

LCD Graphics Display



Introduction:

This lecture introduces the 480 x 320 graphics display on your Pico Breadboard Kit along with the driver routine (LCD.py). To use the LCD display, you need to copy the LCD.py library to your Pico board as well as to a /py subdirectory on your PC. Once copied, these LCD routines are available to be imported and used with Python.

To use the LCD library:

- Copy LCD.py to your Pico chip
- Create a subdirectory on your PC named .../py
- Copy LCD.py to the .../py subdirectory on your PC

Hardware Connections & Data Communications

The heart of the LCD display is a ST7796S driver chip. This chip is hard-wired to your Pico using GP2..7:

GPx	ST7796S Connection	Description
2	CLK	SPI Clock line. Runs at 40MHz (!)
3	DIN	Data In. Data sent from Pico to the LCD
4	DOUT	Data Out. Currently not used
5	CS	Chip Select. 0 indicated writing to the LCD
6	DC	Data / Command. 0 = Data, 1 = Command
7	RST	Reset. Pulse low to reset the display

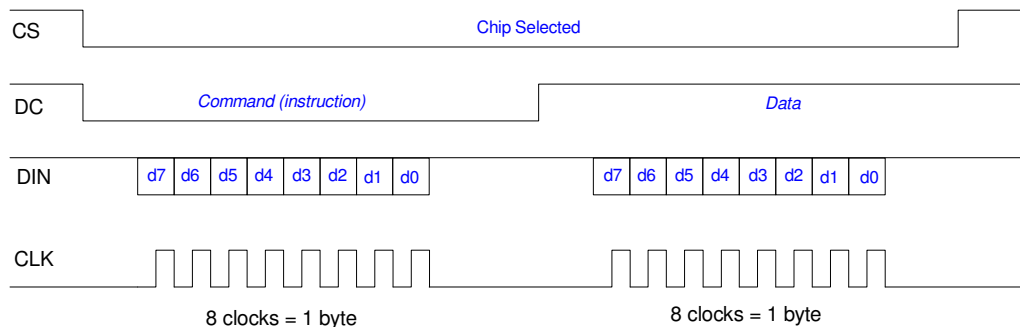
Note that GP2..4 (SPI CLK, DIN, DOUT) can be used with other devices as well as long as each device has a separate chip select.

Also note that this means don't use GP2..7 for other I/O devices if you plan on using the LCD display.

Data communications from the Pico to the LCD follows standard SPI protocol

- At the start of a message, Chip Select goes low
- DC is set to indicate if the message is a command (0) or data (1)
- Data is then sent on the DIN line, each bit valid on the rising edge of the clock
- Once the message is complete, Chip Select goes high

Sample Data Communicaitons:



note: Each command or data must contain 8 bits

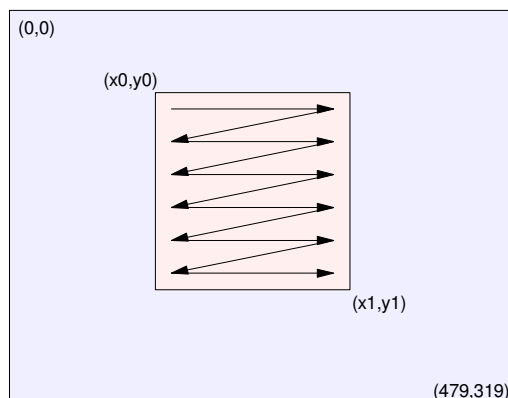
Abbreviated Instrution Set: ST7796S

There are dozens of commands you can send to the ST7796S driver. A brief list of some used in the LCD.py library as follows. For a complete set of commands, please refer to the ST7796S data sheets.

Command	Description
0x01	Software Reset
0x11	Sleep Out (Wake Up)
0x3A	Set color mode to 16 bits per pixel
0x29	Display On
0x21	Inversion Off
0x2A aa bb	Set column range to [aa, bb]
0x2B aa bb	Set row range to [aa, bb]
0x2C xx xx xx xx xx xx	Memory Write

Note on Set Column / Row Range

The *set-column-range* and *set-row-range* commands allow you to define a window on the LCD display. The following bytes will then be written into that window, starting at the upper-left corner and ending up in the lower-right corner.



When writing to the LCD display using *memory write*, a window from (x0,y0) to (x1,y1) is written to as 1st row, 2nd row, etc

If you use the *Memory_Write* command along with the SPI port and a byte-array to store the data, data can be written to the LCD display at 40MHz (!). What this means is:

- Filling in rectangles can be very fast.
- Drawing horizontal or vertical lines can be very fast (these are treated as rectangles with a depth or width of one).

However, diagonal lines, circles, and other shapes are much slower. These need to be filled in pixel-by-pixel. This shows up in the execution times:

- A 300 pixel long horizontal line takes 3.00ms to draw
- A 300 pixel long diagonal line takes 179.9ms to draw
- A 50x50 square takes 1.6ms to draw
- A circle with radius of 50 takes 201ms to draw

To speed up execution time, text is written to the display using the *set column range* commands as well. For each character, an 8x16 rectangle is defined. The bit-map for the character is then written into a byte-array. Once written, the entire array is written to the LCD using the *memory write* command at 40MHz on the SPI bus.

Just so you know.

Color Encoding with the ST7796S:

Note: There are two ways to encode RGB colors with this display

- 6-bit color (uses 3 bytes)
- 5/6/5-bit color (uses 2 bytes)

The LCD driver library uses the latter to speed up the display routines (makes them 33% faster). With this format, the color of each pixel is stored in 16 bits as

- first 5 bits: red
- Next 6 bits: green
- Last 5 bits: blue

color (16 bit variable)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
red brightness					green brightness					blue brightness					
0x1F = 100%, 0x00 = 0%					0x3F = 100%, 0x00 = 0%					0x1F = 100%, 0x00 = 0%					

The routine `LCD.RGB(r,g,b)` converts a desired red / green / blue intensity into this format.

The net result is the LCD is able to display 65,536 different colors with this format (16 bits). Each element of the display can be addressed separately, with point (0,0) in the upper-left corner and (479,319) in the lower right corner

LCD Library Summary

The following is a list of commands in the LCD library as of March 19, 2024. A more detailed description of these routines follows:

Command	Description
<code>import LCD</code>	Gain access to the LCD library functions
<code>LCD.Help()</code>	get a list of commands for the LCD display
<code>LCD.Init()</code>	Initialize the LCD display to 480x320
<code>color = LCD.RGB(r,g,b)</code>	Generate a 16-bit number for a given color
<code>LCD.Clear(color)</code>	Clear the LCD display. Set the color of each pixel
<code>LCD.Pixel(x,y,color)</code>	Set the color of pixel located at (x,y)
<code>LCD.Pixel2(x,y,color)</code>	Set the color of a 2x2 box at (x,y)
<code>LCD.Line(x0,y0,x1,y1,color)</code>	Draw a line from (x0,y0) to (x1,y1)
<code>LCD.Box(x0,y0,x1,y1,color)</code>	Draw a box
<code>LCD.Solid_Box(x0,y0,x1,y1,c)</code>	Draw a solid box
<code>LCD.Light(OnOff, x, y, color)</code>	Draw a 10x10 box at location (x,y) OnOff = 1: solid box (LED on) OnOff = 0: hollow box (LED off)
<code>LCD.Binary_Out(N, x, y)</code>	Draw 16 boxes in a row starting at (x,y) The binary value of N determines which boxes are on and off
<code>LCD.Bar_Out(N, x, y)</code>	Draw 16 boxes in a row starting at (x,y) Box #N is turned on, the rest are off
<code>LCD.Dice(N, x, y, c1, c0)</code>	Draw a 6-sided die at (x,y) Value = N (1..6) c1 = color of pips, c0 = color of background
<code>LCD.Card(Value, Suit, x, y)</code>	Draw a playing card at (x,y) Value = 0..12 (Ace to King) Suit = 0..3 (club, diamond, hearts, spades)
<code>Y = LCD.Sort(X)</code>	Sort vector X Return the order of the elements of X
<code>Deck = LCD.Shuffle()</code>	Generate a 52-element matrix with the numbers 0..51 in random order (shuffle a deck of 52 cards)

LCD Font Routines

Command	Description
LCD.Text (Msg, x, y, c1, c0)	Place text on the LCD with scaling = 1 location = (x,1) c1 = text color, c0 = background color Text uses a 16x8 character font
LCD.Text2 (Msg, x, y, c1, c0)	Same as LCD.Text but uses an 8x16 font scaled by 2x
LCD.Text3 (Msg, x, y, c1, c0)	Same as LCD.Text but uses a 16x24 font (uses up 31k of memory - only included in LCD_16x24.py)
LCD.Text4 (Msg, x, y, c1, c0)	Same as LCD.Text but uses a 16x24 font, scaled by 2x
LCD.Text5 (Msg, x, y, c1, c0)	Same as LCD.Text but uses a 24x32 font (uses up 51k of memory - only included in LCD_24x32.py)
LCD.Number (N, a, b, x, y, c1, c0)	Place number N on the LCD, scaling = 1 fixed decimal format a digits, b decimal places location is (x,y) c1 = text color, c0 = background color Digits use an 8x16 character format
LCD.Number2 (N, a, b, x, y, c1, c0)	LCD Number using an 8x16 font scaled 2x
LCD.Number3 (N, a, b, x, y, c1, c0)	LCD Number using a 16x24 font
LCD.Number4 (N, a, b, x, y, c1, c0)	LCD Number using a 16x24 font scaled 2x
LCD.Number5 (N, a, b, x, y, c1, c0)	LCD Number using a 24x32 font

LCD Library Instructions (greater detail)

LCD.Init()

Initialize the LCD display to

- 480 x 320 resolution
- Non-inversion (0,0,0 = black, 255,255,255 = white)
- Display On

LCD.Clear(color)

Sample Code:

```
Navy = LCD.RGB(0, 0, 50)
LCD.Clear(Navy)
```

Execution Time: 116ms

Clear the display. Set the color of each pixel to *color*

color = LCD.RGB(r, g, b)

Create a 16-bit color for the LCD display.

- r, g, b are the brightness of red / green / blue
- 255 is 100% on, 0 is 0% on

For example

```
Red = LCD.RGB(255, 0, 0)
Green = LCD.RGB(0, 255, 0)
Blue = LCD.RGB(0, 0, 255)
White = LCD.RGB(255, 255, 255)
Grey = LCD.RGB(100, 100, 100)
Black = LCD.RGB(0, 0, 0)
```

The resulting 16-bit value of *color* contains the values of r/g/b in a 5/6/5 format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
red					green						blue				

Box(x0, y0, x1, y1, color)

Draw a box on the LCD display

Example: Draw a pink box from (50,50) to (100,100)

```
import LCD
import time

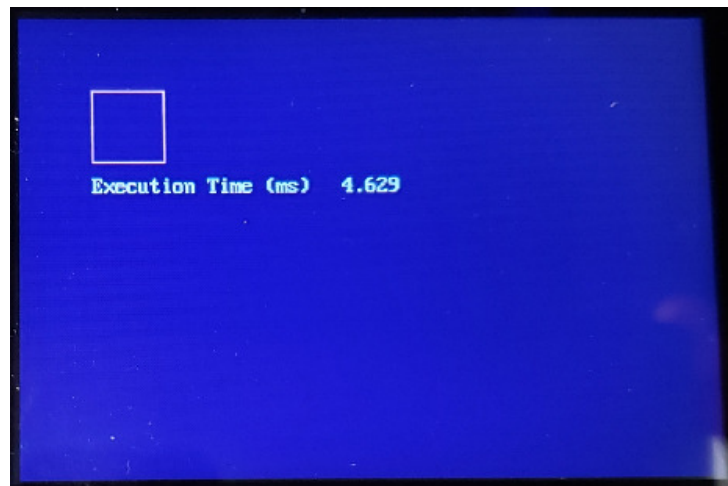
White = LCD.RGB(150,150,150)
Navy = LCD.RGB(0,0,20)
Pink = LCD.RGB(150,80,80)
LCD.Clear(Navy)

X0 = time.ticks_us()

LCD.Box(50,50,100,100,Pink)

X1 = time.ticks_us()
LCD.Text('Execution Time (ms):', 50, 110, White, Navy)
LCD.Number((X1-X0)/1000, 6, 3, 200, 110, White, Navy)
```

Execution time = 4.62ms (varies with box size)



LCD.Solid_Box(x0,y0,x1,y1,color)

Draw a solid box on the LCD display. Execution time = 15.3ms (varies with box size)

Sample Code: Draw a pink box

```
import LCD
import time

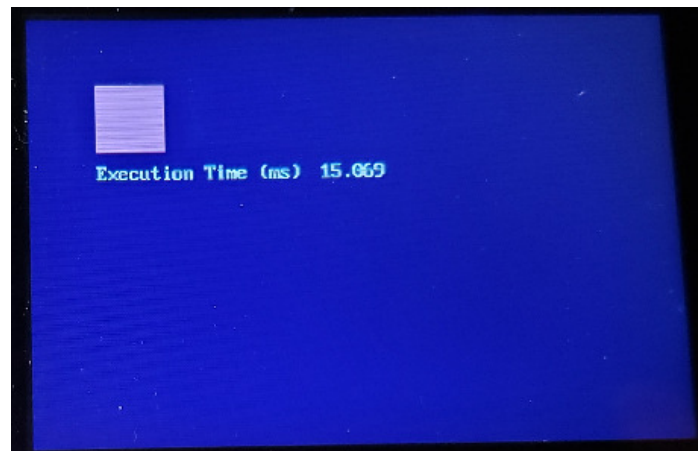
White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,20)
Pink = LCD.RGB(250,100,100)

LCD.Clear(Navy)

X0 = time.ticks_us()

LCD.Solid_Box(50,50,100,100,Pink)

X1 = time.ticks_us()
LCD.Text('Execution Time (ms):', 50, 110, White, Navy)
LCD.Number((X1-X0)/1000, 6, 3, 200, 110, White, Navy)
```



LCD.Line(x0,y0,x1,y1,color)

Draw a line from (x0,y0) to (x1,y1) of the specified color

Execution time varies:

- horizontal or vertical lines: 3.0ms
- diagonal lines: 179.9ms

Example: Draw an octagon on the display

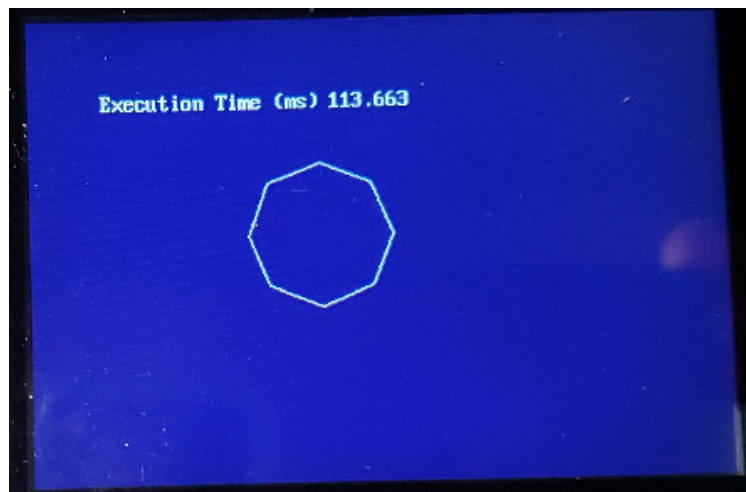
- Center = (200,160)
- Radius = 50

Code:

```
import LCD
from math import sin, cos

White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,50)

x0 = 200
y0 = 150
r = 50
pi = 3.141592654
x = []
y = []
for i in range(0,9):
    x1 = int(x0 + r*cos(i*2*pi/8))
    y1 = int(y0 + r*sin(i*2*pi/8))
    x.append(x1)
    y.append(y1)
for i in range(0,8):
    LCD.Line(x[i],y[i],x[i+1],y[i+1],White)
```



LCD.Circle(x, y, r, color)

Draws a circle

- centered at (x, y)
- radius = r
- color = color

Example: Draw a circle

- Center = (150, 150)
- Radius = 100
- Color = Pink

Code

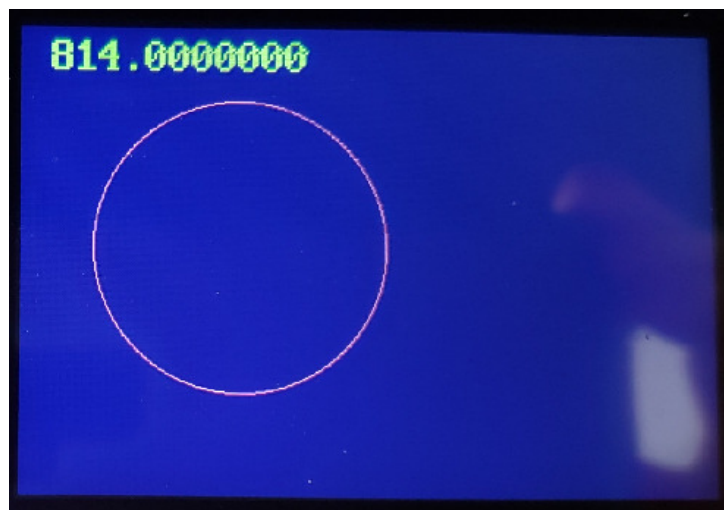
```
import LCD_16x24 as LCD
from time import ticks_us

Navy = LCD.RGB(0,0,10)
Pink = LCD.RGB(150,50,50)
Yellow = LCD.RGB(150,150,0)

LCD.Init()
LCD.Clear(Navy)
t0 = ticks_us()
LCD.Circle(150,150,100,Pink)
t1 = ticks_us()

LCD.Number4(t1-t0, 10, 7, 5, 5, Yellow, Navy)
```

Results:



LCD.Text(Message, x, y, color1, color0)

- Display the string Message on the LCD

LCD.Number(N, A, B, x, y, color1, color0)

- Display the number N on the LCD with A digits and B decimal places
- color1= text color
- color2 = background color

Both use a 16x8 font with fixed character spacing

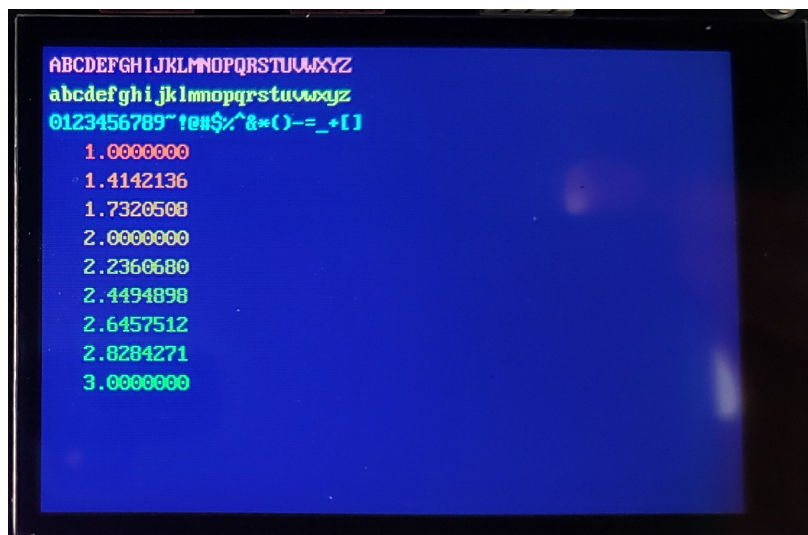
Example:

```
import LCD

Navy = LCD.RGB(0,0,10)
Pink = LCD.RGB(150,50,50)
Yellow = LCD.RGB(150,150,0)
Cyan = LCD.RGB(0,150,150)

LCD.Init()
LCD.Clear(Navy)
LCD.Text('ABCDEFGHIJKLMNOPQRSTUVWXYZ',5,5,Pink,Navy)
LCD.Text('abcdefghijklmnopqrstuvwxyz',5,25,Yellow,Navy)
LCD.Text('0123456789~!@#$%^&*()-=_+[]',5,45,Cyan,Navy)

for i in range(1,10):
    x = i ** 0.5
    LCD.Number(x, 10, 7, 5, 45+20*i, LCD.RGB(150-15*i,15*i,0), Navy)
```



LCD.Text2(Message, x, y, color1, color0)

- Display the string Message on the LCD using a 16x32 font (8x16 scaled 2x)

LCD.Number2(N, A, B, x, y, color1, color0)

- Display the number N on the LCD with A digits and B decimal places
- color1= text color
- color2 = background color

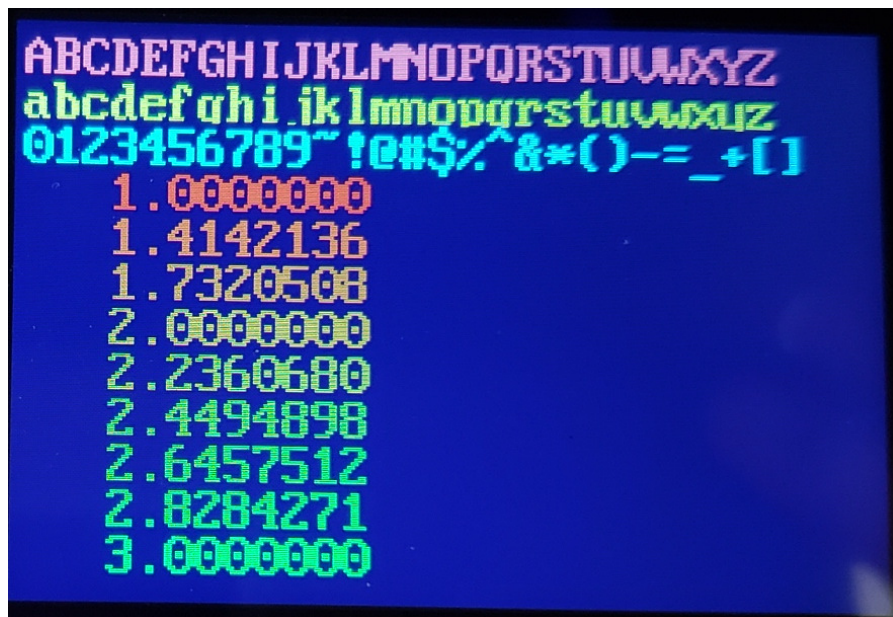
Example:

```
import LCD

Navy = LCD.RGB(0,0,10)
Pink = LCD.RGB(150,50,50)
Yellow = LCD.RGB(150,150,0)
Cyan = LCD.RGB(0,150,150)

LCD.Init()
LCD.Clear(Navy)
LCD.Text2('ABCDEFGHIJKLMNOPQRSTUVWXYZ',5,5,Pink,Navy)
LCD.Text2('abcdefghijklmnopqrstuvwxyz',5,25,Yellow,Navy)
LCD.Text2('0123456789~!@#$/^&*()-=_+[]',5,45,Cyan,Navy)

for i in range(1,10):
    x = i ** 0.5
    LCD.Number2(x, 10, 7, 5, 45+20*i, LCD.RGB(150-15*i,15*i,0), Navy)
```



LCD.Text3(Message, x, y, color1, color0)

- Display the string Message on the LCD using a 16x24 font

LCD.Number3(N, A, B, x, y, color1, color0)

- Display the number N on the LCD with A digits and B decimal places
- color1= text color
- color2 = background color

Note: The 16x24 font bitmap uses up 31k of program memory

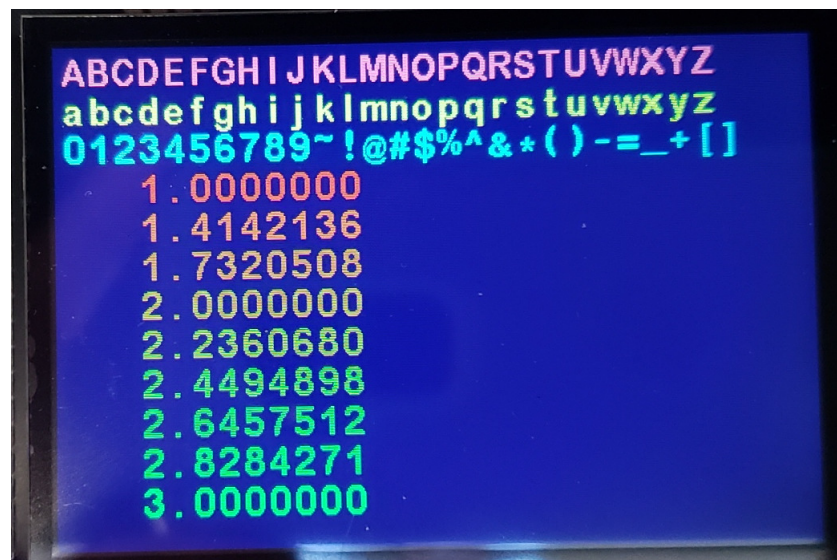
Example:

```
import LCD_16x24 as LCD

Navy = LCD.RGB(0,0,10)
Pink = LCD.RGB(150,50,50)
Yellow = LCD.RGB(150,150,0)
Cyan = LCD.RGB(0,150,150)

LCD.Init()
LCD.Clear(Navy)
LCD.Text3('ABCDEFGHIJKLMNOPQRSTUVWXYZ',5,5,Pink,Navy)
LCD.Text3('abcdefghijklmnopqrstuvwxyz',5,25,Yellow,Navy)
LCD.Text3('0123456789~!@#$%^&*()-=_+[]',5,45,Cyan,Navy)

for i in range(1,10):
    x = i ** 0.5
    LCD.Number3(x, 10, 7, 5, 45+20*i, LCD.RGB(150-15*i,15*i,0), Navy)
```



LCD.Text4(Message, x, y, color1, color0)

- Display the string Message on the LCD using a 24x32 character font

LCD.Number4(N, A, B, x, y, color1, color0)

- Display the number N on the LCD with A digits and B decimal places
- color1= text color
- color2 = background color

Note: 24x32 font uses up 62k of program memory just for the font bitmap

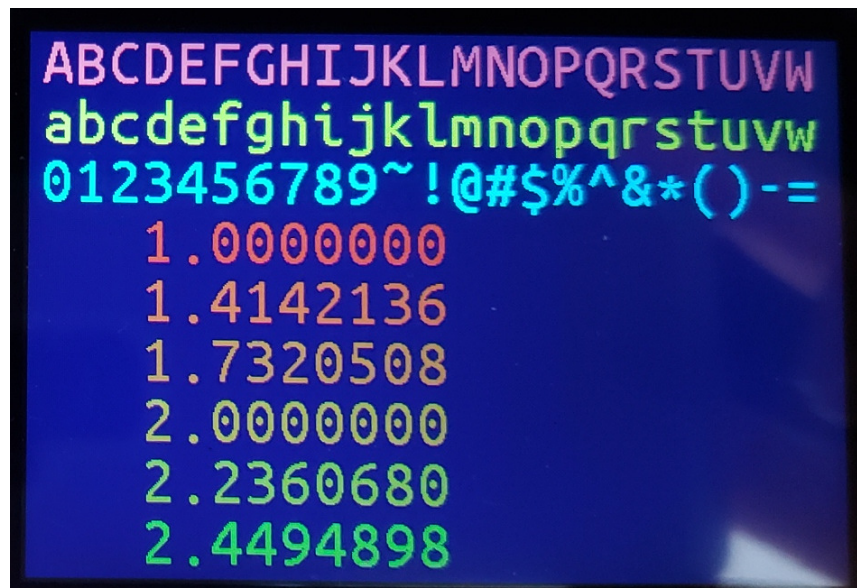
Example:

```
import LCD_24x32 as LCD

Navy = LCD.RGB(0,0,10)
Pink = LCD.RGB(150,50,50)
Yellow = LCD.RGB(150,150,0)
Cyan = LCD.RGB(0,150,150)

LCD.Init()
LCD.Clear(Navy)
LCD.Text4('ABCDEFGHIJKLMNOPQRSTUVWXYZ',5,5,Pink,Navy)
LCD.Text4('abcdefghijklmnopqrstuvwxyz',5,25,Yellow,Navy)
LCD.Text4('0123456789~!@#$%^&*()-=_+',5,45,Cyan,Navy)

for i in range(1,10):
    x = i ** 0.5
    LCD.Number4(x, 10, 7, 5, 45+20*i, LCD.RGB(150-15*i,15*i,0), Navy)
```



LCD.Pixel(x, y, color)

Set the color of the pixel at (x,y).

LCD.Pixel2(x,y,color)

Set the color of a 2x2 set of pixels at (x,y).

Example: Draw a circle with a radius of 50 centered at (200,150). Note, the circumference is 314 pixels, so draw 314 points at radius r

```
import LCD
from math import sin, cos
import time

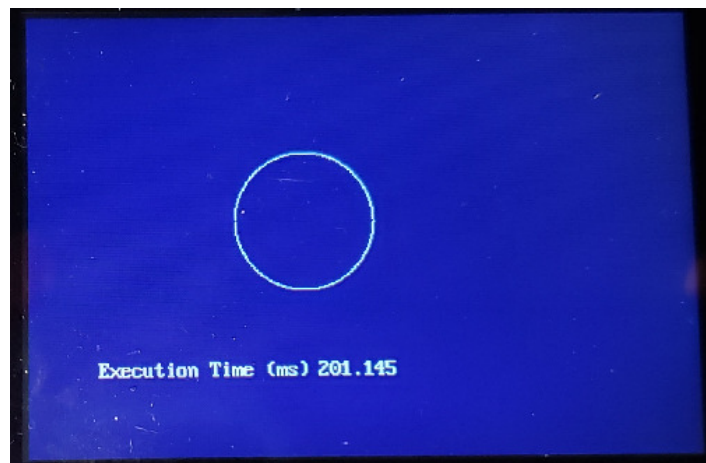
White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,50)

x0 = 200
y0 = 150
r = 50
pi = 3.141592654
x = []
y = []

X0 = time.ticks_us()

for i in range(0,314):
    x = int(x0 + r*cos(i*2*pi/314))
    y = int(y0 + r*sin(i*2*pi/314))
    LCD.Pixel(x, y, White)

X1 = time.ticks_us()
LCD.Text('Execution Time (ms):', 50, 110, White, Navy)
LCD.Number((X1-X0)/1000, 6, 3, 200, 110, White, Navy)
```



LCD.Light(OnOff, x, y, color)

Display a 10x10 box (simulating an LED on the screen) at location (x,y) that's

- A solid box (OnOff = 1), or
- A hollow box (OnOff = 0)

color sets the color of the box. This routine is used in Binary_Out() and Bar_Out()

LCD.Binary_Out(N, x, y)

Display 16 boxes on the screen from left to right to simulate 16 LED lights.

- Turn on LEDs corresponding to the binary value of N
- Total display area is 250 x 10

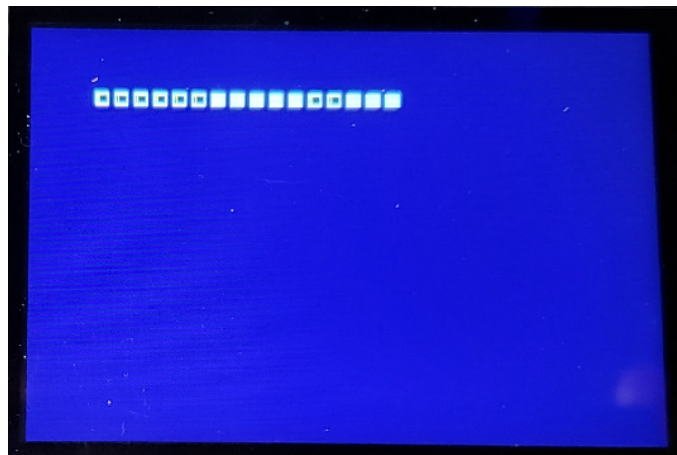
Example: Count in binary

```
import LCD
import time

White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,50)

LCD.Clear(Navy)

for i in range(0, 1000):
    LCD.Binary_Out(i, 50, 50)
    time.sleep(0.001)
```



16 element LED showing the number 0x03E7 = 999

LCD.Bar_Out(N, x, y)

Display 16 boxes on the screen from left to right to simulate 16 LED lights.

- Turn on LED #N and turn the rest off
- Total display area is 250 x 10

Example: Make a light that bounces back and forth

```
import LCD
import time

White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,50)

LCD.Clear(Navy)

dX = 1
X = 0
while(1):
    if(X > 15):
        dX = -1
    if(X < 2):
        dX = 1
    X += dX
    LCD.Bar_Out(X, 50, 50)
    time.sleep(0.1)
```

```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ N = 0
□ □ □ □ □ □ □ □ □ □ □ □ □ □ ■ N = 1
□ □ □ □ □ □ □ □ □ □ □ □ □ ■ □ N = 2
□ □ □ □ □ □ □ □ □ □ □ □ ■ □ □ N = 3
□ □ □ □ □ □ □ □ □ □ □ ■ □ □ □ N = 4
□ □ □ □ □ □ □ □ □ □ ■ □ □ □ □ N = 5
□ □ □ □ □ □ □ □ □ ■ □ □ □ □ □ N = 6
□ □ □ □ □ □ □ □ ■ □ □ □ □ □ □ N = 7
□ □ □ □ □ □ □ ■ □ □ □ □ □ □ □ N = 8
□ □ □ □ □ □ ■ □ □ □ □ □ □ □ □ N = 9
```

LCD.Dice(N, x, y, color1, color0)

Display a 6-sided die at location (x,y)

- The die is 50x50, with the upper left corner at (x,y)
- color1 is the color of the pips
- color2 is the color of the background

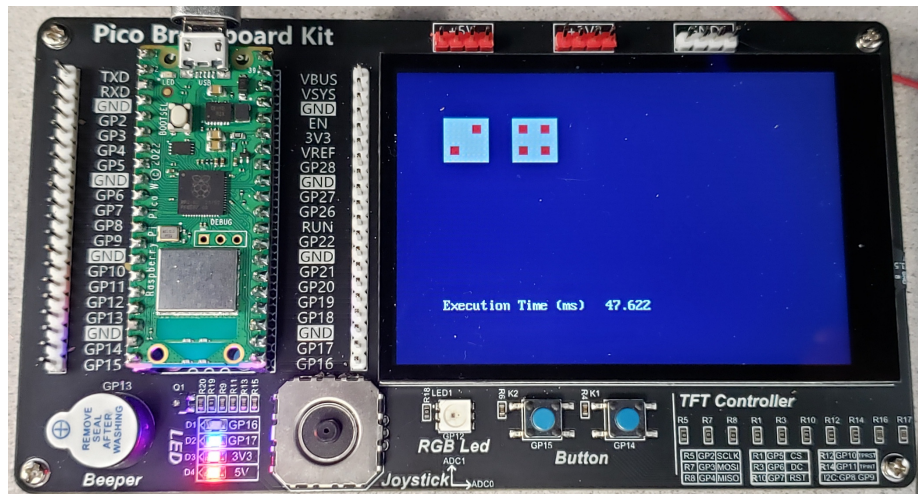
Example: Roll two 6-sided dice and display them on the LCD

```
import LCD
import random

White = LCD.RGB(250,250,250)
Navy = LCD.RGB(0,0,50)

LCD.Clear(Navy)

d1 = random.randrange(1,6)
d2 = random.randrange(1,6)
LCD.Dice(d1,50,50,Red,White)
LCD.Dice(d2,125,50,Red,White)
```



LCD.Card(Value, Suit, x, y)

Display a playing card at location (x,y)

- The playing card is 45 x 65 in size
- Value is 1..13 (1 = Ace, 2 = two, 11 = jack, 12 = queen, 13 = king)
- Suit is 1..4 (1 = club, 2 = diamond, 3 = heart, 4 = spade)

$$Y = \text{LCD.Sort}(X)$$

Return an array which shows the sort order of array X, sorted smallest to largest

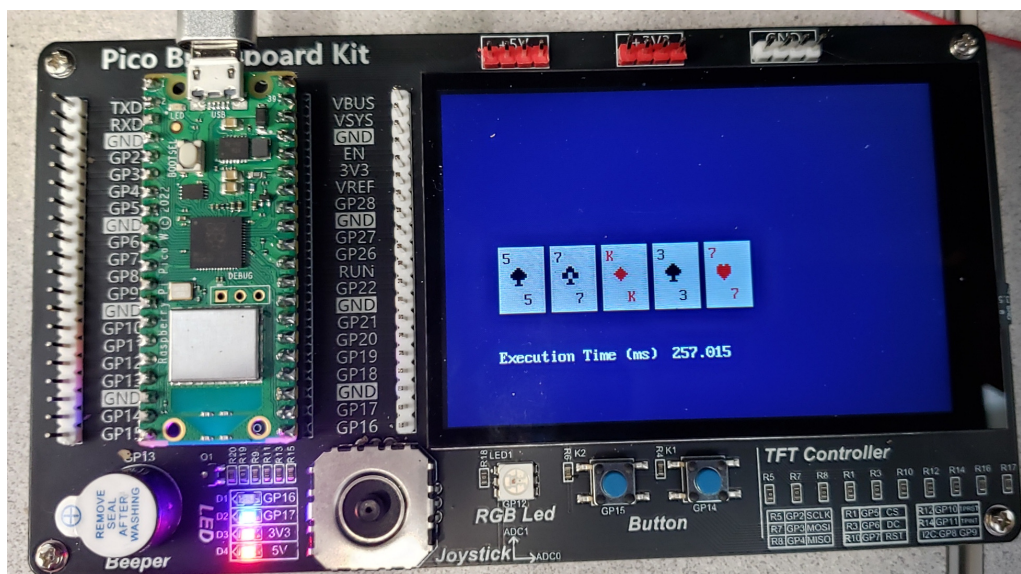
Deck = LCD.Shuffle()

Return a 52-element array of numbers 0..51 in random order. Used to shuffle a deck of 52 playing cards.

Example: Shuffle a deck, draw the first five cards, and display them on the LCD

```
import LCD
import random

Hand = [0,0,0,0,0]
Value = [0,0,0,0,0]
Suit = [0,0,0,0,0]
X = LCD.Shuffle()
for i in range(0,5):
    Hand[i] = Y[i]
    Value[i] = (Hand[i] % 13) + 1
    Suit[i] = int(Hand[i] / 13) + 1
    LCD.Card(Value[i], Suit[i], 50+i*50, 150)
```



References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- <https://learn.pimoroni.com/article/getting-started-with-pico>
- <https://www.w3schools.com/python/default.asp>
- <https://docs.micropython.org/en/latest/pyboard/tutorial/index.html>
- <https://docs.micropython.org/en/latest/library/index.html>
- <https://www.fredscave.com/02-about.html>

Pi-Pico Breadboard Kit

- <https://wiki.52pi.com/index.php?title=EP-0172>

Other

- <https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/>
- <https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/>
- <https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/>
- <https://randomnerdtutorials.com/projects-raspberry-pi-pico/>
- <https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/>