Pico and I2C
Using MicroPython

Steven F. LeBrun

October 8, 2022

Roanoke Robotics & Makers Club
of Southwest Virginia

Overview

- Raspberry Pi Pico
- I2C Protocol
- Raspberry Pi Pico and I2C
- Programming I2C Devices for Pico
- SSD1306 oLED Display
- LCD Display
- ADXL345 Accelerometer
- References

Raspberry Pi Pico

- The Pico is a microcontroller board – not a computer
- Based on the RP2040 chip – Dual core ARM processor
- Powered by 5V USB cable
- GPIO Pins
  - High -- 3.3V
  - Low -- 0V or Ground
- Supports I2C, SPI, UART communication protocols
- 3 available ADC pins

I2C Protocol

- I2C, a.k.a. I^2C, IIC, is formally called **I**nter-**I**ntegrated **C**ircuit
- Two-Wire Serial communications between low speed devices
  - SDA – Serial Data
  - SCL – Serial Clock
- Multiple devices can be connected to the same SDA & SCL lines
- One device is a master and the other devices are slaves
- Each slave device identified with a unique 7 bit address

I2C Protocol

- Transmission starts with the Master Device generating a Start Condition of SDA low and SCL high
- First byte written by Master
  - Contains Slave Address and the I/O bit
- The I/O bit is bit 0 determines I/O direction
  - High – Slave writes next bytes and Master reads them
  - Low -- Master writes next bytes and Slave reads them
- Transmission ends with Master Device generating a Stop Condition of SDA high and SCL high

I2C Protocol

- Each [slave] I2C Device has its own protocol that is on top of I2C.
- Device datasheets explain what those protocols are
- Commonly, the first byte is a register or memory location on the slave device.
- The next n bytes are the data read or written to and from the slave device.
- The format of those bytes are defined in the device datasheet

Pico – I2C GPIO Pins

- The Pico, like the Raspberry Pi, has two I2C buses.
  - Labeled I2C0 for I2C Bus 0, and I2C1 for I2C Bus 1
- On the Raspberry Pi, the command “i2cdetect 0” and “i2cdetect 1” are used to scan the I2C Bus 0 and Bus 1, respectively.
  - Use the command “sudo apt-get install i2c-tools” to install i2cdetect
- There is no such program is available for the Pico
  - Can use method I2C.scan() to obtain the same information on the Pico
  - My program, I2C-scan.py, will perform the scan operation and list the I2C addresses that are active on the two buses.

I2C Scan Program

```
## I2C-scan.py

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))

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 )
```

Output from I2C-scan.py

In Print I2C Device Addresses

I2C ID = I2C(0, freq=399361, scl=9, sda=8)

0x1d

0x27

0x3c

In Print I2C Device Addresses

I2C ID = I2C(1, freq=399361, scl=7, sda=6)

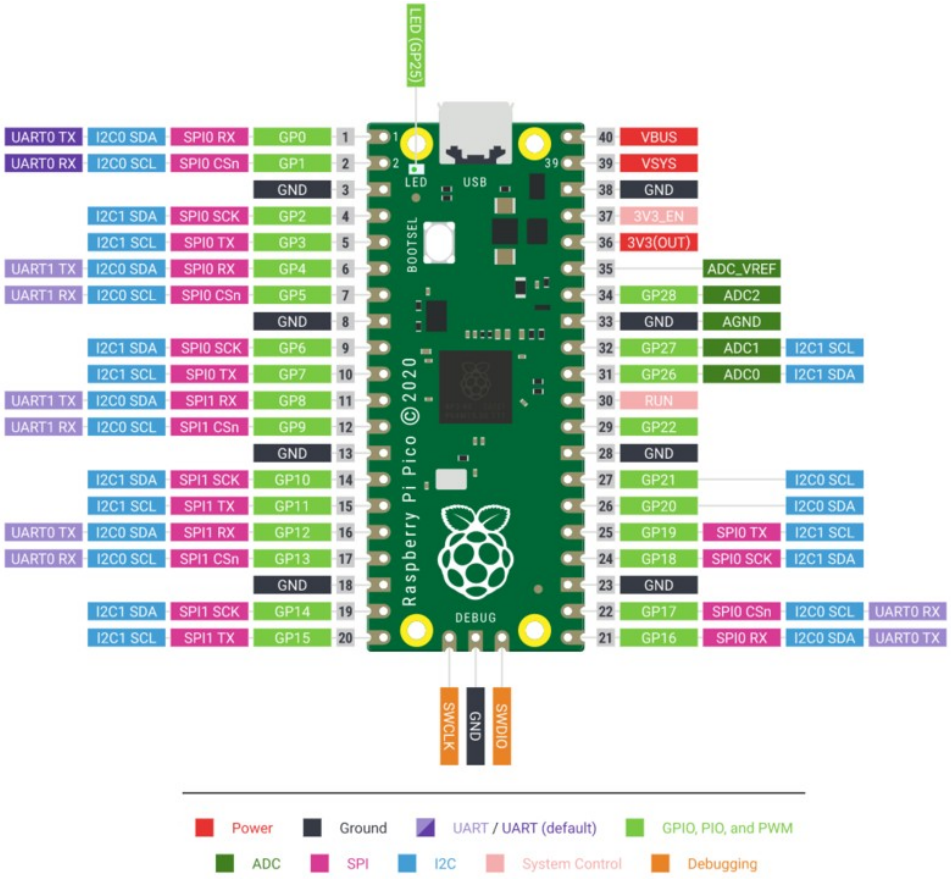
Pico – I2C GPIO Pins

- Different sets of GPIO Pins on the Pico can be used for I2C communications
- See a Raspberry Pi Pico Pinout to determine which GPIO Pins are configured for I2C I/O
- I2C GPIO Pins can be used for other purposes.
- The following table lists the pairs of SDA and SCL GPIO Pins.

Pico – I2C GPIO Pins

I2C Bus 0			I2C Bus 1	
SDA	**SCL**		**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

Pico Pinout


LED (GP25)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging

Programming I2C Device for Pico

- Select a module that supports I2C
  - Examples SSD1306 oLED display, ADXL345 Accelerometer, BME280 Combined Temperature/Humidity/Pressure sensor
- Information needed before writing code:
  - Datasheet
  - MicroPython Library/Package – If any exists
  - Example Programs – If any exist

Programming I2C Device for Pico – Datasheets

- The datasheet provides the following information
  - Pins and how to connect them to Pico
  - Protocol that rides on top of I2C
  - Commands and Data for controlling device
- Control Flow for the device
  - How to initialize
  - How to change modes or behavior
  - How to read and write data

Programming I2C Device for Pico – Libraries

- Performs the low level work needed to control the device
  - Performs the I2C byte-level communication
- Provides API for basic commands
  - Such as display text, read data, initialize device
- Make sure Library is for the programming language you are using
  - MicroPython vs CircuitPython
- Use Thonny or Google to find libraries

Programming I2C Device for Pico – Libraries

- ADXL345 Accelerometer
  - Hass-python-adxl34x library
- BME280 Combined Temperature/Humidity/Pressure Sensor
  - micropython-bme280
- LCD 20x4 Display
  - RPI-PICO-I2C-LCD by T-622 on GitHub
- SSD1306 oLED Display
  - micropython-ssd1306

Programming I2C Device for Pico – Examples

- Example Programs
  - Provide source code for controlling the device.
    - Make sure example uses the same or similar library
  - Provide insight on how to use the device
    - What steps are needed to prepare the device for use
    - What steps are needed to obtain data from device
    - For display devices, how to get text, images, pixels to actually be displayed

Programming I2C Device for Pico

- If no library can be found
  - You will need to write your own low-level library
  - Use the datasheet to:
    - Determine how to write to and read from the device using I2C
    - What methods are needed based on the command set for the device
- Command sets are usually mapped to registers or memory locations
  - Registers are normally represented by a one byte hexadecimal value
  - Registers usually have one byte data values that can be RO, WO, RW

SSD1306 OLED Display

- OLED or oLED == Organic Light Emitting Diodes
- OLED are bitmap displays
- Common SSD1306 module is a 0.96” monochrome display
  - With 128 x 64 pixels
  - The foreground color is blue. Sometimes the first 15-20 rows of pixels may be a different color such as red or yellow
- Multiple libraries are available to handle the low level work of running the display.
  - My choice was micropython-ssd1306

SSD1306 OLED Display

- The ssd1306 defines three classes
  - SSD1306 – Base class derived from frame.FrameBuffer class
    - This is an abstract class used by the following two classes
  - SSD1306_I2C – Class that uses I2C to communicate with display
    - This is the class we will use.
  - SSD1306_SPI – Class that uses SPI to communicate with display
    - SPI is another serial communication protocol that uses two shared wires for Data and Clock, plus a third dedicated wire instead of addresses.

SSD1306 OLED Display

- The underlying **FrameBuffer** class contains methods for managing a bitmap display, such as the SSD1306
- A set of primitive methods control pixels:
  - The are ***fill()***, ***pixel()***, ***hline()***, ***vline()***, ***line()***, ***rect()***, ***elipse()***, ***poly()***
- They are ***text()*** places text strings into the bit map
- Extra methods for more complex operations:
  - They are ***scroll()***, ***blit()***
- Most important, ***SSD1306.show()***, which causes changes to be sent to the display.

Simple OLED Program

```
## 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()
```

LCD Displays

- LCD Displays are character based displays
- Two common sizes are 16x2 characters and 20x4 characters
  - 16 and 20 are the number characters in a row
  - 2 and 4 are the number of rows
- LCD Displays normally use a lot of GPIO pins
- LCD Displays with I2C interfaces uses 2 GPIO pins plus a Vcc and Ground connection

LCD Displays

- MicroPython libraries for LCD Displays are not as common as the libraries for the SSD1306
- The micropython-lcd library on pypi.org is a dummy library.
  - No content or code
  - PyPi is where Thonny searches for packages
- Libraries are available on GitHub
  - I use https://github.com/T-622/RPI-PICO-I2C-LCD.git
  - By Tyler Peppy -- T-622

ADXL345 Accelerometer

- The ADXL345 measures acceleration
  - In X, Y, and Z axes
  - Measures dynamic and static acceleration
    - Dynamic acceleration of movement
    - Static acceleration of gravity
- Uses either SPI or I2C communication protocols
  - Communication protocol selected by CS Pin
  - For this presentation, we will only be using I2C

ADXL345 Accelerometer

- Has an 8 pin interface
  - 4 pins for I2C and SPI communications
  - CS pin to select protocol
    - High for I2C and Low for SPI
  - SDO – Serial Data Output
    - If CS is Low, SDO is for SPI as Serial Data Output
    - If CS is High, I2C Address – High for 0x1D and Low for 0x53
  - 2 pins to generate interrupts

References – This Presentation

- GitHub – Repository Robot-Maker for “Pico and I2C Talk”
  - https://github.com/sflebrun/Robot-Maker
- Directory containing Source Code and Presentation Documents
  - https://github.com/sflebrun/Robot-Maker/tree/main/Pico%20and%20I2C%20Talk

References – ADXL345

- 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

References – I2C

- 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

References – 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

References – 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

References – Pico

- **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

References – SSD1306

- 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