Application Note AN08 :

NET2890 Programming Flowchart

NetChip Technology, Inc.

Document # 660-0057-0003 **Revision:** 0.1 **Date:** 11 / 5 / 98

Applicability

NET2890 Rev 2B

NET2890 PROGRAMMING FLOWCHART

This flowchart presents a simple, yet complete NET2890 programming example. It is a good case study for programmers that are new to the NET2890.

The flowchart applies to a simple bus-powered USB device that supports:
- Enumeration
- Control transfers (read and write)
- Data transfers (IN and OUT)

The flowchart is based on an interrupt-driven CPU. After initialization, all USB events are handled under interrupt control.

For brevity, several limitations apply to the flowchart:
- Only two NET2890 data endpoints are used
- Minimal number of USB requests are supported
- Simple buffer management
- No DMA

With hardware and software support, these limitations can be removed.

NET2890: The Programmer's Perspective:
The NET2890 is a USB interface controller designed to abstract firmware from low-level complexities of USB. For example, to return a packet of data to the host, firmware simply writes data to a FIFO, and (optionally) sets a bit indicating that the NET2890 can send the contents of the FIFO to the host.

When the host successfully reads the packet, an interrupt (if enabled) is generated. The NET2890 automatically handles timing, PID generation, CRC, bit stuffing, retries, etc. Firmware is responsible for all data content however. For example, when a Setup Request successfully arrives, firmware must examine the request, and generate an appropriate response. Descriptor content is completely controlled by firmware. When the host requests a descriptor, firmware loads the descriptor content into a FIFO for the host to read. Note that an endpoint's descriptor content must correspond to the endpoint's programming in the NET2890.

Table of Contents

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

At a minimum, programmers should be familiar with USB specification sections 8.5 and 9.3 - 9.6.

ENUMERATION:
For many USB devices, the most difficult task is enumeration. A device must examine the contents of the Setup Request and respond accordingly. This flowchart supports the minimum set of Setup Requests that are required for enumeration:
- Get Device Descriptor
- Get Configuration Descriptor
- Set Address
- Set Configuration

If the program does not recognize a Setup Request it will instruct the NET2890 to stall endpoint 0.

CONTROL TRANSFERS:
In this flowchart, Control Read data transfers return standard descriptors to the host during enumeration. The model can easily be extended for any Control Read data transfer, including Class or Vendor Specific requests.

Control Write data transfers are not required for enumeration, however the flowchart includes Control Write data transfer programming for completeness. The model can easily be extended for any Control Write data transfer, including Class or Vendor Specific requests.

DATA TRANSFERS:
For brevity, this example uses only two NET2890 data endpoints. Endpoint A is arbitrarily programmed as a 64-byte BULK OUT endpoint, and endpoint B is programmed as a 64-byte BULK IN endpoint. Note that a NET2890 data endpoint can be configured as any type (BULK, ISO, or INT); size (1023 bytes max); or direction (IN or OUT).

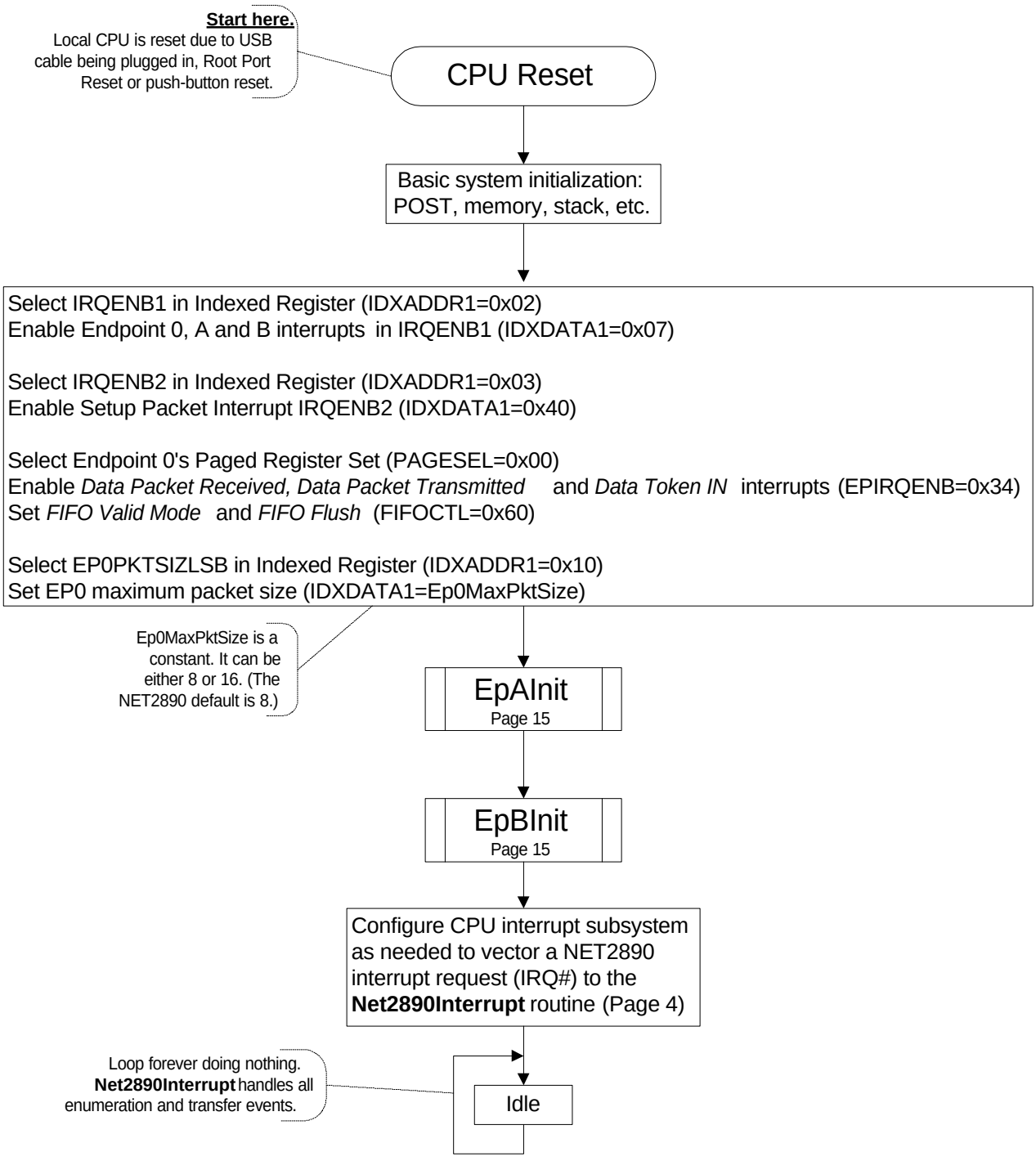
Data transfers are somewhat artificial in this program. A real USB device typically has a mechanism for transfering data from a USB endpoint to an electronic unit (such as a MODEM) and vice-versa. This example, however, simply transfers data to or from the device's memory buffers. Once a receive buffer becomes full or a transmit buffer becomes empty, the endpoint will NAK.

CHIP ERRATA
This flowchart includes a workaround for NET2890 errata 630-0057-0401.

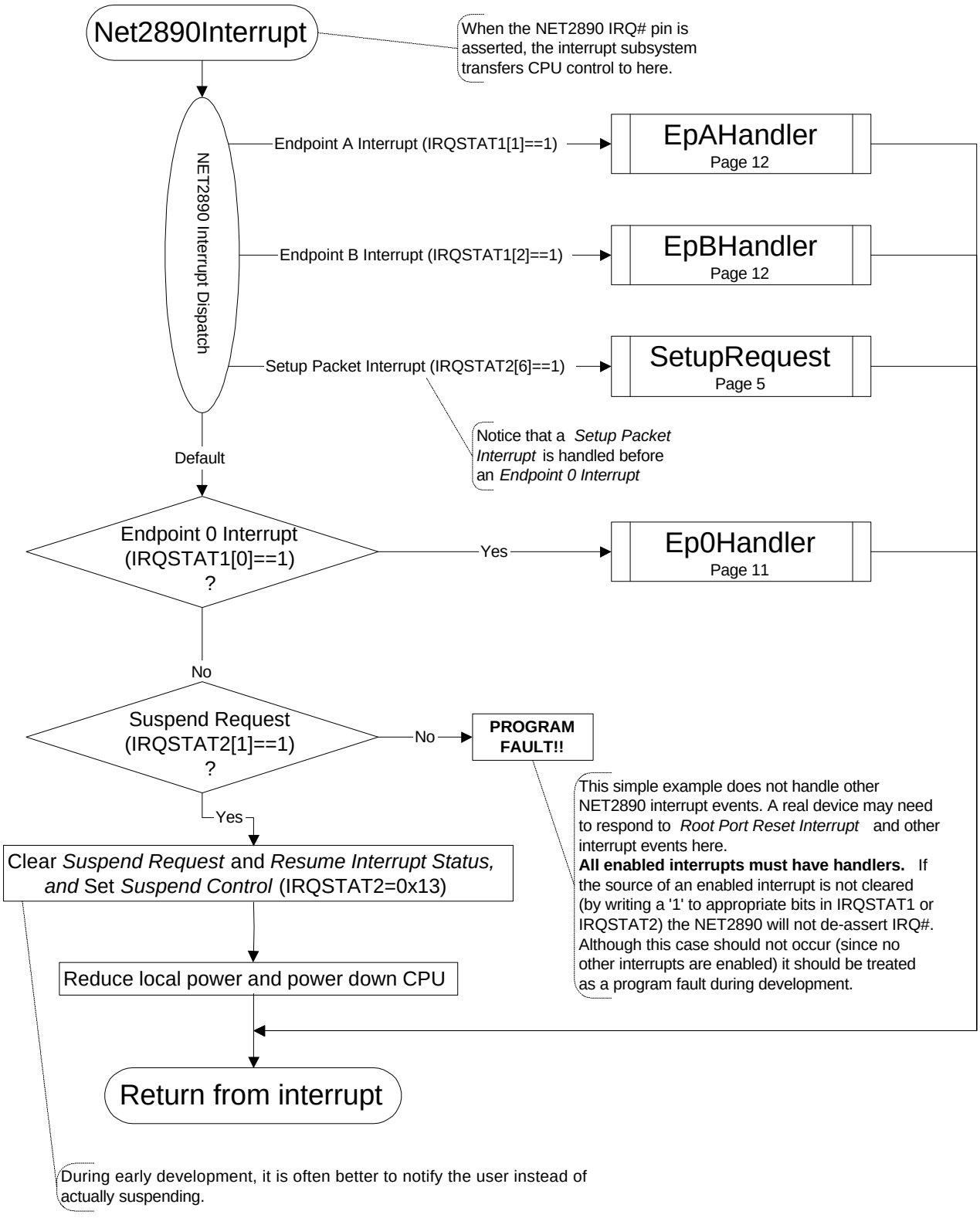
DISCLAIMER
NetChip's NET2890 design team has made every effort to make this document useful, accurate and complete. You should recognize, however, that errors and ommisions may exist. If you have problems with this document, please contact NetChip Technology, Inc.

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

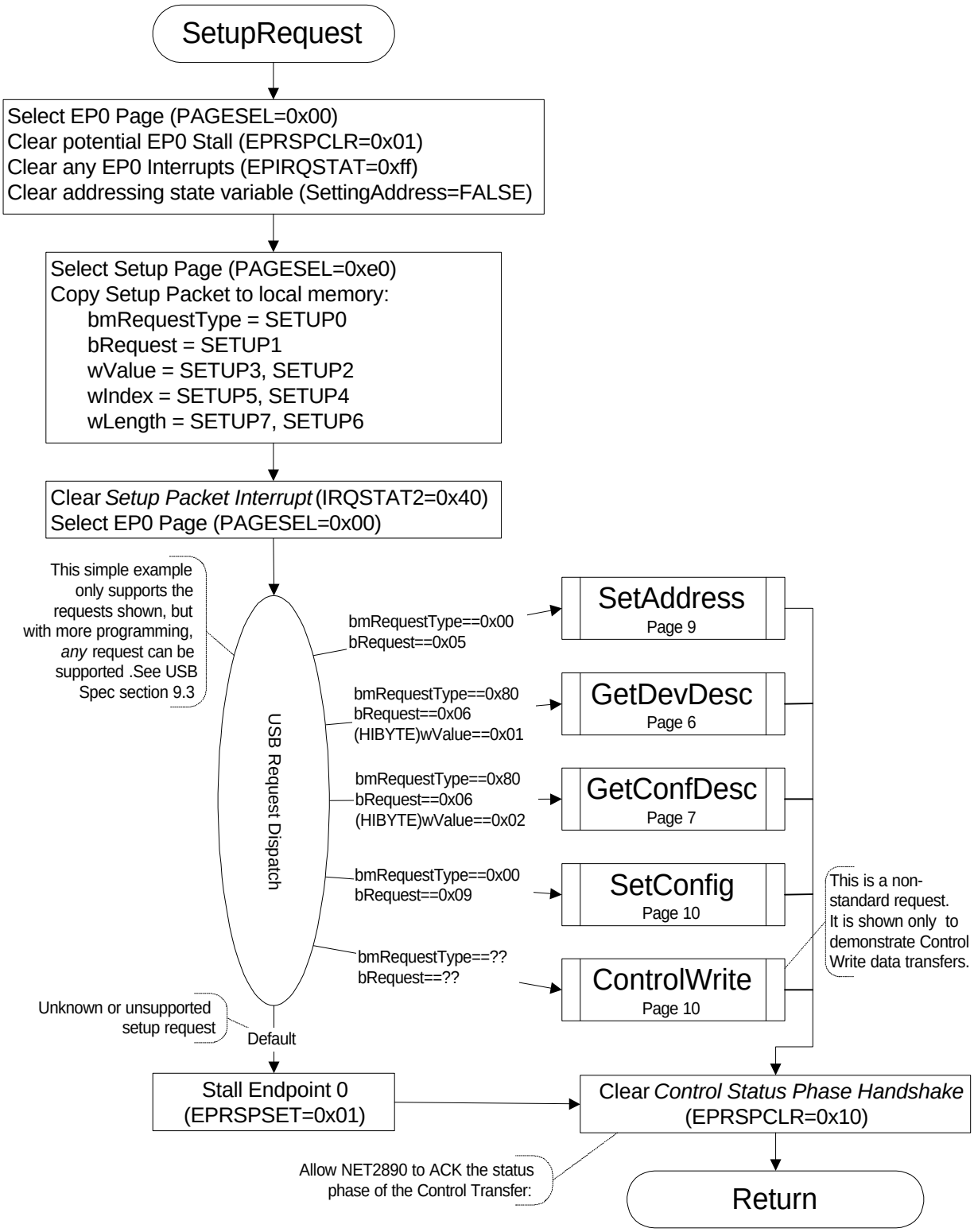
NET2890 Initialization Sequence



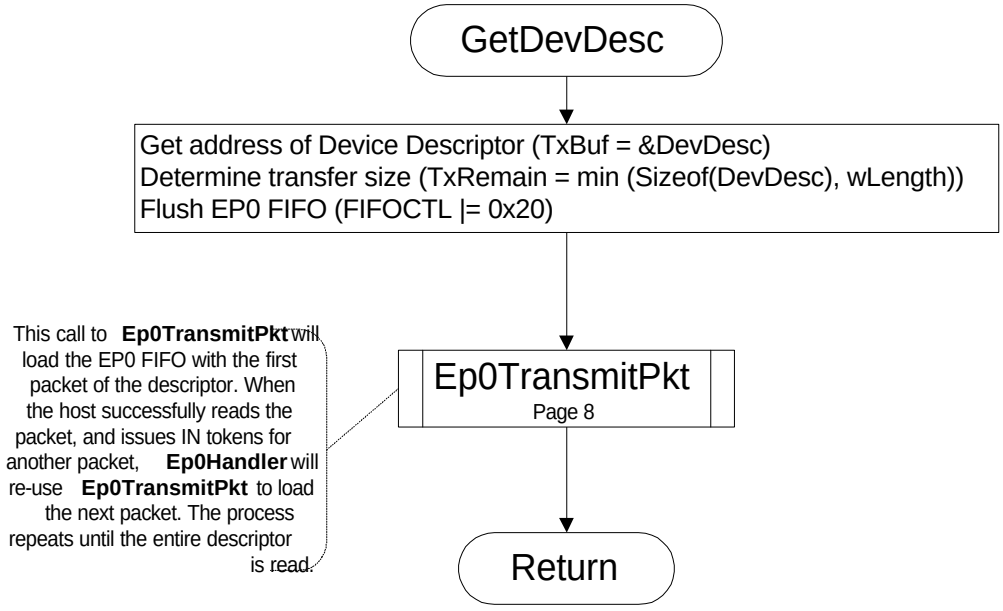
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

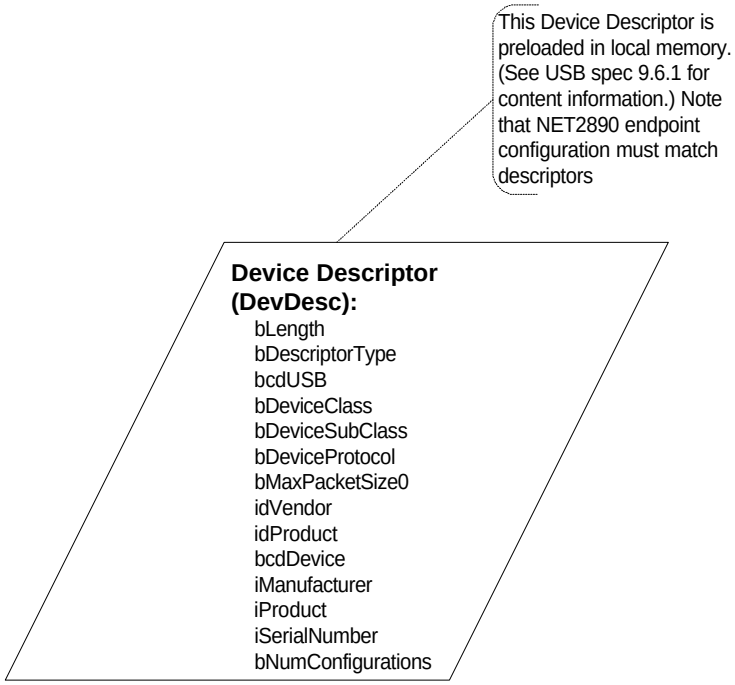


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

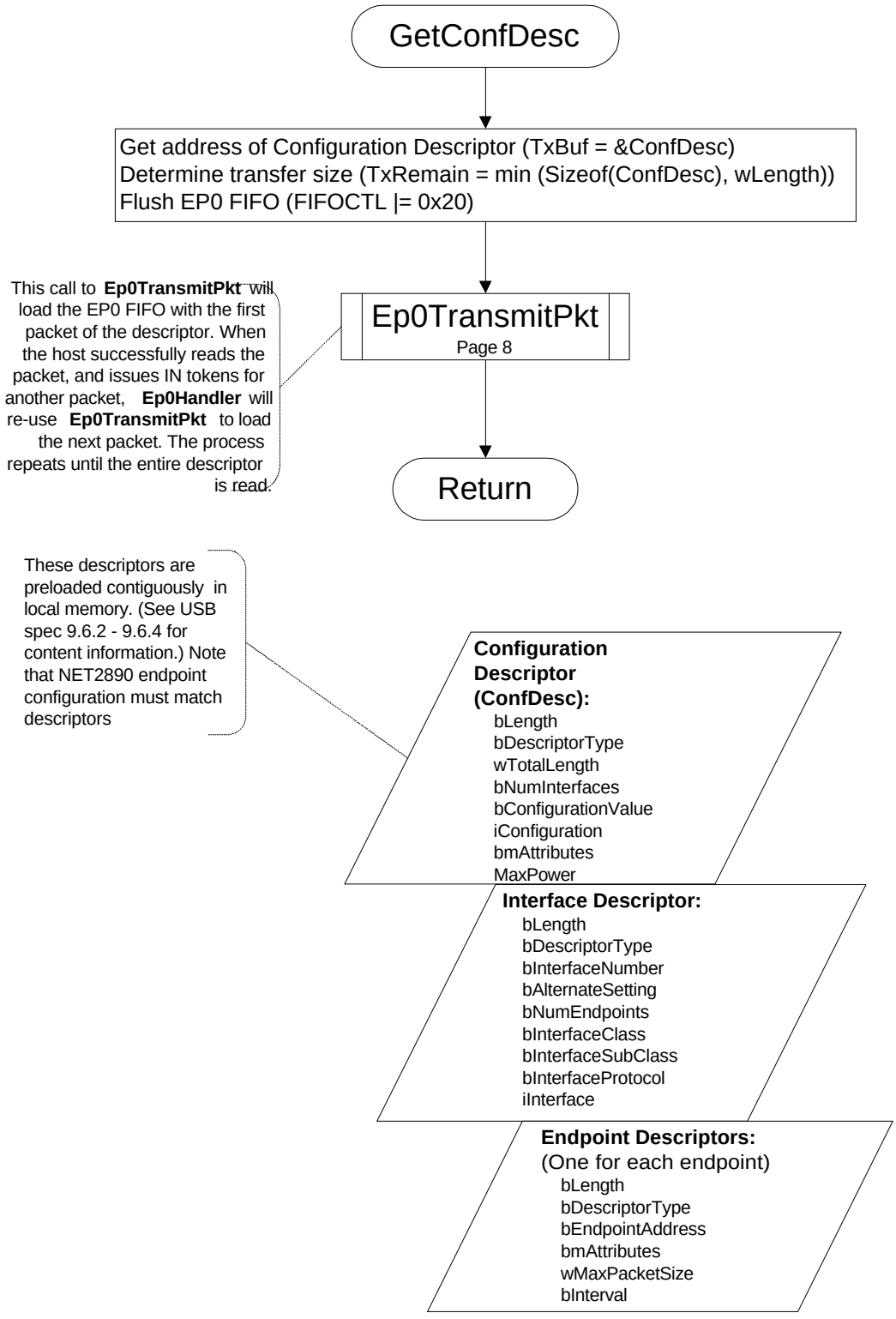


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

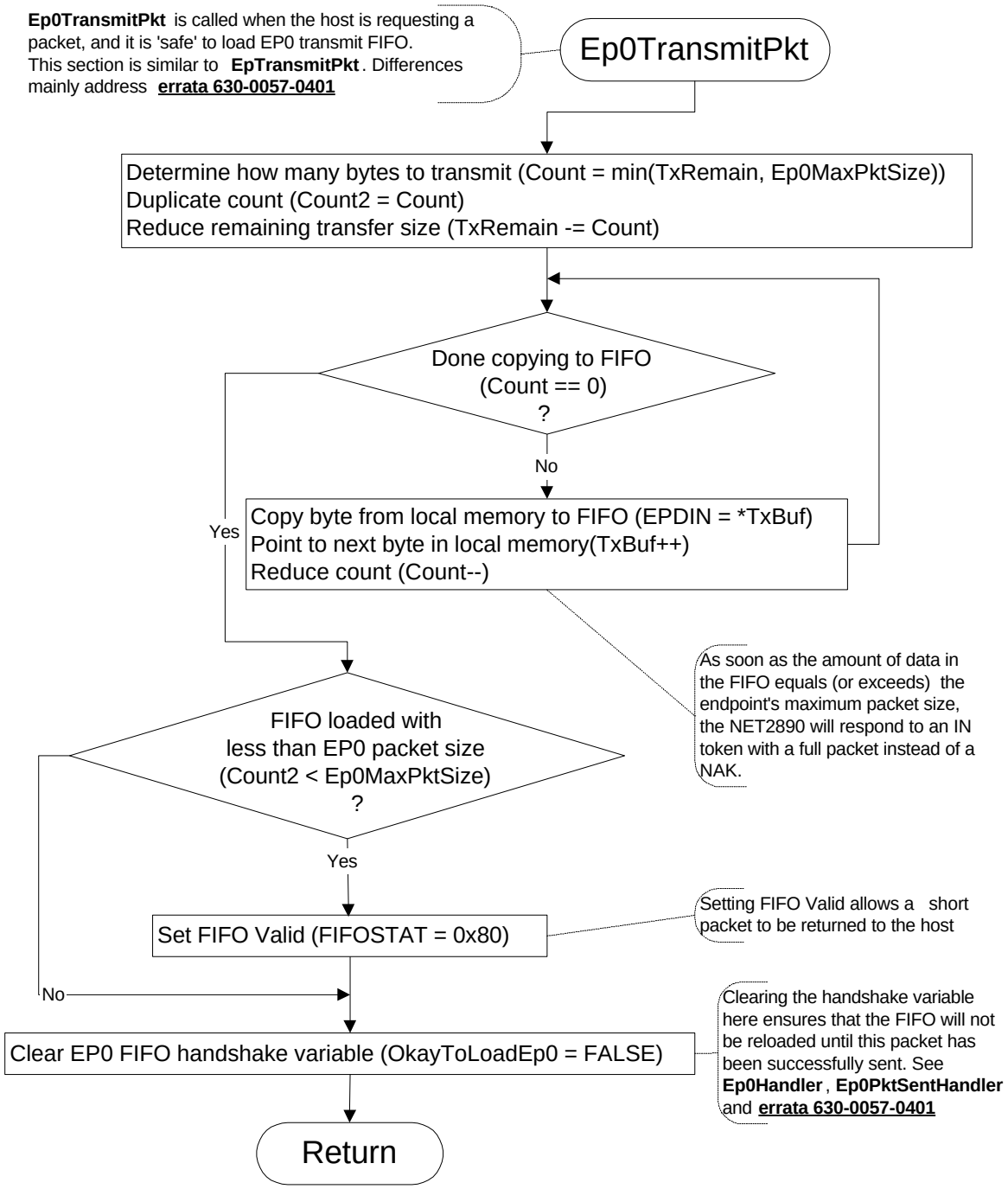


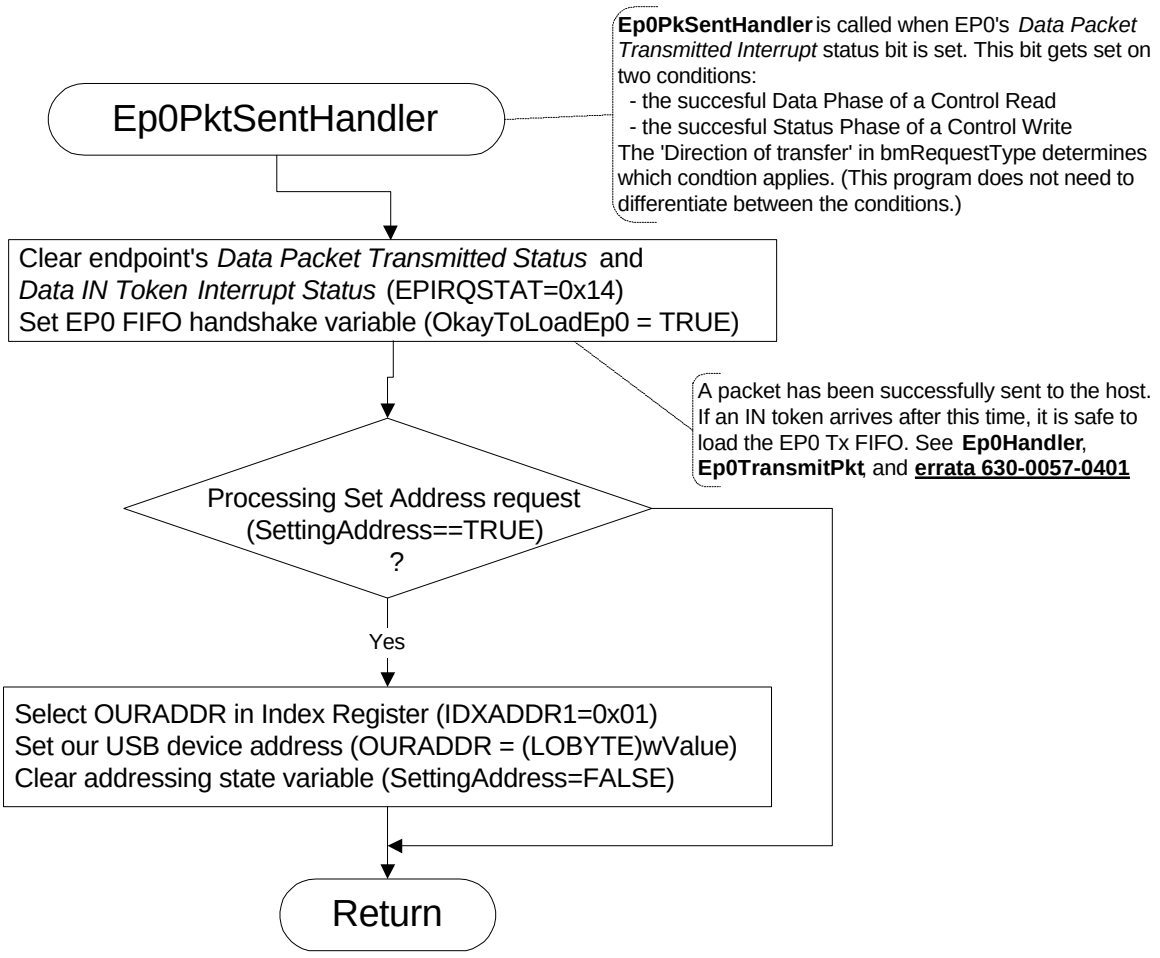


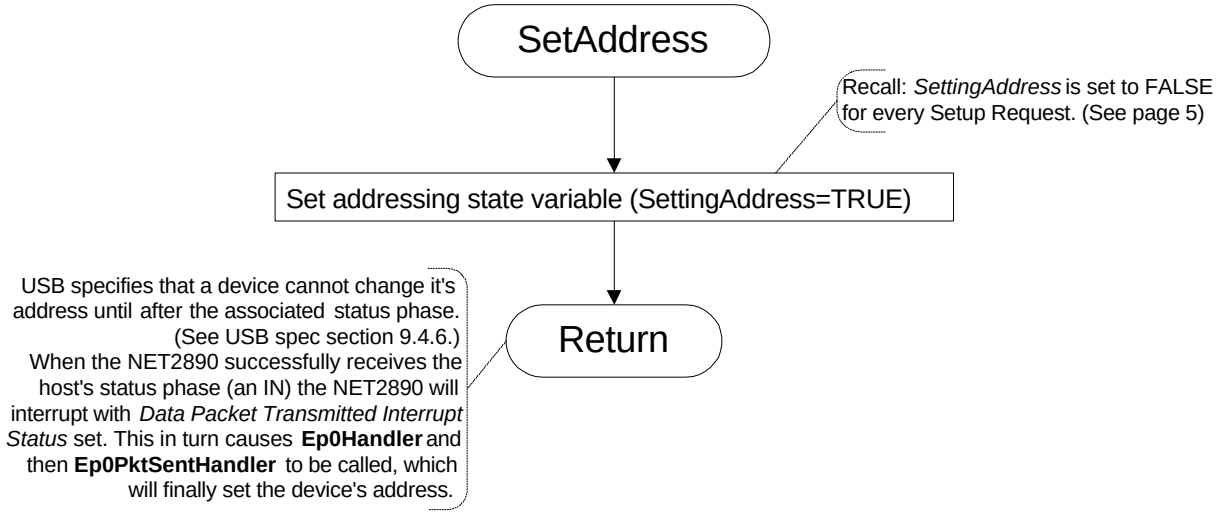
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com



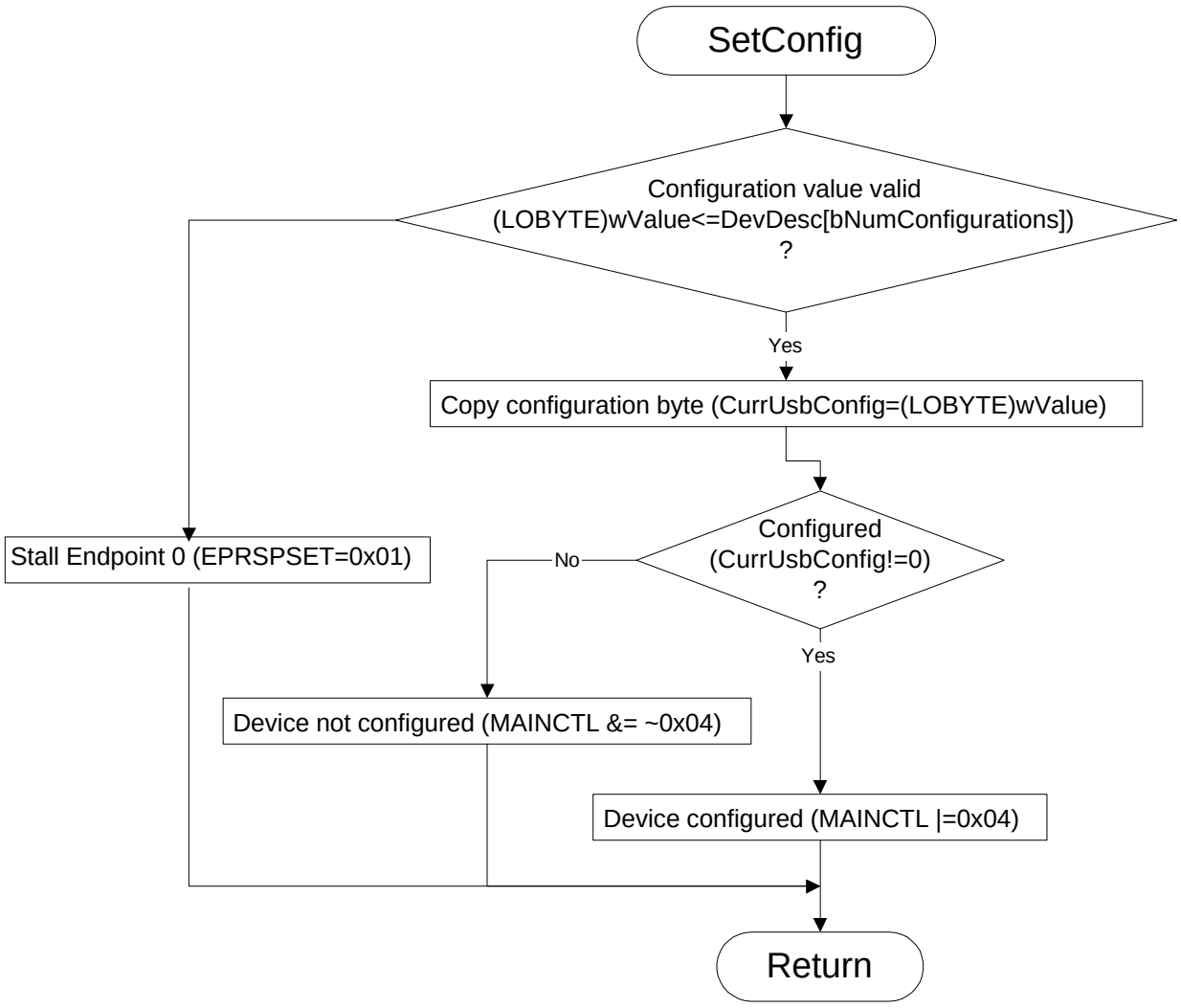
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

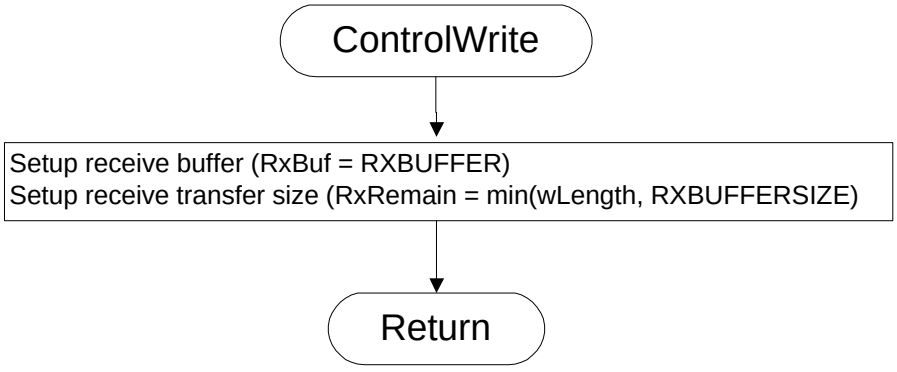

Ep0TransmitPkt is called when the host is requesting a packet, and it is 'safe' to load EP0 transmit FIFO. This section is similar to **EpTransmitPkt**. Differences mainly address **errata 630-0057-0401**
Ep0TransmitPkt
Determine how many bytes to transmit (Count = min(TxRemain, Ep0MaxPktSize))
Duplicate count (Count2 = Count)
Reduce remaining transfer size (TxRemain -= Count)
Done copying to FIFO (Count == 0) ?
No
Yes
Copy byte from local memory to FIFO (EPDIN = *TxBuf)
Point to next byte in local memory(TxBuf++)
Reduce count (Count--)
As soon as the amount of data in the FIFO equals (or exceeds) the endpoint's maximum packet size, the NET2890 will respond to an IN token with a full packet instead of a NAK.
FIFO loaded with less than EP0 packet size (Count2 < Ep0MaxPktSize) ?
Yes
Set FIFO Valid (FIFOSTAT = 0x80)
Setting FIFO Valid allows a short packet to be returned to the host
No
Clear EP0 FIFO handshake variable (OkayToLoadEp0 = FALSE)
Clearing the handshake variable here ensures that the FIFO will not be reloaded until this packet has been successfully sent. See **Ep0Handler**, **Ep0PktSentHandler** and **errata 630-0057-0401**
Return



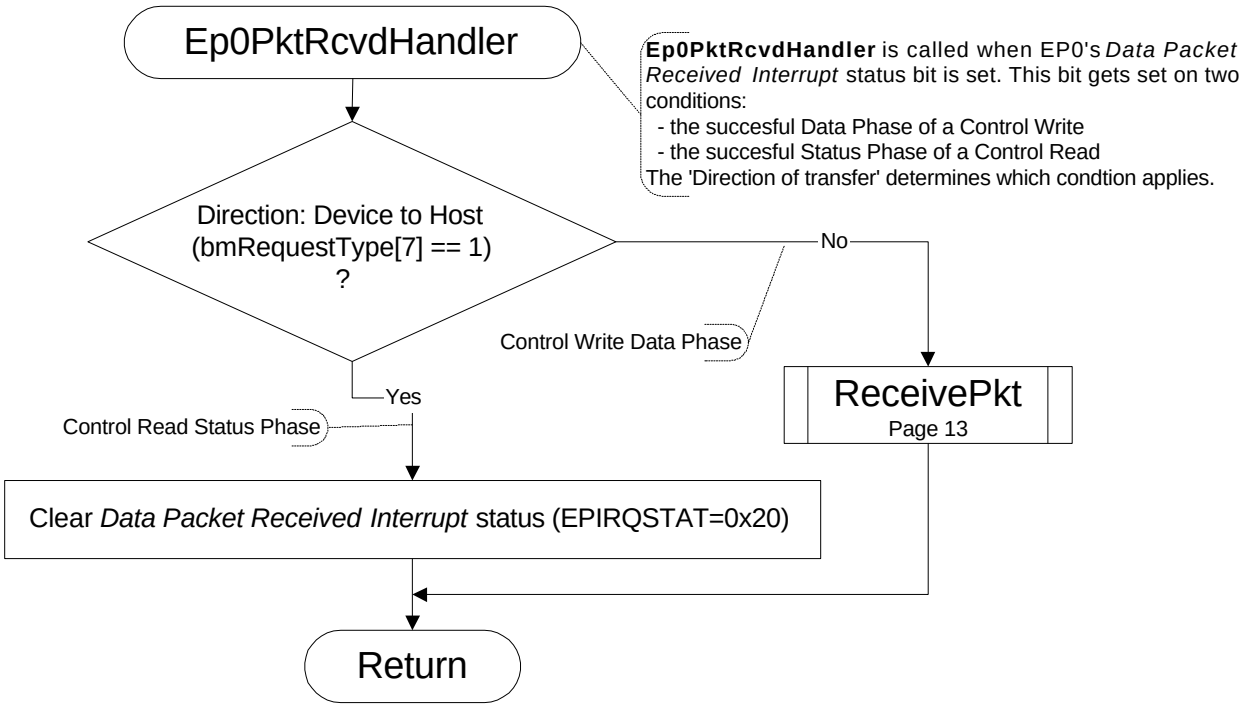


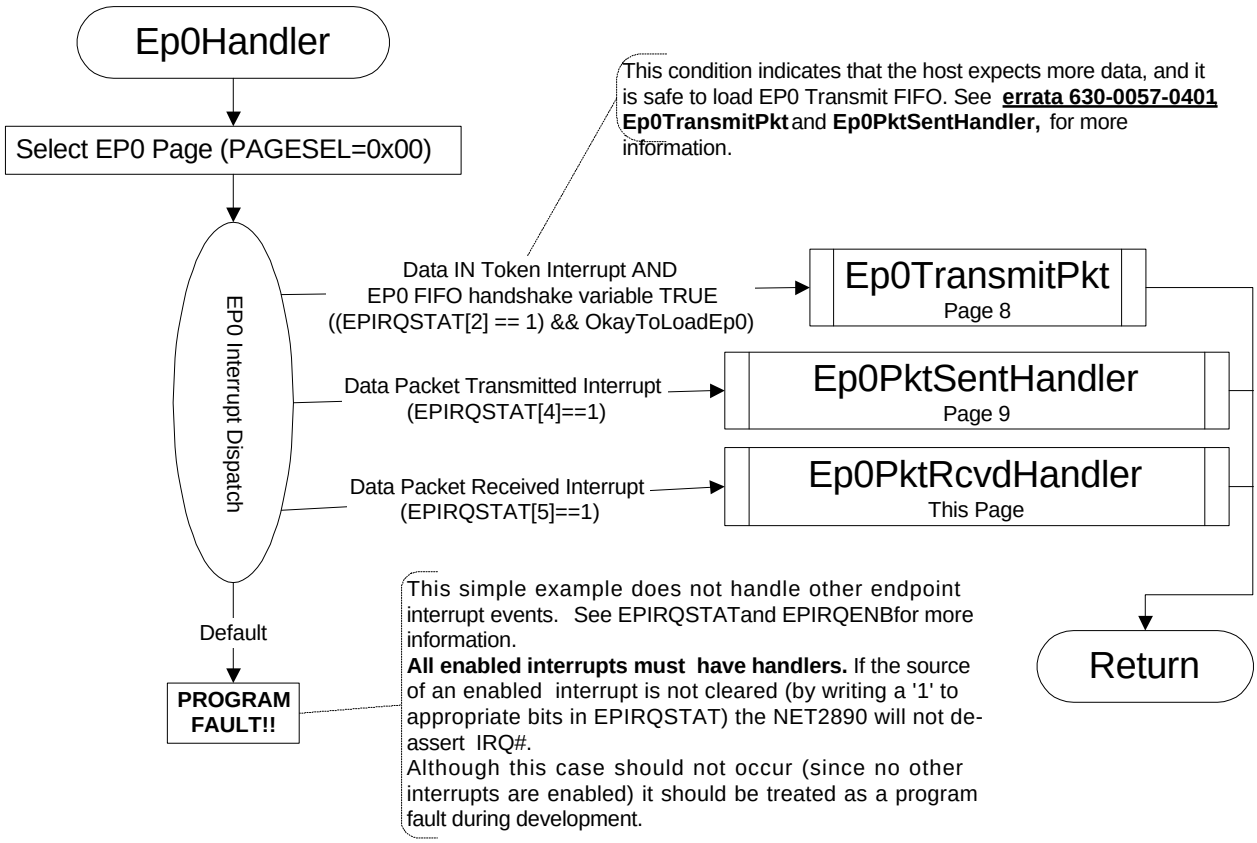
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com



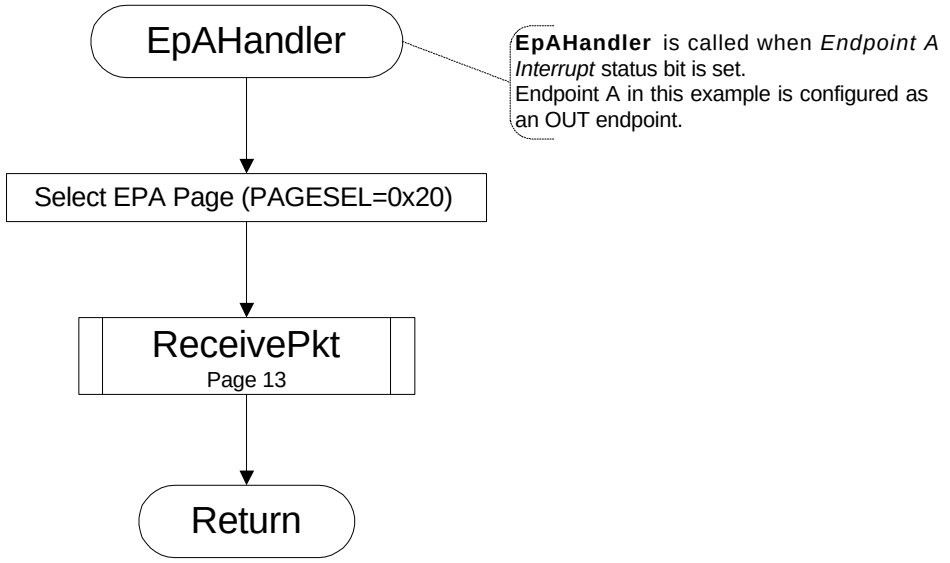


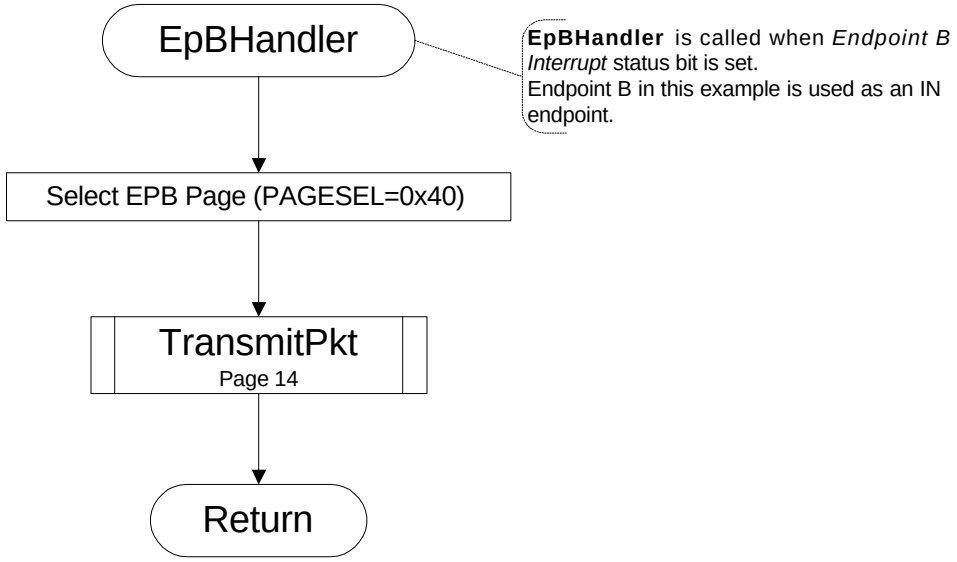
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com



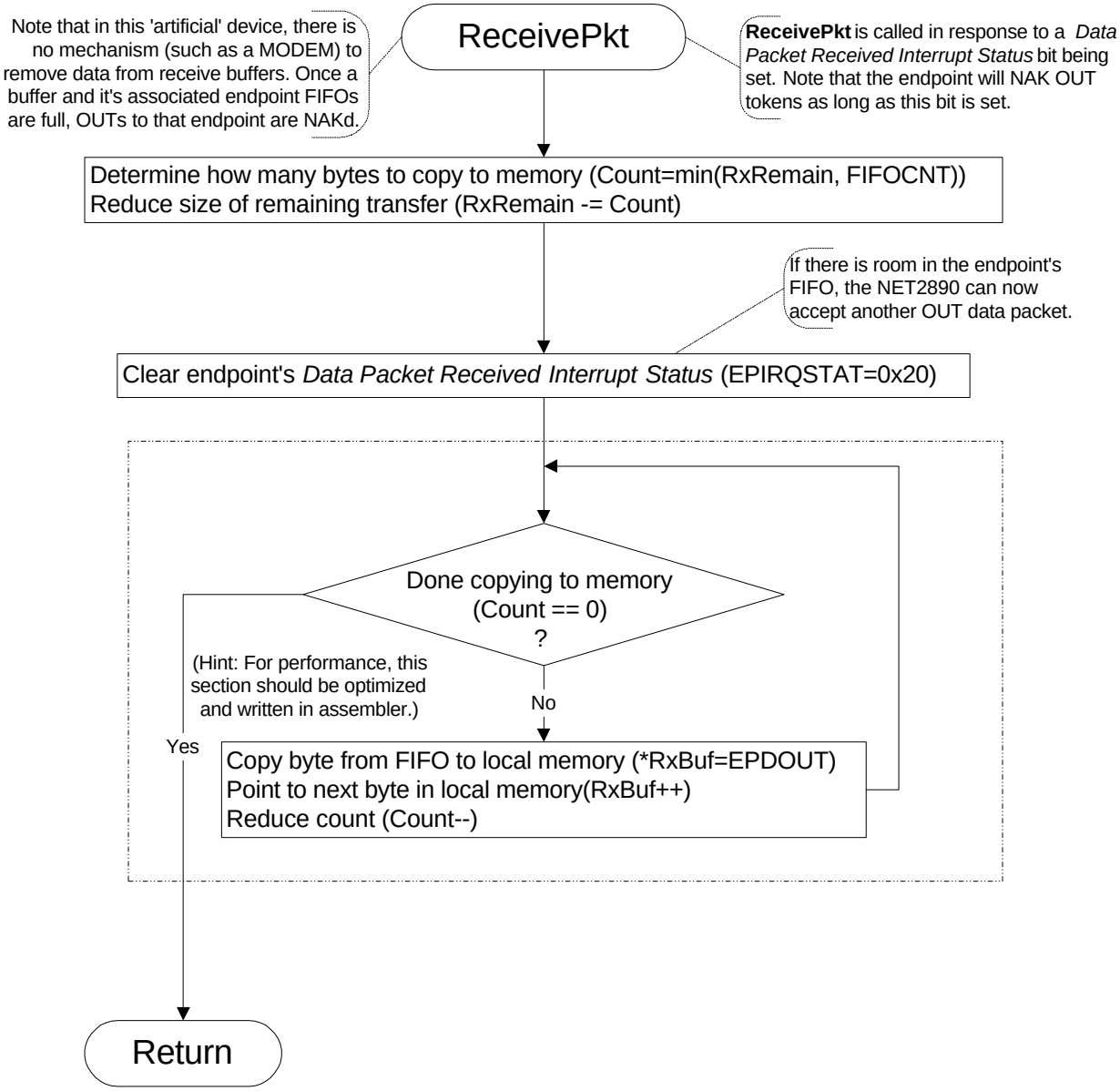


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

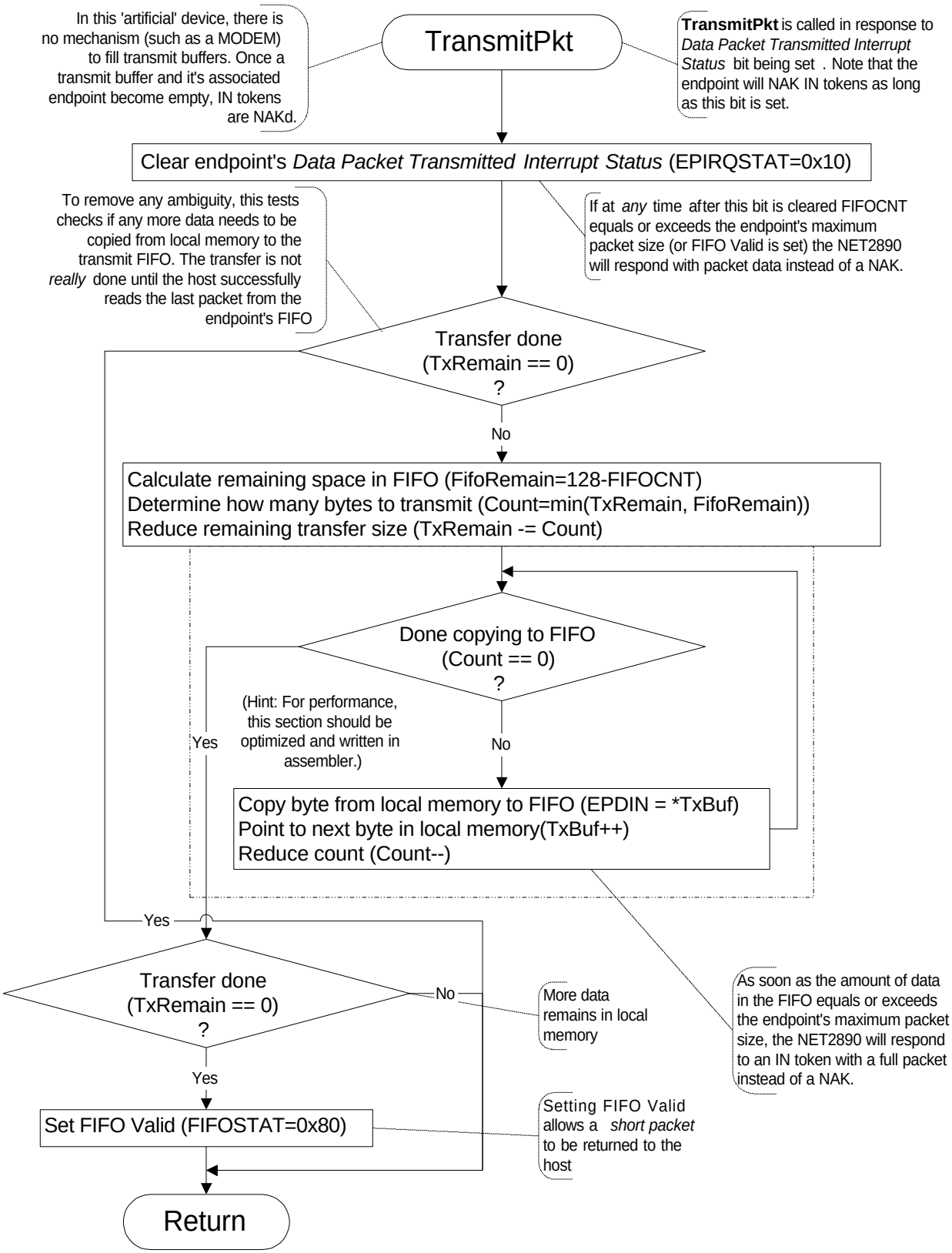




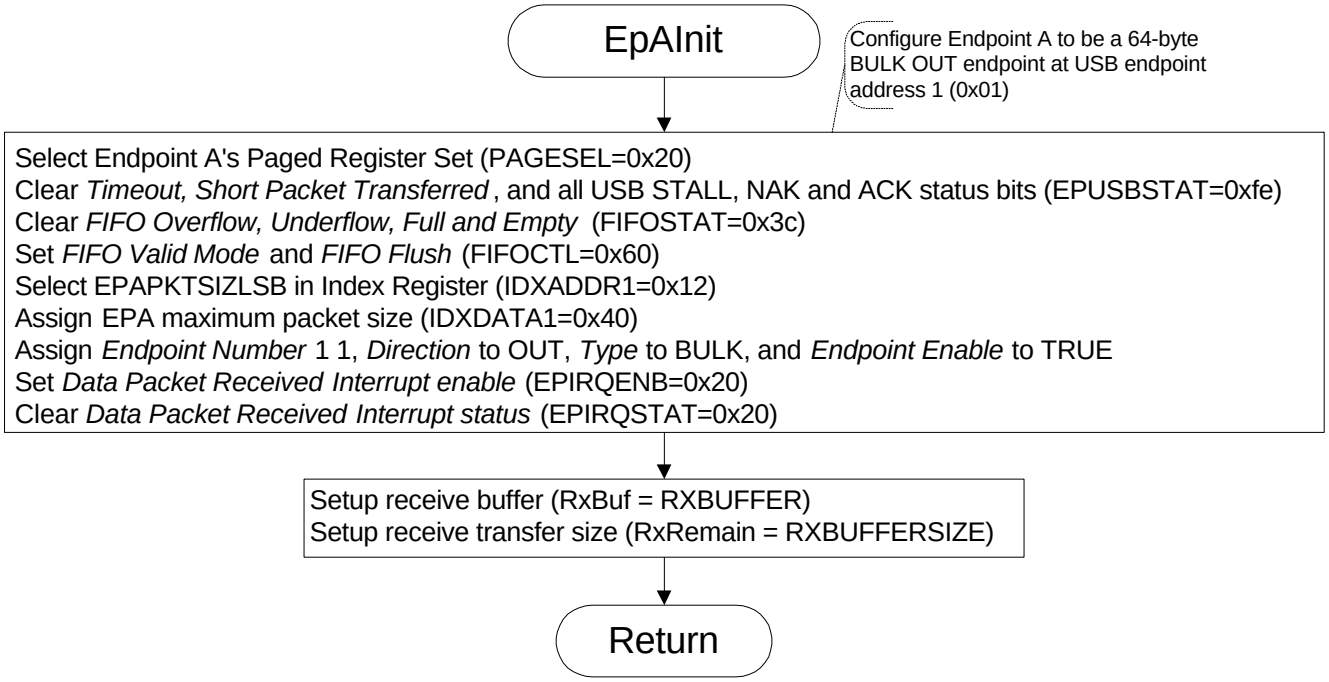
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

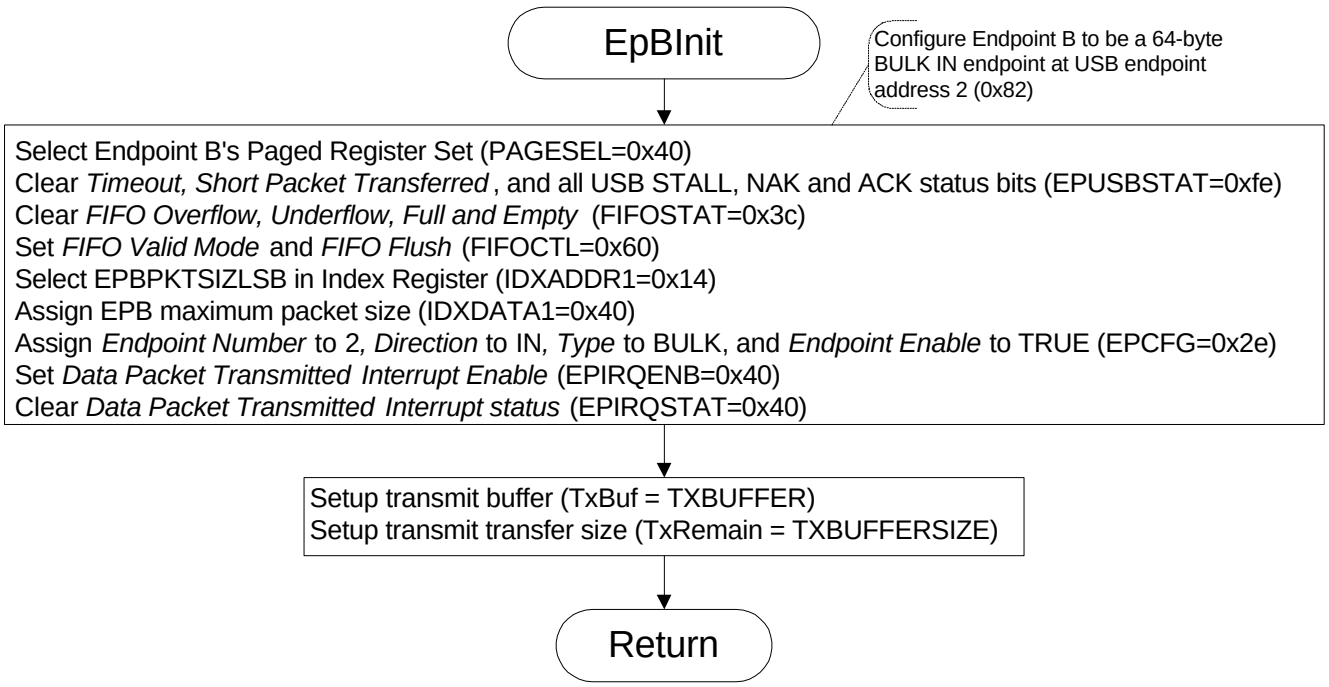


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com



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


EpAInit
Configure Endpoint A to be a 64-byte BULK OUT endpoint at USB endpoint address 1 (0x01)
Select Endpoint A's Paged Register Set (PAGESEL=0x20)
Clear *Timeout, Short Packet Transferred*, and all USB STALL, NAK and ACK status bits (EPUSBSTAT=0xfe)
Clear *FIFO Overflow, Underflow, Full and Empty* (FIFOSTAT=0x3c)
Set *FIFO Valid Mode* and *FIFO Flush* (FIFOCTL=0x60)
Select EPAPKTSIZLSB in Index Register (IDXADDR1=0x12)
Assign EPA maximum packet size (IDXDATA1=0x40)
Assign *Endpoint Number* 1 1, *Direction* to OUT, *Type* to BULK, and *Endpoint Enable* to TRUE
Set *Data Packet Received Interrupt enable* (EPIRQENB=0x20)
Clear *Data Packet Received Interrupt status* (EPIRQSTAT=0x20)
Setup receive buffer (RxBuf = RXBUFFER)
Setup receive transfer size (RxRemain = RXBUFFERSIZE)
Return


EpBInit
Configure Endpoint B to be a 64-byte BULK IN endpoint at USB endpoint address 2 (0x82)
Select Endpoint B's Paged Register Set (PAGESEL=0x40)
Clear *Timeout, Short Packet Transferred*, and all USB STALL, NAK and ACK status bits (EPUSBSTAT=0xfe)
Clear *FIFO Overflow, Underflow, Full and Empty* (FIFOSTAT=0x3c)
Set *FIFO Valid Mode* and *FIFO Flush* (FIFOCTL=0x60)
Select EPBPKTSIZLSB in Index Register (IDXADDR1=0x14)
Assign EPB maximum packet size (IDXDATA1=0x40)
Assign *Endpoint Number* to 2*, Direction* to IN*, Type* to BULK, and *Endpoint Enable* to TRUE (EPCFG=0x2e)
Set *Data Packet Transmitted Interrupt Enable* (EPIRQENB=0x40)
Clear *Data Packet Transmitted Interrupt status* (EPIRQSTAT=0x40)
Setup transmit buffer (TxBuf = TXBUFFER)
Setup transmit transfer size (TxRemain = TXBUFFERSIZE)
Return

Sample Enumeration Sequences

Note: The following enumeration sequences were recorded from a working NET2890-EB evaluation board with a Windows98 host. These sequences are provided as examples only.

No Driver installed in host

Reset
Get Device Descriptor - 64 bytes
Reset
Set Address
Get Device Descriptor - 18 bytes

Get Configuration Descriptor - 255 bytes
Get String Descriptor - 255 bytes
DialogBox installation wizard pop-up

NcUsb.sys installed in host

Reset
Get Device Descriptor - 64 bytes
Reset
Set Address
Get Device Descriptor - 18 bytes

Get Configuration Descriptor - 255 bytes
Get Device Descriptor - 18 bytes
Get Configuration Descriptor - 265 bytes
Set Configuration

USB Compatibility Ch. 9 Test
Reset Get Device Descriptor - 64 bytes Reset Set Address Get Device Descriptor - 18 bytes Get Configuration Descriptor - 9 bytes Get Configuration Descriptor - 255 bytes Get Device Descriptor - 18 bytes