http://hecgeek.blogspot.com/2017/10/wall-o-matic-interfac e-1.html
http://hecgeek.blogspot.com/2017/10/wall-o-matic-interfac e-2.html
http://hecgeek.blogspot.com/2017/10/wall-o-matic-interfac e-3.html

I am trying to get archive.org to create a copy of these files but I am currently having a problem accessing that site.

Attached are 1-3 of the content only of the posts, followed by 1-3 of the page as published.

H.E.C. Geek

Household Enterprise Computing Geek

Sunday, October 22, 2017

Building a Seeburg Wall-O-Matic Interface (Part 1)

Introduction

Last year we moved into a house that had some interesting touches left by a previous owner. Most notably, there was a "play room" decorated in the style of a 50's Diner. It had a booth, table, chairs, and plenty of decoration. We joked that this room's decor was what sold us on the house!

In any case, one notable thing missing from this room was a jukebox! Or at least, taking things slightly more sensibly, the jukebox controller that such diners often have sitting at their tables. It was at this point that I started learning all about the iconic Seeburg Wall-O-Matic.

Flash forward a year, and I was finally in the process of resurrecting my electronics workbench. I started watching a few too many EEVblog videos, built a shelf/bench setup using components from Ikea, unpacked all the gear I'd kept in boxes for a few too many years, and even made a few upgrades.

About Me

Derek

Self-professed ubergeek with a wide variety of technical interests. Can be found evangelizing FreeBSD, tinkering with Solaris, building embedded electronics, developing software, or working on obscure networking projects.

View my complete profile

Followers

Followers (5)

Follow

The one thing I desperately needed? Projects! Given that I knew I was going to have a lot of free time coming up in the near future, I reopened my research into the Wall-O-Matic and began to scour eBay.

Background Research

One of the first things I stumbled across were these commercial "products" designed to provide a modern interface from the wallbox:

CD Adapter
Wallbox2mp3

Unfortunately, these projects were less than desirable for my tastes. I was also looking for a project, not an off-the-shelf solution. These devices also seemed a bit dated, of limited availability, and quite proprietary. They also seemed to focus on playback a bit too "locally," rather than using the wallbox as an actual remote for a real stereo system. My house had in-wall speakers installed in many rooms, including the "diner" room, and I really wanted to use those. Since I had already connected many of my in-wall speakers to a Sonos rig, I kept wondering if there was a way I could just use that.

The next thing I did was dig into these hobbyist projects which seemed much closer to what I actually wanted to accomplish:

Wall Box SONOS Controller [Stephen Devlin]
Seeburg Wall-O-Matic [Retro Future Electrics]
Raspberry Pi Project – A 1960s wallbox interfaced with Sonos [Phil Lavin]

One common theme among these projects was simplicity. Minimal components to interface the wallbox to a Raspberry Pi, and minimal work to control a Sonos system based on the result. They also provided enough schematic and component details to give me a tangible starting point. Even if I decided to take a different path with my own project, at least I had a good foundation to build upon.

Project Goals

So thinking through what I wanted to accomplish with this project, I decided I wanted to build a device that could do the following:

- Provide power to the wallbox
- Read the signal pulses, and decode them into a song selection
- Enqueue selected songs with my Sonos system, simulating the functionality of a jukebox
- Electronically toggle the coin switches, so that inserting actual coins would be optional

At a lower level, I also knew I wanted to take things seriously in the design of the circuit I was going to use to accomplish all of this. That meant:

- Complete and detailed schematic
- Complete and detailed BOM (bill-of-materials)
- Real fabricated PCB (printed circuit board) design

(The BOM and PCB being things that I'd never actually done before. Every prior circuit of mine was a hand-constructed mess of wires on a pre-drilled pad-per-hole PCB. Thankfully, in this day and age, doing it "right" is now quite accessible.)

I'll attempt to break this blog series apart based on the major progression of this project. I may not discuss things in the actual order that I did them, since there was a lot of back-and-forth between the various elements. However, it should flow in an order that makes sense. Most likely it'll be something like this:

- Procuring a Functional Wallbox
- Decoding the Pulses
- Inserting Coins
- Designing the Circuit
- Developing the Software

Posted by Derek at 5:09 PM

Labels: Jukebox, PCB, Seeburg, Sonos, Wall-O-Matic, Wallbox

2 comments:

Steve Hammer said...

Thanks so much for posting about your project. Your pulse decoding circuit saved my bacon on a similar project using a wall-o-matic 100 that I started in December. I tried my own hand at making a circuit for the decoder, but after much testing decided that now matter how much tweaking I did, it wasn't going to work reliably. Sadly I didn't find your post until I had tried it my way. My 3W1 will eventually be hooked up to a Raspberry Pi running Fruitbox, but the premise is similar to yours. A 65 year old front end to a digital device! It's an awesome idea. Thank you, thank you, thank you!

Cheers,
Steve Hammer

8:43 PM

Anthony said...

Following this post ! Very interesting to see how you can do this. I use presently some cdadapter but not really satisfied. If you need funding to help your research, i'm in ! I have a seeburg consolette and it should be awesome to digitalize it, without any damage. Thanks !

5:37 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Awesome Inc. theme. Powered by Blogger.

H.E.C. Geek

Household Enterprise Computing Geek

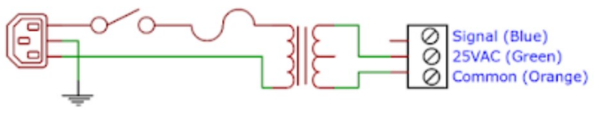
Tuesday, October 24, 2017

Building a Seeburg Wall-O-Matic Interface (Part 2)

Procuring a Functional Wallbox

Providing Power

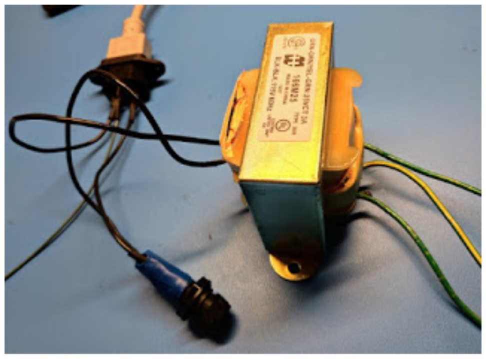
Before you can do very much with a project like this, you need a way of powering up the wallboxes. These things don't run off the mains, nor do they use DC. Rather, these devices run off 25VAC. While you likely don't have a power supply just laying around that can provide this, its pretty simple to assemble one. You really just need an appropriate transformer and a few (optional) support components. This then connects up to a terminal strip right inside the wallbox itself.



In my setup, I used the following components:

- IEC320-C14 Power Receptacle (Q212-ND)
- Rocker Switch (CH755-ND)
- Fuse Holder (283-2712-ND)
- 3A Fuse (283-2841-ND or 283-2917-ND)
- 115V to 25V Transformer (HM556-ND)

I just wired these together as shown in the above schematic, using some heat shrink tubing and electrical tape to cover up the exposed contacts. Specifics aren't all that important here. All that really matters is that you get approximately 25VAC on the output, and won't fry something if there is a short.



(Note: I don't actually have the rocker switch in this version, as I just used a switchable power strip. I do plan to add one on the final version.)

Okay, now to get started...

Seeburg Wall-O-Matic 200

This journey began with the acquisition of a Seeburg Wall-O-Matic (V-3WA) 200 from a rather nice eBay listing. It appeared clean, in good condition, chrome intact, and with the key. On the surface, it seemed like everything I needed to get started:

About Me

Derek

Self-professed ubergeek with a wide variety of technical interests. Can be found evangelizing FreeBSD, tinkering with Solaris, building embedded electronics, developing software, or working on obscure networking projects.

View my complete profile

Blog Archive

Followers

Followers (5)

Follow

Unfortunately this wallbox basically sat untouched for a few weeks, since I still had to buy the necessary components (shown above) to power it up and I was preoccupied with other things at the time.

When I finally powered it up, the first thing I discovered was that multiple light bulbs needed replacing. That much was no big deal, so I just ordered them off Amazon (#51 and #55 bayonet mount light bulbs). I then discovered that the electrical contacts were dirty, the mechanics needed a little fiddling, and it didn't seem to be working correctly.

Thankfully it wasn't too hard to find the repair manual online. Of course it was written in 50's speak, and it was sometimes hard to match the terms and illustrations to what I was seeing inside the actual device.

I spent the next week or so in a state of constant frustration. I replaced the bulbs, cleaned all the contacts, tried to adjust and/or understand what parts of the mechanism I could, kept cursing at the DCU ("dual credit unit") that I was afraid to disassemble, and eventually sorta got it half-working. I got it to the point where I could manually toggle the coin switches and punch in a selection. Of course it would get stuck part-way through the signaling cycle half the time, and I'm not sure if it worked consistently with actual coins. (It was also dirty enough that I felt the need to wash my hands every time I was done fussing with it.)

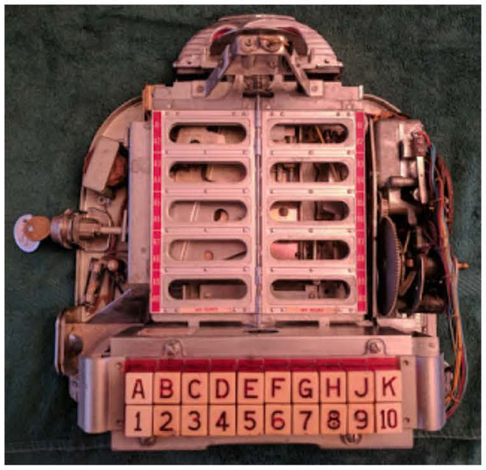
From all of this, at least I learned quite a bit about how these devices operate. These things were designed in an era that pre-dates "electronics" as we know them, and are electro-mechanical in nature. They use a complex assortment of gears, cams, metal strip contactor switches, motors, and solenoids to accomplish what you'd do today in a single $0.50 microcontroller. (Even if it was only the 1970's, chances are you'd do this with a small assortment of transistors and logic chips.)

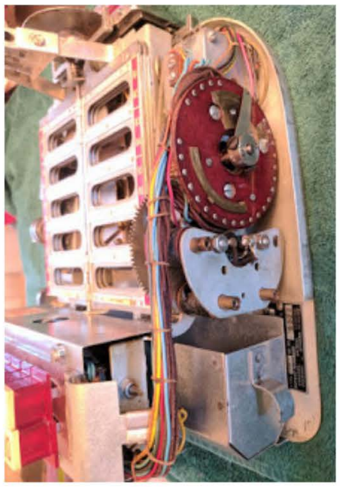
Eventually, I decided it was in my best interest to give up for now. I didn't feel comfortable disassembling the parts that needed the most attention, and I really didn't want to focus all of my energy on this stage of the project. So I decided to just go ahead and actually order a known clean/functional unit, from a dealer that actually specializes in this sort of thing. I can always return to this unit later, and it'll make a nice display piece regardless.

Seeburg Wall-O-Matic 100

This journey continued with me ordering a Seeburg Wall-O-Matic (3W-1) 100 from an actual retro equipment dealer. This time at least I knew I was getting something that had been cleaned and lubricated on the inside, in addition to being in good condition on the outside.

Okay, the buttons could probably use some restoration or replacement, but the rest of it looked excellent. Especially on the inside...





When I powered this unit up, everything magically worked. Okay, I might have had to fiddle with the coin switches a little bit, but those are easy to knock out of place simply by removing the title strip and coin rejector assemblies. Regardless, I was quite happy. I now had a fully functional and reliable wallbox I could use as a foundation for the next stage of the project.

Besides simply being clean, lubricated, and functional, this model had another big advantage over the 200. Its mechanism is a lot simpler. It doesn't have an overly complex "credit unit" in the middle, and I don't think I'd be afraid to try disassembling any of its mechanism if I needed to.

I did later discover this unit had a few modifications done to its coin mechanism, however. It doesn't accept dimes (only nickels and quarters), two of the coin switches were tied together, and one of the coin solenoids was disconnected. I wish these modifications hadn't been done, but they're not a showstopper. I can easily live with them. They basically mean that the device now has only two credit states: A dime adds one song credit, and a quarter adds two song credits.

References

Repair manual for the 100
Repair manual for the 200

Posted by Derek at 1:41 AM

Labels: Jukebox, Seeburg, Wall-O-Matic, Wallbox

1 comment:

laya said...

Very informative post. You can also know ms steel suppliers in chennai offers a best standard and used for various purposes. The quality steels will be used in a wide range. jsw steel dealers in Chennai also earn the customers trust in the same way like ms steel.

6:42 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Awesome Inc. theme. Powered by Blogger.

…Building a Seeburg Wall-O-Matic Interface (Part 3)

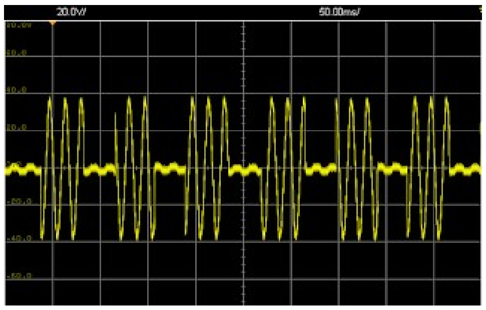
Decoding the Pulses

What the signal looks like

The way these wallboxes signal a song selection might seem a little weird from a modern perspective. When you press the buttons on the front, a collection of contacts are closed and a motor wipes a metal contactor across a studded disk:

Wall-O-Matic 100 Contact Wiper Mechanism

On the other side of this is a signal wire coming out of the wallbox. On that signal wire, you basically get a stream of pulses corresponding to the selection. Of course, those pulses aren't really a clean square wave. Rather, they're slightly noisy 25VAC. If you hook the signal wire up to an oscilloscope, it looks something like this:

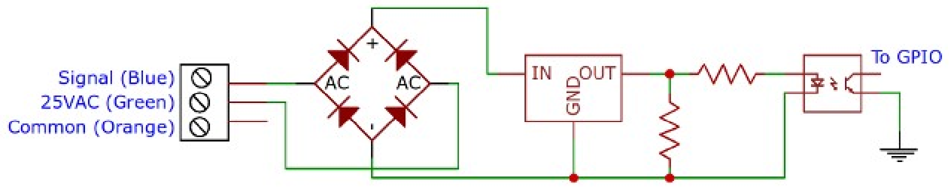


Raw pulsed AC waveform

What the other projects did

Each of the other projects I looked at did things a little bit differently, but they all had a common theme: Rectify the AC, make sure its levels were brought in line with something a microcontroller could handle, and figure out the rest in software. Some of these projects also used an opto-isolator, so that sensitive electronics couldn't be damaged by crap coming from the wallbox.

The basic schematic looked something like this:



Rectifier, Regulator, Resistors, and Opto-Isolator

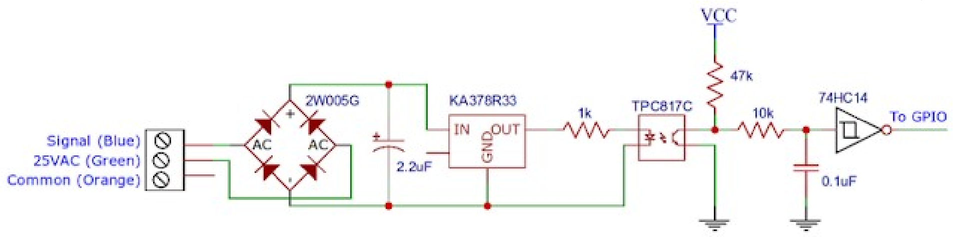
While this approach can be made to work, there's a fair amount of noise you have to account for in software. The pulses are not contiguous, and they are coming from a mechanism that is fundamentally prone to contact bounce.

Preprocessing the signal

Most of the elements of the basic design seemed good to me. I liked using a voltage regulator to bring down the levels, and I liked the idea of isolating the wallbox from the microcontroller. However, I didn't like the idea of having to reliably decode a pulse stream out of noisy rectified AC. With some additional circuitry, I figured that I could get to a significantly cleaner signal.

Circuit diagram

It took a fair amount of research and experimentation to come up with this, but here's the circuit I ended up with. On the input side, it takes pulsed AC from the wallbox's mechanism. On the output side, you get a clean digital pulse stream that is suitable for triggering interrupts and counting with minimal fuss.

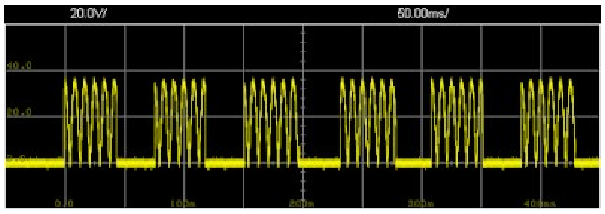


Signal Processing Circuit

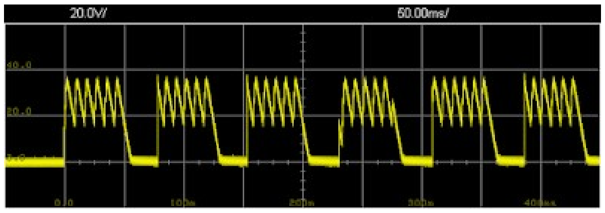
This circuit adds two main elements on top of the previous designs. On the input side, it adds an an appropriately sized capacitor. This capacitor's purpose is to smooth out the rectified wave just enough that it is invisible on the other side of the rectifier, but not so much that it obscures the pulse gaps. On the output side, it adds an RC debouncer designed to make sure the pulses are stable and have clean transitions. (I have to give credit to Jack Ganssle's page on the topic, for providing one of the most useful explanations and examples for figuring out this part.)

Tour of oscilloscope screenshots

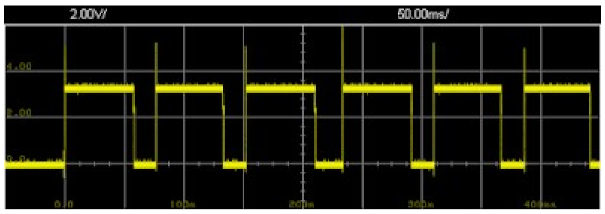
Probably the clearest way to explain what this circuit does, is to actually show what the pulses look like across its elements. So here goes, with a sequence of oscilloscope screenshots:



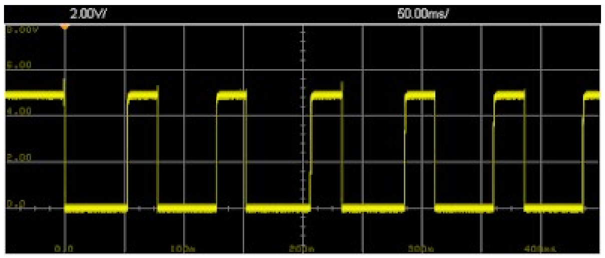
Output of full wave bridge rectifier



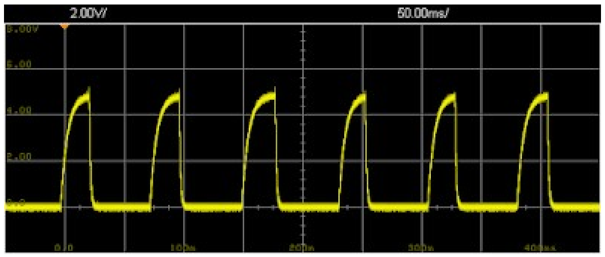
Rectifier output with 2.2uF capacitor



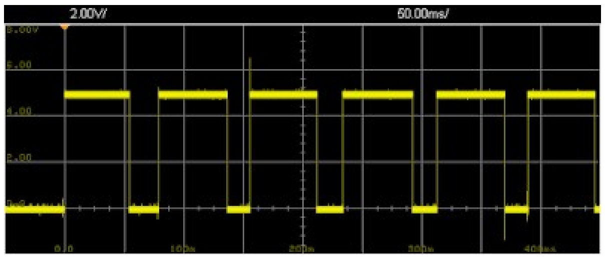
Output of KA78R33 voltage regulator



Output of TPC817C opto-isolator



Input of 74HC14 Schmitt trigger

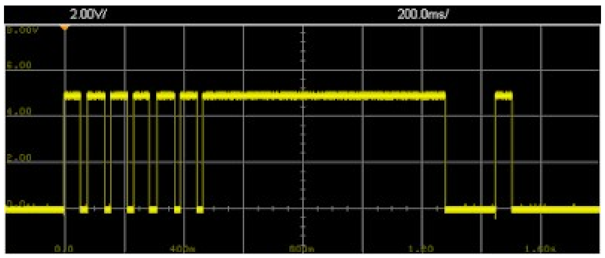


Output of 74HC14 Schmitt trigger

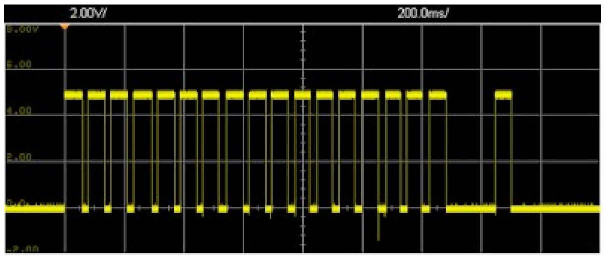
Figuring out the protocol

Reverse engineering

Once I had a clean digital-friendly output, it was time to document the actual protocol. I began by capturing traces like the ones shown below, for a wide range of song selections, while taking notes. Keep in mind that this is specific to the model 100 unit I was working with, and that its entirely possible that other units generate different looking pulse streams.



Pulse stream (Song A-6)



Pulse stream (Song B-6)

I made the following observations:

- Pulses appear to be in two groups
- Each pulse is ~50ms wide
- If the first group has 10 pulses or less, the groups are separated by a long (~814ms) pulse
- If there are more than than 10 pulses in the first group, then the groups are separated by a medium (~174ms) gap
- The full pulse sequence is ~2.1 seconds in duration
- The first group has 1-10, 12-21 pulses, and appears to be the least-significant figure
- The second group has 1-5 pulses, and appears to be the most significant figure

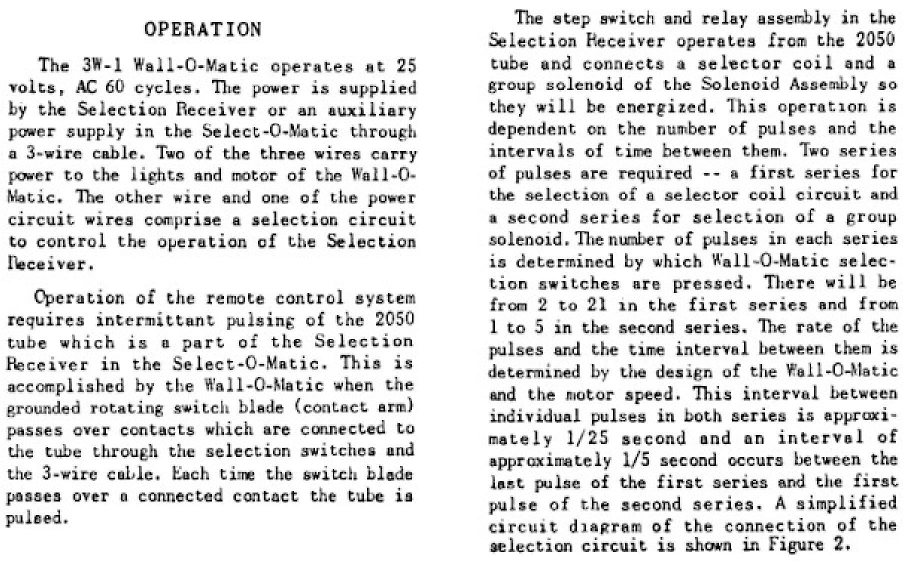
If I chart this out to see how it maps to the song selection buttons, I end up with a sequence like this:

```
A1 ( 1, 1), A2 ( 2, 1), ..., A10 (10, 1)
B1 (12, 1), B2 (13, 1), ..., B10 (21, 1)
C1 ( 1, 2), C2 ( 2, 2), ..., C10 (10, 2)
D1 (12, 2), D2 (13, 2), ..., D10 (21, 2)
(Note: The letter 'I' is skipped.)
```

From this information, a pulse decoding function can be written!

Reading the manual

I later discovered that the service manual actually did contain an excerpt explaining how these pulses work. In case anyone is curious, I've reflowed and pasted it below:



OPERATION

The 3W-1 Wall-O-Matic operates at 25 volts, AC 60 cycles. The power is supplied by the Selection Receiver or an auxiliary power supply in the Select-O-Matic through a 3-wire cable. Two of the three wires carry power to the lights and motor of the Wall-O-Matic. The other wire and one of the power circuit wires comprise a selection circuit to control the operation of the Selection Receiver.

Operation of the remote control system requires intermittant pulsing of the 2050 tube which is a part of the Selection Receiver in the Select-O-Matic. This is accomplished by the Wall-O-Matic when the grounded rotating switch blade (contact arm) passes over contacts which are connected to the tube through the selection switches and the 3-wire cable. Each time the switch blade passes over a connected contact the tube is pulsed.

The step switch and relay assembly in the Selection Receiver operates from the 2050 tube and connects a selector coil and a group solenoid of the Solenoid Assembly so they will be energized. This operation is dependent on the number of pulses and the intervals of time between them. Two series of pulses are required -- a first series for the selection of a selector coil circuit and a second series for selection of a group solenoid. The number of pulses in each series is determined by which Wall-O-Matic selection switches are pressed. There will be from 2 to 21 in the first series and from 1 to 5 in the second series. The rate of the pulses and the time interval between them is determined by the design of the Wall-O-Matic and the motor speed. This interval between individual pulses in both series is approximately 1/25 second and an interval of approximately 1/5 second occurs between the last pulse of the first series and the first pulse of the second series. A simplified circuit diagram of the connection of the selection circuit is shown in Figure 2.

Wired Wall-O-Matic, Type 3W-1

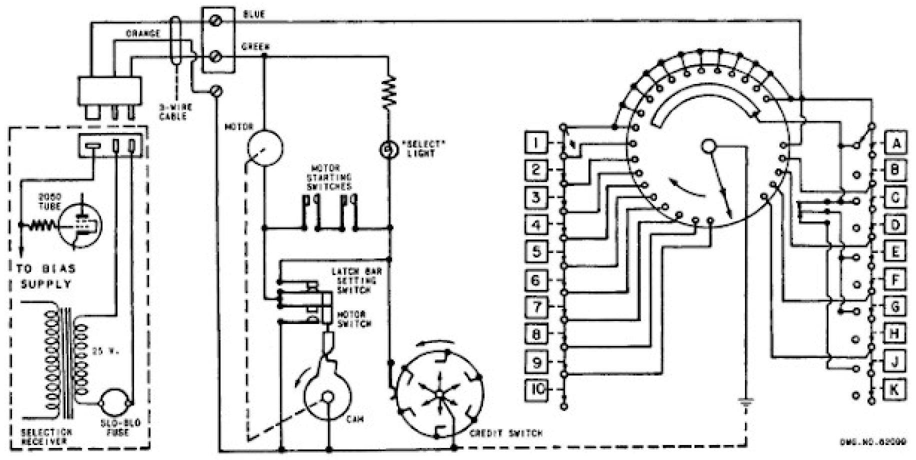


FIG. 2. SIMPLIFIED SCHEMATIC - 3W-1.

From this I gather that they were only really paying attention to the rising edge of the pulses. I'm glad I analyzed the full details, however, as it makes it easier for a robust decoder that can reject invalid/stalled /flaky selection sequences. It also makes it possible to implement short-but-effective timeouts depending on where in the pulse sequence we are.

Test decoding

Many years ago at an unrelated tech conference, I managed to acquire an Arduino Uno. It basically sat in its box until a few months ago, when I realized it could be useful as a "bench tinkering" microcontroller.

Arduino Uno

Despite never having used an Arduino before, this little device turned out to be the perfect way of testing my pulse decoding logic. I plugged it into the output of the signal processing circuit (built on a breadboard) from above, and whipped up a quick-and-dirty sketch that can successfully decode the pulses.

```
/*
   Seeburg 3WA Wall-O-Matic 100
   Test sketch
*/

const unsigned long USEC_PER_SEC = 1000000;
const int pin = 7;

void setup() {
  pinMode(pin, INPUT);
  Serial.begin(9600);
  Serial.println("Wall-O-Matic Pulse Tester");
  Serial.println("-------------------------");
}

void loop() {
  unsigned long lastTimeMs = millis();
  unsigned long durationUs;
  durationUs = pulseIn(pin, HIGH, 5 * USEC_PER_SEC);
  unsigned long pulseTimeMs = millis();
  if (durationUs == 0) {
    return;
  }

  int p1 = 0;
  int p2 = 0;
  bool delimiter = false;

  Serial.println("Start of pulses...");

  do {
    unsigned long elapsed = (pulseTimeMs - lastTimeMs) - (durationUs / 1000);

    lastTimeMs = pulseTimeMs;
    Serial.print("Pulse: ");
    if (durationUs < 1000) {
      Serial.print(durationUs, DEC);
      Serial.print("us");
    } else {
      Serial.print(durationUs / 1000, DEC);
      Serial.print("ms");
    }
    Serial.print(", elapsed: ");
    Serial.print(elapsed, DEC);
    Serial.println("ms");

    if (p1 > 0 && !delimiter && (durationUs / 1000) > 500) {
      delimiter = true;
      Serial.println("----DELIMITER (PULSE)----");
    }
    else {
      if (p1 > 0 && !delimiter && elapsed > 100) {
        delimiter = true;
        Serial.println("----DELIMITER (GAP)----");
      }
      if (!delimiter) {
        p1++;
      }
      else {
        p2++;
      }
    }
    durationUs = pulseIn(pin, HIGH, (delimiter ? 1 : 3) * USEC_PER_SEC);
    pulseTimeMs = millis();
  } while (durationUs > 0);

  Serial.println("Done.");
  Serial.print("-> Signal: ");
  Serial.print(p1, DEC);
  Serial.print(", ");
  Serial.print(p2, DEC);
  Serial.println();

  if (p2 < 1 || p2 > 5) {
    Serial.println("Pulse 2 invalid value");
    return;
  }

  char letter;
  int number;
  if (p1 >= 1 && p1 <= 10) {
    number = p1;
    letter = 'A' + (p2 - 1) * 2;
  }
  else if (p1 >= 12 && p1 <= 21) {
    number = p1 - 11;
    letter = 'A' + ((p2 - 1) * 2) + 1;
  }
  else {
    Serial.println("Pulse 1 invalid value");
    return;
  }

  // Skipping 'I' for some reason
  if (letter > 'H') { letter++; }

  Serial.print("-> Song: ");
  Serial.print(letter);
  Serial.print(number, DEC);
  Serial.println();
  Serial.println();
}
```

Concluding thoughts

Even though this post flows from start to finish, there was actually a lot of back-and-forth as I figured everything out. Part-way through the process, I upgraded from an ancient low-end analog oscilloscope to a modern digital storage oscilloscope. This tooling upgrade made a huge difference in my ability to experiment and refine this design. It enabled me to actually see all the signal transitions and glitches, and to determine all the necessary components to get to a clean pulse train. Early on, the Arduino code was actually capturing (and attempting to overcome) a lot of signal noise. The final version, however, can pretty much ignore it as a factor.

While this was a lengthy post, there's definitely more to come.

Posted by Derek at 1:57 AM

Labels: Arduino, Jukebox, Oscilloscope, Seeburg, Wall-O-Matic, Wallbox

No comments:

Post a Comment

H.E.C. Geek

Household Enterprise Computing Geek

Sunday, October 22, 2017

Building a Seeburg Wall-O-Matic Interface (Part 1)

Introduction

Last year we moved into a house that had some interesting touches left by a previous owner. Most notably, there was a "play room" decorated in the style of a 50's Diner. It had a booth, table, chairs, and plenty of decoration. We joked that this room's decor was what sold us on the house!

In any case, one notable thing missing from this room was a jukebox! Or at least, taking things slightly more sensibly, the jukebox controller that such diners often have sitting at their tables. It was at this point that I started learning all about the iconic Seeburg Wall-O-Matic.

Flash forward a year, and I was finally in the process of resurrecting my electronics workbench. I started watching a few too many EEVblog videos, built a shelf/bench setup using components from Ikea, unpacked all the gear I'd kept in boxes for a few too many years, and even made a few upgrades.

About Me

Derek

Self-professed ubergeek with a wide variety of technical interests. Can be found evangelizing FreeBSD, tinkering with Solaris, building embedded electronics, developing software, or working on obscure networking projects.

View my complete profile

Blog Archive

Followers

Followers (5)

Follow

The one thing I desperately needed? Projects! Given that I knew I was going to have a lot of free time coming up in the near future, I reopened my research into the Wall-O-Matic and began to scour eBay.

Background Research

One of the first things I stumbled across were these commercial "products" designed to provide a modern interface from the wallbox:

CD Adapter
Wallbox2mp3

Unfortunately, these projects were less than desirable for my tastes. I was also looking for a project, not an off-the-shelf solution. These devices also seemed a bit dated, of limited availability, and quite proprietary. They also seemed to focus on playback a bit too "locally," rather than using the wallbox as an actual remote for a real stereo system. My house had in-wall speakers installed in many rooms, including the "diner" room, and I really wanted to use those. Since I had already connected many of my in-wall speakers to a Sonos rig, I kept wondering if there was a way I could just use that.

The next thing I did was dig into these hobbyist projects which seemed much closer to what I actually wanted to accomplish:

Wall Box SONOS Controller [Stephen Devlin]
Seeburg Wall-O-Matic [Retro Future Electrics]
Raspberry Pi Project – A 1960s wallbox interfaced with Sonos [Phil Lavin]

One common theme among these projects was simplicity. Minimal components to interface the wallbox to a Raspberry Pi, and minimal work to control a Sonos system based on the result. They also provided enough schematic and component details to give me a tangible starting point. Even if I decided to take a different path with my own project, at least I had a good foundation to build upon.

Project Goals

So thinking through what I wanted to accomplish with this project, I decided I wanted to build a device that could do the following:

- Provide power to the wallbox
- Read the signal pulses, and decode them into a song selection
- Enqueue selected songs with my Sonos system, simulating the functionality of a jukebox
- Electronically toggle the coin switches, so that inserting actual coins would be optional

At a lower level, I also knew I wanted to take things seriously in the design of the circuit I was going to use to accomplish all of this. That meant:

- Complete and detailed schematic
- Complete and detailed BOM (bill-of-materials)
- Real fabricated PCB (printed circuit board) design

(The BOM and PCB being things that I'd never actually done before. Every prior circuit of mine was a hand-constructed mess of wires on a pre-drilled pad-per-hole PCB. Thankfully, in this day and age, doing it "right" is now quite accessible.)

I'll attempt to break this blog series apart based on the major progression of this project. I may not discuss things in the actual order that I did them, since there was a lot of back-and-forth between the various elements. However, it should flow in an order that makes sense. Most likely it'll be something like this:

- Procuring a Functional Wallbox
- Decoding the Pulses
- Inserting Coins
- Designing the Circuit
- Developing the Software

Posted by Derek at 5:09 PM

Labels: Jukebox, PCB, Seeburg, Sonos, Wall-O-Matic, Wallbox

2 comments:

Steve Hammer said...

Thanks so much for posting about your project. Your pulse decoding circuit saved my bacon on a similar project using a wall-o-matic 100 that I started in December. I tried my own hand at making a circuit for the decoder, but after much testing decided that now matter how much tweaking I did, it wasn't going to work reliably. Sadly I didn't find your post until I had tried it my way. My 3W1 will eventually be hooked up to a Raspberry Pi running Fruitbox, but the premise is similar to yours. A 65 year old front end to a digital device! It's an awesome idea. Thank you, thank you, thank you!

Cheers,
Steve Hammer

8:43 PM

Anthony said...

Following this post ! Very interesting to see how you can do this. I use presently some cdadapter but not really satisfied. If you need funding to help your research, i'm in ! I have a seeburg consolette and it should be awesome to digitalize it, without any damage. Thanks !

5:37 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Awesome Inc. theme. Powered by Blogger.

H.E.C. Geek

Household Enterprise Computing Geek

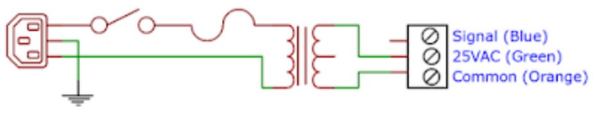
Tuesday, October 24, 2017

Building a Seeburg Wall-O-Matic Interface (Part 2)

Procuring a Functional Wallbox

Providing Power

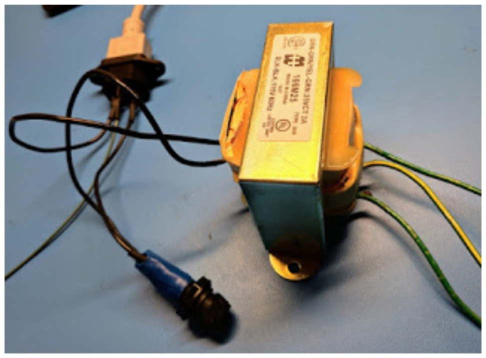
Before you can do very much with a project like this, you need a way of powering up the wallboxes. These things don't run off the mains, nor do they use DC. Rather, these devices run off 25VAC. While you likely don't have a power supply just laying around that can provide this, its pretty simple to assemble one. You really just need an appropriate transformer and a few (optional) support components. This then connects up to a terminal strip right inside the wallbox itself.



In my setup, I used the following components:

- IEC320-C14 Power Receptacle (Q212-ND)
- Rocker Switch (CH755-ND)
- Fuse Holder (283-2712-ND)
- 3A Fuse (283-2841-ND or 283-2917-ND)
- 115V to 25V Transformer (HM556-ND)

I just wired these together as shown in the above schematic, using some heat shrink tubing and electrical tape to cover up the exposed contacts. Specifics aren't all that important here. All that really matters is that you get approximately 25VAC on the output, and won't fry something if there is a short.



(Note: I don't actually have the rocker switch in this version, as I just used a switchable power strip. I do plan to add one on the final version.)

Okay, now to get started...

Seeburg Wall-O-Matic 200

This journey began with the acquisition of a Seeburg Wall-O-Matic (V-3WA) 200 from a rather nice eBay listing. It appeared clean, in good condition, chrome intact, and with the key. On the surface, it seemed like everything I needed to get started:

About Me

Derek

Self-professed ubergeek with a wide variety of technical interests. Can be found evangelizing FreeBSD, tinkering with Solaris, building embedded electronics, developing software, or working on obscure networking projects.

View my complete profile

Blog Archive

Followers

Followers (5)

Follow

Unfortunately this wallbox basically sat untouched for a few weeks, since I still had to buy the necessary components (shown above) to power it up and I was preoccupied with other things at the time.

When I finally powered it up, the first thing I discovered was that multiple light bulbs needed replacing. That much was no big deal, so I just ordered them off Amazon (#51 and #55 bayonet mount light bulbs). I then discovered that the electrical contacts were dirty, the mechanics needed a little fiddling, and it didn't seem to be working correctly.

Thankfully it wasn't too hard to find the repair manual online. Of course it was written in 50's speak, and it was sometimes hard to match the terms and illustrations to what I was seeing inside the actual device.

I spent the next week or so in a state of constant frustration. I replaced the bulbs, cleaned all the contacts, tried to adjust and/or understand what parts of the mechanism I could, kept cursing at the DCU ("dual credit unit") that I was afraid to disassemble, and eventually sorta got it half-working. I got it to the point where I could manually toggle the coin switches and punch in a selection. Of course it would get stuck part-way through the signaling cycle half the time, and I'm not sure if it worked consistently with actual coins. (It was also dirty enough that I felt the need to wash my hands every time I was done fussing with it.)

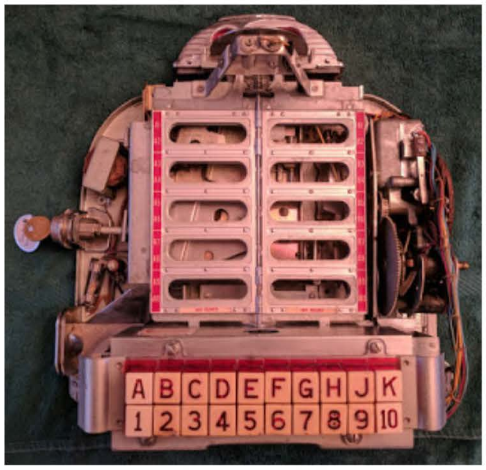
From all of this, at least I learned quite a bit about how these devices operate. These things were designed in an era that pre-dates "electronics" as we know them, and are electro-mechanical in nature. They use a complex assortment of gears, cams, metal strip contactor switches, motors, and solenoids to accomplish what you'd do today in a single $0.50 microcontroller. (Even if it was only the 1970's, chances are you'd do this with a small assortment of transistors and logic chips.)

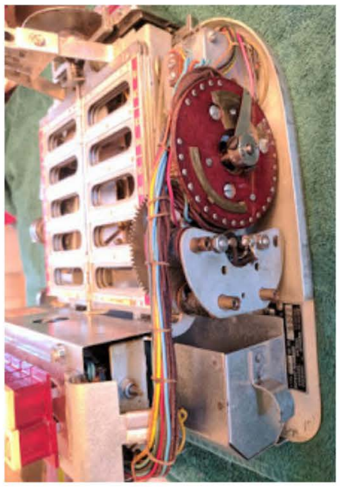
Eventually, I decided it was in my best interest to give up for now. I didn't feel comfortable disassembling the parts that needed the most attention, and I really didn't want to focus all of my energy on this stage of the project. So I decided to just go ahead and actually order a known clean/functional unit, from a dealer that actually specializes in this sort of thing. I can always return to this unit later, and it'll make a nice display piece regardless.

Seeburg Wall-O-Matic 100

This journey continued with me ordering a Seeburg Wall-O-Matic (3W-1) 100 from an actual retro equipment dealer. This time at least I knew I was getting something that had been cleaned and lubricated on the inside, in addition to being in good condition on the outside.

Okay, the buttons could probably use some restoration or replacement, but the rest of it looked excellent. Especially on the inside...





When I powered this unit up, everything magically worked. Okay, I might have had to fiddle with the coin switches a little bit, but those are easy to knock out of place simply by removing the title strip and coin rejector assemblies. Regardless, I was quite happy. I now had a fully functional and reliable wallbox I could use as a foundation for the next stage of the project.

Besides simply being clean, lubricated, and functional, this model had another big advantage over the 200. Its mechanism is a lot simpler. It doesn't have an overly complex "credit unit" in the middle, and I don't think I'd be afraid to try disassembling any of its mechanism if I needed to.

I did later discover this unit had a few modifications done to its coin mechanism, however. It doesn't accept dimes (only nickels and quarters), two of the coin switches were tied together, and one of the coin solenoids was disconnected. I wish these modifications hadn't been done, but they're not a showstopper. I can easily live with them. They basically mean that the device now has only two credit states: A dime adds one song credit, and a quarter adds two song credits.

References

Repair manual for the 100
Repair manual for the 200

Posted by Derek at 1:41 AM

Labels: Jukebox, Seeburg, Wall-O-Matic, Wallbox

1 comment:

laya said...

Very informative post. You can also know ms steel suppliers in chennai offers a best standard and used for various purposes. The quality steels will be used in a wide range. jsw steel dealers in Chennai also earn the customers trust in the same way like ms steel.

6:42 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Awesome Inc. theme. Powered by Blogger.

H.E.C. Geek

Household Enterprise Computing Geek

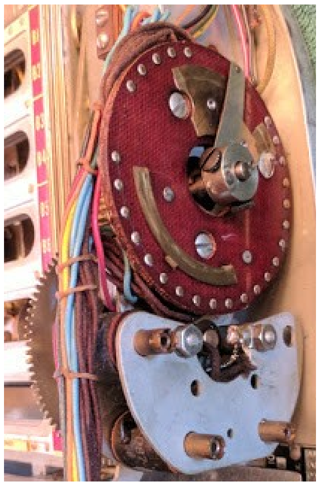
Monday, October 30, 2017

Building a Seeburg Wall-O-Matic Interface (Part 3)

Decoding the Pulses

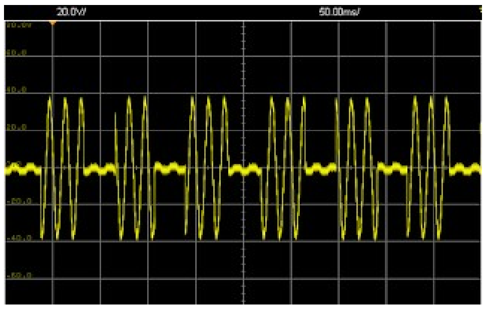
What the signal looks like

The way these wallboxes signal a song selection might seem a little weird from a modern perspective. When you press the buttons on the front, a collection of contacts are closed and a motor wipes a metal contactor across a studded disk:



Wall-O-Matic 100 Contact Wiper Mechanism

On the other side of this is a signal wire coming out of the wallbox. On that signal wire, you basically get a stream of pulses corresponding to the selection. Of course, those pulses aren't really a clean square wave. Rather, they're slightly noisy 25VAC. If you hook the signal wire up to an oscilloscope, it looks something like this:



Raw pulsed AC waveform

What the other projects did

Each of the other projects I looked at did things a little bit differently, but they all had a common theme: Rectify the AC, make sure its levels were brought in line with something a microcontroller could handle, and figure out the rest in software. Some of these projects also used an opto-isolator, so that sensitive electronics couldn't be damaged by crap coming from the wallbox.

The basic schematic looked something like this:

About Me

Derek

Self-professed ubergeek with a wide variety of technical interests. Can be found evangelizing FreeBSD, tinkering with Solaris, building embedded electronics, developing software, or working on obscure networking projects.

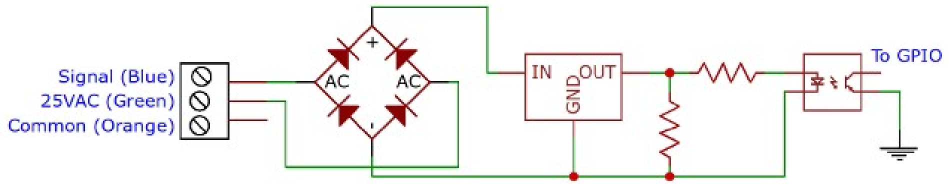
View my complete profile

Blog Archive

Followers

Followers (5)

Follow



Rectifier, Regulator, Resistors, and Opto-Isolator

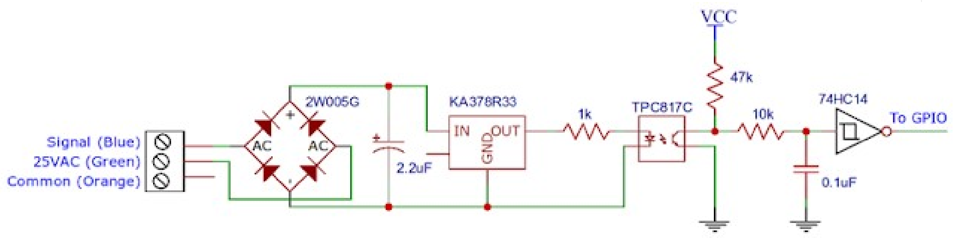
While this approach can be made to work, there's a fair amount of noise you have to account for in software. The pulses are not contiguous, and they are coming from a mechanism that is fundamentally prone to contact bounce.

Preprocessing the signal

Most of the elements of the basic design seemed good to me. I liked using a voltage regulator to bring down the levels, and I liked the idea of isolating the wallbox from the microcontroller. However, I didn't like the idea of having to reliably decode a pulse stream out of noisy rectified AC. With some additional circuitry, I figured that I could get to a significantly cleaner signal.

Circuit diagram

It took a fair amount of research and experimentation to come up with this, but here's the circuit I ended up with. On the input side, it takes pulsed AC from the wallbox's mechanism. On the output side, you get a clean digital pulse stream that is suitable for triggering interrupts and counting with minimal fuss.

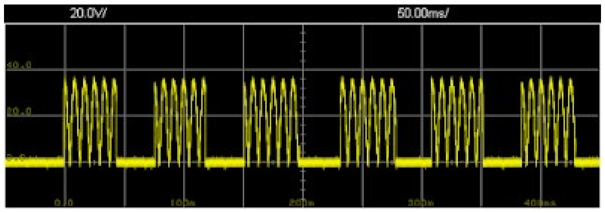


Signal Processing Circuit

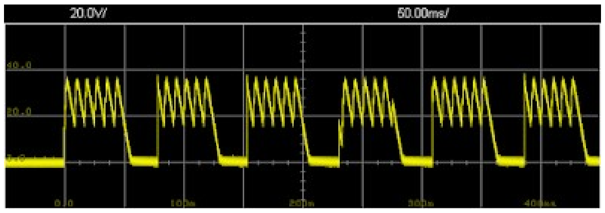
This circuit adds two main elements on top of the previous designs. On the input side, it adds an an appropriately sized capacitor. This capacitor's purpose is to smooth out the rectified wave just enough that it is invisible on the other side of the rectifier, but not so much that it obscures the pulse gaps. On the output side, it adds an RC debouncer designed to make sure the pulses are stable and have clean transitions. (I have to give credit to Jack Ganssle's page on the topic, for providing one of the most useful explanations and examples for figuring out this part.)

Tour of oscilloscope screenshots

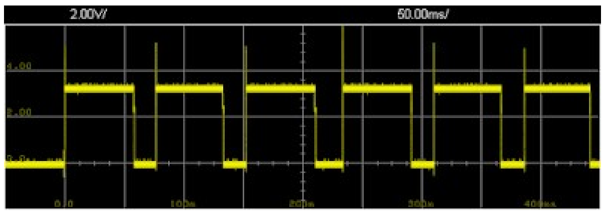
Probably the clearest way to explain what this circuit does, is to actually show what the pulses look like across its elements. So here goes, with a sequence of oscilloscope screenshots:



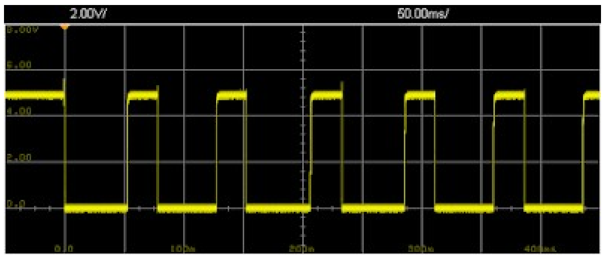
Output of full wave bridge rectifier



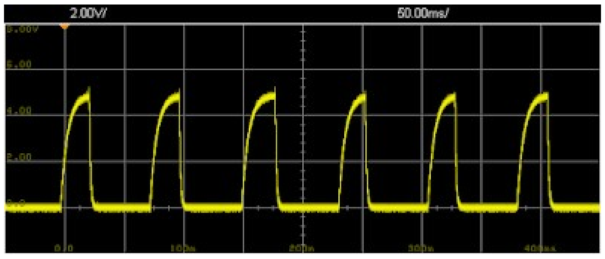
Rectifier output with 2.2uF capacitor



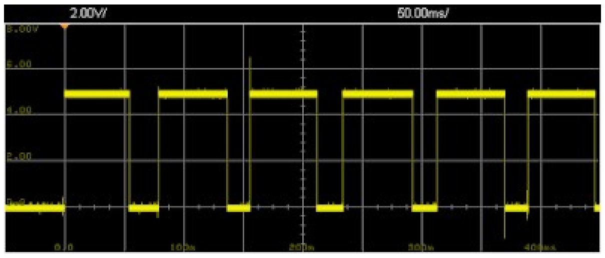
Output of KA78R33 voltage regulator



Output of TPC817C opto-isolator



Input of 74HC14 Schmitt trigger

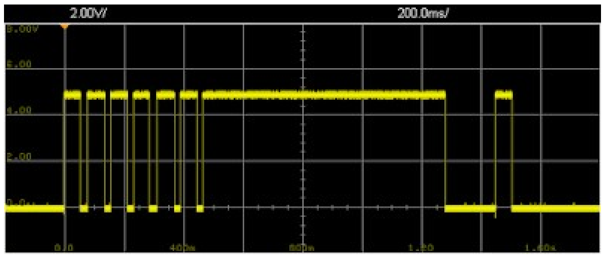


Output of 74HC14 Schmitt trigger

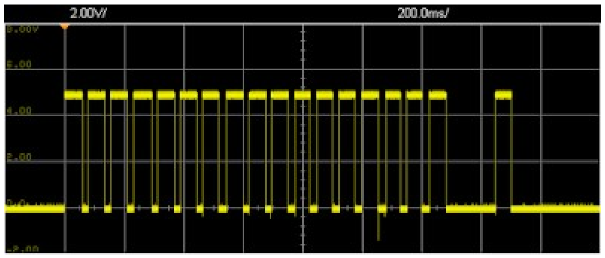
Figuring out the protocol

Reverse engineering

Once I had a clean digital-friendly output, it was time to document the actual protocol. I began by capturing traces like the ones shown below, for a wide range of song selections, while taking notes. Keep in mind that this is specific to the model 100 unit I was working with, and that its entirely possible that other units generate different looking pulse streams.



Pulse stream (Song A-6)



Pulse stream (Song B-6)

I made the following observations:

- Pulses appear to be in two groups
- Each pulse is ~50ms wide
- If the first group has 10 pulses or less, the groups are separated by a long (~814ms) pulse
- If there are more than than 10 pulses in the first group, then the groups are separated by a medium (~174ms) gap
- The full pulse sequence is ~2.1 seconds in duration
- The first group has 1-10, 12-21 pulses, and appears to be the least-significant figure
- The second group has 1-5 pulses, and appears to be the most significant figure

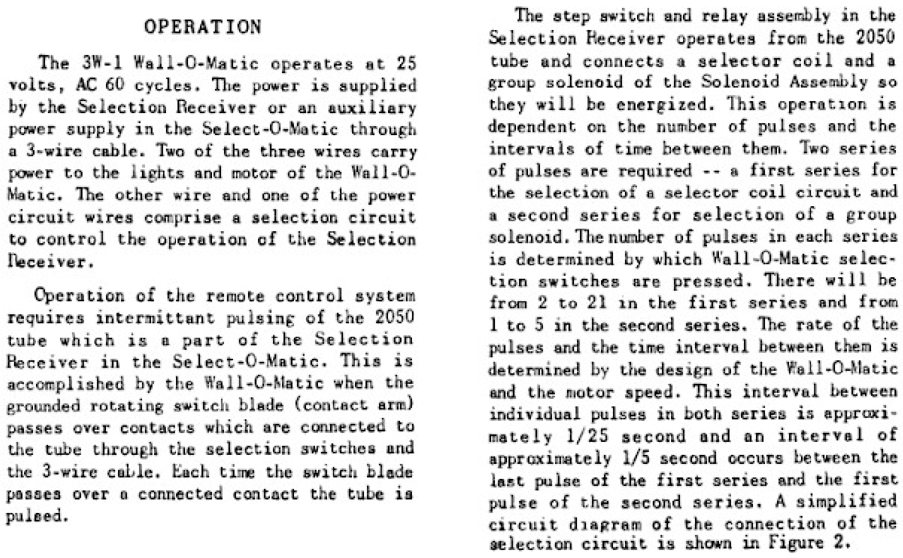
If I chart this out to see how it maps to the song selection buttons, I end up with a sequence like this:

```
A1 ( 1, 1), A2 ( 2, 1), ..., A10 (10, 1)
B1 (12, 1), B2 (13, 1), ..., B10 (21, 1)
C1 ( 1, 2), C2 ( 2, 2), ..., C10 (10, 2)
D1 (12, 2), D2 (13, 2), ..., D10 (21, 2)
(Note: The letter 'I' is skipped.)
```

From this information, a pulse decoding function can be written!

Reading the manual

I later discovered that the service manual actually did contain an excerpt explaining how these pulses work. In case anyone is curious, I've reflowed and pasted it below:



OPERATION

The 3W-1 Wall-O-Matic operates at 25 volts, AC 60 cycles. The power is supplied by the Selection Receiver or an auxiliary power supply in the Select-O-Matic through a 3-wire cable. Two of the three wires carry power to the lights and motor of the Wall-O-Matic. The other wire and one of the power circuit wires comprise a selection circuit to control the operation of the Selection Receiver.

Operation of the remote control system requires intermittant pulsing of the 2050 tube which is a part of the Selection Receiver in the Select-O-Matic. This is accomplished by the Wall-O-Matic when the grounded rotating switch blade (contact arm) passes over contacts which are connected to the tube through the selection switches and the 3-wire cable. Each time the switch blade passes over a connected contact the tube is pulsed.

The step switch and relay assembly in the Selection Receiver operates from the 2050 tube and connects a selector coil and a group solenoid of the Solenoid Assembly so they will be energized. This operation is dependent on the number of pulses and the intervals of time between them. Two series of pulses are required -- a first series for the selection of a selector coil circuit and a second series for selection of a group solenoid. The number of pulses in each series is determined by which Wall-O-Matic selection switches are pressed. There will be from 2 to 21 in the first series and from 1 to 5 in the second series. The rate of the pulses and the time interval between them is determined by the design of the Wall-O-Matic and the motor speed. This interval between individual pulses in both series is approximately 1/25 second and an interval of approximately 1/5 second occurs between the last pulse of the first series and the first pulse of the second series. A simplified circuit diagram of the connection of the selection circuit is shown in Figure 2.

Wired Wall-O-Matic, Type 3W-1

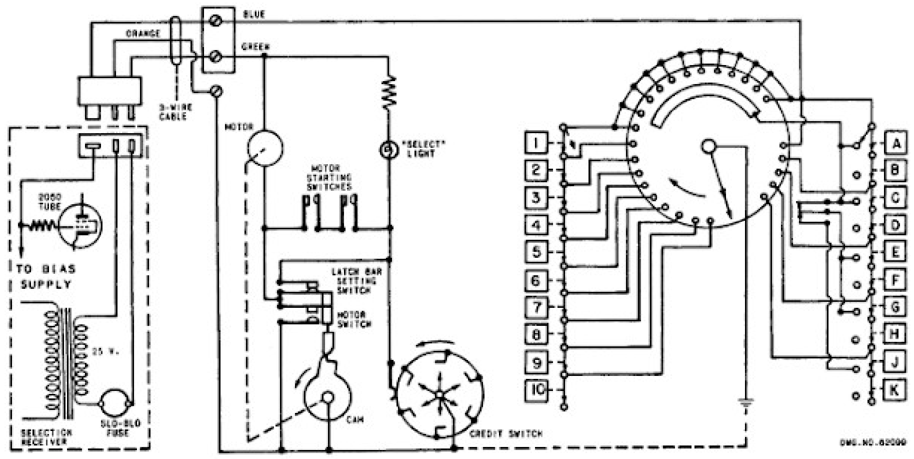


FIG. 2. SIMPLIFIED SCHEMATIC - 3W-1.

From this I gather that they were only really paying attention to the rising edge of the pulses. I'm glad I analyzed the full details, however, as it makes it easier for a robust decoder that can reject invalid/stalled /flaky selection sequences. It also makes it possible to implement short-but-effective timeouts depending on where in the pulse sequence we are.

Test decoding

Many years ago at an unrelated tech conference, I managed to acquire an Arduino Uno. It basically sat in its box until a few months ago, when I realized it could be useful as a "bench tinkering" microcontroller.

Arduino Uno

Despite never having used an Arduino before, this little device turned out to be the perfect way of testing my pulse decoding logic. I plugged it into the output of the signal processing circuit (built on a breadboard) from above, and whipped up a quick-and-dirty sketch that can successfully decode the pulses.

```
/*
   Seeburg 3WA Wall-O-Matic 100
   Test sketch
*/

const unsigned long USEC_PER_SEC = 1000000;
const int pin = 7;

void setup() {
  pinMode(pin, INPUT);
  Serial.begin(9600);
  Serial.println("Wall-O-Matic Pulse Tester");
  Serial.println("-------------------------");
}

void loop() {
  unsigned long lastTimeMs = millis();
  unsigned long durationUs;
  durationUs = pulseIn(pin, HIGH, 5 * USEC_PER_SEC);
  unsigned long pulseTimeMs = millis();
  if (durationUs == 0) {
    return;
  }

  int p1 = 0;
  int p2 = 0;
  bool delimiter = false;

  Serial.println("Start of pulses...");

```

```
    do {
      unsigned long elapsed = (pulseTimeMs - lastTimeMs) - (durationUs / 1000);

      lastTimeMs = pulseTimeMs;
      Serial.print("Pulse: ");
      if (durationUs < 1000) {
        Serial.print(durationUs, DEC);
        Serial.print("us");
      } else {
        Serial.print(durationUs / 1000, DEC);
        Serial.print("ms");
      }
      Serial.print(", elapsed: ");
      Serial.print(elapsed, DEC);
      Serial.println("ms");

      if (p1 > 0 && !delimiter && (durationUs / 1000) > 500) {
        delimiter = true;
        Serial.println("----DELIMITER (PULSE)----");
      }
      else {
        if (p1 > 0 && !delimiter && elapsed > 100) {
          delimiter = true;
          Serial.println("----DELIMITER (GAP)----");
        }
        if (!delimiter) {
          p1++;
        }
        else {
          p2++;
        }
      }
      durationUs = pulseIn(pin, HIGH, (delimiter ? 1 : 3) * USEC_PER_SEC);
      pulseTimeMs = millis();
    } while (durationUs > 0);

    Serial.println("Done.");
    Serial.print("-> Signal: ");
    Serial.print(p1, DEC);
    Serial.print(", ");
    Serial.print(p2, DEC);
    Serial.println();

    if (p2 < 1 || p2 > 5) {
      Serial.println("Pulse 2 invalid value");
      return;
    }

    char letter;
    int number;
    if (p1 >= 1 && p1 <= 10) {
      number = p1;
      letter = 'A' + (p2 - 1) * 2;
    }
    else if (p1 >= 12 && p1 <= 21) {
      number = p1 - 11;
      letter = 'A' + ((p2 - 1) * 2) + 1;
    }
    else {
      Serial.println("Pulse 1 invalid value");
      return;
    }

    // Skipping 'I' for some reason
    if (letter > 'H') { letter++; }

    Serial.print("-> Song: ");
    Serial.print(letter);
    Serial.print(number, DEC);
    Serial.println();
    Serial.println();
  }
```

Concluding thoughts

Even though this post flows from start to finish, there was actually a lot of back-and-forth as I figured everything out. Part-way through the process, I upgraded from an ancient low-end analog oscilloscope to a modern digital storage oscilloscope. This tooling upgrade made a huge difference in my ability to experiment and refine this design. It enabled me to actually see all the signal transitions and glitches, and to determine all the necessary components to get to a clean pulse train. Early on, the Arduino code was actually capturing (and attempting to overcome) a lot of signal noise. The final version, however, can pretty much ignore it as a factor.

While this was a lengthy post, there's definitely more to come.

Posted by Derek at 1:57 AM

Labels: Arduino, Jukebox, Oscilloscope, Seeburg, Wall-O-Matic, Wallbox

No comments:

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)