Building the ZoomFloppy

Nate Lawson
nate@root.org

Presented at ECCC
September 18, 2010
Overview

• Commodore floppy drive architecture
  – How the drive works
  – Copy protection schemes

• The old way to interface a 1541 and PC (printer port)
• USB interfacing: the new way
  – xu1541 (2007)
  – xum1541 (2009)

• Creating the xum1541 firmware
• Building the ZoomFloppy
Commodore floppy drive architecture

- The first dual CPU home computer was not the C128 (apologies to Bil Herd)

- 2040 dual drive
  - 6502 to talk to the host
  - 6504 to do access the media (GCR encoding)

- IEEE-488 (GPIB) interface to PET
Commodore floppy drive architecture

- **1541 series (1540, 1541-II, etc.)**
  - Cost reduction is a kind of performance
  - IEC bus is cheaper serial version of IEEE-488
  - Merged both CPU functions into one with interrupt-driven task queue

- **1571 series**
  - Multiple floppy formats (MFM)
  - Burst mode using SRQ line
Copy protection schemes

• Best ones depend on the hardware limits of drive

• 1541 limits
  – 2 KB of RAM, 8 KB ROM
  – No index hole sensor (soft-sectored)

• Whole-track custom format (RAM too small)
  – v-max: fully custom encoding, 10-bit sync marks
  – Epyx (newer)

• Track sync (no IHS)
  – Bump head between tracks and verify data found at proper location
  – Fat tracks (Activision, EA, RapidLok)
Copying protected floppies

- As opposed to cracking protection schemes
- Hardware mods
  - Maverick 8 KB drive RAM expansion
  - Store entire track at once in drive, stream back to host
  - Detectable by checking mirrored addresses
  - Burst Nibbler parallel cable
  - Attaches to unused 8-bit port on VIA
  - Sends one byte at a time, sufficient for full media bitrate
  - Add an index hole sensor
  - 1571 has one built-in for MFM mode
  - Or clever in-drive timer routines

- nibtools/mnib have support reading and writing protected disks
  - Thanks to Peter Rittwage and Markus Brenner
1541 hardware mods
Interfacing with a PC (old way)

• Printer port (LPT)
  – 8 bits bidirectional plus handshake signals
  – Perfect for IEC bus control + parallel transfer

• Problems
  – XA1541, XM, XE, X, P-variants, too many!
    – Caused by evolving and differing voltages on PC ports
    – Opposing goals of low cost vs. functionality
  – Doesn’t exist on modern PCs and laptops
    – Also Mac hardware
    – Anything non-x86
  – Fast and low-latency, but too slow for some tricks
    – 1 µs per access (1 MB/sec max)
• USB floppy interface by Till Harbaum
  – Cheap
  – All through-hole parts so easy to build
  – Open-source and works well

• Limitations
  – Slow
    – Software USB decoding
    – Serial slower than with an X-cable, parallel transfers as slow as serial
    – Can’t support parallel nibbling and limited RAM so no copying protected floppies
  – USB control transfers only
  – Not readily available

• Good for its time, especially if you just want to transfer files to unprotected disks
Enter: me

• In 2005, got an XAP1541 and began archiving floppies for C64Preservation.com
  – “Just how did some of those more complicated copy protection schemes work anyway?”
  – Reverse-engineered from original image using VICE drive monitor

• Contributed some fixes to nibtools

• Then PC with DOS and printer port died
2008: “Hey, there’s this new Atmel microcontroller with hardware USB support”
- Ported xu1541 firmware to the AT90USB, replacing the software USB stack with hardware routines
  - OpenCBM plugin interface so standard tools all work
  - LUFA library by Dean Camera was very helpful
- AT90USBKEY devel board: $30
  - Built a custom cable to connect to existing XAP1541
  - No PCB design or difficult soldering, just a cable
• More debugging and announced Jan. 2009
• Problems
  – Works fine but have to replug if USB transfer interrupted (^C)
  – No nibbler support, control messages too small for USB overhead (8 bytes each)
  – Only two people built cable (Womo and Christian V.)
• New USB protocol
  – USB bulk transfers (32 or 64 bytes)
  – Inline byte handling (no bucket brigade in RAM)
  – Start/end markers to detect interruptions and reset cleanly

• Finished on train in Europe (fall 2009)
  – Fast d64copy
  – nibtools works!

• Problems
  – Infinite listener hold-off
    – To support printers or other slow devices, you have to wait forever for drive to respond
  – 3.3v level mismatch gave some idle current but still safe
  – Still only two users
Building the ZoomFloppy

• Daughtercard approach
  – New Bumble-B board: $15
  – Would have plugged into this IEC/parallel adapter
  – 7406 for bus isolation

• Assembled several prototypes and works well
  – “Anyone want to build this thing?”
    – Jim Brain: “ok”
Building the ZoomFloppy

- Fully custom PCB by Jim Brain
  - Connectors for IEC, parallel (multiple)
  - Connectors for future IEEE-488 support
  - Fits in a nice enclosure
  - Packaged and for sale (hopefully, soon)
IEEE-488 future support

- Implemented in the XS-1541 currently
  - Thomas Winkler created this device

- Future support planned in xum1541
  - XS-1541 is open source so we can use its routines
  - Autodetect for different cables already designed in
  - ZoomFloppy has connector pads for it
Teaching the 1571 new tricks

• 1571 is quite an interesting device
  – Multi-mode like a C128
    – 1541 compatible (1 MHz, 2 VIAs)
    – 1571 (2 MHz, 2 VIAs, 1 CIA)
    – MFM (WD 1770)
  – Index hole sensor

• Burst mode
  – SRQ is previously unused IEC line
  – Reads at 3.5 KB/s over serial bus but only to C128, writes at old 400 bytes/s rate
  – Backwards-compatible with original transfers
  – Looks interesting, but we need 40 KB/s to keep up with the drive’s raw bitrate
Is it possible?

- We need 40 KB/s (25 µs per byte)
  - Plus a few clock cycles to read a byte from the shift register, toggle handshake lines, loop

- Max CIA transfer rate
  - “Theoretically it is even possible to realize bus transfers at up to 60,000 bytes per second with the C-128’s fast bus hardware.” (1571 Internals, p. 148)
  - Max parameters
    - External clock: 2 MHz
    - Count down timer: start at 1, toggle output on 0
    - 500 Kbits/s (2 µs period)

- Theoretically, it is possible to transfer at 62.5 KB/s, leaving some overhead room
Apparently this is possible

16 μs/byte = 62.5 KB/s
Implementing SRQ nibbling

- A lot more work to do before ready to release
  - Implement drive code
    - Lots of changes needed in 1571 mode like different sync detection
    - Want to keep index hole sensor support also
  - Modify nibtools to load separate IO routines based on drive capabilities
    - Need to autodetect missing SRQ line
    - Fall back to parallel mode automatically

- In the end, all Commodore protected floppies could be archived and remastered with a ZoomFloppy + 1571
  - No hardware mods
Thwarted by Playstation 3 hacking

• Final PCB almost complete, looking for parts suppliers
• Then PS3 USB exploit released
  – Just happened to use the same Atmel chips as us
  – Supply went from “great” to “none” in weeks
    – AT90USB162: Dec 5 2010 delivery
    – ATmega16U2: Nov 15 2010 delivery
• Hey, pirates, there are other exploit delivery methods
  – D2groove: D2Prog (PIC18F)
  – Dingoo A320 console
  – TI-84+/SE calculator
• Note: we don’t condone piracy
  – But get off my lawn!
Conclusion

• xum1541 firmware already available (GPL)
  – Fast IEC transfers
  – Even faster transfers and nibbling with parallel cable
  – Easy to build if you have some soldering skills
  – Stable and well-tested

• New firmware features on the way
  – SRQ nibbling on the 1571
  – IEEE-488 drive support

• ZoomFloppy should be available for sale soon!
  http://root.org/~nate/c64/xum1541
  http://jbrain.net/

• Many thanks to: Wolfgang Moser, Christian Vogelsgang, Jim Brain, Peter Rittwage, Spiro Trikaliotis, Thomas Winkler, Joe Forster, Till Harbaum, C64Preservation.com, OpenCBM