4. Using I2C

1. Code path

RDK X3 enables I2C0 by default on 40Pin, physical pins 3 and 5, IO voltage 3.3V.

Please refer to `/app/40pin_samples/test_i2c.py` for details on how to use I2C.

2. Test method

- Run the test program `python3 /app/40pin_samples/test_i2c.py`
- First list the i2c buses enabled by the current system
- Scan to get which peripherals are connected to the current bus by entering the bus number
- Enter the peripheral address (hexadecimal number), and the test program will read a byte of data from the peripheral

3. Running result

```
Starting demo now! Press CTRL+C to exit
List of enabled I2C controllers:
/dev/i2c-0  /dev/i2c-1
Please input I2C BUS num:1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- UU -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
Please input I2C device num(Hex):40
Read data from device 40 on I2C bus 1
read value= b'``'
```

4. Test code

```
#!/usr/bin/env python3

import sys
import os
import time
```

```python
# 导入i2cdev
from i2cdev import I2C

def i2cdevTest():
    # device, bus = 0x51, 0
    bus = input("Please input I2C BUS num:")
    os.system('i2cdetect -y -r ' + bus)
    device = input("Please input I2C device num(Hex):")
    print("Read data from device %s on I2C bus %s" % (device, bus))
    i2c = I2C(eval("0x" + device), int(bus))
    value = i2c.read(1)
    i2c.write(value)
    print("read value=", value)
    i2c.close()

if __name__ == '__main__':
    print("Starting demo now! Press CTRL+C to exit")
    print("List of enabled I2C controllers:")
    os.system('ls /dev/i2c*')
    while True:
        i2cdevTest()
```