LCD 显示字符

RT-Thread 评估板 RealTouch 裸机例程

版本号：1.0.0

日期：2012/9/19

修订记录

日期	作者	修订历史
2012/9/19	Heyuanjie87	添加例程说明

LCD 显示字符

通过本章学习你将了解如何操作 LCD 显示屏上的像素点以及如何把 ASCII 字符绘制到屏幕上。

1. LCD 控制器 RA8875 简介

RA8875 是一个文字与绘图模式的双图层液晶显示 (TFT-LCD) 控制器，可结合文字或 2D 图形应用，最大可支持到 800*480 点分辨率的中小尺寸数字面板。内建 768KB 显示内存可提供大多数使用者的应用一个更弹性的解决方案。此外，使用者可藉由选用外部串行式 Flash 接口，支持 BIG5/GB 编码，可提供最大达 32*32 像素之的字型输入。在图形的使用上，RA8875 支持 2D 的 BTE 引擎 (Block Transfer Engine)，此功能兼容于一般通用的 2D BitBLT 功能，可处理大量图形数据转换与传送。同时 RA8875 也内建几何图形加速引擎 (Geometric Speed-up Engine)，提供使用者透过简单的设定轻松画出直线、矩形、圆形和椭圆的几何图形。（详细介绍参见工程目录下的 RA8875 中文规格书）

2. 本例程所用硬件资源

1) 7 寸 800*480 像素 LCD
2) RA8875 LCD 控制器
3) FSMC 以及相关 I/O

3. 例程

3.1 LCD 操作接口简介

代码一 画点

```
/* ra8875.c */

void ra8875_lcd_set_pixel(const uint16_t* pixel, int x, int y)
{
    /* 设置内存写入光标位置 */
    _set_write_cursor(x, y);
     /* 准备数据写入 */
    LCD_CmdWrite(MRWC);
     /* 写入像素数据 */
```

```
    LCD_DataWrite(*(uint16_t *)pixel);
  }
```

pixel 是一个 16 位的像素值的地址，x、y 是以屏幕左上方为原点水平向右为 x 增垂直向下为 y 增的坐标值。

代码二　画水平线段

```
  /* ra8875.c */

  void ra8875_lcd_draw_hline(const uint16_t* pixel, int x1, int x2,
int y)
  {
     /* 光标属性设定 */
     LCD_CmdWrite(MWCR0);
     LCD_DataWrite(0x00);
     /* 光标位置设定 */
     _set_write_cursor(x1, y);
     /* 准备写入像素数据 */
     LCD_CmdWrite(MRWC);
     /* 写入数据 */
     for(; x1 < x2; x1++)
     {
        LCD_DataWrite(*(uint16_t *)pixel);
     }
  }
```

Pixel 为 RGB565 格式像素值的地址，x1 为线段左端点、x2 为线段右端点、y 表示线段在哪一行。

代码三　画垂直线段

```
  /* ra8875.c */

  void ra8875_lcd_draw_vline(const uint16_t* pixel, int x, int y1,
int y2)
  {
     /* 设定光标属性为从上向下增长 */
     LCD_CmdWrite(MWCR0);
     LCD_DataWrite(0x00 | 1<<3);
     /* 设置起始点 */
     _set_write_cursor(x,y1);
     /* 准备写入像素数据 */
     LCD_CmdWrite(MRWC);
     /* 绘制垂直方向的像素 */
     for(; y1<y2; y1++)
```

```
    {
        LCD_DataWrite(*pixel);
    }
  }
```

Pixel 为像素地址，x 表示线段在哪一列、y1 表示线段上端点、y2 表示线段下端点。

代码四　画杂色水平线段

```
  /* ra8875.c */

  void ra8875_lcd_blit_line(const char* pixels, int x, int y,
uint16_t size)
  {
     uint16_t *ptr;
     /* 光标水平移动 */
     LCD_CmdWrite(MWCR0);
     LCD_DataWrite(0x00);
     /* 设置起点 */
     _set_write_cursor(x, y);

     ptr = (uint16_t*)pixels;
     /* 告诉 LCD 控制器将写入像素值 */
     LCD_CmdWrite(MRWC);
     /* 写入像素 */
     while(size--)
     {
        LCD_DataWrite(*ptr++);
     }
  }
```

上面介绍的画点画线都是单一色彩的。如果你想实现一个渐变效果，用画点的方式会因函数的反复调用影响效率。这个函数允许你一次设置多种像素值，因此只需一次调用即可实现你的想法。x,y 表示起点、size 表示要绘制的像素个数。

3.2 应用实例

代码五　绘制 ASCII 字符

```
  /* display.c */
```

```
  void display_drawascii(char c, uint16_t x, uint16_t y, uint16_t
color)
  {
      const uint16_t *pdata;
      uint16_t data;
      uint16_t pos;
      uint16_t line,column;
      uint16_t white = 0xFFFF;
      /* ASCII 码表中前 32 个字符为控制字符，不可显示 */
     pos = (c - 32) * _current_font->Height;
      /* 获取字体地址 */
      pdata = &_current_font->Table[pos];

      for (line = y; line < y + _current_font->Height; line ++)
      {
              /* 取出字体的一行点阵数据 */
              data = *pdata;
              /* 指向下一行 */
              pdata ++;
              /* 绘制行的各个像素 */
              for (column = x; column < x + _current_font->Width;
column ++)
              {
                  if (data & 0x01)
                      {
                          /* 显示前景色 color */
                          ra8875_lcd_set_pixel((char*)&color,
column, line);
                      }
                      else
                      {
                          /* 显示白色背景 */
                          ra8875_lcd_set_pixel(&white, column,
line);
                      }

                      data >>= 1;
              }
      }
  }
```

C 为 ACSII 字符、x,y 为字符位置、color 为字符颜色。字体的点阵数组定义在 fonts.c 中的，其中定义了 4 种不同大小的字体_current_font 是一个字体属性结构体指针（参见 fonts.h）用来标示当前使用的哪种大小的

字体。下面将以 16 像素宽 24 像素高的“！”字符给大家讲解一个字符的绘制过程。“！”字符字体定义如下图所示。

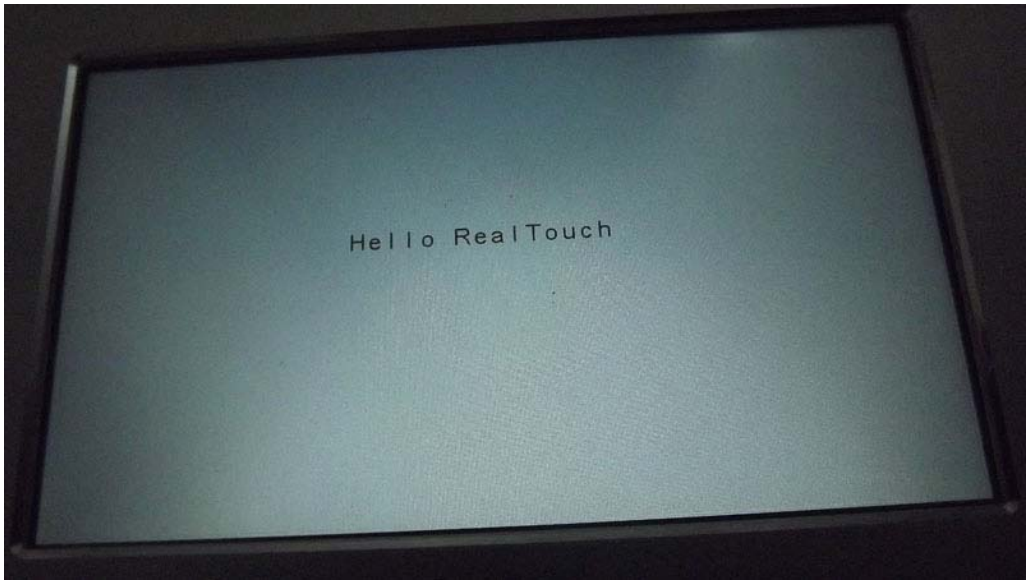
图一 “！”字体定义

```
'!'

0x0000, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180,
0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0000, 0x0000,
0x0180, 0x0180, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
```

字体标示像素的方式跟位图标示像素的方式是不一样的，字体用一位标示一个像素，为 0 的位表示这个地方显示为背景色、为 1 的位表示这个地方显示为前景色。低位为左边的像素高位为右边的像素。这里第一个 0x0000 表示字符最顶行的 16 个像素全显示为背景色，第二个 0x0180 表示第二行当中两个像素显示为前景色左右的像素显示为背景色。

4. 下载运行



这个字符串的输出是在字符输出的基础上实现的，实现过程参见 display.c 还有其他画线的功能就请读者自己尝试了。