

**OS X D2XX Library Version 1.1.12**  
**© Future Technology Devices International Ltd. 2011**

## **DISCLAIMER**

This software is supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Future Technology Devices International Ltd. will not accept any claim for damages howsoever arising as a result of use or failure of this software. Your statutory rights are not affected. This software or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice.

## **Packages**

This package contains 2 different builds of the D2XX driver, located in the following subdirectories:

10.4: PPC and Intel x86 universal binary compatible with OS X 10.4.x (Tiger).

10.5-10.7: PPC, Intel x86 and Intel x86\_64 universal binary compatible with 10.5.x (Leopard), 10.6.x (Snow Leopard) and 10.7.x (Lion).

## **Installation**

Installing the library is a relatively simple operation which involves copying a file and making a symbolic link.

Use the following steps to install (these assume you have copied all of the distribution files to the desktop):

1. Open a Terminal window (Finder->Go->Utilities->Terminal).
2. If the /usr/local/lib directory does not exist, create it (sudo mkdir /usr/local/lib)
3. if the /usr/local/include directory does not exist, create it (sudo mkdir /usr/local/include)
4. Copy the dylib file to /usr/local/lib (sudo cp Desktop/D2XX/bin/libftd2xx.1.1.12.dylib /usr/local/lib/libftd2xx.1.1.12.dylib)
5. Make a symbolic link (sudo ln -sf /usr/local/lib/libftd2xx.1.1.12.dylib /usr/local/lib/libftd2xx.dylib)
6. Copy the D2XX include file (sudo cp Desktop/D2XX/Samples/ftd2xx.h /usr/local/include/ftd2xx.h)
7. Copy the WinTypes include file (sudo cp Desktop/D2XX/Samples/WinTypes.h /usr/local/include/WinTypes.h)
8. You have now successfully installed the D2XX library.

## **Samples**

The samples provided are simple C written command line based applications that must be executed from the Terminal window.

To compile and run the samples perform the following steps (these assume you have copied all of the distribution files to the desktop and installed the library as per

the Installation section above):

1. Open a Terminal window (Finder->Go->Utilities->Terminal).
2. Change to the root samples directory - `cd Desktop/D2XX/Samples`
3. Build the samples by typing "make" then return. If you have issues at this stage revisit the installation section to ensure the library is correctly installed. Read the error messages and try to determine the source of the problem. If you still have issues then contact support detailing your issue with as much information as possible.
4. To run an application have a suitable FTDI device with default VID and PID and change to the Simple directory (`cd Simple`) then type `./simple` then return (make sure the dot and the forward slash precede the simple command).
5. If you have issues at this stage then consult the troubleshooting section later in this document. If the troubleshooting section doesn't help then contact support with your problem details.

### **Upgrading Issues**

Upgrading the D2XX library can cause problems such as a reported bug fix does not appear to be fixed.

This is most likely related to the application executable pointing to a previous version of the library.

To determine which D2XX library your application is using perform the following steps (Examples in brackets assume you have copied all of the files to the desktop and successfully compiled the samples as described above in the Samples section):

1. Open a Terminal window (Finder->Go->Utilities->Terminal).
2. Change directory to the application executable folder (`cd Desktop/D2XX/Samples/Simple`)
3. Use `otool` to determine the library path (`otool -L simple`)
4. The following test is an example of what is displayed  
simple:  
    /usr/local/lib/libftd2xx.1.1.12.dylib (compatibility version 0.1.0, current version 1.1.12)  
    /usr/lib/libSystem.B.dylib (compatibility version 1.1.12, current version 88.1.6)
5. As illustrated the simple application is pointing to the `libftd2xx.1.1.12.dylib`.
6. To alter the library that the simple sample points to use the `install_name_tool` (e.g `install_name_tool -change /usr/local/lib/libftd2xx.1.1.12.dylib /usr/local/lib/libftd2xx.dylib simple`). Please note you may need to change user mode to perform this function depending on the file permissions set on the executable.
7. Run the `otool` (illustrated in step 3) to confirm that the library pointed to by the application has changed and is correct.

### **Multiple VIDs/PIDs**

The current library has a default VID/PID table embedded within to determine if a particular device will be opened / accessed by the library. This table contains FTDI's own VID and PIDs therefore if you use FTDI default VID and PIDs this will not concern your application. There may be a situation when this is not suitable for a

particular application such as custom VIDs and or PIDs. The FT\_SetVIDPID API call can cater for single instances of VID/PID variations however there is a method to include your own range of VID/PIDs with a custom table.

1. Using xcode open the LibTable.xcodeproj in the LibTable folder.
2. Edit the ftdi\_table.c file to include your own VIDs/PIDs
3. Recompile the library.
4. Copy the resultant binary to the /usr/local/lib directory.

If you have trouble with the above procedure (don't know what xcode is / can't compile the library) then contact support with your VID/PID requests and we will provide you with a binary to use. Current workload will ultimately impact on the time we can provide you with a library therefore it is advised to provide us with the details at your earliest possible convenience.

### **Configuration Settings (Advanced users only)**

Configuration settings are considered advanced features and are only needed in certain rare situations. Only alter these settings if it has been suggested by support or you know exactly what you are doing.

The D2XX configuration file MUST be called ftd2xx.cfg. This is a simple text file containing various settings and allows for expansion in the future. The file is read on an FT\_Open/Ex only. 3 sections are available for settings – global, VID/PID and unique. An example of the ftd2xx.cfg file is included in the package and each section should be self explanatory. The [Globals] section applies to all devices, the [VID\_0403&PID\_6001] applies to devices of only VID and PID 0x0403 and 0x6001 and the unique settings apply to only those devices of a particular serial number – in this case [FT000001].

The configuration file must reside in /usr/local/lib or the /usr/lib folder.

### **Multithreaded write**

It has been noted during beta testing that some multi threaded applications can lock up during write communication with the devices. This is due to a conflict in the application run loop and the write run loop. To prevent lockup with a multithreaded application use the 31<sup>st</sup> (ConfigFlags=0x40000000) bit in the configuration file. This will enable an alternative FT\_Write that has its own thread and will not conflict with the current thread run loop operation.

### **Troubleshooting**

Q. Cannot open a port even though installation has been successful.

A1. This is possibly due to the FTDI serial driver holding the port with your VID and PID. Solution is to uninstall the serial driver (see [www.ftdichip.com](http://www.ftdichip.com) knowledgebase on how to do this). To completely eradicate the possibility of this occurring in future it is recommended a new VID and PID is used to distinguish between devices.

A2. Another possibility is an incorrect VID/PID. Try changing your application to use the FT\_SetVIDPID API call to quickly determine if this is the case.

Q. After running an application two or three times – communication will stop.

A. As described above you should be closing the device on application shutdown.

Trapping application exit is demonstrated in some of the sample programs and for simple C applications this is one method to circumvent this issue. If you cannot find a work around then try setting the USB Reset after open bit in the ftd2xx.cfg file (but only as a very last resort!).