

D2XX for Linux 1.4.33

By Future Technology Devices International

Mar 18 2025

Introduction

D2XX for Linux

As Linux distributions vary these instructions are a guide to installation and use. FTDI has tested the driver with LTS Ubuntu distributions for x86 and x86_64 architectures, and Raspbian on Raspberry Pi.

FTDI developed libftd2xx primarily to aid porting Windows applications written with D2XX to Linux. We intend the APIs to behave the same on Windows and Linux so if you notice any differences, please contact us (see <http://www.ftdichip.com/FTSupport.htm>).

FTDI do not release the source code for libftd2xx. If you prefer to work with source code and are starting a project from scratch, consider using the open-source libFTDI.

libftd2xx uses an unmodified version of libusb (<http://libusb.info>) which is distributed under the terms of the GNU Lesser General Public License (see [libusb/COPYING](#) or <http://www.gnu.org/licenses>). Source code for libusb is included in this distribution.

libftd2xx Release Notes ## History * 1.4.33 * Removed ZLP code which is specific to FT4222.

- 1.4.32
 - Update libusb 1.0.22 to 1.0.27.
 - Fixed issues in the jenkins-mac.sh script.
 - Updated the def.mak file to support building on different macOS versions.
 - Fixed Unlock of Unlocked Mutex issue in EventDestroy.
 - Fixed the assertion issue in pthread_mutex_lock, pthread_mutex_unlock, and pthread_setspecific APIs.
- 1.4.31
 - Corrected the timing calculation in the EventWait function.

- Added ArmV5-uclibC-sf support.
 - Corrected the event masking logic.
- 1.4.30
 - Removed “Basic sanity tests for both static and dynamic libraries” from the jankins-mac.sh file.
 - Added version information to the library.
 - Example Makefile correction.
 - Added sample test case for FT-X devices.
 - Added MIPS 64bit support.
 - Updated the Makefiles with updated toolchain path.
 - Updated the toolchain path for the ARM compilation.
 - Added Power delivery sample test cases.
 - Enabled MIPS64bit soft-float build.
 - Change the Rules.make file to access libraries from the current directory.
 - Added install_name_tool into make files.
 - Added changes to create libftd2xx.dylib along with libftd2xx..dylib.
 - Added Arm-uclibC build support.
- 1.4.29
 - fixed FT_CreateDeviceInfoList takes 15sec on Mac OS. (Bug ID:407)
- 1.4.28
 - Removed d3xx code (Now its build only d2xx code)
 - Enabled Mips support
 - Code cleanup
- 1.4.27
 - Add x86_32 support
- 1.4.26
 - Add arm64e support for mac
- 1.4.25
 - ARMv6l hard-float Support included, using ArchLinux ARM Cross compiler.
- 1.4.24
 - Support FT232HP, FT233HP, FT2232HP, FT2233HP, FT4232HP, FT4233HP, FT2232HA and
 - FT4232HA devices. Correct length of product description for multi-interface devices. Add builds for Mac OS X using Apple M1 Silicon.
- 1.4.22

- Improve response time to fast back-to-back FT_Read operations when data is available and buffered. Improve on fix introduced in * 1.4.20.
- 1.4.20
 - Fix issue with possible timeout occurring when reading fast bursts of data.
- 1.4.16
 - Fix issue with occasional data loss on very first packet from device when repeatedly opening and closing driver.
 - Update libusb to 1.0.22.
- 1.4.10
 - Implement functionality for FT_StopInTask and FT_RestartInTask.
- 1.4.8
 - Fix issue where ports opened within the same process do not show reliably in the FT_CreateDeviceInfoList.
- 1.4.6
 - Add ARM V8 build.
- 1.4.4
 - Reduce time taken to perform an FT_CreateDeviceInfoList.
 - Small performance gains on FT_read.
- 1.4.2
 - Rename constructor and destructor to avoid conflict with MySQL.
 - Minor updates to ftd2xx.h to match Windows version.
- 1.4.0
 - Update libusb from 1.0.18 to 1.0.20.
 - Remove lib_table method for specifying custom VID+PIDs.
- 1.3.7
 - Allow FT_SetVIDPID to be called multiple times (no need to save and restore).
 - The LibTable method still works but will be removed soon.
- 1.3.6
 - Release to web as platform-specific tgz packages.
- 1.3.5
 - Add support for ARMv7 uClibc targets.

- 1.3.3
 - Fix potential EEPROM corruption when writing to User Area on type 56 and 66 EEPROMs.
 - Harmonise ftd2xx.h with up-to-date Windows version.
 - Fix out-of-step versions of ftd2xx.h and WinTypes.h in examples directory.
- 1.3.1
 - Add builds for various ARM flavours, including v5, v6 and v7.
- 1.3.0
 - Replace libusb 1.0.12 with libusb 1.0.18 (re-unified with libusb).
- 1.2.8
 - Remove dependence on GLIBC 2.7 again, accidentally re-introduced in 1.2.0 with the switch to libusb.
 - Remove dependence of 64-bit library on GLIBC 2.14.
- 1.2.2
 - Fix bug where small numbers of bytes might be lost at low transfer speeds.
- 1.2.0
 - Replace libusb (1.0.8) with libusb (1.0.12).
 - Fix leaks introduced in 1.1.12: make FT_Close wait for outstanding requests.
 - Fix bug where FT_GetStatus would silently ignore bytes received after waking from sleep: treat time-out like removal.
 - Fix Mac crash discovered investigating above bug by reducing memory churn.
 - Fix FT_SetDeadmanTimeout implementation: now matches Windows driver, and uses the device's default value, rather than the system-wide hard-coded default.
 - Fix a few early-exit crashes.
- 1.1.12
 - Add basic support (not EEPROM) for X-series chips.
 - Reduce memory churn during Bulk-In processing.
 - Avoid potential crashes in FT_Close by not cancelling outstanding transfers: note - this causes FT_Close to leak a few bytes in some scenarios.
 - Improve thread safety to fix potential shutdown crashes.
- 1.1.10
 - Add timeout for FT_Read; zero means infinite.

- Stop reading Bulk-In data when buffer is full, to avoid data loss. (Now matches Windows driver behaviour.)
- Fix a potential crash in FT_Read.
- Disable libusb's internal logging as it may be implicated in crashes.
- Fix potential crash when using Parallel FIFO (fast) reads.
- 1.1.0
 - Improve throughput by using multiple intermediate buffers and asynchronous libusb requests.
 - Use separate threads for requesting and processing Bulk-In data, like the Windows driver.
 - Support use in Android JNI projects.
 - Add FT_CyclePort function. As with the Windows driver, this causes a re-enumeration of the device.
 - Fix bug where FT_SetFlowControl had no effect without explicitly setting RTS or DTR.
 - Improve unplug detection; better Bulk-In error handling after machine sleep.
 - Avoid potential crash in static build when freeing a config descriptor allocated with insufficient permissions (0660).
 - Correct Rx and Tx LED definitions for FT232H device EEPROM CBUS options.
 - Pass Valgrind test: avoid leaking config descriptor, and free the device list when FT_Open exits early.
 - Improve read timeouts.
 - Fix FT_GetBitMode and FT_GetLatencyTimer bugs.
- 1.0.4
 - Add support for FT232H, including access to EEPROM (and its User Area).
 - Fix multi-threaded write bug.
 - Fix potential hang when accessing the device list.
 - Disable libusb's use of timerfd, since this requires GLIBC 2.7.
 - Fix bug opening devices using bad strings.
- 1.0.3 Fix potential crash in FT_ListDevices.
- 1.0.2
 - Remove dependence on GLIBC 2.8.
 - Resolve issue where attempting to open a higher index interface on a device than is available could cause a segmentation violation.
- 1.0.1
 - Remove libusb kernel_driver calls as this could allow competing multiple
 - D2XX processes to access the same device.

- Remove dependence on GLIBC 2.4.
- 1.0.0
 - Major upgrade.
 - Fix numerous bugs.
 - Add support for location IDs.
 - Upgrade libusb to version 1.0.8 - requires udev (present on all 2.6 kernels).
- 0.4.14
 - Add enhancement to enable use in SuSe 10.2 default kernel.
 - Correct FT_GetDeviceInfoList behaviour.
- 0.4.13
 - Fix software flow control (strip out flow control characters).
 - Add FT_SetDeadmanTimeout API call.
- 0.4.12
 - Add configuration file option added to perform a usb reset on device open. (See Config.txt for further details.)
- 0.4.11
 - Enhance FT_SetUSBParameters - check for invalid parameters.
 - Add driver version functions.
 - Alter behaviour of a usb reset - only performed on unopened device (2.6 kernels only).
 - Improved multi-threaded access.
- 0.4.10
 - Add W32 functions and samples (for the extra event handling).
 - Add support for FT232R EEPROM.
 - Fix minor FT_OpenEx bugs.
 - Robust checking of passed in arguments. Fail on invalid handle or NULL pointers instead of causing Segmentation Faults.
- 0.4.9
 - Fix FT_OpenEx bug. Please report any future bugs, as different distributions may behave in a different manner.
 - Fix 100% CPU count when using FT_SetBitMode and not using FT_Read to clear internal buffers.
- 0.4.8
 - Fix FT_Open bug - related to the detach of ftdi_sio kernel driver.
- 0.4.7

- Fix Read-timeout bug.
- 0.4.6
 - Fix bug which prevented programming of FT2232 user area.
- 0.4.5
 - Allow detaching of ftdi_sio kernel driver on device access.
- 0.4.4
 - Add FT_CreateDeviceInfoList, FT_GetDeviceInfoList, FT_GetDeviceInfoDetail,
 - FT_EE_ReadEx and FT_EE_ProgramEx.
- 0.4.3
 - Fix bulk-write bug on 2.6 kernels - requires “/proc/sys/kernel/osrelease” file to be present.
- 0.4.2
 - Fix bug in lib_table functionality. Now works with both static and dynamic library versions.
- 0.4.1
 - Fix modem event notification bug.
- 0.4.0
 - Compile user-space driver with GCC version 3.3.1.
 - Convert all library code to C, so no C++ extensions required - should help with some C++ linkage problems. PLEASE NOTE: due to the conversion to C there is a small chance that you will experience problems; a full recompile of your application may solve this. If not, please contact FTDI Support. Users of the PenScope and Dualscope may experience problems related to this. Please contact FTDI Support for a suitable update.
 - Fix FT_GetBitMode bug.
 - Fix FT_Read timeout bug.
 - Fix FT_GetStatus bug.
 - Static library available (see static folder on how to compile).
 - Static linkage to libusb to avoid installation issues.
- 0.3.0
 - Add timeouts.
 - Add EEPROM reading/writing.
 - Add event handling (see example on how to use).
 - Add more examples to help with development.
 - Add FT_SetVIDPID - see example.

Installation

Installing the D2XX shared library and static library.

1. `tar xfvz libftd2xx-PLATFORM_NAME-FULL_VERSION_NUMBER.tgz`

This unpacks the archive, creating the following directory structure.

```
linux-x86_64
  Event.h
  examples
    BitMode
    EEPROM
    Events
    ftd2xx.h
    loopback
    Makefile
    MultiThread
    Rules.make
    SetVIDPID
    W32
      escapeseq
      events
      simple
    WinTypes.h
    write
  ftd2xx.h
  libftd2xx.so
  libftd2xx.so.1.4.27 -> libftd2xx.so
  libftd2xx-static.a
  libusb.h
  README.pdf
  release-notes.txt
  WinTypes.h
```

2. `cd d2xx_release_packages`

3. `sudo -s`

or, if sudo is not available on your system: `su`

Promotes you to super-user, with installation privileges. If you're already root, then step 3 (and step 7) is not necessary.

4. `cp libftd2xx.* /usr/local/lib`

Copies the libraries to a central location.

5. `chmod 0755 /usr/local/lib/libftd2xx.so.FULL_VERSION_NUMBER`

Allows non-root access to the shared object.


```
6. ln -sf /usr/local/lib/libftd2xx.so.FULL_VERSION_NUMBER /usr/local/lib/libftd2xx.so
```

Creates a symbolic link to the FULL_VERSION_NUMBER version of the shared object.

```
7. cd ..  
   cp ftd2xx.h /usr/local/include  
   cp WinTypes.h /usr/local/include
```

Copies the header files to a central location.

```
8. ldconfig -v
```

Update the linker shared object cache to include the newly installed library.

```
9. exit
```

Ends your super-user session.

Notes on Kernel Built-in Support of FTDI devices.

On most distributions, the linux kernel will have either a built-in or optional module called “ftdi_sio”. This will detect an FTDI device and automatically invoke the “usbserial” module and create devices such as “/dev/ttyUSB0”.

When the ftdi_sio module is controlling an FTDI device it is not available to libftd2xx. If the library attempts to access the device it will receive a message “FT_Open failed”.

There are several methods of preventing ftdi_sio from controlling FTDI devices.

- 1) Remove the ftdi_sio module from the running kernel:

```
sudo lsmod | grep ftdi_sio
```

If “ftdi_sio” is listed unload it (and its helper module, usbserial):

```
sudo rmmod ftdi_sio  
sudo rmmod usbserial
```

To reverse the operation the kernel modules can be reloaded using modprobe instead of rmmod.

- 2) Build a new kernel without the ftdi_sio module.

Refer to your distributions instructions for building a custom kernel.

- 3) Use a udev unbind sysfs interface to disable devices as they are connected.

First identify the device identifier to remove.

```
ls /sys/bus/usb/drivers/ftdi_sio
```

This will show a list of ports on USB devices that are linked to the ftdi_sio driver.

These are in the form 1-2:1.0 (bus number 1 - port number 2 : device 1 . interface 0) for Port A, and 1-2:1.1 would be Port B on the same device.

The `/dev/ttyUSBx` node can be mapped to the device identifier using:

```
lsusb -t
```

The devices will be listed in a tree format which can be mapped to the device identifiers.

Identify the device which you wish to use and then send the identifier of the device to the “unbind” interface in the driver. (The tee function just echos stdin to stdout with privilege escalation through sudo).

```
echo -n 1-2:1.1 | sudo tee /sys/bus/usb/drivers/ftdi_sio/unbind
```

To reverse the process the same device number can be sent to the “bind” interface to re-enable the USB serial device.

This can be scripted through udev rules to happen when a device connection change is detected.

Adding Your Device’s Vendor and Product Identifiers

If your FTDI device has a Vendor ID (VID) or Product ID (PID) that differs from the standard devices then you can call `FT_SetVIDPID` before calling the `FT_Open`, `FT_OpenEx`, `FT_ListDevices` or `FT_CreateDeviceInfoList` functions.

Modified VID and PIDs can also be added to the list of devices which invoke `ftdi_sio` by sending the VID and PID of the device to the “new_id” interface. The VID and PID can be found with `lsusb`.

```
echo -n 0403 1001 | sudo tee /sys/bus/usb-serial/drivers/ftdi_sio/new_id
```

This can either be done after the device is installed or can be done before the device is inserted if an appropriate `modprobe` command is performed.

Building the shared-object examples.

```
1. cd examples
```

```
2. make -B
```

This builds all the shared-object examples in subdirectories.

With an FTDI device connected to a USB port, try one of the examples, e.g. reading EEPROM.

```
3. cd EEPROM/read
```

4. `sudo ./read`

Building the static-library example.

1. `cd examples/static`

2. `rm lib*`

Cleans out any existing libraries built for another target.

3. `cp /usr/local/lib/libftd2xx.a .`

4. `make -B`

5. `sudo ./static_link`

This example demonstrates writing to, and reading from, a device with a loop-back connector attached.

The examples show how to call a small subset of the D2XX API. The full API is available here: http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer%27s_Guide