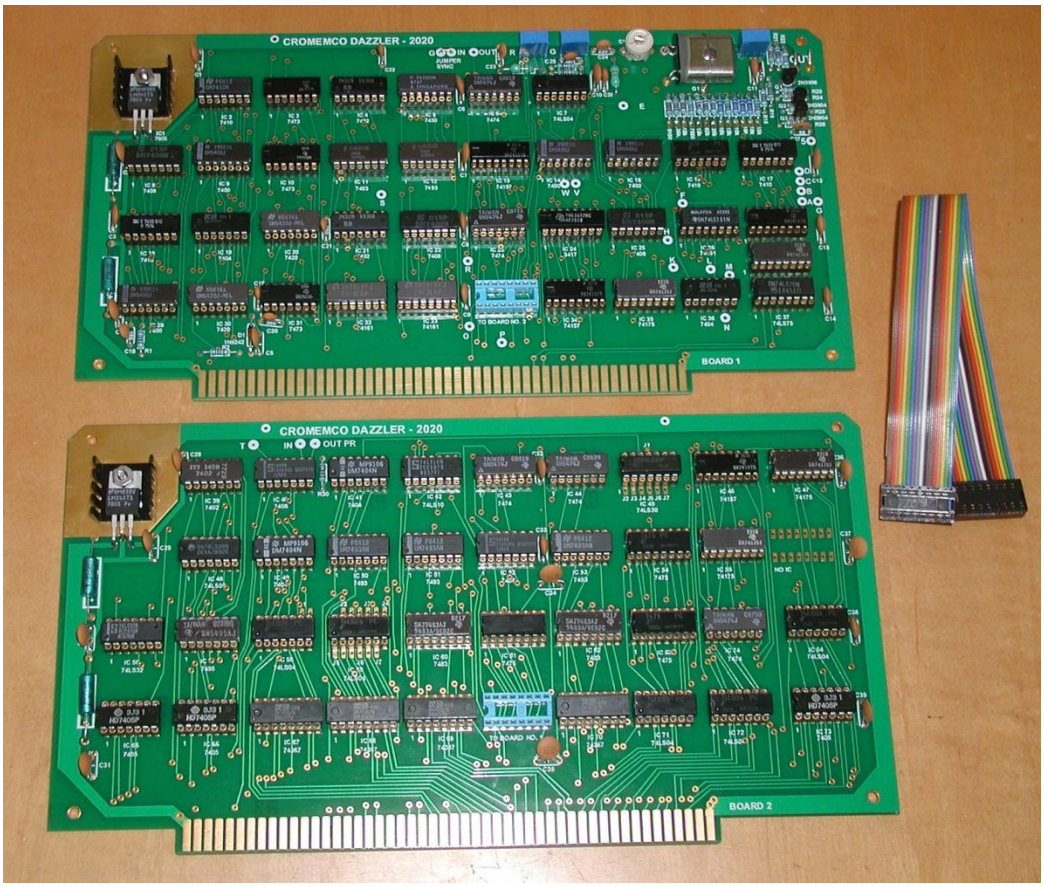
THE CROMEMCO DAZZLER, RE-BORN.

Dr. H. Holden. October 2020.



INTODUCTION:

Computer graphics were coming of age in the mid to late 1970's and efforts were being made to provide home computer enthusiasts with accessory cards which would give them graphics capability, typically to be used in early S-100 computers such as the Altair and others.

Matrox were on the front line then with monochrome graphics cards such as the ALT-256 and the ALT-512 (See Silicon Chip articles October and Nov 2020). Three Matrox

monochrome cards could be deployed to make a RGB color system, it was an expensive purchase.

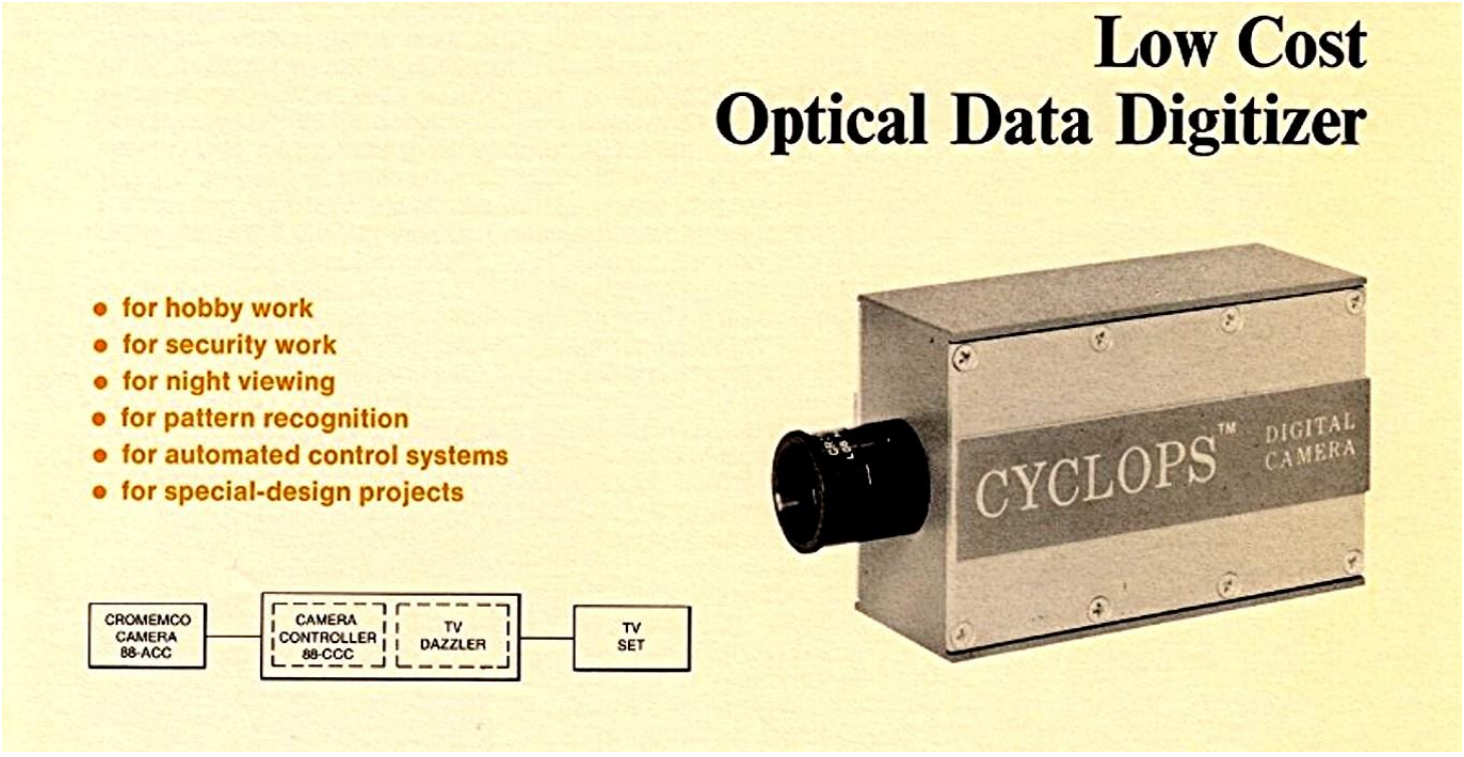
Other companies such as Godbout Electronics offered the “Spectrum” board by 1980, which by then was sophisticated enough to be both color and have on board “Video RAM” (Article on the Spectrum board is in progress).

Prior to this the Cromemco company offered the “DAZZLER” board set in 1976.

DAZZLER HISTORY:

In terms of the history of the development of computer graphics cards, by far and away the most interesting is the Cromemco Dazzler. It was the World’s first color graphics card for S-100 bus computers with an NTSC color composite video signal output.

Essentially the idea behind it was born in 1975 when Roger Melon and Harry Garland created the first solid state video camera. The idea was to use a MOS Dynamic RAM 1k x 1 bit IC, with its top cut off, to act as an optical sensor. They created the “Cyclops” solid state video camera. In addition they formed the company “Cromemco”. The name was derived from Crothers Memorial Hall, where they resided at Stanford University. A photo of the Cyclops camera:



The camera controller board put the camera's pixel data into general ram in the host computer. Then the Dazzler board could read that ram and create a standard (or close to standard) NTSC composite video signal to feed either an NTSC color video monitor, or a domestic TV set with an RF modulator assisting.

Soon the Dazzler board set became an entity of its own. It was presented as a project to build in Popular Electronics magazine in Feb 1976. It became so popular that Cromemco started making it, either in kit form, or selling it fully assembled and tested.

One application it covered was in the production of Color Graphics for Weather Systems and it found its way into the Television industry.

Unlike other graphics cards of the time, which had non-interlaced scanning, the composite video signal generated by the Dazzler was an interlaced scan, compatible with the NTSC color television system.

The Dazzler has no on board video RAM, instead it hijacks the host computer's system RAM for the job with a direct memory access (DMA). For it to work it required that the computers RAM was fast static RAM with an access time of 1 micro-second or faster. Dynamic RAM did not work because the refresh activities interrupt the proceedings.

The Dazzler came as two separate s-100 boards linked by a 16 way ribbon cable as shown in the photo above.

In total the two boards contain 72 IC's, most of which are common garden 74 or 74LS series TTL types. Except for one ***extremely rare IC***, a quad 64 bit shift register, the TMS3417, which apparently was rare even in the 1970's.

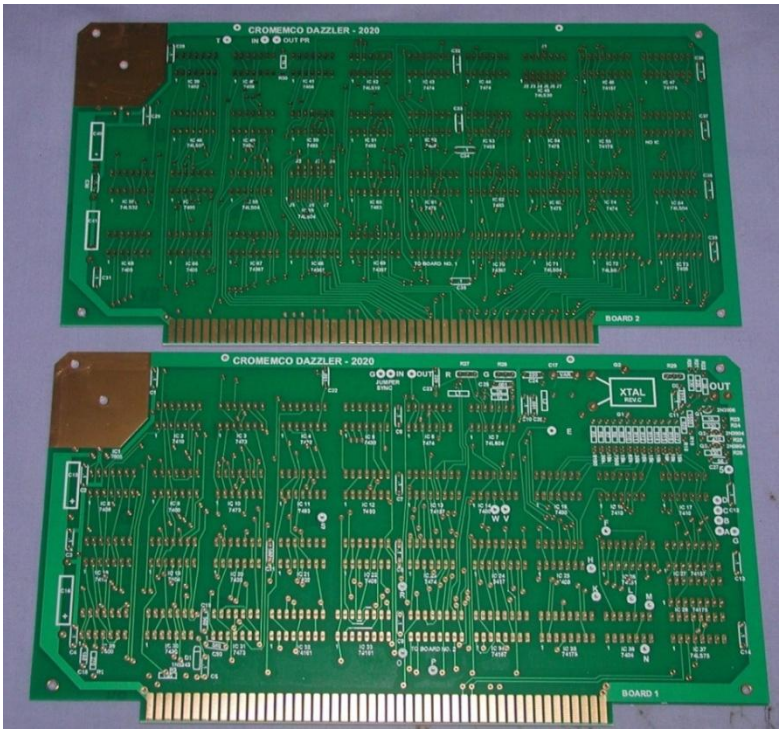
BUILDING THE DAZZLER:

Dazzler board sets are very hard to come by these days, so I realized if I wanted to test and analyse one I would have to make dead replicas of the pcb's.

Cromemco provided the pcb foil patterns in their manual for those who wanted to do this. The old photocopies I could find were not very clear in places. However after some months tracing over them in a drawing program I managed to make clear copies of the top and bottom track patterns of each board. After that I checked against the schematic to correct errors, which took a few late nights.

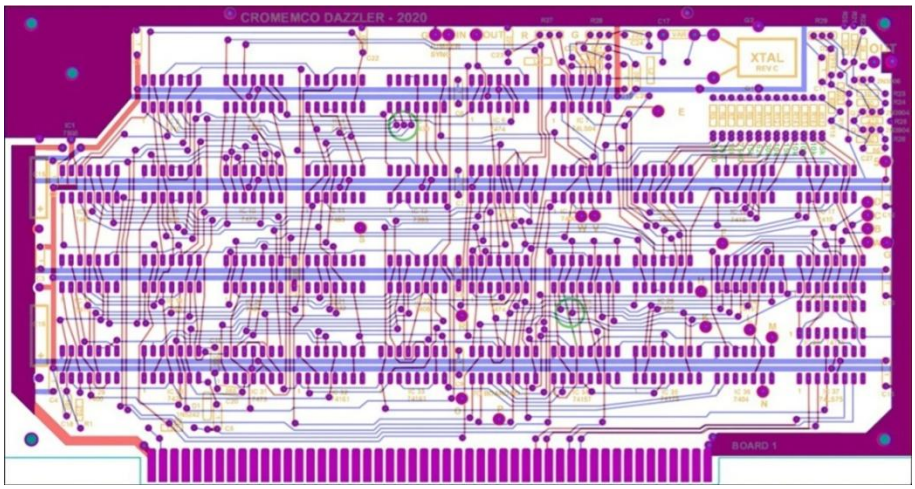
When this was done I sent the image files to LD Electronics (They advertise in Silicon Chip) and they were able to make very high quality pcb's with an exact track pattern replica to my drawings and with Gold plating. George from LD Electronics did an

absolutely terrific job on them, see photo below. These are likely better than the originals.



Cromemco Dazzler replica pcb's unpopulated.

The photo below shows some of the interesting artwork images that got generated along the way for one of the boards:



I acquired all of the components required as per the parts list for the kit in the Cromemco manual. And I sorted them into bags as the original kit had done.

Also I decided to acquire high quality gold plated machine pin sockets for the IC's, so I used NOS Augat all gold machine pin types that I imported from the USA.

It became apparent right away that there was one very difficult to get part. An IC which is a quadruple 64 bit shift register, the TMS3417NC, rated to a 5MHz clock frequency. The closest modern part I could find was the 74HCT7731. However, it has a different pin-out and I was not 100% sure if I used one, if it would work. And since, in a project like this, there could be a lot of scope for it not working initially, due to track errors on my part or initially faulty old stock IC's, it was a very big risk.

At least if I had an original Dazzler "up and running" in my computer, only after that, could I experiment with any substitute shift register IC's.

Another possible IC candidate, as a pin-pin substitute for the TMS3417, is the Fairchild F3342DC. This is a 2MHz rated part. Initially I was put off this. However, there was the Practical Electronics article from 1976 where the F3342DC IC was used in a Dazzler.

After I lot of searching a found a small number of TMS3417 IC's in Germany and a few F3342DC's in the USA, so I am well stocked for these now. When the Dazzler was operating I found that the clock frequency for this shift register is close to 1.8MHz, which explains why both the 2MHz and 5MHz rated shift registers work in the application.

Once the Dazzler was assembled I fitted it to my SOL-20 computer. Much to my surprise, considering all the steps involved in the pcb artwork and the large number of mainly vintage IC's, the Dazzler worked immediately. By that I mean that it responded normally to manipulating its registers and testing its modes and running a software package.

My SOL-20 computer has external 5.25" disk drives where I can run CP/M DOS. This has an assembler, so I was able to assemble the Kaleidoscope program. This was one of the most famous programs that ran with the Dazzler. It puts the Dazzler cards into a 2048k byte picture modes, which is a 64 x 64 pixel display or 4096 pixels total. In this display 4 bits of each byte control a pixel. Three of the bits code the RGB combination and one bit the intensity. So it requires 2048 bytes of the host computer's RAM to accommodate it. The Kaleidoscope program places the image in the computer's RAM starting at address 0200 hex and ending at 09FF hex.

When the Kaleidoscope program is running, it is quite something to observe. A link to a in internet video of it playing is here: www.youtube.com/watch?v=2tDbn1N8EWI

It is hypnotic and mesmerizing. The still frames only give an indication of it, but they are worth looking at, it is more varied than the still images can show, see below.

These stills were photographed directly off the face of the CRT. If the program is terminated (with a CPU reset) this resets the Dazzler hardware and switches the Dazzler off. The last image byte values remain in RAM, so if the Dazzler board is switched back on & into the same mode, the last image is seen there as a “still frame”.

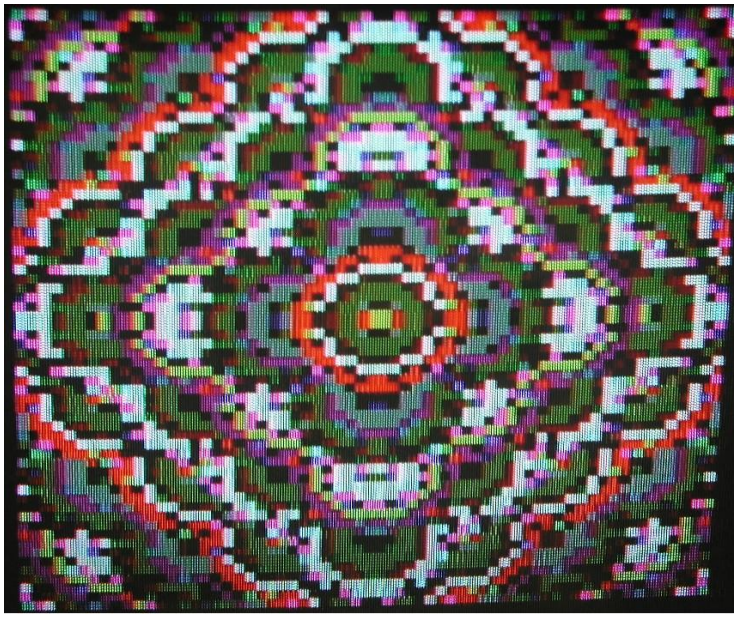
A short program of bytes switches the Dazzler on:

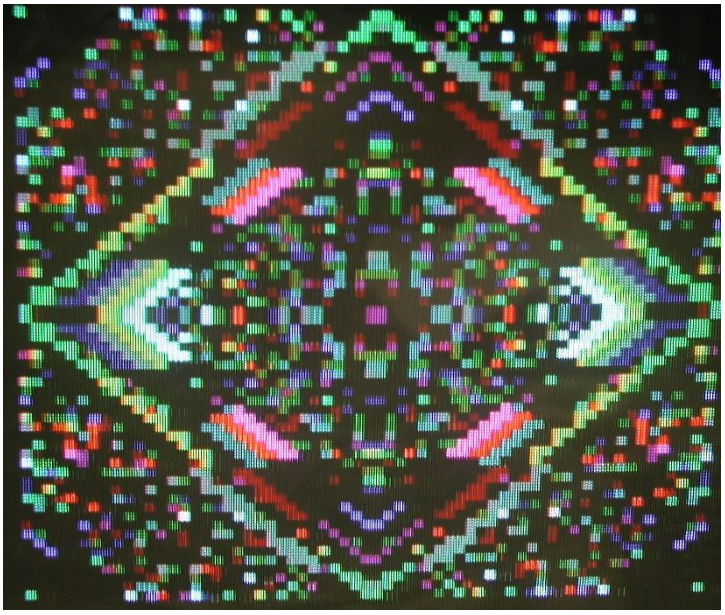
3E 81 D3 0E 3E 30 D3 0F C3 04 C0

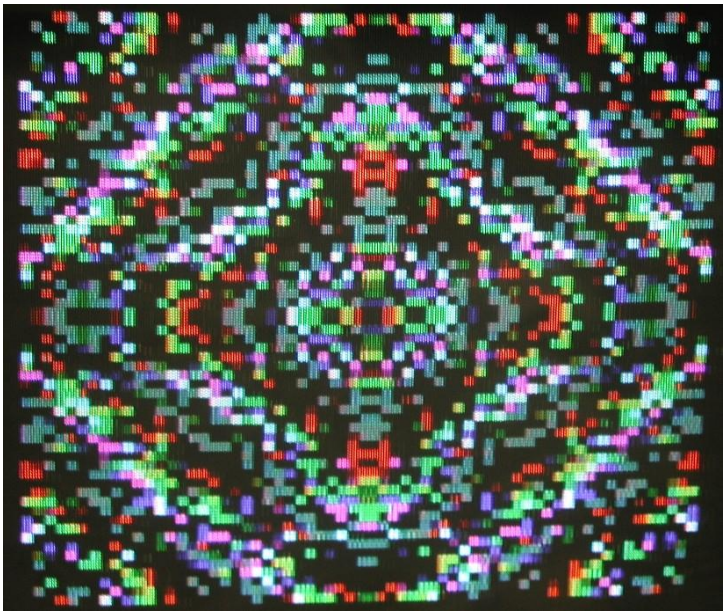
This is equivalent to the short assembly language program:

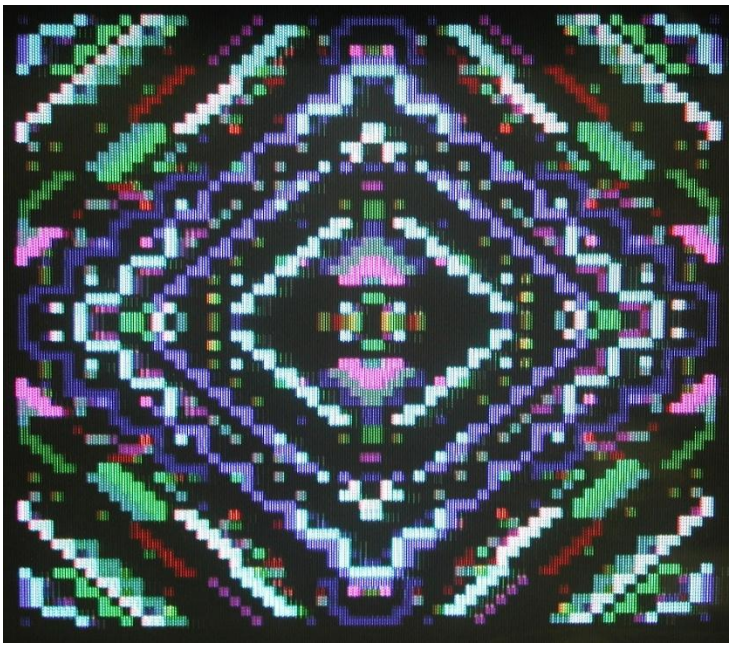
```
MVI   A,81H
OUT   0EH      ;  sends 81H to port 0E on Dazzler card (starts image at 0200H)
MVI   A,30H
OUT   0FH      ;  sends 30H to port 0F on Dazzler card (color mode 2k picture)
JMP   0C004H   ;  returns to the Sol’s operating system without a reset.
```

These are easily entered to memory say at 0100H in the Sol with the EN command and executed with the EX 0100 command.





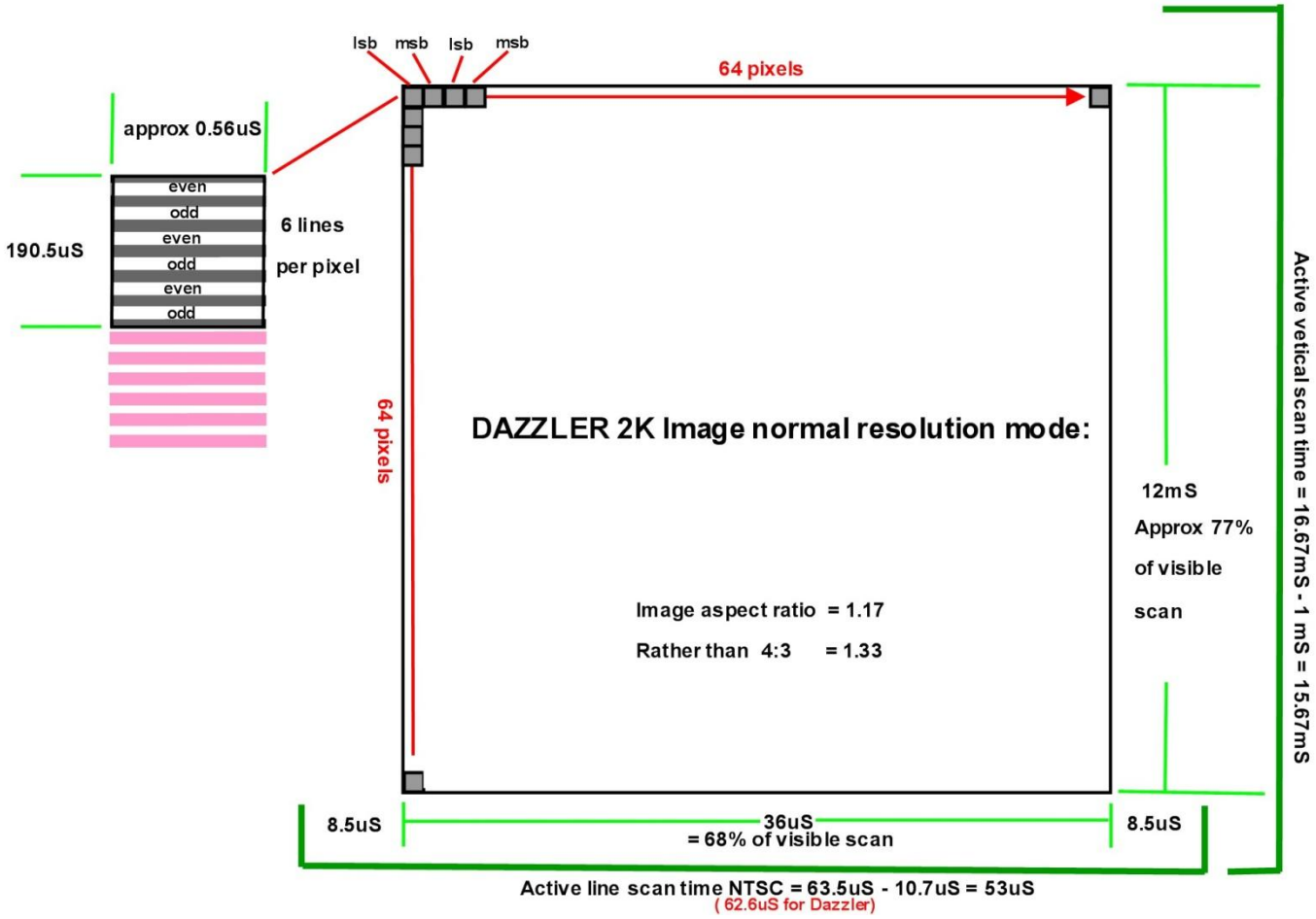




I think it helped that in this image mode, the image is divided into four 512 pixel blocks. So for the program, only one 512 pixel block is dynamically altered and the address rotated into the other 3 blocks to enable the symmetrical effect about centre, which results in the Kaleidoscope like effects.

The diagram below shows the space occupied on the face of the video monitor by the Dazzler’s image:

Space occupied on the screen of a TV or video monitor of the Dazzler's image:



Some curios about the Dazzler's video signal:

It is an interlaced scan format (as is NTSC video) however there are no equalizing pulses around the vertical sync pulse .On account of this the interlace is not perfect and examination shows there is slight line pairing of the scan lines of consecutive even and odd fields. This is only detectable though with a monochrome image in a monochrome monitor, it is much harder to see on a color monitor/TV unless it's a very large screen size.

The Dazzler also has a non standard horizontal line scan period. In the NTSC system it is usually around 63.5uS, for the Dazzler it is around 62.6uS. In addition, the Dazzler

uses a very wide burst gate pulse of around 4.7uS long, this lets through a wider than normal color burst, which as a consequence starts immediately after H Sync and there are more cycles of the color burst as a result.

However, the NTSC color decoders in TV sets usually don't protest with a longer color burst. In addition, the color burst appears on the vertical sync pulse when it is low, due to the way the pulses are combined.

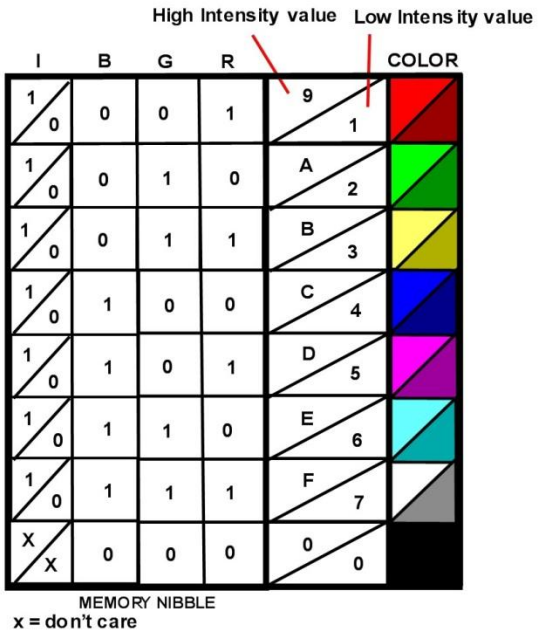
Only a percentage of the active line and active field time scan is used so there is quite a lot of space on the screen around the actual displayed pixel area. This would have allowed for any over-scan on the domestic TV set.

About 77% of the active scan time is used on the vertical and about 68% of the active H scanning time. So the image on the monitor's 4 x 3 screen (1.33 ratio) adopts about a 1.17 ratio, so the overall image (and each pixel) is still a little rectangular and not perfectly square.

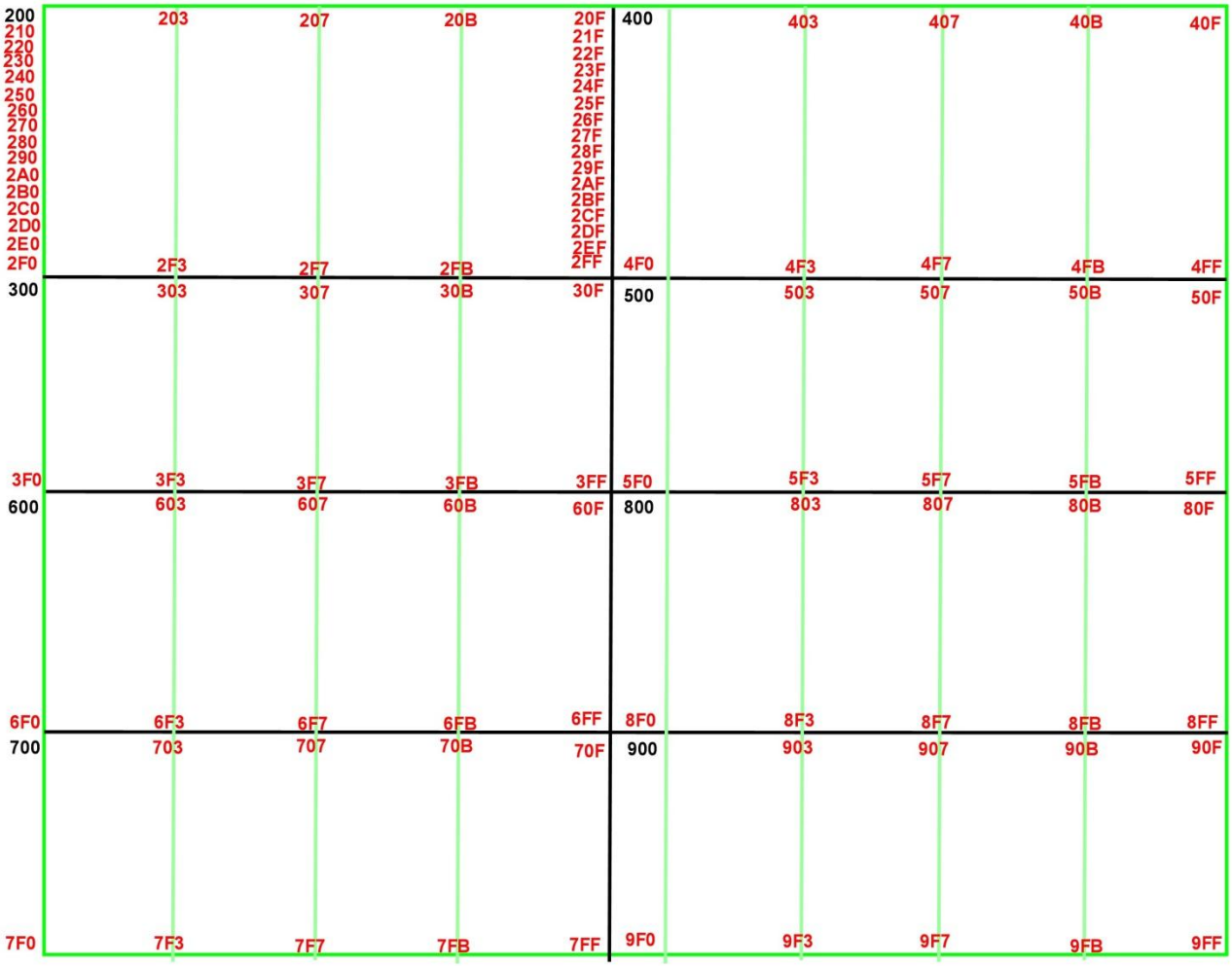
Dazzler Color Encoder:

To check the Dazzler's operation and help to correctly set the Dazzlers Red and Green "color carrier phase adjustments" I wrote a short assembly language program to generate an output that resembled a standard NTSC color test pattern. This enabled the best setting of these controls for the most accurate color rendition and white balance in the color output signal.

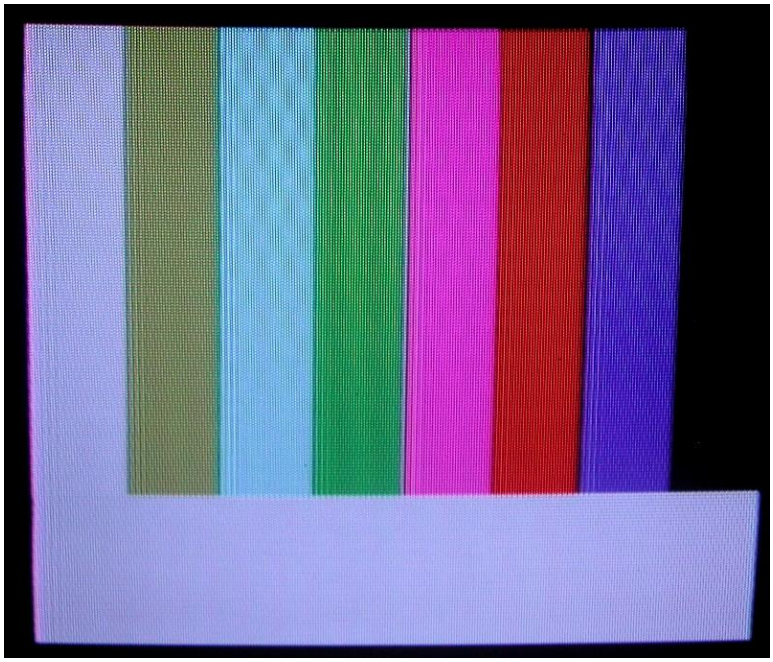
In the 2048 (2k) image mode each nibble (4 bits) of a memory byte controls the pixels color and its intensity. So initially I drew up the table below to help remind me of the byte values corresponding to each color for the high & low intensity values:



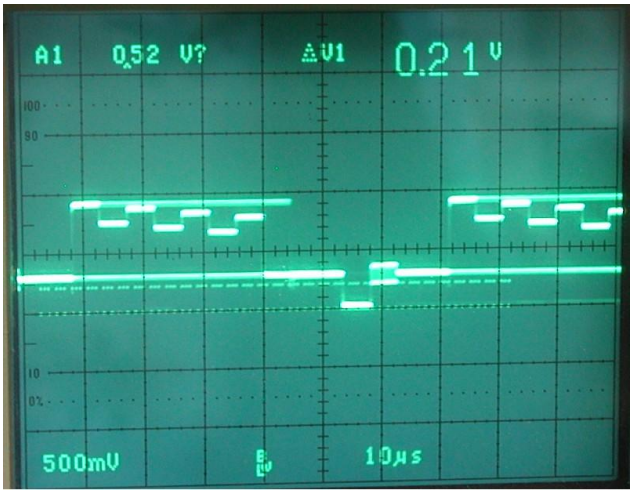
Also I made another table to help remind me of the addresses of the pixel locations when the picture is loaded at 0200H. It is not a progression as it has been divided into 4 blocks or quadrants:



I wrote what would be a standard NTSC test pattern into the memory and this was the result (with optimum adjustments of the R & G phase presets on board 1) with high intensity colors selected:

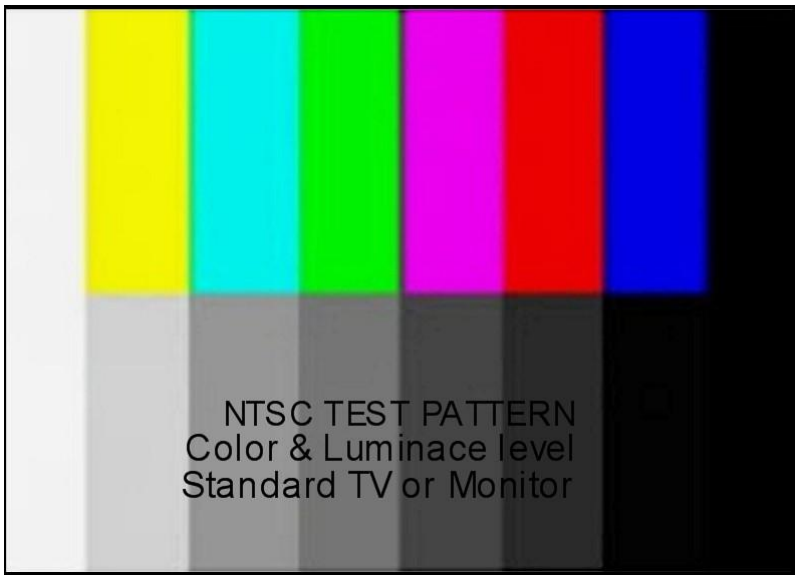


In theory, if the usual NTSC luminance (Y signal) weighting was used, when switched to monochrome (on the TV or monochrome mode on the Dazzler card) it should give a grey scale or descending order of luminance from left to right. However it did not. Cromemco has chosen a different arrangement. The scope recording below shows the levels of the steps in monochrome mode (also notice the wide burst is still there in monochrome mode):

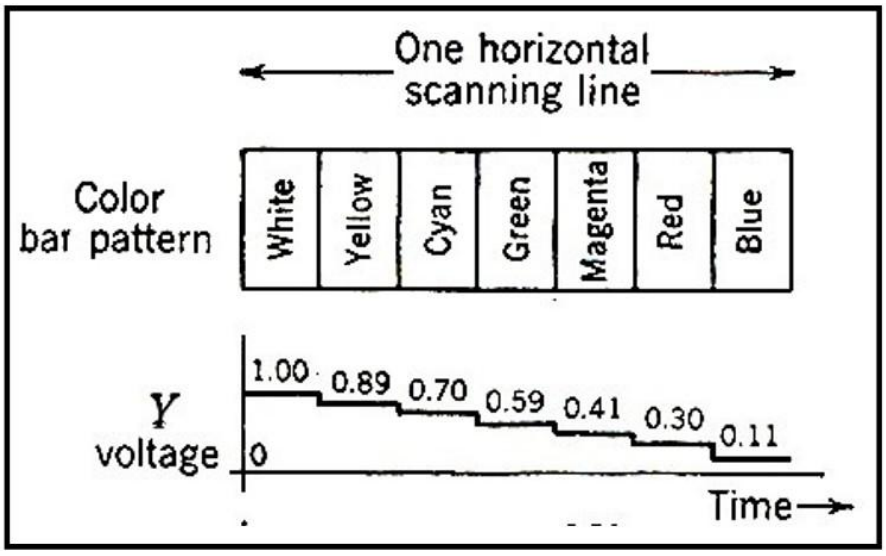


In the Cromemco system, the next step down from white is Cyan, then Magenta, Blue, Yellow, Green and finally Red.

By comparison, in standard NTSC the luminance steps are shown below:



This anomaly came about because of the relative proportions of R,G & B to create white in the Cromemco luminance resistor matrix was not normal. In NTSC the weighting is generally 0.30 Red, 0.59 Green and 0.11 Blue. The diagram below shows how the proportions of R,G & B create the overall luminance component Y in NTSC:

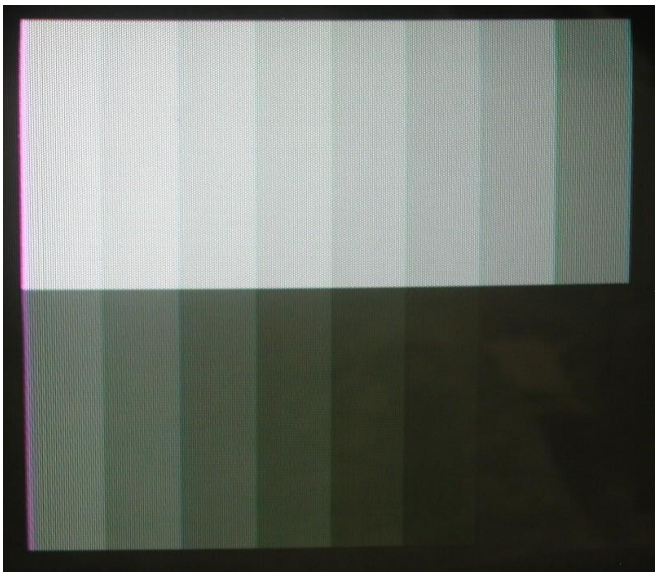


The interesting this here, is despite the luminance values being non standard for an NTSC system, it is hard to see the effect of it on a color image. The reason for this is

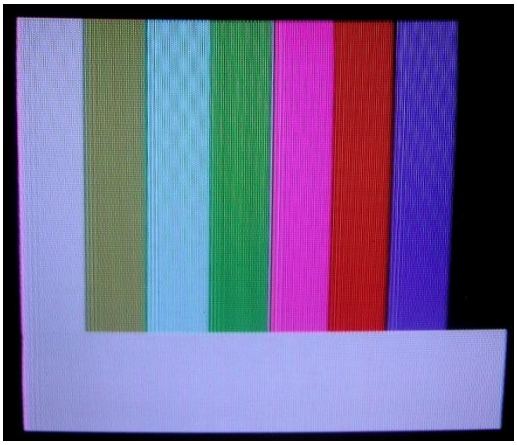
that the colors are very heavily saturated. The problem only shows up when the card is switched monochrome mode.

To investigate this further I programmed another test pattern, to put the colors in the luminance order that Cromemco did. I have drawn onto the photo the programming (memory byte value codes two consecutive pixels) that corresponds to the color and intensity selection. Notice also how the Blue looks just a tad purple, the reason for this explained below.

When switched to monochrome mode the grey scale is very respectable for this color order. And the magnitude of the grey level is proportional to the value of the nibble that codes the pixel. Obviously this is convenient for programming monochrome images:



A standard NTSC pattern looks like this, but as a result of the luminance mix used the grey scale is disordered:



As can be seen in the e Cromemco color scheme, the luminance weighting, rather than being:

0.3 Red

0.59 Green (NTSC COLOR BALANCE)

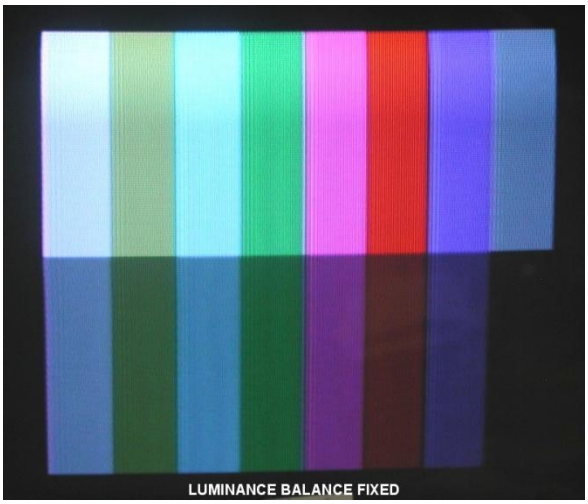
0.11 Blue

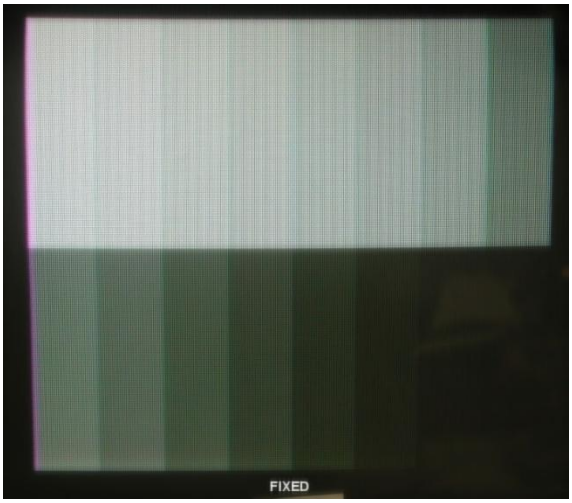
0.29 Green

0.57 Blue (CROMEMCO COLOR BALANCE)

0.14 Red

If the three RGB resistor mixing assignments are switched around to make them conform to an NTSC scheme, the result is shown below:





It became clear after some investigations why this was the case.

The color system which interprets the memory byte (or nibble for a single pixel) used in this mode is MSB…> ..LSB where the 4 bits code I,B,G,R (where I is intensity high or low) and the 3 remaining bits code the pixel in normal resolution mode.

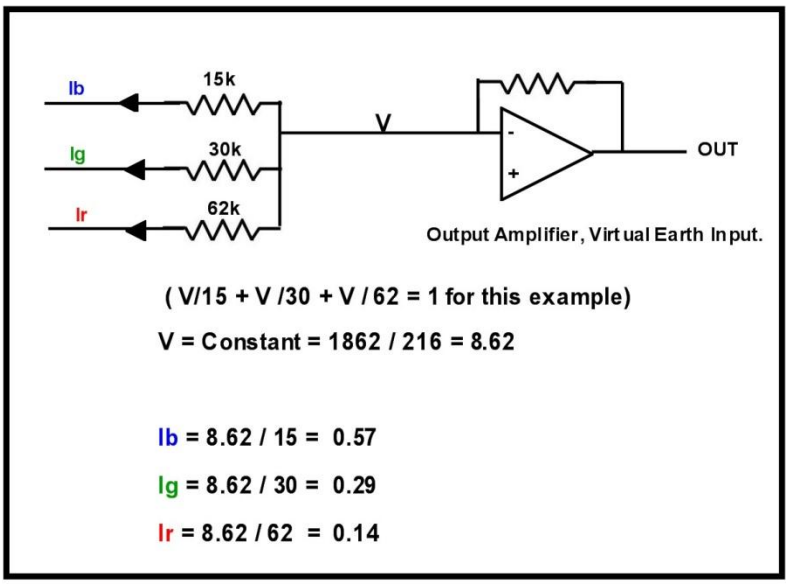
If that was altered to I,G,R,B then Cromemco could have had a color order that matched NTSC along with a grey scale, where the grey scale descended evenly in order with the value of the nibble and got the best of both worlds.

Also an RGB image (if that were to be provided directly from the board) would match up exactly with a composite video image in all respects (except perhaps the superior resolution using RGB video vs composite).

Why Cromemco did not make it like this is a mystery, however, in the field of computer graphics often things like this crop up because most early computer systems used monochrome or RGBI systems (like CGA) and there was less compatibility with domestic television systems.

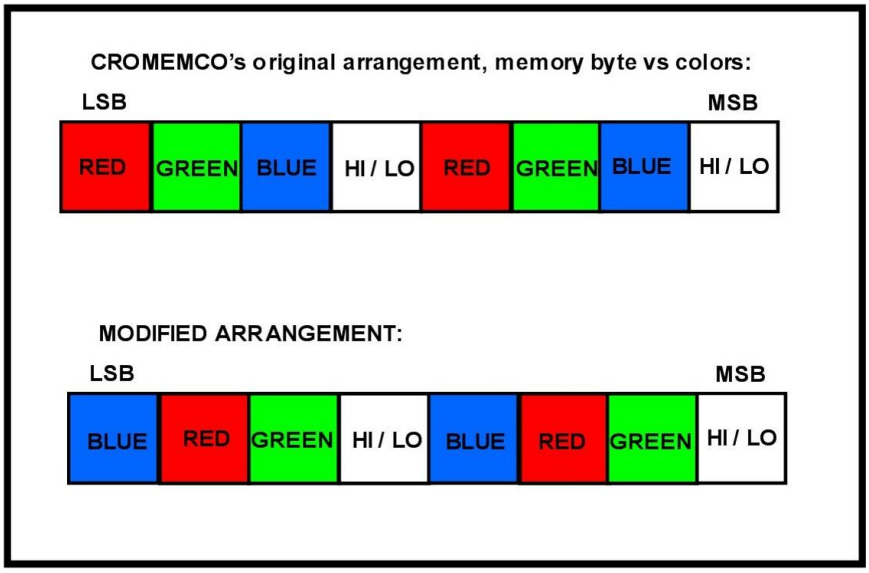
As another example of this sort of thing, in IBM's early computers, from the 1980's era, the CGA output from IBM's CGA card, which had both composite and RGBI outputs, did not match up exactly with what was seen on a composite monitor versus what was seen on a CGA monitor.

The panel below shows how to calculate the relative contributions of the Red, Green and Blue channels (used by Cromemco) to the luminance level of the output:



As noted the result is a uniform grey scale in terms of the magnitude of the memory nibble that codes the grey level, but non standard according to NTSC luminance levels related to the contributions of R, G & B to the luminance levels.

In Cromemco's original scheme, to use the host computer's memory byte to represent two pixels, they assigned them as follows:



In the Cromemco system the blue, then green then red had decreasing binary values.

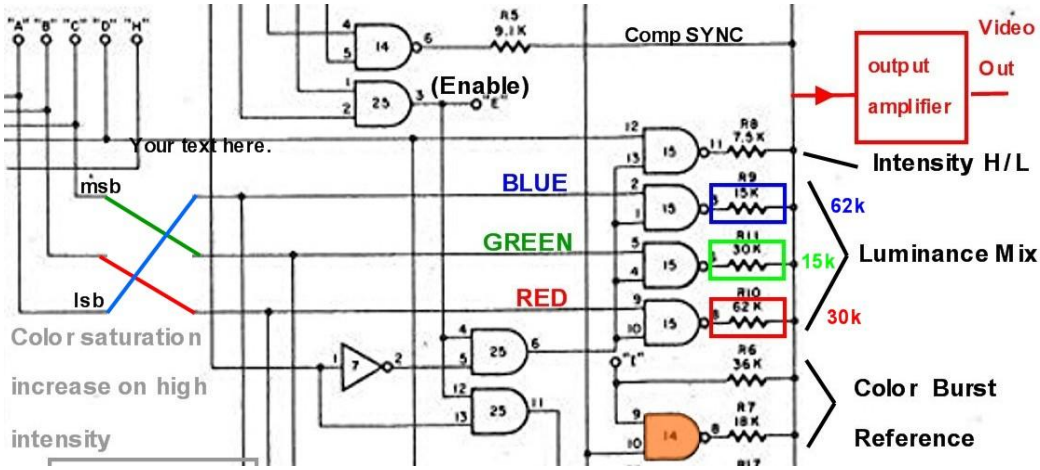
In the NTSC system, where the relative luminance intensities were assigned to the three colors G>R>B, 0.59>0.30>0,11, if this is normalized to make Blue =1 then the proportions are 5.4 Green, 2.7 Red and 1 Blue.

Therefore if the colours are also represented by 3 binary bits per pixel, the intensity weighting is not too far off the magnitude of the bits, which are 4, 2 and 1.This is why the bit order, with respect to colors in a digital system attempting to replicate NTSC video, it is better assigned to Blue = LSB and Green=MSB with red in between. This way, when the bits are mixed together in magnitude, to form a grey scale, it better matches the NTSC system. In other words, in the NTSC system, if a grey scale version of the image is to be preserved, then the magnitude of green is greater than red which is greater than blue.

Due to Cromemco's assignments of these colors, not in that order, it explains why their resistor matrix *appears* mixed up. It is that way so that the grey scale is preserved in terms of higher binary values corresponding to higher white levels in the image.

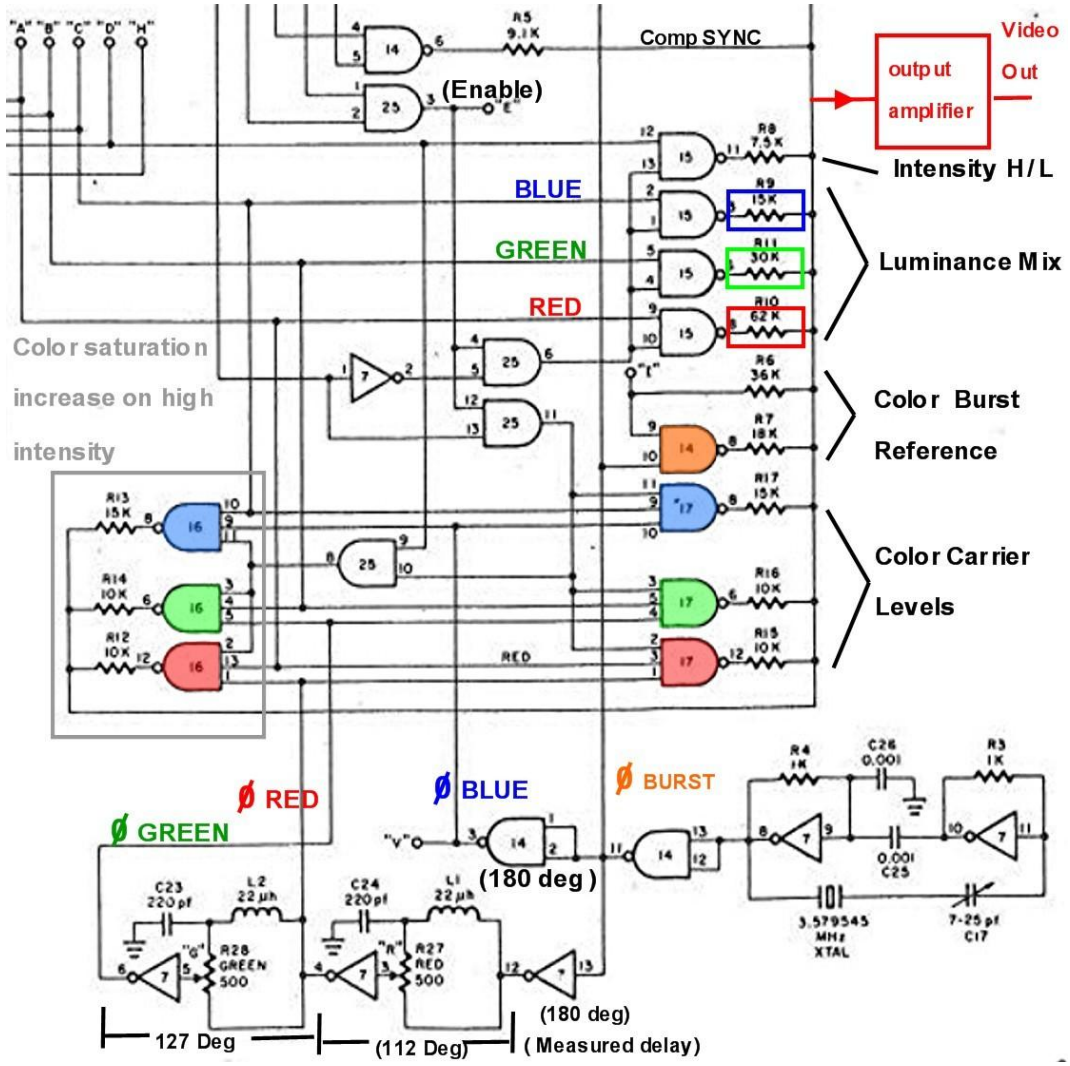
However, it also means that if the color image in memory was derived from an NTSC color image originally, then in monochrome mode, either the Dazzler card or the TV being switched to monochrome, then the picture would not have the correct shades of grey.

It would be simple to modify the Dazzler card to fix this by swapping the three resistors around and by switching the three connections feeding the luminance adder:



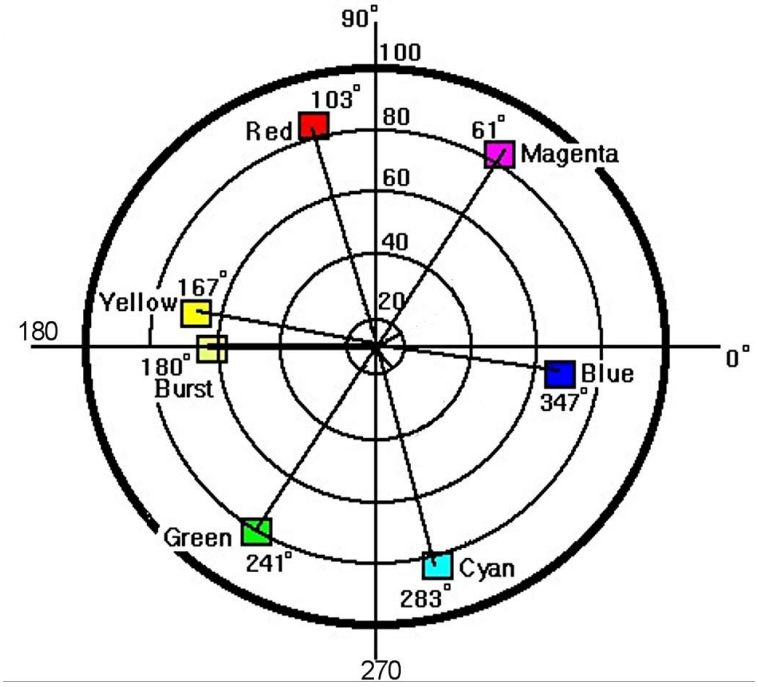
However, I do not propose to modify my Dazzler to fix this color-monochrome greyscale issue, because that would be like trying to change history and I want to keep the Dazzler the way it was designed.

The Cromemco Color Encoder:



The input, to the output amplifier (circuit not shown here), due to it being an inverting input, with heavy negative feedback, acts as a virtual earth. This is so the currents, via the resistors on the gate outputs, can be mixed together without interfering with each other.

The Color Phasors produced by the Cromemco color sub-carrier system:



The standard NTSC diagram above shows the relative color carrier phases of the colors with respect to the color burst (reference) placed at 180 degrees.

Of note, the amplitude of the blue phasor is a little lower than the Red and Green, which explains why they used a 15k rather than 10k resistors on the output of the Blue color carrier gates.

As noted the Blue carrier phase is nearly 180 degrees delayed from the burst. To attain this phase Cromemco simply inverted the Burst signal using a NAND gate wired as an inverter. So it would provide 180 degrees shift . This explains why the Blue bar (on the test pattern) looks just a little Purple, because there is a small phase shift toward Magenta.

With optimum setting of the Red & Green phase controls, looking at the test pattern on the color monitor, I then measured the Red phase delay as (180 + 112) = 292 degrees. As can be seen from the diagram above the exact angle away from the burst reference, for Red, is (180 + 103) = 283 degrees. So that was fairly close.

I measured the total delay for the green carrier as (180 + 112 + 127) = 417 degrees. Subtracting 360 degrees, since the phasor has been delayed longer than one cycle, the

result is 59 degrees. As can be seen from the diagram above, the Green phasor is expected to be about (241-180) = 61 degrees after the burst. That is close enough.

So what does this part of the analysis explain?

With optimum settings of the Green & Red controls, looking at the monitor at least, the red color was slightly shifted 9 degrees towards yellow. The green very close and the blue (not adjustable) shifted approximately (360-347) = 13 degrees toward magenta, explaining why the blue on the color monitor looks just a tad purple.

THE x 4 RESOLUTION MODE:

In this mode each byte of the image memory file in RAM controls 8 pixels, where the bit of the byte turns the pixel either on or off.

The lower 4 bits of the output port 0FH are used to control the intensity and the selected R,G &B colors in any combination.

For 2048 memory bytes there are16384 pixels accounted for and a 128 x 128 pixel array and ***the pixels are three CRT beam scanning lines tall***.

This is unlike the normal resolution mode where on the vertical axis of the pixel there are 6 scanning lines, 3 Even and 3 Odd scanning lines.

I thought it would be worth investigating the x4resolution mode with a still image. One complication is that the image is divided at the hardware level into four 512K blocks, where the address are not sequential. So this required processing the image in four blocks.

The other complication is the way Cromemco organised one byte to represent 4 pixels vertically stacked on each other and not as a linear sequence customary in other systems:

One Byte represents 8 adjacent pixels

LSB D0	D1	D4	D5
D2	D3	D6	D7 MSB

This process was more complicated than it initially seemed. I will explain how I got the file processed and into RAM in my SOL-20 computer.

I started with a 128 x 128 pixel .BMP monochrome high contrast image file in a modern computer running XP and cropped into four separate 64 x 64 pixel files. The 54 byte leader of the .BMP file was manually stripped of in a Hex file editor. The .BMP has three bytes to represent R,G,B, so the actual file size is 3 x 64 x 64 bytes = 12288 Bytes.

This was a manageable size to send to the SOL-20 computer, using the serial port, and Tera-Term and the CP/M program running in the SOL, called PCGET. The four image files were moved to my SOL's floppy disk drive this way.

I had previously written software to move disk files to address 4000H in RAM in the SOL, this helped. After moving the file to memory I acted on it with three custom 8080 programs. I broke the task into three steps to make sure it all worked along the way. The first program COMPF.COM acts on the original file to strip out every second and third byte and overwrite the original file. So now the image was 4096 bytes long. Then that file was operated on with CMPF2.COM. This effectively flips the image vertically by reassigning the addresses. It also moves the start of the resulting file to 2000H to become a source file SF. The reason this is required is that a .BMP file starts coding the pixels from lower left of the image and ends up at the upper right as address values increase. The required file though starts at the upper left and ends at the lower right.

Then the program, CMPF3.COM is run to create the first block of 512 image bytes starting at 1000H in memory and ending at 11FFH, this is the target file TF. The only difference with the programs CMPFB, CMPFC and CMPFD.COM, is they simply load a different starting address. At this point I had not made the software interactive to ask for the starting address.

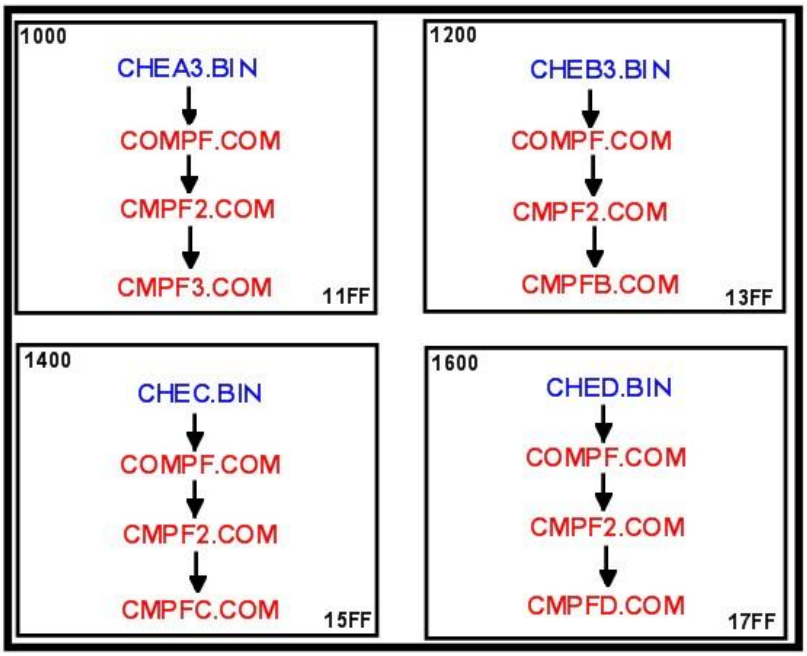
CMPF3.COM was a tricky program. The first 4 pixels of the first line from the SF are examined and a partial byte (4 bits D0,D1,D4,D5) for the TF is created by masking operations on the first 4 bytes. Then the address of the SF is altered to the next line of bytes to create the final 4 bits, D2,D3,D6,D7, of the byte under construction for the TF. When one byte of the TF is made from examination of 8 bytes from the SF, the SF addresses are moved on to examine the next 8 bytes of the SF and create another byte for the TF.

The basic process of using the software is shown below:

DAZZLERS MEMORY MAP FOR 2k BYTE IMAGE

ADDRESS EXAMPLE STARTS AT 1000H, REGISTER 0E,

PROGRAMMED WITH BYTE VALUE 88H



.BIN = 12880 byte image file

4096 Byte file

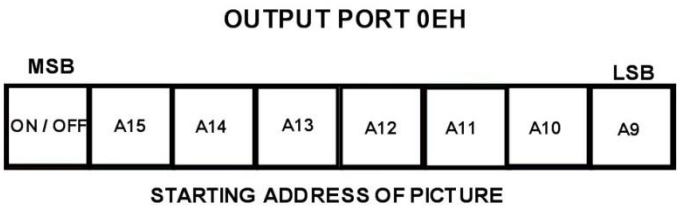
Vertical Addresss flip

512 Byte Dazzler compaitibe file

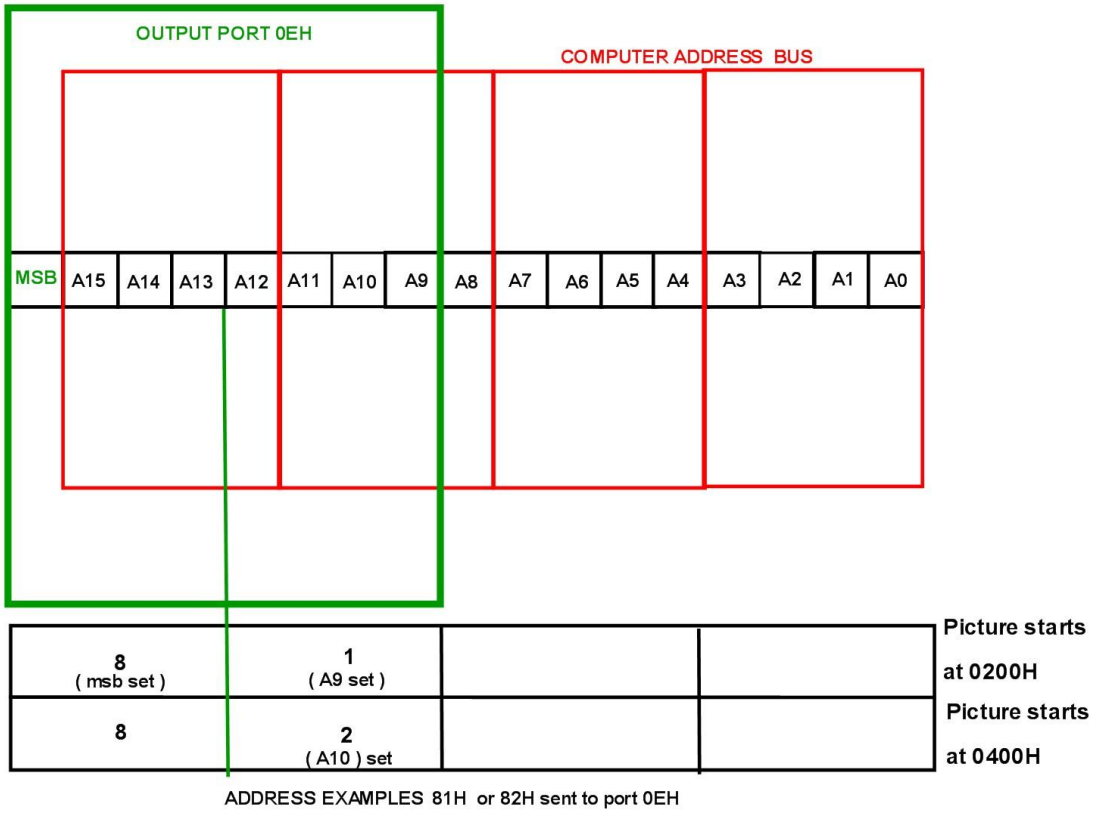
Setting the memory address of the picture for the Dazzler to read:

Initially I was perplexed by the instructions to set the starting address of the image in memory, from the register (output port) assignments shown in the manual. This is because the bits they refer to in their output port have a one bit offset with respect to the actual address lines in the computer. The best way to explain this is with the diagram below:

From the manual:



The MSB here has no counterpart as a bit of a memory address, it is purely to turn the Dazzler on & off. Dividing the memory address bits into the usual bytes:



I had seen in a version of the Kaleidoscope program, when 81H was sent to the port and in the comments it said that "Picture starts at 1000H". This was very odd and incorrect because with 81H sent to the output port 0EH, the picture starts at 0200H as shown in the diagram above.

With 88H sent to the port the address instead, the file starts at 1000H. And this is where my loaded image now was. Running the short program:

3E 88 D3 0E 3E 6F D3 0F C3 04 C0

This switches on the Dazzler and sets the image address start at 1000H in RAM and 6F sets the Dazzler port 0FH to the monochrome resolution x 4 mode.

The image I had stored in RAM appeared at the output of the Dazzler. It is shown below on an amber monochrome computer VDU.

Since the Dazzler was "Revolutionary" when it came out I thought the image was an appropriate choice for a demonstration.

The 4x resolution mode can also be a color mode, using a color monitor, in that the 0F register can be set to R,G and B all on, giving a black & white image, or any of the R,G & B signals can be switched off to globally color the image if wanted and the intensity can be set high or low with another register bit.

SUMMARY:

The Dazzler was an astonishing creation at the time and in my opinion still is. It was designed to bring color graphics into the world of home computer users who were using S-100 computers in the mid 1970's. The Dazzler also found use generating NTSC color graphics for the television industry and provided a way to display images derived from early solid state digital cameras like the Cyclops. The cost of the board/s were kept down due to them not having their own video ram on board and relying on the RAM already present in the host computer.

****