StarFive 赛昉科技

Using VisionFive 2 SPI to Support LCD Display

with Python

Application Note

Version: 1.0

Date: 2022/11/30

Doc ID: VisionFive2-ANEN-006

Legal Statements

Important legal notice before reading our documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Shanghai StarFive Technology Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

All material appearing in this document is protected by copyright and is the property of StarFive. You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. StarFive authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services.

Contact Us

Address: Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com

Email: sales@starfivetech.com(sales) , support@starfivetech.com(support)

Preface

About this guide and technical support information.

About this document

This application note provides steps to use VisionFive 2's SPI to make a 2.4inch LCD display with specified pictures.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2022/11/30	The first official release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.
- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.
- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

Contents

List of Tables

List of Figures

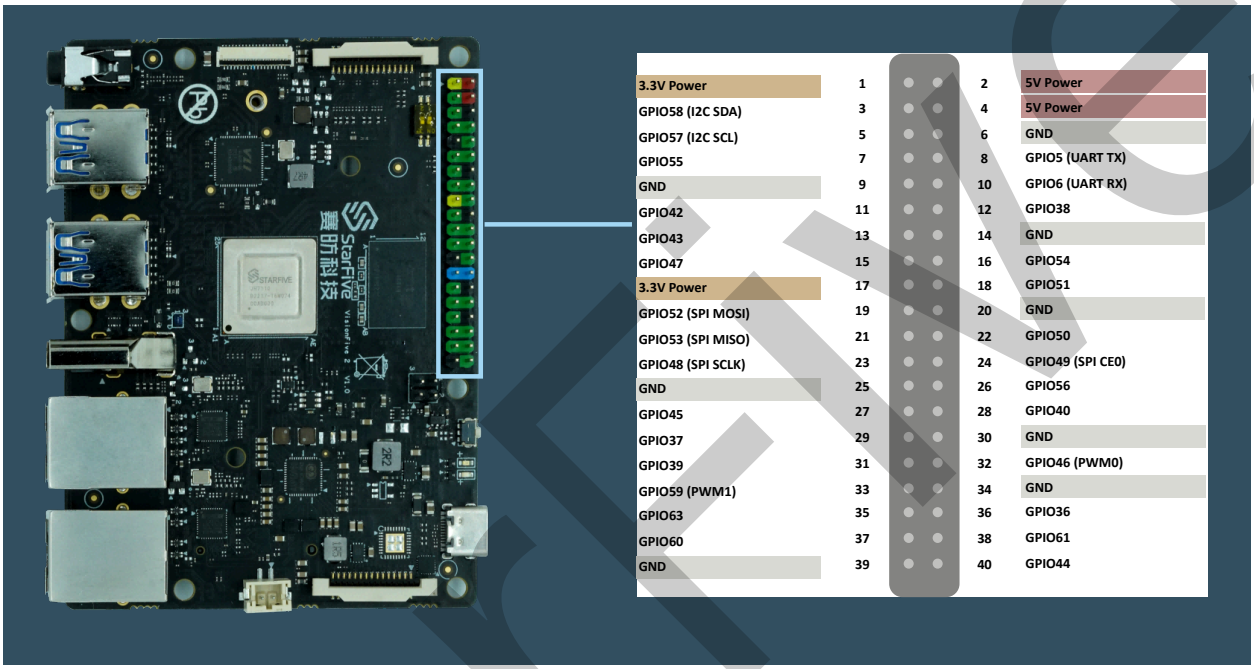
1. Introduction

This application note provides steps to use VisionFive 2's SPI to make a 2.4inch LCD display with specified pictures.

1.1. 40-Pin Header Definition

The following figure shows the location of the 40-pin header on VisionFive 2.

Figure 1-1 40-Pin Definition



2. Preparation

Before executing the demo program, make sure you prepare the following:

2.1. Environment Requirements

The environment requirements are as follows:

- Linux Kernel: Linux 5.15
- OS: Debian 12
- SBC: VisionFive 2
- SoC: JH7110

2.2. Preparing Hardware

Prepare the following hardware items before running the demo code:

Table 2-1 Hardware Preparation

Type	M/O*	Item	Notes
General	M	VisionFive 2 single board computer	-
General	M	• 32 GB (or more) micro-SD card • micro-SD card reader • Computer (Windows/macOS/Linux) • USB to serial converter (3.3 V I/O) • Ethernet cable • Power adapter (5 V / 3 A) • USB Type-C Cable	These items are used for flashing Debian OS into a micro-SD card.
SPI LCD		• 2.4inch LCD Module • Dupont Line	-

Note:
*: M: Mandatory, O: Optional

2.2.1. Hardware Setup

The following table and figure describe how to connect LCD to the 40-pin header:

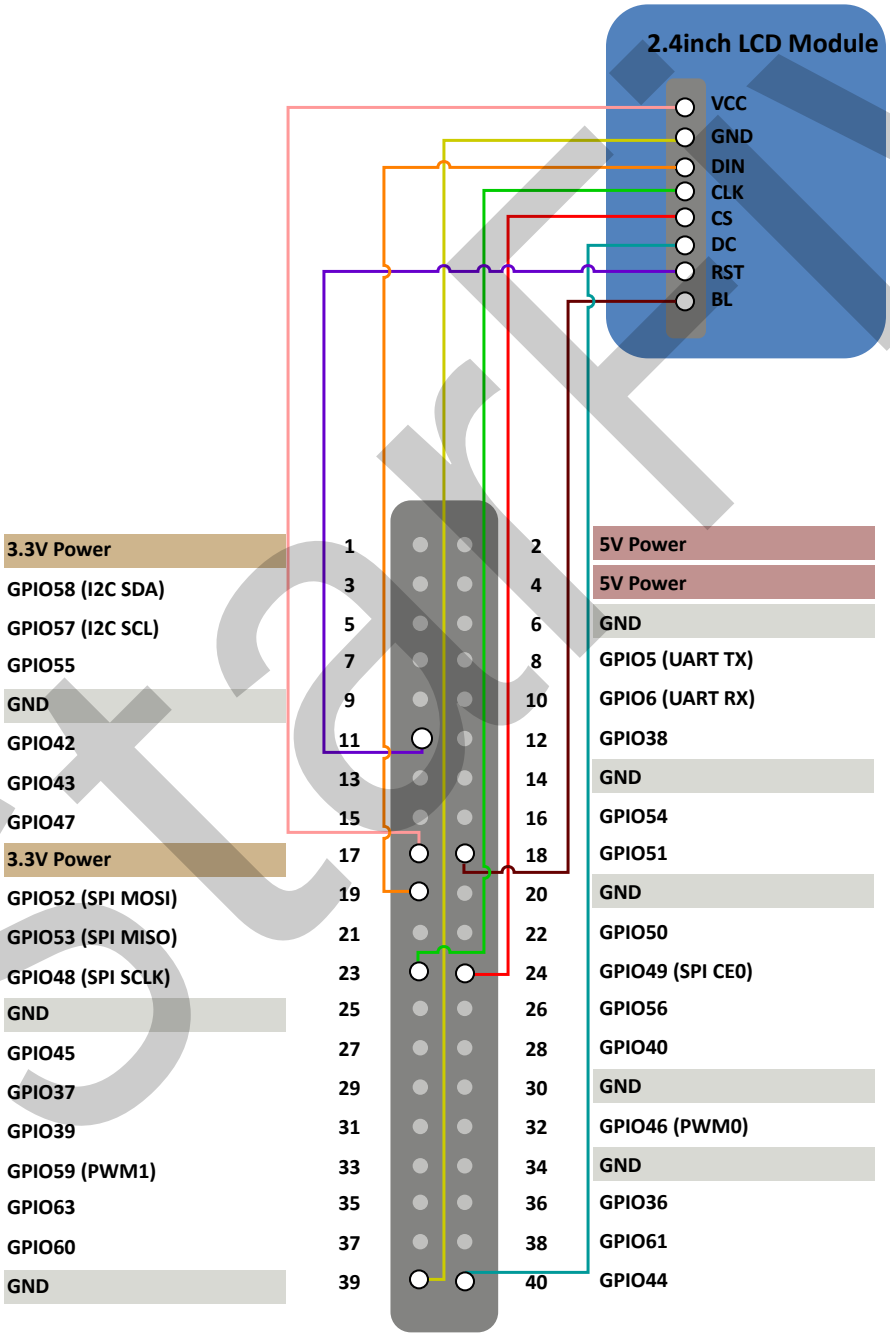
Table 2-2 Connect the 2.4inch LCD to the 40-Pin Header

2.4inch LCD Module	40-Pin GPIO Header	
	Pin Number	Pin Name
VCC	17	3.3V Power
GND	39	GND
DIN	19	GPIO52 (SPI MOSI)

Table 2-2 Connect the 2.4inch LCD to the 40-Pin Header (continued)

2.4inch LCD Module	40-Pin GPIO Header	
	Pin Number	Pin Name
CLK	23	GPIO48 (SPI SCLK)
CS	24	GPIO49 (SPI CE0)
DC	40	GPIO44
RST	11	GPIO42
BL	18	GPIO51

Figure 2-1 Connect the 2.4inch LCD to the 40-Pin Header



2.3. Preparing Software

Make sure the following procedures are performed:

Note:
The Python project, `VisionFive.gpio`, is applicable for VisionFive, VisionFive 2, and JH7110 EVB.

1. Flash Debian OS into a Micro-SD card as described in the *Flashing OS to a Micro-SD Card* section in the *VisionFive 2 Single Board Computer Quick Start Guide*.
2. Log into the Debian and make sure VisionFive 2 is connected to the Internet. For detailed instructions, refer to the *Using SSH over Ethernet* or *Using a USB to Serial Converter* section in the *VisionFive 2 Single Board Computer Quick Start Guide*.
3. Extend the partition on Debian as described in *Extend Partition* in the *VisionFive 2 Single Board Computer Quick Start Guide*.
4. Execute the following command to install PIP on Debian:

   ```
   apt-get install python3-pip
   ```

5. Execute the `pip` command on VisionFive 2 Debian to install the `VisionFive.gpio` package:

   ```
   sudo pip install VisionFive.gpio
   ```

   Alternatively, you can execute the following command:

   ```
   sudo pip3 install VisionFive.gpio
   ```

6. (Optional) If you copy the source code to the local directory under VisionFive 2 Debian, execute the following commands under the source code directory:

   Tip:
   The source code can be downloaded by clicking the following link: VisionFive.gpio.

   ```
   sudo apt-get install python3-dev
   sudo python setup.py install
   ```

   Alternatively, you can execute the following command:

   ```
   sudo python3 setup.py install
   ```

3. Running Demo Code

To run the demo code, perform the following on VisionFive 2 Debian:

1. Locate to the directory where the test code, `2.4inch_LCD_demo`, exists:

    a. Execute the following command to get the directory where `VisionFive.gpio` exists:

    ```
    pip show VisionFive.gpio
    ```

    Example Result:

    ```
    Location: /usr/local/lib64/python3.9/site-packages
    ```

    Note:

    The actual output depends on how the application is installed.

    b. Execute the following to enter the directory, for example, `/usr/local/lib64/python3.9/site-packages` as indicated in the previous step output:

    ```
    cd /usr/local/lib64/python3.9/site-packages
    ```

    c. Execute the following command to enter the `sample-code` directory:

    ```
    cd ./VisionFive/sample-code/
    ```

    d. Execute the following command to enter the directory where the test code, `2.4inch_LCD_demo`, exists:

    ```
    cd ./lcddemo/example/
    ```

2. Under the `sample-code` directory, execute the following command to execute the demo code:

    ```
    sudo python 2.4inch_LCD_demo
    ```

    Alternatively, you can execute the following command:

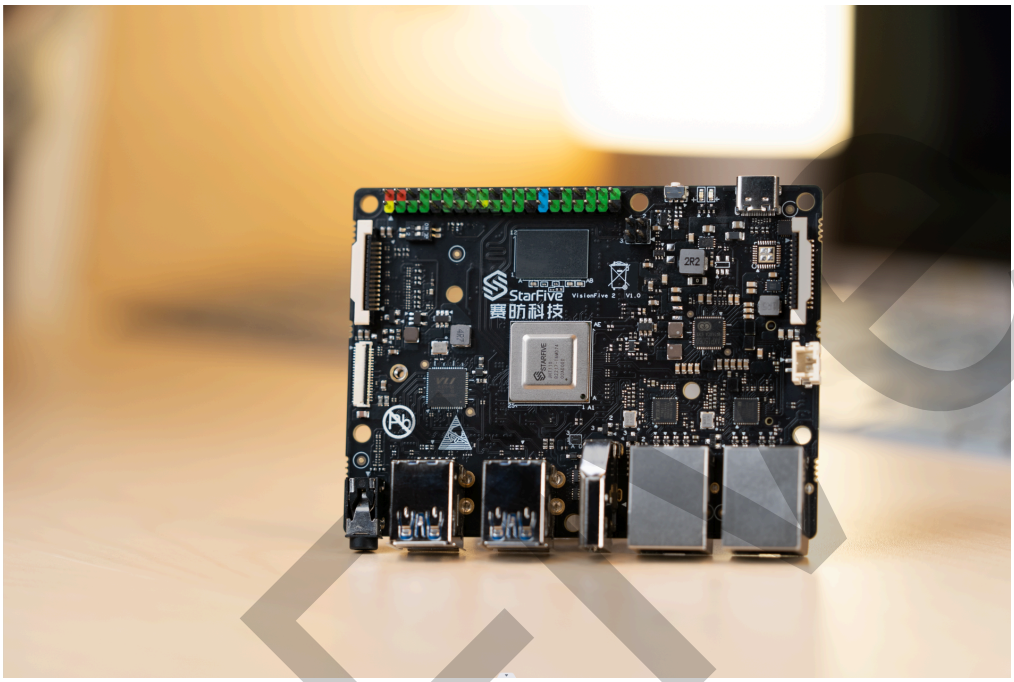
    ```
    sudo python3 2.4inch_LCD_demo
    ```

    Result:

◦ On the 2.4inch LCD:

▪ First, the following picture with the StarFive logo will be displayed for two seconds.

Figure 3-1 Example Output



▪ Then the following two official example figures will be displayed in turn.

Figure 3-2 Example Output

◦ The terminal output is as the following.

```
----------lcd demo----------
Set SPI mode successfully
spi mode: 0x0
bits per word: 8
max speed: 40000000 Hz(40000 kHz)
2022-07-04 16:40:40
2022-07-04 16:40:41
2022-07-04 16:40:41
2022-07-04 16:40:42
2022-07-04 16:40:42
2022-07-04 16:40:43
2022-07-04 16:40:44
2022-07-04 16:40:44
2022-07-04 16:40:45
```

The output indicates:

- that the SPI mode is set successfully
- the SPI mode
- the date and time when all the above three figures are displayed

4. Demo Source Code

The Python source code of this demo is provided for reference purpose only.

`2.4inch_LCD_demo.py`:

```
#!/usr/bin/python
'''
Please make sure the 2.4inch LCD Moudle is connected to the correct pins.
The following table describes how to connect the 2.4inch LCD Module to the 40-pin header.
-------------------------------------------------
__2.4inch LCD Module___Pin Number_____Pin Name
    VCC                  17           3.3 V Power
    GND                  39             GND
    DIN                  19           SPI MOSI
    CLK                  23           SPI SCLK
    CS                   24           SPI CE0
    DC                   40            GPIO44
    RST                  11            GPIO42
    BL                   18            GPIO51
-------------------------------------------------
'''

import os
import sys
import time
import logging
from PIL import Image
sys.path.append("..")

import VisionFive.boardtype as board_t
from  lib import LCD2inch4_lib

'''
Demo modification ans new function description
-------------------------------------------------------------
  I.   add the clear() function to fill LCD screen with white
  II.  give a hexadecimal value of white
  III. cycle through multiple pictures
-------------------------------------------------------------
'''

WHITE = 0xFF

def main():
    print('-----------lcd demo------------')

    #Determining cpu Type: 1 means visionfive1; 2 means visionfive 2
    vf_t = board_t.boardtype()
    if vf_t == 1:
        SPI_DEVICE = "/dev/spidev0.0"
    elif vf_t == 2:
        SPI_DEVICE = "/dev/spidev1.0"
    else:
        print('This module can only be run on a VisionFive board!')
        return 0

    """The initialization settings of 2inch and 2.4inch are distinguished"""
    disp = LCD2inch4_lib.LCD_2inch4(11, 40, SPI_DEVICE)
    #disp.lcd_init()
    disp.lcd_init_2inch4()

    disp.lcd_clear(WHITE)

    if vf_t == 1:
        image = Image.open('./visionfive.bmp')
    elif vf_t == 2:
        image = Image.open('./visionfive2.bmp')
    else:
        return
```

```
    disp.lcd_ShowImage(image, 0, 0)
    time.sleep(2)

    """add the part of displaying pictures circularly"""
    while True:
        print(time.strftime("%Y-%m-%d %H:%M:%S",time.localtime(time.time())))

        """rotate the picture 90 degrees anticlockwise"""
        """to keep consistent with the display direction of other pictures"""
        image = Image.open('./LCD_2inch4_parrot.bmp')
        image = image.transpose(Image.Transpose.ROTATE_90)
        disp.lcd_ShowImage(image, 0, 0)

        image = Image.open('./LCD_2inch.jpg')
        disp.lcd_ShowImage(image, 0, 0)


if __name__=="__main__":
    main()
```

LCD2inch4_lib.py:

```
import os
import sys
import time
import logging
import VisionFive.spi as spi
import VisionFive.gpio as gpio
import numpy as np
from PIL import Image,ImageDraw,ImageFont

class LCD_2inch4():
    width  = 240
    height = 320
    def __init__(self, rst_pin, dc_pin, dev):
        self.rstpin = rst_pin
        self.dcpin  = dc_pin
        self.spidev  = dev
        spi.getdev(self.spidev)

        #Reset the maximum clock frequency of communication.
        #The display speed of the picture is positively correlated with the clock frequency.
        spi.setmode(40000000, 0, 8)
        gpio.setup(self.rstpin, gpio.OUT)
        gpio.setup(self.dcpin, gpio.OUT)

    def __del__(self):
        spi.freedev()

    #Add a short delay for each change of electrical level.
    def lcd_reset(self):
        gpio.output(self.rstpin, gpio.HIGH)
        time.sleep(0.01)
        gpio.output(self.rstpin, gpio.LOW)
        time.sleep(0.01)
        gpio.output(self.rstpin, gpio.HIGH)
        time.sleep(0.01)

    def lcd_spisend(self, data):
        spi.transfer(data)

    def lcd_sendcmd(self, cmd):
        gpio.output(self.dcpin, gpio.LOW)
        spi.transfer(cmd)

    def lcd_senddata(self,data):
        gpio.output(self.dcpin, gpio.HIGH)
        spi.transfer(data)


    #Write multiple bytes.
```

```
    def lcd_sendnbytes(self, data):
        gpio.output(self.dcpin, gpio.HIGH)
        spi.write(data)

    #Common registers' initialization of the 2.4inch LCD module.
    def lcd_init_2inch4(self):
        self.lcd_reset()

        self.lcd_sendcmd(0x11) #Sleep out.

        self.lcd_sendcmd(0xCF) #Power Control B.
        self.lcd_senddata(0x00)
        self.lcd_senddata(0xC1)
        self.lcd_senddata(0x30)

        self.lcd_sendcmd(0xED) #Power on sequence control.
        self.lcd_senddata(0x64)
        self.lcd_senddata(0x03)
        self.lcd_senddata(0x12)
        self.lcd_senddata(0x81)

        self.lcd_sendcmd(0xE8) #Driver Timing Control A.
        self.lcd_senddata(0x85)
        self.lcd_senddata(0x00)
        self.lcd_senddata(0x79)

        self.lcd_sendcmd(0xCB) #Power Control A.
        self.lcd_senddata(0x39)
        self.lcd_senddata(0x2C)
        self.lcd_senddata(0x00)
        self.lcd_senddata(0x34)
        self.lcd_senddata(0x02)

        self.lcd_sendcmd(0xF7) #Pump ratio control.
        self.lcd_senddata(0x20)

        self.lcd_sendcmd(0xEA) #Driver Timing Control B.
        self.lcd_senddata(0x00)
        self.lcd_senddata(0x00)

        self.lcd_sendcmd(0xC0) #Power Control 1.
        self.lcd_senddata(0x1D) #VRH[5:0]

        self.lcd_sendcmd(0xC1) #Power Control 2.
        self.lcd_senddata(0x12) #SAP[2:0], BT[3:0].

        self.lcd_sendcmd(0xC5) #VCOM Control 1.
        self.lcd_senddata(0x33)
        self.lcd_senddata(0x3F)

        self.lcd_sendcmd(0xC7) #VCOM Control 2.
        self.lcd_senddata(0x92)

        self.lcd_sendcmd(0x3A) #COLMOD: Pixel Format Set.
        self.lcd_senddata(0x55)

        self.lcd_sendcmd(0x36) #Memory Access Control.
        self.lcd_senddata(0x08)

        self.lcd_sendcmd(0xB1) #Frame rate control(in normal mode/full colors).
        self.lcd_senddata(0x00)
        self.lcd_senddata(0x12)

        self.lcd_sendcmd(0xB6) #Display function control.
        self.lcd_senddata(0x0A)
        self.lcd_senddata(0xA2)

        self.lcd_sendcmd(0x44) #Set_Tear_Scanline
        self.lcd_senddata(0x02);

        self.lcd_sendcmd(0xF2) #Gamma Function Disable
        self.lcd_senddata(0x00)
```

```
        self.lcd_sendcmd(0x26) #Gamma curve selected.
        self.lcd_senddata(0x01)

        self.lcd_sendcmd(0xE0) #Set Gamma.
        self.lcd_senddata(0x0F)
        self.lcd_senddata(0x22)
        self.lcd_senddata(0x1C)
        self.lcd_senddata(0x1B)
        self.lcd_senddata(0x08)
        self.lcd_senddata(0x0F)
        self.lcd_senddata(0x48)
        self.lcd_senddata(0xB8)
        self.lcd_senddata(0x34)
        self.lcd_senddata(0x05)
        self.lcd_senddata(0x0C)
        self.lcd_senddata(0x09)
        self.lcd_senddata(0x0F)
        self.lcd_senddata(0x07)
        self.lcd_senddata(0x00)

        self.lcd_sendcmd(0XE1); #Set Gamma.
        self.lcd_senddata(0x00)
        self.lcd_senddata(0x23)
        self.lcd_senddata(0x24)
        self.lcd_senddata(0x07)
        self.lcd_senddata(0x10)
        self.lcd_senddata(0x07)
        self.lcd_senddata(0x38)
        self.lcd_senddata(0x47)
        self.lcd_senddata(0x4B)
        self.lcd_senddata(0x0A)
        self.lcd_senddata(0x13)
        self.lcd_senddata(0x06)
        self.lcd_senddata(0x30)
        self.lcd_senddata(0x38)
        self.lcd_senddata(0x0F)
        self.lcd_sendcmd(0x29) #Display On.

    def lcd_setPos(self, Xstart, Ystart, Xend, Yend):

        self.lcd_sendcmd(0x2a)
        self.lcd_senddata(Xstart >>8)
        self.lcd_senddata(Xstart & 0xff)
        self.lcd_senddata((Xend - 1) >> 8)
        self.lcd_senddata((Xend - 1) & 0xff)
        self.lcd_sendcmd(0x2b)
        self.lcd_senddata(Ystart >>8)
        self.lcd_senddata(Ystart & 0xff)
        self.lcd_senddata((Yend - 1) >> 8)
        self.lcd_senddata((Yend - 1) & 0xff)
        self.lcd_sendcmd(0x2C)

    def lcd_clear(self, color):

        #Clear contents of image buffer.
        _buffer = [color]*(self.width * self.height *2)

        self.lcd_setPos(0, 0, self.width, self.height)
        gpio.output(self.dcpin, gpio.HIGH)

        #Multi-byte-write.
        self.lcd_sendnbytes(_buffer)

    def lcd_ShowImage(self, Image, Xstart, Ystart):
        #Set buffer to the value of the Python Imaging Library image.
        #Write display buffer to the physical display.
        imwidth, imheight = Image.size

        if imwidth == self.height and imheight == self.width:
            img = np.asarray(Image)
            pix = np.zeros((self.width, self.height,2), dtype = np.uint8)
```

```
            pix[...,[0]] = np.add(np.bitwise_and(img[...,[0]],0xF8),np.right_shift(img[...,[1]],5))
            pix[...,[1]] = np.add(np.bitwise_and(np.left_shift(img[...,[1]],3),0xE0),
np.right_shift(img[...,[2]],3))
            pix = pix.flatten().tolist()

            self.lcd_sendcmd(0x36) #Define read/write scanning direction of frame memory.
            self.lcd_senddata(0x78)
            self.lcd_setPos(0, 0, self.height, self.width)

            gpio.output(self.dcpin, gpio.HIGH)

            #Multi-byte-write.
            self.lcd_sendnbytes(pix)
        else :
            img = np.asarray(Image)
            pix = np.zeros((imheight, imwidth, 2), dtype = np.uint8)

            pix[...,[0]] = np.add(np.bitwise_and(img[...,[0]],0xF8),np.right_shift(img[...,[1]],5))
            pix[...,[1]] = np.add(np.bitwise_and(np.left_shift(img[...,[1]],3),0xE0),
np.right_shift(img[...,[2]],3))

            pix = pix.flatten().tolist()

            self.lcd_sendcmd(0x36)
            self.lcd_senddata(0x08)
            self.lcd_setPos(0, 0, self.width, self.height)

            gpio.output(self.dcpin, gpio.HIGH)
            #Multi-byte-write.
            self.lcd_sendnbytes(pix)
```