



# Application Note

## AN\_380

# AN\_380 FT900 Bootloader DFU Usage

Version 1.0

Issue Date: 2015-10-13

This application note describes how to utilize the USB DFU feature found in the FT900 bootloader to flash programs via USB.

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.

**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 © 2015 Future Technology Devices International Limited



## Table of Contents

<b>1 Introduction</b>	<b>3</b>
1.1 Overview	3
1.2 Scope	3
<b>2 Package Contents</b>	<b>4</b>
2.1 Required Hardware	4
<b>3 Building and installation of dfu-util</b>	<b>5</b>
3.1 Installation of MinGW and MSYS	5
3.2 Compiling libusb and dfu-util	5
3.2.1 Compiling libusb	5
3.2.2 Compiling dfu-util	5
<b>4 FT900 Board Setup</b>	<b>7</b>
<b>5 Flashing the Bootloader on to FT900</b>	<b>8</b>
<b>6 dfu-util</b>	<b>9</b>
6.1 Windows Driver Signature Enforcement	10
6.2 WinUSB driver installation	10
6.3 Downloading Firmware to the FT900	14
6.4 DFU_application.bin	14
6.5 Preparation for DFU download	16
6.5.1 Padding and Checksum Addition	16
6.5.2 DFU Suffix	17
<b>7 Customizing the Bootloader</b>	<b>18</b>
<b>8 Source Code</b>	<b>19</b>
8.1 DFU_Application.bin (DFU_application.c)	19
8.2 DFU.h	20
<b>9 Tera Term Configuration</b>	<b>22</b>
<b>10 Contact Information</b>	<b>25</b>
<b>Appendix A</b>	<b>26</b>



<b>Document References .....</b>	<b>26</b>
<b>Acronyms and Abbreviations.....</b>	<b>26</b>
<b>Appendix B – List of Tables &amp; Figures .....</b>	<b>27</b>
<b>List of Figures .....</b>	<b>27</b>
<b>List of Tables.....</b>	<b>27</b>
<b>Appendix C – Revision History .....</b>	<b>28</b>

## 1 Introduction

USB Device Firmware Upgrade (DFU) is an official USB device class specification of the USB Implementers Forum. It specifies a vendor and device independent way of updating the firmware of a USB device. The idea is to have only one vendor-independent firmware update tool as part of the operating system, which can then (given a particular firmware image) be downloaded into the device. In addition to firmware download, it also specifies firmware upload, i.e. loading the currently installed device firmware to the USB Host. The FT900 boot loader supports DFU based flashing of programs and this application note describes the steps necessary to achieve it.

dfu-util.exe is a popular open source implementation of a host side implementation of the DFU 1.0 and DFU 1.1 specifications of the USB forum. Using dfu-util.exe users may download and upload firmware to/from devices connected over USB.

### 1.1 Overview

In order to use DFU programming, the user has to have a DFU programmer utility. dfu-util.exe is one such application and it is used in this application note. dfu-util.exe relies on libusb which is an open source C library that gives applications easy access to the USB devices in the system. Libusb in turn relies on WinUSB, which is a generic driver for USB devices that was developed concurrently with the Windows Driver Frameworks (WDF) for Windows XP with SP2. The WinUSB architecture consists of a kernel-mode driver (WinUSB.sys) and a user-mode dynamic link library (WinUSB.dll) that exposes WinUSB functions.

This note describes how to build dfu-util.exe and libusb.dll and install the WinUSB DLL and device drivers. On the device side, it also describes how to prepare a program for DFU download and customize the bootloader to match the user's USB device identification.

dfu-util.exe and dfu-util are used interchangeably in this document. C:\ is used as the example pathname in cmd.exe prompts. The software that is used in this application note is packaged in DFU\_util\_package.zip and is described in 2 Package Contents.

### 1.2 Scope

This application note covers DFU programming using dfu-util.exe built for a Windows system and does not cover Linux or MAC versions of dfu-util.exe

## 2 Package Contents

The package may be obtained from this link - [DFU\\_util\\_package.zip](#). It contains the items listed below:

	Item	Description
1	dfu-util.exe	Windows DFU binary which is used for listing DFU devices connected to the system and flashing binary to device when the device is in DFU-mode
2	DFU_application.bin	First demo application executable
3	FT900ProgGUI.jar*	Programming utility which is used to prepare firmware for DFU download
4	FT900 WinUSB Driver Package	WinUSB device-driver software for FT900 DFU device
5	MinGW-5.1.3.exe	Executable to install MinGW libraries required for dfu-util and libusb compilation
6	MSYS-1.0.10.exe	Executable to install MSYS. Required by dfu-util.exe and libusb-1.0.dll
7	libusb-1.0.dll	USB library file to support WinUSB driver
8	teraterm-4.86.exe	An open source terminal emulator software used for serial communications

**Table 1 Package Contents**

\*Refer to the Toolchain Installation executable for the programming utility file. This file is available as part of `.\FTDI\FT90x Toolchain\Toolchain\programmer\dist\`

### 2.1 Required Hardware

	Item	Description
1	MM900EVxA	FT900 MCU Evaluation Module, $x=1,2$ , or 3
2	UMFTPD2A	FT900 Programmer Board – uses One-wire programming to recover the MM900EVxA if the device corrupts and the DFU becomes unserviceable.
3	Micro USB Cables	For attaching MM900EVxA and UMFTPD2A to PC
4	USB-Serial Cable	For serial communication between MM900EVxA and PC such as <a href="#">TTL-232R-3V3</a> .

**Table 2 Hardware Required**

## 3 Building and installation of dfu-util

dfu-util.exe (Release 0.8) and libusb-1.0.dll (Release 1.0.19) are pre-built and provided in binary form in the package. The following sections show how to build them from source.

1. Extract the contents of the "DFU\_util\_package.zip" package.
2. Copy folders dfu-util-0.8 and libusb-1.0.19 to the desired location. For simplicity, this application note assumes the folders are copied to the C: directory, i.e. "C:\DFU". It is important that the two folders are at the same folder level, for example, C:\DFU\dfu-util-0.8 and C:\DFU\libusb-1.0.19.

*Note*

*Windows platform need MinGW and MSYS for the compilation of dfu-util and libusb.*

### 3.1 Installation of MinGW and MSYS

1. From the package run MinGW-5.1.3.exe. MinGW-5.1.3.exe is an installer which connects to the internet to download and install the MinGW toolchain.
2. From the package run MSYS-1.0.10.exe which guides the user through the installation procedure of MSYS
3. The above installations require an Internet connection.

### 3.2 Compiling libusb and dfu-util

1. Open a MinGW terminal:
  - a. Start -> All Programs -> MinGW -> MSYS -> msys
  - b. msys provides a Unix like shell environment suitable for dfu-util and libusb compilation.
2. Switch to the folder containing dfu-util-0.8 and libusb-1.0.19 and run the following commands In the supplied example, it is C:\DFU

#### 3.2.1 Compiling libusb

```
cd libusb-1.0.19
./configure --prefix=$HOME
# WINVER workaround needed for 1.0.19 only
# MKDIR_P setting should not really be needed...
make CFLAGS="-DWINVER=0x0501" MKDIR_P="mkdir -p"
make install
cd ..
```

#### 3.2.2 Compiling dfu-util

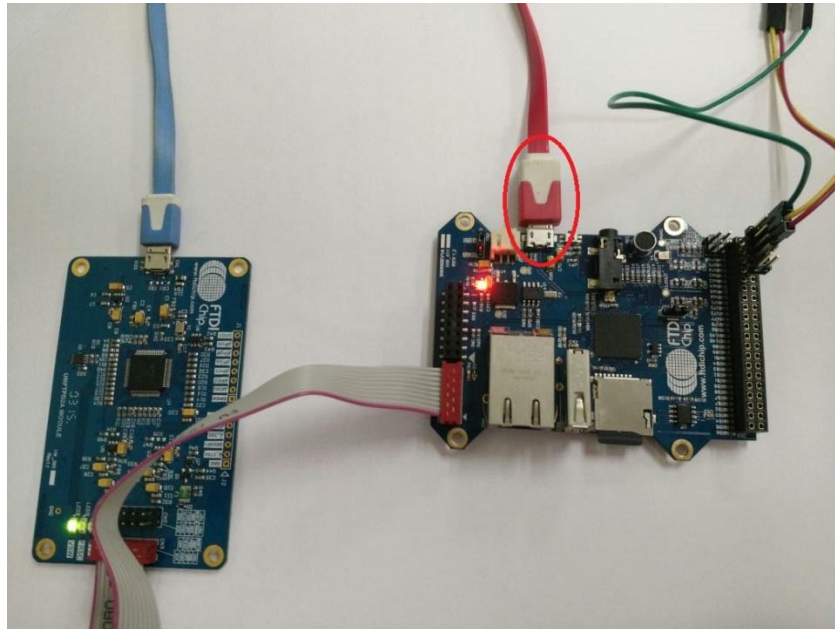
```
cd dfu-util-0.8
./configure USB_CFLAGS="-I$HOME/include/libusb-1.0" \
            USB_LIBS="-L $HOME/lib -lusb-1.0" PKG_CONFIG=true
make
```

```
make install  
cd ..  
make LDFLAGS=-static
```

The built executables (and DLL) will now be under /usr/local/bin.

## 4 FT900 Board Setup

1. Connect the MM900EVxA evaluation module and the UMFTPD2A programmer board to the PC via USB.



**Figure 1 FT900 Board Setup**

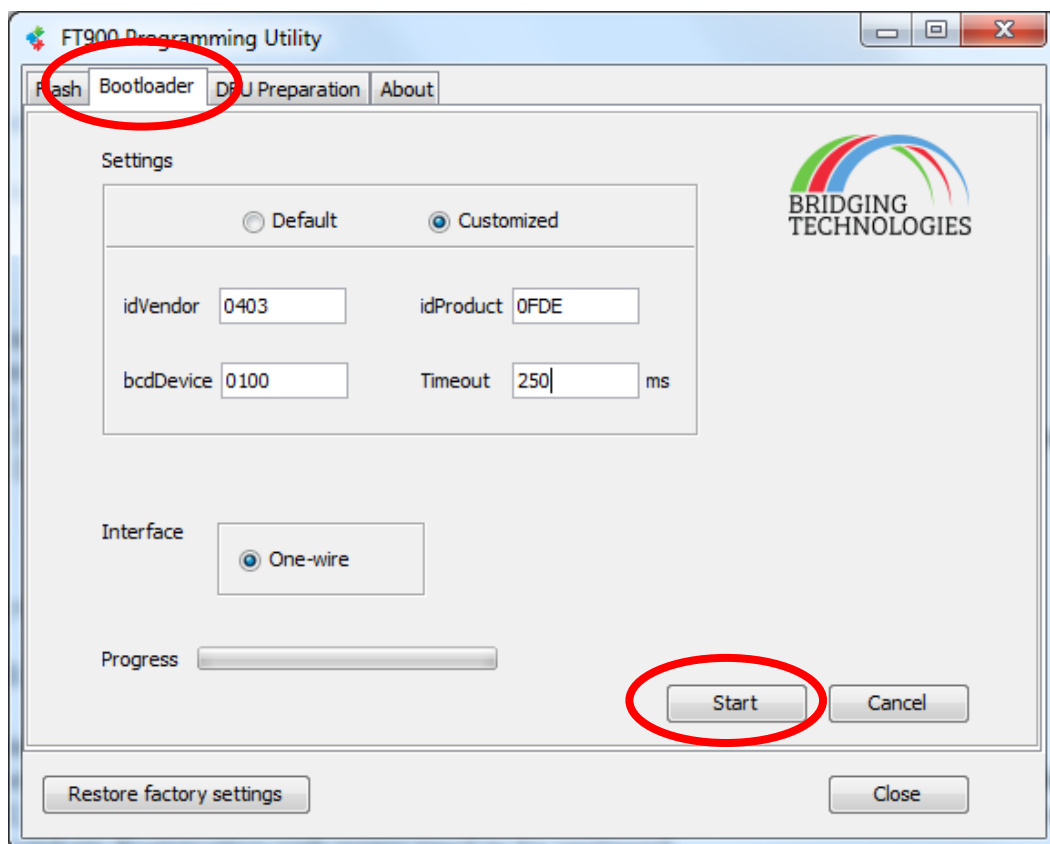
2. Connect the UMFTPD2A module to the MM900EVxA via a ribbon cable. Attach one end of the ribbon cable to CN3 (connector number 3) on UMFTPD2A to CN7 (connector number 7) on MM900EVxA. This module acts as the programmer for FT900.
3. Connect the FT900 UART port to the PC via a TTL UART cable. Minimum connections are CN3 pin 2 (GND), pin 4 (UART0\_TXD) and pin 6 (UART0\_RXD). The sample application (DFU\_application.bin) uses serial communication to display information on a terminal (such as Tera Term).



## 5 Flashing the Bootloader on to FT900

The bootloader is flashed using the FT900 Programmer. By default, the bootloader enumerates with idVendor=0x0403, idProduct=0x0FDE and bcdDevice is set to 2300. These values may be customized in the Bootloader tab of FT900ProgGUI.jar to match the end product requirement. See Section 7 Customizing the Bootloader for more details.

1. Run the FT900ProgGUI.jar
2. Switch to the Bootloader tab.
3. Update the idVendor, idProduct and bcdDevice fields. Leave the 'Timeout' field as 250ms.
4. Press the "Start" button to initiate flashing the bootloader
5. In this application note, the bootloader is flashed with the default values.
6. Once programming is complete, reset the MM900EVxA module.



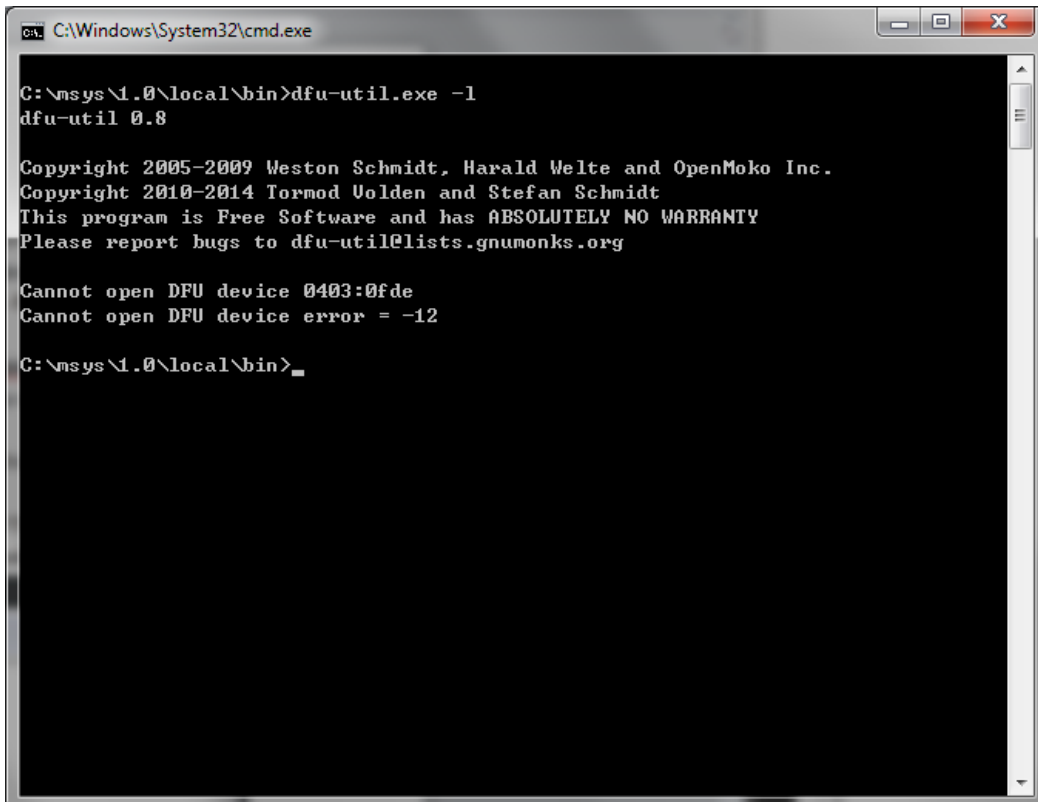
**Figure 2 Flashing bootloader to FT900**

## 6 dfu-util

After flashing the bootloader and resetting the MM900EVxA module, the bootloader detects that there is no valid application and enters DFU mode. In DFU mode, the bootloader initializes the USB device interface and enumerates itself as a USB DFU device. dfu-util may be used to confirm that the bootloader is in DFU mode.

1. Run the following command. The path C:\ is for example and may be different in the user's system.

```
C:\> dfu-util.exe -l
```



```
C:\Windows\System32\cmd.exe

C:\msys\1.0\local\bin>dfu-util.exe -l
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Cannot open DFU device 0403:0fde
Cannot open DFU device error = -12

C:\msys\1.0\local\bin>
```

**Figure 3 List DFU capable devices**

The default idVendor and idProduct reported by the bootloader during USB enumeration are 0x0403 and 0x0FDE respectively. The output listing from dfu-util indicates "Cannot open DFU device 0403:0fde" as shown in Figure 3. This confirms that the bootloader is indeed in DFU mode. Other DFU capable devices may also be listed by dfu-util and the user shall look for the correct Vendor and Device IDs.

The "error" message from dfu-util confirms that the bootloader is waiting in DFU mode. The error is reported because WinUSB is required by libusb-1.0.dll to "open" the USB device.

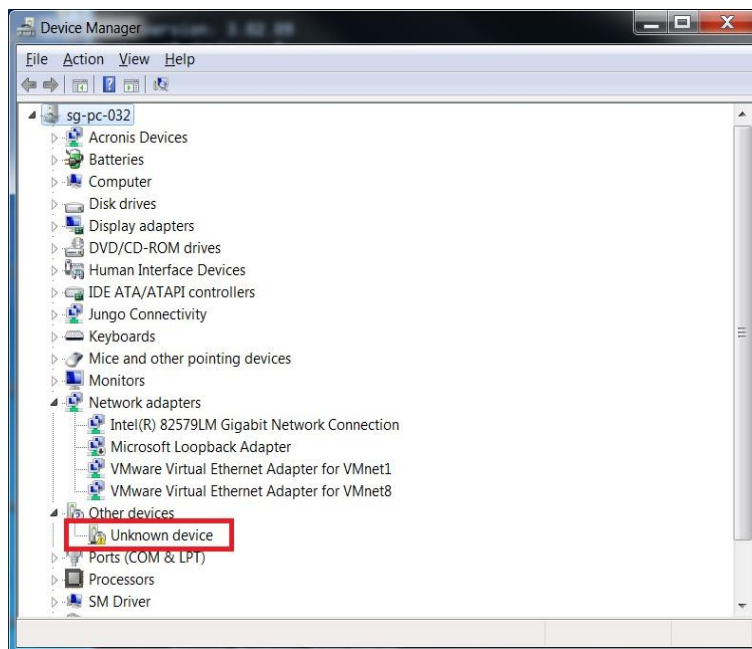
## 6.1 Windows Driver Signature Enforcement

On 64-bit Windows 7 and on Windows 8.x machines, driver signature check enforcement is enabled by default. Any unsigned driver (without a valid certificate) installations are rejected by these operating systems. Windows signature check enforcement has to be bypassed to install WinUSB. Following is a helpful link that shows how driver enforcement may be bypassed:

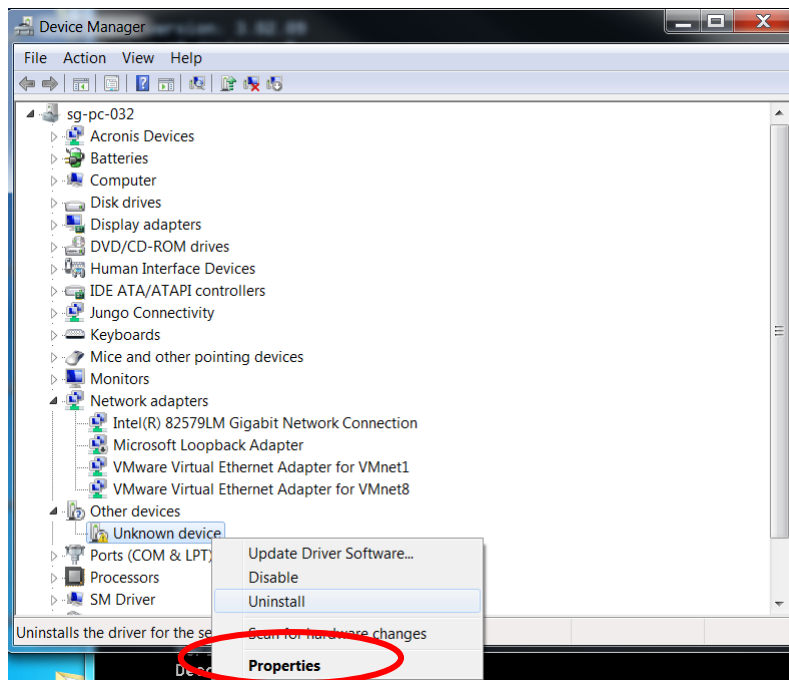
[http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN\\_129\\_Driver%20Release%20Signing.pdf](http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_129_Driver%20Release%20Signing.pdf)

## 6.2 WinUSB driver installation

Open Device Manager from the Control Panel. Look for "Unknown device". Right click on the "Unknown device" and select Properties. Figure 4 and Figure 5 illustrate these.

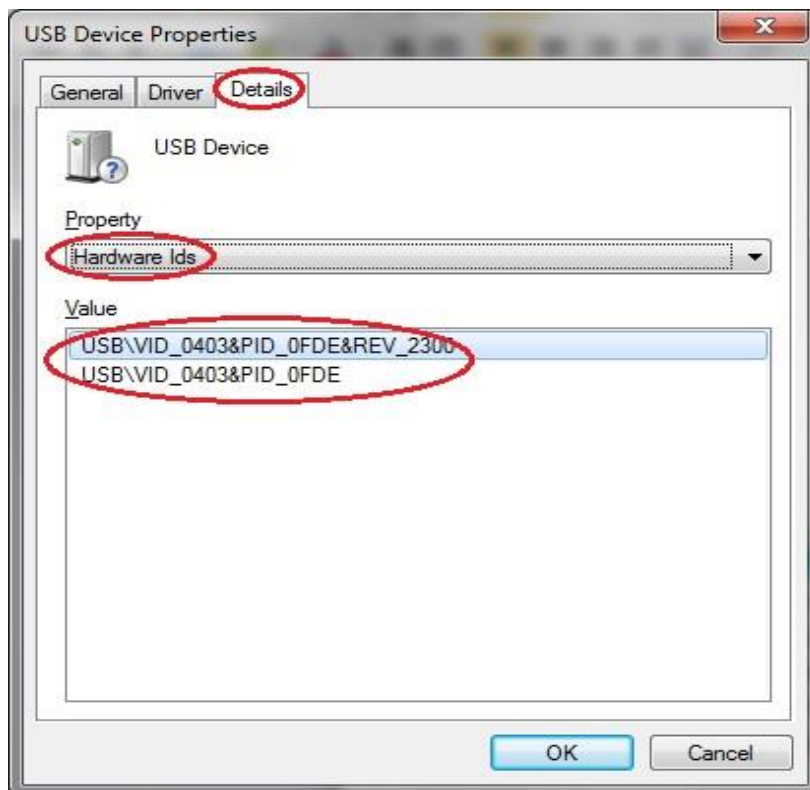


**Figure 4 Device Manager and Unknown device**



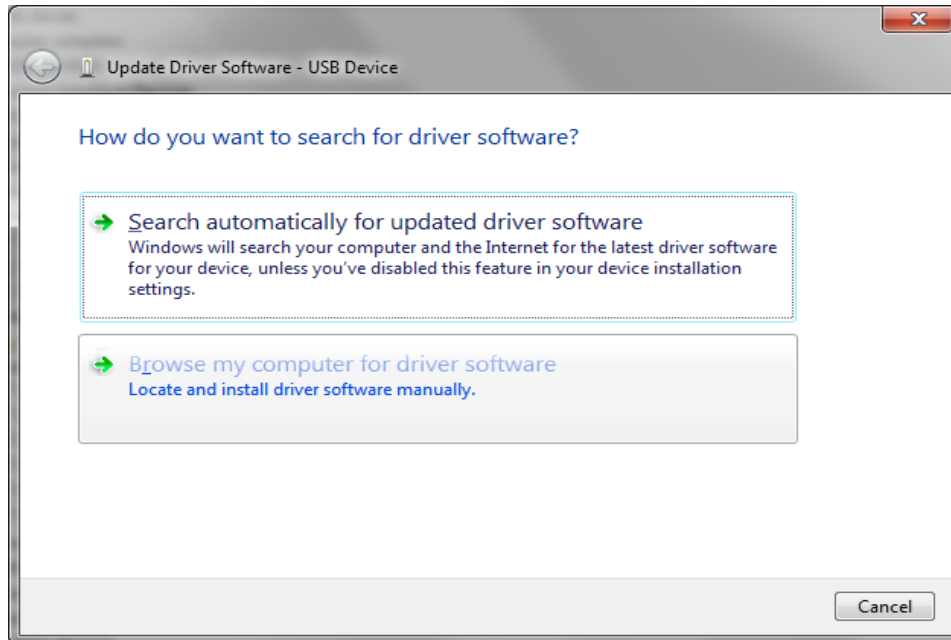
**Figure 5 Right-click and select Properties**

Figure 6 shows how to locate the FT900 USB Device. On the "Properties" page, select the "Details" tab and select "Hardware Ids" property. The "Value" display should show **USB\VID\_0403&PID\_0FDE&REV\_2300**. If the value does not match, then select another "Unknown device" to check the "Hardware Ids".



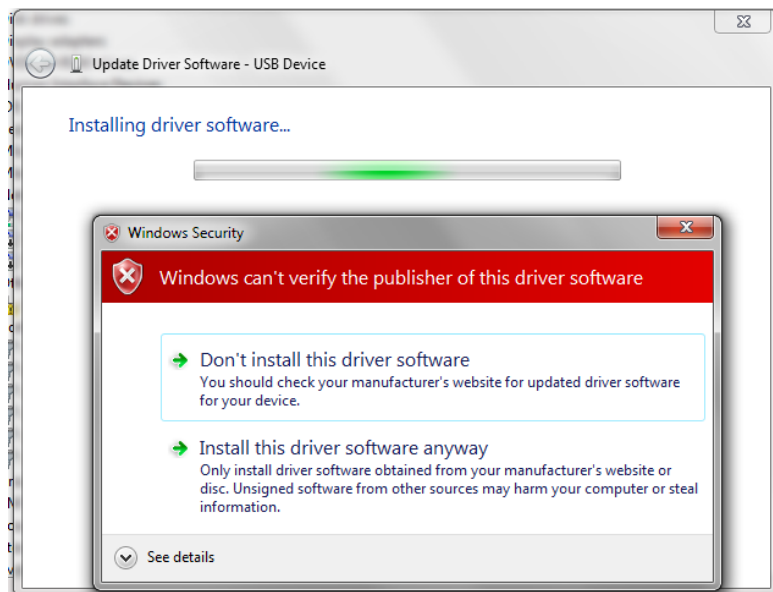
**Figure 6 Locating FT900 USB Device**

Once the "Unknown device" is confirmed to be an FT900 USB device, switch back to the device manager and right-click on the chosen "Unknown devices" and select "Update Driver Software". Refer to Figure 5. This will bring up the next screen. Select "Browse my computer for driver software".



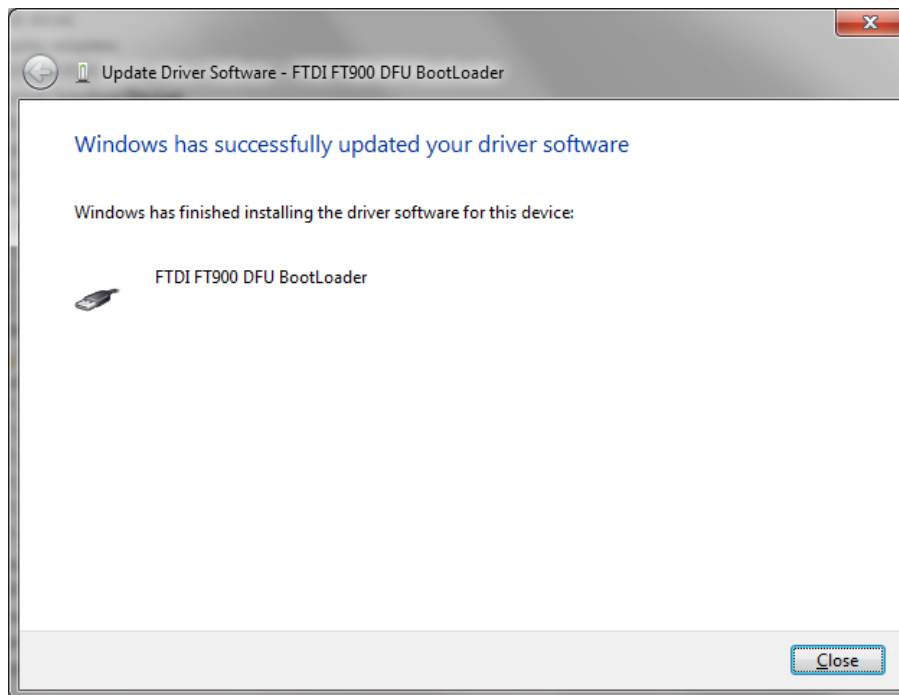
**Figure 7 – Installing WinUSB**

Within the DFU\_util\_package, browse for the WinUSB installation files. The relative path is at ".\FT900 WinUSB Driver Package\DriverInstallation\installer". After navigating to the correct folder, click on next to begin the installation. During installation, Windows Security will warn that it can't verify the publisher of the software. Choose "Install this driver software anyway" to continue the installation.



**Figure 8 – Windows Security warning**

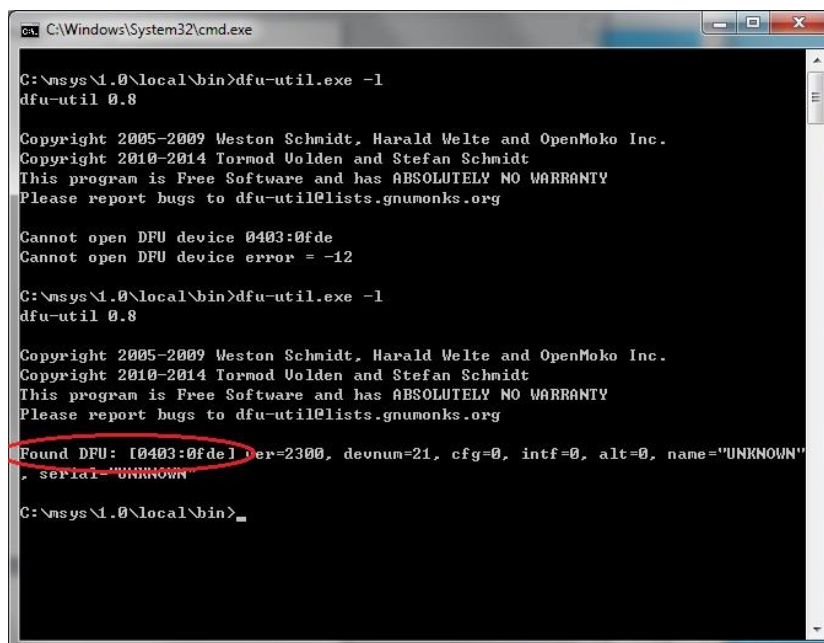
Once driver installation is complete, the following display is shown:



**Figure 9 – Successful Installation of WinUSB**

Switch back to the command prompt box used to run dfu-util.exe (Figure 3 List DFU capable devices) or open a new command prompt box (cmd.exe). Repeat the listing of DFU devices command to confirm that the FT900 USB DFU device is now accessible by dfu-util.

C:\> dfu-util.exe -l



**Figure 10 Confirm FT900 USB DFU device**

This time, the listing will display all the DFU capable devices that are found (red circle in Figure 10 Confirm FT900 USB DFU device).

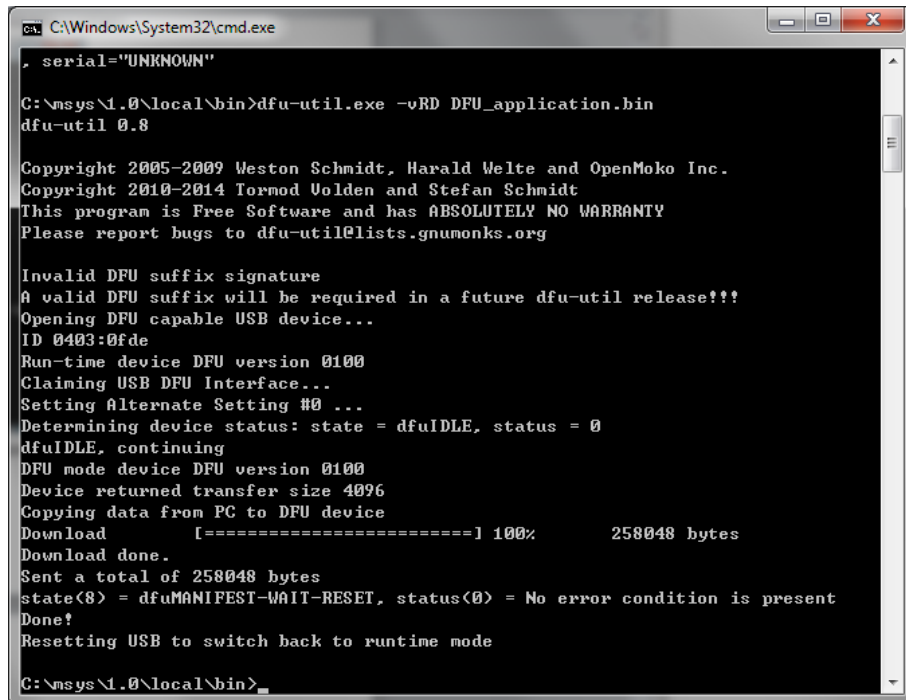
## 6.3 Downloading Firmware to the FT900

When a FT900 is first flashed with the bootloader executable, it does not contain any application program. The bootloader detects that no valid application is present and enters DFU mode waiting for firmware to be downloaded.

Firmware is downloaded using the following command:

```
C:/> dfu-util.exe -d 0403:0fde -vRD DFU_application.bin
```

Note that any other application .bin file can also be programmed, but refer to Section 6.5 Preparation for DFU download for additional requirements.



```

C:\Windows\System32\cmd.exe
. serial="UNKNOWN"

C:\msys\1.0\local\bin>dfu-util.exe -vRD DFU_application.bin
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Torodd Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Invalid DFU suffix signature
A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0403:0fde
Run-time device DFU version 0100
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0100
Device returned transfer size 4096
Copying data from PC to DFU device
Download      [=====] 100%      258048 bytes
Download done.
Sent a total of 258048 bytes
state(8) = dfuMANIFEST-WAIT-RESET, status(0) = No error condition is present
Done!
Resetting USB to switch back to runtime mode

C:\msys\1.0\local\bin>
  
```

**Figure 11 Downloading firmware on FT900**

Figure 11 Downloading firmware on FT900, shows the download command execution and upon download completion, dfu-util reports 100% download completion.

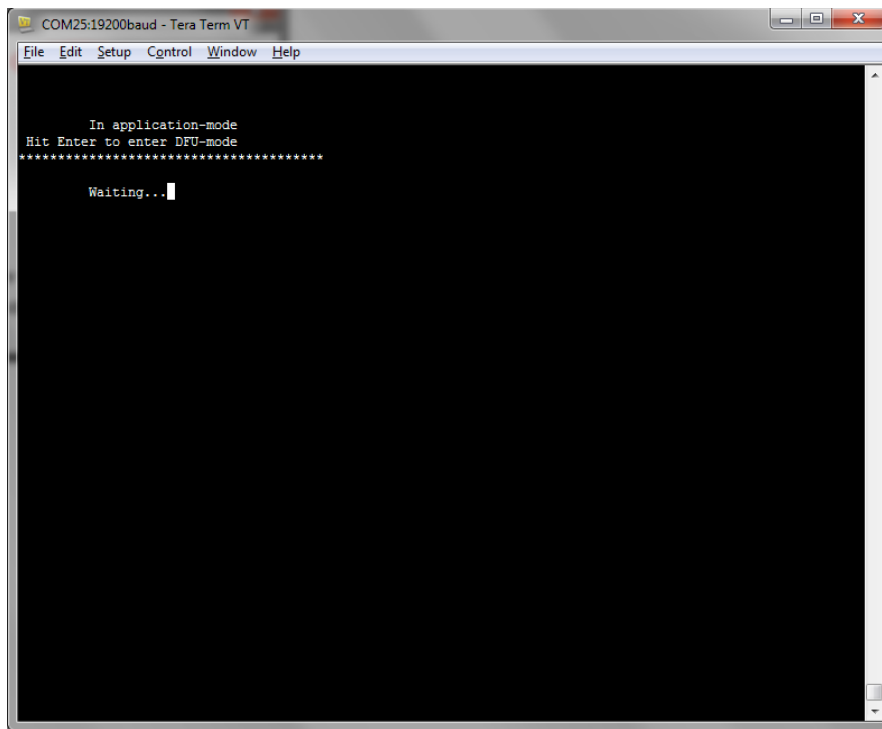
Though dfu-util reports “Resetting USB to switch back to runtime mode”, no USB reset is issued and FT900 remains in DFU mode and a manual reset has to be performed (press reset button or remove and insert the USB cable to the FT900). After reset, the bootloader executes the new firmware.

## 6.4 DFU\_application.bin

DFU\_application.bin is a simple demo application that shows how a user application may be upgraded via DFU. In order to enter DFU mode, an application is required to call into a function in the bootloader and DFU\_application.bin shows how this is done.

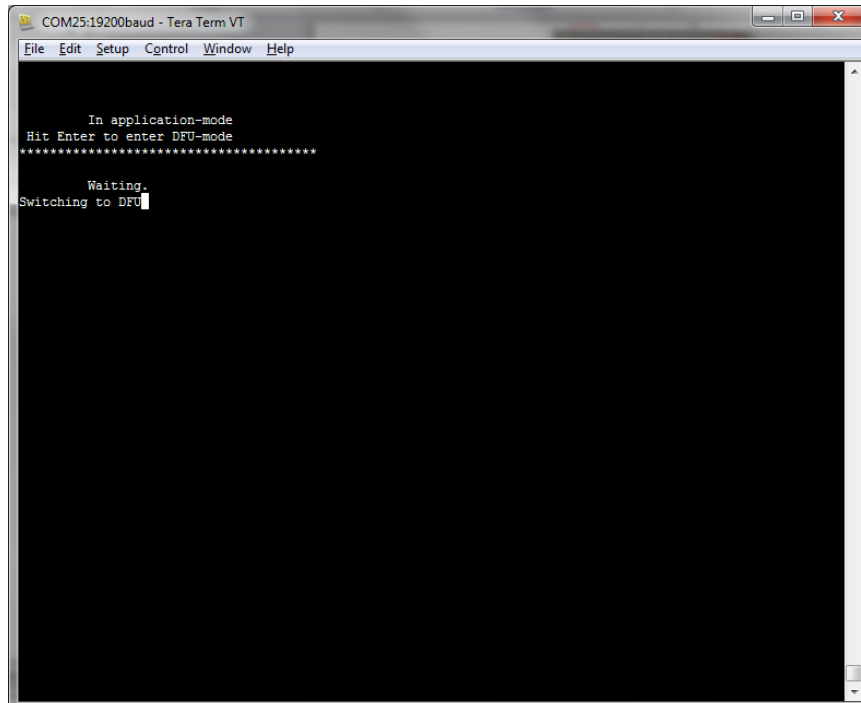
At application startup, DFU\_application.bin displays a message as shown in Figure 12 Startup message from DFU\_application.bin. It waits for the user to press the ENTER key. Once the ENTER key is pressed, the application jumps into DFU mode and stops responding to further keystrokes from the terminal. While the application executes, dfu-util is not able to find the FT900 DFU device because the DFU mode is inactive. However, after the ENTER key is pressed, the application enters DFU mode and dfu-util is able to find the FT900 DFU device again.

See Section 9 Tera Term Configuration for terminal setup.



**Figure 12 Startup message from DFU\_application.bin**

Once the ENTER key is pressed, the application displays "Switching to DFU" and enters DFU mode. This is shown in Figure 13 Switching from application mode to DFU mode.

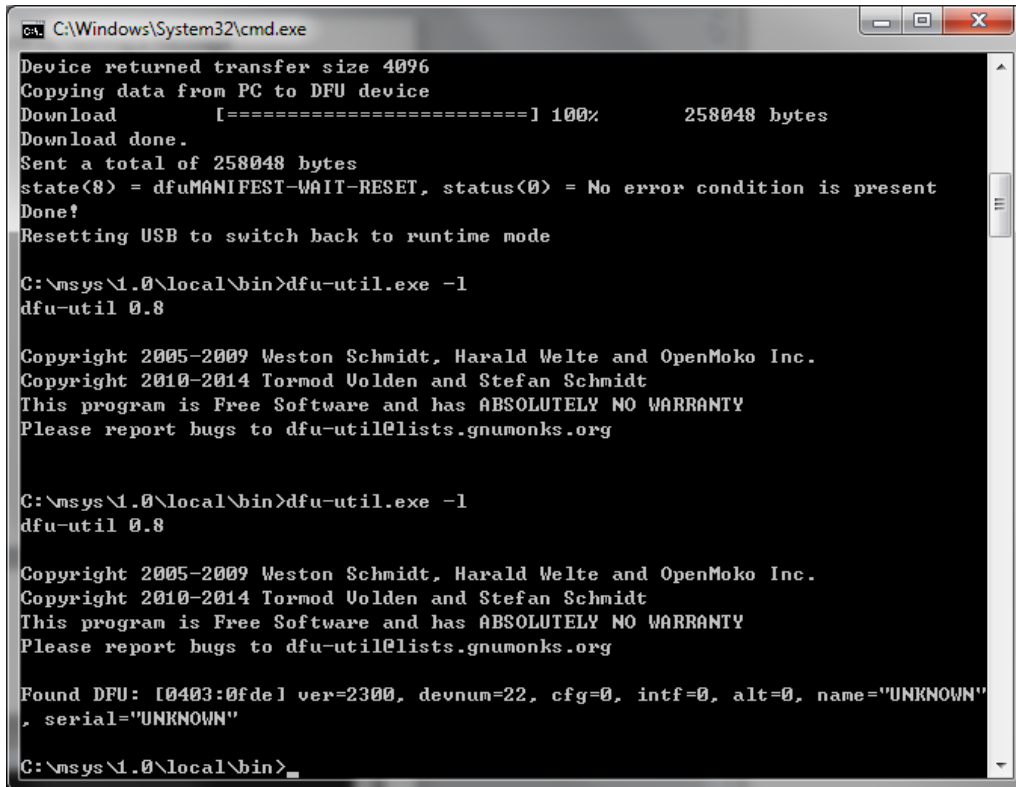


**Figure 13 Switching from application mode to DFU mode**

After switching to DFU mode, dfu-util is able to find and list the device. This is shown in Figure 14 Confirming DFU mode switch.



**Note:** The dfu-util listing command has to be issued a few times before the device is successfully reported as "Found". The delay is caused by USB Bus enumeration and/or re-enumeration requested by dfu-util.



```

C:\Windows\System32\cmd.exe
Device returned transfer size 4096
Copying data from PC to DFU device
Download [=====] 100%      258048 bytes
Download done.
Sent a total of 258048 bytes
state(8) = dfuMANIFEST-WAIT-RESET, status(0) = No error condition is present
Done!
Resetting USB to switch back to runtime mode

C:\msys\1.0\local\bin>dfu-util.exe -l
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

C:\msys\1.0\local\bin>dfu-util.exe -l
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Found DFU: [0403:0fde] ver=2300, devnum=22, cfg=0, intf=0, alt=0, name="UNKNOWN"
, serial="UNKNOWN"

C:\msys\1.0\local\bin>
  
```

**Figure 14 Confirming DFU mode switch**

Now that the device is once again in DFU-Mode, the firmware may be upgraded via DFU.

## 6.5 Preparation for DFU download

When an application is compiled and linked, the toolchain produces the executable and the corresponding binary file. This file is not ready for DFU download and needs to be prepared first. The DFU\_application.bin used in the previous section had already been prepared. The next sections describe the preparation step.

### 6.5.1 Padding and Checksum Addition

Before the firmware is upgraded via DFU, it needs to be appropriately prepared. The application build outputs an executable which is automatically converted, in the toolchain, to a binary file suitable for firmware download via the one-wire programmer.

However, for DFU download, the binary file has to be 252KB in size. The preparation step pads the binary file to 252KB in size. Additionally, the signature and checksum are inserted into the binary file. The padding and checksum are required to ensure that the bootloader is able to successfully verify that a valid firmware image exists in flash memory.

The padding process pads the binary file to a size of 252KB. The signature and checksum takes up 4 bytes and therefore, **the input binary file shall not exceed 252KB-4bytes, i.e. 258,044 bytes.**

### 6.5.2 DFU Suffix

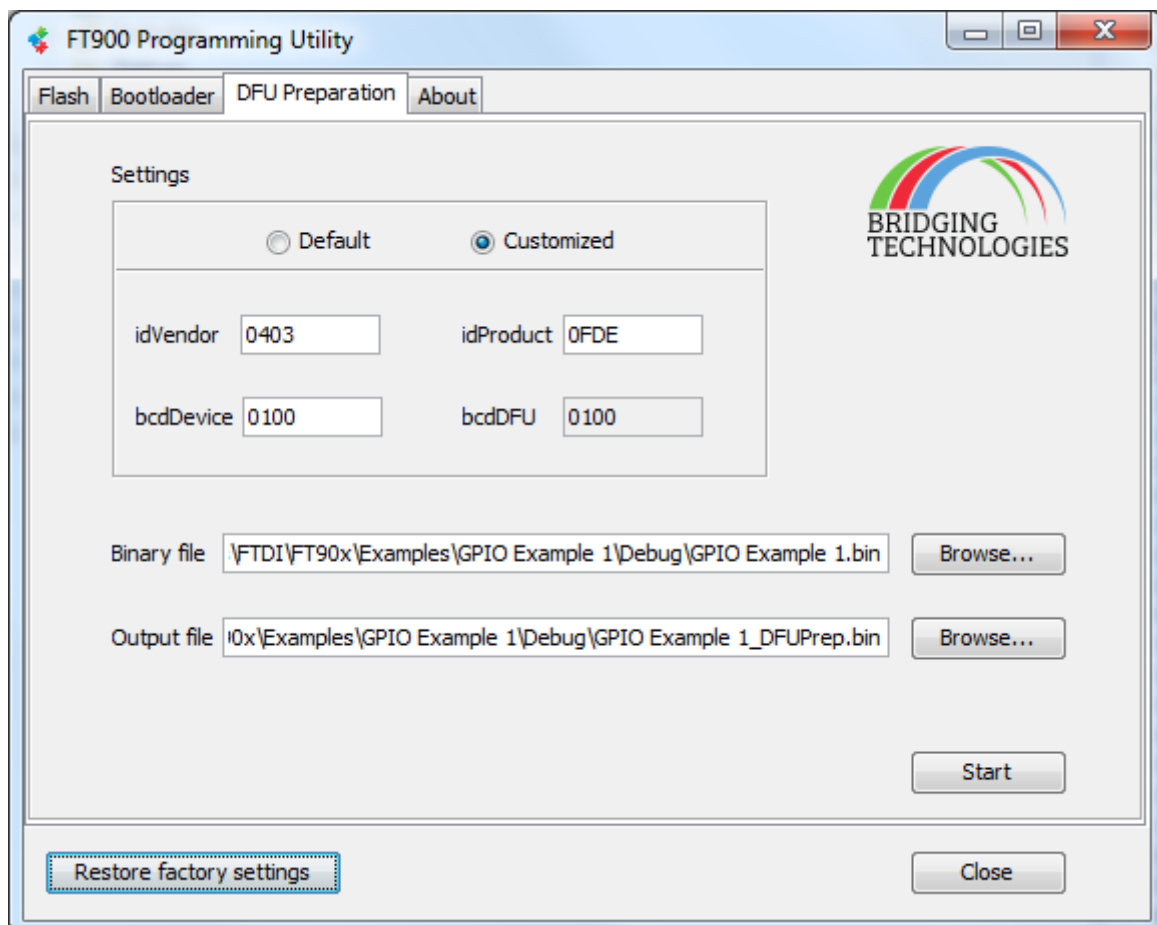
The USB DFU protocol requires that the binary file contain a DFU suffix to identify the file and its corresponding target USB device. The suffix is described in the [USB DFU specification](#) document.

The DFU Suffix contains parameters such as the idVendor, idProduct, bcdDevice and bcdDFU. When dfu-util downloads the firmware, it matches these parameters against the USB Device reported parameters. If there is a mismatch, then dfu-util may not proceed with the download. In the current version as of this writing, dfu-util is at version 0.8 and outputs a warning on a mismatch but proceeds with the download nevertheless. In a future version, the DFU suffix check may be enforced.

Therefore, by using the “DFU Preparation” tab, idVendor, idProduct and bcdDevice settings may be customized to match the values programmed in the bootloader (refer to Section “Downloading Firmware to the FT900”). bcdDfu shall always be set as 0100.

Alternatively, the idVendor, idProduct and bcdDevice settings may be left as “Default”. The default setting is a wildcard setting of idVendor=0xFFFF, idProduct=0xFFFF. These values inform dfu-util to ignore the DFU suffix check.

The next step shows the “DFU Preparation” tab and the “Customized” settings. Fill the input binary file in the “Binary file” entry and the required output file in the “Output file” entry. Click “Start” to begin the preparation and at the end, the “Output file” is produced. The output file may then be used for DFU download.



**Figure 15 DFU Preparation**

## 7 Customizing the Bootloader

The default bootloader that ships with MM900EVxA is programmed to enumerate as a USB DFU device with idVendor=0x0403 and idProduct=0x0FDE. These values may be modified to match the customer's requirements and may be done via the FT900 Programming Utility (FT900ProgGUI.jar).

1. Launch the FT900ProgGUI.jar java executable. Refer to section 5 "Flashing the Bootloader on to FT900".
2. Select the Bootloader tab
3. Enter the required values in the idVendor, idProduct and bcdDevice. Leave the timeout as 250ms.
4. Press the "Start" button to program the new bootloader into the FT900 device. This programming takes place via the One-wire programmer

### **Note**

When the idVendor and idProduct are customized, the bootloader DFU will enumerate with these new values. To support the customized DFU, a corresponding INF file is required so that WinUSB driver is installed for this "new" DFU capable device. The WinUSB installation INF file may be edited with the customized values, i.e. replace the idVendor and idProduct values in the INF file and repeat the steps in 6.2 WinUSB driver installation to install the new DFU device.

## 8 Source Code

### 8.1 DFU\_Application.bin (DFU\_application.c)

Sample source code takes a user input and enters into DFU mode. Look for the GO\_DFU() macro in the uartISR() function below:

```

/*
 * DFU_application.c
 *
 * Created on: Apr 30, 2015
 *
 */

/*
***** INCLUDES *****
#include<uart_command.h>
#include<DFU.h>

/** CONSTANTS ..... */
#define UART0_TXD (48)
#define UART0_RXD (49)

/* FUNCTION DEFINITIONS ..... */
void uart_init(void);
void myputc(void* p, char c);
void displayBanner(void);

/** CODE ..... */

/** Program Entry */
main()
{
    uart_init();
    displayBanner();
    tfp_printf("\033[8;10H");
    tfp_printf("Waiting");
    while(1)
    {
        tfp_printf("\033[8;17H");
        tfp_printf(" ");
        tfp_printf("\033[8;17H");
        tfp_printf(".");
        delayms(500);
        tfp_printf(".");
        delayms(500);
        tfp_printf(".");
        delayms(500);
    }
}

/** displays banner to screen through uart*/
static void displayBanner()
{
    tfp_printf("\033[2J");
    tfp_printf("\033[4;10H");
    tfp_printf("This is NEW FIRMWARE\r\n");
    tfp_printf(" Hit Enter to enter DFU-mode");
    tfp_printf((unsigned char *)"\n\r*****\n\r");
}

```

```
}

/**uart interrupt handler*/
void uartISR()
{
    // Check type & clear interrupt
    if (uart_is_interrupted(UART0, uart_interrupt_rx)
    {
        tfp_printf("\r\nSwitching to DFU");
        GO_DFU(0,0,0); // ← this is the step to switch in the bootloader DFU
    }
}

/** uart initialization*/
void uart_init()
{
    // Initialize local variables
    uint16_t divisor = 0;
    uint8_t prescaler = 0;

    // Initialize UART0
    sys_enable(sys_device_uart0);
    gpio_function(UART0_TXD, pad_uart0_txd); // UART0_TXD/GPIO48
    gpio_function(UART0_RXD, pad_uart0_rxd); // UART0_RXD/GPIO49
    uart_open(UART0, /* Device */
        1, /* Prescaler = 1 */
        UART_DIVIDER_115200_BAUD, /* Divider = 1302 */
        uart_data_bits_8, /* No. Data Bits */
        uart_parity_none, /* Parity */
        uart_stop_bits_1); /* No. Stop Bits */

    // Enable received interrupt for UART0
    uart_enable_interrupt(UART0, uart_interrupt_rx);
    uart_enable_interrupt(UART0, uart_interrupt_tx);
    uart_enable_interrupts_globally(UART0);
    interrupt_enable_globally();
    init_printf(UART0,myputc);

    // Attach ISR function
    interrupt_attach(interrupt_uart0, (uint8_t) interrupt_uart0, uartISR);
}

void myputc(void* p, char c)
{
    uart_write((ft900_uart_regs_t*)p, (uint8_t)c);
}

/* end */
```

## 8.2 DFU.h

```
/*
 * DFU.h
 *
 * Created on: Apr 30, 2015
 *
 */

#ifndef __FT900_DFU_H__
#define __FT900_DFU_H__

#ifdef __cplusplus
```

```
extern "C" {
#endif /* __cplusplus */

/* TYPES
*****/
typedef void (*__dfumain_t)(unsigned int, unsigned int) __attribute__((noreturn));

/* MACROS
*****/
#define _GO_DFU(ID,BCD) ((__dfumain_t)0x3fff4)((ID),(BCD))
#define _concat(a,b) ((a) << 16 | (0xFFFFUL & (b)))

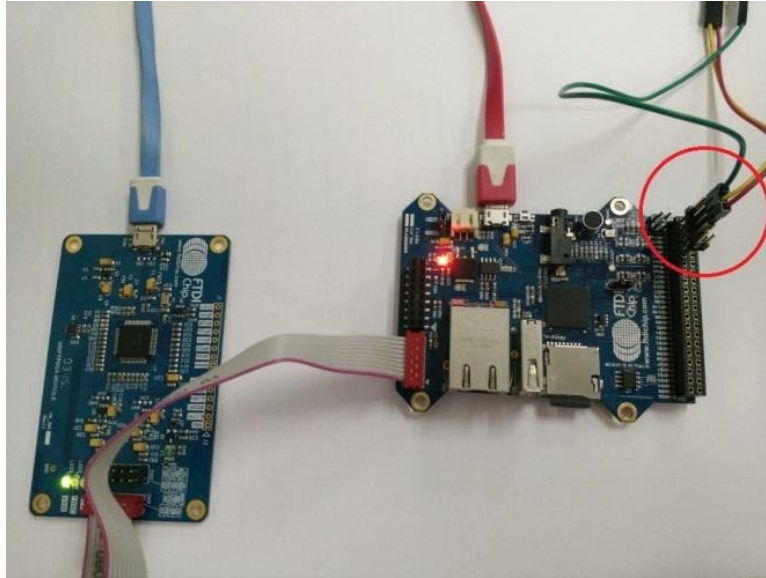
/** @brief macro to Jump into DFU mode
 * @param v Vendor ID
 * @param p Product ID
 * @param b BCD device
 */
#define GO_DFU(v,p,b) do {\
        _GO_DFU((unsigned int) _concat((v),(p)), (unsigned\
int)_concat(0,(b))); \
    } while(0)

#ifdef __cplusplus
} /* extern "C" */
#endif /* __cplusplus */
#endif //__FT900_DFU_H__

/* end */
```

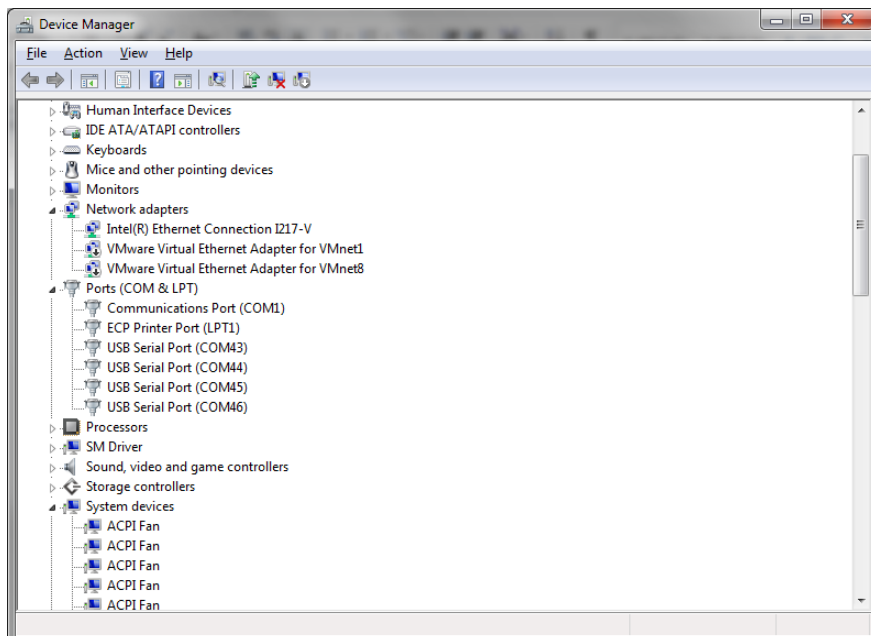
## 9 Tera Term Configuration

Tera Term is used to view UART data sent from the DFU\_Application, or any other application which uses UART0 to send and/or receive data.

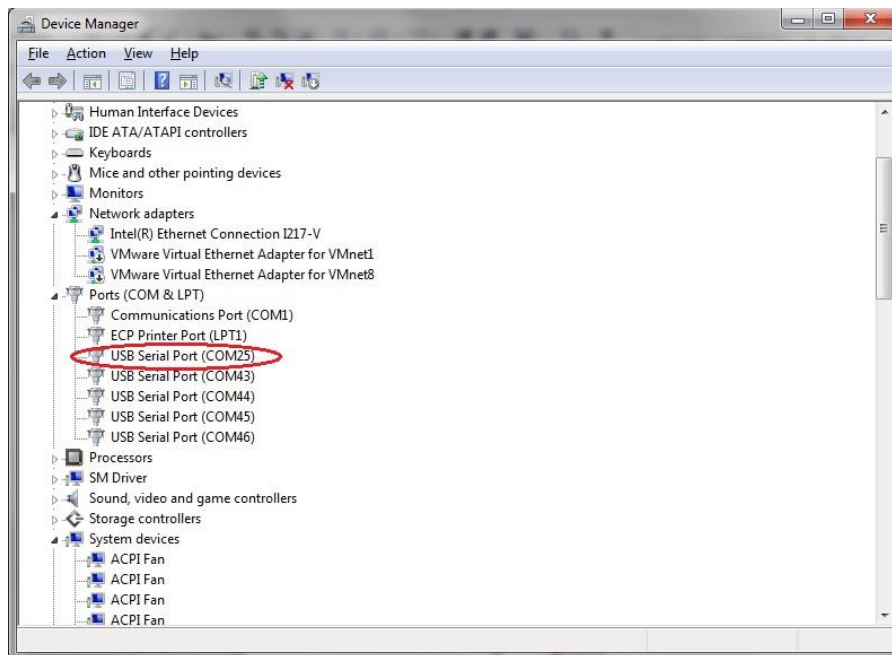


**Figure 16 FT900 UART Connections**

1. Install Tera Term from the package
2. Attach the USB-Serial cable to the FT900 UART pins (GND/TXD/RXD). **Do not yet attach the USB-Serial cable to the PC.**
3. Open Device Manager and then insert the USB-UART cable into a USB port on the PC.
4. Under "Ports (COM & LPT)", verify the **COM port** assigned to the USB-Serial cable.



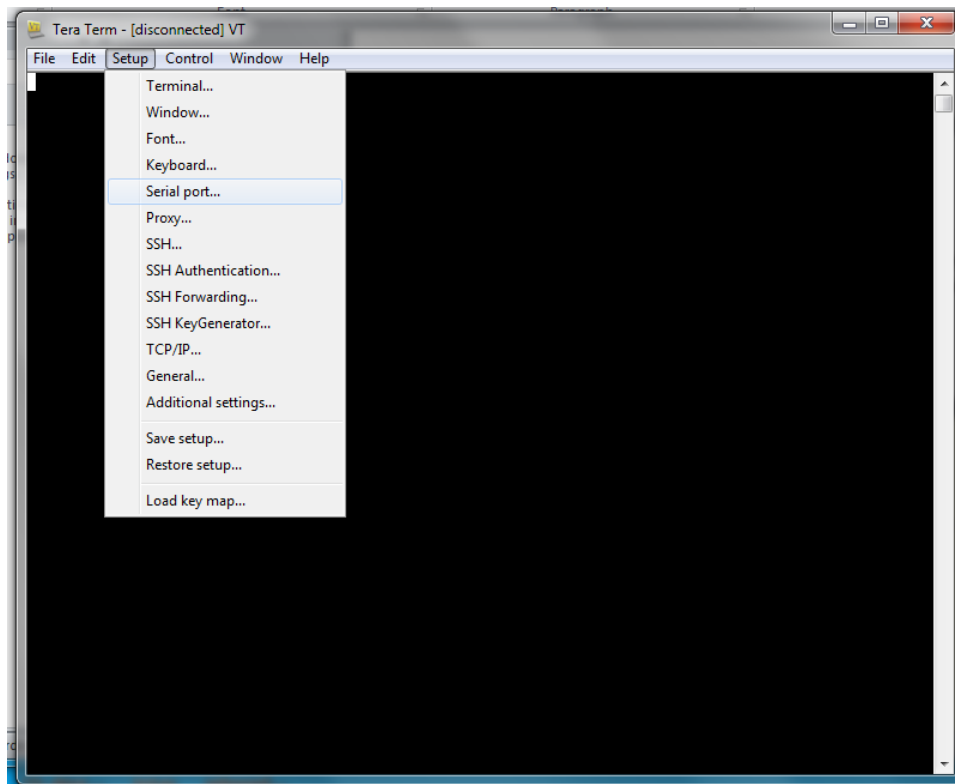
**Figure 17 Device Manager – Ports (COM & LPT)**



**Figure 18 USB Serial Port (COM25)**

In this case, after the USB-Serial cable was inserted, a new "USB Serial Port (COM25)" entry appeared.

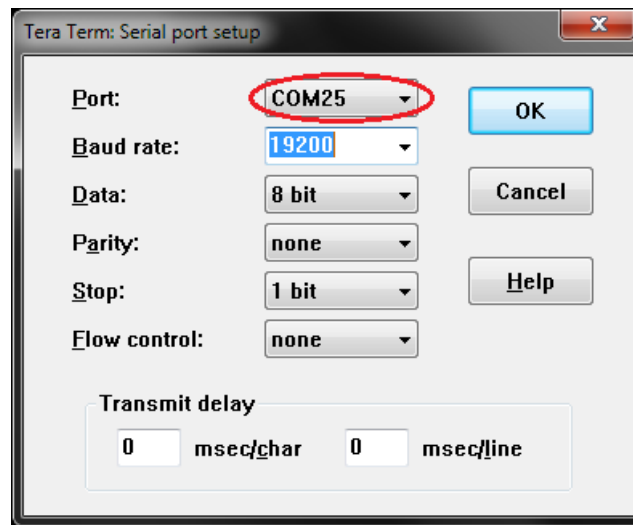
Launch Tera Term.exe and select "Setup" from the menu and choose "Serial port...".



**Figure 19 Configure Serial Port in Tera Term**



Select the port assigned to the USB Serial cable and set the baud-rate to 19200 for communication with DFU\_application.bin and press OK.



**Figure 20 Serial port setup**

Tera Term is configured for communication with DFU\_application.bin so the text in Section 6.4 DFU\_application.bin can be seen.

## 10 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](mailto:sales1@ftdichip.com)  
E-mail (Support) [support1@ftdichip.com](mailto:support1@ftdichip.com)  
E-mail (General Enquiries) [admin1@ftdichip.com](mailto: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](mailto:us.sales@ftdichip.com)  
E-Mail (Support) [us.support@ftdichip.com](mailto:us.support@ftdichip.com)  
E-Mail (General Enquiries) [us.admin@ftdichip.com](mailto: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](mailto:tw.sales1@ftdichip.com)  
E-mail (Support) [tw.support1@ftdichip.com](mailto:tw.support1@ftdichip.com)  
E-mail (General Enquiries) [tw.admin1@ftdichip.com](mailto: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](mailto:cn.sales@ftdichip.com)  
E-mail (Support) [cn.support@ftdichip.com](mailto:cn.support@ftdichip.com)  
E-mail (General Enquiries) [cn.admin@ftdichip.com](mailto: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.

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

### Document References

1. [Universal Serial Bus, Device Class Specification for Device Firmware Upgrade, Version 1.1, Aug 5, 2004](#)
2. <http://dfu-util.sourceforge.net/build.html>
3. [http://www.mingw.org/wiki/howto\\_install\\_the\\_mingw\\_gcc\\_compiler\\_suite](http://www.mingw.org/wiki/howto_install_the_mingw_gcc_compiler_suite)
4. <http://www.ftdichip.com/Products/ICs/FT90x.html>
5. <http://www.ftdichip.com/Support/FTDocuments.htm>

### Acronyms and Abbreviations

Terms	Description
DFU	Device Firmware Update
MinGW	Minimalist GNU for Windows, formerly mingw32
MSYS	MSYS is a collection of GNU utilities such as bash, make, gawk and grep to allow building of applications and programs which depend on traditionally UNIX tools
USB	Universal Serial Bus
USB-IF	USB Implementers Forum

## Appendix B – List of Tables & Figures

### List of Figures

Figure 1 FT900 Board Setup .....	7
Figure 2 Flashing bootloader to FT900 .....	8
Figure 3 List DFU capable devices .....	9
Figure 4 Device Manager and Unknown device.....	10
Figure 5 Right-click and select Properties .....	11
Figure 6 Locating FT900 USB Device.....	11
Figure 7 – Installing WinUSB .....	12
Figure 8 – Windows Security warning .....	12
Figure 9 – Successful Installation of WinUSB .....	13
Figure 10 Confirm FT900 USB DFU device .....	13
Figure 11 Downloading firmware on FT900.....	14
Figure 12 Startup message from DFU_application.bin .....	15
Figure 13 Switching from application mode to DFU mode .....	15
Figure 14 Confirming DFU mode switch.....	16
Figure 16 DFU Preparation .....	17
Figure 17 FT900 UART Connections .....	22
Figure 18 Device Manager – Ports (COM & LPT) .....	22
Figure 19 USB Serial Port (COM25) .....	23
Figure 20 Configure Serial Port in Tera Term .....	23
Figure 21 Serial port setup.....	24

### List of Tables

Table 1 Package Contents .....	4
Table 2 Hardware Required .....	4

## Appendix C – Revision History

Document Title: AN\_380 FT900 Bootloader DFU Usage  
Document Reference No.: FT\_001197  
Clearance No.: FTDI# 472  
Product Page: <http://www.ftdichip.com/FTPProducts.htm>  
Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	Initial Release	2015-10-13