RAiO

RA8871M_Lite

User Guide

Sep 15, 2017

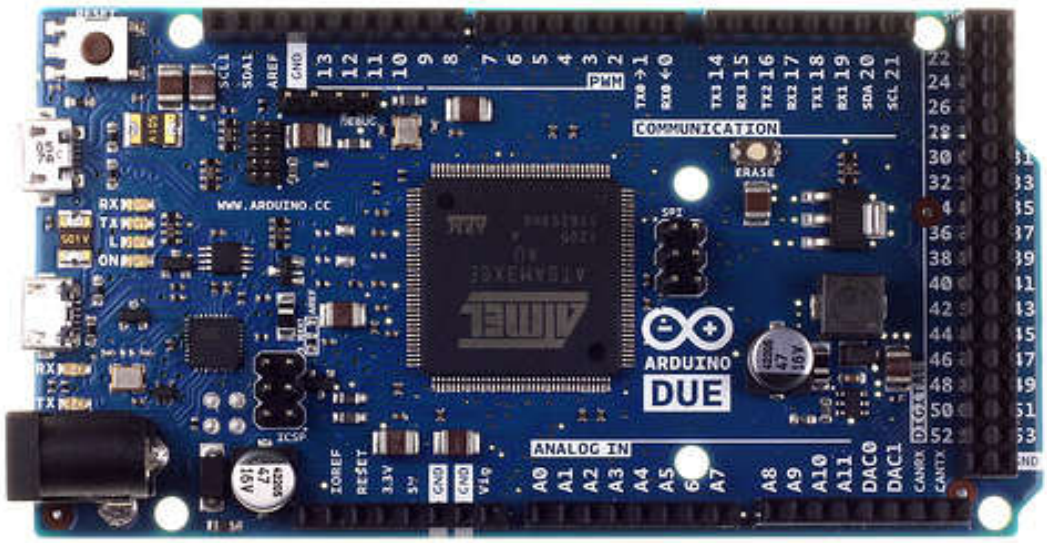
Revise History		
Version	**Date**	**Description**
1.0	2017.09.15	Initial Release

第 1 章 RA8871M_Lite 介绍

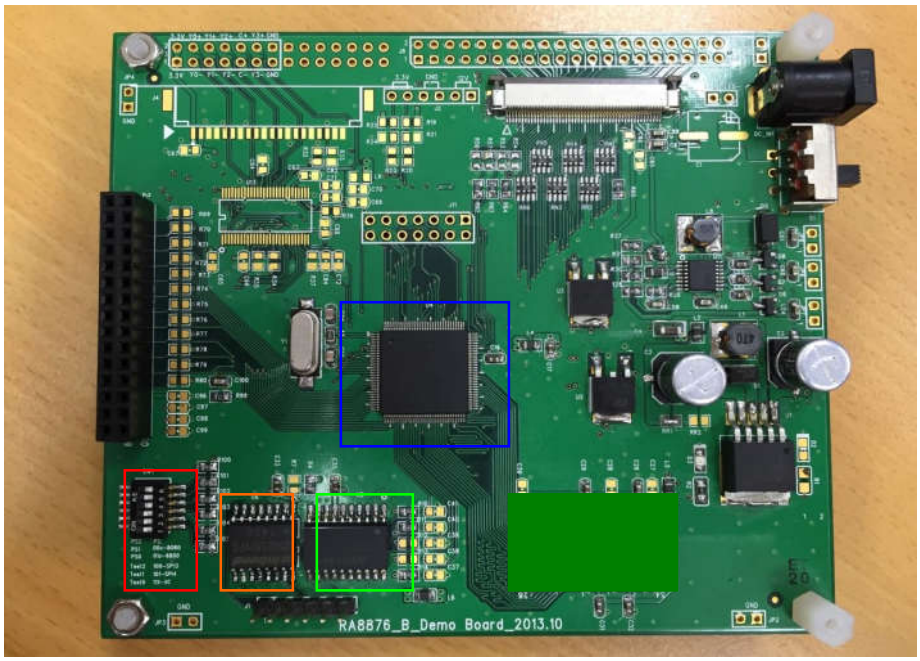
RA8871M_Lite 是基于 Arudino Due 开发板连接 RA8871M 驱动板，与 SD 卡的图像界面应用接口源代码提供，可以协助使用者，快速的利用 Arduino Due 开发环境，驱动由瑞佑科技开发的彩色 TFT-LCD 控制器 RA8871M。

硬件需求

1.Arduino Due 开发板

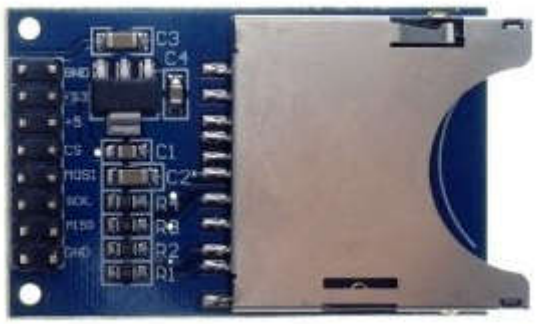


2.RA8871M 驱动板(板载 SPI FLASH，集通字库 IC)



- RA8871M Chip
- 选择 SPI interface
- Serial Flash ROM for DMA function
- 集通字库(Genitop Font ROM)

3.SD 卡转接板



4.SD 卡(容量 4GB 含以下)

软件需求

Arduino IDE 1.5.7 http://arduino.cc/en/Main/Software
Image_Tool_v1.1.0.1 www.raio.com.tw

RA8871M_Lite 特点

RA8871M_Lite 提供 RA8871M 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 RA8871M 显示 16BPP 色深的图像到 TFT-LCD。演示的特点包含下列：

初始化(Initialization)

RA8871M 初始化流程。

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

说明如何规划 RA8871M 内部存储器(Buffer RAM)和各窗口之间的关系与设定。

图像(Graphic)

RA8871M 绘图模式, Arduino Due 写入彩色图像数据。
RA8871M 绘图模式, Arduino Due 写入使用者自建的 ASCII code 字型文字。

文字(Text)

RA8871M 文字模式，Arduino Due 透过 RA8871M 文字功能写入 RA8871M 内建 ASCII 字型，并演示 RA8871M 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)

RA8871M 绘图模式，Arduino Due 透过 RA8871M 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

RA8871M 绘图模式，Arduino Due 透过 RA8871M 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。
BTE 图案填满。
BTE 图案填满与透明色。

DMA

RA8871M 绘图模式，透过 RA8871M DMA 引擎直接读取 Serial Flash 内部图像数据并写入到 RA8871M 内部存储器(Buffer RAM)。

PWM

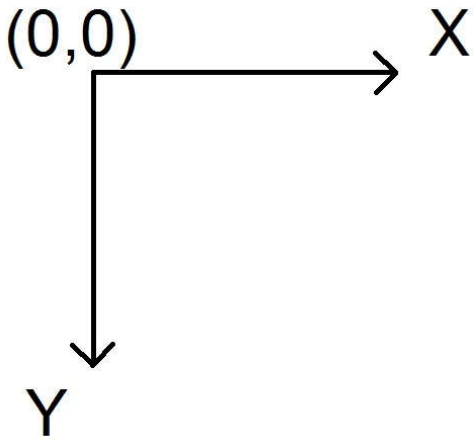
RA8871M PWM 初始设定与频率计算,责任周期规划.(需示波器量测)。

Arduino SD

Arduino Due 读取 SD 卡内图像数据，然后写入 RA8871M 内部存储器(Buffer RAM)。
Arduino Due 读取 SD 卡内图像数据，然后透过 BTE 引擎写入 RA8871M 内部存储器(Buffer RAM)。

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 480*272，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.RA8871M_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

Figure A-2

第 2 章 初始化(Initialization)

RA8871M 初始化主要流程如下：

RA8871M 硬件复位(hardware reset)

↓

RA8871M PLL 初始化

↓

RA8871M BufferRAM 初始化

↓

RA8871M 一般设定

↓

RA8871M TFT 时序设定

↓

RA8871M 影像显示内存与窗口初始化设定

↓

RA8871M TFT 显示开启

2.1 硬件复位(hardware reset)

begin()

RA8871M 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 RA8871M，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 RA8871M 驱动板。

2.2 PLL 初始化

ra8871mPllInitial()

函数会参考以下 User_def.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    120   // SDRAM clock frequency, unit: MHz.
#define CORE_FREQ    100   // Core (system) clock frequency, unit: MHz.
#define SCAN_FREQ    TFT_PCLK   // Panel Scan clock frequency, unit: MHz.
```

定义	说明
OSC_FREQ	连接到 RA8871M 的晶体振荡器频率,建议为 10MHz
DRAM_FREQ	SDRAM 存取频率,建议 40~120MHz
CORE_FREQ	RA8871M 系统核心频率,建议 40~100MHz
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

通常使用者只需要在 User_def.h 选择以下的定义

```
//#define TFT_OUT_320_240
#define TFT_OUT_480_272
```

2.3 BufferRAM 初始化

RA8871M 有内建内存(BufferRAM)做为图像操作与显示内存。

ra8871mBufferRamInitial ()

函数会参考 User_def.h 内定义值 #define BufferRAM_FREQ，并自动执行 BufferRAM 初始化。

通常使用者只需要在 User_def.h 选择以下的定义

```
//#define TFT_OUT_320_240
#define TFT_OUT_480_272
```

2.4 一般设定

以下寄存器在初始化期间设定，参考 RA8871M 规格书与 Ra8871m_Lite.h 寄存器各 bit 定义,根据你的需求设定下列几项。

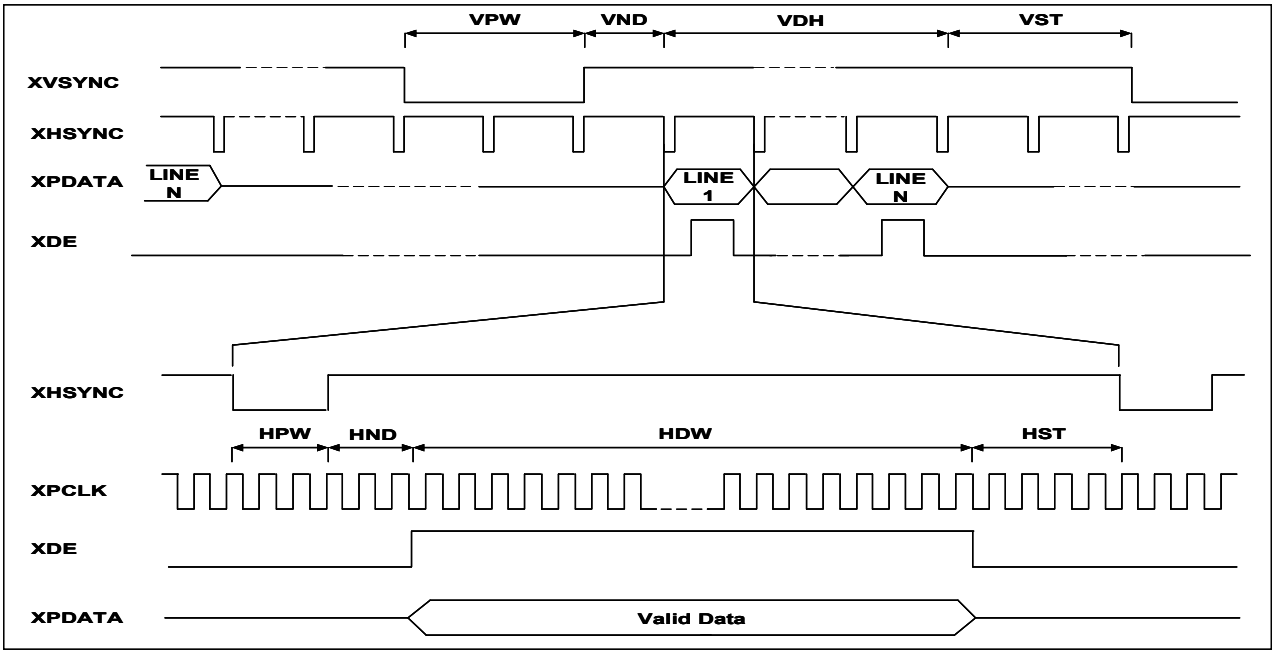
```
lcdRegWrite(RA8871M_CCR);//01h
lcdDataWrite(RA8871M_PLL_ENABLE<<7|RA8871M_WAIT_NO_MASK<<6|RA8871M_KEY_SCAN_DISABLE<<5|RA8871M_TFT_OUTPUT24<<3|RA8871M_I2C_MASTER_DISABLE<<2|RA8871M_SERIAL_IF_ENABLE<<1|RA8871M_HOST_DATA_BUS_SERIAL);

lcdRegWrite(RA8871M_MACR);//02h
lcdDataWrite(RA8871M_DIRECT_WRITE<<6|RA8871M_READ_MEMORY_LRTB<<4|RA8871M_WRITE_MEMORY_LRTB<<1);

lcdRegWrite(RA8871M_ICR);//03h
lcdDataWrite(RA8871M_GRAPHIC_MODE<<2|RA8871M_MEMORY_SELECT_IMAGE);

lcdRegWrite(RA8871M_MPWCTR);//10h
lcdDataWrite(RA8871M_PIP1_WINDOW_DISABLE<<7|RA8871M_PIP2_WINDOW_DISABLE<<6|RA8871M_SELECT_CONFIG_PIP1<<4|RA8871M_IMAGE_COLOCR_DEPTH_16BPP<<2|TFT_MODE);

lcdRegWrite(RA8871M_PIPCDEP);//11h
lcdDataWrite(RA8871M_PIP1_COLOR_DEPTH_16BPP<<2|RA8871M_PIP2_COLOR_DEPTH_16BPP);

lcdRegWrite(RA8871M_AW_COLOR);//5Eh
lcdDataWrite(RA8871M_CANVAS_BLOCK_MODE<<2|RA8871M_CANVAS_COLOR_DEPTH_16BPP);

lcdRegDataWrite(RA8871M_BTE_COLR,RA8871M_S0_COLOR_DEPTH_16BPP<<5|RA8871M_S1_COLOR_DEPTH_16BPP<<2|RA8871M_S0_COLOR_DEPTH_16BPP);//92h
```

2.5 TFT 时序初始化设定

参考以下 User_def.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_PCLK    10    //set PCLK=10MHz

#define TFT_MODE     0    //0:SYNC_mode(SYNC+DE mode), 1: DE mode    //if sync only
                          //mode do not connect DE signal or define XDE_INV = 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           35
#define HDW           480
#define HST           2
#define VPW           2
#define VND           15
#define VDH           272
#define VST           2
```

RA8871M 输出时序参考



本文选用之 **TFT LCD** 为 **AT043tn24**,要求的 **TFT** 时序如下图

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Clock cycle	1/tc	-	9.00	15	MHz	
Hsync cycle	1/fH	-	17.14	-	KHz	
Vsync cycle	1/fv	-	59.94	-	Hz	
Horizontal signal	th	-	525	-	CLK	Note 1
Horizontal display period	thd	-	480	-	CLK	
Horizontal Front porch	thf	2	-	-	CLK	Note 2
Horizontal Pulse width	thp	2	41	-	CLK	Note 2
Horizontal Back porch	thb	2	-	-	CLK	Note 2
Vertical cycle	tv	-	286	-	H	
Vertical display period	tvd	-	272	-	H	
Vertical Front porch	tvf	2	2	-	H	
Vertical Pulse width	tvp	2	10	-	H	
Vertical Back porch	tvb	2	2	-	H	

TFT 时序初始化设定程序：

```
lcdRegWrite(RA8871M_DPCR);//12h
lcdDataWrite(XPCLK_INV<<7|RA8871M_DISPLAY_OFF<<6|RA8871M_OUTPUT_RGB);

lcdRegWrite(RA8871M_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 User_def.h 内定义值：

```
//定义显示屏分辨率宽高
#define SCREEN_WIDTH 480
#define SCREEN_HEIGHT 272

/*RA8871M 提供 3 个 512Kbytes 内存区块(BufferRAM)，我们可以规划为 3 个页面*/
/*使用者影像内存缓存页面规划*/
#define PAGE1_START_ADDR   786432   //0C0000h
#define PAGE2_START_ADDR   1835008 //1C0000h
#define PAGE3_START_ADDR   2883584 //2C0000h
```

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

RA8871M 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，RA8871M 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：

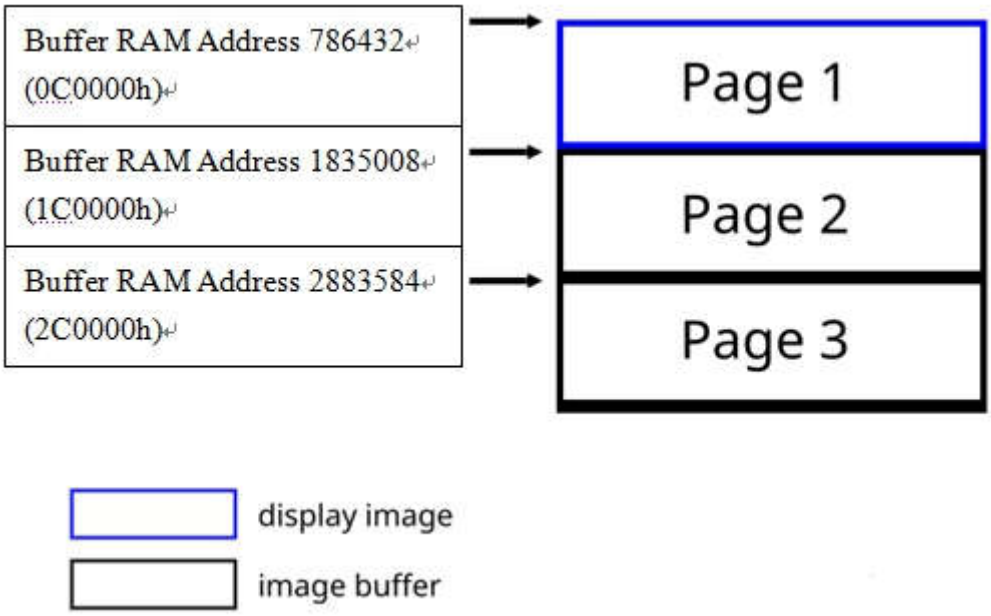
void displayOn(boolean on);

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 3 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 3 个页面，第 1 个页面(Buffer RAM address = 786432)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 480(=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。

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);

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 RA8871M 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 等操作,都必须在当前画布的活动窗口(active window)区域内执行，本文中内存规划为 3 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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);

//指定第 2 页为当前画布
ra8871m.canvasImageStartAddress(PAGE2_START_ADDR);

//指定第 3 页为当前画布
ra8871m.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 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_16bpp()	指定坐标绘制一个像素点
putPicture_16bpp()	指定坐标与宽高然后写入图像数续
putPicture_16bpp()	指定坐标与绘制图像宽高与图像数据指针(Byte 格式)
putPicture_16bpp()	指定坐标与绘制图像宽高与图像数据指针(Word 格式)

注：

参考 RA8871M Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

图像数据使用 Image_Tool_v1.1.0.1 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：

RA8871M 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_16bpp()

描述：
在指定坐标绘制一个像素点。

函数原型：
void putPixel_16bpp(ru16 x,ru16 y,ru16 color);

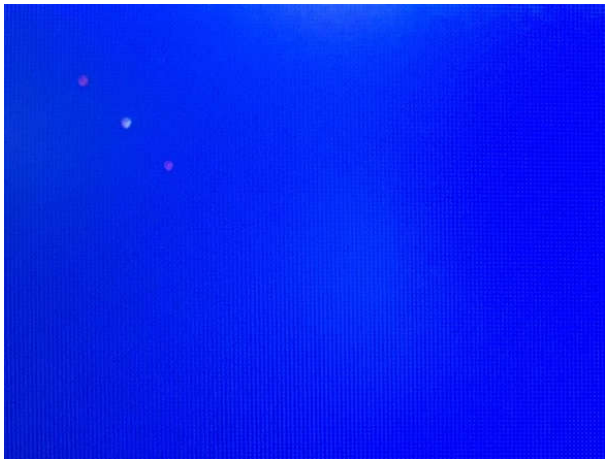
参数	说明
x	X 轴坐标
y	Y 轴坐标
color	RGB565 数据

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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1,
COLOR65K_BLUE);
```

```
//在当前画布(canvas)指定坐标位置(20,20)绘制一个红色像素点
ra8871m.setPixelCursor(20,20);
ra8871m.ramAccessPrepare();
ra8871m.lcdDataWrite(0x00);//RGB565 LSB data
ra8871m.lcdDataWrite(0xf8); //RGB565 MSB data

//在当前画布(canvas)指定坐标位置(30,20)绘制一个白色像素点
ra8871m.setPixelCursor(30,30);
ra8871m.ramAccessPrepare();
ra8871m.lcdDataWrite16bbp(COLOR65K_WHITE);//RGB565 16bpp data

//在当前画布(canvas)指定坐标位置(40,30)绘制一个紫红色像素点
ra8871m.putPixel_16bpp(40,40,COLOR65K_MAGENTA);
```

范例显示截图：



putPicture_16bpp()

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_16bpp(ru16 x,ru16 y,ru16 width, ru16 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_16bpp()

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_16bpp(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_v1.1.0.1 图像工具转换。

putPicture_16bpp()

描述：

设定坐标与图像宽高与图像数据指针(Word 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_16bpp(ru16 x,ru16 y,ru16 width, ru16 height, const unsigned short *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	Word 格式图像数据指针

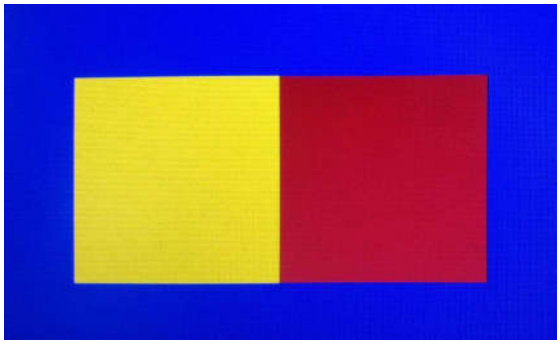
注：

图像数据使用 Image_Tool_v1.1.0.1 图像工具转换。

附注和范例：

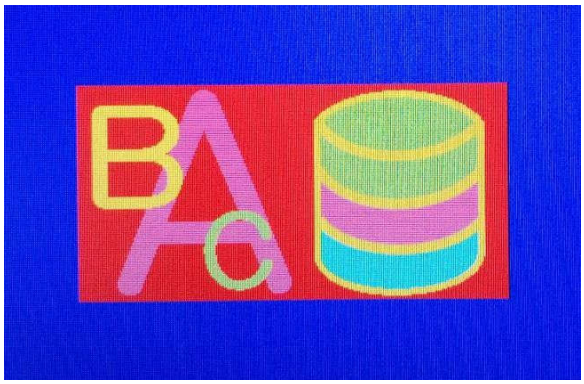
```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
 ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
 ra8871m.canvasImageWidth(SCREEN_WIDTH);
 ra8871m.activeWindowXY(0,0);
 ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
 ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1,
 COLOR65K_BLUE);


//在当前画布(canvas)指定坐标位置填入 128*128 图像
ra8871m.putPicture_16bpp(50,50,128,128);
 for(i=0;i<16384;i++)
 {
 ra8871m.lcdDataWrite16bbp(COLOR65K_YELLOW);//RGB565 16bpp data
 }
ra8871m.putPicture_16bpp(50+128,50,128,128);
    for(i=0;i<16384;i++)
  {
    ra8871m.lcdDataWrite16bbp(COLOR65K_BROWN);//RGB565 16bpp data
  }
```

范例显示截图：



```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
  ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
  ra8871m.canvasImageWidth(SCREEN_WIDTH);
  ra8871m.activeWindowXY(0,0);
  ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
  ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1,
  COLOR65K_BLUE);

//在当前画布(canvas)指定坐标位置填入 128*128 图像
ra8871m.putPicture_16bpp(50,50,128,128,pic16bpp_byte);
ra8871m.putPicture_16bpp(50+128,50,128,128,pic16bpp_word);
```

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：

请参考 User_def.h 文件和设定 DEMO_ASCII_8X12 、DEMO_ASCII_16X24 、DEMO_ASCII_32X48 定义为 1 或 0 如以下，增加 8x12、16x24、32x48 尺寸字的支援。

```
#define DEMO_ASCII_8X12 1
#define DEMO_ASCII_16X24 1
#define DEMO_ASCII_32X48 1

#ifdef DEMO_ASCII_8X12
#include "ascii_table_8x12.h"
#endif

#ifdef DEMO_ASCII_16X24
#include "ascii_table_16x24.h"
#endif

#ifdef DEMO_ASCII_32X48
#include "ascii_table_32x48.h"
#endif
```

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 short fgcolor, unsigned short bgcolor, boolean bg_transparent, unsigned char code)

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

void lcdPutChar32x48(unsigned short x, unsigned short y, unsigned short fgcolor, unsigned short 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 short fgcolor, unsigned short bgcolor, boolean bg_transparent, char *ptr)

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

void lcdPutString32x48(unsigned short x, unsigned short y, unsigned short fgcolor, unsigned short 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
  ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
  ra8871m.canvasImageWidth(SCREEN_WIDTH);
  ra8871m.activeWindowXY(0,0);
  ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
  ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_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
  ra8871m.lcdPutString8x12(0,0,0xFFFF,0x0000,true," !\"#$%&'()*+,-./012345678");
  ra8871m.lcdPutString8x12(0,12,0xFFFF,0x0000,true,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
  ra8871m.lcdPutString8x12(0,24,0xFFFF,0x0000,true,"RSTUVWXYZ[\\]^_`abcdefghij");
  ra8871m.lcdPutString8x12(0,36,0xFFFF,0x0000,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
  ra8871m.lcdPutString16x24(0,78,0xFFFF,0x0000,true,"012345");
  ra8871m.lcdPutString16x24(0,102,0xFFFF,0x0000,true,"Hello!");
#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
  ra8871m.lcdPutString32x48(0,140,0xFFFF,0x0000,false,"012345");
  ra8871m.lcdPutString32x48(0,190,0xFFFF,0x0000,false,"Hello!");
#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 RA8871M 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(ru16 foreground_color, ru16 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	RA8871M_SELECT_INTERNAL_CGROM RA8871M_SELECT_EXTERNAL_CGROM RA8871M_SELECT_USER_DEFINED
size_select	RA8871M_CHAR_HEIGHT_16 RA8871M_CHAR_HEIGHT_24

	RA8871M_CHAR_HEIGHT_32
iso_select	RA8871M_SELECT_8859_1 RA8871M_SELECT_8859_2 RA8871M_SELECT_8859_4 RA8871M_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	RA8871M_TEXT_FULL_ALIGN_DISABLE RA8871M_TEXT_FULL_ALIGN_ENABLE 对齐全型字开启位
chroma_key	RA8871M_TEXT_CHROMA_KEY_DISABLE RA8871M_TEXT_CHROMA_KEY_ENABLE 文字背景色透明开启位
width_enlarge	RA8871M_TEXT_WIDTH_ENLARGEMENT_X1 RA8871M_TEXT_WIDTH_ENLARGEMENT_X2 RA8871M_TEXT_WIDTH_ENLARGEMENT_X3 RA8871M_TEXT_WIDTH_ENLARGEMENT_X4 文字水平放大选择
height_enlarge	RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1 RA8871M_TEXT_HEIGHT_ENLARGEMENT_X2 RA8871M_TEXT_HEIGHT_ENLARGEMENT_X3 RA8871M_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	RA8871M_SERIAL_FLASH_SELECT0 RA8871M_SERIAL_FLASH_SELECT1 选择使用 SPI0 或 SPI1
clk_div	RA8871M_SPI_DIV2 RA8871M_SPI_DIV4 RA8871M_SPI_DIV6 RA8871M_SPI_DIV8 RA8871M_SPI_DIV10 设定集通字库用 SPI clock 除频
rom_select	RA8871M_GT21L16T1W RA8871M_GT30L16U2W RA8871M_GT30L24T3Y RA8871M_GT30L24M1Z RA8871M_GT30L32S4W RA8871M_GT20L24F6Y RA8871M_GT21L24S1W 选择集通字库
character_select	RA8871M_GB2312 RA8871M_GB12345_GB18030 RA8871M_BIG5 RA8871M_ASCII RA8871M_UNICODE RA8871M_UNI_JAPANESE RA8871M_JIS0208 RA8871M_LATIN_GREEK_CYRILLIC_ARABIC_THAI_HEBREW RA8871M_ISO_8859_1_AND_ASCII RA8871M_ISO_8859_2_AND_ASCII RA8871M_ISO_8859_3_AND_ASCII RA8871M_ISO_8859_4_AND_ASCII RA8871M_ISO_8859_5_AND_ASCII

	RA8871M_ISO_8859_7_AND_ASCII RA8871M_ISO_8859_8_AND_ASCII RA8871M_ISO_8859_9_AND_ASCII RA8871M_ISO_8859_10_AND_ASCII RA8871M_ISO_8859_11_AND_ASCII RA8871M_ISO_8859_13_AND_ASCII RA8871M_ISO_8859_14_AND_ASCII RA8871M_ISO_8859_15_AND_ASCII RA8871M_ISO_8859_16_AND_ASCII 选择字型译码器
gt_width	RA8871M_GT_FIXED_WIDTH RA8871M_GT_VARIABLE_WIDTH_ARIAL RA8871M_GT_VARIABLE_FIXED_WIDTH_ROMAN RA8871M_GT_BOLD 选择字体

注：

建议 serial IF0 连接到集通字库，serial IF1 连接到一般的 serial flash。

RA8871M 支持多个集通字库型号，详细参考 RA8871M 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

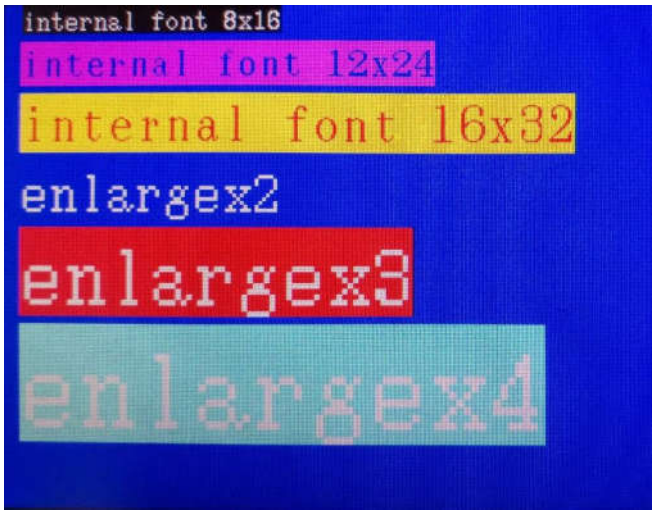
范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
 ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
 ra8871m.canvasImageWidth(SCREEN_WIDTH);
```

```
 ra8871m.activeWindowXY(0,0);
 ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
 ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_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 8x16 ASCII 字符串
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_H
  EIGHT_16,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
 ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
 ra8871m.putString(10,0,"internal font 8x16");

//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
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
 IGHT_24,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
 ra8871m.textColor(COLOR65K_BLUE,COLOR65K_MAGENTA);
 ra8871m.putString(10,20,"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 16x32 ASCII 字符串
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
 IGHT_32,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
 ra8871m.textColor(COLOR65K_RED,COLOR65K_YELLOW);
 ra8871m.putString(10,48,"internal font 16x32");
```

```
//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
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_16,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,RA8871M_TEXT_CHROMA_KEY_ENABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X2,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X2);
 ra8871m.textColor(COLOR65K_WHITE,COLOR65K_RED);
 ra8871m.putString(10,84,"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
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_16,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X3,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X3);
 ra8871m.textColor(COLOR65K_WHITE,COLOR65K_RED);
 ra8871m.putString(10,120,"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
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_16,RA8871M_SELECT_8859_1);//cch
 ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X4,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X4);
 ra8871m.textColor(COLOR65K_WHITE, COLOR65K_LIGHTCYAN);
 ra8871m.putString(10,172,"enlarge x4");
```

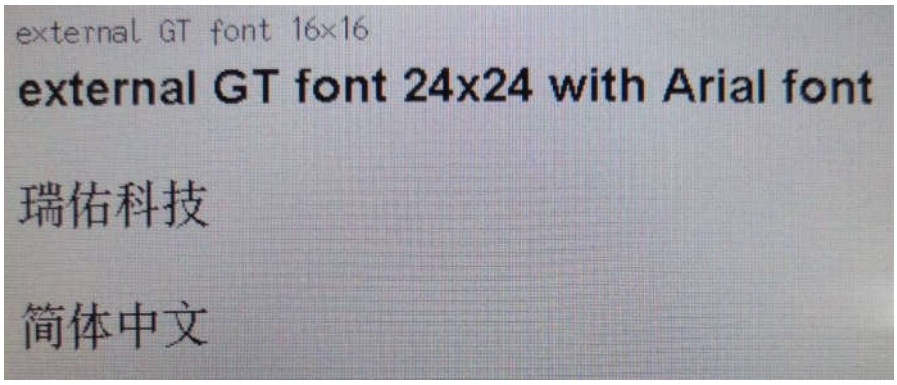
范例显示截图：




```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
 ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
 ra8871m.canvasImageWidth(SCREEN_WIDTH);
 ra8871m.activeWindowXY(0,0);
 ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
 ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//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
ra8871m.setTextParameter1(RA8871M_SELECT_EXTERNAL_CGROM,RA8871M_CHAR_H
EIGHT_16,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);

ra8871m.genitopCharacterRomParameter(RA8871M_SERIAL_FLASH_SELECT0,RA8871M_
SPI_DIV4,RA8871M_GT30L24T3Y,RA8871M_BIG5,RA8871M_GT_FIXED_WIDTH);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.putString(10,10,"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
ra8871m.setTextParameter1(RA8871M_SELECT_EXTERNAL_CGROM,RA8871M_CHAR_HEIGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);

ra8871m.genitopCharacterRomParameter(RA8871M_SERIAL_FLASH_SELECT0,RA8871M_SPI_DIV4,RA8871M_GT30L24T3Y,RA8871M_BIG5,RA8871M_GT_VARIABLE_WIDTH_ARIAL);
ra8871m.putString(10,30,"external GT font 24x24 with Arial font");

ra8871m.putString(10,90,string1);

ra8871m.setTextParameter1(RA8871M_SELECT_EXTERNAL_CGROM,RA8871M_CHAR_HEIGHT_24,RA8871M_SELECT_8859_1);//cch

ra8871m.genitopCharacterRomParameter(RA8871M_SERIAL_FLASH_SELECT0,RA8871M_SPI_DIV4,RA8871M_GT30L24T3Y,RA8871M_GB2312,RA8871M_GT_FIXED_WIDTH);
ra8871m.putString(10,150,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);

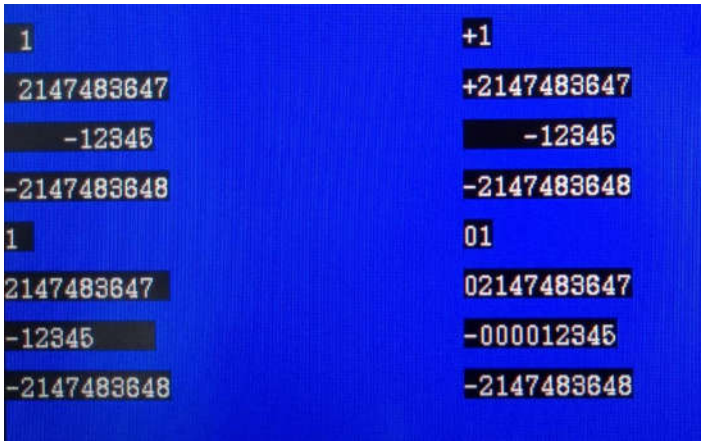
参数	说明
x0	左上角 X 轴坐标
y0	左上角 Y 轴坐标
vaule	输入数值 -2147483648(-2^31) ~ 2147483647(2^31-1)
len	最小显示位数(1~11)
*flag	= “n” ： 显示靠右 = “-” ： 显示靠左 = “+” ： 输出正负号 = “0” ： 在开头处补 0,非补空白

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_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 16x32 ASCII 字符串
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_32,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);

//显示数值
ra8871m.putDec(0,10,1,2,"n");
ra8871m.putDec(0,36,2147483647,11,"n");
ra8871m.putDec(0,62,-12345,10,"n");
ra8871m.putDec(0,88,-2147483648,11,"n");
```

```
ra8871m.putDec(0,114,1,2,"-");
ra8871m.putDec(0,140,2147483647,11,"-");
ra8871m.putDec(0,166,-12345,10,"-");
ra8871m.putDec(0,192,-2147483648,11,"-");

ra8871m.putDec(SCREEN_WIDTH/2,10,1,2,"+");
ra8871m.putDec(SCREEN_WIDTH/2,36,2147483647,11,"+");
ra8871m.putDec(SCREEN_WIDTH/2,62,-12345,10,"+");
ra8871m.putDec(SCREEN_WIDTH/2,88,-2147483648,11,"+");

ra8871m.putDec(SCREEN_WIDTH/2,114,1,2,"0");
ra8871m.putDec(SCREEN_WIDTH/2,140,2147483647,11,"0");
ra8871m.putDec(SCREEN_WIDTH/2,166,-12345,10,"0");
ra8871m.putDec(SCREEN_WIDTH/2,192,-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：

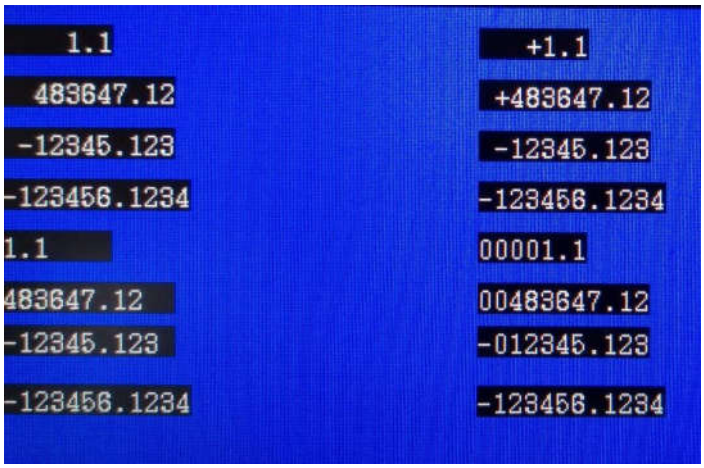
为了得到更高得精准度,这里使用了 double。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_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 16x32 ASCII 字符串
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
IGHT_32,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);

//显示数值
ra8871m.putFloat(0,10,1.1,7,1,"n");
ra8871m.putFloat(0,36,483647.12,11,2,"n");
ra8871m.putFloat(0,62,-12345.123,11,3,"n");
ra8871m.putFloat(0,88,-123456.1234,11,4,"n");

ra8871m.putFloat(0,114,1.1234,7,1,"-");
```

```
ra8871m.putFloat(0,140,483647.12,11,2,"-");
ra8871m.putFloat(0,162,-12345.123,11,3,"-");
ra8871m.putFloat(0,192,-123456.1234,11,4,"-");

ra8871m.putFloat(SCREEN_WIDTH/2,10,1.1,7,1,"+");
ra8871m.putFloat(SCREEN_WIDTH/2,36,483647.12,11,2,"+");
ra8871m.putFloat(SCREEN_WIDTH/2,62,-12345.123,11,3,"+");
ra8871m.putFloat(SCREEN_WIDTH/2,88,-123456.1234,11,4,"+");

ra8871m.putFloat(SCREEN_WIDTH/2,114,1.1,7,1,"0");
ra8871m.putFloat(SCREEN_WIDTH/2,140,483647.12,11,2,"0");
ra8871m.putFloat(SCREEN_WIDTH/2,162,-12345.123,11,3,"0");
ra8871m.putFloat(SCREEN_WIDTH/2,192,-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

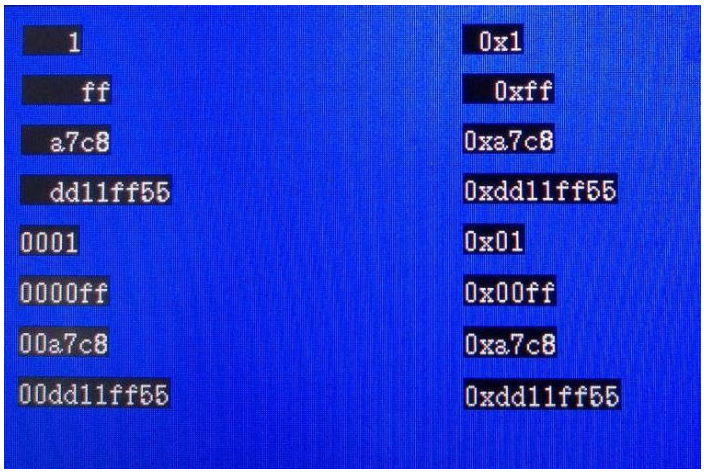
vaule	输入数值 0x00000000~0xffffffff
len	最小显示位数(1~10)
*flag	= “n” ： 显示靠右 = “#” ： 强制输出 0x 作为开头 = “0” ： 在开头处补 0,非补空白 = “x” ： 强制输出 0x 作为开头，补 0

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_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 16x32 ASCII 字符串
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_32,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_DISABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);

//显示数值
ra8871m.putHex(10,10,1,4,"n");
ra8871m.putHex(10,36,255,6,"n");
ra8871m.putHex(10,62,0xa7c8,6,"n");
ra8871m.putHex(10,88,0xdd11ff55,10,"n");

ra8871m.putHex(10,114,1,4,"0");
ra8871m.putHex(10,140,255,6,"0");
ra8871m.putHex(10,166,0xa7c8,6,"0");
ra8871m.putHex(10,192,0xdd11ff55,10,"0");

ra8871m.putHex(SCREEN_WIDTH/2,10,1,4,"#");
```

```
ra8871m.putHex(SCREEN_WIDTH/2,36,255,6,"#");
ra8871m.putHex(SCREEN_WIDTH/2,62,0xa7c8,6,"#");
ra8871m.putHex(SCREEN_WIDTH/2,88,0xdd11ff55,10,"#");

ra8871m.putHex(SCREEN_WIDTH/2,114,1,4,"x");
ra8871m.putHex(SCREEN_WIDTH/2,140,255,6,"x");
ra8871m.putHex(SCREEN_WIDTH/2,166,0xa7c8,6,"x");
ra8871m.putHex(SCREEN_WIDTH/2,192,0xdd11ff55,10,"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 RA8871M 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, ru16 color);

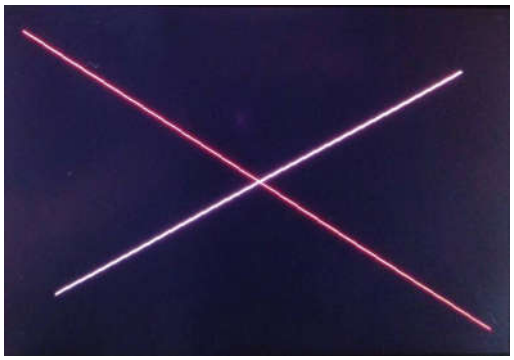
参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标
x1	第 2 点 X 轴坐标
y1	第 2 点 Y 轴坐标
color	设定颜色(RGB565)

范例：

ra8871m.drawLine(20,20,SCREEN_WIDTH-20,SCREEN_HEIGHT-20,COLOR65K_RED);

```
//
ra8871m.foreGroundColor16bpp(COLOR65K_LIGHTRED);
ra8871m.drawLine(SCREEN_WIDTH-50,50,50,SCREEN_HEIGHT-50);
```

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, ru16 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(RGB565)

范例：

```
ra8871m.drawSquare(20, 20, SCREEN_WIDTH-20, SCREEN_HEIGHT-20,
COLOR65K_GRAYSCALE23);
```

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：

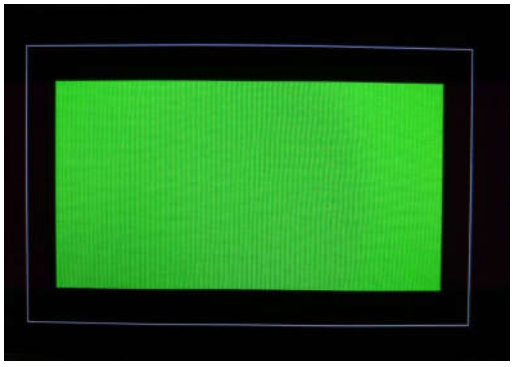
void drawSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 color);

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(RGB565)

范例：

ra8871m.drawSquareFill(50,50,SCREEN_WIDTH-50,SCREEN_HEIGHT-50,
COLOR65K_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, ru16 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(RGB565)

范例：

ra8871m.drawCircleSquare(20,20,SCREEN_WIDTH-20, SCREEN_HEIGHT-20, 20, 20, COLOR65K_BLUE2);

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：

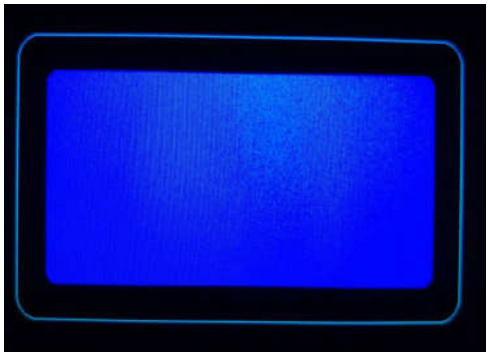
void drawCircleSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 xr, ru16 yr, ru16 color);

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(RGB565)

范例：

ra8871m.drawCircleSquareFill(50,50,SCREEN_WIDTH-50, SCREEN_HEIGHT-50, 10, 10, COLOR65K_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,ru16 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(RGB565)

范例：

ra8871m.drawTriangle(160,20,300,200,50,220,COLOR65K_MAGENTA);

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,ru16 color);

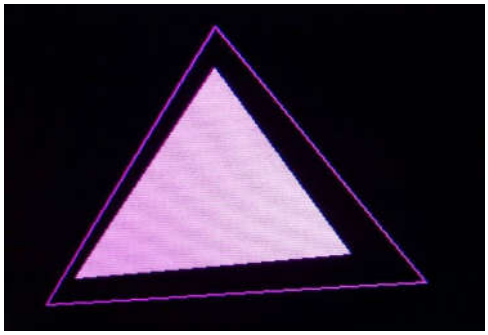
参数	说明
x0	第 1 点 X 轴坐标
y0	第 1 点 Y 轴坐标

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(RGB565)

范例：

ra8871m.drawTriangleFill(160,50,250,180,70,200,COLOR65K_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,ru16 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(RGB565)

范例：

ra8871m.drawCircle(SCREEN_WIDTH/2,SCREEN_HEIGHT/2,80,COLOR65K_YELLOW);

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：

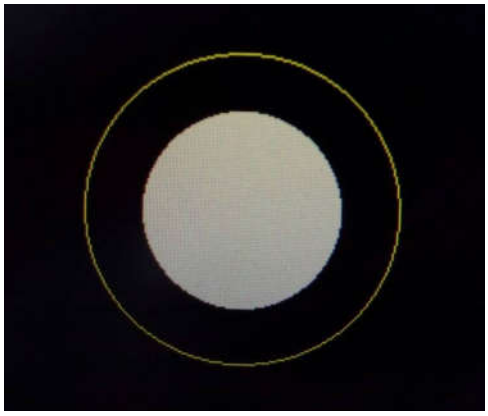
void drawCircleFill(ru16 x0,ru16 y0,ru16 r,ru16 color);

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(RGB565)

范例：

ra8871m.drawCircleFill(SCREEN_WIDTH/2,SCREEN_HEIGHT/2,50,COLOR65K_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,ru16 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(RGB565)

范例：

ra8871m.drawEllipse(SCREEN_WIDTH/2,SCREEN_HEIGHT/2,50,80,COLOR65K_CYAN);

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：

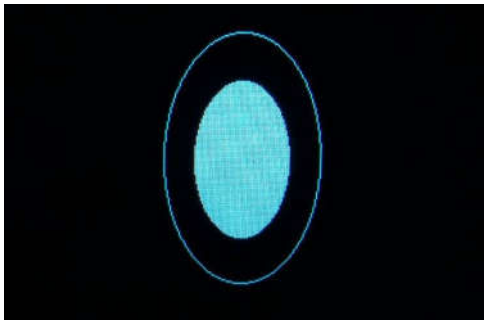
void drawEllipseFill(ru16 x0,ru16 y0,ru16 xr,ru16 yr,ru16 color);

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(RGB565)

范例：

ra8871m.drawEllipseFill(SCREEN_WIDTH/2,SCREEN_HEIGHT/2,30,50,COLOR65K_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，样版图像填充和填充与透明色等功能。

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 1 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 接口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。

样版填充的功能让使用者可以使用尺寸 8*8 或 16*16 的彩色图像(16bpp)，快速的填充指定的内存区块。
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 RA8871M 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
btePatternFill()	样板图像填充
btePatternFillWithChromaKey()	样板图像填充与忽略透明色

注：

参考 RA8871M 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_v1.1.0.1 图像工具转换。

参考图片：

Pic16bpp_word.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(pic16bpp_word.h)，并且包含到程序内。

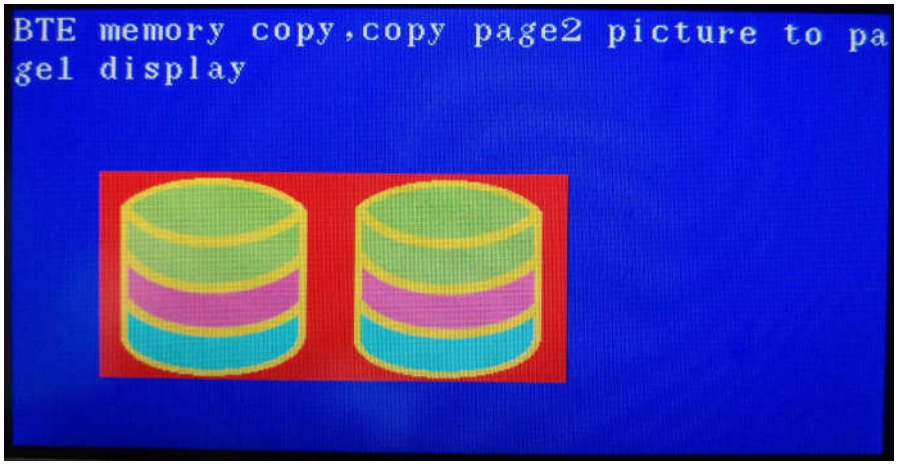
范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

//清除当前画布(canvas) page2 的活动窗口(active window)为红色
ra8871m.canvasImageStartAddress(PAGE2_START_ADDR);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_RED);

//写入图像数据至当前画布 page2 指定位置
ra8871m.putPicture_16bpp(0,0 ,128,128,pic16bpp_word);

//写入字符串至当前画布(canvas) page1 指定位置
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);

ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
IGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
RA8871M_TEXT_CHROMA_KEY_ENABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.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)
ra8871m.bteMemoryCopy(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,PAGE1_START_ADDR,SCREEN_WIDTH, 50,100,128,128);

ra8871m.bteMemoryCopy(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,PAGE1_START_ADDR,SCREEN_WIDTH, 50+128,100,128,128);

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 RA8871M_BTE_ROP_CODE_0 (Blackness) RA8871M_BTE_ROP_CODE_1 ~S0・~S1 or ~ (S0+S1) RA8871M_BTE_ROP_CODE_2 ~S0・S1 RA8871M_BTE_ROP_CODE_3 ~S0 RA8871M_BTE_ROP_CODE_4 S0・~S1 RA8871M_BTE_ROP_CODE_5 ~S1 RA8871M_BTE_ROP_CODE_6 S0^S1 RA8871M_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8871M_BTE_ROP_CODE_8 S0・S1 RA8871M_BTE_ROP_CODE_9 ~ (S0^S1) RA8871M_BTE_ROP_CODE_10 S1 RA8871M_BTE_ROP_CODE_11 ~S0+S1 RA8871M_BTE_ROP_CODE_12 S0 RA8871M_BTE_ROP_CODE_13 S0+~S1 RA8871M_BTE_ROP_CODE_14 S0+S1 RA8871M_BTE_ROP_CODE_15 (Whiteness)

注：

图像数据使用 Image_Tool_v1.1.0.1 图像工具转换。

参考图片：

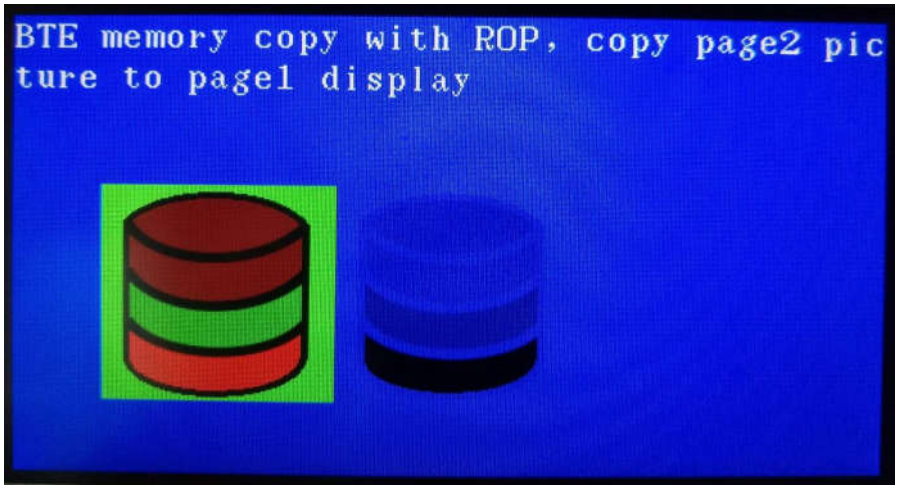
Pic16bpp_word.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(pic16bpp_word.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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.putString(0,0,"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
ra8871m.bteMemoryCopyWithROP(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,PAGE1_ST
ART_ADDR,SCREEN_WIDTH,50,100,PAGE1_START_ADDR,SCREEN_WIDTH,
50,100,128,128,RA8871M_BTE_ROP_CODE_1);

ra8871m.bteMemoryCopyWithROP(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,PAGE1_ST
ART_ADDR,SCREEN_WIDTH,(50+128),100,PAGE1_START_ADDR,SCREEN_WIDTH,
(50+128),100,128,128,RA8871M_BTE_ROP_CODE_2);
```

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, ru16 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_v1.1.0.1 图像工具转换。

参考图片：

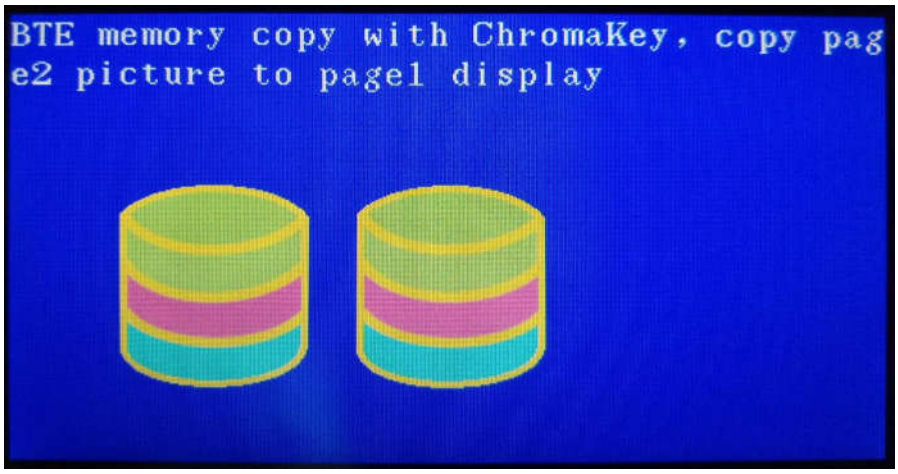
Pic16bpp_word.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(pic16bpp_word.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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.putString(0,0,"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
ra8871m.bteMemoryCopyWithChromaKey(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,
PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,0xf800);

ra8871m.bteMemoryCopyWithChromaKey(PAGE2_START_ADDR,SCREEN_WIDTH,0,0,
PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,0xf800);
```

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 RA8871M_BTE_ROP_CODE_0 (Blackness) RA8871M_BTE_ROP_CODE_1 ~S0・~S1 or ~ (S0+S1) RA8871M_BTE_ROP_CODE_2 ~S0・S1 RA8871M_BTE_ROP_CODE_3 ~S0 RA8871M_BTE_ROP_CODE_4 S0・~S1 RA8871M_BTE_ROP_CODE_5 ~S1 RA8871M_BTE_ROP_CODE_6 S0^S1 RA8871M_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8871M_BTE_ROP_CODE_8 S0・S1 RA8871M_BTE_ROP_CODE_9 ~ (S0^S1) RA8871M_BTE_ROP_CODE_10 S1 RA8871M_BTE_ROP_CODE_11 ~S0+S1 RA8871M_BTE_ROP_CODE_12 S0 RA8871M_BTE_ROP_CODE_13 S0+~S1 RA8871M_BTE_ROP_CODE_14 S0+S1 RA8871M_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_v1.1.0.10 图像工具转换。

参考图片：

Pic16bpp_byte.bmp

Pic16bpp_word.bmp

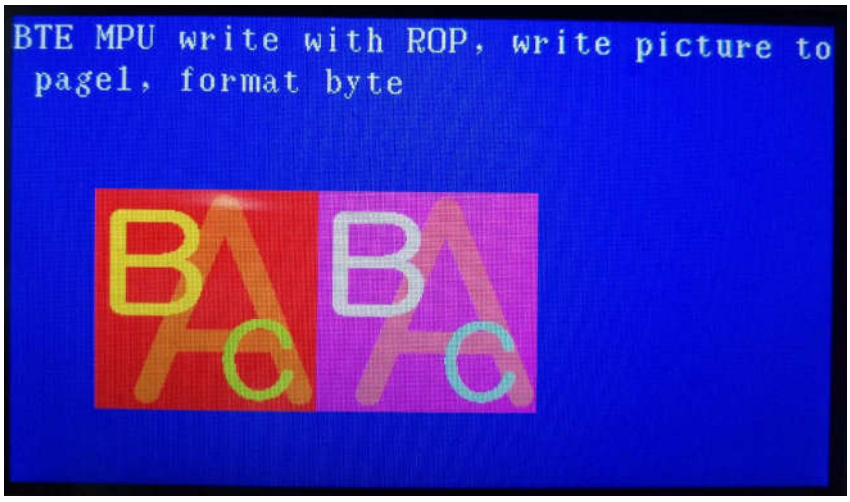
以下范例使用者必须预先转换图片文件为 16bpp 格式(pic16bpp_byte.h，pic16bpp_word.h) ，并且包含到程序内。

范例：

```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
```

```
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
RA8871M_TEXT_CHROMA_KEY_ENABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.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)
ra8871m.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50,100,PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,RA8871M_BTE_ROP_CODE_4,pic16bpp_byte);
ra8871m.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,RA8871M_BTE_ROP_CODE_6,pic16bpp_byte);
```

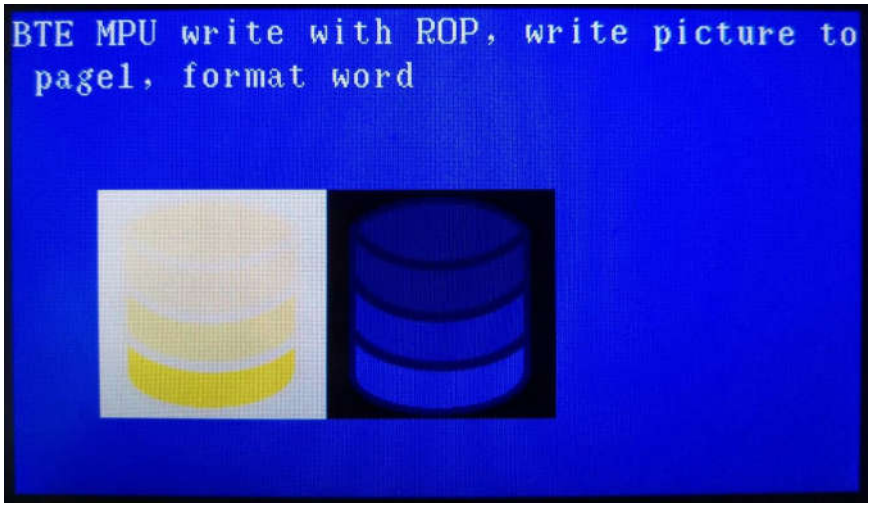
范例显示截图：




```
//清除当前画布(canvas) page1 的活动窗口(active window)为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);
```

```
//写入字符串至当前画布 page1 指定位置
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.putString(0,0,"BTE MPU write with ROP, write picture to page1, format word");

// 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)
ra8871m.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50,100,PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,RA8871M_BTE_ROP_CODE_7,pic16bpp_word);

ra8871m.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,RA8871M_BTE_ROP_CODE_8,pic16bpp_word);
```

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, ru16 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, ru16 chromakey_color, const unsigned short *data);

void bteMpuWriteWithChromaKey(ru32 des_addr, ru16 des_image_width, ru16 des_x, ru16 des_y, ru16 width, ru16 height, ru16 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_v1.1.0.1 图像工具转换。

参考图片：

Pic16bpp_byte.bmp

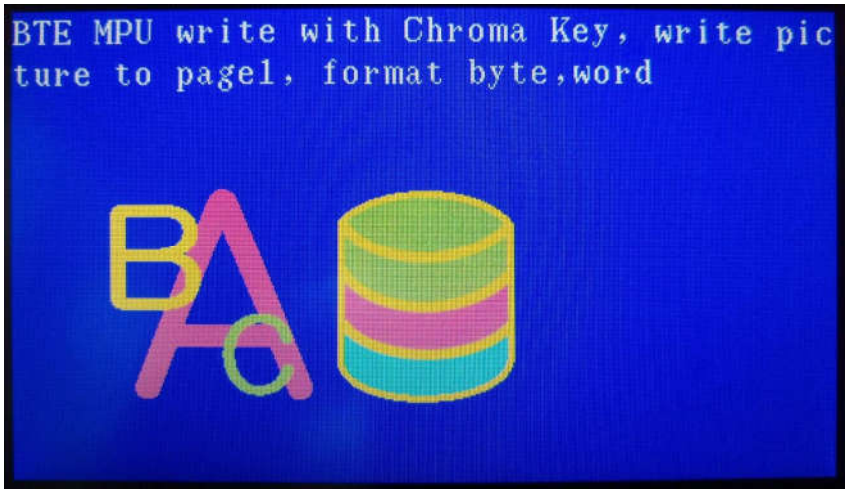
Pic16bpp_word.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(pic16bpp_byte.h，pic16bpp_word.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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.putString(0,0,"BTE MPU write with Chroma Key, write picture to page1, format byte,word");

// MPU 写入数据透过 BTE 透明色数据忽略后写入目的地画布(page1).
ra8871m.bteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,
50,100,128,128,0xf800,pic16bpp_byte);
ra8871m.bteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,
50+128,100,128,128,0xf800,pic16bpp_word);
```

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, ru16 foreground_color, ru16 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, ru16 foreground_color, ru16 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_v1.1.0.1 图像工具转换。

参考图片：

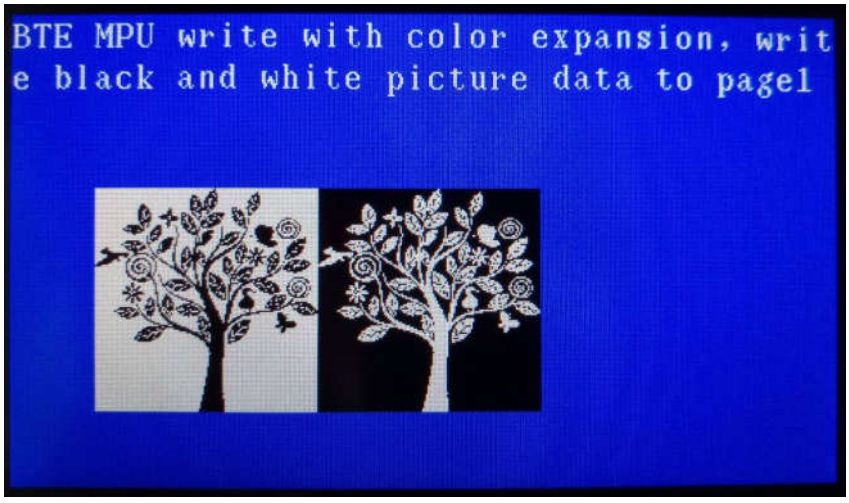
Bw.bmp

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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
RA8871M_TEXT_CHROMA_KEY_ENABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.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
ra8871m.bteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,COLOR65K_BLACK,COLOR65K_WHITE,bw);
ra8871m.bteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,COLOR65K_WHITE,COLOR65K_BLACK,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, ru16 foreground_color, ru16 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, ru16 foreground_color, ru16 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_v1.1.0.1 图像工具转换。

参考图片：

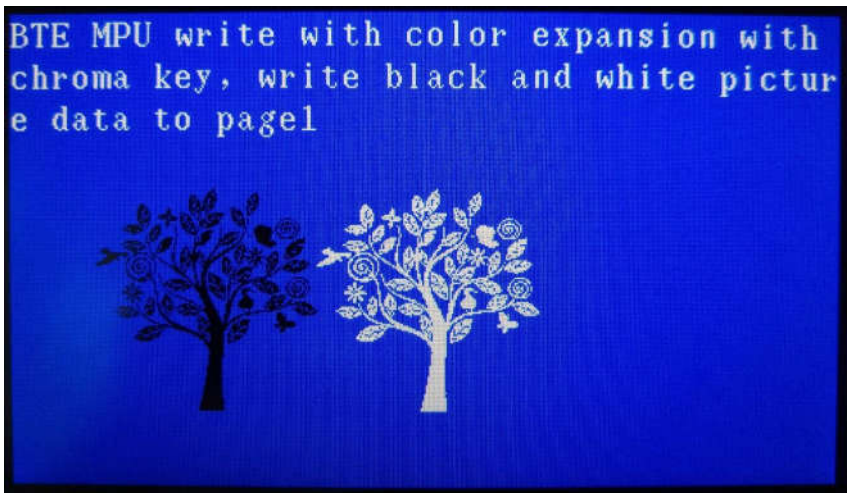
Bw.bmp

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
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//写入字符串至当前画布 page1 指定位置
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.putString(0,0,"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
ra8871m.bteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,COLOR65K_BLACK,COLOR65K_WHITE,bw);

ra8871m.bteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,COLOR65K_WHITE,COLOR65K_BLACK,bw);
```

范例显示截图：



btePatternFill()

描述：

使用样版图像填满指定画布区块。

函数原型：

void btePatternFill(ru8 p8x8or16x16, 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 width, ru16 height);

参数	说明
p8x8or16x16	选择样版尺寸, 0 = 8*8, 1=16*16
s0_addr	样版图像来源(画布)内存起始地址
s0_image_width	样版图像影像(画布)内存宽度
s0_x	样版来源影像 X 轴坐标
s0_y	样版来源影像 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)内存宽度
des_x	目的地影像 X 轴起始坐标
des_y	目的地影像 Y 轴起始坐标
width	要填满的区块宽
height	要填满的区块高

注：

样版图像是预先写入到使用者指定的内存地址。

图像数据使用 Image_Tool_v1.1.0.1 图像工具转换。

参考图片：

pattern6.bmp

pattern11.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(pattern6.h，pattern11.h),并且包含到程序内。

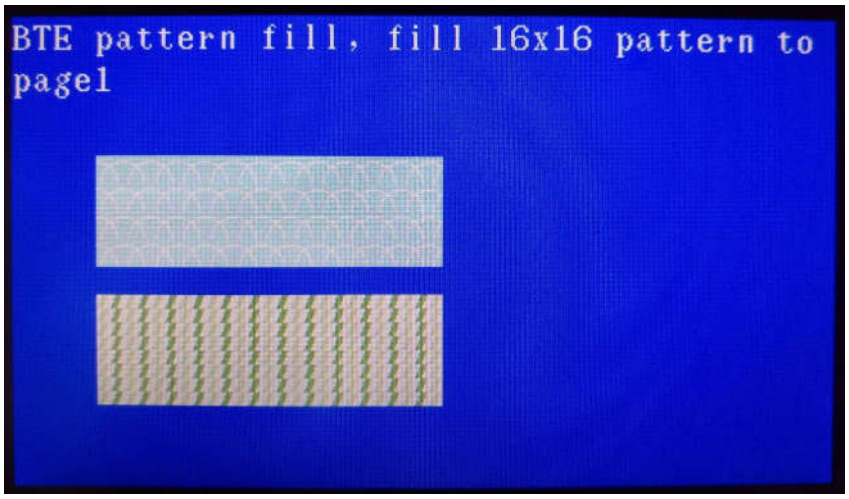
范例：

ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);

```
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//写入图片到指定的 pattern1 内存区块
ra8871m.canvasImageStartAddress(PATTERN1_RAM_START_ADDR);
ra8871m.canvasImageWidth(16);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(16,16);
ra8871m.putPicture_16bpp(0,0,16,16,pattern6);

//写入图片到指定的 pattern2 内存区块
ra8871m.canvasImageStartAddress(PATTERN2_RAM_START_ADDR);
ra8871m.putPicture_16bpp(0,0,16,16,pattern11);

//写入图片到指定的 pattern3 内存区块
ra8871m.canvasImageStartAddress(PATTERN3_RAM_START_ADDR);
ra8871m.putPicture_16bpp(0,0,16,16,bug1);

//set canvas and active window back
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);

ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
IGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
RA8871M_TEXT_CHROMA_KEY_ENABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,
RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.putString(0,0,"BTE pattern fill, fill 16x16 pattern to page1");

ra8871m.btePatternFill(1,PATTERN1_RAM_START_ADDR,16,0,0,PAGE1_START_ADDR,SC
REEN_WIDTH, 50,80,200,64);
```

ra8871m.btePatternFill(1,PATTERN2_RAM_START_ADDR,16,0,0,PAGE1_START_ADDR,SCREEN_WIDTH, 50,160,200,64);

范例显示截图：



btePatternFillWithChromaKey()

描述：

使用样版图像并忽略透明色填满指定画布区块。

函数原型：

void btePatternFill(ru8 p8x8or16x16, 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 width, ru16 height , ru16 chromakey_color);

参数	说明
p8x8or16x16	选择样版尺寸 0 = 8*8, 1=16*16
s0_addr	样版图像来源(画布)内存起始地址
s0_image_width	样版图像影像(画布)内存宽度
s0_x	样版来源影像 X 轴坐标
s0_y	样版来源影像 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)内存宽度
des_x	目的地影像 X 轴起始坐标
des_y	目的地影像 Y 轴起始坐标

width	要填满的区块宽
height	要填满的区块高
chromakey_color	透明色数据

注：

样版图像是预先写入到使用者指定的内存地址。

图像数据使用 Image_Tool_v1.1.0.1 图像工具转换。

参考图片：

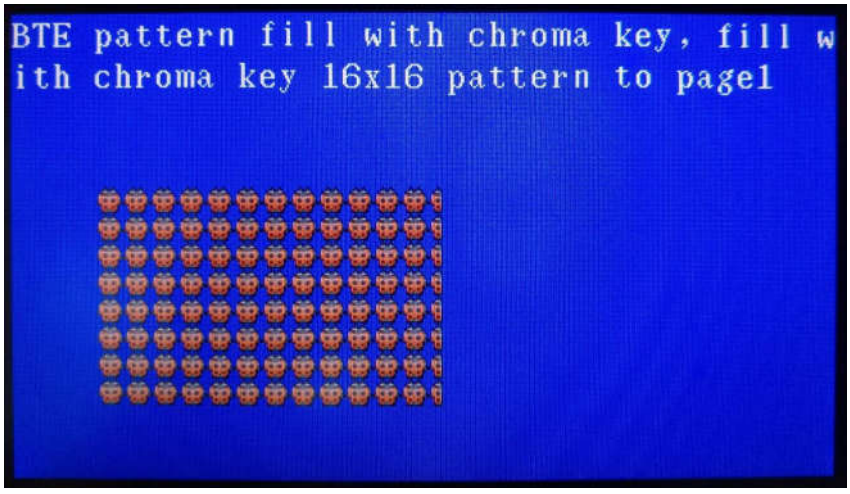
Bug1.bmp

以下范例使用者必须预先转换图片文件为 16bpp 格式(bug1.h)，并且包含到程序内。

范例：

```
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

ra8871m.putString(0,356,"BTE pattern fill with chroma key, fill with chroma key 16x16 pattern to page1");

ra8871m.btePatternFillWithChromaKey(1,PATTERN3_RAM_START_ADDR,16,0,0,PAGE1_START_ADDR,SCREEN_WIDTH, 50,100,200,128,0xe8e4);
```

范例显示截图：



第 8 章 DMA

RA8871M 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
setSerialFlash4BytesMode()	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 RA8871M Arduino Wire Sketch.jpg 接线图或附录 Figure A-1。

Serial Flash 烧录,参考 ArduinoDue_SpiFlashProgramWithSdCard 演示与说明。

第 8 章演示范例，使用者必须先对 Serial Flash 烧录”**ArduinoDue_SpiFlashProgramWithSdCard**”演示项目的”**file2sdcard**”资料夹内”**All_Pic.bin**”文件。

图像数据使用 Image_Tool_v1.1.0.1 图像工具转换。

setSerialFlash4BytesMode()

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：

void setSerialFlash4BytesMode(ru8 scs_select);

参数	说明
scs_select	选择 serial IF0 或 serial IF1

注：

建议 serial IF0 连接到集通字库，serial IF1 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 scs_selct, ru8 clk_div, ru16 x0, ru16 y0, ru16 width, ru16 height, ru16 picture_width, ru32 addr);

参数	说明
scs_selct	RA8871M_SERIAL_FLASH_SELECT0 RA8871M_SERIAL_FLASH_SELECT1 选择 serial IF0 或 serial IF1
clk_div	RA8871M_SPI_DIV2 RA8871M_SPI_DIV4 RA8871M_SPI_DIV6 RA8871M_SPI_DIV8 RA8871M_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：

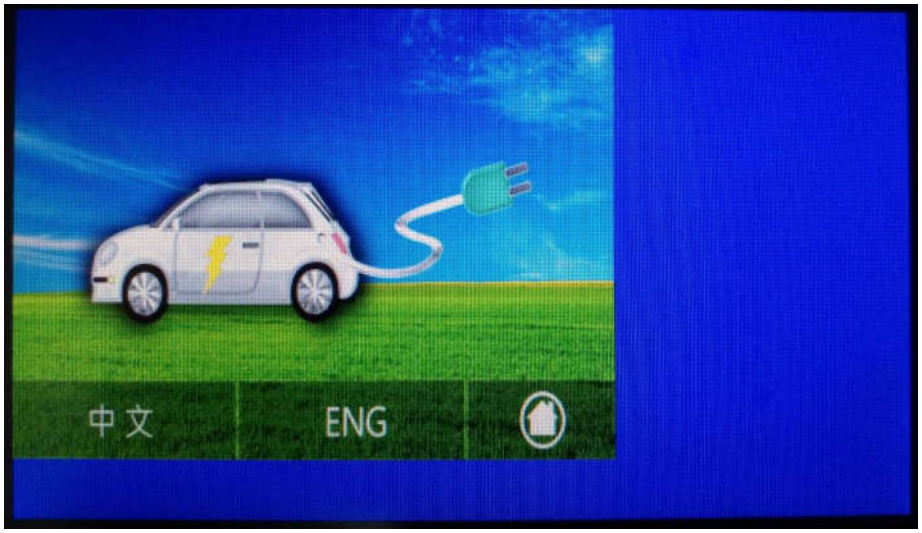
DMA 功能可以执行读取一整个图像数据，然后写入到指定的当前画布区块。

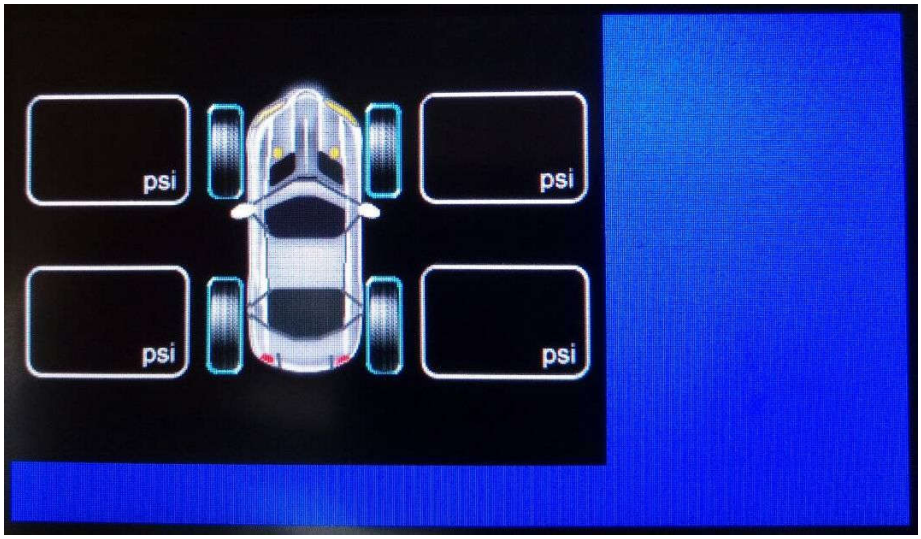
整张图像数据读写范例：

```
//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

ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
```

```
ra8871m.drawSquareFill(0, 0,SCREEN_WIDTH-1,SCREEN_HEIGHT-1,COLOR65K_BLUE);

//DMA 读取 serial flash 内图像写入当前画布指定区块
ra8871m.dma_24bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[carcharge320240].img_width,BINARY_INFO[carcharge320240].img_height,BINARY_INFO[carcharge320240].img_width,BINARY_INFO[carcharge320240].start_addr);

ra8871m.dma_24bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[refrigerator_320240].img_width,BINARY_INFO[refrigerator_320240].img_height,BINARY_INFO[refrigerator_320240].img_width,BINARY_INFO[refrigerator_320240].start_addr);

ra8871m.dma_24bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[tirepressure320240].img_width,BINARY_INFO[tirepressure320240].img_height,BINARY_INFO[tirepressure320240].img_width,BINARY_INFO[tirepressure320240].start_addr);
```

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 scs_selct, ru8 clk_div, ru16 x0, ru16 y0, ru16 width, ru16 height, ru16 picture_width, ru32 addr);

参数	说明
scs_selct	RA8871M_SERIAL_FLASH_SELECT0

	RA8871M_SERIAL_FLASH_SELECT1 选择 serial IF0 或 serial IF1
clk_div	RA8871M_SPI_DIV2 RA8871M_SPI_DIV4 RA8871M_SPI_DIV6 RA8871M_SPI_DIV8 RA8871M_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

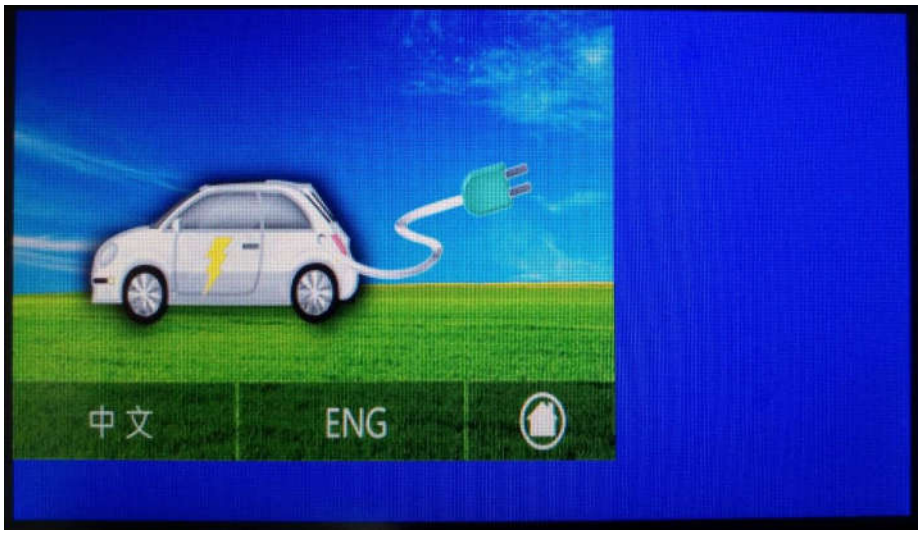
范例：

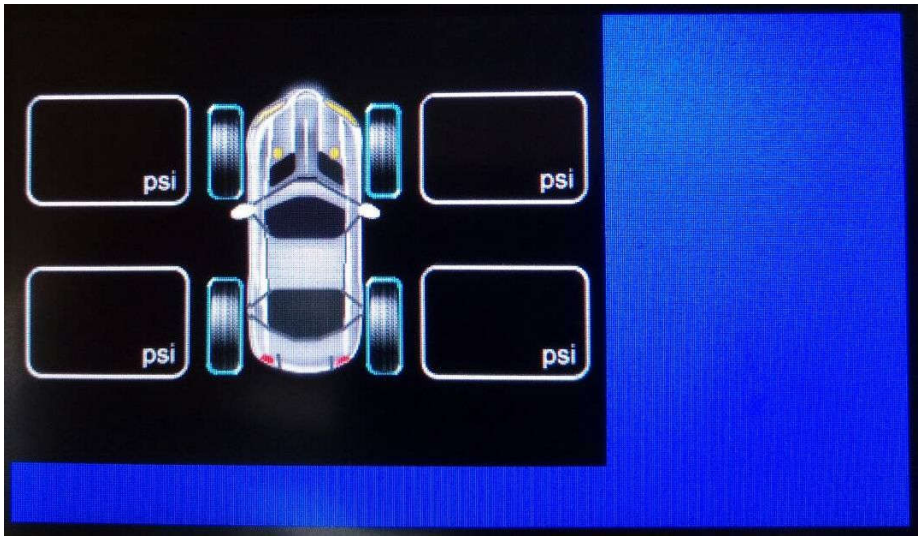
```
//DMA 32bit 地址演示
//当使用 32bit address serial flash 时，必须先设定 serial flash 为 4Bytes mode
//只需要在开电后设定一次
ra8871m.setSerialFlash4BytesMode(1);

//设定当前画布
//清除当前画布(page1)的活动窗口为蓝色
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

//DMA 读取 serial flash 内图像写入当前画布指定区块
ra8871m.dma_32bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[carcharge320240].img_width,BINARY_INFO[carcharge320240].img_height,BINARY_INFO[carcharge320240].img_width,BINARY_INFO[carcharge320240].start_addr);

ra8871m.dma_32bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[refrigerator_320240].img_width,BINARY_INFO[refrigerator_320240
```

].img_height,BINARY_INFO[refrigerator_320240].img_width,BINARY_INFO[refrigerator_320240].start_addr);

ra8871m.dma_32bitAddressBlockMode(RA8871M_SERIAL_FLASH_SELECT1,RA8871M_SPI_DIV2,0,0,BINARY_INFO[tirepressure320240].img_width,BINARY_INFO[tirepressure320240].img_height,BINARY_INFO[tirepressure320240].img_width,BINARY_INFO[tirepressure320240].start_addr);

范例显示截图：




psi
psi
psi
psi

第 9 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 **RA8871M 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	RA8871M_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

	RA8871M_PWM_TIMER_DIV1 RA8871M_PWM_TIMER_DIV2 RA8871M_PWM_TIMER_DIV4 RA8871M_PWM_TIMER_DIV8
pwm0_clk_div	PWM0 基频除频设定 RA8871M_PWM_TIMER_DIV1 RA8871M_PWM_TIMER_DIV2 RA8871M_PWM_TIMER_DIV4 RA8871M_PWM_TIMER_DIV8
xpwm1_ctrl	PWM1 引脚功能选择 RA8871M_XPWM1_OUTPUT_ERROR_FLAG RA8871M_XPWM1_OUTPUT_PWM_TIMER1 RA8871M_XPWM1_OUTPUT_OSC_CLK
xpwm0_ctr	PWM0 引脚功能选择 RA8871M_XPWM0_GPIO_C7 RA8871M_XPWM0_OUTPUT_PWM_TIMER0 RA8871M_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 RA8871M_PWM_TIMER1_INVERTER_OFF RA8871M_PWM_TIMER1_INVERTER_ON
pwm1_auto_reload	PWM1 单次输出或重复输出 RA8871M_PWM_TIMER1_ONE_SHOT RA8871M_PWM_TIMER1_AUTO_RELOAD
pwm1_start	PWM1 停止或开启 RA8871M_PWM_TIMER1_STOP RA8871M_PWM_TIMER1_START

pwm0_dead_zone	PWM0 死区功能选择不启用或启用 RA8871M_PWM_TIMER0_DEAD_ZONE_DISABLE RA8871M_PWM_TIMER0_DEAD_ZONE_ENABLE
pwm0_inverter	PWM0 输出反向 RA8871M_PWM_TIMER0_INVERTER_OFF RA8871M_PWM_TIMER0_INVERTER_ON
pwm0_auto_reload	PWM0 单次输出或重复输出 RA8871M_PWM_TIMER0_ONE_SHOT RA8871M_PWM_TIMER0_AUTO_RELOAD
pwm0_start	PWM0 停止或开启 RA8871M_PWM_TIMER0_STOP RA8871M_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 演示，请用示波器测量频率
ra8871m.pwm_Prescaler(RA8871M_PRESCALER); //if core_freq = 100MHz, pwm base clock
                                          //= 100/(3+1) = 25MHz
ra8871m.pwm_ClockMuxReg(RA8871M_PWM_TIMER_DIV4,  RA8871M_PWM_TIMER_DIV4,
  RA8871M_XPWM1_OUTPUT_PWM_TIMER1,RA8871M_XPWM0_OUTPUT_PWM_TIMER0
  );
//pwm timer clock = 25/4 = 6.25MHz

ra8871m.pwm0_ClocksPerPeriod(1024); // pwm0 = 6.25MHz/1024 = 6.1KHz
ra8871m.pwm0_Duty(10);//pwm0 set 10/1024 duty

ra8871m.pwm1_ClocksPerPeriod(256);    // pwm1 = 7.5MHz/256 = 24.4KHz
ra8871m.pwm1_Duty(5); //pwm1 set 5/256 duty

ra8871m.pwm_Configuration(RA8871M_PWM_TIMER1_INVERTER_ON,RA8871M_PWM_TI
  MER1_AUTO_RELOAD,RA8871M_PWM_TIMER1_START,RA8871M_PWM_TIMER0_DEA
  D_ZONE_DISABLE ,RA8871M_PWM_TIMER0_INVERTER_ON,RA8871M_PWM_TIMER0_
  AUTO_RELOAD,RA8871M_PWM_TIMER0_START);
```

第 10 章 Arduino SD

使用 Arduino due 连接 SD 卡，使用者可以利用 SD 卡做为图片数据的储存器，把已经转换过的图像数据(.bin)透过 PC 存放到 SD，然后使用 Arduino Due 读取 SD 卡内的图像数据填入到 RA8871M 的内存。

函数	说明
sdCardShowPicture16bpp()	读取 SD 卡内指定文件名的图像数据并写入至当前画布指定位置
sdCardShowPicture16bppBteMpuWriteWithROP()	读取 SD 卡内指定文件名的图像数据并透过 BTE 写与逻辑运算写入至目的地画布指定位置
sdCardShowPicture16bppBteMpuWriteWithChromaKey()	读取 SD 卡内指定文件名的图像数据透过 BTE 透明色忽略写入至目的地画布指定位置
sdCardShowPicture16bppBteMpuWriteColorExpansion()	读取 SD 卡内指定文件名的 1bpp 图像数据透过 BTE MPU 写入与颜色扩展至目的地画布指定位置
sdCardShowPicture16bppBteMpuWriteColorExpansionWithChromaKey()	读取 SD 卡内指定文件名的 1bpp 图像数据透过 BTE 执行颜色扩展与透明色忽略写入至目的地画布指定位置

注：

这些子程序是额外提供的，并不包含在 Ra8871m_Lite.cpp 内，如需要使用，参考 Ra8871m_Lite_Arduino_SD.ino，复制需要使用的函数到自己的项目内。

Arduino and SD card and RA8871M 接线图，参考 RA8871MArduinoDueSD Wire Sketch.jpg 或附录 Figure A-2。

图像数据使用 Image_Tool_v1.1.0.10 图像工具转换。

sdCardShowPicture16bpp()

描述：

读取 SD 卡内指定文件名内图像数据，并写入至当前画布指定位置。

函数原型：

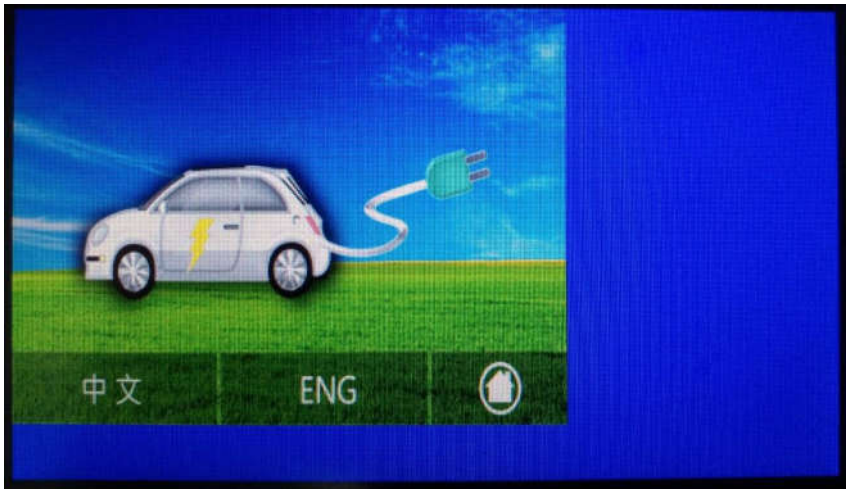
```
void sdCardShowPicture16bpp(unsigned short x, unsigned short y, unsigned short width,
  unsigned short height, char *filename);
```

参数	说明
x	X 轴起始坐标
y	Y 轴起始坐标
width	图像宽度
height	图像高度
*filename	图像文件名

范例：

```
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

sdCardShowPicture16bpp(0,0,320,240,"carc.bin");
```

范例显示截图：



sdCardShowPicture16bppBteMpuWriteWithROP()

描述：

读取 SD 卡内指定文件名的图像数据，并透过 BTE 写与逻辑运算写入至目的地画布指定位置。

函数原型：

```
void sdCardShowPicture16bppBteMpuWriteWithROP(unsigned long s1_addr, unsigned short
```

s1_image_width, unsigned short s1_x, unsigned short s1_y,
unsigned long des_addr, unsigned short des_image_width, unsigned short des_x, unsigned short des_y, unsigned short width, unsigned short height, unsigned char rop_code, char *filename);

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	图像宽度
height	图像高度
rop_code	逻辑运算选择码 RA8871M_BTE_ROP_CODE_0 (Blackness) RA8871M_BTE_ROP_CODE_1 ~S0・~S1 or ~ (S0+S1) RA8871M_BTE_ROP_CODE_2 ~S0・S1 RA8871M_BTE_ROP_CODE_3 ~S0 RA8871M_BTE_ROP_CODE_4 S0・~S1 RA8871M_BTE_ROP_CODE_5 ~S1 RA8871M_BTE_ROP_CODE_6 S0^S1 RA8871M_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8871M_BTE_ROP_CODE_8 S0・S1 RA8871M_BTE_ROP_CODE_9 ~ (S0^S1) RA8871M_BTE_ROP_CODE_10

	S1 RA8871M_BTE_ROP_CODE_11 ~S0+S1 RA8871M_BTE_ROP_CODE_12 S0 RA8871M_BTE_ROP_CODE_13 S0+~S1 RA8871M_BTE_ROP_CODE_14 S0+S1 RA8871M_BTE_ROP_CODE_15 (Whiteness)
*filename	图像文件名

注：

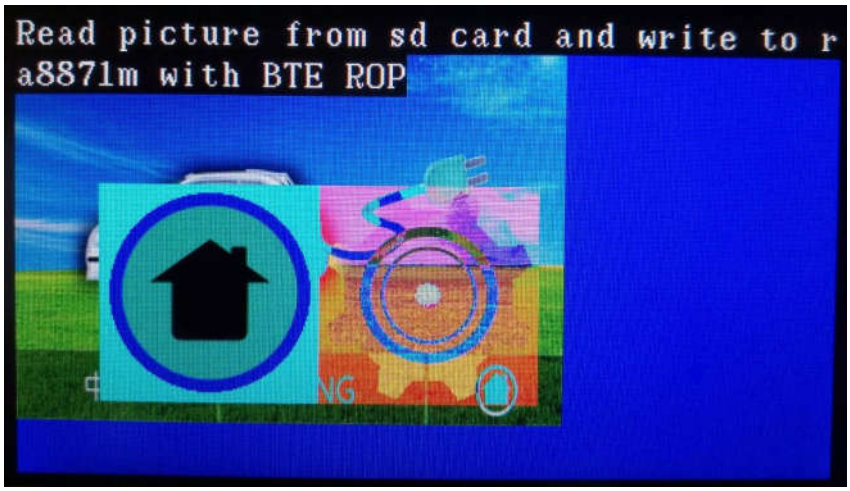
BTE 与 MPU 写入数据相关的功能 S0(Source0) = MPU 写入数据。

S1(Source1)设定可以与 Des(destination)相同。

范例：

```
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);

ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
 IGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
 RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X
 1,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.putString(0,0,"Read picture from sd card and write to ra8871m with BTE ROP");

sdCardShowPicture16bppBteMpuWriteWithROP(PAGE1_START_ADDR, SCREEN_WIDTH,
 50,100,PAGE1_START_ADDR,SCREEN_WIDTH,50,100,128,128,RA8871M_BTE_ROP_CO
 DE_3,"home.bin");
sdCardShowPicture16bppBteMpuWriteWithROP(PAGE1_START_ADDR, SCREEN_WIDTH,
 50+128,100,PAGE1_START_ADDR,SCREEN_WIDTH,50+128,100,128,128,RA8871M_BTE
 _ROP_CODE_6,"appli.bin");
```

范例显示截图：



sdCardShowPicture16bppBteMpuWriteWithChromaKey()

描述：

读取 SD 卡内指定文件名的图像数据，透过 BTE 透明色忽略写入至目的地画布指定位置。

函数原型：

void sdCardShowPicture16bppBteMpuWriteWithChromaKey(unsigned long des_addr , unsigned short des_image_width, unsigned short des_x, unsigned short des_y, unsigned short width, unsigned short height, unsigned short chromakey_color, char *filename);

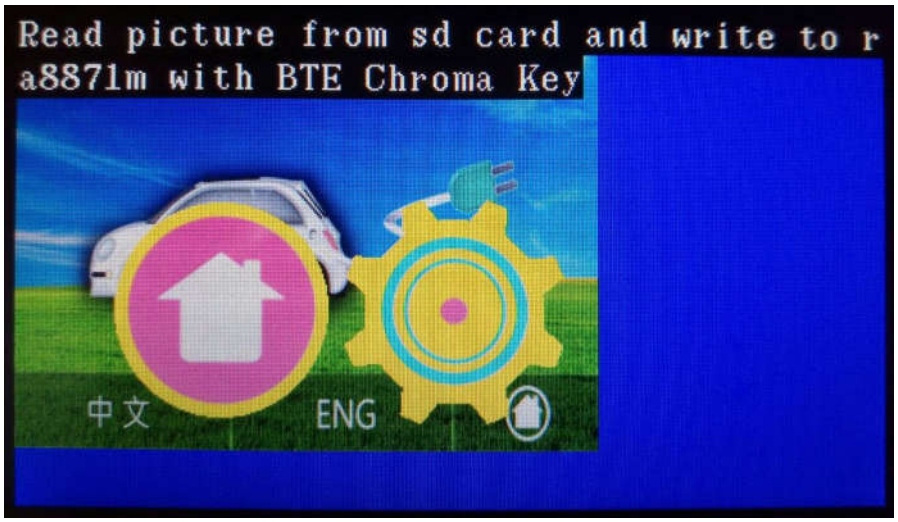
参数	说明
des_addr	目的地(画布)内存起始地址
des_image_width	目的地(画布)内存影像宽度
des_x	目的地 X 轴坐标
des_y	目的地 Y 轴坐标
width	图像宽度
height	图像高度
chromakey_color	透明色数据
* filename	图像文件名

范例：

```
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
```

```
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

sdCardShowPicture16bpp(0,0,320,240,"carc.bin");

ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
  IGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
  RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X
  1,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.putString(0,0,"Read picture from sd card and write to ra8871m with BTE Chroma
  Key");

sdCardShowPicture16bppBteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WI
  DTH,50,100,128,128,0xf800,"home.bin");
sdCardShowPicture16bppBteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WI
  DTH,50+128,100,128,128,0xf800,"appli.bin");
```

范例显示截图：



sdCardShowPicture16bppBteMpuWriteColorExpansion()

描述：

读取 SD 卡内指定文件名的 1bpp 图像数据，透过 BTE MPU 写入与颜色扩展至目的地画布指定位置。

函数原型：

void sdCardShowPicture16bppBteMpuWriteColorExpansion(unsigned long des_addr, unsigned short des_image_width, unsigned short des_x, unsigned short des_y, unsigned short width, unsigned short height, unsigned short foreground_color, unsigned short background_color, char *filename);

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	图像宽度
height	图像高度
foreground_color	前景色
background_color	背景色
* filename	图像文件名

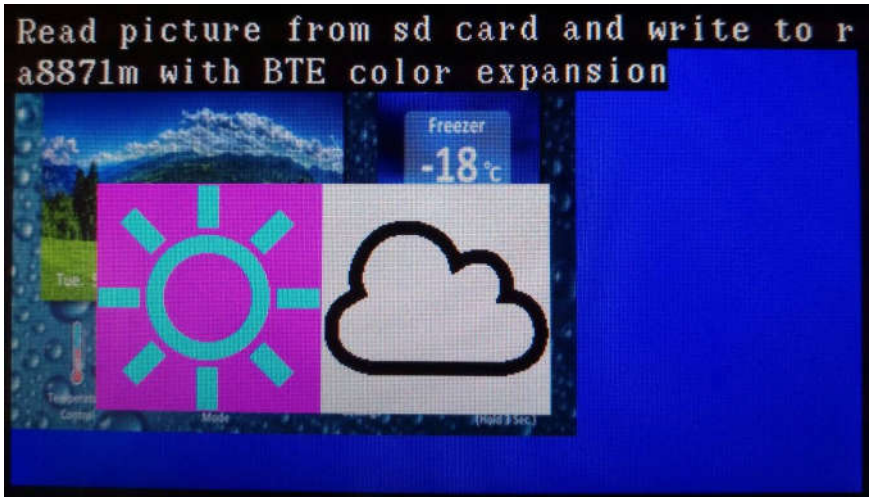
范例：

```
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

sdCardShowPicture16bpp(0,0,320,240,"refri.bin");

ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HEIGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X1,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.putString(0,0,"Read picture from sd card and write to ra8871m with BTE color
```

expansion");

sdCardShowPicture16bppBteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50,100, 128, 128,COLOR65K_CYAN,COLOR65K_MAGENTA,"sun.bin");
sdCardShowPicture16bppBteMpuWriteColorExpansion(PAGE1_START_ADDR,SCREEN_WIDTH,50+128, 100, 128, 128,COLOR65K_BLACK,COLOR65K_WHITE,"cloud.bin");

范例显示截图：



sdCardShowPicture16bppBteMpuWriteColorExpansionWithChromaKey()

描述：

读取 SD 卡内指定文件名的 1bpp 图像数据，透过 BTE 执行颜色扩展与透明色忽略写入至目的地画布指定位置。

函数原型：

void sdCardShowPicture16bppBteMpuWriteColorExpansionWithChromaKey (unsigned long des_addr, unsigned short des_image_width, unsigned short des_x, unsigned short des_y, unsigned short width, unsigned short height, unsigned short foreground_color, unsigned short background_color, char *filename);

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

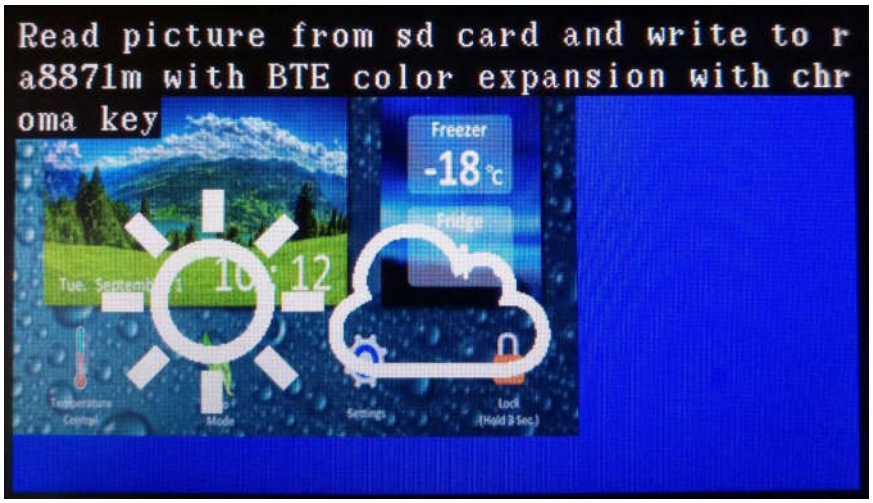
des_y	目的地 Y 轴坐标
width	图像宽度
height	图像高度
foreground_color	前景色
background_color	背景色
* filename	图像文件名

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

范例：

```
ra8871m.canvasImageStartAddress(PAGE1_START_ADDR);
ra8871m.canvasImageWidth(SCREEN_WIDTH);
ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR65K_BLUE);

sdCardShowPicture16bpp(0,0,320,240,"refri.bin");

ra8871m.activeWindowXY(0,0);
ra8871m.activeWindowWH(SCREEN_WIDTH,SCREEN_HEIGHT);
ra8871m.textColor(COLOR65K_WHITE,COLOR65K_BLACK);
ra8871m.setTextParameter1(RA8871M_SELECT_INTERNAL_CGROM,RA8871M_CHAR_HE
 IGHT_24,RA8871M_SELECT_8859_1);//cch
ra8871m.setTextParameter2(RA8871M_TEXT_FULL_ALIGN_ENABLE,
 RA8871M_TEXT_CHROMA_KEY_DISABLE,RA8871M_TEXT_WIDTH_ENLARGEMENT_X
 1,RA8871M_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8871m.putString(0,0,"Read picture from sd card and write to ra8871m with BTE color
 expansion with chroma key");

sdCardShowPicture16bppBteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADD
 R,SCREEN_WIDTH, 50, 100, 128, 128,COLOR65K_WHITE,COLOR65K_BLACK,"sun.bin");
sdCardShowPicture16bppBteMpuWriteColorExpansionWithChromaKey(PAGE1_START_ADD
 R,SCREEN_WIDTH, 50+128, 100, 128,
 128,COLOR65K_WHITE,COLOR65K_BLACK,"cloud.bin");
```

范例显示截图：


Read picture from sd card and write to r
a8871m with BTE color expansion with chr
oma key
Freezer
-18℃
Fridge
Mode
Settings

附录 A

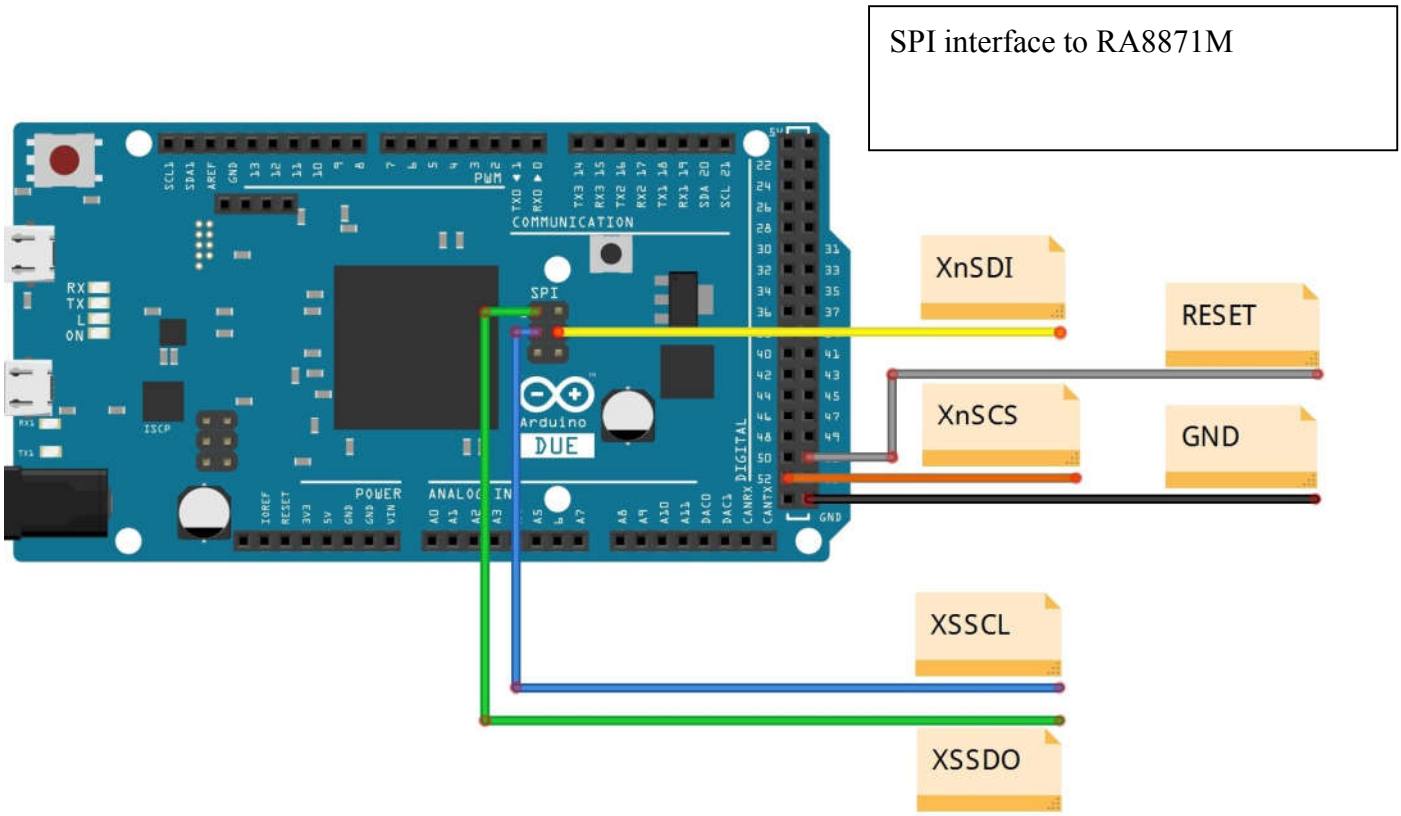


Figure A-1

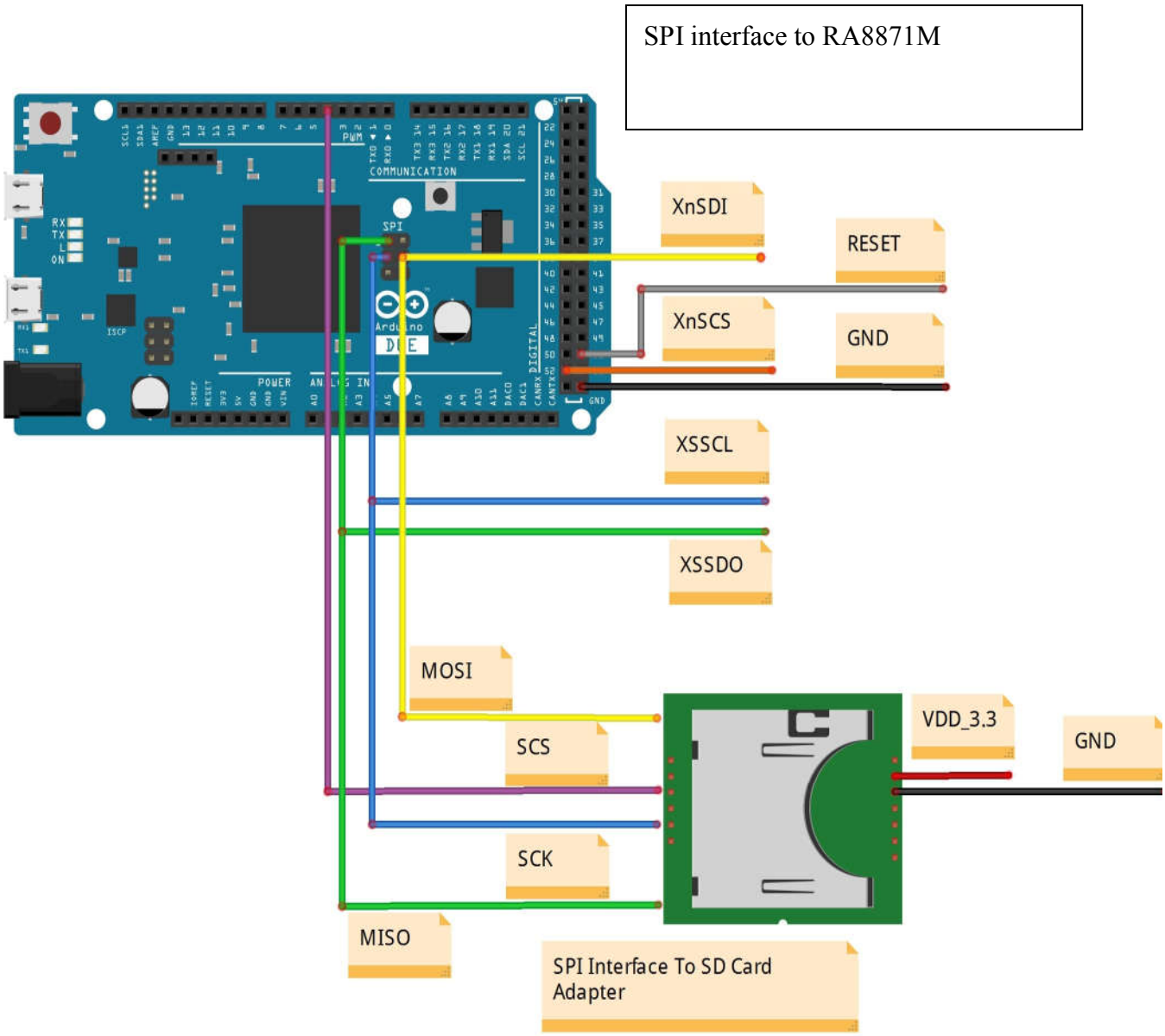


Figure A-2

全文完