Application Note

AN_444

Using the FT260 with Linux

Version 1.0

Issue Date: 2020-04-28

This document shows the user how to configure the FT260 HID chip with the Linux operating system.

Table of Contents

1 Introduction

This document illustrates how to use built-in Linux HID libraries to configure the FTDI FT260 bridge device.

1.1 Overview

This application note will introduce the user to the Linux HID libraries, how HID reports work, how to configure FT260 descriptors using HID reports, and how to perform read/write operations using HID reports. The UMFT260EV module and the C232HD UART cable will be used.

1.2 Scope

This document will cover how to configure the FT260 as a USB to UART bridge. Other FT260 capabilities such as I2C Master and GPIO will not be covered.

1.3 Hardware Required

The following FTDI modules and cables will be used in this application note:

- UMFT260EV1A development module
- C232HD-DDHSP-0 High Speed UART cable
- Linux PC running Ubuntu or other popular Linux distro.

2 Introducing HID Reports

Most people are familiar with USB keyboards and mice as human interface devices (HID class). These devices use report files to communicate with the host PC using the HID class driver that is built into all operating systems. The advantage of using the standard HID driver with a USB bridge device is that you don't need a proprietary driver to interface with the FTDI bridge chip.

All HID Class devices (including the FT260) use Interrupt endpoints, in contrast to standard FT2XX chips that use Bulk endpoints

There are three types of USB HID reports – feature reports, input reports and output reports.

Feature reports are used by the FT260 to set descriptor opcodes – how to configure the FT260 to function in UART, I2C Master, or GPIO modes.

Input and output reports are used by the FT260 to send and receive user data.

2.1 USB HID Libraries for Linux

The HIDRAW and LIBUSB libraries will be used in the demo code.

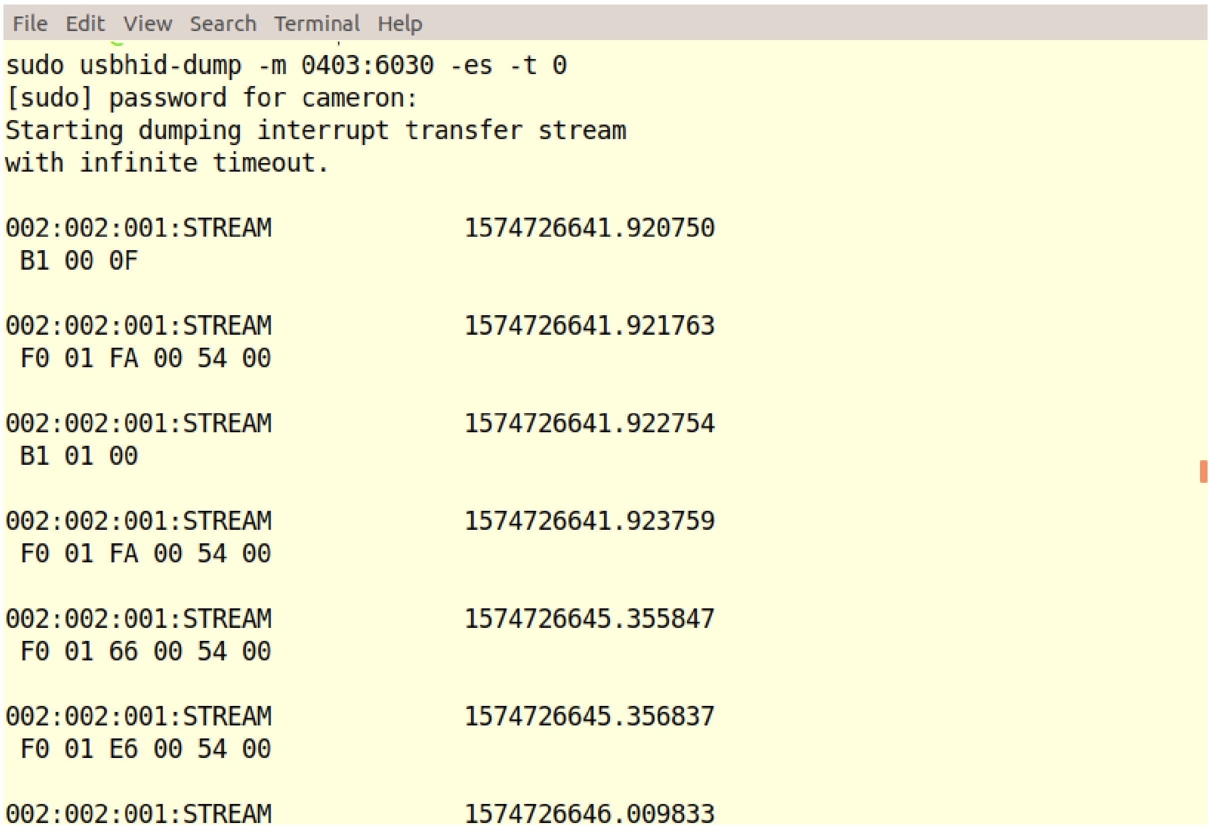
3 USBHID-Dump: A Simple Linux Command Line example

This demo assumes there is streaming UART data applied to the FT260's RXD pin at 9600 baud.

With the UMFT260EV1 plugged in, open a Linux terminal window, and enter the following:

sudo usbhid-dump -m 0403:6030 -es -t 0 <ret>

The commands arguments will look for a HID device with a VID/PID of 0x0403/0x6030, to dump data as a string (-es), and stream data continuously with no timeout (-t 0)



```
sudo usbhid-dump -m 0403:6030 -es -t 0
[sudo] password for cameron:
Starting dumping interrupt transfer stream
with infinite timeout.

002:002:001:STREAM             1574726641.920750
 B1 00 0F

002:002:001:STREAM             1574726641.921763
 F0 01 FA 00 54 00

002:002:001:STREAM             1574726641.922754
 B1 01 00

002:002:001:STREAM             1574726641.923759
 F0 01 FA 00 54 00

002:002:001:STREAM             1574726645.355847
 F0 01 66 00 54 00

002:002:001:STREAM             1574726645.356837
 F0 01 E6 00 54 00

002:002:001:STREAM             1574726646.009833
```

Figure 3.1 Output of usbhid-dump command

This will show the HID report being updated as new bytes of data are received on the RXD pin of the UMFT260EV module.

Note that the ASCII data received is raw data and does not appear 1:1 in the streaming HID report, and requires interpretation. This command was tested with a non-configured UMFT260, and does not show correctly mapped ASCII data. The FT260 requires configuration via USB HID reports from the host PC.

4 HID UART Write Example in C

The **hid_uartcode_wr.c** code example uses the existing HIDRAW libraries built into Linux.

The code first identifies the FT260 using an IOCTL call and reads its descriptors. Next, another IOCTL call is used to send a feature report to configure the FT260 into a specific UART mode. Finally, the code uses a simple output report to send a UART data stream from the Linux host to the FT260.

The specific opcodes used to configure the UART registers can be found in AN_394 "User Guide for FT260". A link for this document can also be found in Appendix A.

This code can be tested using our C232HD cable with the UMFT260 module, and a TTY app.

All source code in this document can be downloaded from the FT260 Linux Code Examples link in Appendix A – References.

```
#include <linux/types.h>
#include <linux/input.h>
#include <linux/hidraw.h>
/*
* For the systems that don't have the new version of hidraw.h in user space.
*/
#ifndef HIDIOCSFEATURE
#warning please have your distro update the userspace kernel headers
#define HIDIOCSFEATURE(len)
_IOC(_IOC_WRITE|_IOC_READ, 'H', 0x06, len)
#define HIDIOCGFEATURE(len)
_IOC(_IOC_WRITE|_IOC_READ, 'H', 0x07, len)
#endif


#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

const char* bus_type_str(int bus) {
switch (bus) {
case BUS_USB:
return "USB";
case BUS_HIL:
return "HIL";
case BUS_BLUETOOTH: return "Bluetooth";
case BUS_VIRTUAL:
return "Virtual";
default: return "Other";
}
}
int main(int argc, char **argv) {
int fd;
int res, desc_size = 0;
char buf[256];
```

```
struct hidraw_report_descriptor rpt_desc;
struct hidraw_devinfo info;
char* device = "/dev/hidraw1"; // was hidraw0 default
if (argc > 1) {
device = argv[1];
}
/* Open the Device with non-blocking reads. */
/* It will be better if use libudev instead of hard coded path.
You can check Appendix A for the example of using libudev */

fd = open(device, O_RDWR|O_NONBLOCK);
if (fd < 0) {
perror("Unable to open device");
return 1;
}
memset(&rpt_desc, 0x0, sizeof(rpt_desc));
memset(&info, 0x0, sizeof(info));
memset(buf, 0x0, sizeof(buf));

/* Demo Instruction */
printf("Connect a TTL-232-WE or C232HD cable to TXD/RXD pins on UMFT260 module \n
\n");
printf("Open a Linux (or Windows) TTY and point to ttyUSB0, configure to 9600-8-N-1 \n
\n");
printf("If no data is seen on TTY, make sure you have selected the correct raw HID
device port on line 45 \n \n");

/* Get Report Descriptor Size */
res = ioctl(fd, HIDIOCGRDESCSIZE, &desc_size);
if (res < 0) {
perror("HIDIOCGRDESCSIZE");
} else {
printf("Report Descriptor Size: %d\n", desc_size);
}

/* Get Raw Name */
res = ioctl(fd, HIDIOCGRAWNAME(256), buf);
if (res < 0) {
perror("HIDIOCGRAWNAME");
} else {
printf("Raw Name: %s\n", buf);
}

/* Get Raw HID Descriptors */
res = ioctl(fd, HIDIOCGRAWINFO, &info);
if (res < 0) {
perror("HIDIOCGRAWINFO");
} else {
printf("Raw Info:\n");
printf("\tbustype: %d (%s)\n",
info.bustype, bus_type_str(info.bustype));
printf("\tvendor: 0x%04hx\n", info.vendor);
printf("\tproduct: 0x%04hx\n", info.product);
}

/* Send Feature Report to Configure FT260 as UART 9600-8-N-1 */

buf[0] = 0xA1; /* SYSTEM_SETTING_ID */
buf[1] = 0x41; /* Configure UART Mode */
buf[2] = 0x04; /* No Flow Control */
```

```
buf[3] = 0x80; /* BaudRate 9600 */
buf[4] = 0x25; /* BaudRate 9600 */
buf[5] = 0x00; /* BaudRate 9600 */
buf[6] = 0x00; /* BaudRate 9600 */
buf[7] = 0x08; /* 8 data bits */
buf[8] = 0x00; /* No Parity bit */
buf[9] = 0x00; /* One Stop bit */
buf[10] = 0x00; /* Break char (none) */
res = ioctl(fd, HIDIOCSFEATURE(11), buf);

/* IOCTL writes Feature Report to the driver */
if (res < 0) {
perror("HIDIOCSFEATURE");
} else {
printf("ioctl HIDIOCGFEATURE returned: %d\n", res);
}

/* Send Output Report (Data Payload) to the FT260 Device */
buf[0] = 0xF6; /* UART write 28 bytes (60 bytes max) */
buf[1] = 0x18; /* 24 Bytes Data */
buf[2] = 0x23; /* Byte 0 # */
buf[3] = 0x46; /* Byte 1 F */
buf[4] = 0x54; /* Byte 2 T */
buf[5] = 0x32; /* Byte 3 2 */
buf[6] = 0x36; /* Byte 4 6 */
buf[7] = 0x30; /* Byte 5 0 */
buf[8] = 0x20; /* Byte 6   */
buf[9] = 0x44; /*  Byte 7 D */
buf[10] = 0x65; /* Byte 8 e */
buf[11] = 0x6D; /* Byte 6 m  */
buf[12] = 0x6F; /* Byte 6 o */
buf[13] = 0x0D; /* Byte 8 <CR> */
buf[14] = 0x0A; /* Byte 9 <LF>*/
buf[15] = 0x46; /* Byte 10 F */
buf[16] = 0x54; /* Byte 11 T */
buf[17] = 0x44; /* Byte 12 D */
buf[18] = 0x49; /* Byte 13 I */
buf[19] = 0x20; /* Byte 14 <SP>*/
buf[20] = 0x55; /* Byte 15 U */
buf[21] = 0x53; /* Byte 16 S */
buf[22] = 0x41; /* Byte 17 A */
buf[23] = 0x0D; /* Byte 18 <CR>*/
buf[24] = 0x0A; /* Byte 19 <LF>*/

res = write(fd, buf, 26);
if (res < 0) {

printf("Error: %d\n", errno);
perror("write");
} else {
printf("write() wrote %d bytes\n", res);
}
close(fd);
return 0;
}
```

You can compile this code with a simple gcc command in a Linux terminal window:

```
gcc  hid_uartcode_wr.c
```

This will create an executable file called a.out. Run this file as follows:

```
sudo ./a.out
```

4.1 Hardware Setup for UART Write Demo

First, connect pins 2 and 3 of jumpers JP7 and JP9. This set the device to UART mode.

Connect the UMFT260 module to the TXD and RXD wires of the C232HD UART cable as shown in Figure 4.1.

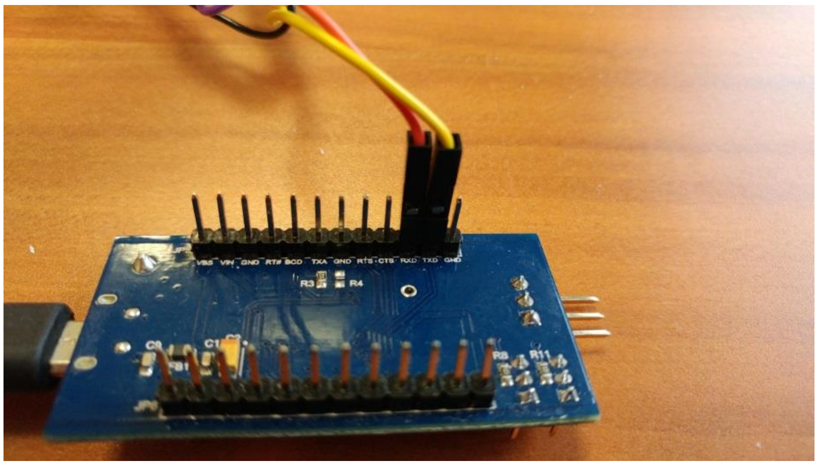


Figure 4.1 Connecting UMFT260 Module RX/TX pins to C232HD cable

4.2 Application SW used in the UART Write demo

Any popular TTY application (TeraTerm, CoolTerm, GtkTerm) can be used to receive data from the hid_uartcode_wr demo code. The terminal (TTY) session will display data received from the UMFT260 module.

4.3 Running the Demo

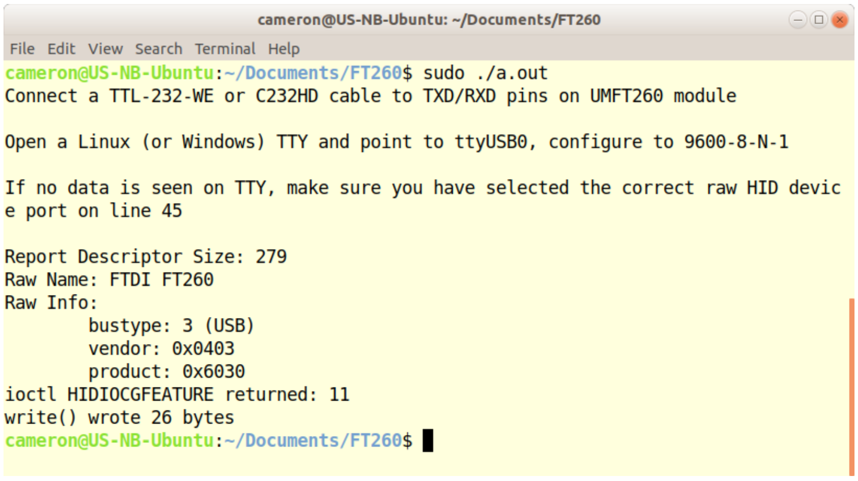
Plug the UMFT260EV module into a Linux PC and connect the TXD/RXD pins to the C232HD cable as specified in Figure 4.1.

Open up a TTY session and configure the TTY for the C232HD COM port, and set the serial interface options to 9600-8-N-1. Open the port.

Note: If you are using GtK Term on Linux, make sure to make the /dev/ttyUSB0 COM port accessible by using the following command:

```
sudo chmod 777 ttyUSB0 <ret>
```

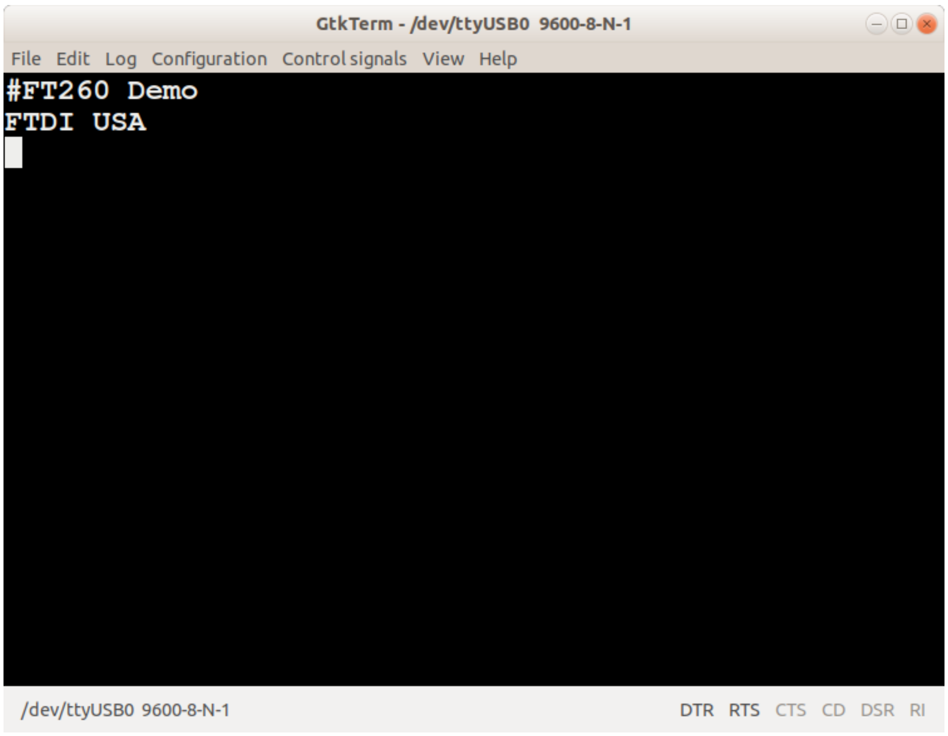
On the Linux PC, go to the FT260 code directory and run the a.out code as shown in **Figure 4.2**.


cameron@US-NB-Ubuntu: ~/Documents/FT260
File Edit View Search Terminal Help

```
cameron@US-NB-Ubuntu:~/Documents/FT260$ sudo ./a.out
Connect a TTL-232-WE or C232HD cable to TXD/RXD pins on UMFT260 module

Open a Linux (or Windows) TTY and point to ttyUSB0, configure to 9600-8-N-1

If no data is seen on TTY, make sure you have selected the correct raw HID devic
e port on line 45

Report Descriptor Size: 279
Raw Name: FTDI FT260
Raw Info:
        bustype: 3 (USB)
        vendor: 0x0403
        product: 0x6030
ioctl HIDIOCGFEATURE returned: 11
write() wrote 26 bytes
cameron@US-NB-Ubuntu:~/Documents/FT260$
```

Figure 4.2 Running the Linux HID Write application

With the GtK Terminal pointing to /dev/ttyUSB0, set to 9600-8-N-1, you should see the data payload from the UART HID Write code:


GtkTerm - /dev/ttyUSB0 9600-8-N-1
File Edit Log Configuration Control signals View Help
/dev/ttyUSB0 9600-8-N-1
DTR RTS CTS CD DSR RI

```
#FT260 Demo
FTDI USA
```

Figure 4.3 UART Terminal App showing received FT260 data

4.3.1 HID Dump Revisited

If you go back and run the usbhid-dump command from Section 3, you can type in characters from the TTY terminal and see them appear as standard ASCII characters, as you have currently set the FT260's UART descriptors to 9600-8-N-1. However, if you unplug the UMFT260EV module, the current UART descriptors will be lost.

5 FT260 Linux Loopback Code

The FT260 developers have provided a more advanced HID application (hid_260_udev.c) that runs a UART loopback test using the FT260. Since the source code is approximately 400 lines long, a code listing is not included in this application note.

All source code in this document can be downloaded from the FT260 Linux Code Examples link in Appendix A – References.

Required Libraries to run the code example

Before running the Make command, you will need to install libusb and libudev.

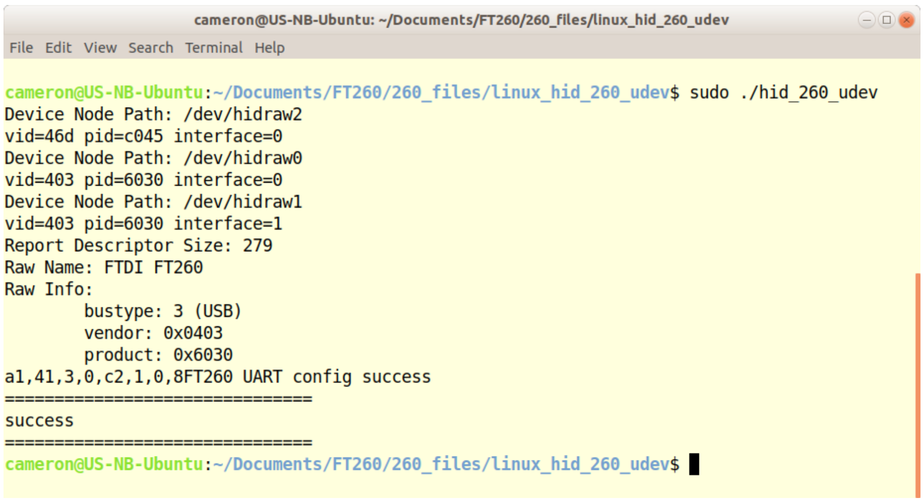
```
sudo apt-get install libusb-1.0-0-dev  <ret>
sudo apt-get install libudev-dev  <ret>
```

5.1 Hardware Setup for UART Loopback Code

On the UMFT260EV module, simply jumper the RXD and TXD pins.

5.2 Compile and run the code

In the Loopback code directory, just run the make command and then run the hid_260_udev code as shown in **Figure 5.1**.


cameron@US-NB-Ubuntu: ~/Documents/FT260/260_files/linux_hid_260_udev
File Edit View Search Terminal Help

```
cameron@US-NB-Ubuntu:~/Documents/FT260/260_files/linux_hid_260_udev$ sudo ./hid_260_udev
Device Node Path: /dev/hidraw2
vid=46d pid=c045 interface=0
Device Node Path: /dev/hidraw0
vid=403 pid=6030 interface=0
Device Node Path: /dev/hidraw1
vid=403 pid=6030 interface=1
Report Descriptor Size: 279
Raw Name: FTDI FT260
Raw Info:
        bustype: 3 (USB)
        vendor: 0x0403
        product: 0x6030
a1,41,3,0,c2,1,0,8FT260 UART config success
==============================
success
==============================
cameron@US-NB-Ubuntu:~/Documents/FT260/260_files/linux_hid_260_udev$ 
```

Figure 5.1 Ouput of Loopback Code

6 Conclusion

This app note has presented an introduction to HID reports, which consist of feature reports for setting device descriptors, and output and input reports for sending and receiving data payloads.

Several code examples are given utilizing the UART functionality of the FT260. By referring to the FT260 descriptor opcodes described in AN_394, the user can also develop their own I2C and GPIO code examples.

7 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
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 – 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 0987

E-Mail (Sales)	us.sales@ftdichip.com
E-Mail (Support)	us.support@ftdichip.com
E-Mail (General Enquiries)	us.admin@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 8791 3570
Fax: +886 (0) 2 8791 3576

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

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.

Appendix A – References

Document References

DS_FT260 Datasheet
AN_394 User Guide for FT260
UMFT260EV1A module
C232HD-DDHSP High Speed UART Cable

Software Links

FT260 Linux Code Examples

Acronyms and Abbreviations

Terms	Description
HID	USB Human Interface Device class.
Opcode	Hex values that are used to set UART/I2C/GPIO descriptors in the FT260.
Reports	Data structures that are used to configure and communicate with HID class devices.
TTY App	Serial Terminal software for interfacing with COM ports. Examples include Tera Term, Cool Term, PuTTY, and GTK Term.
USB	Universal Serial Bus.

Appendix B – List of Figures

List of Figures

List of Figures

NA

Appendix C – Revision History

Document Title:	AN_444 Using the FT260 with Linux
Document Reference No.:	FT_001457
Clearance No.:	FTDI#551
Product Page:	https://www.ftdichip.com/Products/ICs/FT260.html
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	2020-04-28