LCD Graphics Display

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #12

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

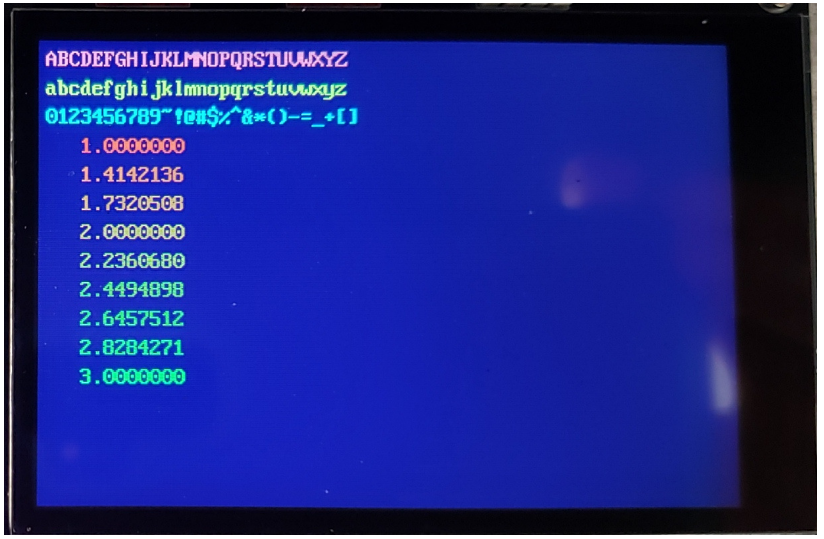
Introduction:

This lecture introduces

- The 480 x 320 graphics display
- The LCD.py library

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

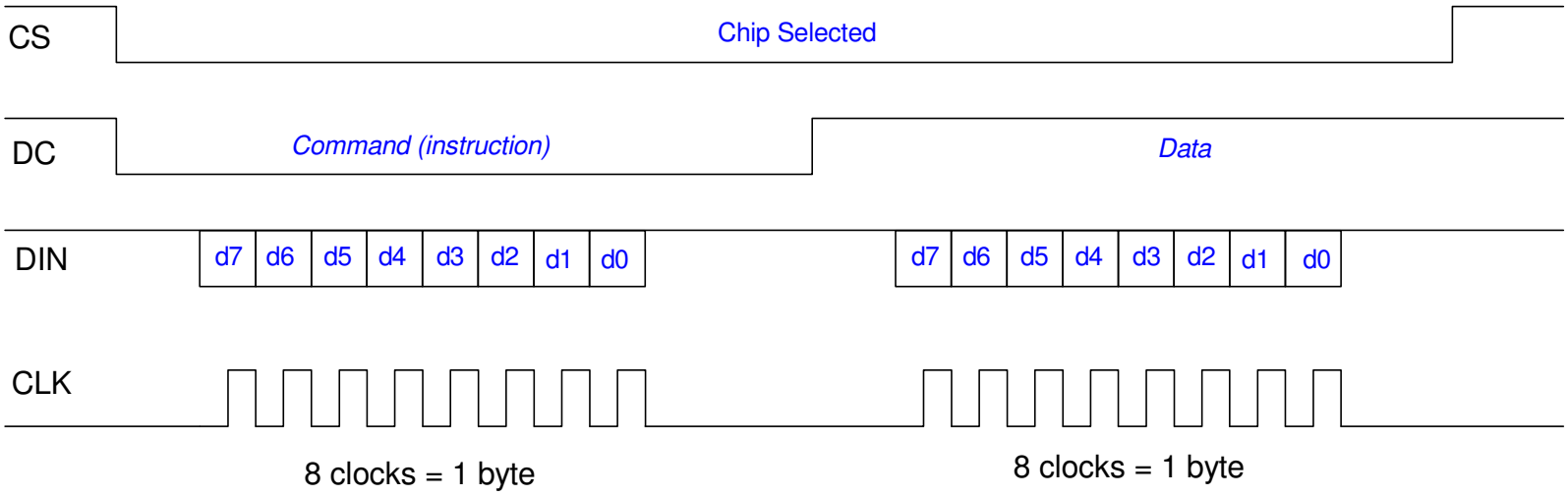
Driver Chip: ST77965

- Hard-wired to your Pico using pins 2..7:
  - Don't use for I/O if you're using the LCD display
- Pins 2..4 (SPI CLK, DIN, DOUT) can be used with other devices
  - SPI communications
  - Give them a different chip-select

Pin	ST77865 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

SPI Communications

- 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
  - note: Each command or data must contain 8 bits
- Once the message is complete, Chip Select goes high



Abbreviated Instrution Set: ST77965

The ST77965 has dozens of commands

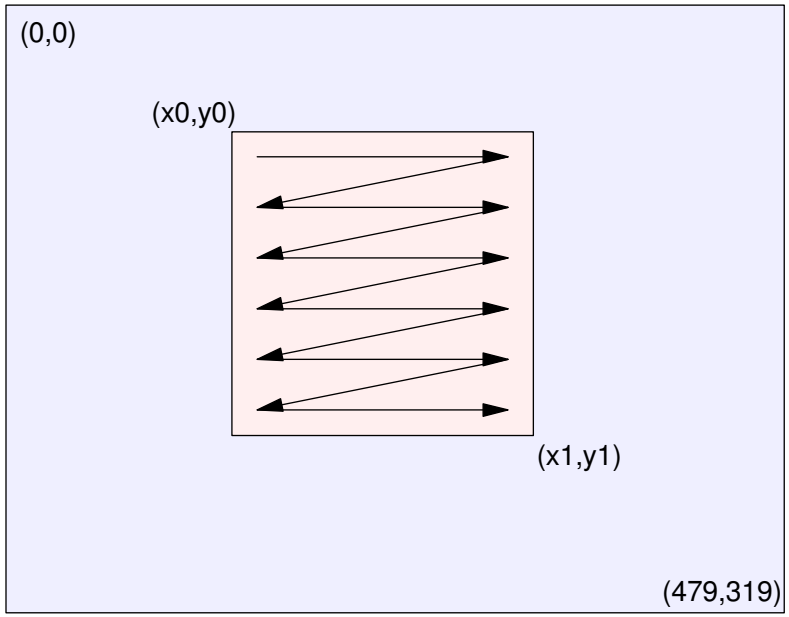
- See the data sheets for a complete list

Some of the main ones used in the LCD.py library are:

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

- 0x2A y0 y1: Set Column Range
- 0x2B x0 x1: Set Row Range



These allow you to define a window

- Rectangle from (x0,y0) to (x1,y1)

Sunsequent *Memory Write* commands fill in the rectangle

- Fill in left to right
- Can write bytes at 40MHz (!)

Meaning...

- Horizonal or vertical lines are fast
- Diagonal lines are slow
    - Have to go pixel by pixel

Execution Times

300 pixel long horizontal line

- 3.00ms

300 pixel long diagonal line

- 179.9ms

50x50 square

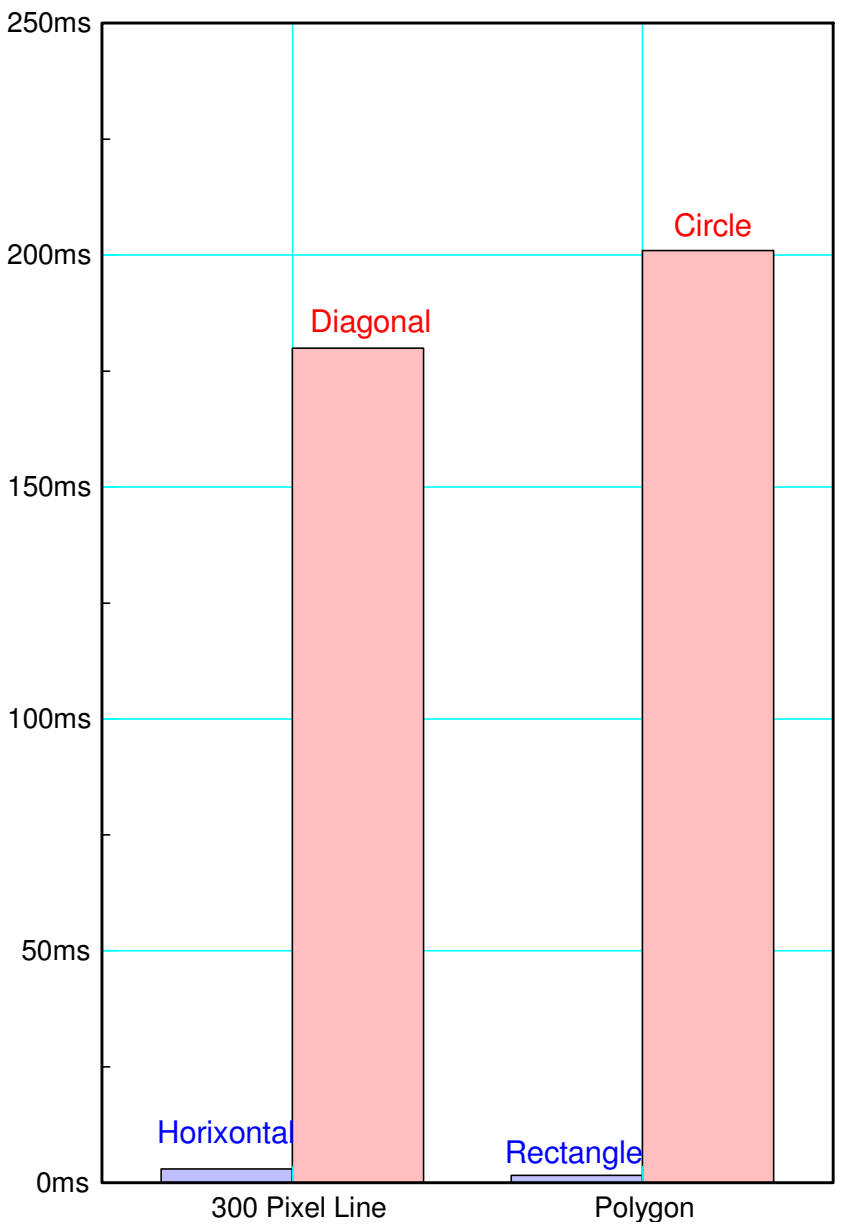
- 1.6ms to draw

A circle with radius of 50

- 201ms

Meaning:

- Try to stick to horizonal and vertical lines if possible



Writing Text

Text uses the *set range* commands

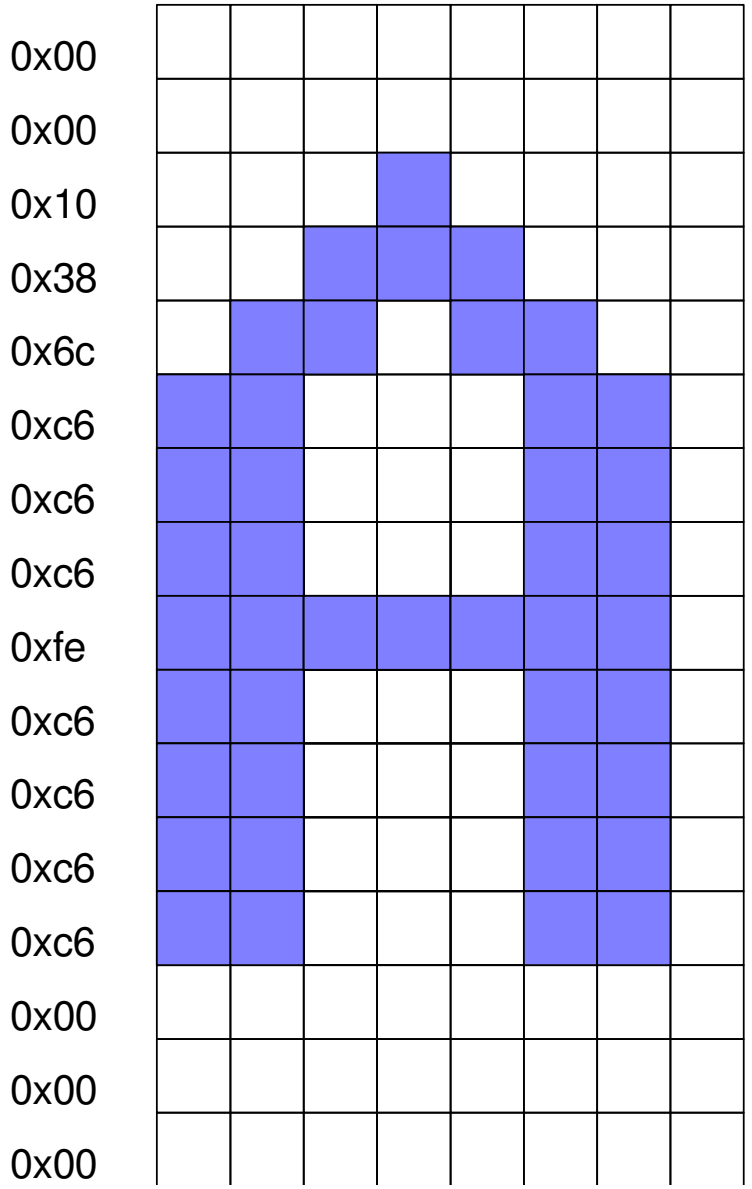
For 8x16 characters

- An 8x16 rectangle is defined
- A table defines each pixel (on/off)
  - defined in the start of LCD.py
- *Memory Write* sends data at 40MHz
  - speeds up execution time

Different table = Different fonts

- 8x16 fonts (standard)
  - takes up 10k of RAM
- 16x24 fonts
  - in file LCD_16x24.py
  - takes up an additional 31k
- 24x32 fonts
  - in file LCD_24x32.py
  - takes up 62k more

Just so you know.



Color Encoding with the ST77965:

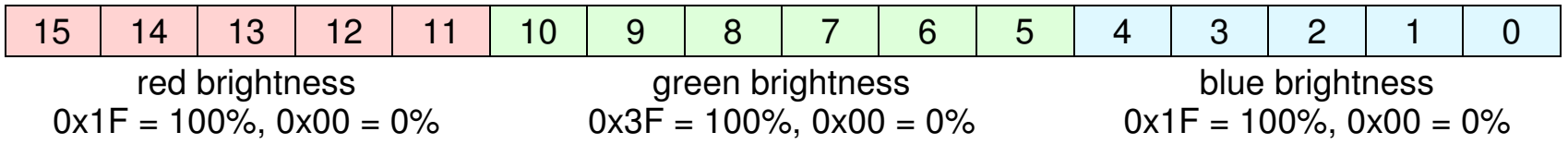
Note: There are two ways to encode RGB colors with this display

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

The LCD driver library uses the latter

- makes routines 33% faster
- Color is stored in 16-bits
  - First 5 bits: red
  - Next 6 bits: green
  - Last 5 bits: blue
- Net results = 65,536 colors avilailable

color (16 bit variable)



LCD Library Summary

- Routines as of March 19, 2024
- A more detailed description of these routines follows:

Command	Description
`import LCD`	Gain access to the LCD library functions
`LCD.Help()`	get a list of commands for the LCD display
`LCD.Init()`	Initialize the LCD display to 480x320
`color = LCD.RGB(r,g,b)`	Generate a 16-bit number for a given color
`LCD.Clear(color)`	Clear the LCD display. Set the color of each pixel
`LCD.Pixel(x,y,color)`	Set the color of pixel located at (x,y)
`LCD.Pixel2(x,y,color)`	Set the color of a 2x2 box at (x,y)
`LCD.Line(x0,y0,x1,y1,color)`	Draw a line from (x0,y0) to (x1,y1)
`LCD.Box(x0,y0,x1,y1,color)`	Draw a box
`LCD.Solid_Box(x0,y0,x1,y1,c)`	Draw a solid box
`LCD.Light(OnOff, x, y, color)`	Draw a 10x10 box at location (x,y) OnOff = 1: solid box (LED on) OnOff = 0: hollow box (LED off)
`LCD.Binary_Out(N, x, y)`	Draw 16 boxes in a row starting at (x,y) The binary value of N determines which boxes are on and off
`LCD.Bar_Out(N, x, y)`	Draw 16 boxes in a row starting at (x,y) Box #N is turned on, the rest are off

Command	Description
`LCD.Dice(N, x, y, c1, c0)`	Draw a 6-sided die at (x,y) Value = N (1..6) c1 = color of pips, c0 = color of background
`LCD.Card(Value, Suit, x, y)`	Draw a playing card at (x,y) Value = 0..12 (Ace to King) Suit = 0..3 (club, diamond, hearts, spades)
`Y = LCD.Sort(X)`	Sort vector X Return the order of the elements of X
`Deck = LCD.Shuffle()`	Generate a 52-element matrix with the numbers 0..51 in random order (shuffle a deck of 52 cards)
`Bar(X)`	Draw a bar graph for X scaled with the maximum of X Similar to Matlab's command bar()
`plot(x,y)`	Draw an axis and plot data x vs. y Similar to Matlab's command plot()
`Title(String, c1, c0)`	Add a title to your graph. Location is fixed: top of display an centered. c1 = text color, c0 = background color

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 (only included in LCD_16x24 and LCD_24x32)
LCD.Text4(Msg,x,y,c1,c0)	Same as LCD.Text but uses a 24x32 font (only included in LCD_24x32)
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 24x32 font *better looking font but uses up a LOT of memory*

LCD.Init()

Initialize the LCD display (only needs to be called once)

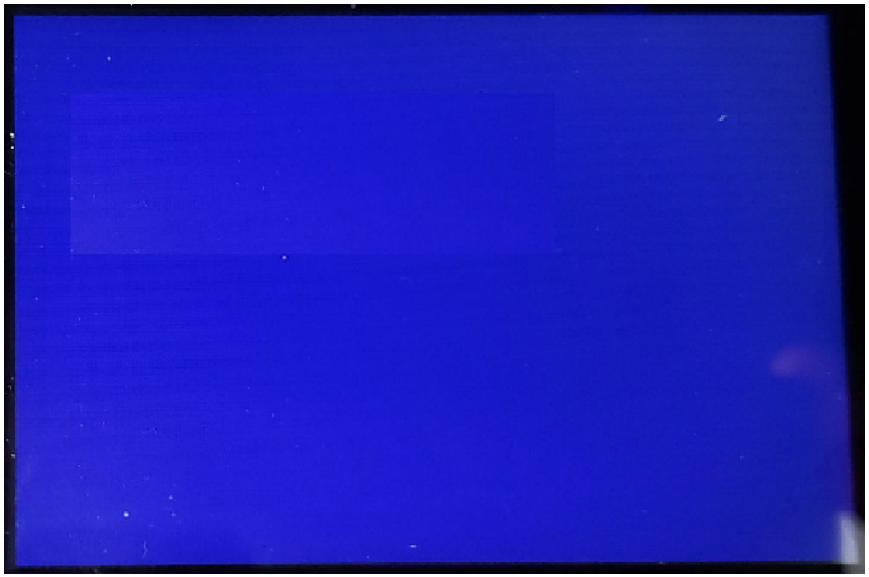
- 480 x 320 resolution
- Non-inversion
  - (0,0,0) = black
  - (255,255,255) = white
- Display On

LCD.Clear(color)

- Clear the display.
- Set the color of each pixel
- Execution Time: 116ms

Sample Code:

```
import LCD

Navy = LCD.RGB(0,0,20)
LCD.Init()
LCD.Clear(Navy)
```



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

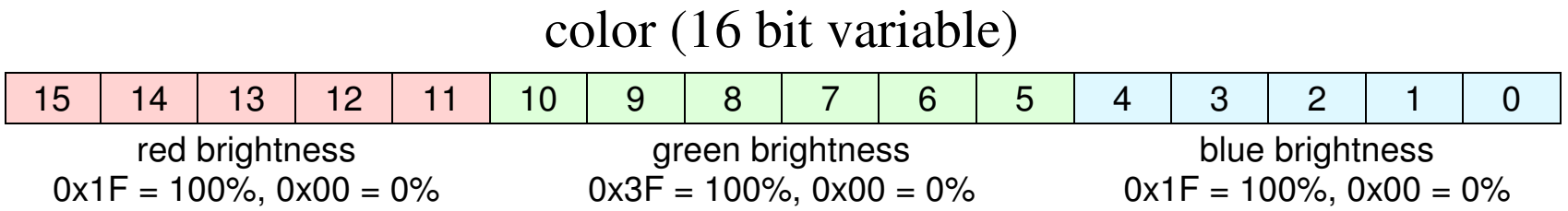
Create a 16-bit color for the LCD display.

- 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 result is a 16-bit number in 5/6/5 format for r/g/b:

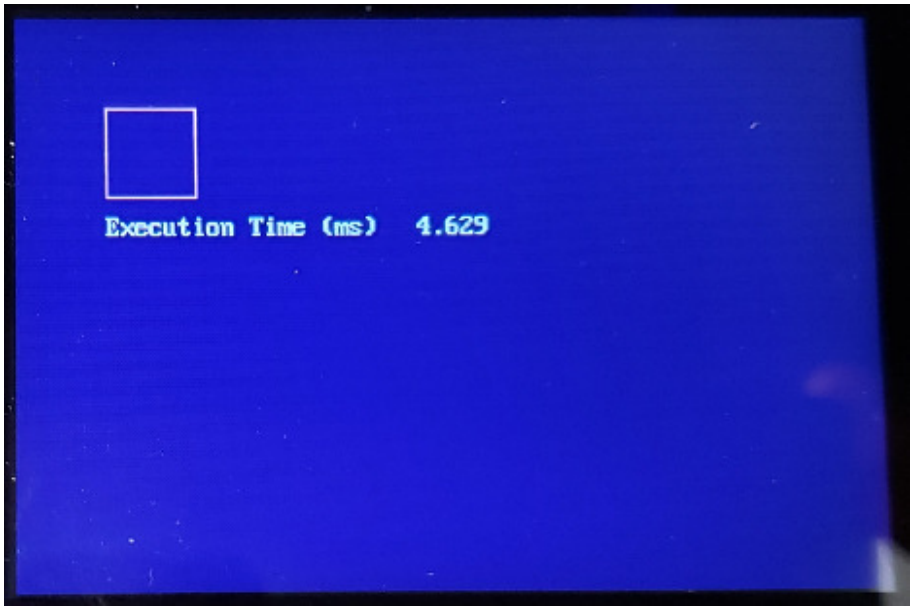


Box(x0, y0, x1, y1, color)

- Draw a box on the LCD display
- Execution time = 4.62ms (varies)

Example:

- Draw a pink box
  - (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)
```

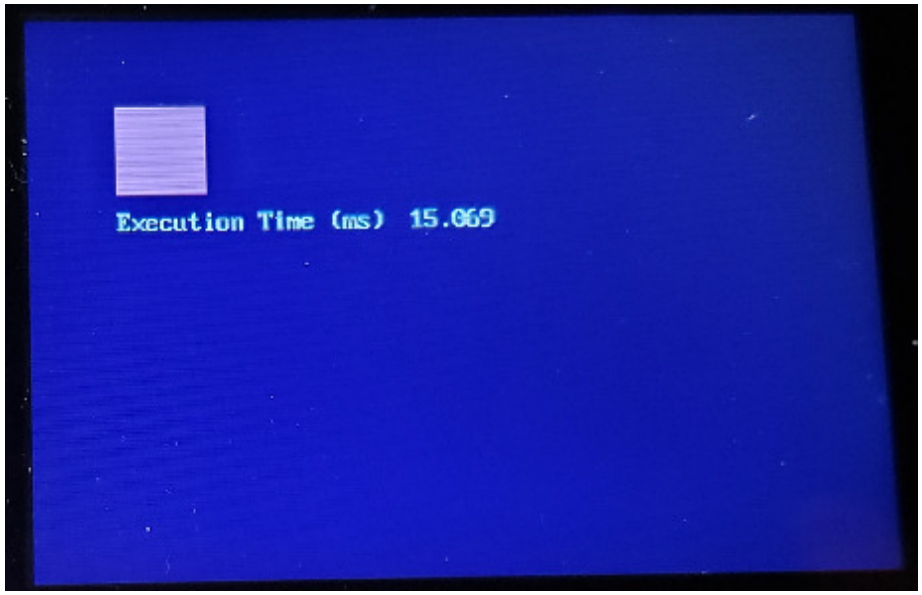
LCD.Solid_Box(x0,y0,x1,y1,color)

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

Sample Code:

- Draw a solid 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)
```

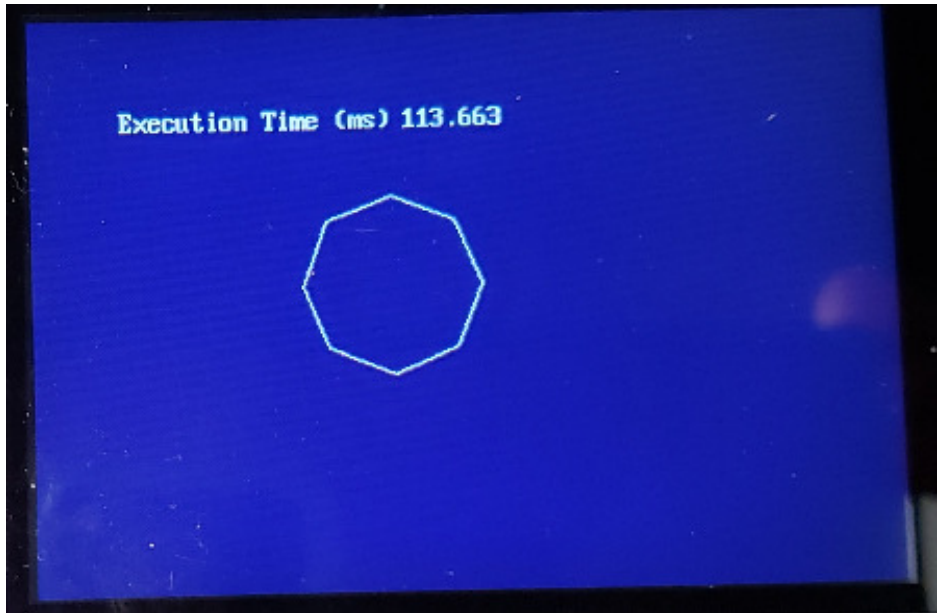


LCD.Line(x0,y0,x1,y1,color)

- Draw a line from (x0,y0) to (x1,y1) of the specified color
- Execution time varies:
  - horizontal of vertical lines: 3.0ms
  - diagonal lines: 179.9ms

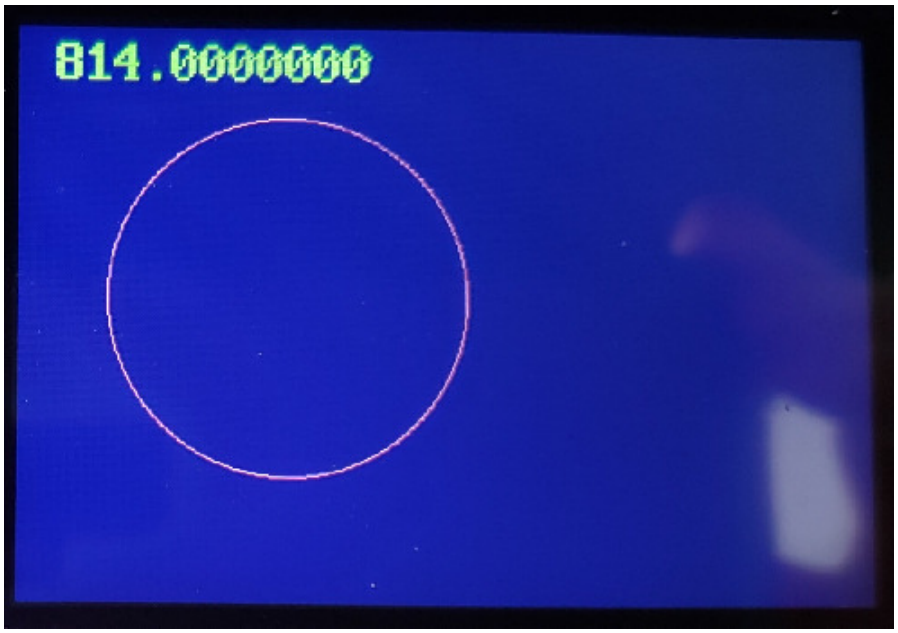
Example: Draw a an octagon on the display

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

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

x0 = 200
y0 = 150
r = 50
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



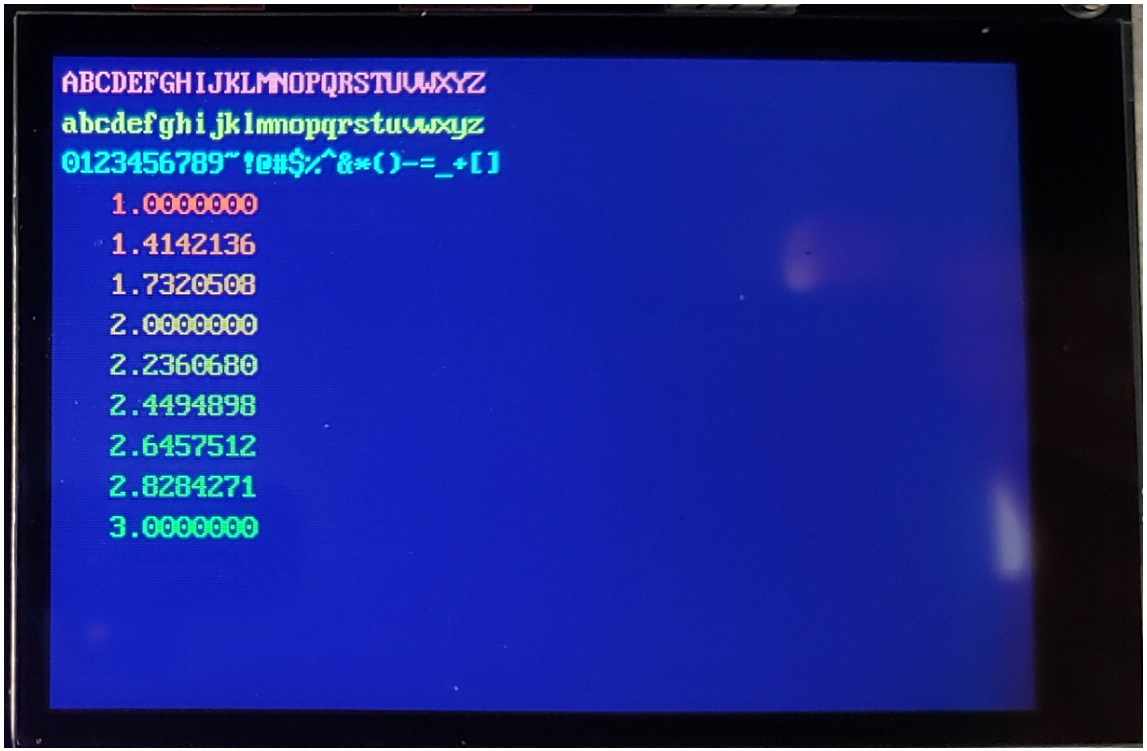
Code: Pink circle with radious 100

```
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)
```

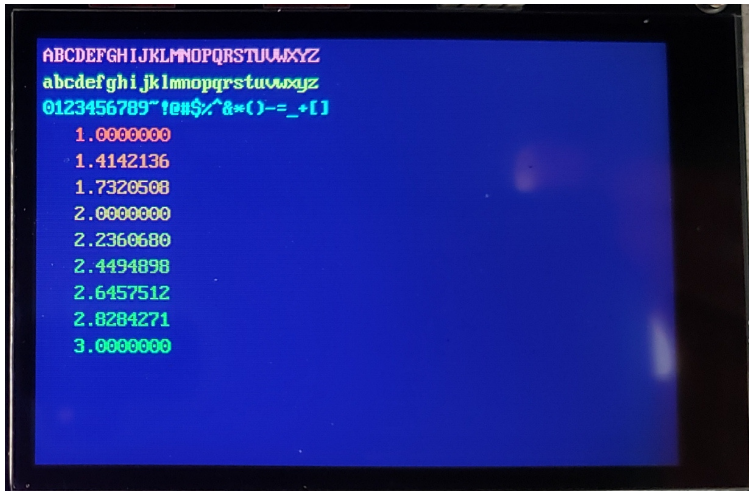
LCD.Text(Message, x, y, color1, color0)

LCD.Number(number, N, M, x, y, color1, color0)

- Display text / number on the LCD starting at position (x,y)
- Numbers have N digits, M decimal places
- color1 = text color, color0 = background color



Sample Code:

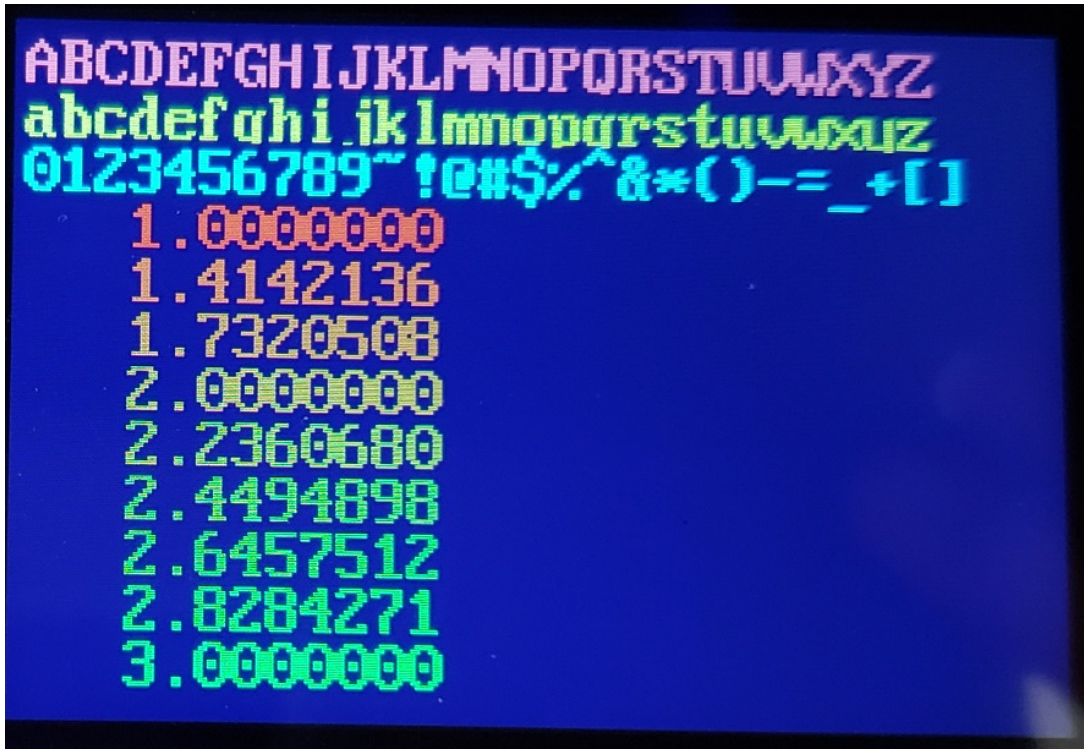



```
import LCD

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)
```

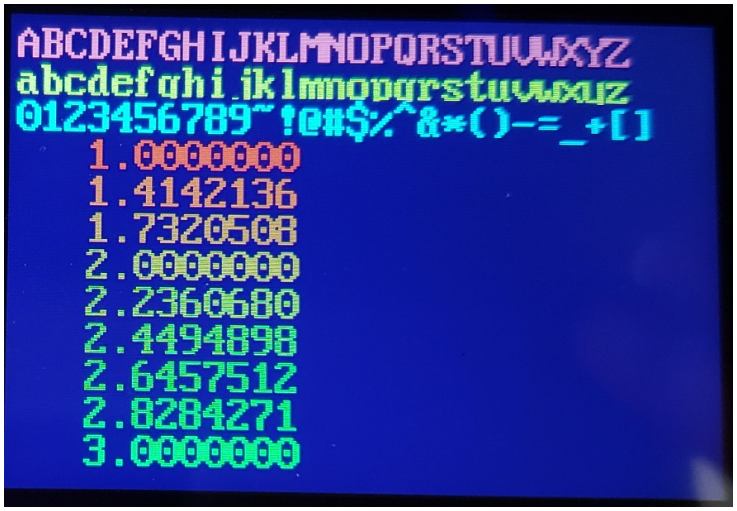
LCD.Text2(Message, x, y, color1, color0)

LCD.Number2(number, N, M, x, y, color1, color0)

- Display text / number on the LCD
- Uses 8x16 font scaled 2x (net is 16x32 font)



Sample Code

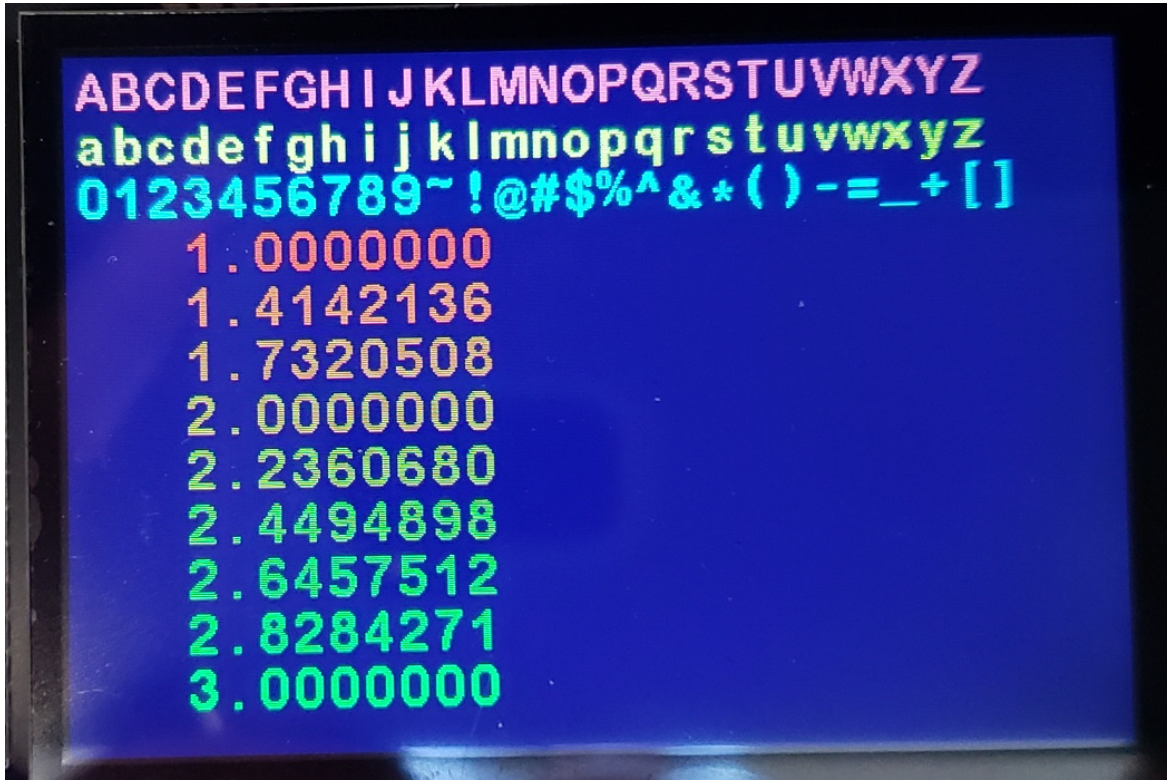



```
import LCD

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)

LCD.Number3(number, N, M, x, y, color1, color0)

- Display text / number on the LCD
- Uses 16x24 font bitmap



Sample Code:

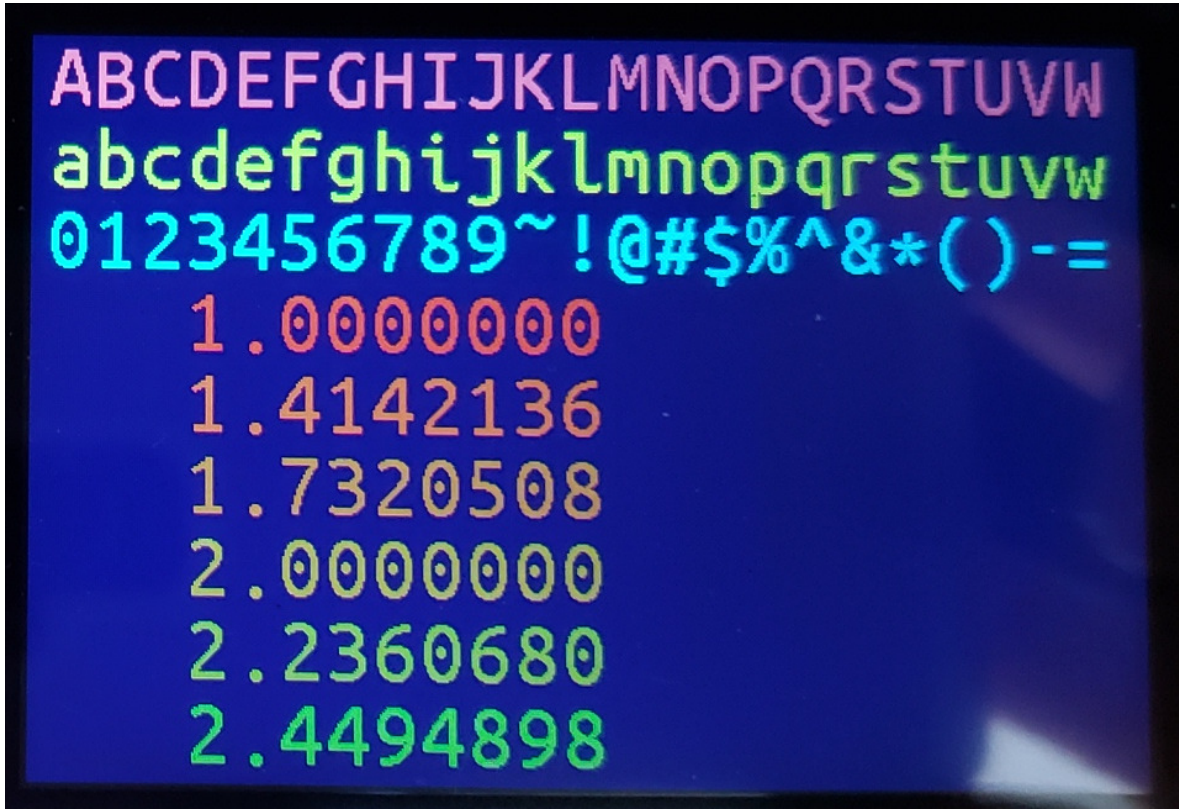
- 16x24 part of LCD_16x24 library
- Also part of LCD_24x32 library
- Uses up 31k or RAM

```
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)

LCD.Number4(number, N, M, x, y, color1, color0)

- Display text / number on the LCD
- Uses 24x32 font



Sample Code

- Only included in LCD_24x32 library
- Uses up 61k or RAM

```
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)
```

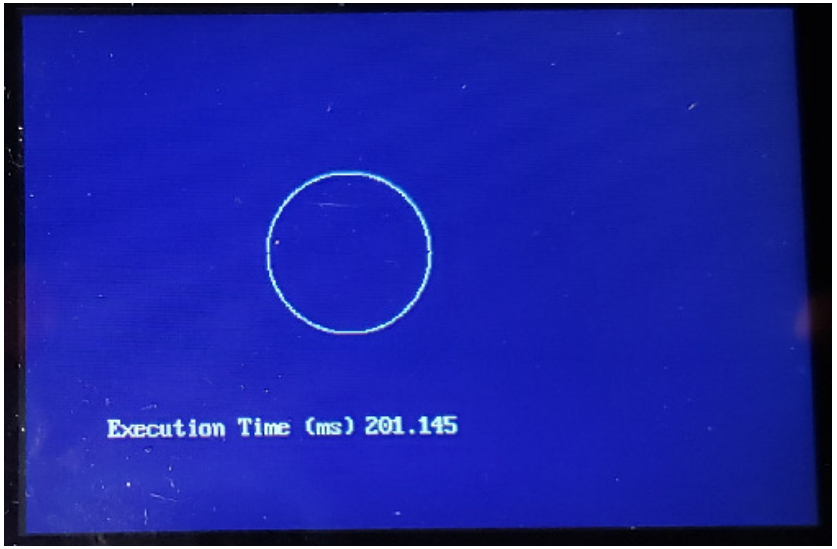
LCD.Pixel(x, y, color)

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).

```
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)
```



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

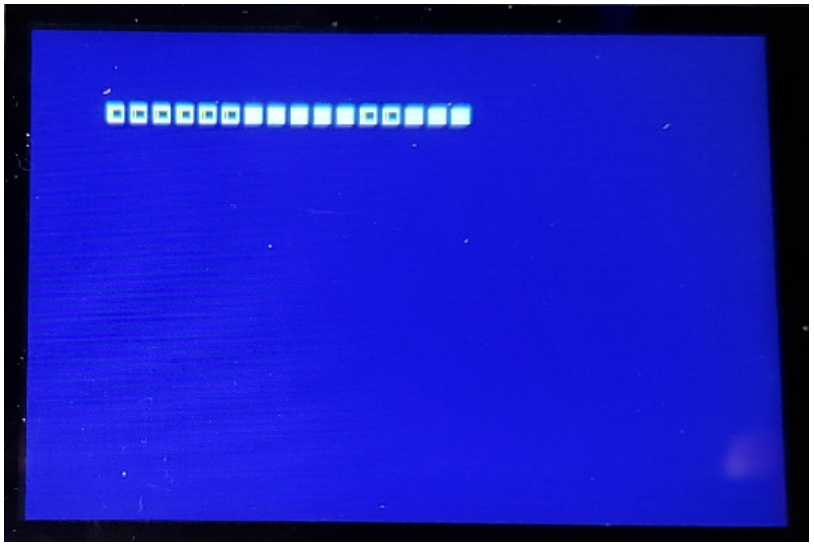
- Display a 10x10 box at (x,y)
  - simulating an LED on the screen
- OnOff = 1 (solid) or 0 (hollow)

LCD.Binary_Out(N, x, y)

Display 16 boxes on the screen

- Display binary value of N

```
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)
```



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)
```

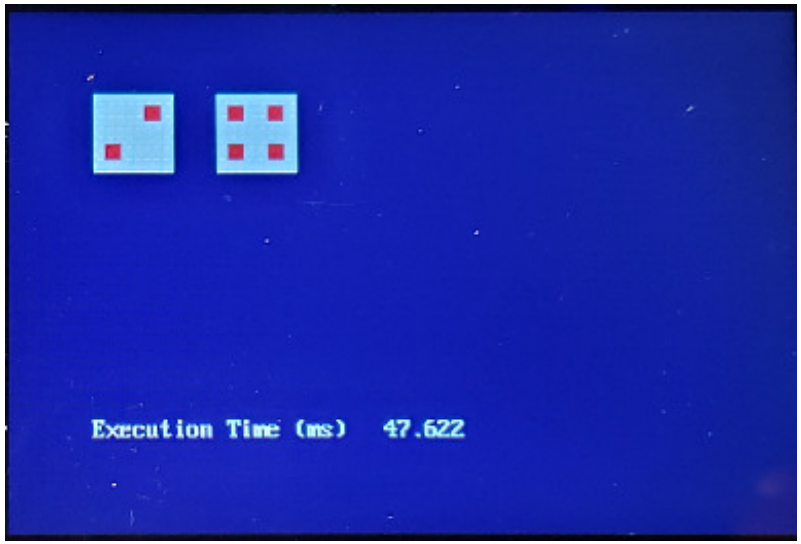
LCD.Dice(N, x, y, color1, color0)

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 = LCD.Sort(X)

- Return an array which shows the sort order of array X
- Sort order is smallest to largest
    - Uses a bubble sort

Deck = LCD.Shuffle()

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

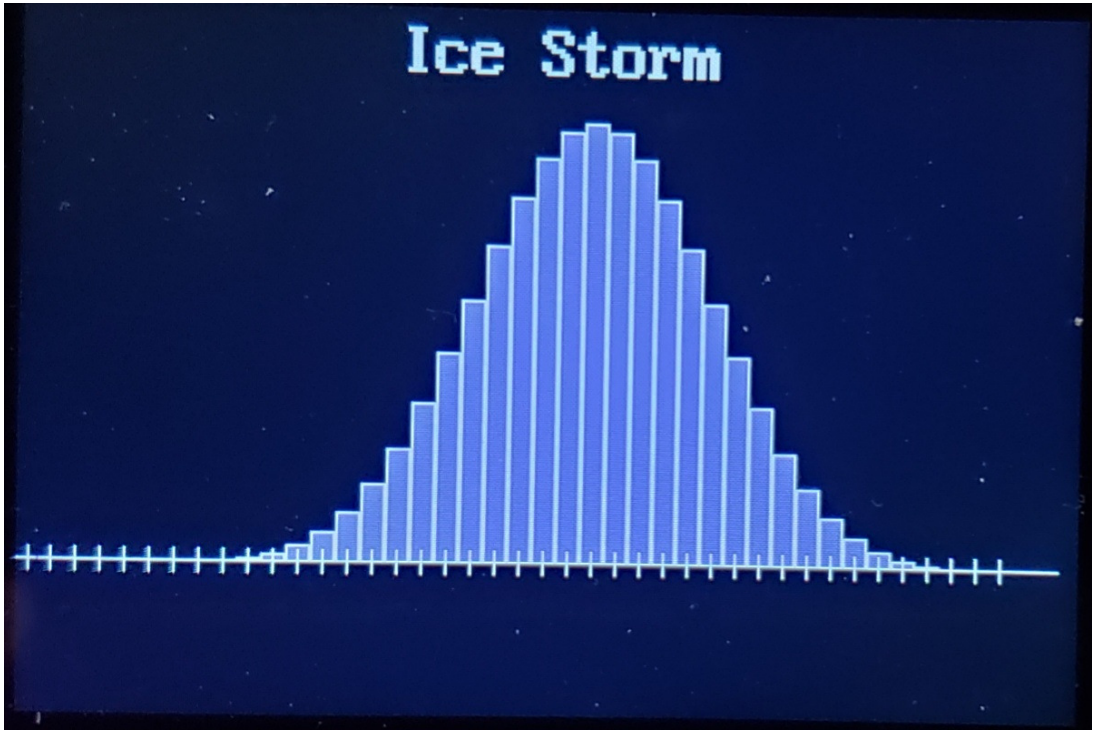
Example: Shuffle, draw five cards, and display them on the LCD

```
import LCD
import random

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

Bar(Data, Edge, Fill)

Display the data on the LCD in a bar graph.

- The peak is scaled to the maximum value of Data
- The x-axis has tic marks for each element in x
- Edge = edge color (white)
- Fill = the fill color (light blue)

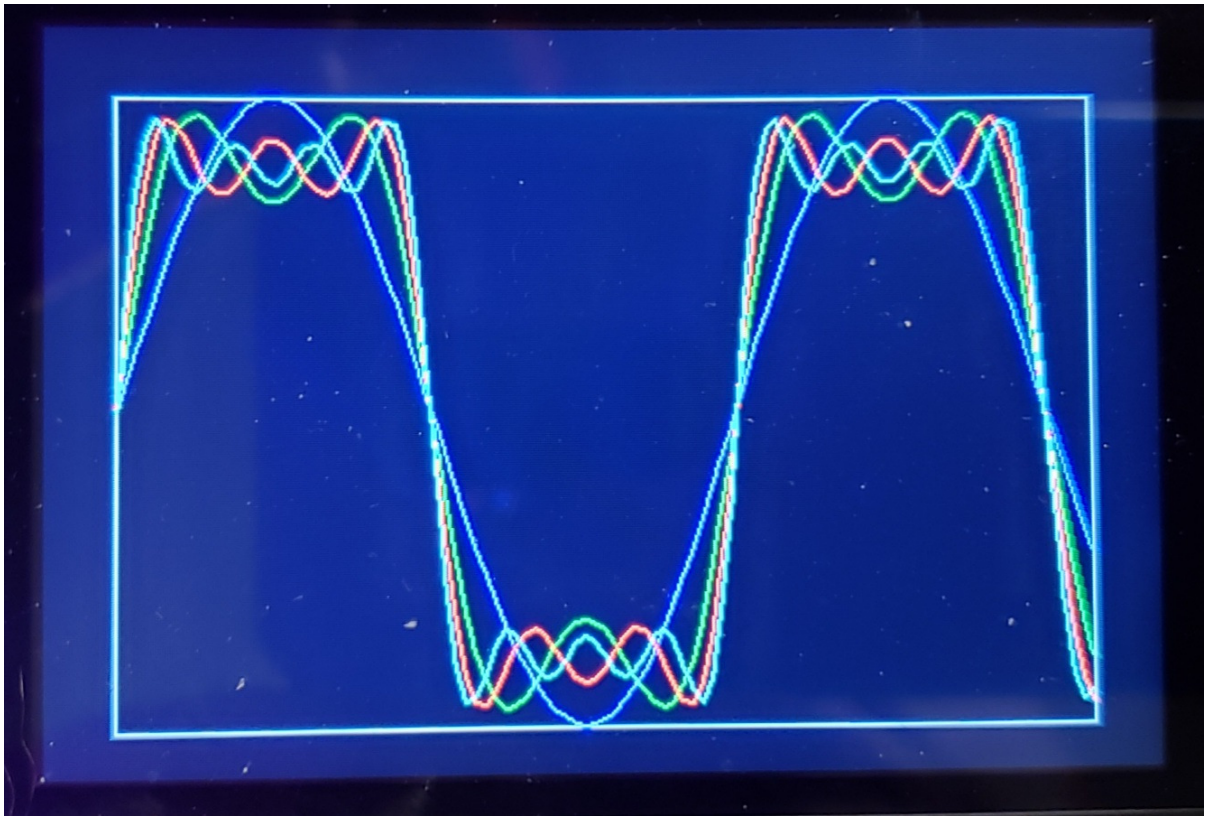


Bar Graph Sample Code

```
import LCD

IceStorm = [0, 0, 0, 0, 1, 4, 10, 20, 35, 56, 82, 112, 143, 172, 196,
212, 218, 212, 196, 172, 143, 112, 82, 56, 35, 20, 10, 4, 1]

Navy = LCD.RGB(0,0,5)
White = LCD.RGB(150,150,150)
Red = LCD.RGB(150,0,0)
LtBlue = LCD.RGB(50,50,150)

LCD.Init()
LCD.Clear(Navy)

LCD.Bar(IceStorm, White, LtBlue)
LCD.Title('IceStorm',White, Navy)
```

Plot(x, y)

- Plot x vs. y on the graphics LCD
- x is a single nx1 vector or matrix
- y is a single nx1 vector or an nxm (or mxn) matrix
    - If y has multiple columns / rows, several plots will be made



Sample Code: plot()

```
import LCD
from math import sin, cos

x = [0]*100
y1 = [0]*100
y2 = [0]*100
y3 = [0]*100
y4 = [0]*100
for i in range(0,100):
    t = i*0.1
    x[i] = t
    y0[i] = sin(t)
    y1[i] = y1[i] + sin(3*t)/3
    y2[i] = y2[i] + sin(5*t)/5
    y3[i] = y3[i] + sin(7*t)/7

LCD.Init()
LCD.Clear(Navy)

LCD.Plot(x1,[y0,y1,y2,y3])
```

Summary

The graphics LCD is a *really* nice way to display data

- You need to copy LCD.py to your Rasbperry Pi Pico to use

With the LCD library, you can display

- Lines,
- Polygons,
- Boxes, and
- Text

65,535 colors are avilable

- Each color is 16 bits in a 5/6/5 format

The standard font size is 8x16

- 16x32 if scaled 2x
- Larger fonts are available
    - LCD_16x24 or LCD_24x32 library
    - These fonts look better, but
    - These take up a lot of memory