Raspberry Pi Pico and I2C using MicroPython

Introduction

Today, we will be discussing how to use Pico or Pico W to manage I2C devices using MicroPython.

Raspberry Pi Pico and Pico W

The Pico and the Pico W are basically the same microcontroller when it comes to I2C two-wire serial communications. The Pico also supports SPI which is a three wire serial communications and UART which is universal asynchronous receiver/transmitter. We will only be covering the I2C protocol.

I2C Protocol

I2C stands for Inter-Integrated Circuit and is also called IIC and I^2C. This protocol is designed for low speed communications, usually in the 100kHz to 400kHz range. Communications takes place on two wires: the SDA for data and the SCL for the clock.

There are two types of devices in I2C. The master device is the one that initiates communication and the slave device(s) respond. Multiple slave devices can be connected to the same I2C bus. Each slave device is identified by a unique 7 bit address. For details on how I2C actually works and transmits bits, see the I2C Specification, reference listed below.

Communications always starts with the master device writing a byte to the I2C bus that contains the 7 bit slave address and an I/O bit. If the I/O bit is low, the master device continues to write bytes to the I2C bus. If the I/O bit is high, the addressed slave device writes bytes to the I2C bus. If the address does not match the address of the slave device, that device ignores the communication.

The I2C specifies how bytes are sent over the I2C bus and the format of the first byte. How to interrupt the rest of the bytes send is device specific and will be described in their datasheets.

I2C on the Pico and Pico W

The Poco boards contain two I2C buses, just like the Raspberry Pi. The buses are labeled as I2C0 and I2C1 and can be referred to as I2C Bus 0 and I2C Bus 1. Each I2C bus uses a different set of GPIO Pins and the pins used determine which of the two I2C buses is used.

On the Raspberry Pi, the command “l2cdetect busNo”[1] will scan the selected bus, 0 or 1, and return print a matrix of I2C Addresses in the command window. There is no equivalent program for the Pico. The MicroPython I2C class does provide API to perform the equivalent function. See the program on the next page for an example:

1 Run the following command, “sudo apt-get install i2c-tools”, to install i2cdetect on the Raspberry Pi.

I2C-scan.py by Steven F. LeBrun

```
## I2C-scan.py
import utime
from machine import I2C, Pin

def print_devices( i2c, devices ):
    print("In Print I2C Device Addresses")
    print("I2C ID = ", i2c)
    if ( devices ):
        for dev in devices:
            print(hex(dev))
            utime.sleep(1)

i2c0 = I2C(0, sda=Pin(8), scl=Pin(9), freq=400000)
i2c1 = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)

devices0 = i2c0.scan()
print_devices( i2c0, devices0 )

devices1 = i2c1.scan()
print_devices( i2c1, devices1 )
```

Pico I2C Pins

The Pico provides 6 sets of SDA/SCL pins for each of the I2C Buses. Only one set should be used for each bus.

Table 1: Pico I2C Pins

I2C Bus 0			I2C Bus 1	
SDA	**SCL**	**Set**	**SDA**	**SCL**
0	1	**1**	2	3
4	5	**2**	6	7
8	9	**3**	10	11
12	13	**4**	14	15
16	17	**5**	18	19
20	**21**	**6**	**26**	**27**

SSD1306 oLED Display and other I2C Devices

The oLED is a bitmap display using organic LED technology. We will be using the 0.96”, 128x64 pixel version of this display. The chip that drives this I2C Device is the SSD1306 chip.

The first place to start with a new I2C Device is its datasheet. This will contain information about how I2C is being used and how to connect the devices pins to the Pico. In this case, there are only 4 I2C pins so they are connected to one set of SDA and SCL pins, to either a 5V or 3.3V power source, and a ground. This display runs on both 5V and 3.3V.

Before getting too deep into the datasheets, look to see if a MicroPython library already exists. If it does, the datasheet will be useful for hooking up the device and the library will handle the protocol that sits onto of the I2C protocol. The two places to look is PyPi, which Thonny[2] will search for you, and GitHub. Use Google or another search engine to search for possible libraries on GitHub or other repositories. Libraries must include the I2C interface. They usually are just passed an I2C object and handle all the I2C communications under the hood.

Another source of information are example programs. There are example programs for most commonly used devices somewhere on the Internet. This is where Google or other search engines help. Some example programs are part of a tutorial which will provide more details about working with the device than just the source code.

Use the example programs, either by running them to see that the device and library works or by using the information in the example program to write your own.

If no example programs exist, then you need to read the library source code file to see what the APIs are. Hopefully, the library will be well documented and/or uses method names that are easy to understand.

If no libraries exist, then you will need to use the datasheet in order to write your own library. This is a last resort since it is the most complicated approach. You will need to determine both what the protocol is that is on top of I2C and how to operate the device. How to initialize the device. How to read and write to the device? Are there registers that need to be addressed before data can be read or written?

2 In Thonny with a Pico connected, go to **Tools → Manage packages…** , type in the chip name in the text box on top and click on the **Search on PyPi** button. A list of potential libraries will be displayed. Pick the one you want and click on **Install** button. The library package will be stored on the Pico in the /lib directory.

Raspberry Pi Pico and I2C using MicroPython

The following program is an example of how to display text on an SSD1306 display using the library micropython-ssd1306 library and its class SSD1306_I2C.

oLED-Demo.py by Steven F. LeBrun

```
## oled-demo.py
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

#i2c = I2C(0, sda=Pin(8), scl=Pin(9), freq=400000 )
i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000 )

display = SSD1306_I2C( 128, 64, i2c, addr=0x3C )

display.text("Hello World!", 0,0,1)
display.hline(0, 10, 128, 1)
display.vline(100, 0, 64, 1)
display.show()
```

References

Example Programs and Presentation Documents

- GitHub Repository
  - https://github.com/sflebrun/Robot-Maker
  - https://github.com/sflebrun/Robot-Maker/tree/main/Pico%20and%20I2C%20Talk

ADXL345 Accelerometer

- Datasheet
  - https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf
- ADXL345 and Pico using MicroPython
  - https://www.digikey.com/en/maker/projects/raspberry-pi-pico-rp2040-i2c-example-with-micropython-and-cc/47d0c922b79342779cdbd4b37b7eb7e2

BME280

- Datasheet
  - https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf

I2C Protocol

- I2C Bus, Interface, and Protocol
  - https://i2c.info/
- I2C Bus Specification
  - https://i2c.info/i2c-bus-specification
- UM10204 – I2C Bus Specification and User Manual
  - https://www.nxp.com/docs/en/user-guide/UM10204.pdf

LCD Display

- Library used in GitHub
  - https://github.com/T-622/RPI-PICO-I2C-LCD.git
- Library contains two files that need to be copied to Pico
  - lcd_api.py
    - Abstract base class LcdApi for operating LCD Displays
    - Imported indirectly pico_i2c_lcd
  - pico_i2c_lcd.py
    - Class I2cLcd, derived from LcdApi
    - Adds I2C functionality to LCD communication

MicroPython

- Overview – MicroPython 1.19.1 Documentation

    - https://docs.micropython.org/en/latest/index.html
- Class I2C – a two-wire serial protocal
    - https://docs.micropython.org/en/latest/library/machine.I2C.html?highlight=i2c#machine.I2C
- Class FrameBuffer – for creating and manipulating bitmaps
    - https://docs.micropython.org/en/latest/library/framebuf.html
- Class Pin – Control GPIO Pins
    - https://docs.micropython.org/en/latest/library/machine.Pin.html

Raspberry Pi Pico and Pico W

- **Raspberry Pi Pico [W]** Pinout
    - https://datasheets.raspberrypi.com/picow/PicoW-A4-Pinout.pdf
- Raspberry Pi Pico SDK Documentation V1.4.0
    - https://raspberrypi.github.io/pico-sdk-doxygen/index.html
- Class machine.I2C for Pico
    - https://docs.micropython.org/en/latest/library/machine.I2C.html
    - https://wiki.sipeed.com/soft/maixpy/en/api_reference/machine/i2c.html

SSD1306 oLED Display

- Datasheet
    - https://www.elecrow.com/download/SSD1306%20Datasheet.pdf
- OLED 4 Pin 128 x 64 Display Module 0.96”
    - https://www.rajguruelectronics.com/Product/1145/OLED%204%20Pin%20128x64%20Display%20module%200.96%20inch%20blue%20color.pdf
- SSD1306 Demo V3
    - https://www.instructables.com/SSD1306-With-Raspberry-Pi-Pico/?fbclid=IwAR0Eg8xfaiJfQH-qZjBUu31SCgZRet8Qgi7HSJ0NvJ9zgLF8q60sfZ62Hp0