SSD1332 OLED 點亮細節

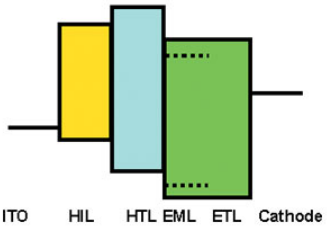
LCM 產品設計課

目錄

一、 OLED 的显示原理

OLED 组件系由 n 型有机材料、p 型有机材料、阴极金属及阳极金属所构成。电子(空穴)由阴极(阳极)注入，经过 n 型(p 型)有机材料传导至发光层(一般为 n 型材料)，经由再结合而放光。一般而言，OLED 组件制作的玻璃基板上先溅镀 ITO 作为阳极，再以真空热蒸镀之方式，依序镀上 p 型和 n 型有机材料，及低功函数之金属阴极。由于有机材料易与水气或氧气作用，产生暗点(Dark spot)而使组件不发亮。因此此组件于真空镀膜完毕后，必须于无水气及氧气之环境下进行封装工艺。

在阴极金属与阳极 ITO 之间，目前广为应用的组件结构一般而言可分为 5 层。如图一所示，从靠近 ITO 侧依序为：空穴注入层、空穴传输层、发光层、电子传输层、电子注入层。OLED 组件系由两层有机材料所构成，分别为空穴传输层及电子传输层。其中空穴传输层为 p 型之有机材料，其特性为具有较高之空穴迁移率，且其最高占据之分子轨域(Highest occupied molecule orbital，HOMO)与 ITO 较接近，可使空穴由 ITO 注入有机层之能障降低。



[图一：OLED 结构图]

而至于电子传输层，系为 n 型之有机材料，其特性为具有较高之电子迁移率，当电子由电子传输层至空穴电子传输层接口时，由于电子传输层之最低非占据分子轨域(Lowest unoccupied molecule orbital，LUMO)较空穴传输层之 LUMO 高出甚多，电子不易跨越此一能障进入空穴传输层，遂被阻挡于此接口。此时空穴由空穴传输层传至接口附近与电子再结合而产生激子(Exciton)，而 Exciton 会以放光及非放光之形式进行能量释放。以一般萤光(Fluorescence)材料系统而言，由选择率(Selection rule)之计算仅得 25%之电子空穴对系以放光之形式做再结合，其余 75%之能量则以放热之形式散逸。近年来，正积极被开发磷光(Phosphorescence)材料成为新一代的 OLED 材料[2]，此类材料可打破选择率之限制，以提高内部量子效率至接近 100%。

在两层组件中，n 型有机材料－即电子传输层－亦同时被当作发光层，其发光波长系由 HOMO 及 LUMO 之能量差所决定。然而，好的电子传输层－即电子迁移率高之材料－并不一定为放光效率佳之材料，因此目前一般之做法，系将高萤光度的有机色料，掺杂(Doped)于电子传输层中靠近空穴传输层之部分，又称为发光层[3]，其体积比约为 1%至 3%。掺杂技术开发系用于增强原材料之萤光量子吸收率的重点技术，一般所选择的材料为萤光量子吸收率高的染料(Dye)。由于有机染料之发展源自于 1970 至 1980 年代染料雷射，因此材料系统齐全，发光波长可涵盖整个可见光区。在 OLED 组件中掺杂之有机染料，能带较差，一般而言小于其宿主(Host)之能带，以利 exciton 由 host 至掺杂物(Dopant)之能量转移。然而，由于 dopant 能带较小，而在电性上系扮演陷阱(trap)之角色，因此，掺杂层太厚将会使驱动电压上升；但若太薄，则能量由 host 转移至 dopant 之比例将会变差，因此，此层厚

度必须最佳化。

阴极之金属材料，传统上系使用低功函数之金属材料(或合金)，如镁合金，以利电子由阴极注入至电子传输层，此外一种普遍之做法，系导入一层电子注入层，其构成为一极薄之低功函数金属卤化物或氧化物，如 LiF 或 Li2O，此可大幅降低阴极与电子传输层之能障[4]，降低驱动电压。

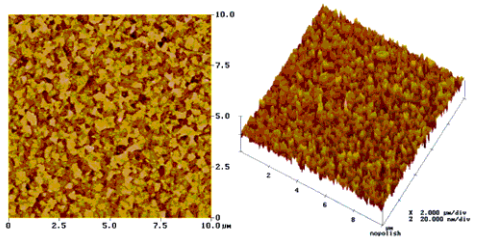
由于空穴传输层材料之 HOMO 值与 ITO 仍有差距，此外 ITO 阳极在长时间操作后，有可能释放出氧气，并破坏有机层产生暗点。故在 ITO 及空穴传输层之间，插入一空穴注入层，其 HOMO 值恰介于 ITO 及空穴传输层之间，有利于空穴注入 OLED 组件，且其薄膜之特性可阻隔 ITO 中之氧气进入 OLED 组件，以延长组件寿命[5]。

二、 OLED 相关关键工艺

氧化铟锡(ITO)基板前处理

(1)ITO 表面平整度

ITO 目前已广泛应用在商业化的显示器面板制造，其具有高透射率、低电阻率及高功函数等优点。一般而言，利用射频溅镀法(RF sputtering)所制造的 ITO，易受工艺控制因素不良而导致表面不平整，进而产生表面的尖端物质或突起物。另外高温锻烧及再结晶的过程亦会产生表面约 10 ~ 30nm 的突起层。这些不平整层的细粒之间所形成的路径会提供空穴直接射向阴极的机会，而这些错综复杂的路径会使漏电流增加。一般有三个方法可以解决这表面层的影响：一是增加空穴注入层及空穴传输层的厚度以降低漏电流，此方法多用于 PLED 及空穴层较厚的 OLED(~200nm)。二是将 ITO 玻璃再处理，使表面光滑。三是使用其它镀膜方法使表面平整度更好(如图二所示)。



[图二：ITO 表面之原子力显微镜照片]

(2) ITO 功函数的增加

当空穴由 ITO 注入 HIL 时，过大的位能差会产生萧基能障，使得空穴不易注入，因此如何降低 ITO / HIL 接口的位能差则成为 ITO 前处理的重点。一般我们使用 O2-Plasma 方式增加 ITO 中氧原子的饱和度，以达到增加功函数之目的。ITO 经 O2-Plasma 处理后功函数可由原先之 4.8eV 提升至 5.2eV，与 HIL 的功函数已非常接近。

加入辅助电极

由于 OLED 为电流驱动组件，当外部线路过长或过细时，于外部电路将会造成严重之电压梯度(voltage drop)，使真正落于 OLED 组件之电压下降，导致面板发光强度减少。由于 ITO 电阻过大(10 ohm / square)，易造成不必要之外部功率消耗，增加一辅助电极以降低电压梯度成了增加发光效率、减少驱动电压的快捷方式。铬(Cr：Chromium)金属是最常被用作辅助电极的材料，它具有对环境因子稳定性佳及对蚀刻液有较大的选择性等优点。然

而它的电阻值在膜层为 100nm 时为 2 ohm / square，在某些应用时仍属过大，因此在相同厚度时拥有较低电阻值的铝(Al：Aluminum)金属(0.2 ohm / square)则成为辅助电极另一较佳选择。但是，铝金属的高活性也使其有信赖性方面之问题；因此，多叠层之辅助金属则被提出，如：Cr / Al / Cr 或 Mo / Al / Mo，然而此类工艺增加复杂度及成本，故辅助电极材料的选择成为 OLED 工艺中的重点之一。

阴极工艺

在高解析的 OLED 面板中，将细微的阴极与阴极之间隔离，一般所用的方法为蘑菇构型法(Mushroom structure approach)，此工艺类似印刷技术的负光阻显影技术。在负光阻显影过程中，许多工艺上的变异因子会影响阴极的品质及良率。例如，体电阻、介电常数、高分辨率、高 Tg、低临界维度(CD)的损失以及与 ITO 或其它有机层适当的黏着接口等。

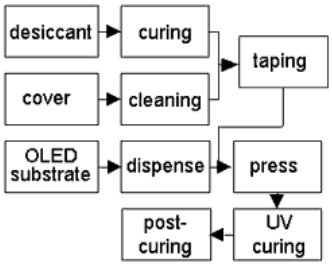
封装

(1)吸水材料

一般 OLED 的生命周期易受周围水气与氧气所影响而降低。水气来源主要分为两种：一是经由外在环境渗透进入组件内，另一种是在 OLED 工艺中被每一层物质所吸收的水气。为了减少水气进入组件或排除由工艺中所吸附的水气，一般最常使用的物质为吸水材(Desiccant)。Desiccant 可以利用化学吸附或物理吸附的方式捕捉自由移动的水分子，以达到去除组件内水气的目的。

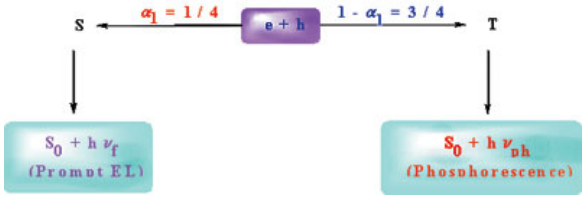
(2)工艺及设备开发

封装工艺之流程如图四所示，为了将 Desiccant 置于盖板及顺利将盖板与基板黏合，需在真空环境或将腔体充入不活泼气体下进行，例如氮气。值得注意的是，如何让盖板与基板这两部分工艺衔接更有效率、减少封装工艺成本以及减少封装时间以达最佳量产速率，已俨然成为封装工艺及设备技术发展的 3 大主要目标。



[图三：传统之 OLED 封装流程]

就原理言，OLED 系与 LED 类似，然而驱动电压相差甚多。LED 之工作电压约为 3 伏特上下，而一般被动矩阵 OLED 之工作电压则为 8 伏特以上。此外，LED 之磊晶厚度为 mm 等级，而 OLED 组件之厚度仅为数百 nm。因此，OLED 组件之操作电场强度甚大于 LED，其主要原因即为 OLED 之低导电度，由于有机材料之电子及空穴迁移律，一般而言，仅为磊晶半导体之百万分之一倍，加上能隙较大，导致载子浓度甚低。因此，如何降低 OLED 组件之驱动电压，以提升电光转换效率遂成为一重要之研究方向。



[图四：OLED 电子空穴对之能量转移]

除了降低驱动电压以外，前文有叙述以目前之萤光材料系统，exciton 由于选择率的限制，只能经由一重项(singlet state)放光，亦即仅有 25%之电能可转换成光能，导致最大之内部量子效率为 25%。有鉴于此，目前有许多研究正朝向磷光材料系统之开发，以提升内部量子效率，由于磷光发光材料其原理为使用如铱(Iridium)、铕(Europium)及铂(Platinum)等过渡重金属元素为基底之有机材料，藉由其半满或空轨域，可进行能量转移至周遭之有机配位基，使 exciton 经由内部系统转换(internal system crossing)至三重项 triplet state 发光，因此其内部量子效率可大幅提升(如图四所示)。其问题点在于：由于相较于萤光发光材料而言，磷光发光之载子生命期为较长，因此当注入电流增加时较易发生 exciton 堆积，使发光效率下降。此外，由于能量转移之机制有异于一般萤光材料，OLED 组件之设计是一重要之课题。此外，发光材料之开发亦为此技术不可或缺之要素

三、OLED 的驅動技朮

從電子學角度簡述有機電致發光顯示器的顯示原理爲：在大于某一閾值的外加電場作用下，空穴和電子以電流的形式分別從陽極和陰極注入倒夾在陽極和陰極間的有機薄膜發光層，兩者結合并生成激子，發生輻射復合而導致發光，發光強度于注入的電流成正比，注入到顯示器件中的每個顯示像素的電流可以單獨控制，不同的顯示像素在驅動信號的作用下，在顯示屏上合成出各種字符、數字、圖形以及圖像。有機電致發光顯示驅動器的功能就是提供這種電流信號。

常用的有機電致發光顯示器件上的驅動方法可分爲有源驅動和無源驅動兩種，有源驅動突出的特點是橫流驅動電路集成在顯示屏上，而且每個發光像素對應集成有一個電荷存儲電容，該存儲電容上的電壓保証在一禎周期內，對應像素一直維持其發光或不發光的狀態。有機電致發光顯示器件具有二極管特性，因此原則上其爲單向電流驅動，但是，由于有有機發光薄膜的厚度在納米量級，發光面積尺寸一般大于一百微米，因此，器件具有很明顯的電容特性，爲了提高顯示器件的刷新頻率，對不發光的像素對應的電容進行快速放電，目前很多驅動電路采用正向橫流反向橫壓的驅動模式。有機電致發光顯示像素上所加的正向電壓大于發光的閾值電壓時，像素將發光顯示；當所加的正向電壓小于閾值電壓時，像素不產生電光效應而不顯示；當所加的正向電壓在閾值電壓附近時，會有微弱的光發出。對于發光的像素，發光強度于注入的電流成正比，因此爲了實現對顯示對比度和亮度的控制，有機電致發光顯示驅動要能夠控制驅動輸出的電流幅值。另外，爲了實現灰度顯示、改善刷新頻率等功能，還要求電致發光顯示驅動器能夠對正向電流的脈寬、反向電壓的幅值和脈寬、頻率等參數進行控制。

對于動態驅動器原理人們把顯示屏上像素的兩個電極做成矩陣型結構，既水平一組顯示像素的同一性質的電極是共同的，總向一組顯示像素的相同性質的另一個電極是共用

的。如果像素可分爲 N 行和 M 列，就可有 N 個行電極和 M 個列電極。上面所提到的行電極和列電極分別對應發光像素的兩個電極，即陰極和陽極。行電極爲陰極或是陽極都可以，在下面分析的情況中我們主要采取行爲陰極的情況來討論。

在實際電路驅動的過程中，要逐行點亮或是逐列點亮像素，通常對于逐行點亮的方式，行電極被稱爲掃描電極，列電極被稱爲數據電極；對于逐行點亮的方式，列電極被稱爲掃描電極，行電極被稱爲數據電極；同樣爲了敘述方便，我們主要采取逐行掃描的方式。

有機電致發光顯示的動態驅動方法是循環地給每行電極施加選擇脈沖，同時所有列電極給出行像素的驅動電流脈沖，從而實現某行所有顯示像素的驅動。這種行掃描是逐行順序進行的，循環掃描一邊所有的行用的時間叫做楨周期。

在一楨中每一行的選擇時間是均等的。假設一楨的掃描行數爲 N，掃描一楨的時間爲 1，那么一行所占有的選擇時間爲一楨時間的 1/N，該值被稱爲占空比系數，在同等電流下，掃描行數的增多將使占空比下降，從而引起有機電致發光像素上的電流注入在一楨中的有效值下降，降低了顯示質量。因此隨著顯示像素的增多，爲了保証顯示質量，就需要適度的提高驅動電流或是采用雙屏電極結構以提高占空比系數。

交叉效應

在動態驅動方式下，某一有機電致發光像素（選擇點）呈顯示效果是由施加在行電極上的選擇電壓與施加在列電極上的選擇電壓的合成來實現的。與該像素不在同一行或同一列的像素點（非選擇點）都處于非選擇狀態下，與該像素在同一行或同一列的像素均有選擇電壓加入，稱之爲半選擇點。該點的電場電壓處于有機電致發光的閾值電壓附近時，屏上將出現不應有的半顯示現象，使得顯示對比度下降，這種現象稱之爲“交叉效應”。

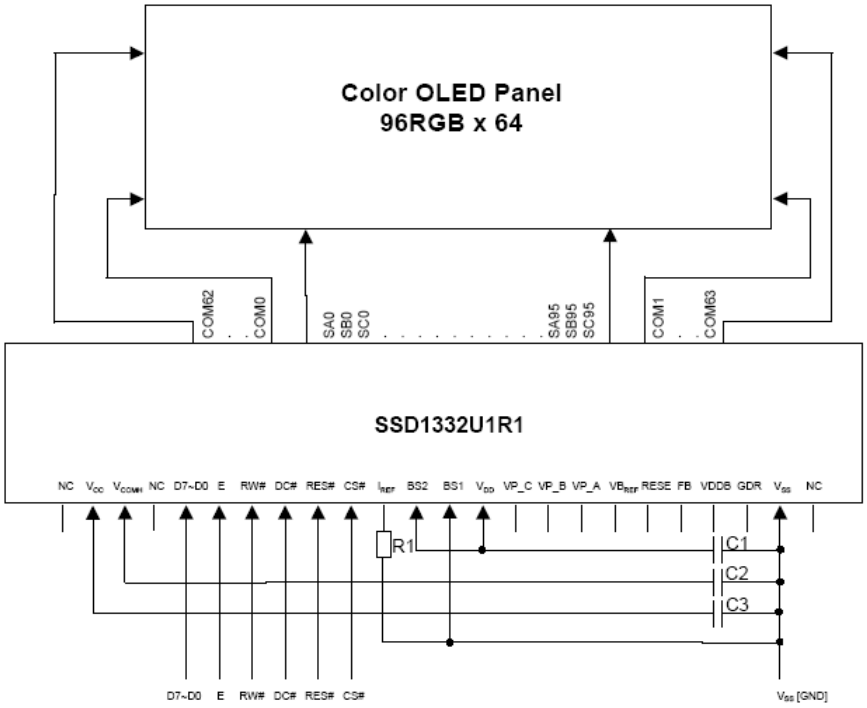
在有機電致發光動態驅動方式中解決“交叉效應”的方法是反向截止法，就是使所有未選中的有機電致發光像素上施加反向電壓。這是因爲有機電致發光的原理是像素中注入電流，正負電荷載流子的復合形成了發光，反向截止強行使可能形成發光的弱場漂移電流、擴散電流都不可能在像素中通過，從而有效地消除了交叉效應。除了由于電極的公用形成“交叉效應”外，有機電致發光顯示屏中正負電荷載流子復合形成發光的機理使任何兩個發光像素，只要組成它們結構的任何一功能膜是直接連接在一起的，那兩個發光像素之間就可能有相互串擾現象，既一個像素發光，另一個像素也可能發出微弱的光。這種現象主要是因爲有機功能薄膜厚度均勻性差，薄膜的橫向絕緣性差造成的。從驅動的角度講，爲了減緩這種不利的串擾，采取反向截止法也是一種有效的方法。

四、具體點亮模組 SSD1332、65K、OLED

SSD1332OLED 是集成了控制器的单一芯片，它可以驱动彩色的无源，公共极为阴极类型的 OLED/PLED 显示。它所具有的 16 比特色彩深度提供了更精细的彩色表达。SSD1332OLED 驅動 IC 不仅有内建的 DC-DC 电压转换器，针对三原色的一个 16 步进的主电流控制和 256 级的对比度控制，而且片内还有一个 SRAM 作为显示缓冲空间。它可支援最大 96RGBx64x16 点阵的 OLED 屏幕，提供 65K 彩色显示。

我們現在點亮的這一款為 96X64 點陣的 OLED 65K 色樣品。點亮模組主要有兩部分組成：外圍電路和調試軟体組成。

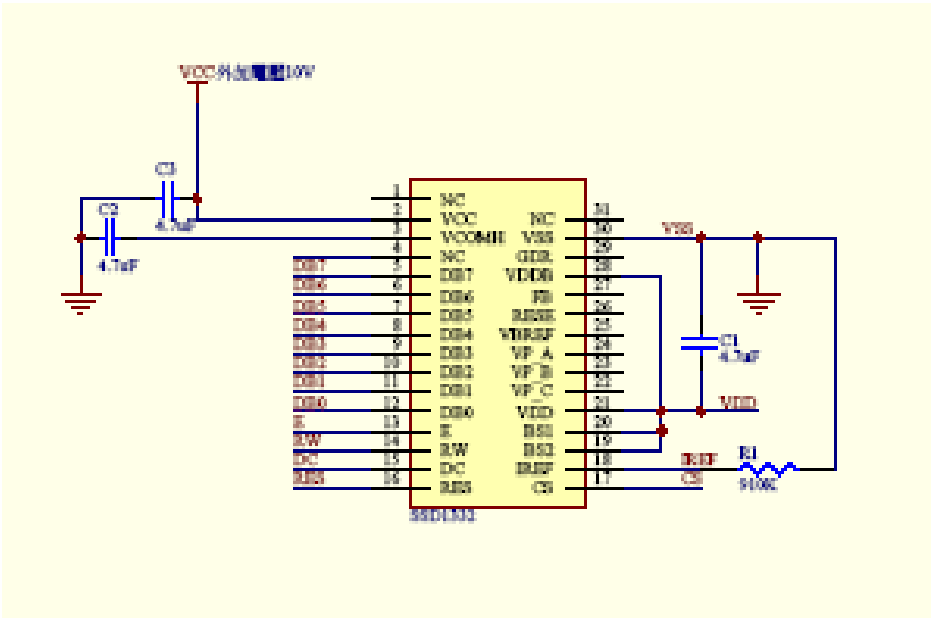
一、外部電路部分：



備註：1、 C1~C3: 4.7uF

2、 R1 = (Voltage at I_{REF} - VSS) / I_{REF} = 910KΩ

DB0-DB7、E、RW、DC、RES、CS、VDD、VSS、VCC(外加電壓)與測試主機連接。



二、調試軟体部分：

利用 C51 語言，在 Keil C51 种編譯執行。圖形編譯軟件為：Image2LCD，以 8080 時序來操作。

以下為 SSD1332 模擬程序：

```
#include<Mreg52.h>                    //富相模組接口及單片機初始定義文件。
#include<delay.c>
```

```
#define Uchar unsigned char
#define Uint    unsigned int

extern unsigned char code tx9[];
extern unsigned char code tx6[];
extern unsigned char code tx7[];
extern unsigned char code tx8[];
extern unsigned char code tx5[];

void Initial(void);
void dispBmp(Uchar *Pbmp);
void ComWrite(Uchar COM);
void DataWrite(Uchar DATA);
void ColorSet(int ptr1);
void SendDATA(int ptr);
void Display1(void);
void Display2(void);
void Display3(void);

unsigned char RGB[]={0x00,0x1f,0x07,0xe0,0xf8,0x00,0x00,0x00,0xff,0xff};
unsigned char
ColorBar[]={0xff,0xff,0xf8,0x00,0xff,0xe0,0x07,0xe0,0x07,0xff,0x00,0x1f,0x00,0x00};

void main(void)
{
    Initial();

    while(1)
    {
            dispBmp(tx9);
            dispBmp(tx5);
            dispBmp(tx8);
            dispBmp(tx7);
            dispBmp(tx6);
            dispBmp(tx5);

            Display1();
            Display2();
```

```
            Display3();
            ColorSet(0x0000);
            ColorSet(0xffff);
            ColorSet(0xf800);
            ColorSet(0x07e0);
            ColorSet(0x001f);
        }
    }

    void Initial(void)
    {
        CS=0;
        delay(500);
        REST=0;
        delay(100);
        REST=1;
        delay(500);

        ComWrite(0x81);     //set High Brightness
        ComWrite(0xdf);
        ComWrite(0x82);     //set High Brightness
        ComWrite(0x1f);
        ComWrite(0x83);     //set High Brightness
        ComWrite(0xff);
        ComWrite(0x87);     //set High Brightness
        ComWrite(0x0f);


        ComWrite(0xa0);
        ComWrite(0x60);    //set 65k color format

        ComWrite(0xa4);     //set Normal Display

        ComWrite(0xa8);     //set Multiplex Ratio
        ComWrite(0x3f);

        ComWrite(0xa9);     //set Power Control
        ComWrite(0x03);
```

```
    ComWrite(0xaf);     //set Display on

    ComWrite(0xb8);
    ComWrite(0x01);
    ComWrite(0x05);
    ComWrite(0x09);
    ComWrite(0x0d);
    ComWrite(0x11);
    ComWrite(0x15);
    ComWrite(0x19);
    ComWrite(0x1d);
    ComWrite(0x21);
    ComWrite(0x25);
    ComWrite(0x29);
    ComWrite(0x2d);
    ComWrite(0x31);
    ComWrite(0x35);
    ComWrite(0x39);
    ComWrite(0x3d);
    ComWrite(0x41);
    ComWrite(0x45);
    ComWrite(0x49);
    ComWrite(0x4d);
    ComWrite(0x51);
    ComWrite(0x55);
    ComWrite(0x59);
    ComWrite(0x5d);
    ComWrite(0x61);
    ComWrite(0x65);
    ComWrite(0x69);
    ComWrite(0x6d);
    ComWrite(0x71);
    ComWrite(0x75);
    ComWrite(0x79);
    ComWrite(0x7d);

}
void ColorSet(int ptr1)
{
```

```
    Uchar i;
    Uint    j;
    ComWrite(0x0a1);
    ComWrite(0x00);
    ComWrite(0x75);
    ComWrite(0x00);
    ComWrite(0x3f);
    for(i=0;i<64;i++)
    {
        ComWrite(0x15);
        ComWrite(0x00);
        ComWrite(0x5f);
        for(j=0;j<96;j++)
        {
            SendDATA(ptr1);
        }
     }
     delay(10000);
}
void dispBmp(Uchar *Pbmp)
{
    register Uint i, j;
    ComWrite(0x0a1);
    ComWrite(0x00);
    ComWrite(0x75);
    ComWrite(0x00);
    ComWrite(0x3f);

    for(i=0;i<64;i++)
    {
        ComWrite(0x15);
        ComWrite(0x00);
        ComWrite(0x5f);
        for(j=0;j<96*2;j++)
        {
        DataWrite(*Pbmp++);

        }
    }
```

```
        delay(10000);
    }

    void ComWrite(Uchar COM)
    {
        E =1;
        RS=0;
        RW=0;
        CS=0;
        P1=COM;
        //E =0;
        CS=1;
        RW=1;
        RS=1;
    }

    void DataWrite(Uchar DATA)
    {
        E =1;
        RS=1;
        RW=0;
        CS=0;
        P1=DATA;
        CS=1;
        RW=1;
        RS=0;
    }
    void SendDATA(int par)
    {
        RS=1;
        RW=0;
        CS=0;
        P1=(par>>8);
        CS=1;
        RW=1;
        RS=0;

        RS=1;
        RW=0;
```

```
        CS=0;
        P1=(0xff&par);
        CS=1;
        RW=1;
        RS=0;
    }
    void Display1(void)
    {
        Uint i,j;
        ComWrite(0x0a1);
        ComWrite(0x00);
        ComWrite(0x75);
        ComWrite(0x00);
        ComWrite(0x3f);
        for(i=0;i<64;i++)
        {
            ComWrite(0x15);
            ComWrite(0x00);
            ComWrite(0x5f);
            for(j=0;j<13;j++)
            {
                DataWrite(ColorBar[0]);
                DataWrite(ColorBar[1]);
            }
            for(j=0;j<14;j++)
            {
                DataWrite(ColorBar[2]);
                DataWrite(ColorBar[3]);
            }
            for(j=0;j<14;j++)
            {
                DataWrite(ColorBar[4]);
                DataWrite(ColorBar[5]);
            }
            for(j=0;j<14;j++)
            {
                DataWrite(ColorBar[6]);
                DataWrite(ColorBar[7]);
            }
```

```
            for(j=0;j<14;j++)
            {
                DataWrite(ColorBar[8]);
                DataWrite(ColorBar[9]);
            }
            for(j=0;j<14;j++)
            {
                DataWrite(ColorBar[10]);
                DataWrite(ColorBar[11]);
            }
            for(j=0;j<13;j++)
            {
                DataWrite(ColorBar[12]);
                DataWrite(ColorBar[13]);
            }
        }
        delay(10000);
    }
    void Display2(void)
    {
        Uint i,j;
        ComWrite(0x0a1);
        ComWrite(0x00);
        ComWrite(0x75);
        ComWrite(0x00);
        ComWrite(0x3f);
        for(i=0;i<32;i++)
        {
            ComWrite(0x15);
            ComWrite(0x00);
            ComWrite(0x5f);
            for(j=0;j<32;j++)
            {
                DataWrite(RGB[0]);
                DataWrite(RGB[1]);
            }
            for(j=0;j<32;j++)
            {
                DataWrite(RGB[2]);
```

```
                DataWrite(RGB[3]);
            }
            for(j=0;j<32;j++)
            {
                DataWrite(RGB[4]);
                DataWrite(RGB[5]);
            }
        }
        for(i=0;i<32;i++)
        {
            for(j=0;j<48;j++)
            {
                DataWrite(RGB[6]);
                DataWrite(RGB[7]);
            }
            for(j=0;j<48;j++)
            {
                DataWrite(RGB[8]);
                DataWrite(RGB[9]);
            }
        }
        delay(10000);
    }
    void display3(void)
    {
        unsigned int i;
        ComWrite(0x0a1);
        ComWrite(0x00);
        ComWrite(0x75);
        ComWrite(0x00);
        ComWrite(0x3f);
        ComWrite(0x15);
        ComWrite(0x00);
        ComWrite(0x5f);
        for(i=0;i<6144;i++)
        {
            DataWrite(0xf8);
            DataWrite(0x00);
        }
```

```
    delay(8000);
    ComWrite(0x0a1);
    ComWrite(0x00);
    ComWrite(0x75);
    ComWrite(0x00);
    ComWrite(0x3f);
    ComWrite(0x15);
    ComWrite(0x00);
    ComWrite(0x5f);
    for(i=0;i<6144;i++)
    {
        DataWrite(0x07);
        DataWrite(0xc0);
    }
    delay(8000);
    ComWrite(0x0a1);
    ComWrite(0x00);
    ComWrite(0x75);
    ComWrite(0x00);
    ComWrite(0x3f);
    ComWrite(0x15);
    ComWrite(0x00);
    ComWrite(0x5f);
    for(i=0;i<6144;i++)
    {
        DataWrite(0x00);
        DataWrite(0x1f);
    }
    delay(8000);
}
```

五、利用 AT90S8515 測試板（新測試板外部可以擴展 8Mbit）點亮模組

此測試板的主要功能就是擴大了外部存儲空間爲 8Mbit,以下相應的硬件設置部分和 C51 測試板相同，以下爲我們 LCM 產品設計課 AVR 單片機程序。主要從硬件測試板中數據地址分配過程請參考 C-STN 測試板原理圖。

```
    #include<io8515.h>
#include <macros.h>

#define DataPort PORTA
```

```
#define CtrlPort PORTD

unsigned char ic=0x18;
unsigned char Vcj=0x3f;


char flash
RGB[]={0x00,0x1f,0x07,0xe0,0xf8,0x00,0x00,0x00,0xff,0xff};
const char
ColorBar[]={0xff,0xff,0xf8,0x00,0xff,0xe0,0x07,0xe0,0x07,0xff,0x00
,0x1f,0x00,0x00};

const
//80-system                        REST   CS     RS     RW    E
  ComWrite=0x11,           //0 0 0    1      0      0      0     1
  ComWrite_set=0x1f,       //0 0 0    1      1      1      1     1
  DataWrite=0x15,          //0 0 0    1      0      1      0     1
  DataWrite_set=0x1b;      //0 0 0    1      1      0      1     1
void Data_out(unsigned char Data)
{
 MCUCR=~BIT(SRE);
 PORTB|=0x18;
 DDRA=0xff;
 DDRD=0x1f;
 CtrlPort=DataWrite;
 DataPort=Data;
 CtrlPort=DataWrite_set;
 MCUCR=BIT(SRE);
 PORTB=ic;
}
void Com_out(unsigned char Com)
{
 MCUCR=~BIT(SRE);
 PORTB|=0x18;
 DDRA=0xff;
 DDRD=0x1f;
 CtrlPort=ComWrite;
 DataPort=Com;
 CtrlPort=ComWrite_set;
```

```
 MCUCR=BIT(SRE);
 PORTB=ic;
}
void DispBmp(unsigned char pg)
{
 unsigned char *ptr;
 unsigned char page,seg;
 switch(pg)
    {
        case 3:ic=0x10;    break;
        case 4:ic=0x11;    break;
        case 5:ic=0x12;    break;
        case 6:ic=0x13;    break;
        case 7:ic=0x14;    break;
        case 8:ic=0x15;    break;
        case 9:ic=0x16;    break;
        case 10:ic=0x17;   break;
        case 11:ic=0x08;   break;
        case 12:ic=0x09;   break;
        case 13:ic=0x0a;   break;
        case 14:ic=0x0b;   break;
        case 15:ic=0x0c;   break;
        case 16:ic=0x0d;   break;
        case 17:ic=0x0e;   break;
        case 18:ic=0x0f;    break;

    }
    ptr=(unsigned char*)0x0260;
    Com_out(0xa1);
    Com_out(0x00);
   Com_out(0x75);
   Com_out(0x00);
   Com_out(0x3f);
    for(page=0;page<64;page++)
    {
        Com_out(0x15);
       Com_out(0x00);
       Com_out(0x5f);
        for(seg=0;seg<96*2;seg++)
```

```
        {
         Data_out(*ptr++);
        }
    }
 delay(20000);
}

void delay(unsigned int t)
{
    unsigned int i,j;

    for(i=0;i<t;i++)
    for(j=0;j<10;j++);
}

void Display(void)
{
 unsigned char i,j;
 Com_out(0xa1);
 Com_out(0x00);
 Com_out(0x75);
 Com_out(0x00);
 Com_out(0x3f);
 for(i=0;i<64;i++)
 {
  Com_out(0x15);
  Com_out(0x00);
  Com_out(0x5f);
  for(j=0;j<13;j++)
  {
   Data_out(ColorBar[0]);
   Data_out(ColorBar[1]);
  }
  for(j=0;j<14;j++)
  {
   Data_out(ColorBar[2]);
   Data_out(ColorBar[3]);
  }
  for(j=0;j<14;j++)
```

```
  {
   Data_out(ColorBar[4]);
   Data_out(ColorBar[5]);
  }
  for(j=0;j<14;j++)
  {
   Data_out(ColorBar[6]);
   Data_out(ColorBar[7]);
  }
  for(j=0;j<14;j++)
  {
   Data_out(ColorBar[8]);
   Data_out(ColorBar[9]);
  }
  for(j=0;j<14;j++)
  {
   Data_out(ColorBar[10]);
   Data_out(ColorBar[11]);
  }
  for(j=0;j<13;j++)
  {
   Data_out(ColorBar[12]);
   Data_out(ColorBar[13]);
  }
 }
 delay(20000);
}
void Display1(void)
{
 unsigned char i,j;
 Com_out(0xa1);
 Com_out(0x00);
 Com_out(0x75);
 Com_out(0x00);
 Com_out(0x3f);
 for(i=0;i<32;i++)
 {
       Com_out(0x15);
       Com_out(0x00);
```

```
            Com_out(0x5f);
            for(j=0;j<32;j++)
            {
                Data_out(RGB[0]);
                Data_out(RGB[1]);
            }
            for(j=0;j<32;j++)
            {
                Data_out(RGB[2]);
                Data_out(RGB[3]);
            }
            for(j=0;j<32;j++)
            {
                Data_out(RGB[4]);
                Data_out(RGB[5]);
            }
        }
        for(i=0;i<32;i++)
        {
            Com_out(0x15);
            Com_out(0x00);
            Com_out(0x5f);
            for(j=0;j<48;j++)
            {
                Data_out(RGB[6]);
                Data_out(RGB[7]);
            }
            for(j=0;j<48;j++)
            {
                Data_out(RGB[8]);
                Data_out(RGB[9]);
            }
        }
        delay(20000);
    }
    void Display2(void)
    {
     unsigned int i;
     Com_out(0xa1);
```

```
Com_out(0x00);
Com_out(0x75);
Com_out(0x00);
Com_out(0x3f);
Com_out(0x15);
Com_out(0x00);
Com_out(0x5f);
for(i=0;i<6144;i++)
{
   Data_out(0xf8);
   Data_out(0x00);
}
delay(20000);
Com_out(0xa1);
Com_out(0x00);
Com_out(0x75);
Com_out(0x00);
Com_out(0x3f);
Com_out(0x15);
Com_out(0x00);
Com_out(0x5f);
for(i=0;i<6144;i++)
{
   Data_out(0x07);
   Data_out(0xe0);
}
delay(20000);
Com_out(0xa1);
Com_out(0x00);
Com_out(0x75);
Com_out(0x00);
Com_out(0x3f);
Com_out(0x15);
Com_out(0x00);
Com_out(0x5f);
for(i=0;i<6144;i++)
{
   Data_out(0x00);
   Data_out(0x1f);
```

```
 }
 delay(20000);
}
void Initial(void)
{
 DDRA=0xff;
 PORTA=0xff;

 DDRD=0b00011000;
 PORTD=0b00000000;    //CS=0;REST=0;
 delay(10);
 PORTD=0b00010000;    //REST=1;
 delay(10);

 Com_out(0x81);
 Com_out(0x9f);
 Com_out(0x82);
 Com_out(Vcj);
 Com_out(0x83);
 Com_out(0xff);
 Com_out(0x87);
 Com_out(0x0f);

 Com_out(0xa0);
 Com_out(0x60);

 Com_out(0xa4);

 Com_out(0xa8);
 Com_out(0x3f);

 Com_out(0xa9);
 Com_out(0x03);

 Com_out(0xaf);

 Com_out(0xb8);
 Com_out(0x01);
 Com_out(0x05);
```

```
 Com_out(0x09);
 Com_out(0x0d);
 Com_out(0x11);
    Com_out(0x15);
    Com_out(0x19);
    Com_out(0x1d);
    Com_out(0x21);
    Com_out(0x25);
    Com_out(0x29);
    Com_out(0x2d);
    Com_out(0x31);
    Com_out(0x35);
    Com_out(0x39);
    Com_out(0x3d);
    Com_out(0x41);
    Com_out(0x45);
    Com_out(0x49);
    Com_out(0x4d);
    Com_out(0x51);
    Com_out(0x55);
    Com_out(0x59);
    Com_out(0x5d);
    Com_out(0x61);
    Com_out(0x65);
    Com_out(0x69);
    Com_out(0x6d);
    Com_out(0x71);
    Com_out(0x75);
    Com_out(0x79);
    Com_out(0x7d);
}

void main(void)
{
 unsigned char step,KS;
 KS=0;
 GIMSK=0x00;
 TIMSK=0x00;
 DDRA=0xff;
```

```
DDRB=0x3f;
DDRC=0xff;
DDRD=0x1f;
Initial();
for(step=0;1;step+=1,step%=20)
   {
       switch(step)
       {

          case 0 : Display();;break;
           case 1 : Display1();break;
           case 2 : Display2();break;
           case 3 : DispBmp(step);break;
           case 4 : DispBmp(step);break;
           case 5 : DispBmp(step);break;
           case 6 : DispBmp(step);break;
           case 7 : DispBmp(step);break;
           case 8 : DispBmp(step);break;
           case 9 : DispBmp(step);break;
           case 10 : DispBmp(step);break;
           case 11 : DispBmp(step);break;
           case 12 : DispBmp(step);break;
           case 13 : DispBmp(step);break;
           case 14 : DispBmp(step);break;
           case 15 : DispBmp(step);break;
           case 16 : DispBmp(step);break;
           case 17 : DispBmp(step);break;
           case 18 : DispBmp(step);break;
       }
       /*if(!(PIND&0x20));       //   轉換畫面調節對比度函數如果 PD.5 口
                                 //     爲 0 執行翻頁程序
   {
     while((PIND&0X20))
        {
         if(!(PINB&0x40))     // 如果 PB.6 口爲 0 執行對比度調節指令
           {
              if(Vcj==0xff) Vcj=0xff;
              else  Vcj+=1;
           }
```

```
                if(!(PINB&0x80))     // 如果 PB.7 口爲 0 執行對比度調節指令
                  {
                     if(Vcj==0) Vcj=0;
                     else Vcj-=1;
                  }
                  Com_out(0x82);
                 Com_out(Vcj);

                 delay(10000);
               }
          }*/
      }
  }
```