
HI-SPEED
CERTIFIED
USB
TM

USB 2.0 Debug Port

John Keys

Intel Corporation

Agenda

- USB 2.0 Debug Port
- USB Debug Device
- Software Components
- Development Status

What Is the Debug Port?

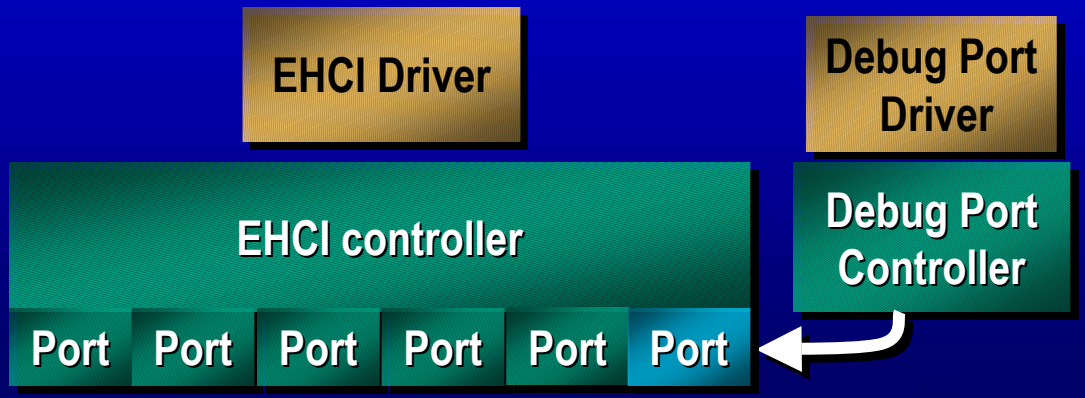
- Optional EHCI feature , details in Appendix C of the EHCI Spec, starting with Revision 0.96
- A simple means of performing single transactions: “pseudo-PIO” USB
- Only works with High-Speed USB devices
- Implemented for a specific port, optionally more
- Operational if port is not suspended and the HC is fully powered

Why Is It Needed?

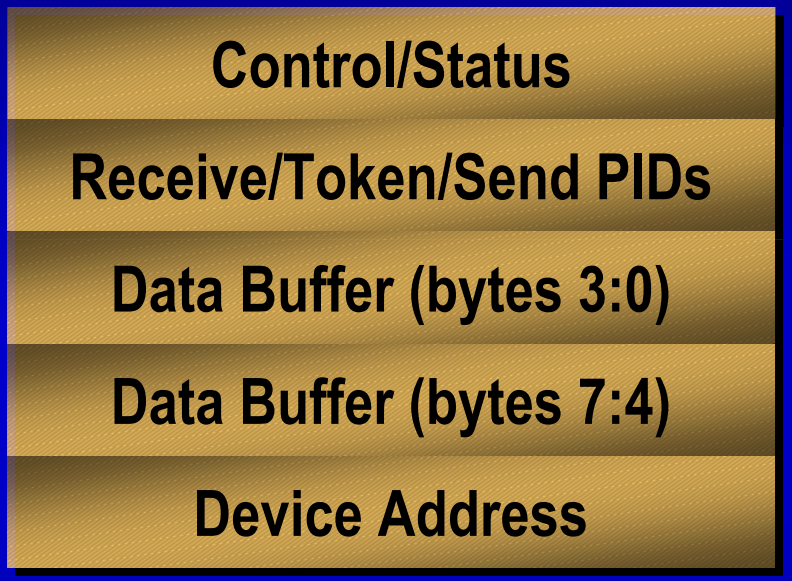
- PC2001 System Design Guide requires debug capabilities in all systems
- Low-cost solution for legacy free debugging
- Low operation complexity = Reliability & Stability
- 5x higher data rate than Super I/O & LPC solutions

Debug Port Architecture

- Debug port provides a memory mapped interface that is independent of EHCI controller
  - Memory space is typically mapped in same area as EHCI
  - Identified and located through ‘PCI capability’ mechanism



Memory Interface



- All registers are dwords
- Registers are located in same memory area as standard EHCI registers
- Debug port capability and register location is determined through PCI Extended Capabilities features

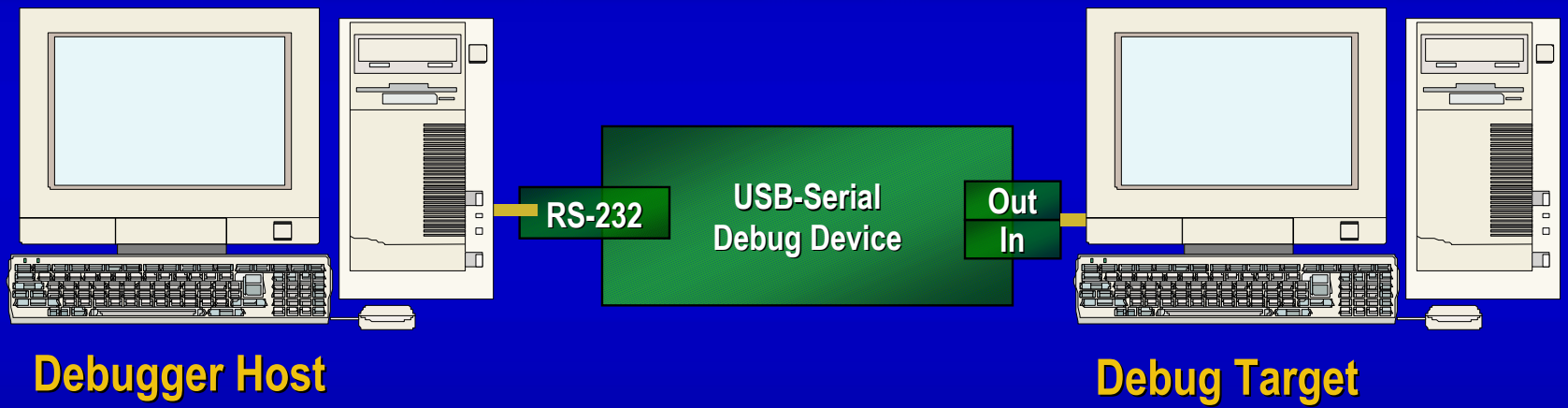
Operational Model

- For OUT transaction:
  - SW initializes token PID to OUT, data PID to DATAx, and puts appropriate data in the data buffer
  - SW tells HW to begin, HW sends token packet and data packet, then waits for handshake packet.
  - HW updates registers with status
  - SW checks status - If no error, SW checks received PID
- For IN transaction
  - SW initializes token PID to IN, and starts the HW
  - HW sends token packet, waits for DATAx packet, and responds with handshake packet if everything OK
  - HW updates registers with status and received length
  - SW checks status - if no error, SW checks PID, gets data

Agenda

- USB 2.0 Debug Port
- USB Debug Device
- Software Components
- Development Status

USB Debug Device

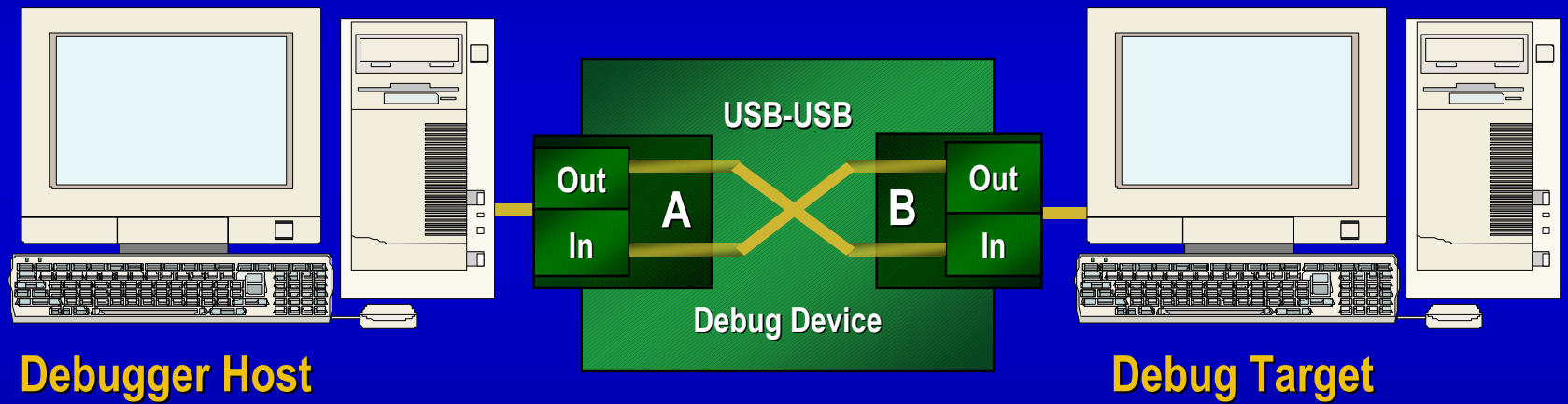


USB to Serial Debug Device Example

USB Debug Device

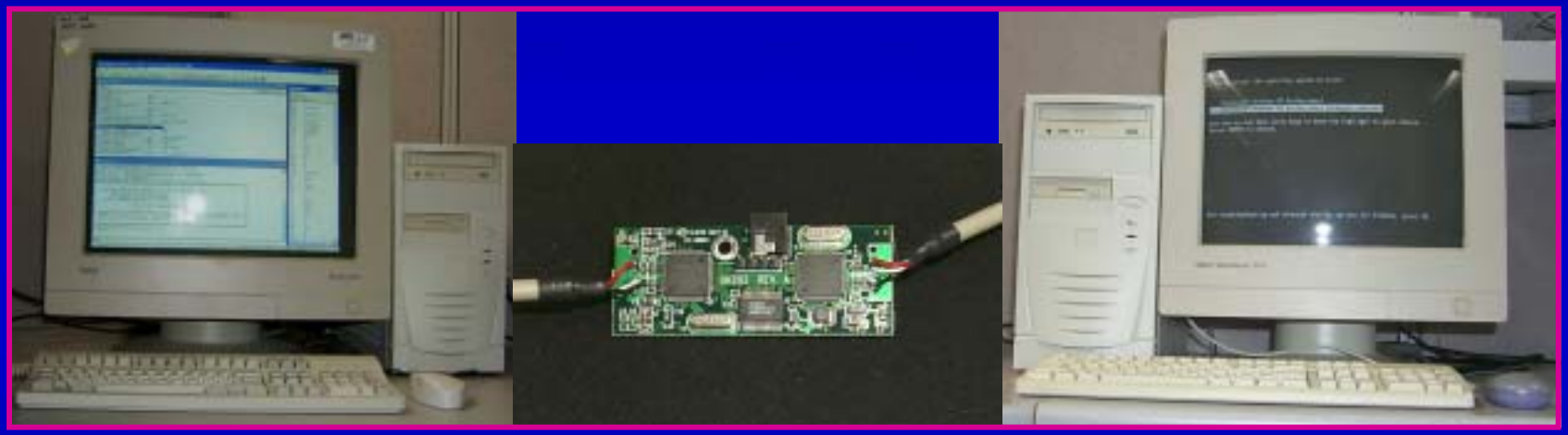
USB to Serial Debug Device Example

USB Debug Device



USB to USB Debug Device Example

USB Debug Device

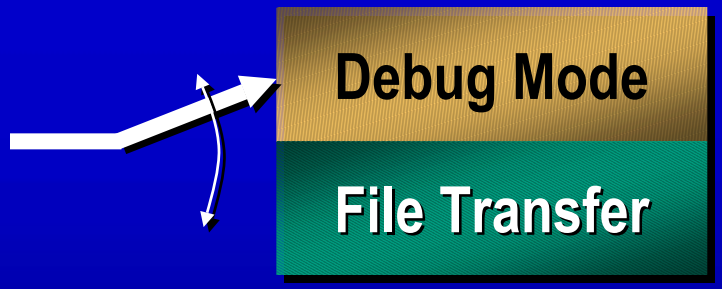


USB to USB Debug Device Example

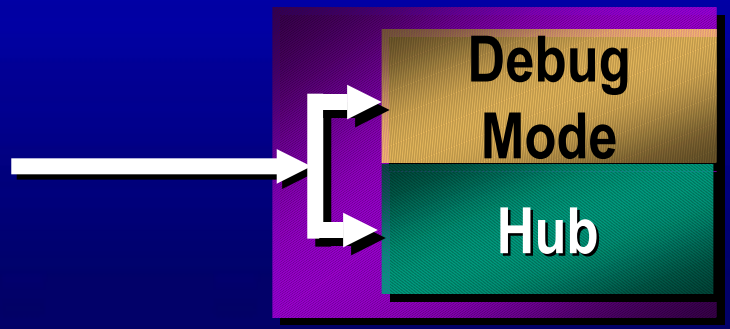
Debug Device Types

- **Fixed Function**
  - Dedicated - only a debug device
  - Has hard-coded address of 127

- **Single Function**
  - Works as a debug device OR as another type of device



- **Multi-function**
  - Combines a fixed-function debug device with another device, a hub for example



Debug Device Requirements

- Must be USB 2.0 High-Speed signaling compliant
- Debug Port Supports Maximum Packet Size of 8, so
  - One bulk-type IN endpoint supporting a maximum packet size of 8 bytes
  - One bulk-type OUT endpoint supporting a maximum packet size of 8 bytes
  - The Control Endpoint must correctly handle short transfer requests of 8 bytes
- Must support Debug Device extensions
- If device has a fixed address, that address must be 127

Debug Device Extensions

Two Extensions for Debug Devices:

- DEBUG Descriptor Type
- DEBUG MODE Feature Selector

Debug Device Extensions

- DEBUG Descriptor Type
  - Provides means of detecting Debug Device functionality
  - Valid from default, addressed, and configured states
  - Provides Target-platform with operational params in < 8 bytes
    (Remember, Debug Port = 8 byte max transfers)

Offset	Field	Value	Description
0	bLength	Number	Length of this Descriptor in bytes: 4
1	bDescriptorType	Constant	DEBUG descriptor type
2	bDebugInEndpoint	Number	Endpoint number of Debug Data IN endpoint
3	bDebugOutEndpoint	Number	Endpoint number of Debug Data IN endpoint

Debug Device Extensions

- DEBUG MODE Feature Selector
  - Tells device to begin “debug device” operational mode
  - Used with SET_FEATURE request
  - Valid only from Default and Addressed state
  - Implied SET_CONFIGURATION request - allows Target to configure device without parsing configuration / interface / endpoint descriptors

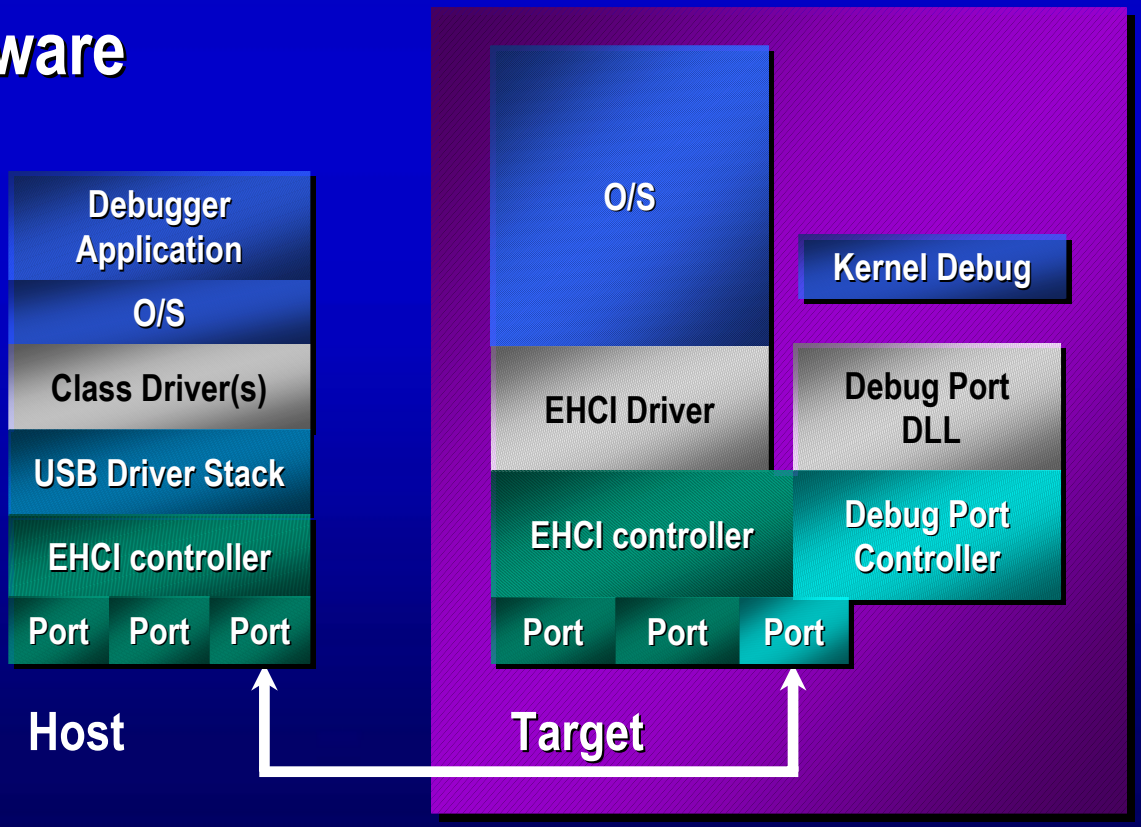
bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_FEATURE	DEBUG MODE	0	0	None

Agenda

- USB 2.0 Debug Port
- USB Debug Device
- Software Components
- Development Status

Target SW Components

- Target Platform Components
  - Debug Port DLL
  - Debug Port-aware EHCI driver



Debug Port DLL

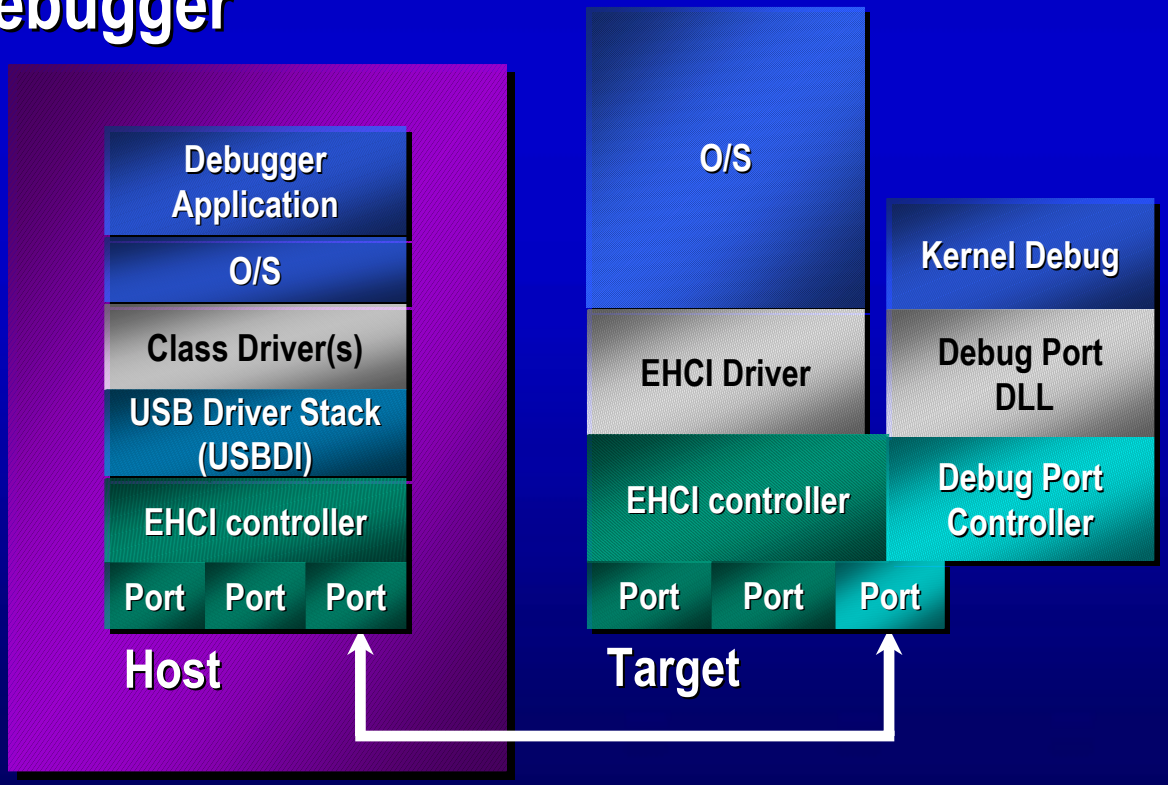
- Connects Kernel Debugging to the USB 2 Debug Port
- Loaded from BOOT.INI
  - DEBUGPORT=USB - O/S Loader loads kdusb.dll
- If device connected to Debug Port:
  - Reset Port
  - Send Get Descriptor:DEBUG descriptor type request to device
  - If device returns descriptor
    - Set device address (typically 127)
    - Send Set Feature:DEBUG MODE request to device to set configuration
    - Sets OWNER and IN-USE bits in Control/Status register to inform EHCI driver that debug port is in use

Debug Port-Aware EHCI Driver

- Cooperates with Debug Port DLL
- Examines Debug Port Control/Status OWNER and IN-USE bits at init time
- If OWNER / IN-USE set:
  - Does not reset host controller
  - Does not include Debug Port in Root Hub ports (Debug Port not available for general use)
  - Does not reset or suspend Debug Port
  - Does not respond to Debug Port status

Debugger Host SW Components

- Debugger Host Platform Components
  - Class Driver(s) that connect USB driver stack to Debugger



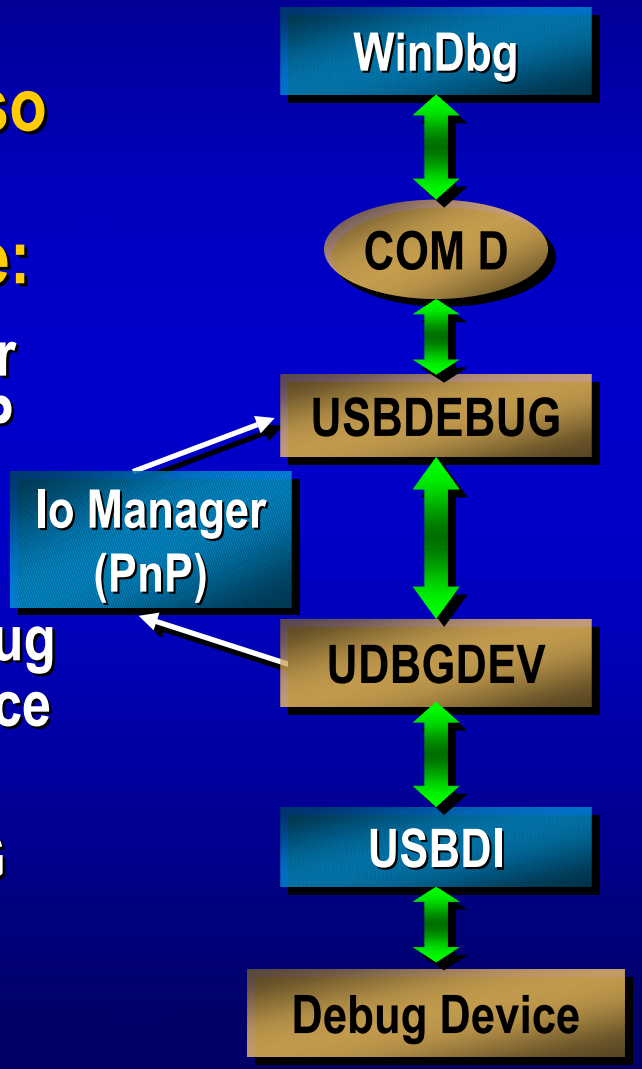
Current WinDbg Solution

Right now WinDbg doesn't know USB, so we use two drivers to create a virtual serial port from the USB debug device:

1) USBDEBUG.SYS creates a Device Object for the virtual serial port (COM D), Waits for PnP to announce arrival of a Debug Device interface

2) UDBGDEV.SYS is a PnP driver for USB Debug Devices. It registers a Debug Device interface with PnP when Debug Device appears

3) WinDbg opens COM D, causing USBDEBUG to open UDBGDEV



Agenda

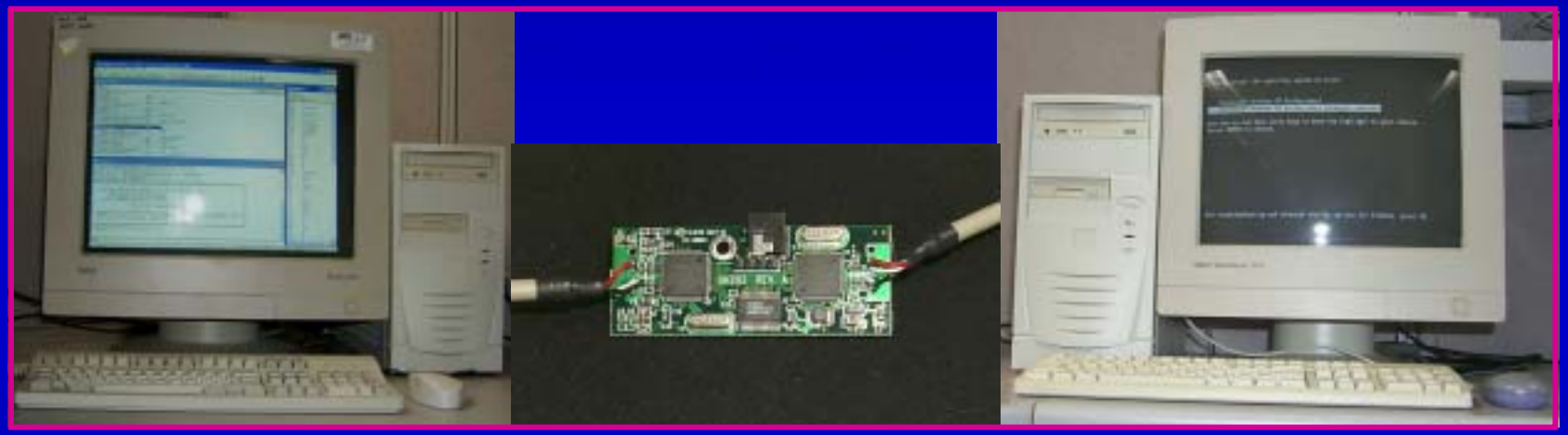
- USB 2.0 Debug Port
- USB Debug Device
- Software Components
- Development Status

Development Status

- Debug Port Implemented in ICH4 Chipset, other vendors nearing completion of Debug Port implementations
- Prototype USB-to-Serial & USB-to-USB Devices Done
  - Netchip has working USB-to-USB reference design for OEMs

- Prototype Software Components Complete
- Intel is working with Microsoft on:
  - Debug Port DLL and Debug Port-Aware EHCI Miniport
  - Deployment Strategy

Demonstration



USB to USB Debug Device

Summary

- USB 2 Debug Port allows kernel debugging on legacy free systems, a viable alternative
- Ingredients coming together nicely: Host controllers, Debug devices, Target software, and Remote software

Call To Action

If you want USB debugging, let Microsoft know how much you want it

Collateral

Available from http://developer.intel.com/technology/usb/spec.htm:

- Enhanced Host Controller Interface Specification for Universal Serial Bus
  Revision 0.96 and 1.00
- USB2 Debug Device - A Functional Device Specification
  Revision 0.9