西安交通大學

嵌入式系统设计实验报告

电信学部 自动化学院

学生姓名 吴思源

班级 自动化钱 71

学号 2171310846

指导教师 刘美兰

2019 年 12 月

目　录

1 实验任务 1
2 实验步骤 2
2.1 完成动态切换图片 2
2.1.1 按键扫描 2
2.1.2 制作图片 2
2.1.3 动图显示 3
3 程序运行结果 4
3.1 按键换图运行结果 4
3.2 动图运行结果 4
4 实验中遇到的问题与解决 5
4.1.1 W90P170 驱动程序问题 5
4.1.2 刷新黑边的问题 5
4.1.3 刷新速率的问题 5
5 附录：关键程序文件 7
5.1 按键换图 7
5.2 动图 8

1 实验任务

- 在 LCD 上显示一张图片
- 在 LCD 上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；
- 在 LCD 上显示一张动态图片（GIF 图）。

2 实验步骤

2.1 完成动态切换图片

2.1.1 按键扫描

为了能够实现按键切换动态图片，需要将例程中的按键部分拷贝到工程文件中，且仅使用其中的一个按键。将中断部分代码做如下改动：

```
if (KeyValue == 0x0)
{
    status ++;
}
        switch(status)      //
   {
        case 0:{
             GLCD_bitmap (0, 0, 320, 240, pic1);
             break;}
        case 1:{
             GLCD_bitmap (0, 0, 320, 240, pic2);
             break;}
        case 2:{
             GLCD_bitmap (0, 0, 320, 240, pic3);
             break;}
   }
```

即识别到代表数字 0 的按键按下后，将状态数加一。之后根据状态数定义为要显示的图片，切换状态数时改变图片的显示。

2.1.2 制作图片

将文件夹中附上的按键换图图片用画图软件裁剪成 320*240 像素大小，然后使用 Image2Lcd 软件进行处理，处理后得到代表 16 位的全彩色图片的 C 语言数组，用来在 LCD 屏幕上显示对应的按键换图图片。

将文件夹中附上的动图显示图片用画图软件裁剪成 320*240 像素大小，然后使用 Image2Lcd 软件进行处理，处理后得到代表 16 位的全彩色图片的 C 语言数组，用来在 LCD 屏幕上显示对应的动图显示图片。

2.1.3 动图显示

动图显示部分的程序在例程上修改得到，即将例程动图显示中的十几张图片替换为程序图片文件文件夹中动图图片。动图图片已经通过 **Image2Lcd** 软件进行处理，并将 C 语言数组复制粘贴到 gImage_pic.c 的 C 语言文件中。

程序通过中断实现了动图自动切换，即每隔固定一段时间发出一个中断信号，收到中断信号后程序刷新 LCD 缓冲区，改变图片的显示。相关部分的关键代码如下所示

```
for (;;)
  {                                 /* Loop forever                        */

    if (Clock100ms)
  {
      Clock100ms = 0;

      if (clk_ani++ == 1)
    {              /* Draw animation picture every 200 ms*/
        clk_ani = 0;

    switch(pic)   //ᶜ
    {
      case 0:
         GLCD_bitmap (0, 0, 320, 240, gImage_1);
         break;
        case 1:
           GLCD_bitmap (0, 0, 320, 240, gImage_2);
         break;
      case 2:
           GLCD_bitmap (0, 0, 320, 240, gImage_3);
         break;
      case 3:
           GLCD_bitmap (0, 0, 320, 240, gImage_4);
         break;
      case 4:
           GLCD_bitmap (0, 0, 320, 240, gImage_5);
         break;
      case 5:
           GLCD_bitmap (0, 0, 320, 240, gImage_6);
         break;
      case 6:
           GLCD_bitmap (0, 0, 320, 240, gImage_7);
         break;
    }
     if (pic++ > 6) pic = 0;

      }
    }
```

3 程序运行结果

3.1 按键换图运行结果

系统启动，按下 0 键之后，可以切换图片。刷新方式是从上到下无缝刷新。具体见视频文件。

3.2 动图运行结果

系统启动，自动显示动图。具体见视频文件。

4 实验中遇到的问题与解决

4.1.1 W90P170 驱动程序问题

例程中的 W90P170 驱动程序存在问题，导致使用例程并不能点亮 ARM 实验板上的 LCD 屏幕。

解决：需要检查电脑，寻找能够点亮并驱动 ARM 实验板上 LCD 屏幕的 W90P170.s 驱动程序，将对应程序拷贝到工程文件夹中，替换原有的 W90P170.s 驱动程序，才能够正常工作。

4.1.2 刷新黑边的问题

在程序的调试中遇到了如下问题：图片从上到下刷新显示的时候，下半部分未刷之前会有黑边。

检查代码发现，从静态显示图片的代码中拷贝来的 GLCD.c 源文件，会在显示之前有一行

```
memset(fb_start, 0x00, FB_SIZE);      /* Clear Frame Buffer      */
```

这一行代码于刷新之前将 LCD 彩屏显示的缓冲区清空，使得刷新过程中，LCD 未刷到的下半部分已经没有要显示的信息，因此导致从上到下刷新时下部分会出现黑边。删去这一行则刷新正常。

另外，在从动图的例程中拷贝得到的 GLCD.c 文件已经做了这样的处理，不会出现黑边的情况。因此直接从动图的例程中拷贝程序不会出现这样的问题。

4.1.3 刷新速率的问题

在动图显示中，遇到如下问题，由于图片从上到下刷新速度过慢，导致动图显示并没有动图的效果，反而像是相似的图片一张张分别从上到下刷新。

检查代码发现，由于图片刷新的延时是中断控制程序控制的，而中断控制程序中提供的中断间隔时间过长，将中断控制程序做如下改动

```
__irq void IRQ_Handler (void) {

  switch (REG_AIC_ISNR & 0x1F) {
    case IRQ_TIMER1:                    /* Timer 0 interrupt index
 */
```

```
      if (!((ticks++ ^ 1) & 0x0F)) {    /* Set Clock100ms to 1 every 100 ms
改动位置，将原来每 100ms 刷新改成每 10ms 刷新*/
        ticks      &= ~0x0F;
        ticks      +=  0x01;
        Clock100ms  =  1;
      }
      if (ticks >= (10 << 4)) {         /* Set Clock1s to 1 every 1 second
*/
        ticks       =  0;
        Clock1s     =  1;
      }
      REG_TISR = 2;                     /* Clear Timer 1 interrupt status
*/
      break;
  }
  REG_AIC_IPER;                         /* Acknowledge interrupt
*/
  REG_AIC_EOSCR = 0;                    /* End of interrupt service routine
*/
    }
```

之后再进行测试，发现能够正常显示，效果比较完美，具体可以参考录制的视频。

5 附录：关键程序文件

5.1 按键换图

```
void KPIInit(void);
void Delay(int t1)
{
    int x = 0;
    int y = 0;
    for(x = 0; x < t1;x++)
    for(y = 0; y < 1000;y++);
};
int status;
int main (void){
      int num     = -1;
  int dir     =  1;
  int off     =  0;
  int pic     =  0;
  int clk_ani =  0;
    WriteReg(REG_EXT0CON, EBI_EXT0_VALUE);
    KPIInit();
    status = 0;

  GLCD_bitmap (0, 0, 320, 240, pic1);
    GLCD_init();
    while(1)
    {

        if (status > 2) status = 0;

    }
}


void KPIInit(void){
     WriteReg(REG_GPIO_CFG2, 0x000aaaaa);
     WriteReg(REG_AIC_SCR29, 0x00000045);
     WriteReg(REG_AIC_MECR,  0x20000000);
     WriteReg(REG_KPICONF,   0x00142fff);
}
```

```
__irq void IRQ_Handler(void){
     u32 KeyValue, temp;
     temp = ReadReg(REG_KPISTATUS);
     temp &= 0x0000000f;
     if(temp > 0xB){
        KeyValue = temp - 0xC;
     }
     else if(temp < 0x4){
        KeyValue = temp + 0xC;
     }
     else{
        KeyValue = temp;
     }
     if (KeyValue == 0x0)
     {
         status ++;
     }
            switch(status)      //
        {
            case 0:{
                 GLCD_bitmap (0, 0, 320, 240, pic1);
                 break;}
            case 1:{
                 GLCD_bitmap (0, 0, 320, 240, pic2);
                 break;}
            case 2:{
                 GLCD_bitmap (0, 0, 320, 240, pic3);
                 break;}
        }

     WriteReg(EBILED_ADDRESS, ~KeyValue);
}
```

5.2 动图

```
int main (void) {                       /* Main Function
 */

  int num     = -1;
  int dir     =  1;
  int off     =  0;
  int pic     =  0;
  int clk_ani =  0;
```

```
 Reg8(0x78000000) = LED_ALL;           /* Turn off all LEDs
*/

 /* Setup Timer1 to generate interrupt every 10 ms
*/
 REG_TCR1  = 0xAC000000 + (15-1);      /* Dbg, IE, period mode, rst, prsc=80
*/
 REG_TCR1  = 0xA8000000 + (15-1);      /* Dbg, IE, period mode,      prsc=80
*/
 REG_TICR1 = (15000000 / (15 * 100));  /* Reload value=15MHz/(prescaler*100)
*/
 REG_TCR1 |= (1 << 30);                /* Start Timer 1
*/

 /* Setup AIC for Timer1 interrupt
*/
 REG_AIC_SCR14 = (1 << 6) | 1;         /* Int trig on high level, priority 1
*/
 REG_AIC_TEST  =  1;                   /* Enable Debugging of AIC
*/
 REG_AIC_MECR  = (1 << IRQ_TIMER1);    /* Enable Timer 1 interrupt
*/

 GLCD_init();
 GLCD_bitmap (0, 0, 320, 240, gImage_1);

 for (;;)
 {                             /* Loop forever                        */

   if (Clock100ms)
 {
     Clock100ms = 0;

     if (clk_ani++ == 1)
   {             /* Draw animation picture every 200 ms*/
       clk_ani = 0;

   switch(pic)   //ᶜ
   {
     case 0:
        GLCD_bitmap (0, 0, 320, 240, gImage_1);
        break;
       case 1:
          GLCD_bitmap (0, 0, 320, 240, gImage_2);
        break;
     case 2:
          GLCD_bitmap (0, 0, 320, 240, gImage_3);
```

```
          break;
        case 3:
             GLCD_bitmap (0, 0, 320, 240, gImage_4);
          break;
        case 4:
             GLCD_bitmap (0, 0, 320, 240, gImage_5);
          break;
        case 5:
             GLCD_bitmap (0, 0, 320, 240, gImage_6);
          break;
        case 6:
             GLCD_bitmap (0, 0, 320, 240, gImage_7);
          break;
      }
       if (pic++ > 6) pic = 0;

        }
      }

      if (Clock1s)
    {                       /* Blink LED every 1 second           */
        Clock1s = 0;

        if (!off)
      {
          /* Calculate 'num': 0, 1, ... , LED_NUM-1, LED_NUM-1, ... , 1, 0, 0,
.*/
          num += dir;
          if (num == LED_NUM) { dir = -1; num =  LED_NUM-1; }
          else if   (num < 0) { dir =  1; num =  0;         }

          Reg8(0x78000000) = ~led_mask[num];  /* Turn  on LED with index 'num'
 */
        } else
      {
         Reg8(0x78000000) = LED_ALL;         /* Turn off all LEDs
*/
        }

        off = !off;
      }
    }
}
```