

1.54-дюймовый модуль электронной бумаги

Из Waveshare Wiki

[инструкция](#) | [Настройка аппаратного / программного обеспечения](#) | [Описание кодов](#) | [Ресурсы](#) | [Часто задаваемые вопросы](#)

О кодах

Мы приводим примеры для четырех популярных аппаратных платформ: Arduino UNO, Jetson UNO, Raspberry Pi и STM32. (Это общий шаблон для всех электронных документов, некоторые описания / функции могут не использоваться имеющейся у вас электронной книгой).

Каждый проект разделен на аппаратный интерфейс, драйвер EPD и функцию приложения;

Языки программирования: C \ C ++ \ python:

- Arduino UNO: C ++
- Джетсон Нано: C и питон
- Raspberry Pi : C и питон
- STM32: C

Примечание.

Драйвер EPD кодов C Jetson Nano, Raspberry Pi и STM32 совместим. За исключением аппаратного интерфейса, коды одинаковы;

C (Используется для Jetson Nano 、 Raspberry Pi 、 STM32)

Аппаратный интерфейс

1.54 дюймовый E-Paper



Панель дисплея E-Ink 200x200, 1,54 дюйма

1.54-дюймовый модуль электронной бумаги



Из-за нескольких аппаратных платформ, мы упаковываем дно, для деталей о том, как это реализуется, вы переходите в соответствующий каталог для определенных кодов

В файле DEV_Config.c (.h):
для Raspberry Pi файлы расположены по адресу: RaspberryPi & JetsonNano \ c \ lib \ Config

Здесь мы используем две библиотеки: bcm2835 и wiringPi
Библиотека WiringPi используется по умолчанию, если вы хотите использовать библиотеки bcm2835, вам просто нужно изменить файл RaspberryPi & JetsonNano \ c \ Makefile, изменить строки 13 и 14, как показано ниже:

```
13 USELIB = USE_BCM2835_LIB
14 # USELIB = USE_WIRINGPI_LIB
15 DEBUG = -D $(USELIB)
16 ifeq ($(USELIB), USE_BCM2835_LIB)
17     LIB = -lbcm2835 -lm
18 else ifeq ($(USELIB), USE_WIRINGPI_LIB)
19     LIB = -lwiringPi -lm
20 endif
```

Для Jetson Nano файлы находятся в RaspberryPi & JetsonNano \ c \ lib \ Config.
Для STM32 файлы находятся в STM32 \ STM32-F103ZET6 \ User \ Config.

- Тип данных:

#define UBYTE uint8_t
#define UWORD uint16_t
#define UDOUBLE uint32_t
- Модуль Init и Exit handle:

void DEV_Module_Init (void) ;
void DEV_Module_Exit (void) ;

Запись:
1.Функции используются для настройки GPIP до и после вождения электронной бумаги.

Модуль дисплея E-Ink 200x200, 1,54 дюйма,
интерфейс SPI

Основной атрибут

Категория: [OLEDs / LCDs](#) , [LCD](#)

Марка: [Waveshare](#)

Веб-сайт

Английский: веб-сайт [Waveshare](#)

Китайский: [官方中文站点](#)

Бортовые интерфейсы

[SPI](#)

сопутствующие товары

- [4.3-дюймовый модуль UART для электронной бумаги](#)
- [12,48-дюймовый модуль электронной бумаги](#)
- [12,48-дюймовый модуль электронной бумаги \(B\)](#)
- [10,3-дюймовый E-Paper HAT \(D\)](#)
- [9,7-дюймовый E-Paper HAT](#)
- [7,8-дюймовый E-Paper HAT](#)
- [7,5-дюймовый E-Paper HAT](#)
- [7,5 дюймовый e-Paper HAT \(B\)](#)
- [7,5-дюймовый E-Paper HAT \(C\)](#)
- [7,5-дюймовый NFC-Powered электронная бумага](#)
- [6-дюймовый E-Paper HAT](#)
- [6-дюймовый HD электронная бумага HAT](#)
- [5.83inch E-Paper HAT](#)
- [5.83inch E-Paper HAT \(B\)](#)
- [5.83inch E-Paper HAT \(C\)](#)
- [4.2-дюймовый модуль электронной бумаги](#)
- [4.2-дюймовый модуль электронной бумаги \(B\)](#)

2. Если на вашей плате напечатано Rev2.1, модуль переходит в режим low-ultra после DEV_Module_Exit (). (как мы тестируем, ток составляет около 0 в этом режиме);

- GPIO чтение / запись:

```
недействительный DEV_Digital_Write ( UWORD Штифт , UBYTE Значение ) ;
UBYTE DEV_Digital_Read ( UWORD Pin ) ;
```

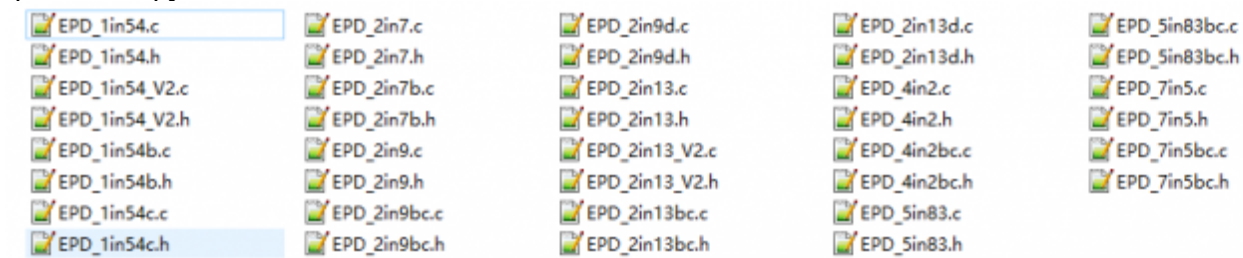
- SPI Запись данных

```
void DEV_SPI_WriteByte (значение UBYTE);
```

EPD драйвер

Для Raspberry Pi и Jetson Nano драйвер epd сохраняется в: RaspberryPi & JetsonNano \ c \ lib \ e-Paper

Для STM32 драйвер epd сохраняется в: STM32 \ STM32-F103ZET6 \ User \ e-Paper Открыть файл .h, функции объявлено здесь



- Инициализация: его следует использовать для инициализации электронной бумаги или пробуждения электронной бумаги из спящего режима.

```
//1.54inch e-Paper , 1.54inch e-Paper V2, 2.13inch e-Paper , 2.13inch e-Paper V2, 2.13inch e-Paper (D) , 2.9inch e-Paper , 2.9inch e-Paper (D)
void EPD_xxx_Init ( режим UBYTE ) ; // Mode = 0 Инициализировать полное обновление; Mode = 1 Инициализировать частичное обновление
// Другие типы
void EPD_xxx_Init ( void ) ;
```

- [4.2-дюймовый модуль электронной бумаги \(C\)](#)
- [4.2inch NFC-Powered электронная бумага](#)
- [2.9-дюймовый модуль электронной бумаги](#)
- [2.9-дюймовый модуль электронной бумаги \(B\)](#)
- [2.9-дюймовый модуль электронной бумаги \(C\)](#)
- [2.9-дюймовый E-Paper HAT \(D\)](#)
- [2.9-дюймовый NFC-Powered электронная бумага](#)
- [2.7-дюймовый E-Paper HAT](#)
- [2.7 дюймовый E-Paper HAT \(B\)](#)
- [2.13-дюймовый E-Paper HAT](#)
- [2.13-дюймовый E-Paper HAT \(B\)](#)
- [2.13-дюймовый E-Paper HAT \(C\)](#)
- [2.13-дюймовый E-Paper HAT \(D\)](#)
- [2.13-дюймовый NFC-Powered электронная бумага](#)
- [1.54inch NFC-Powered электронная бумага \(BB\)](#)
- [1.54-дюймовая NFC-Powered электронная бумага \(BW\)](#)
- 1.54-дюймовый модуль электронной бумаги**
- [1.54-дюймовый модуль электронной бумаги \(B\)](#)
- [1.54-дюймовый модуль электронной бумаги \(C\)](#)
- [1.02-дюймовый модуль электронной бумаги](#)
- [E-Paper Driver HAT](#)
- [EINK-DISP-103](#)
- [E-Paper Shield](#)

- [Электронная бумага ESP8266 Драйвер платы](#)
- [E-Paper ESP32 Драйвер платы](#)
- [Электронная бумага NB-IoT GPRS HAT](#)

- Очистить дисплей: эта функция используется для очистки электронной бумаги до белого

```
void EPD_xxx_Clear ( void ) ;
```

xxx это тип электронной бумаги. Например, если у вас есть электронная бумага 4,2 дюйма, она должна быть EPD-4IN2_Clear ()

- Передача кадра изображения и отображения

```
// Черно-белая электронная бумага
void EPD_xxx_Display ( UBYTE * Image ) ;
// Три цвета электронной бумаги
void EPD_xxx_Display ( const UBYTE * blackimage , const UBYTE * ryimage ) ;
```

Есть некоторые исключения:

```
// Для частичного обновления электронной бумаги 2.13 дюйма (D) , 2.9 дюймовой электронной бумаги (D) следует использовать
void EPD_2IN13D_DisplayPart ( UBYTE * Image ) ;
void EPD_2IN9D_DisplayPart ( UBYTE * Image ) ;
```

```
// Поскольку были обновлены контроллеры e-Paper V2 версии 1.54 и e-Paper V2 версии 2.13inch, вам необходимо использовать EPD_xxx_DisplayPartBaseImage для отображения статического изображения, а десять использовать EPD_xxx_displayPart () для отображения в режиме dumatic при частичном обновлении.
```

```
void EPD_1IN54_V2_DisplayPartBaseImage ( UBYTE * Image ) ;
void EPD_1IN54_V2_DisplayPart ( UBYTE * Image ) ;
void EPD_2IN13_V2_DisplayPart ( UBYTE * Image ) ;
void EPD_2IN13_V2_DisplayPartBaseImage ( UBYTE * Image ) ;
```

```
// Поскольку у STM32103ZET5 недостаточно ОЗУ для изображения, следовательно, 7,5 В, 7,5 С, 5,83 В, 5,83 С может отображать только половину экрана: ' ' '
void EPD_7IN5BC_DisplayHalfScreen ( const UBYTE * blackimage , const UBYTE * ryimage ) ;
void EPD_5IN83BC_DisplayHalfScreen ( const UBYTE * blackimage , const UBYTE * ryimage ) ;
```

xxx это тип электронной бумаги

- Войдите в спящий режим













```
void EPD_xxx_Sleep (void);
```

Примечание. Вам следует выполнить аппаратный сброс или использовать функцию инициализации, чтобы вывести электронную бумагу из спящего режима.

Xxx - это тип электронной бумаги.

Функция приложения

Основные функции рисования представлены здесь. Тогда вы можете найти:
Raspbian Pi & Jetson Nano: RaspberryPi & JetsonNano \ c \ lib \ GUI \ GUI_Paint.c (.h)
STM32: STM32 \ STM32-F103ZET6 \ User \ GUI \ GUI_Paint.c (.h) Шрифты сохранены в каталоге: Raspberry Pi & Jetson Nano: RaspberryPi & JetsonNano \ c \ lib \ Fonts STM32: STM32 \ STM32-F103ZET6 \ User \ Fonts

 GUI_BMPfile.c	2019/6/21 11:14	C 文件	6 KB
 GUI_BMPfile.h	2018/11/12 11:32	H 文件	4 KB
 GUI_Paint.c	2019/6/11 20:58	C 文件	30 KB
 GUI_Paint.h	2019/4/18 17:12	H 文件	7 KB
<div>2019/2021 10:10 10:10</div>			
 font8.c	2018/7/4 17:24	C 文件	18 KB
 font12.c	2018/7/4 17:24	C 文件	27 KB
 font12CN.c	2018/3/6 15:52	C 文件	6 KB
 font16.c	2018/7/4 17:24	C 文件	49 KB
 font20.c	2018/7/4 17:24	C 文件	65 KB
 font24.c	2018/7/4 17:24	C 文件	97 KB
 font24CN.c	2018/3/6 16:02	C 文件	28 KB
 fonts.h	2018/10/29 14:04	H 文件	4 KB

- Создать новый буфер изображения: эта функция используется для создания нового изображения с шириной, высотой, градусом поворота и его цветом.

недействительный Paint_NewImage (UBYTE * изображение , UWORD Ширина , UWORD Высота , UWORD Поворот , UWORD Цвет)

Paratemeters :

Изображение : Буфер изображения , это указатель буфера адреса ;

Ширина : ширина изображения ;

Высота : высота изображения ;

Повернуть: Повернуть степень ;

Цвет: начальный цвет изображения ;

- Выбрать буфер изображения: эта функция используется для выбора буфера изображения. Вы можете создать несколько буферов изображений с последней функцией, а затем выбрать буфер для каждого изображения.

void Paint_SelectImage (UBYTE * image)

Параметры:

image : имя буфера изображения , это указатель на адрес буфера ;

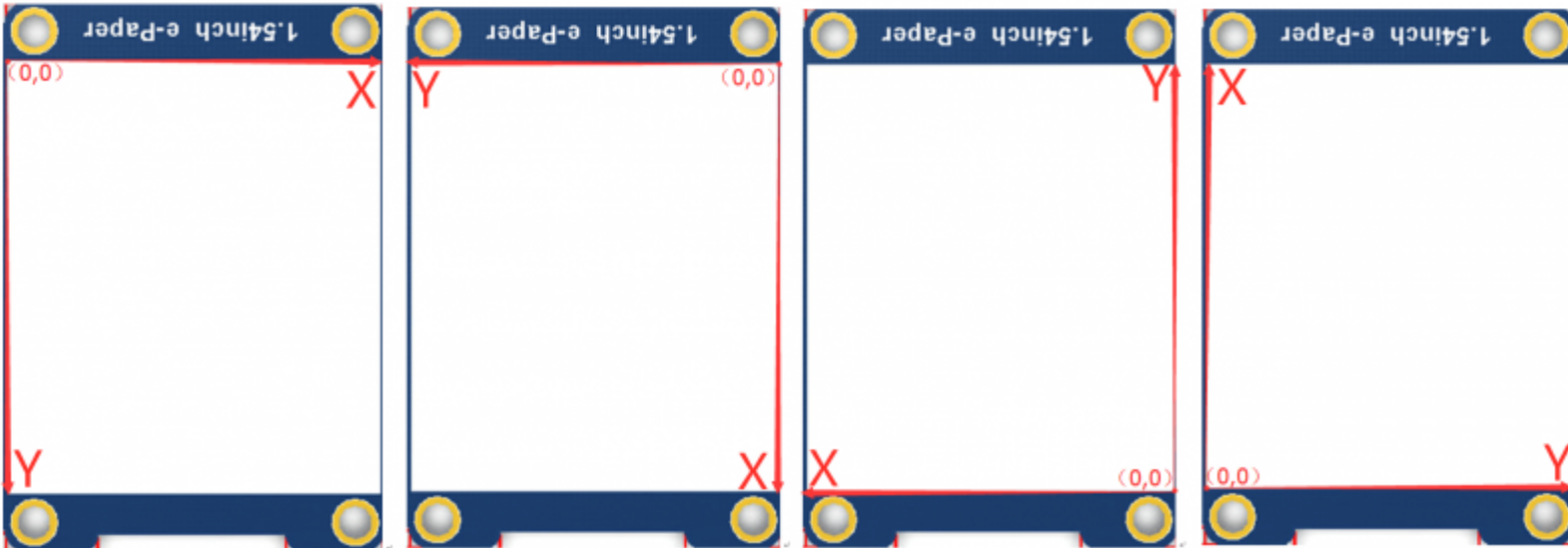
- Установить ориентацию экрана: эта функция используется для установки степени поворота, обычно она используется после Paint_SelectImage (). Вы можете установить угол поворота на 0, 90, 180, 270 градусов.

void Paint_SetRotate (UWORD Rotate)

Параметры:

Поворот : Поворот градусов , вы можете выбрать ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270 который стоит на 0 , 90 , 180 , 270 степени повторно.

【Примечание】 Три цифры ниже показывают эффект отображения в разной степени. (0 °, 90 °, 180 °, 270 °)



- Зеркальное отображение изображений: эта функция используется для зеркального отображения изображений.

```
void Paint_SetMirroring ( зеркало UBYTE )
```

Parameters:

зеркало : вы можете установить MIRROR_NONE , MIRROR_HORIZONTAL , MIRROR_VERTICAL , MIRROR_ORIGIN

- Установить пиксель: эта функция используется для установки положения и цвета пикселей в буфере. Это основная функция GUI.

```
void Paint_SetPixel ( UWORD Xpoint , UWORD Ypoint , UWORD Color )
```

Параметры:

Xpoint : X - оси в буфере ;

Ypoint : Y - оси в буфере ;

Цвет : цвет

- Очистить: эта функция используется для очистки экрана до определенного цвета.

```
void Paint_Clear ( UWORD Color )
```

Параметр:

Цвет :

- Очистить окна: эта функция используется для очистки окна. Обычно используется для отображения времени.

```
void Paint_ClearWindows ( UWORD Xstart , UWORD Ystart , UWORD Xend , UWORD Yend , UWORD Color )
```

Параметры:

Xstart : Начало координат X - оси окна ;

Ystart : начальная координата Y - оси окна ;

Xend : Конечная координата X - оси окна ;

Yend : Конечная координата Y - оси окна ;

Цвет :

- Точка рисования: нарисуйте точку в позиции (Xpoint, Ypoint) в буфере

```
void Paint_DrawPoint ( UWORD Xpoint , UWORD Ypoint , UWORD Color , DOT_PIXEL Dot_Pixel , DOT_STYLE Dot_Style )
```

Параметр:

Xpoint : X координата точки ;

Ypoint : Y координата точки ;

Цвет : цвет точки ;

Dot_Pixel : размер точки , доступно 8 размеров ;

```
typedef enum {  
    DOT_PIXEL_1X1  = 1 ,      // 1 x 1  
    DOT_PIXEL_2X2  ,          // 2 x 2  
    DOT_PIXEL_3X3  ,          // 3 x 3  
    DOT_PIXEL_4X4  ,          // 4 x 4  
    DOT_PIXEL_5X5  ,          // 5 x 5  
    DOT_PIXEL_6X6  ,          // 6 x 6
```



```

        DOT_PIXEL_7X7    ,           // 7 X 7
        DOT_PIXEL_8X8    ,           // 8 X 8
    } DOT_PIXEL ;
Dot_Style : стиль точки.
typedef enum {
    DOT_FILL_AROUND = 1 ,
    DOT_FILL_RIGHTUP ,
} DOT_STYLE ;

```

- Нарисуйте линию: нарисуйте линию для (Xstart, Ystart) до (Xend, Yend)

```

void Paint_DrawLine ( UWORD Xstart , UWORD Ystart , UWORD Xend , UWORD Yend , UWORD Color , LINE_STYLE Line_Style , LINE_STYLE Line_Style )
Параметр:
Xstart : начальная координата X - оси линии ;
Ystart : начальная координата Y - оси линии ;
Xend : Конечная координата X - оси линии ;
Yend : Конечная координата Y - оси линии ;
Цвет : цвет линии
Line_width : ширина линии , 8 размеров доступны ;
    буреиЕ перечисление {
        DOT_PIXEL_1X1 = 1 ,           // 1 x 1
        DOT_PIXEL_2X2 ,               // 2 X 2
        DOT_PIXEL_3X3 ,               // 3 X 3
        DOT_PIXEL_4X4 ,               // 4 X 4
        DOT_PIXEL_5X5 ,               // 5 X 5
        DOT_PIXEL_6X6 ,               // 6 X 6
        DOT_PIXEL_7X7 ,               // 7 X 7
        DOT_PIXEL_8X8 ,               // 8 X 8
    } DOT_PIXEL ;
Line_Style : стиль линии ;
typedef enum {
    LINE_STYLE_SOLID = 0 ,
    LINE_STYLE_DOTTED ,
} LINE_STYLE ;

```

- Рисование прямоугольника: нарисуйте прямоугольник от (Xstart, Ystart) до (Xend, Yend).

```

void Paint_DrawRectangle ( UWORD Xstart , UWORD Ystart , UWORD Xend , UWORD Yend , UWORD Color , DOT_PIXEL Line_width , DRAW_FILL Draw_Fill )
Параметр:
Xstart : начальная координата X - оси прямоугольника
Ystart : начальная координата Y - оси прямоугольника
Xend : Конечная координата X - конец прямоугольника
Yend : Конечная координата Y - конец прямоугольника
Цвет : цвет прямоугольника
Line_width : ширина ребер , доступно 8 сторон ;
    буреиЕ перечисление {
        DOT_PIXEL_1X1 = 1 ,           // 1 x 1
        DOT_PIXEL_2X2 ,               // 2 X 2
        DOT_PIXEL_3X3 ,               // 3 X 3
        DOT_PIXEL_4X4 ,               // 4 X 4
        DOT_PIXEL_5X5 ,               // 5 X 5
        DOT_PIXEL_6X6 ,               // 6 X 6
        DOT_PIXEL_7X7 ,               // 7 X 7
        DOT_PIXEL_8X8 ,               // 8 X 8
    } DOT_PIXEL ;
Draw_Fill : установить прямоугольник полностью или пусто.
typedef enum {
    DRAW_FILL_EMPTY = 0 ,

```



```
        DRAW_FILL_FULL ,
    } DRAW_FILL ;
```

- Нарисуйте круг: нарисуйте круг, используйте (X_Center Y_Center) в качестве центра;

недействительный Paint_DrawCircle (UWORD X_Center , UWORD Y_Center , UWORD Радиус , UWORD Цвет , DOT_PIXEL Line_width , DRAW_FILL Draw_Fill)

Параметр:

```
X_Center : X координата центра
Y_Center : координата Y центра
Радиус: Радиус круга
Цвет : цвет круга
Line_width : ширина круга , 8 размеров доступны
typedef enum {
    DOT_PIXEL_1X1  = 1 ,           // 1 x 1
    DOT_PIXEL_2X2  ,               // 2 X 2
    DOT_PIXEL_3X3  ,               // 3 X 3
    DOT_PIXEL_4X4  ,               // 4 X 4
    DOT_PIXEL_5X5  ,               // 5
    DOT_PIXEL_6X6  ,               // 6 X 6
    DOT_PIXEL_7X7  ,               // 7 X 7
    DOT_PIXEL_8X8  ,               // 8 X 8
} DOT_PIXEL ;
Draw_Fill: стиль круга
typedef enum {
    DRAW_FILL_EMPTY = 0 ,
    DRAW_FILL_FULL ,
} DRAW_FILL ;
```

- Рисование символа (ASCII): установите (Xstart Ystart) как точку начала letf, нарисуйте символ ASCII.

void Paint_DrawChar (UWORD Xstart , UWORD Ystart , const char Ascii_Char , sFONT * Font , UWORD Color_Foreground , UWORD Color_Background)

Параметр:

```
Xstart : X координата левого - верхнего пикселя символа ;
Ystart : Y координата левого - верхнего пикселя символа ;
Ascii_Char : персонаж Ascii ;
Шрифт : доступно 5 шрифтов ;
font8: 5 * 8
font12: 7 * 12
font16: 11 * 16
font20: 14 * 20
font24: 17 * 24
Color_Foreground : цвет символа ;
Color_Background : цвет фона ;
```

- Draw String: Установите точку (Xstart Ystart) как левый верхний пиксель, нарисуйте строку.

void Paint_DrawString_EN (UWORD Xstart , UWORD Ystart , const char * pString , sFONT * Font , UWORD Color_Foreground , UWORD Color_Background)

Параметры:

```
Xstart : X координата левого - верхнего пикселя символов ;
Ystart : Y координата левого - верхний пиксель символов ;
pString ; Указатель строки
Шрифт : доступно 5 шрифтов :
font8 : 5 * 8
font12 : 7 * 12
```

```
font16 : 11 * 16
font20 : 14 * 20
font24 : 17 * 24
Color_Foreground : цвет строки
Color_Background : цвет фона
```

- Рисование китайских символов: эта функция используется для рисования китайских шрифтов на основе шрифтов GB2312.

```
void Paint_DrawString_CN ( UWORD Xstart , UWORD Ystart , const char * pString , cFONT * font , UWORD Color_Foreground , UWORD Color_Background )
```

Параметр:

```
Xstart : координата левого - верхний пиксель символов ;
Ystart : координата левого - верхний пиксель символов ;
pString: указатель строки ;
Шрифт : GB2312 шрифты:
font12CN: 11 * 21 ( ASCII ) , 16 * 21 ( китайский )
font24CN: 24 * 41 ( ASCII ) , 32 * 41 ( китайский )
Color_Foreground : цвет строки
Color_Background : цвет фона
```

- Draw number: нарисуйте строку чисел, (Xstart, Ystart) - левый верхний пиксель.

```
void Paint_DrawNum ( UWORD Xpoint , UWORD Ypoint , int32_t Nummber , sFONT * Font , UWORD Color_Foreground , UWORD Color_Background )
```

Параметр:

```
Xstart : X координата левого - верхнего пикселя ;
Ystart : координата Y слева - в пиксель ;
Nummber: отображаемые числа. эти числа сохраняются в ИНТ формате , максимум 2147483647 ;
Шрифт : 5 шрифтов доступны:
font8: 5 * 8
font12: 7 * 12
font16: 11 * 16
font20: 14 * 20
font24: 17 * 24
Color_Foreground : цвет шрифта ;
Color_Background : цвет фона ;
```

- Время отображения: Время отображения, (Xstart, Ystart) - левый верхний пиксель. Эта функция используется для электронной бумаги, которая поддерживает частичное обновление

```
void Paint_DrawTime ( UWORD Xstart , UWORD Ystart , PAINT_TIME * pTime , sFONT * Font , UWORD Color_Background , UWORD Color_Foreground )
```

Параметр:

```
Xstart : X координата левого - верхнего пикселя символа ;
Ystart : Y координата левого - верхнего пикселя символа ;
pTime: указатель отображаемого времени ;
Шрифт : доступно 5 шрифтов ;
font8: 5 * 8
font12: 7 * 12
font16: 11 * 16
font20: 14 * 20
font24: 17 * 24
Color_Foreground : цвет шрифтов
Color_Background : цвет фона
```

- Draw image: отправить данные изображения из файла bmp в буфер

```
void Paint_DrawBitMap ( const unsigned char * image_buffer )
```

Параметр:

image_buffer : адрес данных изображения в буфере

- Прочитать локальную картинку BMP и записать ее в буфер

Платформа Linux, такая как Jetson Nano и Raspberry Pi, поддерживают непосредственное управление bmp-изображениями
Raspberry Pi & Jetson Nano : RaspberryPi & JetsonNano \ c \ lib \ GUI \ GUI_BMPfile.c (.h)

```
UBYTE GUI_ReadBmp ( const char * path , UWORD Xstart , UWORD Ystart )
```

Параметр:

путь : путь BMP фотографии

Xstart : координата X слева -верху изображения , по умолчанию 0 ;

Ystart : координата Y слева -верх изображения , по умолчанию 0 ;

Тестовый код

В приведенной выше части мы расскажем о древовидных структурах Linux-кодов, здесь мы поговорим о тестировании кода для пользователя.

Raspberry Pi & Jetson Nano: RaspberryPi & JetsonNano \ c \ examples;

Коды в примерах являются тестовым кодом, вы можете изменить определение в файле main.c для различных типов электронных документов. Например, если вы хотите протестировать 7,5-дюймовую электронную бумагу, вам нужно удалить символ «//» в

строке 42.

```
14 int main(void)
15 {
16     // Exception handling:ctrl + c
17     signal(SIGINT, Handler);
18
19     // EPD_1in54_test();
20     // EPD_1in54_V2_test();
21     // EPD_1in54b_test();
22     // EPD_1in54c_test();
23
24     // EPD_2in7_test();
25     // EPD_2in7b_test();
26
27     // EPD_2in9_test();
28     // EPD_2in9bc_test();
29     // EPD_2in9d_test();
30
31     // EPD_2in13_test();
32     // EPD_2in13_V2_test();
33     // EPD_2in13bc_test();
34     // EPD_2in13d_test();
35
36     // EPD_4in2_test();
37     // EPD_4in2bc_test();
38
39     // EPD_5in83_test();
40     // EPD_5in83bc_test();
41
42     // EPD_7in5_test();
43     // EPD_7in5bc_test();
44
45     return 0;
46 }
47
```

```
// EPD_7in5_test ();
```

изменить на

```
EPD_7in5_test ();
```

Затем скомпилируйте его снова и запустите

```
очистить
делать
судо / epd
```

STM32: STM32 \ STM32-F103ZET6 \ User \ Примеры;

тестовые коды сохраняются в этой папке, открывают проект, а затем изменяют определения в файле main.c;

Откройте проект : STM32 \ STM32-F103ZET6 \ MDK-ARM \ epd-demo.uvprojx Например, если вы хотите протестировать 7,5-дюймовую электронную бумагу, вы должны удалить символ «//» в строке 96

```
73 // EPD_lin54_test();
74 // EPD_lin54_V2_test();
75 // EPD_lin54b_test();
76 // EPD_lin54c_test();
77
78 // EPD_2in7_test();
79 // EPD_2in7b_test();
80
81 // EPD_2in9_test();
82 // EPD_2in9bc_test();
83 // EPD_2in9d_test();
84
85 // EPD_2in13_test();
86 // EPD_2in13_V2_test();
87 // EPD_2in13bc_test();
88 // EPD_2in13d_test();
89
90 // EPD_4in2_test();
91 // EPD_4in2bc_test();
92
93 // EPD_5in83_test();
94 // EPD_5in83bc_test();
95
96 // EPD_7in5_test();
97 // EPD_7in5bc_test();
```

```
// EPD_7in5_test ();
```

Измените это на

```
EPD_7in5_test ();
```






















Затем пересоберите проект и загрузите его

Python (используется для Jetson Nano \ Raspberry Pi)

Поддерживает python2.7 и python3.

Питон проще в использовании, чем с-коды

Raspberry Pi и Jetson Nano: RaspberryPi & JetsonNano \ python \ lib \

 epd1in54.py	2019/6/20 15:23	PY 文件	11 KB
 epd1in54_V2.py	2019/6/18 15:11	PY 文件	8 KB
 epd1in54b.py	2019/6/19 11:55	PY 文件	9 KB
 epd1in54c.py	2019/6/19 11:58	PY 文件	6 KB
 epd2in7.py	2019/6/20 15:32	PY 文件	10 KB
 epd2in7b.py	2019/6/21 11:35	PY 文件	10 KB
 epd2in9.py	2019/6/25 15:35	PY 文件	8 KB
 epd2in9bc.py	2019/6/20 15:29	PY 文件	6 KB
 epd2in9d.py	2019/6/20 15:31	PY 文件	13 KB
 epd2in13.py	2019/6/20 15:34	PY 文件	9 KB
 epd2in13_V2.py	2019/6/20 16:35	PY 文件	12 KB
 epd2in13bc.py	2019/6/20 16:35	PY 文件	6 KB
 epd2in13d.py	2019/6/20 11:14	PY 文件	13 KB
 epd4in2.py	2019/6/20 11:27	PY 文件	9 KB
 epd4in2bc.py	2019/6/20 11:54	PY 文件	6 KB
 epd5in83.py	2019/6/20 13:52	PY 文件	8 KB
 epd5in83bc.py	2019/6/20 14:46	PY 文件	8 KB
 epd7in5.py	2019/6/20 14:46	PY 文件	8 KB
 epd7in5bc.py	2019/6/20 14:56	PY 文件	8 KB
 epdconfig.py	2019/6/21 11:42	PY 文件	3 KB
 Font.ttc	2019/6/18 10:47	TrueType Collect...	5,057 KB

epdconfig.py

- Инициализировать модуль и дескриптор выхода:

```
def module_init ( )
def module_exit ( )
```

Запись:

1. Функции используются для настройки GPIO до и после вождения электронной бумаги.

2. Если на вашей плате напечатано Rev2.1, модуль переходит в режим low-ultra после Module_Exit (). (как мы тестируем, ток составляет около 0 в этом режиме);

- GPIO чтение / запись:

```
def digital_write ( pin , value )
def digital_read ( pin )
```

- SPI записывают данные

```
def spi_writebyte ( data )
```

epdxxx.py (xxx - это тип электронной бумаги)

- Initialize e-paper: эту функцию следует использовать в начале. Его также можно использовать для пробуждения электронной бумаги из спящего режима.

```
Для 1,54 дюйма e - Бумага , 1,54 дюйма e - Бумага V2, 2,13 дюйма e - Бумага , 2,13 дюйма e - Бумага V2, 2,13 дюйма e - Бумага ( D ), 2,9 дюйма e - Бумага , 2,9 дюйма e - Бумага ( D )
def init ( self , update ) # update должно быть lut_full_update или lut_partial_update
Другие типы :
def init ( self )
```

- Очистить электронную бумагу: эта функция используется для очистки электронной бумаги от белого;

```
def Clear ( self )
def Clear ( self , color ) # Некоторые типы электронных бумаг должны использовать эту функцию для очистки экрана
```

- Преобразовать изображение в массивы


```
def getbuffer ( self , image )
```

■ Передача одного кадра данных и отображения

```
# Для двухцветного электронной бумаги
дисплея четкости ( самостоятельная , изображений )
# для трехцветного электронной бумаги
DEF дисплея ( самостоятельно , blackimage , redimage )

Есть несколько exсerption: < бр />
Для гибких е - Бумага 2.13inch е - бумага ( D ) , 2.9inch е - бумаги ( D ) , частичное обновление следует использовать
def DisplayPartial ( self , image ) #
Поскольку обновляются контроллеры электронной бумаги 1.54 дюйма V2, 2.13 дюймовой бумаги V2, при частичном обновлении они должны сначала использовать displayPartBaseImage () для
отображения статического фона, а затем использовать displayPart () для динамично отображать.
def displayPartBaseImage ( self , image )
def displayPart ( self , image )
```




















■ Войдите в спящий режим

```
def sleep ( self )
```

erpd_xxx_test.py (xxx - это тип электронной бумаги)

Примеры Python сохраняются в каталоге:

Raspberry Pi & Jetson Nano: RaspberryPi & JetsonNano \ python \ examples \ Если в вашей ОС установлен python2, вы должны запустить примеры, как показано ниже:

 epd_1in54_test.py	2019/6/19 15:30	PY 文件	3 KB
 epd_1in54_V2_test.py	2019/6/19 15:31	PY 文件	3 KB
 epd_1in54b_test.py	2019/6/19 15:31	PY 文件	3 KB
 epd_1in54c_test.py	2019/6/19 15:31	PY 文件	3 KB
 epd_2in7_test.py	2019/6/19 15:31	PY 文件	3 KB
 epd_2in7b_test.py	2019/6/19 15:30	PY 文件	4 KB
 epd_2in9_test.py	2019/6/19 15:55	PY 文件	4 KB
 epd_2in9bc_test.py	2019/6/19 17:35	PY 文件	4 KB
 epd_2in9d_test.py	2019/6/19 18:22	PY 文件	4 KB
 epd_2in13_test.py	2019/6/19 19:46	PY 文件	3 KB
 epd_2in13_V2_test.py	2019/6/20 9:30	PY 文件	3 KB
 epd_2in13bc_test.py	2019/6/20 10:39	PY 文件	4 KB
 epd_2in13d_test.py	2019/6/20 11:15	PY 文件	3 KB
 epd_4in2_test.py	2019/6/20 11:30	PY 文件	3 KB
 epd_4in2bc_test.py	2019/6/20 12:00	PY 文件	4 KB
 epd_5in83_test.py	2019/6/20 13:58	PY 文件	3 KB
 epd_5in83bc_test.py	2019/6/20 14:22	PY 文件	4 KB
 epd_7in5_test.py	2019/6/20 14:35	PY 文件	3 KB
 epd_7in5bc_test.py	2019/6/20 14:50	PY 文件	4 KB

```
sudo python epd_7in5_test. py
```

Если это python3, команды должны быть:

```
sudo python3 epd_7in5_test. py
```

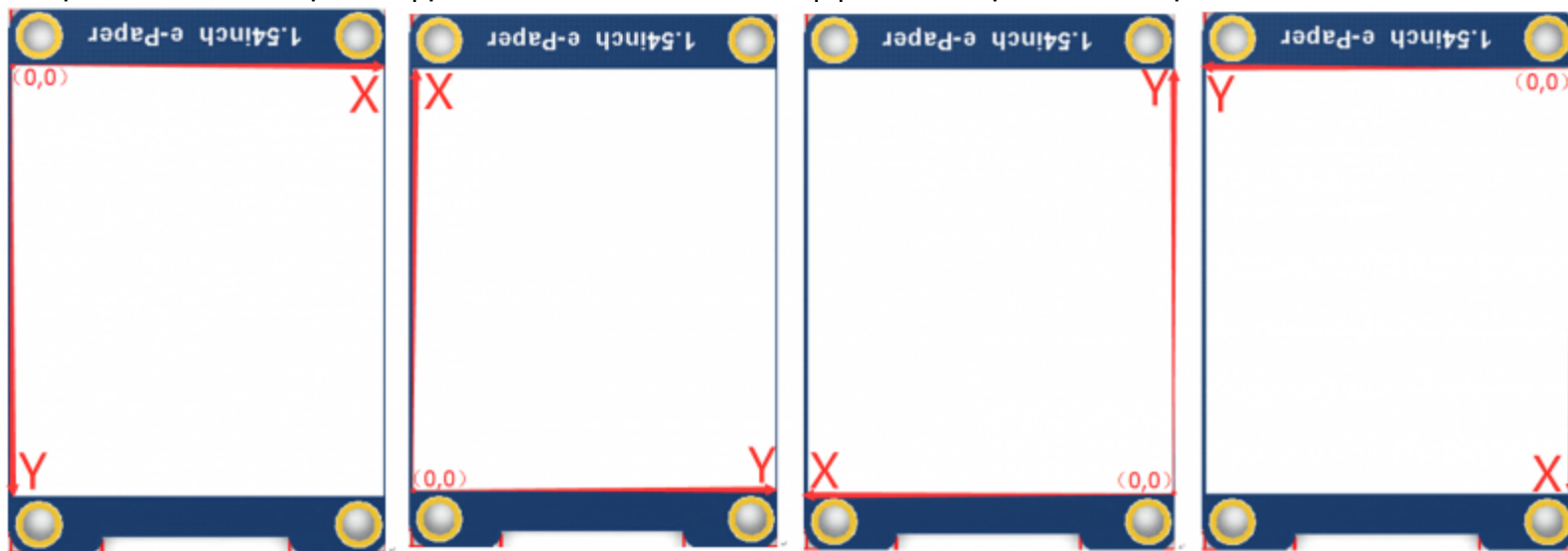
Примечание. Вы можете изменить epd_7inch5_test.py на определенный тип, который вы используете.

ориентация

Чтобы повернуть дисплей, вы можете использовать функцию транспонирования

```
blackimage = blackimage.transpose (Image.ROTATE_270)
redimage = redimage.transpose (Image.ROTATE_270)
```

【Примечание】 Три цифры ниже показывают эффект отображения в разной степени. (0 °, 90 °, 180 °, 270 °)



Arduino

Поскольку Arduino не имеет полной оперативной памяти для отображения динамического изображения, мы не предоставляем другие функции для него. Если вы хотите использовать Arduino, мы рекомендуем вам использовать электронную бумагу Waveshare Shield .

Служба поддержки



Свяжитесь с продавцом (быстрый ответ и наиболее рекомендуется)

или отправьте электронное **письмо** на адрес **service@waveshare.com** (недостаточно быстро, но, пожалуйста, наберитесь терпения) за помощью.

Наше рабочее время: 09: 00-18: 00 (**UTC + 8 c** понедельника по субботу)

- Последнее изменение этой страницы - 26 марта 2020 года, в 12:18.