TEENSY++ USB CDROM EMULATOR

V0.9.1 – svn68

(README rev 1.2)

TEENSY++ BASED USB CDROM EMULATOR USING MINI SD CARD FOR BACKEND ISO STORAGE. HANDLES CD'S AND DVD'S UP TO 4 GIG IN SIZE. SWITCHABLE ON THE FLY – NO MORE STACKS OF CD'S

USB CD EMULATOR

TEENSY++ VERSION 0.9.1

Hardware Hookup

Hardware needed:

Without LCD

- Teensy++ 1.0 or 2.0
- miniSD for teensy
- miniSD card 2-32gig
- 1 switch - momentary push button

Just use the setup procedure here

For the button – just solder one side of the switch on Port E, Pin 0 The other pin is soldered to gnd.

(no need to change startup.cfg file)

With LCD

(same as above) plus…

Standard hd44780 lcd 1-4 lines, 16-40 chars wide

LCD Hookup as follow:

- Pins PC0-PC3 are connected to LCD D4-D7
- Pin PC4 connected to LCD RS
- Pin PC5 connected to LCD enable
- Pin PC6 connected to LCD ebable2
- Pin PC7 connected to LCD RW

Note: I use a 4 x 40 lcd on my development board, that way I use the other space to debug with. But the emulator only addresses 1 x 16.

(need to change startup.cfg file – line3 – 11016)

Firmware Setup:

Read here and download loader here. Then download the efl file here v1.0 for teensy++ 1.0 and v2.0 for teensy++ 2.0 and follow the directions from the first link. It should only take a few minutes. If there are any problems please post to the forum here.

SD Card Prep

As of 0.9.0 mini SD cards 2-32gig in size are supported (SDSC and SDHC).

Note: Any writing of big files like ISO's should be done in the computer first using a sd card reader. The multi-sector writing is not complete yet, so writing is slow.

Format SD card as FAT32 w/64k clusters. The bigger the cluster size the faster the sd card enumerates when used from the emulator in ms mode.

Once formatted, create the ISO fold on the root. Start coping iso files to ISO folder.

Make sure all iso files are in standard 8.3 file format- LFN not supported yet.

Put card back into emulator and plug emulator in. At which time the emulator will see a missing startup.cfg file and write a new one on the root of the sd card with defaults set.

After your computer detects the emulator and see it's in ms mode you should have access to the card from the emulator – this is a good time to set your lcd options if you have one (line3-11016).

Start using the emulator. Use interface menu to select an iso, and switch modes – back to cd.

ENJOY ☺ Post results here

Menu interface

The current menu interface is based off of the 1 button 1 led concept.

The led is used to feedback to the end user on what's going on.

LED Blink Meanings:

- Pre-startup - Led Stays on until it sees and init's a SD card
- Startup - Led blinks mode 1=CDRom, 2=MS -pause- Led blinks cd number
- Feedback=6 rapid blinks used for user feedback while in menu
- Data Read = Led blinks on read access to sd card

Main Menu Control:

- Hold button till led blinks once and release - ISO Selection menu
- Hold button till led blinks twice and release - Mode Switch (cd/ms)
- If held for more then about 6 seconds, menu loop resets (with feedback blink)
- Upon entering a menu selection (iso or mode) led will produce feedback blink then blink the menu selection 1(iso) or 2(mode)

ISO Selection memu:

- Upon entering iso selection menu – emulator will blink the number of iso's available
- Press button x number of times to select cdrom number
- Upon selection timeout about 6 seconds, Led will give feedback blink followed by the number selected, menu will return to idle state and load new cdrom.
- Led will also blink back the number currently selected after 2-3 second pause after pushing button, that way you know exactly which one will be loaded.

MODE Switch:

- no input just toggles between modes and menu returns to idle state

Note: Menu Changes with LCD support. No Startup Blinks on mode or iso. While in ISO switch mode the LCD is showing the current iso and shows the iso being switched to so less led feedback. It's more of a smoother interface with LCD support turned on. Check out bit 0 of line3.

Background

This documentation covers the teensy++ usb cd emulation project, a project that started back in '09. At the time this started out as a thought stuck in my head. A thought that what's missing for a service tech is an easy way to carry all your cd's around on service calls without worry about weather I scratched, lost or broke any throughout the day. Not to mention that is seems a good portion of the time the customer had a cdrom that doesn't read anymore.

So this project needed to be able to boot cd's, be external and able to carry around all the cd's on one medium.

After some thought on the idea I ordered a teensy++ v1.0 board and dove into the biggest most mentally challenged project I have done to date. Not only did I need to sharpen up on my coding skills (which I haven't worked on in about 10years), I also had to learn the details of iso file structure and the anatomy of a cdrom drive. I read allot of stuff about SCSI / FAT32 / ISO / USB / ATAHDD / SD Cards etc… It was a daunting task at first.

Jumping into coding for the avr chips took a little working with. I'm very behind the times on coding for avr chips. I haven't coded in years and have a full time job and family that keeps me busy most of the day. What did I jump into!!! But I found time and the patience to learn and move on. Eventually had my old 4x40 lcd screen working – carried that screen around with me for years knowing I'd use it one day. I figured the lcd screen was the first thing that had to be working – that way I was able to debug on the fly by printing out results to the screen. It was a smart idea, I needed all the info I could get.

My first big break was stumbling onto LUFA. The usb interface is complicated at best! Thanks to Dean's LUFA usb framework I didn't have to worry about writing my own usb interface. He already had example codes using the dataflash chip used on atmels devel board. It used scsi calls, read and write. It got me almost half way to my goal!! At that point I just needed to add my own storage medium (hdd / sd), some sort of fat32 read/write coded, figure out how cd rom drives work and figure out iso files.

The storage medium came next. At this point I haven't even though about sd card as an options. I was ambitious I wanted to have 10's of gig of storage – it had to be a hdd. Well – let me tell you there are allot of wires used to talk to the ata interface on a hdd. It took most of the remaining io lines left on the teensy++ board. The ATA library I used was one I found on AVRFreaks.net Web site. The library incorporated both ata and fat32 layers. That helped allot. Now I don't have to write any code for the low level storage medium access and don't have to write any midlevel fat32 code either. There were a couple of bugs I had to figure out with the code, but It saved a ton of time. Also had to make a special cable to use as the hdd interface to the teensy++. Using a standard ide cable – cut one side off, and tinned all the wires needed to connect into the breadboard. Then used a laptop ide->2.5in hdd adapter on the other end to interface the donor 80gig laptop hdd.

Buttons, I need buttons. It was time to add some sort of user input. I ended up reusing 4 reset switches from 4 donor computer cases. They worked perfectly. Also found a good debounce code library on AVRFreaks.net.

Then came the part where I was on my own. This is the part that ended up giving me more grey hairs. The Idea that this hardware / code framework could all be glued together in a way that it acts like a cdrom drive. Talked with Dean a couple of times for usb code pointers, ended up getting it to look like a cdrom to the computer. At that point it was pretty easy. Watch what scsi codes are requested then look them up with the appropriate pdf file, then figure out how to code the correct response – repeat. After I had enough core cdrom scsi cmd's working it was time to take the iso file and figure out how to convert the scsi reads to iso sectors on the hdd. And the rest is history.

As of Version 0.9.1 the project is going in a cheapest is best approach. The hdd is replaced with sd(spi mode). No more time consuming custom hdd cables to be made or expensive hdd drives need to be sacrificed. The 4 buttons were cut down to 1. The LCD isn't needed anymore. Just watch the LED for feedback. This was all done to make it easy for anyone out there to build there own usb cd emulator.

Teensy++ CD Emu - Change Log

teensy++ CD Emu - 0.9.1 (Released 2011-04-23)
==========

[General] startup.cfg major change

teensy++ CD Emu - 0.9 (Released 2011-04-10)
==========

Inital release of the grub-less menu control

[ISO Processing] cd switching using sense instead of usb disconnect/reconnect
[User Interface] Move lcd for debugging, updated - moved to port c 0-8
[General] Change the version number - still reported at 0.8beta from a year ago
[User Interface] Remember last loaded cdrom and start selection from there
[Fat Interface] Slight problem with onboard fat, corruption happens around 2 gig full

[General] Lockup and have to restart the emulator after first insertion of a newly formatted sd card
[User Interface] change 1 button interface for faster input when using lcd

References:

USB CDROM EMULATOR home, forum, bugtracker, svn

LUFA – home

Teensy++ home

AVRFreaks.net