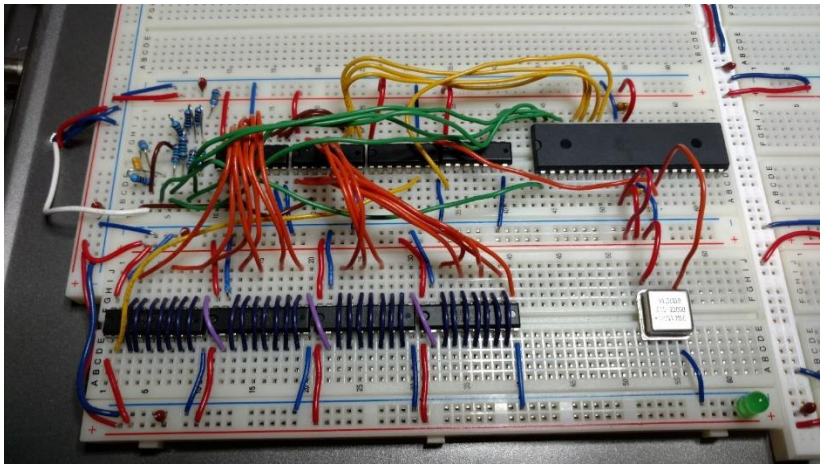
Composite Video Example Circuit

John Smigel

21 February 2018



1.0 Basic Electronics Introduction

All electronic circuits work by controlling how electrons move around. Electrons are tiny negatively charged particles. More electrons jammed together with more energy produces a more negative potential (to do work) or voltage. A lack of electrons produces a more positive voltage. Electrons will flow from a negative to a less negative or positive voltage, V, if connected by something that conducts electrons (electricity). How well something conducts electricity depends on the material properties and the number of free electron in the material. Metals are usually good conductors. Silver, copper, and gold are three of the best conductors. How well electrons flow is characterized by the material's resistance, R. The flow of electrons is called current, I, and is measured in amperes or just amps, A. Ohms law tells the relationship between voltage, current, and resistance in a circuit,

V = I R (voltage equals current times resistance)

You can tell the resistance of a resistor by the pattern of colored bands around it, but I find it much easier and faster to just measure it with a meter or keep it in a labeled container.

In addition to resistors and wires, electronic circuits will have a voltage source, and electronic components; usually capacitors, C, and integrated circuits (IC). Capacitors store electrons and resist a change in voltage.

Analog vs. Digital

The electronic components will be analog, digital, or some combination. Analog means that the voltage levels used in the circuit have many values (continuous), where digital circuits use only 2 voltages representing 'on' (1) and 'off' (0). Actually, it is just voltages above some level that are 'on' and below some other level are 'off.' In digital circuits, information is stored in binary form where everything is represented by a string of ones and zeros. For example, two binary digits can represent: 00 = 0, 01 = 1, 10=2, and 11=3. If we add 1 to 11=3 we need another digit and would get 100=4. The n^{th} digit from the right represents a value of 2 raised to the n^{th} power (2^n) if its value is 1. The total value is the sum of all the values represented by all the 1's. Letters are also represented by a string of 1's and 0's. The standard codes for letters and numbers are called ASCII and contain 8 bits each, called one byte of data. Digital integrated circuits (IC) manipulate and/or store digital data in some way. These digital circuits are often called gates. For example, an AND gate would take two binary (1 or 0) inputs and output a 1 if both inputs are 1 and output zero otherwise. An OR gate outputs a 1 if either input is a 1.

Television (TV) Video Signal Basics

The early TV signals were analog and just black and white. The picture was created by what is called a picture tube or a cathode ray tube (CRT). A CRT is just a device that shoots electrons at a screen that glows when electrons hit it. The air is sucked out of the glass tube (called a vacuum tube) to make the electrons flow easier. The cathode is a plate that has a high negative voltage or charge (many excess jammed-together electrons). The electrons are pulled off the plate and shot toward the screen by a positively-charged anode. Magnetic fields steer the electron beam vertically and horizontally to draw the picture on the inside of the tube. We see what is drawn on the outside of the tube as the TV picture. The input analog voltage controlling the black and white level is called the luminance (brightness) or luma signal. It controls how strong the electrons are that shoot at a given point on the screen. No

electrons produces black and many strong electrons produces white. The inside of the tube is coated with a phosphorescent material that glows when electrons hit it.

The electron gun is steered horizontally across the screen at a particular rate and also vertically. The sync signals tell the TV scanning circuits when to start scanning. When color was added, it was added in a way that the original black and white luma signal stayed the same and just the new color signal, chrominance or chroma, controlled the color. Within an old TV, color is produced by having three electron guns that are aimed at phosphor locations that have different colors, red, green, or blue. All colors can be constructed by some different amounts of red, green, and blue (RGB). The red, green, and blue dots are so close together that we just see the resulting combined color. The committee that produced the TV signal specification is called the National Television System Committee (NTSC), originally in 1941 and with color added in 1953. Some people say NTSC stands for "Never The Same Color."

So the three main components of the color TV signal are the luma, chroma, and sync signals. I'll give more details on the TV signals and format as I describe how the circuits operate.

2. Video Interface Circuit

Electronic circuits are designed and represented by what is called schematic or circuit diagrams. I am using a free open source electronic design tool package called KiCad. Such tools are often called Electronic Design Automation (EDA) tools. KiCad includes a schematic design tool called Eeschema. I'm not going to try to teach about KiCad or Eechema here because there are already good tutorials included with that package. The video circuits drawn in Eeschema are shown in Figures 1 and 2.

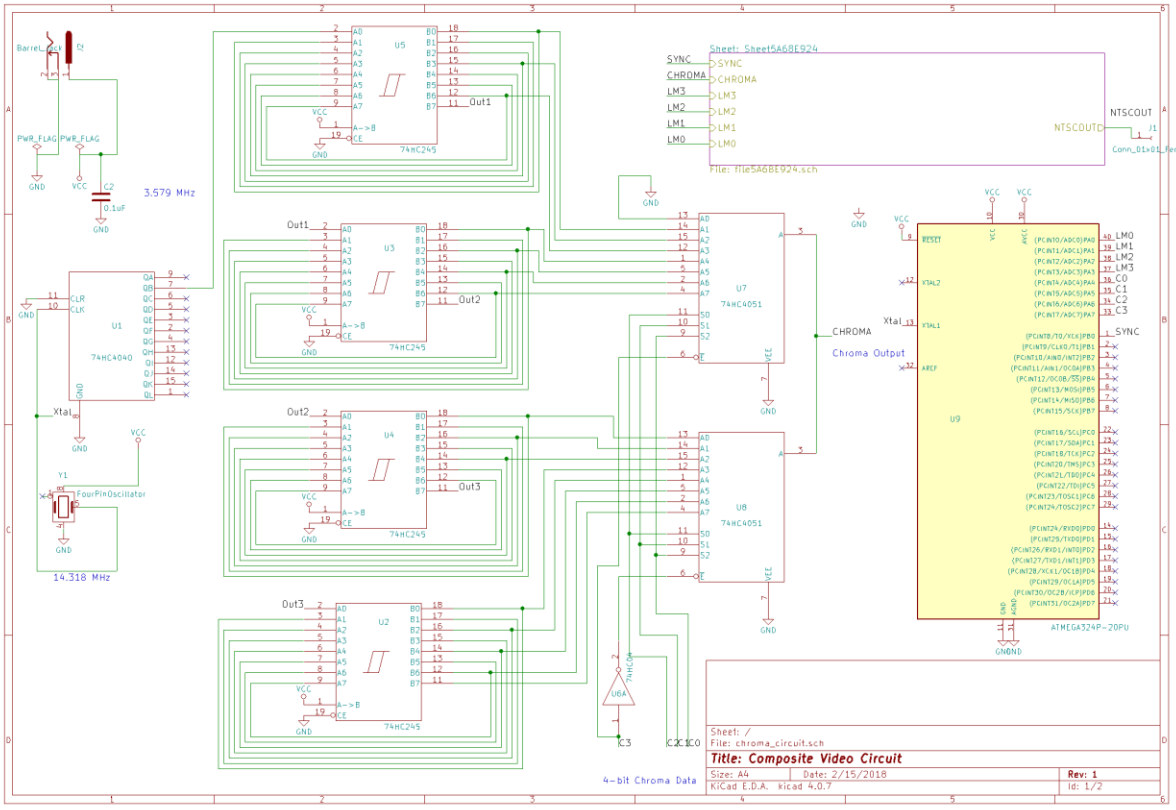


Figure 1. Main Video Circuit Schematic

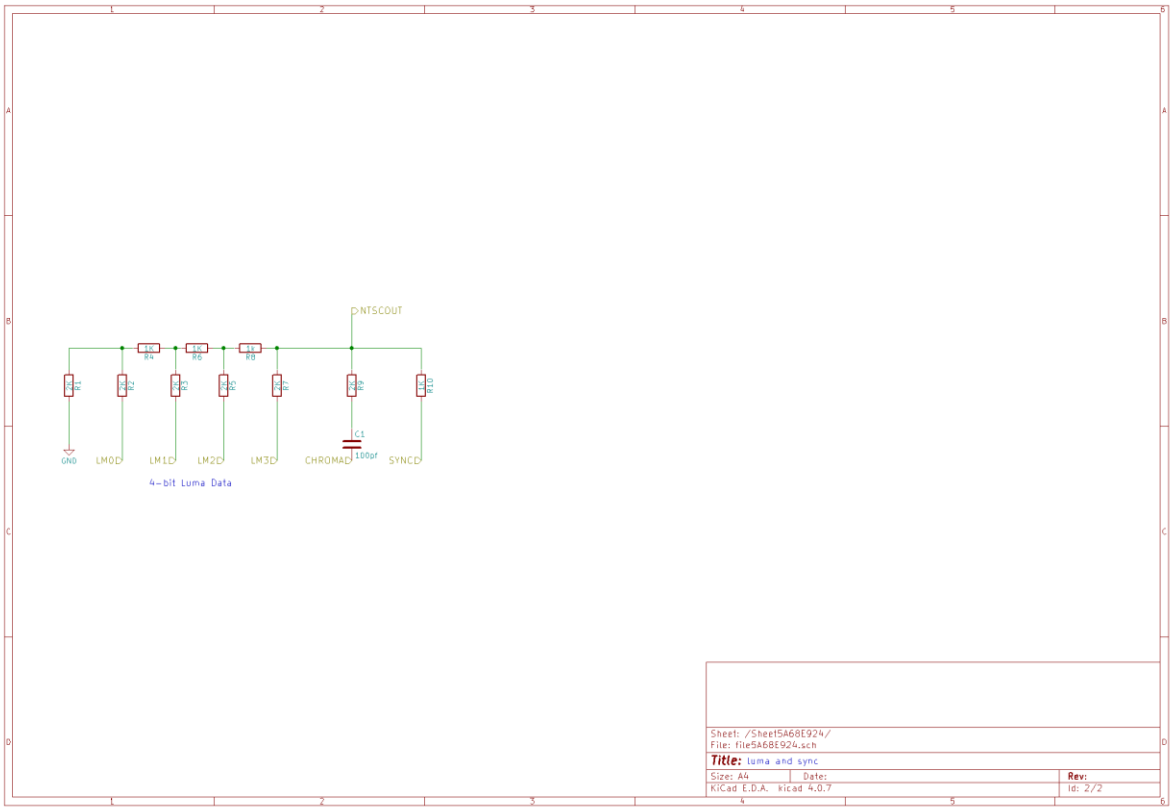


Figure 2. Subsheet of Schematic for Luma Generation and Combining with Chroma and Sync

The Figure 2 subsheet is the part in the upper right corner of the Figure 1 schematic.

I'll describe the circuit components and how they work from the left to the right. At the top left is a connector for the power supply input. I am using a +5 volt power supply because that works with most digital logic devices. The capacitor between the +5 V and ground helps keep the voltage constant. The capacitor value is 0.1 uF, the uF stands for micro ($x10^{-6}$) Farads. At the bottom left is an oscillator device that outputs a clock signal at a frequency of 14.318 MHz. That means the signal voltage moves up and down in a repeating pattern, repeating 14.318 million times in a second (pretty fast). This clock is used to derive other lower-frequency clock signals. The highest frequency clock is called the master clock. Clock signals derived from the master are all in sync and are said to be coherent and synchronized. The oscillator output goes into the U1 device, 74HC4040, and the microcontroller (MCU), Atmega324P-20PU crystal XTAL1 input. Often just a crystal is used to generate a clock signal, but I'm using a stand-alone oscillator circuit because this clock is generating the chroma signal too. So why 14.318 MHz? First I'll explain the units of frequency, Hertz or Hz. A Hz unit is one cycle per second. When signals are sinusoidal and the angle changes over the cycle, one complete cycle occurs after 360 degrees or 2*pi radian angle units (pi~=3.1415926). The picture image can be thought of as being made up of individual dots on the screen where the level and color value change from dot to dot. The dots are changed across the screen horizontally first and then vertically. A horizontal sequence of dots is called a line. So if you

see a CRT screen mode listed as 800x600 (HxV) this means there are 800 dots (also called pixels) horizontally and 600 pixels vertically for a total of 800x600=480,000 pixels. Using this terminology, the TV display produced by the example circuit here is 224x242 pixels. For a TV, the time the electron beam scans across a horizontal line from left to right is fixed at 63.56us. The unit us is microseconds or 10^{-6} seconds. This time is represented by, H, in the TV specs. Most other times are specified relative to H. A frequency or rate is obtained by taking 1 divided by the time (reciprocal of time). So the line rate is 1 cycle/(63.56us) = 15.73k cycles/s = 15.73kHz. The kHz unit is kilohertz or 10^3 = 1000 Hz. This means that there can be 15.73 thousand lines drawn in one second.

The two frequencies that we need to generate from the master clock are the chroma signal frequency and the pixel clock frequency. The chroma signal frequency is specified as 3.579 MHz. The pixel clock frequency is the rate at which we change the pixel value (draw a new pixel). We need to know the time between pixels. How fast a pixel can be changed depends on the bandwidth of the luma signal. The bandwidth is the difference between the largest and the smallest frequencies that are simultaneously present in the signal. All signals can be broken down into a sum of components at different frequencies. This is called the frequency spectrum of the signal. The time needed to change the signal is inversely proportional to the signal bandwidth. Bandwidth through space is limited by what has been allocated for a particular purpose, by the Federal Communications Commission (FCC) in the US.

TV channel total bandwidth is limited to between 6 and 8 MHz, depending on channel frequency. The pixel rate will need to be less than the available bandwidth for it to be possible to generate that rate. For this circuit we are using 224 pixels on a horizontal line. The horizontal line time actually includes additional time for the sync signal and some other signals. The time spent drawing one line is about 47us. The time we have to change pixels is then this line-drawing time, 47us, divided by 224 pixels or 0.21us/pixel. The pixel frequency rate is 1 over this time, 1/0.21us=4.77 MHz. The MCU needs to change the pixels at this rate. It takes 3 MCU instructions (3 clock cycles) to change the MCU output data driving the luma and chroma values. Therefore, the MCU clock needs to be 3 times the desired pixel rate or 3x4.77MHz=14.318MHz=master clock frequency. The necessary chroma signal frequency is generated by dividing the 14.318 MHz master clock frequency by 4, or 14.318MHz/4=3.579MHz.

The first component is U1 and the component type is 74HC4040. The 74HC specifies a particular digital logic family of devices. The HC is high-speed CMOS (Complementary Metal Oxide Semiconductor). The different families have different ways they are constructed, resulting in different power, voltage inputs, number of outputs they can drive, temperature range, etc. Appendix B summarizes some differences in logic families. All the information for a particular device is in the specification (spec) sheet for that device. The number at the end, 4040 in this case, is a code for what specific function is performed. 4040 is the code for a synchronous 12-bit binary counter. A counter can be used to generate a clock signal that is lower by a power of 2 than the input clock signal. Since the device counts input cycles in binary, the output bits cycle at a lower rate as they go toward more significant bits. Here we take the 2^{nd} least significant counter output bit, QB, that divides the input clock by 2^2=4.

To understand the next part of the circuit, you need to understand a little about the chroma or color signal used by the TV. Because the TV signal would originally be sent over the air, all the individual signal components needed to be combined into a single channel. This means they all need to be combined in a way that they can be separated by the TV. Combining signals is called multiplexing them and separating them is called demultiplexing. To multiplex signals together in a way they can be

separated requires some dimension or domain in which they are separated. Some possible separation domains are: 1) time, 2) amplitude, 3) frequency, and 4) phase. The luma and chroma signals are separated in frequency by having the chroma signal at a very narrow and specific frequency that can be extracted. The color information is coded in the phase of this chroma signal. The phase is how far the signal has reached into its cycle. The cycle phase spans from 0 to 360 degrees. For more accuracy, the color phase is defined as the difference in phase between a reference signal sent at the beginning of each scan line and the phase sent at each pixel. The measured phase difference then maps to a particular color for the TV to generate. The reference signal is called the color burst signal.

The next part of the circuit is 4 identical components, U2 to U4, of type 74HC245. These are called Octal BUS transceivers. The octal means that there are 8 channels (1 byte). A transceiver takes an input with some signal characteristics and converts it to a corresponding signal with specific output characteristics. In this case we are taking advantage of a fixed and known delay between the input and output of these 74HC devices. This delay is about 7ns (n=nano is 10^{-9}, s=seconds). This configuration where we have a sequence of delays all connected together is called a tapped delay line. We can choose a signal that is delayed by a specific amount relative to the input by selecting a particular tap.

So how does the delay relate to the phases that we need to generate to represent specific colors? The frequency of a signal is how fast the phase angle changes (phase angles in a cycle per second). The chroma signal has a frequency of 3.579 MHz. This means the signal goes through 360 times 3.579×10^6 degrees per second. So the time it takes to go through one cycle is the reciprocal of the frequency, $1/3.579 \times 10^6 = 0.28$us. The amount of phase shift produced by a 7ns delay is 360 degrees/cycle$*3.579 \times 10^6$ degrees/s * 7ns = 9 degrees. In order to span about 360 degrees, we would need 360/9 = 40 gates. This circuit uses 8x4=32 gates. This many gates will span about 32*9= 288 degrees of phase. We only need 16 phases that are different enough to produce different colors. Each different phase is interpreted as a different color by the TV. This circuit uses 16 different color phase values that are selected from the 32 different phases produced by the U2 to U5 integrated circuits (ICs). These selected chroma taps are sent to components U7 and U8 that are 8-to-1 multiplexers, 74HC4051. These devices take 8 inputs and output one of these eight, depending on the control input value. We are creating a 16-to-1 multiplexer by combining two 8-to-1 multiplexers together. Selecting one of 8 values requires 3 control bits. Selecting 16 values requires 4-bits of control. The extra bit is the most significant bit (MSB) of the chroma digital value from the MCU. This bit is put through an inverter gate in U6, 74HC04 inverter IC and used as the chip select for the two multiplexer chips, inverted for one chip, not inverted for the other. The inverter device just makes an output that is low (logic 0) if the input is high(logic 1) and high (logic 1) if the input is low (logic 0). Because the two chip selects are always inverted, only one of the chips is selected at a time and the multiplexer outputs can be tied together. Generally, two outputs should not be tied together if one output can be trying to be set high when the other output is trying to be set low. KiCad outputs a warning for connections that might cause a problem or any missing connections.

Figure 2 above shows the luma circuit and the circuits to combine the luma, chroma, and sync signals. The right side of the circuit has digital inputs labeled luma bits LM0,LM1,LM2, and LM3. LM0 is the least significant bit. This circuit is called a resistor ladder network and produces a simple digital-to-analog (D-to-A) converter. The purpose of this is to convert the 4-bit digital input to an analog voltage that corresponds to that digital value. A higher value of the digital input number, the higher the voltage. As the input is stepped through the 16 possible values, the voltage is stepped up from some minimum to

maximum value. You can calculate the voltages that will be produced at the top of R7 (luma output) by knowing the LM0-LM3 voltages (0 or 1 voltages) and the combined resistances. For resistors in series, the resistances just add. For resistances in parallel, the combined resistance is given by Rparallelcombined = (R1+R2)/(R1*R2). For example, two 2K ohm resistors in parallel will result in a resistance of (2+2)/(2*2) = 4/4 = 1K ohm. The rest of the circuit just adds the analog luma, chroma, and sync signals together.

Finally, the microcontroller (MCU) function is briefly described here. The microcontroller used is an Atmega324P-20PU device. The 324P number is the code for the specific device. Part of this device code sometimes refers to the amount of memory, 32 is for 32K of flash memory in this case. The -20PU is the maximum clock rate (20MHz) and the packaging (PU is the 40-pin DIP). The Atmega family is made by Microchip Technologies (used to be Atmel). There are many different MCUs available. The Atmega family is an 8-bit device popular with hobbyists and experimenting. The popular Anduino boards are basically an Atmega chip on a board with connectors to the pins added. The microcontroller is a small computer on a single chip.

Here is the MCU device summary from the spec sheet:

"The Atmel® ATmega324P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega324P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed. The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATmega324P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, two serial programmable USARTs , one byte-oriented 2-wire Serial Interface (I2C), a 8channel 10-bit ADC with optional differential input stage with programmable gain, a programmable Watchdog Timer with internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run."

This MCU chip is U9 shown in yellow with 40 pins. Most of the pins are not used. The pins usually have multiple possible uses. The 32 pins on the right can be used as 4 separate 8-bit input/output ports (ports A-D). For this application we are only using port A to output the 4-bit luma and 4-bit chroma

values and bit zero (B0=LSB) of port B for the sync signal. The program to generate the color test pattern is listed in Appendix A. Most of the code is all the (224 different) sets of 3 instructions to output each individual pixel on a line. I could not think of a way to do this in a loop and still only use 3 cycles per pixel. The whole program (app) is just a big loop that changes the values on the port A and B pins to the correct levels for the correct amount of time.

Table 1 lists each of the video signal components, how long they are, and the output values of the sync, luma, and chroma (only specified during the horizontal sync period). Here is a link to the NTSC specification defining the signals: https://antiqueradio.org/art/NTSC%20Signal%20Specifications.pdf.

Table 1. Video Signal Timing Summary

Signal	Nominal Width (us)	Width (MCU Clocks)	Clocks Used	Width Used (us)	Sync	Luma	Chroma	Comments
Front Porch	1.27	18.18386	19	1.327	H	1	0	240 Visible Vertical Lines (H sync pulse periods)
H Sync	4.763	68.196634	69	4.81911	L	0	0	Negative polarity (pulse is inverted)
Breezeway	0.38	5.44084	6	0.41905	H	1	0	
Color Burst	2.7	38.6586	39	2.72384		1	1	
Breezeway Back	1.27	18.18386	19	1.327		1	0	
H Blanking Left (border)	3	42.954	43	3.00321				Note: adjust to center display
H Blanking Right (border)	3	42.954	43	3.00321				Note after active data
Subtotal	16.383	234.571794	238	16.6224				
Active Data	46.9339293	672	672	46.9339				Use 224 horizontal pixels(16 colors x 14 intensities)
Total	63.3169293	906.571794	910	63.5564				
Rear (Top) Blanking								6 Lines as above with zero luma, no color burst
EQ Gap	2.54	36.36772	37	2.58416				
EQ Width	29.2	418.0856	419	29.2639				Positive Polarity
EQ Total	31.74			31.848				6 EQ pulses at just above blanking level

V Sync Gap	27.3	390.8814	391	27.3083				6 Vsync Pulses
V Sync Pulse	4.45	63.7151	64	4.4699				Positive Polarity
V Sync Total	31.75	454.5965	455	31.7782				
EQ Gap	2.54	36.36772	37	2.58416				6 EQ pulses
EQ Width	29.2	418.0856	419	29.2639				
EQ Total	31.74			31.848				
Front (Bottom) Blanking								5 Lines as above with zero luma, no color burst
Repeat from top								

In this implementation there are 262 horizontal lines of which 20 are not visible and used for vertical sync. One standard full TV picture consists of 525 lines. This is two sets of 262.5 lines. The half-line (0.5) is to vertically offset the two sets by ½ of a line. This is called interleaving. The sets of lines are interleaved, drawn sequentially. The full set of lines is drawn at a rate of 30 times a second. Note that the circuit test code used here does not bother to interleave 2 different sets of lines because all the lines are the same.

Physical Construction of the Circuit

I built the circuit using prototyping boards (breadboards). This circuit fits on two 63-hole set breadboards. Each hole set has 2 sets of 5 holes that are connected and can be used to insert components or wires. The connected holes are labeled A-E and F-J. There are many integrated circuit (IC) packaging types. The old package type used here is called Dual In-line Package (DIP). This package type can be easily inserted into the breadboard holes. Wires are inserted into the appropriate holes to implement the schematic diagram connections. Note it is getting harder to find components in DIP packages because most new devices have too many inputs/outputs and are too small to fit in this format. Also, as speeds of the signals increase, the wire lengths need to be kept shorter. So breadboarding of this type will not work for everything.

KiCad has a circuit board layout tool that can be used to generate the data needed to have a custom circuit card made. This is surprisingly inexpensive to get from China (on the order of $1 per board plus shipping in quantity of 10 small boards).

For practice, I made a printed circuit board design in KiCad shown in Figure 3. It can be challenging to route all the connections without any crossing. I used 4 layers in this example board design (not all the connections are visible). KiCad makes it pretty easy to convert the schematic into a printed circuit board (PCB) design. KiCad also lets you show a 3D rendering of what the board will look like.

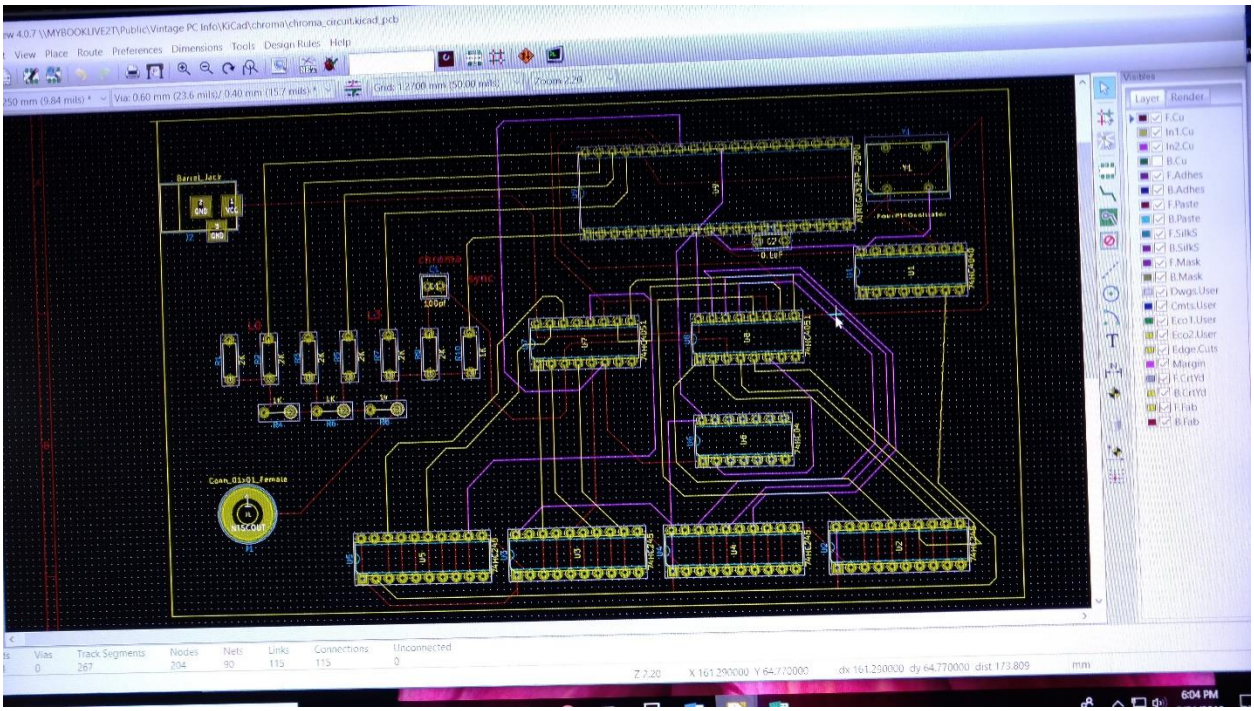


Figure 3. KiCad Example Printed Circuit Board (PCB) Design

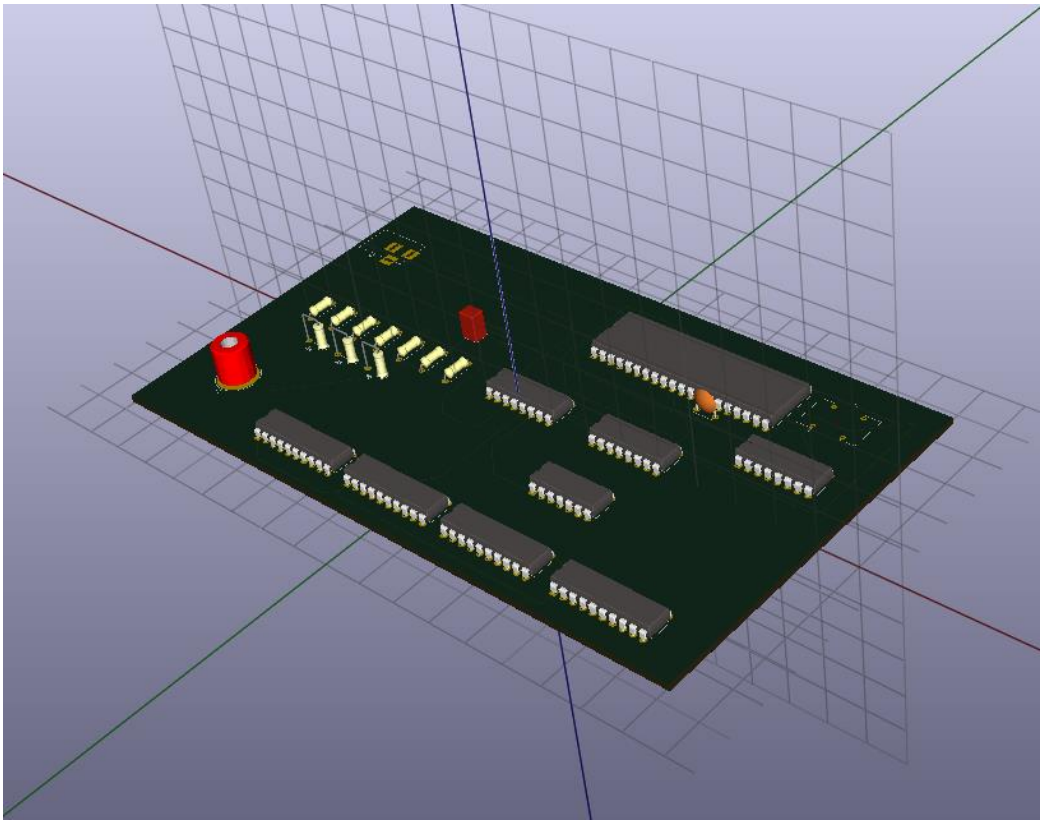


Figure 4. 3D Example Board Rendering

The breadboard implementation is shown in Figure 5. Note that the layout is different than the PCB design shown in Figure 3. The MCU is the big 40-pin IC. The four ICs with the blue wires over them are the tapped delay line. The green and yellow wires coming from the MCU are the port A and port B data controlling the luma, chroma, and sync. Ground (GND=0 volt reference) wires are light blue and +5V power wires (Vcc) are red. The oscillator is the silver box.

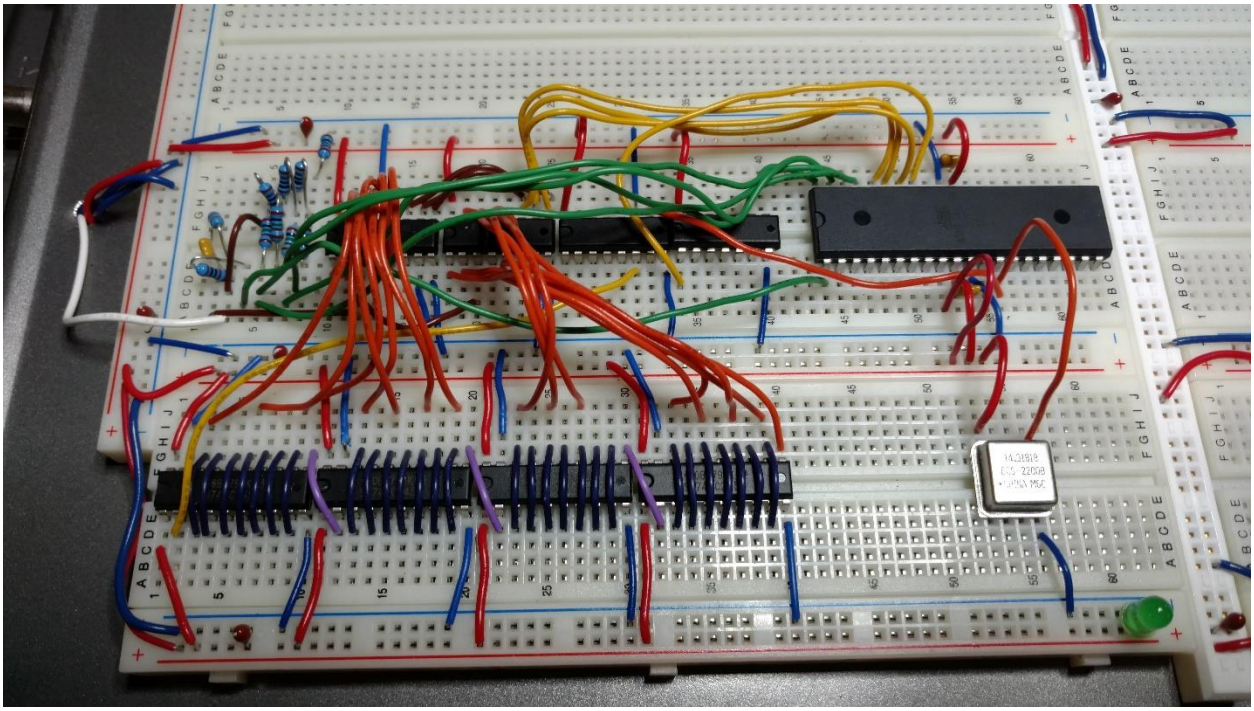


Figure 5. Video Circuit Breadboard

The video test pattern output is shown in Figure 6. There are 16 vertical bars with 15 colors (chroma taps) plus a bar of gray scale (bar on the left, black and white, no chroma, just luma). Each bar has 14 different luma values, from 2 to 15.

Figure 6. Video Output Test Signal

An oscilloscope (scope) is useful for debugging circuits. Figure 7 shows an example scope output of the vertical sync time period. The sync signal is shown in yellow and the luma signal is shown in pink. Note that the horizontal sync pulses are inverted. This means the pulse goes to a lower voltage from a high voltage value. Horizontal sync pulses can be seen in the right part of the yellow output. The vertical sync pulses are not inverted and the six vertical sync pulses are seen in the center yellow. A luma output ramping through the 14 levels can be seen in pink at the lower left.

Now that I look at it, I see a minor problem (or at least an inconsistency) with the sync output shown in Figure 7. For the purposes of this experiment, it does not make any difference. Do you know what it is? It could be fixed by a slight change in the MCU code listed in Appendix A.

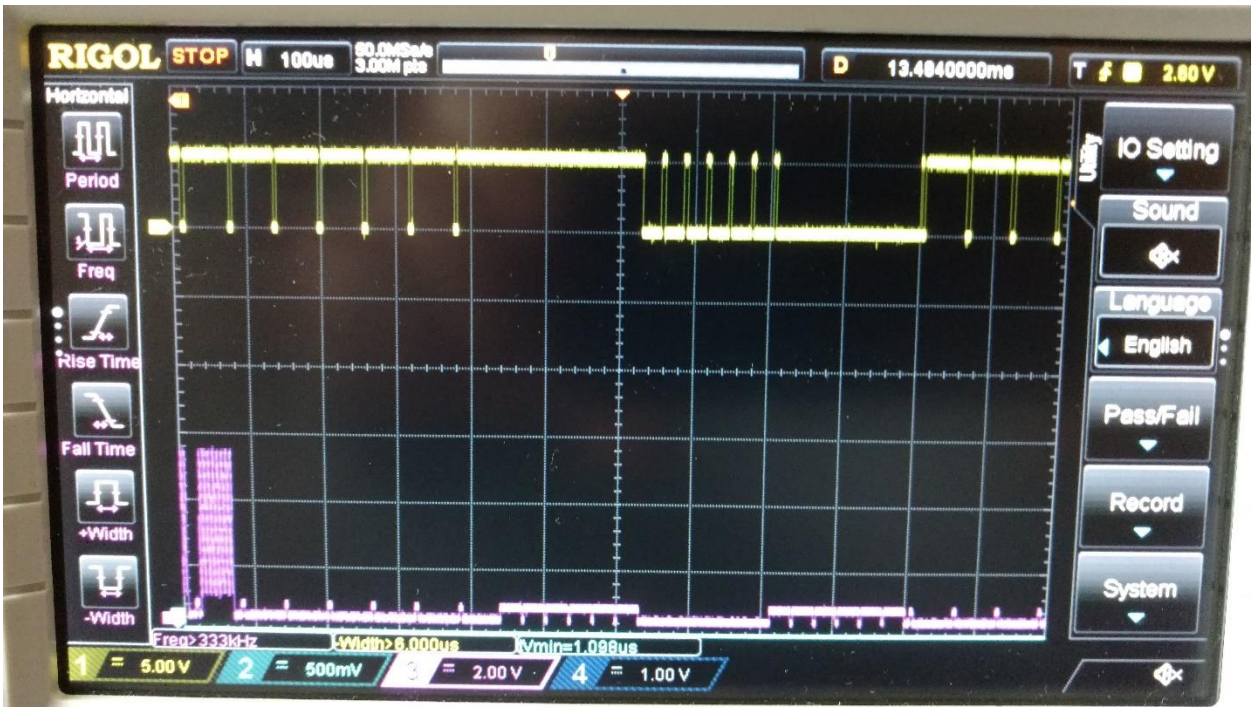


Figure 7. Example Oscilloscope Output

Appendix A. MCU Assembly Language Code for Video Test Pattern Generation

```
;
; VideoTest2.asm
;
; Created: 2/14/2018 1:06:25 PM
; Author : john
;
;
********************************************************************************
*******************************
; ***** VideoTest2 SYNC AND COLOR TEST
;
********************************************************************************
*******************************
; ------------------------------------------------------------------------
----------------------------
; HORIZONTAL MODE MICROSECONDS CYCLES ADJUSTED NOTES
; ------------------------------------------------------------------------
----------------------------
; FRONT PORCH 1.500 21.477 22
; SYNC PULSE 4.700 67.295 67
; BREEZEWAY 0.600 8.590 8
; COLOR BURST 2.500 35.795 36 9 CYCLES @ 3.579 MHZ
; BACK PORCH 1.600 22.909 23
; LEFT BORDER 2.833 40.536 41
; ACTIVE PIXELS 46.934 672.000 672 672/3 = 224 PIXELS
; RIGHT BORDER 2.833 40.563 41
; TOTAL HORIZONTAL 63.556 910.006 910
; ------------------------------------------------------------------------
----------------------------
; HORIZONTAL MODE LINES SYNC BURST BLANKING
; ------------------------------------------------------------------------
---------------------------
; TOP BORDER 28 NEGATIVE ACTIVE OFF
; ACTIVE LINES 200 NEGATIVE ACTIVE OFF
; BOTTOM BORDER 28 NEGATIVE ACTIVE OFF
; VERTICAL SYNC 6 POSITIVE NONE ON
; TOTAL VERTICAL 262
;
********************************************************************************
*******************************
; ***** PORT AND REGISTER INITIALIZATION
;
********************************************************************************
*******************************
.INCLUDE "M324pdef.inc"
.org 0
start:
; setting bit defines as output
; PORT A
sbi ddra,0 ; CHROMA BIT 0 sbi=set bit in I/O register
```

```
sbi ddra,1 ; CHROMA BIT 1
sbi ddra,2 ; CHROMA BIT 2
sbi ddra,3 ; CHROMA BIT 3
sbi ddra,4 ; LUMA BIT 0
sbi ddra,5 ; LUMA BIT 1
sbi ddra,6 ; LUMA BIT 2
sbi ddra,7 ; LUMA BIT 3
; PORT B
sbi ddrb,0 ; COMPOSITE SYNC
.def t1 = r16 ; TEMP REGISTER
.def t2 = r17 ; TEMP REGISTER
.def lc1 = r18 ; LINE COUNTER 1
.def lc2 = r19 ; LINE COUNTER 2
.def t3 = r20; TEMP REGISTER 3
;.def syp = r20 ; SYNC POLARITY
.def act = r21 ; BLANKING FLAG
; STACK POINTER
ldi t1,low(ramend) ; load immediate ldi Rd,k Rd <- k
out spl,t1 ; Output port out P,Rr,  P <- Rr
ldi t1,high(ramend)
out sph,t1
clr lc1
;disable interrupts
cli ; Clear interrupts
;
**********************************************************************
*********************************
; ***** VIDEO FRAME MAIN LOOP = 910 CYCLES TOTAL
;
**********************************************************************
*********************************
main:
; FRONT PORCH TIME = 19 CYCLES
ldi t2,6 ;1
fp:
dec t2 ;1
brne fp ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; horizontal sync on (low)
; SYNC PULSE TIME = 69 CYCLES
cbi portb,0 ;2 1
ldi t2,23 ;1
hsp:
dec t2 ;1
brne hsp ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; BREEZEWAY TIME = 6 CYCLES
; horizontal sync off
sbi portb,0 ;2
nop ;1
nop ;1
nop ;1
nop ;1
; turn on color burst during back porch
```

```
ldi t1,17 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
; COLOR BURST TIME = 39 CYCLES
ldi t2,11 ;1  3 cycles per loop with branch
cb:
dec t2 ;1
brne cb ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop ;1
nop ;1
nop ;1
; Breezeway back TIME = 19 CYCLES
ldi t1,1 ;1 burst off, luma to 1
out porta,t1 ;1
ldi t2,5 ;1  3 cycles per loop with branch
bb:
dec t2 ;1
brne bb ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop; 1
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
; LEFT BORDER TIME = 43 CYCLES
ldi t2,14 ;1  3 cycles per loop with branch
lb:
dec t2 ;1
brne lb ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; ACTIVE PIXEL TIME = 672 CYCLES
; want 14 intensities luma(L) = 2-15 and 16 colors chroma 0-15
ldi t1,2; 1
out porta,t1 ;1 L2, C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C0
```

```
nop;
inc t1 ; 1
out porta,t1 ;1 L10,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C0
nop; 1
inc t1 ; 1
out porta,t1 ;1 L14,C0
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C0
nop ;1
ldi t1,18; 1
out porta,t1 ;1 L2,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C1
nop ; 1
inc t1 ; 1
out porta,t1 ;1 L10,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C1
nop ;1
```

```
inc t1 ; 1
out porta,t1 ;1 L13,C1
nop ; 1
inc t1 ; 1
out porta,t1 ;1 L14,C1
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C1
nop ;1
ldi t1,34; 1
out porta,t1 ;1 L2,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C2
nop; 1
inc t1 ; 1
out porta,t1 ;1 L10,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C2
nop; 1
inc t1 ; 1
out porta,t1 ;1 L14,C2
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C2
nop ;1
ldi t1,50; 1
```

```
out porta,t1 ;1 L2,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C3
nop ; 1
inc t1 ; 1
out porta,t1 ;1 L10,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C3
nop; 1
inc t1 ; 1
out porta,t1 ;1 L14,C3
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C3
nop ;1
ldi t1,66; 1
out porta,t1 ;1 L2,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C4
```

```
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C4
nop ; 1
inc t1 ; 1
out porta,t1 ;1 L10,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C4
nop; 1
inc t1 ; 1
out porta,t1 ;1 L14,C4
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C4
nop ;1
ldi t1,82; 1
out porta,t1 ;1 L2,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C5
nop ;1
```

```
inc t1 ; 1
out porta,t1 ;1 L9,C5
nop ; 1
inc t1 ; 1
out porta,t1 ;1 L10,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L12,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C5
nop; 1
inc t1 ; 1
out porta,t1 ;1 L14,C5
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C5
nop ;1
ldi t1,98; 1
out porta,t1 ;1 L2,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C6
nop; 1
inc t1 ; 1
out porta,t1 ;1 L11,C6
nop ;1
inc t1 ; 1
```

```
out porta,t1 ;1 L12,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C6
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C6
nop; 1
inc t1 ; 1
out porta,t1 ;1 L15,C6
nop ;1
ldi t1,114; 1
out porta,t1 ;1 L2,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C7
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C7
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C7
```

```
nop; 1
ldi t1,130; 1
out porta,t1 ;1 L2,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C8
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C8
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C8
nop; 1
ldi t1,146; 1
out porta,t1 ;1 L2,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C9
nop ;1
```

```
inc t1 ; 1
out porta,t1 ;1 L5,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C9
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C9
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C9
nop; 1
ldi t1,162; 1
out porta,t1 ;1 L2,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C10
nop ;1
inc t1 ; 1
```

```
out porta,t1 ;1 L8,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C10
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C10
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C10
nop; 1
ldi t1,178; 1
out porta,t1 ;1 L2,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C11
```

```
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C11
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C11
nop; 1
ldi t1,194; 1
out porta,t1 ;1 L2,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C12
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C12
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C12
nop ;1
```

```
inc t1 ; 1
out porta,t1 ;1 L15,C12
nop; 1
ldi t1,210; 1
out porta,t1 ;1 L2,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C13
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C13
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C13
nop; 1
ldi t1,226; 1
out porta,t1 ;1 L2,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C14
nop ;1
inc t1 ; 1
```

```
out porta,t1 ;1 L4,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C14
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C14
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C14
nop; 1
ldi t1,242; 1
out porta,t1 ;1 L2,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L3,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L4,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L5,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L6,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L7,C15
```

```
nop ;1
inc t1 ; 1
out porta,t1 ;1 L8,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L9,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L10,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L11,C15
nop; 1
inc t1 ; 1
out porta,t1 ;1 L12,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L13,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L14,C15
nop ;1
inc t1 ; 1
out porta,t1 ;1 L15,C15
nop; 1
;nop; 1
wdr ; Watchdog reset
; BLANKING Right PIXEL TIME = 43 CYCLES
ldi t1,0 ;1
out porta,t1 ;1
ldi t1,14 ;1
blk22:
dec t1 ;1
brne blk22 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop ;1
; Repeat 239 more times for visible scan lines
; line counter
inc lc1 ;1
cpi lc1,239
breq done
jmp main
done:
;done with active data lines
; do blanked and vertical sync lines
ldi t3,6 ; counter for number of blanking lines
rearb: ; rear (bottom) blanking - 6 lines, no color burst
; FRONT PORCH TIME = 19 CYCLES
ldi t2,6 ;1
fp2:
dec t2 ;1
brne fp2 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; horizontal sync on (low)
```

```
; SYNC PULSE TIME = 69 CYCLES
cbi portb,0 ;2
ldi t2,23 ;1
hsp2:
dec t2 ;1
brne hsp2 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; BREEZEWAY TIME = 6 CYCLES
; horizontal sync off
sbi portb,0 ;2
nop ;1
nop ;1
nop ;1
nop ;1
; turn off color burst during back porch
ldi t1,1 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
; COLOR BURST TIME = 39 CYCLES
ldi t2,11 ;1  3 cycles per loop with branch
cb2:
dec t2 ;1
brne cb2 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop ;1
nop ;1
nop ;1
; Breezeway back TIME = 19 CYCLES
ldi t1,1 ;1 burst off, luma to 1
out porta,t1 ;1
ldi t2,5 ;1  3 cycles per loop with branch
bb2:
dec t2 ;1
brne bb2 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop; 1
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
; LEFT BORDER TIME = 43 CYCLES
ldi t2,14 ;1  3 cycles per loop with branch
lb4:
dec t2 ;1
brne lb4 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1

; ACTIVE PIXEL TIME (blanked) = 672 CYCLES
ldi t2,223 ;1  3 cycles per loop with branch
ab:
dec t2 ;1
brne ab ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
dec t3; 1
brne rearb;  2/1

; Equalizer Gap 37 cycles
;next is 6 equalizer pulses with luma just above blanking
```

```
ldi t3,6; 1
ep:
;cbi portb,0 ;2
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,11 ;1
epg:
dec t2 ;1
brne epg ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop;

; Equalizer Pulse 419 cycles
ldi t1,1 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,139 ;1
epp:
dec t2 ;1
brne epp ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop;
nop;
dec t3
brne ep;

; Vertical sync Gap 391 cycles
; not trying to use luma for vertical sync luma=1 for pulse, =0 for
pulse off
ldi t3,6; 1
nop ; 1
vsp:
cbi portb,0 ; 2  set to zero, positive polarity
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,118 ;1
vspg:
dec t2 ;1
brne vspg ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1

; Vertical Sync Pulse 64 cycles
sbi portb,0 ; 2
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,20 ;1
vspp:
dec t2 ;1
brne vspp ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1

cbi portb,0; 2
dec t3; 1
```

```
brne vsp; 2/1
nop; 1

; Equalizer Gap 37 cycles
;next is 6 equalizer pulses with luma just above blanking
nop
ldi t3,6; 1
ep3:
;cbi portb,0 ;2
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,10 ;1
epg3:
dec t2 ;1
brne epg3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop;

; Equalizer Pulse 419 cycles
ldi t1,1 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
ldi t2,139 ;1
epp3:
dec t2 ;1
brne epp3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop;
nop;

dec t3 ; 1
brne ep3; 2/1

; FRONT PORCH TIME = 19 CYCLES
; Last is front(bottom) blanking of 5 lines
; do blanked and vertical sync lines
ldi t3,5 ; 1 counter for number of blanking lines
frontb: ; front (bottom) blanking - 5 lines, no color burst
sbi portb,0 ;2
ldi t1,0 ;1
out porta,t1 ;1
ldi t2,5 ;1
fp3:
dec t2 ;1
brne fp3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; horizontal sync on (low)
; SYNC PULSE TIME = 69 CYCLES
cbi portb,0 ;2 1
ldi t2,23 ;1
hsp3:
dec t2 ;1
brne hsp3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; BREEZEWAY TIME = 6 CYCLES
```

```
; horizontal sync off
sbi portb,0 ;2
nop ;1
nop ;1
nop ;1
nop ;1
; turn off color burst during back porch
ldi t1,1 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1
; COLOR BURST TIME = 39 CYCLES
ldi t2,11 ;1  3 cycles per loop with branch
cb3:
dec t2 ;1
brne cb3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop ;1
nop ;1
; nop ;1
; Breezeway back TIME = 19 CYCLES
ldi t1,1 ;1 burst off, luma to 1
out porta,t1 ;1
ldi t2,5 ;1  3 cycles per loop with branch
bb3:
dec t2 ;1
brne bb3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
nop; 1
ldi t1,0 ;1  0-15, color burst is off (C3-C0=0 selects gnd out of
mux),
out porta,t1 ;1

; LEFT BORDER TIME = 43 CYCLES
ldi t2,14 ;1  3 cycles per loop with branch
lb3:
dec t2 ;1
brne lb3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1
; ACTIVE PIXEL TIME (blanked) = 672 CYCLES
ldi t2,223 ;1  3 cycles per loop with branch
ab3:
dec t2 ;1
brne ab3 ;2/1  branch if not equal: brne k  if (Z=0) PC <- PC + k +1

dec t3; 1
brne frontb;  2/1
;get ready to start over
clr lc1 ; 1 clear line counter
jmp main ;2
```

Appendix B - Logic Families Overview

TTL -- Transistor / Transistor Logic	TTL chips require a fairly narrow range of supply voltage -- 5 volts, +/- 0.5V. For TTL circuits, a logic "1" is usually defined as a signal of about 2.4 V, while logic "0" is about 0.5 V above ground; any signal between those values is undefined. Also, note that all unused inputs should be tied to ground. Bipolar TTL logic comes in many flavors (usually numbered in the "74XX" format, occasionally in "54XX" format) including 74LS/ALS types. These all require 1000s of times more power than TTL-compatible CMOS logic (like the 74HCT type, see below). Note that the 74HCT* ICs have the same logic thresholds as other TTL chips, allowing them used in a mixed circuits of TTL and HCT logic. Here are some TTL "subfamilies": 74LXX -- low-power TTL (1/10 the speed, 1/10 the power of "regular" TTL) 74HXX -- high-speed TTL (twice as fast, twice as much power) 74SXX -- Schottky TTL (for high-frequency uses) 74LSXX -- combination of low-power & Schottky, same speed as regular TTL, but at 1/5 the power consumption
CMOS -- Complementary Metal Oxide Silicon	CMOS ICs use much less power than TTL, plus they are fairly forgiving of "slop" in input voltage -- they're happy with anything between 3 (some, like 1381s, go lower) and 12 volts. CMOS, though, is much more susceptible to damage from static electricity (so get that grounding strap out!). CMOS comes in two flavors, with corresponding numbering schemes: "CD40XX" metal gate CMOS, and "74CXX" silicon gate CMOS (note that this second subfamily borrows its numbering scheme from TTL ICs). **Metal gate CMOS** (CD40XX)has a rated working voltage of 3 - 15 V but can be used down to 2V. **Silicon gate CMOS** (74CXX) logic has a working voltage range of 2 -6V but can be used to less than 1V. For microwatt power applications you want to use the lowest possible voltage.

Numbering systems and IC "sub-families"

One of the more common numbering systems used for CMOS ICs is borrowed from TTL -- that of 74xxxnnn -- here, "xxx" is AC, ACT, HC, or HCT (subfamily); and "nnn" is the specific chip ID.

So, basically, any given chip will come in a whole mess of variations (one per subfamily); if it's any consolation, we only need to be concerned with 4. For any given chip, you'll need to consider the plusses and minuses of two comparisons -- AC vs HC subfamilies, and AC / HC vs ACT / HCT (essentially, CMOS vs. TTL input levels).

AC vs HC
AC / ACT stands for Advanced CMOS Logic (ACL for short). HC / HCT stands for High-speed CMOS Logic (HCL).

The AC and ACT subfamilies are faster than the HC and HCT subfamilies, and draw some more power in some circuits. All chips in the AC* subfamily have lower output resistance than HC* and can sink and source 24 mA at logic levels and up 70 mA (typ) per gate for motor loads. As a result AC* gates can handle more than twice the current of HC* gates (50 - 70 mA vs. 24 mA). Note, though, that while most HC* chips have a 25 mA limit, the HC* driver chips such as the 74HC240 and the 74HC245 (i.e., buffers) can handle 35 mA per device, and a maximum of 75 mA per chip.

The AC & ACT families also draw about twice as much current as the HC & HCT chips (but we speaking here of microamps, so it's usually not a huge deal).

It is occasionally possible to find (high quality) motors that you can drive directly from an HC chip. For intermittent operation, such as you get with a quadcore, you COULD drive very efficient (i.e., very low-current) motors directly. You would definitely need a capacitor (say 0.47 uF) across each motor to keep the noise under control.

CMOS vs. TTL inputs
The main difference between AC- and HC-chips vs. ACT- and HCT-chips is that AC / HC chips work with CMOS-level inputs, while ACT / HCT chips work from TTL-level inputs and outputs. AC / HC chips allow a relatively-wide supply-voltage range (2 - 6 V), and works with CMOS levels on input and output (i.e., doesn't like inputs or outputs to be ill-defined, prefers close to 0V for low and Vdd for high). ACT / HCT works only with a supply voltage of 5V +/- 10%, and works with TTL levels on input and output (a valid input is close to 0V for low, and ~2.4 - 5.0 V for high).