Reverse Engineering the iPod Shuffl remote protocol

David Carne

Background

On March 11th, 2009, Apple introduced the third generation iPod shuffle. Sh regarding the presence of a proprietary chip inside the shuffle headphones, sa DRM measure. The "DRM" argument was unsupported by the facts, given that with a non-remote headphone connected, albeit with no remote functionality dismissed shortly thereafter by reports that the chip was simply a "Dual Mode

I purchased an iPod shuffle in June of 2009 to use while running, since my However, the first time I took the shuffle with me, the headphone remote died remote module was not designed to deal with sweat, or moisture of any kind. design flaw for a product advertised for use during exercise. Now that I had a what else was there to do but hack them! I only got around to finishing and w hence the delayed creation of this report.

Disassembly and Hardware Overview

Disassembly of the shuffle headphones is simple - the enclosure snaps toget and the remote control board sits within. No photos were taken of this stag the process, but a disassembly photo can be found at iFixit. Once disassemb the remote board can be desoldered from the two wires attached to it. Figur and 2 show the button side and the circuit side [hereafter referred to as "Top" "Bottom"] of the remote.

The Top of the remote contains 3 sets of pads, each composed of an outer r and a centre contact. Over each pad a metal dome held down with an adhe pad is placed, and the dome, when pressed, makes contact between the outer r and the central pad.

On the bottom of the board there is an integrated circuit, a diode for reve polarity protection and 7 passive components. Two connection pads can be fo on the bottom of the board, and the headphone cable is soldered to these pa See Figure 3 for a connection diagram.

Fig. 4 is a colorized image of the tracks and components, as viewed through bottom of the board. By following the tracks, it becomes apparent that play/pause button simply shorts the CTL line and the GND line. The Volume and Volume down buttons are connected to the proprietary IC and functiona cannot be determined from simply looking at the hardware.

Investigating the Signalling

To figure out the behaviour of the Volume Up and Volume Down buttons, we m look at the signalling as it is on the control line. The control line is driven from shuffle with a known impedance, such that if a load is placed on it, the voltage the control line will drop. See Figure 5 for captured waveforms of up/down but presses. Volume up was measured to cause voltage droop from a nominal 2. down to 1.68V. Volume Down was measured to cause a voltage droop to 1.5 The voltage droop was present for the entire duration of the button push. At point, the meaning and source of the 4.8 millisecond grounding of the signal unknown.

To verify the behaviour gathered so far, I built a test circuit, described in Figur I left the shuffle remote in-circuit so it could generate the "power-up" signal [described later]. This circuit worked as expected, and it even worked perfe with the original remote removed, as long as the remote was attached at powe to identify the remote to the shuffle. Interestingly, the 4.8ms low-pulse was present, with only the pushbutton/resistor circuit connected, without the orig remote. That indicated that the shuffle was actually generating the low pulse, the remote. This may serve as an acknowledgement of some kind. From now this kind of low pulse will be termed an "ACK" pulse.

Power-On signalling

Given that the remote signalling works after power-on authentication, the r logical step was to look at the power on signalling. Figure 7 is two captures of Shuffle power-on behaviour. The upper signal [in grey] is a succes authentication of a remote, and power is kept on to the remote, from the shu At bottom is the same power on behaviour, except without a remote connec The waveform seemed fairly simple, and I figured that in this case, the "low" pu was likely generated by the remote, as I couldn't see any other distinguish features in the waveform.

I proceeded to build increasingly complex circuitry to accurately mimic t authentication [RC circuits to get rise times right] culminating in the wav emulated waveform in black overlapping the original remote waveform in grey and out different impedances, with timing controlled by a microcontroller. authenticated as a remote, despite the high level of similarity between the orig

To determine where the low pulse was coming from, I monitored the curr flowing to the device - a simple 1K resistor works as a current sensor. T confirmed my suspicions that the low pulse was actually another "ack", and generated by the shuffle. So the question then was - what the heck is the shu seeing in the signal?

Power-On signalling - take 2

After digging around a bunch, and building an amplified current sensor, I fo the power-on detection was actually a 100mV/10uA ultrasonic chirp occurring msec after power up. This would seem to coincide with the mention "ultrasonic" by iLounge. Figure 9 has a blown up view of the chirp. In figure 9, upper signal is voltage, and the lower is current. The current signal, measu with a 1k resistor, was run through a 5x preamp made from a precision op-am bring it up to the level seen in the figure. The ultrasonic chirp is actuall dual-frequency chirp, consisting of 1.5msec of 280khz, followed by 4.6mse 244khz. [Doing a binary-search by hand using the scope trigger to find transition point is no fun].

There is a configurable pulldown applied by the remote during the period bet ack pulse. The shuffle expects this pull to be gone after the ack pulse, and ope on. The shuffle will not recognize the remote without the presence of this pulld

The shuffle changes the power supply after the ack pulse. During the power- around 2.72V, and rests at 2.4V with the expected pulldown. After the power-c at 2.08V. [See Fig. 8 for detail].

Emulating the Remote

Now that I knew how to successfully authenticate, I could extend the circui Figure 6 to add the power-on chirp, and power-on pulldown. For initial testin used an ATmega88 microcontroller to generate the chirp and control pulldown, as I had a bunch kicking around in my parts drawers. The downsid using that part is that it won't run on the voltage or current the shuffle can sup meaning for this test, I used an external power supply.

First off - the microcontroller had to detect when the power came on from

shuffle. V_{ih} for the ATmega88 operating at 5V is ~ 2.6V, so I couldn't use a regu
IO pin to watch the line. The ATmega88 has a handy internal analog compara
and it also has a convenient 1.1V reference. The 1.1V ref can be connec
internally to the positive side of the comparator, so I connected the CTL line
AIN1, Pin 13, which is the negative input to the comparator. To watch for
rising edge, the firmware polls bit AC0 of the ACSR register, watching for it to
low [to indicate the input has risen above 1.1V]. At this point the firmware sta
timing to the start of the chirp + enables the pulldown.

The controllable pulldown is simple - a 17.2k resistor is connected from CTL
the drain of a 2n7002 mosfet, which in turn, has its source tied to GND. [See
10 for more detail]. The 2n7002 is driven via a 100ohm resistor by pin PC5 of
ATmega88. Driving PC5 high will enable the pulldown, and driving PC5 low
disable it.

Creating the chirp is slightly more complex, but the onboard PWM hardware
be used as a starting point. I used the Fast PWM mode, and OCR0A + OCR0B a
frequency generator, and then filtered out the high frequency components
quick thrown-together bandpass filter is shown in the schematic. The b
frequency is not in the middle of the filter response [See Figure 11], but the fi
suffices because we only care about eliminating DC and the higher harmon
and attenuating the generated frequency to match the original chirp. At 276l
the filter attenuates the fundamental frequency to 186mV_{p-p}, and the 3rd harmo
is down at 24mV_{p-p} which is sufficient for my purposes.

I created a quick test prototype - Figure 12, which ran the code in Listing 1.

```
#define F_CPU 8000000
#include <avr/io.h>

#include <util/delay.h>

#define CTL_PIN (1<<5)
#define CTL_HI (!(ACSR & (1<<ACO)))

void main()
{
        CLKPR = 0x80;
        CLKPR = 0x0;
        ADCSRB &= ~(1<< ACME);
        ACSR |= ( 1<< ACBG);
        TCCR0A = 0x03;
        TCCR0B = 0x09;
        OCR0B = 7;
        DDRC = CTL_PIN;

        while(1){
                PORTC = 0;
                while (!CTL_HI);
                _delay_ms(4.1);
                PORTC |= CTL_PIN;
                _delay_ms(4.2);
                // 280khz burst
                TCCR0A = 0x23;
                TCNT0 = 0x0;
                OCR0A = 27;
                DDRD |= 1<<5;
                _delay_ms(1.5);

                // 245khz burst
                OCR0A = 33;
                TCNT0 = 0x0;
                _delay_ms(4.6);

                // burst off
                TCCR0A = 0x03;
                DDRD &= ~(1<<5);

                // wait for ack pulse
                while (CTL_HI);
                // kill init pulldown
                PORTC &= ~CTL_PIN;

                // Just hang around and
                // wait for loss-of-signal
                while (1) {
                        while (CTL_HI);
                        _delay_ms(100);
                        if (!CTL_HI)
                                break;
                }
        }
}
```

Listing 1 - ATmega88 demo source.

After a bit of frequency + timing tweaking, the replacement remote reliably
Volume controls, play pause, and power on detection all worked.

Conclusion

So, is that chip a "DRM" or "Authentication" chip? Well, the jury is still out
startup is a bit suspicious, but it may be to indicate the remote type to allow
can behave differently when connected to a phone type device.

Given the simplicity of the chirp - my thoughts are that it is purely an identifie
Any manufacturer that wanted to design a compatible accessory without
reverse engineer the chirp just as I did.

Future Plans

At some point in the future when I have time, I plan to build a low power MSI solely off the power supply generated by the shuffle. I'd like to investigat remote, since that could be made completely waterproof.

[1] http://www.eff.org/deeplinks/2009/03/apple-adds-still-more-drm-ipod-shuffle
[2] http://www.ilounge.com/index.php/backstage/comments/technical-business-details-on-ipod-shuffle-3gs-remot