RAiO

RA8889_Lite

User Guide

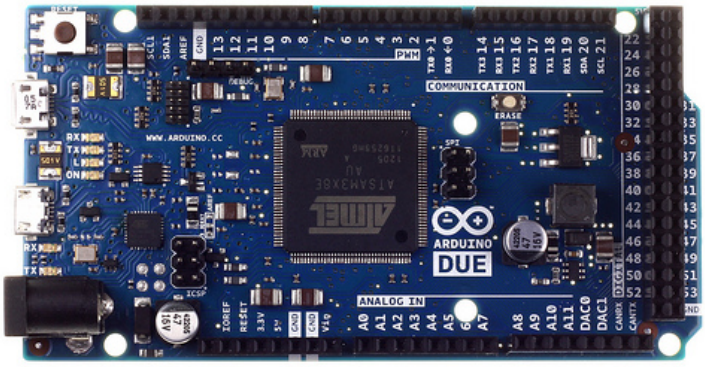
Revise History		
Version	**Date**	**Description**
1.0	MAY 07, 2020	Initial Release

第 1 章 RA8889_Lite 介绍

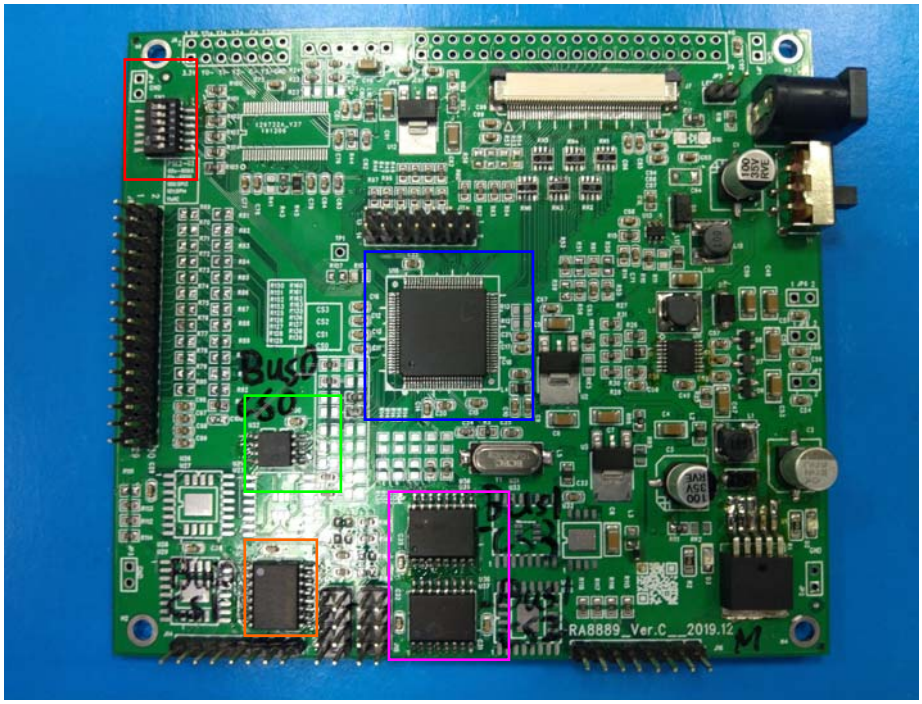
RA8889_Lite 是基于 Arudino Due 开发板连接 RA8889 驱动板源代码提供，可以协助使用者，快速的利用 Arduino Due 开发环境，驱动由瑞佑科技开发的彩色 TFT-LCD 控制器 RA8889。

硬件需求

1.Arduino Due 开发板



2.RA8889 驱动板(板载 SPI FLASH，集通字库 IC)



RA8889 Chip

- 选择 Host SPI 4 interface
- Serial Flash ROM for DMA function Bus0 xnsfcs1
- 集通字库(Genitop Font ROM) on Bus0 xnsfcs0
- Serial Flash ROM for IDEC function JPEG/AVI decode on Bus1 xnsfcs2/xnsfcs3

软件需求

Arduino IDE 1.5.7 http://arduino.cc/en/Main/Software
RA8889 Image_Tool_0.1.0.0 www.raio.com.tw

RA8889_Lite 特点

RA8889_Lite 提供 RA8889 TFTLCD 控制器主要的内建功能的应用接口(API)函数，本文所有的演示皆是基于 Arduino Due 开发板 SPI 界面，驱动 RA8889 显示 24BPP 色深的图像到 TFT-LCD。演示的特点包含下列：

初始化(Initialization)

RA8889 初始化流程。

内存规划与窗口(Memory configuration & Window)

说明如何规划 RA8889 内存内存(SDRAM)和各窗口之间的关系与设定。

图像(Graphic)

RA8889 绘图模式, Arduino Due 写入彩色图像数据。
RA8889 绘图模式, Arduino Due 写入使用者自建的 ASCII code 字型文字。

文字(Text)

RA8889 文字模式，Arduino Due 透过 RA8889 文字功能写入 RA8889 内建 ASCII 字型，并演示 RA8889 字体放大功能。
搭配支持的集通字库显示 ASCII code，BIG5，GB2312 等字型。

几何绘图(Geometric Draw)

RA8889 绘图模式，Arduino Due 透过 RA8889 绘图引擎绘制线，矩形，矩形填满，圆角矩形，圆角矩形填满，三角形，三角形填满，,圆形，圆形填满，椭圆形，椭圆型填满。

BTE

RA8889 绘图模式，Arduino Due 透过 RA8889 BTE 引擎演示：

BTE 内存复制。
BTE 内存 ROP 逻辑运算与复制。
BTE 内存复制与透明色。
Arduino Due 透过 BTE 引擎执行内存写入与 ROP 逻辑运算。
Arduino Due 透过 BTE 引擎执行内存写入与透明色。
Arduino Due 透过 BTE 引擎执行内存写入与颜色扩充。
Arduino Due 透过 BTE 引擎执行内存写入与颜色扩充和透明色。

DMA

RA8889 绘图模式，透过 RA8889 DMA 功能直接读取 Serial Flash 内部图像数据(bitmap format)并写入到 RA8889 内存内存。

IDEC

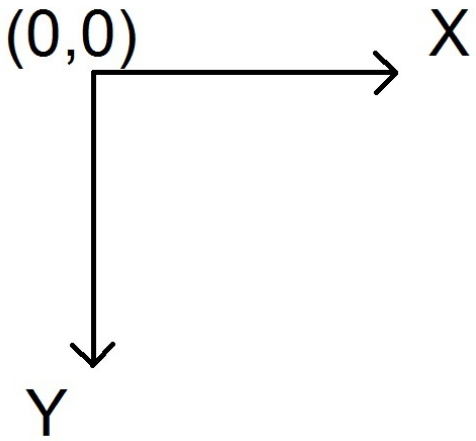
RA8889 绘图模式，透过 RA8889 IDEC 功能直接读取 Serial Flash 内部图像数据**(JPEG/AVI format)**并透过媒体译码单元**(MDU)**译码后写入到 RA8889 内存内存。

PWM

RA8889 PWM 初始设定与频率计算,责任周期规划.(需示波器量测)。

注：

1.本文件的显示坐标系统：



2.本文使用的显示屏分辨率为 800*600，如需其它分辨率参考第 2 章初始化与第 3 章内存规划与窗口。

3.RA8889_Lite 使用的自定义变量型态如下：

typedef	signed char	rs8;
typedef	signed short	rs16;
typedef	signed long	rs32;
typedef	unsigned char	ru8;
typedef	unsigned short	ru16;
typedef	unsigned long	ru32;

4.接线图参考附录 A ：

Figure A-1

第 2 章 初始化(Initialization)

RA8889 初始化主要流程如下：

RA8889 硬件复位(hardware reset)

↓

RA8889 PLL 初始化

↓

RA8889 SDRAM 初始化

↓

RA8889 一般设定

↓

RA8889 TFT 时序设定

↓

RA8889 影像显示内存与窗口初始化设定

↓

RA8889 TFT 显示开启

2.1 硬件复位(hardware reset)

begin()

RA8889 硬件复位程序包含在 begin()函数内。
当 begin()函数返回值为 true，表示硬件复位成功且正确地连接 RA8889，如返回值为 false，表示连接失败，检查 Arduino SPI bus 是否有正确的连接到 RA8889 驱动板。

2.2 PLL 初始化

ra8889PllInitial(DRAM_FREQ,CORE_FREQ,SCAN_FREQ)

函数会参考以下 Ra8889_Lite.h 内定义值，自动完成 PLL 初始化，使用者只要根据显示的需求设定下列数值。

```
#define OSC_FREQ     10   // OSC clock frequency, unit: MHz.
#define DRAM_FREQ    140   // SDRAM clock frequency, unit: MHz.
#define CORE_FREQ    120   // Core (system) clock frequency, unit: MHz.
#define SCAN_FREQ    35   // Panel Scan clock frequency, unit: MHz.
```

定义	说明
OSC_FREQ	连接到 RA8889 的晶体振荡器频率,建议为 10MHz
DRAM_FREQ	SDRAM 存取频率,建议 100~160MHz
CORE_FREQ	RA8889 系统核心频率,建议 100~130MHz
SCAN_FREQ	TFT 显示驱动频率 PCLK,参考 LCD SPEC 指定的 PCLK 频率需求

注： DRAM_FREQ >= CORE_FREQ
CORE_FREQ >= 2 * SCAN_FREQ

2.3 SDRAM 初始化

RA8889 内建 128Mbit(16MByte) SDRAM 做为图像操作与显示内存。

ra8889SdramInitial()

函数会参考 Ra8889_Lite.h 内#define DRAM_FREQ 定义值，自动执行 SDRAM 初始化。

2.4 一般设定

以下寄存器在初始化期间设定，参考 RA8889 规格书与 RA8889_Lite.h 寄存器各 bit 定义,根据你的需求设定下列几项。

```
lcdRegWrite(RA8889_CCR);//01h
lcdDataWrite(RA8889_PLL_ENABLE<<7|RA8889_WAIT_NO_MASK<<6|RA8889_KEY_SCAN_DISABLE<<5|RA8889_TFT_OUTPUT24<<3|RA8889_I2C_MASTER_DISABLE<<2|RA8889_SERIAL_IF_ENABLE<<1|RA8889_HOST_DATA_BUS_SERIAL);

lcdRegWrite(RA8889_MACR);//02h
```

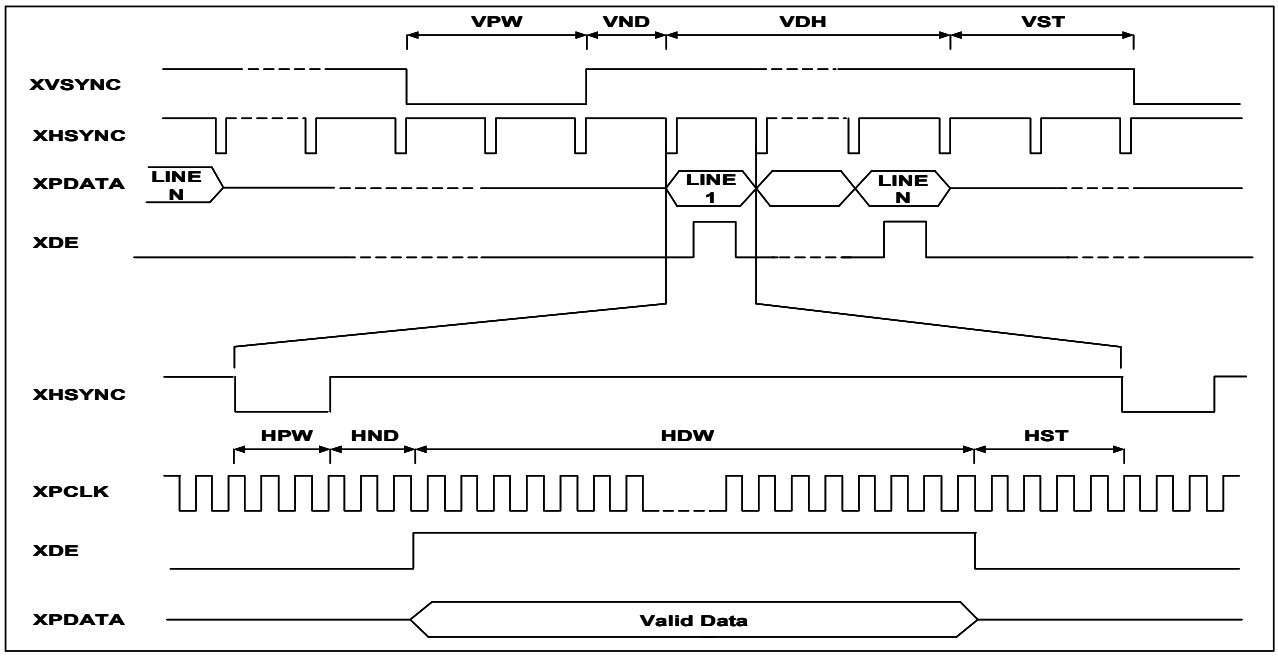
```
lcdDataWrite(RA8889_DIRECT_WRITE<<6|RA8889_READ_MEMORY_LRTB<<4|RA8889_W
RITE_MEMORY_LRTB<<1);
lcdRegWrite(RA8889_ICR);//03h
lcdDataWrite(RA8889_GRAPHIC_MODE<<2|RA8889_MEMORY_SELECT_IMAGE);

#ifdef COLOR_DEPTH_16BPP
lcdRegWrite(RA8889_MPWCTR);//10h
lcdDataWrite(RA8889_PIP1_WINDOW_DISABLE<<7|RA8889_PIP2_WINDOW_DISABLE<<6
|RA8889_SELECT_CONFIG_PIP1<<4|RA8889_IMAGE_COLOCR_DEPTH_16BPP<<2|TFT_
MODE);
lcdRegWrite(RA8889_PIPCDEP);//11h
lcdDataWrite(RA8889_PIP1_COLOR_DEPTH_16BPP<<2|RA8889_PIP2_COLOR_DEPTH_1
6BPP);
lcdRegWrite(RA8889_AW_COLOR);//5Eh
lcdDataWrite(RA8889_CANVAS_BLOCK_MODE<<2|RA8889_CANVAS_COLOR_DEPTH_16
BPP);
lcdRegDataWrite(RA8889_BTE_COLR,RA8889_S0_COLOR_DEPTH_16BPP<<5|RA8889_S
1_COLOR_DEPTH_16BPP<<2|RA8889_S0_COLOR_DEPTH_16BPP);//92h
  #endif

#ifdef COLOR_DEPTH_24BPP
lcdRegWrite(RA8889_MPWCTR);//10h
lcdDataWrite(RA8889_PIP1_WINDOW_DISABLE<<7|RA8889_PIP2_WINDOW_DISABLE<<6
|RA8889_SELECT_CONFIG_PIP1<<4|RA8889_IMAGE_COLOCR_DEPTH_24BPP<<2|TFT_
MODE);
lcdRegWrite(RA8889_PIPCDEP);//11h

lcdDataWrite(RA8889_PIP1_COLOR_DEPTH_24BPP<<2|RA8889_PIP2_COLOR_DEPTH_2
4BPP);
lcdRegWrite(RA8889_AW_COLOR);//5Eh
lcdDataWrite(RA8889_CANVAS_BLOCK_MODE<<2|RA8889_CANVAS_COLOR_DEPTH_24
BPP);
lcdRegDataWrite(RA8889_BTE_COLR,RA8889_S0_COLOR_DEPTH_24BPP<<5|RA8889_S
1_COLOR_DEPTH_24BPP<<2|RA8889_S0_COLOR_DEPTH_24BPP);//92h
  #endif
```

2.5 TFT 时序初始化设定

参考以下 Ra8889_Lite.h 内定义值，使用者需参考 LCD 规格书设定以下 TFT 时序数值。

```
#define TFT_MODE     0    //0:SYNC_mode(SYNC+DE mode), 1: DE mode
//if sync only mode do not connect DE signal or set XDE_INV to 1
#define XHSYNC_INV    0 // 0:no inversion, 1:inversion
#define XVSYNC_INV    0 // 0:no inversion, 1:inversion
#define XDE_INV       0 // 0:no inversion, 1:inversion
#define XPCLK_INV     1    // 0:no inversion, 1:inversion
#define HPW           8    //
#define HND           38
#define HDW           800
#define HST           16
#define VPW           8
#define VND           15
#define VDH           600
#define VST           12
```

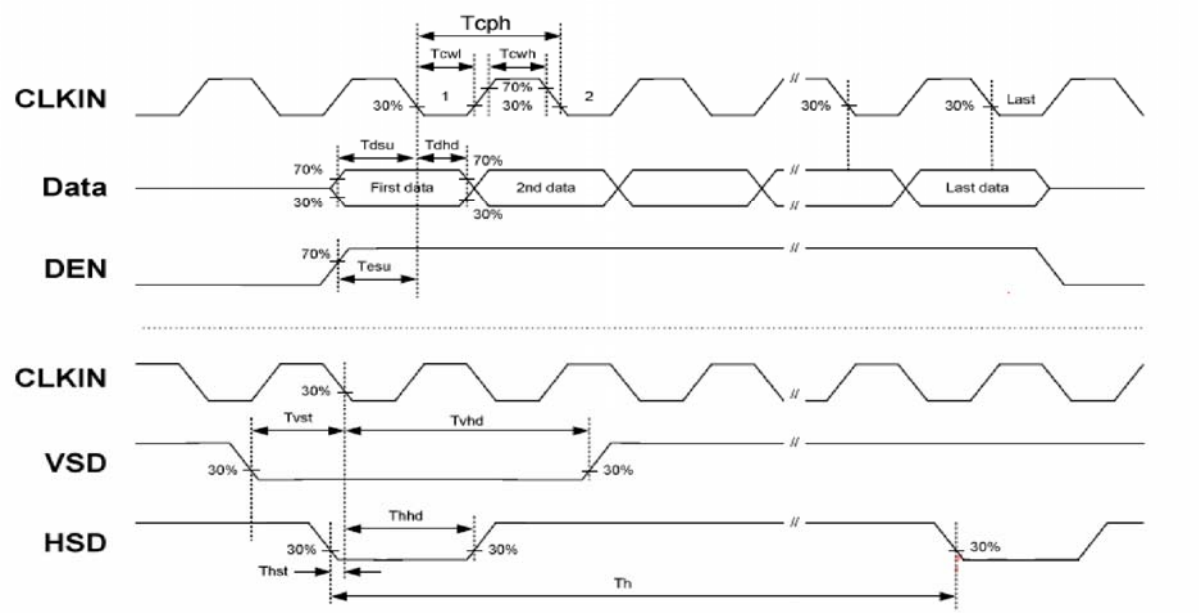
RA8889 输出时序参考



本文选用之 **TFT LCD 为 AT080TN52,要求的 TFT** 时序如下图

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Horizontal Display Area	thd	-	800	-	DCLK	
DCLK Frequency	fclk	-	40	50	MHz	
One Horizontal Line	th	862	1056	1200	DCLK	
HS pulse width	thpw	1	-	40	DCLK	
HS Back Porch(Blanking)	thb	46	46	46	DCLK	
HS Front Porch	thfp	16	210	354	DCLK	

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Vertical Display Area	tvd	-	600	-	TH	
VS period time	tv	624	635	700	TH	
VS pulse width	tvpw	1	-	20	TH	
VS Back Porch(Blanking)	tvb	23	23	23	TH	
VS Front Porch	tvfp	1	12	77	TH	



TFT 时序初始化设定程序：

```
lcdRegWrite(RA8889_DPCR);//12h
lcdDataWrite(XPCLK_INV<<7|RA8889_DISPLAY_OFF<<6|RA8889_OUTPUT_RGB);

lcdRegWrite(RA8889_PCSR);//13h
lcdDataWrite(XHSYNC_INV<<7|XVSYNC_INV<<6|XDE_INV<<5);

lcdHorizontalWidthVerticalHeight(HDW,VDH);
lcdHorizontalNonDisplay(HND);
lcdHsyncStartPosition(HST);
lcdHsyncPulseWidth(HPW);
lcdVerticalNonDisplay(VND);
lcdVsyncStartPosition(VST);
lcdVsyncPulseWidth(VPW);
```

2.6 影像显示内存初始化设定

参考以下 Ra8889_Lite.h 内定义值，使用者需定义下列数值：

```
//定义显示屏分辨率宽高
#define SCREEN_WIDTH 800
#define SCREEN_HEIGHT 600

//使用者影像内存缓存页面规划
//最大页面数量取决于 SDRAM 的容量还有该页使用的色深，宽度，高度。
//例如 SDRAM 容量= 16Mbyte
//page_size = 800*600*3byte(24bpp) = 1440000byte
//maximum number = 16/1.44 = 11.11
//所以最多可以规划 11 页给显示与应用
//本文则规划 10 个页给显示与应用如下列，每个页的尺寸都与显示尺寸相同为 800*600，色深
//24bpp，是垂直方向的多页缓存应用。

#define PAGE1_START_ADDR    0
#define PAGE2_START_ADDR    800*600*3
#define PAGE3_START_ADDR    800*600*3*2
#define PAGE4_START_ADDR    800*600*3*3
#define PAGE5_START_ADDR    800*600*3*4
```

```
#define PAGE6_START_ADDR    800*600*3*5
#define PAGE7_START_ADDR    800*600*3*6
#define PAGE8_START_ADDR    800*600*3*7
#define PAGE9_START_ADDR    800*600*3*8
#define PAGE10_START_ADDR    800*600*3*9
```

窗口初始化程序：

```
displayImageStartAddress(PAGE1_START_ADDR);
displayImageWidth(SCREEN_WIDTH);
displayWindowStartXY(0,0);
canvasImageStartAddress(PAGE1_START_ADDR);
canvasImageWidth(SCREEN_WIDTH);
activeWindowXY(0,0);
activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
```

2.7 TFT 显示开启

RA8889 初始化设定完成后，通常先对显示内存执行写入图像数据，然后再将显示开启；开启后，RA8889 TFT LCD 时序控制器，将会自动提取影像显示内存的显示窗口区块内的影像数据，并且输出到 LCD 显示。

displayOn()

描述：

显示开启或关闭.

函数原型：

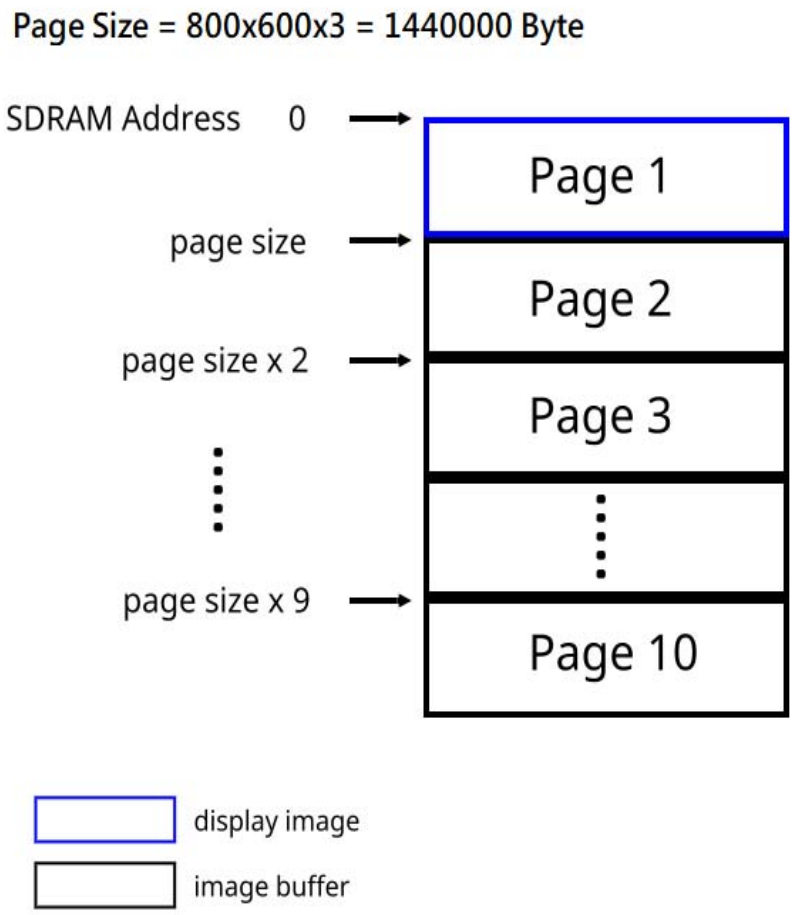
void displayOn(boolean on);

参数	说明
on	= true 显示开启 = false 显示关闭

第 3 章 内存规划与窗口(Memory Configuration & Window)

在本文中，内存被分配成 10 页，第 1 页分配给影像显示内存，其它页做为图像缓存，例如图像更新到第 2 页，然后使用 BTE memory copy 功能，将该页数据复制到第 1 页影像显示内存，这方法可以避免直接对影像显示内存，执行图形更新而造成画面显示的闪动，迭加效果。

内存规划图标：



内存和窗口相关函数如下表：

函数	说明
displayImageStartAddress()	设定影像显示内存的起始地址
displayImageWidth()	设定影像显示内存的宽度
displayWindowStartXY()	设定显示窗口在影像显示内存的左上角起始点
canvasImageStartAddress()	设定画布影像内存的起始地址
canvasImageWidth()	设定画布影像内存的宽度
activeWindowXY()	设定活动窗口在画布上的左上角起始点
activeWindowWH()	设定活动窗口宽高

displayImageStartAddress()

描述：
设定影像显示内存的起始地址。

函数原型：
void displayImageStartAddress(ru32 addr);

参数	说明
addr	影像显示内存的起始位置

附注和范例：
影像显示内存为显示窗口的数据来源内存，建议设定为 0。在本文中，内存规划为 10 个页面，第 1 个页面则分配给影像显示内存，所以初始化阶段设定如下：

displayImageStartAddress(PAGE1_START_ADDR);

displayImageWidth()

描述：
设定影像显示内存的宽度。

函数原型：
void displayImageWidth(ru16 width);

参数	说明
width	影像显示内存的宽度

附注和范例：

影像显示内存的宽度设定必须与页面(画布)宽度相等。

将每个页面(画布)宽度设定 800(=SCREEN_WIDTH),所以初始化设定如下：

displayImageWidth(SCREEN_WIDTH);

此函数只需要在初始化期间设定一次。

使用者也可以规划页面(画布)宽度 > SCREEN_WIDTH

例如：

//规划影像显示页面(画布)起始点为内存地址 0，宽度为 1600，高度为 600

displayImageStartAddress(0);

displayImageWidth(1600);

//则下一个页面的内存地址起始点 = 1600*600*3(byte)

displayWindowStartXY()

描述：

设定显示窗口在影像显示内存的左上角起始点。

函数原型：

void displayWindowStartXY(ru16 x0,ru16 y0);

参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标

附注和范例：

显示窗口的宽度和高度是参考 TFT 时序设定 HDW 和 VDH，使用者只要设定显示窗口位于影像显示内存左上角起始点。

设定如下：

displayWindowStartXY(0,0);

当影像显示内存页(画布)的宽度与高度 > LCD 分辨率宽高，X 轴坐标与 Y 轴坐标是可以改变的，X 轴的最小偏移量为 4 的倍数，Y 轴的最小偏移量为 1。

显示窗口与当前影像显示内存为子父关系，显示窗口(子)永远依附在当前指定影像显示内存(父)。

显示窗口的内容会由 RA8889 TFT 时序控制器输出到 LCD 上显示，当设定 displayOn(true)。

canvasImageStartAddress()

描述：
设定画布影像内存的起始位置

函数原型：
void canvasImageStartAddress(ru32 addr);

参数	说明
addr	画布影像内存的起始位置

canvasImageWidth()

描述：
设定画布影像内存的宽度

函数原型：
void canvasImageWidth(ru16 width);

参数	说明
width	画布影像内存的宽度

附注和范例：
图像(Graphic)，文字(Text)，绘图(Draw)，DMA，IDEC 等操作，都必须在当前画布的活动窗口(active window)区块内执行，本文中内存规划为 10 个页，每一个页都可以指定为当前画布，例如：

```
//指定第 1 页为当前画布
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);

//指定第 2 页为当前画布
ra8889lite.canvasImageStartAddress(PAGE2_START_ADDR);

//指定第 3 页为当前画布
ra8889lite.canvasImageStartAddress(PAGE3_START_ADDR);
```

activeWindowXY()

描述：
设定活动窗口在画布上的左上角起始点

函数原型：
void activeWindowXY(ru16 x0,ru16 y0);

参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标

activeWindowWH()

描述：
设定活动窗口区块宽高

函数原型：
void activeWindowWH(ru16 width,ru16 height);

参数	说明
width	活动窗口区块宽
height	活动窗口区块高

附注和范例：
图像(Graphic)，文字(Text)，绘图(Draw)，DMA，IDEC 等操作，都必须在当前画布的活动窗口(active window)区块内执行。
活动窗口与当前画布为子与父关系，活动窗口为(子)永远依附在当前画布(父) 。
活动窗口设定必须在当前画布区域内。

第 4 章 图像(Graphic)

函数	说明

graphicMode()	切换图形模式或文字模式
setPixelCursor()	设定画素光标坐标
ramAccessPrepare()	内存存取预指令
putPixel_24bpp()	指定坐标绘制一个像素点
putPicture_24bpp()	指定坐标与宽高然后写入图像数续
putPicture_24bpp()	指定坐标与绘制图像宽高与图像数据指针(Byte 格式)

注：

参考RA8889 Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

图像数据使用 Image_Tool_V1.0 图像工具转换。

graphicMode()

描述：

切换图形模式或文字模式。

函数原型：

void graphicMode(boolean on);

参数	说明
on	= true 设定为图像模式 = false 设定为文字模式

注：

RA8889 初始化设定缺省为图形模式。

setPixelCursor()

描述：

设定画素光标坐标。

函数原型：

void setPixelCursor(ru16 x,ru16 y);

参数	说明

x	X 轴坐标
y	Y 轴坐标

ramAccessPrepare()

描述：
内存存取预指令。

函数原型：
void ramAccessPrepare(void);

附注：
内存存取前必须调用此函数。

putPixel_24bpp()

描述：
在指定坐标绘制一个像素点。

函数原型：
void putPixel_24bpp(ru16 x,ru16 y,ru32 color);

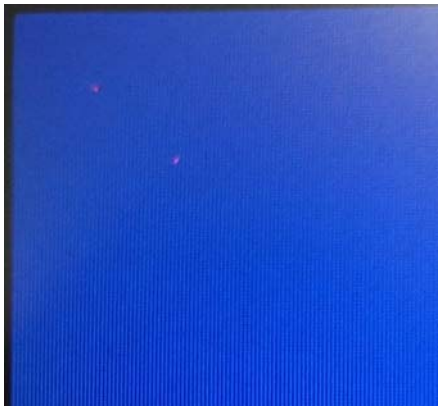
参数	说明
x	X 轴坐标
y	Y 轴坐标
color	RGB888 数据

附注和范例：

```
//清除当前画布(canvas) page1 指定的活动窗口(active window)为蓝色
  ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
  ra8889lite.canvasImageWidth(SCREEN_WIDTH);
  ra8889lite.activeWindowXY(0,0);
  ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
  ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);
```

```
//在当前画布(canvas)指定坐标位置(20,20)绘制一个红色像素点
  ra8889lite.setPixelCursor(20,20);
  ra8889lite.ramAccessPrepare();
  ra8889lite.lcdDataWrite(0x00);//RGB888 Blue data
  ra8889lite.lcdDataWrite(0x00);//RGB888 Green data
  ra8889lite.lcdDataWrite(0xff);//RGB888 Red data

//在当前画布(canvas)指定坐标位置(40,40)绘制一个紫红色像素点
  ra8889lite.putPixel_24bpp(40,40,COLOR16M_MAGENTA);
```

范例显示截图：



putPicture_24bpp()

描述：

设定左上角起始坐标与图像宽高,设定完成后，使用者可以接续写入图像数据。

函数原型：

void putPicture_24bpp(ru16 x,ru16 y,ru16 width, ru32 height);

参数	说明
x	左上角 X 轴坐标
y	左上角 Y 轴坐标
width	图像宽(水平像素尺寸)
height	图像高(垂直像素尺寸)

putPicture_24bpp()

描述：

设定坐标与图像宽高与图像数据指针(Byte 格式)，设定完后，函数会参考数据指针并自动写入图像数据到当前画布的活动窗口内指定坐标。

函数原型：

void putPicture_24bpp(ru16 x,ru16 y,ru16 width, ru16 height, const unsigned char *data);

参数	说明
x	左上角 X 轴坐标
y	左上角 Y 轴坐标
width	图像宽(水平像素尺寸)
height	图像高(垂直像素尺寸)
*data	Byte 格式图像数据指针

注：

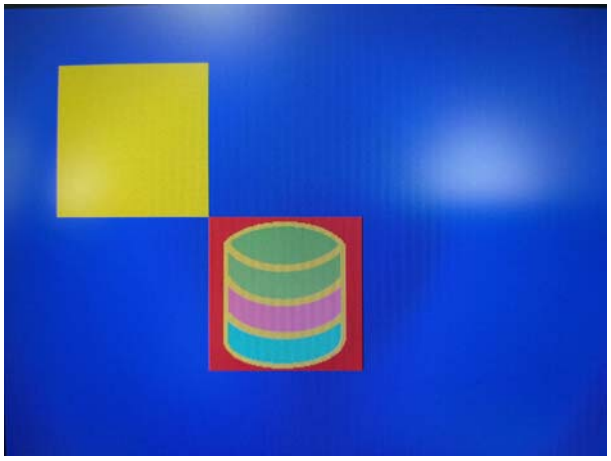
图像数据使用 Image_Tool_V1.0 图像工具转换。

附注和范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
 ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
 ra8889lite.canvasImageWidth(SCREEN_WIDTH);
 ra8889lite.activeWindowXY(0,0);
 ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
 ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//在当前画布(canvas)指定坐标位置填入 128*128 图像
ra8889lite.putPicture_24bpp(50,50,128,128);
 for(i=0;i<16384;i++)
 {
    ra8889lite.lcdDataWrite(COLOR16M_YELLOW);//RGB888 blue data
    ra8889lite.lcdDataWrite(COLOR16M_YELLOW>>8);//RGB888 blue data
    ra8889lite.lcdDataWrite(COLOR16M_YELLOW>>16);//RGB888 blue data
 }
```

//在当前画布(canvas)指定坐标位置填入 128*128 图像
ra8889lite.putPicture_24bpp(50+128+128,50+128+128,128,128, pic24bpp_1);

范例显示截图：



额外的涵数与范例

函数	说明
lcdPutChar8x12()	绘制 8x12 ASCII 字符
lcdPutString8x12()	绘制 8x12 ASCII 字符串
lcdPutChar16x24()	绘制 16x24 ASCII 字符
lcdPutString16x24()	绘制 16x24 ASCII 字符串
lcdPutChar32x48()	绘制 32x48 ASCII 字符
lcdPutString32x48()	绘制 32x48 ASCII 字符串

注：
以上函数参考 RA8889_Lite_Graphic.ino 如需要使用，增加这些函数至自己的项目内。

lcdPutChar8x12()
lcdPutChar16x24()
lcdPutChar32x48()

描述：
在当前画布的活动窗口内指定坐标绘制 ASCII 字符。

函数原型：

void lcdPutChar8x12(unsigned short x,unsigned short y,unsigned long fgcolor, unsigned long bgcolor, boolean bg_transparent, unsigned char code)

void lcdPutChar16x24(unsigned short x, unsigned short y, unsigned long fgcolor, unsigned long bgcolor, boolean bg_transparent, unsigned char code);

void lcdPutChar32x48(unsigned short x, unsigned short y, unsigned long fgcolor, unsigned long bgcolor, boolean bg_transparent, unsigned char code);

参数	说明
x	左上角 X 轴坐标
y	左上角 Y 轴坐标
fgcolor	文字前景色
bgcolor	文字背景色
bg_transparent	= ture ： 选择背景透明 ， =false ： 选择背景色
code	ASCII 码

lcdPutString8x12()
lcdPutString16x24()
lcdPutString32x48()

描述：
在当前画布与活动窗口内指定坐标位置绘制 ASCII 字符串。

函数原型：
void lcdPutString8x12(unsigned short x, unsigned short y, unsigned lonf fgcolor, unsigned long bgcolor, boolean bg_transparent, char *ptr)

void lcdPutString16x24(unsigned short x, unsigned short y, unsigned long fgcolor, unsigned long bgcolor, boolean bg_transparent, char *ptr)

void lcdPutString32x48(unsigned short x, unsigned short y, unsigned long fgcolor, unsigned long bgcolor, boolean bg_transparent, char *ptr)

参数	说明

x	左上角 X 坐标
y	左上角 Y 坐标
fgcolor	文字前景色
bgcolor	文字背景色
bg_transparent	= ture：选择背景透明 ， =false：选择背景色
*ptr	字符串或数据指针

附注和范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
  ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
  ra8889lite.canvasImageWidth(SCREEN_WIDTH);
  ra8889lite.activeWindowXY(0,0);
  ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//在当前画布的活动窗口内指定坐标位置填入 8x12 ASCII  字符串
#ifdef DEMO_ASCII_8X12
  lcdPutString8x12(0,0,0xFFFFFF,0x000000,true," !\"#$%&'()*+,-./012345678");
  lcdPutString8x12(0,12,0xFFFFFF,0x000000,true,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
  lcdPutString8x12(0,24,0xFFFFFF,0x000000,true,"RSTUVWXYZ[\\]^_`abcdefghij");
  lcdPutString8x12(0,36,0xFFFFFF,0x000000,true,"klmnopqrstuvwxyz{|}~");
#endif

//在当前画布的活动窗口内指定坐标位置填入 16x24 ASCII  字符串
#ifdef DEMO_ASCII_16X24
    lcdPutString16x24(0,48,0xFFFFFF,0x000000,true," !\"#$%&'()*+,-./012345678");
    lcdPutString16x24(0,72,0xFFFFFF,0x000000,true,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
    lcdPutString16x24(0,96,0xFFFFFF,0x000000,true,"RSTUVWXYZ[\\]^_`abcdefghij");
    lcdPutString16x24(0,120,0xFFFFFF,0x000000,true,"klmnopqrstuvwxyz{|}~");
#endif

//在当前画布的活动窗口内指定坐标位置填入 32x48 ASCII  字符串
#ifdef DEMO_ASCII_32X48
    lcdPutString32x48(0,144,0xFFFFFF,0x000000,false," !\"#$%&'()*+,-./012345678");
    lcdPutString32x48(0,192,0xFFFFFF,0x000000,false,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
    lcdPutString32x48(0,240,0xFFFFFF,0x000000,false,"RSTUVWXYZ[\\]^_`abcdefghij");
```

```
  lcdPutString32x48(0,288,0xFFFFFF,0x000000,false,"klmnopqrstuvwxyz{|}~");
#endif
```

范例显示截图:

第 5 章 文字(Text)和数值(Value)

函数	说明

textMode()	切换文字模式或图像模式
textColor()	设定文字前景色与背景色
setTextCursor()	设定文字光标坐标
setTextParameter1()	设定文字功能参数 1
setTextParameter2()	设定文字功能参数 2
genitopCharacterRomParameter()	设定集通字库文字功能参数
putString()	指定坐标位置写入字符串
putDec()	指定坐标位置写入十进制数值
putFloat()	指定坐标位置写入浮点数数值
putHex()	指定坐标位置写入十六进制数值

注：

参考RA8889 Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

textMode()

描述：

切换文字模式或图像模式。

函数原型：

void textMode (boolean on);

参数	说明
on	= true 设定为文字模式 = false 设定为图像模式

注：

建议进行文字功能操作时，设定为文字模式，操作完成后，设定返回图像模式。

textColor()

描述：

设定文字前景色与背景色。

函数原型：

void textColor(ru32 foreground_color, ru32 background_color);

参数	说明
foreground_color	文字前景色
background_color	文字背景色

setTextCursor()

描述：
设定文字坐标位置。

函数原型：
void setTextCursor(ru16 x, ru16 y);

参数	说明
x	X 轴坐标
y	Y 轴坐标

setTextParameter1()

描述：
设定文字功能参数 1。

函数原型：
void setTextParameter1(ru8 source_select, ru8 size_select, ru8 iso_select);

参数	说明
source_select	RA8889_SELECT_INTERNAL_CGROM RA8889_SELECT_EXTERNAL_CGROM RA8889_SELECT_USER_DEFINED
size_select	RA8889_CHAR_HEIGHT_16 RA8889_CHAR_HEIGHT_24 RA8889_CHAR_HEIGHT_32
iso_select	RA8889_SELECT_8859_1 RA8889_SELECT_8859_2

	RA8889_SELECT_8859_4 RA8889_SELECT_8859_5

setTextParameter2()

描述：

设定文字功能参数 2。

函数原型：

void setTextParameter2(ru8 align, ru8 chroma_key, ru8 width_enlarge, ru8 height_enlarge);

参数	说明
align	RA8889_TEXT_FULL_ALIGN_DISABLE RA8889_TEXT_FULL_ALIGN_ENABLE 对齐全型字开启位
chroma_key	RA8889_TEXT_CHROMA_KEY_DISABLE RA8889_TEXT_CHROMA_KEY_ENABLE 文字背景色透明开启位
width_enlarge	RA8889_TEXT_WIDTH_ENLARGEMENT_X1 RA8889_TEXT_WIDTH_ENLARGEMENT_X2 RA8889_TEXT_WIDTH_ENLARGEMENT_X3 RA8889_TEXT_WIDTH_ENLARGEMENT_X4 文字水平放大选择
height_enlarge	RA8889_TEXT_HEIGHT_ENLARGEMENT_X1 RA8889_TEXT_HEIGHT_ENLARGEMENT_X2 RA8889_TEXT_HEIGHT_ENLARGEMENT_X3 RA8889_TEXT_HEIGHT_ENLARGEMENT_X4 文字垂直放大选择

genitopCharacterRomParameter()

描述：

设定集通字库文字功能参数。

函数原型：

void genitopCharacterRomParameter(ru8 scs_select, ru8 clk_div, ru8 rom_select, ru8 character_select, ru8 gt_width);

参数	说明
scs_select	RA8889_SERIAL_FLASH_SELECT0 RA8889_SERIAL_FLASH_SELECT1 选择使用 SPI0 或 SPI1
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4 RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 设定集通字库用 SPI clock 除频
rom_select	RA8889_GT21L16T1W RA8889_GT30L16U2W RA8889_GT30L24T3Y RA8889_GT30L24M1Z RA8889_GT30L32S4W RA8889_GT20L24F6Y RA8889_GT21L24S1W 选择集通字库
character_select	RA8889_GB2312 RA8889_GB12345_GB18030 RA8889_BIG5 RA8889_ASCII RA8889_UNICODE RA8889_UNI_JAPANESE RA8889_JIS0208 RA8889_LATIN_GREEK_CYRILLIC_ARABIC_THAI_HEBREW RA8889_ISO_8859_1_AND_ASCII RA8889_ISO_8859_2_AND_ASCII RA8889_ISO_8859_3_AND_ASCII RA8889_ISO_8859_4_AND_ASCII RA8889_ISO_8859_5_AND_ASCII RA8889_ISO_8859_7_AND_ASCII RA8889_ISO_8859_8_AND_ASCII RA8889_ISO_8859_9_AND_ASCII

	RA8889_ISO_8859_10_AND_ASCII RA8889_ISO_8859_11_AND_ASCII RA8889_ISO_8859_13_AND_ASCII RA8889_ISO_8859_14_AND_ASCII RA8889_ISO_8859_15_AND_ASCII RA8889_ISO_8859_16_AND_ASCII 选择字型译码器
gt_width	RA8889_GT_FIXED_WIDTH RA8889_GT_VARIABLE_WIDTH_ARIAL RA8889_GT_VARIABLE_FIXED_WIDTH_ROMAN RA8889_GT_BOLD 选择字体

注：

建议使用 serial IF0(xnsfcs0)连接到集通字库，serial IF1(xnsfcs1)连接到一般的 serial flash。RA8889 支持多个集通字库型号，详细参考 RA8889 DataSheet。

putString()

描述：

在当前画布的活动窗口内的指定坐标位置写入字符串。

函数原型：

void putString(ru16 x0, ru16 y0, char *str);

参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标
*str	字符串或数据指针

范例：

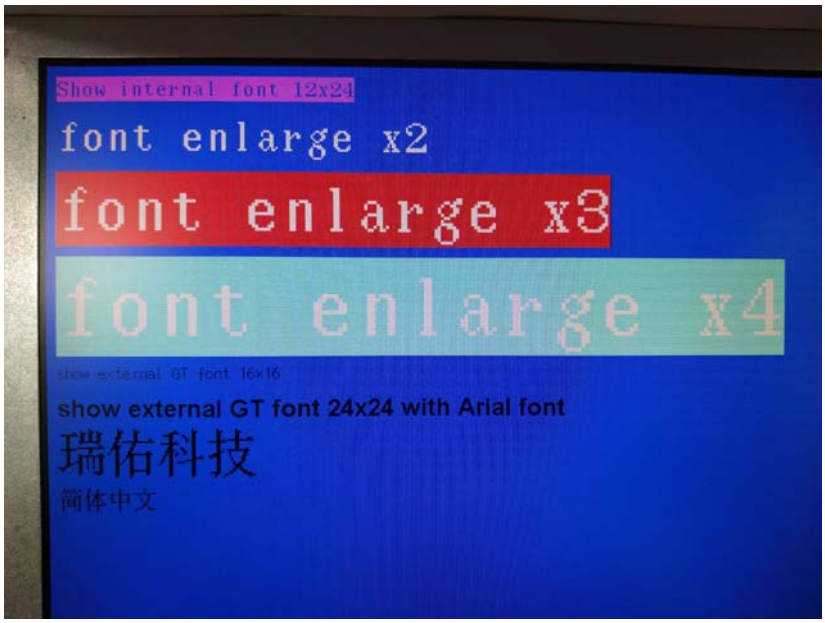
```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
  ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
  ra8889lite.canvasImageWidth(SCREEN_WIDTH);
  ra8889lite.activeWindowXY(0,0);
  ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
  ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);
```

```
//设定文字功能参数
//设定文字颜色
//在指定坐标位置显示内建字型 12x24 ASCII 字符串
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
  HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_BLUE,COLOR16M_MAGENTA);
ra8889lite.putString(10,10,"Show internal font 12x24");

//设定文字功能参数
//设定文字颜色
//在指定坐标位置显示内建字型 2 倍放大
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X2,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X2);
  ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_RED);
  ra8889lite.putString(10,44,"font enlarge x2");

//设定文字功能参数
//设定文字颜色
//在指定坐标位置显示内建字型 3 倍放大
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
  ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X3,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X3);
  ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_RED);
  ra8889lite.putString(10,102,"font enlarge x3");

//设定文字功能参数
//设定文字颜色
//在指定坐标位置显示内建字型 4 倍放大
```

```
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
 HT_24,RA8889_SELECT_8859_1);//cch
 ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X4,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X4);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_LIGHTCYAN);
ra8889lite.putString(10,184,"font enlarge x4");

//设定文字功能参数
//设定集通字库参数
//设定文字颜色
//在指定坐标位置显示集通字型字符串
 ra8889lite.setTextParameter1(RA8889_SELECT_EXTERNAL_CGROM,RA8889_CHAR_HEI
GHT_16,RA8889_SELECT_8859_1);//cch
 ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X1);

ra8889lite.genitopCharacterRomParameter(RA8889_SERIAL_FLASH_SELECT0,RA8889_SP
I_DIV4,RA8889_GT30L24T3Y,RA8889_BIG5,RA8889_GT_FIXED_WIDTH);
ra8889lite.textColor(COLOR16M_BLACK,COLOR16M_RED);
ra8889lite.putString(10,290,"show external GT font 16x16");

//设定文字功能参数
//设定集通字库参数
//设定文字颜色
//在指定坐标位置显示集通字型字符串
ra8889lite.setTextParameter1(RA8889_SELECT_EXTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X2,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X2);

ra8889lite.genitopCharacterRomParameter(RA8889_SERIAL_FLASH_SELECT0,RA8889_SP
I_DIV4,RA8889_GT30L24T3Y,RA8889_BIG5,RA8889_GT_VARIABLE_WIDTH_ARIAL);
ra8889lite.putString(10,316,"show external GT font 24x24 with Arial font");
```

```
ra8889lite.putString(10,350,string1);

ra8889lite.setTextParameter1(RA8889_SELECT_EXTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X1);

ra8889lite.genitopCharacterRomParameter(RA8889_SERIAL_FLASH_SELECT0,RA8889_SP
I_DIV4,RA8889_GT30L24T3Y,RA8889_GB2312,RA8889_GT_FIXED_WIDTH);
ra8889lite.putString(10,408,string2);
```

范例显示截图：



putDec()

描述：

在当前画布的活动窗口内的指定坐标位置写入十进制数值。

函数原型：

void putDec(ru16 x0,ru16 y0,rs32 vaule,ru8 len,const char *flag);

参数	说明
x0	左上角 X 轴坐标

y0	左上角 Y 轴坐标
vaule	输入数值 -2147483648(-2^31) ~ 2147483647(2^31-1)
len	最小显示位数(1~11)
*flag	= “n” ： 显示靠右 = “-” ： 显示靠左 = “+” ： 输出正负号 = “0” ： 在开头处补 0,非补空白

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//设定文字功能参数
//设定文字颜色
//在指定坐标位置绘制内建字型 12x24 ASCII 字符串
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

//显示数值
ra8889lite.putDec(10,10,1,2,"n");
ra8889lite.putDec(10,44,2147483647,11,"n");
ra8889lite.putDec(10,78,-12345,10,"n");
ra8889lite.putDec(10,112,-2147483648,11,"n");

ra8889lite.putDec(10,146,1,2,"-");
ra8889lite.putDec(10,180,2147483647,11,"-");
ra8889lite.putDec(10,214,-12345,10,"-");
ra8889lite.putDec(10,248,-2147483648,11,"-");
```

```
ra8889lite.putDec(10,282,1,2,"+");
ra8889lite.putDec(10,316,2147483647,11,"+");
ra8889lite.putDec(10,350,-12345,10,"+");
ra8889lite.putDec(10,384,-2147483648,11,"+");

ra8889lite.putDec(10,418,1,2,"0");
ra8889lite.putDec(10,452,2147483647,11,"0");
ra8889lite.putDec(10,486,-12345,10,"0");
ra8889lite.putDec(10,520,-2147483648,11,"0");
```

范例显示截图：

putFloat()

描述：

在当前画布的活动窗口内的指定坐标位置写入浮点数数值。

函数原型：

void putFloat (ru16 x0,ru16 y0, double vaule,ru8 len, ru8 precision,const char *flag);

参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标

vaule	输入数值(3.4E-38 ~ 3.4E38)
len	最小显示位数(1~11)
precision	小数点右边精准位数(1~4 位)
*flag	= “n” ： 显示靠右 = “-” ： 显示靠左 = “+” ： 输出正负号 = “0” ： 在开头处补 0,非补空白

注：

为了得到更高得精准度,这里使用了 double。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//设定文字功能参数
//设定文字颜色
//在指定坐标位置绘制内建字型 12x24 ASCII  字符串
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

//显示数值
ra8889lite.putFloat(10,10,1.1,7,1,"n");
ra8889lite.putFloat(10,44,483647.12,11,2,"n");
ra8889lite.putFloat(10,78,-12345.123,11,3,"n");
ra8889lite.putFloat(10,112,-123456.1234,11,4,"n");

ra8889lite.putFloat(10,146,1.1234,7,1,"-");
ra8889lite.putFloat(10,180,483647.12,11,2,"-");
ra8889lite.putFloat(10,214,-12345.123,11,3,"-");
```

```
ra8889lite.putFloat(10,248,-123456.1234,11,4,"-");

ra8889lite.putFloat(10,282,1.1,7,1,"+");
ra8889lite.putFloat(10,316,483647.12,11,2,"+");
ra8889lite.putFloat(10,350,-12345.123,11,3,"+");
ra8889lite.putFloat(10,384,-123456.1234,11,4,"+");

ra8889lite.putFloat(10,418,1.1,7,1,"0");
ra8889lite.putFloat(10,452,483647.12,11,2,"0");
ra8889lite.putFloat(10,486,-12345.123,11,3,"0");
ra8889lite.putFloat(10,520,-123456.1234,11,4,"0");
```

范例显示截图：

putHex()

描述：

在当前画布的活动窗口内的指定坐标位置写入十六进制数值。

函数原型：

void putHex(ru16 x0,ru16 y0,ru32 vaule,ru8 len,const char *flag);

参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标

vaule	输入数值 0x00000000~0xffffffff
len	最小显示位数(1~10)
*flag	= “n” ： 显示靠右 = “#” ： 强制输出 0x 作为开头 = “0” ： 在开头处补 0,非补空白 = “x” ： 强制输出 0x 作为开头，补 0

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//设定文字功能参数
//设定文字颜色
//在指定坐标位置绘制内建字型 12x24 ASCII 字符串
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA
8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

//显示数值
ra8889lite.putHex(10,10,1,4,"n");
ra8889lite.putHex(10,44,255,6,"n");
ra8889lite.putHex(10,78,0xa7c8,6,"n");
ra8889lite.putHex(10,112,0xdd11ff55,10,"n");

ra8889lite.putHex(10,146,1,4,"0");
ra8889lite.putHex(10,180,255,6,"0");
ra8889lite.putHex(10,214,0xa7c8,6,"0");
ra8889lite.putHex(10,248,0xdd11ff55,10,"0");

ra8889lite.putHex(10,282,1,4,"#");
```

```
ra8889lite.putHex(10,316,255,6,"#");
ra8889lite.putHex(10,350,0xa7c8,6,"#");
ra8889lite.putHex(10,384,0xdd11ff55,10,"#");

ra8889lite.putHex(10,418,1,4,"x");
ra8889lite.putHex(10,452,255,6,"x");
ra8889lite.putHex(10,486,0xa7c8,6,"x");
ra8889lite.putHex(10,520,0xdd11ff55,10,"x");
```

范例显示截图：

第 6 章 几何绘图(Geometric Draw)

函数	说明
drawLine()	绘制线
drawSquare()	绘制矩形
drawSquareFill()	绘制矩形填满
drawCircleSquare()	绘制圆角矩形
drawCircleSquareFill()	绘制圆角矩形填满
drawTriangle()	绘制三角型
drawTriangleFill()	绘制三角型填满
drawCircle()	绘制圆形
drawCircleFill()	绘制圆形填满
drawEllipse()	绘制椭圆形
drawEllipseFill()	绘制椭圆形填满

注：

参考RA8889 Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

drawLine()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定两点绘制线。

函数原型：

void drawLine(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
color	设定颜色(RGB888)

范例：

```
ra8889lite.drawLine(40,40,159,159,COLOR16M_RED);
ra8889lite.drawLine(40,159,159,40,COLOR16M_LIGHTRED);
```

范例显示截图：

drawSquare()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定两点绘制矩形。

函数原型：

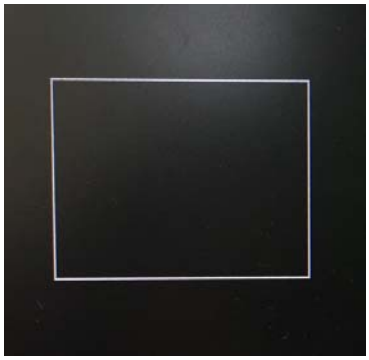
void drawSquare(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
color	设定颜色(RGB888)

范例：

ra8889lite.drawSquare(200+30, 50, 399-30, 199-50, COLOR16M_GRAYSCALE13);

范例显示截图：



drawSquareFill()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定两点绘制矩形填满。

函数原型：

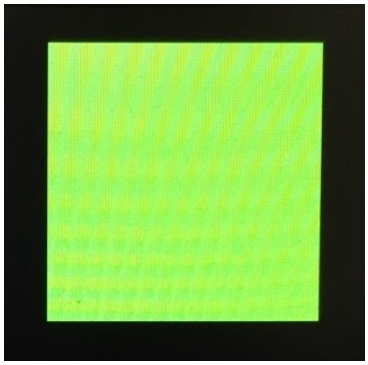
void drawSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
color	设定颜色(RGB888)

范例：

ra8889lite.drawSquareFill(420, 20, 579, 179, COLOR16M_GREEN);

范例显示截图：



drawCircleSquare()

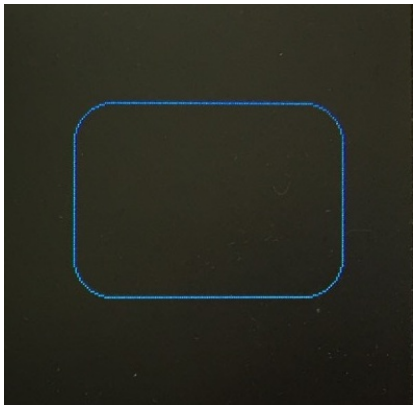
描述：
在当前画布(canvas)的活动窗口(active window)内的任意指定两点绘制圆角矩形。

函数原型：
void drawCircleSquare(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 xr, ru16 yr, ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
xr	圆角水平半径
yr	圆角垂直半径
color	设定颜色(RGB888)

范例：
ra8889lite.drawCircleSquare(600+30,0+50, 799-30, 199-50, 20, 20, COLOR16M_BLUE2);

范例显示截图：



drawCircleSquareFill()

描述：
在当前画布(canvas)的活动窗口(active window)内的任意指定两点绘制圆角矩形填满。

函数原型：

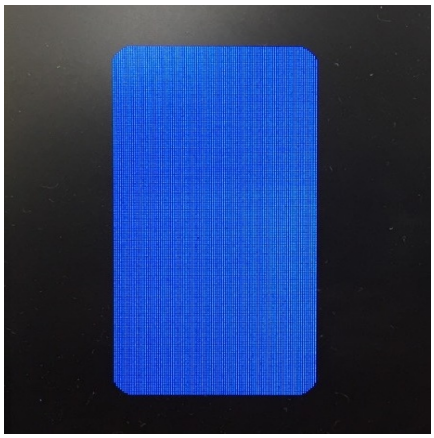
void drawCircleSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 xr, ru16 yr, ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
xr	圆角水平半径
yr	圆角垂直半径
color	设定颜色(RGB888)

范例：

ra8889lite.drawCircleSquareFill(50,200, 149, 399, 10, 10, COLOR16M_BLUE);

范例显示截图：



drawTriangle()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定三点绘制三角型。

函数原型：

void drawTriangle(ru16 x0,ru16 y0,ru16 x1,ru16 y1,ru16 x2,ru16 y2,ru32 color);

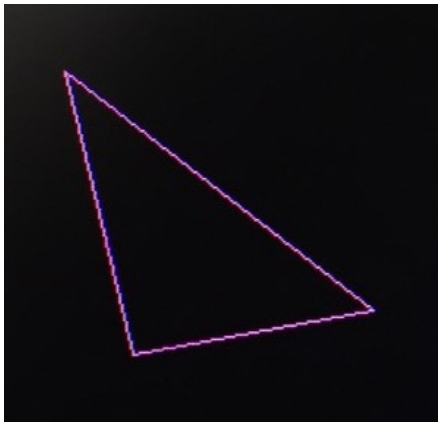
参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标

x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
x2	第 3 点 X 轴坐标
y2	第 3 点 Y 轴坐标
color	设定颜色(RGB888)

范例：

ra8889lite.drawTriangle(220,250,360,360,250,380,COLOR16M_MAGENTA);

范例显示截图：



drawTriangleFill()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定三点绘制三角型填满。

函数原型：

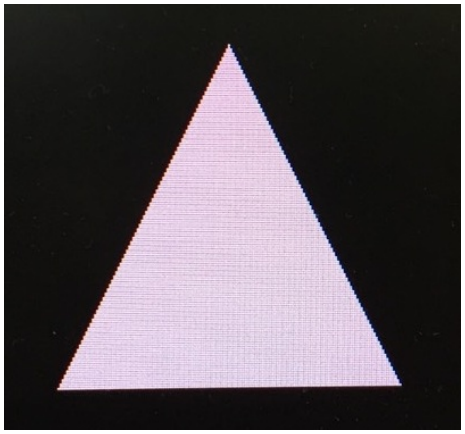
void drawTriangleFill(ru16 x0,ru16 y0,ru16 x1,ru16 y1,ru16 x2,ru16 y2,ru32 color);

参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
x2	第 3 点 X 轴坐标
y2	第 3 点 Y 轴坐标
color	设定颜色(RGB888)

范例：

ra8889lite.drawTriangleFill(500,220,580,380,420,380,COLOR16M_LIGHTMAGENTA);

范例显示截图：



drawCircle()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定中心点绘制圆。

函数原型：

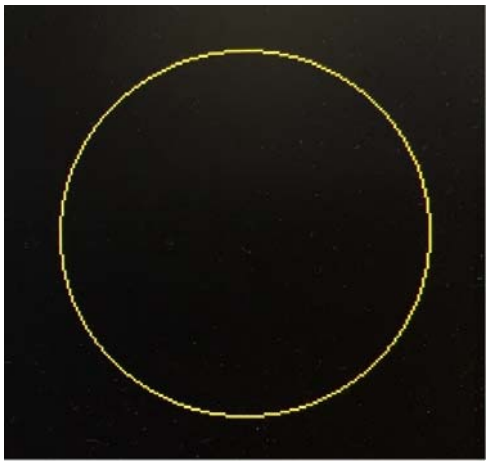
void drawCircle(ru16 x0,ru16 y0,ru16 r,ru32 color);

参数	说明
x0	中心点 X 轴坐标
y0	中心点 Y 轴坐标
r	半径
color	设定颜色(RGB888)

范例：

ra8889lite.drawCircle(700,300,80,COLOR16M_YELLOW);

范例显示截图：



drawCircleFill()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定中心点绘制圆填满。

函数原型：

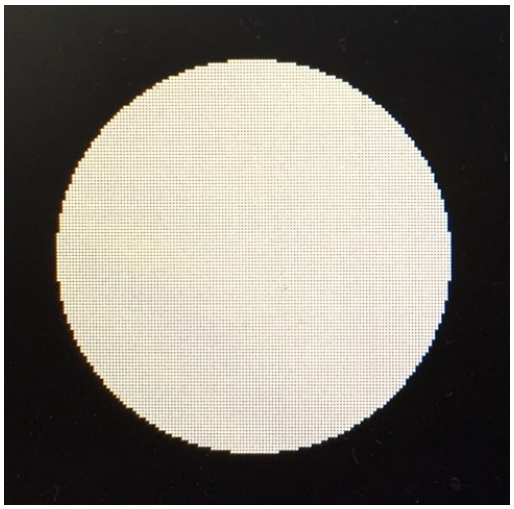
void drawCircleFill(ru16 x0,ru16 y0,ru16 r,ru32 color);

参数	说明
x0	中心点 X 轴坐标
y0	中心点 Y 轴坐标
r	半径
color	设定颜色(RGB888)

范例：

ra8889lite.drawCircleFill(100,500,60,COLOR16M_LIGHTYELLOW);

范例显示截图：



drawEllipse()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定中心点绘制椭圆。

函数原型：

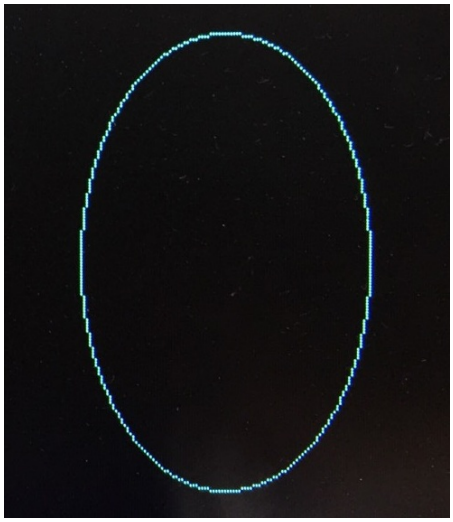
void drawEllipse(ru16 x0,ru16 y0,ru16 xr,ru16 yr,ru32 color);

参数	说明
x0	中心点 X 轴坐标
y0	中心点 Y 轴坐标
xr	水平半径
yr	垂直半径
color	设定颜色(RGB888)

范例：

ra8889lite.drawEllipse(300,500,50,80,COLOR16M_CYAN);

范例显示截图：



drawEllipseFill()

描述：

在当前画布(canvas)的活动窗口(active window)内的任意指定中心点绘制椭圆填满。

函数原型：

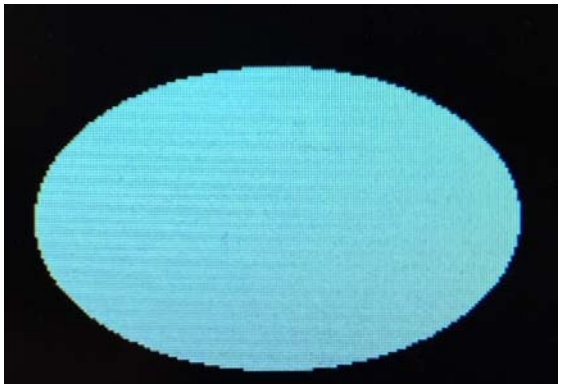
void drawEllipseFill(ru16 x0,ru16 y0,ru16 xr,ru16 yr,ru32 color);

参数	说明
x0	中心点 X 轴坐标
y0	中心点 Y 轴坐标
xr	水平半径
yr	垂直半径
color	设定颜色(RGB888)

范例：

ra8889lite.drawEllipseFill(500,500,80,50,COLOR16M_LIGHTCYAN);

范例显示截图：



第 7 章 BTE

Block Transfer Engine 是个 2D 加速功能引擎，提供了快速的内存数据传送的复制和逻辑运算，数据的透明色忽略，单色(1bpp)数据转换彩色数据的颜色扩充和颜色扩充与透明色等功能。

彩色显示的数据量是巨大的，如果 MPU 写入的速度不够快，就可以看到显示画面数据更新时像瀑布一样的扫描线，又或者需要显示丨个背景图是静态(如桌布) ，前景的文字或图像是变动的动态的效果，这样的效果就必须重新写入背景图后再写入前景的文字或图像，如果直接在当前的显示内存执行，就会因为更新背景图导致画面出现闪动，如果不重新写入背景图就直接更新前景的文字或图像，则会造成图像的迭加，所以想要得到良好的显示效果，就可以透过 BTE 功能的协助，使用者可以先将图像数据透过 MPU 接口或 DMA/IDEC 功能的方式写入到非显示区的内存，然后再使用 BTE memory copy 的功能将数据复制搬移到显示区的内存，就可以避免上述的不良效果。

颜色扩充的功能可以把 0 或 1 的数据转换成指定的彩色数据，由于 MPU 的内建 ROM 是有限制的，通常小于 512Kbyte，如果把 16bpp 图像数据转换成 1bpp 的格式存放到 MPU ROM，就可以减少 16 倍的数据量，例如使用者可能需要 64*128 尺寸的数字 0~9 用于显示，就可以将这些数字影像转换成 1bpp 格式数据，存放到 MPU ROM，然后透过 BTE color expansion 功能把数字影像写入到内存。

关于相关详细的 BTE 功能， 请参考以下的章节的说明，或者是参考 RA8889 的规格书。

函数	说明
bteMemoryCopy()	内存数据复制搬移
bteMemoryCopyWithROP()	内存数据复制搬移与逻辑运算
bteMemoryCopyWithChromaKey()	内存数据复制搬移并忽略透明色
bteMpuWriteWithROP()	MPU 写入数据与内存逻辑运算(含数据指针，Byte 格式)
bteMpuWriteWithROP()	MPU 写入数据与内存逻辑运算(含数据指针，Word 格式)
bteMpuWriteWithROP()	MPU 写入数据与内存逻辑运算
bteMpuWriteWithChromaKey()	MPU 写入数据并忽略透明色(含数据指针，Byte 格式)
bteMpuWriteWithChromaKey()	MPU 写入数据并忽略透明色(含数据指针，Word 格式)
bteMpuWriteWithChromaKey()	MPU 写入数据并忽略透明色
bteMpuWriteColorExpansion()	MPU 写入数据并执行颜色扩充(含数据指针)

bteMpuWriteColorExpansion()	MPU 写入数据并执行颜色扩充
bteMpuWriteColorExpansionWithChromaKey()	MPU 写入数据并执行颜色扩充与忽略透明色(含数据指针)
bteMpuWriteColorExpansionWithChromaKey()	MPU 写入数据并执行颜色扩充与忽略透明色

注：

参考RA8889 Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

bteMemoryCopy()

描述：

执行画布内或画布与画布间的内存数据复制与搬移。

函数原型：

void bteMemoryCopy(ru32 s0_addr, ru16 s0_image_width, ru16 s0_x, ru16 s0_y, ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 copy_width, ru16 copy_height);

参数	说明
s0_addr	来源画布的内存起始地址
s0_image_width	来源画布的图像内存宽度
s0_x	来源图像在画布上的 X 轴坐标
s0_y	来源图像在画布上的 Y 轴坐标
des_addr	目的地画布的内存起始地址
des_image_width	目的地画布的内存影像宽度
des_x	目的地图像在画布上的 X 轴坐标
des_y	目的地图像在画布上的 Y 轴坐标
copy_width	复制的图像宽度
copy_height	复制的图像高度

注：

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

pic24bpp_1.bmp

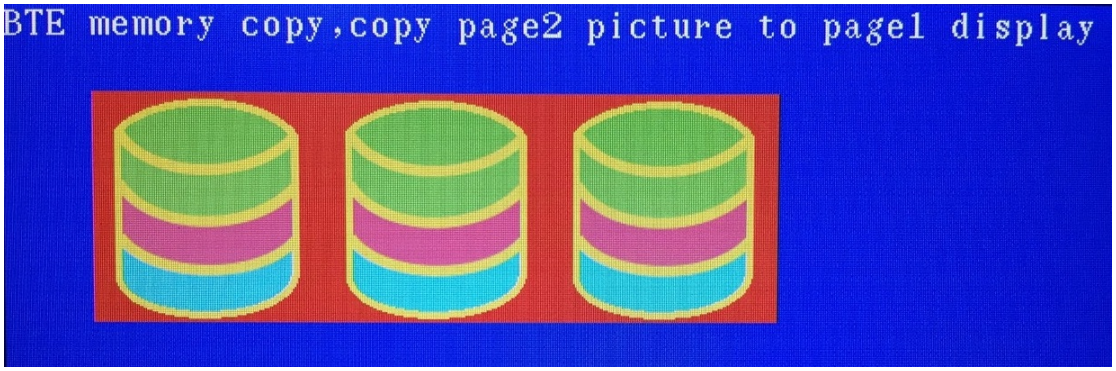
以下范例使用者必须预先转换图片文件为 24bpp 格式(pic24bpp_1.h)，并且包含到程序内。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1,
COLOR16M_BLUE);

//清除当前画布(canvas) page2 的活动窗口(active window)为红色
ra8889lite.canvasImageStartAddress(PAGE2_START_ADDR);
ra8889lite.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR16M_RED);

//写入图像数据至当前画布 page2 指定位置
ra8889lite.putPicture_24bpp(50,50,128,128, pic24bpp_1);

//写入字符串至当前画布(canvas) page1 指定位置
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_ENABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.putString(0,0,"BTE memory copy,copy page2 picture to page1 display");

//复制 page2 画布(来源)的图像数据写入 page1 画布(目的地)
```

ra8889lite.bteMemoryCopy(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH, 50,50,128,128);
ra8889lite.bteMemoryCopy(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH, (50+128),50,128,128);
ra8889lite.bteMemoryCopy(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH, (50+128+128),50,128,128);

范例显示截图：



bteMemoryCopyWithROP()

描述：

执行画布内或画布与画布间的内存数据复制与逻辑运算搬移。

函数原型：

void bteMemoryCopy WithROP (ru32 s0_addr, ru16 s0_image_width, ru16 s0_x, ru16 s0_y, ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 copy_width, ru16 copy_height, ru8 rop_code);

参数	说明
s0_addr	来源内存起始地址
s0_image_width	来源内存影像宽度
s0_x	来源 X 轴坐标位置
s0_y	来源 Y 轴坐标位置
des_addr	目的地内存起始地址
des_image_width	目的地内存影像宽度
des_x	目的地 X 轴坐标位置
des_y	目的地 Y 轴坐标位置
copy_width	复制的图像宽度

copy_height	复制的图像高度
rop_code	逻辑运算选择码 RA8889_BTE_ROP_CODE_0 (Blackness) RA8889_BTE_ROP_CODE_1 ~S0・~S1 or ~ (S0+S1) RA8889_BTE_ROP_CODE_2 ~S0・S1 RA8889_BTE_ROP_CODE_3 ~S0 RA8889_BTE_ROP_CODE_4 S0・~S1 RA8889_BTE_ROP_CODE_5 ~S1 RA8889_BTE_ROP_CODE_6 S0^S1 RA8889_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8889_BTE_ROP_CODE_8 S0・S1 RA8889_BTE_ROP_CODE_9 ~ (S0^S1) RA8889_BTE_ROP_CODE_10 S1 RA8889_BTE_ROP_CODE_11 ~S0+S1 RA8889_BTE_ROP_CODE_12 S0 RA8889_BTE_ROP_CODE_13 S0+~S1 RA8889_BTE_ROP_CODE_14 S0+S1 RA8889_BTE_ROP_CODE_15 (Whiteness)

注：

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

pic24bpp_1.bmp

以下范例使用者必须预先转换图片文件为 24bpp 格式(pic24bpp_1.h)，并且包含到程序内。

范例：

```
//写入字符串至当前画布 page1 指定位置
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.putString(0,178,"BTE memory copy with ROP, copy page2 picture to page1 display");

//复制 page2 画布(来源)的图像数据与 page1 画布(目的地)数据运算后写入 page1 画布目的地
ra8889lite.bteMemoryCopyWithROP(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1
_START_ADDR,SCREEN_WIDTH,50,228,PAGE1_START_ADDR,SCREEN_WIDTH,50,228,1
28,128,RA8889_BTE_ROP_CODE_1);
ra8889lite.bteMemoryCopyWithROP(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1
_START_ADDR,SCREEN_WIDTH,(50+128),228,PAGE1_START_ADDR,SCREEN_WIDTH,(5
0+128),228,128,128,RA8889_BTE_ROP_CODE_2);
ra8889lite.bteMemoryCopyWithROP(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1
_START_ADDR,SCREEN_WIDTH,(50+128+128),228,PAGE1_START_ADDR,SCREEN_WID
TH,(50+128+128),228,128,128,RA8889_BTE_ROP_CODE_3);
```

范例显示截图：

bteMemoryCopyWithChromaKey()

描述：

执行画布内或画布与画布间的内存数据复制搬移忽略透明色。

函数原型：

void bteMemoryCopyWithChromaKey(ru32 s0_addr, ru16 s0_image_width, ru16 s0_x, ru16 s0_y, ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 copy_width, ru16 copy_height, ru32 chromakey_color);

参数	说明
s0_addr	来源内存起始地址
s0_image_width	来源内存影像宽度
s0_x	来源 X 轴坐标位置
s0_y	来源 Y 轴坐标位置
des_addr	目的地内存起始地址
des_image_width	目的地内存影像宽度
des_x	目的地 X 轴坐标位置
des_y	目的地 Y 轴坐标位置
copy_width	复制的图像宽度
copy_height	复制的图像高度
chromakey_color	透明色数据

注：

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

pic24bpp_1.bmp

以下范例使用者必须预先转换图片文件为 24bpp 格式(pic24bpp_1.h)，并且包含到程序内。

范例：

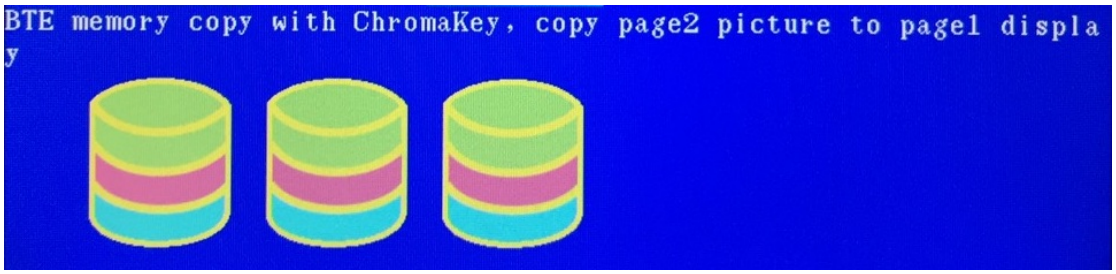
//写入字符串至当前画布 page1 指定位置

ra8889lite.putString(0,356,"BTE memory copy with ChromaKey, copy page2 picture to page1

display");

//复制 page2 画布(来源)的图像数据并忽略透明色数据写入 page1 画布目的地指定位置
ra8889lite.bteMemoryCopyWithChromakey(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH,50,406,128,128,0xff0000);
ra8889lite.bteMemoryCopyWithChromakey(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH,50+128,406,128,128,0xff0000);
ra8889lite.bteMemoryCopyWithChromakey(PAGE2_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH,50+128+128,406,128,128,0xff0000);

范例显示截图：



bteMpuWriteWithROP()

描述：
MPU(Source0)透过 BTE 与画布(Source1)指定区块执行逻辑运算后写入目的地画布(Destination)。

函数原型：
void bteMpuWriteWithROP(ru32 s1_addr,ru16 s1_image_width,ru16 s1_x,ru16 s1_y,ru32 des_addr,ru16 des_image_width,ru16 des_x,ru16 des_y,ru16 width,ru16 height,ru8 rop_code,const unsigned char *data);

void bteMpuWriteWithROP(ru32 s1_addr,ru16 s1_image_width,ru16 s1_x,ru16 s1_y,ru32 des_addr,ru16 des_image_width,ru16 des_x,ru16 des_y,ru16 width,ru16 height,ru8 rop_code,const unsigned short *data);

void bteMpuWriteWithROP(ru32 s1_addr,ru16 s1_image_width,ru16 s1_x,ru16 s1_y,ru32 des_addr,ru16 des_image_width,ru16 des_x,ru16 des_y,ru16 width,ru16 height,ru8 rop_code);

参数	说明
s1_addr	Source1 内存起始地址
s1_image_width	Source1 内存影像宽度
s1_x	Source1 X 轴坐标
s1_y	Source1 Y 轴坐标
des_addr	目的地(画布)内存起始地址
des_image_width	目的地(画布)内存影像宽度
des_x	目的地 X 轴坐标
des_y	目的地 Y 轴坐标
width	写入的图像宽度
height	写入的图像高度
rop_code	逻辑运算选择码 RA8889_BTE_ROP_CODE_0 (Blackness) RA8889_BTE_ROP_CODE_1 ~S0・~S1 or ~ (S0+S1) RA8889_BTE_ROP_CODE_2 ~S0・S1 RA8889_BTE_ROP_CODE_3 ~S0 RA8889_BTE_ROP_CODE_4 S0・~S1 RA8889_BTE_ROP_CODE_5 ~S1 RA8889_BTE_ROP_CODE_6 S0^S1 RA8889_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8889_BTE_ROP_CODE_8 S0・S1 RA8889_BTE_ROP_CODE_9 ~ (S0^S1) RA8889_BTE_ROP_CODE_10 S1 RA8889_BTE_ROP_CODE_11 ~S0+S1 RA8889_BTE_ROP_CODE_12

	S0 RA8889_BTE_ROP_CODE_13 S0+~S1 RA8889_BTE_ROP_CODE_14 S0+S1 RA8889_BTE_ROP_CODE_15 (Whiteness)
*data	数据指针(Byte 或 Word 格式)

注：

BTE 与 MPU 写入数据相关的功能 S0(Source0) = MPU 写入数据。

S1(Source1)设定可以与 Des(destination)相同。

无指针函数，调用后使用者可以继续写入影像数据。

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

Pic24bpp_1.bmp

Pic24bpp_2.bmp

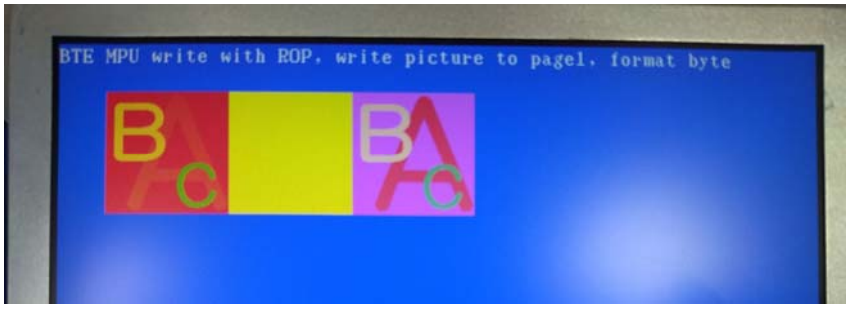
以下范例使用者必须预先转换图片文件为 24bpp 格式(pic24bpp_1.h，pic24bpp_2.h) ，并且包含到程序内。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
```

```
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_ENABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.putString(0,0,"BTE MPU write with ROP, write picture to page1, format byte");

// MPU(Source0)写入数据透过 BTE 引擎与来源画布(page1)指定位置数据运算
//后写入目的地画布(page1)
ra8889lite.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_ST
ART_ADDR,SCREEN_WIDTH,50,50,128,128,RA8889_BTE_ROP_CODE_4, pic24bpp_2);
ra8889lite.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,50,PAGE
1_START_ADDR,SCREEN_WIDTH,50+128,50,128,128,RA8889_BTE_ROP_CODE_5,
pic24bpp_2);
ra8889lite.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50+128+128,50,P
AGE1_START_ADDR,SCREEN_WIDTH,50+128+128,50,128,128,RA8889_BTE_ROP_CODE
_6, pic24bpp_2);
```

范例显示截图：



bteMpuWriteWithChromaKey()

描述：

MPU 透过 BTE 与透明色数据忽略写入数据到目的地画布。

函数原型：

void bteMpuWriteWithChromaKey(ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 width, ru16 height, ru32 chromakey_color, const unsigned char *data);

void bteMpuWriteWithChromaKey(ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 width, ru16 height, ru32 chromakey_color);

参数	说明
des_addr	目的地(画布)内存起始地址
des_image_width	目的地(画布)内存影像宽度
des_x	目的地 X 轴坐标
des_y	目的地 Y 轴坐标
width	写入的图像宽度
height	写入的图像高度
chromakey_color	透明色数据
*data	数据指针

注：

无指针函数,调用后使用者可以继续写入影像数据。

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

Pic24bpp_1.bmp

以下范例使用者必须预先转换图片文件为 24bpp 格式(pic24bpp_1.h) ， 并且包含到程序内。

范例：

//写入字符串至当前画布 page1 指定位置
ra8889lite.putString(0,356,"BTE MPU write with Chroma Key, write picture to page1, format byte");

// MPU 写入数据透过 BTE 透明色数据忽略后写入目的地画布(page1).
ra8889lite.bteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH, 50,406,128,128,0xff0000,pic24bpp_1);

范例显示截图：

bteMpuWriteColorExpansion()

描述：
MPU 写入 1bpp 图像数据透过 BTE 颜色扩展后写入至目的地画布指定区块。

函数原型：
void bteMpuWriteColorExpansion(ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 width, ru16 height, ru32 foreground_color, ru32 background_color, const unsigned char *data);

void bteMpuWriteColorExpansion(ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 width, ru16 height, ru32 foreground_color, ru32 background_color);

参数	说明
des_addr	目的地(画布)内存起始地址
des_image_width	目的地(画布)内存影像宽度
des_x	目的地 X 轴坐标
des_y	目的地 Y 轴坐标
width	写入的图像宽度

height	写入的图像高度
foreground_color	指定前景色
background_color	指定背景色
*data	数据指针(Byte 格式)

注：

无指针函数,调用后使用者可以继续写入影像数据。

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

Bw.bmp

以下范例使用者必须预先转换图片文件为 1bpp 格式(bw.h)，并且包含到程序内。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIG
HT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_ENABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8
889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.putString(0,0,"BTE MPU write with color expansion, write black and white picture
data to page1");
```

```
//MPU 写入 1bpp 图像数据透过 BTE 执行颜色扩展后写入至目的地画布
ra8889lite.bteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50,50,128,1
  28,COLOR16M_BLACK,COLOR16M_WHITE,bw);
ra8889lite.bteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,50,
128,128,COLOR16M_WHITE,COLOR16M_BLACK,bw);
ra8889lite.bteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50+128+12
8,50,128,128,COLOR16M_YELLOW,COLOR16M_CYAN,bw);
```

范例显示截图：

bteMpuWriteColorExpansionWithChromaKey()

描述：

MPU 写入 1bpp 图像数据透过 BTE 颜色扩展和忽略背景色写入至目的地画布指定位置。

函数原型：

```
void bteMpuWriteColorExpansionWithChromaKey(ru32 des_addr, ru16 des_image_width, ru16
des_x, ru16 des_y, ru16 width, ru16 height, ru32 foreground_color, ru32 background_color,
const unsigned char *data);

void bteMpuWriteColorExpansionWithChromaKey(ru32 des_addr, ru16 des_image_width, ru16
des_x, ru16 des_y, ru16 width, ru16 height, ru32 foreground_color, ru32 background_color);
```

参数	说明
des_addr	目的地(画布)内存起始地址
des_image_width	目的地(画布)内存影像宽度
des_x	目的地 X 轴坐标
des_y	目的地 Y 轴坐标
width	写入的图像宽度
height	写入的图像高度

foreground_color	指定转换前景色
background_color	指定转换背景色
*data	数据指针(Byte 格式)

注：

foreground_color 和 background_color 必须设定为不同颜色数据。

无指针函数，调用后使用者可以继续写入影像数据。

图像数据使用 Image_Tool_V1.0 图像工具转换。

参考图片：

Bw.bmp

以下范例使用者必须预先转换图片文件为 1bpp 格式(bw.h)，并且包含到程序内。

范例：

```
//写入字符串至当前画布 page1 指定位置
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);
ra8889lite.putString(0,178,"BTE MPU write with color expansion with chroma key, write black and white picture data to page1");

//MPU 写入 1bpp 图像数据透过 BTE 执行颜色扩展后忽略背景色写入至目的地
//画布指定位置
ra8889lite.bteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,50,228,128,128,COLOR16M_BLACK,COLOR16M_WHITE,bw);
ra8889lite.bteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,228,128,128,COLOR16M_WHITE,COLOR16M_BLACK,bw);
ra8889lite.bteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,50+128+128,228,128,128,COLOR16M_YELLOW,COLOR16M_BLACK,bw);
```

范例显示截图：

bteMemoryCopyWith_ARGB8888()

描述：

ARGB 图像数据透过 BTE 结合透明度的内存复制与搬移至目的地画布指定位置。
(ARGB 图像数据必须先储存于 serial flash 并且搭配使用 DMA Linear mode 功能将数据搬移至指定的非显示区画布当作 BTE Source1 层数据来源)

函数原型：

void Ra8889_Lite::bteMemoryCopyWith_ARGB8888(ru32 s1_addr,ru16 s1_image_width, ru32 des_addr,ru16 des_image_width, ru16 des_x,ru16 des_y,ru16 copy_width,ru16 copy_height);

参数	说明
S1_addr	来源内存起始地址
S1_image_width	来源内存影像宽度
des_addr	目的地内存起始地址
des_image_width	目的地内存影像宽度
des_x	目的地 X 轴坐标位置
des_y	目的地 Y 轴坐标位置
copy_width	复制的图像宽度
copy_height	复制的图像高度

注：

图像数据使用 Image_Tool_V1.0 图像工具转换。

范例：

```
//demo BTE memory with ARGB after DMA function
//clear page1
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
```

```
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

ra8889lite.spimSetSerialFlash4BytesMode(0,1);//

ra8889lite.dma_32bitAddressLinearMode(0,1,2,BINARY_INFO[Aircraft_PNG].start_addr,PAGE9_START_ADDR,BINARY_INFO[Aircraft_PNG].img_size);

ra8889lite.bteMemoryCopyWith_ARGB8888(PAGE9_START_ADDR,BINARY_INFO[Aircraft_PNG].img_width,PAGE1_START_ADDR,SCREEN_WIDTH,10,20,BINARY_INFO[Aircraft_PNG].img_width,BINARY_INFO[Aircraft_PNG].img_height);

ra8889lite.dma_32bitAddressLinearMode(0,1,2,BINARY_INFO[Car_PNG].start_addr,PAGE9_START_ADDR,BINARY_INFO[Car_PNG].img_size);

ra8889lite.bteMemoryCopyWith_ARGB8888(PAGE9_START_ADDR,BINARY_INFO[Car_PNG].img_width,PAGE1_START_ADDR,SCREEN_WIDTH,100,20,BINARY_INFO[Car_PNG].img_width,BINARY_INFO[Car_PNG].img_height);

ra8889lite.dma_32bitAddressLinearMode(0,1,2,BINARY_INFO[Lion_PNG].start_addr,PAGE9_START_ADDR,BINARY_INFO[Lion_PNG].img_size);

ra8889lite.bteMemoryCopyWith_ARGB8888(PAGE9_START_ADDR,BINARY_INFO[Lion_PNG].img_width,PAGE1_START_ADDR,SCREEN_WIDTH,150,100,BINARY_INFO[Lion_PNG].img_width,BINARY_INFO[Lion_PNG].img_height);
```

范例显示截图：

第 8 章 DMA

RA8889 提供了 DMA 功能，可以快速的读取其外扩的 serial flash 内图像数据，并写入到指定的画布区域。外扩的 serial flash 提供了让使用者存放图像数据的空间，彩色的图像数据量是庞大的，低阶的 MPU 内建的 ROM 通常小于 512Kbyte，只能存放少量图型数据，低阶 MPU 频率通常低于 50MHz，如果写入大量的数据就需要较久的时间，因此使用者可以选择使用 DMA 功能，把图像数据先烧录到 serial flash，然后利用 DMA 功能执行快速的图像存取。

函数	说明
spimSetSerialFlash4BytesMode()	设定 serial flash 为 4Bytes 模式
dma_24bitAddressBlockMode()	DMA 读取 24bit serial flash,区块模式
dma_32bitAddressBlockMode()	DMA 读取 32bit serial flash,区块模式

注：

参考RA8889 Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

图像数据使用 **Image_Tool_V1.0** 图像工具转换。

spimSetSerialFlash4BytesMode ()

描述：

当使用 32bit address serial flash 时，必须先调用此函数，设定 serial flash 为 4Bytes mode。

函数原型：

spimSetSerialFlash4BytesMode(ru8 bus_selct, ru8 scs_selct);

参数	说明
bus_select	选择 Bus0 或 Bus1
scs_select	选择 serial IF0(xnsfcs0)或 serial IF1(xnsfcs1)

注：

建议 serial IF0(xnsfcs0)连接到集通字库，serial IF1(xnsfcs1)连接到 serial flash。

dma_24bitAddressBlockMode()

描述：

由指定的 serial IF 从 24bit address serial flash 读出图像数据，并写入到指定的当前画布区块。

函数原型：

void dma_24bitAddressBlockMode(ru8 bus_selct, ru8 scs_selct, ru8 clk_div, ru16 x0, ru16 y0, ru16 width, ru16 height, ru16 picture_width, ru32 addr);

参数	说明
bus_selct	0~1 选择 Bus0 或 Bus1
scs_selct	0~3 选择 serial IF0~3 (xnsfcs0~3)
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4 RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 选择 SPI clock 除频
x0	当前画布上 X 轴坐标
y0	当前画布上 Y 轴坐标
width	DMA 区块宽度
height	DMA 区块高度
picture_width	Serial flash 内图像宽度
addr	Serial flash 内图像的起始地址

范例：

DMA 功能可以执行读取图像数据，然后写入到指定的当前画布区块。

范例：

```
//设定当前画布
//清除当前画布(page1)的活动窗口为蓝色
//DMA 读取 Serial Flash 内图像写入当前画布指定区块

ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

ra8889lite.dma_24bitAddressBlockMode(0,1,RA8889_SPI_DIV2,0,0,BINARY_INFO[p1].img_w
```

```
idth,BINARY_INFO[p1].img_height,BINARY_INFO[p1].img_width,BINARY_INFO[p1].start_add
r);
delay(1000);

ra8889lite.dma_24bitAddressBlockMode(0,1,RA8889_SPI_DIV2,0,0,BINARY_INFO[p2].img_w
idth,BINARY_INFO[p2].img_height,BINARY_INFO[p2].img_width,BINARY_INFO[p2].start_add
r);
delay(1000);

ra8889lite.dma_24bitAddressBlockMode(0,1,RA8889_SPI_DIV2,0,0,BINARY_INFO[p3].img_w
idth,BINARY_INFO[p3].img_height,BINARY_INFO[p3].img_width,BINARY_INFO[p3].start_add
r);
delay(1000);
```

范例显示截图：

dma_32bitAddressBlockMode()

描述：

由指定的 serial IF 从 32bit address serial flash 读出图像数据，并写入到指定的当前画布区块。

函数原型：

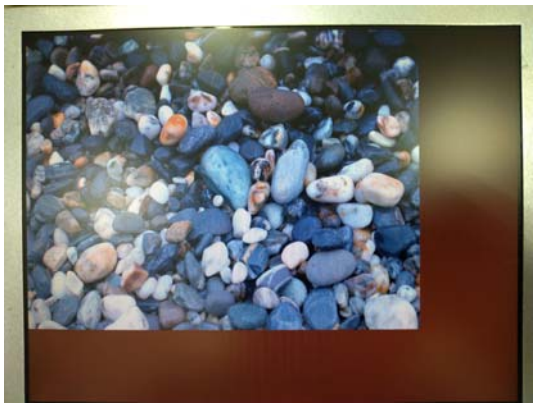
void dma_32bitAddressBlockMode(ru8 bus_selct,ru8 scs_selct, ru8 clk_div, ru16 x0, ru16 y0, ru16 width, ru16 height, ru16 picture_width, ru32 addr);

参数	说明
bus_selct	0~1 选择 Bus0 或 Bus1
scs_selct	0~3 选择 serial IF0~3 (xnsfcs0~3)
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4 RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 选择 SPI clock 除频
x0	当前画布上 X 轴坐标
y0	当前画布上 Y 轴坐标
width	DMA 区块宽度
height	DMA 区块高度
picture_width	Serial flash 内图像宽度
addr	Serial flash 内图像的起始地址

范例：

```
/*DMA demo 32bit address*/
//when using the 32bit address serial flash, must be setting serial flash to 4Bytes mode
//only needs set one times after power on
ra8889lite. spimSetSerialFlash4BytesMode (1);
while(1)
{
//设定当前画布
//清除当前画布(page1)的活动窗口为淡青色
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_LIGHTCYAN);

//DMA 读取 serial flash 内图像写入当前画布指定区块
   //demo 32bit address serial flash DMA function

ra8889lite.dma_32bitAddressBlockMode(0,1,RA8889_SPI_DIV2,30,20,BINARY_INFO[p1].img
_width,BINARY_INFO[p1].img_height,BINARY_INFO[p1].img_width,BINARY_INFO[p1].start_a
ddr);
   delay(1000);

ra8889lite.dma_32bitAddressBlockMode(0,1,RA8889_SPI_DIV2,30,20,BINARY_INFO[p2].img
_width,BINARY_INFO[p2].img_height,BINARY_INFO[p2].img_width,BINARY_INFO[p2].start_a
ddr);
   delay(1000);

ra8889lite.dma_32bitAddressBlockMode(0,1,RA8889_SPI_DIV2,30,20,BINARY_INFO[p3].img
_width,BINARY_INFO[p3].img_height,BINARY_INFO[p3].img_width,BINARY_INFO[p3].start_a
ddr);
   delay(1000);}
```

范例显示截图：



第 9 章 IDEC

RA8889 提供了 IDEC 功能，可以快速的读取其外扩的 serial flash 内部图像数据(JPEG/AVI format)并透过媒体译码单元(MDU)译码后写入到指定的画布区域。

函数	说明
spimSetSerialFlashQuadMode();	设定 serial flash 为 Quad Mode
spimSetSerialFlash4BytesMode()	设定 serial flash 为 4Bytes Mode
idec_24bitAddressQuadMode6B_24bpp_JPEG()	IDEC 读取 24bit serial flash 内 JPEG 图片数据并译码写入指定位置
idec_32bitAddressQuadMode6B_24bpp_JPEG()	IDEC 读取 32bit serial flash 内 JPEG 图片数据并译码写入指定位置
idec_24bitAddressQuadMode6B_24bpp_AVI()	IDEC 读取 24bit serial flash 内 AVI 影像数据并译码写入指定位置
idec_32bitAddressQuadMode6B_24bpp_AVI()	IDEC 读取 32bit serial flash 内 AVI 影像数据并译码写入指定位置
aviWindowOn()	AVI 显示窗口开启/关闭

注：

图像数据使用 **Image_Tool_V1.0** 图像工具转换。

建议 IDEC 功能使用 Bus1 与 serial IF2(xnsfcs2)或 serial IF3(xnsfcs3)硬件配置。

spimSetSerialFlashQuadMode()

描述：

当使用 IDEC 与 MDU 时，serial flash 必须支持 Quad mode，并调用此函数，设定 serial flash 为 Quad mode。

函数原型：

spimSetSerialFlashQuadMode(ru8 bus_select,ru8 scs_select,ru8 flash_select,ru8 data1, ru8 data2)

参数	说明
bus_select	选择 Bus0 或 Bus1
scs_select	选择 serial IF2(xnsfcs2)或 serial IF3(xnsfcs3)
flash_select	0: MXIC flash 1: Winbond flash Others: 透过 data1 和 data2 推送命令
data1	Quad Mode 命令 1
data2	Quad Mode 命令 2

spimSetSerialFlash4BytesMode ()

描述：

当使用 32bit address serial flash 时，必须先调用此函数，设定 serial flash 为 4Bytes mode。

函数原型：

spimSetSerialFlash4BytesMode(ru8 bus_selct, ru8 scs_selct);

参数	说明
bus_select	选择 Bus0 或 Bus1
scs_select	选择 serial IF2(xnsfcs2)或 serial IF3(xnsfcs3)

注：

建议 IDEC 功能使用 Bus1 与 serial IF2(xnsfcs2)或 serial IF3(xnsfcs3)硬件配置。

idec_24bitAddressQuadMode6B_24bpp_JPEG()
idec_32bitAddressQuadMode6B_24bpp_JPEG()

描述：

由指定的 serial IF 从 24bit /32bit address serial flash 读出 JPEG 图像数据透过 MDU 译码，并写入到指定的当前画布区块。

函数原型：

void idec_24bitAddressQuadMode6B_24bpp_JPEG(ru8 bus_select,ru8 scs_select,ru16 x0,ru16 y0,ru32 addr,ru32 number,ru16 des_image_width,ru32 des_start_addr);
void idec_32bitAddressQuadMode6B_24bpp_JPEG(ru8 bus_select,ru8 scs_select,ru16 x0,ru16 y0,ru32 addr,ru32 number,ru16 des_image_width,ru32 des_start_addr

参数	说明
bus_selct	0~1 选择 Bus0 或 Bus1
scs_selct	0~3 选择 serial IF0~3 (xnsfcs0~3)
x0	目的地画布上 X 轴坐标
y0	目的地画布上 Y 轴坐标
addr	Serial flash 内图像的起始地址
number	数据量
des_image_width	目的地画布宽度

des_start_addr	目的地画布起始地址

idec_24bitAddressQuadMode6B_24bpp_AVI()
idec_32bitAddressQuadMode6B_24bpp_AVI()

描述：
由指定的 serial IF 从 24bit address serial flash 读出 AVI 图像数据透过 MDU 译码，并写入到指定的当前画布区块。

函数原型：
```
void idec_24bitAddressQuadMode6B_24bpp_AVI(ru8 bus_select,ru8 scs_select,ru16 x0,ru16 y0,ru32 addr,ru32 number,ru16 width,ru16 height,ru32 shadow_buffer_addr,ru32 pip_image_addr,ru16 pip_image_width);

void idec_32bitAddressQuadMode6B_24bpp_AVI(ru8 bus_select,ru8 scs_select,ru16 x0,ru16 y0,ru32 addr,ru32 number,ru16 width,ru16 height,ru32 shadow_buffer_addr,ru32 pip_image_addr,ru16 pip_image_width);
```

参数	说明
bus_selct	0~1 选择 Bus0 或 Bus1
scs_selct	0~3 选择 serial IF0~3 (xnsfcs0~3)
x0	AVI 窗口显示 X 轴坐标
y0	AVI 窗口显示 Y 轴坐标
addr	Serial flash 内图像的起始地址
number	数据量
width	影像宽
height	影像高
shadow_buffer_addr	影子缓存内存起始地址
pip_image_addr	PIP1 影像缓存内存地址
pip_image_width	PIP1 影像缓存内存宽度

aviWindowOn()

描述：

AVI 显示窗口开启与关闭，AVI 窗口透过 PIP1 显示，开启后永远显示在最上层。

函数原型：

aviWindowOn(boolean enable);

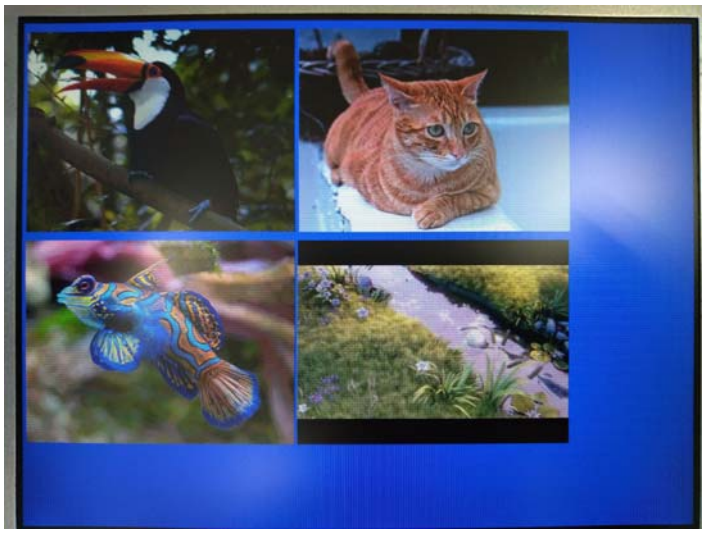
参数	说明
enable	= 1： AVI 显示窗口开启 = 0： AVI 显示窗口关闭

范例：

```
ra8889lite.spimSetSerialFlashQuadMode(1,3,0,0x00,0x00);

while(1)
{
//clear page1
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);
ra8889lite.activeWindowXY(0,0);
ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);


#ifdef COLOR_DEPTH_24BPP
ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_BLUE);

ra8889lite.idec_24bitAddressQuadMode6B_24bpp_JPEG(1,3,5,10,BINARY_INFO[bird].start
_addr,BINARY_INFO[bird].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
delay(1000);

ra8889lite.idec_24bitAddressQuadMode6B_24bpp_JPEG(1,3,330,10,BINARY_INFO[cat].st
art_addr,BINARY_INFO[cat].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
delay(1000);

ra8889lite.idec_24bitAddressQuadMode6B_24bpp_JPEG(1,3,5,260,BINARY_INFO[fish].sta
rt_addr,BINARY_INFO[fish].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
delay(1000);
```

```
    ra8889lite.idec_24bitAddressQuadMode6B_24bpp_AVI(1,3,330,260,BINARY_INFO[demo_v
    ideo320240].start_addr,BINARY_INFO[demo_video320240].img_size,320,240,
    PAGE2_START_ADDR,PAGE3_START_ADDR,SCREEN_WIDTH);
    ra8889lite.aviWindowOn(1);
    while( ra8889lite.getMediaDecodeBusyFlag());    //check busy flag until media decode not
    busy, user can run font/dma function with different sfi bus when idec function run avi decoding
    ra8889lite.aviWindowOn(0);

    #endif
}
```

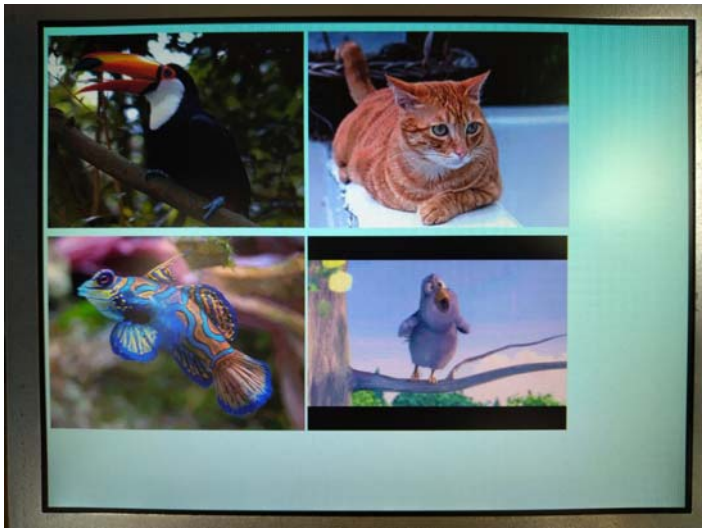
范例显示截图：



```
    /*IDEC demo 32bit address*/
    //when using the 32bit address serial flash, must be setting serial flash to 4Bytes mode and
    Quad mode for IDEC function
    //only needs set one times after power on

    ra8889lite.spimSetSerialFlashQuadMode(1,2,1,0x00,0x00);
    ra8889lite.spimSetSerialFlash4BytesMode(1,2);

    while(1)
    {
    ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
    ra8889lite.canvasImageWidth(SCREEN_WIDTH);
```

```
  ra8889lite.activeWindowXY(0,0);
  ra8889lite.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);

  #ifdef COLOR_DEPTH_24BPP
  ra8889lite.drawSquareFill(0, 0, 799, 599, COLOR16M_LIGHTCYAN);


  ra8889lite.idec_32bitAddressQuadMode6B_24bpp_JPEG(1,2,5,10,BINARY_INFO[bird].start
  _addr,BINARY_INFO[bird].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
  delay(1000);

  ra8889lite.idec_32bitAddressQuadMode6B_24bpp_JPEG(1,2,330,10,BINARY_INFO[cat].st
  art_addr,BINARY_INFO[cat].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
  delay(1000);

  ra8889lite.idec_32bitAddressQuadMode6B_24bpp_JPEG(1,2,5,260,BINARY_INFO[fish].sta
  rt_addr,BINARY_INFO[fish].img_size,SCREEN_WIDTH,PAGE1_START_ADDR);
  delay(1000);

  ra8889lite.idec_32bitAddressQuadMode6B_24bpp_AVI(1,2,330,260,BINARY_INFO[demo_v
  ideo320240].start_addr,BINARY_INFO[demo_video320240].img_size,320,240,
  PAGE2_START_ADDR,PAGE3_START_ADDR,SCREEN_WIDTH);

  ra8889lite.aviWindowOn(1);
  while( ra8889lite.getMediaDecodeBusyFlag());    //check busy flag until media decode not
  busy, user can run font/dma function with different sfi bus when idec function run avi decoding
  ra8889lite.aviWindowOn(0);
  #endif
}
```

范例显示截图：



第 10 章 PWM

函数	说明
pwm_Prescaler()	预分频器设定
pwm_ClockMuxReg()	PWM 除频器与 PWM 引脚的功能选择
pwm_Configuration()	PWM 功能设定与开启
pwm0_ClocksPerPeriod()	PWM0 每个周期频率数量设定
pwm0_Duty()	PWM0 责任周期
pwm1_ClocksPerPeriod()	PWM1 每个周期频率数量设定
pwm1_Duty()	PWM1 责任周期

注：参考**RA8889 Arduino Wire Sketch.jpg** 接线图或附录 Figure A-1。

pwm_Prescaler()

描述：

预分频器设定

函数原型：

```
void pwm_Prescaler(ru8 prescaler);
```

参数	说明
prescaler	RA8889_PRESCALER

注：

PWM0 和 PWM1 的基频 = Core_Freq(核心频率) / (Prescaler + 1)

pwm_ClockMuxReg()

描述：

PWM 除频器与 PWM 引脚的功能选择

函数原型：

```
void pwm_ClockMuxReg(ru8 pwm1_clk_div, ru8 pwm0_clk_div, ru8 xpwm1_ctrl, ru8 xpwm0_ctrl);
```

参数	说明
pwm1_clk_div	PWM1 基频除频设定

	RA8889_PWM_TIMER_DIV1 RA8889_PWM_TIMER_DIV2 RA8889_PWM_TIMER_DIV4 RA8889_PWM_TIMER_DIV8
pwm0_clk_div	PWM0 基频除频设定 RA8889_PWM_TIMER_DIV1 RA8889_PWM_TIMER_DIV2 RA8889_PWM_TIMER_DIV4 RA8889_PWM_TIMER_DIV8
xpwm1_ctrl	PWM1 引脚功能选择 RA8889_XPWM1_OUTPUT_ERROR_FLAG RA8889_XPWM1_OUTPUT_PWM_TIMER1 RA8889_XPWM1_OUTPUT_OSC_CLK
xpwm0_ctr	PWM0 引脚功能选择 RA8889_XPWM0_GPIO_C7 RA8889_XPWM0_OUTPUT_PWM_TIMER0 RA8889_XPWM0_OUTPUT_CORE_CLK

pwm_Configuration()

描述：

PWM 功能设定与开启

函数原型：

void pwm_Configuration(ru8 pwm1_inverter, ru8 pwm1_auto_reload, ru8 pwm1_start,ru8 pwm0_dead_zone, ru8 pwm0_inverter, ru8 pwm0_auto_reload,ru8 pwm0_start);

参数	说明
pwm1_inverter	PWM1 输出反向 RA8889_PWM_TIMER1_INVERTER_OFF RA8889_PWM_TIMER1_INVERTER_ON
pwm1_auto_reload	PWM1 单次输出或重复输出 RA8889_PWM_TIMER1_ONE_SHOT RA8889_PWM_TIMER1_AUTO_RELOAD
pwm1_start	PWM1 停止或开启 RA8889_PWM_TIMER1_STOP RA8889_PWM_TIMER1_START

pwm0_dead_zone	PWM0 死区功能选择不启用或启用 RA8889_PWM_TIMER0_DEAD_ZONE_DISABLE RA8889_PWM_TIMER0_DEAD_ZONE_ENABLE
pwm0_inverter	PWM0 输出反向 RA8889_PWM_TIMER0_INVERTER_OFF RA8889_PWM_TIMER0_INVERTER_ON
pwm0_auto_reload	PWM0 单次输出或重复输出 RA8889_PWM_TIMER0_ONE_SHOT RA8889_PWM_TIMER0_AUTO_RELOAD
pwm0_start	PWM0 停止或开启 RA8889_PWM_TIMER0_STOP RA8889_PWM_TIMER0_START

pwm0_ClocksPerPeriod()
pwm1_ClocksPerPeriod()
描述：
PWM0 每个周期的频率数量设定.
PWM1 每个周期的频率数量设定.

函数原型：
void pwm0_ClocksPerPeriod(ru16 clocks_per_period);
void pwm1_ClocksPerPeriod(ru16 clocks_per_period);

参数	说明
clocks_per_period	每个周期频率数量(1~65535)

注：
此设定也可以说是 PWM 分辨率的设定，例如设定值为 1000，那么责任周期(Duty cycle)可以调整的范围就是 0~1000。

pwm0_Duty()
pwm1_Duty()

描述：
PWM0 责任周期设定.
PWM1 责任周期设定.

函数原型：

void pwm0_Duty(ru16 duty);

void pwm1_Duty(ru16 duty);

参数	说明
duty	责任周期设定值

注：

责任周期 duty 范围为 clocks_per_period 设定值决定。

范例：

```
/*pwm demo please measure by oscilloscope */
ra8889lite.pwm_Prescaler(RA8889_PRESCALER); //if core_freq = 120MHz, pwm base clock =
                                            //120/(3+1) = 30MHz

ra8889lite.pwm_ClockMuxReg(RA8889_PWM_TIMER_DIV4,RA8889_PWM_TIMER_DIV4,RA
  8889_XPWM1_OUTPUT_PWM_TIMER1,RA8889_XPWM0_OUTPUT_PWM_TIMER0);
//pwm timer clock = 30 MHz /4 = 7.5MHz

ra8889lite.pwm0_ClocksPerPeriod(1024); // pwm0 = 7.5MHz/1024 = 7.3KHz
ra8889lite.pwm0_Duty(10);//pwm0 set 10/1024 duty

ra8889lite.pwm1_ClocksPerPeriod(256);    // pwm1 = 7.5MHz/256 = 29.2KHz
ra8889lite.pwm1_Duty(5); //pwm1 set 5/256 duty

ra8889lite.pwm_Configuration(RA8889_PWM_TIMER1_INVERTER_ON,RA8889_PWM_TIME
  R1_AUTO_RELOAD,RA8889_PWM_TIMER1_START,RA8889_PWM_TIMER0_DEAD_ZON
  E_DISABLE ,RA8889_PWM_TIMER0_INVERTER_ON,RA8889_PWM_TIMER0_AUTO_REL
  OAD,RA8889_PWM_TIMER0_START);
```

附录 A

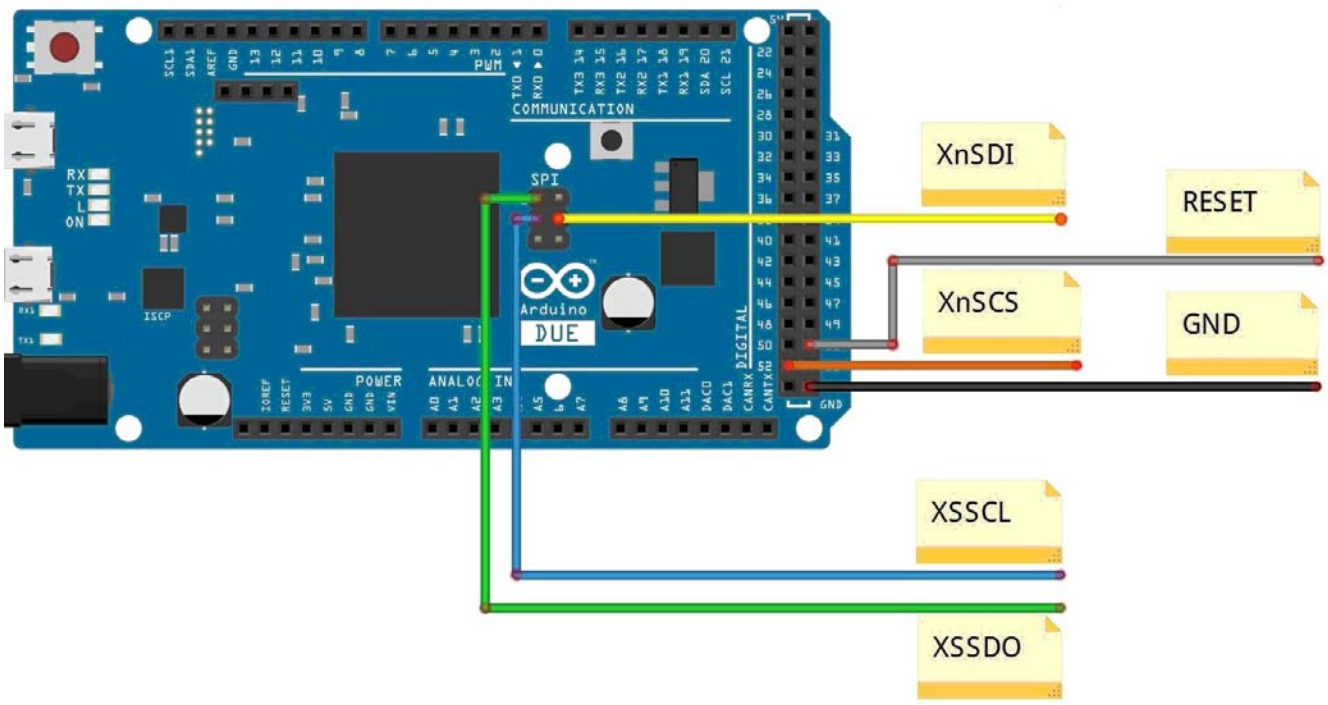


Figure A-1

全文完