Hardware and Software for Mechanical Television System

This document provides all the required information for being able to build / tweak a mechanical narrow band television similar to the prototype show in this Youtube video: http://www.youtube.com/watch?v=tslWfnsTwPg .

Building-up and tweaking this kind of mechanical and electronic hardware requires some experience and a lot of patience. Electromechanical parts (speed response vs torque of the available motor , disc inertia and size) have an impact on the global system behavior, so that the prototype will not be “plug-and-play”. A lot of adjustment means have been implemented in the schematics to face this kind of issues. However, it remains possible that you will have to modify some values/components to match with your mechanical configuration, especially in the motor synchronization system (as in all other kinds of mechanical TV designs). If experienced in homebrew design but not familiar at all with NBTV and/or mechanical television, first have a look to this well documented website: http://www.earlytelevision.org/dupouy.html

The design proposed in this document is modular so that it is possible to chose some parts from it, if not all of them, according to one's preferences:

* Video synchronization extractor + Y / green extractor
* Motor driving system disc synchronization with video
* Vsync triggered stroboscope for an easy adjustment of vertical synchronization by the user.
* Low loss LED anode voltage regulator (avoids thermal dissipation issues and reduces power consumption), made with very common components.
* SECAM-like color decoder + audio separator, if willing to build a color TV (includes a PIC1F819).

No particular design is proposed for the audio amplifier since any kind of audio amplifier (having an input impedance higher than 10K in case of a color system) can match for this purpose. For the prototype shown in the video I simply used as audio amplifier a small IC demo kit that I had saved from the trash at my workplace…

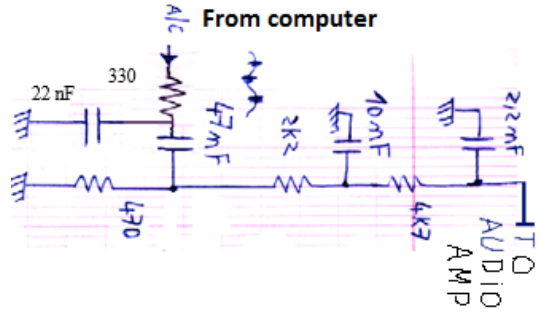
Input signal and main features

Like many other NBTV systems, this proposed piece of hardware handles a narrow band video signal stored in a specifically generated computer file. The file type is WAV, commonly used for audio recordings: A classical lossless 2-channel (stereo) recording file at 48000 spl/s. Therefore it makes sense to talk about pixels like in digital video, because the video signal sampling rate directly gives the number of produced pixels per second when scanning a line (even if you plan to reproduce this signal later with some custom analog equipment).

In case of a black-and-white system:

- The first channel is used for composite monochrome video (Vsync/Hsync/Lum)
- The second channel is used exclusively for audio

In this case where you don't use the color decoder, you will have at least to insert this filter between the computer audio channel output and the audio amplifier input to suppress the color carrier superimposed by default by the software converter. (If the amplifier has no volume control, insert a 10K potentiometer - as a voltage divider - between the filter output and the amplifier input)



In case of a color system:

- The first channel is used for composite green video (Vsync/Hsync/Green)
- The second channel is used for low-pass filtered audio + SECAM-like color carrier
  (alternatively providing the blue and the red color signals. No YUV optimization is handled at this time).

Mechanical rasterizer, resolution and size

The mechanical scanning system is based on a Nipkow's disc. However the electronics could be used to drive other mechanisms (drums, mirror screws...). But we will assume for paper simplification that a Nipkow's disc is used.

Experimenters willing to build their own Nipkow's disc can use this helping SW tool:
http://users.tpg.com.au/users/gmillard/nbtv/DXFNipkow.zip

The electronics and the video generation software tools are designed to be compatible with 30-line and 60-line discs, while reserving the first line for a HSYNC signal. In other words, the first line sector is present on the disc (and the corresponding hole can exist) but is it always black, bringing the effective resolutions at 29 and 59 lines. The rasterization is horizontal like in classical video (unless the original video has already been rotated before creating the NBTV video), and its speed is 12,5 frames per second.

The prototype shown in the Youtube video is based on a Nipkow's disc spinning at 12,5 turns per second. It is relatively small, especially if dealing with 60 lines (*).

(*) This experience helped to conclude that such a small disc featuring so many lines can only be built with advanced machining equipment. Even a printed template and a careful driller will not succeed into reaching a satisfying result. This is why generated pictures shown in the video are disappointing. Handmade discs should be 30-line discs with a diameter close to what can be seen in the video (27 cm) or 60-line discs, two times bigger if targeting an aspect ratio close to 1:1 and a reasonable picture size.

Aspect ratio considerations

The goal of building such a small disc was to optimize the ratio [Picture width] / [Disc diameter] by creating shrunk pictures on the disc which could be horizontally extended with the help of an anamorphic lens. However, this prototype particularity is just an example and all the stuff presented here can be used with more classical Nipkow's discs, provided that some care is taken for the aspect ratio:

The original video file to be processed by supplied software (detailed further) must have the same aspect ratio than the final picture to be created on the disc (directly depending on disc geometry). The video converting program will adapt the entire original images to fit in a NBTV video at 57x59 pixels (or 121x29 pixels for 30-line variant), whatever are the initial and the final picture resolutions. In other words, pixels may not be "squares" depending on what you do: With 60 lines the effective pixel shape proportions are almost the same than the disc-created picture proportions, while with 30 lines the ratio height / width is the quadruple. (Pixels are in fact more similar to small bars perpendicular to lines, like in the most current cases of 30-line television systems. Ideally the disc holes should have the same proportions and shapes but this is not possible).

Controls, output devices and sensors

a) External controls are

- One user-controlled pushbutton for vertical synchronization adjustment
  (The original design of the Youtube video had a far too much complex system)

- One user-controller potentiometer for fine adjusting the free run disc speed

- One user-controller potentiometer for adjusting the sound volume (optional)

- One power switch, of course ;-)

b) Output devices or actuators are:

- A white power LED in case of a black and white system, for disc lighting (video driven back light):
  For example, a 900mA LED - Avago ASMT MxE2 or Mx22.

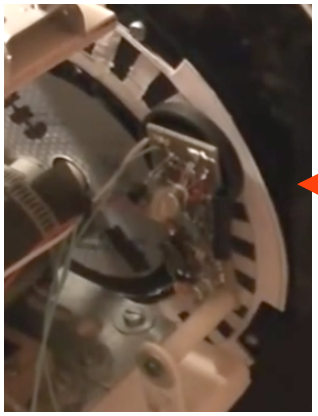
- An RGB triple LED in case of a color system, for disc lighting (video driven back light):
  For example, a 3x300mA RGB LED – Broadcom/Avago ASMT-MT100-00001

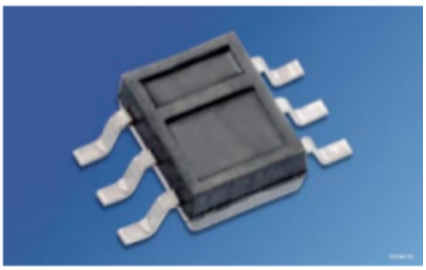
- The disc driving motor:
  A DC motor reaching the unloaded speed of 12,5 rps (750 rpm) at a voltage between 6 and 15V.

- A simple LED used as stroboscope to help synchronizing the disc (any LED rated for 30mA pulsed)

c) Sensor:

An IR reflection detector SFH-9201 (Infineon / OSRAM) is used as optical sensor for disc speed control. Each disc revolution generates 30 pulses. So in case of a 60-line system it counts “double lines”. The synchronization system can handle an incoming video at 30 synchronization pulses per second as well as at 60 synchronization pulses per second (1 video pulse on 2 is then used).





This small SMD component is mounted on a separate circuit board and protected from a too strong disturbing ambient light by a plastic cap taken from an old photo film box. The black and white cardboard counting track features 30 B/W cycles.

Adjustments

The schematic diagrams include many adjustable potentiometers (to be mounted on circuit boards) which are intended for system tuning and should not require further settings as soon as everything is properly working. Unless otherwise specified, all potentiometers tagged “Px” are these kinds of potentiometers that the user will not have to use.

Power supplies

The primary power supply shall be a DC voltage source within 12-16V. If the disc driving motor requires more than 9V5 for reaching the target speed, the low limiting power supply value will not be 12V but the required motor voltage + 2V5. In the schematic diagrams this power supply voltage is marked:
“V DC IN (+)”.

The post-diode voltage “VP” is provided (in this example) from the synchronization extractor schematic. This power source is used by circuits requiring power but without accurate voltage regulation.

The regulated 9V voltage used by several different circuits is provided by a 7809 regulator mounted on the synchronization extractor. (This regulator location choice is also not mandatory). It is marked “P9V”.

If used, the color decoder has its own power supply architecture: Directly connected on “V DC IN”, it has its own internal voltage regulator providing 5V through the source signal “VDD”. The only dependency that this circuit has from the rest of the appliance is the video synchronization signal (positive composite synchronization Hsync/Vsync from the synchronization extractor). By processing the SECAM carrier channel it is able to drive the red and blue LEDs directly.

Other signals shared between circuits

Wire terminating arrows tagged with signal names are signals:

- Generated by the circuit if the arrow is oriented output-wise.
- Used from another circuit of the arrow is oriented input-wise.

Schematic diagram style and default conventions

Diagrams are hand-made drafts. No nicer CAD sheets will be provided. However, care has been taken for clarity. Regulator and transistor pin-outs are assumed to be known (or easily found) by people having enough experience to get into this adventure. No BOM tables will be provided.

The default ratings are the following, unless otherwise specified:

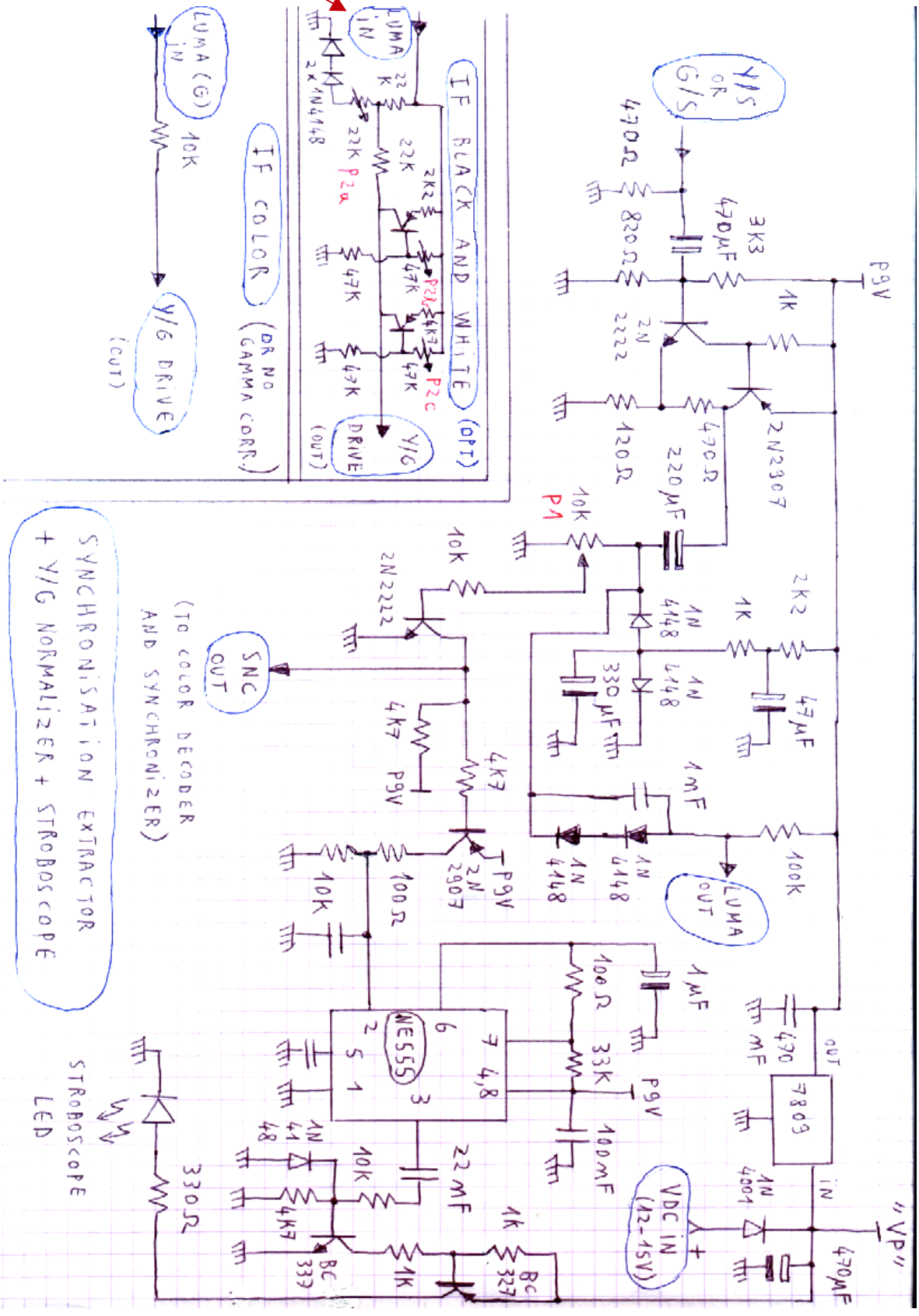
- Resistors and potentiometers: 1/4W

- Electrolytic capacitor voltages: 16V minimum
  (except for the color decoder and the synchronization extractor parts running under P9V where 10V is allowed)

- Simple heatsink might be required on the BD139/BD138 used in the motor driving circuit, depending on the disc inertia and motor efficiency.

Synchronization and luminance / green extractor

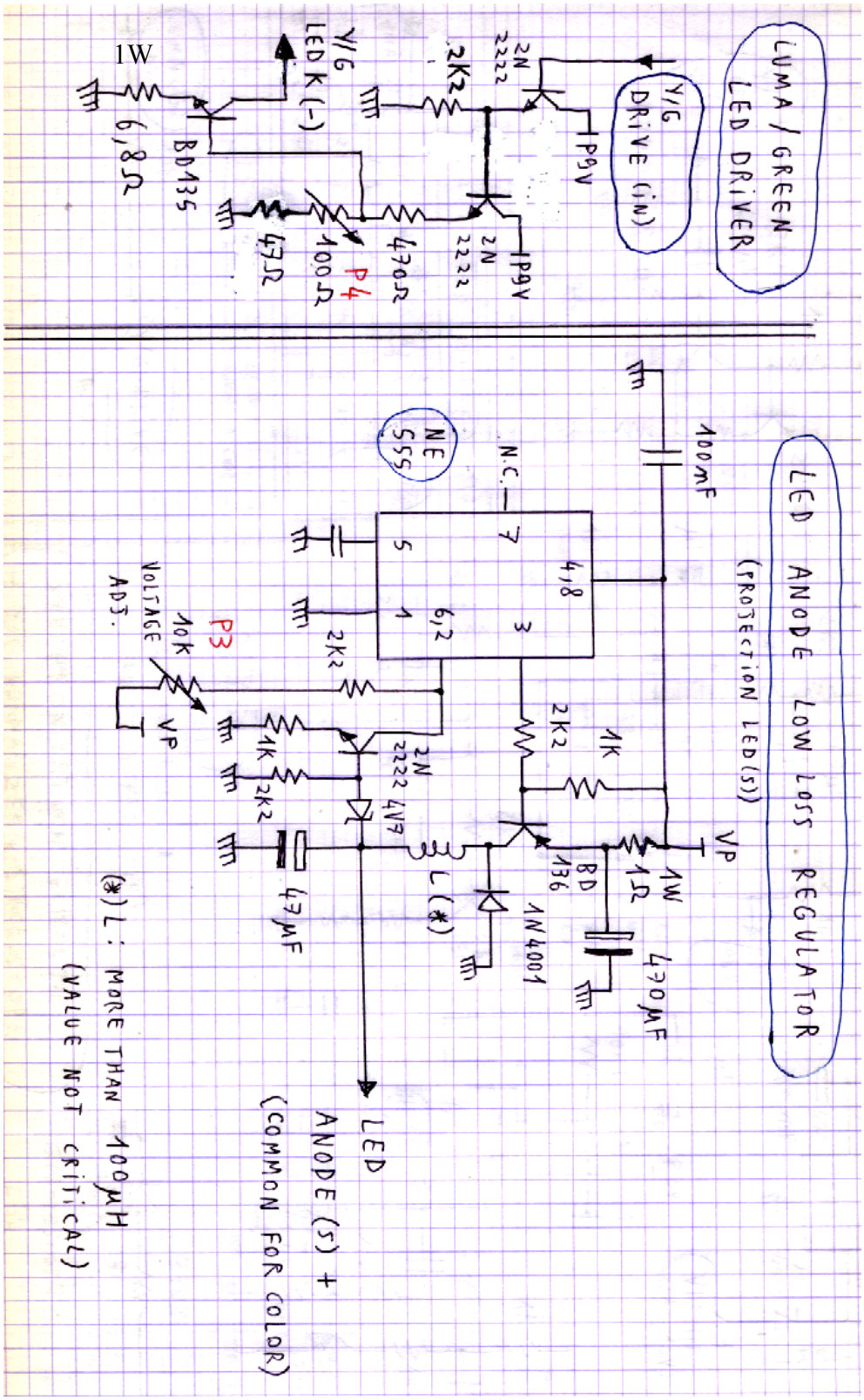
("LUMA OUT" can go through the "gamma correction" circuit to improve picture crispness in B/W. But this is not mandatory)



("Y/S or G/S" means: Input signal proving composite black and white or composite green video)

LED power stages

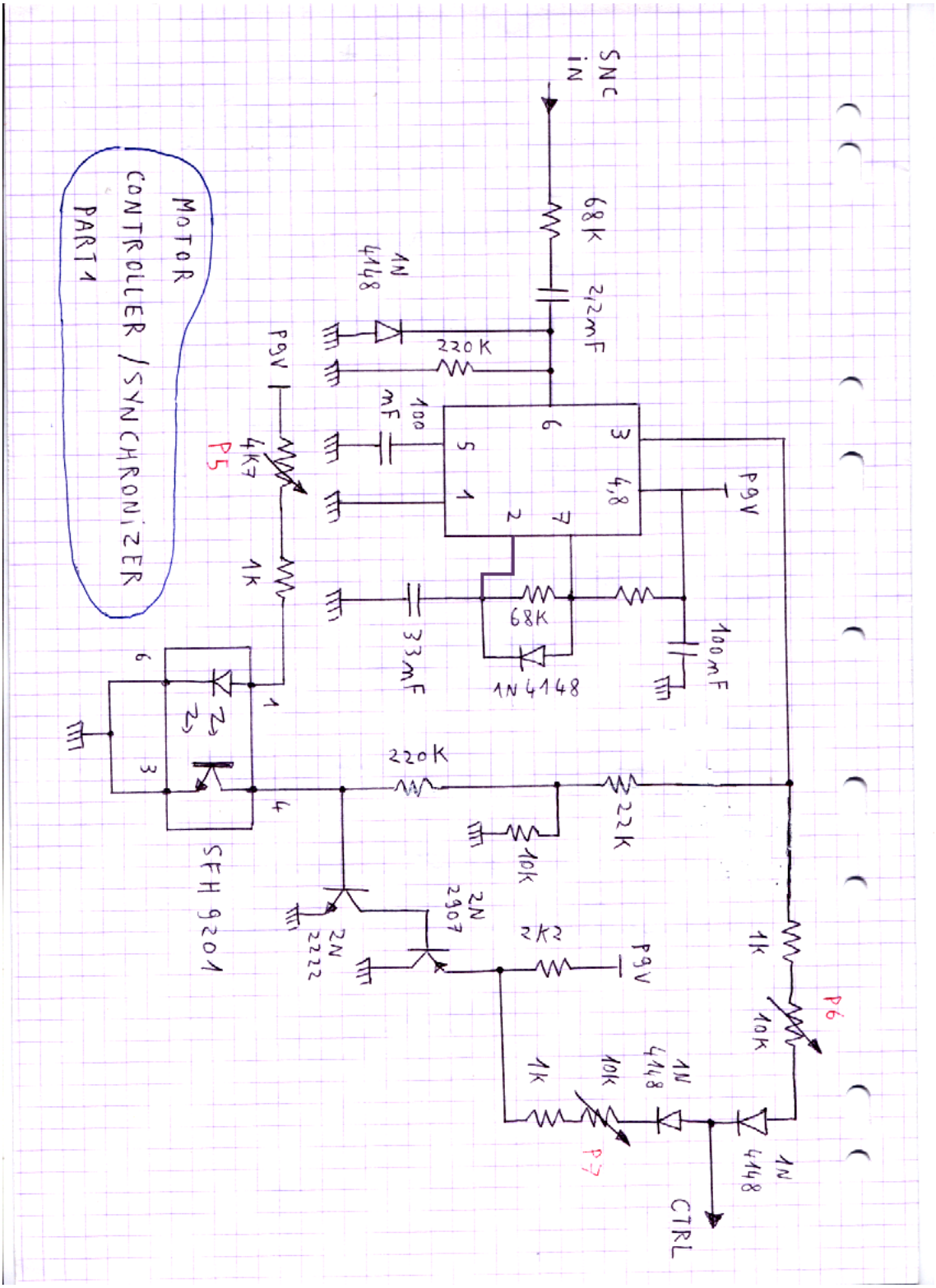
This is the only part requiring an inductor (but with low constraints on characteristics: L > 100uH, I > 0,8A).



(If not sure about the possible inductor value you may have in your salvaged component stocks, just test in in this circuit: After having mounted the inductor and started the circuit, load the regulator with a 6.8 ohms 5W resistor and check that the BD136 stays at a temperature allowing to touch it permanently. If not, the inductance is too low. If the inductor is getting hot, it is rated for a too small current).

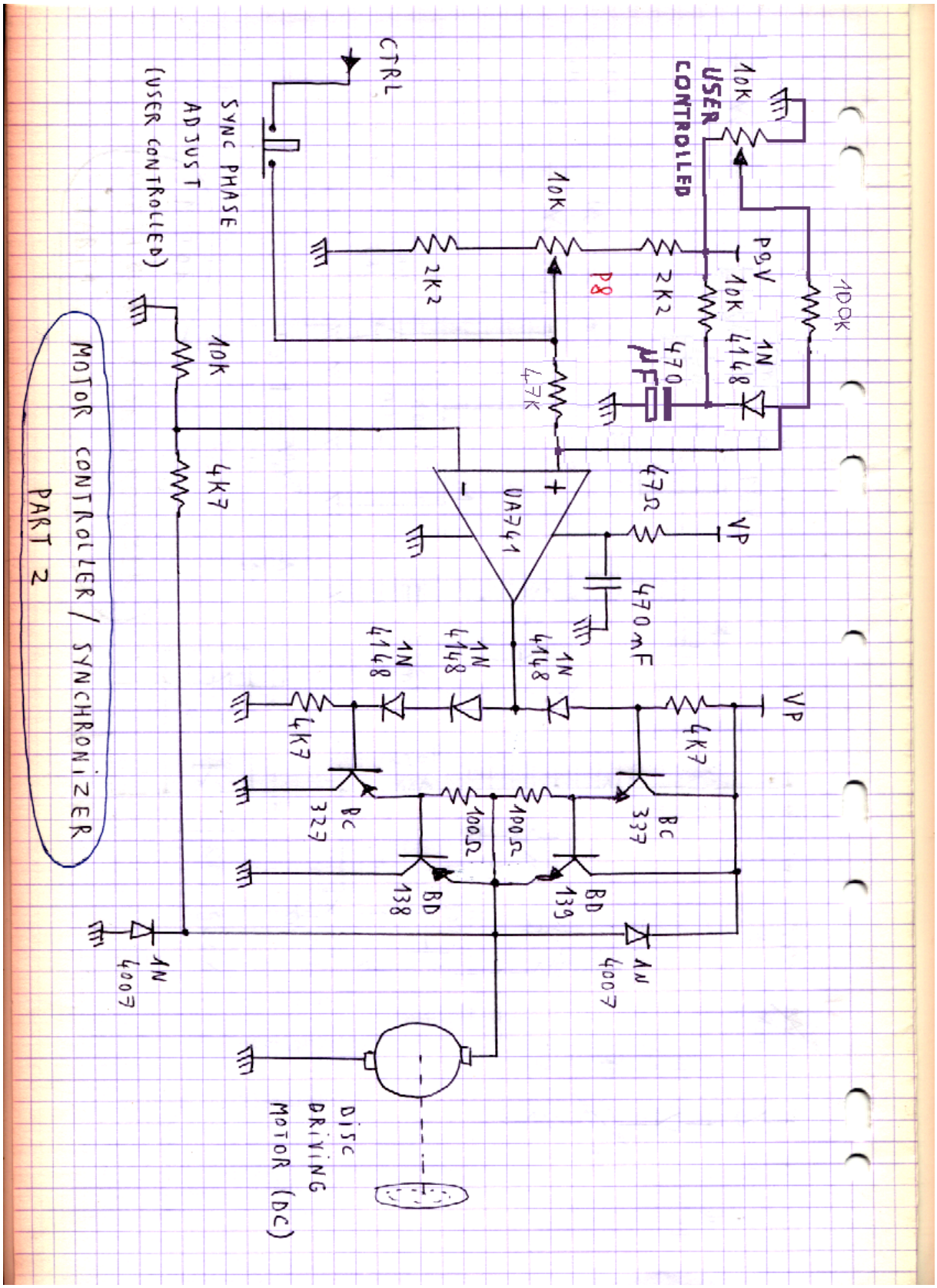
Motor driving – Control circuit

This circuit receives the extracted synchronization (like does the color decoder) and handle the optical sensor for issuing a control signal “CTRL” (polarity modulated pulses) which is connected to the power driver.



The NE555 has a monostable time constant set just slightly longer than a 60-line synchronization pulse interval, so that divides the frequency by 2 in this case. In case of a 30-line video this division does not happen but in all cases the output duty cycle is within [40%-60%].

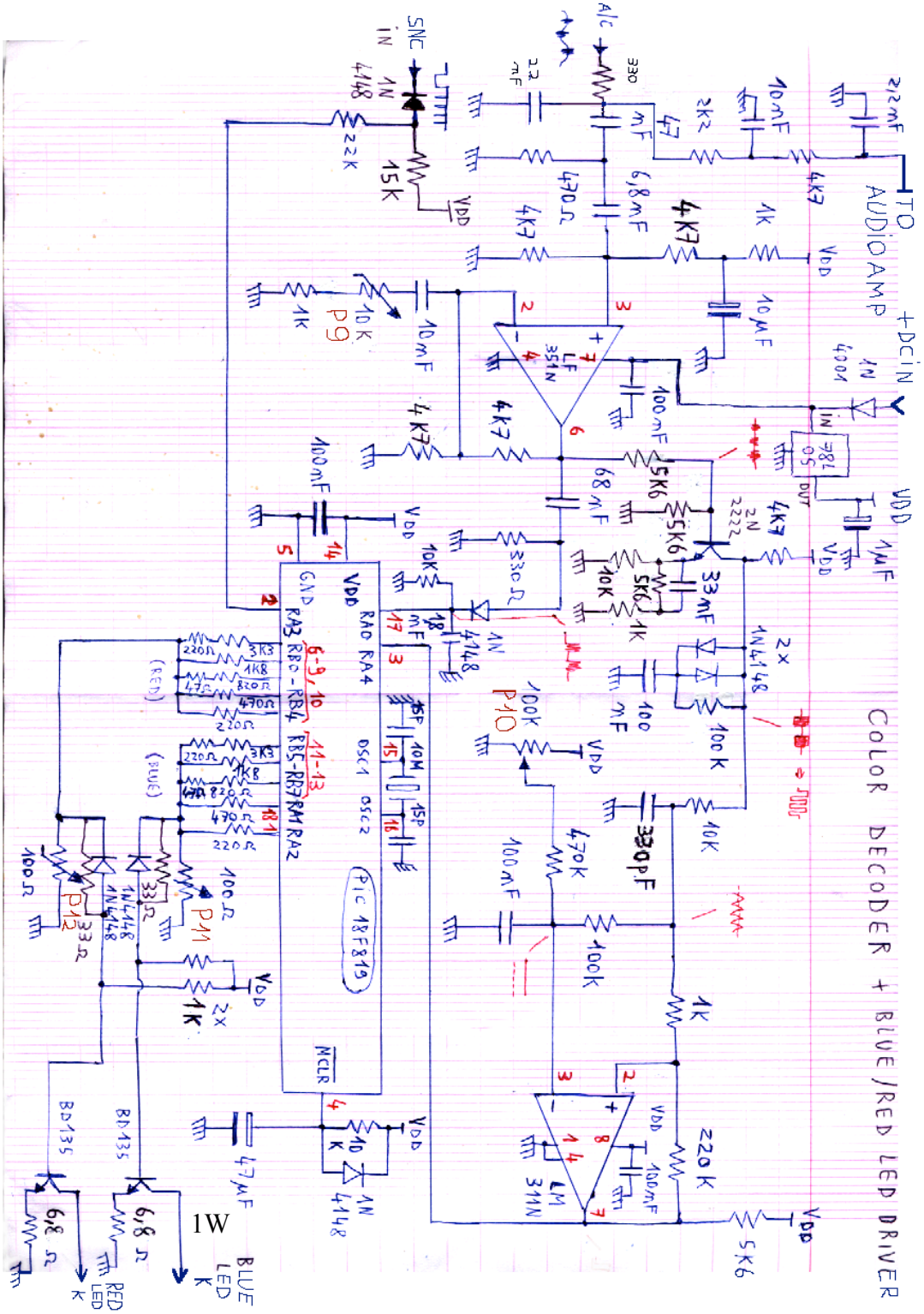
Motor driving – Power circuit



The free-run speed adjustment and the vertical synchronization pushbutton are visible here. P8 is used only during the tuning stage.

SECAM-like color decoder

This is the only part using a programmable component (code supplied): A common Microchip PIC16F819 micro-controller used to handle carrier synchronous demodulation (by synchronous sampling) as well as acting as the delay line required for SECAM (color alternated transmission, but here in AM). The frequency of the color carrier to be demodulated is the half of the WAV sampling rate (maximum possible frequency), so. 24KHz. The color pixel resolution is then the half of the green/white pixel resolution, while the same factor 2 applies for lines (due to color alternation). Therefore we can say that we use “quarter pixels” for color as it is usually done in high resolution video.



Recommendations applying for color systems

Whenever Windows or Linux is running on the computer to be used for playing the WAV files, the player application (Audacity, or the provided "playloop-48k" program, or any other) cannot directly send its stream to the sound card. The stream indeed flows through an audio mixer so that several applications can produce sound at the same time even if their output audio streams are at different sample rates. This is why the mixer resamples every source at a targeted output sample rate for being able to mix.

When just playing audio or even when sending just black and white NBTV video, the re-sampling frequency is not critical since it is expected to be equal or higher than the WAV sample rate. However, if a color carrier is part of the signal, some care has to be taken to avoid a totally disappointing behavior of the color decoder:

The mixer re-sampling frequency can be:
- Equal to 48Kspl/s
- Equal to the double, so: 96Kspl/s (synchronous resampling)

It must not be 44.1Kspl/s or any multiple of this frequency. Otherwise the resampling process will totally flaw the color carrier modulation.

The choice of 48Kspl/s as the source file and mixer sample rates has been made because PC audio hardware often runs by default at this frequency or at least supports it natively in any case. Soundcards usually re-sample any stream which is sent to them if this stream is sampled at a multiple of 44.1Kspl/s (and we absolutely don't want that).

If not sure about your settings, check the following:

a) Mixer settings under Windows (7 and further)

In the contextual menu of the speaker icon located in the task bar, select the playback devices. Then, edit the properties of the output you plan to use and select the "advanced" tab. The sample rate should be 48000 Hz (16-bit) by default. Otherwise set it to this value.

b) Mixer settings under Linux

* If running Pulse Audio, edit the following file as superuser:
  /etc/pulse/daemon.conf

  Then update the following line (while removing the ";" at the begining, if present):
  default-sample-rate = 48000

* If running Alsa, edit the following file as superuser:
  /etc/asound.conf

  Find the following section and update it this way:

```
pcm.device{
    format S16_LE
    rate 96000  (or: rate 48000)
    type hw
    card 0    \
    device 0  / Do not modify this section if different
}
```

c) The final real check: Output monitoring with an oscilloscope

This test can be led without any additional hardware, except the cable that you will use anyway to connect to the NBTV television system. Even a cheap and/or very old analog oscilloscope can fit for this test.

Just connect an oscilloscope input to the computer audio channel corresponding to the color carrier. Launch the color pattern file "colorploop-x0.wav" (detailed in the "software tools" chapter) and adjust the time base to see only a few carrier periods. The following must be observed:

* For a soundcard output rate at 48000Hz: Raw square wave, varying between only a few discrete amplitude levels (because of the simple color bar modulation used).

* For a soundcard output rate at 96000Hz: The square wave may be slightly distorted but in a stable way. For instance, two-step low / high levels like this:

The time varying signal amplitude must also show only a few discrete levels.

In both cases, adjusting the time base to get two video lines must allow to see long and clean carrier bursts with stable amplitudes, similar to what can be viewed in the WAV file via an audio editor like Audacity.

If the applied resampling is inappropriate, then the carrier seen over one video line on the oscilloscope will appear as sinus modulated, completely flawing color demodulation.

d) Avoid any kind of equalization

If your soundcard management utility and/or mixer includes an audio equalization feature, don't forget to disable it!

e) Audio CD players

The the reasons exposed here above, it is obvious that using an audio CD player (sampling at 44,1KHz and filtering at 22KHz) cannot lead to successful results with the presented color system.

Composite Y/Green video characteristics

The hardware expects positive video. This means that synchronization pulses step down at the lowest voltage level (under the video black level) while visible signal varies above the black level.

WAV files are recorded this way but it is not guaranteed that the computer will not invert signal polarity. (It is designed for audio where polarity has no importance while it is the same on both channels). The color decoder is not sensitive to polarity inversion but the synchronization extractor does. Therefore the oscilloscope must also be used to check if the video signal is inverted or not. If it is inverted, you can easily use the "soft" way: Process the WAV files by using the feature "Invert" in the "Effects" menu of the Audacity sound editor (http://www.audacityteam.org/). You may also add a selectable hardware inverter according to your preferences...

Remark: To help keeping a rock stable synchronization extraction, even in the case of fast brightness changes (very important for a mechanical TV), it has been prefered to apply a black level at 50%, leaving the remaining 50% for the visible signal, while classical video standards define 30% and 70%.

Software tools

The following software tools are provided in the archive containing the present document:

a) Basic tools for tuning the system

* The WAV files "colorploop-60.wav" and "colorploop-30.wav" containing a prepared color bar pattern:

(Black line and mid-height, and darker level in the bottom part)

* For Linux users, the executable "playloop-48k" allows to continuously play in loop the WAV file to be used. It can be compiled as follows:
# gcc playloop-48k.c -o playloop-48k (oss-devel package to install through your package manager)

Then launched this way:
./playloop-48k colorploop-60.wav (for a 60-line system)

* For Windows users, no loop playing tool is available from me at this time. So, unless you already have one preferred player which does not implement equalization (!), Audacity (http://www.audacityteam.org/) is highly recommended. You will have to use the "looping" feature (http://manual.audacityteam.org/man/tutorial_looping.html) or to import and paste the WAV file several times on the track. Indeed, "colorloop-x0.wav" is too short to be played in single shot while giving the time to analyze and tune. Also ensure that loop join is perfect and does not bring any phase step in the synchronization. Otherwise you would encounter troubles during synchronization tuning!

b) Tools for converting videos to NBTV WAV files

Once the system is tuned, everyone is impatient to see how it behaves with "real" videos.
The third party software to be used in addition to the provided tools is FFMPEG. It is available as well for Linux (installed through the package management wizard of your distribution) as for Windows: http://ffmpeg.zeranoe.com/builds/

Windows binaries are provided, while Linux users can compile the single file sources as follows:
gcc nbtv60color.c -o nbtv60color
gcc nbtv30color.c -o nbtv30color
gcc grey2rgb.c -o grey2rgb

These binaries, as well as FFMPEG, are intended to be called by scripts which perform an "all-in-one" process. So once everything is installed, the script "video2nbtv_x0.sh" or "video2nbtv_x0.bat" has to be launched, according on the OS you run. However, considering your disc spinning direction and your spiral profile, first edit the script to check at the last line if the arguments passed to nbtvx0color correspond to what you need:

Argument "-i" sets line rasterization order from the bottom to the top (but remains left to right)
No argument means classical rasterization: Left to right, top to bottom.

(This applies if you create the picture at the top of disk while encoding non-rotated videos. Otherwise you will have to cogitate a little bit with spatial considerations)

If the process ends successfully you should get an output file "NBTV60-OUT.wav" or "NBTV30-OUT.wav". Having a short look at it with Audacity can help checking that it looks like NBTV video. You can then rename this output file in a more user-friendly way, so that it will not be overwritten at the next script call.

c) Possible line offset correction

When creating NBTV WAV files it is possible to take into account a possible positioning error made when drilling the disc holes. Of course this correction can only apply in the line scanning direction by adding variable delays in video lines. This is why each version of converter gives the possibility to add a text file named "shifdef.dat" which will be loaded from the place where the converting script is called.

In this file featuring 29 or 59 lines depending on the used video resolution, line delays are expressed in pixels via decimal signed integer values which just have to be entered without any other characters.

Example:

-1
2
-3
2
(... until the 29th or 59th line...)

A negative value means a "negative delay" (first pixels lost and black pixels added at end-of-line).

A positive value means a "positive delay" (black pixels added at the beginning and last pixels lost).

The settings can be tweaked through several iterations to get the best result while using an encoded bar pattern. An interactive tool can help for this (end of document), so that Linux users can create "shiftdef.dat" automatically after having "moved" bars line by line to align them through keyboard controls. This kind of tool was used for the prototype shown in the Youtube video.

Note for color systems:

In a SECAM transmission one single color line is shared by two scanned lines (blue is acquired while red is replayed, then red is acquired while blue is replayed, etc...). This means that line offsets cannot be corrected as well as in black and white. This can be observed in the Youtube video (case of a very bad Nipkow's disc) where color pictures are much more scrambled than black and white pictures shown earlier.

However, the color NBTV converting tools do handle "shiftdef.dat" anyway by applying a trade-off for color: The offset applied to the currently transmitted color is the intermediate value (average) between the currently scanned line offset and the next coming line offset. This can help in case of moderate corrections, but not in the case of the disc prototype spinning in the Youtube video.

Code for programming the color decoder micro-controller

The assembled code file "colordec.hex" is ready for programming the PIC16F819 with any Microchip programmer (for example Pic Kit2 or more recent versions like this one: http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=PG164130 , very affordable and efficient tools). The source file "colordec.asm" is provided for information or for your own modifications, if interested in.

Note: It's easier to program the micro-controller on a separate DIP IC socket where you will solder 5 wires according to the programmer device documentation.

Proposed tuning procedure

These guidelines are not strict procedure steps but intends to explain a possible method that I consider as simple and quite logic. It is probable that sometimes you will have to step back and forth in this procedure for finer adjustments during tests. It is obvious that you will need an oscilloscope. Even a basic, old and analog oscilloscope can fit.

Starting conditions:

First set the potentiometers in the following positions:

* P1 set at mid-range
* P2a set to maximum resistance \
* P2b set to maximum resistance | - If the gamma correction circuit is used
* P2c set to maximum resistance /
* P3 set at mid-range
* P4 set to zero resistance
* P5 set at maximum resistance
* P6 set at maximum resistance
* P7 set at maximum resistance
* P8 set at mid-range
* User-controller speed potentiometer set at mid-range
* P9 set to maximum resistance \
* P10 set at mid-range | If the color decoder is used
* P11 set at zero resistance |
* P12 set at zero resistance /

After having checked all the usual basics for each separate circuit as in other hardware preliminary tests (power supply voltages, reasonable current consumption, no suspicious heating, etc...), power-up the complete system, except the color decoder if present. Also disconnect at this time the disc driving motor from its power driver as well as the green or white LED from its driver.

First tuning phase:

Start playing the color bar pattern and adjust the computer volume so that the composite signal (green or luminance) is about 1V peak-peak at the maximum levels.

Then check the signal on “SYNC OUT” in the synchronization extractor circuit, while viewing an entire video frame. While turning P1, look for the two limits where the extracted synchronization signals becomes degraded and then replace the cursor at the middle of these two limits.

Once the synchronization correctly operates, the stroboscope LED must flash at periodic intervals (12.5Hz, the video frame refresh rate equal to the disc rotation speed in RPS).

Then check the back-light (video controlled) LED anode supply voltage (in the low loss regulator circuit), while loading this output by about 100 ohms, just to drain some current from it. This voltage appears like a DC voltage plus a superimposed ripple which is not an issue. Considering the maximum expected VF for the white or blue LED (about 3.5V), the common anode voltage should be set above VF + 2.4V ~ 6V to have the LED current driver(s) working properly. So adjust P3 to have the lower ripple peak at 6V.

It's time know to set the maximum green LED current by using P4. Replacing the LED by a 4.7-ohm resistor to reduce transistor heating, set P4 so that the maximum peak voltage on the 6.8-ohm resistor (BD135 emitter) is: [target_current] x 6.8.

For the green LED (part of the RGB LED) I recommend not to exceed 240mA per LED, while the proposed white LED (for the black and white version) can handle much more. If wanted, you will not reach the maximum rated LED current of 900mA with this circuit, unless you change the 6.8-ohm emitter resistor for a smaller value and increase the 47-ohm resistor in series with P4. You will also need a heat-sink for the BD135.

Once the video circuits are set, it's time to adjust and test the optical sensor. It is mandatory to have it mechanically installed in front of the fully equipped disc, as in the definitive setup. To have the disc spinning it's possible to move it by the hand or to power the motor with an external power supply (for instance a battery, even if the disc spins slowly). If the "counting track" sectors are correctly made (dark black and bright white), if the distance to sensor is correctly adjusted and if the sensor is correctly protected from strong ambient light, then there must be a position for P5 for which the 2N2907 emitter "toggles" at each counting sector all along the disc rotation angle. This should happen flawlessly all over the disc rotation. By adjusting P5, look for the limits where operation is degraded then set it back in the middle.

Now it's time to check that the NE555 correctly generates on pin 3 a square wave synchronized with Hsync pulses. No particular adjustments should be required if using 30 or 60 lines per second.

In the motor power driving circuit, keep the user-controlled potentiometer at mid-range and, while pushing the "Sync Phase" button to disconnected "CTRL", set P8 so that its cursor voltage equals the motor voltage required to reach 12.5rps (from your previous motor evaluation tests), plus 5%.

Release the pushbutton. The P8 cursor voltage will then show some superimposed positive and negative pulses (unsynchronized) when rotating the disc. Perform a first adjustment of P6 and P7 to have these both pulse amplitudes at about 20% of the DC voltage.

Second tuning phase:

We are now ready for a tuning in real conditions. After having switched the system off, connect the motor and the white / green LED to their respective drivers. Then turn the power supply back on while keeping the "SYNC PHASE" button pressed.

The motor driver has a "soft start" feature which allow limiting the motor starting current. However, after speed stabilization (a few seconds) it has the ability to relaunch and brake disc rotation with low impedance against inertia to keep the best possible stability. When the free run speed is stable, keep on holding the button pressed while adjusting the user-controlled potentiometer so that the stroboscope shows that the disc is spinning slightly too fast (apparent speed at about 0.25 RPS). If this cannot be obtained then fine adjust P8 to reach this result. Then release the button…

If you are lucky, the stroboscope shows that the disc rotation is synchronized! (Apparent immobility). Then you will just have to play with the button to get the angle mark (as shown in the Youtube video) at its appropriate location. A stable picture must then be seen on the back-light illuminated disc sector!

If you are not lucky you'll see the disc apparently coming back and forth under the stroboscope light, or even totally out of sync. This means that you have to fine tune P6 and P7, or even P8.

A synchronized disc showing an excessive angle jitter (apparent oscillation under the stroboscope light) is the result of a too strong inertia against motor torque capabilities. Try anyway to change the settings of P6 and P7 to change the error gain and see if satisfying results can be obtained.

A disc totally out of sync can result from:
- P6-adjusted control pulses too weak versus P7-adjusted control pulses.
- P7-adjusted control pulses too weak versus P6-adjusted control pulses.
- P8 set too far from the appropriate free-run speed.

There is no simple magic. Just try to get the good recipe. ;-)

Color decoder tuning (if used):

The color decoder can now be connected and adjusted if you planned to use it. In that case you should be impatient to add blue and red to your green and unclear picture!

First check the signal on pin 6 of the LF351N (front amplifier). You should see the same as on the color carrier channel from the computer, plus a DC component. Adjust P9 to get a peak-to-peak maximum amplitude (occurring during the biggest color bursts) at 6V.

Then check the signal on pin 7 of the LM311 (right part). Adjust P10 to obtain a square wave at a duty cycle of 50% and perfectly synchronized with the color carrier. (If the signal is stable, then it is synchronized).

Then check the voltage on one of the two 6.8-ohm resistors (LED drivers, bottom right) while slightly rotating the corresponding potentiometer (P11 or P12). Increase the level until the maximum voltage (during the biggest color bursts) reaches 1,7V. Process the same way for the second LED driver.

If everything happens as expected, the picture on the disc must now be in full color!

Gamma correction tuning:

If this small circuit is inserted before the white LED driver (black and white version), the goal is not create this kind of exponential function called gamma correction. The "scientific" way for tuning requires the creation of a test pattern where video lines are linear increases. Both transistors are intended for changing director coefficients within the ramp-up while the diode network sets the upper clamping. Then the three P2x potentiometers can be adjusted until the gamma curve is seen at the output, shown by the oscilloscope.

The more empiric way is the watching of real videos while adjusting the three potentiometers until it appears as pleasant, crisp and clear… ;-)

Using disc error correction – Calibrating tool

In case where the line offsets caused by disc's holes position errors makes the result being too disappointing, it is possible to apply a software correction at WAV file generation time, as explained on page 14. The program "shiftune" (available only for Linux at this time) makes this far easier than a manual editing of "shiftdef.dat" leading to many iterations:

Launch the corresponding "shiftune-x0" corresponding to your resolution. The possible arguments are:
- No argument: Classical scanning, left to right, top to bottom
- Argument "-i": Scanning from left to right but from bottom to top
- Argument "-n": If the PC audio output inverts video polarity
- Argument "-ni": Activates simultaneously "-i" and "-n"

When started, synchronize the disc and then look at the two displayed vertical bars (in green or white depending on if you built a color or black-and-white system). The interactive console based user interface on the PC screen guides you to adjust a correct alignment, line by line. Once done, press "Esc" and "shiftdef.dat" is automatically created, ready to use!

FINAL NOTE:

This set of guidelines does not deal with the common but necessary checks that any careful experimenter has to perform for any kind of electronic hardware. Just a few examples to avoid damaging the most critical and expensive components before starting the tuning procedure:

- Ensure that LED drivers cannot damage the LED if transistors are not mounted correctly (first tests with dummy loads).

- Ensure than VDD is well regulated at 5V in the color decoder before inserting the micro-controller on its socket.

- Ensure that transistor bias points do not show aberrant behaviors (pin-out errors, damage…) and that they are not overheating.

The author, V. Portet
vpapounet@gmail.com