The Atomic Pinball Clock

I'm not much of a hacker. I'm more of a hacker wannabe. I follow along with Hackaday.com and Make magazine and similar places and admire the ingenuity regularly demonstrated there. I see lots of things I'd love to try to build or do but rarely seem to be able to get started. What follows is an account of the project I did finally do along with a few of the pitfalls I encountered on the way.

My primary hobby is restoring and collecting pinball machines, predominantly the electromechanical (EM) machines made before the late 1970s when solid state machines took over. While solid state games are cool and all, I find the tangibility of switches, relays, motors and solenoids to be especially alluring. I'm not much of an electronics hobbyist either but reading all those project accounts using microcontrollers had left me with a nagging desire to learn to use a microcontroller for something useful.

It all came together for me a while back when a friend and fellow collector gave me the orphaned head of a 1968 Williams Cue-T pinball machine.

The playfield and main cabinet had long since been lost to history but the head and its contents were still good at least for spare parts if not for something more.

My friend had no use for it but just couldn't bring himself to throw it out. He thought that I could find a use, or maybe a better home for it. It was complete with all its score reels, relays, lights and a stepper unit. It could

easily have been restored had it not been separated from the rest of the game.

It didn't take me long to conclude that the head was too good to part out and that it deserved a better fate. I decided that it needed to be a clock. Actually what I decided was that I needed to learn to design what would make it behave like a clock. In fact that became one of my design constraints. I wanted to keep the existing wiring in the head as close as possible to its original state. Part of the reason was that I might want to be able to reclaim the head should I ever find a Cue-T cabinet in need of one. I also liked the idea that I would essentially have to reverse engineer the interface and design a clock driver that would look to the orphaned head just like the game it was once attached to.

My second constraint was that the clock needed to be self sufficient. I knew that I wouldn't leave the clock running. It would be a little too loud and annoying for that. Once finished, the clock would be more something to show to folks who come over to enjoy the other games and to take to the occasional pinball show. I wanted this clock to figure things out for itself when power is applied, without batteries or buttons or dials to jump start it.

So combining the Cue-T head with my constraints and nagging desire to learn more about microcontrollers I set out to design a clock that would receive the WWVB (or "Atomic Clock") time signal and display the current time on the score reels of the Cue-T head. Pretty simple I thought. Straightforward. Really nothing more than combining bits and pieces I had read about into perhaps a unique combination with a modest Wow factor. And so began the design phase.

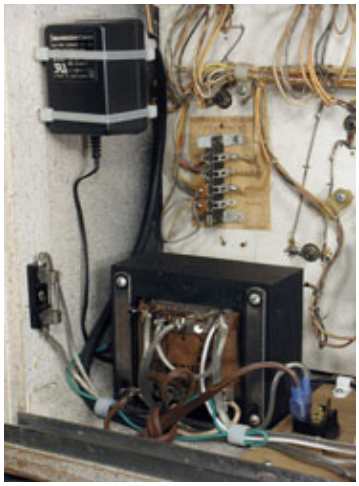
The design

Since most of this was new to me I had a fair bit of research to do. I worked my way backwards from the solenoids and relays in the head. I read up on controlling AC relays with DC signals and ended up combining a few optocoupled triac driver circuits I found. To generate the DC signals I chose the Arduino Duemilenove microcontroller. It had the right combination for me of low cost to get started (since I had no programmer or development board) and wealth of online support. Finally I found an inexpensive WWVB receiver module that I thought the Arduino could poll regularly for the current time. I drew up some schematics, ordered my parts and got to work.

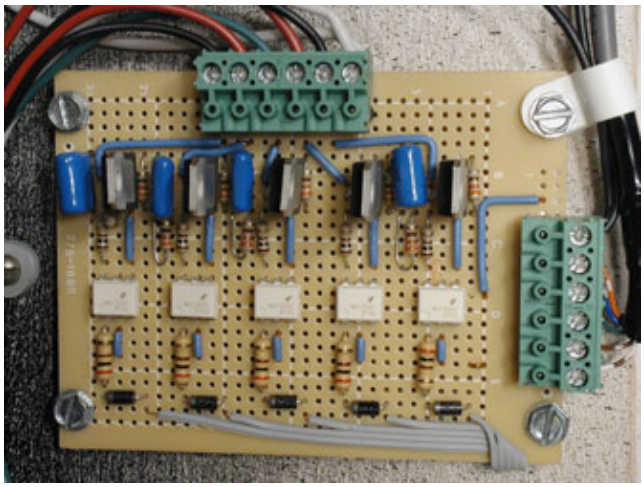
The head has a pair of sockets where jones plugs from the main cabinet connected. Through the jones plugs the main cabinet sent AC signals to advance the score reels and stepper and to power the lights in the head. By tracing faded wire colors and using a meter I mapped out what each of the socket connections was used for and how it would need to be driven by the new clock driver.

The first thing I needed was a way to power the 24 VAC relays and solenoids in the head and the 6.3 VAC lights which illuminate the backglass. Rather than trying to contrive my own solution, I just bought an old transformer from a parted out EM pinball machine from everyone's favorite online auction site and mounted it in the head.

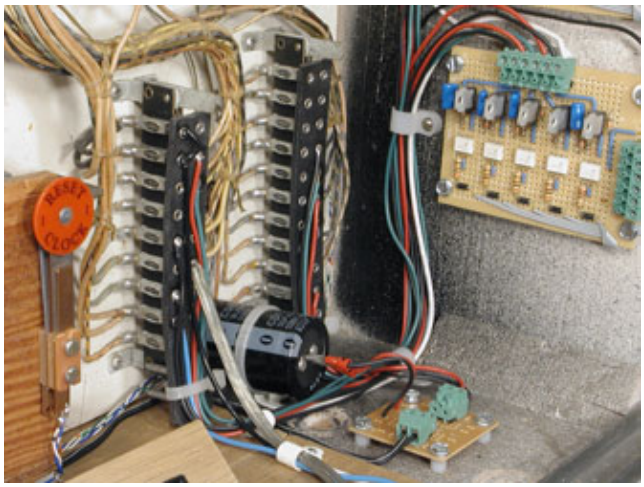
With jumpers from the transformer I could tap individual connections on the jones sockets to verify which relay each one was tied to. I also mounted an old 9 VDC wall wart just above the transformer to supply power to the logic circuits.



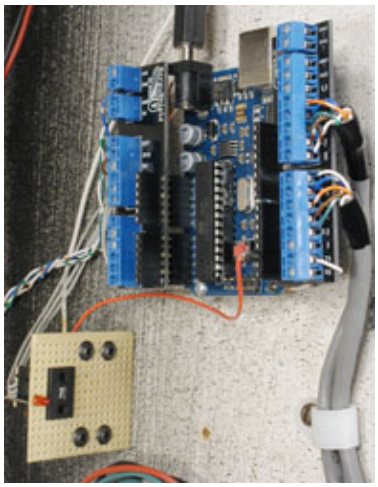
The next step was to build optocoupled triac drivers to switch the 24 VAC to each of the relays in the head. It took a little planning and some trial and error to find a layout that would fit the ten individual drivers I needed onto the two perf boards I bought (although just one is shown here). In fact, board layout and wire routing were two things I learned to pay more attention to as I went along to keep things from devolving into an unmanageable rats nest.



With the driver boards assembled, installed and routed to a pair of new jones plugs set into the sockets I could drive the individual relays by tapping the various logic level inputs with 5 VDC jumpers and get the same results I'd gotten earlier by tapping 24 VAC directly to the socket connections. The big red button is a pinball playfield target that I wired up as a reset switch.



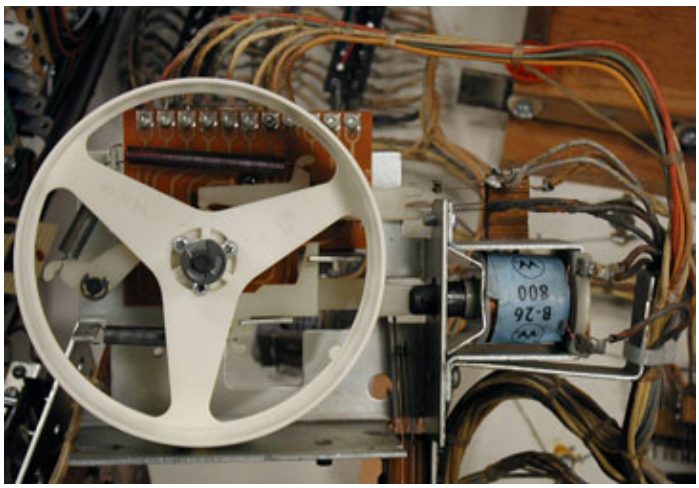
So far so good. It was time to break out the Arduino to drive the logic level control signals.

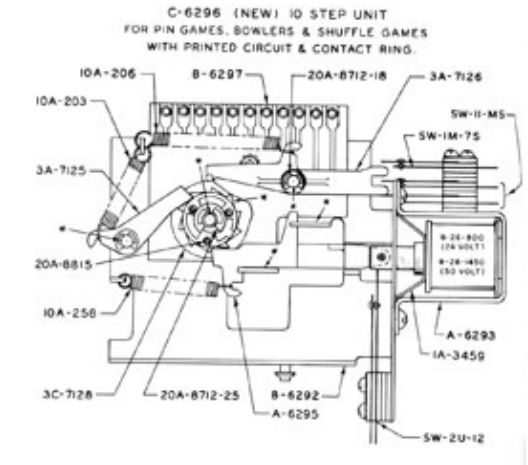


After reading through the startup guide and trying an example or two it took no time at all to code up a simple loop to drive one of the relays repeatedly. I spent a fair bit of time tuning the basic relay driving routine. An EM pinball machine has natural cadence, determined by the speed of the score motor and the shape of its cams, which all the scoring mechanisms follow. The pulses which activate the relays are usually just long enough to complete the required mechanical motion. Any longer and they tend to buzz annoyingly. And the speed at which the score reels and other mechanism advance is constant and familiar. As a collector, matching that sound was important to me even though the score motor was lost with the main cabinet. The illusion of having the head look and sound as if it were still installed on the game was worth the little extra effort it would require.

Once I could drive the relays in the head I wrote the routines to display an arbitrary time on the score reels. In an EM pinball machine the set of score reels is nothing more than a simple, specialized adder. The game requires the score reels to do just two simple operations: reset and advance (increment by one). When the game starts all the score reels are reset to zero. During game play, each score reel is just required to advance to the next digit either due to some event on the play field, or as the carry out when its lower neighboring score reel advances from nine to zero.

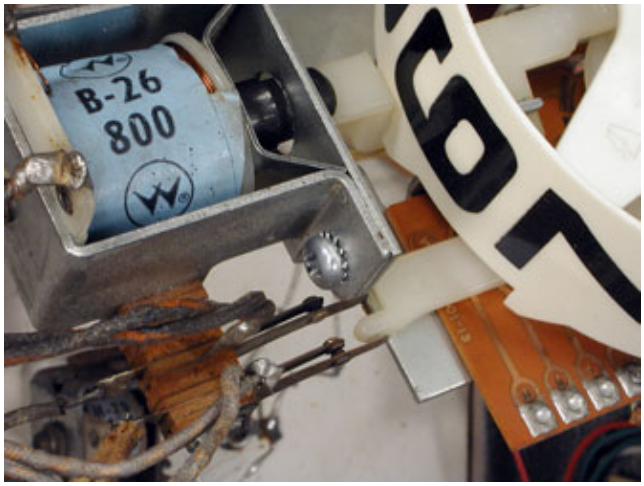
The score reels are mechanical ratcheting wheels driven by a solenoid and springs.



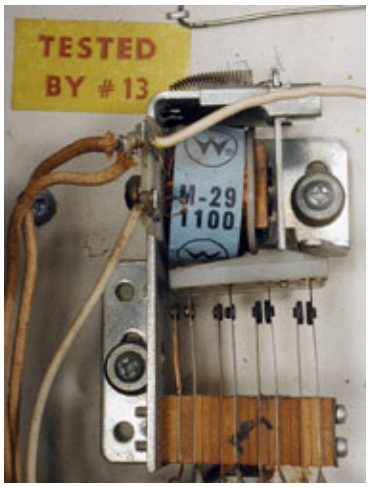


When a score reel solenoid gets an AC pulse, its plunger and the attached ratcheting arm stretch the return spring as they move into position to advance the score reel. When the pulse subsides and the solenoid relaxes, the return spring pulls the plunger and ratcheting arm back into their rest position and advances the score reel

in one smooth motion. To reset the score reels at the start of a game, the score motor in the main cabinet sends ten pulses to each reel through its reset circuit which is wired in series with a switch that opens when the score reel shows a zero. So all the reels advance to zero from whatever position they happen to be in, but no further.



To advance a score reel an intermediate relay in the head is used to close the high current contacts of the score reel solenoid. You can see that the high current contacts on the right of the relay are larger than the normal contacts on the left.

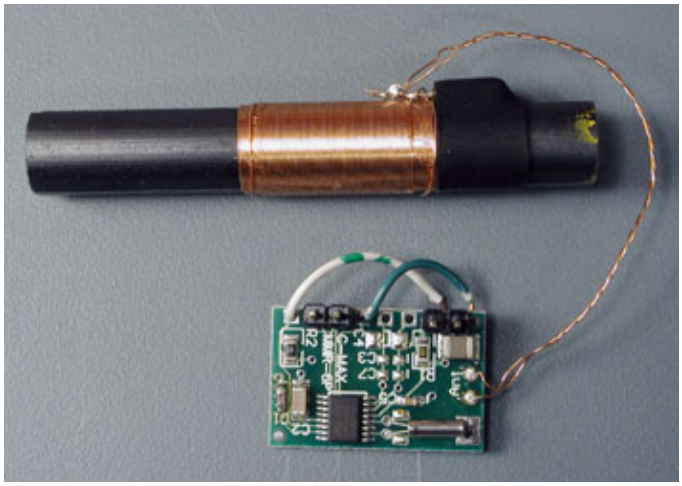


The score reel advance circuits are wired in parallel to the reset circuits and have no series cutoff switches as the reset circuits do. Each advance circuit does use another switch which closes when its score reel shows nine. This switch is used to advance the neighboring score reel on the next advance pulse. For example if the score is 190 and the 10 point relay fires, the pulse created when the relay closes the high current score reel contacts will advance the tens digit from 9 to 0 as usual. But a second pulse will also be created by the 10 point relay if the tens digit 9-position switch is closed which will advance the hundreds digit from 1 to 2 at the same time. In this case the 10 point relay advances two reels to get from 190 to 200.

To use the original game wiring in the head the clock would need to control the four individual reset circuits and the three advance relays (10s, 100s and 1000s) through the jones plug. Note that there is no advance relay for the 10000s digit because the game apparently had no 10000 point target. After resetting to zero the 10000s digit would only ever advance as a carry out from the 1000s digit through the 9-position switch. In the clock design the 10000s digit corresponds to the 10s hour digit of the current time and can similarly be controlled by pulsing the 1s hour digit as needed. Given that the score reels are already wired to reset and increment, driving

them to show the current time is a pretty trivial process.

The last step was to add the WWVB receiver module to the clock.



The module comes with a ferrite rod antenna that looks like it belongs in an AM radio and a three pin interface: 3V, GND and the serial output. The power requirements are low enough that the Arduino can supply the required 3V directly. I mounted it right above the Arduino inside the head, patched it in to the analog inputs and turned my attention to decoding the signal.

The automated WVVB transmission takes an entire minute to broadcast the current time from a NIST atomic clock nationwide from its transmitter in northern Colorado.

Each second another bit is transmitted and together the 60 bits sent each minute indicate the start frame, the current time, the date, and indicators for leap year, daylight savings and other adjustments. The value of each bit (0 or 1) is indicated by the duration of the pulse received in that second: a short pulse for 0 and a longer pulse for 1. The one bit of information not transmitted by WWVB is the time zone, which makes perfect sense since the transmission is intended for a nationwide audience. I added a small BCD rotary switch that the Arduino reads just after reset to determine what offset to add to the decoded time for the selected time zone.

Decoding the signal was pretty straightforward once I read that sampling and averaging the signal would be much more reliable at identifying transitions than relying on individual samples. I thought I was pretty close to being done when I had all the AC drive circuitry in place and the Arduino sending the current time to my

laptop through its USB interface. Then stuff started to happen.

The Pitfalls

At some point late in the WWVB decoder programming stage the power supply for my old laptop gave out. No problem I thought. I bought a cheap generic replacement from everyone's favorite online auction site and after a week or so I was back in business. But not really. Somehow my results got all wonky. I couldn't read the time from the receiver anymore. Had I saved a bad version of the routine? Had I somehow ruined the receiver?

After some testing of the Arduino to convince myself that it was still working I ordered another receiver. It didn't work either. What was going on? In desperation I removed everything but the Arduino and the receiver and tried debugging with just the Arduino's on board LED. It seemed to work fine. Slowly adding more and more back into the system I discovered that things went bad when I tied the Ardiuno to my laptop with the USB cable. Further experimentation revealed that the new laptop power supply was the culprit. The clock would work fine with the laptop on battery power, but not with the power supply plugged in. Somehow my cheap generic replacement power supply was putting some kind of noise out through the USB cable to the Arduino that in turn confused the receiver. Ugh. Another, brand name replacement laptop power supply restored the system reliability and my sanity.

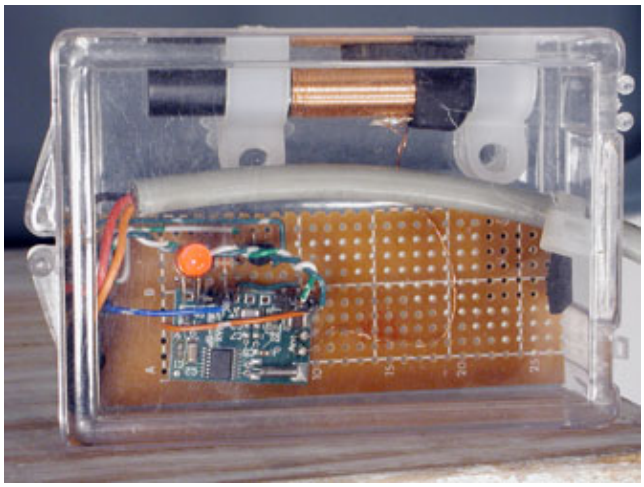
So all that was left was to write the routines to take the decoded time from the receiver and put the appropriate numbers up on the score reels and I'd be home free. The coding went quickly and the initial results looked pretty good. But occasionally I'd get the wrong time on the score reels. Upon closer inspection I found that sometimes I would get an unexpected score reel to advance along with the expected reel(s). Hmm. Did I have an intermittent short?

I also found that once I started displaying the time on the score reels instead of on the laptop, my radio started acting up again. Various experiments led me nowhere and I pondered what might be happening for quite a while. Finally it hit me. Since I wanted to use the original game wiring I controlled the score reels through their intermediate relays which open and close the high current solenoid contacts. These contacts often arc as they open or close due to some combination of the magnetic field in the score reel solenoid changing quickly, and the mechanical bouncing of the contacts as the relay opens and closes.

<embed arcing contact video here>

Diodes are used in more modern pinball machines to suppress the arcing across contacts but older EM games generally don't use them because the arcing has little effect on the components other than pitting the contacts a bit. Could the arcing be the source of my problems?

I removed the radio receiver from inside the head and mounted it in a clear plastic box at the end of a long coax tether that could be placed much farther from the arcing contacts.



That resolved the radio issues. I can imagine that the results I had been getting from the receiver were probably about as good as those of an AM radio in a thunderstorm. I also added a transistor to drive the signal back through the coax and an LED to show me what was being sent back. When the LED flashes at about 1Hz, it lets me know that the receiver likely has a good signal and is sending it back to the microcontroller.

Suspecting that the arcing was causing some triacs to fire when they shouldn't, I looked for ways to make the triacs more resilient. Normally a triac should only conduct the AC between its A1/A2 pins with an applied gate current. With a little more research I found that a transient AC spike with enough di/dt (or a high current slew rate) could cause a traic to conduct even without an applied gate current. This is apparently due to capacitive coupling between the A1 pin and the gate. The solution is to add a snubber (or series resistor and capacitor) between the A1 and A2 pins, presumably to smooth out the AC spikes and reduce the worst case di/dt. I couldn't really quantify what the AC spikes might look like so I experimented with some high voltage capacitors left over from my tube radio restoration days. Early results improved the situation but didn't fix the problem until I added snubbers with .022uF capacitors to the score reel triacs. After that everything started working as expected.

Pleased with myself and proud of my clock, even though it took much longer than anticipated, I offered to take it to California to be part of an exhibit at a pinball show. Arrangements were made and I took the clock out to the show only to find during set up the day before that it didn't work. The radio wasn't receiving (no flashing on the receiver LED). Was it the building? Was I in a dead spot? It wasn't until much after I'd returned home that I realized what had gone wrong.

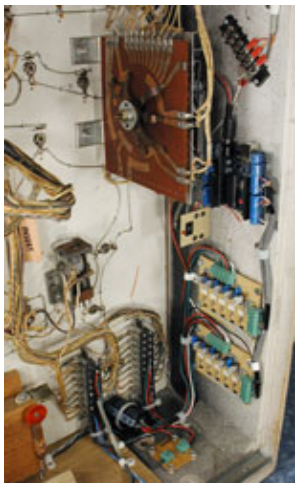
It turns out that while the WWVB signal is broadcast nationally and continuously, local reception can vary wildly. In fact the coverage area expands and contracts each day as the radio signal propagation conditions change. Generally speaking, most of the continental US can receive the signal from Northern Colorado overnight as the coverage area expands. But during the day the coverage area contracts to something less than the whole country. If you live in the middle of the country as I do, you can probably receive the signal at any time. Those on the coasts however are likely to have outages during the day.

So my design which relies on a fresh signal update each minute falls down at the periphery of the WWVB continuous coverage area. It turns out that the designs of commercial "atomic clock" clocks and watches allow their clocks to run independently until they receive a good WWVB signal at which point they resynchronize.

Final details

Once the basic clock (receiver, microcontroller and score reels) were working and playing well together, it was time to finish out the clock with a few extra details. I added the big red button mentioned earlier to restart the clock and send it though its animated reset sequence. This is mostly to make it easier to demonstrate how the clock works when I'm explaining it to someone because otherwise, the clock sits idle most of the time.

I also incorporated the big stepper unit in the upper right corner of the head.

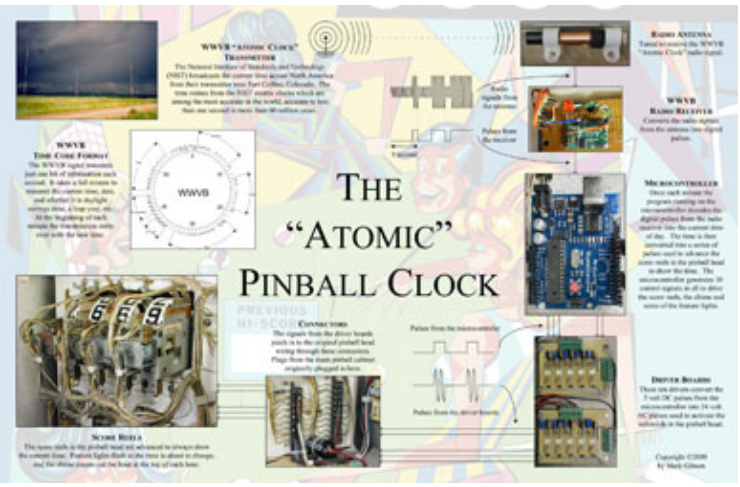


The stepper unit is used to show the number of balls left to play in the complete game and like everything else was working just fine. It seemed silly not to use it so I have it count out the minutes (0 to 9) of the time about to be displayed on the score reels.

The one change I did end up making to the original game wiring is to change how the chime works. Originally the chime would fire any time the 10 point relay fired to advance the score reel. I wanted to have control over the chime so I added a relay and wired it to the chime in place of the 10 point relay. That allows me to do things like to have the chime count off the hour at the top of each hour after the score reels have been set.

I added a little serial LCD display to the inside of the cabinet because I wanted to learn to use one, and to make the clock a little more self explanatory to the casual observer.

I also added a clear polycarbonate back to expose the inner workings and a nice poster board to explain how it all works when it's set up at shows.



Other improvements made to the software are strictly for me since chances are good that no one will ever realize they're there. I've added some error tolerance to the decoding routines because occasionally the receiver does glitch for whatever reason. But of the 60 bits it takes to transmit the current time, the routines really only need about one third of them. Even if there is an error in receiving the time, the routines can register that another minute has passed if they can recognize the next frame marker . So the routines keep track of the current time and the expected next time and allow for a programmable number of consecutive bad time transmissions. If the clock knows the current time it could in theory just watch for the new frame markers and derive the time from those. Once the software has passed the threshold of consecutive bad time transmissions it resets the clock and waits for the next good signal. I've also added code to handle Daylight Savings Time adjustments just in case the clock happens to be running overnight on the transition days.

On the cosmetic side of things the original Cue-T backglass was too badly damaged to use. Much of the ink had lifed from the glass or was missing altogether. So I scanned the original, cleaned it up in a graphic editor and had it printed on translucent film complete with a black mask layer to use in place of the original. I had to make one small change to the backglass art though to make it work as a clock. The original backglass has windows for five score digits to show through even though there are only four score reels. The reason is that the least significant digit is a dummy made to look like a score reel that always shows a zero. In fact 1968 (when Cue-T was made) was about the time when pinball manufacturers decided that games needed to score more points. Most just kept the same number of score reels and added a dummy in the lowest position to make it look like 10 times more points were being scored. Call it score inflation. In any event, I had to black out the window for the least significant digit in my new blackglass so only the four digits of the current time would show through. I suppose I could have also added a colon to the backglass between the windows for the hours and minutes but that was more of an alteration than I wanted to do. I wanted it to look as much like the original as possible.

So there it is. My big goofy atomic pinball clock is more boondoggle that precision time piece, but that’s ok.

Clock Demonstration Video

I spent more time and money on it than I thought I would but I learned a lot and folks seem to enjoy it when they see it. In fact it's fun to set it out and watch people's reactions. Most need a few minutes or even a hint or two to figure out what's going on but the reaction after that is usually very positive.

For those who need more detail I've also posted the driver schematics (page 1 and page 2) and the source code.