StarFive
赛昉科技

使用昉·星光单板计算机的SPI点亮LCD屏幕

Python版本
应用说明
版本：1.0
日期：2022/07/29
Doc ID: VisionFive-ANCH-013-1.0

法律声明

阅读本文件前的重要法律告知。

版权注释

本文档中的说明均基于“视为正确”提供，可能包含部分错误。内容可能因产品开发而定期更新或修订。上海赛昉科技有限公司（以下简称“赛昉科技”）保留对本协议中的任何内容进行更改的权利，恕不另行通知。

赛昉科技明确否认任何形式的担保、解释和条件，无论是明示的还是默示的，包括但不限于适销性、特定用途适用性和非侵权的担保或条件。

赛昉科技无需承担因应用或使用任何产品或电路而产生的任何责任，并明确表示无需承担任何及所有连带责任，包括但不限于间接、偶然、特殊、惩戒性或由此造成的损害。

联系我们：

地址： 浦东新区盛夏路61弄张润大厦2号楼502，上海市，201203，中国

网站： http://www.starfivetech.com

邮箱：sales@starfivetech.com （销售） support@starfivetech.com（支持）

前言

关于本指南和技术支持信息

关于本手册

本应用说明提供使用昉·星光单板计算机的SPI让2.4英寸的LCD显示器显示出指定图片的步骤。

修订历史

表 0-1 修订历史

版本	发布说明	修订
V1.0	2022/07/29	首次发布。

注释和注意事项

本指南中可能会出现以下注释和注意事项：

- **提示：**
  建议如何在某个主题或步骤中应用信息。
- **注：**
  解释某个特例或阐释某个重要的点。
- **重要：**
  指出与某个主题或步骤有关的重要信息。
- **警告：**
  表明某个操作或步骤可能会导致数据丢失、安全问题或性能问题。
- **警告：**
  表明某个操作或步骤可能导致物理伤害或硬件损坏。

目录

表格清单

插图清单

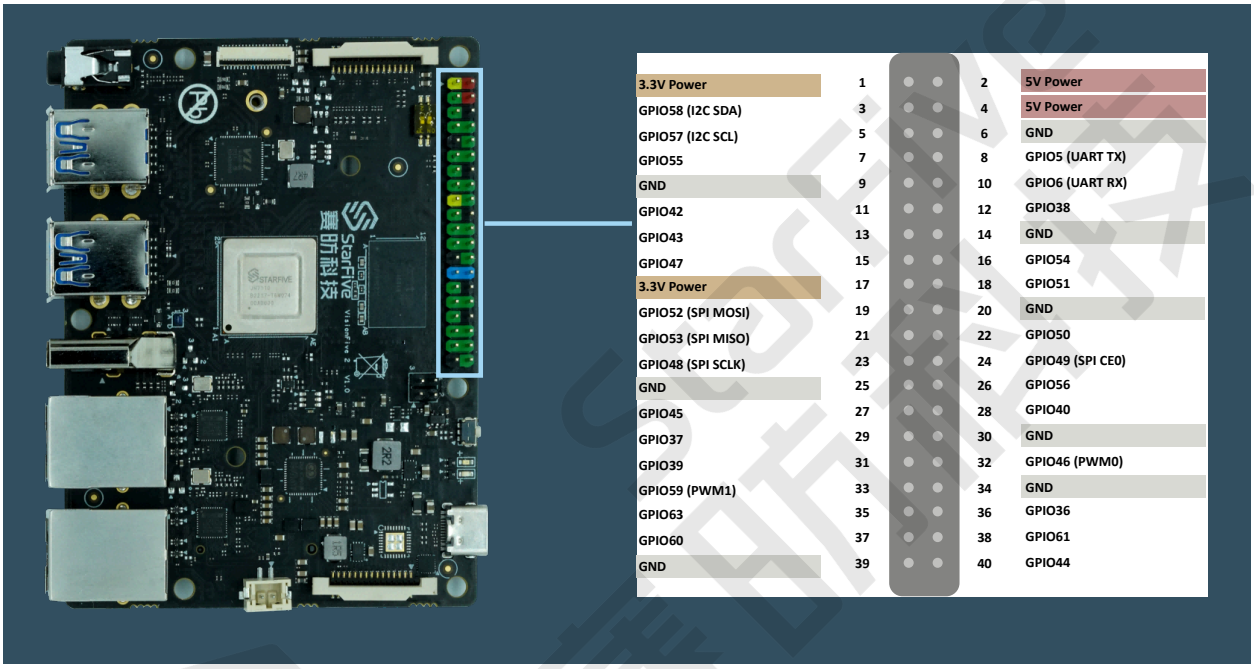
1. 介绍

本应用说明提供使用昉·星光单板计算机的SPI让2.4英寸的LCD显示器显示出指定图片的步骤。

1.1. 40-Pin Header定义

下图以昉·星光单板计算机开发板为例说明40-Pin Header的位置：

图 1-1 40-Pin定义



2. 准备

在执行演示程序之前，务必确认已准备好以下项目：

2.1. 准备硬件

在执行演示程序前，请务必准备以下硬件：

表 2-1 硬件准备

类型	M/O*	项目	注释
通用	M	赛昉科技 单板计算机	可使用以下单板计算机： • 星光板 • 昉·星光单板计算机
通用	M	• 容量不低于32 GB的Micro-SD卡 • Micro-SD卡读卡器 • 计算机（PC/Mac/Linux） • USB转串口转换器（3.3 V I/O） • 网线 • 电源适配器（5 V/ 3 A） • USB Type-C数据线	上述项目用于将Fedora OS烧录到Micro-SD上。
SPI LCD		• 2.4英寸LCD模块 • 杜邦线	-

注：
*: M：必须。O：可选

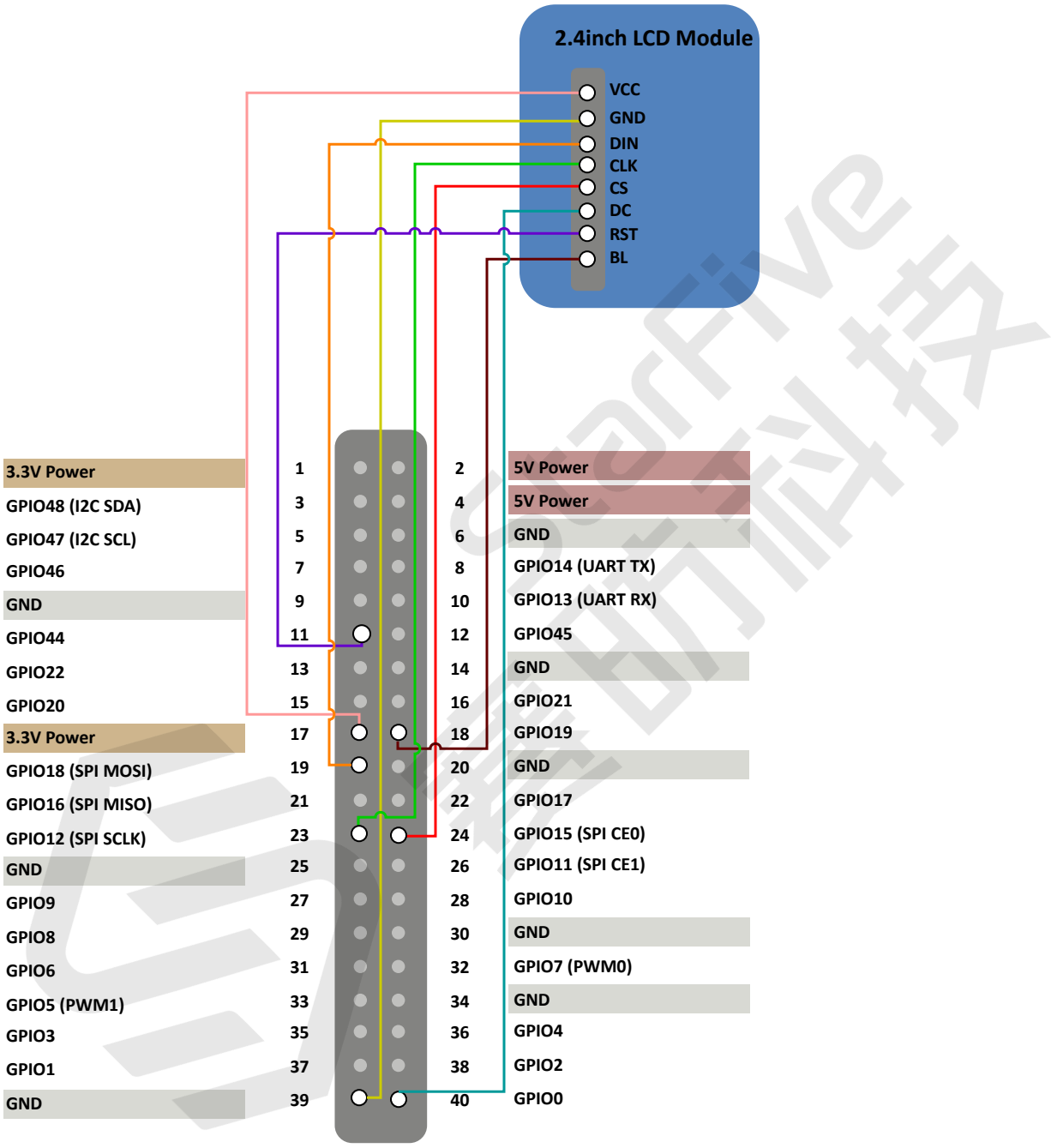
2.1.1. 连接硬件

以下图表描述了如何将LCD连接到40-Pin Header上：

表 2-2 将2.4英寸LCD连接到40-Pin Header上

2.4英寸LCD模块	40-Pin GPIO Header	
	引脚序号	引脚名
VCC	17	3.3V Power
GND	39	GND
DIN	19	GPIO18 (SPI MOSI)
CLK	23	GPIO12 (SPI SCLK)
CS	24	GPIO15 (SPI CE0)
DC	40	GPIO0
RST	11	GPIO44
BL	18	GPIO19

图 2-1 将2.4英寸LCD连接到40-Pin Header上



2.2. 准备软件

确认按照以下步骤进行操作:

1. 按照《昉·星光单板计算机快速入门指南》中的“将*Fedora*烧录到*Micro SD*上”章节，将Fedora OS烧录到Micro SD卡上。
2. 登录Fedora并确保昉·星光单板计算机已联网。有关详细说明，请参阅《昉·星光单板计算机快速入门指南》中“通过以太网使用*SSH*登录”或“使用*USB*转串口转换器连接并登录”章节。
3. 在昉·星光单板计算机Fedora上执行`pip`命令，以安装`VisionFive.gpio`包：

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

   或者，您也可以执行以下命令：

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

4. （可选）如果您将源代码复制到昉·星光单板计算机Fedora的本地目录下，请在源代码目录下执行以下命令：

   提示：
   点击以下链接可下载源代码：VisionFive.gpio。

   ```
   sudo yum install python-devel python3-devel
   sudo python setup.py install
   ```

   或者，您也可以执行以下命令：

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

3. 执行演示代码

执行以下操作，以在昉·星光单板计算机Fedora上运行演示代码：

1. 找到测试代码`2.4inch_LCD_demo`所在的目录：

   a. 执行以下命令以获取`VisionFive.gpio`所在的目录：

   ```
   pip show VisionFive.gpio
   ```

   示例结果：

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

   注：

   实际输出取决于应用的安装方式。

   b. 如前一步输出中所示，执行以下操作进入目录`/usr/local/lib64/python3.9/site-packages`：

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

   c. 执行以下命令进入`sample-code`目录：

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

   d. 执行以下命令，以进入测试代码`2.4inch_LCD_demo`所在的目录：

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

2. 在`sample-code`目录下，执行以下命令以运行演示代码：

   ```
   sudo python 2.4inch_LCD_demo
   ```

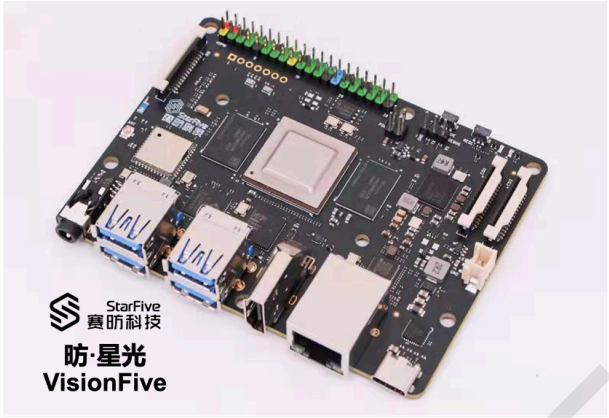
   或者，您也可以执行以下命令：

   ```
   sudo python3 2.4inch_LCD_demo
   ```

   结果：

- 在2.4英寸LCD显示器上：
  - 首先，以下带有赛昉科技徽标的图片将会显示两秒钟：

图 3-1 示例输出



  - 然后依次显示以下两个官方示例图：

图 3-2 示例输出

- 终端输出如下：

```
-----------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
```

该输出表示:

- SPI模式设置成功
- SPI模式
- 上述三张图片显示的日期和时间

4. 演示源代码

本演示中的资源代码仅作为参考。

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.png")
    else:
        return

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

    """add the part of displaying pictures circularly"""
    while True:
        try:
            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)
            time.sleep(0.5)

            image = Image.open("./LCD_2inch.jpg")
            disp.lcd_ShowImage(image, 0, 0)
            time.sleep(0.5)

            if vf_t == 1:
                image = Image.open("./visionfive.bmp")
            elif vf_t == 2:
```

```
                image = Image.open("./visionfive2.png")
            else:
                return
            disp.lcd_ShowImage(image, 0, 0)
            time.sleep(0.5)

        except KeyboardInterrupt:
            break

    print("Exit demo!")


if __name__ ==
```

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):
        gpio.setmode(gpio.BOARD)

        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(500000, 0, 8)
        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 2.4inch LCD module"""

    def lcd_init_2inch4(self):
        self.lcd_reset()

        self.lcd_sendcmd(0x11)  # sleep out

        self.lcd_sendcmd(0xCF)  # Ppower 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)  # 3Gamma 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)

        """modify the original single byte write to multi byte write"""
        # for i in range(0,len(_buffer)):
        #    self.lcd_spisend(_buffer[i])
        self.lcd_sendnbytes(_buffer)

    def lcd_ShowImage(self, Image, Xstart, Ystart):
        """Set buffer to value of Python Imaging Library image."""
        """Write display buffer to 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)
            # RGB888 >> RGB565
            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)

            """modify the original single byte write to multi byte write"""
            # for i in range(0,len(pix),1):
            #   self.lcd_spisend(pix[i])
            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)

            """modify the original single byte write to multi byte write"""
            # for i in range(0,len(pix)):
            #    self.lcd_spisend(pix[i])
            self.lcd_sendnbytes(pix)
```