TFT LCD with SPI Interface

General Information

Dear customers,
Thank you very much for choosing our product.
In following, we will introduce you to what to observe while starting up and using this product.
Should you encounter any unexpected problem during use, please do not hesitate to contact us.

TABLE OF CONTENTS

I. INTRODUCTION 1
1.1. About Driver 1
1.2. Funtion Description 1
II. KERNEL CONFIGULATION 1
Step 1: Install the Device Tree Overlay 1
III. INTERFACE DESCRIPTION 2
3.1. Interface Definition: 2
3.2. IOCTL Command Description: 3
❖ TFT_IOCTL_SET_GPIO: 3
❖ TFT_SET_WINDOW: 4
❖ TFT_SET_ORIENTATION_PORTRAIT: 5
❖ TFT_SET_ORIENTATION_LANDSCAPE: 6
❖ TFT_Clear: 6
❖ TFT_DRAW_PIXEL: 7
❖ TFT_WRITE_STRING: 8

I. INTRODUCTION

This code is a Linux kernel module that provides a driver for controlling a TFT display using the SPI interface. The driver is designed to interface with a TFT display module, managing its initialization, configuration, and rendering operations. The main functionalities include controlling the display orientation, drawing pixels, clearing the screen, and writing strings.

1.1. About Driver

The driver includes the following 3 files:

- Kernel Object File: tft_driver.ko
- Kernel Header File: tft_driver.h
- Device Tree File: testoverlay_2.dtb

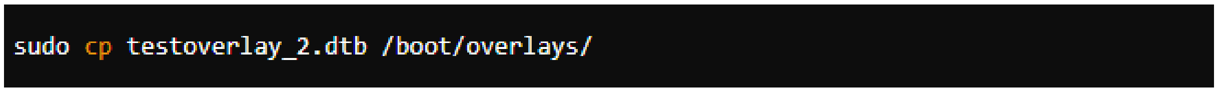
1.2. Funtion Description

- Support SPI0 interfaces, Chip Select 0(CE0), Max Speed 30MHz.
- Support selection GPIO for Back Light and A0 Pin of ST7735.
- Provide Ioctl Interface with Character Driver.
- Provide API for interface with Driver to config TFT LCD Parameter.

II. KERNEL CONFIGULATION

Step 1: Install the Device Tree Overlay

- Copy the testoverlay_2.dtb file to the /boot/overlays/ directory on your Raspberry Pi:



```
sudo cp testoverlay_2.dtb /boot/overlays/
```

- Open the /boot/config.txt file for editing:

```
sudo nano /boot/config.txt
```

- Add the following line to the end of the file to activate the overlay:

```
dtoverlay=testoverlay_2
```

- Save the file and reboot your Raspberry Pi.

Step 2: Load the Kernel Module

- After the system reboots, copy the tft_driver.ko file to an appropriate directory, such as /lib/modules/$(uname -r)/kernel/drivers/:

```
sudo cp tft_driver.ko /lib/modules/$(uname -r)/kernel/drivers/
```

- Load kernel module:

```
sudo insmod /lib/modules/$(uname -r)/kernel/drivers/tft_driver.ko
```

III. INTERFACE DESCRIPTION

3.1.Interface Definition:

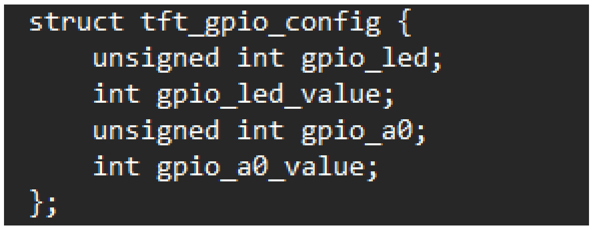
❖ The definition is stored in file tft_driver.h, following some define:

- **Constants for Color**: Defined constants represent various colors in 16-bit RGB format.
- **Font Definition**: The header defines a font structure (FontDef) which includes the width, height, and data for a specific font (Font7x10). This font is represented as a 2D array of 16-bit values.
- **IOCTL Commands**:
    - ➢ Commands for resetting the display, setting display intensity, configuring GPIO pins, setting display orientation, clearing the display, writing a byte to the display, drawing a pixel, writing a string, and writing a character.

- ➢ Each command is associated with a unique IOCTL number and may include additional parameters.

3.2.IOCTL Command Description:

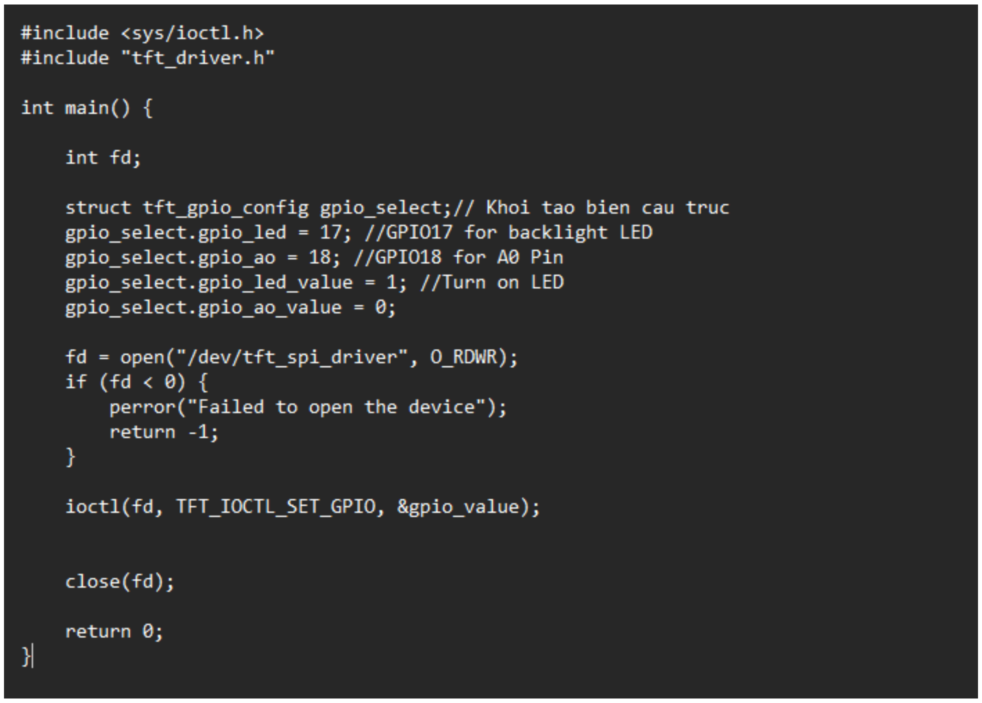
- ❖ TFT_IOCTL_SET_GPIO:
  - **Description**: Select GPIO Pin for Back Light LED and A0 PIN of ST7735.
  - **Commad Parameter**: struct tft_gpio_config.



```
struct tft_gpio_config {
    unsigned int gpio_led;
    int gpio_led_value;
    unsigned int gpio_a0;
    int gpio_a0_value;
};
```

- ➢ **gpio_led**: December Number of GPIO PIN to config Back Light LED pin of ST7735.
- ➢ **gpio_led_value**: logic output value (0 or 1).
- ➢ **gpio_ao**: December Number of GPIO PIN to config A0 pin of ST7735.
- ➢ **gpio_ao_value**: logic output value (0 or 1).

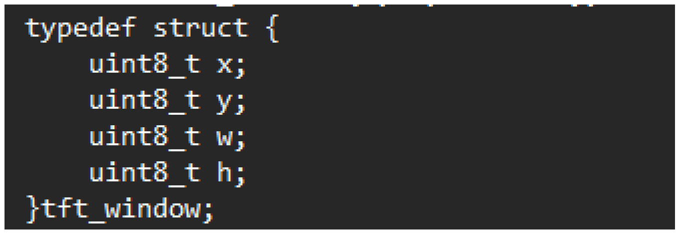
Example in User Code:



```
#include <sys/ioctl.h>
#include "tft_driver.h"

int main() {

    int fd;

    struct tft_gpio_config gpio_select;// Khoi tao bien cau truc
    gpio_select.gpio_led = 17; //GPIO17 for backlight LED
    gpio_select.gpio_ao = 18; //GPIO18 for A0 Pin
    gpio_select.gpio_led_value = 1; //Turn on LED
    gpio_select.gpio_ao_value = 0;

    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

    ioctl(fd, TFT_IOCTL_SET_GPIO, &gpio_value);


    close(fd);

    return 0;
}
```

❖ TFT_SET_WINDOW:

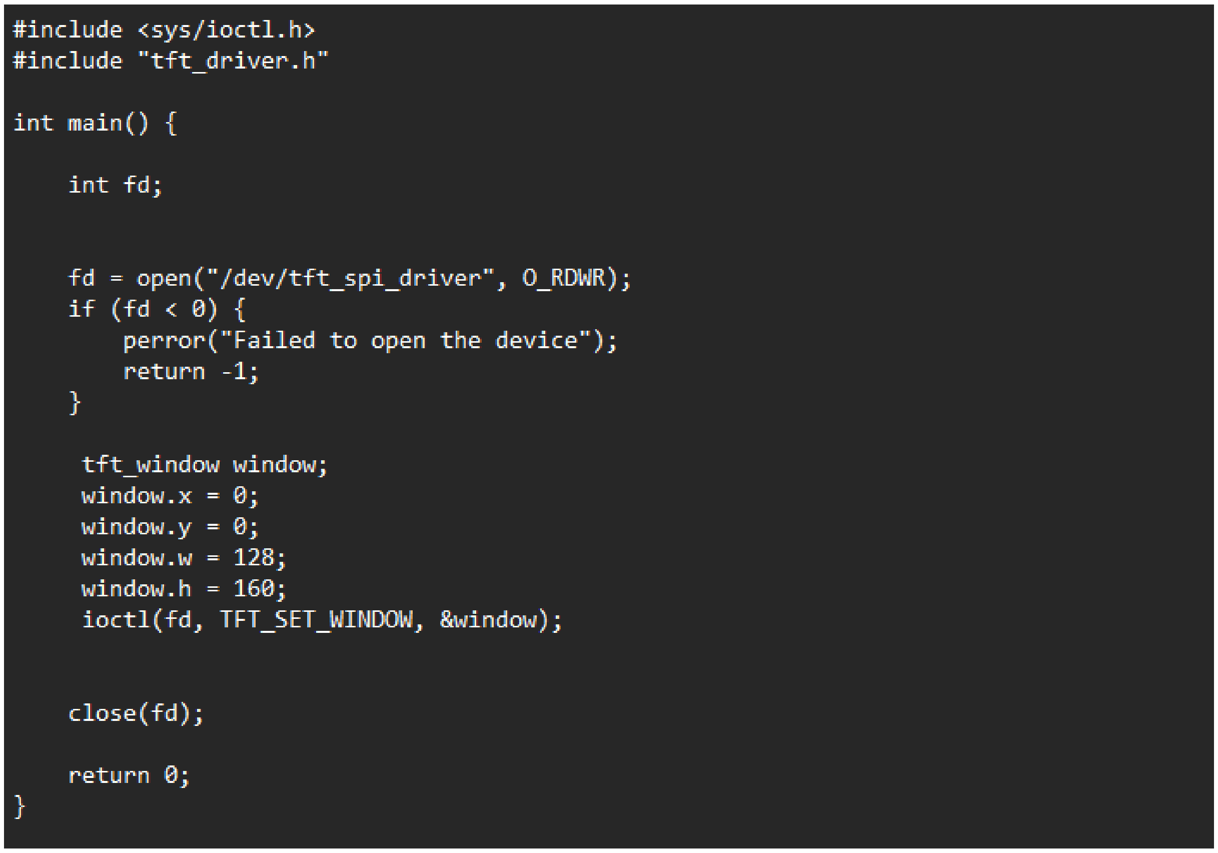
- **Description**: set window display for TFT CLD.
- **Command Parameter**: s typedef struct following window parameter.



```
typedef struct {
    uint8_t x;
    uint8_t y;
    uint8_t w;
    uint8_t h;
}tft_window;
```

- **uint8_t x**: Represents the x-coordinate (horizontal position) of the window.
- **uint8_t y**: Represents the y-coordinate (vertical position) of the window.
- **uint8_t w**: Represents the width of the window.
- **uint8_t h**: Represents the height of the window.

<u>Example in user code</u>

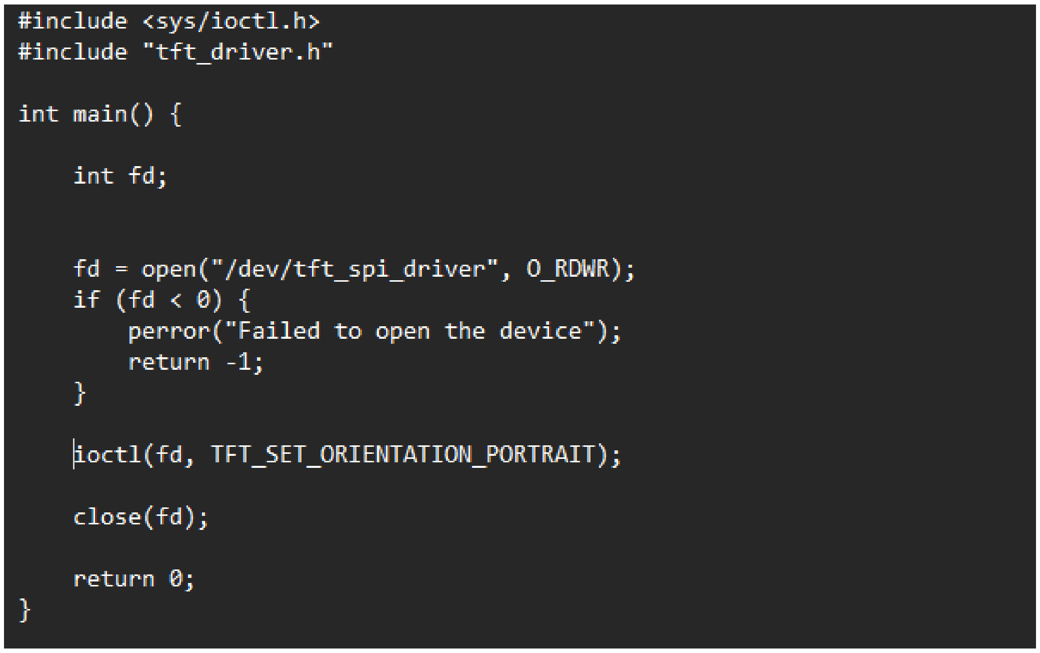


```c
#include <sys/ioctl.h>
#include "tft_driver.h"

int main() {

    int fd;


    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

     tft_window window;
     window.x = 0;
     window.y = 0;
     window.w = 128;
     window.h = 160;
     ioctl(fd, TFT_SET_WINDOW, &window);


    close(fd);

    return 0;
}
```

❖ TFT_SET_ORIENTATION_PORTRAIT:

- **Description**: Select display orientation (Portrait).
- **Commad Parameter**: None

<u>Example in user code:</u>

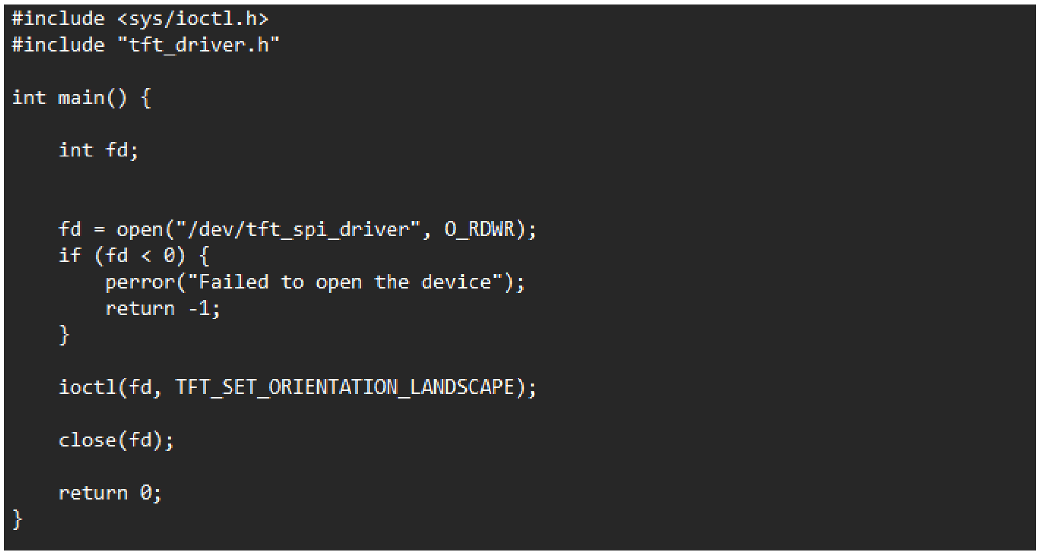


```c
#include <sys/ioctl.h>
#include "tft_driver.h"

int main() {

    int fd;


    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

    ioctl(fd, TFT_SET_ORIENTATION_PORTRAIT);

    close(fd);

    return 0;
}
```

❖ TFT_SET_ORIENTATION_LANDSCAPE:

- **Description**: Select display orientation (Lascape).
- **Commad Parameter**: None.

Example in user code:

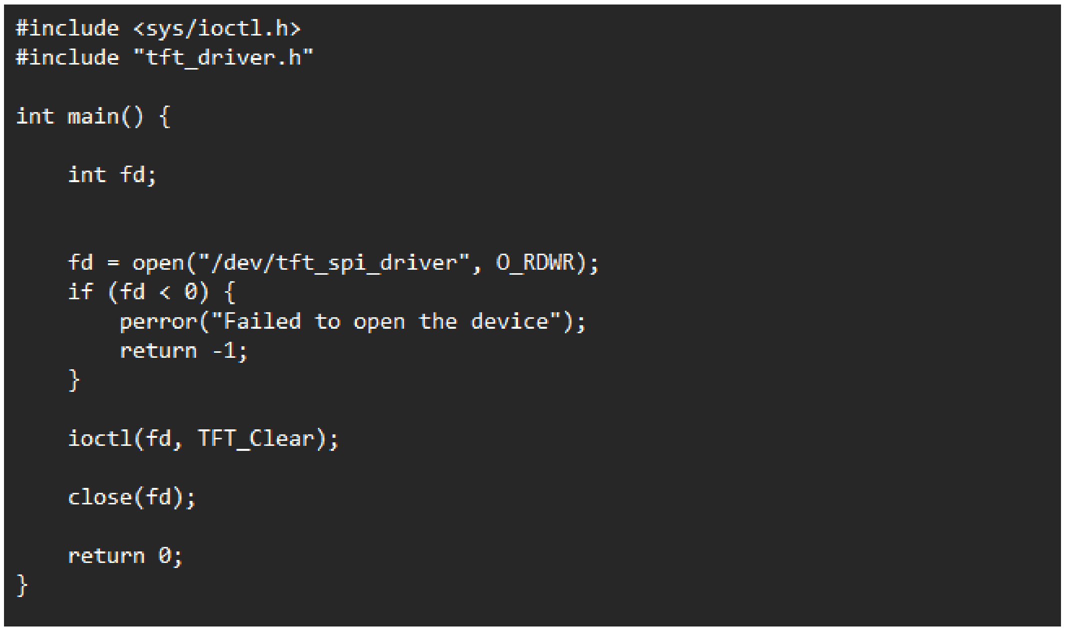


```
#include <sys/ioctl.h>
#include "tft_driver.h"

int main() {

    int fd;


    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

    ioctl(fd, TFT_SET_ORIENTATION_LANDSCAPE);

    close(fd);

    return 0;
}
```

❖ TFT_Clear:

- **Description**: Clear all display to black backgroud.
- **Commad Parameter**: None

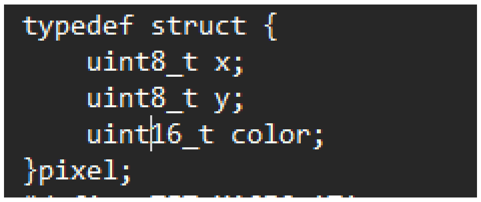
Example in user code:



```
#include <sys/ioctl.h>
#include "tft_driver.h"

int main() {

    int fd;


    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

    ioctl(fd, TFT_Clear);

    close(fd);

    return 0;
}
```

❖ TFT_DRAW_PIXEL:

- **Description**: Draw a pixel to display TFT LCD (Lascape).
- **Commad Parameter**: typedef struct defines a structure named pixel, which represents a single pixel on a display.



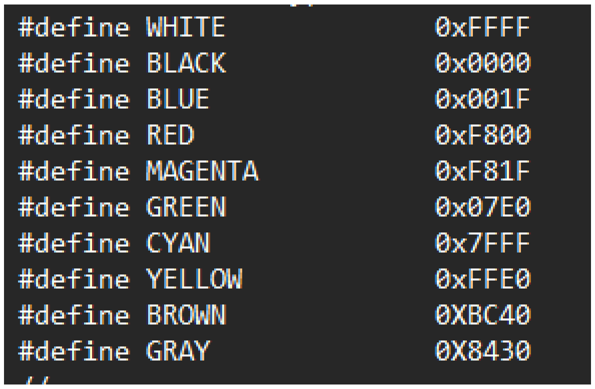
```
typedef struct {
    uint8_t x;
    uint8_t y;
    uint16_t color;
}pixel;
```

➢ **uint8_t x**: Represents the x-coordinate (horizontal position) of the pixel.

➢ **uint8_t y**: Represents the y-coordinate (vertical position) of the pixel.

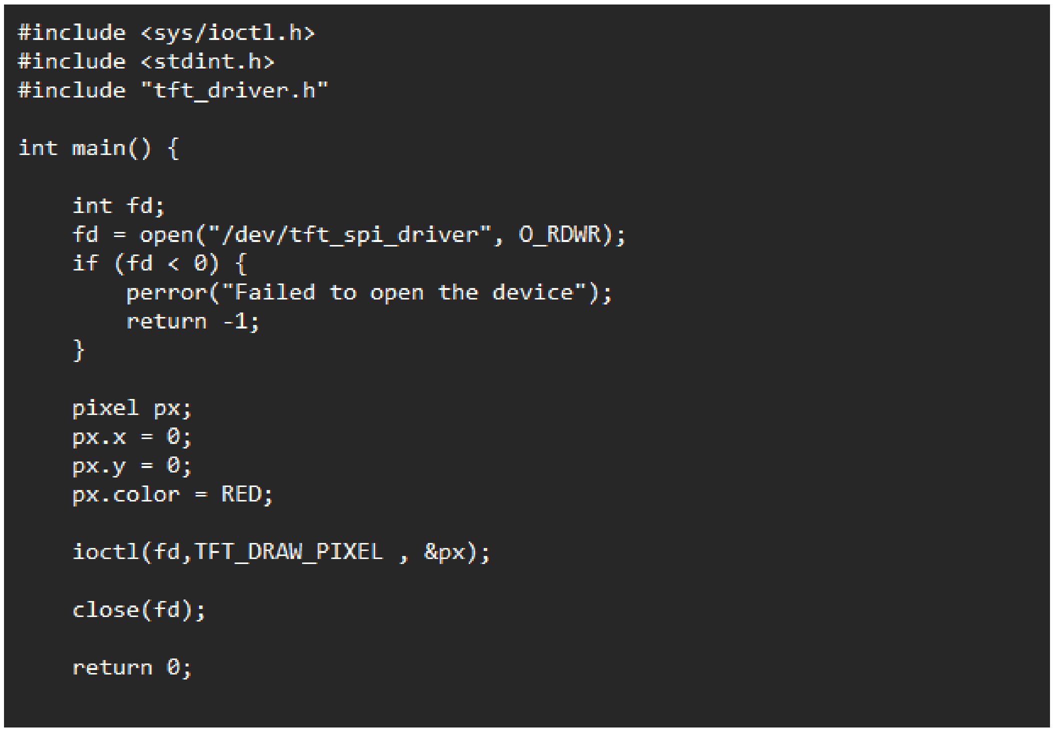
➢ **uint16_t color**: Represents the color of the pixel. The color is typically encoded as a 16-bit value, allowing a wide range of colors to be represented.

- *Color value: 16 bit value mode RGB565*



```
#define WHITE           0xFFFF
#define BLACK           0x0000
#define BLUE            0x001F
#define RED             0xF800
#define MAGENTA         0xF81F
#define GREEN           0x07E0
#define CYAN            0x7FFF
#define YELLOW          0xFFE0
#define BROWN           0XBC40
#define GRAY            0X8430
```

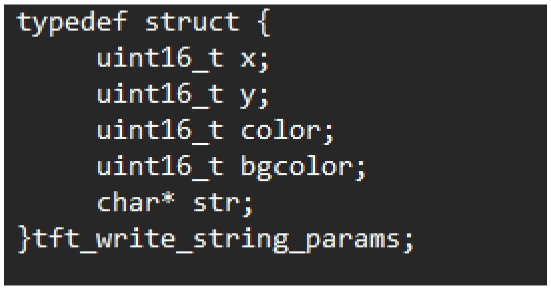
Example in user code:



```
#include <sys/ioctl.h>
#include <stdint.h>
#include "tft_driver.h"

int main() {

    int fd;
    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }

    pixel px;
    px.x = 0;
    px.y = 0;
    px.color = RED;

    ioctl(fd,TFT_DRAW_PIXEL , &px);

    close(fd);

    return 0;
```

❖ TFT_WRITE_STRING:

- **Description**: Display a string on TFT LCD.
- **Commad Parameter**: typedef struct defines a structure named tft_write_string_params, used to specify parameters for writing a string of characters to a display.



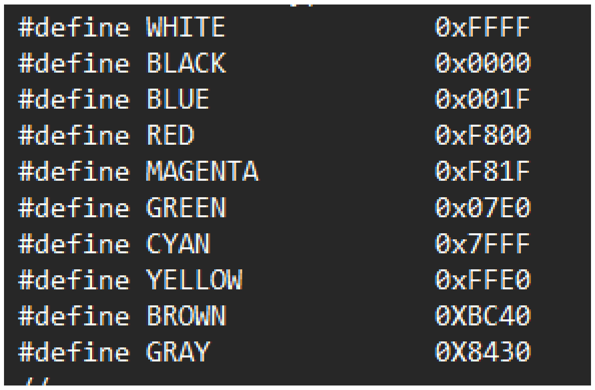
```
typedef struct {
      uint16_t x;
      uint16_t y;
      uint16_t color;
      uint16_t bgcolor;
      char* str;
}tft_write_string_params;
```

➢ **uint16_t x**: Represents the x-coordinate (horizontal position) where the string will be written.

➢ **uint16_t y**: Represents the y-coordinate (vertical position) where the string will be written.

➢ **uint16_t color**: Represents the color of the characters in the string. The color is typically encoded as a 16-bit value, allowing a wide range of colors to be represented.

- *Color value: 16 bit value mode RGB565*

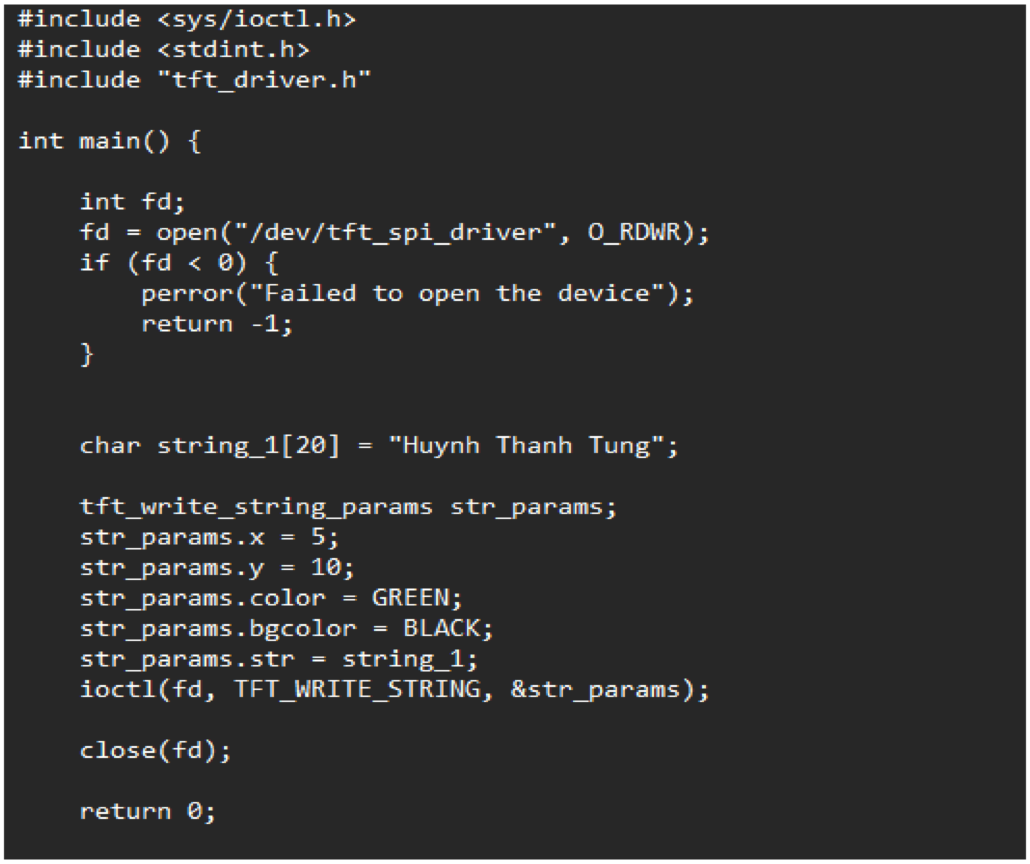


```
#define WHITE              0xFFFF
#define BLACK              0x0000
#define BLUE               0x001F
#define RED                0xF800
#define MAGENTA            0xF81F
#define GREEN              0x07E0
#define CYAN               0x7FFF
#define YELLOW             0xFFE0
#define BROWN              0XBC40
#define GRAY               0X8430
```

➢ **uint16_t bgcolor**: Represents the background color behind the characters. The color is typically encoded as a 16-bit value, allowing a wide range of colors to be represented.

➢ **char* str**: A pointer of string to be displayed.

Example in user code:



```
#include <sys/ioctl.h>
#include <stdint.h>
#include "tft_driver.h"

int main() {

    int fd;
    fd = open("/dev/tft_spi_driver", O_RDWR);
    if (fd < 0) {
        perror("Failed to open the device");
        return -1;
    }


    char string_1[20] = "Huynh Thanh Tung";

    tft_write_string_params str_params;
    str_params.x = 5;
    str_params.y = 10;
    str_params.color = GREEN;
    str_params.bgcolor = BLACK;
    str_params.str = string_1;
    ioctl(fd, TFT_WRITE_STRING, &str_params);

    close(fd);

    return 0;
```