



Python Support for FTDI Products Software User Guide

[Home](#) » [FTDI](#) » Python Support for FTDI Products Software User Guide 

Contents

- 1 Python Support for FTDI Products Software
- 2 Introduction
- 3 What is Python programming?
- 4 Python Driver Support
 - 4.1 VCP
 - 4.2 D2xx
- 5 Third-Party Python Examples
 - 5.1 PyUSB
 - 5.2 pylibftdi
 - 5.3 PyFtdi
 - 5.4 Adafruit
 - 5.5 ftdi-python
 - 5.6 ftd2xx ctypes Wrapper
 - 5.7 python-ft4222
- 6 Conclusion
- 7 Contact Information
- 8 Appendix A – References
- 9 Acronyms and Abbreviations
- 10 Appendix B – List of Tables and Figures
- 11 Appendix C – Serialtest.py
- 12 Appendix D– GetComPortNumber.py
- 13 Appendix E – GetDeviceInfoDetail.py
- 14 Appendix F – Revision History
- 15 CUSTOMER SERVICES
- 16 Documents / Resources
 - 16.1 References
- 17 Related Posts



The purpose of this Technical Note is to show how to use Python programming with FTDI products.

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold FTDI harmless from any and all damages, claims, suits, or expense resulting from such use.

Introduction

Python is commonly used for software development, task automation, data analysis, etc. Since it is relatively easy to learn its popularity is increasing.

This Technical Note shows ways in which Python programming can be used by FTDI products which rely on FTDI's standard [VCP Driver](#) or [D2xx Driver](#).

There are also many third-party python libraries appearing and these are listed here too.

Python support is also available for FT600/1 ICs which use the D3xx driver but is not covered in this document. Please see the product pages for these ICs for more information.

What is Python programming?

Python is becoming a very popular programming language.

It can be used for rapid prototyping, or for production-ready software development.

It works on different platforms (e.g., Windows, MacOS, Linux, etc.) and has a simple syntax like the English language. This syntax allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written which means that prototyping can be very quick.

There are lots of resources online available for Python with this being the main website: <https://www.python.org/>



Figure 1 – Python Logo

Python Driver Support

This section details how to use FTDI's VCP and D2xx driver with Python programming.

VCP

This section has a simple example to open a COM port using the VCP driver, display the COM port opened to the screen and then send and receive a few characters to the open port and display those on the screen. The output from the code is shown below. Note that an FTDI cable with TX to RX loopback was used for the simple test.

```
Opened COM port: COM7
Data received: b'Hello'
```

The code listing is found in **Appendix C – Serialtest.py**.

To run this code, you will need to install the pyserial library. You can do this by running the following command in your terminal:

pip install pyserial

Once the pyserial library is installed, you can run the code by saving it as a .py file and then running the following command in your terminal:

your_filename.py

D2xx

This section has two simple examples which use basic D2xx APIs.

- Get COM Port Number
- Get Device Info Detail

Refer to the [D2XX Programmer's Guide](#) for more information on these APIs.

Get COM Port Number

This simple example loads the d2xx library, opens the first FTDI device found, finds the COM Port number using FT_GetComPortNumber and then closes it.

The code listing is found in **Appendix D– GetComPortNumber.py**.

When executed the output looks like this:

```
===== Python D2XX Get Com Port =====
D2XX library loaded OK
Com Port Number: 6
```

Get Device Info Detail

This simple example loads the d2xx library and uses FT_Create Device Info List and FT_Get Device Info Detail to get the FTDI device details and lists them.

The code listing is found in **Appendix E – GetDeviceInfoDetail.py**.

When executed the output looks like this:

```
===== Python D2XX Get Device Info Detail =====
D2XX library loaded OK
Number of devices is: 1
Dev: 0
  Flags=0x0
  Type=0x5
  ID=0x4036001
  LocId=0x15
  SerialNumber=AUDG9VX7
  Description=FT232R USB UART
  ftHandle=0x0
```

Third-Party Python Examples

There are many third-party libraries and examples appearing as the programming language popularity expands, some are listed here but more may be found.

PyUSB

PyUSB provides easy access to the host machine's Universal Serial Bus (USB) system for Python 3. PyUSB is an API rich, backend neutral Python USB module easy to use. <https://pyusb.github.io/pyusb/>

pylibftdi

Please note that this uses [libftdi](#) which is not an FTDI supported driver however can be used to control FTDI devices.

pylibftdi is a minimal Pythonic interface to FTDI devices. <https://pypi.org/project/pylibftdi/>

PyFtdi

PyFtdi aims at providing a user-space driver for popular FTDI devices, implemented in pure Python language.

<https://pypi.org/project/pyftdi/>

Adafruit

Adafruit has created a guide to show how to use an FT232H to connect to I2C and SPI sensors and breakouts from your desktop PC running Windows, Mac OSX, or Linux.

<https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h>

ftdi-python

This example demonstrates a step-by-step approach to driving FTDI ICs from Python to learn about their functionality.

<https://iosoft.blog/ftdi-python/>

ftd2xx ctypes Wrapper

ftd2xx is a simple python wrapper around the D2XX DLL from FTDI using ctypes. <https://pypi.org/project/ftd2xx/>

python-ft4222

This example provides python binding to LibFT4222 which must be used with FT4222H IC and provides a similar API to LibFT4222. <https://pypi.org/project/ft4222/>

Conclusion

This Technical Note shows how to use Python programming with FTDI products with the VCP and D2xx driver and has listed some third-party examples.

Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited (UK)
Unit 1, 2 Seaward Place, Centurion Business Park Glasgow G41 1HH United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com

E-mail (Support) support1@ftdichip.com

E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan, R.O.C.
Tel: +886 (0) 2 8797 1330
Fax: +886 (0) 2 8751 9737

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited (USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 098

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 1103, No. 666 West Huaihai Road, Shanghai, 200052 China
Tel: +86 (21) 62351596
Fax: +86 (21) 62351595
E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.c

Web Site <http://ftdichip.com>

Distributor and Sales Representatives

Please visit the Sales Network page of the [FTDI Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold harmless FTDI from any and all damages, claims, suits, or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted, or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640.

Appendix A – References

Document References

[D2xx Driver](#)
[VCP Driver](#)
<https://www.python.org/>
[D2XX Programmer's Guide](#)
<https://pyusb.github.io/pyusb/>
<https://pypi.org/project/pylibftdi/>
<https://pypi.org/project/pyftdi/>
<https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h>

<https://iosoft.blog/ftdi-python/>
<https://pypi.org/project/ftd2xx/>
<https://pypi.org/project/ft4222/>
[libftdi](#)

Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
IC	Integrated Circuit
USB	Universal Se
VCP	Virtual COM Port

Appendix B – List of Tables and Figures

List of Tables

N/A

List of Figures

Figure 1 – Python Logo..... 5

Appendix C – Serialtest.py

```
import serial
import time
# Set the COM port number
portName = 'COM7'
# Set the baud rate
baudRate = 115200
# Open the serial port
try:
    ser = serial.Serial(portName, baudRate)
    print('Opened COM port: ' + portName)
except:
    print("An exception occurred")

# Send 'Hello' to start the program
ser.write(b'Hello')
time.sleep(1)
# Read line
read_data = ser.read(5)
print("Data received: " + str(read_data))
# Close the serial port
ser.close
```

Appendix D– GetComPortNumber.py

```
#!/usr/bin/env python
import os
import sys
import ctypes
```

```
#####
# D2XX definitions
def check(f):
    if f != 0:
        names = [
            "FT_OK",
            "FT_INVALID_HANDLE",
            "FT_DEVICE_NOT_FOUND",
            "FT_DEVICE_NOT_OPENED",
            "FT_IO_ERROR",
            "FT_INSUFFICIENT_RESOURCES",
            "FT_INVALID_PARAMETER",
            "FT_INVALID_BAUD_RATE",
            "FT_DEVICE_NOT_OPENED_FOR_ERASE",
            "FT_DEVICE_NOT_OPENED_FOR_WRITE",
            "FT_FAILED_TO_WRITE_DEVICE",
            "FT_EEPROM_READ_FAILED",
            "FT_EEPROM_WRITE_FAILED",
            "FT_EEPROM_ERASE_FAILED",
            "FT_EEPROM_NOT_PRESENT",
            "FT_EEPROM_NOT_PROGRAMMED",
            "FT_INVALID_ARGS",
            "FT_NOT_SUPPORTED",
            "FT_OTHER_ERROR"] raise IOError("Error: (status %d: %s)" % (f, names[f]))
#####
# Main Program
#
# Implements simple GetComPortNumber example from D2XX programmers guide.

class D2XXTest(object):
    def __init__(self):
        #Load driver binaries
        if sys.platform.startswith('linux'):
            self.d2xx = ctypes.cdll.LoadLibrary("libftd2xx.so")
        elif sys.platform.startswith('darwin'):
            self.d2xx = ctypes.cdll.LoadLibrary("libftd2xx.1.1.0.dylib")
        else:
            self.d2xx = ctypes.windll.LoadLibrary("ftd2xx")
        print 'D2XX library loaded OK'
        print
        sys.stdout.flush()
        self.getCom()
        def getCom(self):
            #create FT Handle variable
            self.ftHandle = ctypes.c_void_p()
            #Open the first device on the system
            check(self.d2xx.FT_Open(0, ctypes.byref(self.ftHandle)))
            #com port number variable
            IComPortNumber = ctypes.c_long()
            #retrieve com # with FT_GetComPortNumber
            check(self.d2xx.FT_GetComPortNumber(self.ftHandle,
            ctypes.byref(IComPortNumber)))
            if IComPortNumber.value == -1:
                print "No Com Port Assigned"
            else:
                print "Com Port Number: %d" % (IComPortNumber.value)
            #call FT_Close to close connection
            check(self.d2xx.FT_Close(self.ftHandle))
        if __name__ == '__main__':
```

```

print "===== Python D2XX Get Com Port ====="
print
app = D2XXTest()

```

Appendix E – GetDeviceInfoDetail.py

```

#!/usr/bin/env python
import os
import sys
import ctypes

#####
# D2XX definitions
def check(f):
    if f != 0:
        names = [
            "FT_OK",
            "FT_INVALID_HANDLE",
            "FT_DEVICE_NOT_FOUND",
            "FT_DEVICE_NOT_OPENED",
            "FT_IO_ERROR",
            "FT_INSUFFICIENT_RESOURCES",
            "FT_INVALID_PARAMETER",
            "FT_INVALID_BAUD_RATE",
            "FT_DEVICE_NOT_OPENED_FOR_ERASE",
            "FT_DEVICE_NOT_OPENED_FOR_WRITE",
            "FT_FAILED_TO_WRITE_DEVICE",
            "FT_EEPROM_READ_FAILED",
            "FT_EEPROM_WRITE_FAILED",
            "FT_EEPROM_ERASE_FAILED",
            "FT_EEPROM_NOT_PRESENT",
            "FT_EEPROM_NOT_PROGRAMMED",
            "FT_INVALID_ARGS",
            "FT_NOT_SUPPORTED",
            "FT_OTHER_ERROR"] raise IOError("Error: (status %d: %s)" % (f, names[f]))
#####
# Main Program

###***** maybe add boolens for operating system so when you try and open a device you can do it right for the
right OS. Linux cant use indexs to open (?) check linux examples maybe?
class D2XXTest(object):
    def __init__(self):
        #Load driver binaries
        if sys.platform.startswith('linux'):
            self.d2xx = ctypes.cdll.LoadLibrary("libftd2xx.so")
        elif sys.platform.startswith('darwin'):
            self.d2xx = ctypes.cdll.LoadLibrary("libftd2xx.1.1.0.dylib")
        else:
            self.d2xx = ctypes.windll.LoadLibrary("ftd2xx")
        print "D2XX library loaded OK\n"
        sys.stdout.flush()
        #call example fuction
        self.getDevInfoList()
        def getDevInfoList(self):
            #declare vairables needed in function
            numDevs = ctypes.c_long()
            check(self.d2xx.FT_CreateDeviceInfoList(ctypes.byref(numDevs)))
            print "Number of devices is: %d" % (numDevs.value)
            # if there is at least one device connected
            if numDevs.value > 0:

```



```

#obtain device info for all devices on the system
for i in range (numDevs.value):
#create FT Handle variable
ftHandleTemp = ctypes.c_long()
Flags = ctypes.c_long()
ID = ctypes.c_long()
Type = ctypes.c_long()
LocId = ctypes.c_long()
SerialNumber = ctypes.create_string_buffer(16)
Description = ctypes.create_string_buffer(64)

#call GetDeviceInfoDetail function to obtain device

details
check(self.d2xx.FT_GetDeviceInfoDetail(i,
ctypes.byref(Flags),ctypes.byref(Type), ctypes.byref(ID), ctypes.byref(LocId),
ctypes.byref(SerialNumber), ctypes.byref(Description), ctypes.byref(ftHandleTemp)))
#print the device details
self.printDetails(i,Flags.value, Type.value, ID.value,
LocId.value, SerialNumber.value, Description.value, ftHandleTemp.value)
else:
#if no devices exit the program
sys.exit()
def printDetails(self,dev,flags,ty,i_d,locid,serial,desc,handle):
print "Dev: %d" % (dev)
print " Flags=0x%x" % (flags)
print " Type=0x%x" % (ty)
print " ID=0x%x" % (i_d)
print " LocId=0x%x" % (locid)
print " SerialNumber=%s" % (serial)
print " Description=%s" % (desc)
print " ftHandle=0x%s" % (handle)
if __name__ == '__main__':
print "===== Python D2XX Get Device Info Detail =====\n"
app = D2XXTest()

```

Appendix F – Revision History

Document Title: TN_187 Python Support for FTDI Products

Document Reference No.: FT_001561

Clearance No.: FTDI#590

Product Page: <https://ftdichip.com/product-category/products/ic/>

Document Feedback: [Send Feedback](#)

CUSTOMER SERVICES

[Product Page](#)

[Document Feedback](#)

Copyright © Future Technology Devices International Limited Future Technology Devices International Limited
(FTDI) Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom


Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

Web Site: <http://ftdichip.com>
















Copyright © Future Technology Devices International Limited



Documents / Resources

	FTDI Python Support for FTDI Products Software [pdf] User Guide FT4222H, Python Support for FTDI Products Software, Support for FTDI Products Software, FT DI Products Software, Software
---	---

References

-  [Home - FTDI](#)
-  [Home - FTDI](#)
-  [Sales Network - FTDI](#)
-  [D2XX Drivers - FTDI](#)
-  [VCP Drivers - FTDI](#)
-  [IC - FTDI](#)
-  [Programming FTDI devices in Python – Lean2](#)
-  [Overview | CircuitPython Libraries on any Computer with FT232H | Adafruit Learning System](#)
-  [ft4222 · PyPI](#)
-  [ftd2xx · PyPI](#)
-  [pyftdi · PyPI](#)
-  [pylibftdi · PyPI](#)
-  [PyUSB](#)
-  [libFTDI » FTDI USB driver with bitbang mode](#)
-  [Welcome to Python.org](#)
- [User Manual](#)

[Manuals+](#), [Privacy Policy](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.