Converting from Intel 8x930Ax to Cypress EZ-USB

Introduction

In 1999, Cypress partnered with Intel in an exclusive product licensing agreement for the USB Intel product family. As a leader in USB peripheral solutions, Cypress pursued this opportunity for two reasons, to provide another option for the Intel customer base and to promote the USB technology.

This product licensing agreement consisted of two phases. The first phase was an exclusive buy/resale agreement between Cypress and Intel. As previously agreed to by both parties in the product licensing agreement, the buy/resale phase will expire in October of 2000.

The second phase is an option to manufacture these former Intel USB devices in Cypress fabs. After several months of conducting market research, the overall demand for the former N8x93x family of USB microcontrollers does not offset the costs associated with transferring and manufacturing of these devices in Cypress fabs. As a result, Cypress is officially announcing the obsolescence of these former Intel USB microcontroller devices. Cypress will accept lifetime buys on all of the listed products per the following schedule:

Pruned Cypress Part Number	Corresponding Former Intel Part Number	Package	Hub
CY7C69300AD-JC	N80930AD	68 PLCC	No
CY7C69300HF-JC	N80930HF	68 PLCC	Yes
CY7C69310AA-JC	N80931AA	68 PLCC	No
CY7C69310AA-NC	S80931AA	64 MQFP	No
CY7C69310HA-JC	N80931HA	68 PLCC	Yes
CY7C69310HA-NC	S80931HA	64 MQFP	Yes

All parts are for commercial temperature.

Cypress will accept end-of-life buys on the affected products through May 12, 2000, for delivery from now until October 12, 2000. In addition to offering a lifetime buy opportunity, Cypress will provide design conversion assistance for transitioning your design to a comparable Cypress USB microcontroller.

The Cypress family of USB microcontrollers can offer a number of benefits relative to the former Intel N8x93x family, including improved USB performance, greater flexibility, generally lower system costs and greater availability with short lead-times. In addition, many applications may benefit from a nearly seamless transfer to our equivalent 8051-based USB microcontroller family. For instance, a MJPEG reference design by Zoran was converted in four weeks with an increased performance of 40% and still achieved a 15% reduction in system cost. This white paper provides pointers on how to convert the 8x930AX design into an EZ-USB family member, along with comparisons between the two USB microcontroller families.

Intel 8x930Ax vs. EZ-USB Family Architectural Differences

The Intel 8x930Ax device is a first-generation generic USB microcontroller, merging existing MCS 51/251 controllers with standard USB SIE technology. To make it easier for peripheral manufacturers to ease into the complex world of USB, EZ-USB took the concept of the general-purpose USB microcontroller further by implementing a Smart USB Engine Core. This Smart USB Engine Core is so intelligent that it can respond to host controller commands and pass a USB Chapter 9 compliance test without the need for the microcontroller. EZ-USB handles USB protocol commands more efficiently than the earlier Intel architecture, as the majority of USB requests are handled without the involvement of the internal microcontroller.

With this architecture, USB peripheral developers can minimize their knowledge about USB and concentrate on their application firmware, while still achieving the benefits of a generic USB microcontroller. The diagrams below illustrate the efficiency of EZ-USB by comparing how the two USB microcontrollers handle the “Get Descriptor” host controller command.

Figure 1 shows how conventional USB controllers, such as the Intel 8x930Ax, handle a three-stage USB SETUP transfer called “Get Descriptor”. The serial data flowing over the USB is shown as three stages: Setup, Data, and Status. The numbered arrows indicate transfers between the USB, endpoint FIFOs, and microprocessor memory. Significant CPU overhead is required to transfer the data to and from the endpoint FIFOs (2,3,5) and to divide the descriptor table data into packets for transmission using multiple USB data packets (4,6).

Figure 2 shows how the EZ-USB handles the same “Get Descriptor” command. The EZ-USB core directly transfers setup packet data into a dedicated eight-byte setup data buffer for CPU inspection (1). Then the 8051 loads an EZ-USB pointer with the start address of the requested descriptor data (2). The EZ-USB Smart Engine Core does the rest. The Smart Engine Core automatically takes care of error checking and retries, dividing the table into packets for the various IN transfer and responding to the Status stage.

A good way to measure the efficiency of the Cypress solution is to compare the firmware sizes between EZ-USB and Intel for the same function. The following functions were downloaded from Intel website and compiled.

- USB Chapter 9 Test
- String Descriptors
- USB Suspend and Resume
- Remote Wake-up
- Bulk Endpoint Loopback

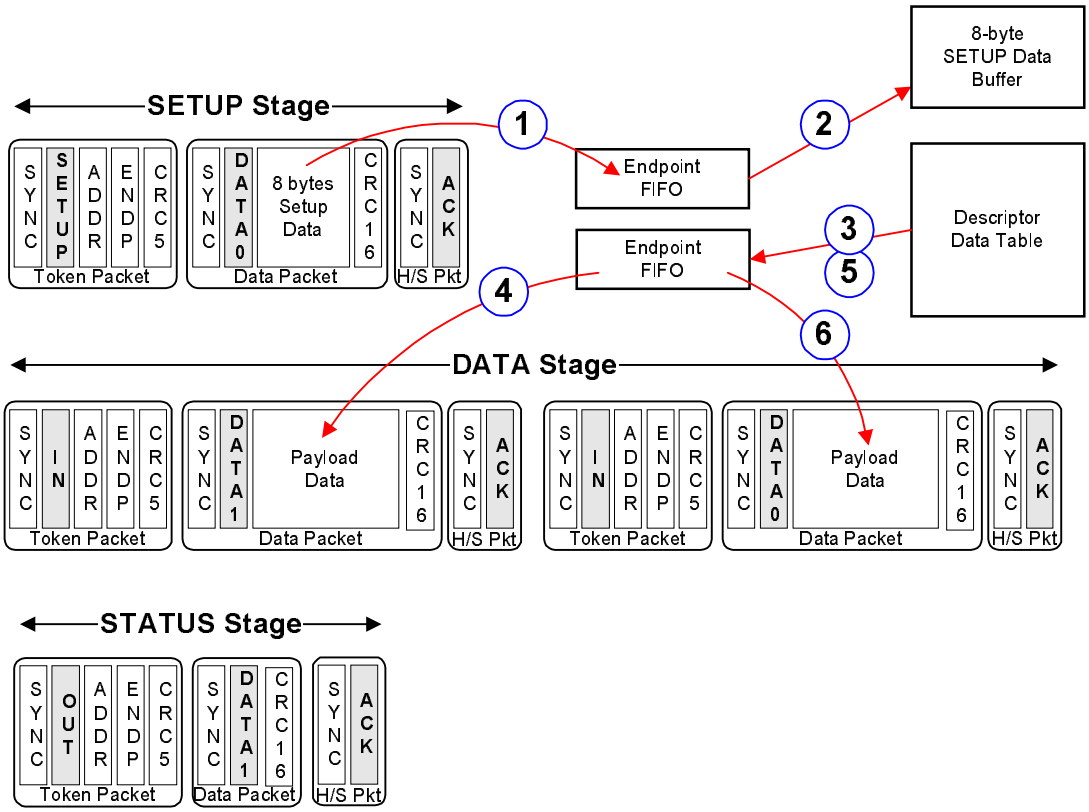

Figure 1. How the 8x930Ax Handles a "Get Descriptor" Request

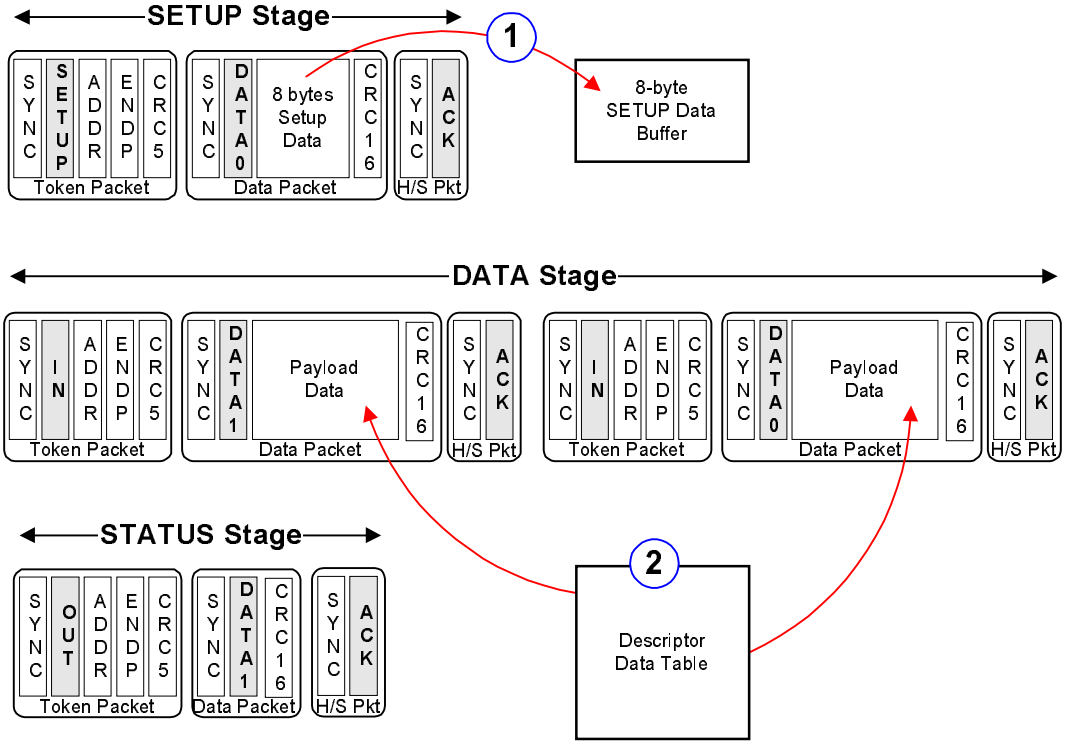

Figure 2. EZ-USB Logic Does Most of the Work for a "Get Descriptor" Request

Table 1. Compilation Results and Comparison

Firmware File Type	EZ-USB Family	Intel 8x930Ax
Source Size	730 lines of C code	5445 lines of assembly code
Binary Size	< 1K Byte	> 5K Byte

Table 1 shows the results of our compilation and a comparison of the same function implemented using EZ-USB.

It is the manner in which EZ-USB handles the USB protocols that will need to be addressed when converting from the Intel 8x930Ax to the EZ-USB family. Firmware code which deals with the peripheral function can remain the same, even if the firmware code was previously compiled for the 251 architecture. If the firmware code was written in C, then recompiling under the 8051 option instead of the 251 option should keep the effort to convert to Cypress's EZ-USB to a minimum.

EZ-USB Hardware Considerations Versus the Intel 8x930Ax

While architectural differences on how each of the USB microcontrollers handle the USB protocol must be observed, there are hardware differences which can yield a better USB hardware design. These are:

- Elimination of external ROM or EPROM
- Elimination of external components
- Higher I/O performance -> Yielding maximum USB performance
- Additional endpoints
- Support of maximum packet sizes for all endpoints
- Lower power/Easier power management
- Smaller board density
- Multiple product options

The firmware developer and hardware designer will need to make slight modifications in order to take advantage of these hardware advantages. The following is a quick summary of these differences and benefits.

1. Elimination of external ROM or EPROM

   While the Intel family includes an internal ROM option, most users have implemented the ROMless option (80390Ax) with external ROM or EPROM in order to maximize flexibility in firmware changes. While the EZ-USB family also supports external firmware memory, the unique EZ-USB ability to download firmware from the host PC eliminates the need for external ROM or EPROM. The EZ-USB firmware download feature provides firmware flexibility coupled with the cost advantage of an internal ROM.

2. Elimination of external components

   While the Intel family uses a multiplexed address and databus, the EZ-USB family has a separate address and databus, eliminating the need for an external demultiplexing latch. In addition, with only 16 PIOs of the 8x930Ax compared to 24 PIOs of the AN2131QC, USB designs which require additional I/Os may be able to eliminate additional external components.

3. Higher I/O performance -> Yielding maximum USB performance

   Intel's architecture supports standard transfer whereby the 8051 instructions initiate a data transfer through a MOVX and then increments the address. For high-performance applications, this can result in USB performance limited by the USB microcontroller as the 8051 cannot empty FIFOs fast enough to keep pace with a high data transfer rate. Some users have reported losing data every 3rd frame due to this performance limitation. The suggested solution was to reduce the packet size to 8 or 16 bytes per frame.

   However, the EZ-USB has a special data transfer mode called Turbo mode which yields significant USB performance improvement. With Turbo mode, the EZ-USB core monitors certain "MOVX" transfers, and directly broadcasts or receives their data to or from the databus, and generates read and write strobes. In the bulk Turbo mode, a third datapointer in EZ-USB auto increments the RAM address with each read or write cycle. Thus the bulk data RAM buffer acts as a fast FIFO with the ability to transfer 1 byte in 330 ns. As a result, EZ-USB can support 17 bulk packets of a 64-byte size within a single USB frame (1 ms) in a standard Windows® environment (see white paper on AN2131 bulk performance). In this illustration, performance was limited by the host controller, not the EZ-USB. The EZ-USB microcontroller never generated a NAK. This yields an equivalent data transfer rate of 8.9 Mbps.

   Similarly, in isochronous mode, 8a bytes can be transferred in or out of an ISO endpoint FIFO in 330 ns. Thus a 1024-byte packet can be transferred in or out of an isochronous endpoint FIFO in 1/3 time of a single USB frame providing leftover time for the internal 8051 to perform other functions.

   To take advantage of the EZ-USB turbo mode, firmware will need to be adjusted so that read and write operations occur on pins PA4/FWR# and PA5/FRD#.

4. Additional endpoints/Max. endpoints packet size

   The Intel 8x930Ax supports 2 endpoint configurations; 4 or 6 endpoint pairs. In either configuration, endpoint 1 is used to support a 64-byte bulk packet size or a 512-byte isochronous packet size. A 1024-byte packet is available, but does not support the USB requirement of an isochronous packet being double buffered. Endpoints 2–5 are either 16- or 32-byte packets.

   With EZ-USB, 16 endpoint pairs are available. Endpoints 1 through 7 support 64-byte bulk packet sizes. In addition, to obtain maximum performance, these endpoints can be paired, providing double buffering capability to maximize bulk performance.

   For isochronous support, the EZ-USB family provides eight endpoints, numbered 8–15. A full 2 Kbytes of isochronous FIFO is provided, so that a 1024-byte packet (double buffered) can be supported in a single USB frame.

   In converting to the EZ-USB product, bulk peripheral designs can still use endpoint 1 for primary data transfers. If the firmware is using other endpoints for control and sta-

tus, they can be expanded to 64-byte maximum packet sizes, which will improve USB throughput. For isochronous designs, endpoints will need to be converted from endpoint 1 to endpoints 8 through 15.

Feature	Cypress EZ-USB Family	Intel 8x930Ax
Endpoint Pairs	16	4/6
Max. Packet Size		
Isochronous (bytes)	1024	512/256
Bulk (bytes)	64	16/32

5. Lower power/Easier power management

   The Intel 8x930Ax has a 150-mA maximum current specification under normal operation. The USB specification requires a device to consume no more than 100 mA during the initial plug-in. This is because bus-powered hubs can only supply a maximum of 100 mA per port during bus enumeration. Following the chip reset, the 8x930Ax operates in low-clock mode, wherein the CPU and on-chip peripherals are clocked at a reduced rate until bus enumeration is complete. This reduces the I_{CC} to meet the USB 100-mA requirement. Thus the Intel 8x930Ax requires the USB microcontroller to be in idle mode operation during initial power-on.

   Remote wake-up on the 8x930Ax is performed using a register bit (RWU). Firmware must be used to drive resume signaling on the USB lines. The USB microcontroller must be awake in order for firmware to be accessed and generate the remote wake-up operation. However, this makes it difficult for remote activity peripherals such as modems and telephones, which are usually in a suspended state when not in use to save power. An external stimulus, such as a phone call, requires careful timing by the peripheral designer as it must pull the USB microcontroller out of suspend operation and write to firmware before initiating the next series of events. Normally, an external stimulus should trigger a series of events, including the resume signalling in the host PC right away.

   In addition, there is a restriction on the Intel 8x930Ax. If the 8x930Ax is put into power-down mode prior to receiving a USB Suspend signal from the host, a USB Resume will not properly wake up the 8x930Ax from the power-down mode.

   EZ-USB uses ¼ of the power of the Intel 8x930Ax, making it ideal for power-sensitive or portable peripherals. As a result, there is no idle operation mode, since EZ-USB only has a 50-mA (25-mA typical) maximum current specification under the normal operating conditions. If a peripheral requires between 500 and 600 mA, then the reduction of 100 mA by switching to EZ-USB will allow the peripheral manufacturer to eliminate the power supply and become a bus-powered peripheral, thereby saving significant costs.

   In addition, EZ-USB resume operation can be performed with the toggle of an external pin (WAKEUP#), thus making it easy for hardware to implement the wake-up operation. No timing considerations due to waking the USB microcontroller are necessary.

   Converting to EZ-USB requires a 3.3V regulator, as EZ-USB is powered under 3.3V as opposed to the Intel 8x930Ax which can run off Vbus.

Feature	Cypress EZ-USB Family	Intel 8x930Ax
Voltage (volts)	3.0–3.6	4.0–5.25
Max. Power (Active)	180 mW	788 mW
Max. Current (Active)	50 mA	150 mA

6. Smaller board density

   The Intel 8x930Ax is housed in a 68 PLCC, which has an approximate body size of 24 x 24. The EZ-USB family is packaged in both the 44 PQFP (10 x 10 mm) and 80 PQFP (14 x 10 mm). The 44 PQFP is useful for applications not needing the address bus. As a result, a 44 PQFP can use ¼ the board space of the Intel solution.

Feature	Cypress EZ-USB Family	Intel 8x930Ax
Package	44 PQFP 80 PQFP	68 PLCC
Package Board Density	170 sq. mm 432 sq. mm	625 sq. mm

7. More product options

   While the Intel 8x930Ax offers only one product option which supports both isochronous and bulk, the EZ-USB family has 10 product options. All the options differ in internal RAM size, I/O performance, and the support of bulk only versus isochronous support (see *Table 2*).

Additional Hardware Issues To Consider When Converting

8. VID/PID/DID Implementation

   With the Intel 8x930Ax, VID/PID/DID information was embodied into the internal ROM or external ROM/EPROM.

   If the peripheral designer is implementing the firmware download feature of EZ-USB, then the VID/PID/DID information is housed in a tiny 16-byte (or larger) EEPROM which is connected through the EZ-USB I^2C port. Should the peripheral designer use an 8K EEPROM to load the entire firmware through the I^2C port, then the VID/PID/DID information is contained in the 8K EEPROM. Of course, if the peripheral designer requires more than 8K of firmware and uses external memory for firmware, the VID/PID/DID information is embodied in external memory as with the 8x930Ax.

9. 1.5-kΩ Pull-up Resistor

   The 8x930Ax uses a standard implementation for USB, attaching a 1.5-kΩ resistor from D+ to Vbus. However, EZ-USB performs special tricks with the USB signal lines in performing the ReNumeration operation. As a result, the 1.5-kΩ resistor is connected directly to our DISCON# pin. Even if ReNumeration operation is not required, the designer should still connect the 1.5-kΩ resistor to the DISCON# pin.

Table 2. Product Options

Part Number	Package	RAM Size	I/O Rate (Bytes/s)	# of Prog I/Os	8-bit Databus?	ISO Support?
AN2121SC	44 PQFP	4K	600K	16	No	Yes
AN2122SC	44 PQFP	4K	600K	16	No	No
AN2122TC	48 TQFP	4K	600K	19	No	No
AN2125SC	44 PQFP	4K	2M	8	Yes	Yes
AN2126SC	44 PQFP	4K	2M	8	Yes	No
AN2126TC	48 TQFP	4K	2M	11	Yes	No
AN2131SC	44 PQFP	8K	600K	16	No	Yes
AN2135SC	44 PQFP	8K	2M	8	Yes	Yes
AN2136SC	44 PQFP	8K	2M	8	Yes	No
AN2131QC	80 PQFP	8K	2M	24	Yes+Addr	Yes

10. Passive Networks

    Both the 8x930Ax and EZ-USB have active-HIGH RESET pins, but while the 8x930Ax requires only a capacitor to V_{CC}, the EZ-USB family parts require a resistor-capacitor network.

11. Connect any programmable I/Os that were used to generate I^2C signals directly to the EZ-USB I^2C port.

    If a peripheral required an I^2C bus master, the 8x930Ax would require the use of a programmable I/O along with the appropriate firmware to emulate the I^2C bus. EZ-USB has a dedicated I^2C port that is a master and can be used to control other I^2C devices on the peripheral board.

Additional Firmware Issues To Consider When Converting

1. Compile as 8051 (not 251)

   Since the EZ-USB core operates in the 8051 mode only, you must recompile your application in only this configuration. The ‘251 mode uses different instructions and formats and will not work. Most compilers for the 8051 family have compile switches that make this translation transparent.

2. Add EZ-USB frameworks code

   - Handles endpoint 0 transfers (getdesc…)
   - Provides hooks to USB events
   - Performs ReNumeration

   All of the firmware that is required to handle default device requests, set up the processor environment, perform ReNumeration, and set up hooks to all of the USB events is included in the Application Frameworks which is included in the EZ-USB Developer’s Kit. Although much of the endpoint 0 traffic is handled by the EZ-USB Smart core, the 8051 still has USB overhead to handle. The frameworks code serves as a good example of how to handle the remaining processing necessary by the 8051.

3. Remove control transfer code

   Since the EZ-USB Smart core automatically handles much of the Endpoint 0 Control transfer processing, a good portion of the firmware required by the 930 can be removed from the code after conversion.

4. Use Turbo transfer mode when USB performance is desired

   To utilize the full bandwidth of the USB and still achieve the maximum processing power, all data transfers for high-bandwidth data should use the Turbo transfer mode. The firmware required to perform these transfers is very simple and is written in assembly to utilize the full speed of this mode. These transfers will replace standard XDATA accesses in the 930 firmware.

5. Remove 251 configuration setup

   Since there are no configuration features that require set-up for the 8051, the configuration setup code for the ‘251 can be removed.

6. Endpoint data is in RAM, not FIFOs

   The 930 provides all of the bulk endpoint data in FIFOs, making the processor read each byte out in series to process a block of endpoint data. The EZ-USB architecture places its data in RAM, which can be accessed in any order necessary. This process can greatly reduce the processing overhead, especially when processing control transfers.

7. EZ-USB registers are in XDATA space, not SFRs

   All of the registers in the EZ-USB chip are contained in the XDATA memory space. This requires MOVX instructions to access them.

8. I/O Ports are in XDATA space

   The Intel 8x930Ax family uses SFR (Special Function Register) bits to control their I/O port pins. The EZ-USB family memory maps I/O port control registers, accessible with MOVX instructions. Thus bit set and clear instructions cannot be used with the EZ-USB I/O Ports.

Case Example– Zoran Video Inlet Reference Design

Zoran manufacturers a Motion JPEG device, ZR36060, performing on-the-fly hardware compression and yielding 30 fps performance for video captures at CIF resolutions over USB. With the advent of supporting JPEG compression, end users can still capture video frame and maintain sharp images during the capture. The Video Inlet reference design offers the ability for the consumer to use a standard consumer video-

cam for video capture over the USB bus, while offering high-quality video motion and video capture performance.

The following schematic (Exhibit A) includes only the USB portion of the reference design. With added notes, this example illustrates the straightforward method of converting an Intel 8x930Ax design to the EZ-USB family.

In performing this conversion, the following hardware changes were required.

- Remove boot EPROM
- Remove address/data latch (not needed)
- Remove 16V8 PAL
- Remove double width bus switch
- Add 3.3V regulator
- Add tiny I^2C EEPROM for VID/PID, code
- Reconnect 1.5K D+ pull-up to DISCON
- Include RESET cap and resistor

The EXCLUSIVE ORs required to generate a suspend signal were eliminated as it is easier to use the CLK24 output pin of the EZ-USB pin to indicate when the USB microcontroller goes into a suspend state. CLK24 stops at the last known state when EZ-USB goes into suspend.

The use of an external address of the 8x930Ax only leaves 16 programmable I/Os for other functions. The 16V8 PAL is used to generate additional signals due to this limitation. With EZ-USB, 24 Programmable I/O pins are available. This provided enough Programmable I/O pins to allow the designer to eliminate the 16V8 PAL.

The double width bus switch, isolating the USB microcontroller from the video subsystem can be eliminated. The bus switch was incorporated to isolate the USB portion for the video subsystem in case it became inoperable due to ESD zap. Instead, substitute an ESD protection device specifically designed for USB. An example of an USB surge suppressor is TI's SN75240PW which is rated to 8 KV.

Since MacWorld in Tokyo was only in 4 weeks from the start of their conversion process, Zoran was able to meet an accelerated development schedule to complete both hardware and firmware conversion in time to make a press announcement at MacWorld. In addition, the board was much simpler, eliminating most all of the glue logic and external memory components surrounding the USB microcontroller. As a result, the system cost was reduced by 15%. Another added benefit was the improvement in performance. Zoran was able to take advantage of the EZ-USB Turbo mode to increase the video capture compression rate by 40%.

Exhibit A

Zoran Video Inlet Schematic Prior to Conversion