Designing Embedded Systems using Micro Python

Class 5: Testing MicroPython Projects

June 14, 2019
Jacob Beningo

Course Overview

Topics:

- Designing Products with MicroPython
- Getting Started with the Pyboard D-Series
- Customizing the MicroPython Kernel for Production
- Developing Real-time Application Projects
- **Testing MicroPython Projects**

Session Overview

- Test Harnesses
- Hardware Requirements
- Software Requirements
- Testing

Presented by:

Test Harnesses

A **test harness** is a collection of software and data that is used to automatically test application modules under various conditions in order to determine whether they meet the design requirements. A test harness will often consist of three main components which include:

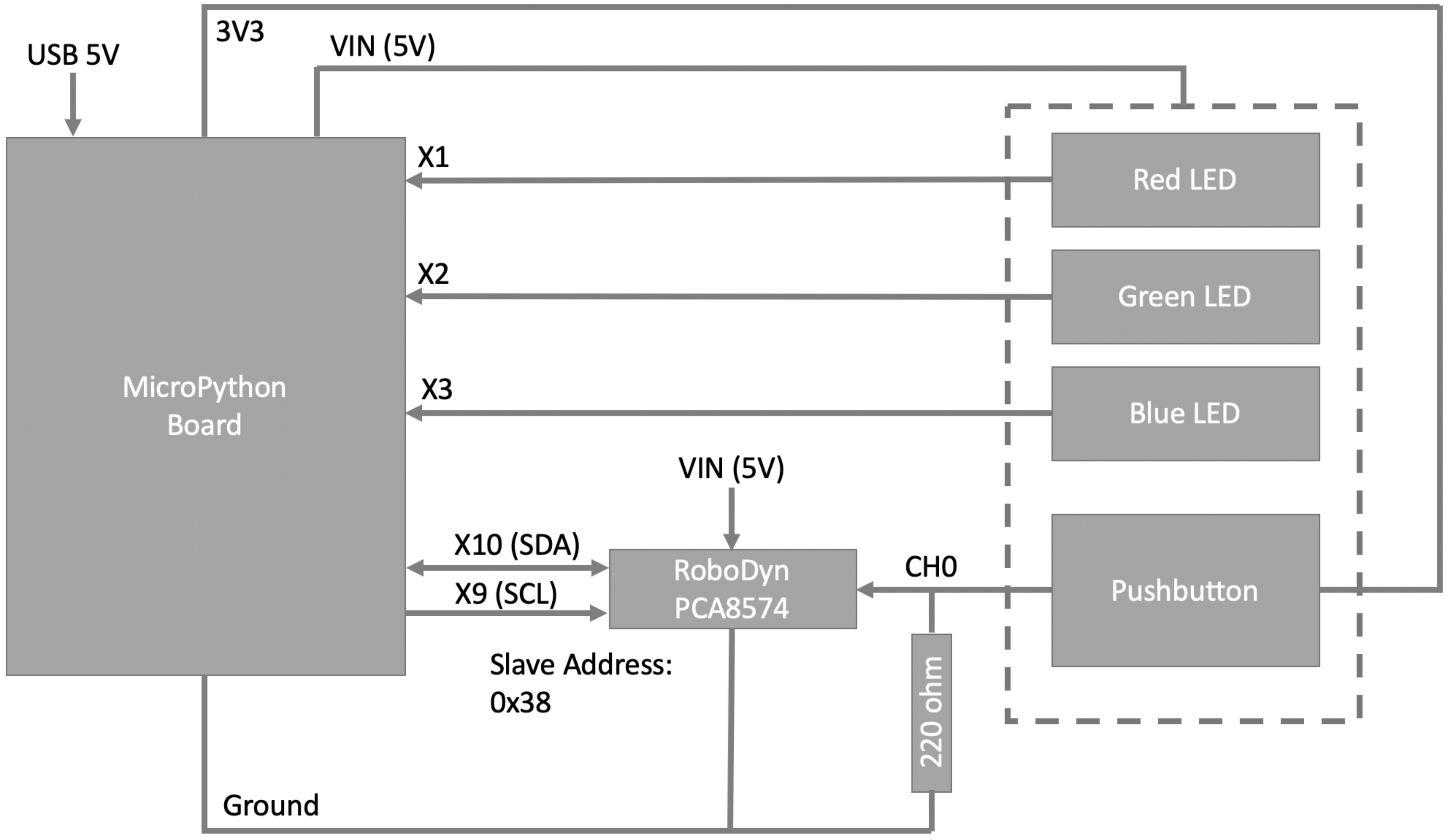
- A test execution engine
- A repository of tests
- A test reporting mechanism

Test Harnesses

A test harness can be a very powerful tool for developers and brings several advantages to the testing process such as:

- Automating testing that then free up developers to focus on other activities
- Performing regression testing which can verify recent changes haven't broken other pieces of code
- Increased code quality

TH - Hardware Requirements



TH - Hardware Requirements

Define your testing requirements

- The test harness shall be capable of recording the output signals on X1, X2 and X3 for manual review and verification.
- The test harness shall monitor I2C communication on X9 and X10.
- The test harness shall replace the pushbutton CH0 input with an I/O line controllable by the test harness.

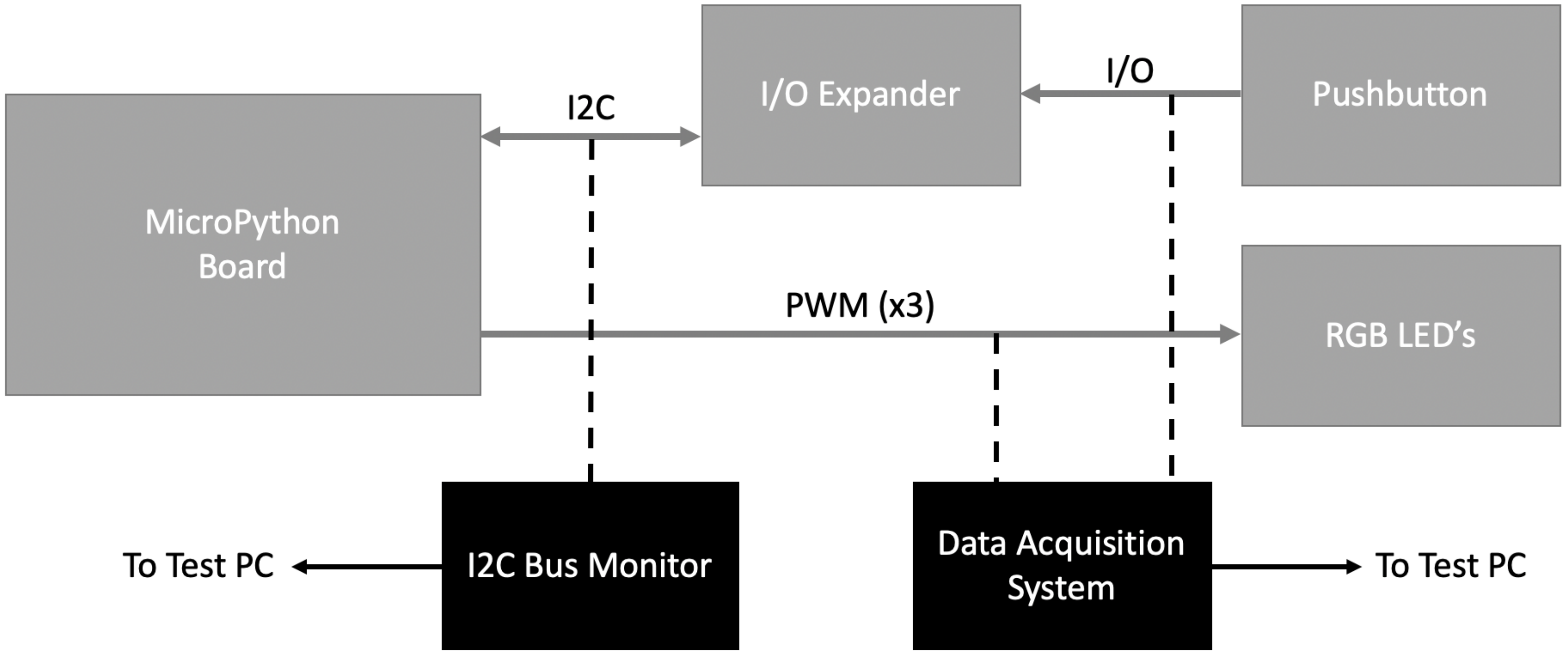
TH - Software Requirements

1. The test harness shall be configurable to run a subset of tests or the entire test suite based on the test harness configuration settings.
2. The test harness shall be modular to allow test harness features to be reused across multiple projects.
3. The test harness shall record how long it takes to perform each test along with the total test time.
4. The test harness shall generate a report which will be saved to the file system once the tests are complete that specifies

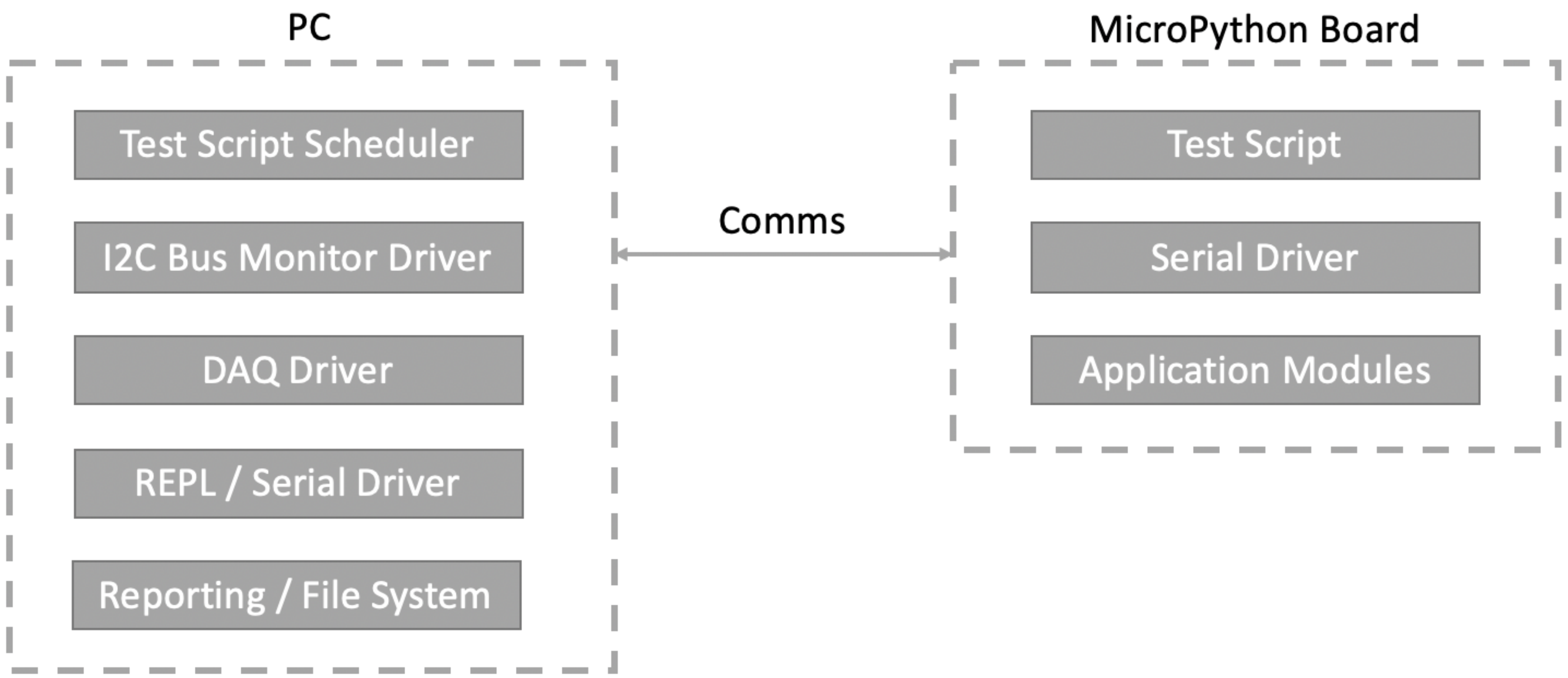
TH – Software Requirements

- Software version under test
- Test harness version
- Hardware version
- Number of tests to execute
- Test start time
- Test specific information
    - Name of test executed
    - Input parameters
    - Expected output
    - Actual output
    - Time to run the test
- Test Stop time
- Number of tests that pass
- Number of tests that fail

TH Hardware Architecture



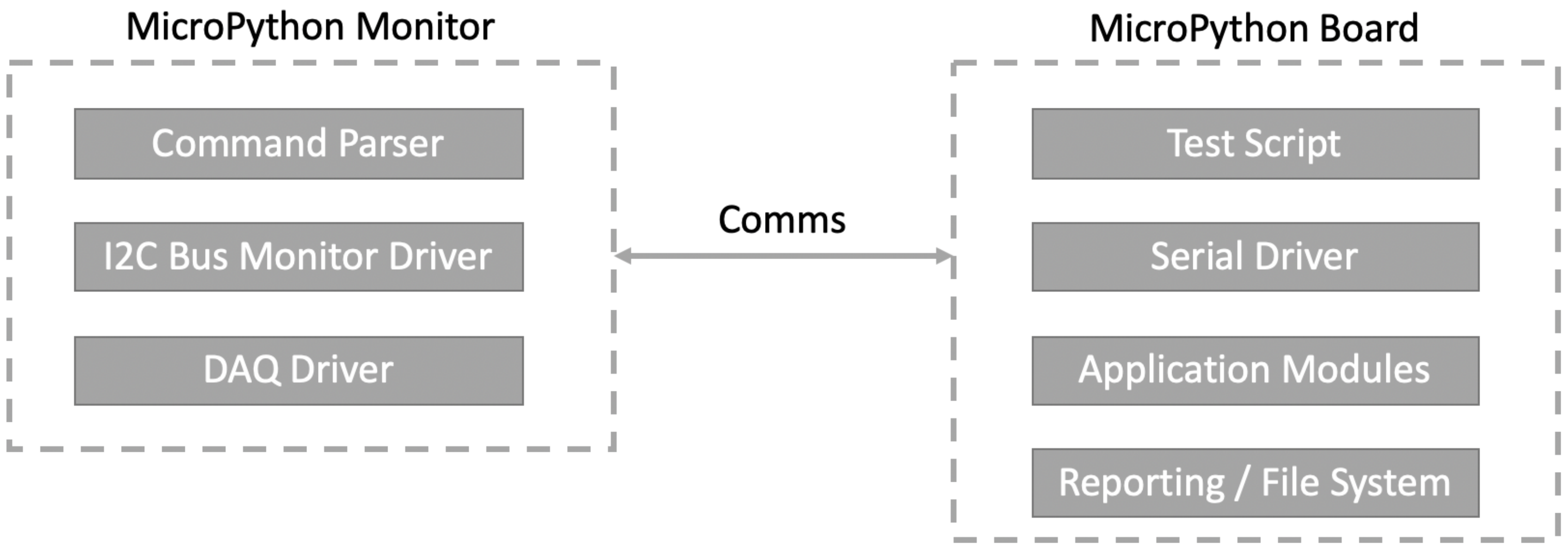
TH – Software Architecture #1



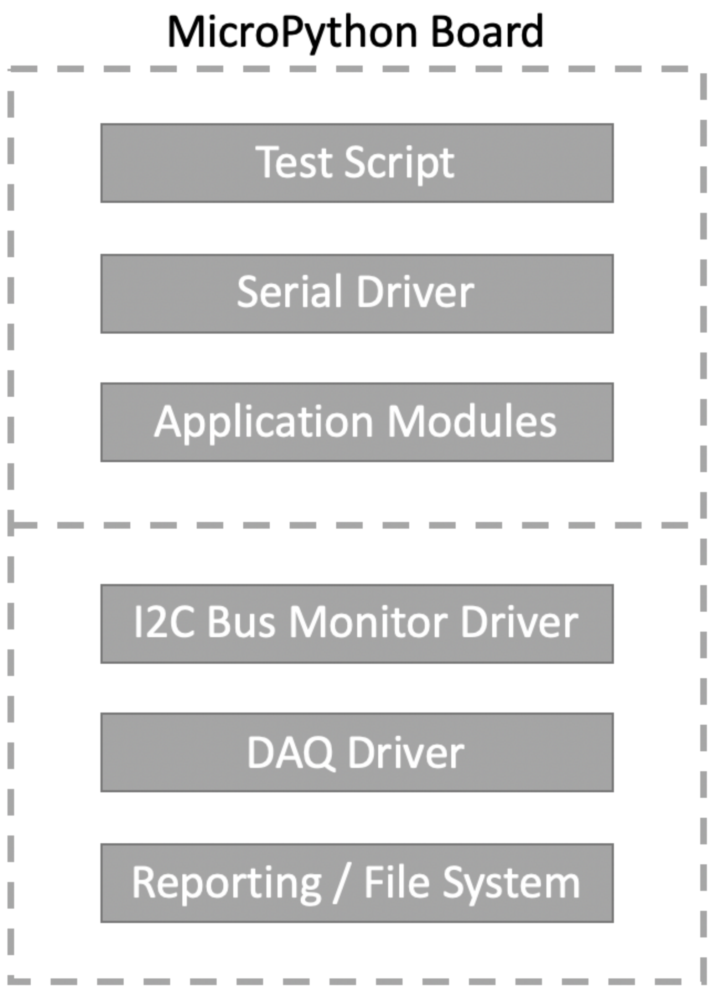
TH – Software Architecture #1

- Easily scalable
- Can support multiple external test equipment devices
- Easy access to test data
- Developers can leverage existing Python libraries to simplify and speed up development
- Portable for use in other projects

TH – Software Architecture #2



TH – Software Architecture #3



TH – Software Architecture #3

Disadvantages

- The test harness could impact the testing depending on the end application
- The test harness may use too much memory or CPU cycles
- Complexity of managing multiple functions on a single device
- Advantages
- Decreased hardware costs
- Decreased overall test harness complexity
- All the code in a single place

TH - Testing

```
try:
        # Initialize I2C 1
        i2c = I2C(I2C_BUS1, I2C.MASTER, baudrate=100000)
        # returns list of slave addresses
        I2C_List = i2c.scan()

        if I2C_List:
          print("I2C Slaves Present =", I2C_List)
        else:
          print("There are no I2C devices present! Exiting application.")
          sys.exit(0)
except Exception as e: print(e)
```

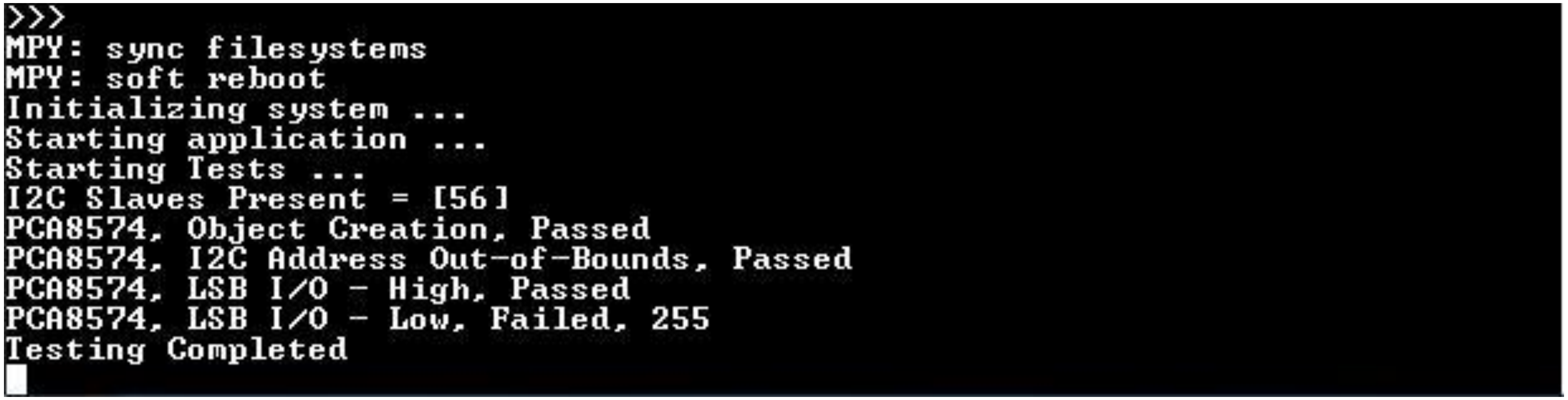
Presented by:

TH - Testing

```
# Test that we can initialize the object
        try:
                PCA8574_Object = PCA8574_IO(i2c, I2C_List[0])
                print("PCA8574, Object Creation, Passed")
        except:
                print("PCA8574, Object Creation, Failed")
```

TH - Testing

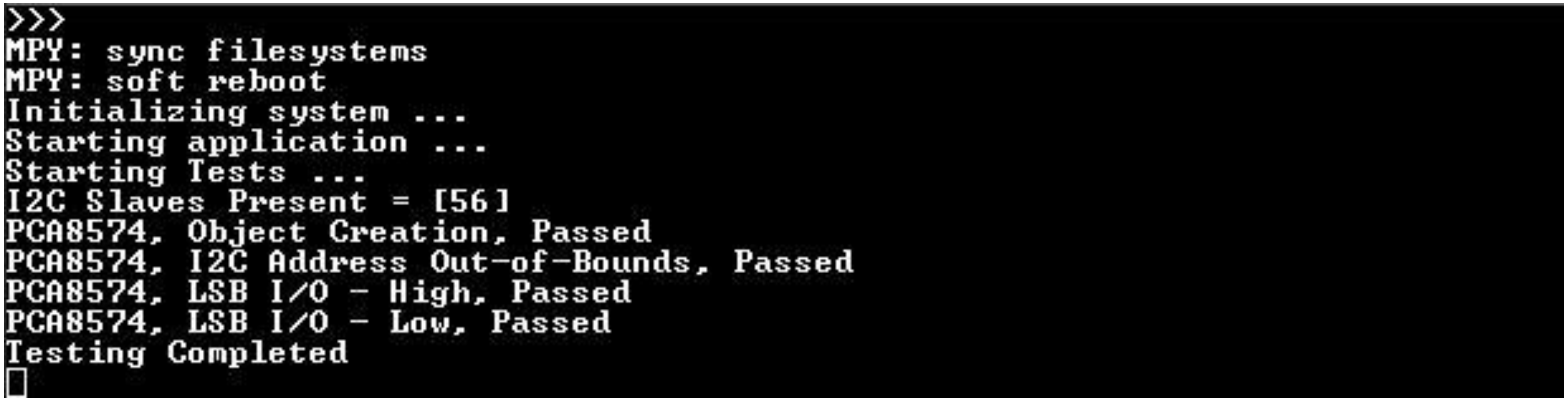
Example Failed Test:



```
>>>
MPY: sync filesystems
MPY: soft reboot
Initializing system ...
Starting application ...
Starting Tests ...
I2C Slaves Present = [56]
PCA8574, Object Creation, Passed
PCA8574, I2C Address Out-of-Bounds, Passed
PCA8574, LSB I/O - High, Passed
PCA8574, LSB I/O - Low, Failed, 255
Testing Completed
```

TH - Testing

Example Passed Tests:



```
>>>
MPY: sync filesystems
MPY: soft reboot
Initializing system ...
Starting application ...
Starting Tests ...
I2C Slaves Present = [56]
PCA8574, Object Creation, Passed
PCA8574, I2C Address Out-of-Bounds, Passed
PCA8574, LSB I/O - High, Passed
PCA8574, LSB I/O - Low, Passed
Testing Completed
```

Additional Resources

- Download Course Material for
  - http://bit.ly/MicroPythonProjects
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Designing Embedded Systems using MicroPython