Application Note AN 14:

High-Bandwidth Isochronous Traffic in the NET2890

NetChip Technology, Inc.

Document # 632-0057-1401 **Date:** 7/09/99

1 Applicability

NET2890 – FlexUSB – Flex Performance USB Interface Controller (All Revisions)

2 General Description

The NET2890 is an ideal candidate for isochronous data flows consisting of large packets, up to the USB maximum of 1023 bytes. The NET2890 outperforms all other USB controllers in performance with both bulk and isochronous data flows. System throughput is maximized by allowing full-size isochronous packets to be transmitted and received continuously, and CPU usage is minimized by the fast register cycle time, low interrupt overhead, and programmer-oriented architecture.

The Isochronous transfer type is defined by the USB specification for applications requiring real-time data streaming (e.g. video or audio playback). Unlike other USB traffic types, isochronous data has constrained latency and guaranteed data rate, but error-free delivery is not guaranteed. Isochronous data transfers are "unacknowledged", meaning there are no ACK or NAK handshakes for flow control or error recovery. An isochronous endpoint sends or receives one packet per USB frame (1 ms), and an isochronous packet may be up to 1023 bytes in length.

In high-bandwidth isochronous data flows, the packet size may be larger than the NET2890's FIFO size. The FIFOs are read or written while the packet is still in progress. Section 3 analyzes the performance of high-bandwidth isochronous data flows, and shows that the NET2890 is faster than other USB interface controllers in full-size isochronous transfers. Section 4 discusses isochronous transfers under CPU control, and Section 5 discusses DMA-controlled isochronous data flows.

3 Performance Analysis

3.1 Performance with DMA control

For transfers using the NET2890's DMA interface, the 128-byte FIFO imposes a maximum latency constraint for reading or writing the first byte of data from/to the FIFO. This maximum latency is found to be approximately 85.4 μs in Section 5, which is easily achieved by any DMA implementation. The DMA controller must also maintain the average data rate of USB, which is approximately 660 ns to transfer each byte. Note that this requirement is true regardless of the USB controller used. Therefore, the 128-byte FIFO does not limit the performance of a device using DMA-controlled transfers.

© NetChip Technology, Inc., 1999
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

3.2 Performance with CPU control

For transfers done under CPU control, the local CPU fields multiple interrupts per packet transmitted or received. The multiple interrupts add some extra interrupt overhead. However, the NET2890's fast local-bus access time allows a CPU to read or write data several times faster than other USB controllers. The NET2890's fast local bus cycle time more than compensates for the extra interrupt overhead. The performance of the NET2890 (in terms of % CPU utilization to transfer maximum-sized isochronous packets) depends on CPU interrupt latency. Figure 1 illustrates this:

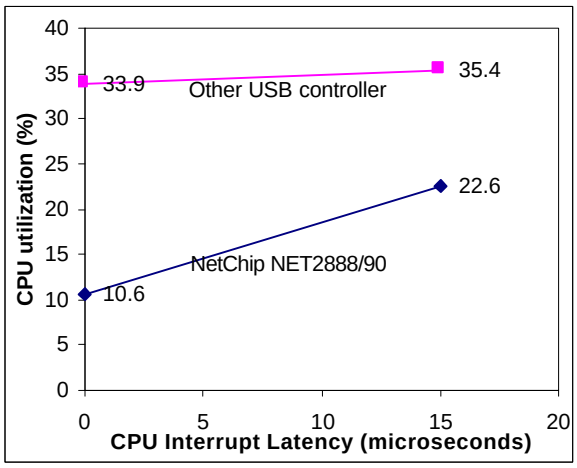

Figure 1: CPU Utilization of NET2888/90 and other USB controllers.

The relative performance of NetChip's NET2888/90 and other controllers depends on the CPU interrupt latency. For a CPU with high interrupt latency (10-15 µs), other controllers require about 50% more CPU time than the NET2888/90. The performance advantage is even more apparent with low interrupt latency; other controllers require roughly 3 times more CPU time than the NET2888/90 for interrupt latencies less than 1 µs. The remainder of this section discusses the details of the calculations.

Under CPU control, a maximum-sized isochronous packet of 1023 bytes will require 8 interrupts for the NET2888/90 (since a FIFO is 128 bytes in length). Each interrupt will incur overhead from three sources:

- CPU interrupt latency at the beginning of the interrupt to begin executing the interrupt routine and push registers (unrelated to the NET2890).
- Four NET2890 register accesses (read IRQSTAT1, write PAGESEL, read EPIRQSTAT, read FIFOCOUNT)
- CPU interrupt latency at the end of the interrupt to pop registers and return from the interrupt (unrelated to the NET2890).

The interrupt overhead will therefore be (8*(CPU interrupt latency) + 8*4*(NET2890 Register cycle)). In addition, there will be 1023 bytes of data transferred into or out of the NET2890 (which are also register cycles). The total time spent by the CPU during one USB frame to service a 1023-byte isochronous packet is therefore:

NET2890 CPU utilization per frame = 8* CPU interrupt latency + (1023 + 32) *NET2890 Register cycle
= 8 * latency + 105.5 µs (using a typical read/write cycle of 100 ns)

A USB interface controller using a 1kbyte FIFO will require:

Other controller CPU utilization per frame = 1 * CPU interrupt latency + (1023 + 4) * Register cycle

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

= latency + 339 µs (using a typical read/write cycle of 330 ns)

The NET2890's fast local-bus timing allows fast local CPU read and write cycles. The access time depends on the local CPU; a typical value used in this example is 100 ns. By contrast, the register access cycle of competing chips is on the order of 330-500 ns. The example uses 330 ns as a typical read/write cycle time, which is on the fast end of the range of access times for competing chips.

From these results, we find that the NET2890 will be faster for all reasonable CPU interrupt latencies, as shown in Figure 1. Lower CPU utilization results in more time available for the CPU to perform other tasks, including servicing other USB endpoints and performing processing on the data.

The next sections describe implementation details to achieve the highest bandwidth with the NET2890 in isochronous applications.

4 Implementing CPU-Controlled Isochronous Transfers

To transmit and receive large isochronous packets using the NET2890 under CPU control, the CPU is responsible for sinking or sourcing data while the transaction is in progress. The 128-byte FIFOs provide buffering to reduce latency requirements for the local CPU. The following sections describe the timing of large isochronous packets.

4.1 Isochronous Receive with Local CPU Control

4.1.1 Isochronous Receive with a Slow CPU

In some applications, the local CPU may be slower reading bytes out of the NET2890 than the USB byte rate. In this case, the cycle time for reading a byte (including loop overhead) is more than 660 ns. Receiving isochronous packets is simple for a slow CPU. The local CPU's firmware can be interrupted on an OUT token interrupt, and the interrupt routine performs the following steps:

- read FIFOCOUNT to see how many bytes are available in the FIFO
- read as many bytes from the receive FIFO as are available
- loop until FIFOCOUNT is zero.

If the CPU is very slow, it may not be capable of receiving a maximum-sized (1023-byte) isochronous packet. Looking strictly at data transfer, the CPU can only receive a packet up to the point where it has lagged 128 bytes behind the incoming data. At that point, the NET2890's FIFO will have filled, and data will be lost. The maximum packet size that can be received by a slow CPU is approximately:

Max packet size = 128 / (1-660 / CPU cycle time)

The CPU cycle time is the cycle time to read one byte (in ns), including loop overhead if applicable. If the maximum packet size given by the equation is larger than 1023, the maximum is 1023 bytes.

This approximation does not take into account interrupt overhead time, so the actual maximum packet size the local CPU can successfully receive may be slightly lower due to interrupt response latency.

4.1.2 Isochronous Receive with a Fast CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

The “FIFO Almost Full Interrupt” (in register EPIRQSTAT) is often used to assist the firmware in receiving large isochronous packets. The interrupt activates when the FIFO crosses a firmware-programmable threshold (in register Fn_AFTH) while the FIFO is filling. When the interrupt occurs, the firmware responds by clearing the interrupt status bit, and reading data from the FIFO.

The almost-full threshold level (register Fn_AFTH) is typically set once, during the firmware’s initialization routine. The level of this threshold determines the latency requirements on the local CPU. Figure 2 below shows the details of the timing requirements.

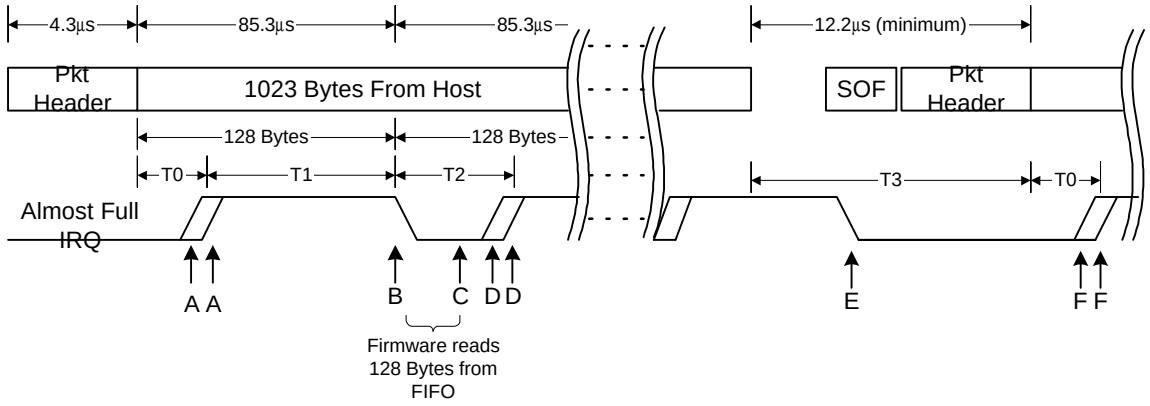

Figure 2: Timing of Large Isochronous Packet Received from Host PC

Timing Point	Description
A	Almost Full Interrupt activates at firmware-programmed FIFO threshold.
B	Firmware clears interrupt, begins reading data from FIFO.
C	Firmware finishes reading data from FIFO and exits the interrupt routine. The FIFO count must be below the Almost Full threshold at some time while reading data, so that the FIFO will cross the threshold again at point (D).
D	Almost Full Interrupt activates again as FIFO crosses Almost Full threshold. Firmware should have completed previous interrupt routine by this time.
E	Firmware clears interrupt, begins reading data from FIFO.
F	Almost Full Interrupt activates at firmware-programmed FIFO threshold.

The figure shows the NET2890 receiving a maximum-sized isochronous packet (1023 bytes). The almost-full interrupt activates at point (A), when the FIFO fills up past the threshold. Point (B) corresponds to the time at which a full 128 bytes have been received. By this time, the firmware must have started reading data from the FIFO. The maximum firmware latency to respond to the almost full interrupt and read the first byte of data is therefore T1.

The firmware continues reading data from the FIFO while data is coming in, and finishes reading data at point (C). At point (D), the FIFO almost full threshold activates again.

After one isochronous packet, the next packet will occur in the following USB frame. This puts a lower bound on the time between received packets. T3 in the figure shows the time from the last data byte of an isochronous receive packet to the first data byte of the next isochronous receive packet. T3 consists of a minimum of:

- 2.50 bytes: CRC16 + SE0 + bus turn-around time
- 5.00 bytes: end-of-frame idle time
- 4.25 bytes: SOF packet (Sync, PID, timestamp, CRC5, EOP)

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

- 6.50 bytes: Packet header (OUT + DATA0/1)

T3 is therefore at least 18.25 USB byte times (12.2µs)

Details of the timing parameters are shown in the table below:

Parameter	Equation	Example with Fn_AFTH = 80 (decimal) bytes
T0: Beginning of data in one packet to almost-full interrupt activating	Fn_AFTH * 0.667 µs	53.4 µs
T1: Firmware Interrupt Response Latency (maximum)	**(128-Fn_AFTH) * 0.667 µs**	**32.0 µs**
T2: Time available for firmware to read FIFO (maximum)	**Fn_AFTH * 0.667 µs**	**53.4 µs**
T3: Time from end of data in one packet to beginning of data in next packet (minimum)	12.2 µs	12.2 µs

The T1 and T2 timing constraints bound the firmware. Select the almost-full threshold so that these parameters are met. The setting of the almost full threshold level should be as high as possible, but still low enough so that the firmware latency is guaranteed to be less than T1.

4.2 Isochronous Transmit with Local CPU Control

For large isochronous packets being transmitted to the Host PC, the first 128 bytes of the transfer are placed in the transmit FIFO by the firmware as soon as they are available, often before the Host PC begins requesting data. This "primes" the FIFO so that when the Host PC sends an IN token requesting data, the NET2890 responds with this data while additional bytes are loaded into the transmit FIFO.

4.2.1 Isochronous Transmit with a Slow CPU

In some applications, the local CPU may be slower writing bytes into the NET2890 than the USB byte rate. In this case, the cycle time for writing a byte (including loop overhead) is more than 660 ns. Transmitting isochronous packets is simple for a slow CPU. The local CPU's firmware can be interrupted on an IN token interrupt, and the interrupt routine would perform the following steps:

- read FIFOCOUNT to determine how many empty locations are available in the FIFO (e.g. 128 – FIFOCOUNT)
- write as many bytes to the transmit FIFO as there are empty locations
- test the "Data Packet Transmitted" interrupt status bit. Loop until it is set.
- Clear the "Data Packet Transmitted" interrupt status bit. Begin priming FIFO for next packet to be transmitted to Host PC.

If the CPU is slow enough, it may not be capable of transmitting a maximum-sized (1023-byte) isochronous packet. Looking strictly at data transfer, the CPU can only transmit a packet up to the point where it has lagged 128 bytes behind the outgoing data. At that point, the NET2890's FIFO will have emptied, and the packet will be terminated by the NET2890. The maximum packet size that can be transmitted by a slow CPU is approximately:

Max packet size = 128 / (1-660 / CPU cycle time)

The CPU cycle time is the cycle time to write one byte (in ns), including loop overhead if applicable. If the maximum packet size given by the equation is larger than 1023, the maximum is 1023 bytes.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

This approximation does not take into account interrupt overhead time, so the actual maximum packet size the local CPU can successfully transmit may be slightly lower due to interrupt response latency.

4.2.2 Isochronous Transmit with a Fast CPU

The "FIFO Almost Empty Interrupt" (in register EPIRQSTAT) is often used to assist the firmware in transmitting large isochronous packets. The interrupt activates when the FIFO crosses a firmware-programmable threshold (in register Fn_AETH) while emptying. When the interrupt activates, the firmware responds by clearing the interrupt status bit, and writing data to the FIFO.

The almost-empty threshold level (register Fn_AETH) is typically set once, during the firmware's initialization. The level of this threshold determines the latency requirements on the local CPU. Figure 3 below shows the details of the timing requirements.

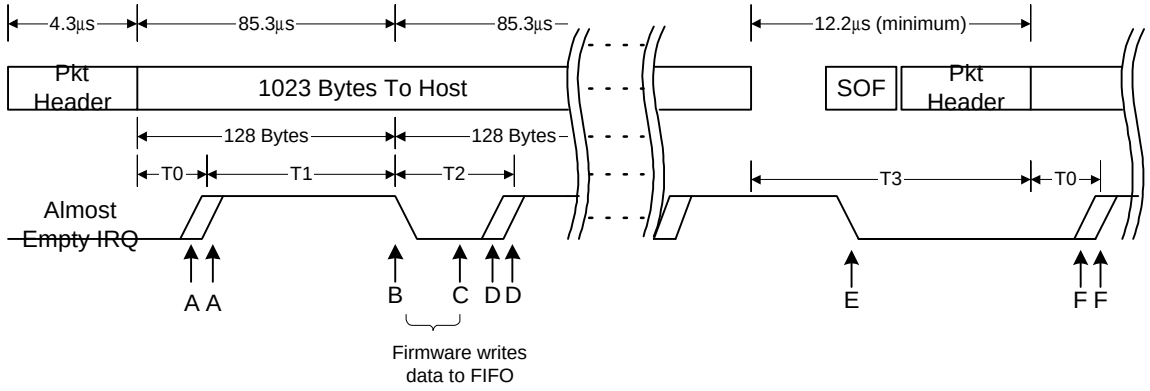

Figure 3: Timing of Large Isochronous Packet Transmitted to Host PC

Timing Point	Description
A	Almost Empty Interrupt activates at firmware-programmed FIFO threshold.
B	Firmware clears interrupt, begins writing data to FIFO.
C	Firmware finishes writing data to the FIFO and exits the interrupt routine. The FIFO count must be above the Almost Empty threshold at some time while writing data, so that the FIFO will cross the threshold again at point (D).
D	Almost Empty Interrupt activates again as FIFO crosses Almost Empty threshold. Firmware should have completed previous interrupt routine by this time.
E	Firmware clears interrupt, begins writing data to FIFO.
F	Almost Empty Interrupt activates at firmware-programmed FIFO threshold.

The figure shows the NET2890 transmitting a maximum-sized isochronous packet (1023 bytes). The almost-empty interrupt activates at point (A), when the FIFO empties below the threshold. Point (B) corresponds to the time at which a full 128 bytes have been transmitted. By this time, the firmware must have started writing additional data to the FIFO. T1 is the maximum firmware latency response to the almost full interrupt and writing of the first byte of data.

The firmware continues writing data to the FIFO while data is being transmitted, and finishes writing data at point (C). At point (D), the FIFO almost empty threshold activates again.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

As in the case of receive data transfers under CPU control, a second isochronous packet occurs in the following USB frame. T3, the time from the last data byte of an isochronous receive packet to the first data byte of the next isochronous receive packet, is therefore at least 18.25 USB byte times (12.2µs), as calculated for receive data transfers under CPU control.

Details of the timing parameters are shown in the table below:

Parameter	Equation	Example with Fn_AETH = 48 (decimal) bytes
T0: Beginning of data in one packet to almost-empty interrupt activating	(128-Fn_AETH) * 0.667 µs	53.4 µs
T1: Firmware Interrupt Response Latency (maximum)	**Fn_AETH * 0.667 µs**	**32.0 µs**
T2: Time available for firmware to begin writing additional bytes to transmit FIFO (maximum)	**(128-Fn_AETH) * 0.667 µs**	**53.4 µs**
T3: Time from end of data in one packet to beginning of data in next packet (minimum)	12.2µs	12.2 µs

The T1 and T2 timing constraints bound the firmware. Select the almost-empty threshold so that these parameters are met. The setting of the almost empty threshold level should be as low as possible, but still high enough so that the firmware latency is guaranteed to be less than T1.

5 Implementing DMA-Controlled Isochronous Transfers

With the NET2890's interface to a DMA controller, receiving and transmitting isochronous packets is easy. The DMA controller can be programmed to receive or transmit a large stream of data, spanning multiple packets, without any local CPU intervention. The 128-byte FIFO attached to the isochronous endpoint minimizes latency requirements of the DMA controller: bytes are received or transmitted through the FIFO, so the DMA needs to access the local bus only frequently enough to prevent over- or under-flows of this FIFO during the packet. The latency requirements are detailed below.

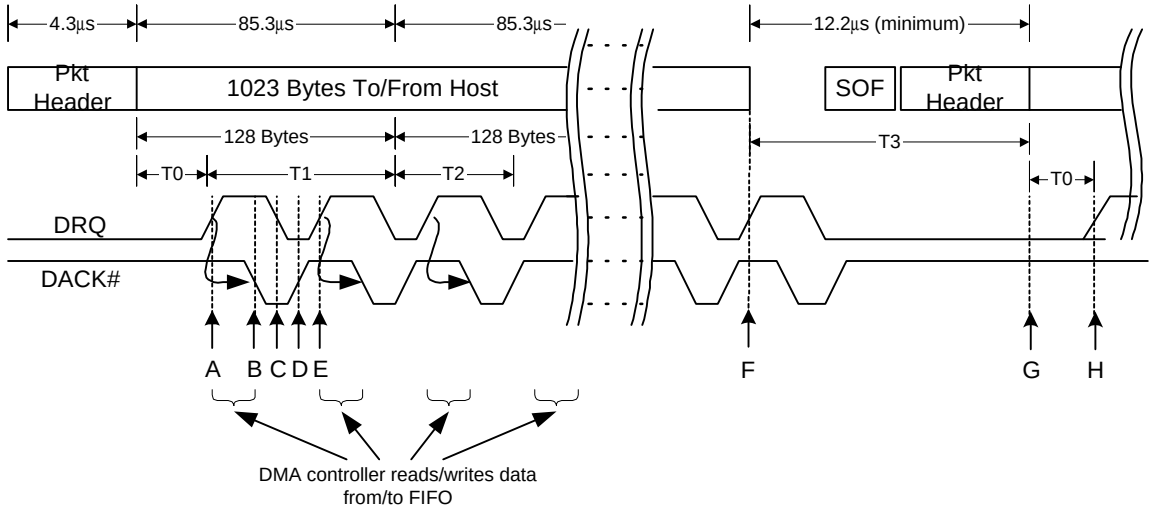

Figure 4: Timing of Large Isochronous Packet Transmitted or Received with DMA Control

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Timing Point	Description
A	NET2890 requests DMA cycle (read or write) by activating DRQ.
B	DMA controller acknowledges DMA cycle by asserting DACK#. The data bus is valid now.
C	NET2890 responds to DACK# by clearing DRQ. If this was a DMA write, the NET2890 latches the data.
D	DMA controller acknowledges end of DMA cycle by de-asserting DACK#.
E	NET2890 requests DMA cycle (read or write) by activating DRQ. This occurs when another byte is available in the receive FIFO (for received packets), or when there is room in the transmit FIFO (for transmitted packets).
F	Last byte of data packet received or transmitted. There will be additional DMA request cycles until the FIFO is empty (for received packets) or full (for transmitted packets). For transmitted packets, the data in the full FIFO become the first 128 bytes of the next isochronous packet.
G	Beginning of next data packet. This occurs in the next USB frame, at minimum after a SOF and packet header.
H	NET2890 requests DMA cycle (read or write) by activating DRQ.

When transmitting packets, the NET2890 "primes" the FIFO by requesting DMA transfers until the FIFO is full. This occurs even before the USB Host sends an IN token. Then, when the Host PC has read the first byte of data from the transmit FIFO, the NET2890 will request a DMA cycle again since there is one free location in the transmit FIFO (point A). Similarly, after the end of the transmitted packet (point F), the NET2890 will request DMA cycles for the next packet until the transmit FIFO is full.

When receiving large isochronous packets from the Host PC, the DMA controller will begin requesting DMA cycles when the first byte is received. The DMA controller will continue to request DMA cycles while there are bytes in the FIFO. The receive FIFO will simultaneously be filling with additional bytes from the Host PC.

As in the case of CPU-controlled data transfers, a second isochronous packet occurs in the following USB frame. T3, the time from the last data byte of an isochronous packet to the first data byte of the next isochronous packet, is therefore at least 18.25 USB byte times (12.2µs), as calculated for CPU transfers.

Details of the timing parameters are shown in the table below:

Parameter	Value for OUT Transaction (Device to Host)	Value for IN Transaction (Device to Host)
T0: Beginning of data in one packet to DRQ asserting	0.667 µs	0.667 µs
T1: DMA Controller latency to respond to first DMA request	**128*0.667 µs = 85.4 µs (maximum)**	**128*0.667 µs = 85.4 µs (maximum)**
T2: Average access time to perform a single DMA cycle (one byte of data transferred)	**0.667 µs**	**0.667 µs**
T3: Time from end of data in one packet to beginning of data in next packet (minimum)	12.2 µs	12.2 µs

6 Isochronous Failed Packets

Isochronous transfer type is unique in that there is no handshaking or retry protocol. The CRC field, however, is still present. The firmware decides how to handle the data in failed packets.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

First, firmware will never know about errors that occur in packets transmitted to the Host PC. Since the PC does not provide an acknowledgment, there is no way to distinguish a successful transmission from a failed one, and no built-in way to retry the missing data. This is a characteristic of the isochronous transfer type.

For received packets under CPU control, the firmware can determine whether the packet was received successfully. EPUSBSTAT register bits are set for an isochronous endpoint even though there is no handshake sent to the Host PC. If the "Timeout" bit is set, then the received packet failed a CRC check or had some other error. The firmware should clear the bit. The received data from the failed packet may be used or discarded, depending on the application.

For received packets under DMA control, the EPUSBSTAT register bits are also set. If the firmware is not fielding Data Packet Received interrupts, the firmware can read the "Timeout" bit when the end-of-transfer interrupt occurs. If the bit is set, there were one or more failed packets in the previous transfer. If the firmware must know which packets failed, it can respond to the Data Packet Received Interrupt, and read the "Timeout" bit after each received packet.

7 Summary

The NET2890 is designed specifically to support the highest-performance USB peripherals using either bulk or isochronous transfers. The programmer-oriented architecture and fast local-bus access time reduce CPU utilization for high-bandwidth USB transfers. The FIFOs provide a latency buffer when receiving or transmitting large isochronous packets, reducing the demands on the local CPU. The local CPU can achieve the maximum isochronous data rate using either DMA or CPU-controlled data transfers.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com