



# Developing a Device Based on the Certified Wireless USB Specification

**Jerome Tjia**  
Philips Semiconductors

*Content also provided by:  
Bart Vertenten, Philips Semiconductor*

# Outline



- Usage Models and Consumer Experience
- DWA vs. Native
- Main Problems to Solve
- Designing Your Own Certified 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

# 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



# 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 !



# From USB to 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 Solutions



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

# 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 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 Certified 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 power vs. size

# Possible Pitfalls



- Self-beaconing vs. directed beaconing
  - Self-beaconing is must when:
    - Running multiple PALs on same PHY-MAC
      - Solve problem when PHY Channel Change is happening on 1 PAL and not on the other
  - More info during presentation on “Self-Beaconing and Dual-Role Device Design Considerations”

# Possible Pitfalls



- WCTA inside MMC
  - IN endpoint
    - Max 1  $W_{DT}$ CTA for each endpoint per MMC
  - OUT endpoint
    - Max 1  $W_{DR}$ CTA and 1  $W_{DT}$ CTA for each endpoint per MMC
- Multiple IEs in one MMC
  - Host: Watch for the ordering rules for MMC\_IEs
  - Device: Ignore (and skip) unrecognizable IE (“forward” compatible)

# Possible Pitfalls



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



# 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
    - More info during presentation on “Databursting”

# 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 maxpacket size
    - 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

# Design for Performance



- What to do to have optimal throughput on your device
  - Certified Wireless 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 databursting
- 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 2006

Taipei, Taiwan