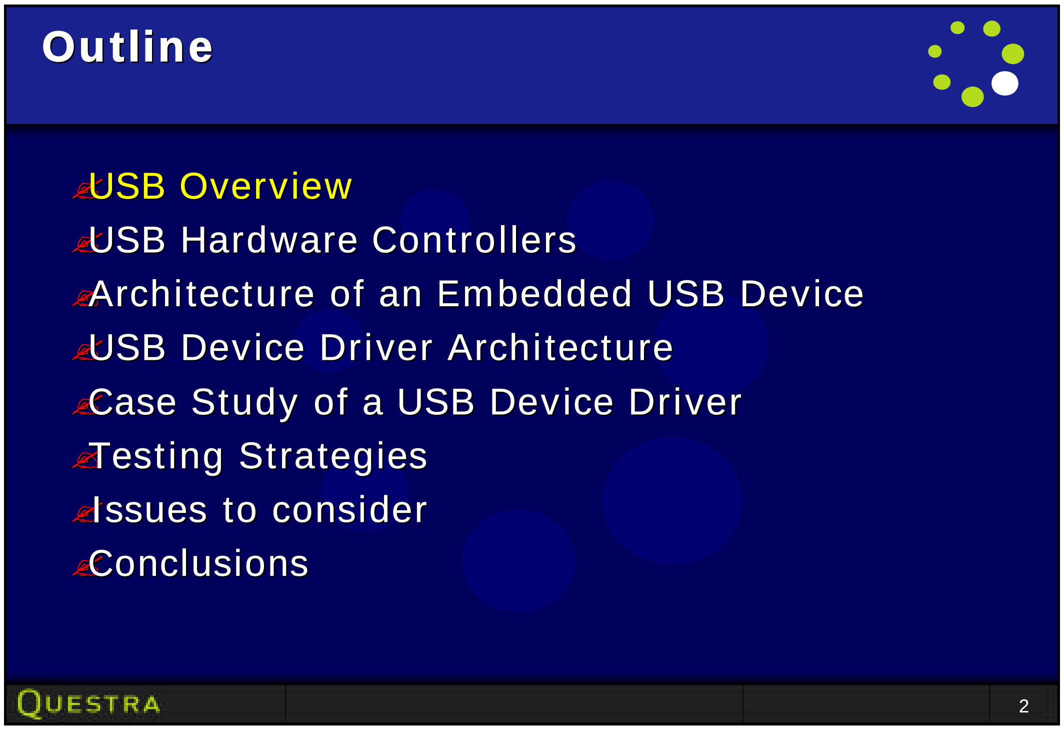
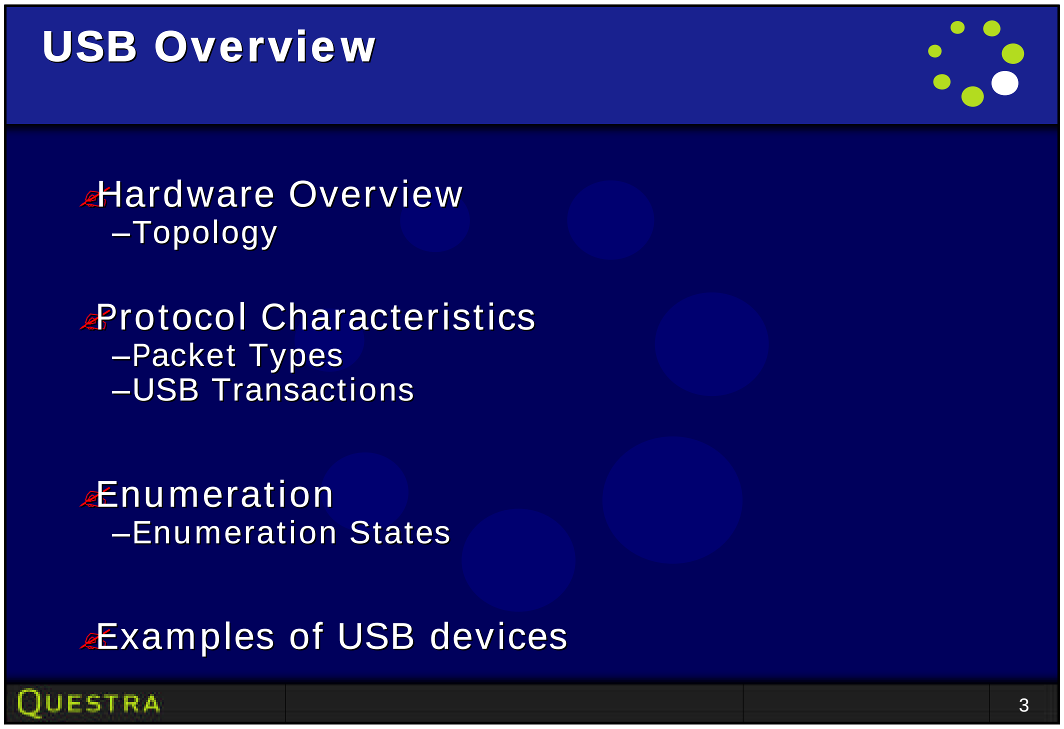
Design of a USB Device Driver

Joe Flynn
Questra Corporation
jflynn@questra.com
(716)381-0260



Outline

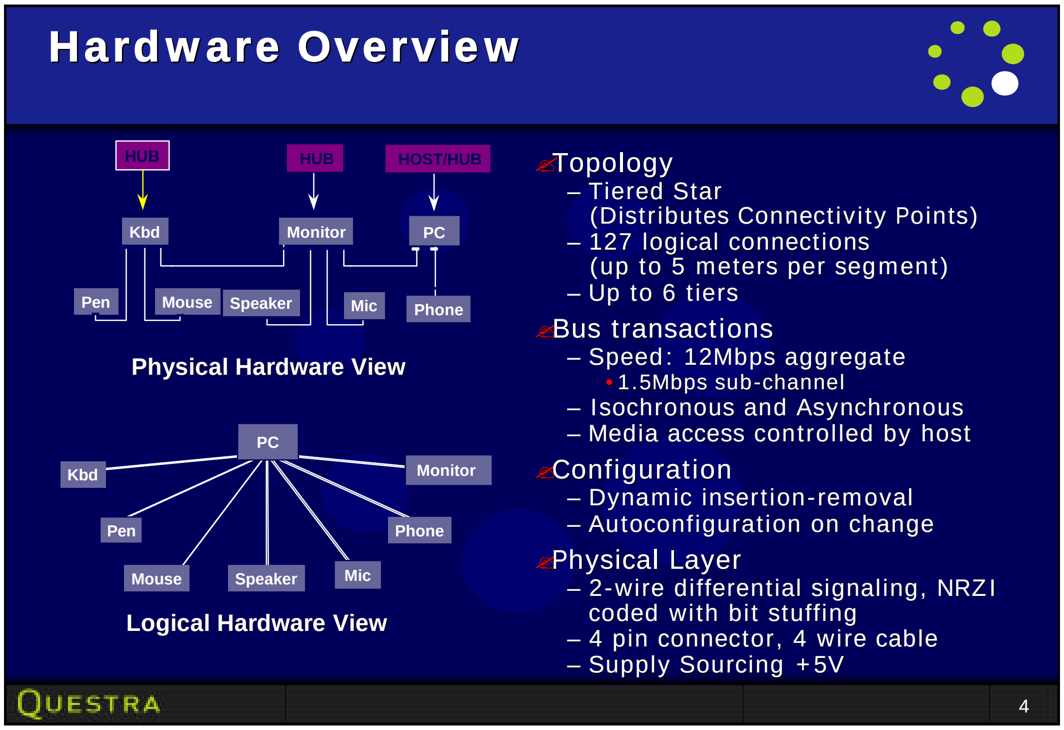
- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions



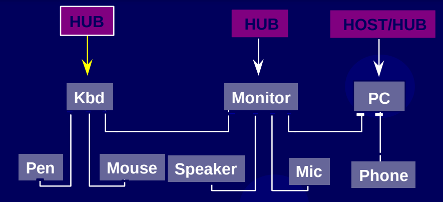
USB Overview

- Hardware Overview
  - –Topology
- Protocol Characteristics
  - –Packet Types
  - –USB Transactions
- Enumeration
  - –Enumeration States
- Examples of USB devices

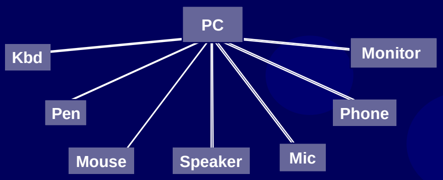
QUESTRA 3



Hardware Overview



Physical Hardware View



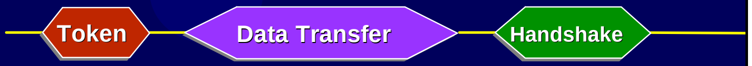
Logical Hardware View

- Topology
  - – Tiered Star
    (Distributes Connectivity Points)
  - – 127 logical connections
    (up to 5 meters per segment)
  - – Up to 6 tiers
- Bus transactions
  - – Speed: 12Mbps aggregate
    - 1.5Mbps sub-channel
  - – Isochronous and Asynchronous
  - – Media access controlled by host
- Configuration
  - – Dynamic insertion-removal
  - – Autoconfiguration on change
- Physical Layer
  - – 2-wire differential signaling, NRZI
    coded with bit stuffing
  - – 4 pin connector, 4 wire cable
  - – Supply Sourcing +5V

QUESTRA 4

The Transaction Protocol is Host Based

- Host based token polling
  - Data from host-to-function and function-to-host
  - Host handles most of the protocol complexity
  - Peripheral design is simple and low-cost
- Robustness
  - Handshake to acknowledge data transfer and flow control
  - Very low raw physical bit error rate ($<10^{-10}$)
  - CRC protection plus hardware retry option
  - Data Toggle Sequence bits
- Bounded transfer characteristics
  - Data transfer bandwidth and latency prenegotiated
  - Flow control for peripheral buffer management



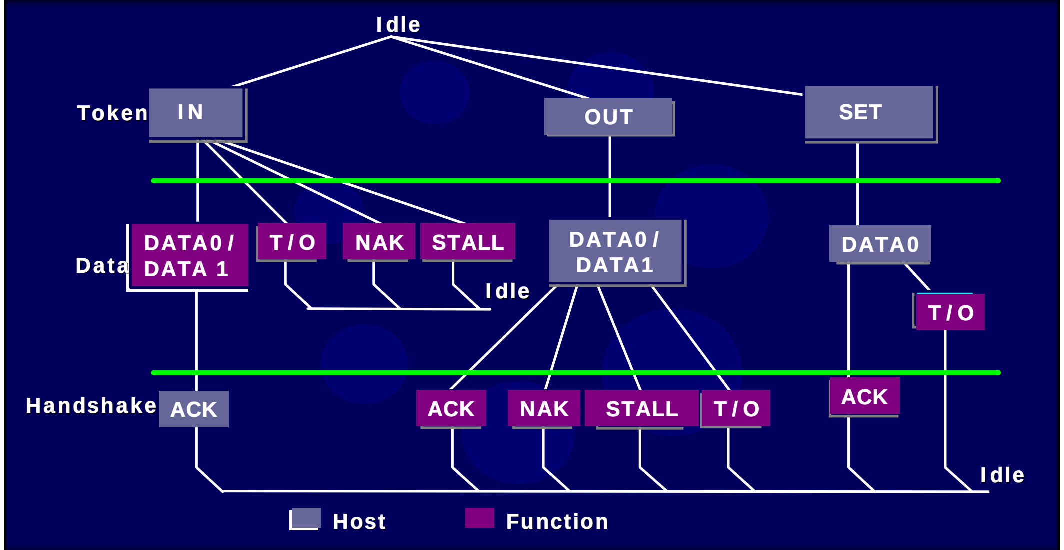
QUESTRA 5

Packet Types

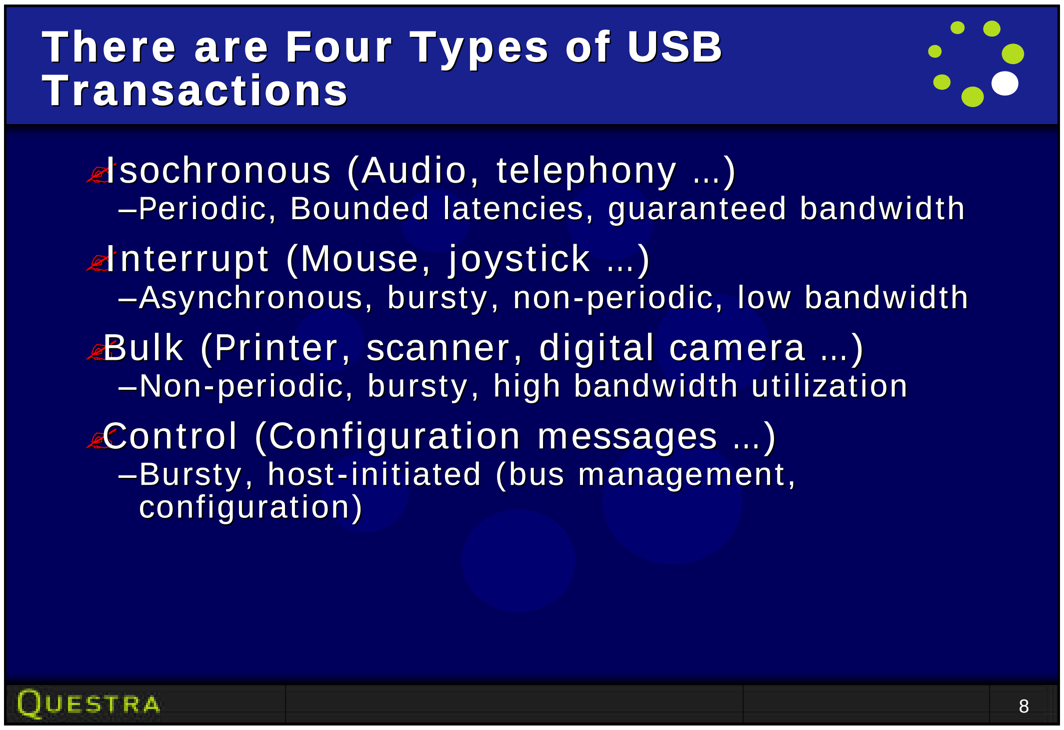
- Token - OUT, IN, SOF, SETUP
  - First packet in any transaction
  - Specifies function address, endpoint
  - Specifies data direction
- Data - DATA0, DATA1
  - 0 - 1023 bytes
- Handshake - ACK, NAK, STALL
  - Report status of data transaction
  - Flow control
  - Stall conditions
- Special - PRE
  - Enables Hub for low speed communications

QUESTRA 6

A Typical USB Transaction Consists of Three Packets



QUESTRA 7



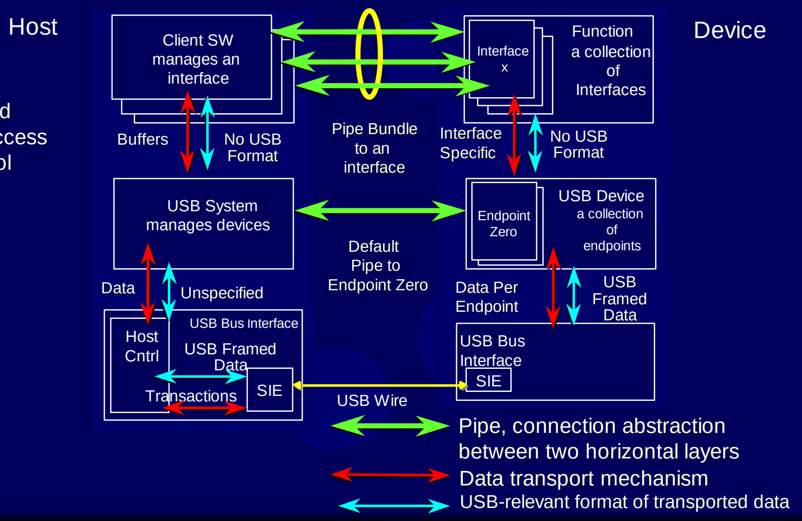
There are Four Types of USB Transactions

- Isochronous (Audio, telephony ...)
  - Periodic, Bounded latencies, guaranteed bandwidth
- Interrupt (Mouse, joystick ...)
  - Asynchronous, bursty, non-periodic, low bandwidth
- Bulk (Printer, scanner, digital camera ...)
  - Non-periodic, bursty, high bandwidth utilization
- Control (Configuration messages ...)
  - Bursty, host-initiated (bus management, configuration)

QUESTRA 8

The Basic USB Model has Several Layers of Abstraction

Endpoint 0
- Required, shared
- Configuration access
- Capability control

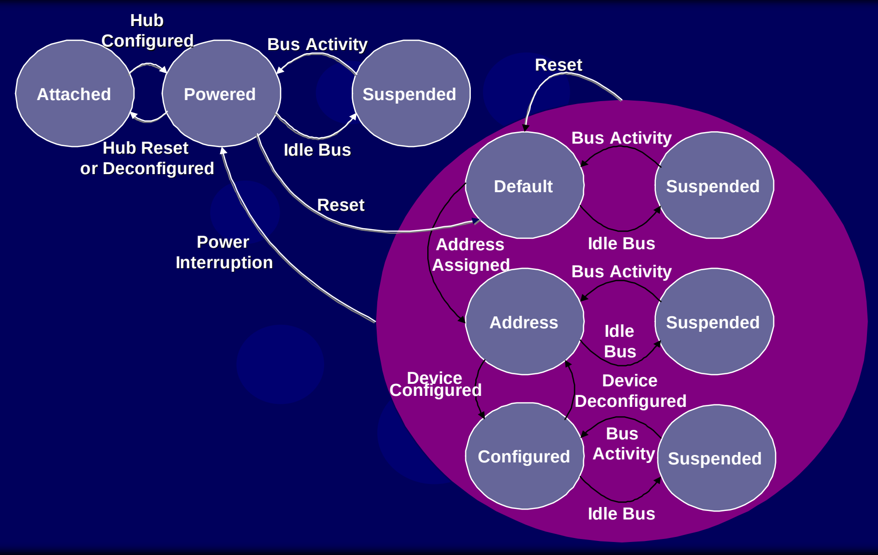


Host
Device
Client SW manages an interface
Buffers
No USB Format
Pipe Bundle to an interface
Interface x
Function a collection of Interfaces
Interface Specific
No USB Format
USB System manages devices
Default Pipe to Endpoint Zero
Endpoint Zero
USB Device a collection of endpoints
Data
Unspecified
Data Per Endpoint
USB Framed Data
Host Cntrl
USB Bus Interface
USB Framed Data
Transactions
SIE
USB Wire
USB Bus Interface
SIE
Pipe, connection abstraction between two horizontal layers
Data transport mechanism
USB-relevant format of transported data

Enumeration: Device perspective

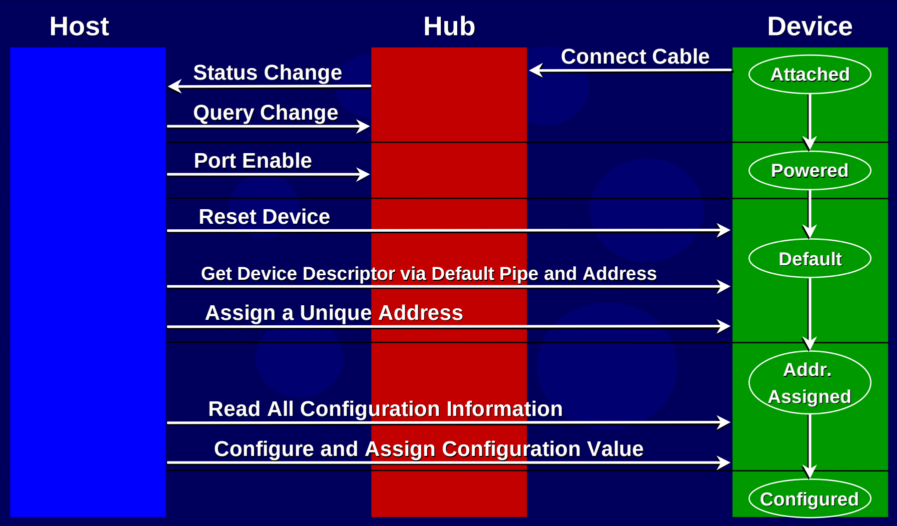
- Attached State
  - –Entered by attaching USB Cable
- Powered State
  - –USB Host Applies power
- Default state
  - –USB Host resets bus
- Addressed State
  - –USB Host sends Set Address with non-zero address
- Configured state
  - –USB Host sends Set Configuration with non-zero value
- Suspended state
  - –USB Host stops sending SOF for 3 msec

The Device State Machine

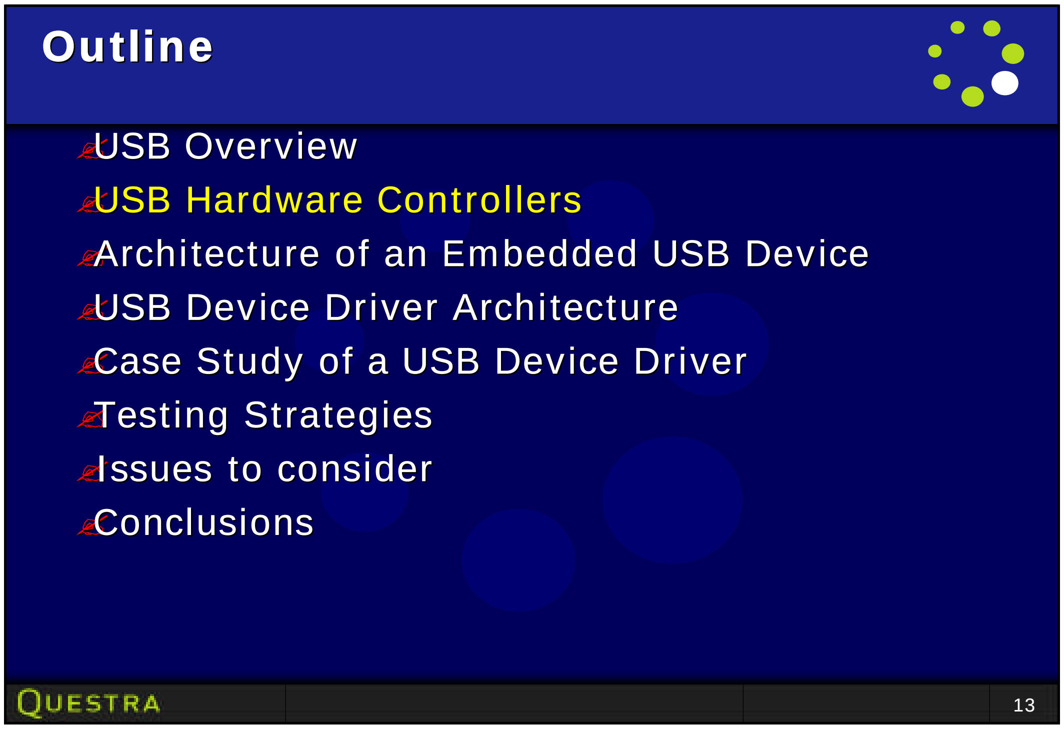


QUESTRA 11

Enumeration is the Process of Assigning Addresses and Setting Configurations



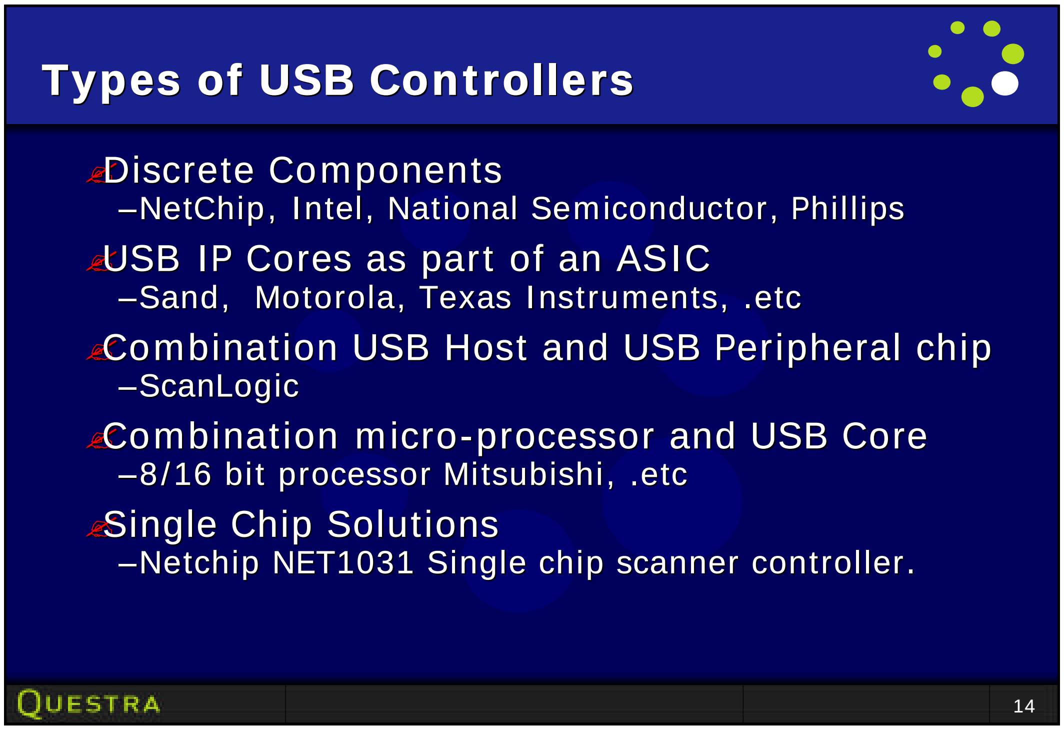
QUESTRA 12



Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

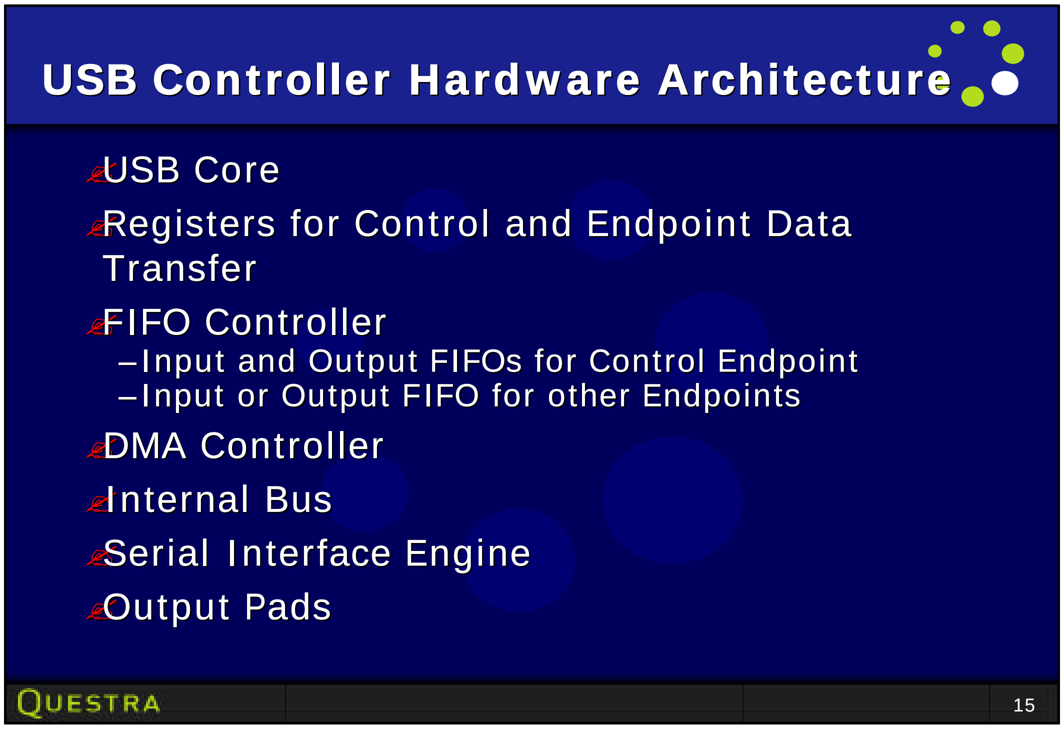
QUESTRA 13



Types of USB Controllers

- Discrete Components
  - NetChip, Intel, National Semiconductor, Phillips
- USB IP Cores as part of an ASIC
  - Sand, Motorola, Texas Instruments, .etc
- Combination USB Host and USB Peripheral chip
  - ScanLogic
- Combination micro-processor and USB Core
  - 8/16 bit processor Mitsubishi, .etc
- Single Chip Solutions
  - Netchip NET1031 Single chip scanner controller.

QUESTRA 14

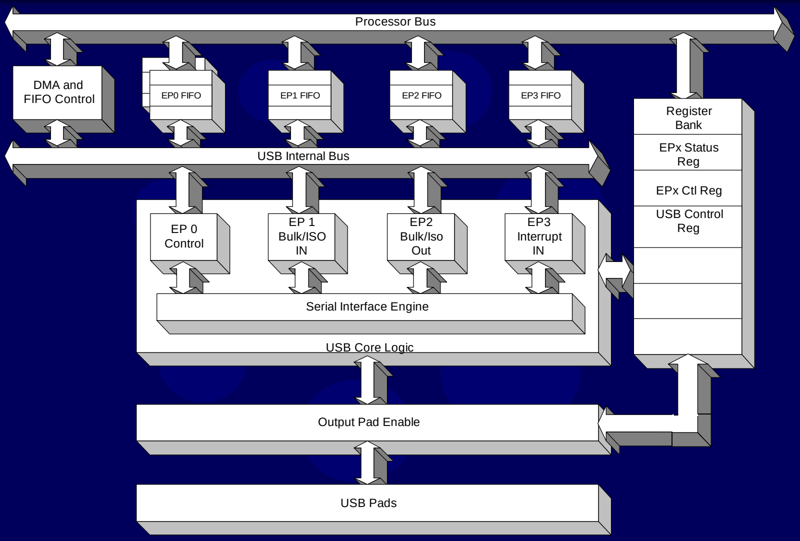


USB Controller Hardware Architecture

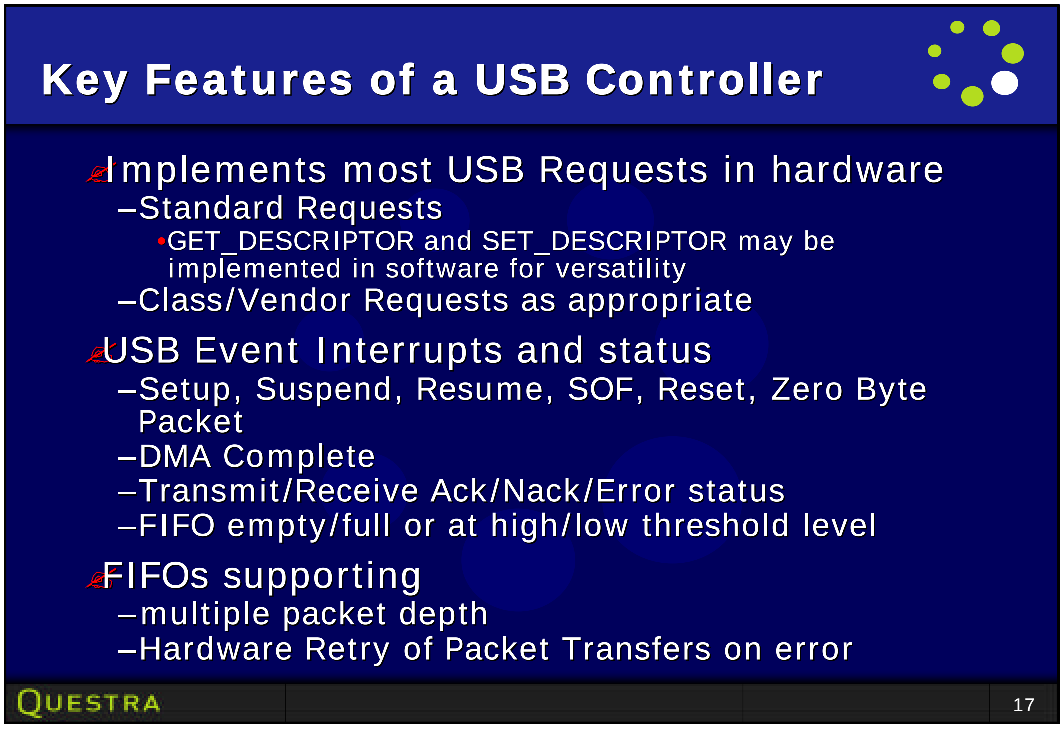
- USB Core
- Registers for Control and Endpoint Data Transfer
- FIFO Controller
  - Input and Output FIFOs for Control Endpoint
  - Input or Output FIFO for other Endpoints
- DMA Controller
- Internal Bus
- Serial Interface Engine
- Output Pads

QUESTRA 15

Example of USB Controller



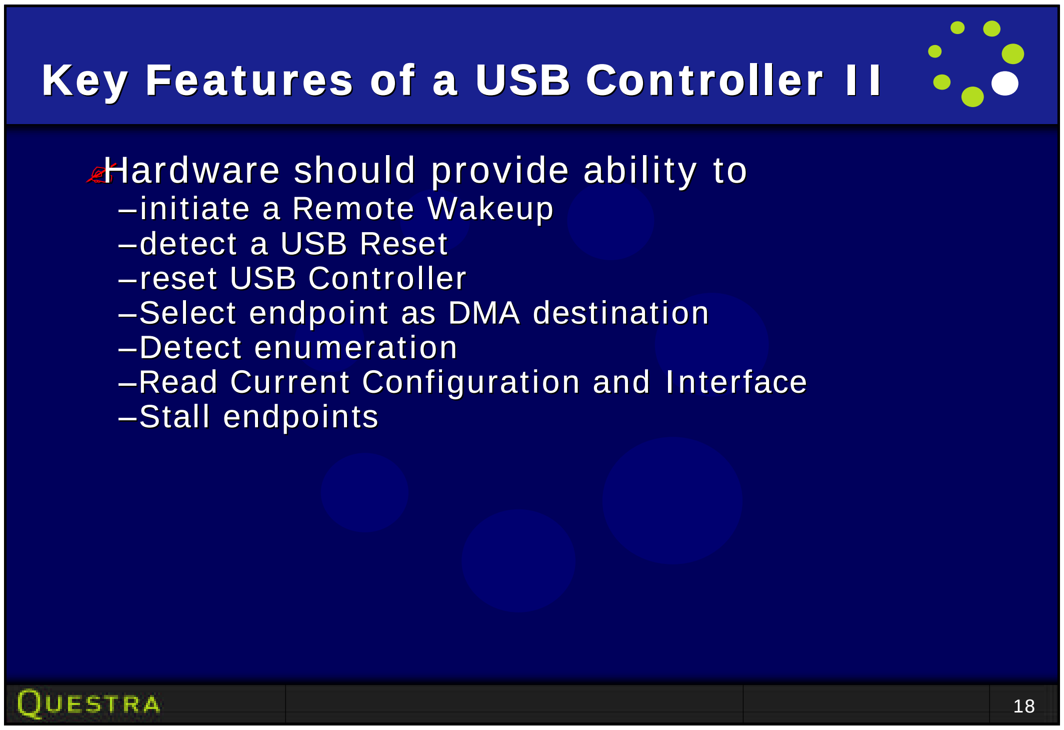
QUESTRA 16



Key Features of a USB Controller

- Implements most USB Requests in hardware
  - –Standard Requests
    - •GET_DESCRIPTOR and SET_DESCRIPTOR may be implemented in software for versatility
  - –Class/Vendor Requests as appropriate
- USB Event Interrupts and status
  - –Setup, Suspend, Resume, SOF, Reset, Zero Byte Packet
  - –DMA Complete
  - –Transmit/Receive Ack/Nack/Error status
  - –FIFO empty/full or at high/low threshold level
- FIFOs supporting
  - –multiple packet depth
  - –Hardware Retry of Packet Transfers on error

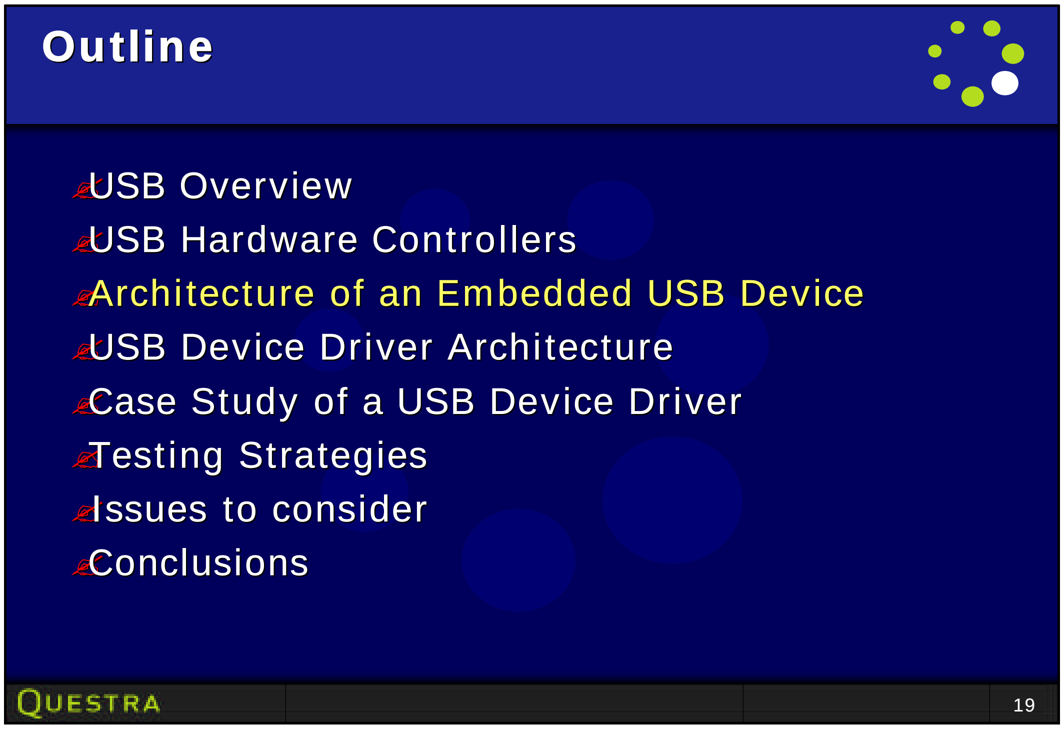
QUESTRA 17



Key Features of a USB Controller II

- Hardware should provide ability to
  - –initiate a Remote Wakeup
  - –detect a USB Reset
  - –reset USB Controller
  - –Select endpoint as DMA destination
  - –Detect enumeration
  - –Read Current Configuration and Interface
  - –Stall endpoints

QUESTRA 18

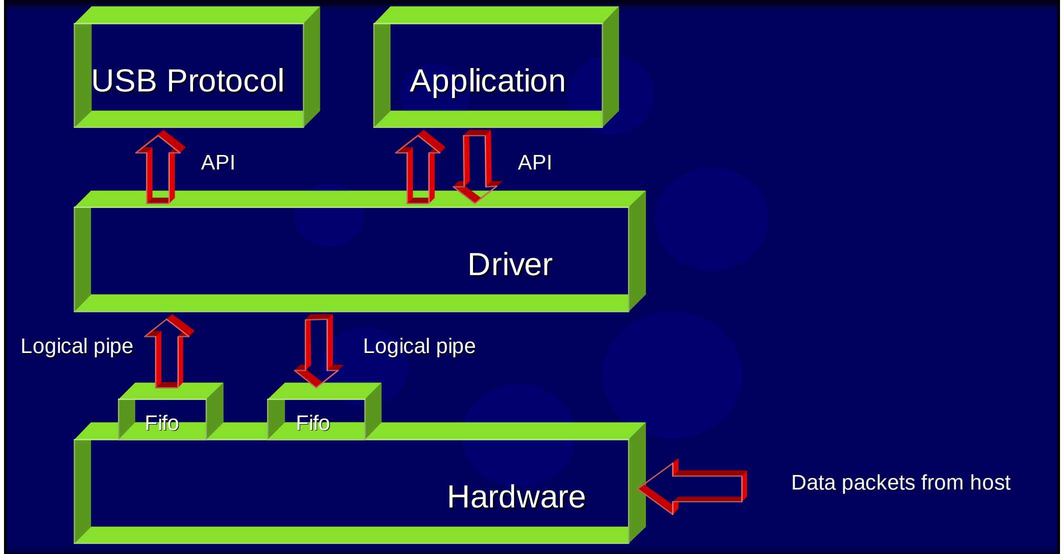


Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

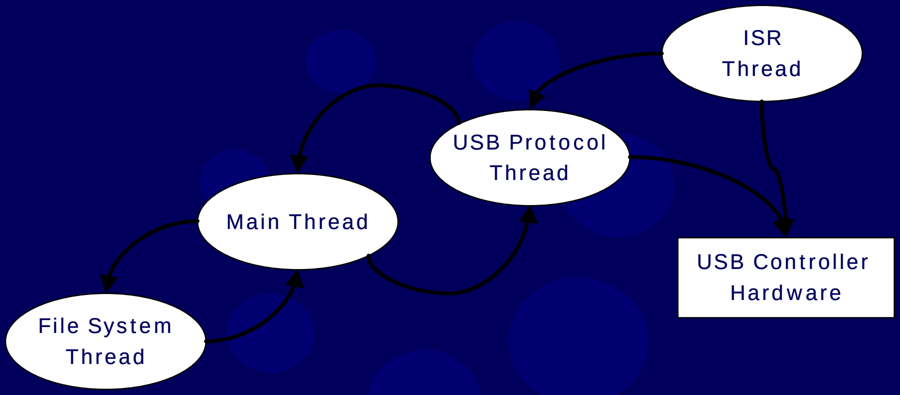
QUESTRA 19

Architecture of an Embedded USB Device



QUESTRA 20

System Architecture

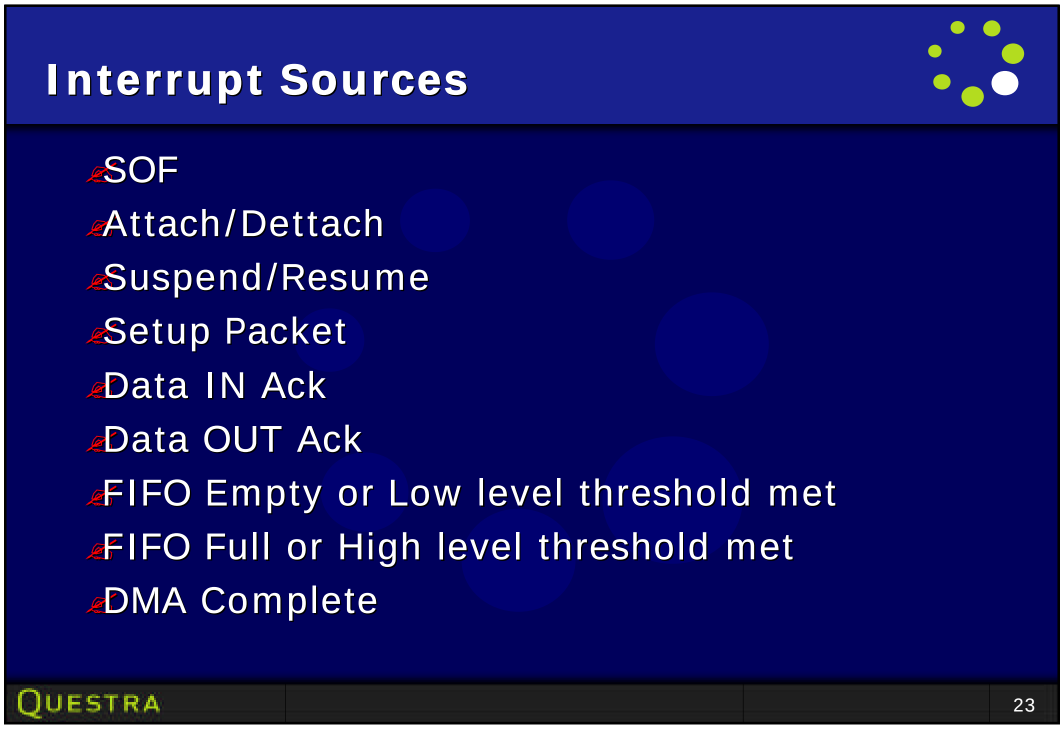


QUESTRA 21

USB Peripheral Threads

- ISR Thread
  - Low Level Interrupt Service routine(s)
    - USB Controller Interrupt
    - DMA Controller Interrupt
- USB Protocol Thread
  - Task which implements USB Protocol
    - Control, Bulk, Isochrnous, Interrupt Endpoints
    - Attach/Dettach, SOF, Suspend/Resume
- Main Thread
  - Thread which executes the product application
  - Calls and is triggered by Callback from USB Driver layer
- File System Thread
  - Lower Priority File System Thread

QUESTRA 22

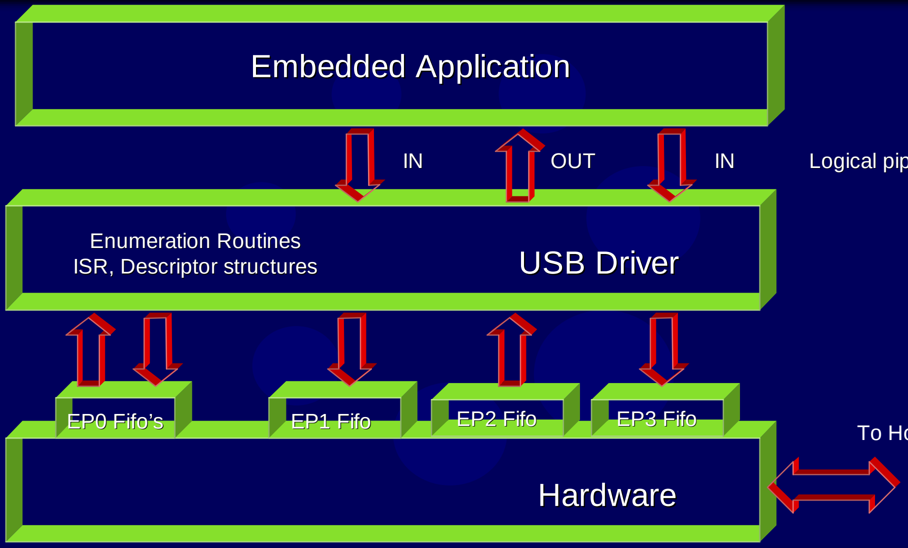


Interrupt Sources

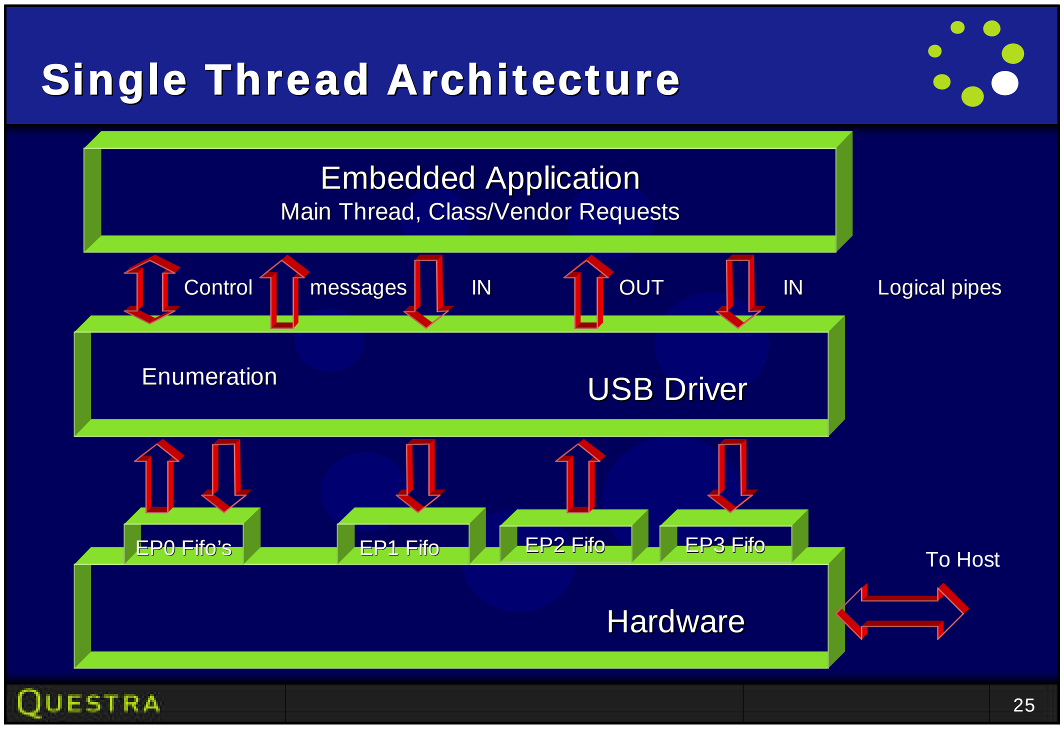
- SOF
- Attach/Dettach
- Suspend/Resume
- Setup Packet
- Data IN Ack
- Data OUT Ack
- FIFO Empty or Low level threshold met
- FIFO Full or High level threshold met
- DMA Complete

QUESTRA 23

Driver/Hardware Enumeration Architecture

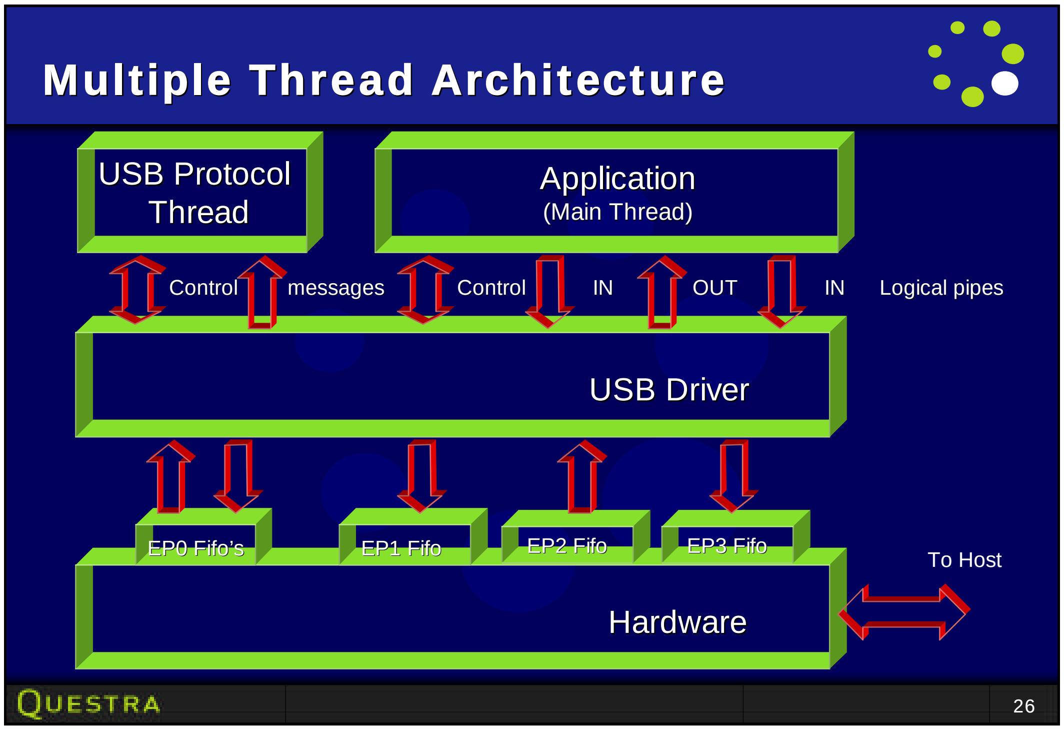


QUESTRA 24



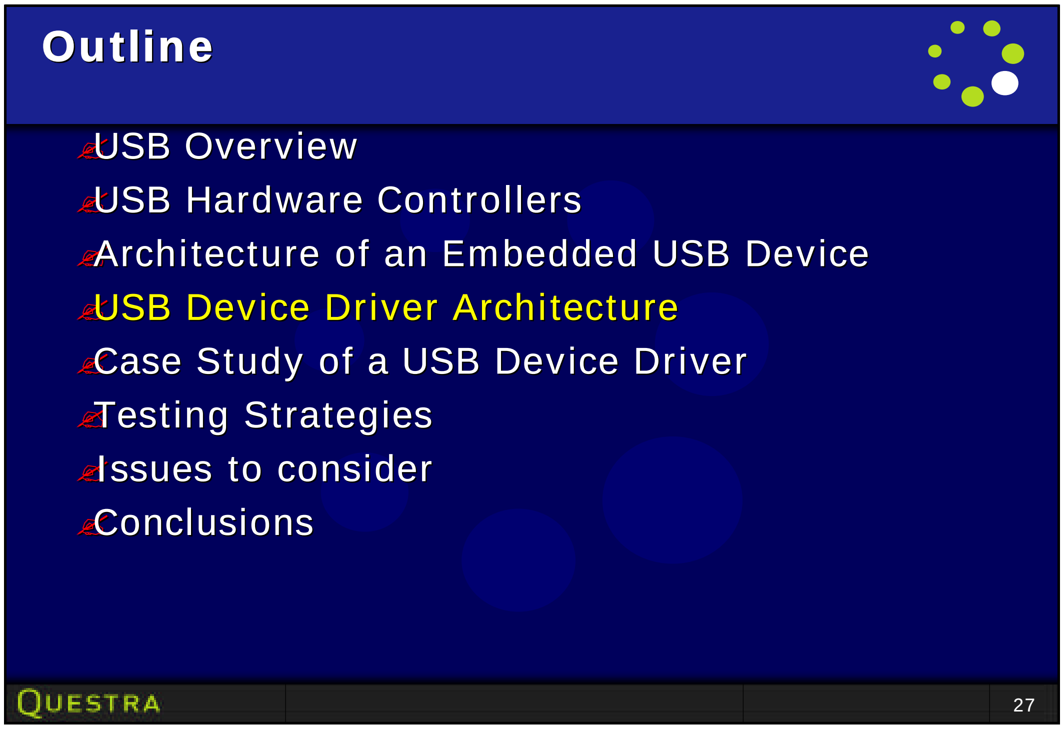
Single Thread Architecture

QUESTRA 25



Multiple Thread Architecture

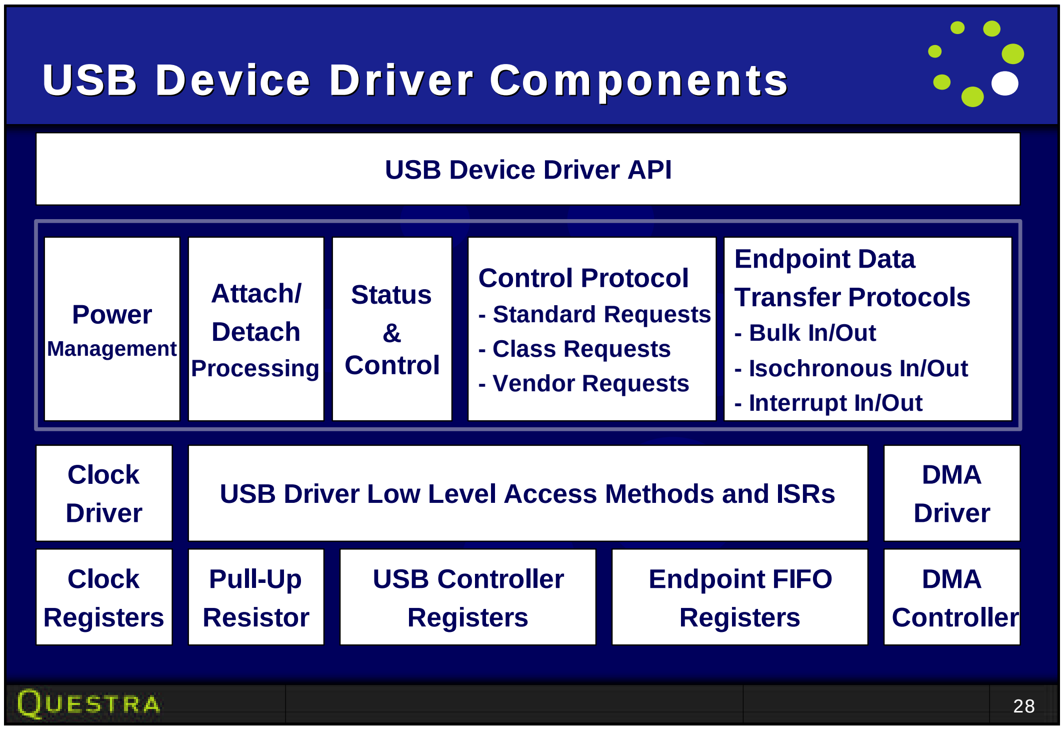
QUESTRA 26



Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

QUESTRA 27



USB Device Driver Components

USB Device Driver API

Power Management

Attach/ Detach Processing

Status & Control

Control Protocol
- Standard Requests
- Class Requests
- Vendor Requests

Endpoint Data Transfer Protocols
- Bulk In/Out
- Isochronous In/Out
- Interrupt In/Out

Clock Driver

USB Driver Low Level Access Methods and ISRs

DMA Driver

Clock Registers

Pull-Up Resistor

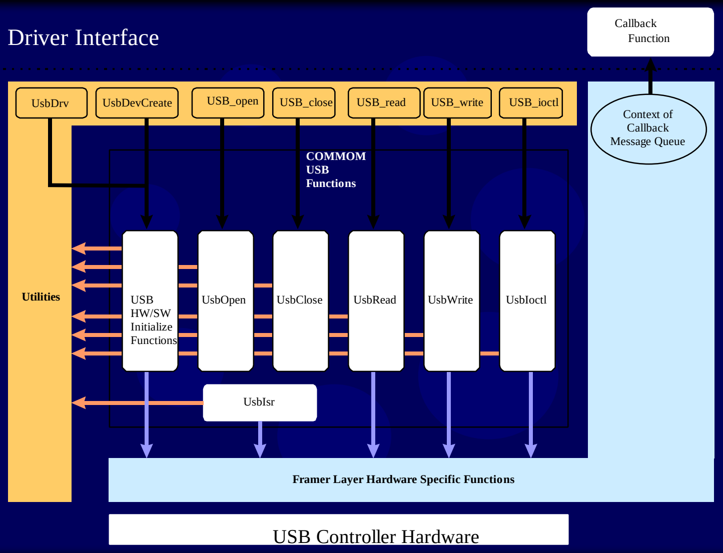
USB Controller Registers

Endpoint FIFO Registers

DMA Controller

QUESTRA 28

USB Device Driver Architecture



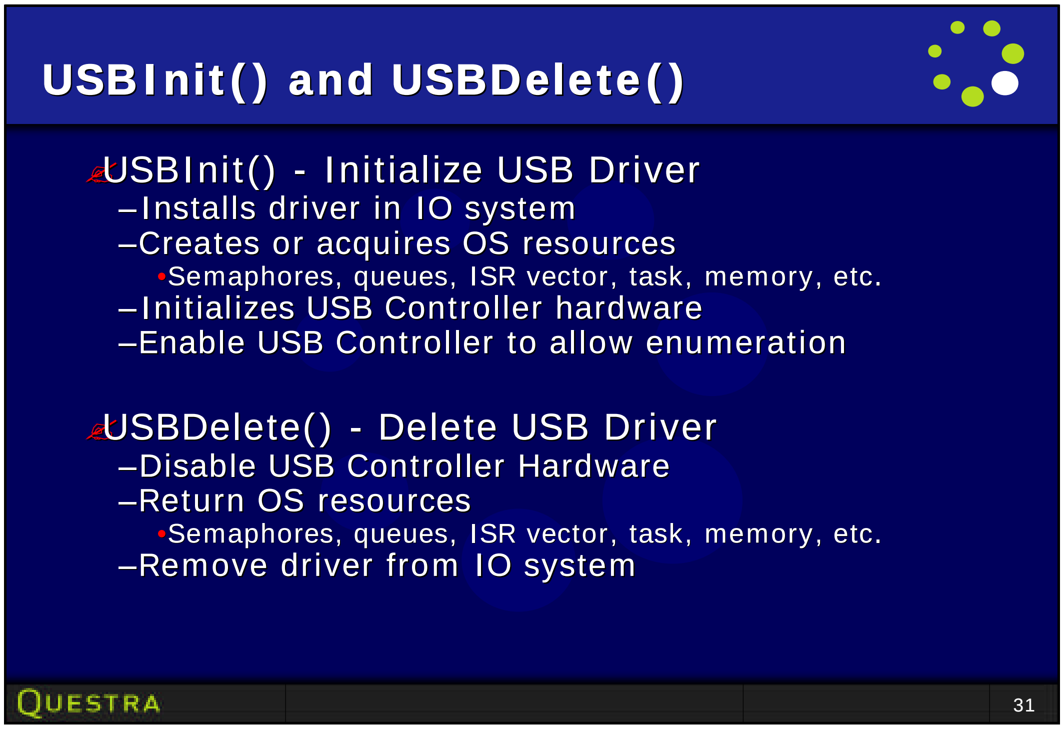
QUESTRA

USB Device Driver API

- USBInit()
- USBDelete()
- USBOpen()
- USBClose()
- USBRead()
- USBWrite()
- USBIoctl()

- Callback Message Queue

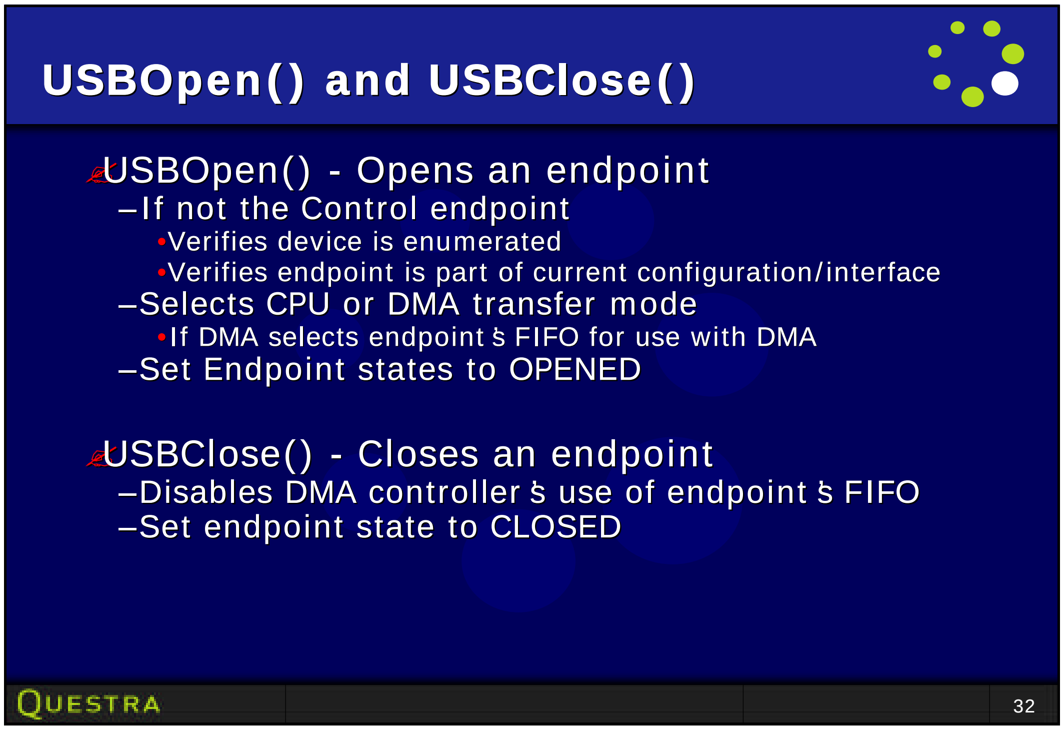
QUESTRA



USBInit() and USBDelete()

- USBInit() - Initialize USB Driver
  - –Installs driver in IO system
  - –Creates or acquires OS resources
    - •Semaphores, queues, ISR vector, task, memory, etc.
  - –Initializes USB Controller hardware
  - –Enable USB Controller to allow enumeration
- USBDelete() - Delete USB Driver
  - –Disable USB Controller Hardware
  - –Return OS resources
    - •Semaphores, queues, ISR vector, task, memory, etc.
  - –Remove driver from IO system

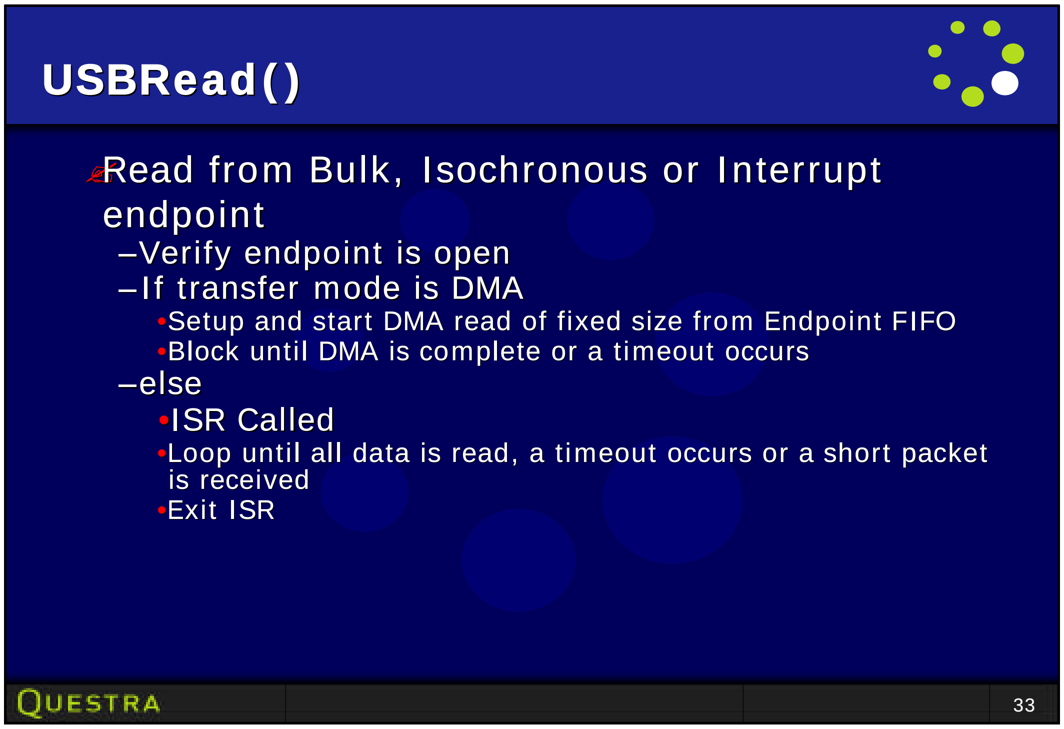
QUESTRA



USBOpen() and USBClose()

- USBOpen() - Opens an endpoint
  - –If not the Control endpoint
    - •Verifies device is enumerated
    - •Verifies endpoint is part of current configuration/interface
  - –Selects CPU or DMA transfer mode
    - •If DMA selects endpoint’s FIFO for use with DMA
  - –Set Endpoint states to OPENED
- USBClose() - Closes an endpoint
  - –Disables DMA controller’s use of endpoint’s FIFO
  - –Set endpoint state to CLOSED

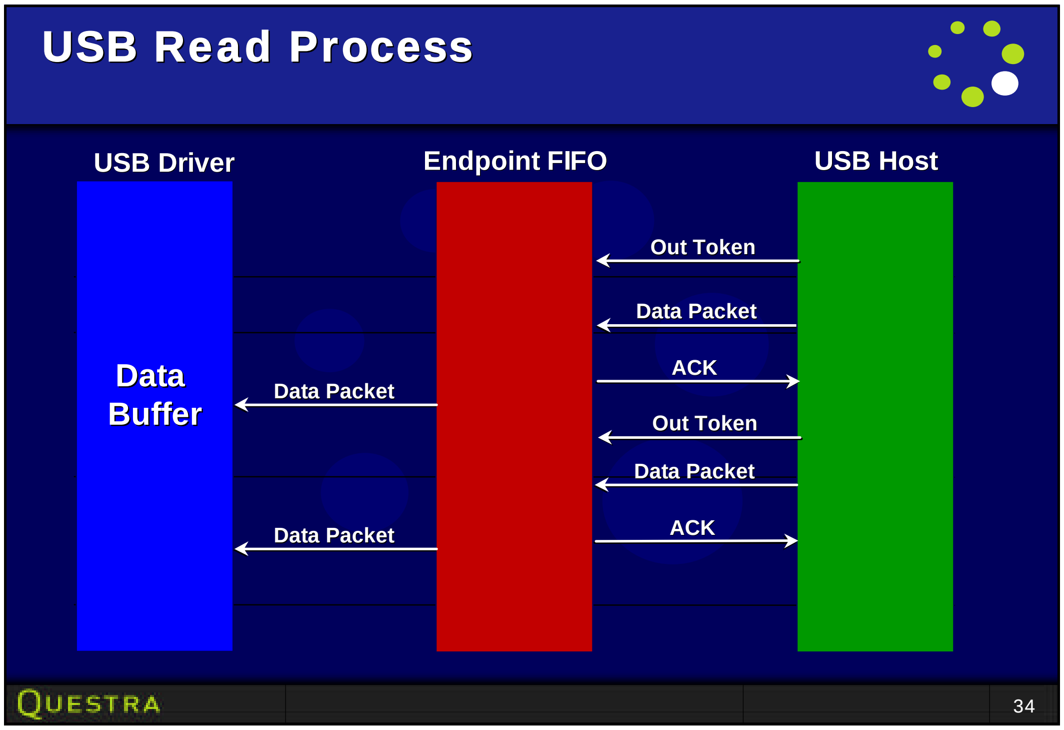
QUESTRA



USBRead()

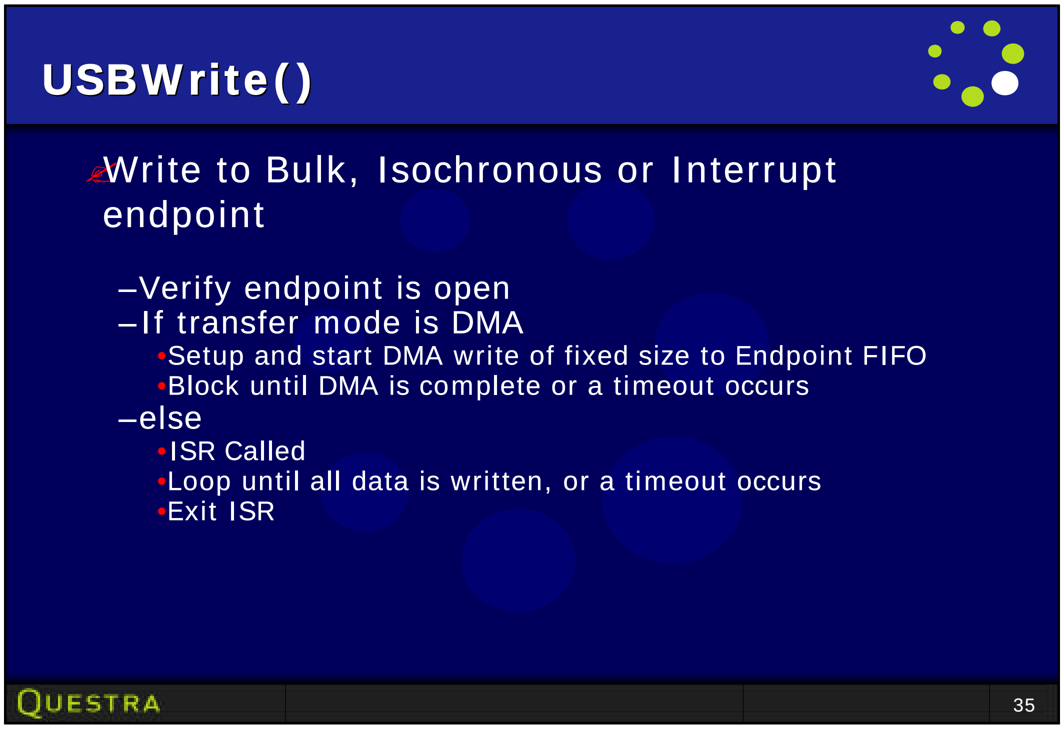
- Read from Bulk, Isochronous or Interrupt endpoint
  - –Verify endpoint is open
  - –If transfer mode is DMA
    - •Setup and start DMA read of fixed size from Endpoint FIFO
    - •Block until DMA is complete or a timeout occurs
  - –else
    - •ISR Called
    - •Loop until all data is read, a timeout occurs or a short packet is received
    - •Exit ISR

QUESTRA 33



USB Read Process

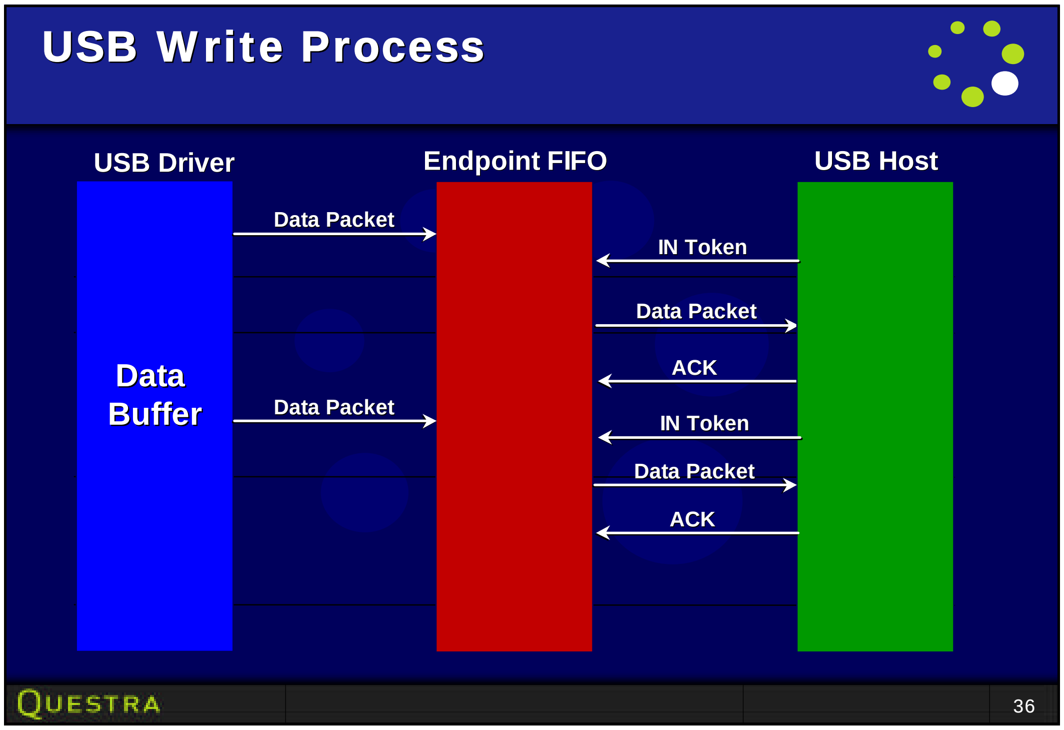
QUESTRA 34



USBWrite()

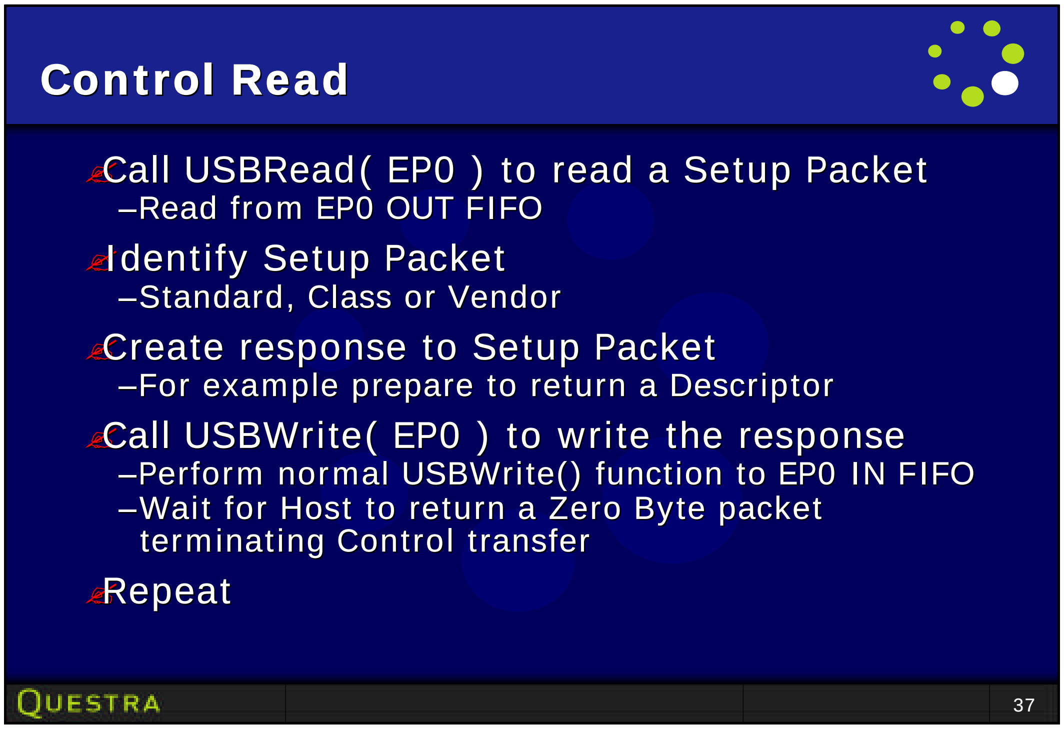
- Write to Bulk, Isochronous or Interrupt endpoint
  - –Verify endpoint is open
  - –If transfer mode is DMA
    - Setup and start DMA write of fixed size to Endpoint FIFO
    - Block until DMA is complete or a timeout occurs
  - –else
    - ISR Called
    - Loop until all data is written, or a timeout occurs
    - Exit ISR

QUESTRA 35



USB Write Process

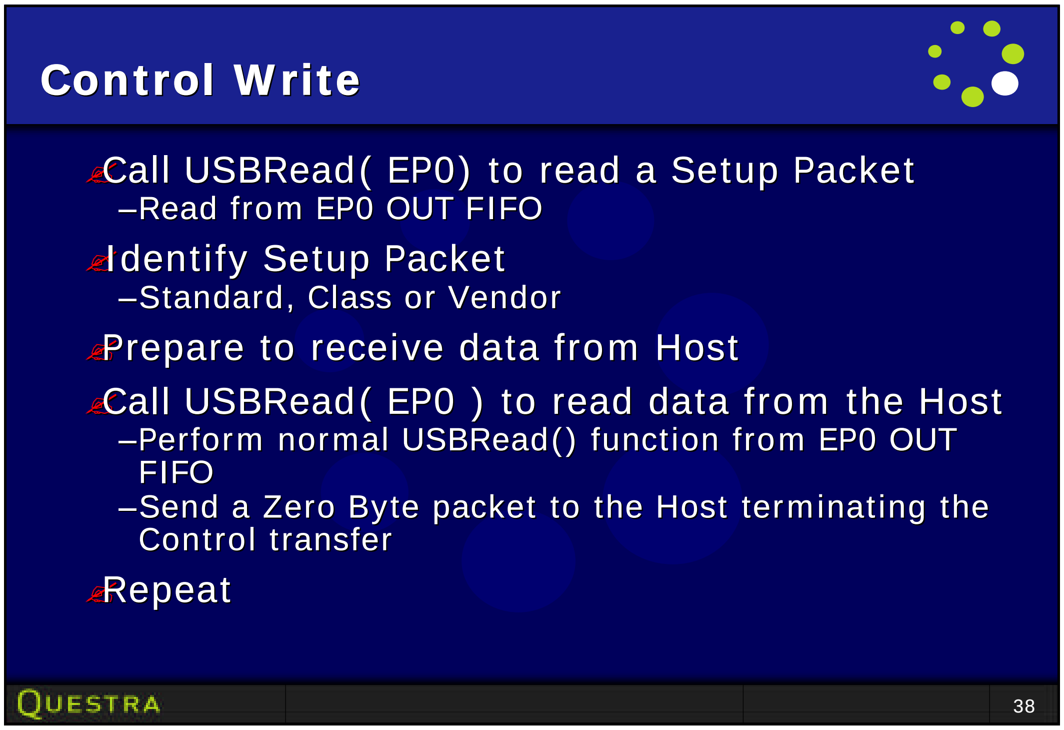
QUESTRA 36



Control Read

- Call USBRead(EP0) to read a Setup Packet
  - Read from EP0 OUT FIFO
- Identify Setup Packet
  - Standard, Class or Vendor
- Create response to Setup Packet
  - For example prepare to return a Descriptor
- Call USBWrite(EP0) to write the response
  - Perform normal USBWrite() function to EP0 IN FIFO
  - Wait for Host to return a Zero Byte packet terminating Control transfer
- Repeat

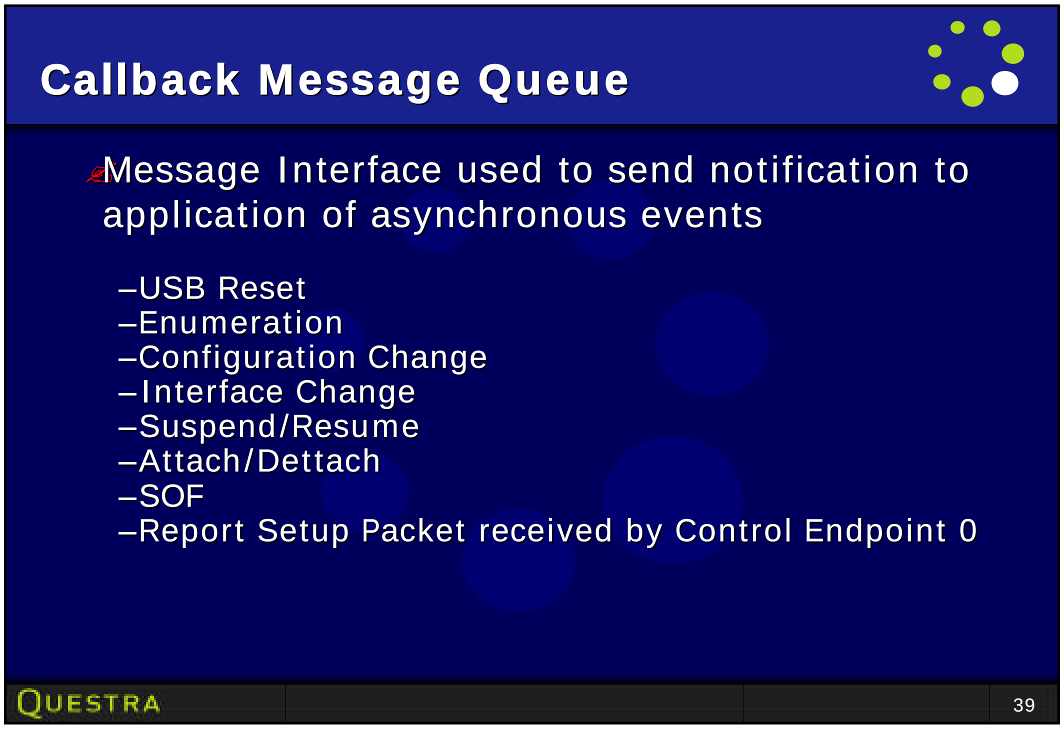
QUESTRA 37



Control Write

- Call USBRead(EP0) to read a Setup Packet
  - Read from EP0 OUT FIFO
- Identify Setup Packet
  - Standard, Class or Vendor
- Prepare to receive data from Host
- Call USBRead(EP0) to read data from the Host
  - Perform normal USBRead() function from EP0 OUT FIFO
  - Send a Zero Byte packet to the Host terminating the Control transfer
- Repeat

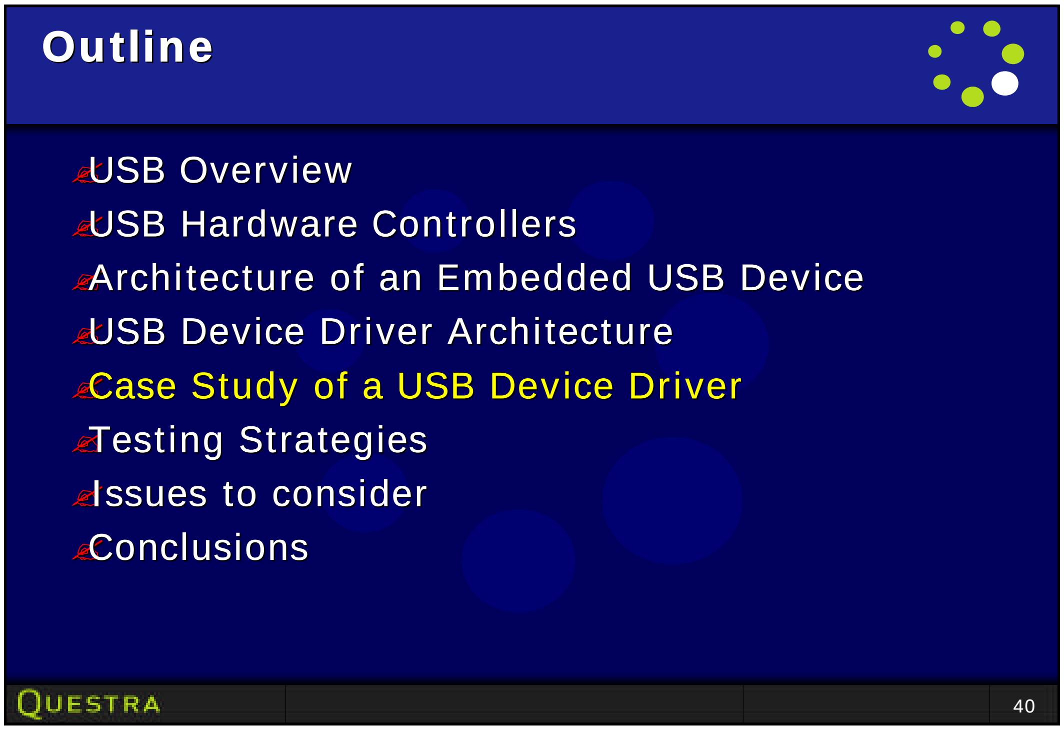
QUESTRA 38



Callback Message Queue

- Message Interface used to send notification to application of asynchronous events
  - –USB Reset
  - –Enumeration
  - –Configuration Change
  - –Interface Change
  - –Suspend/Resume
  - –Attach/Dettach
  - –SOF
  - –Report Setup Packet received by Control Endpoint 0

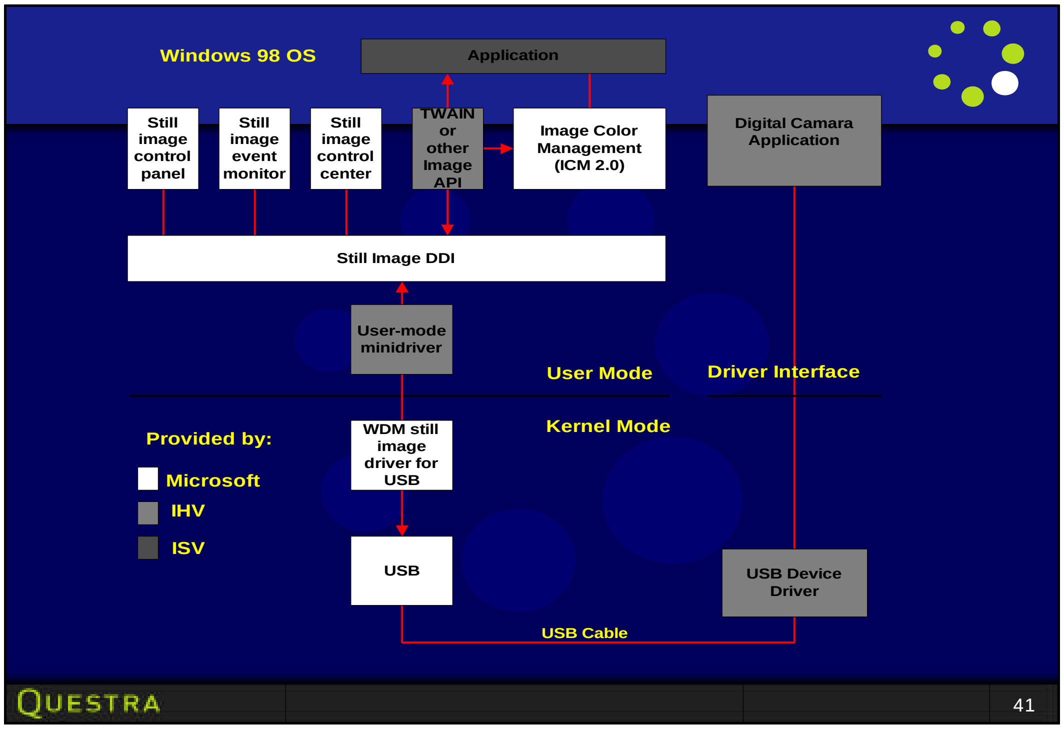
QUESTRA 39

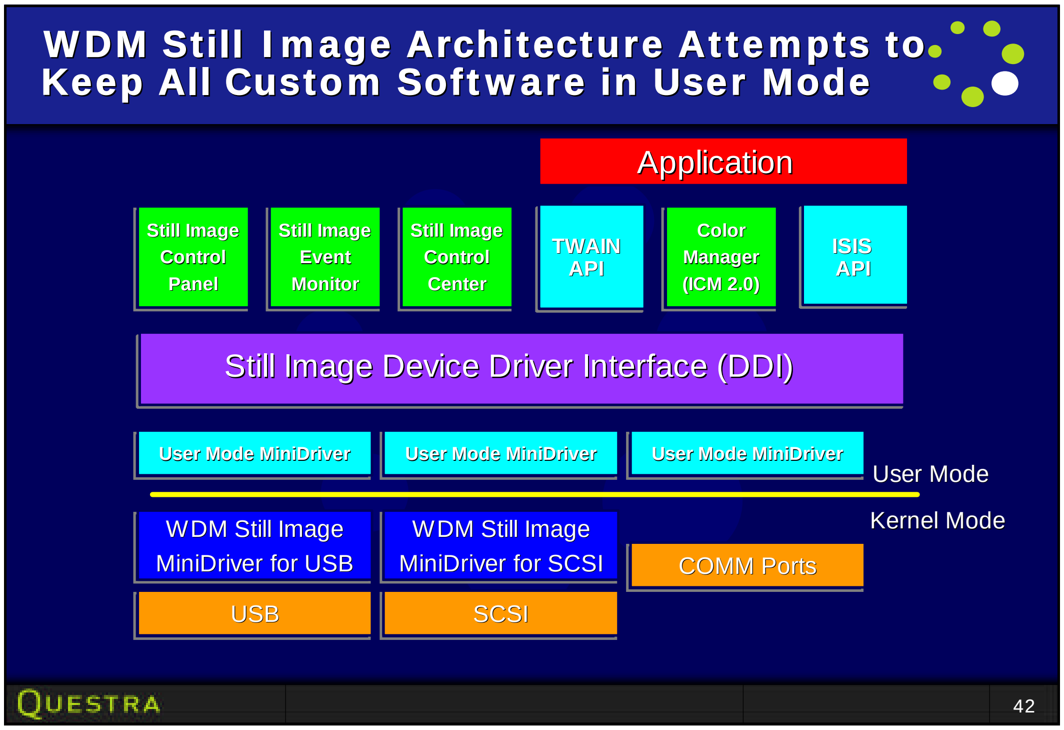


Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

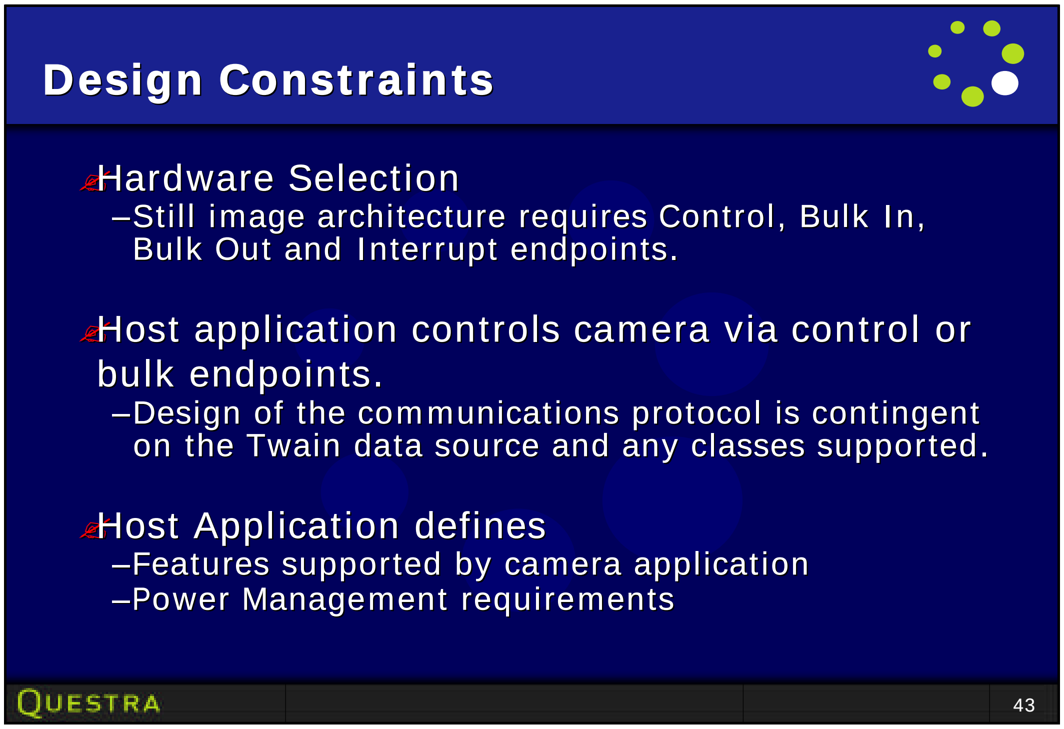
QUESTRA 40





WDM Still Image Architecture Attempts to Keep All Custom Software in User Mode

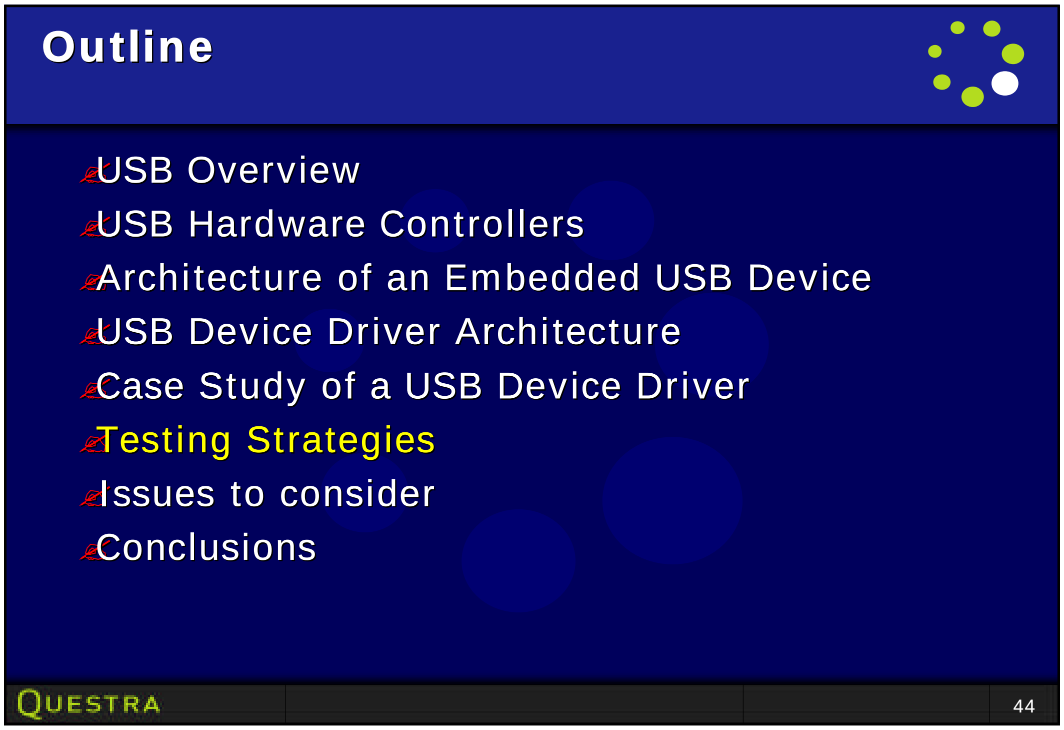
Application
Still Image Control Panel
Still Image Event Monitor
Still Image Control Center
TWAIN API
Color Manager (ICM 2.0)
ISIS API
Still Image Device Driver Interface (DDI)
User Mode MiniDriver
User Mode MiniDriver
User Mode MiniDriver
User Mode
Kernel Mode
WDM Still Image MiniDriver for USB
WDM Still Image MiniDriver for SCSI
COMM Ports
USB
SCSI
QUESTRA
42



Design Constraints

- Hardware Selection
  - –Still image architecture requires Control, Bulk In, Bulk Out and Interrupt endpoints.
- Host application controls camera via control or bulk endpoints.
  - –Design of the communications protocol is contingent on the Twain data source and any classes supported.
- Host Application defines
  - –Features supported by camera application
  - –Power Management requirements

QUESTRA 43



Outline

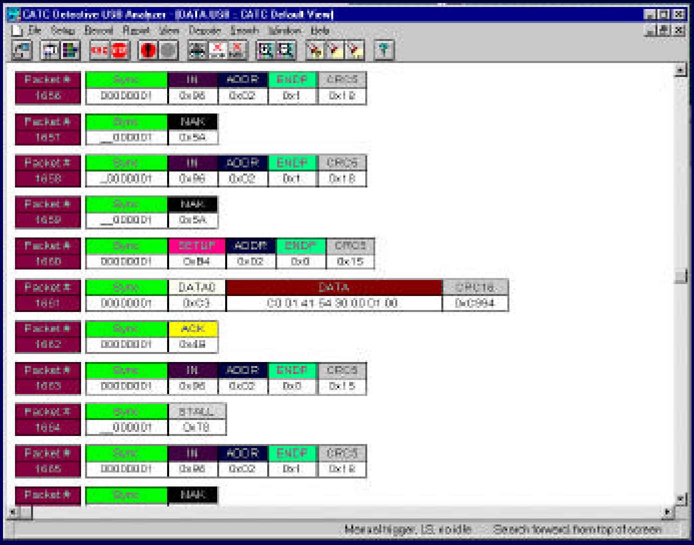
- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

QUESTRA 44

Effective Testing Strategies

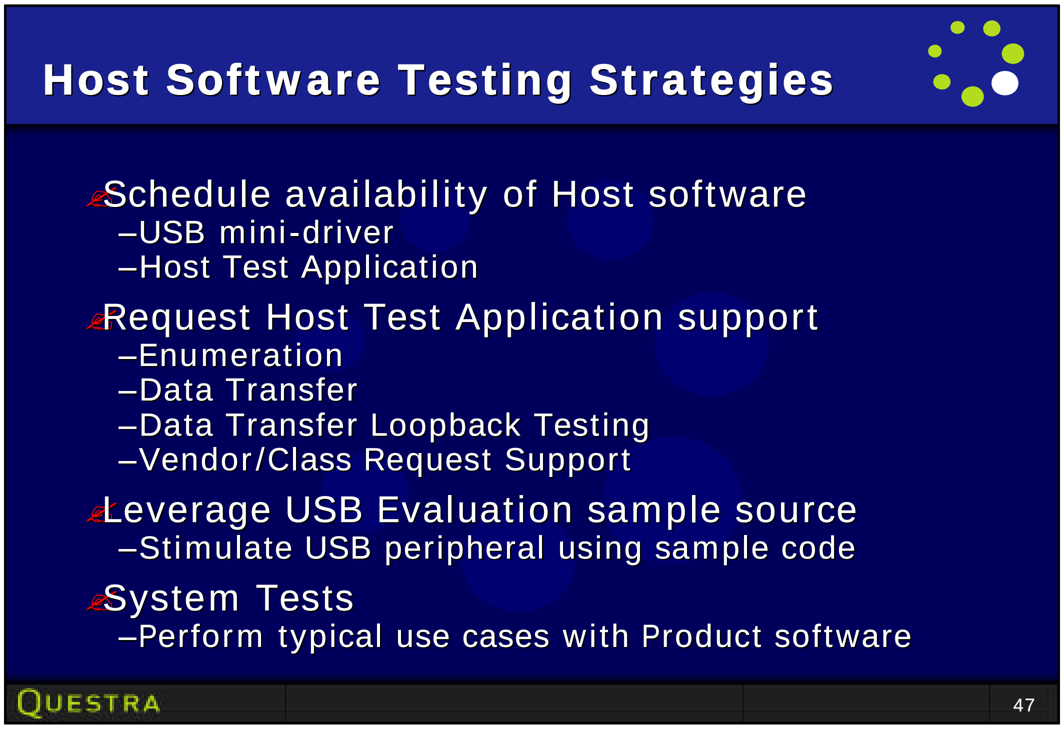
- Develop Written Test Plans
  – Define Unit Tests
  – Define System Tests
- Define minimum USB Host Driver Test
  – Capabilities
  – Enumeration, Data Transfer, Loopback, etc.
- Acquire an USB Analyzer
  – Use analyzer for documenting test results
  – Debug Driver enumeration and Data Transfer
  – Verify System level behavior with analyzer
  – Execute Compliance Test in loop mode (>1000x)
- Purchase a USB Evaluation Board & source code
- Utilize USB Organizations Test Resources

Analyzer View


CATC Detective USB Analyzer - [DATA.USB : CATC Default View]
File Setup Record Report View Decode Search Window Help

```
Packet #  Sync        IN      ADDR   ENDP   CRC5
1656      00000001    0x96    0x02   0x1    0x18

Packet #  Sync        NAK
1657      __000001    0x5A

Packet #  Sync        IN      ADDR   ENDP   CRC5
1658      _00000001   0x96    0x02   0x1    0x18

Packet #  Sync        NAK
1659      __000001    0x5A

Packet #  Sync        SETUP   ADDR   ENDP   CRC5
1660      00000001    0xB4    0x02   0x0    0x15

Packet #  Sync        DATA0   DATA                           CRC16
1661      00000001    0xC3    C0 01 41 54 30 00 01 00        0xC994

Packet #  Sync        ACK
1662      00000001    0x4B

Packet #  Sync        IN      ADDR   ENDP   CRC5
1663      00000001    0x96    0x02   0x0    0x15

Packet #  Sync        STALL
1664      __000001    0x78

Packet #  Sync        IN      ADDR   ENDP   CRC5
1665      00000001    0x96    0x02   0x1    0x18

Packet #  Sync        NAK
```

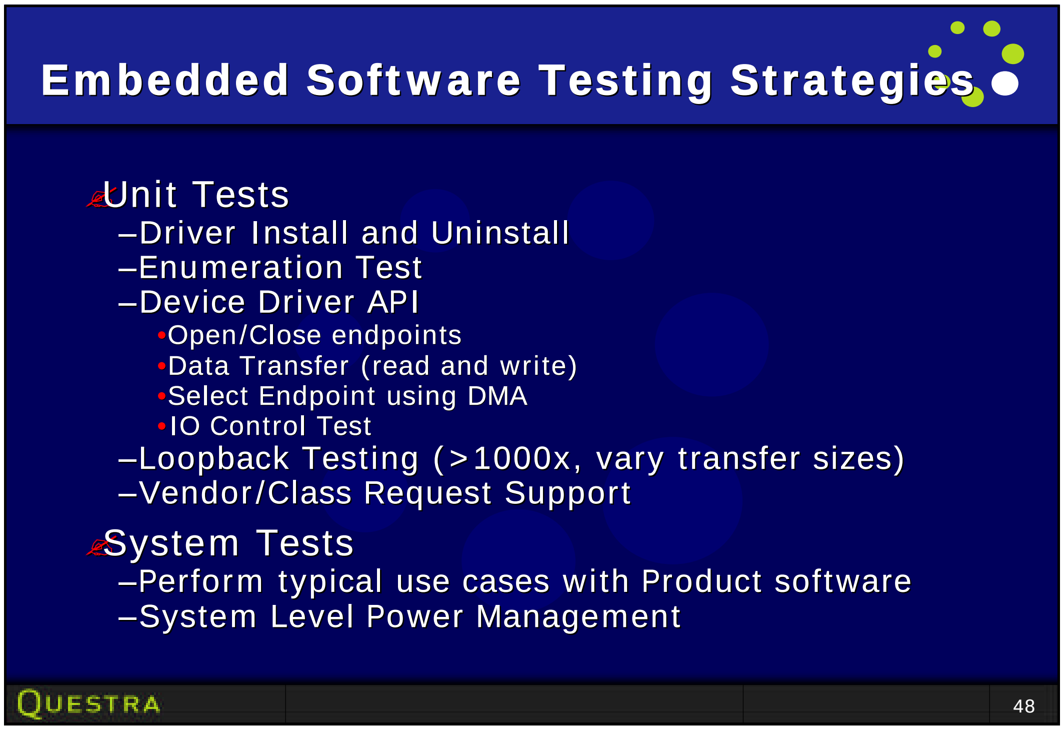
Manual trigger, LS, no idle   Search forward from top of screen



Host Software Testing Strategies

- Schedule availability of Host software
  - –USB mini-driver
  - –Host Test Application
- Request Host Test Application support
  - –Enumeration
  - –Data Transfer
  - –Data Transfer Loopback Testing
  - –Vendor/Class Request Support
- Leverage USB Evaluation sample source
  - –Stimulate USB peripheral using sample code
- System Tests
  - –Perform typical use cases with Product software

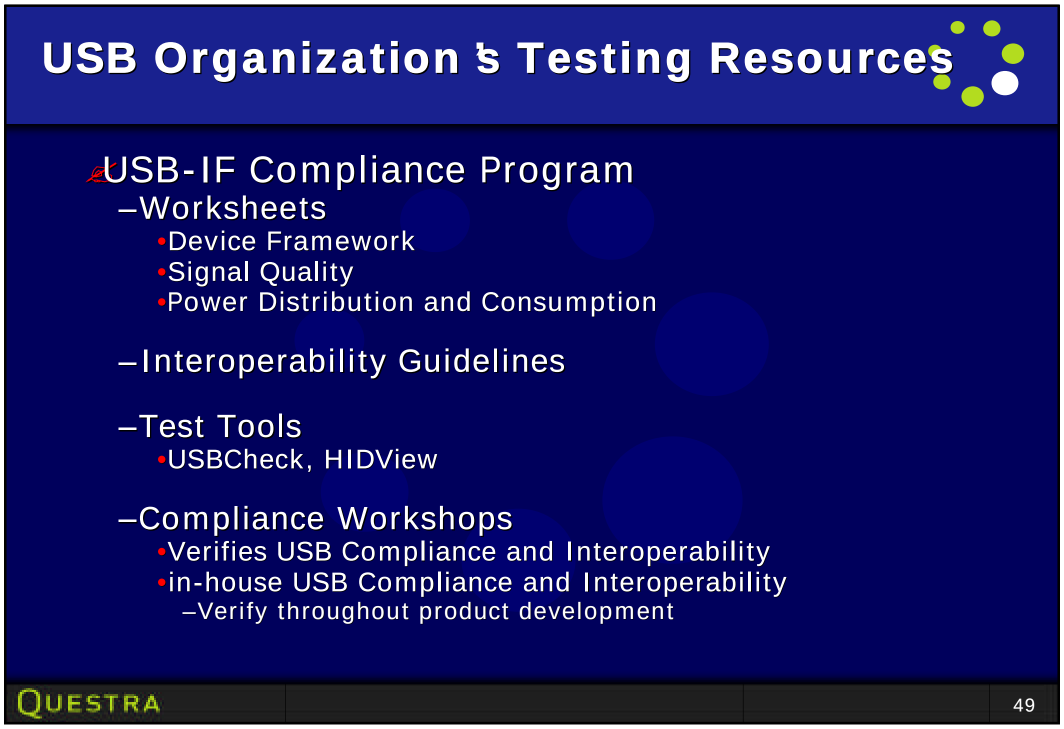
QUESTRA 47



Embedded Software Testing Strategies

- Unit Tests
  - –Driver Install and Uninstall
  - –Enumeration Test
  - –Device Driver API
    - Open/Close endpoints
    - Data Transfer (read and write)
    - Select Endpoint using DMA
    - IO Control Test
  - –Loopback Testing (>1000x, vary transfer sizes)
  - –Vendor/Class Request Support
- System Tests
  - –Perform typical use cases with Product software
  - –System Level Power Management

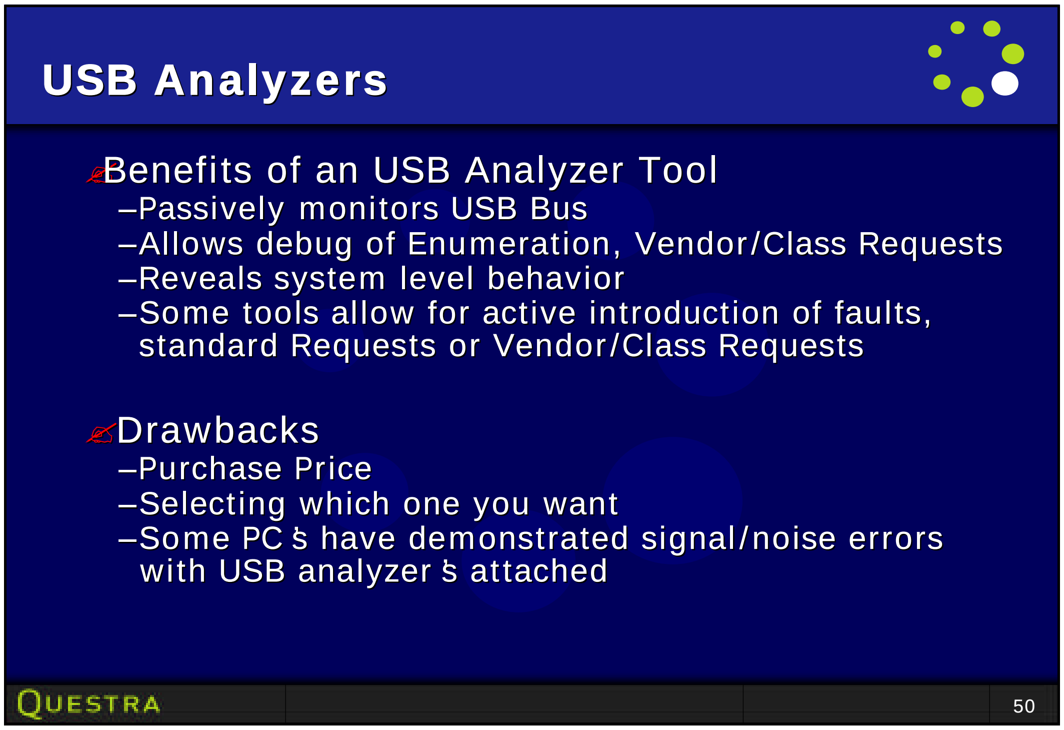
QUESTRA 48



USB Organization’s Testing Resources

- USB-IF Compliance Program
  - –Worksheets
    - Device Framework
    - Signal Quality
    - Power Distribution and Consumption
  - –Interoperability Guidelines
  - –Test Tools
    - USBCheck, HIDView
  - –Compliance Workshops
    - Verifies USB Compliance and Interoperability
    - in-house USB Compliance and Interoperability
      - –Verify throughout product development

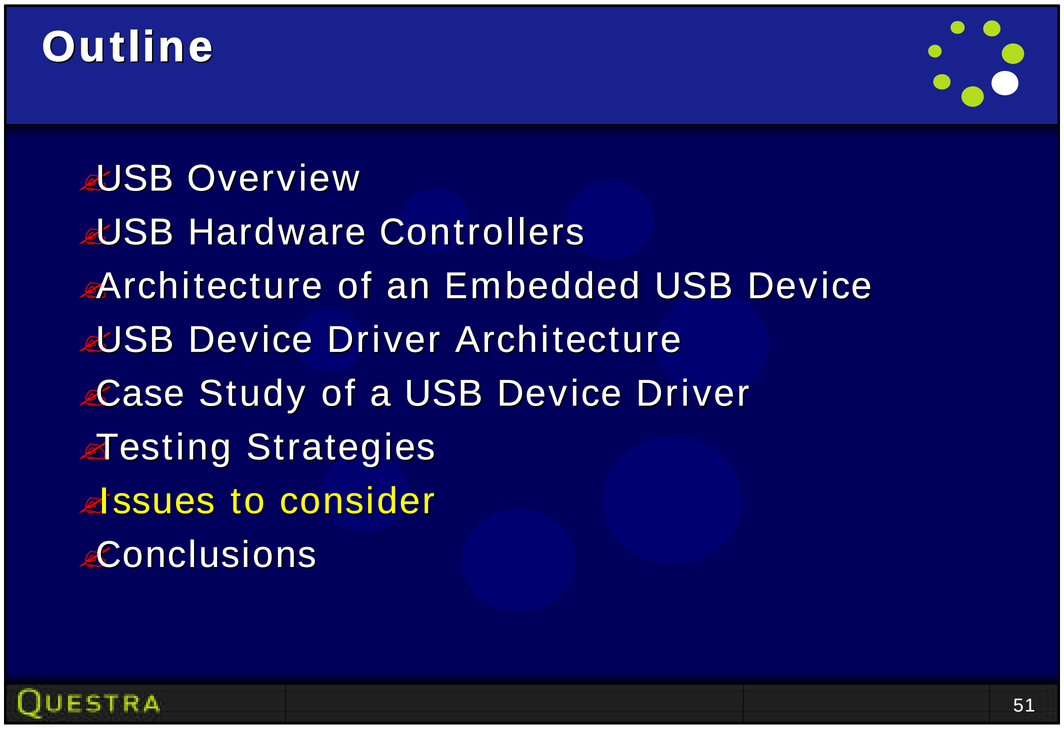
QUESTRA 49



USB Analyzers

- Benefits of an USB Analyzer Tool
  - –Passively monitors USB Bus
  - –Allows debug of Enumeration, Vendor/Class Requests
  - –Reveals system level behavior
  - –Some tools allow for active introduction of faults, standard Requests or Vendor/Class Requests
- Drawbacks
  - –Purchase Price
  - –Selecting which one you want
  - –Some PC’s have demonstrated signal/noise errors with USB analyzer’s attached

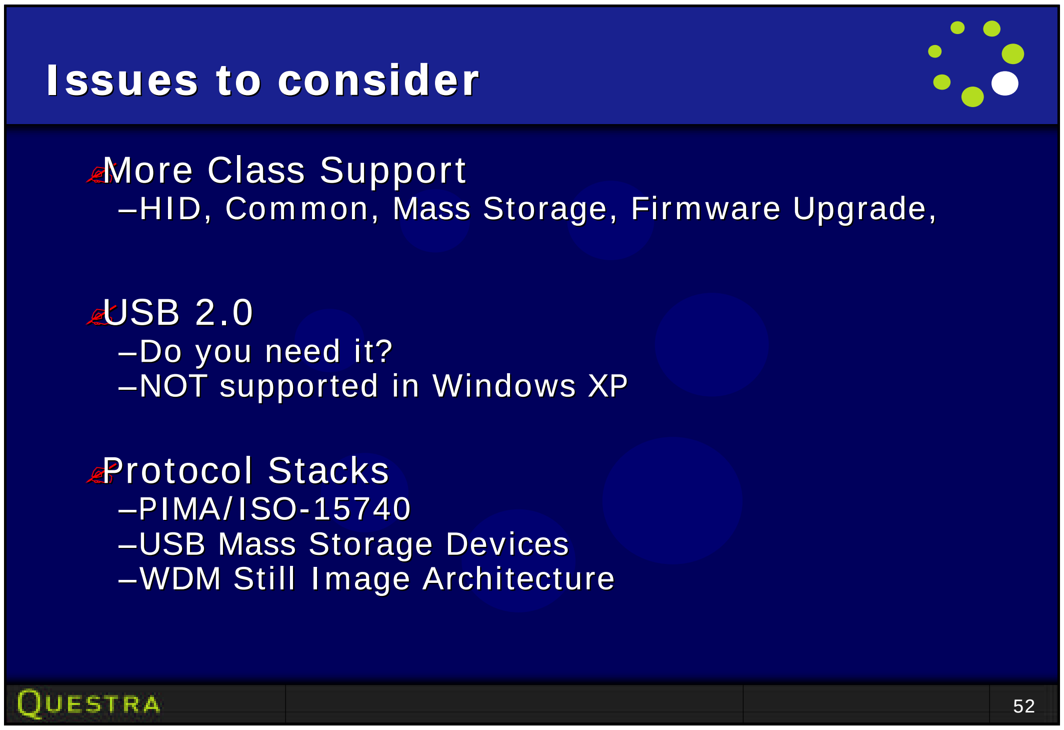
QUESTRA 50



Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

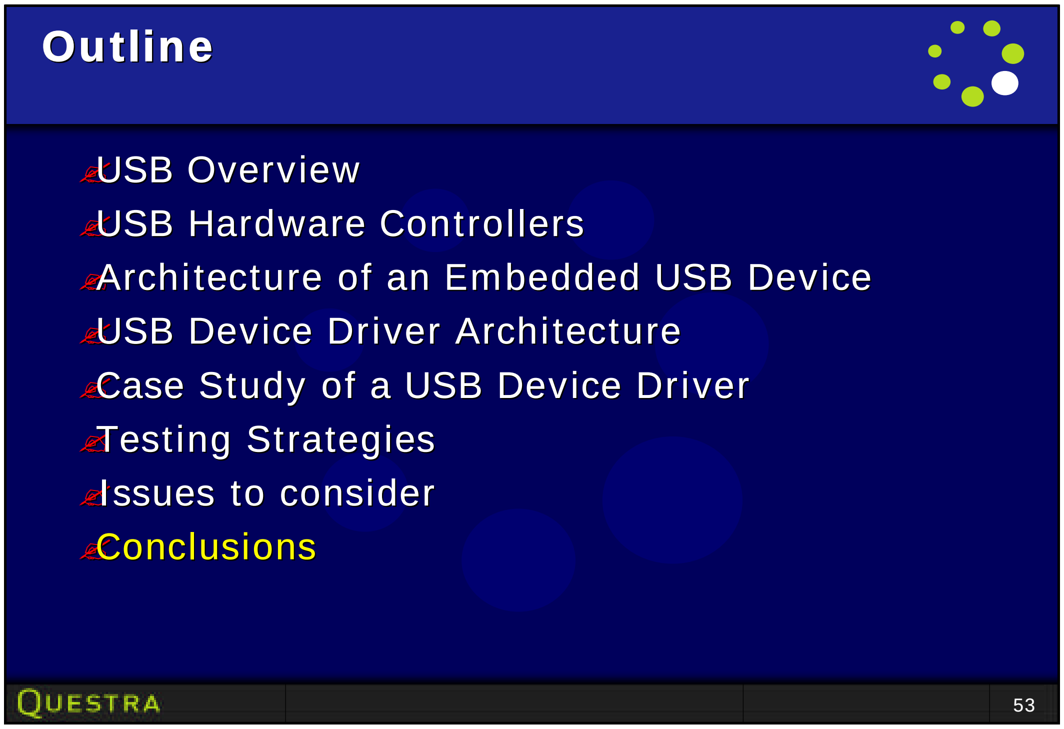
QUESTRA 51



Issues to consider

- More Class Support
  - HID, Common, Mass Storage, Firmware Upgrade,
- USB 2.0
  - Do you need it?
  - NOT supported in Windows XP
- Protocol Stacks
  - PIMA/ISO-15740
  - USB Mass Storage Devices
  - WDM Still Image Architecture

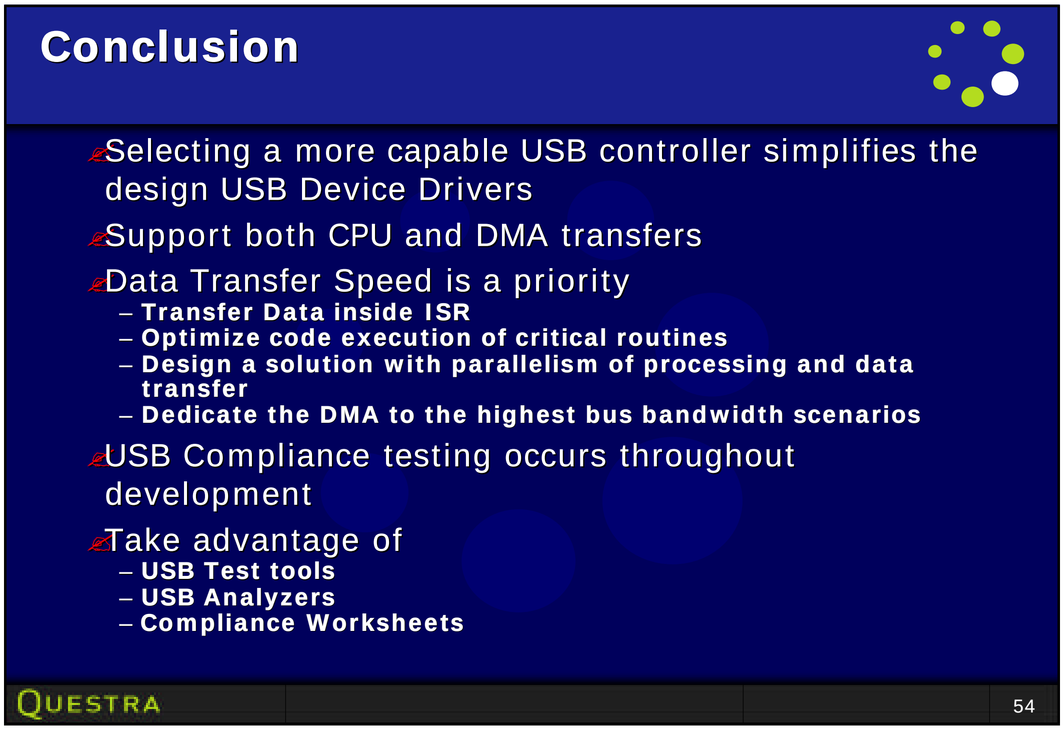
QUESTRA 52



Outline

- USB Overview
- USB Hardware Controllers
- Architecture of an Embedded USB Device
- USB Device Driver Architecture
- Case Study of a USB Device Driver
- Testing Strategies
- Issues to consider
- Conclusions

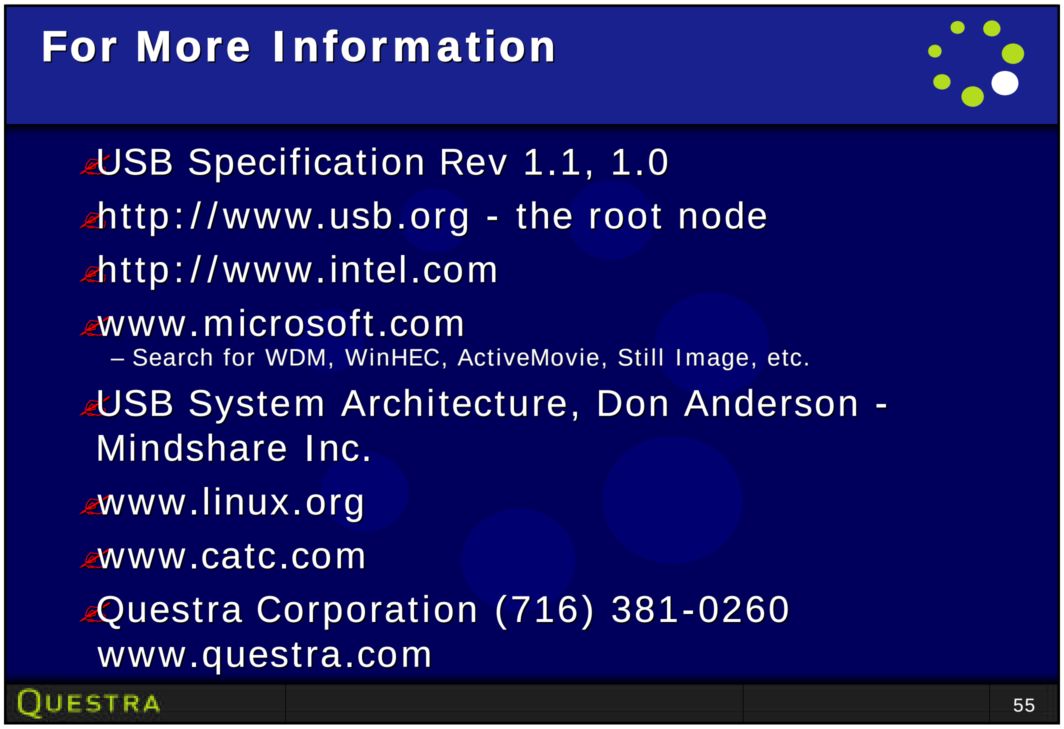
QUESTRA 53



Conclusion

- Selecting a more capable USB controller simplifies the design USB Device Drivers
- Support both CPU and DMA transfers
- Data Transfer Speed is a priority
  - **Transfer Data inside ISR**
  - **Optimize code execution of critical routines**
  - **Design a solution with parallelism of processing and data transfer**
  - **Dedicate the DMA to the highest bus bandwidth scenarios**
- USB Compliance testing occurs throughout development
- Take advantage of
  - **USB Test tools**
  - **USB Analyzers**
  - **Compliance Worksheets**

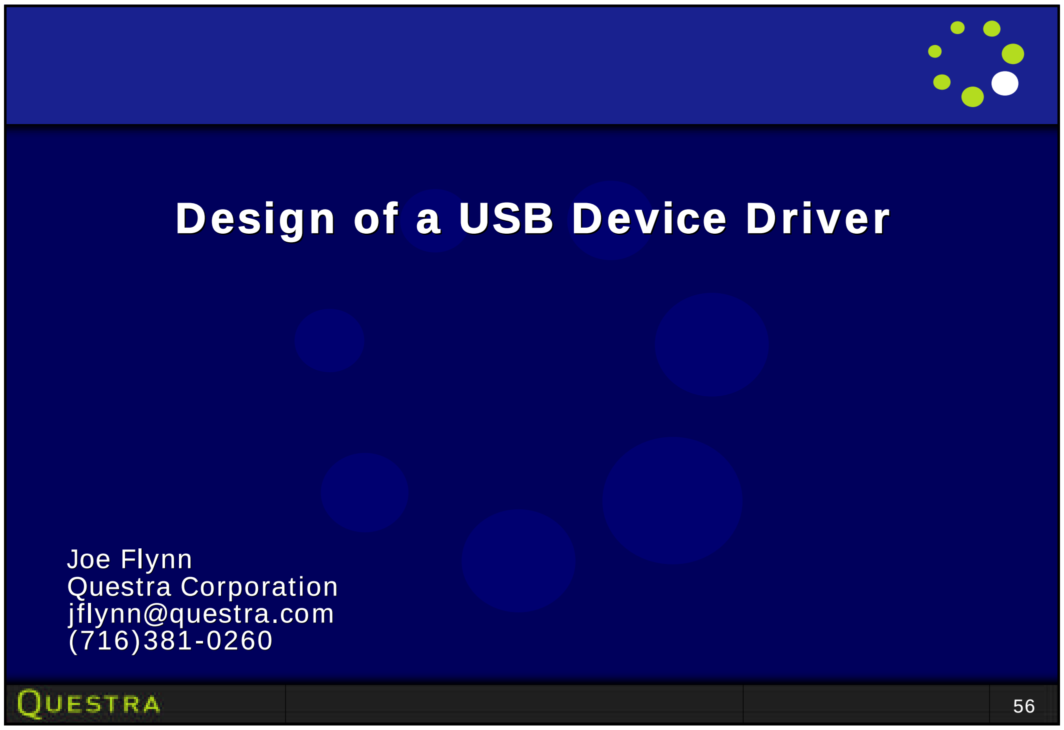
QUESTRA 54



For More Information

- USB Specification Rev 1.1, 1.0
- http://www.usb.org - the root node
- http://www.intel.com
- www.microsoft.com
  - Search for WDM, WinHEC, ActiveMovie, Still Image, etc.
- USB System Architecture, Don Anderson - Mindshare Inc.
- www.linux.org
- www.catc.com
- Questra Corporation (716) 381-0260 www.questra.com

QUESTRA 55



Design of a USB Device Driver

Joe Flynn
Questra Corporation
jflynn@questra.com
(716)381-0260

QUESTRA 56