Implementing USB 1.0 on the fido1100 Microcontroller

An Innovasic Semiconductor Application Note
fido1100 Application Note 100

Version 1.1
December 2006

Innovasic Semiconductor

Table of Contents

Abstract

This application note discusses how the fido1100 can be used in conjunction with Universal Serial Bus (USB) Version 1.0. or Version 1.1 for industrial communications. The purpose of this application note is to help guide board developers and designers who wish to implement USB in designs that also use the fido1100.

Introduction

Universal Serial Bus is currently the ubiquitous standard for interfacing between personal computers and peripheral devices. In the industrial automation space, there is a similar need for a host to be able to integrate and control several external peripheral devices.

Universal Serial Bus (USB) offers several benefits such as low cost, expandability, auto-configuration, and hot-plugging. It also provides power to the bus, enabling many peripherals to operate without the added need for an AC power adapter. Since USB is a cross-platform standard, compliant third-party devices and peripherals are compatible with compliant computers from different manufacturers, differing only in the software required for a specific operating system.

USB 1.1 can operate at 1.5 Megabits per second (Mbps), or 12 Mbps, or both. Typical USB 1.x devices include keyboards, mice, joysticks, game pads, and other low-bandwidth, low-cost devices. While these devices are low bandwidth that could be connected directly to a microcontroller, they are not independently powered. This implies that the USB host provides the power for these devices.

USB Tutorial

USB Protocols

Unlike RS-232 and similar serial interfaces where the format of data being sent is not defined, USB is made up of several layers of protocols. While this sounds complicated, don't give up now. Once you understand what is going on, you really only have to worry about the higher level layers. In fact most USB controller I.C.s will take care of the lower layer, thus making it almost invisible to the end designer.

Each USB transaction consists of a

- Token Packet (Header defining what it expects to follow), an
- Optional Data Packet, (Containing the payload) and a

- Status Packet (Used to acknowledge transactions and to provide a means of error correction)

As we have already discussed, USB is a host centric bus. The host initiates all transactions. The first packet, also called a token is generated by the host to describe what is to follow and whether the data transaction will be a read or write and what the device's address and designated endpoint is. The next packet is generally a data packet carrying the payload and is followed by an handshaking packet, reporting if the data or token was received successfully, or if the endpoint is stalled or not available to accept data.

Common USB Packet Fields

Data on the Universal Serial Bus is transmitted LS-Bit first. USB packets consist of the following fields:

SYNC – Sync field

PID – Packet ID

ADDR – Address of packet destination

ENDP – Endpoint field

CRC – 16-bit cyclic redundancy check

EOP – End of packet

•Sync

All packets must start with a sync field. The sync field is 8 bits long at low and full speed or 32 bits long for high speed and is used to synchronize the clock of the receiver with that of the transmitter. The last two bits indicate where the PID fields starts.

•PID

PID stands for Packet ID. This field is used to identify the type of packet that is being sent. The following table shows the possible values.

Group	PID Value	Packet Identifier
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1

	0111	DATA2
	1111	MDATA
Handshake	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No Response Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

There are 4 bits to the PID, however to insure it is received correctly, the 4 bits are complemented and repeated, making an 8 bit PID in total. The resulting format is shown below.

PID_0	PID_1	PID_2	PID_3	nPID 0	nPID 1	nPID 2	nPID 3

- **ADDR**

  The address field specifies which device the packet is designated for. Being 7 bits in length allows for 127 devices to be supported. Address 0 is not valid, as any device which is not yet assigned an address must respond to packets sent to address zero.

- **ENDP**

  The endpoint field is made up of 4 bits, allowing 16 possible endpoints. Low speed devices, however can only have 2 additional endpoints on top of the default pipe. (4 endpoints max)

- **CRC**

  Cyclic Redundancy Checks are performed on the data within the packet payload. All token packets have a 5 bit CRC while data packets have a 16 bit CRC.

- **EOP**

End of packet. Signal led by a Single Ended Zero (SE0) for approximately 2 bit times followed by a J for 1 bit time.

USB Packet Types

USB has four different packet types. Token packets indicate the type of transaction to follow, data packets contain the payload, handshake packets are used for acknowledging data or reporting errors and start of frame packets indicate the start of a new frame.

- **Token Packets**

  There are three types of token packets,

  - **In** - Informs the USB device that the host wishes to read information.
  - **Out** - Informs the USB device that the host wishes to send information.
  - **Setup** - Used to begin control transfers.

  Token Packets must conform to the following format,

Sync	PID	ADDR	ENDP	CRC5	EOP

- **Data Packets**

  There are two types of data packets each capable of transmitting up to 1024 bytes of data.

  - Data0
  - Data1

  High Speed mode defines another two data PIDs, DATA2 and MDATA.

  Data packets have the following format,

Sync	PID	Data	CRC16	EOP

  - Maximum data payload size for low-speed devices is 8 bytes.
  - Maximum data payload size for full-speed devices is 1023 bytes.
  - Maximum data payload size for high-speed devices is 1024 bytes.
  - Data must be sent in multiples of bytes.

- **Handshake Packets**

  There are three type of handshake packets which consist simply of the PID

  - **ACK** - Acknowledgment that the packet has been successfully received.
  - **NAK** - Reports that the device temporary cannot send or received data. Also used during interrupt transactions to inform the host there is no data to send.
  - **STALL** - The device finds its in a state that it requires intervention from the host.

Handshake Packets have the following format,

Sync	PID	EOP

- **Start of Frame Packets**

The SOF packet consisting of an 11-bit frame number is sent by the host every 1ms ± 500ns on a full speed bus or every 125 µs ± 0.0625 µs on a high speed bus.

Sync	PID	Frame Number	CRC5	EOP

USB Functions

When we think of a USB device, we think of a USB peripheral, but a USB device could mean a USB transceiver device used at the host or peripheral, a USB Hub or Host Controller IC device, or a USB peripheral device. The standard therefore makes references to USB functions which can be seen as USB devices which provide a capability or function such as a Printer, Zip Drive, Scanner, Modem or other peripheral.

Approach

In general, interfacing USB to the **fido1100** is accomplished by connecting a "USB to Parallel PHY" chip to one to one of the **fido1100**'s Universal I/O Controller (UIC) interfaces. The UIC is then configured to act as a standard parallel interface. Each **fido1100** microcontroller contains four Universal I/O Controllers which may be configured according to the user's specifications. The UIC is a very flexible hardware solution designed to support numerous interface requirements. In this solution the UIC will use two of its three banks of digital I/O. Each UIC has three banks of digital I/O: Bank A 8-bits, Bank B 8-bits, and Bank C 2-bits.

Using a USB to Parallel PHY chip to act as an intermediary between the **fido1100** UIC and the actual Universal Serial Bus allows the **fido1100**, operating a only 60MHz, to communicate at the higher transmission rates of USB1.x (typically 100Mbps). This is accomplished by accumulating the single-bit communications of USB to an 8-bit data bus operating at 48MHz supplied by the USB to Parallel PHY chip. Additionally, using an external controller protects the **fido1100** against potential power spikes that may occur while connecting and disconnecting USB devices to and from the **fido1100**.

Interfacing a UIC to a USB to parallel PHY chip varies according to the particular vendor and model of controller however, the general principal is the same across all PHY devices. One 8-bit bank of the UIC is used for the 48MHz data bus while another 8-bit bank is used for the control lines between the UIC and the USB to parallel PHY chip. In this particular solution we will discuss the use of the FT245R USB FIFO from Future Technology Devices Ltd.

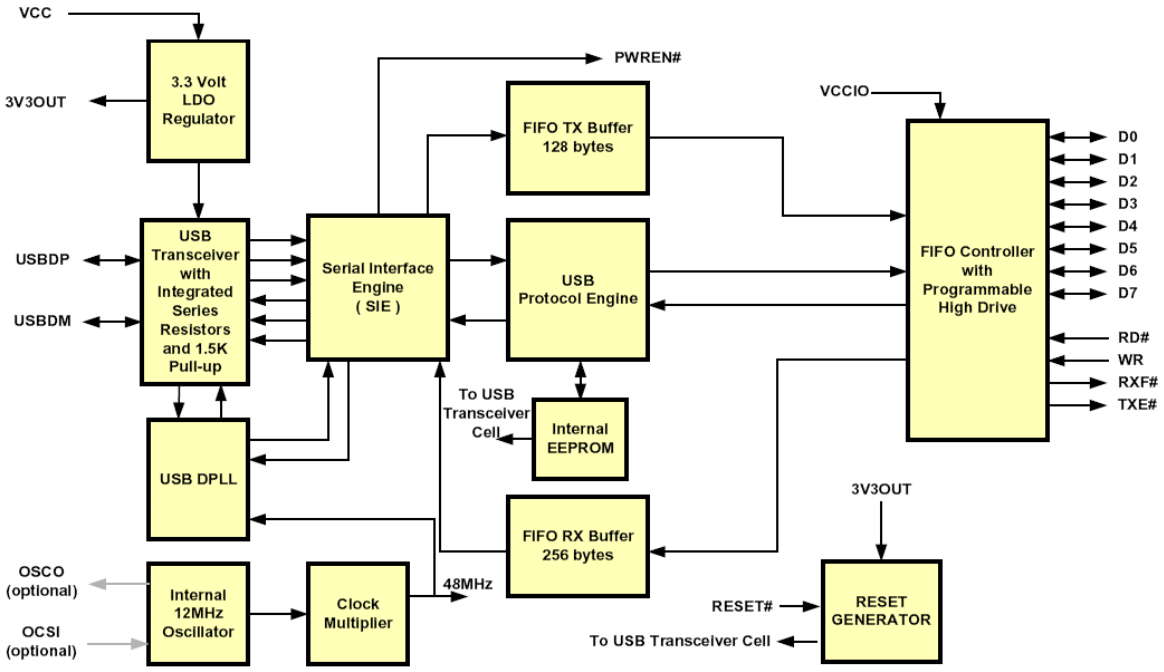

Illustration 1: FT245 USB FIFO Simple Block Diagram

Connecting the USB FIFO to the fido1100

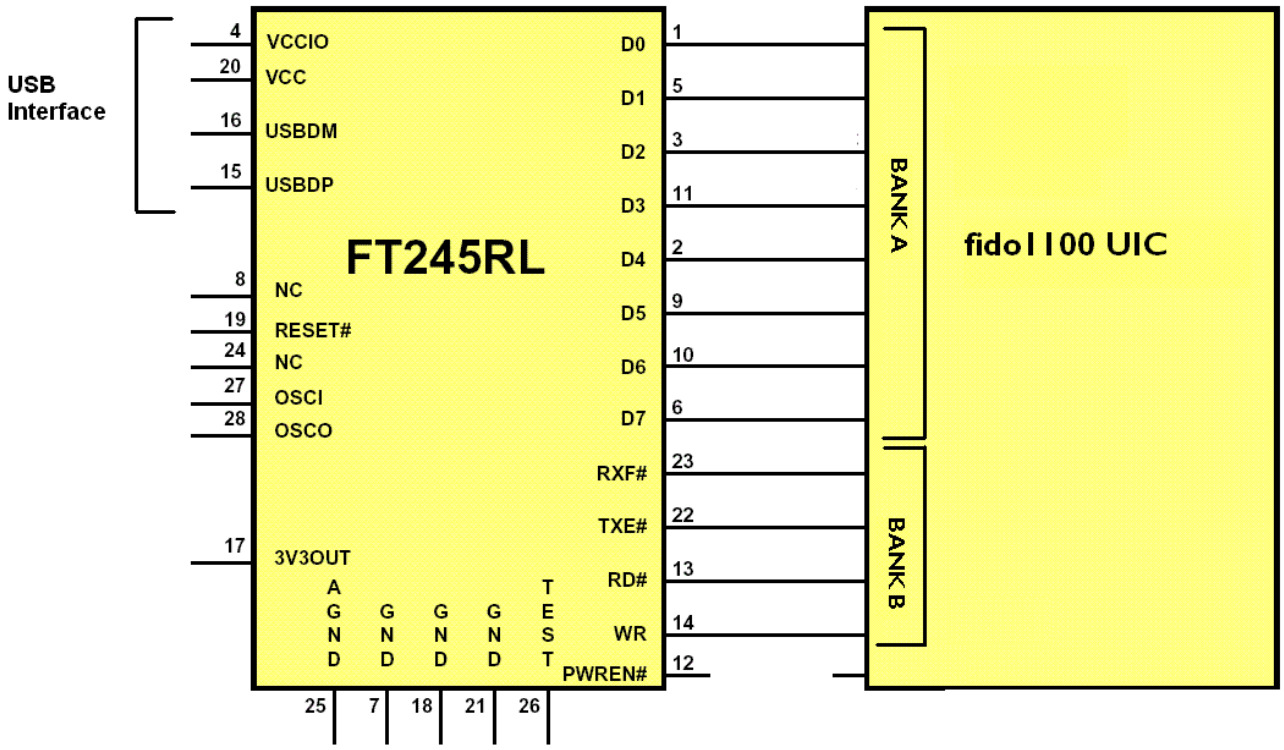

Illustration 2: Connecting the fido1100 UIC to the FT245RL USB to Parallel PHY

Notice that in Illustration 2 pin 12 of the FT245RL is not connected to the **fido1100** UIC. This pin is used for different modes for powering attached USB devices. Please refer to the data sheet for the FT245RL for details on USB device powering modes. This application note only discusses the Bus Power mode for powering USB devices.

Power Considerations

It is important to remember that the USB to Parallel PHY will be providing power to USB devices. It is therefore recommended that sufficient isolation be provided between the **fido1100** and the USB to Parallel PHY source power supplies as well as the final output to the USB device. This will protect the **fido1100** against
potential power spikes that may occur when USB devices are connected and disconnected.

There are two modes of supplying power to USB devices using the FT245RL Self Power and Bus Power This application note only addresses Bus Power Mode described below.

Bus Power Mode

1. On plug-in to USB, the device must draw no more than 100mA.
2. On USB Suspend the device must draw no more than 500µA.
3. A Bus Powered High Power USB Device (one that draws more than 100mA) should use the PWREN# pin to keep the current below 100mA on plug-in and 500µA on USB suspend.

4. A device that consumes more than 100mA can not be plugged into a USB Bus Powered Hub
5. No device can draw more that 500mA from the USB Bus.

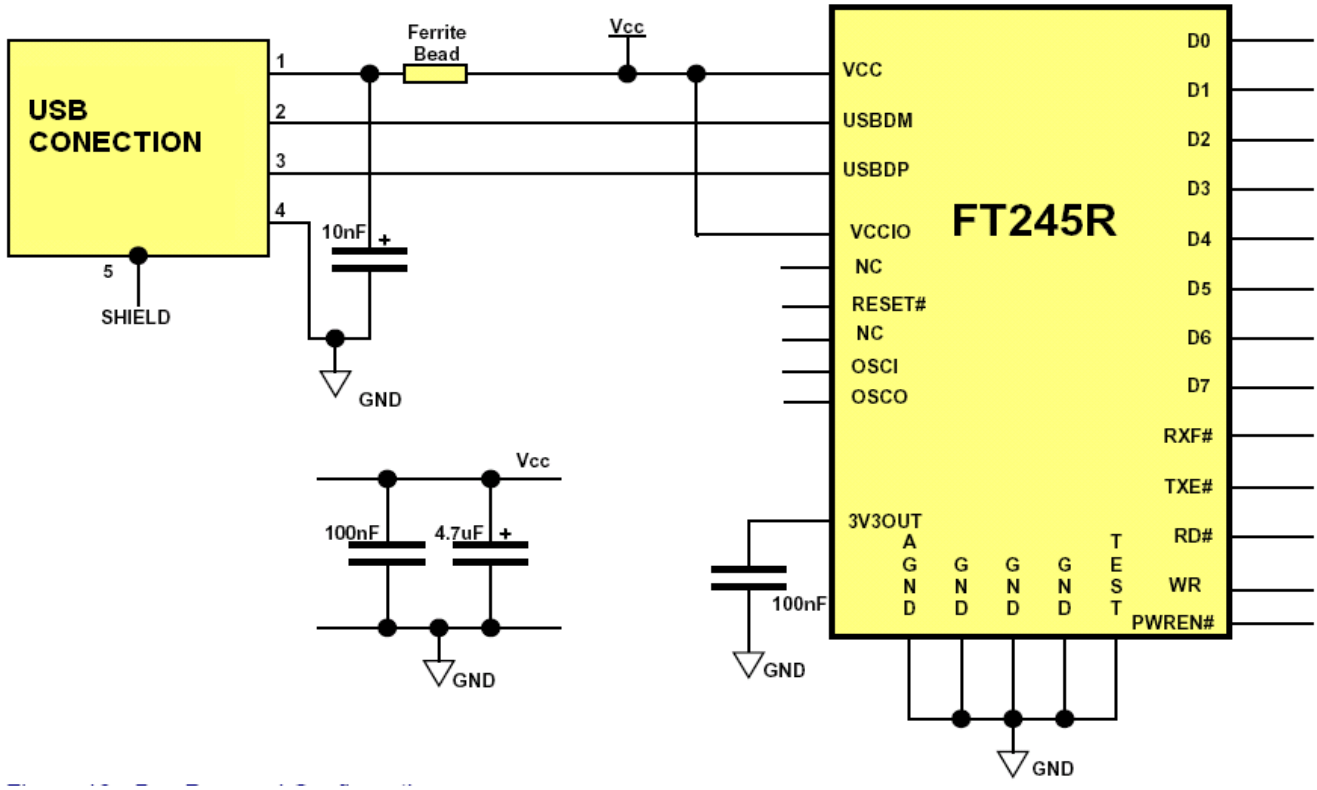

Illustration 3: Bus Power Mode for powering USB devices

When providing power to USB devices the use of a ferrite bead is recommended to help isolate the USB to parallel PHY from power spikes. A ferrite bead is a passive electric component used to suppress high frequency noise in electronic circuits. Ferrite beads employ the mechanism of high dissipation of high frequency currents in a ferrite to build extremely effective high frequency noise suppression devices. Ferrite bead is similar in construction to inductor, but working in especially that area that is parasitic for general purpose inductors.

Parallel Port Timing

Parallel Port Read Cycle

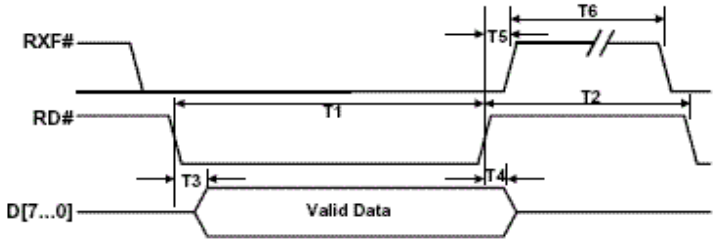

Illustration 4: Parallel Port Read Cycle

Time	Description	Min	Max	Unit
T1	RD Activate Pules Width	50		Nano Seconds
T2	RD to RD Pre-Charge Time	50 + T6		Nano Seconds
T3	RD Active to valid Data	20	50	Nano Seconds
T4	Valid Data Hold Time from RD Inactive	0		Nano Seconds
T4	RD Inactive to RXF#	0	25	Nano Seconds
T6	RXF Inactive After RD Cycle	80		Nano Seconds

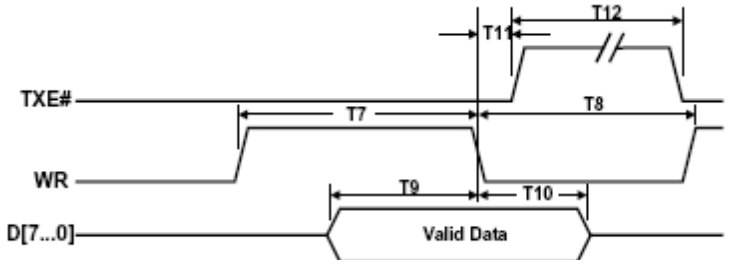

Illustration 5: Parallel Port Write Cycle

Parallel Port Write Cycle

Time	Description	Min	Max	Unit
T7	WR Activate Pules Width	50		Nano Seconds
T8	WR to RD Pre-Charge Time	50		Nano Seconds
T9	Data Setup Time Before WR Inactive	20		Nano Seconds
T10	Data Hold Time from WR Inactive	0		Nano Seconds
T11	WR Inactive to TXE#	5	25	Nano Seconds
T12	TXE Inactive After WR Cycle	80		Nano Seconds

Software and Drivers

To implement a USB interface using the **fido1100** a low-level driver is required to implement the link layer (Parallel Port Interface). Innovasic Semiconductor plans to provide a driver and a standard USB stack in the near future. Please contact Innovasic for more information on availability. In the meantime, it is possible for the customer to write a low-level driver with no additional UIC firmware required. The UIC I/O pins are directly accessible like standard GPIO pins, allowing the customer to "bit-bang" the communications interface using the fido main processor.

Additional Considerations

A variety of vendors supply USB to parallel PHY interface chips. Many factors should be taken into consideration when implementing these chips to the **fido1100** in your design. Primary factors to consider include (but are not limited to):

Total power consumption of devices
Thermal dissipation
Physical placement and positioning of the chips

Thank You

Thank you for taking the time to review this application note. We hope you have found the information included in this application useful and east to understand. Please feel free to contact the Innovasic Support Team any time with questions or comments.

Innovasic Support Team
3737 Princeton NE
Suite 130
Albuquerque, NM, 87107

(505) 883-5263

support@innovasic.com
http://www.innovasic.com