RAiO

RA8889_Lite User Guide

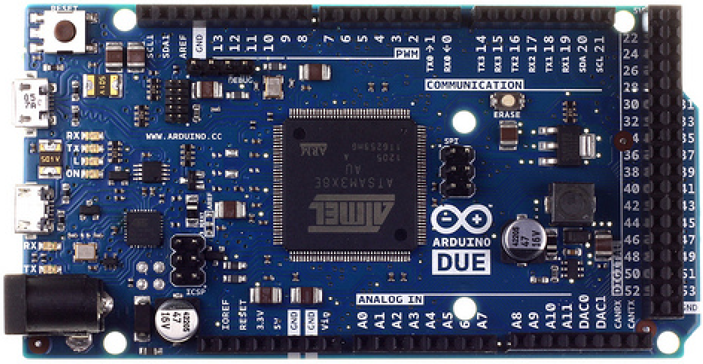
Revise History		
Version	**Date**	**Description**
1.0	MAY 07, 2020	Initial Release

Chapter 1 RA8889_Lite introduction

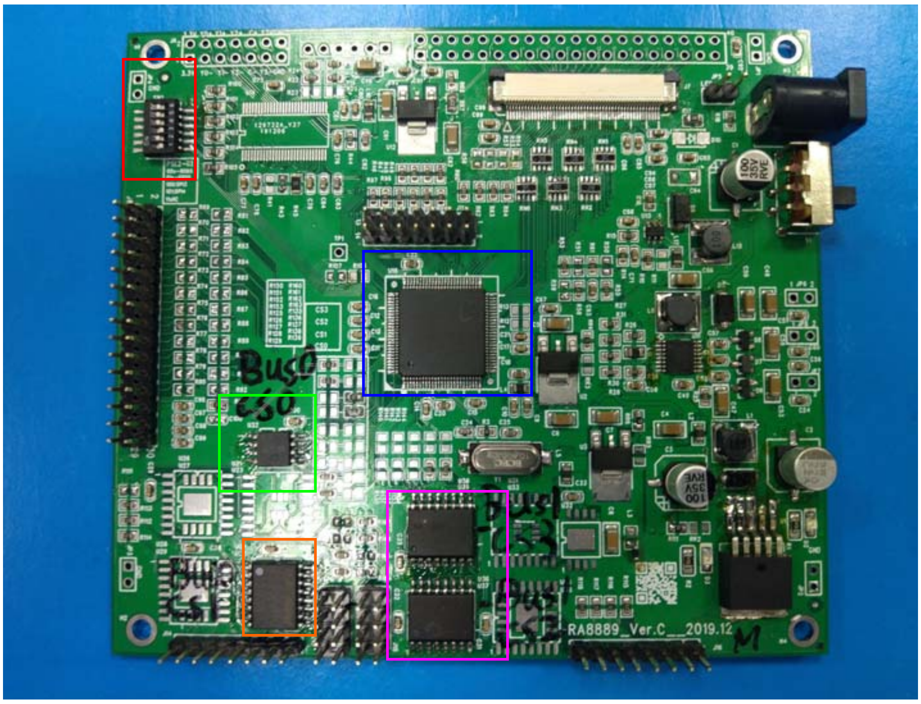
RA8889_Lite provides GUI application source code that is based on the Arudino Due development board, and it can be connected to the driver board of RA8889. This document will help users to rapidly realize how to apply the Arduino Due development environment with RA8889 for the TFT – LCD solutions.

Hardware requirements

1. Arduino Due development board



2. RA8889 evaluation board (mounted SPI FLASH ROM and Genitop Font ROM IC on board)



🟦 RA8889 Chip
🟥 Select Host SPI 4 wire interface
🟧 Serial Flash ROM for DMA function
🟩 Genitop Font ROM
🟪 Serial Flash ROM for IDEC function JPEG/AVI decode on Bus1 xnsfcs2/xnsfcs3

Software requirements

Arduino IDE 1.5.7	http://arduino.cc/en/Main/Software
RA8889 Image_Tool_1.0	www.raio.com.tw

RA8889_Lite features

RA8889_Lite provides application interface (API) that is used for the major built-in functions of RA8889 TFT LCD controller, all demonstrations in this document are based on the SPI interface of Arduino Due development board, it is used with RA8889 for displaying the 24BPP color depth image on the TFT-LCD. The following is the API features in this document:

I. Initialization

RA8889's initialized procedures.

II. Memory configuration & Window

Describe how to configure the external memory of RA8889 which is corresponded to the distinct operating windows.

III. Graphic

In graphic mode, there are two examples here, which is used for describing how to display the image data or customized fonts with RA8889.

a. RA8889 is set in Graphic Mode, the Arduino Due writes the color image data.

b. RA8889 is set in Graphic Mode, the Arduino Due writes user's customized ASCII fonts.

IV. Text

This function provides a simple way to implement displaying fonts on the LCD. When RA8889 is set in the Text Mode, without the complex programming, user just need to send a few commands of standard ASCII code/text string to RA8889, and RA8889 will display the related font data on the LCD automatically. the Arduino Due can writes standard ASCII code to RA8889, RA8889 also supports so many useful functions with the font display. such as the font enlargement, ASCII code, Traditional Chinese characters (BIG5) and Simple Chinese character (GB2312) display. In addition, The function with Chinese characters is implemented by the external font ROM from Genitop Inc.

V. Geometric Draw

This function provides a simple way to implement displaying geometric pattern on the LCD. When RA8889 is set in the Graphic Mode, without the complex programming, user can just send a few commands of drawing function to RA8889, and RA8889 will display the related geometric pattern on the LCD automatically. The geometric patterns are including drawing line, square, square fill, circle square, circle square fill, triangle, triangle fill, circle, circle fill, ellipse, ellipse fill and so on.

VI. BTE

The RA8889 embedded a built-in 2D Block Transfer Engine(BTE) which can increase the performance of block transfer operation. When a block of data needs to be moved or do some logic operation with dedicated data, RA8889 can speed up the operation by BTE hardware and also simplify the MCU program. This section will discuss the BTE engine operation and functionality. This application note will focus on some of the BTE functions as below.

- ♦ BTE memory copy
- ♦ BTE memory ROP logic operation and copy
- ♦ BTE memory copy with chroma key
- ♦ Arduino Due executes memory write with ROP logic operation through BTE engine
- ♦ Arduino Due executes memory write with chroma key through BTE engine
- ♦ Arduino Due executes memory write with color expansion through BTE engine
- ♦ Arduino Due executes memory write with color expansion and chroma key through BTE engine

VII. DMA

With a few command setting, when RA8889 is set in Graphic Mode, the RA8889 can automatically read image data (bitmap format) from external serial flash directly, and then write into its internal memory through the DMA function.

VIII. IDEC

With just a few command setting, when RA8889 is set in Graphic Mode, the RA8889 can automatically reads display data (JPEG/AVI format) from serial flash directly by the IDEC

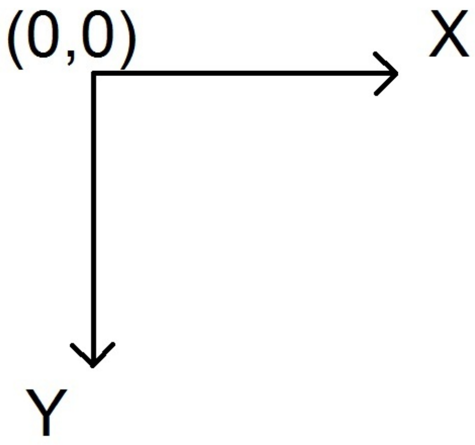
Function, the IDEC will rapidly pass the compressed data to MDU (Media Decoder Unit), the related data will be decode by MDU and then write the decompressed image data into the internal memory of RA8889.

IX. PWM

Description in this section, it is about RA8889 PWM initial setting, frequency calculations and duty cycle configuration. (Users may need an oscilloscope to measure the output frequency)

Note:

1. Display coordinate system in this document:



2. The display resolution is 800 * 600 in this document, for other resolutions, please refer to Chapter 2 Initialization and Chapter 3 Memory configuration & Window.

3. Variables type of user define for RA8889_Lite are listed as below:

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. Please refer to appendix A for the related circuitry connection: Figure A-1

Chapter 2 Initialization

RA8889 initial procedures are shown as follows:

RA8889 hardware reset

↓

RA8889 PLL initialization

↓

RA8889 SDRAM initialization

↓

RA8889 General setting

↓

RA8889 TFT timing setting

↓

RA8889 Image display memory and windows initialized setting

↓

RA8889 TFT Display on

2.1 Hardware reset

begin()

The hardware reset program for RA8889 is included in the function "begin()". If the result of the function "begin()" is return to “true”, it indicates the hardware reset is successful and the hardware connection between RA8889 is correct. If the result is returned to “false”, meaning the hardware reset and its connection is failed. So please check the connection between Arduino Board and RA8889 is correct or not with SPI bus?

2.2 PLL initialization

ra8889PllInitial()

This PLL initialized subroutine will automatically finish the related initialization works depending on the parameters which defined in the RA8889_Lite.h. So according to the display requirement, users just need to define the parameters as the following.

```
#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.
```

Define	Description
OSC_FREQ	Crystal resonator for RA8889, suggested 10MHz
DRAM_FREQ	SDRAM access clock, suggested 100~160MHz
CORE_FREQ	RA8889 system core clock, suggested 100~130MHz
SCAN_FREQ	TFT driving clock PCLK, refer to LCD SPEC specified PCLK frequency requirements

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

2.3 SDRAM initialization

RA8889 have built-in 128Mbit(16MByte) SDRAM which is used as the image operating buffer and display memory.

ra8889SdramInitial()

The function "ra8889SdramInitial()" will refer to RA8889_Lite.h #define DRAM_FREQ, and execute SDRAM initialize automatically

2.4 General setting

According to customer’s display requirement, the following registers should be set during executing the initialization for RA8889. The relevant information please refer to RA8889 specification and the bit definition of each register in the RA8889_Lite.h

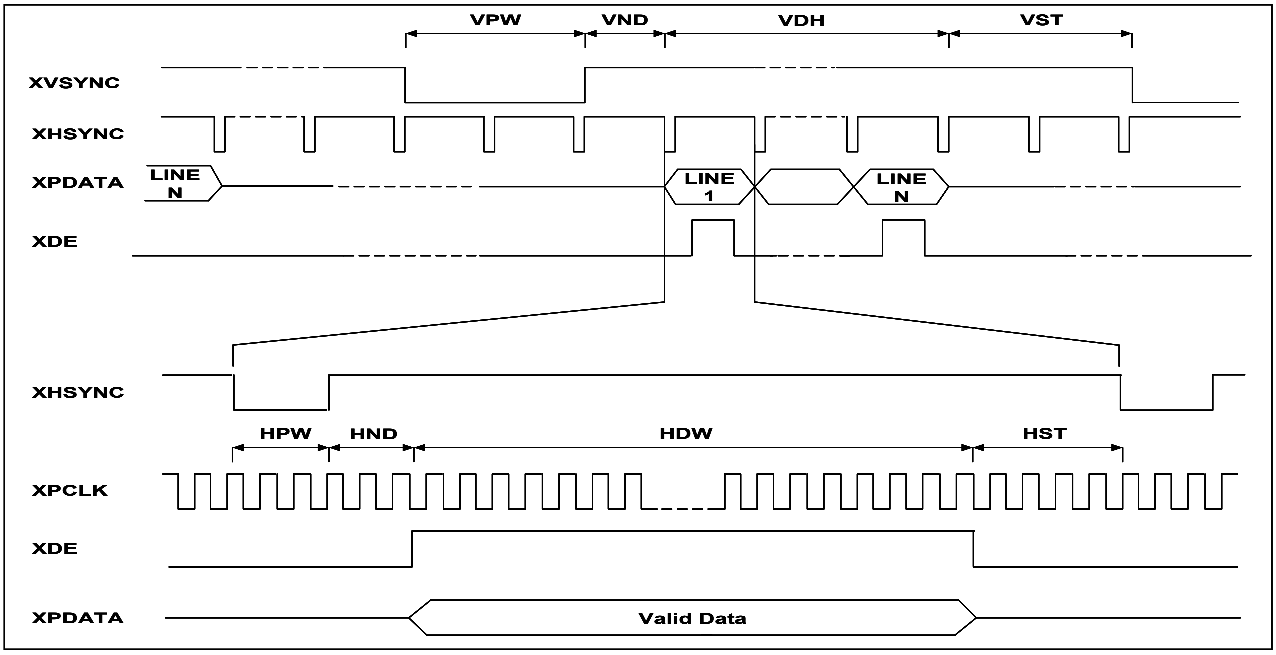
```
lcdRegWrite(RA8889_CCR);//01h
lcdDataWrite(RA8889_PLL_ENABLE<<7|RA8889_WAIT_NO_MASK<<6|RA8889_KEY_SCA
N_DISABLE<<5|RA8889_TFT_OUTPUT24<<3|RA8889_I2C_MASTER_DISABLE<<2|RA888
9_SERIAL_IF_ENABLE<<1|RA8889_HOST_DATA_BUS_SERIAL);

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 timing setting

According to the TFT LCD's datasheet, the relevant timing conditions should be set for RA8889 as below. The following definitions are defined in the RA8889_Lite.h.

```
#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
```

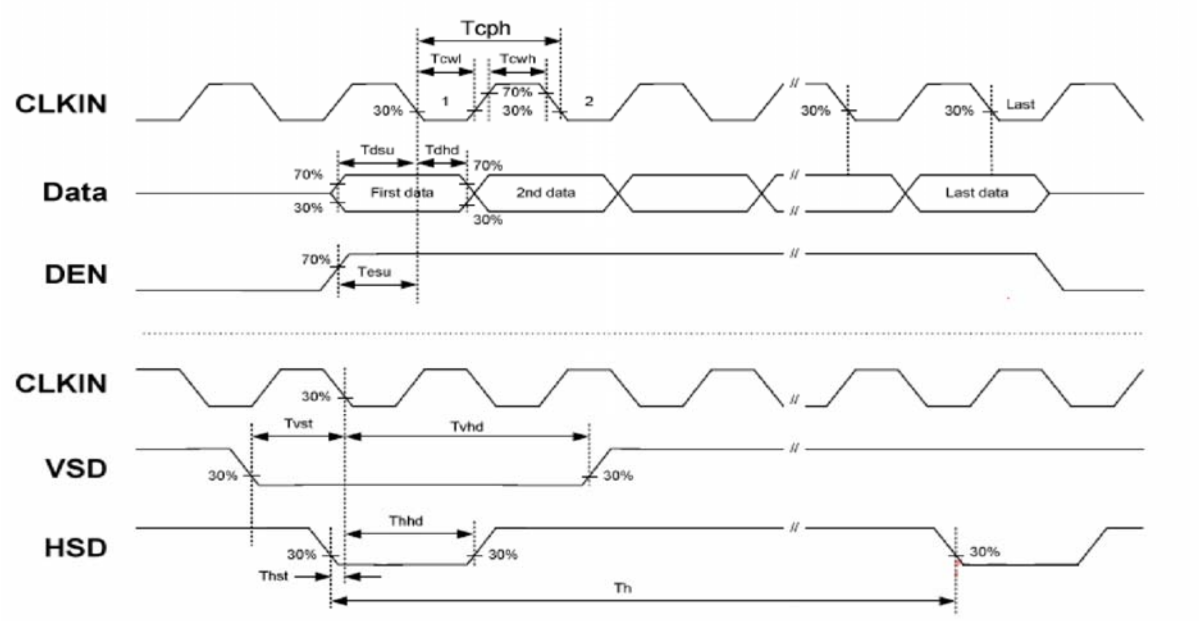
RA8889 Output Timing Reference



The TFT LCD AT080TN52, TFT timing requirements as below:

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 timing initialization setup program:

```
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 Image display memory initialization setting

Please refer to RA8889_Lite.h for the related definitions as the following, user need to define the following values:

```
// define screen resolution
#define SCREEN_WIDTH 800
#define SCREEN_HEIGHT 600

// user image memory buffer page configure
// the maximum number of pages depending on the capacity of the SDRAM and what the page
//use of color depth, width, height.
// for example, the SDRAM capacity = 16Mbyte
//page_size = 800*600*3byte(24bpp) = 1440000byte
//maximum number = 16/1.44 = 11.11
//so maximum configure page is 11 for application
// this article is configure 10 pages to display applications such as the following, the size of
//each page is the same to the display size 800 * 600, 24bpp color depth, that is configure for
//vertical direction and mulit-page buffer application

#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
```

Windows initialization program:

```
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 display on

After running the RA8889 initialization setting, the next step, we will usually executes writing image data into display memory firstly, then turn the display on. With the operations above, the TFT LCD timing controller of RA8889 will automatically fetch the image data from the display windows block of the image display memory and then output to the LCD to display, after turning on the display.

displayOn()

Description:

Display on/off.

Function prototype:

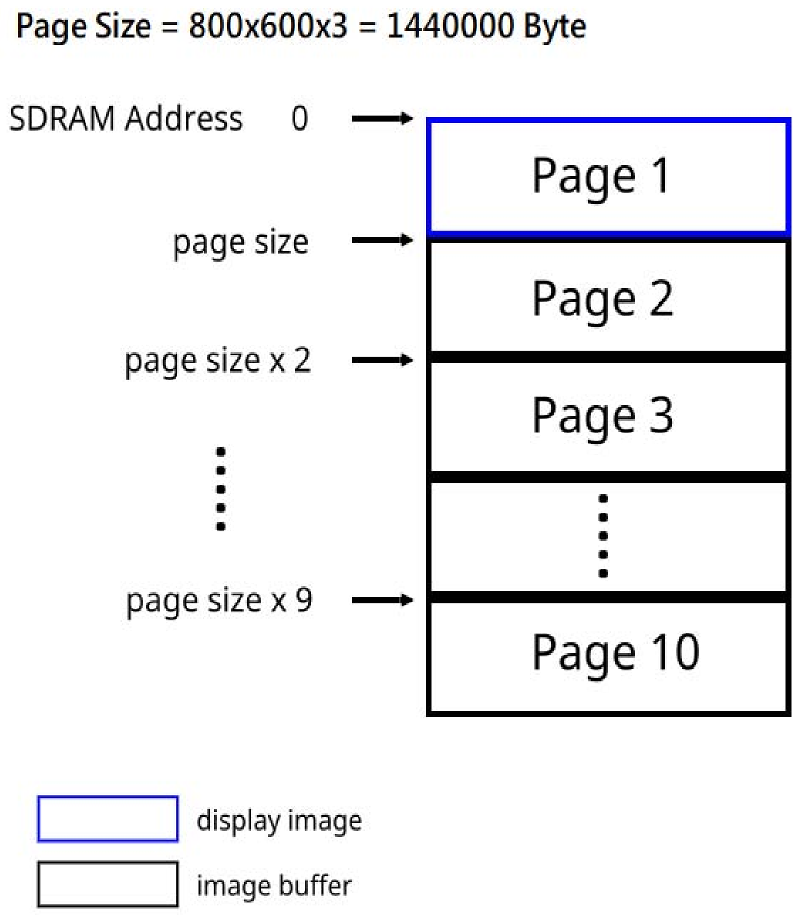
```
void displayOn(boolean on);
```

Parameter	Description
on	= true Display on = false Display off

Chapter 3 Memory Configuration & Window

In this document, the memory be configured to 10 pages, the first page is assigned to image display memory, the others are used for image buffer; for example, we update image to image buffer page 2, and then use BTE memory copy function, copy the image data from page2 to page1 which we define as the image display memory. This method can avoid to bring on the flicker effect or the overlap effect when updating the display data to image display memory directly.

Memory configuration diagram:



The related functions for Memory and Windows are shown as below:

Function	Description
displayImageStartAddress()	Set the start address of the image display memory
displayImageWidth()	Set the width of image display memory
displayWindowStartXY()	Set the display window start point of the upper left corner of the image display memory
canvasImageStartAddress()	Set the start address of the canvas image memory
canvasImageWidth()	Set the width of the canvas image memory
activeWindowXY()	Set the active window start point of the upper left corner of canvas
activeWindowWH()	Set the width and height of the active window

displayImageStartAddress()

Description:

Set the start address of the image display memory.

Function prototype:

void displayImageStartAddress(ru32 addr);

Parameter	Description
addr	Start address of image display memory

Note and example:

Image display memory is the data source of the display window, the start address is recommended to arrange at address 0. In this document, the memory buffer of RA8889 is configured to 10 pages, the first page is assigned for image display memory, the initialization setting is shown as the following:

```
displayImageStartAddress(PAGE1_START_ADDR);
```

displayImageWidth()

Description:

Set the width of image display memory.

Function prototype:

void displayImageWidth(ru16 width);

Parameter	Description
width	Width of the image display memory

Note and example:

The Width of the image display memory must be set to equal to the page (canvas) width.
Set each page (canvas) width to 800(=SCREEN_WIDTH), so initialization is set as the following:
displayImageWidth(SCREEN_WIDTH);
This function only needs to set one time when we set the initialization of RA8889.

User can also configure the page (canvas) width> SCREEN_WIDTH
For example:
//configure image display page (canvas) start point to address 0 of the memory, width is 1600,
//height is 600.
displayImageStartAddress(0)
displayImageWidth(1600);
*//start point of the memory address of the next page = 1600*600*3(byte)*

displayWindowStartXY()

Description:

Set the coordinate of start point for the display window at the upper left corner of the image display memory.

Function prototype:

void displayWindowStartXY(ru16 x0,ru16 y0);

Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate

Note and example:

Width and height of the display window are referenced to the TFT display timing setting HDW and VDH, so user only need to set start point for display window which is located at the upper left corner of the image display memory.

Setting is shown as the following:
displayWindowStartXY(0,0);

When width and height of the image display memory page (canvas) > width and height of the LCD resolution, the coordinates of "displayWindowStartXY (x,y)" can be changed to the other address but the minimum offset for X-axis is multiples of 4 and the minimum offset for Y-axis is 1.

The corresponding relation between the display window and the current image display memory is like child and parent, the display window (child) is always attached to the current specified image display memory (parent).

The Contents of display window will output to the TFT-LCD display by RA8889 TFT timing controller, after the function "displayOn (true)" is set.

canvasImageStartAddress()

Description:
Set the coordinate of start address for the canvas image memory.

Function prototype:
void canvasImageStartAddress(ru32 addr);

Parameter	Description
addr	Start address of the canvas image memory

canvasImageWidth()

Description:
Set the width of the canvas image memory.

Function prototype:
void canvasImageWidth(ru16 width);

Parameter	Description

width	Width of the canvas image memory

Note and example:

With the functional operations for RA8889 such as Graphic, Text, Draw or DMA, IDEC and so on, all the display manipulations must be executed within the area of the active window, and the active window is located within the current canvas, in this document, the internal memory for RA8889 is configured to 10 pages, all pages can be specified as the current canvas, for example:

```
// specify the page 1 for the current canvas
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.canvasImageWidth(SCREEN_WIDTH);

// specify the page 2 for the current canvas
ra8889lite.canvasImageStartAddress(PAGE2_START_ADDR);

// specify the page 3 for the current canvas
ra8889lite.canvasImageStartAddress(PAGE3_START_ADDR);
```

activeWindowXY()

Description:

Set the coordinate of start point for the active window which is located on the upper left corner of canvas.

Function prototype:

void activeWindowXY(ru16 x0,ru16 y0);

Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate

activeWindowWH()

Description:

Set the width and height of the active window.

Function prototype:

void activeWindowWH(ru16 width,ru16 height);

Parameter	Description
width	Width of the active window
height	height of the active window

Note and example:

With the functional operations for RA8889 such as Graphic, Text, Draw or DMA, IDEC and so on, all the display manipulations must be executed within the area of the active window, and the active window is located within the current canvas. The corresponding relation between the active window and the current canvas is like child and parent, the active window (child) is always attached to the current canvas (parent).

Chapter 4 Graphic

Function	Description
graphicMode()	Switch to graphics mode or text mode
setPixelCursor()	Set the pixel cursor coordinate
ramAccessPrepare()	Pre-instruction for the memory access
putPixel_24bpp()	Draw a pixel at the specified coordinate
putPicture_24bpp()	Specify coordinate and width, height and then write image data
putPicture_24bpp()	Specify coordinate and width, height image data pointer (Byte format)

Note:

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg*" or appendix Figure A-1.

All the image data in this section, are converted by using "*Image_Tool_V1.0*" image tool.

graphicMode()

Description:

Option for selecting that RA8889 is worked in the graphics mode or text mode.

Function prototype:

void graphicMode(boolean on);

Parameter	Description
on	= true Set to graphic mode = false Set to Text mode

Note:

The default value for RA8889 is stayed in graphic mode.

setPixelCursor()

Description:

Set the pixel cursor’s coordinate.

Function prototype:

void setPixelCursor(ru16 x,ru16 y);

Parameter	Description
x	X-axis coordinate
y	Y-axis coordinate

ramAccessPrepare()

Description:

Pre-instruction for the memory access

Function prototype:

void ramAccessPrepare(void);

Note:

This function must be called before the memory access.

putPixel_24bpp()

Description:

Draw a pixel at the specified coordinate.

Function prototype:

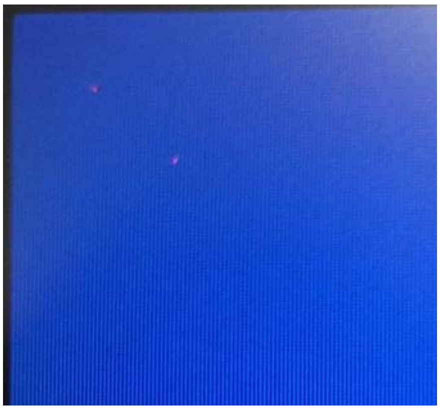
void putPixel_24bpp(ru16 x,ru16 y,ru32 color);

Parameter	Description
x	X-axis coordinate
y	Y-axis coordinate
color	RGB888 data

Note and example:

```
 //clean current canvas page1 specified active window to color blue
   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.setPixelCursor(20,20);
   ra8889lite.ramAccessPrepare();
   ra8889lite.lcdDataWrite(0x00);//RGB888 Blue data
   ra8889lite.lcdDataWrite(0x00);//RGB888 Green data
   ra8889lite.lcdDataWrite(0xff);//RGB888 Red data

   ra8889lite.putPixel_24bpp(40,40,COLOR16M_MAGENTA);
```

Screenshot of the example:



putPicture_24bpp()

Description:

Set the start coordinate for the upper left corner of wanted image data, and then set the width and height for the intended image, after setting the relevant parameters, user will is able to proceed with writing image data.

Function prototype:

void putPicture_16bpp(ru16 x,ru16 y,ru16 width, ru16 height);

Parameter	Description
X	Upper left corner X-axis coordinate
Y	Upper left corner Y-axis coordinate
Width	Image width(horizontal pixel size)
Height	Image height(vertical pixel size)

putPicture_24bpp()

Description:

Set the coordinate, width and height of the image and the image data pointer (Byte format), after the previous settings, the function will depend on the data pointer, and then it will start to write the image data automatically to the specified address, besides, the specified address should be defined within the current active window of the current canvas.

Function prototype:

void putPicture_16bpp(ru16 x,ru16 y,ru16 width, ru16 height, const unsigned char *data);

Parameter	Description
x	Upper left corner X-axis coordinate
y	Upper left corner Y-axis coordinate
width	Image width(horizontal pixel size)
height	Image height(vertical pixel size)
*data	Byte format image data pointer

Note:

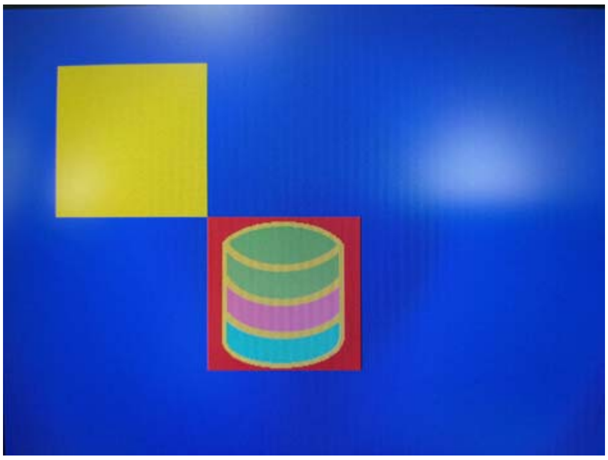
All the image data in this document , are converted by using “*Image_Tool_V1.0*” image tool.

Note and example:

```
//clean current canvas page1 specify active window to color blue
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);
```

```
//write 128*128 image to the specified coordinate of the current canvas
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
 }
 // write 128*128 image to the specified coordinate of the current canvas
ra8889lite.putPicture_24bpp(50+128+128,50+128+128,128,128, pic24bpp_1);
```

Screenshot of the example:



Additional functions and examples

Function	Description
lcdPutChar8x12()	Draw 8x12 ASCII character
lcdPutString8x12()	Draw 8x12 ASCII string
lcdPutChar16x24()	Draw 16x24 ASCII character
lcdPutString16x24()	Draw 16x24 ASCII string
lcdPutChar32x48()	Draw 32x48 ASCII character
lcdPutString32x48()	Draw 32x48 ASCII string

Note:

Please refer to the file “*RA8889_Lite_Graphic.ino”* for getting the relevant information of the above functions. If user needs the functions for their display requirement, please migrate the needed functions or program to their own firmware project.

lcdPutChar8x12()
lcdPutChar16x24()
lcdPutChar32x48()

Description:
Show ASCII character at specified coordinate which is located in the current active window of the current canvas.

Function prototype:
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);

Parameter	Description
x	Upper left corner X-axis coordinate
y	Upper left corner Y-axis coordinate
fgcolor	Text foreground color
bgcolor	Text background color
bg_transparent	= ture : select background transparent, =false : select background color
code	ASCII code

lcdPutString8x12()
lcdPutString16x24()
lcdPutString32x48()

Description:
Show ASCII string at specified coordinate which is located in the current active window of the current canvas.

Function prototype:

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)

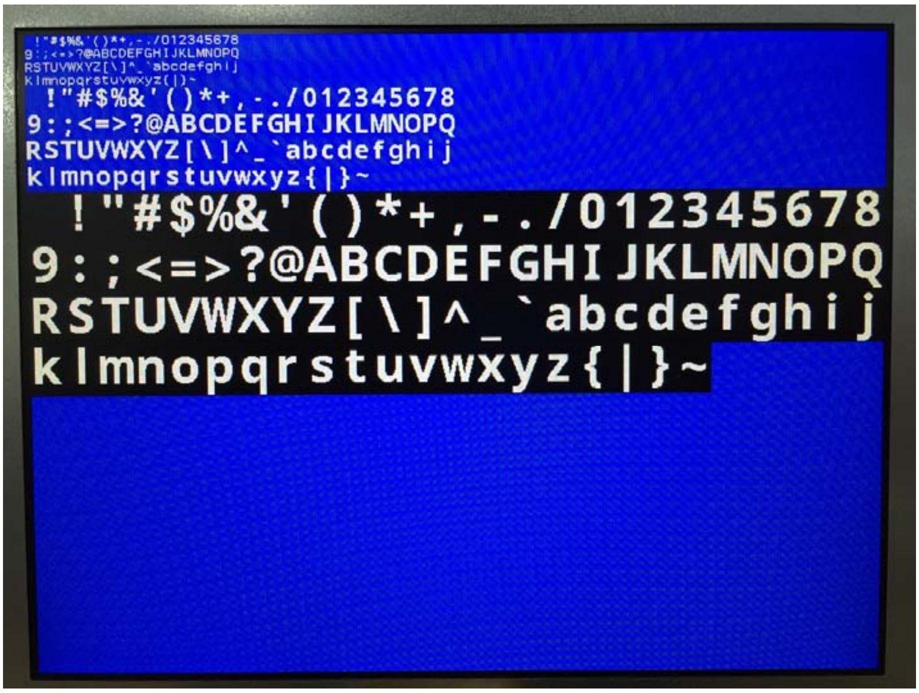
Parameter	Description
x	Upper left corner X-axis coordinate
y	Upper left corner Y-axis coordinate
fgcolor	Text foreground color
bgcolor	Text background color
bg_transparent	= ture : select background transparent , =false : select background color
*ptr	String or data pointer

Note and example:

```
//clean current canvas page1 specified active window to color blue
  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);

// draw 8*12 ASCII character to specified coordinate in the active window of the current canvas.
#ifdef DEMO_ASCII_8X12
  lcdPutString8x12(0,0,0xFFFF,0x0000,true," !\"#$%&'()*+,-./012345678");
  lcdPutString8x12(0,12,0xFFFF,0x0000,true,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
  lcdPutString8x12(0,24,0xFFFF,0x0000,true,"RSTUVWXYZ[\\]^_`abcdefghij");
  lcdPutString8x12(0,36,0xFFFF,0x0000,true,"klmnopqrstuvwxyz{|}~");
#endif
```

```
// draw 16*24 ASCII character to specified coordinate in the active window of the current
//canvas.
#ifdef DEMO_ASCII_16X24
   lcdPutString16x24(0,48,0xFFFF,0x0000,true," !\"#$%&'()*+,-./012345678");
   lcdPutString16x24(0,72,0xFFFF,0x0000,true,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
   lcdPutString16x24(0,96,0xFFFF,0x0000,true,"RSTUVWXYZ[\\]^_`abcdefghij");
   lcdPutString16x24(0,120,0xFFFF,0x0000,true,"klmnopqrstuvwxyz{|}~");
#endif

// draw 32*48 ASCII character to specified coordinate in the active window of the current
//canvas.

#ifdef DEMO_ASCII_32X48
   lcdPutString32x48(0,144,0xFFFF,0x0000,false," !\"#$%&'()*+,-./012345678");
   lcdPutString32x48(0,192,0xFFFF,0x0000,false,"9:;<=>?@ABCDEFGHIJKLMNOPQ");
   lcdPutString32x48(0,240,0xFFFF,0x0000,false,"RSTUVWXYZ[\\]^_`abcdefghij");

  lcdPutString32x48(0,288,0xFFFF,0x0000,false,"klmnopqrstuvwxyz{|}~");
#endif
```

Screenshot of the example:



```
 !"#$%&'()*+,-./012345678
9:;<=>?@ABCDEFGHIJKLMNOPQ
RSTUVWXYZ[\]^_`abcdefghij
klmnopqrstuvwxyz{|}~
 !"#$%&'()*+,-./012345678
9:;<=>?@ABCDEFGHIJKLMNOPQ
RSTUVWXYZ[\]^_`abcdefghij
klmnopqrstuvwxyz{|}~
 !"#$%&'()*+,-./012345678
9:;<=>?@ABCDEFGHIJKLMNOPQ
RSTUVWXYZ[\]^_`abcdefghij
klmnopqrstuvwxyz{|}~
```

Chapter 5 Text and Value

Function	Description
textMode()	Switch to text mode or graphic mode
textColor()	Set the text foreground color and background color
setTextCursor()	Set the text cursor coordinate
setTextParameter1()	Set the text function parameter1
setTextParameter2()	Set the text function parameter2
genitopCharacterRomParameter()	Set the Genitop font function parameter
putString()	Write string to specified coordinate
putDec()	Write decimal value to specified coordinate
putFloat()	Write floating value to specified coordinate
putHex()	Write hexadecimal value to specified coordinate

Note:

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg*" or appendix Figure A-1.

textMode()

Description:

Option for selecting that RA8889 is worked in the graphics mode or text mode.

Function prototype:

void textMode (boolean on);

Parameter	Description
on	= true Set to text mode = false Set to graphic mode

Note:

It is recommended that set the operating mode of RA8889 back to the graphic mode after each time user finish~~ed~~ the text mode operation in text mode.

textColor()

Description:

Set the foreground color and the background color for the displayed text.

Function prototype:

void textColor(ru32 foreground_color, ru32 background_color);

Parameter	Description
foreground_color	Color for text foreground
background_color	Color for text background

setTextCursor()

Description:

Set the coordinate for text cursor .

Function prototype:

void setTextCursor(ru16 x, ru16 y);

Parameter	Description
x	X-axis coordinate
y	Y-axis coordinate

setTextParameter1()

Description:

Set the text function's parameter1.

Function prototype:

void setTextParameter1(ru8 source_select, ru8 size_select, ru8 iso_select);

Parameter	Description
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()

Description:

Set the text function’s parameter2.

Function prototype:

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

Parameter	Description
align	RA8889_TEXT_FULL_ALIGN_DISABLE RA8889_TEXT_FULL_ALIGN_ENABLE Full-width font aligment enable bit
chroma_key	RA8889_TEXT_CHROMA_KEY_DISABLE RA8889_TEXT_CHROMA_KEY_ENABLE Text background color transparent enable bit
width_enlarge	RA8889_TEXT_WIDTH_ENLARGEMENT_X1 RA8889_TEXT_WIDTH_ENLARGEMENT_X2 RA8889_TEXT_WIDTH_ENLARGEMENT_X3 RA8889_TEXT_WIDTH_ENLARGEMENT_X4 Text horizontal enlarge select
height_enlarge	RA8889_TEXT_HEIGHT_ENLARGEMENT_X1 RA8889_TEXT_HEIGHT_ENLARGEMENT_X2 RA8889_TEXT_HEIGHT_ENLARGEMENT_X3 RA8889_TEXT_HEIGHT_ENLARGEMENT_X4 Text vertical enlarge select

genitopCharacterRomParameter()

Description:

Set the parameters for the Genitop font function.

Function prototype:

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

Parameter	Description
scs_select	RA8889_SERIAL_FLASH_SELECT0 RA8889_SERIAL_FLASH_SELECT1 Select use SPI0 or SPI1
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4 RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 Set Genitop font SPI clock divider
rom_select	RA8889_GT21L16T1W RA8889_GT30L16U2W RA8889_GT30L24T3Y RA8889_GT30L24M1Z RA8889_GT30L32S4W RA8889_GT20L24F6Y RA8889_GT21L24S1W Select Genitop font
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 Select font decoder
gt_width	RA8889_GT_FIXED_WIDTH RA8889_GT_VARIABLE_WIDTH_ARIAL RA8889_GT_VARIABLE_FIXED_WIDTH_ROMAN RA8889_GT_BOLD Select font

Note:

It is recommended to use the serial IF0 (xnsfcs0)for the GENITOP's font ROM, and use the serial IF1 (xnsfcs1)for the serial flash memory, for more detailed information, please refer to the datasheet of RA8889.

putString()

Description:

Write a string to specified coordinate within the current active window of the current canvas.

Function prototype:

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

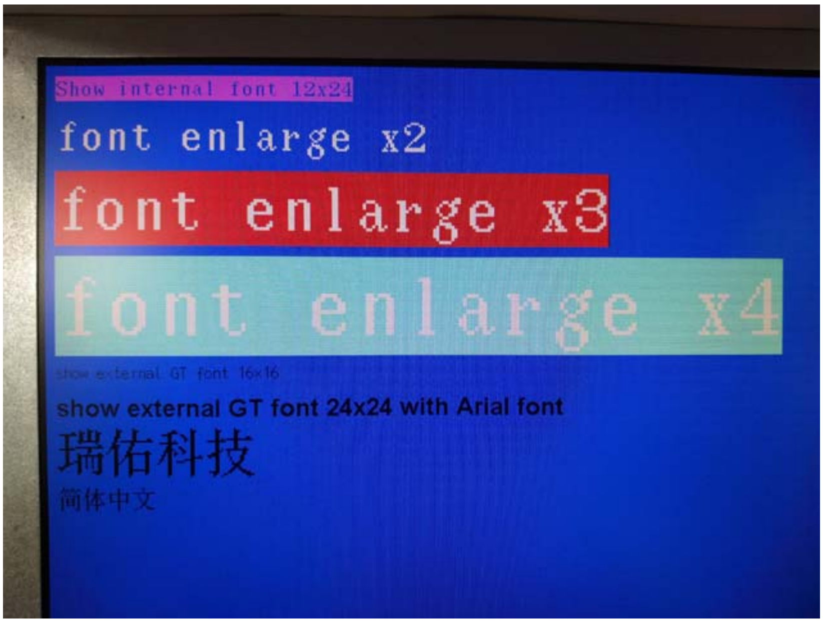
Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate
*str	String or data pointer

Example:

```
//clean current canvas page1 specified active window to color blue
  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);

  //set the text function parameter
  //set the text color
  //write build-in font 12x24 ASCII string to specified coordinate
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");

// set the text function parameter
// set the text color
// write build-in font and enlarge 2 times to specified coordinate
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");

// set the text function parameter
// set the text color
// write build-in font and enlarge 3 times to specified coordinate
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");

// set the text function parameter
// set the text color
// write build-in font and enlarge 4 times to specified coordinate
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");

// set the text function parameter
// set the Genitop font function parameter
// set the text color
// write string of the Genitop font to specified coordinate
  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");

// set the text function parameter
// set the Genitop font function parameter
// set the text color
// write string of the Genitop font to specified coordinate
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,RA8889_TEXT_HEIGHT_ENLARGEMENT_X2);

ra8889lite.genitopCharacterRomParameter(RA8889_SERIAL_FLASH_SELECT0,RA8889_SPI_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_HEIGHT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8889_TEXT_HEIGHT_ENLARGEMENT_X1);

ra8889lite.genitopCharacterRomParameter(RA8889_SERIAL_FLASH_SELECT0,RA8889_SPI_DIV4,RA8889_GT30L24T3Y,RA8889_GB2312,RA8889_GT_FIXED_WIDTH);
ra8889lite.putString(10,408,string2);
```

Screenshot of the example:



```
Show internal font 12x24
font enlarge x2
font enlarge x3
font enlarge x4
show external GT font 16x16
show external GT font 24x24 with Arial font
瑞佑科技
简体中文
```

putDec()

Description:

Write decimal number to specified coordinate within the current active window of the current canvas.

Function prototype:

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

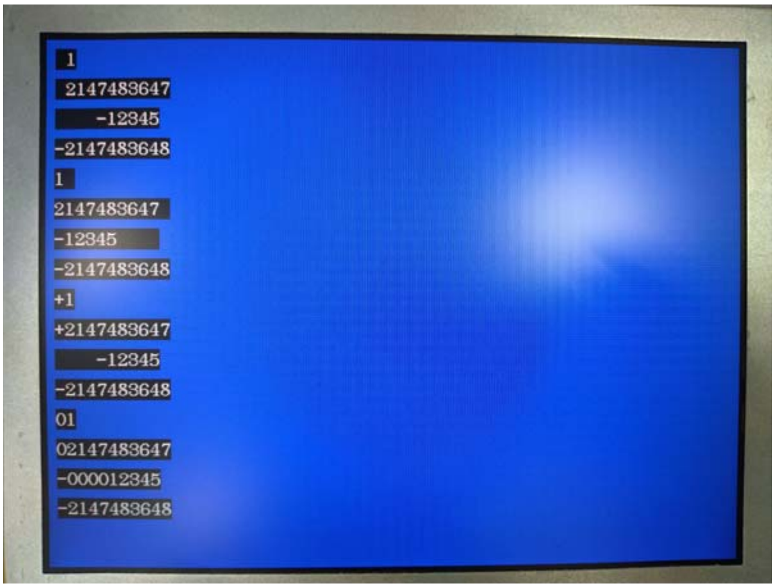
Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate
vaule	Input value -2147483648(-2^31) ~ 2147483647(2^31-1)
len	Minimum display number of bits(1~11)
*flag	= “n” : Display to the right = “-” : Display to the left = “+” : Output sign = “0” : fill 0 at the beginning, not fill space

Example:

```
//clean current canvas page1 specified active window to color blue
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);

// set text function parameter
// set text color
//write build-in font 12x24 ASCII string to specified coordinate
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIGHT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

//display value
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");
```

Screenshot of the example:



```
 1
 2147483647
     -12345
-2147483648
1
2147483647
-12345
-2147483648
+1
+2147483647
     -12345
-2147483648
01
02147483647
-000012345
-2147483648
```

putFloat()

Description:

Write floating value to specified coordinate within the current active window of the current canvas.

Function prototype:

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

Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate
vaule	Input value (3.4E-38 ~ 3.4E38)
len	Minimum display number of bits (1~11)
precision	The precise number of bits to the right of the decimal point (1~4bits)
*flag	= "n" : Display to the right = "-" : Display to the left = "+" : Output sign = "0" : fill 0 at the beginning, not fill space

Note:

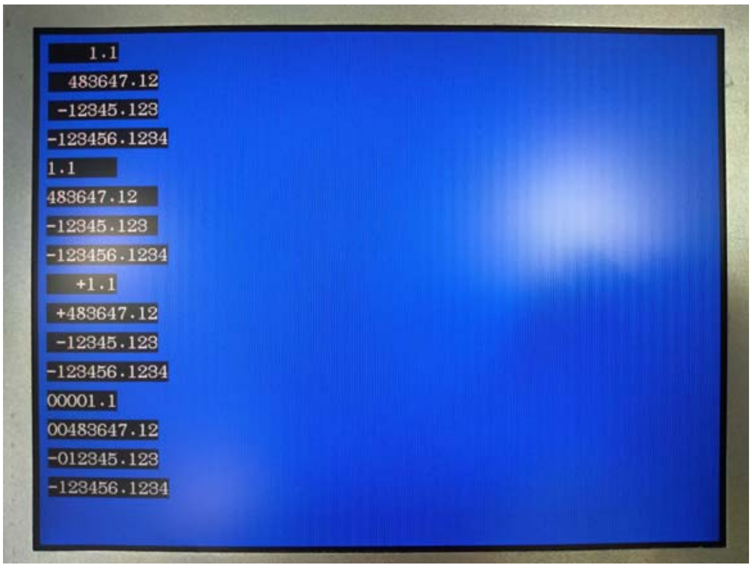
Use a variable "double" for getting more precise accuracy.

Example:

```
//clean current canvas page1 specified active window to color blue
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);

//set text function parameter
//set text color
//write build-in font 12x24 ASCII string to specified coordinate
ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIGHT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_DISABLE,
RA8889_TEXT_CHROMA_KEY_DISABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);
```

```
//display value
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");
```

Screenshot of the example:



```
    1.1
  483647.12
 -12345.123
-123456.1234
1.1
483647.12
-12345.123
-123456.1234
   +1.1
 +483647.12
 -12345.123
-123456.1234
00001.1
00483647.12
-012345.123
-123456.1234
```

putHex()

Description:

Write hexadecimal value to specify coordinate within the current active window of the current canvas.

Function prototype:

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

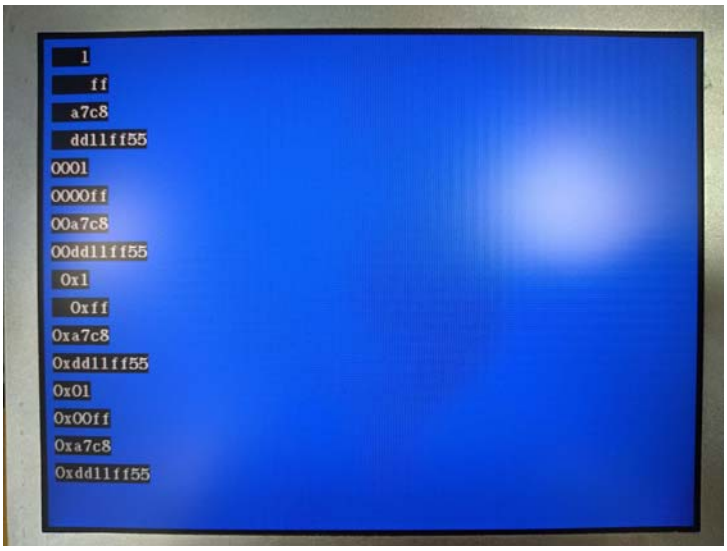
Parameter	Description
x0	Upper left corner X-axis coordinate
y0	Upper left corner Y-axis coordinate
vaule	Input value 0x00000000~0xffffffff
len	Minimum display number of bits (1~10)
*flag	= “n” : Display to the right = “#” : Force output 0x as the beginning = “0” : fill 0 at the beginning, not fill space = “x” : Force output 0x as the beginning，fill 0

Example:

```
//clean current canvas page1 specified active window to color blue
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);

// set text function parameter
// set text color
//write build-in font 12x24 ASCII string to specified coordinate
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);
```

```
//display value
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");
```

Screenshot of the example:



```
   1
    ff
  a7c8
  dd11ff55
0001
0000ff
00a7c8
00dd11ff55
 0x1
  0xff
0xa7c8
0xdd11ff55
0x01
0x00ff
0xa7c8
0xdd11ff55
```

Chapter 6 Geometric Draw

Function	Description
drawLine()	Draw a line
drawSquare()	Draw a square
drawSquareFill()	Draw a square fill
drawCircleSquare()	Draw a circle square
drawCircleSquareFill()	Draw a circle square fill
drawTriangle()	Draw a triangle
drawTriangleFill()	Draw a triangle fill
drawCircle()	Draw a circle
drawCircleFill()	Draw a circle fill
drawEllipse()	Draw a ellipse
drawEllipseFill()	Draw a ellipse fill

Note:

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg"* or appendix Figure A-1.

drawLine()

Description:

Specify any two points to draw a colorful line in the active window of the current canvas.

Function prototype:

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

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
color	Set color(RGB888)

Example:

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

Screenshot of the example:

drawSquare()

Description:

Specify any two points to draw a colorful square in the active window of the current canvas.

Function prototype:

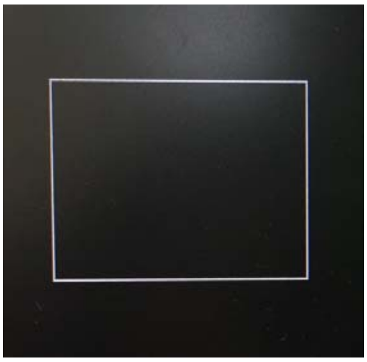
void drawSquare(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru32 color);

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawSquareFill()

Description:

Specify any two points to draw a colorful square fill in the active window of the current canvas.

Function prototype:

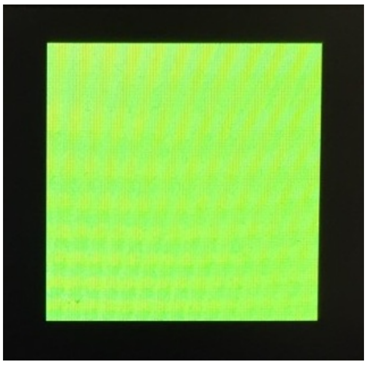
void drawSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru32 color);

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawCircleSquare()

Description:

Specify any two points to draw a colorful circle square in the active window of the current canvas.

Function prototype:

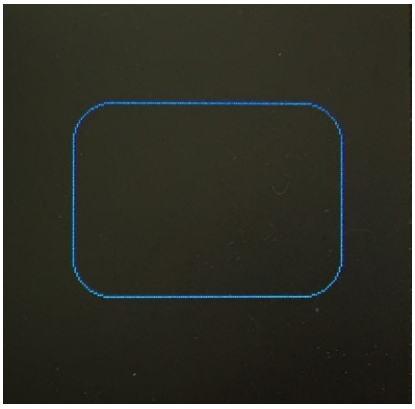
void drawCircleSquare(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 xr, ru16 yr, ru32 color);

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
xr	Horizontal radius of the rounded corner
yr	Vertical radius of the rounded corner
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawCircleSquareFill()

Description:

Specify any two points to draw a colorful circle square fill in the active window of the current canvas.

Function prototype:

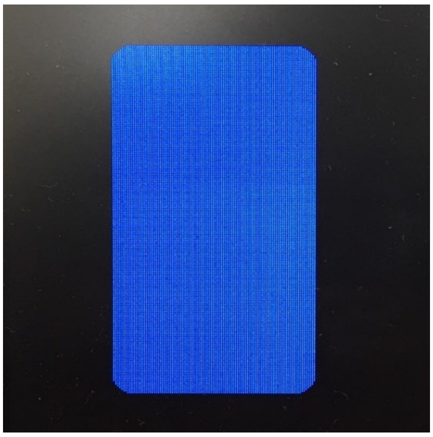
void drawCircleSquareFill(ru16 x0, ru16 y0, ru16 x1, ru16 y1, ru16 xr, ru16 yr, ru32 color);

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
xr	Horizontal radius of the rounded corner
yr	Vertical radius of the rounded corner
color	Set color(RGB888)

Example:

ra8889lite.drawCircleSquareFill(50,200, 149, 399, 10, 10, COLOR65K_BLUE);

Screenshot of the example:



drawTriangle()

Description:

Specify any three points to draw a colorful triangle in the active window of the current canvas.

Function prototype:

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

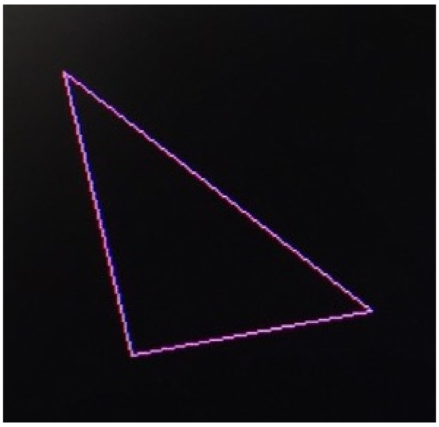
Parameter	Description

x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
x2	X-axis coordinate of point 3
y2	Y-axis coordinate of point 3
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawTriangleFill()

Description:

Specify any three points to draw a colorful triangle fill in the active window of the current canvas.

Function prototype:

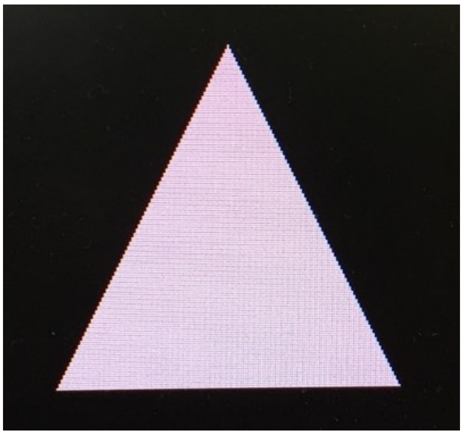
void drawTriangleFill(ru16 x0,ru16 y0,ru16 x1,ru16 y1,ru16 x2,ru16 y2,ru32 color);

Parameter	Description
x0	X-axis coordinate of point 1
y0	Y-axis coordinate of point 1
x1	X-axis coordinate of point 2
y1	Y-axis coordinate of point 2
x2	X-axis coordinate of point 3
y2	Y-axis coordinate of point 3
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawCircle()

Description:

Specify any points as a center and define the radius for drawing a colorful circle in the active window of the current canvas.

Function prototype:

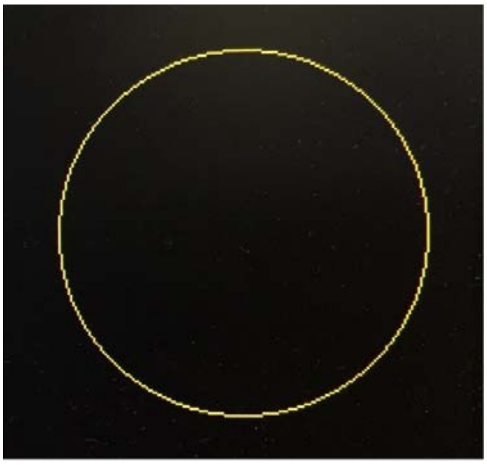
void drawCircle(ru16 x0,ru16 y0,ru16 r,ru32 color);

Parameter	Description
x0	X-axis coordinate of the center
y0	Y-axis coordinate of the center
r	Radius
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawCircleFill()

Description:

Specify any points as a center and define the radius for drawing a colorful filled circle in the active window of the current canvas.

Function prototype:

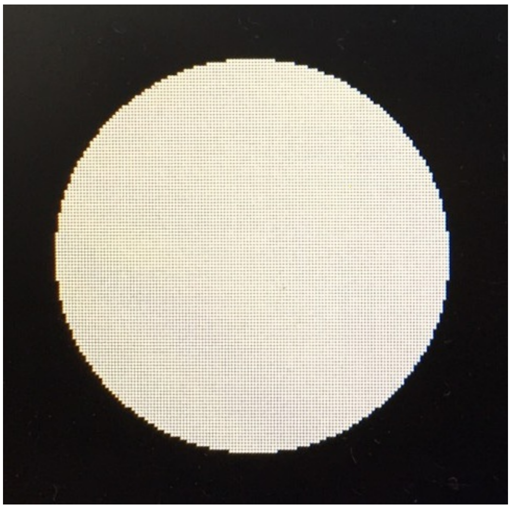
void drawCircleFill(ru16 x0,ru16 y0,ru16 r,ru32 color);

Parameter	Description
x0	X-axis coordinate of the center
y0	Y-axis coordinate of the center
r	Radius
color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawEllipse()

Description:

Specify any points as a center and define the horizontal radius and the vertical radius for drawing a colorful ellipse in the active window of the current canvas.

Function prototype:

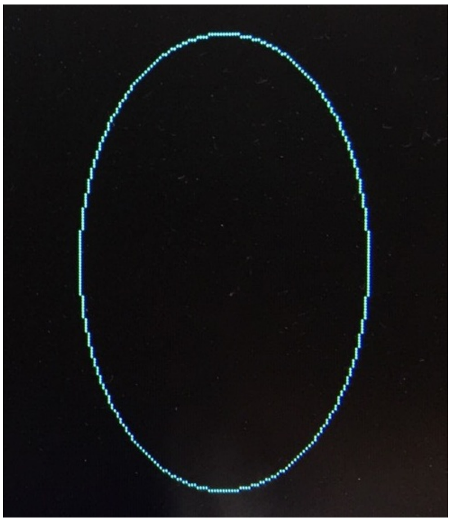
void drawEllipse(ru16 x0,ru16 y0,ru16 xr,ru16 yr,ru32 color);

Parameter	Description
x0	X-axis coordinate of the center
y0	Y-axis coordinate of the center
Xr	Horizontal radius
Yr	Vertical radius
Color	Set color(RGB888)

Example:

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

Screenshot of the example:



drawEllipseFill()

Description:

Specify any points as a center and define the radius for drawing a colorful filled ellipse in the active window of the current canvas.

Function prototype:

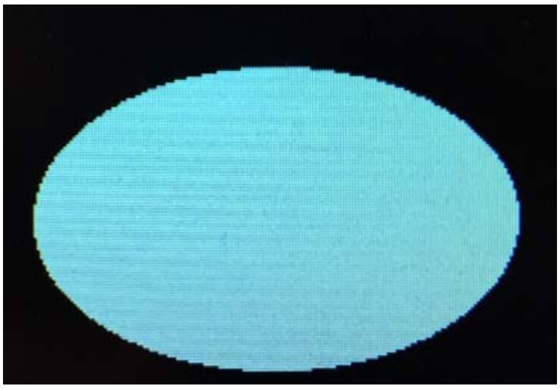
void drawEllipseFill(ru16 x0,ru16 y0,ru16 xr,ru16 yr,ru32 color);

Parameter	Description
x0	X-axis coordinate of the center
y0	Y-axis coordinate of the center
xr	Horizontal radius
yr	Vertical radius
color	Set color(RGB888)

Example:

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

Screenshot of the example:



Chapter 7 BTE

Block Transfer Engine is a 2D acceleration engine for RA8889, it provides so many useful functions, including the fast memory data transfer with copy and logic operation, the chroma key with the color ignoring function and the color expansion with chroma key function which can used to change the monochrome (1bpp) data to colorful data.

The display data is huge for the colorful display system, if the operating speed of MPU writes is not fast enough, we may see the update scan line on the display when the system is updating the display data, the visual effect is looked like the waterfall. For the other application, user might need a dynamic effects for their display requirement, such as the background image keeps still (such as wallpaper), and the foreground text or image is changed as time goes on, traditionally with this kind of display request, the host system must re-write the background data and then update the foreground text or the image data over and over again. Besides, if we directly change the current contents of the display memory, it will lead to the screen flicker effect which is caused by updating huge display data. If user update foreground text or image data directly without re-write the background data, this operation will cause the image overlay issue, so if we want to get a better display effect, we can try a more easily using the BTE function of RA8889, the image data can be written to the non-display area of the display memory through the MPU interface or DMA /IDEC function firstly, and then use the memory copy of BTE function to duplicate and move the image data to the display memory area, this manipulation can avoid the bad display effect which is described above.

Color expansion function can convert monochrome data like 0 or 1 to the specifically colorful data, due to the MPU's ROM is limited, typically is under 512Kbyte, if we convert the image data from 16bpp to 1bpp format and store the converted image data into the MPU’s ROM, therefore we can reduce the ROM usage of MPU/MCU. For example, users may need 64 * 128 resolution numeric digits 0-9 for display; they can convert the numeric image data to 1bpp data format, and store into the MPU's ROM. If we want to show a colorful and customized number on the display, we can use the color expansion function, the color expansion with BTE function will automatically take the image data from the MPU/MCU’s ROM, converting the monochrome image data to specified colorful image data, and write the colorful image data into the memory of RA8889.

The detailed information for all of BTE functions, please refer to the description in the following sections, or refer to the datasheet.

Function	Description
bteMemoryCopy()	Memory data copy and move
bteMemoryCopyWithROP()	Memory data copy and move with logic operation
bteMemoryCopyWithChromakey()	Memory data copy and move with chroma key color ignore
bteMpuWriteWithROP()	MPU write data with logic operation(included data pointer ,Byte format)
bteMpuWriteWithROP()	MPU write data with logic operation(included data pointer, Word format)
bteMpuWriteWithROP()	MPU write data with logic operation
bteMpuWriteWithChromaKey()	MPU write data with chroma key color ignore(included data pointer ,Byte format)
bteMpuWriteWithChromaKey()	MPU write data with chroma key color ignore(included data pointer, Word format)
bteMpuWriteWithChromaKey()	MPU write data with chroma key color ignore
bteMpuWriteColorExpansion()	MPU write data with color expansion(included data pointer)
bteMpuWriteColorExpansion()	MPU write data with color expansion
bteMpuWriteColorExpansionWithChromaKey()	MPU write data with color expansion and chroma key color ignore (included data pointer)
bteMpuWriteColorExpansionWithChromaKey()	MPU write data with color expansion and chroma key color ignore

Note:

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg"* or appendix Figure A-1.

bteMemoryCopy()

Description:

Perform memory data copy means that duplicate the memory data from the specified memory source to the specified memory destination, the moving range for the memory data is specified within the current canvas or is specified between two canvases.

Function prototype:

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);

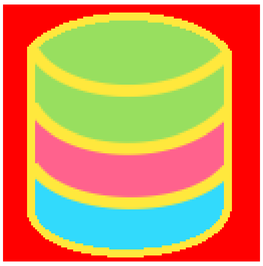
Parameter	Description
s0_addr	Start address memory of the source 0 canvas
s0_image_width	Width of the image memory of the source 0 canvas
s0_x	Source 0 image X-axis coordinate of the canvas
s0_y	Source 0 image Y-axis coordinate of the canvas
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
copy_width	Image width for copy
copy_height	Image height for copy

Note:

All the image data in this document, are converted by using “*Image_Tool_V1.0*” image tool.

Reference picture:

pic24bpp_1.bmp



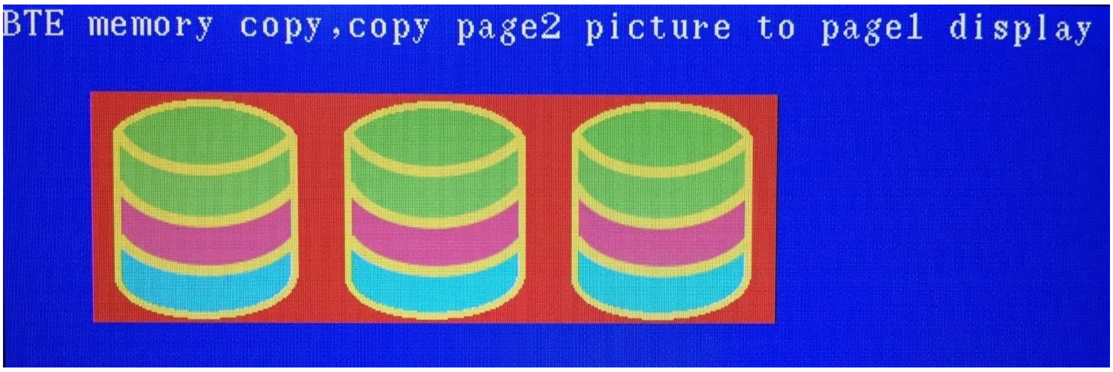
Before performing the following example, we will need an image data source, so user should prepare a converted 24bpp image data file (such as pic24bpp_1.h) , include the relevant header files to the main programming project and then we can start the related operations as below..

Example:

```
//clean current canvas page1 specified active window to color blue
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);
```

```
//clean current canvas page2 specified active window to color red
ra8889lite.canvasImageStartAddress(PAGE2_START_ADDR);
ra8889lite.drawSquareFill(0, 0, SCREEN_WIDTH-1, SCREEN_HEIGHT-1, COLOR16M_RED);

//write image data to current canvas page2 specified position
ra8889lite.putPicture_24bpp(50,50,128,128, pic24bpp_1);

//write string to current canvas page1 specified position
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIGHT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_ENABLE,
RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.putString(0,0,"BTE memory copy,copy page2 picture to page1 display");

//copy image data from page2 canvas(source) and written to page1 canvas (destination)
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);
```

Screenshot of the example:



BTE memory copy,copy page2 picture to page1 display

bteMemoryCopyWithROP()

Description:

Perform the memory data copy with ROP function means that duplicate the memory data from specified memory source to the specified memory destination with the ROP logic operation, the memory moving range is specified within the current canvas or it is specified between two canvases.

Function prototype:

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);

Parameter	Description
s0_addr	Start address of the memory of the source 0 canvas
s0_image_width	Width of the image memory of the source 0 canvas
s0_x	Source 0 image X-axis coordinate of the canvas
s0_y	Source 0 image Y-axis coordinate of the canvas
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
copy_width	Image width for copy
copy_height	Image height for copy
rop_code	Select of the logic operation 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

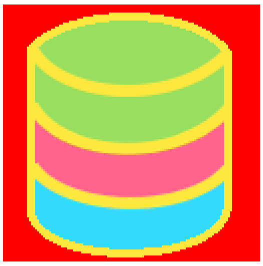
	RA8889_BTE_ROP_CODE_6 S0^S1 RA8889_BTE_ROP_CODE_7 ~S0+~S1 or ~ (S0・S1) RA8889_BTE_ROP_CODE_8 S0・S1 RA8889_BTE_ROP_CODE_9 ~ (S0^S1) RA8889_BTE_ROP_CODE_10 S1 RA8889_BTE_ROP_CODE_11 ~S0+S1 RA8889_BTE_ROP_CODE_12 S0 RA8889_BTE_ROP_CODE_13 S0+~S1 RA8889_BTE_ROP_CODE_14 S0+S1 RA8889_BTE_ROP_CODE_15 (Whiteness)

Note:

All the image data in this document, are converted by using “*Image_Tool_V1.0*” image tool.

Reference picture:

pic24bpp_1.bmp

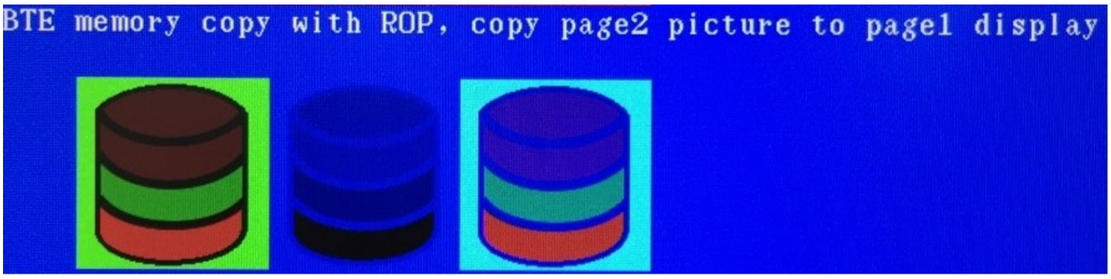


Before performing the following example, we will need an image data source, so user should prepare a converted 24bpp image data file (such as pic24bpp_1.h) , include the relevant header files to the main programming project and then we can start the related operations as below..

Example:

```
//write string to current canvas page1 specified position
ra8889lite.canvasImageStartAddress(PAGE1_START_ADDR);
ra8889lite.putString(0,178,"BTE memory copy with ROP, copy page2 picture to page1 display");

//copy image data from page2 canvas(source) and logic operation with page1
//canvas(destination) and then written to page1 canvas (destination)
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);
```

Screenshot of the example:



BTE memory copy with ROP, copy page2 picture to page1 display

bteMemoryCopyWithChromaKey()

Description:

Perform the memory data copy with chroma key function, the chroma key means that RA8889 will ignore the indicated background data and the memory data copy function will move the foreground data from the specified memory source to the specified memory destination. The moving range for the memory copy is specified within the current canvas or is specified between the two canvases.

Function prototype:

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);

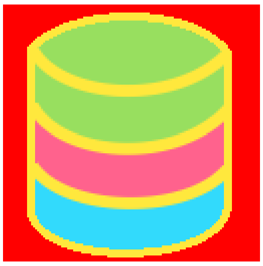
Parameter	Description
s0_addr	Start address of the memory of the source 0 canvas
s0_image_width	Width of the image memory of the source 0 canvas
s0_x	Source 0 image X-axis coordinate of the canvas
s0_y	Source 0 image Y-axis coordinate of the canvas
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
copy_width	Image width for copy
copy_height	Image height for copy
chromakey_color	Data of chroma key color

Note:

All the image data in this document, are converted by using "*Image_Tool_V1.0*" image tool.

Reference picture:

pic24bpp_1.bmp



Before performing the following example, we will need an image data source, so user should prepare a converted 24bpp image data file (such as pic24bpp_1.h) and then include the relevant header files to the main programming project and then we can start the related operations as below..

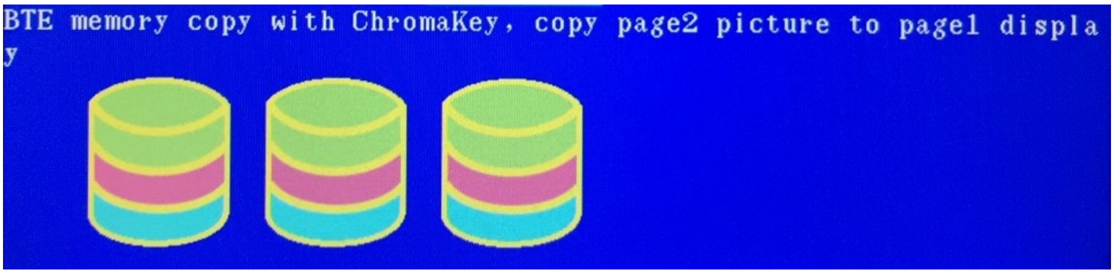
Example:

//write string to current canvas page1 specified position

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

```
//copy image data from page2 canvas(source) and then written to page1 canvas (destination)
//with chroma key color ignore.
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);
```

Screenshot of the example:



BTE memory copy with ChromaKey, copy page2 picture to page1 displa
y

bteMpuWriteWithROP()

Description:

For this function, the image data are written by MCU and provided from the source 0, these data from source 0 will be performed the logic operation with the source1 image data, and then the operating results will be moved into the specified memory destination.

Function prototype:

```
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);
```

Parameter	Description
s1_addr	Start address of the memory of the source 1 canvas
s1_image_width	Width of the image memory of the source 1 canvas
s1_x	Source 1 image X-axis coordinate of the canvas
s1_y	Source 1 image Y-axis coordinate of the canvas
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
width	Image width for write
height	Image height for write
rop_code	Select of the logic operation 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	Data pointer (Byte or Word format)

Note:

BTE function with MPU data write, S0 (Source0) = MPU data write.

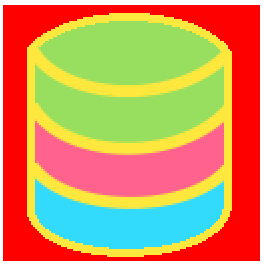
S1 (Source1) can be set the same as Des (destination).

User can continuously write the image data after calling the function which without pointer.

All the image data in this document , are converted by using “*Image_Tool_V1.0*” image tool.

Reference picture:

Pic24bpp_1.bmp



Pic24bpp_2.bmp

Before performing the following example, we will need an image data source, so user should prepare the converted 16bpp image data files (such as pic24bpp_1.h and pic24bpp_2.h) and then include the relevant header files to the main programming project and then we can start the related operations as below..

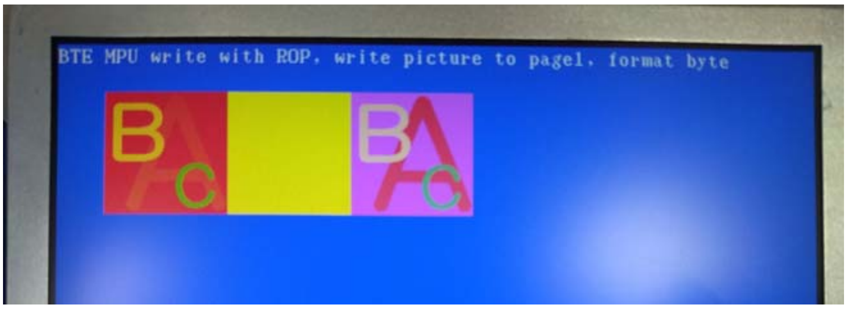
Example:

//clean current canvas page1 specified active window to color blue

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);

//write string to current canvas page1 specified position
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

ra8889lite.setTextParameter1(RA8889_SELECT_INTERNAL_CGROM,RA8889_CHAR_HEIGHT_24,RA8889_SELECT_8859_1);//cch
ra8889lite.setTextParameter2(RA8889_TEXT_FULL_ALIGN_ENABLE, RA8889_TEXT_CHROMA_KEY_ENABLE,RA8889_TEXT_WIDTH_ENLARGEMENT_X1,RA8889_TEXT_HEIGHT_ENLARGEMENT_X1);
ra8889lite.putString(0,0,"BTE MPU write with ROP, write picture to page1, format byte");

//MPU(Source0) written data to destination canvas(Destination) through BTE engine after
//execute logic operation with specified block of canvas(Source1).
ra8889lite.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50,50,PAGE1_START_ADDR,SCREEN_WIDTH,50,50,128,128,RA8889_BTE_ROP_CODE_4, pic24bpp_2);
ra8889lite.bteMpuWriteWithROP(PAGE1_START_ADDR,SCREEN_WIDTH,50+128,50,PAGE1_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,PAGE1_START_ADDR,SCREEN_WIDTH,50+128+128,50,128,128,RA8889_BTE_ROP_CODE_6, pic24bpp_2);
```

Screenshot of the example:



BTE MPU write with ROP, write picture to page1, format byte

bteMpuWriteWithChromaKey()

Description:

MPU write data to the destination with the chroma key function.

Function prototype:

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);

Parameter	Description
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
width	Image width for write
height	Image height for write
chromakey_color	Data of chroma key color
*data	Data pointer

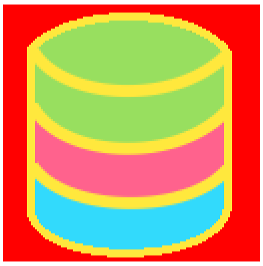
Note:

User can continuously write the image data after calling the function which without pointer.

All the image data in this document, are converted by using "*Image_Tool_V1.0*" image tool.

Reference picture:

Pic24bpp_1.bmp



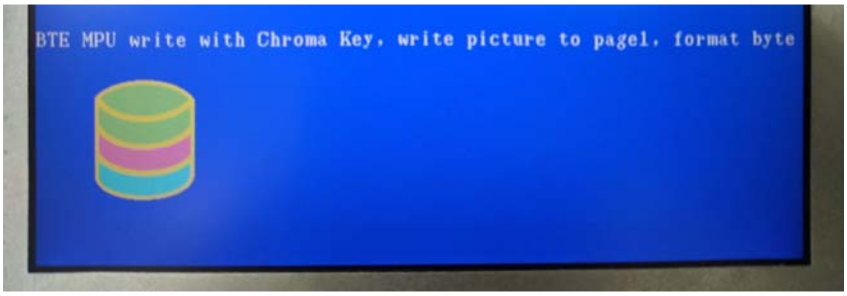
Before performing the following example, we will need an image data source, so user should prepare the converted 24bpp image data files (such as pic24bpp_1.h) and then include the relevant header files to the main programming project and then we can start the related operations as below..

Example:

//write string to current canvas page1 specified position
ra8889lite.putString(0,356,"BTE MPU write with Chroma Key, write picture to page1, format byte");

// MPU write data to destination canvas(page1) through BTE with chroma key color ignore.
ra8889lite.bteMpuWriteWithChromaKey(PAGE1_START_ADDR,SCREEN_WIDTH, 50,406,128,128,0xff0000,pic24bpp_1);

Screenshot of the example:



BTE MPU write with Chroma Key, write picture to page1, format byte

bteMpuWriteColorExpansion()

Description:
MPU writes 1bpp data to the specified block of destination in the canvas using color expansion.

Function prototype:
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);

Parameter	Description
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas

des_y	Destination image Y-axis coordinate of the canvas
width	Image width for write
height	Image height for write
foreground_color	Foreground color
background_color	Background color
*data	Data pointer(Byte format)

Note:

User can continuously write the image data after calling the function which without pointer.

All the image data in this document, are converted by using "*Image_Tool_V1.0*" image tool.

Reference picture:

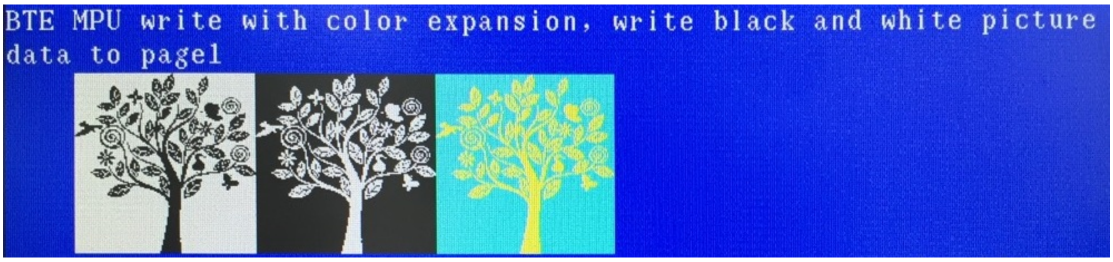
Bw.bmp

Before performing the following example, we will need an 1bpp image data source, so user should prepare a converted 1bpp image data file (such as bw.h) and then include the relevant header file to the main programming project and then we can start the related operations as below..

Example:

```
//clean current canvas page1 specify active window to color blue
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);

//write string to current canvas page1 specified position
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 written 1bpp data to specified block of destination canvas through BTE after excute
//color expansion.
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);
```

Screenshot of the example:



BTE MPU write with color expansion, write black and white picture
data to page1

bteMpuWriteColorExpansionWithChromaKey()

Description:

MPU writes 1bpp data to the specified block of destination in the canvas using color expansion with chroma key function.

Function prototype:

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);

Parameter	Description
des_addr	Start address of the memory of the destination canvas
des_image_width	Width of the image memory of the destination canvas
des_x	Destination image X-axis coordinate of the canvas
des_y	Destination image Y-axis coordinate of the canvas
width	Image width for write
height	Image height for write
foreground_color	Foreground color
background_color	Background color
*data	Data pointer(Byte format)

Note:

The foreground_color and the background_color must be set to the different color data.

User can continuously write the image data after calling the function which without pointer.

All the image data in this document , are converted by using "*Image_Tool_V1.0*" image tool.

Reference picture:

Bw.bmp

Before performing the following example, we will need an 1bpp image data source, so user should prepare a converted 1bpp image data file (such as bw.h) and then include the relevant header file to the main programming project and then we can start the related operations as below..

Example:

//write string to current canvas page1 specified position

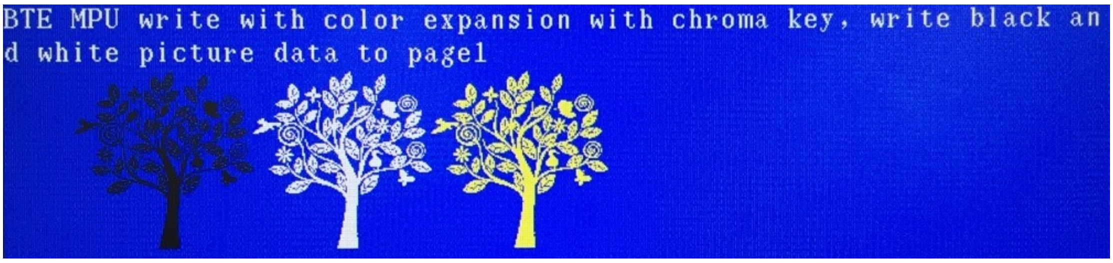
ra8889lite.textColor(COLOR16M_WHITE,COLOR16M_BLACK);

ra8889lite.putString(0,178,"BTE MPU write with color expansion with chroma key, write black and white picture data to page1");

//MPU written 1bpp data to specified block of destination canvas through BTE after execute color //expansion with chroma key

```
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);
```

Screenshot of the example:



BTE MPU write with color expansion with chroma key, write black and white picture data to page1

bteMemoryCopyWith_ARGB8888()

Description：

ARGB image data is copy and move to the specified location on the destination canvas through BTE memory copy with opacity function。

(ARGB image data must be stored in the serial flash first, and use DMA Linear mode function To move the data into specified non-display canvas as the BTE source 1 layer data source)

Function prototype：

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);

Parameter	Description
S1_addr	Start address for Source 1
S1_image_width	The image width of the Source 1
des_addr	Start address for Destination
des_image_width	The image width of the Destination
des_x	X-axis coordinate position for Destination
des_y	Y-axis coordinate position for Destination

copy_width	The width of the copied image
copy_height	The height of the copied image

Note：

All the image data in this document, are converted by using “*Image_Tool_V1.0*” image tool.

Example：

```
//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_PN
G].img_width,BINARY_INFO[Lion_PNG].img_height);
```

Reference picture：

Chapter 8 DMA

RA8889 provides the DMA function, DMA function can read image data from the serial flash connected to the RA8889 and write the image data into to the specified memory block of the canvas quickly, external expansion of serial flash provides a large space to store user's image data, the amount of the color image data is huge, built-in ROM to low-end MPU is usually less than 512Kbyte, so the storing space for MCU can store a small amount of image data only. Besides, the system clock with the low-end MPU, it is usually less than 50MHz, if we get to write big amount of image data to the display system, it ought to take so long time to finish the data movement. So user can get a more efficient way of working with the DMA function of RA8889, this just need to prepare the required image data in advance, and store the image data into the external Serial Flash, and then we can use the DMA function to get a fast image access.

Function	Description
spimSetSerialFlash4BytesMode()	Set serial flash to 4Bytes mode
dma_24bitAddressBlockMode()	DMA read 24bit serial flash, block mode
dma_32bitAddressBlockMode()	DMA read 32bit serial flash, block mode

Note:

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg"* or appendix Figure A-1.

All the image data in this document, are converted by using "*Image_Tool_V1.0*" image tool.

spimSetSerialFlash4BytesMode ()

Description:

User must call the function "spimSetSerialFlash4BytesMode ()" first for setting the serial flash memory as 4Bytes mode when we want to use the 32bit address for serial flash memory,.

Function prototype:

spimSetSerialFlash4BytesMode(ru8 bus_selct, ru8 scs_selct);

Parameter	Description
bus_select	Select Bus0 or Bus1
scs_select	Select serial IF0(xnsfcs0) or serial IF1(xnsfcs1)

Note:

It is recommended to use the serial IF0(xnsfcs0) for the GENITOP's font ROM, and use the serial IF0(xnsfcs1) for the serial flash memory.

dma_24bitAddressBlockMode()

Description:

Read the image data from a 24bit address serial flash memory via the specified serial I/F, and the write the data into the specified memory block of the current canvas.

Function prototype:

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);

Parameter	Description
bus_selct	0~1 Select Bus0 or Bus1
scs_selct	0~3 Select serial IF0~3 (xnsfcs0~3)
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4 RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 Select SPI clock divider
x0	X-axis coordinate of the current canvas
y0	Y-axis coordinate of the current canvas
width	Width of the DMA block
height	Height of the DMA block
picture_width	Image width of the serial flash
addr	Image data start address of the serial flash

Example:

DMA function can be executed to read the entire image data or read the partial block data of the image, and then write the data into the specified memory block of the current canvas.

Example, the entire image data read and write:

//set current canvas
//clean current canvas page1 specify active window to color blue
//DMA reads image data from Serial Flash and writes to specified block of the current canvas

```
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_width,BINARY_INFO[p1].img_height,BINARY_INFO[p1].img_width,BINARY_INFO[p1].start_addr);
delay(1000);

ra8889lite.dma_24bitAddressBlockMode(0,1,RA8889_SPI_DIV2,0,0,BINARY_INFO[p2].img_width,BINARY_INFO[p2].img_height,BINARY_INFO[p2].img_width,BINARY_INFO[p2].start_addr);
delay(1000);

ra8889lite.dma_24bitAddressBlockMode(0,1,RA8889_SPI_DIV2,0,0,BINARY_INFO[p3].img_width,BINARY_INFO[p3].img_height,BINARY_INFO[p3].img_width,BINARY_INFO[p3].start_addr);
delay(1000);
```

Screenshot of the example:

dma_32bitAddressBlockMode()

Description:

Read the image data from a 32bit address serial flash memory via the specified serial I/F, and the write the data into the specified memory block of the current canvas.

Function prototype:

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

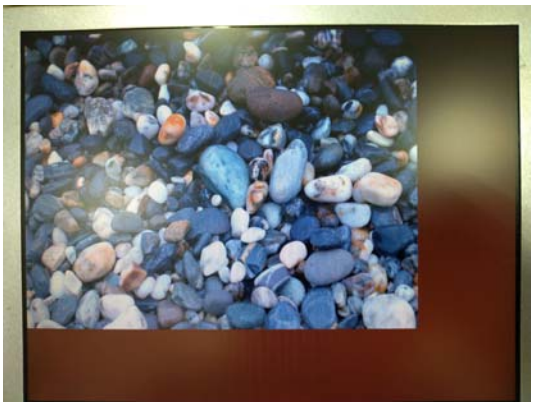
Parameter	Description
bus_selct	0~1 Select Bus0 or Bus1
scs_selct	0~3 Select serial IF0~3 (xnsfcs0~3)
clk_div	RA8889_SPI_DIV2 RA8889_SPI_DIV4

	RA8889_SPI_DIV6 RA8889_SPI_DIV8 RA8889_SPI_DIV10 Select SPI clock divider
x0	X-axis coordinate of the current canvas
y0	Y-axis coordinate of the current canvas
width	Width of the DMA block
height	Height of the DMA block
picture_width	Image width of the serial flash
addr	Image data start address of the serial flash

Example:

```
/*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)
{
//set current canvas
// clean current canvas page1 specify active window to color light cyan
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 read image data from Serial Flash and write to specified block of the current canvas
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_addr);
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_addr);
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);}
}
```

Screenshot of the example:



Chapter 9 IDEC

RA8889 provides the IDEC function, IDEC function can read image data (JPEG/AVI format) from the expanded serial flash of the RA8889, decode compressed data through Media Decoder Unit (MDU) and the decompress data will be written to specified block of the canvas quickly.

Function	Description
spimSetSerialFlashQuadMode();	Set serial flash to Quad Mode
spimSetSerialFlash4BytesMode()	Set serial flash to 4Bytes mode
idec_24bitAddressQuadMode6B_24bpp_JPEG()	IDEC read 24bit serial flash JPEG image data then decode and write into specified location
idec_32bitAddressQuadMode6B_24bpp_JPEG()	IDEC read 32bit serial flash JPEG image data then decode and write into specified location
idec_24bitAddressQuadMode6B_24bpp_AVI()	IDEC read 24bit serial flash AVI image data then decode and write into specified location
idec_32bitAddressQuadMode6B_24bpp_AVI()	IDEC read 32bit serial flash AVI image data then decode and write into specified location
aviWindowOn()	AVI display window on/off

Note:

It is recommended that the IDEC function is worked with the serial flash which connected to the Bus1 and selected by serial IF2 (xnsfcs2) or connected to the Bus1 and selected by serial IF3 (xnsfcs3).

spimSetSerialFlashQuadMode ()

Description:

If we want to use IDEC and MDU function for RA8889, the external serial flash must support Quad mode, and call this function to set the serial flash stayed in the Quad mode.

Function prototype:

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

Parameter	Description
bus_select	Select Bus0 or Bus1
scs_select	Select serial IF2(xnsfcs2) or serial IF3(xnsfcs3)
flash_select	0: MXIC flash 1: Winbond flash Others: through data1 and data2 push command
data1	Quad Mode command1
data2	Quad Mode command2

Note:

It is recommended that the IDEC function is worked with the serial flash which connected to the Bus1 and selected by serial IF2 (xnsfcs2) or connected to the Bus1 and selected by serial IF3 (xnsfcs3).

spimSetSerialFlash4BytesMode ()

Description:

When we use the serial flash memory with 32bit address, user must call this function first for setting the serial flash memory worked as 4Bytes mode.

Function prototype:

spimSetSerialFlash4BytesMode(ru8 bus_selct, ru8 scs_selct);

Parameter	Description
bus_select	Select Bus0 or Bus1
scs_select	Select serial IF0(xnsfcs0) or serial IF1(xnsfcs1)

Note:

It is recommended that the IDEC function is worked with the serial flash which connected to the Bus1 and selected by serial IF2 (xnsfcs2) or connected to the Bus1 and selected by serial IF3 (xnsfcs3).

idec_24bitAddressQuadMode6B_24bpp_JPEG()
idec_32bitAddressQuadMode6B_24bpp_JPEG()

Description:

IDEC read JPEG image data from 24bit /32bit serial flash then decode through MDU and write the decompress data into specified location.

Function prototype:

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

Parameter	Description
bus_selct	0~1 Select Bus0 or Bus1
scs_selct	0~3 Select serial IF0~3 (xnsfcs0~3)
x0	X-axis coordinate of the destination canvas
y0	Y-axis coordinate of the destination canvas
addr	Image data start address of the serial flash
number	The amount of data
des_image_width	Width of the destination canvas
des_start_addr	Start address of the destination canvas

idec_24bitAddressQuadMode6B_24bpp_AVI()
idec_32bitAddressQuadMode6B_24bpp_AVI()

Description:

IDEC read AVI image data from 24bit /32bit serial flash then decode through MDU and write the decompress data into specified location.

Function prototype:

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);

Parameter	Description

bus_selct	0~1 Select Bus0 or Bus1
scs_selct	0~3 Select serial IF0~3 (xnsfcs0~3)
x0	X-axis coordinate of the AVI window
y0	Y-axis coordinate of the AVI window
addr	Image data start address of the serial flash
number	The amount of data
width	Width of the AVI
height	Height of the AVI
shadow_buffer_addr	Specified shadow buffer start address of memory
pip_image_addr	PIP1 image start address of the memory
pip_image_width	PIP1 image width of the memory

aviWindowOn()

Description:

The function is used as an on/off switch to turn-on/turn-off the AVI display window. The AVI window is displayed through PIP1, it always be displayed on the top layer after turning on the AVI display window.

Function prototype:

aviWindowOn(boolean enable);

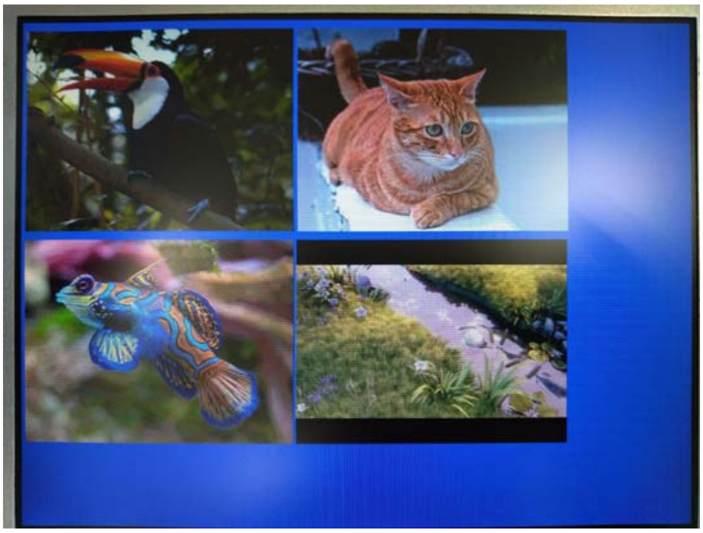
參數	說明
enable	= 1 : AVI display window on = 0 : AVI display window off

Example:

```
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
}
```

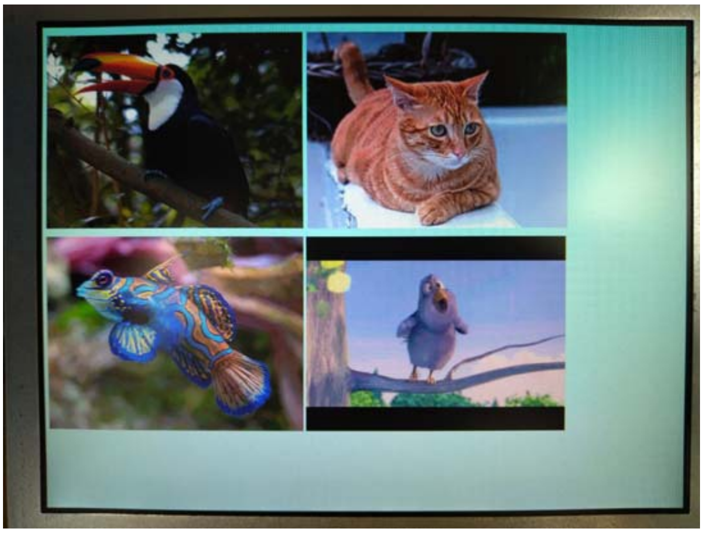
Screenshot of the example:



```
/*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].start_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_video320240].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
}
```

Screenshot of the example:



Chapter 10 PWM

Function	Description
pwm_Prescalar()	Set Prescalar
pwm_ClockMuxReg()	PWM frequency divider and the PWM pin function selection
pwm_Configuration()	Setting and start PWM function
pwm0_ClocksPerPeriod()	Setting amount of the each duty cycle clock for PWM0
pwm0_Duty()	PWM0 duty cycle
pwm1_ClocksPerPeriod()	Setting amount of the each duty cycle clock for PWM1
pwm1_Duty()	PWM1 duty cycle

The related circuitry or hardware connection in this chapter, please refer to "*RA8889 Arduino Wire Sketch.jpg"* or appendix Figure A-1.

pwm_Prescalar()

Description:

Set prescalar.

Function prototype:

void pwm_Prescalar(ru8 Prescalar);

Parameter	Description
Prescalar	RA8889_PRESCALAR

Note:

Base frequency of the PWM0 and PWM1 = Core_Freq / (Prescalar + 1)

pwm_ClockMuxReg()

Description:

It is used for decided the PWM frequency divider and the PWM pin function selection

Function prototype:

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

Parameter	Description
pwm1_clk_div	PWM1 base frequency divider setting

	RA8889_PWM_TIMER_DIV1 RA8889_PWM_TIMER_DIV2 RA8889_PWM_TIMER_DIV4 RA8889_PWM_TIMER_DIV8
pwm0_clk_div	PWM0 base frequency divider setting RA8889_PWM_TIMER_DIV1 RA8889_PWM_TIMER_DIV2 RA8889_PWM_TIMER_DIV4 RA8889_PWM_TIMER_DIV8
xpwm1_ctrl	PWM1 pin function selection RA8889_XPWM1_OUTPUT_ERROR_FLAG RA8889_XPWM1_OUTPUT_PWM_TIMER1 RA8889_XPWM1_OUTPUT_OSC_CLK
xpwm0_ctr	PWM0 pin function selection RA8889_XPWM0_GPIO_C7 RA8889_XPWM0_OUTPUT_PWM_TIMER0 RA8889_XPWM0_OUTPUT_CORE_CLK

pwm_Configuration()

Description:

Set and start PWM function

Function prototype:

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);

Parameter	Description
pwm1_inverter	PWM1 output inverter off or on RA8889_PWM_TIMER1_INVERTER_OFF RA8889_PWM_TIMER1_INVERTER_ON
pwm1_auto_reload	PWM1 output one shot or auto reload RA8889_PWM_TIMER1_ONE_SHOT RA8889_PWM_TIMER1_AUTO_RELOAD
pwm1_start	PWM1 stop or start RA8889_PWM_TIMER1_STOP RA8889_PWM_TIMER1_START

pwm0_dead_zone	PWM0 dead zone disable or enable RA8889_PWM_TIMER0_DEAD_ZONE_DISABLE RA8889_PWM_TIMER0_DEAD_ZONE_ENABLE
pwm0_inverter	PWM0 output inverter off or on RA8889_PWM_TIMER0_INVERTER_OFF RA8889_PWM_TIMER0_INVERTER_ON
pwm0_auto_reload	PWM0 output one shot or auto reload RA8889_PWM_TIMER0_ONE_SHOT RA8889_PWM_TIMER0_AUTO_RELOAD
pwm0_start	PWM0 stop or start RA8889_PWM_TIMER0_STOP RA8889_PWM_TIMER0_START

pwm0_ClocksPerPeriod()
pwm1_ClocksPerPeriod()

Description:

The function “pwm0_ClocksPerPeriod()” sets the number of pulses of each duty cycle for the channel the “PWM0”.

The function “pwm1_ClocksPerPeriod()” sets the number of pulses of each duty cycle for the channel the “PWM1”.

Function prototype:

```
void pwm0_ClocksPerPeriod(ru16 clocks_per_period);
void pwm1_ClocksPerPeriod(ru16 clocks_per_period);
```

Parameter	Description
clocks_per_period	Pulse amount of the each duty cycle (1~65535)

Note:

Another meaning for the setting is PWM resolution, for example, the setting is 1000, then the duty cycle range can be adjusted from 0 to 1000.

pwm0_Duty()
pwm1_Duty()

Description:

"**pwm0_Duty()**" is the duty cycle setting for PWM0.

"**pwm1_Duty()**" is the duty cycle setting for PWM1.

Function prototype:

```
void pwm0_Duty(ru16 duty);
void pwm1_Duty(ru16 duty);
```

Parameter	Description
duty	Value of the duty cycle

Note:

Duty cycle's duty range is decided by clocks_per_period setting value.

Example:

```
//pwm demo please measure by oscilloscope
ra8889lite.pwm_Prescalar(RA8889_PRESCALAR); //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);
```

Appendix A

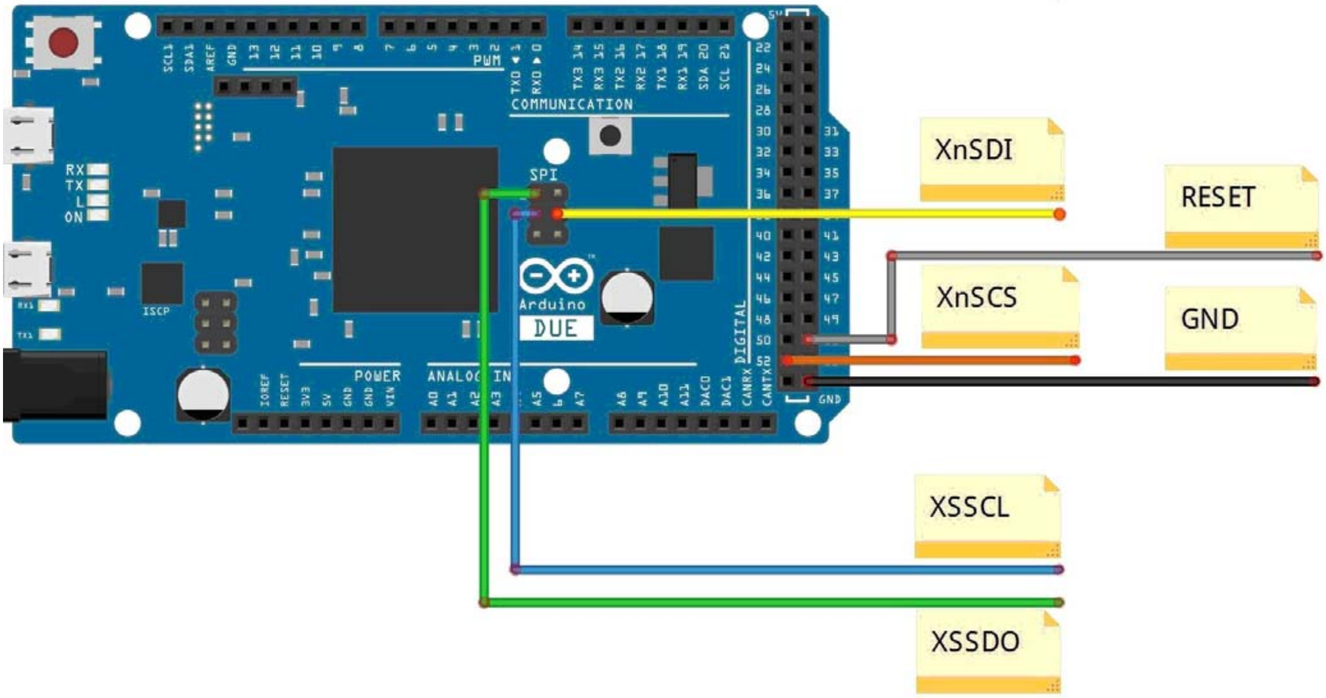


Figure A-1

End