



SY826

段码屏驱动原理



请认真阅读关于 Amicro 知识产权政策

本文中提及的“珠海一微半导体股份有限公司”（简称“本公司”）对此产品持有知识产权及其对应的法定权益。未经合法授权，使用本公司的 MCU 或其他相关产品的行为将被视为侵权。对于任何未经授权而侵犯本公司知识产权的实体或个人，本公司有权采取法律手段保护权益，并将对由此造成的损害寻求赔偿。

*本公司保留对产品规格书中，关于产品设计、功能和可靠性方面的改进作进一步说明的权利，但对于规格内容的使用并不承担责任。文档中所描述的应用案例仅供参考，本公司不保证和不表示这些应用，在没有更深入地更改和修正就能适用。同时，本公司不推荐产品使用在可能会对人身造成危害的场景。本公司的产品未经特别授权，不得用于救生、维生器件或系统中或作为关键器件使用。

*本公司保留在未经预告的情况下修改其产品的权利。

目录

引言 3

一、 扫灯入门 4

二、 扫灯新手 5

三、 扫灯进阶 9

四、 点灯大师 9



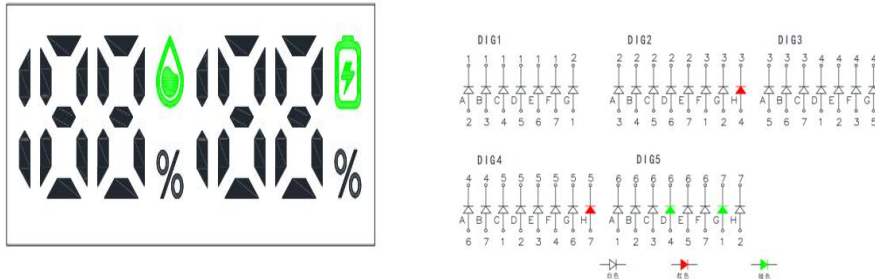
引言

在使用 SY826 标案开发时，有很多工程师对驱动段码屏的代码有疑惑，不知道如何入手，为了让工程师们了解段码屏扫灯的基本原理，在拿到新屏幕后能基于我们的标案快速把屏幕点亮，特建立此文档。

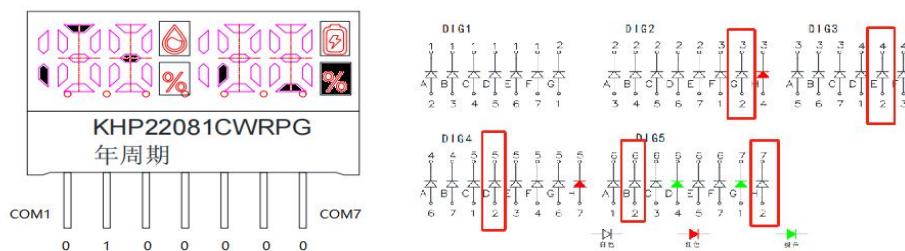


一、扫灯入门

以 7pin 的屏为例子，拿到一个段码屏后，第一步先要到它的规格书。如下图，利用二极管的单向导通性，如果 COM2 置高电平，COM1 置低电平，则 A1 会被点亮，依次类推，即可点亮所有的段。又因为人的眼睛有视觉残留效应，所有我们每次只需点亮一段，只要刷新的速率足够快，视觉上看起来就是同时点亮的。

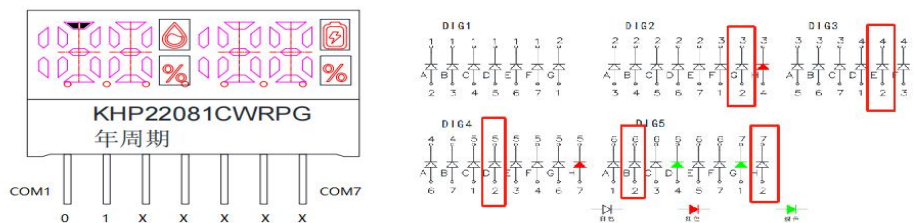


按上述方法操作，聪明如你，肯定会发现一个问题：我只想点亮几个段，但是其他的一段或多端也亮起来了。这是为什么呢？我们来看一个实际案例：现在我们想要点亮 A1，其他的段都熄灭。如果按照上述点灯方法，就是 COM1 置低电平，COM2 置高电平，COM3 ~ COM7 置低电平。这个时候 A1 确实是被点亮了。但是同时 G2、E3、D4、B5、H5 也会被点亮。



为了解决这个问题，我们引入高阻态这个概念（不懂的同学自行百度）。

现在我们的 IO 就有三个状态了（高电平：1 低电平：0 高阻态：X），那我们再来看，COM1 低电平，COM2 高电平，COM3 ~ COM7 高阻态，这时候你就会发现，真的只有 A1 在亮耶。恭喜你，你已经学会点灯了。



二、扫灯新手

如果你只了解了第一步《扫灯原理》，那么只能说你学会点灯了。那么，怎样才能做一个合格的点灯人呢？从现在开始，就需要您有一点点的 C 编程基础了。这里我们引入位域的概念（还是自行百度，求人不如求己）。

看到这里，你已经了解了位域的概念，那么请看下面的结构体定义

```
#define DIGITS_NUM (8)
```

```
typedef struct
```

```
{
```

```
    u8 a : 1;
```

```
    u8 b : 1;
```

```
    u8 c : 1;
```

```
    u8 d : 1;
```

```
    u8 e : 1;
```

```
    u8 f : 1;
```

```
    u8 g : 1;
```

```
    u8 h : 1;
```

```
} DIGITS_T;
```

```
typedef union
```

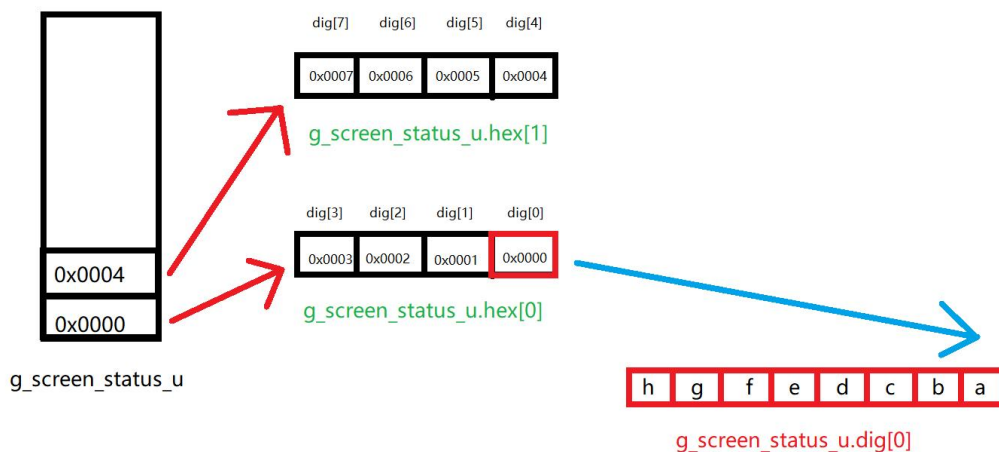
```
{
```

```
    DIGITS_T dig[DIGITS_NUM];
```

```
    u32      hex[2];
```

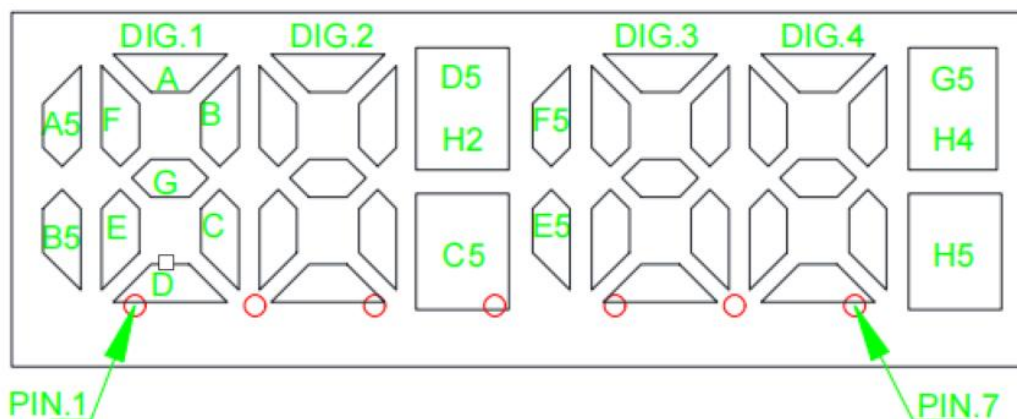
```
} LEDSCREEN_STATUS_U;
```

我们来看一下他的内存分配，假设 "LEDSCREEN_STATUS_U g_screen_status_u;" 被分配到了 0x0000 地址，那么他的成员变量分布如下：



是的，你猜怎么着？DIGITS_T 这个结构体，它的 8 个成员 a~h,刚好可以对应上段码屏上的一个数码“8”，这个时候，又有眼尖的朋友站出来说：不对，结构体里不是还有个 h 吗？这个对应哪里？很简单，你可以直接不用，另外一种做法是，你可以利用它来映射其他多出来的段，只要你能记住它们的对应关系即可。（如 dig[0].h -> H1, dig[1].h -> H2），同理，有的屏幕上，还有油量等图标，也是一样的做法，自由发挥，把他看成是一个或者多个 DIGITS_T 就可以了。

下面举个栗子：



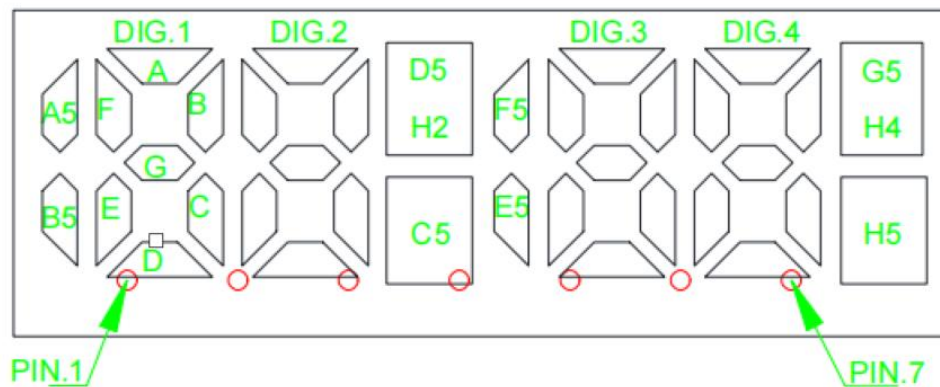
你当然可以按照规格书上的原始排序，一一对应到扫灯函数里面去。但是有些时候，屏厂为了走线方便或者某些原因（偷懒），本该同属一个区域的段，却分布得七零八落。所以，为了方便写程序（特别是屏幕的 pin 脚较多时），通常我会将他们重新映射。如下图，是根据规格书写出来的真值表

	A	B	C	D	E	F	G	H	I	J
1		COM1	COM2	COM3	COM4	COM5	COM6	COM7	高电平	
2	COM1		A1	B1	C1	D1	E1	F1		
3	COM2	G1		A2	B2	C2	D2	E2		
4	COM3	F2	G2		H2	A3	B3	C3		
5	COM4	D3	E3	F3		G3	A4	B4		
6	COM5	C4	D4	E4	F4		G4	H4		
7	COM6	A5	B5	C5	D5	E5		F5		
8	COM7	G5	H5							
9	低电平									
10										

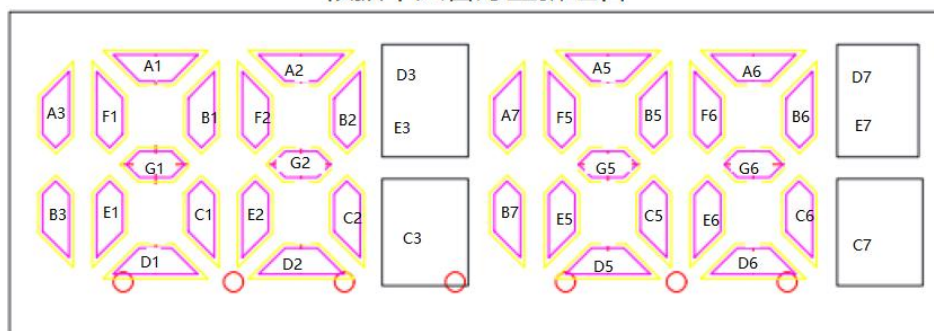
我们可以根据自己的个人习惯或者写程序方便，在原始真值表的基础上进行重新映射。

	A	B	C	D	E	F	G	H	I	J
12		COM1	COM2	COM3	COM4	COM5	COM6	COM7	高电平	
13	COM1		A1	B1	C1	D1	E1	F1		
14	COM2	G1		A2	B2	C2	D2	E2		
15	COM3	F2	G2		H2/E3	A3/A5	B3/B5	C3/C5		
16	COM4	D3/D5	E3/E5	F3/F5		G3/G5	A4/A6	B4/B6		
17	COM5	C4/C6	D4/D6	E4/E6	F4/F6		G4/G6	H4/E7		
18	COM6	A5/A3	B5/B3	C5/C3	D5/D3	E5/B7		F5/A7		
19	COM7	G5/D7	H5/C7							
20	低电平									
21										
22										

重新映射后，我们就得到了一个新的段码排序。下图是原始排序和重新映射后的段码排序，大家可以仔细思考一下两者有什么区别。



根据个人喜好重新组合



接下来，我们只要把这些对应关系一一填到下面的驱动函数里，如果你眼不迷手不抖没打错字的话，那个恭喜你，这个驱屏函数你就实现了。我们来看看扫灯函数的具体实现，每次进入 `screen_driver_scan_loop()` 这个函数，我们通过调用 `LED_ALLCOM_SET_NONE()`；将所有 COM 脚都设置为高阻态，然后依次扫描，例如某次扫描里，我们先将 `COM1` 置低电平，然后根据对应段点的成员状态，如果该位为 1，则将对应的 COM 置高电平，否则保持高阻态。这里可能有工程师会疑惑：扫 `COM1` 的时候为什么还要嵌套一个 `switch`，直接一次扫完 `COM2-COM7` 不可以吗？如果你拿到的是屏幕是单色的，那么一次扫多段和每次扫一段，视觉效果是一样的，但是，如果你的屏幕是彩色的，那么你会发现，一次扫多段的时候部分段点的亮度是不一样的。这里推荐使用每次只扫一段的方式，使屏幕的亮度均匀一些。

```

6 void screen_driver_scan_loop(void)
7 {
8     static xdata COM_E scan_com = COM_1;
9     static xdata u8 cnt;
10    LED_ALLCOM_SET_NONE();
11
12    switch (scan_com)
13    {
14        case COM_1:
15            SET_COM_OUTVALUE(CFG_PIN_COM1_PORT, CFG_PIN_COM1_PINNUM, SCAN_COM_USING_LEVEL);
16
17            switch (cnt)
18            {
19                case 0:
20                    com_set_mode_by_state(COM_2, g_screen_status_u.dig[0].a);
21                    break;
22                case 1:
23                    com_set_mode_by_state(COM_3, g_screen_status_u.dig[0].b);
24                    break;
25                case 2:
26                    com_set_mode_by_state(COM_4, g_screen_status_u.dig[0].c);
27                    break;
28                case 3:
29                    com_set_mode_by_state(COM_5, g_screen_status_u.dig[0].d);
30                    break;
31                case 4:
32                    com_set_mode_by_state(COM_6, g_screen_status_u.dig[0].e);
33                    break;
34            }
35            cnt++;
36            if (cnt == 5)
37            {
38                scan_com = COM_2;
39                cnt = 0;
40            }
41        }
42    }

```


又有同学会问：LEDScreen_Status_U 这个联合体里，hex 成员是干嘛用的？

答案：快速赋值。举个例子：我想把屏幕中的第一个数码“8”（dig1）点亮，一种做法是：

```
g_screen_status_u.dig[0].a = 1;
g_screen_status_u.dig[0].b = 1;
g_screen_status_u.dig[0].c = 1;
g_screen_status_u.dig[0].d = 1;
g_screen_status_u.dig[0].e = 1;
g_screen_status_u.dig[0].f = 1;
g_screen_status_u.dig[0].g = 1;
```

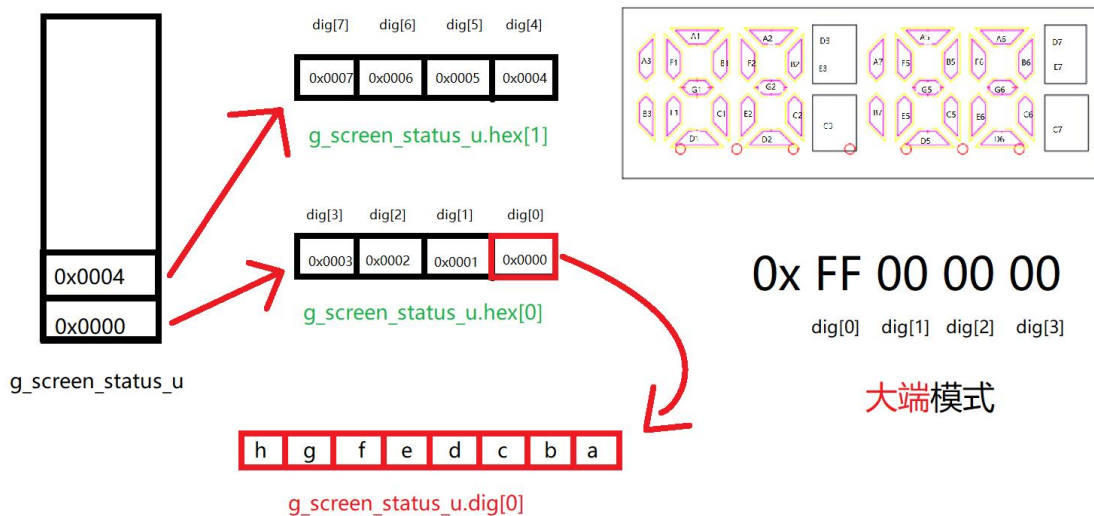
另一种做法是：

```
g_screen_status_u.hex[0] = 0xFF000000;
```

反正我是喜欢用第二种做法。

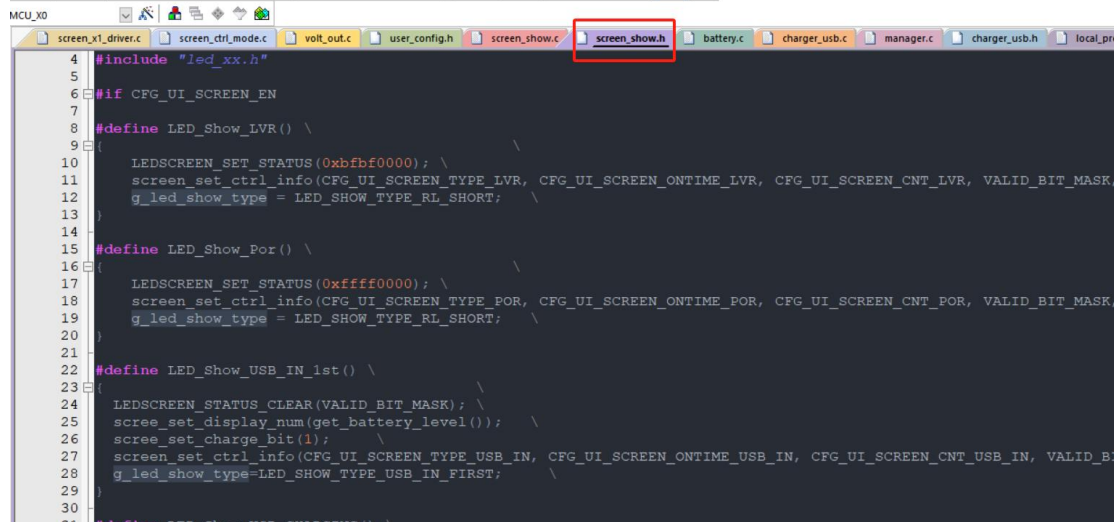
又有眼睛的同学：你这个写法有问题啊，dig[0]不是低地址吗，为什么在 hex 里对应的是高位啊？

答案：这个就跟字节序（还是自行百度，求人不如求己）有关系了，keil C51 是大端模式，我可以很肯定的告诉你，就是这样对应的，没有错。



三、扫灯进阶

如无意外，你已经可以任意点亮你想要点亮的内容了。那么，常亮、常灭、闪烁这些灯效，又改怎么处理呢？实际上，上电、吸烟、短路、低电等等这些灯效，我们都已经用宏写好了，你只需要在设置灯效前，把你要显示的内容（要亮哪些段）再设置一下就好了。

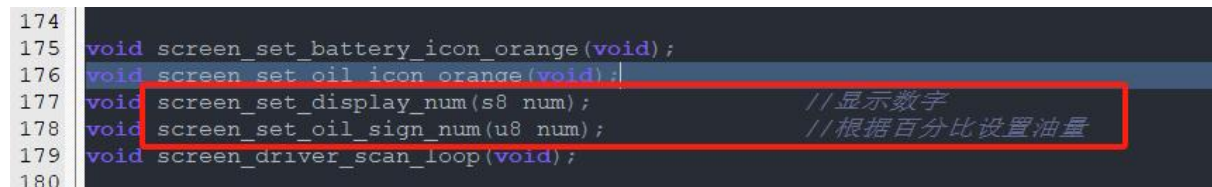


```

4 #include "led_xx.h"
5
6 #if CFG_UI_SCREEN_EN
7
8 #define LED_Show_LVR() \
9 {
10     LEDSCREEN_SET_STATUS(0xbfbf0000); \
11     screen_set_ctrl_info(CFG_UI_SCREEN_TYPE_LVR, CFG_UI_SCREEN_ONTIME_LVR, CFG_UI_SCREEN_CNT_LVR, VALID_BIT_MASK); \
12     g_led_show_type = LED_SHOW_TYPE_RL_SHORT; \
13 }
14
15 #define LED_Show_Por() \
16 {
17     LEDSCREEN_SET_STATUS(0xffff0000); \
18     screen_set_ctrl_info(CFG_UI_SCREEN_TYPE_POR, CFG_UI_SCREEN_ONTIME_POR, CFG_UI_SCREEN_CNT_POR, VALID_BIT_MASK); \
19     g_led_show_type = LED_SHOW_TYPE_RL_SHORT; \
20 }
21
22 #define LED_Show_USB_IN_1st() \
23 {
24     LEDSCREEN_STATUS_CLEAR(VALID_BIT_MASK); \
25     screen_set_display_num(get_battery_level()); \
26     screen_set_charge_bit(1); \
27     screen_set_ctrl_info(CFG_UI_SCREEN_TYPE_USB_IN, CFG_UI_SCREEN_ONTIME_USB_IN, CFG_UI_SCREEN_CNT_USB_IN, VALID_BIT_MASK); \
28     g_led_show_type = LED_SHOW_TYPE_USB_IN_FIRST; \
29 }
30
31 #define LED_Show_USB_CHARGING() \

```

通常，我们会抽出几个接口来（电量、油量、闪灯图标、百分号图标等等），对不同类型的屏幕来说，他的具体实现是不一样的，所以需要工程师们自行实现。



```

174 void screen_set_battery_icon_orange(void);
175 void screen_set_oil_icon_orange(void);
176 void screen_set_display_num(s8 num); //显示数字
177 void screen_set_oil_sign_num(u8 num); //根据百分比设置油量
178 void screen_driver_scan_loop(void);
179
180

```

四、点灯大师

段码屏呼吸灯效??? 是的，没有听错，这种灯效也是可以实现的。实际上就是模拟 pwm，在扫灯的基础上再套上一层模拟 pwm 控制，只有在 pwm 的正脉宽时间里，如果该段需要点亮，才点亮，否则其余时间都是熄灭的。（感兴趣的同学自行实现）