

# OLED Driver API Manual For SSD 2119 Controller

© 2009 Gwentech LLC. All rights reserved.

---

4456 N. Abbe Road, Suite 391, Sheffield Village, OH 44054

Website: [www.graphicsdriversource.com](http://www.graphicsdriversource.com)

---

## Table of Contents

1.	INTRODUCTION .....	4
1.1.	SCOPE .....	4
1.2.	Terms and Definitions .....	4
2.	OLED IO Driver .....	4
2.1.	Hardware interface communication modes: .....	4
2.2.	Memory allocation for Display Buffer: .....	6
2.3.	Function:: init_io_port() .....	6
2.4.	Function:: void oled_Reset() .....	7
2.5.	Function:: unsigned char oled_ReadData() .....	7
2.6.	Function:: unsigned char oled_ReadStatus() .....	7
2.7.	Function:: void oled_WriteData() .....	8
2.8.	Function:: void oled_WriteCommand () .....	8
3.	OLED API Driver .....	8
3.1.	Initialization: .....	8
3.2.	Function:: oled_ssd2119_init() .....	8
4.	Configuration .....	8
4.1.	Function:: oled_GoTo() .....	9
4.2.	Function:: oled_ClearScreen() .....	9
4.3.	Function:: oled_InverseScreen () .....	10
4.4.	Function:: oled_NormalScreen () .....	10
5.	Graphics .....	10
5.1.	Display Bitmap .....	13
5.2.	Function:: oled_SetPixel(): .....	13
5.3.	Function:: oled_Rectangle() .....	14
5.4.	Function:: oled_Circle() .....	16
5.5.	Function:: oled_Line() .....	16
5.6.	Function:: oled_Bitmap () .....	17
6.	Character and String .....	18
6.1.	Function:: oled_WriteChar () .....	18

**OLED Driver API Manual For SSD  
2119 Controller**

Document Type: API Manual

Classification: Confidential

---

6.2.	Function:: oled_WriteString ().....	19
7.	Display Buffer.....	19
7.1.	Function:: oled_CreateBuffer ().....	20
7.2.	Function:: oled_RestoreBuffer ().....	21
7.3.	Function:: oled_SaveBuffer ().....	21
8.	Sample Programming Examples.....	21
8.1.	Example 1: Put String on Display.....	21
8.2.	Example 2: Creates Rectangle on Display.....	22
8.3.	Example 3: Creates Line on Display.....	23
8.4.	Example 4: Creates Circle on Display.....	23
8.5.	Example 5: Creates Bitmap on Display.....	24
8.6.	Example 6: Puts String in Buffer and then Display.....	24
8.7.	Example 7: Creates Rectangle in buffer and then Display.....	25
8.8.	Example 8: Creates Line in buffer and then Display.....	26
8.9.	Example 9: Creates Circle in Buffer and then Display.....	26
8.10.	Example 10: Creates Bitmap in Buffer and then Display.....	27
8.11.	Example 11: Display Inverses.....	28
8.12.	Example 12: Saves display image in buffer and then copy it on display.....	29

## 1. INTRODUCTION

### 1.1. SCOPE

The OLED IO driver and the driver API are software modules, which are distributed as a set of source files. They have been written using the C language in a minimum processor-independent manner. The code is highly portable and can run on any general purpose processor.

The software consists of two parts

- 1) The OLED driver IO functions are located in the oled\_io.c and
- 2) The API functions are located in the oled\_api.c.

### 1.2. Terms and Definitions

Sl. No.	Acronym	Description
1	OLED	Organic Light Emitting Diode
2	API	Application Programming Interface
3	IO	Input Output
4	SPI	Serial Peripheral Interfaces

## 2. OLED IO Driver

The hardware interface with host processor depends on circuit connection. The IO port definition and memory allocation are compiler related and required to be defined according to host processor and compiler definitions.

### 2.1. Hardware interface communication modes:

The display module is operated using following hardware interface communication modes.

- MODE\_6800
- MODE\_8080
- MODE\_SPI

MODE\_6800 and MODE\_8080 are parallel data interface mode and MODE\_SPI is serial data interface mode. The hardware IO port definition is required to be done in definition of oled\_io.h

The hardware interface communication mode is defined using MACRO definitions.

```
#define MODE_6800 0
```

```
#define MODE_8080 1
```

```
#define MODE_SPI 2
```

```
//defines Hardware Interface communication mode of display module
```

```
#define MPU_MODE MODE_SPI
```

## 2.2. Memory allocation for Display Buffer:

Display buffer memory is use for storing the image. The image created in the Display Buffer can be used for displaying the image on the module. The pool of memory for display buffer is allocated initially. The API use to create the display image object will allocate the memory block from this pool.

The macro definition in SSD2119.h will define the size of memory pool. The size depends on memory available and display image size requirement. For SSD2119 module the memory size required is 153600 Bytes (320 X 240 X 16 Bits ) for 65K Color or 172800 Bytes (320 X 240 X 18 Bits) for 262K color image. When more images are required to be stored the size can be calculated accordingly.

```
#define DISPLAY_BUFFER_SIZE 500
```

The memory allocation is done in ssd2119.c and location given by Display Buffer. The variable DisplayBufferFree indicates memory allocated from the pool.

```
unsigned char DisplayBuffer [DISPLAY_BUFFER_SIZE];  
int DisplayBufferFree = 0;
```

## 2.3. Function:: init\_io\_port()

Synopsis: void init\_io\_port()

Description: The initialization of the ports as Output or Input is done. This function host processor and compiler specific and need to be modified accordingly.

Arguments: None

Return Value: None

#### **2.4. Function:: void oled\_Reset()**

Synopsis: void oled\_Reset()

Description: The OLED Display will reset before initialization, when powered on. This will make display module command registers in default mode.

Arguments: None

Return Value: None

#### **2.5. Function:: unsigned char oled\_ReadData()**

Synopsis: unsigned char oled\_ReadData(void)

Description: The functions read data from the module. When communication interface mode is MODE\_SPI this function is not usable.

Arguments: None

Return Value: The data from the Display module.

#### **2.6. Function:: unsigned char oled\_ReadStatus()**

Synopsis: unsigned char oled\_ReadStatus(void)

Description: The function reads status data from the module. When communication interface mode is MODE\_SPI this function is not usable.

Arguments: None

Return Value: The status data from the Display module.

## 2.7. Function:: void oled\_WriteData()

Synopsis: void oled\_WriteData(unsigned char dataToWrite)

Description: The function writes data to the module.

Arguments: The data to be written

Return Value: None

## 2.8. Function:: void oled\_WriteCommand ()

Synopsis: void oled\_WriteCommand(unsigned char commandToWrite)

Description: The function writes command data to the module.

Arguments: The command data to be written

Return Value: None

## 3. OLED API Driver

### 3.1. Initialization:

The MACRO definitions for SSD2119 commands are defined in oled\_ssd2119.h The API functions uses these commands to communicate with Display module. The oled\_ssd2119\_init() API requires to used before starting new operation. This will initialize required settings of the display.

The display module size is 320X240. The corresponding row address size is 00-0x13F. The column address size is 0x00-0xEF.

The module has 65k color mode and 262K color mode. The mode is selected initially using global variable color\_mode.

### 3.2. Function:: oled\_ssd2119\_init()

Synopsis: void oled\_ssd2119\_init()

Description: Functions initializes the display module. After switching on the module, the initialization API does all the required settings for the module.

Arguments: None

Return Value: None

## 4. Configuration

The functions are used to configure the display according to the requirement.

#### 4.1. Function:: oled\_GoTo()

Synopsis: oled\_GoTo(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2)

Description: The display write location is set. The start location is top left and the end location is right bottom.

Arguments:

- unsigned char x1  
X Location of Start of Display, Left location
- unsigned char y1,  
Y Location of Start of Display, Top location
- unsigned char x2,  
X Location of End of Display, Right location
- unsigned char y2  
Y Location of End of Display, Bottom location

Return Value: None

#### 4.2. Function:: oled\_ClearScreen()

Synopsis: oled\_ClearScreen(void)

Description: The entire display module memory data clears to RGB scale level 0. This will have effect of clear screen

Arguments: None

Return Value: None

### 4.3. Function:: oled\_InverseScreen ()

Synopsis: oled\_InverseScreen(void)

Description: The RBG scale level of display data are swapped having effect of inverse display.

Arguments: None

Return Value: None

### 4.4. Function:: oled\_NormalScreen ()

Synopsis: oled\_NormalScreen(void)

Description: Reset the "Entire Display ON, Entire Display OFF or Inverse Display" effects and turn the data to ON at the corresponding RBG level.

Arguments: None

Return Value: None

## 5. Graphics

The graphic API functions can operated on the module directly or use the display buffer memory for storing the image depending on flag\_draw\_now argument. When flag\_draw\_now is YES, the functions operates on display module. When flag\_draw\_now is NO, the functions stores in display buffer memory.

The OLED\_USER\_DISPLAY structure defines the attributes of display buffer memory. API oled\_CreateBuffer () creates the display buffer (Refer API description)

The pixel point has following attributes which is defined in oled\_api.h

- X: X location of pixel point
- Y: Y location of pixel point
- color: Structure for RGB color

typedef struct point

```
{
    int x;
    int y;
    COLOR color;
}POINT;

typedef struct color_
{
    unsigned char A;
    unsigned char B;
    unsigned char C;
}COLOR;
```

For 65K Color the sub pixel data is as follows:

A Red: 5 Bits (00-31)

B Blue: 6 Bits (00-63)

C Green: 5 Bits (00-31)

For 262K Color the sub pixel data is as follows:

A Red: 6 Bits (0-63)

B Blue: 6 Bits (0-63)

C Green: 6 Bits (0-63)

The global variable `color_mode` decides color mode for the display.



## 5.1. Display Bitmap

Sixteen bits of data are required for each pixel for 65K Color and Eighteen bits of data are required for 612K Color for allowing possible RGB values. In total, is 12288 Bytes (320 X 240 X 16 Bits ) for 65K Color or 6144 Bytes (320 X 240 X 18 Bits) for 612K color image are required to hold the complete display content for the 320x240 pixels. Address mapping is as below.

For 65K COLOR

Address Pixel		Column						
		0	1	2	3	---	638	639
		0		1		---	319	
Row	0							
	1							
	-							
	239							

For 612K COLOR

Address Pixel		Column									
		0	1	2	3	4	5	---	957	958	959
		0			1			---	319		
Row	0										
	1										
	-										
	239										

## 5.2. Function:: oled\_SetPixel():

Synopsis: oled\_SetPixel(

POINT pt,  
unsigned char flag\_draw\_now,  
OLED\_USER\_DISPLAY disp)

Description: Function draws the pixel point on display module or stores in display buffer memory. The attributes of the pixel point pt decides location and RGB scale level.

Arguments:

- POINT pt:  
The co-ordinates and RGB scale level of the pixel point
- unsigned char flag\_draw\_now  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp  
The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

### 5.3. Function:: oled\_Rectangle()

Synopsis: void oled\_Rectangle(  
POINT start,  
unsigned char width,  
unsigned char height,  
unsigned char flag\_draw\_now,  
OLED\_USER\_DISPLAY disp)

Description: Function draws the rectangle on display module or stores in display buffer memory. The attributes of the pixel point start decides top left location of rectangle and RGB scale level of rectangle.

Arguments:

- POINT start:  
The top left co-ordinates of the rectangle and RGB scale level of the rectangle
- unsigned char width:  
Width of the rectangle
- unsigned char height:

Height of the rectangle

- unsigned char flag\_draw\_now:

YES: Function operates on display module

NO: Function operates on Display buffer memory

- OLED\_USER\_DISPLAY disp:

The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

#### 5.4. Function:: oled\_Circle()

Synopsis: Void oled\_Circle(  
          POINT center,  
          unsigned char radius,  
          unsigned char flag\_draw\_now,  
          OLED\_USER\_DISPLAY disp)

Description: Function draws the circle on display module or stores in display buffer memory. The attributes of the pixel point center decide center location of circle and RBG scale level of circle.

Arguments:

- POINT center:  
The co-ordinates of the center of circle and RBG scale level of the circle
- unsigned char radius:  
Radius of the circle
- unsigned char flag\_draw\_now:  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp:

The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

#### 5.5. Function:: oled\_Line()

Synopsis: oled\_Line( POINT start, POINT end, unsigned char flag\_draw\_now,  
                  OLED\_USER\_DISPLAY disp)

Description: Function draws the line on display module or stores in display buffer memory. The attributes of the pixel point start decide start location of line and RGB scale level of line.

Arguments:

- POINT start:  
The co-ordinates of the start of line and RGB scale level of the line
- POINT end:  
The co-ordinates of the end of line and RGB scale level of the line
- unsigned char flag\_draw\_now:  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp:  
The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

## 5.6. Function:: oled\_Bitmap ()

Synopsis: oled\_Bitmap(  
    char \* bitmap,  
    POINT start,  
    unsigned char width,  
    unsigned char height,  
    unsigned char flag\_draw\_now,  
    OLED\_USER\_DISPLAY disp)

Description: This function draws bitmap in display module or stores in display buffer module. The pixel data for bitmap is already stored in bitmap buffer. Refer memory map of the display for generating bitmap data.

Arguments:

- char \* bitmap:  
The bitmap storage location.
- POINT start:  
The co-ordinates of the start of Bitmap
- unsigned char width:  
Width of the bitmap
- unsigned char height:  
Height of the bitmap
- unsigned char flag\_draw\_now:  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp:

The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

## 6. Character and String

The bitmap for the characters is stored in font5x7.h. The character code will get the desired bitmap data. One character requires 7 bytes of data. 5 LSBs are only significant.

### 6.1. Function:: oled\_WriteChar ()

Synopsis: oled\_WriteChar(  
    POINT start,  
    char charCode,  
    unsigned char flag\_draw\_now,  
    OLED\_USER\_DISPLAY disp)

Description: The function displays the character using ASCII code of the character. The ASCII character code gets the bitmap of the character from the font table. The bitmap is displayed at the location determine by start point or stores in display memory buffer.

Arguments:

- POINT start:  
The co-ordinates of the start of line and RBG scale level of the character
- char charCode: The ascii value of the character to be displayed.
- unsigned char flag\_draw\_now:  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp:  
The display buffer memory block created. Applicable when flag\_draw\_now is NO

Return Value: None

## 6.2. Function:: oled\_WriteString ()

Synopsis: oled\_WriteString(  
POINT start,  
char \* string,  
unsigned char flag\_draw\_now,  
OLED\_USER\_DISPLAY disp)

Description: The functions display the character string at the location defined at start point or stores in display memory buffer.

Arguments:

- POINT start:  
The co-ordinates of the start of line and RBG scale level of the string
- char \* string:  
The array pointer of the string to be displayed
- unsigned char flag\_draw\_now:  
YES: Function operates on display module  
NO: Function operates on Display buffer memory
- OLED\_USER\_DISPLAY disp:  
The display buffer memory block created. Applicable when flag\_draw\_now is NO.

Return Value: None

## 7. Display Buffer

---

The display object can be created in the display memory buffer. The display object once created can modify and displayed on module. The display objects created from the memory pool available for display purpose. The memory pool required to be created using available memory allocation methods. The various display objects having different sizes can be created considering memory availability. Once display object created cannot be destroyed to avoid memory fragmentation.

### 7.1. Function:: oled\_CreateBuffer ()

Synopsis: OLED\_USER\_DISPLAY oled\_CreateBuffer(  
    unsigned char width,  
    unsigned char height)

Description: Display object is created from display memory pool.

Arguments:

- unsigned char width:  
    Width of the Display object
- unsigned char height:  
    Height of the display object

Return Value: Display object created

## 7.2. Function:: oled\_RestoreBuffer ()

Synopsis: oled\_RestoreBuffer(

unsigned char left,  
unsigned char top,  
OLED\_USER\_DISPLAY disp)

Description: Function displays on the module the display image from the display memory buffer which was created and modified using various API functions

Arguments:

- unsigned char left:  
The left location on display module where object image to be displays
- unsigned char top:  
The top location on display module where object image to be displays
- OLED\_USER\_DISPLAY disp:  
Display object already created

Return Value: None

## 7.3. Function:: oled\_SaveBuffer ()

Synopsis: oled\_SaveBuffer(POINT start, OLED\_USER\_DISPLAY disp)

Description: The data from the display module is stores in the display memory buffer, which was already created.

Arguments:

- POINT start:  
The start co-ordinates of the display image on the display module.
- OLED\_USER\_DISPLAY disp:  
Display object already created

Return Value: None

# 8. Sample Programming Examples

## 8.1. Example 1: Put String on Display

- P1: Start location of string having co-ordinate (0,0) and RGB scale level.
- str: String stored buffer location terminated by NULL.
- flag\_draw\_now YES: The String to be displayed directly.
- Window: buffer location, not useful for this.

P1.x=0;

P1.y=0;

P1.color.A=31;

P1.color.B=0;

P1.color.C=0;

oled\_WriteString(P1, str,YES>window);

P1: Start location of string having co-ordinate (0,0) and RGB scale level.

Str: String stored buffer location terminated by NULL.

flag\_draw\_now YES: The String to be displayed directly.

Window: buffer location, not useful for this.

P1.x=0;

P1.y=0;

P1.color.A=31;

P1.color.B=0;

P1.color.C=0;

oled\_WriteString(P1, str,YES>window);

## 8.2. Example 2: Creates Rectangle on Display

- P1: Start location of Rectangle i.e. top and left point co-ordinate (0,0) and RGB scale level.
- Width in pixel: 10
- Height in pixel: 10
- flag\_draw\_now YES: The Rectangle to be displayed directly.
- Window: buffer location, not useful for this.

P1.x=0;

P1.y=0;

P1.color.A=31;

P1.color.B=0;

```
P1.color.C=0;
```

```
oled_Rectangle(P1,10, 10,YES>window);
```

### 8.3. Example 3: Creates Line on Display

- P1: Start location of Line having co-ordinate (0,0) and RGB scale level.
- P2: Start location of Line having co-ordinate (40,50)
- flag\_draw\_now YES: The Line to be displayed directly.
- Window: buffer location, not useful for this.

```
P1.x=0;
```

```
P1.y=0;
```

```
P1.color.A=31;
```

```
P1.color.B=0;
```

```
P1.color.C=0;
```

```
P2.x=40;
```

```
P2.y=50;
```

```
oled_Line(P1,P2,YES>window);
```

### 8.4. Example 4: Creates Circle on Display

- P1: Center location of circle having co-ordinate (30,30) and RGB scale level.
- Radius: Radius of circle 10
- flag\_draw\_now YES: The circle to be displayed directly.
- Window: buffer location, not useful for this.

```
P1.x=30;
```

```
P1.y=30;
```

```
P1.color.A=31;
```

```
P1.color.B=0;
```

```
P1.color.C=0;
```

```
oled_Circle(P1,10,YES>window);
```

### 8.5. Example 5: Creates Bitmap on Display

- Bmp: Buffer location where bitmap data is stored according to address map of display.
- P1: Start location of Bitmap, left/top co-ordinate (0,0) and RGB scale level.
- Width of Bitmap 10
- Height of Bitmap 5
- flag\_draw\_now YES: The bitmap to be displayed directly.
- Window: buffer location, not useful for this.

```
P1.x=0;
```

```
P1.y=0;
```

```
P1.color.A=31;
```

```
P1.color.B=0;
```

```
P1.color.C=0;
```

```
oled_Bitmap(bmp,P1,10,5,YES>window);
```

### 8.6. Example 6: Puts String in Buffer and then Display

The example describes putting string in buffer and then displaying this data on display. The buffer is created from the memory pool for putting display data. The flag\_draw\_now NO will stores the string in buffer. Finally, the restore function displays the buffer data on display.

- P1: Start location of string having co-ordinate (0,0) and RGB scale level.
- Str: String stored buffer location terminated by NULL.
- flag\_draw\_now NO: The String to be stored in display buffer.
- Window: buffer location for storing display data.

```
P1.x=0;
```

```
P1.y=0;
```

```
P1.color.A=31;
```

```
P1.color.B=0;
```

```

P1.color.C=0;
str[2]=0;
window=oled_CreateBuffer(20, 20);
if (window.buffer==0)
{
    // error handling
};
oled_WriteString(P1, str,NO,window);
oled_RestoreBuffer(20, 20, window);

```

### 8.7. Example 7: Creates Rectangle in buffer and then Display

The example describes putting rectangle in buffer and then displaying this data on display. The buffer is created from the memory pool for putting display data. The flag\_draw\_now NO will stores the rectangle in buffer. Finally, the restore function displays the buffer data on display.

- P1: Start location of Rectangle i.e. top and left point co-ordinate (0,0) and RBG scale level.
- Width in pixel: 6
- Height in pixel: 6
- flag\_draw\_now NO: The Rectangle to be stored in display buffer.
- Window: buffer location for storing display data.

```

P1.x=0;
P1.y=0;
P1.color.A=31;
P1.color.B=0;
P1.color.C=0;
window=oled_CreateBuffer(20, 20);
if (window.buffer==0)
{
    // error handling
};
oled_Rectangle(P1,6, 6,NO,window);

```

```
oled_RestoreBuffer(20, 20, window);
```

### 8.8. Example 8: Creates Line in buffer and then Display

The example describes putting line in buffer and then displaying this data on display. The buffer is created from the memory pool for putting display data. The flag\_draw\_now NO will stores the line in buffer. Finally, the restore function displays the buffer data on display.

- P1: Start location of Line having co-ordinate (0,0) and RGB scale level.
- P2: Start location of Line having co-ordinate (6,6)
- flag\_draw\_now NO: The line to be stored in display buffer.
- Window: buffer location for storing display data.

```
P1.x=0;
P1.y=0;
P1.color.A=31;
P1.color.B=0;
P1.color.C=0;
P2.x=6;
P2.y=6;
window=oled_CreateBuffer(20, 20);
if (window.buffer==0)
{
    // error handling
};
oled_Line(P1,P2,NO,window);
oled_RestoreBuffer(20, 20, window);
```

### 8.9. Example 9: Creates Circle in Buffer and then Display

The example describes putting circle in buffer and then displaying this data on display. The buffer is created from the memory pool for putting display data. The flag\_draw\_now NO will stores the circle in buffer. Finally, the restore function displays the buffer data on display.

- P1: Center location of circle having co-ordinate (5,5) and RGB scale 16.
- Radius: Radius of circle 4
- flag\_draw\_now NO: The circle to be stored in display buffer.
- Window: buffer location for storing display data.

```

P1.x=5; // center of Circle
P1.y=5;
P1.color.A=31;
P1.color.B=0;
P1.color.C=0;
window=oled_CreateBuffer(20, 20);
if (window.buffer==0)
{
    // error handling
};
oled_Circle(P1,4,NO>window);
oled_RestoreBuffer(0, 12, window);

```

### 8.10. Example 10: Creates Bitmap in Buffer and then Display

The example describes putting bitmap in buffer and then displaying this data on display. The buffer is created from the memory pool for putting display data. The flag\_draw\_now NO will stores the bitmap in buffer. Finally, the restore function displays the buffer data on display.

- Bmp: Buffer location where bitmap data is stored according to address map of display.
- P1: Start location of Bitmap, left/top co-ordinate (0,0) and RGB scale 16.
- Width of Bitmap 10
- Height of Bitmap 5
- flag\_draw\_now NO: The bitmap to be stored in display buffer.
- Window: buffer location for storing display data.

```

P1.x=0;
P1.y=0;

```

```
P1.color.A=31;
P1.color.B=0;
P1.color.C=0;
window=oled_CreateBuffer(20, 20);
if (window.buffer==0)
{
    // error handling
};
oled_Bitmap(bmp,P1,10,5,NO,window);
oled_RestoreBuffer(20, 20, window);
```

### 8.11. Example 11: Display Inverses

The display will inverses the display data i.e. The RBG level data change. If RBG level data is 0xf it will become 0x0 and vice versa.

```
oled_InverseScreen();
```

### 8.12. Example 12: Saves display image in buffer and then copy it on display

The display image already on display, copies in the buffer and this will display at other location.

The buffer is created from the memory pool for putting display data. The display image is stores in the buffer and finally, the restore function displays the buffer data on display.

```
P1.x=0;
P1.y=0;
P1.color.A=31;
P1.color.B=0;
P1.color.C=0;
window=oled_CreateBuffer(20, 20);
oled_SaveBuffer(P1,window);
oled_RestoreBuffer(20, 20, window);
```