  ***********************************************************//
//*****************************************************************************//
extern void LCD_CmdWrite(unsigned char, unsigned char) small;
extern unsigned char LCD_CmdRead(unsigned char) small;
extern void LCD_DataWrite(unsigned char) small;
extern void delay (int i);
extern void LCD_ChkBusy(void) small;
extern void LCD_Reset(void) small;
extern void LCD_Initial(void) small;
extern void LCD_Clear(void) small;
extern void GotoXY(unsigned char x1,unsigned char y1) small;
extern void PrintStr(char *ptr,int delay_time) small;
extern void putHEX2(unsigned char var) small;


sbit P0_0 = 0x80;
sbit P0_1 = 0x81;
sbit P0_2 = 0x82;
sbit P0_3 = 0x83;
sbit P0_4 = 0x84;
sbit P0_5 = 0x85;
sbit P0_6 = 0x86;
sbit P0_7 = 0x87;

sbit P3_0 = P3^0;
sbit P3_1 = P3^1;
sbit P3_2 = P3^2;
sbit P3_3 = P3^3;
sbit P3_4 = P3^4;
sbit P3_5 = P3^5;
sbit P3_6 = P3^6;
sbit P3_7 = P3^7;


//*****************************************************************************//
// ***  缓存器 写入 子程序  ***************************************************//
//*****************************************************************************//
void LCD_CmdWrite(unsigned char cmdReg, unsigned char cmdData) small
{
  LCD_cmdReg = cmdReg;                                  // 写入缓存器的地址
  LCD_CS1 =0;                                           // 致能 RA8803 读写动作
  LCD_RD = 1;                                           // 禁能读取动作
  LCD_RS = 0;                                           // 设定为写入资至缓存器模式
  LCD_WR = 0;                                           // 开始写入缓存器的地址
  LCD_WR = 1;
  LCD_RS = 1;
  LCD_CS1 =1;                                           // 致能 RA8803 读写动作
  //……
  LCD_cmdReg = cmdReg;                                  // 写入缓存器的数据
  LCD_CS1 =0;                                           // 致能 RA8803 读写动作
  LCD_RD = 1;                                           // 禁能读取动作
```

```
  LCD_RS = 0;                                   // 设定为写入数据至缓存器模式

  LCD_WR = 0;                                   // 开始写入缓存器的数据
  LCD_WR = 1;
  LCD_RS = 1;
  LCD_CS1 =1;                                   // 致能 RA8803 读写动作
}


//*************************************************************************//
// ***  缓存器 读取 子程序  ***************************************************//
//*************************************************************************//
unsigned char LCD_CmdRead(unsigned char cmdReg) small
{
  LCD_ChkBusy();                                // 判断 RA8803 是否忙碌中
  LCD_cmdReg = cmdReg;                          // 写入缓存器的地址
  LCD_CS1 =0;                                   // 致能 RA8803 读写动作
  LCD_RD = 1;                                   // 禁能读取动作
  LCD_RS = 0;                                   // 设定为写入数据至缓存器模式

  LCD_WR = 0;                                   // 开始写入缓存器的地址
  LCD_WR = 1;
  LCD_RS = 1;
  LCD_CS1 =1;                                   // 致能 RA8803 读写动作

//……
//
  LCD_DATA = 0xff;
  LCD_CS1 =0;                                   // 致能 RA8803 读写动作
  LCD_WR = 1;                                   // 禁能写入动作
  LCD_RS = 0;                                   // 设定从缓存器读取数据模式

  LCD_RD = 0;                                   // 开始读取缓存器的数据
  REG_Read = LCD_DATA;
  LCD_RD = 1;                                   // 致能读取缓存器

  LCD_RS = 1;
  LCD_CS1 =1;                                   // 禁能 RA8803 读写动作
  return REG_Read;                              // 回传读取缓存器的数据
}


//*************************************************************************//
// ***  写入数据至显示内存 子程序  ********************************************//
//*************************************************************************//
void LCD_DataWrite(unsigned char WrData) small
{
  LCD_ChkBusy();                                // 判断 RA8803 是否忙碌中
  LCD_DATA = WrData;                            // 准备预写入显示内存的数据
  LCD_CS1 =0;                                   // 致能 RA8803 读写动作
  LCD_RD = 1;                                   // 禁能读取动作
  LCD_RS = 1;                                   // 设定为写入数据至显示内存模式
  LCD_WR = 0;                                   // 开始写数据至显示内存
```

```
  LCD_WR = 1;

  LCD_RS = 1;
  LCD_CS1 =1;                                          // 禁能 RA8803 读写动作
}


//*************************************************************************//
//***  检查忙碌旗标 子程序  ***********************************************//
//*************************************************************************//
void LCD_ChkBusy(void) small
{
  do
  {
  }
  while(LCD_BUSY == 1);
}


//*************************************************************************//
// ***  重置 子程序  ******************************************************//
//*** RA8803/22 重置时间至少需要维持低准位为 250 豪秒  " >= (250ms)" ********//
//*************************************************************************//
void LCD_Reset(void) small
{
  LCD_READY = 0xff;
  LCD_CS1 = 0;
  delay(1000);                                         // "使用 8051 外挂频率为 6MHz"
  LCD_RST = 0;
  delay(11000);                                        // 低准位大于 250 毫秒
  LCD_RST = 1;
  delay(1000);
}


//*************************************************************************//
// *** 清除显示内存 子程序  ********************************************//
//*************************************************************************//
void LCD_Clear(void) small
{
  unsigned char READ_REG;
  LCD_CmdWrite(0xE0,0x00);                             // 设定将显示内存的数据全部写入"0x00"
  READ_REG = LCD_CmdRead(0xF0);
  READ_REG &= 0xF7;

  READ_REG |= 0x08;
  LCD_CmdWrite(0xF0,READ_REG);                         // 当缓存器[0xF0]之位 3 设定为"1"时,
  delay(1000);                                         // 硬件自动将显示内存全部写入缓存器[E0]的资
                                                       // 料
}
```

```
//*****************************************************************************//
// *** 设定 显示坐标 (内存地址) 子程序 ********************************//
//*****************************************************************************//
extern void GotoXY(unsigned char x1,unsigned char y1) small
{
  LCD_CmdWrite(0x60,x1);                                  // 设定水平坐标地址
  LCD_CmdWrite(0x70,y1);                                  // 设定垂直坐标地址
}

//*****************************************************************************//
// *** 延迟 子程序  ***************************************************************//
//*****************************************************************************//
void delay(int i)                                         // 延迟时间子程序
{
  int k ;
  for ( k=0 ; k < i ; k++ );
}


//*****************************************************************************//
// *** 定义 所有 缓存器的 初始值 ***************************************//
//*****************************************************************************//
void LCD_Initial(void) small
{
  LCD_CmdWrite(0x00,0xC9);                                // LCD 基本显示功能设定-1
  LCD_CmdWrite(0x01,0xF1);                                // 系统工作频率与中断准位设定
  LCD_CmdWrite(0x02,0x10);                                // LCD 内存读写速度与功能设定
  LCD_CmdWrite(0x03,0x80);                                // LCD 特殊显示功能
  LCD_CmdWrite(0x10,0x6B);                                // LCD 基本显示功能设定-2
  LCD_CmdWrite(0x11,0x22);                                // 设定光标高度与行距
  LCD_CmdWrite(0x12,0x91);                                // LCD 显示图层设定

  LCD_CmdWrite(0x20,0x27);                                // 设定实际显示窗口垂直起始地址(X1)
  LCD_CmdWrite(0x30,0xEF);                                // 设定实际显示窗口水平起始地址(Y1)
  LCD_CmdWrite(0x40,0x00);                                // 设定实际显示窗口垂直结束地址(X2)
  LCD_CmdWrite(0x50,0x00);                                // 设定实际显示窗口水平结束地址(Y2)

  LCD_CmdWrite(0x21,0x27);                                // 设定 LCD 模块垂直起始地址(X1)
  LCD_CmdWrite(0x31,0xEF);                                // 设定 LCD 模块水平起始地址(Y1)
  LCD_CmdWrite(0x41,0x00);                                // 设定 LCD 模块垂直结束地址(X2)
  LCD_CmdWrite(0x51,0x00);                                // 设定 LCD 模块水平结束地址(Y2)

  LCD_CmdWrite(0x60,0x00);                                // 设定光标水平地址(Common)
  LCD_CmdWrite(0x61,0x00);                                // 设定垂直开始显示地址(Segment)
  LCD_CmdWrite(0x70,0x00);                                // 设定光标垂直地址(Segment)
  LCD_CmdWrite(0x71,0x00);                                // 设定于水平区块移动模式下,起始地址
  LCD_CmdWrite(0x72,0xEF);                                // 设定于水平区块移动模式下,结束地址

  LCD_CmdWrite(0x80,0x33);                                // 光标闪烁时间设定
  LCD_CmdWrite(0x81,0x00);                                // 变换 FRM 极性的起始地址(Common)
  LCD_CmdWrite(0x91,0x00);                                // 变换 FRM 极性的结束地址(Common)
  LCD_CmdWrite(0x90,0x04);
```

```
  LCD_CmdWrite(0xA0,0x11);                    // 中断功能设定
  LCD_CmdWrite(0xA1,0x00);                    // 键盘扫描控制缓存器
  LCD_CmdWrite(0xA2,0x00);                    // 键盘扫描控制输出数据
  LCD_CmdWrite(0xA3,0x00);                    // 键盘扫描控制输入数据

  LCD_CmdWrite(0xB0,0x27);                    // 设定水平方向断点的地址
  LCD_CmdWrite(0xB1,0xEF);                    // 设定垂直方向断点的地址

  LCD_CmdWrite(0xC0,0xD0);                    // 触控屏幕功能设定
  LCD_CmdWrite(0xC1,0x0A);                    // 读写并可侦测触控屏幕的状态
  LCD_CmdWrite(0xC8,0x80);                    // 读取触控屏幕水平轴的数据(高字节)
  LCD_CmdWrite(0xC9,0x80);                    // 读取触控屏幕水平轴的数据(高字节)
  LCD_CmdWrite(0xCA,0x00);                    // 读取触控屏幕水平轴与垂直轴的数据(低字节)

  LCD_CmdWrite(0xD0,0x0C);                    // 设定 DAC 电流输出，可调整 LCD 亮度

  LCD_CmdWrite(0xE0,0x00);                    // 设定写入显示内存的数据(需搭配缓存器[F0]
                                              // 使用”
  LCD_CmdWrite(0xF0,0x90);                    // 设定中文字型为”BIG5”
  LCD_CmdWrite(0xF1,0x0F);                    // 改变字型垂直与水平显示大小
}


//***************************************************************************//
// *** LCD 显示字符串 子程序  ***************************************************//
//***************************************************************************//
void PrintStr(char *ptr,int delay_time) small
{
  while(*ptr != '\0')                         // 判断是否为空字符，若不是，继续写入与显示下
  {                                           // 一个字
    LCD_DataWrite(*ptr);                      // 写入显示内存
    ++ptr;
    delay(delay_time);                        // 每个字显示的时间间隔
  }
}


//***************************************************************************//
// *** LCD 显示字符 子程序  ***************************************************//
//***************************************************************************//
void putHEX2(unsigned char var) small
{                                             // LCD 显示两个 16 进制文字
  unsigned char div_val,base;
  base = 16;                                  // 设定参数”两个字节”
  div_val = 0x10;
  do
  {
    LCD_DataWrite(ASCIITable[var / div_val]); // 写入显示内存
    var %= div_val;
    div_val /= base;
   } while (div_val);
}
```

附录 H. 字型与字码表(GB)

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	{	|	}	￣	

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

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

A6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
B	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
C		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
D	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
E	⏤	⏥	⏠	⏡	⏢	⏣	⏦	⏧	⏤	⏥	⏤	⏥			⏤	⏥
F	⏤	⏥	|		|	|										

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																

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

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

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

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

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

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

B0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		啊	阿	埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱	隘
B	鞍	氨	安	俺	按	暗	岸	胺	案	肮	昂	盎	凹	敖	熬	翱
C	袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	笆	八	疤	巴	拔	跋
D	靶	把	耙	坝	霸	罢	爸	白	柏	百	摆	佰	败	拜	稗	斑
E	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	邦	帮
F	梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤	苞	胞	包	褒	剥	

B1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		薄	雹	保	堡	饱	宝	抱	报	暴	豹	鲍	爆	杯	碑	悲
B	卑	北	辈	背	贝	钡	倍	狈	备	惫	焙	被	奔	苯	本	笨
C	崩	绷	甭	泵	蹦	迸	逼	鼻	比	鄙	笔	彼	碧	蓖	蔽	毕
D	毙	毖	币	庇	痹	闭	敝	弊	必	辟	壁	臂	避	陛	鞭	边
E	编	贬	扁	便	变	卞	辨	辩	辫	遍	标	彪	膘	表	鳖	憋
F	别	瘪	彬	斌	濒	滨	宾	摈	兵	冰	柄	丙	秉	饼	炳	

B2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		病	并	玻	菠	播	拨	钵	波	博	勃	搏	铂	箔	伯	帛
B	舶	脖	膊	渤	泊	驳	捕	卜	哺	补	埠	不	布	步	簿	部
C	怖	擦	猜	裁	材	才	财	睬	踩	采	彩	菜	蔡	餐	参	蚕
D	残	惭	惨	灿	苍	舱	仓	沧	藏	操	糙	槽	曹	草	厕	策
E	侧	册	测	层	蹭	插	叉	茬	茶	查	碴	搽	察	岔	差	诧
F	拆	柴	豺	搀	掺	蝉	馋	谗	缠	铲	产	阐	颤	昌	猖	

B3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		场	尝	常	长	偿	肠	厂	敞	畅	唱	倡	超	抄	钞	朝
B	嘲	潮	巢	吵	炒	车	扯	撤	掣	彻	澈	郴	臣	辰	尘	晨
C	忱	沉	陈	趁	衬	撑	称	城	橙	成	呈	乘	程	惩	澄	诚
D	承	逞	骋	秤	吃	痴	持	匙	池	迟	弛	驰	耻	齿	侈	尺
E	赤	翅	斥	炽	充	冲	虫	崇	宠	抽	酬	畴	踌	稠	愁	筹
F	仇	绸	瞅	丑	臭	初	出	橱	厨	躇	锄	雏	滁	除	楚	

B4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		础	储	矗	搐	触	处	揣	川	穿	椽	传	船	喘	串	疮
B	窗	幢	床	闯	创	吹	炊	捶	锤	垂	春	椿	醇	唇	淳	纯
C	蠢	戳	绰	疵	茨	磁	雌	辞	慈	瓷	词	此	刺	赐	次	聪
D	葱	囱	匆	从	丛	凑	粗	醋	簇	促	蹿	篡	窜	摧	崔	催
E	脆	瘁	粹	淬	翠	村	存	寸	磋	撮	搓	措	挫	错	搭	达
F	答	瘩	打	大	呆	歹	傣	戴	带	殆	代	贷	袋	待	逮	

B5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		怠	耽	担	丹	单	郸	掸	胆	旦	氮	但	惮	淡	诞	弹
B	蛋	当	挡	党	荡	档	刀	捣	蹈	倒	岛	祷	导	到	稻	悼
C	道	盗	德	得	的	蹬	灯	登	等	瞪	凳	邓	堤	低	滴	迪
D	敌	笛	狄	涤	翟	嫡	抵	底	地	蒂	第	帝	弟	递	缔	颠
E	掂	滇	碘	点	典	靛	垫	电	佃	甸	店	惦	奠	淀	殿	碉
F	叼	雕	凋	刁	掉	吊	钓	调	跌	爹	碟	蝶	迭	谍	叠	

B6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		丁	盯	叮	钉	顶	鼎	锭	定	订	丢	东	冬	董	懂	动
B	栋	侗	恫	冻	洞	兜	抖	斗	陡	豆	逗	痘	都	督	毒	犊
C	独	读	堵	睹	赌	杜	镀	肚	度	渡	妒	端	短	锻	段	断
D	缎	堆	兑	队	对	墩	吨	蹲	敦	顿	囤	钝	盾	遁	掇	哆
E	多	夺	垛	躲	朵	跺	舵	剁	惰	堕	蛾	峨	鹅	俄	额	讹
F	娥	恶	厄	扼	遏	鄂	饿	恩	而	儿	耳	尔	饵	洱	二	

B7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		贰	发	罚	筏	伐	乏	阀	法	珐	藩	帆	番	翻	樊	矾
B	钒	繁	凡	烦	反	返	范	贩	犯	饭	泛	坊	芳	方	肪	房
C	防	妨	仿	访	纺	放	菲	非	啡	飞	肥	匪	诽	吠	肺	废
D	沸	费	芬	酚	吩	氛	分	纷	坟	焚	汾	粉	奋	份	忿	愤
E	粪	丰	封	枫	蜂	峰	锋	风	疯	烽	逢	冯	缝	讽	奉	凤
F	佛	否	夫	敷	肤	孵	扶	拂	辐	幅	氟	符	伏	俘	服	

B8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		浮	涪	福	袱	弗	甫	抚	辅	俯	釜	斧	脯	腑	府	腐
B	赴	副	覆	赋	复	傅	付	阜	父	腹	负	富	讣	附	妇	缚
C	咐	噶	嘎	该	改	概	钙	盖	溉	干	甘	杆	柑	竿	肝	赶
D	感	秆	敢	赣	冈	刚	钢	缸	肛	纲	岗	港	杠	篙	皋	高
E	膏	羔	糕	搞	镐	稿	告	哥	歌	搁	戈	鸽	胳	疙	割	革
F	葛	格	蛤	阁	隔	铬	个	各	给	根	跟	耕	更	庚	羹	

B9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		埂	耿	梗	工	攻	功	恭	龚	供	躬	公	宫	弓	巩	汞
B	拱	贡	共	钩	勾	沟	苟	狗	垢	构	购	够	辜	菇	咕	箍
C	估	沽	孤	姑	鼓	古	蛊	骨	谷	股	故	顾	固	雇	刮	瓜
D	剐	寡	挂	褂	乖	拐	怪	棺	关	官	冠	观	管	馆	罐	惯
E	灌	贯	光	广	逛	瑰	规	圭	硅	归	龟	闺	轨	鬼	诡	癸
F	桂	柜	跪	贵	刽	辊	滚	棍	锅	郭	国	果	裹	过	哈	

BA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		骸	孩	海	氦	亥	害	骇	酣	憨	邯	韩	含	涵	寒	函
B	喊	罕	翰	撼	捍	旱	憾	悍	焊	汗	汉	夯	杭	航	壕	嚎
C	豪	毫	郝	好	耗	号	浩	呵	喝	荷	菏	核	禾	和	何	合
D	盒	貉	阂	河	涸	赫	褐	鹤	贺	嘿	黑	痕	很	狠	恨	哼
E	亨	横	衡	恒	轰	哄	烘	虹	鸿	洪	宏	弘	红	喉	侯	猴
F	吼	厚	候	后	呼	乎	忽	瑚	壶	葫	胡	蝴	狐	糊	湖	

BB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		弧	虎	唬	护	互	沪	户	花	哗	华	猾	滑	画	划	化
B	话	槐	徊	怀	淮	坏	欢	环	桓	还	缓	换	患	唤	痪	豢
C	焕	涣	宦	幻	荒	慌	黄	磺	蝗	簧	皇	凰	惶	煌	晃	幌
D	恍	谎	灰	挥	辉	徽	恢	蛔	回	毁	悔	慧	卉	惠	晦	贿
E	秽	会	烩	汇	讳	诲	绘	荤	昏	婚	魂	浑	混	豁	活	伙
F	火	获	或	惑	霍	货	祸	击	圾	基	机	畸	稽	积	箕	

BC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		肌	饥	迹	激	讥	鸡	姬	绩	缉	吉	极	棘	辑	籍	集
B	及	急	疾	汲	即	嫉	级	挤	几	脊	己	蓟	技	冀	季	伎
C	祭	剂	悸	济	寄	寂	计	记	既	忌	际	妓	继	纪	嘉	枷
D	夹	佳	家	加	荚	颊	贾	甲	钾	假	稼	价	架	驾	嫁	歼
E	监	坚	尖	笺	间	煎	兼	肩	艰	奸	缄	茧	检	柬	碱	硷
F	拣	捡	简	俭	剪	减	荐	槛	鉴	践	贱	见	键	箭	件	

BD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		健	舰	剑	饯	渐	溅	涧	建	僵	姜	将	浆	江	疆	蒋
B	桨	奖	讲	匠	酱	降	蕉	椒	礁	焦	胶	交	郊	浇	骄	娇
C	嚼	搅	铰	矫	侥	脚	狡	角	饺	缴	绞	剿	教	酵	轿	较
D	叫	窖	揭	接	皆	秸	街	阶	截	劫	节	桔	杰	捷	睫	竭
E	洁	结	解	姐	戒	藉	芥	界	借	介	疥	诫	届	巾	筋	斤
F	金	今	津	襟	紧	锦	仅	谨	进	靳	晋	禁	近	烬	浸	

BE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		尽	劲	荆	兢	茎	睛	晶	鲸	京	惊	精	粳	经	井	警
B	景	颈	静	境	敬	镜	径	痉	靖	竟	竞	净	炯	窘	揪	究
C	纠	玖	韭	久	灸	九	酒	厩	救	旧	臼	舅	咎	就	疚	鞠
D	拘	狙	疽	居	驹	菊	局	咀	矩	举	沮	聚	拒	据	巨	具
E	距	踞	锯	俱	句	惧	炬	剧	捐	鹃	娟	倦	眷	卷	绢	撅
F	攫	抉	掘	倔	爵	觉	决	诀	绝	均	菌	钧	军	君	峻	

BF	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		俊	竣	浚	郡	骏	喀	咖	卡	咯	开	揩	楷	凯	慨	刊
B	堪	勘	坎	砍	看	康	慷	糠	扛	抗	亢	炕	考	拷	烤	靠
C	坷	苛	柯	棵	磕	颗	科	壳	咳	可	渴	克	刻	客	课	肯
D	啃	垦	恳	坑	吭	空	恐	孔	控	抠	口	扣	寇	枯	哭	窟
E	苦	酷	库	裤	夸	垮	挎	跨	胯	块	筷	侩	快	宽	款	匡
F	筐	狂	框	矿	眶	旷	况	亏	盔	岿	窥	葵	奎	魁	傀	

C0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		馈	愧	溃	坤	昆	捆	困	括	扩	廓	阔	垃	拉	喇	蜡
B	腊	辣	啦	莱	来	赖	蓝	婪	栏	拦	篮	阑	兰	澜	谰	揽
C	览	懒	缆	烂	滥	琅	榔	狼	廊	郎	朗	浪	捞	劳	牢	老
D	佬	姥	酪	烙	涝	勒	乐	雷	镭	蕾	磊	累	儡	垒	擂	肋
E	类	泪	棱	楞	冷	厘	梨	犁	黎	篱	狸	离	漓	理	李	里
F	鲤	礼	莉	荔	吏	栗	丽	厉	励	砾	历	利	傈	例	俐	

C1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		痢	立	粒	沥	隶	力	璃	哩	俩	联	莲	连	镰	廉	怜
B	涟	帘	敛	脸	链	恋	炼	练	粮	凉	梁	粱	良	两	辆	量
C	晾	亮	谅	撩	聊	僚	疗	燎	寥	辽	潦	了	撂	镣	廖	料
D	列	裂	烈	劣	猎	琳	林	磷	霖	临	邻	鳞	淋	凛	赁	吝
E	拎	玲	菱	零	龄	铃	伶	羚	凌	灵	陵	岭	领	另	令	溜
F	琉	榴	硫	馏	留	刘	瘤	流	柳	六	龙	聋	咙	笼	窿	

C2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		隆	垄	拢	陇	楼	娄	搂	篓	漏	陋	芦	卢	颅	庐	炉
B	掳	卤	虏	鲁	麓	碌	露	路	赂	鹿	潞	禄	录	陆	戮	驴
C	吕	铝	侣	旅	履	屡	缕	虑	氯	律	率	滤	绿	峦	挛	孪
D	滦	卵	乱	掠	略	抡	轮	伦	仑	沦	纶	论	萝	螺	罗	逻
E	锣	箩	骡	裸	落	洛	骆	络	妈	麻	玛	码	蚂	马	骂	嘛
F	吗	埋	买	麦	卖	迈	脉	瞒	馒	蛮	满	蔓	曼	慢	漫	

C3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		谩	芒	茫	盲	氓	忙	莽	猫	茅	锚	毛	矛	铆	卯	茂
B	冒	帽	貌	贸	么	玫	枚	梅	酶	霉	煤	没	眉	媒	镁	每
C	美	昧	寐	妹	媚	门	闷	们	萌	蒙	檬	盟	锰	猛	梦	孟
D	眯	醚	靡	糜	迷	谜	弥	米	秘	觅	泌	蜜	密	幂	棉	眠
E	绵	冕	免	勉	娩	缅	面	苗	描	瞄	藐	秒	渺	庙	妙	蔑
F	灭	民	抿	皿	敏	悯	闽	明	螟	鸣	铭	名	命	谬	摸	

C4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		摹	蘑	模	膜	磨	摩	魔	抹	末	莫	墨	默	沫	漠	寞
B	陌	谋	牟	某	拇	牡	亩	姆	母	墓	暮	幕	募	慕	木	目
C	睦	牧	穆	拿	哪	呐	钠	那	娜	纳	氖	乃	奶	耐	奈	南
D	男	难	囊	挠	脑	恼	闹	淖	呢	馁	内	嫩	能	妮	霓	倪
E	泥	尼	拟	你	匿	腻	逆	溺	蔫	拈	年	碾	撵	捻	念	娘
F	酿	鸟	尿	捏	聂	孽	啮	镊	镍	涅	您	柠	狞	凝	宁	

C5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		拧	泞	牛	扭	钮	纽	脓	浓	农	弄	奴	努	怒	女	暖
B	虐	疟	挪	懦	糯	诺	哦	欧	鸥	殴	藕	呕	偶	沤	啪	趴
C	爬	帕	怕	琶	拍	排	牌	徘	湃	派	攀	潘	盘	磐	盼	畔
D	判	叛	乓	庞	旁	耪	胖	抛	咆	刨	炮	袍	跑	泡	呸	胚
E	培	裴	赔	陪	配	佩	沛	喷	盆	砰	抨	烹	澎	彭	蓬	棚
F	硼	篷	膨	朋	鹏	捧	碰	坯	砒	霹	批	披	劈	琵	毗	

C6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		啤	脾	疲	皮	匹	痞	僻	屁	譬	篇	偏	片	骗	飘	漂
B	瓢	票	撇	瞥	拼	频	贫	品	聘	乒	坪	苹	萍	平	凭	瓶
C	评	屏	坡	泼	颇	婆	破	魄	迫	粕	剖	扑	铺	仆	莆	葡
D	菩	蒲	埔	朴	圃	普	浦	谱	曝	瀑	期	欺	栖	戚	妻	七
E	凄	漆	柒	沏	其	棋	奇	歧	畦	崎	脐	齐	旗	祈	祁	骑
F	起	岂	乞	企	启	契	砌	器	气	迄	弃	汽	泣	讫	掐	

C7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		恰	洽	牵	扦	钎	铅	千	迁	签	仟	谦	乾	黔	钱	钳
B	前	潜	遣	浅	谴	堑	嵌	欠	歉	枪	呛	腔	羌	墙	蔷	强
C	抢	橇	锹	敲	悄	桥	瞧	乔	侨	巧	鞘	撬	翘	峭	俏	窍
D	切	茄	且	怯	窃	钦	侵	亲	秦	琴	勤	芹	擒	禽	寝	沁
E	青	轻	氢	倾	卿	清	擎	晴	氰	情	顷	请	庆	琼	穷	秋
F	丘	邱	球	求	囚	酋	泅	趋	区	蛆	曲	躯	屈	驱	渠	

C8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		取	娶	龋	趣	去	圈	颧	权	醛	泉	全	痊	拳	犬	券
B	劝	缺	炔	瘸	却	鹊	榷	确	雀	裙	群	然	燃	冉	染	瓤
C	壤	攘	嚷	让	饶	扰	绕	惹	热	壬	仁	人	忍	韧	任	认
D	刃	妊	纫	扔	仍	日	戎	茸	蓉	荣	融	熔	溶	容	绒	冗
E	揉	柔	肉	茹	蠕	儒	孺	如	辱	乳	汝	入	褥	软	阮	蕊
F	瑞	锐	闰	润	若	弱	撒	洒	萨	腮	鳃	塞	赛	三	叁	

C9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		伞	散	桑	嗓	丧	搔	骚	扫	嫂	瑟	色	涩	森	僧	莎
B	砂	杀	刹	沙	纱	傻	啥	煞	筛	晒	珊	苫	杉	山	删	煽
C	衫	闪	陕	擅	赡	膳	善	汕	扇	缮	墒	伤	商	赏	晌	上
D	尚	裳	梢	捎	稍	烧	芍	勺	韶	少	哨	邵	绍	奢	赊	蛇
E	舌	舍	赦	摄	射	慑	涉	社	设	砷	申	呻	伸	身	深	娠
F	绅	神	沈	审	婶	甚	肾	慎	渗	声	生	甥	牲	升	绳	

CA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		省	盛	剩	胜	圣	师	失	狮	施	湿	诗	尸	虱	十	石
B	拾	时	什	食	蚀	实	识	史	矢	使	屎	驶	始	式	示	士
C	世	柿	事	拭	誓	逝	势	是	嗜	噬	适	仕	侍	释	饰	氏
D	市	恃	室	视	试	收	手	首	守	寿	授	售	受	瘦	兽	蔬
E	枢	梳	殊	抒	输	叔	舒	淑	疏	书	赎	孰	熟	薯	暑	曙
F	署	蜀	黍	鼠	属	术	述	树	束	戍	竖	墅	庶	数	漱	

CB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		恕	刷	耍	摔	衰	甩	帅	栓	拴	霜	双	爽	谁	水	睡
B	税	吮	瞬	顺	舜	说	硕	朔	烁	斯	撕	嘶	思	私	司	丝
C	死	肆	寺	嗣	四	伺	似	饲	巳	松	耸	怂	颂	送	宋	讼
D	诵	搜	艘	擞	嗽	苏	酥	俗	素	速	粟	僳	塑	溯	宿	诉
E	肃	酸	蒜	算	虽	隋	随	绥	髓	碎	岁	穗	遂	隧	祟	孙
F	损	笋	蓑	梭	唆	缩	琐	索	锁	所	塌	他	它	她	塔	

CC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		獭	挞	蹋	踏	胎	苔	抬	台	泰	酞	太	态	汰	坍	摊
B	贪	瘫	滩	坛	檀	痰	潭	谭	谈	坦	毯	袒	碳	探	叹	炭
C	汤	塘	搪	堂	棠	膛	唐	糖	倘	躺	淌	趟	烫	掏	涛	滔
D	绦	萄	桃	逃	淘	陶	讨	套	特	藤	腾	疼	誊	梯	剔	踢
E	锑	提	题	蹄	啼	体	替	嚏	惕	涕	剃	屉	天	添	填	田
F	甜	恬	舔	腆	挑	条	迢	眺	跳	贴	铁	帖	厅	听	烃	

CD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		汀	廷	停	亭	庭	挺	艇	通	桐	酮	瞳	同	铜	彤	童
B	桶	捅	筒	统	痛	偷	投	头	透	凸	秃	突	图	徒	途	涂
C	屠	土	吐	兔	湍	团	推	颓	腿	蜕	褪	退	吞	屯	臀	拖
D	托	脱	鸵	陀	驮	驼	椭	妥	拓	唾	挖	哇	蛙	洼	娃	瓦
E	袜	歪	外	豌	弯	湾	玩	顽	丸	烷	完	碗	挽	晚	皖	惋
F	宛	婉	万	腕	汪	王	亡	枉	网	往	旺	望	忘	妄	威	

CE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		巍	微	危	韦	违	桅	围	唯	惟	为	潍	维	苇	萎	委
B	伟	伪	尾	纬	未	蔚	味	畏	胃	喂	魏	位	渭	谓	尉	慰
C	卫	瘟	温	蚊	文	闻	纹	吻	稳	紊	问	嗡	翁	瓮	挝	蜗
D	涡	窝	我	斡	卧	握	沃	巫	呜	钨	乌	污	诬	屋	无	芜
E	梧	吾	吴	毋	武	五	捂	午	舞	伍	侮	坞	戊	雾	晤	物
F	勿	务	悟	误	昔	熙	析	西	硒	矽	晰	嘻	吸	锡	牺	

CF	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		稀	息	希	悉	膝	夕	惜	熄	烯	溪	汐	犀	檄	袭	席
B	习	媳	喜	铣	洗	系	隙	戏	细	瞎	虾	匣	霞	辖	暇	峡
C	侠	狭	下	厦	夏	吓	掀	锨	先	仙	鲜	纤	咸	贤	衔	舷
D	闲	涎	弦	嫌	显	险	现	献	县	腺	馅	羡	宪	陷	限	线
E	相	厢	镶	香	箱	襄	湘	乡	翔	祥	详	想	响	享	项	巷
F	橡	像	向	象	萧	硝	霄	削	哮	嚣	销	消	宵	淆	晓	

D0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		小	孝	校	肖	啸	笑	效	楔	些	歇	蝎	鞋	协	挟	携
B	邪	斜	胁	谐	写	械	卸	蟹	懈	泄	泻	谢	屑	薪	芯	锌
C	欣	辛	新	忻	心	信	衅	星	腥	猩	惺	兴	刑	型	形	邢
D	行	醒	幸	杏	性	姓	兄	凶	胸	匈	汹	雄	熊	休	修	羞
E	朽	嗅	锈	秀	袖	绣	墟	戌	需	虚	嘘	须	徐	许	蓄	酗
F	叙	旭	序	畜	恤	絮	婿	绪	续	轩	喧	宣	悬	旋	玄	

D1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		选	癣	眩	绚	靴	薛	学	穴	雪	血	勋	熏	循	旬	询
B	寻	驯	巡	殉	汛	训	讯	逊	迅	压	押	鸦	鸭	呀	丫	芽
C	牙	蚜	崖	衙	涯	雅	哑	亚	讶	焉	咽	阉	烟	淹	盐	严
D	研	蜒	岩	延	言	颜	阎	炎	沿	奄	掩	眼	衍	演	艳	堰
E	燕	厌	砚	雁	唁	彦	焰	宴	谚	验	殃	央	鸯	秧	杨	扬
F	佯	疡	羊	洋	阳	氧	仰	痒	养	样	漾	邀	腰	妖	瑶	

D2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		摇	尧	遥	窑	谣	姚	咬	舀	药	要	耀	椰	噎	耶	爷
B	野	冶	也	页	掖	业	叶	曳	腋	夜	液	一	壹	医	揖	铱
C	依	伊	衣	颐	夷	遗	移	仪	胰	疑	沂	宜	姨	彝	椅	蚁
D	倚	已	乙	矣	以	艺	抑	易	邑	屹	亿	役	臆	逸	肄	疫
E	亦	裔	意	毅	忆	义	益	溢	诣	议	谊	译	异	翼	翌	绎
F	茵	荫	因	殷	音	阴	姻	吟	银	淫	寅	饮	尹	引	隐	

D3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		印	英	樱	婴	鹰	应	缨	莹	萤	营	荧	蝇	迎	赢	盈
B	影	颖	硬	映	哟	拥	佣	臃	痈	庸	雍	踊	蛹	咏	泳	涌
C	永	恿	勇	用	幽	优	悠	忧	尤	由	邮	铀	犹	油	游	酉
D	有	友	右	佑	釉	诱	又	幼	迂	淤	于	盂	榆	虞	愚	舆
E	余	俞	逾	鱼	愉	渝	渔	隅	予	娱	雨	与	屿	禹	宇	语
F	羽	玉	域	芋	郁	吁	遇	喻	峪	御	愈	欲	狱	育	誉	

D4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		浴	寓	裕	预	豫	驭	鸳	渊	冤	元	垣	袁	原	援	辕
B	园	员	圆	猿	源	缘	远	苑	愿	怨	院	曰	约	越	跃	钥
C	岳	粤	月	悦	阅	耘	云	郧	匀	陨	允	运	蕴	酝	晕	韵
D	孕	匝	砸	杂	栽	哉	灾	宰	载	再	在	咱	攒	暂	赞	赃
E	脏	葬	遭	糟	凿	藻	枣	早	澡	蚤	躁	噪	造	皂	灶	燥
F	责	择	则	泽	贼	怎	增	憎	曾	赠	扎	喳	渣	札	轧	

D5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		铡	闸	眨	栅	榨	咋	乍	炸	诈	摘	斋	宅	窄	债	寨
B	瞻	毡	詹	粘	沾	盏	斩	辗	崭	展	蘸	栈	占	战	站	湛
C	绽	樟	章	彰	漳	张	掌	涨	杖	丈	帐	账	仗	胀	瘴	障
D	招	昭	找	沼	赵	照	罩	兆	肇	召	遮	折	哲	蛰	辙	者
E	锗	蔗	这	浙	珍	斟	真	甄	砧	臻	贞	针	侦	枕	疹	诊
F	震	振	镇	阵	蒸	挣	睁	征	狰	争	怔	整	拯	正	政	

D6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		帧	症	郑	证	芝	枝	支	吱	蜘	知	肢	脂	汁	之	织
B	职	直	植	殖	执	值	侄	址	指	止	趾	只	旨	纸	志	挚
C	掷	至	致	置	帜	峙	制	智	秩	稚	质	炙	痔	滞	治	窒
D	中	盅	忠	钟	衷	终	种	肿	重	仲	众	舟	周	州	洲	诌
E	粥	轴	肘	帚	咒	皱	宙	昼	骤	珠	株	蛛	朱	猪	诸	诛
F	逐	竹	烛	煮	拄	瞩	嘱	主	著	柱	助	蛀	贮	铸	筑	

D7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		住	注	祝	驻	抓	爪	拽	专	砖	转	撰	赚	篆	桩	庄
B	装	妆	撞	壮	状	椎	锥	追	赘	坠	缀	谆	准	捉	拙	卓
C	桌	琢	茁	酌	啄	着	灼	浊	兹	咨	资	姿	滋	淄	孜	紫
D	仔	籽	滓	子	自	渍	字	鬃	棕	踪	宗	综	总	纵	邹	走
E	奏	揍	租	足	卒	族	祖	诅	阻	组	钻	纂	嘴	醉	最	罪
F	尊	遵	昨	左	佐	柞	做	作	坐	座						

D8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		亍	丌	兀	丐	廿	卅	丕	亘	丞	鬲	孬	噩	丨	禺	丿
B	匕	毛	夭	爻	卮	氐	囟	胤	馗	毓	睾	鼗	丶	亟	鼐	乜
C	乩	亓	芈	孛	啬	嘏	仄	厍	厝	厣	厥	厮	靥	匚	叵	
D	匦	匮	匾	赜	卦	卣	刂	刈	刎	刭	刳	刿	剀	剌	剞	剡
E	剜	蒯	剽	劂	劁	劐	剖	冂	罔	亻	仃	仉	仂	仨	仡	仫
F	仞	伛	仳	伢	佤	仵	伥	伧	伉	伫	佞	佧	攸	佚	佝	

D9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		佟	佗	伲	伽	佶	佴	侑	侉	侃	侏	佾	佻	侪	佼	侬
B	侔	俦	俨	俪	俅	俚	俣	俜	俑	俟	俸	倩	偌	俳	倬	倏
C	倮	倭	俾	倜	倌	倥	倨	偾	偃	偕	偈	偎	偬	偻	傥	傧
D	傩	傺	僖	儆	僭	僬	僦	僮	儇	儋	仝	氽	佘	佥	俎	龠
E	汆	籴	兮	巽	黉	馘	冁	夔	勹	匍	訇	匐	凫	夙	兕	亠
F	兖	亳	衮	袤	亵	脔	裒	禀	嬴	蠃	羸	冫	冱	冽	冼	

DA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		凇	冖	冢	冥	讠	讦	讧	讪	讴	讵	讷	诂	诃	诋	诏
B	诎	诒	诓	诔	诖	诘	诙	诜	诟	诠	诤	诨	诩	诮	诰	诳
C	诶	诹	诼	诿	谀	谂	谄	谇	谌	谏	谑	谒	谔	谕	谖	谙
D	谛	谘	谝	谟	谠	谡	谥	谧	谪	谫	谮	谯	谲	谳	谵	谶
E	卩	卺	阝	阢	阡	阱	阪	阽	阼	陂	陉	陔	陟	陧	陬	陲
F	陴	隈	隍	隗	隰	邗	邛	邝	邙	邬	邡	邴	邳	邶	邺	

DB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		邸	邰	郏	郅	邾	郐	郄	郇	郓	郦	郢	郜	郗	郛	郫
B	郯	郾	鄄	鄢	鄞	鄣	鄱	鄯	鄹	酃	酆	刍	奂	劢	劬	劭
C	劾	哿	勐	勖	勰	叟	燮	矍	廴	凵	凼	鬯	厶	弁	畚	巯
D	坌	垩	垡	塾	墼	壅	壑	圩	圬	圪	圳	圹	圮	圯	坜	圻
E	坂	坩	垅	坫	垆	坼	坻	坨	坭	坶	坳	垭	垤	垌	垲	埏
F	垧	垴	垓	垠	埕	埘	埚	埙	埒	垸	埴	埯	埸	埤	埝	

DC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		堋	堍	埽	埭	堀	堞	堙	塄	堠	塥	塬	墁	墉	墚	墀
B	馨	鼙	懿	艹	艽	艿	芏	芊	芨	芄	芎	芑	芗	芙	芫	芸
C	芾	芰	苈	苊	苣	芘	芷	芮	苋	苌	苁	芩	芴	芡	芪	芟
D	苄	苎	芤	苡	茉	苷	苤	茏	茇	苜	苴	苒	苘	茌	苻	苓
E	茑	茚	茆	茔	茕	苠	苕	茜	荑	荛	荜	茈	莒	茼	茴	茱
F	莛	荞	茯	荏	荇	荃	荟	荀	茗	荠	茭	茺	茳	荦	荥	

DD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		荨	茛	荩	荬	荪	荭	荮	莰	荸	莳	莴	莠	莪	莓	莜
B	莅	荼	莶	莩	荽	莸	荻	莘	莞	莨	莺	莼	菁	萁	菥	菘
C	堇	萘	萋	菝	菽	菖	萜	萸	萑	萆	菔	菟	萏	萃	菸	菹
D	菪	菅	菀	萦	菰	菡	葜	葑	葚	葙	葳	蒇	蒈	葺	蒉	葸
E	萼	葆	葩	葶	蒌	蒎	萱	葭	蓁	蓍	蓐	蓦	蒽	蓓	蓊	蒿
F	蒺	蓠	蒡	蒹	蒴	蒗	蓥	蓣	蔌	甍	蔸	蓰	蔹	蔟	蔺	

DE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		蕖	蔻	蓿	蓼	蕙	蕈	蕨	蕤	蕞	蕺	瞢	蕃	蕲	蕻	薤
B	薨	薇	薏	蕹	薮	薜	薅	薹	薷	薰	藓	藁	藜	藿	蘧	蘅
C	蘩	蘖	蘼	廾	弈	夼	奁	耷	奕	奚	奘	匏	尢	尥	尬	尴
D	扌	扪	抟	抻	拊	拚	拗	拮	挢	拶	挹	捋	捃	掭	揶	捱
E	捺	掎	掴	捭	掬	掊	捩	掮	掼	揲	揸	揠	揿	揄	揞	揎
F	摒	揆	掾	摅	摁	搋	搛	搠	搌	搦	搡	摞	撄	摭	撖	

DF	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		摺	撷	撸	撙	撺	擀	擐	擗	擤	擢	攉	攥	攮	弋	忒
B	甙	弑	卟	叱	叽	叩	叨	叻	吒	吖	吆	呋	呒	呓	呔	呖
C	呃	吡	呗	呙	吣	吲	咂	咔	呷	呱	呤	咚	咛	咄	呶	呦
D	咝	哐	咭	哂	咴	哒	咧	咦	哓	哔	呲	咣	哕	咻	咿	哌
E	哙	哚	哜	咩	咪	咤	哝	哏	哞	唛	哧	唠	哽	唔	哳	唢
F	唣	唏	唑	唧	唪	啧	喏	喵	啉	啭	啁	啕	唿	啐	唼	

E0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		唷	啖	啵	啶	啷	唳	唰	啜	喋	嗒	喃	喱	喹	喈	喁
B	喟	啾	嗖	喑	啻	嗟	喽	喾	喔	喙	嗪	嗷	嗉	嘟	嗑	嗫
C	嗬	嗔	嗦	嗝	嗄	嗯	嗥	嗲	嗳	嗌	嗍	嗨	嗵	嗤	辔	嘞
D	嘈	嘌	嘁	嘤	嘣	嗾	嘀	嘧	嘭	噘	嘹	噗	嘬	噍	噢	噙
E	噜	噌	噔	嚆	噤	噱	噫	噻	噼	嚅	嚓	嚯	囔	囗	囝	囡
F	囵	囫	囹	囿	圄	圊	圉	圜	帏	帙	帔	帑	帱	帻	帼	

E1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		帷	幄	幔	幛	幞	幡	岌	屺	岍	岐	岖	岈	岘	岙	岑
B	岚	岜	岵	岢	岽	岬	岫	岱	岣	峁	岷	峄	峒	峤	峋	峥
C	崂	崃	崧	崦	崮	崤	崞	崆	崛	嵘	崾	崴	崽	嵬	嵛	嵯
D	嵝	嵫	嵋	嵊	嵩	嵴	嶂	嶙	嶝	豳	嶷	巅	彳	彷	徂	徇
E	徉	後	徕	徙	徜	徨	徭	徵	徼	衢	彡	犭	犰	犴	犷	犸
F	狃	狁	狎	狍	狒	狨	狯	狩	狲	狴	狷	猁	狳	猃	狺	

E2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		狻	猗	猓	猡	猊	猞	猝	猕	猢	猹	猥	猬	猸	猱	獐
B	獍	獗	獠	獬	獯	獾	舛	夥	飧	夤	夂	饣	饧	饨	饩	饪
C	饫	饬	饴	饷	饽	馀	馄	馇	馊	馍	馐	馑	馓	馔	馕	庀
D	庑	庋	庖	庥	庠	庹	庵	庾	庳	赓	廒	廑	廛	廨	廪	膺
E	忄	忉	忖	忏	怃	忮	怄	忡	忤	忾	怅	怆	忪	忭	忸	怙
F	怵	怦	怛	怏	怍	怩	怫	怊	怿	怡	恸	恹	恻	恺	恂	

E3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		恪	恽	悖	悚	悭	悝	悃	悒	悌	悛	惬	悻	悱	惝	惘
B	惆	惚	悴	愠	愦	愕	愣	惴	愀	愎	愫	慊	慵	憬	憔	憧
C	憷	懔	懵	忝	隳	闩	闫	闱	闳	闵	闶	闼	闾	阃	阄	阆
D	阈	阊	阋	阌	阍	阏	阒	阕	阖	阗	阙	阚	丬	爿	戕	氵
E	汔	汜	汊	沣	沅	沐	沔	沌	汨	汩	汴	汶	沆	沩	泐	泔
F	沭	泷	泸	泱	泗	沲	泠	泖	泺	泫	泮	沱	泓	泯	泾	

E4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		洹	洧	洌	浃	浈	洇	洄	洙	洎	洫	浍	洮	洵	洚	浏
B	浒	浔	洳	涑	浯	涞	涠	浞	涓	涔	浜	浠	浼	浣	渚	淇
C	淅	淞	渎	涿	淠	渑	淦	淝	淙	渖	涫	渌	涮	渫	湮	湎
D	湫	溲	湟	溆	湓	湔	渲	渥	湄	滟	溱	溘	滠	漭	滢	溥
E	溧	溽	溻	溷	滗	溴	滏	溏	滂	溟	潢	潆	潇	漤	漕	滹
F	漯	漶	潋	潴	漪	漉	漩	澉	澍	澌	潸	潲	潼	潦	濑	

E5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		濉	澧	澹	澶	濂	濡	濮	濞	濠	濯	瀚	瀣	瀛	瀹	瀵
B	灏	灞	宀	宄	宕	宓	宥	宸	甯	骞	搴	寤	寮	褰	寰	蹇
C	謇	辶	迓	迕	迥	迮	迤	迩	迦	迳	迨	逅	逄	逋	逦	逑
D	逍	逖	逡	逵	逶	逭	逯	遄	遑	遒	遐	遨	遘	遢	遛	暹
E	遴	遽	邂	邈	邃	邋	彐	彗	彖	彘	尻	咫	屐	屙	孱	屣
F	屦	羼	弪	弩	弭	艴	弼	鬻	屮	妁	妃	妍	妩	妪	妣	

E6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		妗	姊	妫	妞	妤	姒	妲	妯	姗	妾	娅	娆	姝	娈	姣
B	姘	姹	娌	娉	娲	娴	娑	娣	娓	婀	婧	婊	婕	娼	婢	婵
C	胬	媪	媛	婷	婺	媾	嫫	媲	嫒	嫔	媸	嫠	嫣	嫱	嫖	嫦
D	嫘	嫜	嬉	嬗	嬖	嬲	嬷	孀	尕	尜	孚	孥	孳	孑	孓	孢
E	驵	驷	驸	驺	驿	驽	骀	骁	骅	骈	骊	骐	骒	骓	骖	骘
F	骛	骜	骝	骟	骠	骢	骣	骥	骧	纟	纡	纣	纥	纨	纩	

E7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		纭	纰	纾	绀	绁	绂	绉	绋	绌	绐	绔	绗	绛	绠	绡
B	绨	绫	绮	绯	绱	绲	缍	绶	绺	绻	绾	缁	缂	缃	缇	缈
C	缋	缌	缏	缑	缒	缗	缙	缜	缛	缟	缡	缢	缣	缤	缥	缦
D	缧	缪	缫	缬	缭	缯	缰	缱	缲	缳	缵	幺	畿	巛	甾	邕
E	玎	玑	玮	玢	玟	珏	珂	珑	玷	玳	珀	珉	珈	珥	珙	顼
F	琊	珩	珧	珞	玺	珲	琏	琪	瑛	琦	琥	琨	琰	琮	琬	

E8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		琛	琚	瑁	瑜	瑗	瑕	瑙	瑷	瑭	瑾	璜	璎	璀	璁	璇
B	璋	璞	璨	璩	璐	璧	瓒	璺	韪	韫	韬	杌	杓	杞	杈	杩
C	枥	枇	杪	杳	枘	枧	杵	枨	枞	枭	枋	杷	杼	柰	栉	柘
D	栊	柩	枰	栌	柙	枵	柚	枳	柝	栀	柃	枸	柢	栎	柁	柽
E	栲	栳	桠	桡	桎	桢	桄	桤	梃	栝	桕	桦	桁	桧	桀	栾
F	桊	桉	栩	梵	梏	桴	桷	梓	桫	棂	楮	棼	椟	椠	棹	

E9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		椤	棰	椋	椁	楗	棣	椐	楱	椹	楠	楂	楝	榄	楫	榀
B	榘	楸	椴	槌	榇	榈	槎	榉	楦	楣	楹	榛	榧	榻	榫	榭
C	槔	榱	槁	槊	槟	榕	槠	榍	槿	樯	槭	樗	樘	橥	槲	橄
D	樾	檠	橐	橛	樵	檎	橹	樽	樨	橘	橼	檑	檐	檩	檗	檫
E	猷	獒	殁	殂	殇	殄	殒	殓	殍	殚	殛	殡	殪	轫	轭	轱
F	轲	轳	轵	轶	轸	轷	轹	轺	轼	轾	辁	辂	辄	辇	辋	

EA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		辍	辎	辏	辘	辚	軎	戋	戗	戛	戟	戢	戡	戥	戤	戬
B	臧	瓯	瓴	瓿	甏	甑	甓	攴	旮	旯	旰	昊	昙	杲	昃	昕
C	昀	炅	曷	昝	昴	昱	昶	昵	耆	晟	晔	晁	晏	晖	晡	晗
D	晷	暄	暌	暧	暝	暾	曛	曜	曦	曩	贲	贳	贶	贻	贽	赀
E	赅	赆	赈	赉	赇	赍	赕	赙	觇	觊	觋	觌	觎	觏	觐	觑
F	牮	犟	牝	牦	牯	牾	牿	犄	犋	犍	犏	犒	挈	挲	掰	

EB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		搿	擘	耄	毪	毳	毽	毵	毹	氅	氇	氆	氍	氕	氘	氙
B	氚	氡	氩	氤	氪	氲	攵	敕	敫	牍	牒	牖	爰	虢	刖	肟
C	肜	肓	肼	朊	肽	肱	肫	肭	肴	肷	胧	胨	胩	胪	胛	胂
D	胄	胙	胍	胗	朐	胝	胫	胱	胴	胭	脍	脎	胲	胼	朕	脒
E	豚	脶	脞	脬	脘	脲	腈	腌	腓	腴	腙	腚	腱	腠	腩	腼
F	腽	腭	腧	塍	媵	膈	膂	膑	滕	膣	膪	臌	朦	臊	膻	

EC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		臁	膦	欤	欷	欹	歃	歆	歙	飑	飒	飓	飕	飙	飚	殳
B	彀	毂	觳	斐	齑	斓	於	旆	旄	旃	旌	旎	旒	旖	炀	炜
C	炖	炝	炻	烀	炷	炫	炱	烨	烊	焐	焓	焖	焯	焱	煳	煜
D	煨	煅	煲	煊	煸	煺	熘	熳	熵	熨	熠	燠	燔	燧	燹	爝
E	爨	灬	焘	煦	熹	戾	戽	扃	扈	扉	礻	祀	祆	祉	祛	祜
F	祓	祚	祢	祗	祠	祯	祧	祺	禅	禊	禚	禧	禳	忑	忐	

ED	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		怼	恝	恚	恧	恁	恙	恣	悫	愆	愍	慝	憩	憝	懋	懑
B	戆	肀	聿	沓	泶	淼	矶	矸	砀	砉	砗	砘	砑	斫	砭	砜
C	砝	砹	砺	砻	砟	砼	砥	砬	砣	砩	硎	硭	硖	硗	砦	硐
D	硇	硌	硪	碛	碓	碚	碇	碜	碡	碣	碲	碹	碥	磔	磙	磉
E	磬	磲	礅	磴	礓	礤	礞	礴	龛	黹	黻	黼	盱	眄	眍	盹
F	眇	眈	眚	眢	眙	眭	眦	眵	眸	睐	睑	睇	睃	睚	睨	

EE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		睢	睥	睿	瞍	睽	瞀	瞌	瞑	瞟	瞠	瞰	瞵	瞽	町	畀
B	畎	畋	畈	畛	畲	畹	疃	罘	罡	罟	詈	罨	罴	罱	罹	羁
C	罾	盍	盥	蠲	钅	钆	钇	钋	钊	钌	钍	钏	钐	钔	钗	钕
D	钚	钛	钜	钣	钤	钫	钪	钭	钬	钯	钰	钲	钴	钶	钷	钸
E	钹	钺	钼	钽	钿	铄	铈	铉	铊	铋	铌	铍	铎	铐	铑	铒
F	铕	铖	铗	铙	铘	铛	铞	铟	铠	铢	铤	铥	铧	铨	铪	

EF	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		铩	铫	铮	铯	铳	铴	铵	铷	铹	铼	铽	铿	锃	锂	锆
B	锇	锉	锊	锍	锎	锏	锒	锓	锔	锕	锖	锘	锛	锝	锞	锟
C	锢	锪	锫	锩	锬	锱	锲	锴	锶	锷	锸	锼	锾	锿	镂	锵
D	镄	镅	镆	镉	镌	镎	镏	镒	镓	镔	镖	镗	镘	镙	镛	镞
E	镟	镝	镡	镢	镤	镥	镦	镧	镨	镩	镪	镫	镬	镯	镱	镲
F	镳	锺	矧	矬	雉	秕	秭	秣	秫	稆	嵇	稃	稂	稞	稔	

F0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		稹	稷	穑	黏	馥	穰	皈	皎	皓	皙	皤	瓞	瓠	甬	鸠
B	鸢	鸨	鸩	鸪	鸫	鸬	鸲	鸱	鸶	鸸	鸷	鸹	鸺	鸾	鹁	鹂
C	鹄	鹆	鹇	鹈	鹉	鹋	鹌	鹎	鹑	鹕	鹗	鹚	鹛	鹜	鹞	鹣
D	鹦	鹧	鹨	鹩	鹪	鹫	鹬	鹱	鹭	鹳	疒	疔	疖	疠	疝	疬
E	疣	疳	疴	疸	痄	疱	疰	痃	痂	痖	痍	痣	痨	痦	痤	痫
F	痧	瘃	痱	痼	痿	瘐	瘀	瘅	瘌	瘗	瘊	瘥	瘘	瘕	瘙	

F1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		瘛	瘼	瘢	瘠	癀	瘭	瘰	瘿	瘵	癃	瘾	瘳	癍	癞	癔
B	癜	癖	癫	癯	翊	竦	穸	穹	窀	窆	窈	窕	窦	窠	窬	窨
C	窭	窳	衤	衩	衲	衽	衿	袂	袢	裆	袷	袼	裉	裢	裎	裣
D	裥	裱	褚	裼	裨	裾	裰	褡	褙	褓	褛	褊	褴	褫	褶	襁
E	襦	襻	疋	胥	皲	皴	矜	耒	耔	耖	耜	耠	耢	耥	耦	耧
F	耩	耨	耱	耋	耵	聃	聆	聍	聒	聩	聱	覃	顸	颀	颃	

F2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		颉	颌	颍	颏	颔	颚	颛	颞	颟	颡	颢	颥	颦	虍	虔
B	虬	虮	虿	虺	虼	虻	蚨	蚍	蚋	蚬	蚝	蚧	蚣	蚪	蚓	蚩
C	蚶	蛄	蚵	蛎	蚰	蚺	蚱	蚯	蛉	蛏	蚴	蛩	蛱	蛲	蛭	蛳
D	蛐	蜓	蛞	蛴	蛟	蛘	蛑	蜃	蜇	蛸	蜈	蜊	蜍	蜉	蜣	蜻
E	蜞	蜥	蜮	蜚	蜾	蝈	蜴	蜱	蜩	蜷	蜿	螂	蜢	蝽	蝾	蝻
F	蝠	蝰	蝌	蝮	螋	蝓	蝣	蝼	蝤	蝙	蝥	螓	螯	螨	蟒	

F3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		蟆	螈	螅	螭	螗	螃	螫	蟥	螬	螵	螳	蟋	蟓	螽	蟑
B	蟀	蟊	蟛	蟪	蟠	蟮	蠖	蠓	蟾	蠊	蠛	蠡	蠹	蠼	缶	罂
C	罄	罅	舐	竺	竽	笈	笃	笄	笕	笊	笫	笏	筇	笸	笪	笙
D	笮	笱	笠	笥	笤	笳	笾	笞	筘	筚	筅	筵	筌	筝	筠	筮
E	筻	筢	筲	筱	箐	箦	箧	箸	箬	箝	箨	箅	箪	箜	箢	箫
F	箴	篑	篁	篌	篝	篚	篥	篦	篪	簌	篾	篼	簏	簖	簋	

F4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		簟	簪	簦	簸	籁	籀	臾	舁	舂	舄	臬	衄	舡	舢	舣
B	舭	舯	舨	舫	舸	舻	舳	舴	舾	艄	艉	艋	艏	艚	艟	艨
C	衾	袅	袈	裘	裟	襞	羝	羟	羧	羯	羰	羲	籼	敉	粑	粝
D	粜	粞	粢	粲	粼	粽	糁	糇	糌	糍	糈	糅	糗	糨	艮	暨
E	羿	翎	翕	翥	翡	翦	翩	翮	翳	糸	絷	綦	綮	繇	纛	麸
F	麴	赳	趄	趔	趑	趱	赧	赭	豇	豉	酊	酐	酎	酏	酤	

F5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		酢	酡	酰	酩	酯	酽	酾	酲	酴	酹	醌	醅	醐	醍	醑
B	醢	醣	醪	醭	醮	醯	醵	醴	醺	豕	鹾	趸	跫	踅	蹙	蹩
C	趵	趿	趼	趺	跄	跖	跗	跚	跞	跎	跏	跛	跆	跬	跷	跸
D	跣	跹	跻	跤	踉	跽	踔	踝	踟	踬	踮	踣	踯	踺	蹀	踹
E	踵	踽	踱	蹉	蹁	蹂	蹑	蹒	蹊	蹰	蹶	蹼	蹯	蹴	躅	躏
F	躔	躐	躜	躞	豸	貂	貊	貅	貘	貔	斛	觖	觞	觚	觜	

F6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		觥	觫	觯	訾	謦	靓	雩	雳	雯	霆	霁	霈	霏	霎	霪
B	霭	霰	霾	龀	龃	龅	龆	龇	龈	龉	龊	龌	黾	鼋	鼍	隹
C	隼	隽	雎	雒	瞿	雠	銎	銮	鋈	錾	鍪	鏊	鎏	鐾	鑫	鱿
D	鲂	鲅	鲆	鲇	鲈	稣	鲋	鲎	鲐	鲑	鲒	鲔	鲕	鲚	鲛	鲞
E	鲟	鲠	鲡	鲢	鲣	鲥	鲦	鲧	鲨	鲩	鲫	鲭	鲮	鲰	鲱	鲲
F	鲳	鲴	鲵	鲶	鲷	鲺	鲻	鲼	鲽	鳄	鳅	鳆	鳇	鳊	鳋	

F7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		鳌	鳍	鳎	鳏	鳐	鳓	鳔	鳕	鳗	鳘	鳙	鳜	鳝	鳟	鳢
B	靼	鞅	鞑	鞒	鞔	鞯	鞫	鞣	鞲	鞴	骱	骰	骷	鹘	骶	骺
C	骼	髁	髀	髅	髂	髋	髌	髑	魅	魃	魇	魉	魈	魍	魑	飨
D	餍	餮	饕	饔	髟	髡	髦	髯	髫	髻	髭	髹	鬈	鬏	鬓	鬟
E	鬣	麽	麾	縻	麂	麇	麈	麋	麒	鏖	麝	麟	黛	黜	黝	黠
F	黟	黢	黩	黧	黥	黪	黯	鼢	鼬	鼯	鼹	鼷	鼽	鼾	齄	

注: 繁体版的字型与字码表请参考繁体版应用手册: RA8803_8822_AP_v26_Chi.pdf