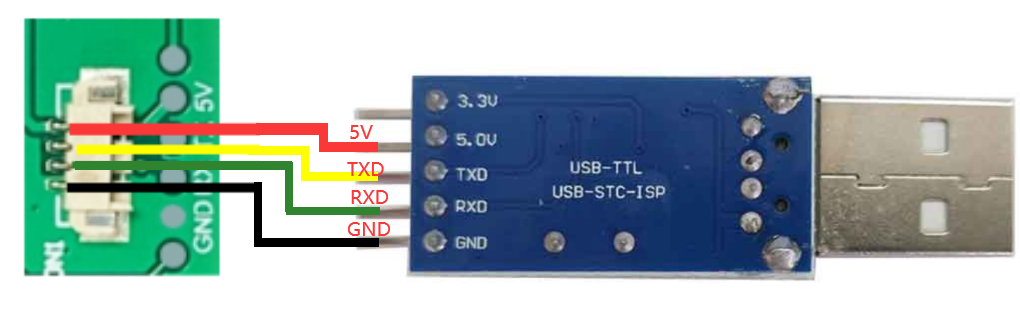
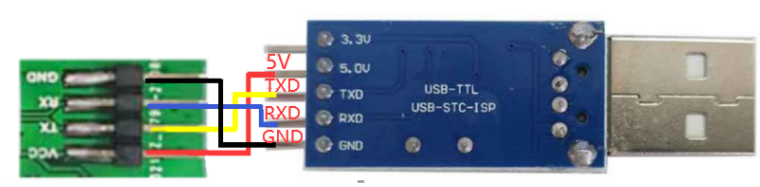
**1.How to connect and use?**

1.1 Test it by computer

Universal version：

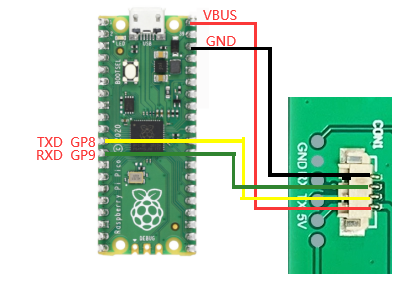


Pico version：

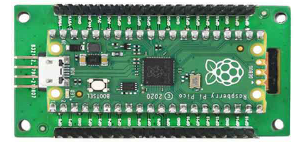


1.2 Test it by Pico

Universal version：

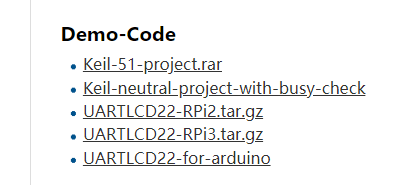


Pico version：



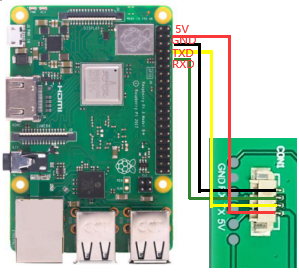
1.3 Test it by Raspberry Pi/Arduino/51  
You can find the demo code at here

<http://www.spotpear.com/index/study/detail/id/585.html>

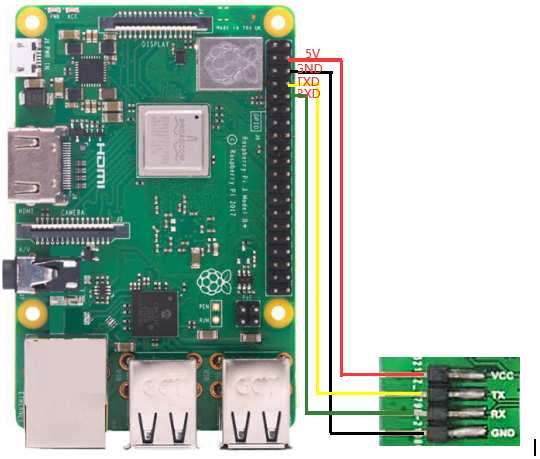


If you use it at raspberry pi. You need connect like this

Universal version：

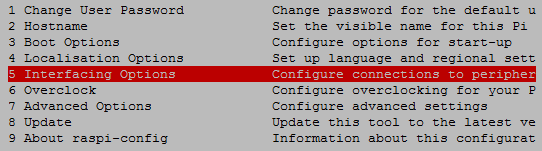


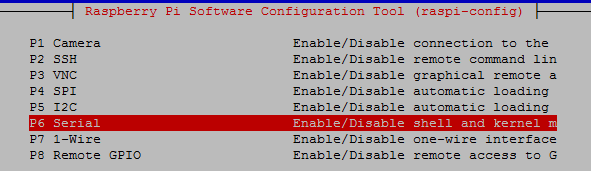
Pico version：

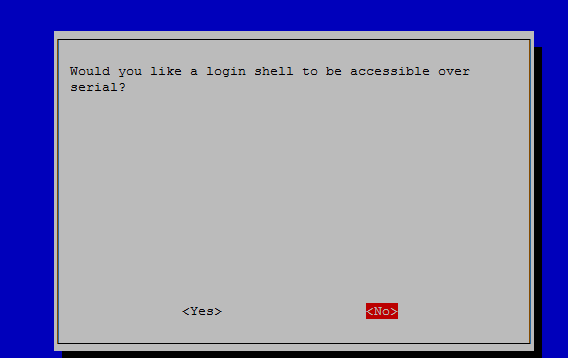


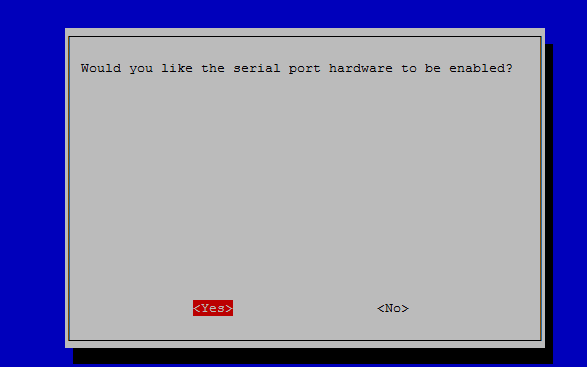
Before you run the program, you need to configure to disable the UART

sudo raspi-config









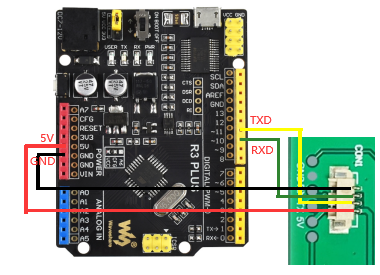
After reboot. On the Terminal, enter  
 sudo wget http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/UARTLCD22-RPi3.tar.gz  
 sudo tar xvf UARTLCD22-RPi3.tar.gz

cd UARTLCD22-RPi3/   
sudo rm UART

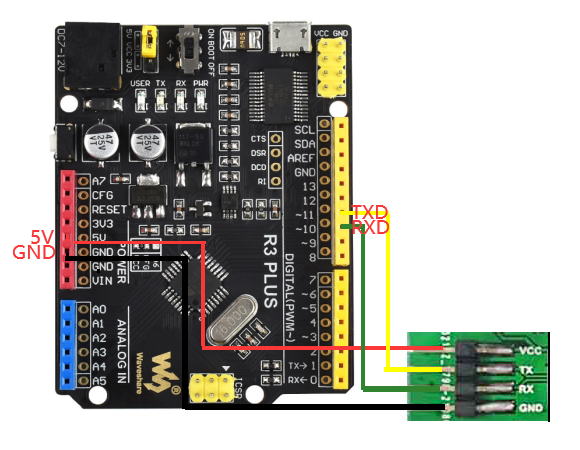
sudo make  
sudo ./UART

**1.4** If you use it at ARDUINO. You need connect like this

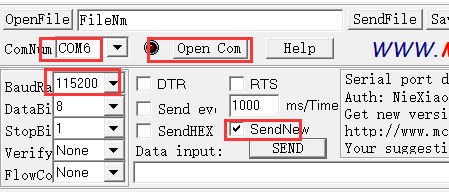
Universal version：

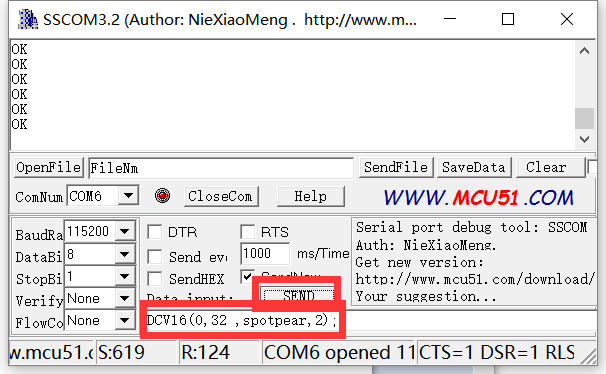


Pico version：



1.5 How to install the USB TO TTL module PL2303 driver？  
http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/UART-driver.rar

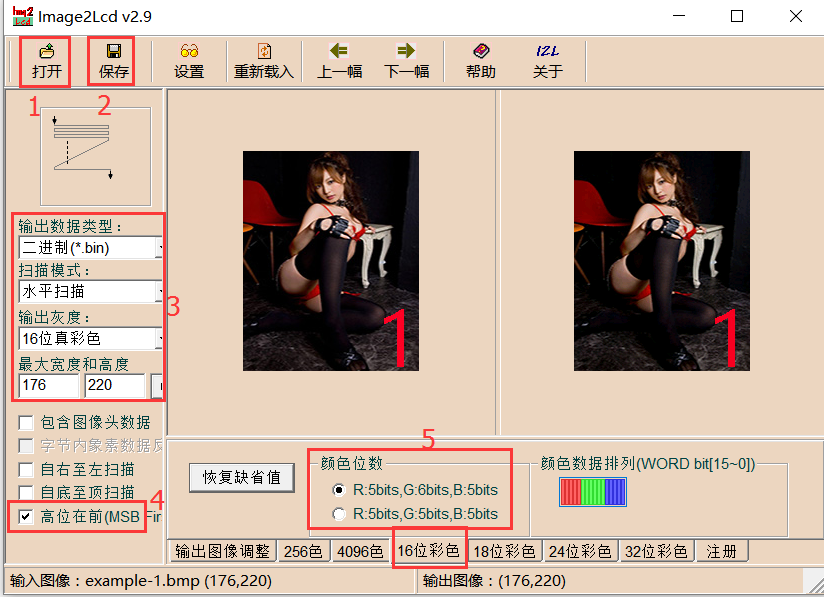
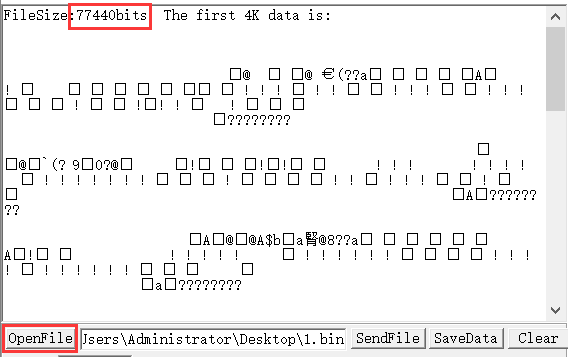
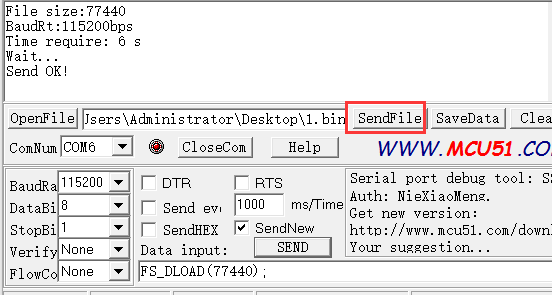
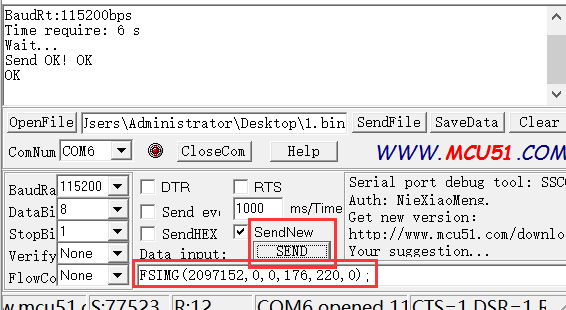
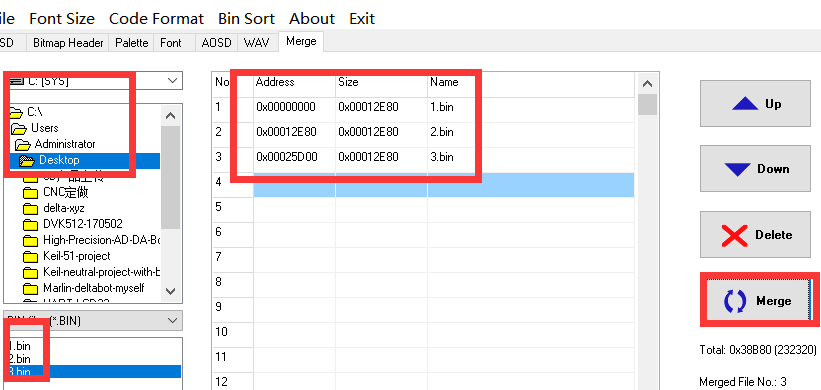
1.6 Open sscom32：http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/sscom32E.rar  


1.6.1 Test the LCD with following simple CMD:  
  
DIR(1);  
CIRF(40,80,20,3);  
CIR(40,120,20,3);  
BOXF(90,30,100,40,1);  
BOX(110,40,120,60,1);

DCV32(0,0 ,spotpear,0);  
CLR(0);  
PL(0,0,220,176,1);  
PS(40,40,3);  
SBC(1);   
DCV16(0,32 ,spotpear,2);  
DIR(0);  
DCV24(0,0 ,spotpear,0);  
  
About above CMD, you can see more detailed introduction later in the article

1.6.2 How to display a picture？  
->1.6.2-1 Get a picture （you need to change your picture to bmp format）  
[click to download it](http://www.spotpear.com/download/LCD/uart/UART-LCD22/example-picture/) （example-1.bmp）

->1.6.2-2 change .bmp picture to bin file：  
Download this software to change .bmp to .bin file  
http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/Image2Lcd%202.9.rar

1->open ; set as 3,4,5; 2->save. You will get the .bin file  
  
or you [click to get it](http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/1.bin) （The file“1.bin”is made by us）  
->1.6.2-3 display the picture  
Use SSCOM3.2 to open the .bin file ，you will see the file Size 77440bits （220x176 picture）  
  
Enter FS\_DLOAD(77440);   
  
Then send file  
  
Enter FSIMG(2097152,0,0,176,220,0);  
And press “SEND”to display the picture  
  
If no picture showed，you need to press “SEND” again  
->6.2.4 Download 3 pictures  
Do “->1.6.2-1”and “->1.6.2-2” to get file“2.bin”and“3.bin”from“example-2.bmp”“example-3.bmp”  
Put“1.bin”“2.bin”“3.bin”together to “Pic.BIN” by this software：  
http://spotpear.com/uploads/picture/learn/common-lcd/lcd/uart-lcd-22/EzOSD\_v015T9.rar  
  
Do “->1.6.2-3” to download “Pic.BIN”（change file“1.bin”to“Pic.BIN”）and show“example-1.bmp”  
If you want to display example-2.bmp and example-3.bmp， You need to change to enter  
FSIMG(2174592,0,0,176,220,0);  
to display example-2.bmp  
And enter  
FSIMG(2252032,0,0,176,220,0);  
to display example-3.bmp  
  
PS:display picture ， the starting address is 2097152，each adding picture need add 77440，So you can see   
FSIMG(2097152,0,0,176,220,0); ->picture 1  
FSIMG(2174592,0,0,176,220,0); ->picture 2   
FSIMG(2252032,0,0,176,220,0); ->picture 3  
FSIMG(2329472,0,0,176,220,0); ->picture 4  
FSIMG(2406912,0,0,176,220,0); ->picture 5  
FSIMG(2484352,0,0,176,220,0); ->picture 6

...............

**2.Software Description**

(1)Pre-knowledge

1. Display memory：DDRAM:176\*220\*2=77440
2. Memory address：

Horizontal screen mode DDRAM address arrangement

Vertical screen mode DDRAM address arrangement

(2) Initialize the LCD

POWER the host

The host configures the serial port and waits 900ms

LCD enters the ready state

(3) UART command introduction

|  |  |  |
| --- | --- | --- |
|  | ***CMD*** | ***DESCRIPTION*** |
| Reset | RESET; |  |
| This command allows the module to enter the software reset, receive this command, the module's peripheral components and system parameters will be restored to the power value. |
| Get the version information instructions for the module | VER; |  |
| Through the VER; you can get the firmware version of this module information, and displayed on the screen |
| Set the baud rate command | BPS(bps); | The default baud rate is 115200 when the system is powered on. |
|  |
| Clear command | CLR(c); | Note that the range of c is 0 to 15, and if the value of c exceeds 15, the system will not respond to the CMD, and the range of c values will look at the following color list. |
| CLR for the script, c for the clear use of the background color, the specific code see the following color list. If you want to fill the screen with black, then CLR (0); |
| LCD control CMD | LCDON（on\_off）; | On\_off parameters only 0 or 1, the system ignores other parameters. |
| LCDON for the script, on\_off, respectively, that start or turn off the LCD. Such as LCDON (1); that start LCD, LCDON (0); turn off the LCD. |
| Display the LCD | FSIMG(addr,x，y，w,h,mode); | When Mode is 1, the white background of the picture will not be displayed. This mode is used to overlay the icon and the background image. Addr is the flash start address for storing pictures, starting at 2097152 |
| FSIMG for the script, addr for the picture stored in the flash address, x, y for the picture to be displayed on the screen above the starting position, w for the picture width, h for the picture height, mode for the picture display: 1 for the transparent display , 0 is normal display. Such as FSIMG (2097152,0,0,240,400,1); that from the 2097152 FLASH address removed 240 \* 400 pictures and 0,0 position transparent display. |
| Image download to FLASH command | FS\_DLOAD(SIZE); | Picture will be downloaded to the FLASH 2M high storage space, so from 2M (2097152 position to start storing pictures) a total of 2M  This command supports the merger of the picture programming, does not support a single picture file programming. |
| FS\_DLOAD is the script, and SIZE is the total size of the picture to be downloaded. Such as FS\_DLOAD (192000); that 192000 bytes of pictures downloaded to the flash, the total size of the picture can not exceed 2097152 bytes, if the SIZE assignment greater than 2097152 bytes, the system only to identify 2097152 bytes. |
| SDIMG for the script, x, y for the picture to be displayed at the beginning of the screen position, w, h were the width and height of the picture, 'name' for the file name, currently only supports English name. SDIMG (0,0,240,400, '6.bin'); that is, the SD card stored 6.bin file in the module 0,0 position display |
| Vertical and horizontal screen switch CMD | DIR(H\_V); | The LCD is displayed by default for DIR (0); for vertical screen |
| Such as DIR (0); for vertical screen. DIR (1); for horizontal screen |
| Set the brightness of the backlight | BL (p); | After the system is powered on, the brightness of the backlight is 20 |
| where BL is the instruction code, p is the brightness value of the backlight, the adjustment range is: 0 ~ 255, where 0 is full display, 255 is off display. |
| Draw points | PS (x, y, c); | This instruction does not apply to large areas of speculation, if there is a need to recommend built-in internal modules |
| where PS is the instruction code, x, y is the starting position of the display, c is the color of the point |
| Draw lines | PL (x1, y1, x2, y2, c) | Note that the range of c is 0 to 15, and if the value of c exceeds 15, the system will ignore this operation. |
| where PL is the instruction code, x1, y1 is the starting point, x2, y2 is the position of the end point, c is the color of the line |
| Draw box | BOX (x1, y1, x2, y2, c) |  |
| where BOX is the instruction code, x1, y1, the position of the starting point, x2, y2 is the position of the end point, c is the color of the box |
| Draw box with Filled color | BOXF (x1, y1, x2, y2, c); |  |
| where BOXF is the instruction code, x1, y1, the position of the starting point, x2, y2 is the position of the end point, c is the color of the box |
| Draw a circle | CIR (x, y, r, c); |  |
| where CIR is the instruction code, x, y is the center of the circle, r is the radius of the circle, c is the circle color |
| Draw a circle with Filled color | CIRF (x, y, r, c); |  |
| where CIRF is the instruction code, x, y is the center of the circle, r is the radius of the circle, c is the color of the circle |
| Set background color | SBC (c); |  |
| where SBC is the instruction code, c is the background color value, and c ranges from 0 to 63. |
| Display 16 high character With background color instruction | DCV16 (x, y, \* str, c); |  |
| where DCV16 is the instruction code, x, y is the starting position of the character, \* str is the pointer of the character, c is the color of the character |
| Display 24 high character With background color instruction | DCV24 (x, y, \* str, c); |  |
| where DCV24 is the instruction code, x, y is the starting position of the character, \* str is the pointer of the character, c is the color of the character |
| Display 32 high character With background color instruction | DCV32 (x, y, \* str, c); |  |
| where DCV32 is the instruction code, x, y is the starting position of the character, \* str is the pointer of the character, c is the color of the character |  |

Color list

|  |  |
| --- | --- |
| black | 0 |
| red | 1 |
| green | 2 |
| blue | 3 |
| yellow | 4 |
| cyan-blue | 5 |
| purple | 6 |
| gray | 7 |
| Light gray | 8 |
| brown | 9 |
| Dark green | 10 |
| Navy blue | 11 |
| Dark yellow | 12 |
| Orange | 13 |
| Light red | 14 |
| white | 15 |

(4)

Each instruction must end with a semicolon symbol, and each operation must end with a newline symbol.   
semicolon symbol ;   
semicolon symbol \r\n

(5)Programming examples：  
 STM32F103RBT6:

void uart\_init(u32 bound){

// GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1|RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_AFIO, ENABLE);//USART1\_TX PA.9

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//USART1\_RX PA.10

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//Usart1 NVIC ÅäÖÃ

NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 3; //

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

USART\_InitStructure.USART\_BaudRate = bound;//

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(USART1, &USART\_InitStructure);

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//

USART\_Cmd(USART1, ENABLE);

}

void UartSend(char \* databuf) //

{

u8 i=0;

while (1)

{

if (databuf[i]!=0)//

{

USART\_SendData(USART1, databuf[i]); //

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET){}; //

i++;

}

else return;

}

}

int main(void)

{

SystemInit();//

delay\_init(72); //

uart\_init(115200); //

delay\_ms(500);

for(;;)

{

UartSend("SBC(15);DIR(0);FSIMG(2329472,0,0,176,220,0);DIR(1);SBC(10);\r\n");

CheckBusy();

UartSend("DC32(0,0,'spotpear',1);\r\n");

CheckBusy();

UartSend("DC24(0,32,'spotpear1',2);\r\n");

CheckBusy();

UartSend("DC24(0,56,'spotpear2',4);\r\n");

CheckBusy();

UartSend("DC16(0,80,'spotpear3',3);\r\n");

CheckBusy();

UartSend("DC16(0,96,'spotpear4',1);\r\n");

CheckBusy();

UartSend("DC16(0,112,'spotpear5',1);\r\n");

CheckBusy();

UartSend("PS(10,10,14);\r\n");

CheckBusy();

UartSend("BOX(120,140,150,160,3);\r\n");

CheckBusy();

UartSend("CIRF(70,150,20,1);\r\n");

CheckBusy();

while(1);

}

}

**2.2 Pico（Python）**

**Note: Please modify the blue and red characters, according to the actual parameters.**

from machine import UART, Pin

import time

import sys

uart1 = UART(1, baudrate=115200, bits=8, parity=None, stop=1,tx=Pin(8), rx=Pin(9))

uart0 = UART(0, baudrate=115200, bits=8, parity=None, stop=1,tx=Pin(0), rx=Pin(1))

txData = u'CLR(0);\r\n'

uart1.write(txData)

time.sleep(0.1)

txData = b"DIR(1);DC24(20,0,\'spotpear\',1);DC24(20,70,\' UART LCD for Pico\',2);BOX(120,140,160,180,3);CIRF(70,150,30,4);DELAYMS (500000);DELAYMS (500000);CLR(0);DIR(1);DELAYMS(400);CLR(6);DELAYMS (400);FSIMG(2097152,0,0,376,240,0);DELAYMS(600);CLR(4);DELAYMS(400);FSIMG(2277632,0,0,376,240,0);;DELAYMS(600);CLR(5);DELAYMS(400);FSIMG(2458112,0,0,376,240,0);\r\n"

uart1.write(txData.decode('unicode'))

time.sleep(0.1)

rxData = bytes()

while uart0.any() > 0:

rxData += uart0.read(1)

print(rxData.decode('utf-8'))

**2.3 Raspberry Pi 3**

**Note: Please modify the blue and red characters, according to the actual parameters.**

#include <stdio.h>

#include <wiringPi.h>

#include <wiringSerial.h>

int main()

{

int fd;

if(wiringPiSetup() < 0)return 1;

// if((fd = serialOpen("/dev/ttyAMA0",115200)) < 0)return 1;

if((fd = serialOpen("/dev/ttyS0",115200)) < 0)return 1;

printf("serial test start ...\n");

delay(800);

serialPrintf(fd,"RESET;\r\n");//reset the LCD

delay(100);

serialPrintf(fd,"BPS(115200);\r\n");//Set Baud rate

delay(100);

serialPrintf(fd,"CLR(0);\r\n");//Clean LCD with black color

delay(100);

serialPrintf(fd,"CLR(1);\r\n");//Clean LCD with red color

delay(100);

serialPrintf(fd,"CLR(15);\r\n");//Clean LCD with white color

delay(100);

serialPrintf(fd,"DIR(0);\r\n");//Vertical display

delay(100);

serialPrintf(fd,"DCV24(0,0,spotpear,0);\r\n");

//display "spotpear" at coordinate（0.0），Font color ：0-black；background color ：default black

delay(100);

serialPrintf(fd,"SBC(1);\r\n");//set background color red

delay(100);

serialPrintf(fd,"DCV24(0,24,spotpear,0);\r\n");

//display "spotpear" at coordinate（X-0.Y-24）

delay(500);

serialPrintf(fd,"DCV24(0,24,spotpear,3);\r\n");//，Font color ：3-;

delay(500);

serialPrintf(fd,"CLR(0);\r\n");//Clean LCD with black color

delay(500);

serialPrintf(fd,"DIR(1);\r\n");//Horizontal display

delay(500);

serialPrintf(fd,"DCV16(0,24,spotpear,0);\r\n");

delay(500);

serialPrintf(fd,"DCV32(0,0,spotpear,0);\r\n");

delay(500);

serialPrintf(fd,"CIRF(40,80,20,3);\r\n");//filling circle coordinate（X-40.Y-80,r-20,color-3）

delay(100);

serialPrintf(fd,"CIR(70,150,20,1);\r\n");//circle coordinate（X-70.Y-150,r-20,color-1）

delay(500);

serialPrintf(fd,"BOXF(70,150,90,170,3);\r\n");//rectangle coordinate

delay(500);

serialPrintf(fd,"BOX(40,80,70,110,3);\r\n");//rectangle coordinate

delay(500);

serialPrintf(fd,"PL(0,0,220,176,6);\r\n");//line: color-6,

delay(500);

serialPrintf(fd,"PS(110,110,4);\r\n");//line: color-6,

delay(1000);

serialPrintf(fd,"DIR(0);\r\n");//Vertical display

delay(100);

serialPrintf(fd,"FSIMG(2097152,0,0,176,220,0);\r\n");

//load picture-1 from LCD（picture loaded by computer UART software in advance）

delay(500);

serialPrintf(fd,"FSIMG(2174592,0,0,176,220,0);\r\n");//load picture-2 from LCD

delay(500);

serialPrintf(fd,"FSIMG(2252032,0,0,176,220,0);\r\n");

delay(500);

serialPrintf(fd,"BL(1023);\r\n");////Backlight ightness:1024-open display

delay(1000);

serialPrintf(fd,"BL(0);\r\n");//Backlight ightness:0-stop display

delay(300);

// serialPrintf(fd,"RESET;\r\n");//reset\*/

// delay(300);

serialPrintf(fd,"DCV24(0,0,spotpear,0);\r\n");

delay(300);

//while(1)

//{

// serialPutchar(fd,serialGetchar(fd));

//}

serialClose(fd);

return 0;

}

**2.4 Arduino**

**Note: Please modify the blue and red characters, according to the actual parameters.**

UARTLCD22-1

/\*

Software serial multple serial test

Receives from the hardware serial, sends to software serial.

Receives from software serial, sends to hardware serial.

The circuit:

\* RX is digital pin 10 (connect to TX of other device)

\* TX is digital pin 11 (connect to RX of other device)

Note:

Not all pins on the Mega and Mega 2560 support change interrupts,

so only the following can be used for RX:

10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69

Not all pins on the Leonardo support change interrupts,

so only the following can be used for RX:

8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

created back in the mists of time

modified 25 May 2012

by Tom Igoe

based on Mikal Hart's example

This example code is in the public domain.

\*/

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()

{

mySerial.begin(115200);

delay(800);

mySerial.println("RESET;\r\n");

delay(100);

mySerial.println("BPS(115200);\r\n");

delay(100);

mySerial.println("CLR(1);\r\n");

delay(500);

mySerial.println("CLR(15);\r\n");

delay(500);

mySerial.println("DIR(0);\r\n");

delay(100);

mySerial.println("DCV24(0,0,spotpear,0);\r\n");

delay(100);

mySerial.println("SBC(1);\r\n");

delay(100);

mySerial.println("DCV24(0,24,spotpear,0);\r\n");

delay(300);

mySerial.println("DCV24(0,24,spotpear,3);\r\n");

delay(300);

mySerial.println("CLR(0);\r\n");

delay(300);

mySerial.println("FSIMG(2097152,0,0,176,220,0);\r\n");

delay(300);

mySerial.println("FSIMG(2174592,0,0,176,220,0);\r\n");

delay(300);

mySerial.println("FSIMG(2252032,0,0,176,220,0);\r\n");

delay(300);

mySerial.println("BL(1023);\r\n");

delay(1000);

mySerial.println("BL(0);\r\n");

delay(1000);

}

void loop() // run over and over

{

delay(300);

}

UARTLCD22-2

/\*

Software serial multple serial test

Receives from the hardware serial, sends to software serial.

Receives from software serial, sends to hardware serial.

The circuit:

\* RX is digital pin 10 (connect to TX of other device)

\* TX is digital pin 11 (connect to RX of other device)

Note:

Not all pins on the Mega and Mega 2560 support change interrupts,

so only the following can be used for RX:

10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69

Not all pins on the Leonardo support change interrupts,

so only the following can be used for RX:

8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

created back in the mists of time

modified 25 May 2012

by Tom Igoe

based on Mikal Hart's example

This example code is in the public domain.

\*/

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()

{

mySerial.begin(115200);

delay(800);

mySerial.println("RESET;\r\n");

delay(300);

mySerial.println("DIR(1);\r\n");

delay(500);

mySerial.println("CLR(0);\r\n");

delay(500);

mySerial.println("DCV16(0,24,spotpear,0);\r\n");

delay(300);

mySerial.println("DCV32(0,0,spotpear,0);\r\n");

delay(300);

mySerial.println("CIRF(40,80,20,3);\r\n");

delay(300);

mySerial.println("CIR(70,150,20,1);\r\n");

delay(300);

mySerial.println("BOXF(70,150,90,170,3);\r\n");

delay(300);

mySerial.println("BOX(40,80,70,110,3);\r\n");

delay(300);

mySerial.println("PL(0,0,220,176,6);\r\n");

delay(300);

mySerial.println("PS(110,110,4);\r\n");

delay(300);

}

void loop() // run over and over

{

delay(1000);

}