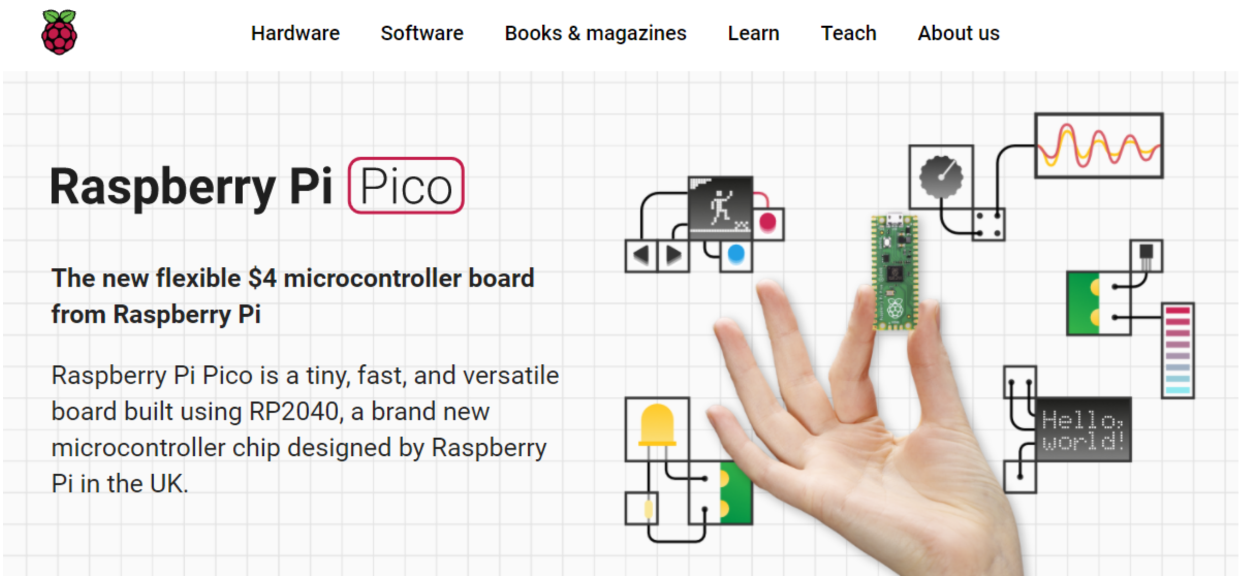
Raspberry Pi Pico 扩展板 ST7789 SPI IPS 240x240 LCD使用教程

作者：骑驴玩儿漂移 AKA 漂移菌 2021-03-11 0:32:15

操作步骤：

1. 首先将树莓派Pico焊接好排针，然后像下图扣入硬禾学堂的Pico扩展板
2. 去树莓派官方网站下载microPython的uf2文件：https://www.raspberrypi.org/products/raspberry-pi-pico/

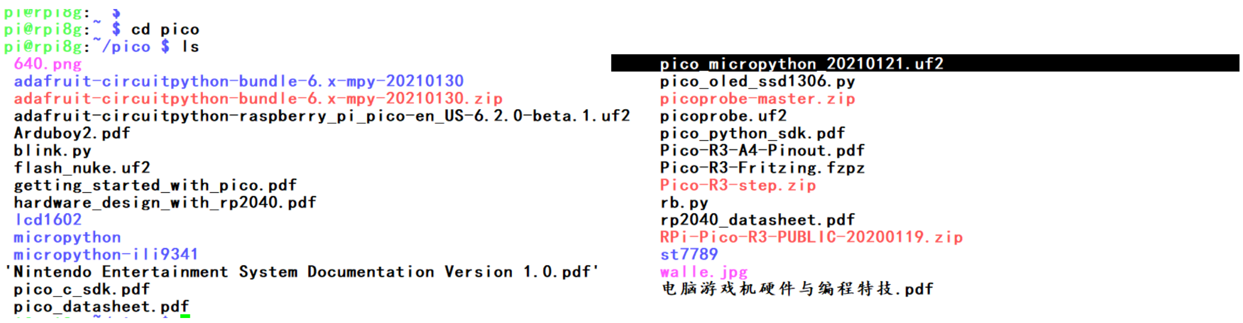


Raspberry Pi Pico

The new flexible $4 microcontroller board from Raspberry Pi

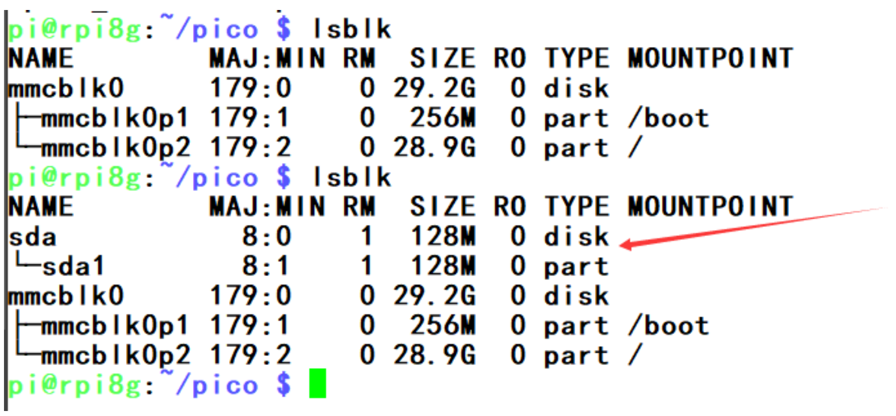
Raspberry Pi Pico is a tiny, fast, and versatile board built using RP2040, a brand new microcontroller chip designed by Raspberry Pi in the UK.

3. 我通过在树莓派上构建了一个pico的目录将所有需要用到的文件都丢进去了



```
pi@rpi8g:~ $ cd pico
pi@rpi8g:~/pico $ ls
 640.png                                                        pico_micropython_20210121.uf2
 adafruit-circuitpython-bundle-6.x-mpy-20210130                 pico_oled_ssd1306.py
 adafruit-circuitpython-bundle-6.x-mpy-20210130.zip             picoprobe-master.zip
 adafruit-circuitpython-raspberry_pi_pico-en_US-6.2.0-beta.1.uf2 picoprobe.uf2
 Arduboy2.pdf                                                   pico_python_sdk.pdf
 blink.py                                                       Pico-R3-A4-Pinout.pdf
 flash_nuke.uf2                                                 Pico-R3-Fritzing.fzpz
 getting_started_with_pico.pdf                                  Pico-R3-step.zip
 hardware_design_with_rp2040.pdf                                rb.py
 lcd1602                                                        rp2040_datasheet.pdf
 micropython                                                    RPi-Pico-R3-PUBLIC-20200119.zip
 micropython-ili9341                                            st7789
'Nintendo Entertainment System Documentation Version 1.0.pdf'   walle.jpg
 pico_c_sdk.pdf                                                 电脑游戏机硬件与编程特技.pdf
 pico_datasheet.pdf
```

4. 按住Pico上的BOOTSEL 键，然后插入树莓派。
   树莓派会识别出一个sda设备：



```
pi@rpi8g:~/pico $ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 29.2G  0 disk
├─mmcblk0p1 179:1    0  256M  0 part /boot
└─mmcblk0p2 179:2    0 28.9G  0 part /
pi@rpi8g:~/pico $ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    1  128M  0 disk
└─sda1        8:1    1  128M  0 part
mmcblk0     179:0    0 29.2G  0 disk
├─mmcblk0p1 179:1    0  256M  0 part /boot
└─mmcblk0p2 179:2    0 28.9G  0 part /
pi@rpi8g:~/pico $
```

5. 通过挂载命令挂载这个硬盘到你的树莓派，然后将micropython的固件拷贝进去即可完成MicroPython的固件的配置。

```
sudo mount /dev/sda1 /mnt
sudo cp  ~/pico/pico_micropython_20210121.uf2  /mnt
sudo sync
sudo umount /mnt
```

6. 然后在树莓派上安装rshell来进行USB串口的接入，非常方便无图形化编程环境，在字符界面就搞定一切的方法，我这里加入了一些编译工具，方便后期用C或者C++开发。

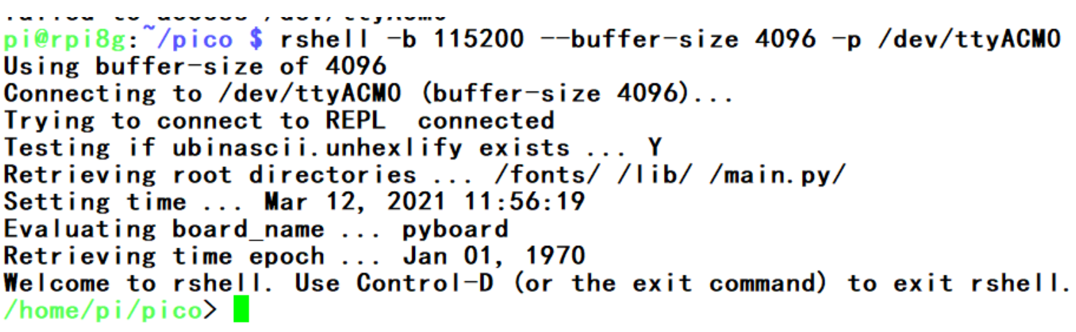
```
sudo apt update
sudo apt -y install  cmake gcc-arm-none-eabi libnewlib-arm
-none-eabi build-essential
sudo apt -y install minicom vim tree
```

然后如果用micropython的环境，可以 使用REPL 进行调试，非常方便。
REPL - Read Evaluate Print Loop 就是一个交互的终端。当Pico插入树莓派USB接口的时候，会被识别为一个`/dev/ttyACM0`的设备，通过串口工具链接这个设备就可以进行调试了
安装rshell就比较方便访问这个串口，所以我用下面命令安装rshell：

```
sudo apt -y install python3-pip
sudo pip3 install rshell
```

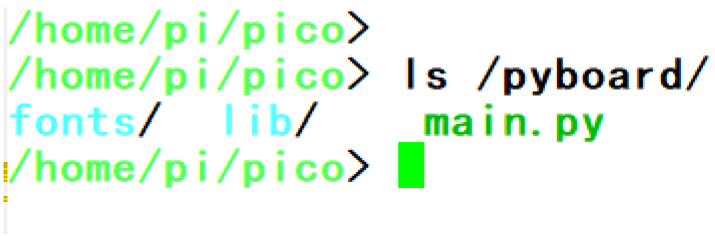
访问使用：

```
rshell -p /dev/ttyACM0 -b 115200 --buffer-size 4096
```



```
pi@rpi8g:~/pico $ rshell -b 115200 --buffer-size 4096 -p /dev/ttyACM0
Using buffer-size of 4096
Connecting to /dev/ttyACM0 (buffer-size 4096)...
Trying to connect to REPL  connected
Testing if ubinascii.unhexlify exists ... Y
Retrieving root directories ... /fonts/ /lib/ /main.py/
Setting time ... Mar 12, 2021 11:56:19
Evaluating board_name ... pyboard
Retrieving time epoch ... Jan 01, 1970
Welcome to rshell. Use Control-D (or the exit command) to exit rshell.
/home/pi/pico>
```

成功后就进入了一个交互的命令界面。
可以通过CTRL+D退出这个rshell，也可以像访问文件夹一样访问pico的/pyboard 目录，这个目录里面就可以放置你的库文件，字体文件和主程序main.py了，这个main.py就是Micropython在设备中自动运行的一个文件，当你把主程序命名成这个名字的时候，单片机一上电就会运行这个文件。
我简单看一下我的pico的目录结构：

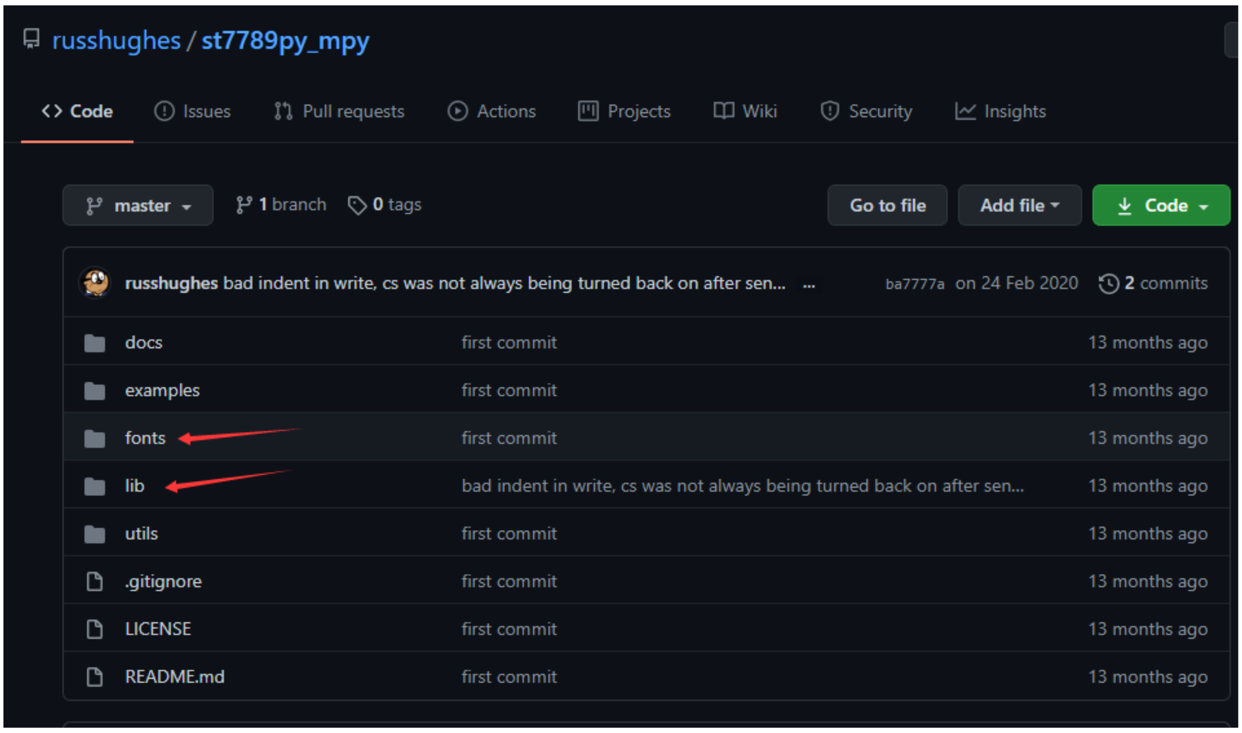


```
/home/pi/pico>
/home/pi/pico> ls /pyboard/
fonts/   lib/      main.py
/home/pi/pico>
```

有fonts，lib，这两个目录是我自己创建的，也可以通过thonny ide生成。

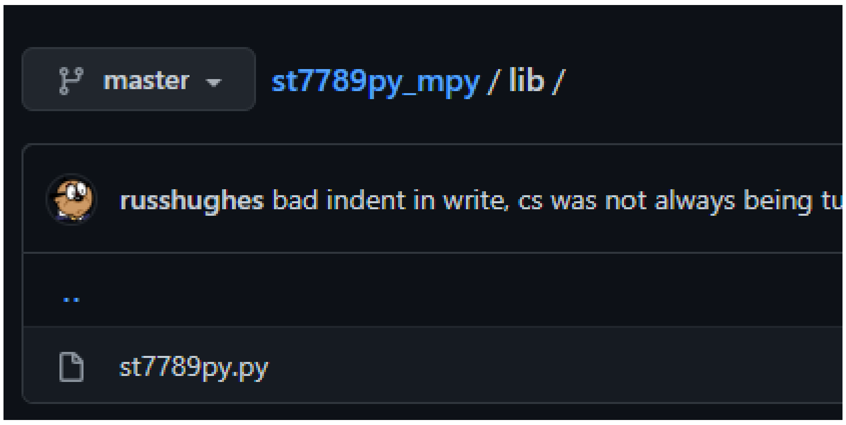
```
mkdir  /pyboard/fonts
mkdir /pyboard/lib
```

为了驱动ST7789的屏幕，我fork了一个github的仓库：https://github.com/russhughes/st7789py_mpy
他是从devbis' st7789py_mpy module from https://github.com/devbis/st7789py_mpy. fork 过来的。
但是他的程序里面有bug，需要自己改一下。
我们只需要从这里面弄到lib和fonts文件即可。


russhughes / st7789py_mpy

russhughes bad indent in write, cs was not always being turned back on after sen... ba7777a on 24 Feb 2020 2 commits

docs	first commit	13 months ago
examples	first commit	13 months ago
fonts	first commit	13 months ago
lib	bad indent in write, cs was not always being turned back on after sen...	13 months ago
utils	first commit	13 months ago
.gitignore	first commit	13 months ago
LICENSE	first commit	13 months ago
README.md	first commit	13 months ago


master st7789py_mpy / lib /

russhughes bad indent in write, cs was not always being tu

..

st7789py.py

如果你访问不到github，下面的就是st7789py.py文件的内容：

```
"""
st7789 tft driver in MicroPython based on devbis' st7789py
_mpy module from
https://github.com/devbis/st7789py_mpy.

I added support for display rotation, scrolling and drawin
g text using 8 and 16
bit wide bitmap fonts with heights that are multiples of
 8.  Included are 12
bitmap fonts derived from classic pc text mode fonts.
"""

import time
from micropython import const
import ustruct as struct
```

```
# commands
ST7789_NOP = const(0x00)
ST7789_SWRESET = const(0x01)
ST7789_RDDID = const(0x04)
ST7789_RDDST = const(0x09)

ST7789_SLPIN = const(0x10)
ST7789_SLPOUT = const(0x11)
ST7789_PTLON = const(0x12)
ST7789_NORON = const(0x13)

ST7789_INVOFF = const(0x20)
ST7789_INVON = const(0x21)
ST7789_DISPOFF = const(0x28)
ST7789_DISPON = const(0x29)
ST7789_CASET = const(0x2A)
ST7789_RASET = const(0x2B)
ST7789_RAMWR = const(0x2C)
ST7789_RAMRD = const(0x2E)

ST7789_PTLAR = const(0x30)
ST7789_VSCRDEF = const(0x33)
ST7789_COLMOD = const(0x3A)
ST7789_MADCTL = const(0x36)
ST7789_VSCSAD = const(0x37)

ST7789_MADCTL_MY = const(0x80)
ST7789_MADCTL_MX = const(0x40)
ST7789_MADCTL_MV = const(0x20)
ST7789_MADCTL_ML = const(0x10)
ST7789_MADCTL_BGR = const(0x08)
ST7789_MADCTL_MH = const(0x04)
ST7789_MADCTL_RGB = const(0x00)

ST7789_RDID1 = const(0xDA)
ST7789_RDID2 = const(0xDB)
ST7789_RDID3 = const(0xDC)
ST7789_RDID4 = const(0xDD)

COLOR_MODE_65K = const(0x50)
```

```
COLOR_MODE_262K = const(0x60)
COLOR_MODE_12BIT = const(0x03)
COLOR_MODE_16BIT = const(0x05)
COLOR_MODE_18BIT = const(0x06)
COLOR_MODE_16M = const(0x07)

# Color definitions
BLACK = const(0x0000)
BLUE = const(0x001F)
RED = const(0xF800)
GREEN = const(0x07E0)
CYAN = const(0x07FF)
MAGENTA = const(0xF81F)
YELLOW = const(0xFFE0)
WHITE = const(0xFFFF)

_ENCODE_PIXEL = ">H"
_ENCODE_POS = ">HH"
_DECODE_PIXEL = ">BBB"

_BUFFER_SIZE = const(256)

_BIT7 = const(0x80)
_BIT6 = const(0x40)
_BIT5 = const(0x20)
_BIT4 = const(0x10)
_BIT3 = const(0x08)
_BIT2 = const(0x04)
_BIT1 = const(0x02)
_BIT0 = const(0x01)

def color565(red, green=0, blue=0):
    """
    Convert red, green and blue values (0-255) into a 16-b
it 565 encoding.
    """
    try:
        red, green, blue = red  # see if the first var is
 a tuple/list
    except TypeError:
        pass
```

```python
    return (red & 0xf8) << 8 | (green & 0xfc) << 3 | blue
 >> 3

def _encode_pos(x, y):
    """Encode a postion into bytes."""
    return struct.pack(_ENCODE_POS, x, y)

def _encode_pixel(color):
    """Encode a pixel color into bytes."""
    return struct.pack(_ENCODE_PIXEL, color)

class ST7789():
    """
    ST7789 driver class

    Args:
        spi (spi): spi object
        width (int): display width
        height (int): display height
        reset (pin): reset pin
        dc (pin): dc pin
        cs (pin): cs pin
        backlight(pin): backlight pin
        xstart (int): display xstart offset
        ystart (int): display ystart offset
        rotation (int): display rotation
    """
    def __init__(self, spi, width, height, reset, dc, cs=N
one, backlight=None,
                 xstart=-1, ystart=-1, rotation=0):
        """
        Initialize display.
        """
        if (width, height) != (240, 240) and (width, heigh
t) != (135, 240):
            raise ValueError(
                "Unsupported display. Only 240x240 and 135
x240 are supported."
            )

        self._display_width = self.width = width
        self._display_height = self.height = height
        self.spi = spi
        self.reset = reset
        self.dc = dc
        self.cs = cs
        self.backlight = backlight
        self._rotation = rotation % 4

        self.hard_reset()
        self.soft_reset()
        self.sleep_mode(False)

        self._set_color_mode(COLOR_MODE_65K|COLOR_MODE_16B
IT)
        time.sleep_ms(50)
        self.rotation(self._rotation)
        self.inversion_mode(True)
        time.sleep_ms(10)
        self.write(ST7789_NORON)
        time.sleep_ms(10)
        if backlight is not None:
            backlight.value(1)
        self.fill(0)
        self.write(ST7789_DISPON)
        time.sleep_ms(500)

    def write(self, command=None, data=None):
        """SPI write to the device: commands and data."""
        if self.cs:
            self.cs.off()

        if command is not None:
            self.dc.off()
            self.spi.write(bytes([command]))
        if data is not None:
            self.dc.on()
            self.spi.write(data)

        if self.cs:
            self.cs.on()
    def hard_reset(self):
        """
        Hard reset display.
        """
        if self.cs:
            self.cs.off()

        if self.reset:
            self.reset.on()
        time.sleep_ms(50)
        if self.reset:
            self.reset.off()
        time.sleep_ms(50)
        if self.reset:
            self.reset.on()
        time.sleep_ms(150)

        if self.cs:
            self.cs.on()

    def soft_reset(self):
        """
        Soft reset display.
        """
        self.write(ST7789_SWRESET)
        time.sleep_ms(150)

    def sleep_mode(self, value):
        """
        Enable or disable display sleep mode.

        Args:
            value (bool): if True enable sleep mode. if Fa
lse disable sleep
            mode
        """
        if value:
            self.write(ST7789_SLPIN)
        else:
            self.write(ST7789_SLPOUT)
    def inversion_mode(self, value):
        """
        Enable or disable display inversion mode.

        Args:
            value (bool): if True enable inversion mode. i
f False disable
            inversion mode
        """
        if value:
            self.write(ST7789_INVON)
        else:
            self.write(ST7789_INVOFF)

    def _set_color_mode(self, mode):
        """
        Set display color mode.

        Args:
            mode (int): color mode
                COLOR_MODE_65K, COLOR_MODE_262K, COLOR_MOD
E_12BIT,
                COLOR_MODE_16BIT, COLOR_MODE_18BIT, COLOR_
MODE_16M
        """
        self.write(ST7789_COLMOD, bytes([mode & 0x77]))

    def rotation(self, rotation):
        """
        Set display rotation.

        Args:
            rotation (int): 0-Portrait, 1-Landscape, 2-Inv
erted Portrait,
            3-Inverted Landscape
        """
        self._rotation = rotation % 4
        if self._rotation == 0:         # Portrait
            madctl = ST7789_MADCTL_RGB
            self.width = self._display_width
            self.height = self._display_height
```

```python
            if self._display_width == 135:
                self.xstart = 52
                self.ystart = 40

        elif self._rotation == 1:       # Landscape
            madctl = ST7789_MADCTL_MX | ST7789_MADCTL_MV | ST7789_MADCTL_RGB
            self.width = self._display_height
            self.height = self._display_width
            if self._display_width == 135:
                self.xstart = 40
                self.ystart = 53

        elif self._rotation == 2:       # Inverted Portrait
            madctl = ST7789_MADCTL_MX | ST7789_MADCTL_MY | ST7789_MADCTL_RGB
            self.width = self._display_width
            self.height = self._display_height
            if self._display_width == 135:
                self.xstart = 53
                self.ystart = 40
        else:                           # Inverted Landscape
            madctl = ST7789_MADCTL_MV | ST7789_MADCTL_MY | ST7789_MADCTL_RGB
            self.width = self._display_height
            self.height = self._display_width
            if self._display_width == 135:
                self.xstart = 40
                self.ystart = 52

        self.write(ST7789_MADCTL, bytes([madctl]))

    def _set_columns(self, start, end):
        """
        Send CASET (column address set) command to display.

        Args:
            start (int): column start address
            end (int): column end address
        """
        if start <= end <= self.width:
            self.write(ST7789_CASET, _encode_pos(
                start+self.xstart, end + self.xstart))

    def _set_rows(self, start, end):
        """
        Send RASET (row address set) command to display.

        Args:
            start (int): row start address
            end (int): row end address
       """
        if start <= end <= self.height:
            self.write(ST7789_RASET, _encode_pos(
                start+self.ystart, end+self.ystart))

    def set_window(self, x0, y0, x1, y1):
        """
        Set window to column and row address.

        Args:
            x0 (int): column start address
            y0 (int): row start address
            x1 (int): column end address
            y1 (int): row end address
        """
        self._set_columns(x0, x1)
        self._set_rows(y0, y1)
        self.write(ST7789_RAMWR)

    def vline(self, x, y, length, color):
        """
        Draw vertical line at the given location and colo
r.

        Args:
            x (int): x coordinate
            Y (int): y coordinate
            length (int): length of line
            color (int): 565 encoded color
        """
        self.fill_rect(x, y, 1, length, color)

    def hline(self, x, y, length, color):
        """
        Draw horizontal line at the given location and col
or.

        Args:
            x (int): x coordinate
            Y (int): y coordinate
            length (int): length of line
            color (int): 565 encoded color
        """
        self.fill_rect(x, y, length, 1, color)

    def pixel(self, x, y, color):
        """
        Draw a pixel at the given location and color.

        Args:
            x (int): x coordinate
            Y (int): y coordinate
            color (int): 565 encoded color
        """
        self.set_window(x, y, x, y)
        self.write(None, _encode_pixel(color))

    def blit_buffer(self, buffer, x, y, width, height):
        """
        Copy buffer to display at the given location.

        Args:
            buffer (bytes): Data to copy to display
            x (int): Top left corner x coordinate
            Y (int): Top left corner y coordinate
            width (int): Width
            height (int): Height
        """
        self.set_window(x, y, x + width - 1, y + height -
1)
        self.write(None, buffer)

    def rect(self, x, y, w, h, color):
        """
        Draw a rectangle at the given location, size and c
olor.

        Args:
            x (int): Top left corner x coordinate
            y (int): Top left corner y coordinate
            width (int): Width in pixels
            height (int): Height in pixels
            color (int): 565 encoded color
        """
        self.hline(x, y, w, color)
        self.vline(x, y, h, color)
        self.vline(x + w - 1, y, h, color)
        self.hline(x, y + h - 1, w, color)

    def fill_rect(self, x, y, width, height, color):
        """
        Draw a rectangle at the given location, size and f
illed with color.

        Args:
            x (int): Top left corner x coordinate
            y (int): Top left corner y coordinate
            width (int): Width in pixels
            height (int): Height in pixels
            color (int): 565 encoded color
        """
        self.set_window(x, y, x + width - 1, y + height -
1)
        chunks, rest = divmod(width * height, _BUFFER_SIZ
E)
        pixel = _encode_pixel(color)
        self.dc.on()
        if chunks:
            data = pixel * _BUFFER_SIZE
            for _ in range(chunks):
                self.write(None, data)
        if rest:
            self.write(None, pixel * rest)

    def fill(self, color):
        """
        Fill the entire FrameBuffer with the specified col
or.

        Args:
            color (int): 565 encoded color
        """
        self.fill_rect(0, 0, self.width, self.height, colo
r)

    def line(self, x0, y0, x1, y1, color):
        """
        Draw a single pixel wide line starting at x0, y0 a
nd ending at x1, y1.

        Args:
            x0 (int): Start point x coordinate
            y0 (int): Start point y coordinate
            x1 (int): End point x coordinate
            y1 (int): End point y coordinate
            color (int): 565 encoded color
        """
        steep = abs(y1 - y0) > abs(x1 - x0)
        if steep:
            x0, y0 = y0, x0
            x1, y1 = y1, x1
        if x0 > x1:
            x0, x1 = x1, x0
            y0, y1 = y1, y0
        dx = x1 - x0
        dy = abs(y1 - y0)
        err = dx // 2
        if y0 < y1:
            ystep = 1
        else:
            ystep = -1
        while x0 <= x1:
            if steep:
                self.pixel(y0, x0, color)
            else:
                self.pixel(x0, y0, color)
            err -= dy
            if err < 0:
                y0 += ystep
                err += dx
            x0 += 1

    def vscrdef(self, tfa, vsa, bfa):
        """
        Set Vertical Scrolling Definition.

        To scroll a 135x240 display these values should be
 40, 240, 40.
        There are 40 lines above the display that are not
 shown followed by
        240 lines that are shown followed by 40 more lines
 that are not shown.
        You could write to these areas off display and scr
oll them into view by
        changing the TFA, VSA and BFA values.

        Args:
            tfa (int): Top Fixed Area
            vsa (int): Vertical Scrolling Area
            bfa (int): Bottom Fixed Area
        """
        struct.pack(">HHH", tfa, vsa, bfa)
        self.write(ST7789_VSCRDEF, struct.pack(">HHH", tf
a, vsa, bfa))

    def vscsad(self, vssa):
        """
        Set Vertical Scroll Start Address of RAM.

        Defines which line in the Frame Memory will be wri
tten as the first
```

```
        line after the last line of the Top Fixed Area on
 the display

        Example:

            for line in range(40, 280, 1):
                tft.vscsad(line)
                utime.sleep(0.01)

        Args:
            vssa (int): Vertical Scrolling Start Address

        """
        self.write(ST7789_VSCSAD, struct.pack(">H", vssa))

    def _text8(self, font, text, x0, y0, color=WHITE, back
ground=BLACK):
        """
        Internal method to write characters with width of
 8 and
        heights of 8 or 16.

        Args:
            font (module): font module to use
            text (str): text to write
            x0 (int): column to start drawing at
            y0 (int): row to start drawing at
            color (int): 565 encoded color to use for char
acters
            background (int): 565 encoded color to use for
 background
        """
        for char in text:
            ch = ord(char)
            if (font.FIRST <= ch < font.LAST
                    and x0+font.WIDTH <= self.width
                    and y0+font.HEIGHT <= self.height):

                if font.HEIGHT == 8:
                    passes = 1
                    size = 8
                    each = 0
                else:
                    passes = 2
                    size = 16
                    each = 8

                for line in range(passes):
                    idx = (ch-font.FIRST)*size+(each*line)
                    buffer = struct.pack('>64H',
                        color if font.FONT[idx] & _BIT7 el
se background,
                        color if font.FONT[idx] & _BIT6 el
se background,
                        color if font.FONT[idx] & _BIT5 el
se background,
                        color if font.FONT[idx] & _BIT4 el
se background,
                        color if font.FONT[idx] & _BIT3 el
se background,
                        color if font.FONT[idx] & _BIT2 el
se background,
                        color if font.FONT[idx] & _BIT1 el
se background,
                        color if font.FONT[idx] & _BIT0 el
se background,
                        color if font.FONT[idx+1] & _BIT7
else background,
                        color if font.FONT[idx+1] & _BIT6
else background,
                        color if font.FONT[idx+1] & _BIT5
else background,
                        color if font.FONT[idx+1] & _BIT4
else background,
                        color if font.FONT[idx+1] & _BIT3
else background,
                        color if font.FONT[idx+1] & _BIT2
else background,
                        color if font.FONT[idx+1] & _BIT1
else background,
                        color if font.FONT[idx+1] & _BIT0
else background,
                                            color if font.FONT[idx+2] & _BIT7
    else background,
                                            color if font.FONT[idx+2] & _BIT6
    else background,
                                            color if font.FONT[idx+2] & _BIT5
    else background,
                                            color if font.FONT[idx+2] & _BIT4
    else background,
                                            color if font.FONT[idx+2] & _BIT3
    else background,
                                            color if font.FONT[idx+2] & _BIT2
    else background,
                                            color if font.FONT[idx+2] & _BIT1
    else background,
                                            color if font.FONT[idx+2] & _BIT0
    else background,
                                            color if font.FONT[idx+3] & _BIT7
    else background,
                                            color if font.FONT[idx+3] & _BIT6
    else background,
                                            color if font.FONT[idx+3] & _BIT5
    else background,
                                            color if font.FONT[idx+3] & _BIT4
    else background,
                                            color if font.FONT[idx+3] & _BIT3
    else background,
                                            color if font.FONT[idx+3] & _BIT2
    else background,
                                            color if font.FONT[idx+3] & _BIT1
    else background,
                                            color if font.FONT[idx+3] & _BIT0
    else background,
                                            color if font.FONT[idx+4] & _BIT7
    else background,
                                            color if font.FONT[idx+4] & _BIT6
    else background,
                                            color if font.FONT[idx+4] & _BIT5
    else background,
                                            color if font.FONT[idx+4] & _BIT4
    else background,
                                color if font.FONT[idx+4] & _BIT3
else background,
                                color if font.FONT[idx+4] & _BIT2
else background,
                                color if font.FONT[idx+4] & _BIT1
else background,
                                color if font.FONT[idx+4] & _BIT0
else background,
                                color if font.FONT[idx+5] & _BIT7
else background,
                                color if font.FONT[idx+5] & _BIT6
else background,
                                color if font.FONT[idx+5] & _BIT5
else background,
                                color if font.FONT[idx+5] & _BIT4
else background,
                                color if font.FONT[idx+5] & _BIT3
else background,
                                color if font.FONT[idx+5] & _BIT2
else background,
                                color if font.FONT[idx+5] & _BIT1
else background,
                                color if font.FONT[idx+5] & _BIT0
else background,
                                color if font.FONT[idx+6] & _BIT7
else background,
                                color if font.FONT[idx+6] & _BIT6
else background,
                                color if font.FONT[idx+6] & _BIT5
else background,
                                color if font.FONT[idx+6] & _BIT4
else background,
                                color if font.FONT[idx+6] & _BIT3
else background,
                                color if font.FONT[idx+6] & _BIT2
else background,
                                color if font.FONT[idx+6] & _BIT1
else background,
                                color if font.FONT[idx+6] & _BIT0
else background,
                        color if font.FONT[idx+7] & _BIT7
else background,
                        color if font.FONT[idx+7] & _BIT6
else background,
                        color if font.FONT[idx+7] & _BIT5
else background,
                        color if font.FONT[idx+7] & _BIT4
else background,
                        color if font.FONT[idx+7] & _BIT3
else background,
                        color if font.FONT[idx+7] & _BIT2
else background,
                        color if font.FONT[idx+7] & _BIT1
else background,
                        color if font.FONT[idx+7] & _BIT0
else background
                    )
                    self.blit_buffer(buffer, x0, y0+8*lin
e, 8, 8)

                x0 += 8

    def _text16(self, font, text, x0, y0, color=WHITE, bac
kground=BLACK):
        """
        Internal method to draw characters with width of 1
6 and heights of 16
        or 32.

        Args:
            font (module): font module to use
            text (str): text to write
            x0 (int): column to start drawing at
            y0 (int): row to start drawing at
            color (int): 565 encoded color to use for char
acters
            background (int): 565 encoded color to use for
 background
        """
        for char in text:
            ch = ord(char)
            if (font.FIRST <= ch < font.LAST
                    and x0+font.WIDTH <= self.width
                    and y0+font.HEIGHT <= self.height):

                if font.HEIGHT == 16:
                    passes = 2
                    size = 32
                    each = 16
                else:
                    passes = 4
                    size = 64
                    each = 16

                for line in range(passes):
                    idx = (ch-font.FIRST)*size+(each*line)
                    buffer = struct.pack('>128H',
                        color if font.FONT[idx] & _BIT7 el
se background,
                        color if font.FONT[idx] & _BIT6 el
se background,
                        color if font.FONT[idx] & _BIT5 el
se background,
                        color if font.FONT[idx] & _BIT4 el
se background,
                        color if font.FONT[idx] & _BIT3 el
se background,
                        color if font.FONT[idx] & _BIT2 el
se background,
                        color if font.FONT[idx] & _BIT1 el
se background,
                        color if font.FONT[idx] & _BIT0 el
se background,
                        color if font.FONT[idx+1] & _BIT7
else background,
                        color if font.FONT[idx+1] & _BIT6
else background,
                        color if font.FONT[idx+1] & _BIT5
else background,
                        color if font.FONT[idx+1] & _BIT4
else background,
                          color if font.FONT[idx+1] & _BIT3
else background,
                          color if font.FONT[idx+1] & _BIT2
else background,
                          color if font.FONT[idx+1] & _BIT1
else background,
                          color if font.FONT[idx+1] & _BIT0
else background,
                          color if font.FONT[idx+2] & _BIT7
else background,
                          color if font.FONT[idx+2] & _BIT6
else background,
                          color if font.FONT[idx+2] & _BIT5
else background,
                          color if font.FONT[idx+2] & _BIT4
else background,
                          color if font.FONT[idx+2] & _BIT3
else background,
                          color if font.FONT[idx+2] & _BIT2
else background,
                          color if font.FONT[idx+2] & _BIT1
else background,
                          color if font.FONT[idx+2] & _BIT0
else background,
                          color if font.FONT[idx+3] & _BIT7
else background,
                          color if font.FONT[idx+3] & _BIT6
else background,
                          color if font.FONT[idx+3] & _BIT5
else background,
                          color if font.FONT[idx+3] & _BIT4
else background,
                          color if font.FONT[idx+3] & _BIT3
else background,
                          color if font.FONT[idx+3] & _BIT2
else background,
                          color if font.FONT[idx+3] & _BIT1
else background,
                          color if font.FONT[idx+3] & _BIT0
else background,
                                            color if font.FONT[idx+4] & _BIT7
     else background,
                                            color if font.FONT[idx+4] & _BIT6
     else background,
                                            color if font.FONT[idx+4] & _BIT5
     else background,
                                            color if font.FONT[idx+4] & _BIT4
     else background,
                                            color if font.FONT[idx+4] & _BIT3
     else background,
                                            color if font.FONT[idx+4] & _BIT2
     else background,
                                            color if font.FONT[idx+4] & _BIT1
     else background,
                                            color if font.FONT[idx+4] & _BIT0
     else background,
                                            color if font.FONT[idx+5] & _BIT7
     else background,
                                            color if font.FONT[idx+5] & _BIT6
     else background,
                                            color if font.FONT[idx+5] & _BIT5
     else background,
                                            color if font.FONT[idx+5] & _BIT4
     else background,
                                            color if font.FONT[idx+5] & _BIT3
     else background,
                                            color if font.FONT[idx+5] & _BIT2
     else background,
                                            color if font.FONT[idx+5] & _BIT1
     else background,
                                            color if font.FONT[idx+5] & _BIT0
     else background,
                                            color if font.FONT[idx+6] & _BIT7
     else background,
                                            color if font.FONT[idx+6] & _BIT6
     else background,
                                            color if font.FONT[idx+6] & _BIT5
     else background,
                                            color if font.FONT[idx+6] & _BIT4
     else background,
                                        color if font.FONT[idx+6] & _BIT3
    else background,
                                        color if font.FONT[idx+6] & _BIT2
    else background,
                                        color if font.FONT[idx+6] & _BIT1
    else background,
                                        color if font.FONT[idx+6] & _BIT0
    else background,
                                        color if font.FONT[idx+7] & _BIT7
    else background,
                                        color if font.FONT[idx+7] & _BIT6
    else background,
                                        color if font.FONT[idx+7] & _BIT5
    else background,
                                        color if font.FONT[idx+7] & _BIT4
    else background,
                                        color if font.FONT[idx+7] & _BIT3
    else background,
                                        color if font.FONT[idx+7] & _BIT2
    else background,
                                        color if font.FONT[idx+7] & _BIT1
    else background,
                                        color if font.FONT[idx+7] & _BIT0
    else background,
                                        color if font.FONT[idx+8] & _BIT7
    else background,
                                        color if font.FONT[idx+8] & _BIT6
    else background,
                                        color if font.FONT[idx+8] & _BIT5
    else background,
                                        color if font.FONT[idx+8] & _BIT4
    else background,
                                        color if font.FONT[idx+8] & _BIT3
    else background,
                                        color if font.FONT[idx+8] & _BIT2
    else background,
                                        color if font.FONT[idx+8] & _BIT1
    else background,
                                        color if font.FONT[idx+8] & _BIT0
    else background,
                        color if font.FONT[idx+9] & _BIT7
else background,
                        color if font.FONT[idx+9] & _BIT6
else background,
                        color if font.FONT[idx+9] & _BIT5
else background,
                        color if font.FONT[idx+9] & _BIT4
else background,
                        color if font.FONT[idx+9] & _BIT3
else background,
                        color if font.FONT[idx+9] & _BIT2
else background,
                        color if font.FONT[idx+9] & _BIT1
else background,
                        color if font.FONT[idx+9] & _BIT0
else background,
                        color if font.FONT[idx+10] & _BIT7
 else background,
                        color if font.FONT[idx+10] & _BIT6
 else background,
                        color if font.FONT[idx+10] & _BIT5
 else background,
                        color if font.FONT[idx+10] & _BIT4
 else background,
                        color if font.FONT[idx+10] & _BIT3
 else background,
                        color if font.FONT[idx+10] & _BIT2
 else background,
                        color if font.FONT[idx+10] & _BIT1
 else background,
                        color if font.FONT[idx+10] & _BIT0
 else background,
                        color if font.FONT[idx+11] & _BIT7
 else background,
                        color if font.FONT[idx+11] & _BIT6
 else background,
                        color if font.FONT[idx+11] & _BIT5
 else background,
                        color if font.FONT[idx+11] & _BIT4
 else background,
                                    color if font.FONT[idx+11] & _BIT3
    else background,
                                    color if font.FONT[idx+11] & _BIT2
    else background,
                                    color if font.FONT[idx+11] & _BIT1
    else background,
                                    color if font.FONT[idx+11] & _BIT0
    else background,
                                    color if font.FONT[idx+12] & _BIT7
    else background,
                                    color if font.FONT[idx+12] & _BIT6
    else background,
                                    color if font.FONT[idx+12] & _BIT5
    else background,
                                    color if font.FONT[idx+12] & _BIT4
    else background,
                                    color if font.FONT[idx+12] & _BIT3
    else background,
                                    color if font.FONT[idx+12] & _BIT2
    else background,
                                    color if font.FONT[idx+12] & _BIT1
    else background,
                                    color if font.FONT[idx+12] & _BIT0
    else background,
                                    color if font.FONT[idx+13] & _BIT7
    else background,
                                    color if font.FONT[idx+13] & _BIT6
    else background,
                                    color if font.FONT[idx+13] & _BIT5
    else background,
                                    color if font.FONT[idx+13] & _BIT4
    else background,
                                    color if font.FONT[idx+13] & _BIT3
    else background,
                                    color if font.FONT[idx+13] & _BIT2
    else background,
                                    color if font.FONT[idx+13] & _BIT1
    else background,
                                    color if font.FONT[idx+13] & _BIT0
    else background,
                        color if font.FONT[idx+14] & _BIT7
 else background,
                        color if font.FONT[idx+14] & _BIT6
 else background,
                        color if font.FONT[idx+14] & _BIT5
 else background,
                        color if font.FONT[idx+14] & _BIT4
 else background,
                        color if font.FONT[idx+14] & _BIT3
 else background,
                        color if font.FONT[idx+14] & _BIT2
 else background,
                        color if font.FONT[idx+14] & _BIT1
 else background,
                        color if font.FONT[idx+14] & _BIT0
 else background,
                        color if font.FONT[idx+15] & _BIT7
 else background,
                        color if font.FONT[idx+15] & _BIT6
 else background,
                        color if font.FONT[idx+15] & _BIT5
 else background,
                        color if font.FONT[idx+15] & _BIT4
 else background,
                        color if font.FONT[idx+15] & _BIT3
 else background,
                        color if font.FONT[idx+15] & _BIT2
 else background,
                        color if font.FONT[idx+15] & _BIT1
 else background,
                        color if font.FONT[idx+15] & _BIT0
 else background
                    )
                    self.blit_buffer(buffer, x0, y0+8*lin
e, 16, 8)
            x0 += font.WIDTH

    def text(self, font, text, x0, y0, color=WHITE, backgr
ound=BLACK):
        """
        Draw text on display in specified font and colors.
 8 and 16 bit wide
        fonts are supported.

        Args:
            font (module): font module to use.
            text (str): text to write
            x0 (int): column to start drawing at
            y0 (int): row to start drawing at
            color (int): 565 encoded color to use for char
acters
            background (int): 565 encoded color to use for
 background
        """
        if font.WIDTH == 8:
            self._text8(font, text, x0, y0, color, backgro
und)
        else:
            self._text16(font, text, x0, y0, color, backgr
ound)
```

但是他的初始化配置里面缺少了两行：

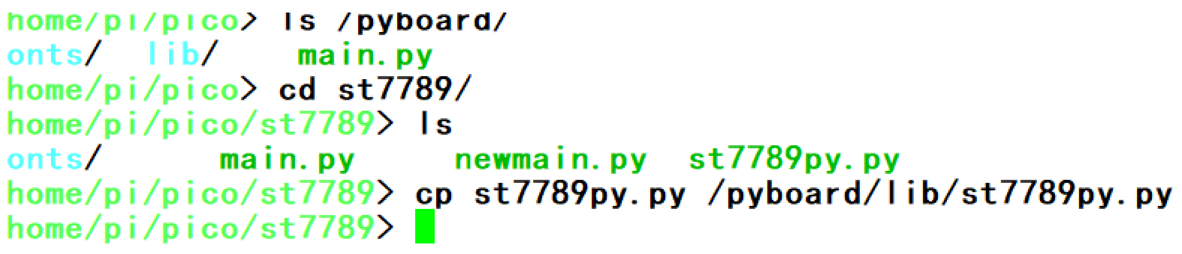
```
def __init__(self, spi, width, height, reset, dc, cs=None, backlight=None,
             xstart=-1, ystart=-1, rotation=0):
    """
    Initialize display.
    """
    if (width, height) != (240, 240) and (width, height) != (135, 240):
        raise ValueError(
            "Unsupported display. Only 240x240 and 135x240 are supported."
        )

    self._display_width = self.width = width
    self._display_height = self.height = height
    self.spi = spi
    self.reset = reset
    self.dc = dc
    self.cs = cs
    self.backlight = backlight
    self._rotation = rotation % 4

    self.hard_reset()
    self.soft_reset()
    self.sleep_mode(False)
```

在def **init** 函数的self.hard_reset()上方加入：

```
self.ystart = ystart
self.xstart = xstart
```

然后保存的时候请把名字保存成：st7789py.py 然后拷贝到pico的/pyboard/lib/st7789py.py中。

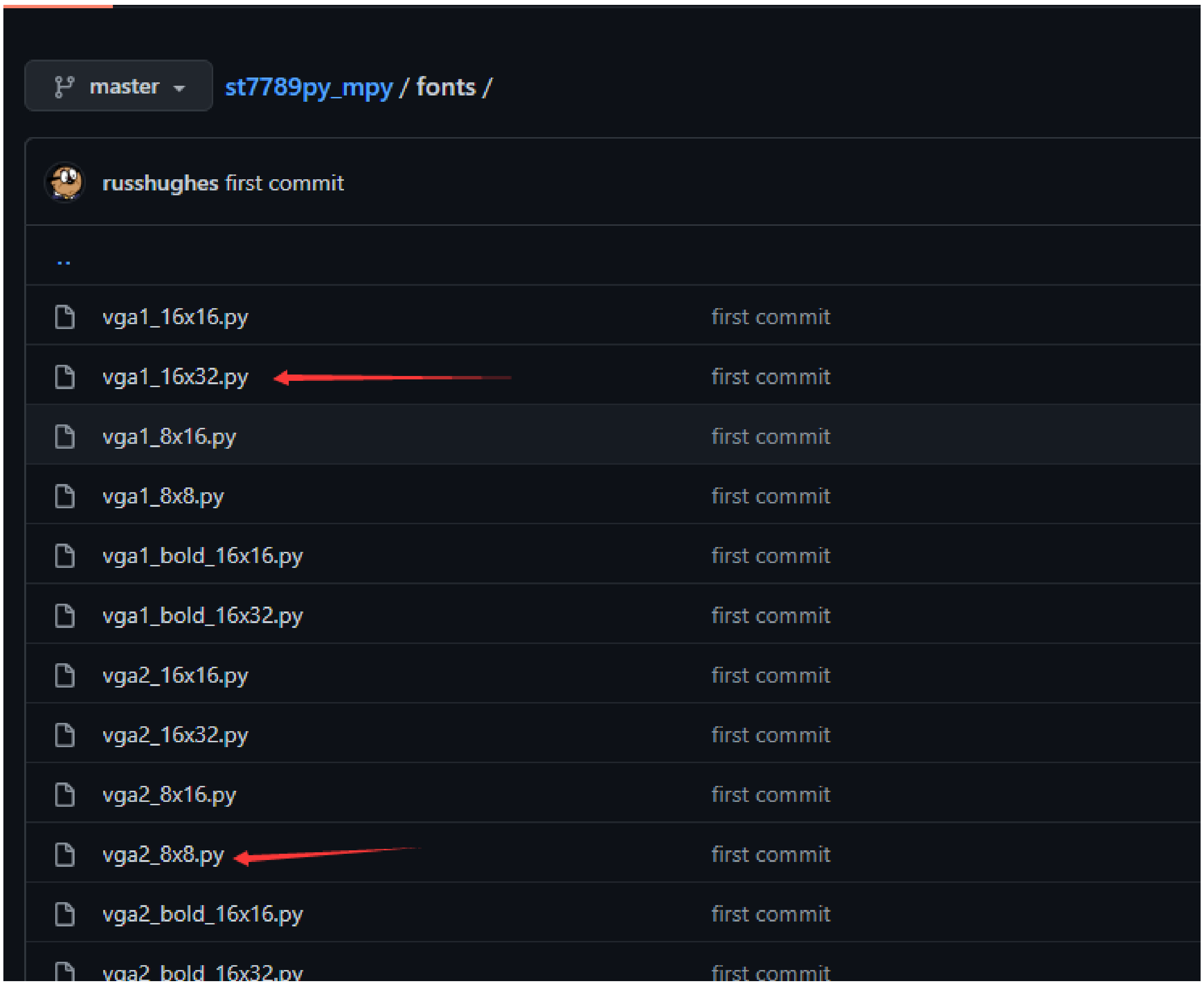
```
cp st7789py.py /pyboard/lib/st7789py.py
```



```
home/pi/pico> ls /pyboard/
onts/   lib/     main.py
home/pi/pico> cd st7789/
home/pi/pico/st7789> ls
onts/        main.py       newmain.py   st7789py.py
home/pi/pico/st7789> cp st7789py.py /pyboard/lib/st7789py.py
home/pi/pico/st7789> 
```

字体选择

接下来要在fonts里面创建两个文件，就是做好的字体文件，看喜欢什么类型字体就换什么字体文件，在GitHub的fonts目录里面：



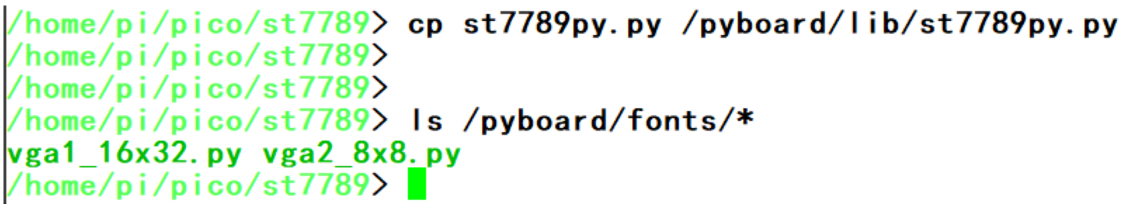
master st7789py_mpy / fonts /

russhughes first commit

..

vga1_16x16.py	first commit
vga1_16x32.py	first commit
vga1_8x16.py	first commit
vga1_8x8.py	first commit
vga1_bold_16x16.py	first commit
vga1_bold_16x32.py	first commit
vga2_16x16.py	first commit
vga2_16x32.py	first commit
vga2_8x16.py	first commit
vga2_8x8.py	first commit
vga2_bold_16x16.py	first commit
vga2_bold_16x32.py	first commit

我选择了这两个：
然后点开文件选择raw按钮，然后复制，粘贴到对应的文件名即可，都放在了fonts/目录里面



```
/home/pi/pico/st7789> cp st7789py.py /pyboard/lib/st7789py.py
/home/pi/pico/st7789>
/home/pi/pico/st7789>
/home/pi/pico/st7789> ls /pyboard/fonts/*
vga1_16x32.py vga2_8x8.py
/home/pi/pico/st7789>
```

vga1_16x32.py 的内容：

```
WIDTH = 16
HEIGHT = 32
FIRST = 0x20
LAST = 0x7f
```

```
_FONT = \
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00
\x00\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38
\x1c\x38\x1c\x38\x1c\x38\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1c\x38
\x1c\x38\x1c\x38\x1c\x38\x7f\xfe\x7f\xfe\x1c\x38\x1c\x38\x
1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x7f\xfe\x7f
\xfe\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x01\x80\x01\x80\x0f\xf0\x0f\xf0\x39\x9c
\x39\x9c\x71\x8e\x71\x8e\x71\x80\x71\x80\x39\x80\x39\x80\x
0f\xf0\x0f\xf0\x01\x9c\x01\x9c\x01\x8e\x01\x8e\x71\x8e\x71
\x8e\x39\x9c\x39\x9c\x0f\xf0\x0f\xf0\x01\x80\x01\x80\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x1e\x1c\x1e\x1c\x1e\x38\x1e\x38\x00\x70\x00\x70\x
00\xe0\x00\xe0\x01\xc0\x01\xc0\x03\x80\x03\x80\x07\x00\x07
\x00\x0e\x3c\x0e\x3c\x1c\x3c\x1c\x3c\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xc0\x07\xc0\x1c\x70
\x1c\x70\x38\x38\x38\x38\x1c\x70\x1c\x70\x07\xc0\x07\xc0\x
0f\xce\x0f\xce\x38\xfc\x38\xfc\x70\x78\x70\x78\x70\x78\x70
\x78\x38\xfc\x38\xfc\x0f\xce\x0f\xce\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\xe0\x00\xe0\x01\xc0\x01\xc0\x03\x80
\x03\x80\x07\x00\x07\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe0\x00\xe0\x01\xc0
\x01\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x07\x00\x07\x00\x
07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x03\x80\x03
\x80\x01\xc0\x01\xc0\x00\xe0\x00\xe0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\x00\x07\x00\x03\x80
\x03\x80\x01\xc0\x01\xc0\x00\xe0\x00\xe0\x00\xe0\x00\xe0\x
00\xe0\x00\xe0\x00\xe0\x00\xe0\x00\xe0\x00\xe0\x01\xc0\x01
\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x0e\x38\x0e\x38\x03\xe0\x03\xe0\x
3f\xfe\x3f\xfe\x03\xe0\x03\xe0\x0e\x38\x0e\x38\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
3f\xfe\x3f\xfe\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x03\xc0\x03\xc0\x03\xc0\x03\xc0\x03\x80\x03\x80\x07\x
00\x07\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
3f\xfe\x3f\xfe\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x03\xc0\x03\xc0\x03\xc0\x03\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1c
\x00\x1c\x00\x38\x00\x38\x00\x70\x00\x70\x00\xe0\x00\xe0\x
01\xc0\x01\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x0e\x00\x0e
\x00\x1c\x00\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xe0\x07\xe0\x1c\x38
\x1c\x38\x38\x3c\x38\x3c\x38\x7c\x38\x7c\x38\xdc\x38\xdc\x
39\x9c\x39\x9c\x3b\x1c\x3b\x1c\x3e\x1c\x3e\x1c\x3c\x1c\x3c
\x1c\x1c\x38\x1c\x38\x07\xe0\x07\xe0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x03\xc0
\x03\xc0\x0f\xc0\x0f\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf0\x0f\xf0\x38\x1c
\x38\x1c\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x1c\x00\x1c\x
00\x70\x00\x70\x01\xc0\x01\xc0\x07\x00\x07\x00\x1c\x00\x1c
\x00\x38\x00\x38\x00\x3f\xfe\x3f\xfe\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf0\x0f\xf0\x38\x1c
\x38\x1c\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x1c\x00\x1c\x
01\xf0\x01\xf0\x00\x1c\x00\x1c\x00\x0e\x00\x0e\x00\x0e\x00
\x0e\x38\x1c\x38\x1c\x0f\xf0\x0f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xf0\x01\xf0\x03\xf0
\x03\xf0\x07\x70\x07\x70\x0e\x70\x0e\x70\x1c\x70\x1c\x70\x
38\x70\x38\x70\x3f\xfc\x3f\xfc\x00\x70\x00\x70\x00\x70\x00
\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfe\x3f\xfe\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x3f\xf0\x3f\xf0\x
00\x1c\x00\x1c\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00
\x0e\x38\x1c\x38\x1c\x0f\xf0\x0f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x00
\x1c\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x
3f\xf0\x3f\xf0\x38\x1c\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf8\x3f\xf8\x00\x38
\x00\x38\x00\x38\x00\x38\x00\x70\x00\x70\x00\xe0\x00\xe0\x
01\xc0\x01\xc0\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03
\x80\x03\x80\x03\x80\x03\x80\x03\x80\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x1c\x1c\x1c\x1c\x
07\xf0\x07\xf0\x1c\x1c\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x1c\x0e\x1c\x0e\x
07\xfe\x07\xfe\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00
\x0e\x00\x1c\x00\x1c\x0f\xf0\x0f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x03\x80\x03\x80\x03\x80\x03\x80\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x80\x03
\x80\x03\x80\x03\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x03\x80\x03\x80\x03\x80\x03\x80\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x80\x03
\x80\x03\x80\x03\x80\x07\x00\x07\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe0\x00\xe0\x01\xc0
\x01\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x0e\x00\x0e\x00\x
1c\x00\x1c\x00\x0e\x00\x0e\x00\x07\x00\x07\x00\x03\x80\x03
\x80\x01\xc0\x01\xc0\x00\xe0\x00\xe0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfc\x3f\xfc\x
00\x00\x00\x00\x00\x00\x00\x00\x3f\xfc\x3f\xfc\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\x00\x07\x00\x03\x80
\x03\x80\x01\xc0\x01\xc0\x00\xe0\x00\xe0\x00\x70\x00\x70\x
00\x38\x00\x38\x00\x70\x00\x70\x00\xe0\x00\xe0\x01\xc0\x01
\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xe0\x07\xe0\x1c\x38
\x1c\x38\x38\x1c\x38\x1c\x00\x38\x00\x38\x00\x70\x00\x70\x
00\xe0\x00\xe0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00
\x00\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf0\x0f\xf0\x38\x1c
\x38\x1c\x70\x0e\x70\x0e\x71\xfe\x71\xfe\x73\x8e\x73\x8e\x
73\x8e\x73\x8e\x73\x8e\x73\x8e\x71\xfc\x71\xfc\x70\x00\x70
\x00\x38\x00\x38\x00\x0f\xfc\x0f\xfc\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x03\xc0\x03\xc0\x07\xe0
\x07\xe0\x0e\x70\x0e\x70\x1c\x38\x1c\x38\x38\x1c\x38\x1c\x
38\x1c\x38\x1c\x3f\xfc\x3f\xfc\x38\x1c\x38\x1c\x38\x1c\x38
\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x38\x1c
\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x1c\x38\x1c\x
3f\xf0\x3f\xf0\x38\x1c\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x1c\x38\x1c\x3f\xf0\x3f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x00\x38\x00\x38\x00\x38\x00\x
38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x38\x1c
\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x1c\x38\x1c\x3f\xf0\x3f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfc\x3f\xfc\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x
3f\xe0\x3f\xe0\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x00\x38\x00\x3f\xfc\x3f\xfc\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfc\x3f\xfc\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x
3f\xe0\x3f\xe0\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x00\x38\x00\x38\x00\x38\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x00\x38\x00\x38\x00\x38\x00\x
38\x00\x38\x00\x38\x3e\x38\x3e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x38\x0e
\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x
3f\xfe\x3f\xfe\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1c\x00\x1c\x00\x1c
\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x
00\x1c\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x00\x1c\x1c\x1c\x1c
\x1c\x0e\x38\x0e\x38\x03\xe0\x03\xe0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x1c\x38\x1c\x38\x1c\x70
\x1c\x70\x1c\xe0\x1c\xe0\x1d\xc0\x1d\xc0\x1f\x80\x1f\x80\x
1f\x80\x1f\x80\x1d\xc0\x1d\xc0\x1c\xe0\x1c\xe0\x1c\x70\x1c
\x70\x1c\x38\x1c\x38\x1c\x1c\x1c\x1c\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x00\x38\x00\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x
38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x00\x38\x00\x3f\xfc\x3f\xfc\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x78\x1e\x78\x1e\x7c\x3e
\x7c\x3e\x7e\x7e\x7e\x7e\x77\xee\x77\xee\x73\xce\x73\xce\x
71\x8e\x71\x8e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x70
\x0e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x3c\x0e
\x3c\x0e\x3e\x0e\x3e\x0e\x3f\x0e\x3f\x0e\x3b\x8e\x3b\x8e\x
39\xce\x39\xce\x38\xee\x38\xee\x38\x7e\x38\x7e\x38\x3e\x38
\x3e\x38\x1e\x38\x1e\x38\x0e\x38\x0e\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x38\x1c
\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x1c\x38\x1c\x
3f\xf0\x3f\xf0\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x00\x38\x00\x38\x00\x38\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x1c\x1c
\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\xee\x38
\xee\x1c\x7c\x1c\x7c\x07\xf8\x07\xf8\x00\x1c\x00\x1c\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x38\x1c
\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x1c\x38\x1c\x
3f\xf0\x3f\xf0\x38\xe0\x38\xe0\x38\x70\x38\x70\x38\x38\x38
\x38\x38\x1c\x38\x1c\x38\x0e\x38\x0e\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf0\x0f\xf0\x38\x1c
\x38\x1c\x70\x0e\x70\x0e\x70\x00\x70\x00\x38\x00\x38\x00\x
0f\xf0\x0f\xf0\x00\x1c\x00\x1c\x00\x0e\x00\x0e\x70\x0e\x70
\x0e\x38\x1c\x38\x1c\x0f\xf0\x0f\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfe\x3f\xfe\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x38\x0e
\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x1c\x38\x1c\x38\x1c
\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x
38\x1c\x38\x1c\x38\x1c\x38\x1c\x1c\x38\x1c\x38\x0e\x70\x0e
\x70\x07\xe0\x07\xe0\x03\xc0\x03\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x70\x0e\x70\x0e\x70\x0e
\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x70\x0e\x
70\x0e\x70\x0e\x71\x8e\x71\x8e\x73\xce\x73\xce\x77\xee\x77
\xee\x3e\x7c\x3e\x7c\x1c\x38\x1c\x38\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x1c\x38\x1c\x38\x1c
\x38\x1c\x1c\x38\x1c\x38\x0e\x70\x0e\x70\x07\xe0\x07\xe0\x
03\xc0\x03\xc0\x07\xe0\x07\xe0\x0e\x70\x0e\x70\x1c\x38\x1c
\x38\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x1c\x1c
\x1c\x1c\x0e\x38\x0e\x38\x07\x70\x07\x70\x03\xe0\x03\xe0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfe\x3f\xfe\x00\x1c
\x00\x1c\x00\x38\x00\x38\x00\x70\x00\x70\x00\xe0\x00\xe0\x
01\xc0\x01\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x0e\x00\x0e
\x00\x1c\x00\x1c\x00\x3f\xfe\x3f\xfe\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x07\x00
\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x
07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07\x00\x07
\x00\x07\x00\x07\x00\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1c\x00
\x1c\x00\x0e\x00\x0e\x00\x07\x00\x07\x00\x03\x80\x03\x80\x
01\xc0\x01\xc0\x00\xe0\x00\xe0\x00\x70\x00\x70\x00\x38\x00
\x38\x00\x1c\x00\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x00\x70
\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x
00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00
\x70\x00\x70\x00\x70\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x03\xc0\x03\xc0\x07\xe0\x07\xe0\x0e\x70\x0e\x70\x1c\x38
\x1c\x38\x38\x1c\x38\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\x
ff\xff\xff\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x07\x00\x07\x00\x03\x80\x03\x80\x01\xc0
\x01\xc0\x00\xe0\x00\xe0\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf8\x0f\xf8\x
00\x0e\x00\x0e\x0f\xfe\x0f\xfe\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x0e\x38\x0e\x0f\xfe\x0f\xfe\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x00\x38\x00\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x3f\xf0\x3f\xf0\x
38\x1c\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x0e\x38\x0e\x3f\xf8\x3f\xf8\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf8\x0f\xf8\x
38\x0e\x38\x0e\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x0e\x38\x0e\x0f\xf8\x0f\xf8\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0e\x00\x0e\x00\x0e
\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x07\xfe\x07\xfe\x
1c\x0e\x1c\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x0e\x38\x0e\x0f\xfe\x0f\xfe\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf8\x0f\xf8\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x3f\xfe\x3f\xfe\x38\x00\x38
\x00\x38\x00\x38\x00\x0f\xfc\x0f\xfc\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf8\x00\xf8\x03\x80
\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x
0f\xf0\x0f\xf0\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03
\x80\x03\x80\x03\x80\x03\x80\x03\x80\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xf8\x0f\xf8\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x0f\xfe\x0f\xfe\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x1f\x
f8\x1f\xf8\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x38\x00\x38\x00\x38\x00
\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x3b\xf8\x3b\xf8\x
3c\x0e\x3c\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x70
\x00\x70\x00\x70\x00\x70\x00\x00\x00\x00\x00\x70\x00\x70\x
00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00
\x70\x00\x70\x00\x70\x00\x70\x00\x70\x00\xe0\x00\xe0\x0f\x
80\x0f\x80\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x0e\x00\x0e\x00\x0e\x00
\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x38\x0e\x38\x
0e\x70\x0e\x70\x0e\xe0\x0e\xe0\x0f\xc0\x0f\xc0\x0e\xe0\x0e
\xe0\x0e\x70\x0e\x70\x0e\x38\x0e\x38\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3e\x78\x3e\x78\x
39\xce\x39\xce\x39\xce\x39\xce\x39\xce\x39\xce\x39\xce\x39
\xce\x39\xce\x39\xce\x39\xce\x39\xce\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xe0\x3f\xe0\x
38\x38\x38\x38\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38
\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xf0\x07\xf0\x
1c\x1c\x1c\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38
\x0e\x1c\x1c\x1c\x1c\x07\xf0\x07\xf0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x
38\x1c\x38\x1c\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x1c\x38
\x1c\x3f\xf0\x3f\xf0\x38\x00\x38\x00\x38\x00\x38\x00\x38\x
00\x38\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xfe\x07\xfe\x
1c\x0e\x1c\x0e\x38\x0e\x38\x0e\x38\x0e\x38\x0e\x1c\x0e\x1c
\x0e\x07\xfe\x07\xfe\x00\x0e\x00\x0e\x00\x0e\x00\x0e\x00\x
0e\x00\x0e\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xf0\x3f\xf0\x
38\x1c\x38\x1c\x38\x00\x38\x00\x38\x00\x38\x00\x38\x00\x38
\x00\x38\x00\x38\x00\x38\x00\x38\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0f\xfc\x0f\xfc\x
38\x00\x38\x00\x38\x00\x38\x00\x0f\xf8\x0f\xf8\x00\x0e\x00
\x0e\x00\x0e\x00\x0e\x1f\xf8\x1f\xf8\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x1f\xfc\x1f\xfc\x
01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x38\x1c\x38\x1c\x
38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38\x1c\x38
\x1c\x38\x1c\x38\x1c\x0f\xfc\x0f\xfc\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x70\x0e\x70\x0e\x
38\x1c\x38\x1c\x1c\x38\x1c\x38\x0e\x70\x0e\x70\x07\xe0\x07
\xe0\x03\xc0\x03\xc0\x01\x80\x01\x80\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x
38\x0e\x38\x0e\x38\x0e\x38\x0e\x39\xce\x39\xce\x3b\xee\x3b
\xee\x1f\x7c\x1f\x7c\x0e\x38\x0e\x38\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1c\x38\x1c\x38\x
0e\x70\x0e\x70\x07\xe0\x07\xe0\x03\xc0\x03\xc0\x07\xe0\x07
\xe0\x0e\x70\x0e\x70\x1c\x38\x1c\x38\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x38\x0e\x38\x0e\x
1c\x1c\x1c\x1c\x0e\x38\x0e\x38\x07\x70\x07\x70\x03\xe0\x03
\xe0\x01\xc0\x01\xc0\x03\x80\x03\x80\x07\x00\x07\x00\x0e\x
00\x0e\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x3f\xfe\x3f\xfe\x
00\x1c\x00\x1c\x00\x70\x00\x70\x01\xc0\x01\xc0\x07\x00\x07
\x00\x1c\x00\x1c\x00\x3f\xfe\x3f\xfe\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf8\x00\xf8\x01\xc0
\x01\xc0\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x
1e\x00\x1e\x00\x03\x80\x03\x80\x03\x80\x03\x80\x03\x80\x03
\x80\x01\xc0\x01\xc0\x00\xf8\x00\xf8\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0
\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
00\x00\x00\x00\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x1f\x00\x1f\x00\x03\x80
\x03\x80\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x
00\x78\x00\x78\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01\xc0\x01
\xc0\x03\x80\x03\x80\x1f\x00\x1f\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x07\x9e\x07\x9e\x3c\xf0
\x3c\xf0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x01\xc0\x01\xc0\x07\x70\x07\x70\x1c\x1c\x1c\x1c\x
70\x07\x70\x07\x70\x07\x70\x07\x7f\xff\x7f\xff\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00'\
FONT = memoryview(_FONT)
```

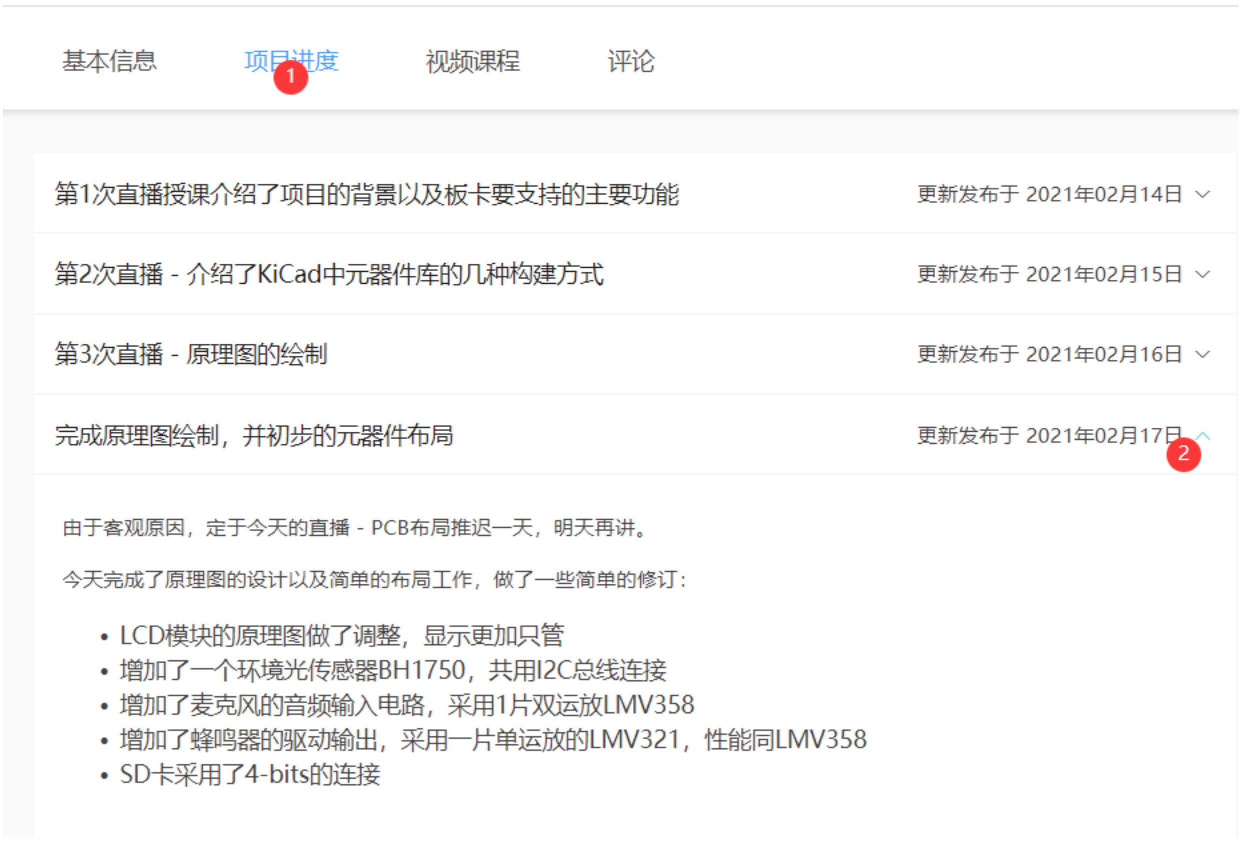
vga2_8x8.py 的内容:

```
"""converted from vga_8x8.bin """
WIDTH = 8
HEIGHT = 8
FIRST = 0x00
LAST = 0xff
_FONT =\
b'\x00\x00\x00\x00\x00\x00\x00\x00'\
b'\x7e\x81\xa5\x81\xbd\x99\x81\x7e'\
b'\x7e\xff\xdb\xff\xc3\xe7\xff\x7e'\
b'\x6c\xfe\xfe\xfe\x7c\x38\x10\x00'\
b'\x10\x38\x7c\xfe\x7c\x38\x10\x00'\
b'\x38\x7c\x38\xfe\xfe\xd6\x10\x38'\
b'\x10\x38\x7c\xfe\xfe\x7c\x10\x38'\
b'\x00\x00\x18\x3c\x3c\x18\x00\x00'\
b'\xff\xff\xe7\xc3\xc3\xe7\xff\xff'\
b'\x00\x3c\x66\x42\x42\x66\x3c\x00'\
b'\xff\xc3\x99\xbd\xbd\x99\xc3\xff'\
b'\x0f\x07\x0f\x7d\xcc\xcc\xcc\x78'\
b'\x3c\x66\x66\x66\x3c\x18\x7e\x18'\
b'\x3f\x33\x3f\x30\x30\x70\xf0\xe0'\
b'\x7f\x63\x7f\x63\x63\x67\xe6\xc0'\
b'\x18\xdb\x3c\xe7\xe7\x3c\xdb\x18'\
b'\x80\xe0\xf8\xfe\xf8\xe0\x80\x00'\
b'\x02\x0e\x3e\xfe\x3e\x0e\x02\x00'\
b'\x18\x3c\x7e\x18\x18\x7e\x3c\x18'\
b'\x66\x66\x66\x66\x66\x00\x66\x00'\
b'\x7f\xdb\xdb\x7b\x1b\x1b\x1b\x00'\
b'\x3e\x61\x3c\x66\x66\x3c\x86\x7c'\
b'\x00\x00\x00\x00\x7e\x7e\x7e\x00'\
b'\x18\x3c\x7e\x18\x7e\x3c\x18\xff'\
b'\x18\x3c\x7e\x18\x18\x18\x18\x00'\
b'\x18\x18\x18\x18\x7e\x3c\x18\x00'\
b'\x00\x18\x0c\xfe\x0c\x18\x00\x00'\
b'\x00\x30\x60\xfe\x60\x30\x00\x00'\
b'\x00\x00\xc0\xc0\xc0\xfe\x00\x00'\
b'\x00\x24\x66\xff\x66\x24\x00\x00'\
b'\x00\x18\x3c\x7e\xff\xff\x00\x00'\
b'\x00\xff\xff\x7e\x3c\x18\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00'\
b'\x18\x3c\x3c\x18\x18\x00\x18\x00'\
b'\x66\x66\x24\x00\x00\x00\x00\x00'\
b'\x6c\x6c\xfe\x6c\xfe\x6c\x6c\x00'\
b'\x18\x3e\x60\x3c\x06\x7c\x18\x00'\
b'\x00\xc6\xcc\x18\x30\x66\xc6\x00'\
b'\x38\x6c\x38\x76\xdc\xcc\x76\x00'\
b'\x18\x18\x30\x00\x00\x00\x00\x00'\
b'\x0c\x18\x30\x30\x30\x18\x0c\x00'\
b'\x30\x18\x0c\x0c\x0c\x18\x30\x00'\
b'\x00\x66\x3c\xff\x3c\x66\x00\x00'\
b'\x00\x18\x18\x7e\x18\x18\x00\x00'\
b'\x00\x00\x00\x00\x00\x18\x18\x30'\
b'\x00\x00\x00\x7e\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x18\x18\x00'\
b'\x06\x0c\x18\x30\x60\xc0\x80\x00'\
b'\x38\x6c\xc6\xd6\xc6\x6c\x38\x00'\
b'\x18\x38\x18\x18\x18\x18\x7e\x00'\
b'\x7c\xc6\x06\x1c\x30\x66\xfe\x00'\
b'\x7c\xc6\x06\x3c\x06\xc6\x7c\x00'\
b'\x1c\x3c\x6c\xcc\xfe\x0c\x1e\x00'\
b'\xfe\xc0\xc0\xfc\x06\xc6\x7c\x00'\
b'\x38\x60\xc0\xfc\xc6\xc6\x7c\x00'\
b'\xfe\xc6\x0c\x18\x30\x30\x30\x00'\
b'\x7c\xc6\xc6\x7c\xc6\xc6\x7c\x00'\
b'\x7c\xc6\xc6\x7e\x06\x0c\x78\x00'\
b'\x00\x18\x18\x00\x00\x18\x18\x00'\
b'\x00\x18\x18\x00\x00\x18\x18\x30'\
b'\x06\x0c\x18\x30\x18\x0c\x06\x00'\
b'\x00\x00\x7e\x00\x00\x7e\x00\x00'\
b'\x60\x30\x18\x0c\x18\x30\x60\x00'\
b'\x7c\xc6\x0c\x18\x18\x00\x18\x00'\
b'\x7c\xc6\xde\xde\xde\xc0\x78\x00'\
b'\x38\x6c\xc6\xfe\xc6\xc6\xc6\x00'\
b'\xfc\x66\x66\x7c\x66\x66\xfc\x00'\
b'\x3c\x66\xc0\xc0\xc0\x66\x3c\x00'\
b'\xf8\x6c\x66\x66\x66\x6c\xf8\x00'\
b'\xfe\x62\x68\x78\x68\x62\xfe\x00'\
b'\xfe\x62\x68\x78\x68\x60\xf0\x00'\
b'\x3c\x66\xc0\xc0\xce\x66\x3a\x00'\
b'\xc6\xc6\xc6\xfe\xc6\xc6\xc6\x00'\
b'\x3c\x18\x18\x18\x18\x18\x3c\x00'\
b'\x1e\x0c\x0c\x0c\xcc\xcc\x78\x00'\
b'\xe6\x66\x6c\x78\x6c\x66\xe6\x00'\
b'\xf0\x60\x60\x60\x62\x66\xfe\x00'\
b'\xc6\xee\xfe\xfe\xd6\xc6\xc6\x00'\
b'\xc6\xe6\xf6\xde\xce\xc6\xc6\x00'\
b'\x7c\xc6\xc6\xc6\xc6\xc6\x7c\x00'\
b'\xfc\x66\x66\x7c\x60\x60\xf0\x00'\
b'\x7c\xc6\xc6\xc6\xc6\xce\x7c\x0e'\
b'\xfc\x66\x66\x7c\x6c\x66\xe6\x00'\
b'\x3c\x66\x30\x18\x0c\x66\x3c\x00'\
b'\x7e\x7e\x5a\x18\x18\x18\x3c\x00'\
b'\xc6\xc6\xc6\xc6\xc6\xc6\x7c\x00'\
b'\xc6\xc6\xc6\xc6\xc6\x6c\x38\x00'\
b'\xc6\xc6\xc6\xd6\xd6\xfe\x6c\x00'\
b'\xc6\xc6\x6c\x38\x6c\xc6\xc6\x00'\
b'\x66\x66\x66\x3c\x18\x18\x3c\x00'\
b'\xfe\xc6\x8c\x18\x32\x66\xfe\x00'\
b'\x3c\x30\x30\x30\x30\x30\x3c\x00'\
b'\xc0\x60\x30\x18\x0c\x06\x02\x00'\
b'\x3c\x0c\x0c\x0c\x0c\x0c\x3c\x00'\
b'\x10\x38\x6c\xc6\x00\x00\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\xff'\
b'\x30\x18\x0c\x00\x00\x00\x00\x00'\
b'\x00\x00\x78\x0c\x7c\xcc\x76\x00'\
b'\xe0\x60\x7c\x66\x66\x66\xdc\x00'\
b'\x00\x00\x7c\xc6\xc0\xc6\x7c\x00'\
b'\x1c\x0c\x7c\xcc\xcc\xcc\x76\x00'\
b'\x00\x00\x7c\xc6\xfe\xc0\x7c\x00'\
b'\x3c\x66\x60\xf8\x60\x60\xf0\x00'\
b'\x00\x00\x76\xcc\xcc\x7c\x0c\xf8'\
b'\xe0\x60\x6c\x76\x66\x66\xe6\x00'\
b'\x18\x00\x38\x18\x18\x18\x3c\x00'\
b'\x06\x00\x06\x06\x06\x66\x66\x3c'\
b'\xe0\x60\x66\x6c\x78\x6c\xe6\x00'\
b'\x38\x18\x18\x18\x18\x18\x3c\x00'\
b'\x00\x00\xec\xfe\xd6\xd6\xd6\x00'\
b'\x00\x00\xdc\x66\x66\x66\x66\x00'\
b'\x00\x00\x7c\xc6\xc6\xc6\x7c\x00'\
b'\x00\x00\xdc\x66\x66\x7c\x60\xf0'\
b'\x00\x00\x76\xcc\xcc\x7c\x0c\x1e'\
b'\x00\x00\xdc\x76\x60\x60\xf0\x00'\
b'\x00\x00\x7e\xc0\x7c\x06\xfc\x00'\
b'\x30\x30\xfc\x30\x30\x36\x1c\x00'\
b'\x00\x00\xcc\xcc\xcc\xcc\x76\x00'\
b'\x00\x00\xc6\xc6\xc6\x6c\x38\x00'\
b'\x00\x00\xc6\xd6\xd6\xfe\x6c\x00'\
b'\x00\x00\xc6\x6c\x38\x6c\xc6\x00'\
b'\x00\x00\xc6\xc6\xc6\x7e\x06\xfc'\
b'\x00\x00\x7e\x4c\x18\x32\x7e\x00'\
b'\x0e\x18\x18\x70\x18\x18\x0e\x00'\
b'\x18\x18\x18\x18\x18\x18\x18\x00'\
b'\x70\x18\x18\x0e\x18\x18\x70\x00'\
b'\x76\xdc\x00\x00\x00\x00\x00\x00'\
b'\x00\x10\x38\x6c\xc6\xc6\xfe\x00'\
b'\x7c\xc6\xc0\xc0\xc6\x7c\x0c\x78'\
b'\xcc\x00\xcc\xcc\xcc\xcc\x76\x00'\
b'\x0c\x18\x7c\xc6\xfe\xc0\x7c\x00'\
b'\x7c\x82\x78\x0c\x7c\xcc\x76\x00'\
b'\xc6\x00\x78\x0c\x7c\xcc\x76\x00'\
b'\x30\x18\x78\x0c\x7c\xcc\x76\x00'\
b'\x30\x30\x78\x0c\x7c\xcc\x76\x00'\
b'\x00\x00\x7e\xc0\xc0\x7e\x0c\x38'\
b'\x7c\x82\x7c\xc6\xfe\xc0\x7c\x00'\
b'\xc6\x00\x7c\xc6\xfe\xc0\x7c\x00'\
b'\x30\x18\x7c\xc6\xfe\xc0\x7c\x00'\
b'\x66\x00\x38\x18\x18\x18\x3c\x00'\
b'\x7c\x82\x38\x18\x18\x18\x3c\x00'\
b'\x30\x18\x00\x38\x18\x18\x3c\x00'\
b'\xc6\x38\x6c\xc6\xfe\xc6\xc6\x00'\
b'\x38\x6c\x7c\xc6\xfe\xc6\xc6\x00'\
b'\x18\x30\xfe\xc0\xf8\xc0\xfe\x00'\
b'\x00\x00\x7e\x18\x7e\xd8\x7e\x00'\
b'\x3e\x6c\xcc\xfe\xcc\xcc\xce\x00'\
b'\x7c\x82\x7c\xc6\xc6\xc6\x7c\x00'\
b'\xc6\x00\x7c\xc6\xc6\xc6\x7c\x00'\
b'\x30\x18\x7c\xc6\xc6\xc6\x7c\x00'\
b'\x78\x84\x00\xcc\xcc\xcc\x76\x00'\
b'\x60\x30\xcc\xcc\xcc\xcc\x76\x00'\
b'\xc6\x00\xc6\xc6\xc6\x7e\x06\xfc'\
b'\xc6\x38\x6c\xc6\xc6\x6c\x38\x00'\
b'\xc6\x00\xc6\xc6\xc6\xc6\x7c\x00'\
b'\x18\x18\x7e\xc0\xc0\x7e\x18\x18'\
b'\x38\x6c\x64\xf0\x60\x66\xfc\x00'\
b'\x66\x66\x3c\x7e\x18\x7e\x18\x18'\
b'\xf8\xcc\xcc\xfa\xc6\xcf\xc6\xc7'\
b'\x0e\x1b\x18\x3c\x18\xd8\x70\x00'\
b'\x18\x30\x78\x0c\x7c\xcc\x76\x00'\
b'\x0c\x18\x00\x38\x18\x18\x3c\x00'\
b'\x0c\x18\x7c\xc6\xc6\xc6\x7c\x00'\
b'\x18\x30\xcc\xcc\xcc\xcc\x76\x00'\
b'\x76\xdc\x00\xdc\x66\x66\x66\x00'\
b'\x76\xdc\x00\xe6\xf6\xde\xce\x00'\
b'\x3c\x6c\x6c\x3e\x00\x7e\x00\x00'\
b'\x38\x6c\x6c\x38\x00\x7c\x00\x00'\
b'\x18\x00\x18\x18\x30\x63\x3e\x00'\
b'\x00\x00\x00\xfe\xc0\xc0\x00\x00'\
b'\x00\x00\x00\xfe\x06\x06\x00\x00'\
b'\x63\xe6\x6c\x7e\x33\x66\xcc\x0f'\
b'\x63\xe6\x6c\x7a\x36\x6a\xdf\x06'\
b'\x18\x00\x18\x18\x3c\x3c\x18\x00'\
b'\x00\x33\x66\xcc\x66\x33\x00\x00'\
b'\x00\xcc\x66\x33\x66\xcc\x00\x00'\
b'\x22\x88\x22\x88\x22\x88\x22\x88'\
b'\x55\xaa\x55\xaa\x55\xaa\x55\xaa'\
b'\x77\xdd\x77\xdd\x77\xdd\x77\xdd'\
b'\x18\x18\x18\x18\x18\x18\x18\x18'\
b'\x18\x18\x18\x18\xf8\x18\x18\x18'\
b'\x18\x18\xf8\x18\xf8\x18\x18\x18'\
b'\x36\x36\x36\x36\xf6\x36\x36\x36'\
b'\x00\x00\x00\x00\xfe\x36\x36\x36'\
b'\x00\x00\xf8\x18\xf8\x18\x18\x18'\
b'\x36\x36\xf6\x06\xf6\x36\x36\x36'\
b'\x36\x36\x36\x36\x36\x36\x36\x36'\
b'\x00\x00\xfe\x06\xf6\x36\x36\x36'\
b'\x36\x36\xf6\x06\xfe\x00\x00\x00'\
b'\x36\x36\x36\x36\xfe\x00\x00\x00'\
b'\x18\x18\xf8\x18\xf8\x00\x00\x00'\
b'\x00\x00\x00\x00\xf8\x18\x18\x18'\
b'\x18\x18\x18\x18\x1f\x00\x00\x00'\
b'\x18\x18\x18\x18\xff\x00\x00\x00'\
b'\x00\x00\x00\x00\xff\x18\x18\x18'\
b'\x18\x18\x18\x18\x1f\x18\x18\x18'\
b'\x00\x00\x00\x00\xff\x00\x00\x00'\
b'\x18\x18\x18\x18\xff\x18\x18\x18'\
b'\x18\x18\x1f\x18\x1f\x18\x18\x18'\
b'\x36\x36\x36\x36\x37\x36\x36\x36'\
b'\x36\x36\x37\x30\x3f\x00\x00\x00'\
b'\x00\x00\x3f\x30\x37\x36\x36\x36'\
b'\x36\x36\xf7\x00\xff\x00\x00\x00'\
b'\x00\x00\xff\x00\xf7\x36\x36\x36'\
b'\x36\x36\x37\x30\x37\x36\x36\x36'\
b'\x00\x00\xff\x00\xff\x00\x00\x00'\
b'\x36\x36\xf7\x00\xf7\x36\x36\x36'\
b'\x18\x18\xff\x00\xff\x00\x00\x00'\
b'\x36\x36\x36\x36\xff\x00\x00\x00'\
b'\x00\x00\xff\x00\xff\x18\x18\x18'\
b'\x00\x00\x00\x00\xff\x36\x36\x36'\
b'\x36\x36\x36\x36\x3f\x00\x00\x00'\
b'\x18\x18\x1f\x18\x1f\x00\x00\x00'\
b'\x00\x00\x1f\x18\x1f\x18\x18\x18'\
b'\x00\x00\x00\x00\x3f\x36\x36\x36'\
b'\x36\x36\x36\x36\xff\x36\x36\x36'\
b'\x18\x18\xff\x18\xff\x18\x18\x18'\
b'\x18\x18\x18\x18\xf8\x00\x00\x00'\
b'\x00\x00\x00\x00\x1f\x18\x18\x18'\
b'\xff\xff\xff\xff\xff\xff\xff\xff'\
b'\x00\x00\x00\x00\xff\xff\xff\xff'\
b'\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0'\
b'\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f'\
b'\xff\xff\xff\xff\x00\x00\x00\x00'\
b'\x00\x00\x76\xdc\xc8\xdc\x76\x00'\
b'\x78\xcc\xcc\xd8\xcc\xc6\xcc\x00'\
b'\xfe\xc6\xc0\xc0\xc0\xc0\xc0\x00'\
b'\x00\x00\xfe\x6c\x6c\x6c\x6c\x00'\
b'\xfe\xc6\x60\x30\x60\xc6\xfe\x00'\
b'\x00\x00\x7e\xd8\xd8\xd8\x70\x00'\
b'\x00\x00\x66\x66\x66\x66\x7c\xc0'\
b'\x00\x76\xdc\x18\x18\x18\x18\x00'\
b'\x7e\x18\x3c\x66\x66\x3c\x18\x7e'\
b'\x38\x6c\xc6\xfe\xc6\x6c\x38\x00'\
b'\x38\x6c\xc6\xc6\x6c\x6c\xee\x00'\
b'\x0e\x18\x0c\x3e\x66\x66\x3c\x00'\
b'\x00\x00\x7e\xdb\xdb\x7e\x00\x00'\
b'\x06\x0c\x7e\xdb\xdb\x7e\x60\xc0'\
b'\x1e\x30\x60\x7e\x60\x30\x1e\x00'\
b'\x00\x7c\xc6\xc6\xc6\xc6\xc6\x00'\
b'\x00\xfe\x00\xfe\x00\xfe\x00\x00'\
b'\x18\x18\x7e\x18\x18\x00\x7e\x00'\
b'\x30\x18\x0c\x18\x30\x00\x7e\x00'\
b'\x0c\x18\x30\x18\x0c\x00\x7e\x00'\
b'\x0e\x1b\x1b\x18\x18\x18\x18\x18'\
b'\x18\x18\x18\x18\x18\xd8\xd8\x70'\
b'\x00\x18\x00\x7e\x00\x18\x00\x00'\
b'\x00\x76\xdc\x00\x76\xdc\x00\x00'\
b'\x38\x6c\x6c\x38\x00\x00\x00\x00'\
b'\x00\x00\x00\x18\x18\x00\x00\x00'\
b'\x00\x00\x00\x18\x00\x00\x00\x00'\
b'\x0f\x0c\x0c\x0c\xec\x6c\x3c\x1c'\
b'\x6c\x36\x36\x36\x36\x00\x00\x00'\
b'\x78\x0c\x18\x30\x7c\x00\x00\x00'\
b'\x00\x00\x3c\x3c\x3c\x3c\x00\x00'\
b'\x00\x00\x00\x00\x00\x00\x00\x00'\

FONT = memoryview(_FONT)
```

保存到fonts的方法和之前拷贝库一样用cp就好。拷贝完成后就可以继续下面的操作了。
这里说明一下，我是创建了一个st7789的目录，里面创建的这个文件，并写了一个main.py的文件做测试。
接下来就要进入主题了，需要编写一个main.py的程序，但是在编写之前，要去这里看看电路图的引脚接驳方式：
https://www.eetree.cn/project/detail/103

器件类型	器件型号	原理图符号	PCB封装	3D模型
PICO核心模块插座	PICO-R3	自己创建	来自VGA参考设计	GrabCAD下载
LCD模块	ST7789_1.54_240*240	OpenHandheld项目中提取	OpenHandheld项目中提取	GrabCAD下载
姿态传感器	MMA7660	MMA7660	KiCad库自带	KiCad库自带
环境光传感器	BH1750	KiCad库自带	KiCad库自带	KiCad库自带
电机插座	通用插座	KiCad库自带	KiCad库自带	KiCad库自带
SD卡座	通用插座	KiCad库自带	KiCad库自带	OpenHandheld项目中提取
蜂鸣器/SPeaker	通用器件	KiCad库自带	KiCad库自带	KiCad库自带
光电旋转编码器	KiCad库自带	KiCad库自带	KiCad库自带	GrabCAD下载
按键	通用器件	KiCad库自带	KiCad库自带	KiCad库自带
麦克风	通用器件	KiCad库自带	KiCad库自带	KiCad库自带
运算放大器	LMV358及其它，管脚/封装兼容	KiCad库自带	KiCad库自带	KiCad库自带
LDO	MIC5504-3.3及其它，管脚/封装兼容	KiCad库自带	KiCad库自带	KiCad库自带

这里点开，往下翻：



第1次直播授课介绍了项目的背景以及板卡要支持的主要功能 更新发布于 2021年02月14日

第2次直播 - 介绍了KiCad中元器件库的几种构建方式 更新发布于 2021年02月15日

第3次直播 - 原理图的绘制 更新发布于 2021年02月16日

完成原理图绘制，并初步的元器件布局 更新发布于 2021年02月17日

由于客观原因，定于今天的直播 - PCB布局推迟一天，明天再讲。

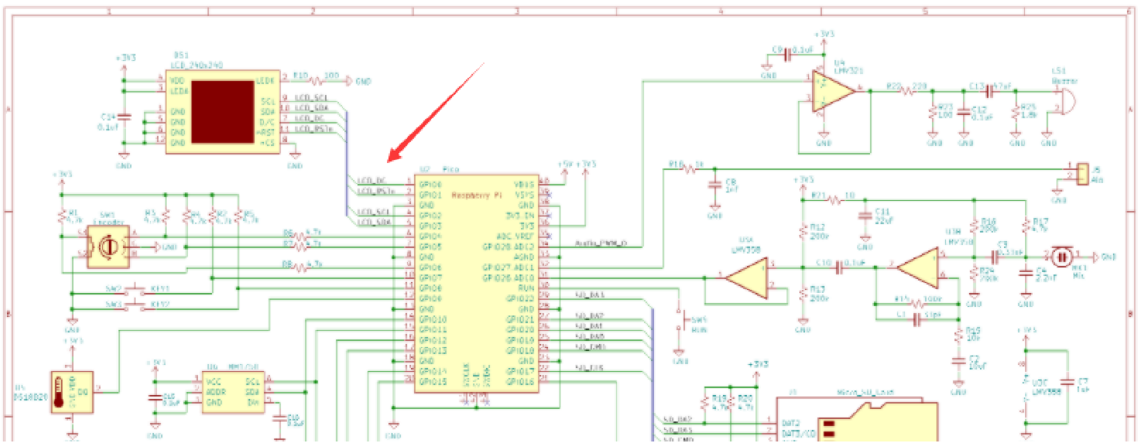
今天完成了原理图的设计以及简单的布局工作，做了一些简单的修订：

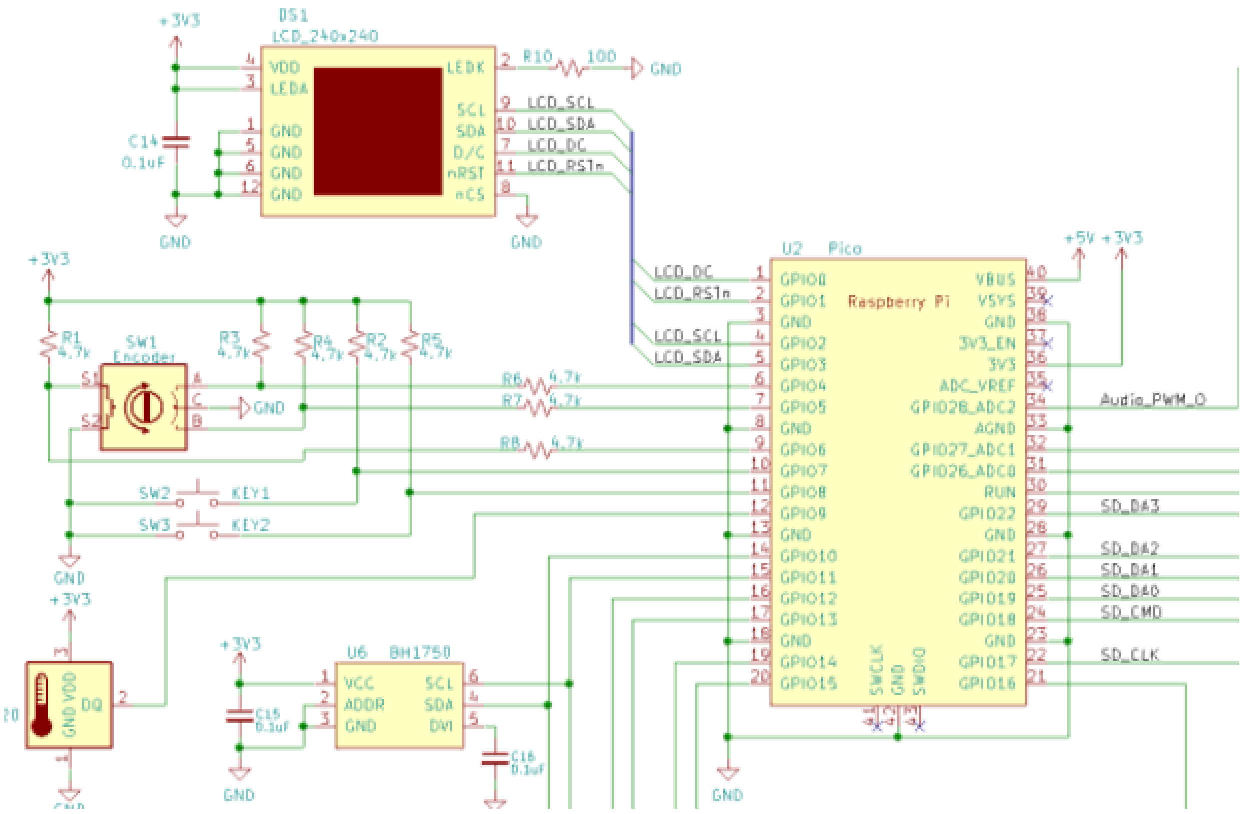
- LCD模块的原理图做了调整，显示更加只管
- 增加了一个环境光传感器BH1750，共用I2C总线连接
- 增加了麦克风的音频输入电路，采用1片双运放LMV358
- 增加了蜂鸣器的驱动输出，采用一片单运放的LMV321，性能同LMV358
- SD卡采用了4-bits的连接

看到这里放大：

今天完成了原理图的设计以及简单的布局工作，做了一些简单的修订：

- LCD模块的原理图做了调整，显示更加只管
- 增加了一个环境光传感器BH1750，共用I2C总线连接
- 增加了麦克风的音频输入电路，采用1片双运放LMV358
- 增加了蜂鸣器的驱动输出，采用一片单运放的LMV321，性能同LMV358
- SD卡采用了4-bits的连接





在这里我要整理一下思路：

ST7789 LCD	RPI-Pico	Pico Function
LCD_SCL	GP2	SCK
LCD_SDA	GP3	TX
LCD_DC	GP0	RX

LCD_RSTn	GP1	CS
LCD_CSn	GND	GND

这里要注意一下，这里LCD_DC和LCD_RSTn 在写代码的时候要交换一下，否则屏幕亮不起来。
下面是主程序main.py，我先列出来，然后给大家讲解做了什么操作。

```
import uos
import machine
import st7789py as st7789
from fonts import vga2_8x8 as font1
from fonts import vga1_16x32 as font2
import random
import time
import utime


# 定义默认SPI(0)引脚
sck  = 2
mosi = 3
rst  = 0
dc   = 1

# 定义屏幕信息
width = 240
height = 240

CENTER_Y = int(width/2)
CENTER_X = int(height/2)

# 打印系统信息
print(uos.uname())

# 初始化一个spi0对象
spi0 = machine.SPI(0, baudrate=40000000, polarity=1, phase
=0, sck=machine.Pin(sck), mosi=machine.Pin(mosi))
print(spi0)

# 初始化内部的温度sensor，接4号引脚的
adc = machine.ADC(4)
```

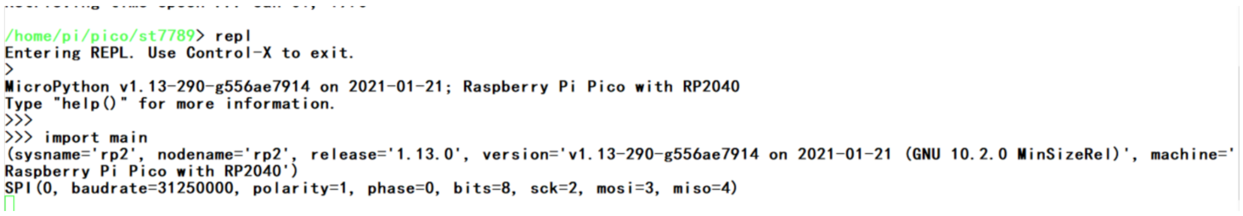
```
# 实例化一个display对象
display = st7789.ST7789(spi0, width, width, reset=machine.
Pin(rst, machine.Pin.OUT),dc=machine.Pin(dc, machine.Pin.O
UT),xstart=0, ystart=0, rotation=0)
仔细看看，引脚定义和之前的表格稍微换了了一下，reset和dc交换了一下。

#  利用对象的fill方法填充颜色，  r ，g ，b但是要用st7789的color56
5转换。
display.fill(st7789.color565(0, 255, 120))
time.sleep(2)
display.fill(st7789.BLACK)

# text就是刷文字上去，参数格式是：字体，内容，  x坐标，y坐标
display.text(font2, "Hello!", 10, 10)
time.sleep(.2)
display.text(font2, "RPi Pico", 10, 40)
time.sleep(.2)
display.text(font2, "EETREE NICE", 10, 70)
time.sleep(.2)
display.text(font1, "ST7789 SPI 240*240 IPS", 10, 100)
time.sleep(.2)
display.text(font1, "eetree.cn", 10, 110)
time.sleep(.2)
display.text(font1, "Piday, let's have fun!", 10, 120)
time.sleep(.2)

# pixel允许打点，就是一次可以画一个点，  一个pixel，所以，如果我随机
画点在屏幕上，就会有高斯模糊的效果。但是我注释掉了，主要是有点儿乱。
"""
for i in range(5000):
    display.pixel(random.randint(0, width),
          random.randint(0, height),
          st7789.color565(random.getrandbits(8),random.get
randbits(8),random.getrandbits(8)))
"""

# 下面这个画圆形的是从别出抄来的，参数内容是x坐标，y坐标，半径，颜
色。
# Helper function to draw a circle from a given position w
ith a given radius
```

```
# This is an implementation of the midpoint circle algorit
hm,
# see https://en.wikipedia.org/wiki/Midpoint_circle_algori
thm#C_example
# for details
def draw_circle(xpos0, ypos0, rad,
col=st7789.color565(255, 255, 255)):
    x = rad - 1
    y = 0
    dx = 1
    dy = 1
    err = dx - (rad << 1)
    while x >= y:
        display.pixel(xpos0 + x, ypos0 + y, col)
        display.pixel(xpos0 + y, ypos0 + x, col)
        display.pixel(xpos0 - y, ypos0 + x, col)
        display.pixel(xpos0 - x, ypos0 + y, col)
        display.pixel(xpos0 - x, ypos0 - y, col)
        display.pixel(xpos0 - y, ypos0 - x, col)
        display.pixel(xpos0 + y, ypos0 - x, col)
        display.pixel(xpos0 + x, ypos0 - y, col)
        if err <= 0:
            y += 1
            err += dy
            dy += 2
        if err > 0:
            x -= 1
            dx += 2
            err += dx - (rad << 1)

# draw_circle(CENTER_X, CENTER_Y, 100)
# 我这里通过遍历一个20~100的x坐标，然后在y坐标为160的位置上画一个半
径为15像素的圆圈。
for i in range(20,100,20):
    draw_circle(i,160,15)
# 因为画了一个奥迪的标志，不写个标题奥迪不给打钱。
display.text(font1, "Audi", 20, 180)

# 这是尝试实时刷新看看刷5个数字。
for i in range(5):
    display.text(font2, "Last: "+str(i)+" s", 20, 200)
    time.sleep(1)

# 这里定义采样的精度因子然后进入一个循环读取ADC这边的温度计的信息，然后显示在x20，y10的位置上，没有调好，还有点儿乱。
factor = 3.3 / (65535)
try:
    while True:
        reading = adc.read_u16() * factor
        temperature = 27 - (reading - 0.706)/0.001721
        utime.sleep(2)
        display.text(font2,"CPU_Temp:"+str(temperature), 20,10)
except KeyboardInterrupt:
    display.fill(st7789.color565(255,0,0))
    display.text(font2, "Good Bye!", 20, 110)
```

接下来就将这个文件拷贝到pyboard里面，然后进入repl， 然后import main就可以看到效果。

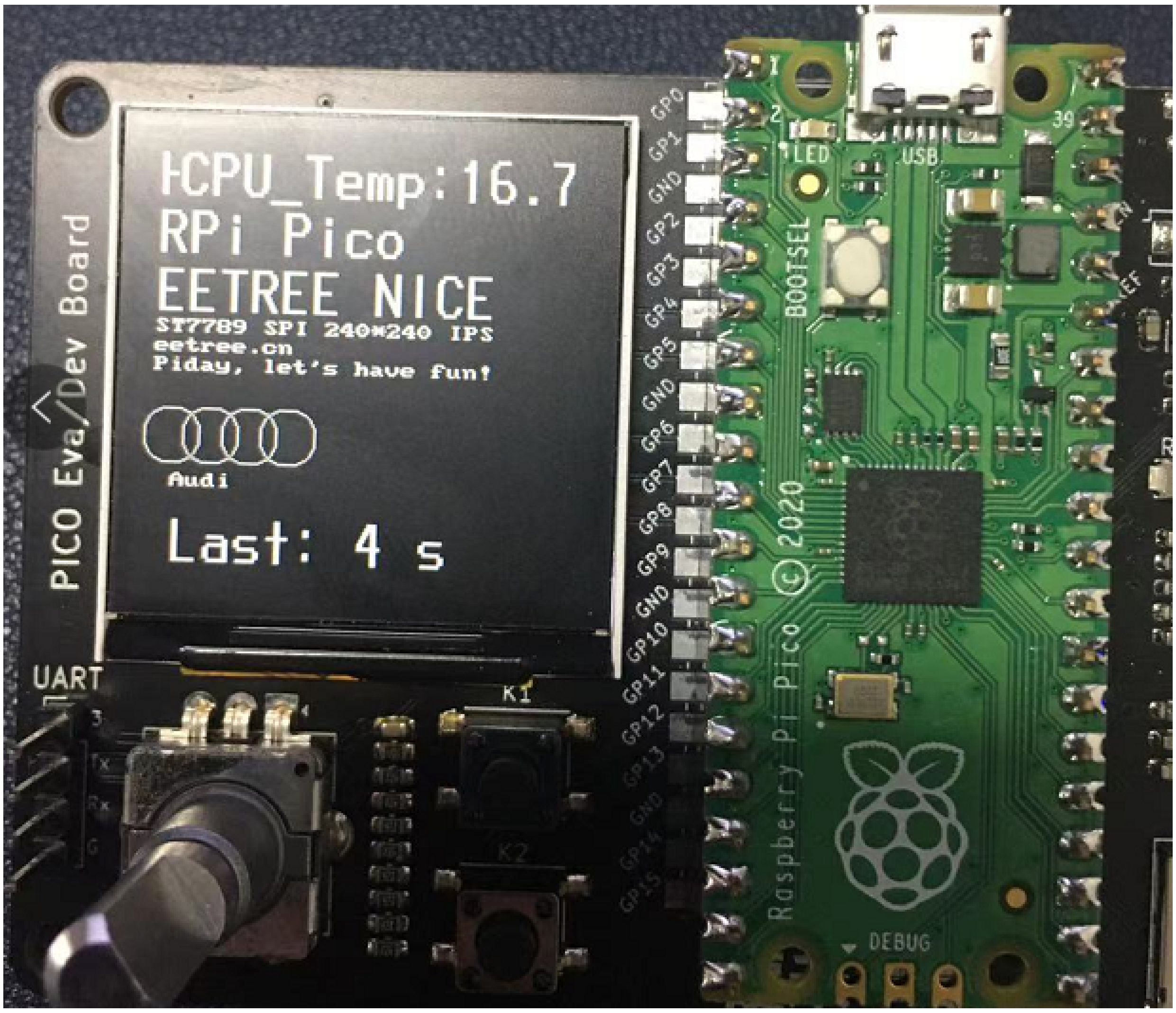
```
cp main.py  /pyboard/main.py
repl
```

执行效果如图：

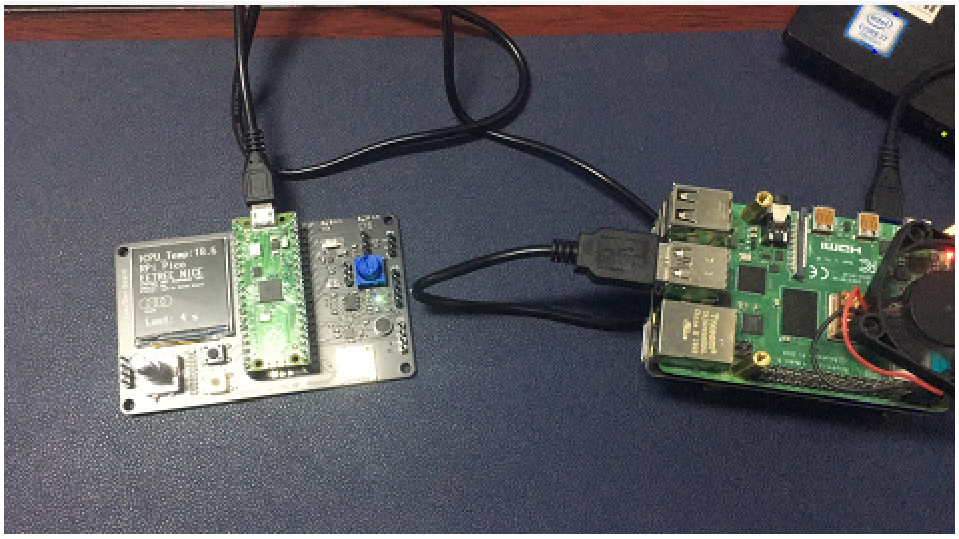


```
/home/pi/pico/st7789> repl
Entering REPL. Use Control-X to exit.
>
MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
>>> import main
(sysname='rp2', nodename='rp2', release='1.13.0', version='v1.13-290-g556ae7914 on 2021-01-21 (GNU 10.2.0 MinSizeRel)', machine='
Raspberry Pi Pico with RP2040')
SPI(0, baudrate=31250000, polarity=1, phase=0, bits=8, sck=2, mosi=3, miso=4)
```

它就执行了，屏幕上就开始显示内容了。
当需要退出，按下CTRL+X退出REPL，如果要退出RSHELL，CTRL+D
下面是内容展示：



整体图：



到这里，通过pico点亮ST7789 SPI 屏幕搞定。