A Mac IR Detector Reveals All

John Calhoun
Scheherazade Software
softdoroth@aol.com

This article describes a simple, inexpensive hardware device that allows you to record infra-red waveforms from devices like hand-held remote controls. Using a sound-capture program and a Mac with sound-input capabilities, you can use this device as if it were a microphone plugged into the Mac. The infra-red is recorded in the same manner sound is recorded. It ain't a pretty hack, but it's cheap, simple and can give good results

OTHERWORLDLY

Infra-red is rather cool – there is this whole other world around us that we can't see, hear or feel. With all these remote controls, toys and things (and now the Newton) sending infra-red back and forth and to other devices, it's getting noisy in that neighborhood of the spectrum – and under our noses. If you build a detector you've suddenly added a new "color" to your spectrum and have sort of opened up a new door. For example, I built an IR detector and had it running into my Mac – the Mac was running a sound program that would audibly "play" any IR coming into the detector. I kept getting a short blip every second or so for some unknown reason. My Newton appeared to just be sitting there contentedly, but I figured out that it was indeed the source of the stray IR pulse. I checked the Prefs on the Newt and found that I had "Receive beams automatically" checked. I unchecked it and the stray blips went away. I had no idea that all this time my Newt (named Little Nemo, by the way) had been sending out silent pleas for a beam.

To digress even further, I don't like dropping the dash from the word infra-red. I'll use the abbreviated "IR", but not "infrared". When I was a kid, I used to see "infrared" in E*dmund's Scientific Catalog* and thought it was two syllables (pronounced like the word "impaired").

I have a cheap Magnavox TV and a cheap Emerson VCR. They each have a remote control and I wanted to use the Newt to replace these little devices. The IR Seed Kit I was sent had resources for controlling a Sony CD and other things I didn't have. So I needed to create my own resources for my two remotes. To do this I needed to know the pattern of IR pulses that my remotes sent out so I could define resources for them and thus allow the Newt to emulate them – I had to build the device below.

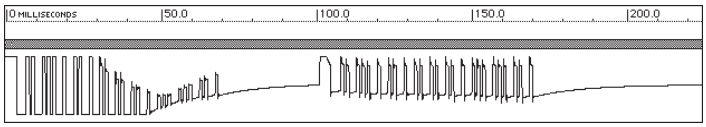

Figure 2 - Basic remote control command.

BUILDING THE IR INPUT DEVICE

Step one was to build a device that could detect, record and display the pulses coming out of my remotes. I recalled a 1992 MacHack hack called "IR Man." In case you've never heard of it, it was an INIT that you dropped into your Mac's System Folder that would allow you to control various Mac functions with a hand-held remote control (you could Play, Stop, and Rewind QuickTime movies for example). The hardware responsible for detecting the IR from the remote was an easy-to-build device that plugs into the sound-input jack of your Mac. The device was pretending to be a microphone as far as the Mac knew. Instead of sending a signal in response to sound however, the device sent current in response to infra-red – how's the Sound Manager to know the difference? So, this little box detects infra-red. To record and display it, you could use any sound editing software.

The device did in fact prove easy to build. Here is the list of parts you need:

- IR Detector Module (part 276-137 from Radio Shack, $3.59)
- 100K resistor
- Phono jack (one that fits into your Mac's sound-input jack)
- 9V battery clip
- Small toggle switch
- Small project case (so that it looks professional)

I suspect that the modest hardware hacker has all the above parts lying around with the possible exception of the IR Detector module. And incidentally, you need a Mac that has sound input capability (let's see, the LC, IIsi, Quadra's, AV Macs, PowerBook Duo's – any others?) and some kind of sound capture software (SoundEdit is the most obvious example).

Figure 1 shows a schematic for the device. The circuit is so simple that a circuit board isn't really necessary. What I did was arrange the 9V battery and IR module around in the project case in order to determine where everything would fit. Then I drilled holes for everything (don't forget a hole for the IR Module so it can "see"). I used a small amount of super glue to mount the IR module in place and to keep the cord to the phone jack from pulling out. I then mounted the switch. With the main components mounted, I started soldering the parts to one another in place. I soldered the 9V battery clip to the switch and IR module (black lead to the switch, red lead to the module – the package the module comes in, by the way, diagrams which lead of the module is which). I soldered a small wire between the other lead of the switch to the IR module. Then I soldered the 100K resistor

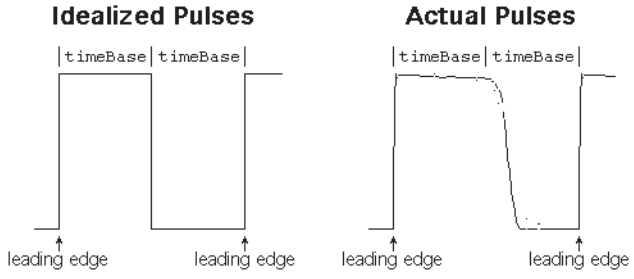

Figure 4

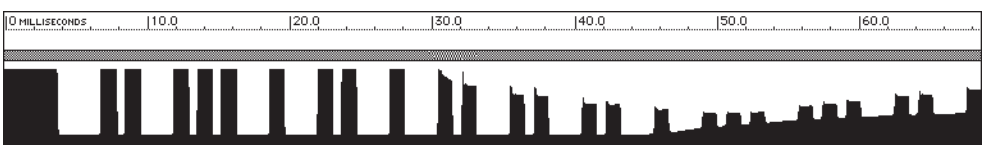

Figure 3

between the IR module and the core of the phono jack. Finally I soldered the shield of the phono jack to the negative (black) lead of the 9V battery clip (attached now to the switch). That's it.

Did it work the first time I tried it out? In fact it did.

TROUBLESHOOTING

When you're done, you should have the IR box sitting near your Mac with it's phono jack cable running around back and plugged into the sound-input port. As a simple test, go to the Sound control panel under System 7 and opt to record a new sound for your System beep. With the IR box switched on, point a remote control at it and hit a few buttons on the remote. You should hear something coming through the Mac's speaker and see the "VU meter" on the Sound control panel jump. Success – hopefully. If you don't see anything, the usual troubleshooting applies (check connections, see that the switch on your IR box is turned on, try another 9V battery, double check all your wiring – there is a list of other possible problems below).

By the way, I have found that the signal coming from the IR box is quite strong – it equates to someone shouting extremely loudly into a microphone. It hasn't harmed my Mac in any way, but as a safety precaution, I've avoided using powerful batteries in the IR box (like alkaline batteries). Instead, I use the cheapest, old-technology batteries I can find (does the color green and the words "Radio Shack" call any images to mind?). The circuit uses a 100K resistor between `Vout` and the jack that runs into the Mac – my pseudo-understanding of electronics suggests that a larger resistor might lessen the strength of the signal. To be honest though, I know very little about electronics and so I can't guarantee that this will help.

If you're detector isn't working very well for you (as in, the waveforms look a far cry from ideal), here are some things I have found that may help you:

- Batteries make a difference – try cheap batteries, fresh batteries, or different brands.
- Keep the box away from your monitor – ELF and other radiations introduce a lot of noise.
- Try turning out extraneous lights, especially fluorescent ones.
- Try firing the remote from different distances.
- Try placing colored gels over the IR detector module to screen out extraneous light frequencies.
- Make sure you record with the highest possible quality (22KHz on my machine).
- View your sampled waveforms as lines, not dots (many sound-recording packages give this option).
- You might be able to clarify your waveforms by using some of the effects in the sound-recording package on them (noise gate, emphasize, and amplify can do wonders).

RECORDING INFRA-RED

With a functioning detector, a Mac with sound-input capability, and sound-capture software you can begin the more tedious and mundane part. Without pretending to know a lot about how remote controls work, let me just begin with a typical waveform I recorded using SoundEdit (see Figure 2). I took the liberty of editing out all the "silence" to the left of the waveforms and "amplifying" the sound at 80% (so you can see the tops and bottoms of the waves more clearly).

Macro Analysis

Something to notice at this point is that (although I swear I hit the button on the remote control as briefly as I could) two distinct waveforms are recorded. In fact, timing-wise, the two waveforms are identical. The distances between the individual waves on the left waveform match those on the right. When you write your Newton application and go to call `SendRemoteControlCode(cookie, command, count)` for this remote control, you want to send the number two for `count` for the waveform above.

Here's the structure for an IR Toolkit infra-red command:

```
struct IRCodeWord
{
    unsigned long name;
    unsigned long timeBase;
    unsigned long leadIn;
    unsigned long repeat;
    unsigned long leadOut;
    unsigned long transitions[];
};
```

I first make note of the time between the two chunks of data (for the `repeat`). I get something close to 32.8 ms for the above waveform (I've found that once you've determined these two values for a single command on your remote control, all other commands on that remote control are the same). I also need to determine the `timeBase` and convert this 32.8 ms into units of `timeBase` before I can use it. Notice too that after the command on the left, the signal gradually comes to a horizontal line before the repeat starts. You can use the length of this line to determine the `leadOut` value in the above struct.

All right, we've obtained some of that macro-information, so I'll chop the data down to a single chunk.This time, I take the liberty of filling the area below the wave with black in order to make the pulses more distinct (see Figure 3). Remember, this is not the "idealized" waveform, but the real thing recorded straight from the remote, so you're going to get little jags and such in the waveform.

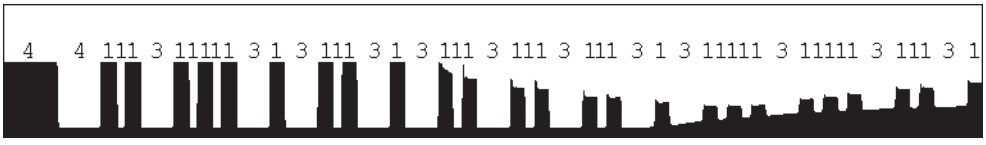

Figure 5 - The transitions.

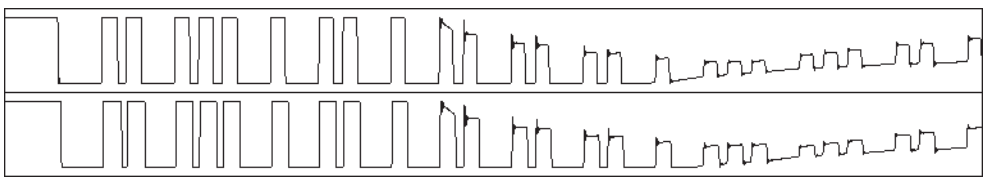

Figure 6 - The original and the pretender.

There are a number of things you're trying to determine at this point: what the carrier frequency is (`timeBase`), what the `leadIn` value is, and what the actual pattern to the pulses is (the `transitions`).

Determining the timeBase

The very first pulse beginning on the left is the lead-in pulse. It's a high pulse and often the longest pulse as well. Right at the conclusion of sending this pulse, the IR signal switches from high to low and is ready to begin the first transition from low to high which happens next. The final transition should end on a low so there is always an even number of transitions.

Since the Newton wants the `leadIn` pulse (and all subsequent transitions) to be measured in `timeBase`, reading the time for the `leadIn` pulse off the sound editing software ruler (in microseconds) is pointless. Instead, we ignore the `leadIn` for now until we can determine `timeBase`. We determine this from looking at transitions.

Note that for this remote control waveform all the high pulses seem to be the same length (measuring left to right – along the time axis). This is the approximate length of our `timeBase`. However, you may have noticed that some of the low pulses appear to be slightly narrower than the high pulses. Since only whole numbers of `timeBase` are allowed here, how should we represent the length of time for those skinny low transitions? They appear to be about 0.75 `timeBase`.

It appears to be an analog artifact from the little infra-red LEDs glowing for a brief period of time after the power to them has been cut off. I suspect it's of no consequence because the receiving hardware is just looking for the leading edge of the high pulses and doesn't really care how brief the low pulses are. To keep things simple I define the distance between the leading edges of two of those high peaks close together as `2 * timeBase` (a one-unit high transition followed by a one-unit low transition).

If you having trouble determining the `timeBase` for your remote control, here's a tip. Go through your waveform and look for the smallest distance between two leading edges (see Figure 4). This is probably two, three or four `timeBase`s — different remote controls use different multipliers.

Looking at the example waveform in Figure 3, do you see the first two peaks on the left (they look something like the World Trade Center towers)? The distance from the left edge of the left tower to the left edge of the right tower (one tower plus the space between) equals `2 * timeBase`. Using this as a ruler, we can determine how wide (in time units) the whole waveform is. I get about 76 time units for the length of the transitions (the towers = 1, the skinny spaces between the towers = 1, the wide spaces between the towers = 3, and the first really wide space = 4 – I told you this was tedious).

Now using a ruler to measure the total length of the transitions, I get about 65.0 milliseconds. The Newton wants the time unit expressed in microseconds (1 milli = 1000 micro), so this is 65,000 microseconds. Dividing that by our 76 `timeBase`'s gives us roughly 855 microseconds – that's the length of `timeBase` (the carrier wave period). Now we need to calculate the other command parameters.

leadIn, leadOut and repeat

Looking at the `leadIn` pulse, I measure about 3.7 milliseconds (3700 microseconds). When I divide that by our time unit, I get 4.3 time units. To keep the numbers whole (and since high pulses typically seem to be long) I round down to four.

Going way back to where we measured the time between the two separate commands (32.8 msecs), we can now calculate `repeat` in `timeBase`. I get approximately 38 (32,800 microseconds / 855 microseconds). We do the same thing for `leadOut`, yielding 50.

Perhaps you've noticed that measuring these waves doesn't appear to require remarkable precision. I find any errors I may have from rounding off are of no consequence in the final scheme of things. This is confirmed when I show the Newton's IR waveform I'm creating (the "synthesized" waveform) alongside the original remote's waveform later.

If we fill in IR Toolkit's `IRCodeWord` struct with the information we have so far, it looks like this:

```
struct IRCodeWord
{
    name :=              'play'
    timeBase :=          855
    leadIn :=        4
    repeat :=        38
    leadOut :=       50
    transitions[] := ???? // still to come
};
```

Determining the Transitions

Tedium becomes an art form now — the transitions must be measured. I won't walk through every pulse in the sample waveform but instead just give you some general tips.

- Print the waveform out as large as you can, and print out all your remote control waveforms to the same scale.
- Make a "ruler" using a piece of paper with your timeBase marked in little tick marks along it.
- Measure from leading edge to leading edge. In my sample waveform, all the high pulses are clearly one `timeBase` long, so the only challenge is determining the low transition lengths.

Figure 5 shows my sample waveform marked up with the transitions. Based on that transition data, the final field in the `IRCodeWord` struct looks like this:

```
transitions[] :=

[4,1,1,1,3,1,1,1,1,1,3,1,3,1,1,1,3,1,3,1,1,1,3,

1,1,1,3,1,1,1,3,1,3,1,1,1,1,1,3,1,1,1,1,1,3,1];
```

That's it. Use ResEdit to enter the above data into an 'IRCD' resource (get the ResEdit template for 'IRCD' resources from the IR Toolkit sample code).

HOW DOES IT COMPARE?

Well, below are the two waveforms. The top pulse came directly from the remote control and the bottom one came from the Newton using the data arrived at in this article. Close enough? You be the judge (hint, it has my VCR fooled).

ONWARD

It occurs to me that someone could write a slick hack that automates the entire process of capturing/decoding remote control commands. I've begun such a hack, and at the time of this article, have it recording and accurately determining all the required data with the exception of the transitions (it gets them wrong frequently). This public domain code, designed to run on a Macintosh, is available on the *PIE Developers* 2.3 source code disk. Feel free to use it as the basis for a polished, automated solution to the process I've described in this article. ▲

John Calhoun lives in Kansas and writes games for the Mac and Newton full time in an effort to avoid entering the "real" workplace. He enjoys collecting technological gadgets that allow him to feel superior to his two cats.