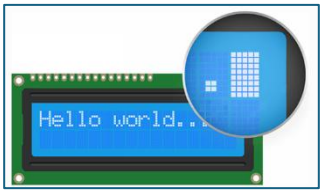
LCD Display 1602 ic2 Module



https://thepihut.com/products/lcd1I2C 16x2 LCD Module | The Pi Hut602-i2c-module

Today we are using the Waveshare i2c 1602 LCD Display v2.0.

This is a simple low cost text display screen, used to output sensor data such as temperature.

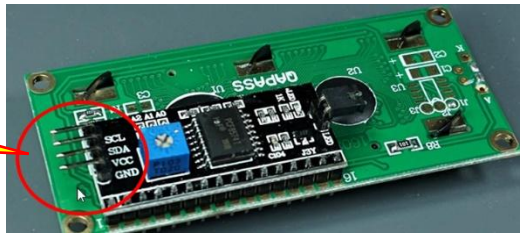
There are other versions, such as the FreeNove i2c 1602 LCD Display v2.0. The pins and labelling of the Freenove version is slightly different. See the **separate guide** to wire up and program the Freenove LCD display.

Contents

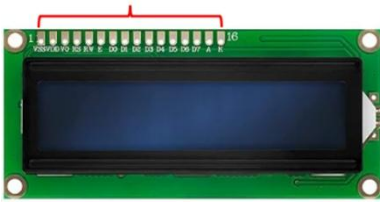
There are two rows of 16 cells. Each cell can display a letter.

Note that some display screens do not have the i2c converter module built in. This refers to the black board with 4 pins.

I2C converter module with 4 pins



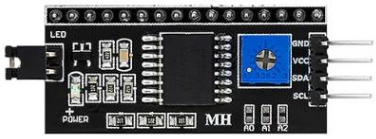
Without the 4-pin converter module, we would have to make use of all these pins along the side! But, we want the 4 pins!



Using the 16 pins is technically possible, but it is more work and can be confusing. If you are interested, here is a video wiring an Arduino device to the 1602 using the 16 pins.

Source: Pi My Life Up

If you do accidentally buy an I2C LCD without the 4 pin converter, you can buy this black board separately and solder it to the back of the I2C LCD screen.



Source: Amazon

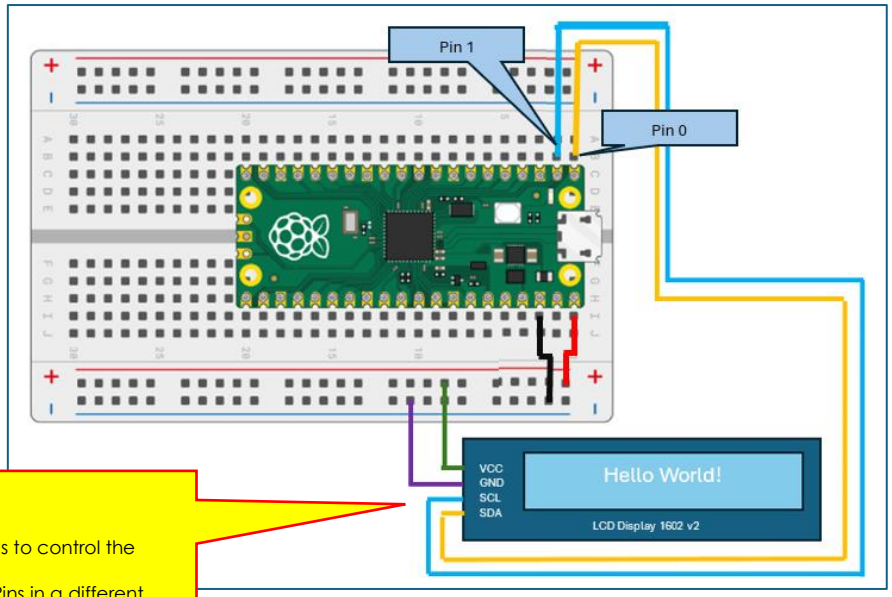
Search for "i2c 1602 display converter module" to order.

The Freenove i2c 1602 LCD display **version 2.0** has the 4-pin converter **module built into** the main board, so you will not see the black board, but it does have the 4 pins.

Specification

- 16x2 LCD panel LCD1602
- **i2C** control interface (only two signal pins are required)
- Compatible with 3.3V/5V operating voltage (this is what the Pico outputs)
- Data sheet here.
- Dimensions and further information here: LCD1602 I2C Module - Waveshare Wiki

Wiring Diagram



VCC = **+** power
GND = **-** power
SCL and **SDA** pins send signals to control the screen.
Some displays are built with Pins in a different order. Check the writing on your hardware.

- Green VCC Pin to + rail on the breadboard
- Purple GND Pin to – rail on the breadboard
- Yellow wire from SDA Pin to GPIO Pin 0 on the breadboard

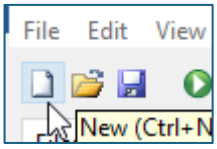
- Blue wire from SCL on the screen to GPIO Pin 1 on the breadboard

The SCL and SDA wires can only connect to certain 'analogue' pins on the Pico. Alternative pins to use are Pin 16 and Pin 17. Click here if you want to read more about this.

Install the LCD1602 Library

This is a library of code that our program will refer to. We do not need to edit this. But it does need to be saved onto the Pico.

Go to the Thonny IDE. Open a new Thonny page.



The code in blue below is small because you do not need to see it.

Copy this library of code:

```
# -*- coding: utf-8 -*-
import time
from machine import Pin,I2C

LCD1602_SDA = Pin(0)
LCD1602_SCL = Pin(1)

LCD1602_I2C = I2C(0,sda = LCD1602_SDA,scl = LCD1602_SCL ,freq = 400000)

#Device I2C Arress
LCD_ADDRESS   = (0x7c>>1)

LCD_CLEARDISPLAY = 0x01
LCD_RETURNHOME = 0x02
LCD_ENTRYMODESET = 0x04
LCD_DISPLAYCONTROL = 0x08
LCD_CURSORSHIFT = 0x10
LCD_FUNCTIONSET = 0x20
LCD_SETCGRAMADDR = 0x40
LCD_SETDDRAMADDR = 0x80

#flags for display entry mode
LCD_ENTRYRIGHT = 0x00
LCD_ENTRYLEFT = 0x02
LCD_ENTRYSHIFTINCREMENT = 0x01
LCD_ENTRYSHIFTDECREMENT = 0x00

#flags for display on/off control
LCD_DISPLAYON = 0x04
LCD_DISPLAYOFF = 0x00
LCD_CURSORON = 0x02
LCD_CURSOROFF = 0x00
LCD_BLINKON = 0x01
LCD_BLINKOFF = 0x00

#flags for display/cursor shift
LCD_DISPLAYMOVE = 0x08
LCD_CURSORMOVE = 0x00
LCD_MOVERIGHT = 0x04
LCD_MOVELEFT = 0x00

#flags for function set
LCD_8BITMODE = 0x10
LCD_4BITMODE = 0x00
LCD_2LINE = 0x08
LCD_1LINE = 0x00
LCD_5x8DOTS = 0x00

class LCD1602:
  def __init__(self, col, row):
    self._row = row
    self._col = col

    self._showfunction = LCD_4BITMODE | LCD_1LINE | LCD_5x8DOTS;
    self.begin(self._row,self._col)

  def command(self,cmd):
    LCD1602_I2C.writeto_mem(LCD_ADDRESS, 0x80, chr(cmd))

  def write(self,data):
    LCD1602_I2C.writeto_mem(LCD_ADDRESS, 0x40, chr(data))

  def setCursor(self,col,row):
    if(row == 0):
      col |=0x80
    else:
      col |=0xc0;
    LCD1602_I2C.writeto(LCD_ADDRESS, bytearray([0x80,col]))

  def clear(self):
    self.command(LCD_CLEARDISPLAY)
    time.sleep(0.002)
  def printout(self,arg):
    if(isinstance(arg,int)):
      arg=str(arg)

    for x in bytearray(arg,'utf-8'):
      self.write(x)


  def display(self):
    self._showcontrol |= LCD_DISPLAYON
    self.command(LCD_DISPLAYCONTROL | self._showcontrol)

  def begin(self,col,lines):
    if (lines > 1):
      self._showfunction |= LCD_2LINE

    self._numlines = lines
    self._currline = 0

    time.sleep(0.05)

    # Send function set command sequence
    self.command(LCD_FUNCTIONSET | self._showfunction)
    #delayMicroseconds(4500);  # wait more than 4.1ms
    time.sleep(0.005)
    # second try
    self.command(LCD_FUNCTIONSET | self._showfunction);
    time.sleep_us(150)
    # third go
    self.command(LCD_FUNCTIONSET | self._showfunction)
    time.sleep(0.001)
    # finally, set # lines, font size, etc.
    self.command(LCD_FUNCTIONSET | self._showfunction)
    # turn the display on with no cursor or blinking default
    self._showcontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF
    self.display()
    # clear it off
    self.clear()
    # Initialize to default text direction (for romance languages)
    self._showmode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT
    # set the entry mode
    self.command(LCD_ENTRYMODESET | self._showmode);
    # backlight init
```

If you are using a Pico W (the wireless Pico with the silver chip), you need an alternative library. This can be found here (yet to be tested).

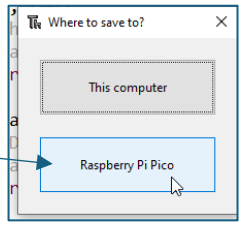
Paste the code into the Thonny page.

You should see a page like this:

Note that this is where the Pico pins are set. If you are not using Pin 0 and Pin 1 you need to change these lines of code.

Alternative Pins are Pin 16 and 17. See the PinOut Diagram.

```
[ LCD1602.py ]
1  # -*- coding: utf-8 -*-
2  import time
3  from machine import Pin,I2C
4
5  LCD1602_SDA = Pin(0)
6  LCD1602_SCL = Pin(1)
7
```



In Thonny, click File / Save as…

Select the Pico option:

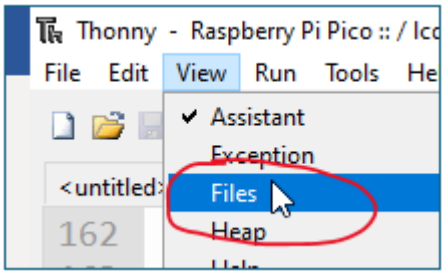
Save it as **LCD1602.py** (onto the Pico).

Saving with the correct file name is important.

You must save using capital letters for **LCD**1602.py

Let's check if the file has been correctly saved onto the Pico.

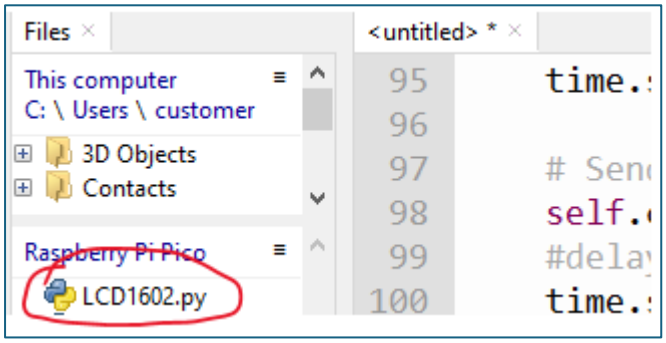
In Thonny, click **View** / **Files:**



You should now see a window appear on the left side of your screen.

This shows the files stored on the Pico device.

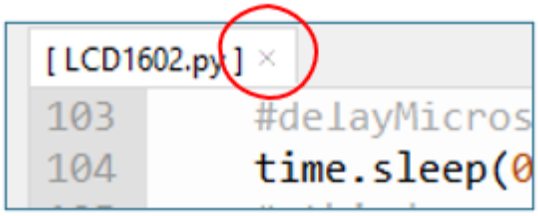
Can you see the LCD1602.py file?



We can now close this Thonny page.

Remember, we do not need to edit this library file.

We just need it to be saved onto the Pico.

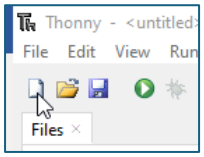


Ok, now we have installed the library file, we can start to make text appear on the LCD screen. Let's see if we can output text like this:

Display Text on the LCD Screen

Let's see if we can output some text to the display.

In Thonny, create a new blank page:



Copy this test code and paste it into the new Thonny page:

Test code version 1 – show and hide lines of text

```
import LCD1602
import time

lcd=LCD1602.LCD1602(16,2)

lcd.setCursor(0, 0)
lcd.printout("Hello")
lcd.setCursor(0, 1)
lcd.printout("World!")
time.sleep(2)
lcd.clear()
lcd.setCursor(0, 0)
lcd.printout("i2c1602 display")
lcd.setCursor(0, 1)
lcd.printout("screen")
time.sleep(2)
lcd.clear()
del lcd
```

Test code version 2 – use a while loop

```
import LCD1602
import time
lcd=LCD1602.LCD1602(16,2)
try:
    while True:
        # set the cursor to column 0, line 1
        lcd.setCursor(0, 0)
        lcd.printout("Waveshare")
        lcd.setCursor(0, 1)
        lcd.printout("Hello World!")
        time.sleep(0.1)
except(KeyboardInterrupt):
    lcd.clear()
    del lcd
```

You should see this:

```
<untitled> *
1  import LCD1602
2  import time
3
4  lcd=LCD1602.LCD1602(16,2)
5
6  try:
7      while True:
8          # set the cursor to column 0, line 1
9          lcd.setCursor(0, 0)
10
11         lcd.printout("Waveshare")
```

Run the program.

Hopefully you are getting this output on your display screen.

Trouble shooting.

If you are not getting text appearing on the LCD screen check the following:

- Are you wiring SDA to Pin 0
- Are you wiring SCL to Pin 1
- Have you saved LCD1602.py onto the Pico with the correct file name
- Is the VCC wire going to the positive rail?
- Is the GND wire going to the negative rain?
- Are the rails connected to GND and power pins?

Challenge 1 – Change the text to be output.

The test code text produced this output.

Can you edit the test code to output different text?

Make it output “Hello” on the first row, and “Python Ninja” on the second row.

Challenge 2 – Make text output, clear the screen and make another text output.

You could make it tell a joke!

Challenge 3 – Display the temperature

The Pico has an on-board temperature sensor.

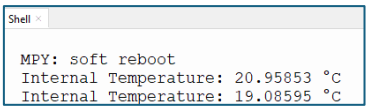
We can output a reading from this built in sensor, to the i2C1602 display.

This code will output a temperature reading to the Thonny shell. You will need to develop it to make the output display on the i2c1602 screen.

Copy and paste this into Thonny and run it.

```
from machine import ADC
import time
# Internal temperature sensor is connected to ADC channel 4
temp_sensor = ADC(4)

while True:
    adc_value = temp_sensor.read_u16() # Read the raw ADC value
    voltage = adc_value * (3.3 / 65535.0)# Convert ADC value to voltage
    temp = 27 - (voltage - 0.706) / 0.001721 # Temperature calculation based on sensor characteristics
    print("Internal Temperature:", temp, "°C")
    time.sleep(1)
```

You should see the output in the Thonny shell, like this. Put your finger on the Pico to see the temperature change.



How can you combine this code with the i2C1602 display screen code to output the temperature?

Challenge 4 - Display the current time.

Try this test code to see if you can display the time:

```
import LCD1602
import time

lcd=LCD1602.LCD1602(16,2)

try:
    while True:
        # set the cursor to column 0, line 1
        lcd.setCursor(0, 0)
        # print the number of seconds since reset:

        # print the number of seconds since reset:
        T=list(time.localtime())
        T[6]+=1
        T=["{:0>2}".format(str(i)) for i in T]
        lcd.printout(T[0]+' '+T[1]+' '+T[2]+' '+T[6])

        lcd.setCursor(0, 1)

        lcd.printout(T[3]+":"+T[4]+":"+T[5])
        time.sleep(0.1)
except(KeyboardInterrupt):
    lcd.clear()
    del lcd
```

Can you edit this so the year appears after the date?

I think one of the numbers is the day of the week e.g. Tuesday = 2. How could you develop this so it output “Tue” instead of “2”? I have not tried this myself yet 😊

Challenge 5 – Display output from ultrasonic distance sensor

This challenge is all about merging two projects. Have a look a the guide on the ultrasonic distance sensor. Get it working so the output appears in the thonny shell.

Then try to output the variable to the display screen.

Remember, the display screen cannot handle float. I am not sure whether it can handle integers. It could be everything has to be cast to string before output.

Sources and Further Reading

https://docs.sunfounder.com/projects/thales-kit/en/latest/micropython/liquid_crystal_display.html

https://www.waveshare.com/wiki/LCD1602_I2C_Module#Working_With_Pico

Solution 1 to temperature challenge

```
#i2c1602 display libaries
import LCD1602
import time
lcd=LCD1602.LCD1602(16,2)
#temp sensor library and variable
from machine import ADC
temp_sensor = ADC(4)

def check_temp():
    adc_value = temp_sensor.read_u16() # Read the raw ADC value
    voltage = adc_value * (3.3 / 65535.0)# Convert ADC value to voltage
    temp = 27 - (voltage - 0.706) / 0.001721 # Temperature calculation based on sensor characteristics
    temp_int = int(temp)
    print("Internal Temperature:", temp, "°C")#outputs to shell
    return temp_int

while True:
    reading = check_temp()
    lcd.setCursor(0, 0)
    lcd.printout("Temperature =")
    lcd.setCursor(0, 1)
    lcd.printout(reading)
    time.sleep(0.5)
```

Solution 2 to the Temperature challenge

The first solution only output integer (whole numbers).

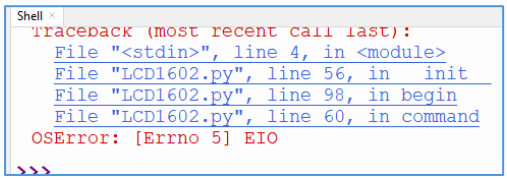
This solution converts (casts) to string before outputting to the display.

```
#i2c1602 display libraries
import LCD1602
import time
lcd=LCD1602.LCD1602(16,2)
#temp sensor library and variable
from machine import ADC
temp_sensor = ADC(4)

while True:
    adc_value = temp_sensor.read_u16() # Read the raw ADC value
    voltage = adc_value * (3.3 / 65535.0)# Convert ADC value to voltage
    temp = 27 - (voltage - 0.706) / 0.001721 # Temperature calculation based on sensor characteristics
    print("Internal Temperature:", temp, "°C")#outputs to shell
    x = round(temp,2)#rounds the output to two decimal places
    y = str(x)#converts the output to string(the display cannot handle float)
    lcd.setCursor(0, 0)
    lcd.printout(y)
    time.sleep(1)
    lcd.clear()
```

More trouble shooting

An error like this suggests a connection issue.

I got the Waveshare 1602 display to work but could not get the Ali Express version to connect. Possible solder bridging issue?



You can scan to see if you Pico can see the connection to the display

```
from machine import I2C, Pin
i2c = I2C(0, sda=Pin(0), scl=Pin(1))
print(i2c.scan())
```

i2c.scan() should return a short list with the device addresses on the bus. These are decimal numbers. If the list is either empty or very long, then there is still a wiring problem.

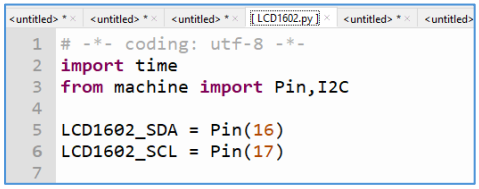
https://forum.micropython.org/viewtopic.php?t=12941

This is alternative code for scanning for a connection:

```
import machine

sdaPin = machine.Pin(0)
sclPin = machine.Pin(1)
i2c = machine.I2C(0, sda=sdaPin, scl=sclPin, freq=400000)

devices = i2c.scan()

if len(devices) == 0:
  print('No i2c device found!')
else:
  print('i2c device found', len(devices))

for device in devices:
  print("At address: ", hex(device))
```

Try Pin 16 and Pin 17

Change the lines of code in LCD1602.py



When I run this code (edited to use pin 16 and 17) I now get the address

```
import machine

sdaPin = machine.Pin(16)
sclPin = machine.Pin(17)
i2c = machine.I2C(0, sda=sdaPin, scl=sclPin, freq=400000)

devices = i2c.scan()

if len(devices) == 0:
  print('No i2c device found!')
else:
  print('i2c device found', len(devices))

for device in devices:
  print("At address: ", hex(device))
```

TRY THIS

https://docs.sunfounder.com/projects/umsk/en/latest/04_pi_pico/pico_lesson26_lcd.html

Sources and Further Reading

How to Use I2C LCD with Arduino | Very Easy Arduino LCD I2C Tutorial | Arduino 16x2 LCD I2C Tutorial

Note: MicroPython and CircuitPython are not the same, although they look similar. The Pico will run either, but be aware micropython has been installed on the Pico's at school.

I2C Liquid Crystal Displays | Arduino Project Hub