AN75705

Getting Started with EZ-USB® FX3™

Author: Sonia Gandhi
Associated Project: No
Associated Part Family: EZ-USB® FX3™
Software Version: SDK 1.2
Related Application Notes: For a complete list of application notes, click here

Abstract

AN75705 gets you started with the EZ-USB® FX3™ USB 3.0 Device Controller. It provides background information on USB 3.0 and comparisons to USB 2.0. It details hardware, firmware, and software aspects of working with the FX3. Finally, it details the steps to verify the working condition of DVK and its compatibility with the host system.

Contents

Introduction

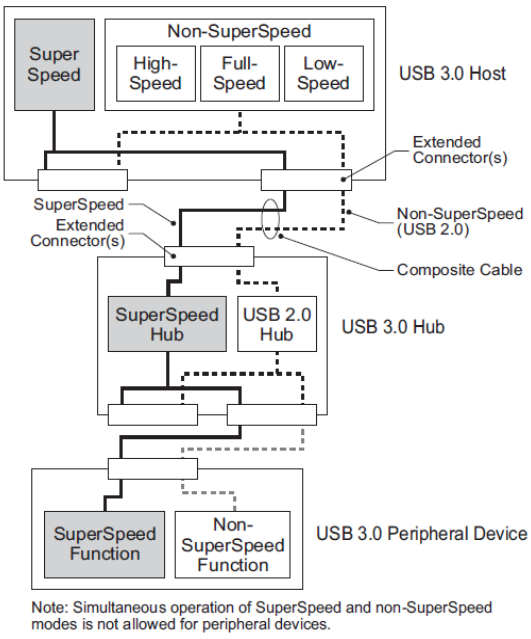
The Cypress EZ-USB® FX3 peripheral controller lets developers add USB 3.0 functionality to any system.

The FX3 Developer Kit (DVK) helps designers to build firmware prior to prototyping their own boards. This application note provides all that is necessary for a developer to get up and running with the FX3 DVK. It provides the background information for the USB 3.0 interface as well as the basics of FX3 and its architecture. This application note also introduces the reader to the firmware and software provided as part of the FX3 SDK. Finally this application note guides the developer to set up the USB 3.0 host and confirm proper operation with the FX3 DVK.

USB 3.0 Overview

USB 3.0 enables an increased data rate of 5 Gbps, reduces power consumption and is backward-compatible with USB 2.0. The USB specification is published by USB-IF and can be found here. The architectural components of USB 2.0 and USB 3.0 are similar, and Figure 1 shows the USB 3.0 architecture.

Figure 1. USB 3.0 Dual Bus Architecture



Courtesy: www. http://usb.org

The physical interface of USB 3.0 consists of two differential pairs and a ground for SuperSpeed transfers in addition to the USB 2.0 connections. This allows USB 3.0 to ensure the backwards compatibility with USB 2.0.

Electrical Interface

The USB 3.0 pinout is different from that of USB 2.0. In addition to the VBUS, D-, D+, and GND pins required for USB 2.0, USB 3.0 has five additional pins – two differential pairs plus one ground (GND_DRAIN). The two differential pairs are for SuperSpeed data transfer, supporting dual simplex SuperSpeed signaling. The GND_DRAIN pin is for drain wire termination, management of signal integrity, and EMI performance. The following table shows a description of the nine pins.

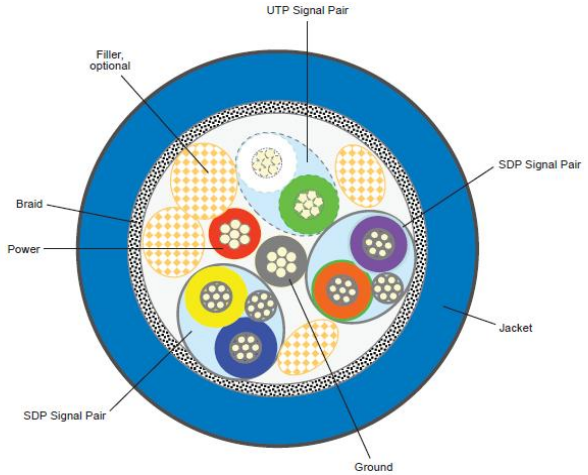
Table 1. USB 3.0 Pin Description

Pins Name	Description
VBUS	Power
D-	USB 2.0 differential pair
D+	
GND	Ground for power return
SSRX-	SuperSpeed receiver differential pair
SSRX+	
SSTX-	SuperSpeed transmit differential pair
SSTX+	
GND_DRAIN	Ground for signal return

Cables and Connectors

USB 3.0 has four additional data lines (SSRX+, SSRX-, SSTX+, SSTX-) for data transfer and one additional ground lines for drain wire termination, signal integrity management, and EMI performance. Figure 2 shows the architecture of USB 3.0 cable. Table 2 shows the description of these lines.

Figure 2. USB 3.0 Cable Architecture



Courtesy: www. http://usb3expresscard.com

Table 2. USB 3.0 Cable Description

Name	Description	Color
PWR	VBUS	Red
UTP_D-	USB 2.0 D-	White
UTP_D+	USB 2.0 D+	Green
Ground	Ground for power drain	Black
SDP1-	Shielded differential pair 1	Blue
SDP1+		Yellow
SDP1_Drain	Drain line for SDP1	
SDP2-	Shielded differential pair 2	Purple
SDP2+		Orange
SDP2_Drain	Drain line for SDP2	

The USB 3.0 specification also defines the following connectors:

- **USB 3.0 Standard-A Plug and Receptacle**

  The USB 3.0 Standard-A connector is defined in the SuperSpeed standard as the host connector. It is based off the design of the USB 2.0 Standard-A connector but has the addition SuperSpeed signals. A USB 3.0 Standard-A receptacle accepts either a USB 3.0 Standard-A plug or a USB 2.0 Standard-A plug. USB 3.0 capable Standard-A connectors use unique color in order to make identification easy. Figure 3 shows the color coding recommendation.

- **USB 3.0 Standard-B Plug and Receptacle**

  The USB 3.0 Standard-B connector is defined for large, stationary peripherals, such as external hard drives and printers. The USB 3.0 Standard-B receptacle accepts either a USB 3.0 Standard-B plug or a USB 2.0 Standard-B plug. You cannot insert a USB 3.0 Standard-B plug into a USB 2.0 Standard-B receptacle.

- **USB 3.0 Powered-B Plug and Receptacle**

  The USB 3.0 Powered-B connector enables a USB 3.0 device to provide power to a USB adapter without an external power supply. It is identical to the USB 3.0 Standard-B connector in form factor but has two more pins: one for power (DPWR) and one for ground (DGND).

- **USB 3.0 Micro-B Plug and Receptacle**

  The USB 3.0 Micro-B connector is defined for small handheld devices.

- **USB 3.0 Micro-AB and USB 3.0 Micro-A Connectors**

  The USB 3.0 Micro-AB receptacle is similar to the USB 3.0 Micro-B receptacle, except keying is different. It accepts a USB 3.0 Micro-A plug, a USB 3.0 Micro-B plug, a USB 2.0 Micro-A plug, or a USB 2.0 Micro-B plug. The USB 3.0 Micro-AB receptacle is allowed only on OTG products, which may function as either a host or a device. All other uses of the USB 3.0 Micro-AB receptacle are prohibited.

  The USB 3.0 Micro-A plug is similar to the USB 3.0 Micro-B plug, except for different keying and ID pin connections. The USB 3.0 Micro-A plug, the USB 3.0 Micro-AB receptacle, and the USB 3.0 Micro-B receptacle and plug, belong to the USB 3.0 Micro connector family. Their interfaces differ only in keying. Similar to the USB 2.0 Micro-A plug, the USB 3.0 Micro-A plug is defined only for OTG applications.

USB 3.0 versus 2.0

USB 3.0 has a dual-bus architecture that supports USB 2.0 and 3.0. The following table shows the differences between USB 3.0 and USB 2.0.

Table 3. Differences Between USB 3.0 and USB 2.0

Feature	USB 2.0	USB 3.0
Data rate	480 Mbits/s (High Speed) 12 Mbits/s (Full Speed) 1.5 Mbits/s (Low Speed)	5.0 Gbits/s (SuperSpeed) 480 Mbits/s (High Speed) 12 Mbits/s (Full Speed) 1.5 Mbits/s (Low Speed)
Data interface	1. Half-duplex 2. Two-wire differential signaling	1. Dual-simplex 2. Four-wire differential signaling
Cable signal count	Four signals: 1. Two for USB 2.0 data (D, D–) 2. Two for VBUS and GND	Nine signals: 1. Four for SuperSpeed data 2. Two for USB 2.0 data (D, D–) 3. Three for VBUS and GND
Bus transaction protocol	1. Host directed 2. Polled traffic flow 3. Packets broadcast to all downstream devices 4. No multiplexing of data streams	1. Host directed 2. Asynchronous notifications 3. Packets routed only to target device 4. Multiple data streams possible for bulk transfers
Power management	Two modes 1. Active 2. Suspend	Four modes 1. Active (U0) 2. Idle, Fast (U1) 3. Idle, Slow (U2) 4. Suspend, Slow (U3)
Bus power	Low-power device : 100 mA High-power device : 500 mA	Low-power device : 150 mA High-power device : 900 mA
Port state	Port hardware detects connect events. System software uses port commands to transition the port into an enabled state.	Port hardware detects connect events and brings the port into operational state ready for SuperSpeed data communication
Max Cable Length	5 meters	Based on electrical specification. In practice 3 meters for 26 AWG copper
Data transfer types	Four data transfer types: control, bulk, interrupt, and isochronous.	USB 2.0 types with SuperSpeed capabilities. Bulk has streams capability.

Introduction to FX3

FX3 has a USB 3.0 device controller, USB 2.0 OTG controller and contains 512 KB of SRAM for code and data. FX3 supports a highly flexible programmable parallel interface known as GPIF II. It also has UART, SPI, I^2C, and I^2S interfaces to connect to serial peripherals.

FX3 versus FX2LP

As the next-generation USB peripheral controller based on EZ-USB FX2LP, FX3 provides enhancements on certain FX2LP features. The following table shows the differences between FX3 and FX2LP.

Table 4. Differences Between FX3 and FX2LP

Feature	FX2LP	FX3
USB support	1. Supports USB 2.0 2. Supports 6 physical endpoints	1. Supports USB 3.0 and USB 2.0 2. Supports High-speed On-The-Go (HS-OTG) host and peripheral 3. Supports battery charging Spec 1.1 and accessory charger adaptor (ACA) detection 4. Supports 32 physical endpoints
MCU architecture	Standard enhanced 8051 core with 48-MHz operation	32-bit ARM926EJ Core with 200-MHz operation
SRAM	16 kB	512 kB
Boot options	1. Boot from USB 2. Boot from I^2C	1. Boot from USB 2. Boot from I^2C 3. Boot from SPI 4. Boot from GPIF II ASync ADMUX mode 5. Boot from GPIF II Sync ADMUX mode 6. Boot from GPIF II ASync SRAM mode
GPIF	1. 48 MHz with 8/16 bit data bus 2. Up to six configurable control signals 3. 8 firmware programmable states	1. 100 MHz with 8/16/32 bit data bus 2. 16 configurable control signals for 8-/16-bit data bus and 14 signals for 32-bit data bus 3. 256 firmware programmable states
I^2C	I^2C master support 100/400 KHz for 3.3-V device	I^2C master support 100 kHz, 400 KHz, 1 MHz (1.8 V/2.5 V/3.3 V required for 400 KHz and 1 MHz)
Power mode	Supports two power modes: ▪ Normal mode ▪ Low power mode	Supports five modes: ▪ Normal mode ▪ Suspend mode with USB 3.0 PHY enabled (L1) ▪ Suspend mode with USB 3.0 PHY disabled (L2) ▪ Standby mode (L3) ▪ Core power down mode (L4)
Operation system	N/A	ThreadX RTOS
Compiler and IDE	Keil C for 8051.	Eclipse with GNU ARM tool chain
Clock	Crystal with 24 MHz	1. Crystal with 19.2 MHz 2. External clock: 19.2, 26, 38.4, and 52 MHz

Benefits

FX3 comes with the Cypress SuperSpeed USB Suite which speeds application development. The software development kit comes with application examples to accelerate time to market. Some of the major benefits are

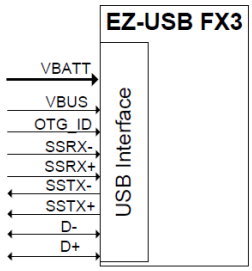
- 10x performance compared with USB 2.0
- Connects with any industry standard or non standard interfaces in systems with FPGA, ASIC, ASSP, and image sensors
- Fully accessible core with 512 KB of on-chip SRAM to build custom applications
- Better control over low power states
- Faster battery charging
- Enables connectivity to on-system peripherals

FX3 Architectural Overview

USB Interface

FX3 supports USB peripheral functionality compliant with USB 3.0 Specification Revision 1.0 and is compliant with OTG Supplement Revision 2.0. The following figure shows the signals of the FX3 USB interface.

Figure 3. EZ-USB® FX3 USB Interface



- **OTG**

  FX3 complies with the OTG Specification Revision 2.0. In OTG mode, FX3 supports both A and B device modes as well as Control, Interrupt, Bulk, and Isochronous data transfers. FX3 requires an external charge pump to power VBUS in OTG A-device mode and does not support the Attach Detection Protocol (ADP).

- **EZ-Dtect®**

  FX3's EZ-Dtect®, mechanism complies with Battery Charging Specification Revision 1.1. FX3 provides hardware support to detect resistance values on the ID pin to meet the specification request. EZ-Dtect can detect a dedicated wall charger, host/hub charger, and host/hub.

GPIF II Engine Overview

FX3 offers a high-performance GPIF II interface that enables functionality similar to, but more advanced than, FX2LP's GPIF and Slave FIFO interfaces.
The GPIF II is a programmable state machine that enables a flexible interface that may function either as a master or a slave in industry standard or proprietary interfaces.

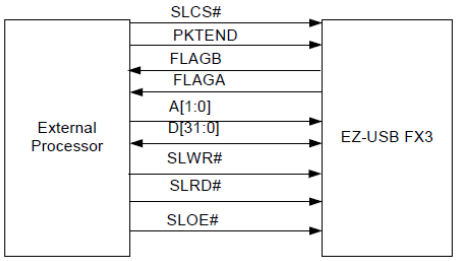
GPIF has the following features:

- Functions as a master or a slave.
- 256 firmware programmable states.
- 8-bit, 16-bit, and 32-bit parallel data bus.
- Enables interface frequencies of as high as 100 MHz.
- Supports 14 configurable control pins when the 32-bit data bus is used. All control pins can be either input/output or bidirectional.
- Supports 16 configurable control pins when 16/8 data bus is used. All control pins can be either input/output or bidirectional.

For master mode, it is easy to use GPIF II to implement various different parallel and serial interface host controllers.

For slave mode, GPIF II allows an external processor to directly access as many as 32 buffers internal to FX3 with 5-bit Slave FIFO. Figure 4 shows how the processor connects to FX3 with Slave FIFO interface using a 2-bit addressing scheme.

Figure 4. Slave FIFO Interface



Other Interfaces

FX3 supports different types of peripherals, including UART, I^2C, I^2S, and SPI.

- **UART**

  FX3 UART supports full duplex. It can generate a range of baud rates, from 300 bps to 4608 Kbps, selectable by the firmware.

- **I^2C**

  FX3 has an I^2C interface compatible with the I^2C Bus Specification Revision 3. The FX3 I^2C interface can operate only as an I^2C master.

  The bus frequencies supported by the I^2C controller are 100 kHz, 400 kHz, and 1 MHz. When the power domain of I^2C is 1.2 V, the maximum operating frequency supported is 100 kHz. When the power domain of I^2C is 1.8 V, 2.5 V, or 3.3 V, the operating frequencies supported are 100 KHz, 400 kHz and 1 MHz. Both SCL and SDA signals of the I^2C interface require external pull-up resistors, which must be connected to the I^2C power domain.

- **I^2S Master**

  The FX3 has the capability to act as an I^2S master transmitter. This means that the FX3 will generate the clock as well as the data for the I^2S interface. The I^2S interface consists of four signals: clock line

(I2S_CLK), serial data line (I2S_SD), word select line (I2S_WS), and master system clock (I2S_MCLK). The sampling frequencies supported by the I^2S interface are 32 kHz, 44.1 kHz, and 48 kHz.

- **SPI**

  FX3 supports a SPI Master interface. The maximum frequency of operation is 33 MHz.

DMA Overview

FX3 implements an efficient and flexible DMA engine that can control data transfer between peripherals (such as, USB, GPIF II, I2S, SPI, and UART). DMA requires firmware only for initialization. Once running there is no processor intervention required.

The DMA engine module in firmware consists of a set of low-level functions that manage the registers that control the various data flows within the system. All of the other firmware modules make calls into this module to set up and check the data transfers.

FX3 Solutions Overview

As a general purpose USB 3.0 device controller, FX3 is capable of providing a wide range of solutions. Typical solutions being created today are products which require bandwidth which exceeds the capability of USB 2.0. This includes but is not limited to: HD webcams, Industrial imaging, medical devices, logic analyzers/generators, high resolution scanners, and professional audio mixing devices. Next we will look into how such solutions are created.

Introduction to FX3 DVK

Cypress's FX3 DVK provides all of the hardware that you need in order to get started. It provides all of the necessary clocks and voltages for the FX3 as well configurable GPIO voltages. The DVK has standard high speed mictor connectors for interfacing with your own devices. The DVK also provides the ability to work with various boot mode settings and includes a SPI flash and an I^2C EEPROM socket for testing these boot modes.

If you do not already have a DVK you can find one **here**.

Two important entities that are external to the FX3 are:

- USB host/device
  - When the FX3 is connected to a USB host, it functions as a USB device. The FX3 enumerates as a super-speed, high-speed, or full-speed USB peripheral corresponding to the host type.
  - When a USB device is connected, the FX3 plays the role of the corresponding high-speed, full-speed, or low-speed USB host.
- GPIF II master/slave
  - GPIF II is a fully configurable interface that can realize any application-specific protocol. Any processor, ASIC, DSP, or FPGA can be used to interface with the FX3. GPIF II interfaces can be programmed via firmware.

Introduction to FX3 SDK

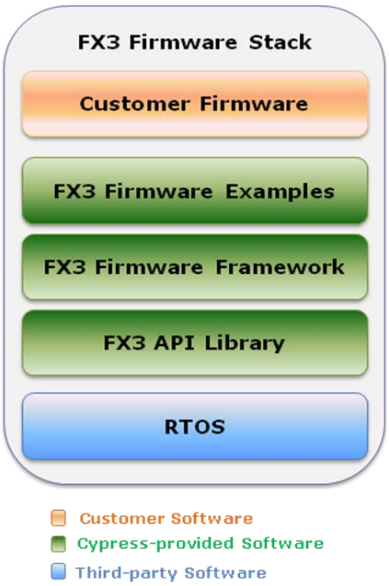
Cypress delivers the complete software and firmware stack for FX3 to integrate USB applications in the embedded system environment. The software development kit comes with application examples that accelerate application development. The SDK is available for download from the Cypress website.

Firmware Stack

Cypress provides a powerful API library to make developing firmware easier even for complex designs. Here are some of the advantages provided.

- RTOS
  - The included ThreadX Real Time OS simplifies the firmware development process. With the RTOS multiple threads can easily be created to simplify the firmware flow.
- Modular approach
  - An API-based approach means that the customer doesn't have to worry about FX3 internals but can focus on the firmware logic and flow. This approach is easy to use, debug, modify, and support.

Figure 5. Structure of Firmware SDK



Firmware API

A full-fledged API library comes with the FX3 SDK. The API library and the corresponding header files provide all the APIs required for programming the different blocks of the FX3. All of these APIs are documented in the API Guide that is distributed as part of the SDK. A small subset of the APIs will be discussed in this application note. The APIs can be classified as high level and advanced.

High-Level APIs

These include APIs for USB, GPIF II, DMA, serial interfaces (UART, I^2C, I^2S, SPI, GPIO), and OS system call functionality. This set of APIs provides:

- Programming each individual block of the FX3 device, the GPIF, USB, and serial interfaces
- Programming the DMA engine and setting up of data flow between these blocks
- ThreadX OS calls as required by the application
- Debug capability

Advanced APIs

These include APIs for debug, power management, low-level DMA blocks, and advanced OS system call functionality. This set of APIs provides:

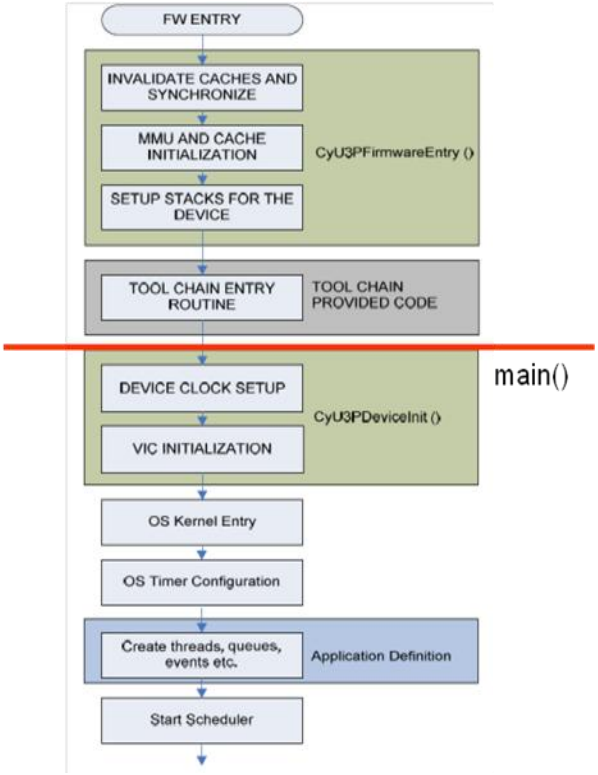
- USB host mode of operation
- USB power mode handling
- Programming of the low-level DMA engine
- Power management features
- Porting to a different OS

Framework API

The firmware (or application) framework has all the startup and initialization codes. Also included are the individual drivers for the USB, GPIF, and serial interface blocks. The framework performs the following functions:

- Defines the program entry point
- Performs the stack setup
- Performs kernel initialization
- Provides placeholders for application thread startup code

Figure 6. Firmware Sequence



Initialization Code

CyU3PFirmwareEntry

The entry point for the FX3 firmware is the `CyU3PFirmwareEntry()` function. It is defined in the FX3 API library and is not visible to the user. As part of the linker options, the entry point is specified as the `CyU3PFirmwareEntry()` function. The firmware entry function performs the following actions:

- Invalidates the caches (used by the bootloader)
- Initializes the Memory Management Unit (MMU) and the caches
- Initializes the SYS, FIQ, IRQ, and SVC modes of stacks
- Transfers the execution the tool chain initialization (`CyU3PToolChainInit()`) function.

CyU3PToolChainInit

The next step in the initialization sequence is the tool chain initialization. This is defined by the specific tool chain and provides a method to initialize the stacks and the C library.

Note: The required stack initialization is performed by the firmware entry function, the tool chain initialization is overridden; that is, the stacks are not re-initialized.

In this function, only two actions are performed:

- Clears the BSS area
- Transfers the control to `main()`

CyU3PDeviceinit

This is the first user-defined function in the initialization sequence. The function `main()` is the C programming language entry for the FX3 firmware. This function does the following:

- Device initialization
  - Sets up the CPU clock. A NULL is passed as an argument for `CyU3PDeviceInit()`, which selects the default clock configuration.
  - Initializes the VIC
  - Configures the GCTL and the PLLs
- I/O matrix configuration

  The second step is to configure the I/Os required, including the GPIF and the serial interfaces (SPI, I^2C, I^2S, GPIO and UART). It initializes the I/O matrix configuration data structure and invokes the `CyU3PDeviceConfigureIOMatrix()` function (in the library).
- Invocation of the OS

  The final step in the `main()` function is invocation of the OS.

CyFxApplicationDefine

The function `CyFxApplicationDefine()` is called by the FX3 library after the OS is invoked. This function creates application-specific threads.

FX3 Firmware Examples

There are many firmware examples included in the firmware directory of the SDK. This section details several of the more common example firmware projects. Complete list of currently available firmware examples is in the SDK release notes.

Note: All example firmwares are provided as Eclipse projects.

USB Bulk Data Loopback Examples

These illustrate a loopback mechanism between two or three USB bulk endpoints. The example consists of Vendor Class USB enumeration descriptors with two or three bulk endpoints. The DMA multichannel examples use three endpoints for the loopback.

USB Isochronous Data Loopback Examples

These examples illustrate a loopback mechanism between two USB isochronous endpoints. The example consists of Vendor Class USB enumeration descriptors with two isochronous endpoints.

USB Video Class Example (cyfxuvcinmem)

This example implements a USB Video Class Driver using the appropriate USB enumeration descriptors. With these descriptors, the FX3 device enumerates as a USB Video Class device on the USB host. The video streaming is accomplished with the help of a DMA Manual Out channel. Video frames are stored in contiguous memory locations as a constant array. The streaming of these frames continuously from the device produces a video-like appearance on the USB host. The example works at both USB 2.0 and USB 3.0 speeds.

Slave FIFO Application Examples

The Slave FIFO application example demonstrates data loopback between the USB host and an external master. The example consists of two unidirectional data pipes between the USB host and the external master. Data is transmitted by the USB host and then echoed back by the external master. The loopback is observed on the USB host. The GPIF II interface can be configured for either synchronous or asynchronous Slave FIFO. There is also the option for a 16 or 32 bit wide data bus.

Serial Interface Examples

These examples demonstrate data accesses to the GPIOs, I^2C, SPI, and UART.

USB Bulk/Isochronous Data Source Sink Examples

These examples illustrate data source and data sink mechanism with two USB bulk/isochronous endpoints. The example consists of Vendor Class USB enumeration descriptors with two bulk/isochronous endpoints. The examples work at both USB 2.0 and USB 3.0 speeds.

USB Bulk Streams Example

This example illustrates the data source and data sink mechanism with two USB bulk endpoints using the bulk streams. A set of bulk streams is defined for each of the endpoints (OUT and IN). Data source and data sink happen through these streams, which apply to USB 3.0 speeds only. The example works in a fashion similar to the bulk source sink example (cyfxbulksrcsink) when connected to USB 2.0.

Flash Programmer example

This example illustrates the programming of I^2C EEPROMS and SPI flash devices from USB. The read or write operations are done using pre-defined vendor commands. The utility can be used to flash the boot images to these devices.

Mass Storage Class example

This example illustrates the implementation of a USB mass storage class (Bulk Only Transport) device using a small section of the FX3 device RAM as the storage device. The example shows how mass storage commands can be parsed and handled in the FX3 firmware.

USB Audio Class Example

This example creates a USB Audio Class compliant microphone device, which streams PCM audio data stored on the SPI flash memory to the USB host. This example works only at USB 2.0 speeds.

Two Stage Booter Example

A simple set of APIs have been provided as a separate library to implement two stage booting. This example demonstrates the use of these APIs. Configuration files that can also be used for Real View Tool chain are provided.

USB host and OTG examples

These examples demonstrate the host mode and OTG mode operation of the FX3 USB port.

FX3 SDK Documentation

The FX3 SDK includes the following documentation:

- Getting Started with FX3 SDK
- FX3 Programmer's Manual
- FX3 API Guide

Directory Structure

On installation of the FX3 SDK, the directory structure shown in Figure 7 is created.

Figure 7. Directory Structure

Windows Software Overview

Structure

Cypress delivers the complete device driver and interface APIs with library for FX3 to use all USB applications in Windows.

Windows USB Device Driver

CYUSB3.SYS is a USB device driver for 32-bit Windows XP, 32/64-bit Windows Vista, 32/64-bit Windows 7 and 32/64-bit Windows 8, which is capable of communicating with any devices that complies with USB 2.0 and USB 3.0. The driver is general-purpose and understands primitive USB commands. However, it does not implement higher-level commands and USB device-class specific commands. Therefore, the driver cannot interface a USB mass storage-class device to the Windows file system.

Note that this release does not include the USB 3.0 bulk streaming interface. As of SDK 1.2 this generic Cypress driver has been signed for a VID of 0x04B4 and a PID of 0x00F0, 0x00F1 and 0x00F3. If you wish to use your own VID/PID pair you must get the driver signed by Microsoft WHQL. During the development phase you can use the Cypress driver unsigned and disable driver signature enforcement on your computer. How to do this is detailed in the "Connection and Verifying your Setup" section of the document.

The driver is ideal for communicating with a vendor-specific device from a custom USB application. Or, the driver might be used to send low-level USB requests to any USB device for experimental or diagnostic applications. To use the driver to communicate with a device, Windows must match the device to the driver. The class library, CyAPI.lib and Cyusb.dll, provides a high-level programming interface to the driver.

Features

- Windows Driver Foundation (WDF)-compliant
- Compatible with any USB 2.0-compliant device
- Compatible with Cypress USB 3.0-compliant device
- Supports basic USB 3.0 features
- Supports Windows plug-and-play and power management
- Supports USB remote wake-up
- Supports control, bulk, interrupt, and isochronous endpoints
- Supports multiple USB devices connected at once
- Supports customizable driver GUID without rebuilding the driver
- Supports high-bandwidth data transfers passing multiple packets per frame

It is not required that you use the Cypress supplied driver. FX3 can be programmed to implement standard USB class devices. When implementing such a device the USB class driver should be used instead. For example if an FX3 is implementing the USB Video Class (UVC) it would not need the Cypress generic driver and instead would use the OS provided UVC driver. This makes Windows, Linux and Mac support simple as drivers are already provided as part of the class implementation.

Application Interface

CyAPI.lib

CyAPI.lib provides a simple, powerful C++ programming interface to USB devices. It is a C++ class library that provides a high-level programming interface to the CyUsb3.sys device driver. The library can communicate only with USB devices served by this driver.

For further details on CyAPI.lib refer to the either the Programmer Reference C# Library or the Programmers Reference C++ Library. These are distributed in the USB Suite as part of the SDK.

CyUSB.dll

CyUSB.dll is a managed Microsoft .NET class library that provides a high-level, powerful programming interface to USB devices. Rather than communicate with the USB device drivers directly via Win32 API calls such as SetupDiXxxx and DeviceIoControl, applications can access USB devices through library methods such as XferData and properties such as AltIntfc. Because CyUSB.dll is a managed .NET library, its classes and methods can be accessed from any of the Microsoft Visual Stuido.NET managed languages, such as Visual Basic.NET, C#, Visual J# and managed C++. To use the library, add a reference to CyUSB.dll to your project's **References** folder. Then, any source file that accesses the CyUSB namespace must include a line to add the namespace in the appropriate syntax.

Examples:

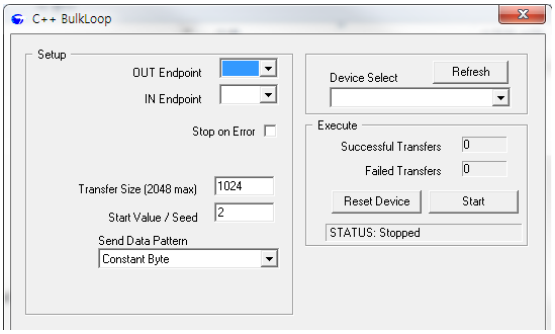
- Visual Basic.net : Imports CyUSB
- Visual C# : using CyUSB;
- Visual C++ (Win Forms App): using namespace CyUSB;
- Visual J# : import CyUSB.*;

The library employs a model of DeviceList, Devices, and EndPoints. An application normally creates an instance of the USBDeviceList class, which represents a list of USB devices. Each of those devices then can be accessed individually. The devices represented in the device list are vendor-specific USB devices (non-USB-class devices) served by the CyUSB.sys driver. Such members of the device list are instances of the CyUSBDevice class and expose one or more CyUSBEndPoints through which data transfers can occur.

Window Software Examples

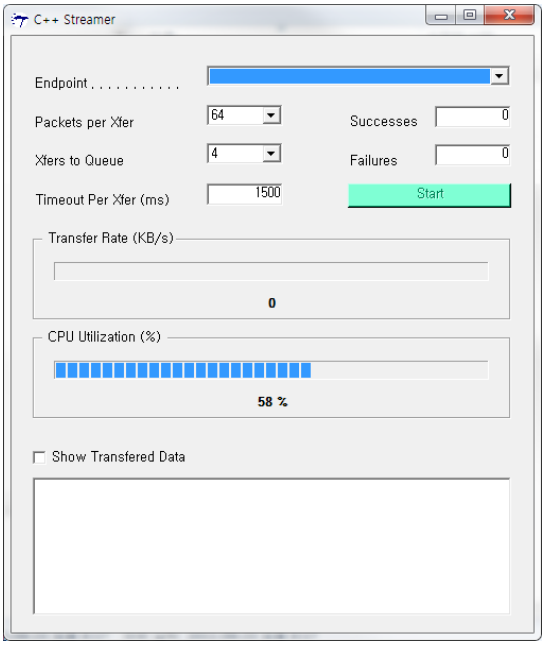
BulkLoop example

The application BulkLoop is used to test the loopback of data transfer through bulk endpoints.



Streamer Example

The application Streamer is used to test the data transfer on isochronous/bulk endpoints.



FX3 Firmware Development Tools

Eclipse IDE

The Eclipse IDE for C/C++ Developer is provided as part of the FX3 SDK. This IDE comprises the base Eclipse platform (3.5.2) and the CPP feature (1.2.2). Plug-ins required for development is bundled with the IDE.

- GNU ARM C/C++ Development support
- Zylin Embedded CDT
- Java(TM) Platform, Standard Edition Runtime Environment Version 7 (JRE)

GNU Tool Chain

The GNU tool chain provided as part of the FX3 SDK comprises the following:

- GCC compiler (gcc)
- GNU Linker (ld)
- GNU Assembler (as)
- GNU Debugger (gdb)

These executables are invoked by the Eclipse IDE.

GPIF II Designer

GPIF II Interface Design Tool is a Windows application that FX3 customers get as part of the FX3 SDK. The tool provides a graphical user interface to specify the necessary interface for the target device. The tool generates a C header file to be included in the Eclipse project.

JTAG Debugger

The Segger J-Link probe is the preferred JTAG probe for the FX3 SDK. This probe, along with the Segger J-Link ARM GDB Server, is used for debug. The Eclipse IDE will connect to the J-link GDB server when debugging your firmware. To get Eclipse working with the GDB server you need to create a debug configuration for the J-link. Details on how to do this are in the Programmers Manual, which is distributed as part of the SDK.

Debug Utility

USB 3.0 Protocol Analyzer

A USB 3.0 Protocol analyzer is a very useful debugging tool. It analyzes the traffic on the USB bus between the FX3 and the host. Software tools included with each analyzer then decode the data into USB transfer packets. By analyzing this data issues can be easily identified and performance can be maximized. There are several USB 3.0 analyzers available on the market today. Cypress does not recommend any specific analyzer but here are several options:

- **Standalone USB 3.0 Protocol Analyzer**
  - Ellisys USB Explorer 280
  - LeCory USB Voyager M3i
  - Beagle USB 5000 SuperSpeed Protocol Analyzer
- **PC Software USB 3.0 Protocol Analyzer**
  - SourceQuest SourceUSB
  - SysNucleus USBTrace

Logic Analyzer

Logic analyzers allow simple analysis of digital signals. They can be used to look at various GPIOs between the FX3 and other peripherals. Two types of logic analyzer are on the market.

- **Standalone Type L.A.**
  - Agilent 16800 Series Portable Logic Analyzer
- **PC-Based Type L.A.**
  - USBee Logic Analyzer
  - ZeroPlus Logic Analyzer

The most important factor to consider is the signal frequency range. If the analyzer supports only a lower frequency range to the signal that is watched, then the analyzer cannot function with some high-frequency signals.

Configuring and Verifying your Setup

Configuring your Development Kit

The FX3 DVK has the necessary jumper settings to reconfigure the development board. (See Table 5) These jumper settings can be used as a reference for verifying your DVK and host environment. Further details of all the jumpers and switches can be looked up in the DVK schematic. The settings that are not given in Table 5 can be left disconnected.

PMODE settings configure booting options for the device. This example uses PMODE setting F11 which is USB boot. Additional details on PMODE Settings can be found in AN76405 – EZ-USB® FX3 Boot Options.

Table 5. Default Configuration

Purpose	Jumpers	Setting
Set Voltage Levels to 3.3 Volts	J134-J136 J144-J146	V3P3
Set VBATT regulator	J143	V5P0
Use USB VBUS for power	J53	Pins 1-2 Both Pairs
Setup JTAG	J52	Pins 2-3
Connect I^2C lines to P-Port	J42, J45	Pins 2-3
Enable Reset Switch	J72	Pins 1-2
Set PMODE Boot option	SW25	Off, Off, Off, Off
Set PMODE Boot option	J96, J97	Pins 2-3
Set PMODE Boot option	J98	Disconnected

Configuring your USB 3.0 Host

The next step is to setup the USB host environment with the FX3 SDK and proper drivers.

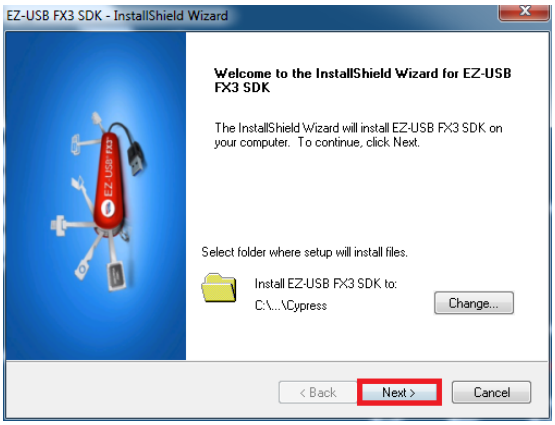
USB 3.0 Host Drivers

First check whether the USB 3.0 host has the latest drivers for your USB 3.0 host controller. Check with the USB 3.0 host controller manufacturer for verification. For best performance we recommend that you use a setup with a built-in chipset rather than an ExpressCard because an ExpressCard-based USB 3.0 host may show reduced performance.

Installing the SDK

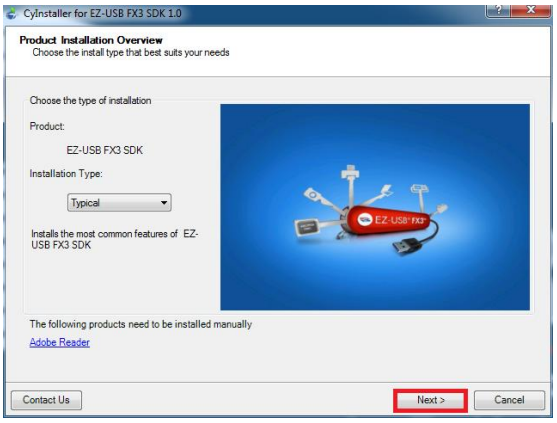
The SDK includes all the necessary software components for developing your application. This includes software tools, development environment, drivers, compiler, example code, API and API documentation. The SDK installer is included on the CD contained in your DVK. You may also download the SDK from the web. (Click here for download). If an older version of the SDK installed it should be uninstalled before installing the SDK.

1. Launch the SDK Installer.

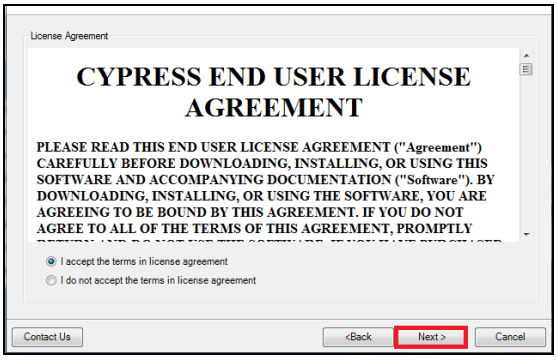


Note: Leave the default settings for the installer. The rest of the application note references these settings.

2. Choose Installation Options

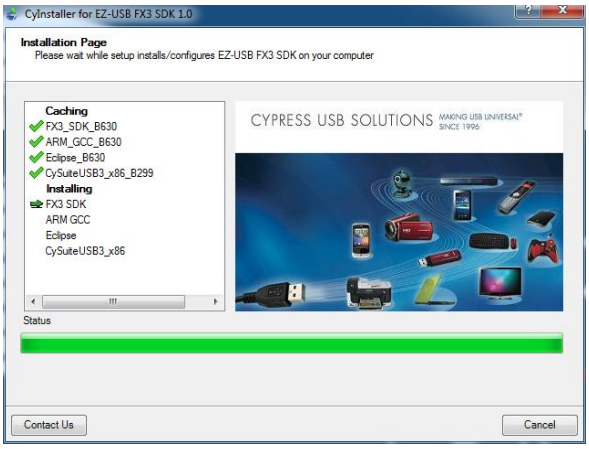


3. Accept License Agreements



Note: The SDK installer contains several software packages that require you to get license for these different software packages. The licenses are for the Cypress tools, CodeSourcery compiler, and Eclipse IDE.

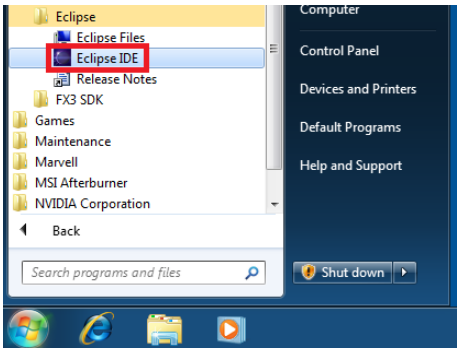
4. The following figure shows the installation of all the individual components of the SDK package.



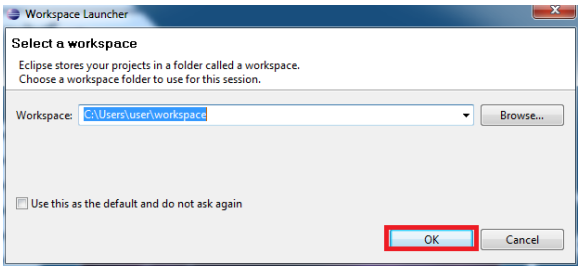
Setting Up and Using Eclipse IDE

The FX3 SDK includes Eclipse and sets it up as your IDE. Also included in the SDK are example code and API libraries. In this section, we set up the environment and compile the example projects including USBBulkLoopAuto which we later use for verification.

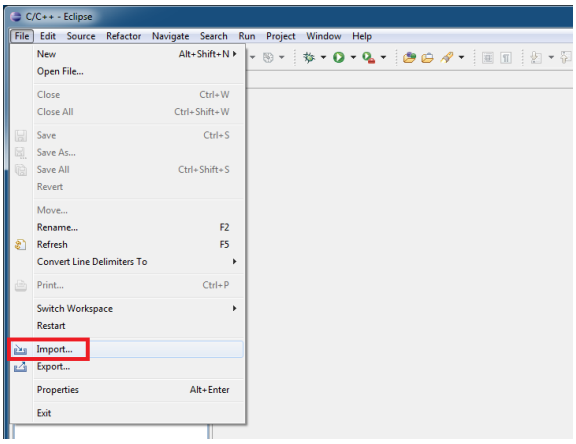
1. Launch **Eclipse IDE** from the Windows **Start** Menu **> Cypress > Eclipse > Eclipse IDE**. The Eclipse IDE appears.



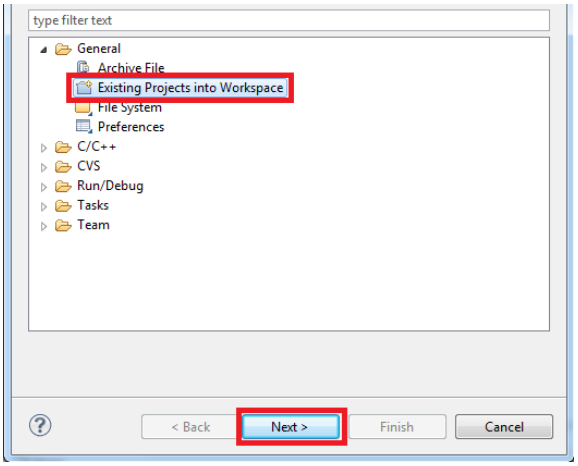
2. Use the **Browse** button to locate your workspace, and then click the **OK** button.



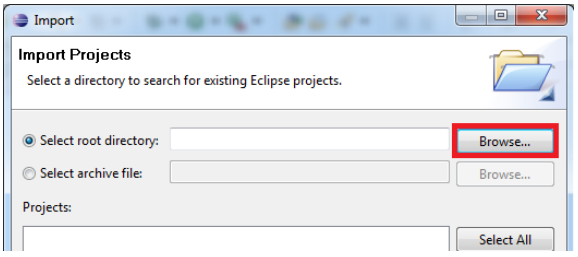
3. To import an example project, select **File > Import.**



4. Select **Existing Projects into Workspace**, and then click **Next**.

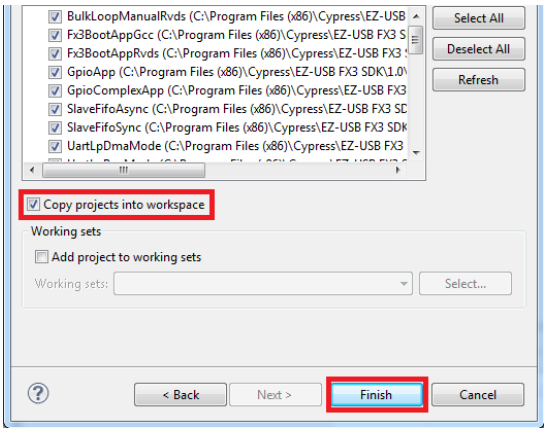


5. Click the **Browse** button, and then select the firmware directory of the SDK installation for the root directory.



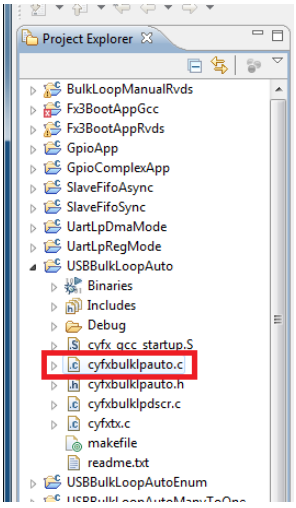
Note: This directory contains all of the Eclipse projects for the example firmware. Eclipse recognizes the directory format and imports them as projects. Default firmware directory is "`C:\Cypress\EZ-USB FX3 SDK\1.2\firmware`"

6. Select all the projects in the SDK installation directory. Verify that the **Copy projects into workspace** check box is ticked. Click **Finish**.



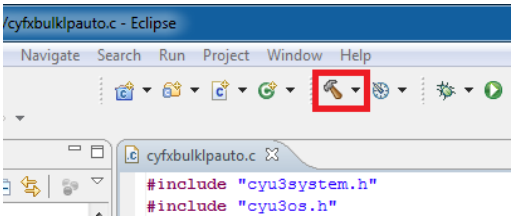
Imports all the projects into the directory and copies the project files to the Workspace.

7. Now all the example code projects have been imported. Each project contains a comment at the top of the main C file that describes what the project does. More details on these projects can be found in the firmware overview section of the "Getting Started with FX3 SDK" guide included in the SDK. The guide details the differences between example projects, gives additional information and provides use cases. For this application note we only focus on USBBulkLoopAuto.

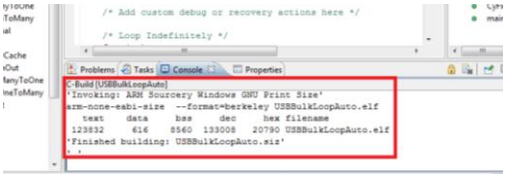


Note: We recommend that you enable the **Save Automatically Before Build** setting. From the tab **General** > **Workspace**, select **Window** > **Preferences**.

8. The last step is to build the project. From the Eclipse IDE, click the **Build** button on the toolbar.



9. The console displays the "Finished building:" message after the build completed successfully



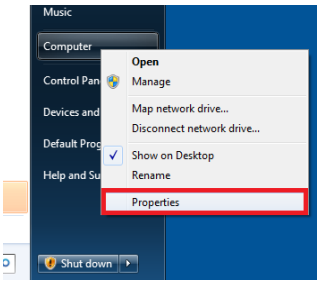
Installing Drivers

After building the example firmware you must install the necessary drivers for the FX3 bootloader. The bootloader is a firmware burned into the FX3 ROM that loads when the device first powers up. Depending on the PMODE settings the bootloader displays different functionality. In this example the bootloader enumerates over USB 2.0 and waits for a firmware image. The bootloader uses Vendor ID 0x04B4 and Product ID 0x00F3.

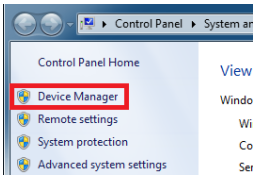
The cyusb3.sys driver allows us to communicate with the FX3 after it boots. With this driver we can use the Cypress Control Center to program the FX3 with our example firmware. For this procedure you must plug the FX3 DVK into your USB 3.0 port. Also make sure the device is powered on with SW9. When not powered on only the LED3 and D24 are lit. When powered on D14-D18 are lit.

Note: If you are using Windows 7 (64 bit) and your own VID/PID, you must disable Driver Signature Enforcement because the current driver is still in development and has not been signed by Microsoft. To do this, press F8 at startup and choose **Disable Driver Signature Enforcement**. Repeat this step every time the computer is booted.

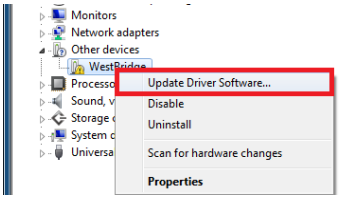
1. From the Windows desktop, click **Start**. Right-click **Computer**, and then select **Properties**.



2. Click Device Manager

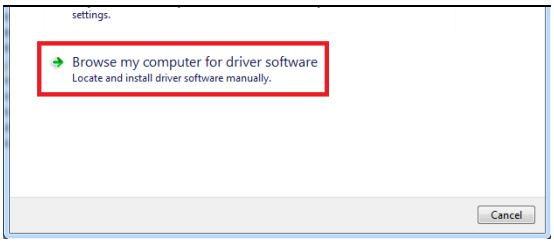


3. Expand **Other devices**, right-click **WestBridge**, and then select **Updated Driver Software**.

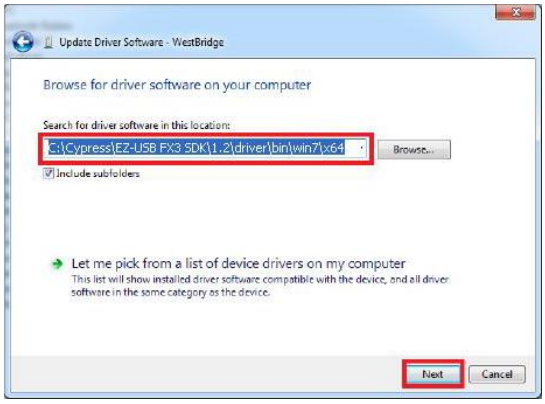


Note: When the FX3 is plugged in without a driver installed, it appears as "WestBridge" under "Other devices".

4. Click **Browse my computer for driver software**.



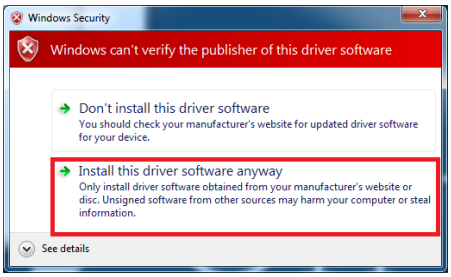
5. Click the **Browse** button. Locate the directory which contains the driver. Click **Next**.



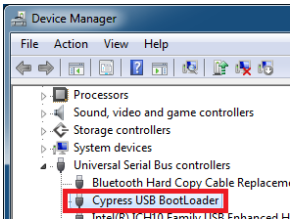
The default location is: `C:\Cypress\EZ-USB FX3 SDK\1.2\driver\bin\win7\x64`.

Note: For 32-bit systems or for Windows XP/Vista systems change the last two directories as necessary. Also if you changed the default install location of the SDK change this path accordingly.

6. Click **Install this drive software anyway**. This confirms the installation of an unsigned driver.



7. After successful installation **Cypress USB Bootloader** appears under the Universal Serial Bus controllers menu.



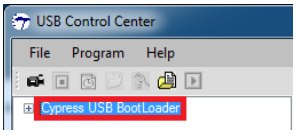
Verifying Your Setup

After setting up your host and host environment you can now run a test of the entire setup. The USBBulkLoopAuto demonstration project sends data from the USB OUT endpoint to the FX3. The data is written to a buffer that is then sent automatically out on the IN endpoint back to the host.

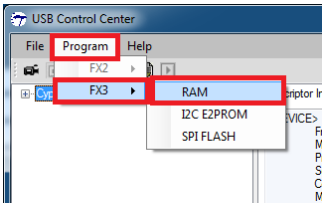
Loading USBBulkLoopAuto Firmware

We use the Cypress Control Center to load the example firmware onto the FX3 DVK.

1. To open Cypress Control Center, select **Start > Cypress > Cypress SuperSpeed USBSuite.**
2. To program the device, select the Cypress USB Bootloader



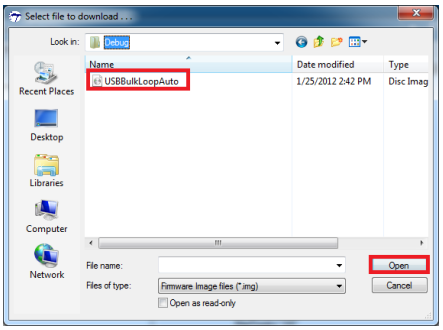
3. To load the firmware in the FX3 device, select **Program > FX3 > RAM**.



Note: Other options allow you to program the I^2C EEPROM or SPI FLASH for other booting options.

Details on these boot options can be found in AN76405 – EZ-USB® FX3 Boot Options.

4. Select the firmware to load. Firmware generated with the SDK and Eclipse ends in the IMG file extension. Click **Open**.



Note: By default the firmware can be found in the Eclipse workspace directory beneath `USBBulkLoopAuto\ Debug`

After programming, the FX3 device disappears from the Control Center menu. This is because the device has reenumerated with a new Vendor ID 0x04B4 and Product ID 0x00F0. The Control Center only shows devices that bind with cyusb.sys.

Installing the Driver for Bulk Loop Firmware

The next procedure is to install the driver for the Bulk Loop firmware. The previous driver was installed for the bootloader firmware which enumerates when the device is booted. The bulk loop firmware uses the same default cyusb3 driver but since it has a different VID/PID the driver must be installed again. The process is similar.

1. You can modify the INF file to add specific VID/PID pairs. To do so, open the INF file that corresponds to your driver in a text editor. For Windows 7 64-bit open:

   ```
   C:\Cypress\EZ-USB FX3
   SDK\1.2\driver\bin\ win7\x64\cyusb3.inf
   ```

2. Use the comments on lines 32, 38, and 45 to add additional VID/PID pairs. For the example project, add to all three locations the following:

   ```
   %VID_04B4&PID_00F0.DeviceDesc%=CyUsb3,
   USB\VID_04B4&PID_00F0
   ```

3. Add a line in the strings section following the template on line 144. For this example add

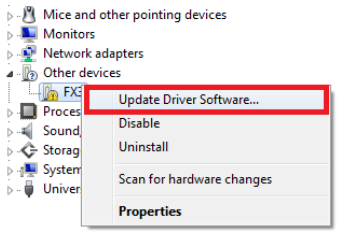
   ```
   VID_04B4&PID_00F0.DeviceDesc="FX3  Bulk
   Loop Demo"
   ```

4. To modify a firmware example and to use a different VID/PID pair simply follow the example lines and insert the new VID, PID and description.

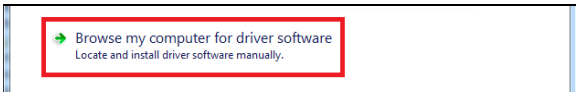
   Note: If you do not change the INF file you must force the Device Manager to use this driver since it does not recognize it as a match. It has no effect on functionality.

5. Open the **Device Manager**. From the Windows desktop, click **Start**. Right-click **Computer**, and then select **Properties**. Click **Device Manager**.

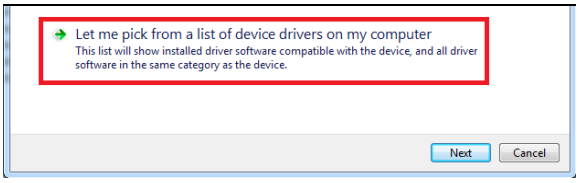
6. Expand **Other devices**, and then click **Update Driver Software**.



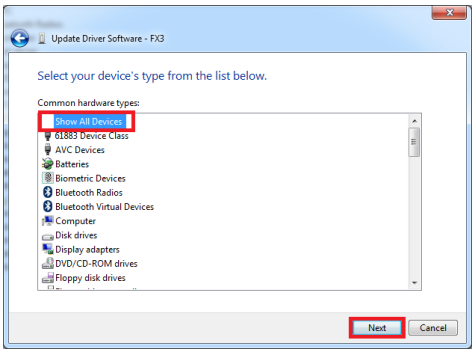
7. Click **Browse my computer for driver software**.



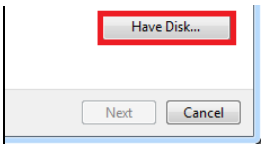
8. To pick the Driver Manually if the VID/PID do not Match. Click **Let me pick from a list of device drivers on my computer**. Click **Next**.



9. Click **Show All Devices**. Click **Next**.



10. Click the **Have Disk** button.

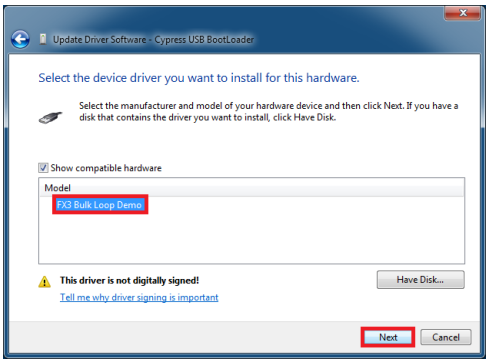


11. Browse to the cyusb3 driver. For Windows 7 64-bit computers, the default location for the cyusb3.sys driver is:

    ```
    C:\ Cypress\EZ-USB FX3
    SDK\1.2\driver\bin\win7\x64.
    ```

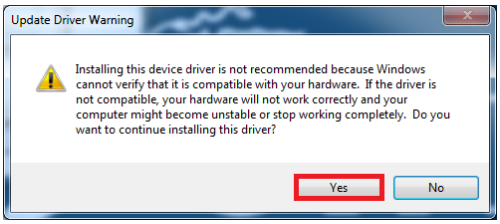
    If you have a 32-bit computer, or if you are using XP/Vista modify the path accordingly.

12. To select a USB bootloader, click F**X3 Bulk Loop Demo**, and then click **Next**.



Note: This selection does not affect the functionality but only affects the name of the device in the device manager.

13. Confirm the Correct Driver



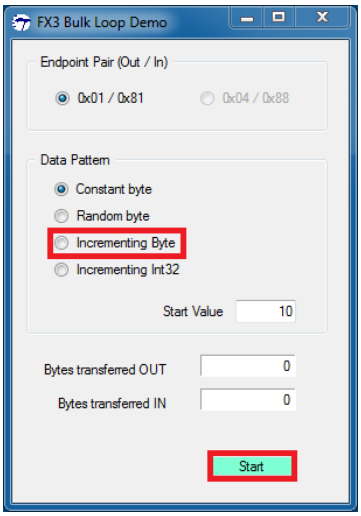
Verifying with Bulk Loop Tool

With the firmware properly loaded and the driver properly installed, we can now run the bulk loop software tool included in the SDK.

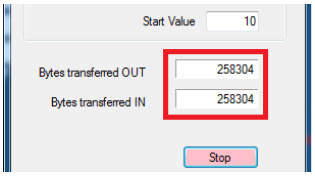
The bulk loop tool lets you to test your firmware image and USB 3.0 connection, by sending data on an OUT endpoint and then waiting to receive it on an IN endpoint. When paired with a bulk loop back firmware that echoes back the data from the OUT endpoint onto the IN endpoint, the bulk loop tool sends and receives the data continuously to the FX3.

Since the bulk loop tool requires a specific firmware, it will only work if it detects a specific VID/PID. In this case, it looks for 0x04B4 and 0x00F0. If no such devices are found it reports no device found.

1. From the Windows desktop, select **Start** > **Cypress** > **Cypress SuperSpeed USBSuite** > **Bulk Loop**.
2. To set bulk Loop tool, in the Data Pattern section, select Incrementing Byte.



3. Click the **Start** button. The following image is of a successful demonstration project.



FX3 SDK and Software for Linux

EZ-USB FX3 SDK for Linux

The FX3 SDK version 1.2 supports firmware development with Eclipse IDE, and debug using the J-Link JTAG debugger probe on a Linux platform.

The EZ-USB FX3 SDK for Linux is released in the form of a gzipped tar archive called FX3_SDK.tar.gz. On extraction, this tar archive contains the following gzipped tar archives:

1. **FX3_Firmware.tar.gz**: The FX3 firmware library and examples.
2. **ARM_GCC.tgz**: Sourcery ARM GNU toolchain.
3. **eclipse_x86.tgz**: Eclipse IDE for 32-bit Linux OS installations.
4. **eclipse_x64.tgz**: Eclipse IDE for 64-bit Linux OS installations.
5. **cyusb_linux_1.0.2.tar.gz**: The CyUSB Suite for Linux software.

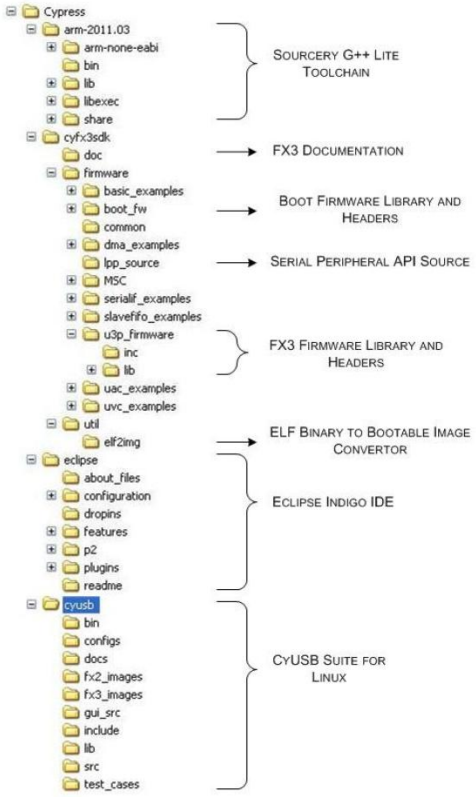
The installation procedure involves extraction of these archives and the setting of a couple of environment variables.

The installation procedure is as follows:

1. Extract the contents of the FX3_SDK.tar.gz archive at a preferred location, say, $HOME/Cypress.

2. Change to the install location ($HOME/Cypress) and extract the contents of the FX3_Firmware.tar.gz, ARM_GCC.tgz and eclipse_x86.tgz (or eclipse_x64.tgz) files.

   This creates a directory structure shown as follows:



Note: Select the appropriate archive based on the OS type (32-bit or 64-bit).

1. Add the folder containing the ARM GNU toolchain binaries to the PATH environment variables.
   For example: `export PATH=$PATH:$HOME/Cypress/arm-2011.03/bin`
2. Create an environment variable called FX3_INSTALL_PATH that points to the directory where the FX3 firmware package has been extracted.
   For example: `export FX3_INSTALL_PATH=$HOME/Cypress/cyfx3sdk`
3. Change to the Cypress `/cyfx3sdk/util/elf2img` folder and compile the elf2img program that converts ELF firmware binaries into the .img binaries that can be used to boot the FX3 device.
   For example:
   ```
   cd $FX3_INSTALL_PATH/util/elf2img
   gcc elf2img.c –o elf2img –Wall
   ```

CyUSB Suite for Linux

The CyUSB Suite for Linux software enables you to download firmware images onto FX3 devices, and test the various interfaces on the device.

CyUSB Suite for Linux Installation

The software CyUSB Suite for Linux is available as a tarball as follows:

cyusb_linux_1.0.2.tar.gz; where the 1.0.2 refers to the version number of the software

Copy the tar file to any of the local folder and extract it as shown below

```
[user@desktop /home/Cypress]$ gunzip cyusb_linux_1.0.2.tar.gz
[user@desktop /home/Cypress]$ tar xvf cyusb_linux_1.0.2.tar
```

The above creates a directory structure as follows:

The ***bin*** directory contains the main executable program called cyusb_linux.

The ***configs*** directory contains 3 configuration files as under:

- **cyusb.conf** -> Contains VID and PID of list of Cypress devices, which are of interest. User can add edit the file and add new VID PID on to the list.
- **88-cyusb.rules**-> Udev rules written for cypress devices
- cyusb**_renumerate.sh**-> Udev script that runs when ever Cypress device is connected to Host machine.

The ***docs*** subdirectory contains the documentation in PDF format

The ***fx3_images*** subdirectory contains some standard frmware images for the FX3 devkit

The ***gui_src*** subdirectory contains the source code for this software

The ***include*** subdirectory contains the header files controlcenter.h and cyusb.h

The ***lib*** subdirectory contains the library file libcyusb.so. This would need to be accessible for the main program to work.

The ***test_cases*** directory contains a simple program to generate test case data, if required.

CyUSB Suite for Linux Features

The CyUSB Suite for Linux has been developed and tested on the following environment:

- Linux Kernel - 3.4.2 (stock kernel from http://www.kernel.org)
- Fedora - 14 distribution, Ubuntu 10.04, Ubuntu 11.04, Ubuntu 12.04
- Qt version 4.7
- libusb-1.0.9

The CyUSB Suite for Linux software allows you to do the following:

1. View the device, configuration, interface, alternate-interface and endpoint descriptors of attached devices;
2. Select a specific interface+alt interface to communicate with;
3. Program the device (download firmware) to the FX3 device > download into RAM, I^2C based EEPROM or SPI based flash
4. Test your own commands (Vendor Extensions) after downloading a specific firmware that implements your command(s);
5. Test the Bulk OUT endpoint(s) by sending either constant or random or incrementing data patterns and also testing the Bulk IN endpoints by looping back data after downloading the relevant images.
6. Test the Isoc OUT and IN endpoints and ascertaining the data transfer rates.

CyUSB Suite for Linux Documentation

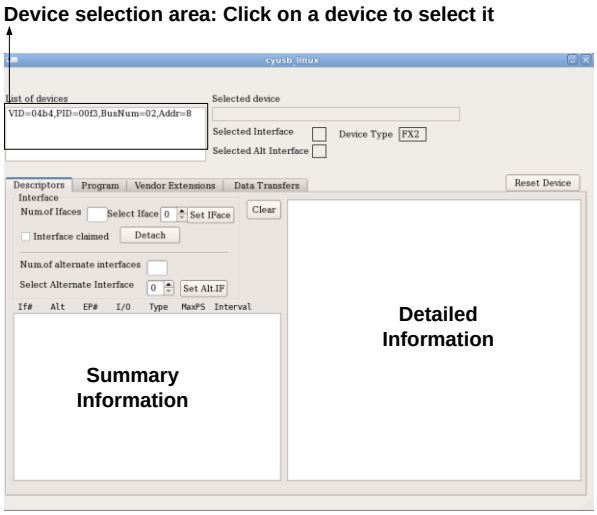
There are two companion documents available for Linux:

1. The CyUSB Suite for Linux User Guide, which describes how to install the CyUSB Suite for Linux software, and how to use the software to download firmware to FX3, to test Vendor Extensions, Bulk OUT/IN transfers and Isoc OUT/IN transfers.
2. The CyUSB Suite for Linux - Programmers Reference Manual, which describes the cyusb library for Linux and how to build and integrate user-written applications with the library.

Starting the application

After installing the software, open a terminal and launch the application by typing “cyusb_linux”. For example: `[user@desktop /home]$ cyusb_linux.`

Figure 8. CyUSB Suite for Linux



This application enables you to view the device details, program firmware, test vendor extensions, and perform bulk and isochronous data transfers to and from the device. For details on how to use this application, refer to the CyUSB Suite for Linux User Guide.

Troubleshooting Steps

In the event that any of the above steps do not work as expected refer to Table 6 for troubleshooting steps and solutions.

Table 6. Possible Issues and Solutions

Problem	Solutions
Bootloader does not enumerate	Check PMODE boot settings are set to F11
	Check that your USB 3.0 Host driver is updated to the latest version
	Make sure DVK is powered. Check VBUS power settings(J53) and power switch (SW9)
Device can’t start (code 10)	Check that your USB 3.0 Host driver is updated to the latest version
Cannot install cyusb3 driver	Disable Driver Signature Enforcement with F8 at Windows boot
	Make sure you are using correct cyusb3.sys driver for your OS.
No device in Control Center	Check device manager for FX3 device
	Hit reset switch (SW8)
Bulk Loop “no device” error	Check to see if driver is installed for Bulk Loop
	Reset Device (SW8) and reload firmware
	Check that correct firmware is used with correct VID/PID of 0x04B4 and 0x00F0.

Available Collaterals

Datasheet

Before you start to design with FX3, download the EZUSB-FX3 CYUSB3014 Datasheet from the Cypress website.

Application Notes

To help you design with FX3, Cypress has several related application notes. Additional FX3 application notes can be found here.

AN75779 – Interfacing an Image Sensor to EZ-USB® FX3 in a USB video class (UVC) Framework
This document describes how to create a video class device with FX3. It describes both the firmware and hardware requirements for such a device.

AN70707 – EZ-USB® FX3 Hardware Design Guidelines
This document helps with schematic design. It includes a checklist to ensure your hardware design could make FX3 work. In addition, a layout guide tells you how to design USB 3.0 trace from the FX3 to the connector.

AN65974 – Designing with the EZ-USB® FX3 Slave FIFO Interface
This application note describes why you need to choose a Slave FIFO interface with FX3 in your application. The details about hardware pin mapping with a Slave FIFO interface, the timing sequence in each Slave FIFO mode and details of Slave FIFO configuration are also provided in this application note.

AN76405 – EZ-USB® FX3 Boot Options
This application note details the various boot options of FX3. Included are the necessary considerations for each option. This includes USB, I^2C, SPI, and Synchronous ADMux. This application note helps with the selection of the best boot option for your design.

Cypress Support

Cypress provides an efficient tool on its website to enable quick responses to technical questions. First, register on the Cypress website and obtain credentials for the MyCases feature. Then, you can log into submit technical questions using the **Create Case** button.

About the Author

Name:	Sonia Gandhi
Title:	Application Engineer Sr. Staff
Contact:	osg@cypress.com

Document History

Document Title: Getting Started with EZ-USB® FX3™

Document Number: 001-75705

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3427934	ATAM	02/23/2012	New Application Note.
*A	3715875	ATAM/OSG	08/17/2012	Merged with AN75432 USB 3.0 EZ-USB® FX3™ Orientation Update Application Note for FX3 SDK 1.2

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com