



# Developers Conference 2007

Amsterdam, The Netherlands



# Developing a Device Based on the Certified Wireless USB Spec

**Bart Vertenten**  
NXP Semiconductors

# Agenda



- Usage Models and Consumer Experience
- DWA vs. Native
- Main Problems to Solve
- Designing Your Own Wireless USB Device IP
  - Possible Pitfalls

# Usage Models



## CE Segment



- Eliminate cable hassles for portable device content upload / display
- Ease install/config of next gen digital displays and home theater
- Ease AV load on WLAN

## PC Segment



- Eliminate USB cables
- Highest value for portable devices
- Unwire high rate devices
- Applicable to digital home and digital office

## Mobile Segment



- Enable high rate apps
- Wireless notebook docking station
- Handheld + notebook linkage w/ resource sharing
- Wireless AV display from notebook and handheld

# Example : Audio Files



- Need for speed: Audio

- MP3 Music Download:

- 15 songs @ 5MB / song ~ 75MB
    - 128 Kbps for Playback
    - Time to download
      - 10 Mbps takes 1 minute → Not acceptable
      - 30 Mbps takes 20 seconds → Better
      - 300 Mbps takes 2 seconds → Way to go !



- CD Download:

- 74 minutes audio takes ~ 640 MB
    - 1.4 Mbps for Playback
    - Time to download
      - 9 Mbps takes 10 minutes → Not acceptable
      - 45 Mbps takes 2 minutes → Better
      - 300 Mbps takes 17 seconds → Way to go !



# Need for Fast Download



- Certified Wireless USB Storage Application
  - 480Mbps wirelessly
- Advantages
  - Reduced installation costs
    - Build on current drivers
  - Improved range
  - Portability
  - Simplicity of use
  - Easy connections between several sources
  - No “non-standard” card interface concerns



# From USB to a Certified Wireless USB Device



- 3 possibilities
  - Embedding DWA
  - Native Device
  - Device IP (Intellectual Property)

# DWA vs. Native



## ***Embedded DWA***

- No knowledge of Certified Wireless USB
- Firmware to display and verify numeric association model
- Fastest TTM
- More expensive

## ***Native Device***

- Minimal knowledge of Certified Wireless USB
- Integration of Certified Wireless USB firmware (no change on application firmware)
- Fast TTM
- Less expensive

## ***Device IP***

- Full knowledge of Certified Wireless USB
- All wireless features must be developed
- Slowest TTM
- Least expensive



# Main Problems to Solve

- Power Delivery
- Association Model
- Selecting the Right Solution for the Application

# Power Delivery



- Self-powered USB devices
  - No problem
  - Architecture can remain same
- Bus-powered USB device
  - Removing USB cable = removing power cable
  - Integrator will need to include power source
    - Example : USB thumb drive

# Power Management

## Channel Stop



- Send by host when PC is put in power down
- 2 options for device :
  - User intervention to wakeup device
    - Power down device and wake up when there is user intervention
    - Preferred for bus-powered devices
  - Device must be active again when PC is awake
    - Device regularly checks if Wireless USB host is sending MMCs again
    - Preferred for self-powered devices

# Power Management

## Sleep



- Device sends DN\_Sleep if it wants to go into sleep mode
- While in sleep, device sends this to host from time-to-time to check if host has data to send
- To be used by self-powered devices
- For battery-powered devices, it is better to use the disconnect function

# Association Model

## Models to Be Implemented



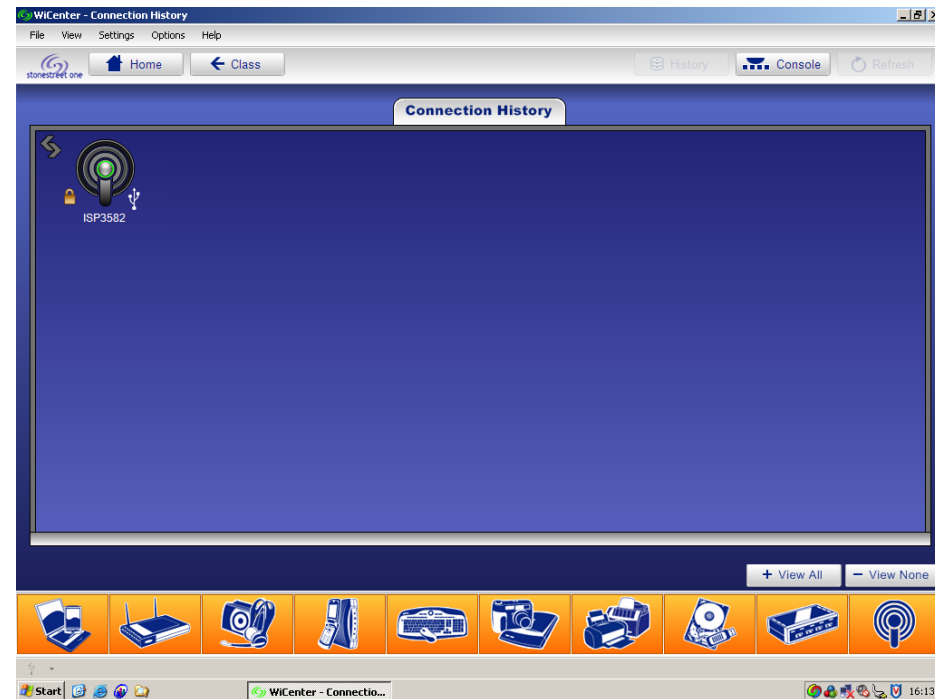
- Form factor of product defines which model to implement
  - Display on product
    - Numeric model must be implemented
  - USB B-receptacle on product
    - Cable model must be implemented
  - Display **and** B-receptacle on product
    - **Both** numeric **and** cable models must be implemented

# Association Model

## Which Host to Connect to?



- How many host to support?
  - Size of non-volatile memory to store the number of CCs (Connection Context)
- User Interface to select which host to connect
- How to remove host from list?
  - Temporary access to device



# Selecting the Right Solution for the Application

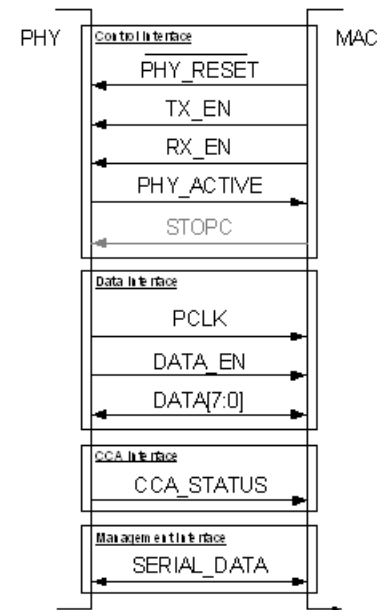


- Throughput
  - Buffer mechanism
    - Data bursting supported or not
    - Sufficient buffers to minimize device NAKing
      - Host 'un-schedules' endpoint upon NAK until DN\_EPRdy indication
  - Maximum Sequence Number supported
    - Allows for the number of "smashed" packets
  - Number of Rpipes implemented in DWA
    - Equal to number of endpoints active at the same time
- Association model
  - Does it support the model you want?
- Power consumption

# Designing Your Own Wireless USB Device IP



- WiMedia MAC-PHY interface spec
  - Use a building block for PHY
- SW-based architecture
  - Faster processor
  - More flexible
  - Some timing can affect performance
    - Better to be implemented in the hardware
- HW-based architecture
  - Optimized solution for power and size



# Possible Pitfalls



- Self-beaconing
  - All devices must beacon to comply with WiMedia
  - Running multiple PALs on same PHY-MAC
    - Remaining problem :
      - When PHY Channel Change is happening on 1 PAL and not on the other
  - More info during presentation on “Self-Beaconing, Dual-Role Device, and Multi-PAL Design Considerations” – *this room, today at 10:30am*

# Possible Pitfalls



- WCTA inside MMC

- IN endpoint

- Max 1  $W_{DT}$ CTA for each endpoint per MMC

MMC	Dir	Host Id	Chan TS	IE Id	CTA 0	Addr	Endp	EDir	bvDINAck	CTA 1
	-->	0x1D5D	1962.735 ms	WCTA_IE	DT	1	1	IN	0x00000080	EOL

- OUT endpoint

- Max 1  $W_{DR}$ CTA and 1  $W_{DT}$ CTA for each endpoint per MMC

MMC	Dir	Host Id	Chan TS	IE Id	CTA 0	Addr	Endp	CTA 1	Addr	Endp	EDir	bvDINAck	CTA 2	IE Id
	-->	0x1D5D	1961.314 ms	WCTA_IE	DR	1	1	DT (Hnd)	1	1	OUT	0x00000000	EOL	WHOSTINFO_IE

# Possible Pitfalls



- Multiple IEs in one MMC
  - Host: Watch for the ordering rules for MMC\_IEs
  - Device: Ignore (and skip) not recognizable IE (“forward” compatible)

▶ MMC	Dir	Host Id	▶ IE Id	▶ CTA 0	# Notif TS's	▶ CTA 1	▶ IE Id	▶ IE Id
-->	0xBEEF	WCTA_IE	DNTS	16	EOL	WCONNECTACK_IE	WHOSTINFO_IE	

# Possible Pitfalls



- DN\_EPRdy
  - Signal to host that the endpoint is ready to resume data transfer
  - Accurate reflection of buffers available
  - Used by host to update internal bvAckcode

MMC	Dir	Host Id	IE Id	CTA 0	# Notif TS's	CTA 1
	-->	0x0C62	WCTA_IE	DNTS	2	EOL

DN	Dir	Host Id	Addr	DN Type	Length	Endp	EDir	bvAckCode	Endp	EDir	bvDINAck
	<--	0x0C62	2	Endpoints Rdy	2	1	OUT	0x00000040	1	IN	0x00000800



# Possible Pitfalls

- Slot time of a DNTS slot
  - $t_{\text{NOTIFICATIONSLOT}} = 26 \text{ us}$  (not 24)
  - Read Errata !!!

# Possible Pitfalls



- Data bursting
  - Stuck-at wrap condition
    - 2 possible ways to get out of it
    - Refer to presentation on “Data bursting”

# Possible Pitfalls



- Adjusted MaxPacketSize
  - Optional feature
  - To increase the chance of successful transmission in a bad channel
  - Implement buffer mechanism that can support this
  - bvAckcode must be calculated on the actual maxpacketsize
    - Regular condition
    - Last frame

# Possible Pitfalls



- MaxPacketSize for Control Endpoint 0
  - Fixed 512 bytes
  - Except for Data Loopback mode
    - Largest of all the available endpoints  
MaxPacketSize

# Possible Pitfalls



- SetAddress
  - Be able to accept WCTA from 2 different device addresses
  - Case 1 : ACK of status successfully received by host
    - New setup phase to new device address
  - Case 2 : ACK of status not received by host
    - Retry of status phase to old device address

# Possible Pitfalls



- SetAddress : case 1

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	Data	CTA 2	IE Id	IE Id	FCS
	-->	WCTA_IE	DNTS	8	DT	128	0	IN	0x00000001	00 05 00 00 00 00 00 00	EOL	WCONNECTACK_IE	WHOSTINFO_IE	0x71F1FC91

Hand	Dir	Host Id	NAK	Addr	Endp	EDir
	<--	0xAA55	2	128	0	IN

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	CTA 2
	-->	WCTA_IE	DNTS	8	DT	128	0	IN	0x00000001	EOL

Hand	Dir	Host Id	ACK	Addr	Endp	EDir	bvAckCode
	<--	0xAA55	1	128	0	IN	0x00000001

← ACK of status successfully received by host

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	Data
	-->	WCTA_IE	DNTS	1	DT	0	0	IN	0x00000001	8 bytes

Hand	Dir	Host Id	NAK	Addr	Endp	EDir
	<--	0xAA55	2	0	0	IN

← Device answers on address 0

MMC	Dir	IE Id	CTA 0	Addr	Endp	EDir	bvDINAck	CTA 1
	-->	WCTA_IE	DT	0	0	IN	0x00000001	EOL

IN	Dir	Host Id	DATA	Addr	Endp	EDir	Secure Payload
	<--	0xAA55	0	0	0	OUT	18 bytes

# Possible Pitfalls



- SetAddress : case 2

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	Data	CTA 2	IE Id	IE Id	FCS
	-->	WCTA_IE	DNTS	8	DT	128	0	IN	0x00000001	00 05 00 00 00 00 00 00	EOL	WCONNECTACK_IE	WHOSTINFO_IE	0x71F1FC91

Hand	Dir	Host Id	NAK	Addr	Endp	EDir
	<--	0xAA55	2	128	0	IN

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	CTA 2
	-->	WCTA_IE	DNTS	8	DT	128	0	IN	0x00000001	EOL

Hand	Dir	Host Id	ACK	Addr	Endp	EDir	bvAckCode
	<--	0xAA55	1	128	0	IN	0x00000001

← ACK of status not received by host

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	CTA 2
	-->	WCTA_IE	DNTS	8	DT	128	0	IN	0x00000001	EOL

Hand	Dir	Host Id	ACK	Addr	Endp	EDir	bvAckCode
	<--	0xAA55	1	128	0	IN	0x00000001

← Device answers again status on address 128

MMC	Dir	IE Id	CTA 0	# Notif TS's	CTA 1	Addr	Endp	EDir	bvDINAck	Data
	-->	WCTA_IE	DNTS	1	DT	0	0	IN	0x00000001	8 bytes

Hand	Dir	Host Id	NAK	Addr	Endp	EDir
	<--	0xAA55	2	0	0	IN

MMC	Dir	IE Id	CTA 0	Addr	Endp	EDir	bvDINAck	CTA 1
	-->	WCTA_IE	DT	0	0	IN	0x00000001	EOL

IN	Dir	Host Id	DATA	Addr	Endp	EDir	Secure Payload
	<--	0xAA55	0	0	0	OUT	18 bytes

# Numeric Association



- Big endian vs. little endian
  - Number is defined as any field  $\leq 64$  bits
    - All numbers in Assoc spec are presented in big endian
    - All numbers are required to be sent over USB as little endian
    - All numbers used for cryptographic calculations must be stored in memory in big-endian
  - Fields  $\geq 64$  bits are always byte-arrays
    - UNICODE strings are always byte-arrays
  - Read FAQ#69 for clear explanation !

# Numeric Association



- Big endian vs. little endian

- Numbers (little endian) :

- AssociationTypeId
    - AssociationSubTypeId
    - Length
    - AssociationStatus
    - LangID
    - BandGroups
    - Flags
    - AssociationTypeInfoSize

- Byte arrays :

- DeviceFriendlyName
    - HostFriendlyName
    - CHID
    - CDID
    - CK
    - ConnectionContext
    - SHA-256(M3)
    - PK\_H
    - PK\_D

# Design for Performance



- What to do to have optimal throughput on your device
  - Wireless USB protocol
    - Data burst size = 16
    - Max Packet Size = 3584 bytes
    - Max Sequence Number = 31
  - Buffer mechanism
    - 32 buffers
      - 16 buffers to receive max burst from host and 16 buffers to move data to application

# Design for a Good Host



- Support for data bursting
- Idle time between last WCTA slot and next MMC frame
  - Used as calculation time for next MMC
  - Major impact on throughput

# Get the Logo



- Interop testing with multiple HWA and WHCI
  - Different timings can show different problems
- Run WUSBCV to check specific corner cases

# Summary



- Multiple solutions to build a Certified Wireless USB device
- Analyze your selection criteria
- Do interop testing with multiple host solutions



# Developers Conference 2007

Amsterdam, The Netherlands