

REAL TIME CLOCK:

```

          +-----+ NTSC
          +--->| /4 |-----+
+-----+ +-----+ | +-----+ |
| COLOR_CLOCK |-->| /9 |--+          +---> RTC
+-----+ +-----+ | +-----+ PAL |
          +--->| /5 |-----+
          +-----+
```

NTSC RTC = 3.579545 MHz /36 = 99.431806 KHz ; 10.05714 us (+0.5714%)

PAL RTC = 4.43361875 MHz /45 = 98.534878 KHz ; 10.14869 us (+1.4869%)

-----  
PAL slower by 0.9155%

```

          +-----+ NTSC
          +--->| /72 |-----+
+-----+ | +-----+ |
| COLOR_CLOCK*2 |--+          +---> RTC
+-----+ | +-----+ PAL |
          +--->| /89 |-----+
          +-----+
```

NTSC RTC = 7.15909 MHz /72 = 99.431806 KHz ; 10.057140 us (+0.57140%)

PAL RTC = 8.8672375 MHz /89 = 99.6319 KHz ; 10.036948 us (+0.36948%)

-----  
PAL faster by 0.20192%

PROGRAMMING VIVIAN:

```
BLOCK = ADDRESS[23,14]
BLKADRS = ADDRESS[13,0]
```

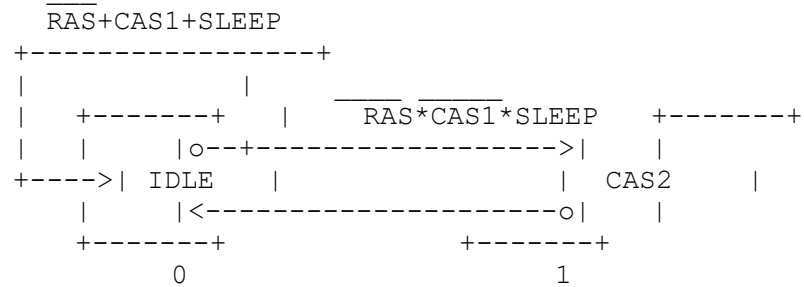
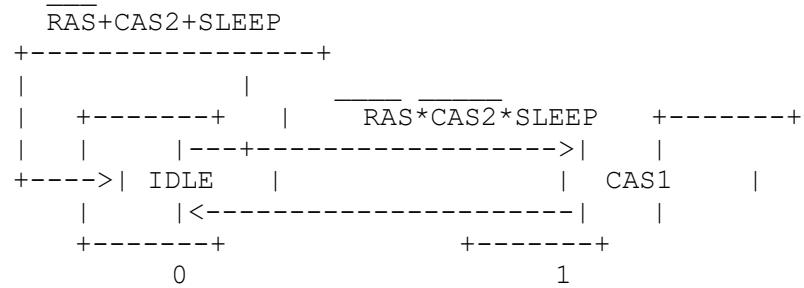
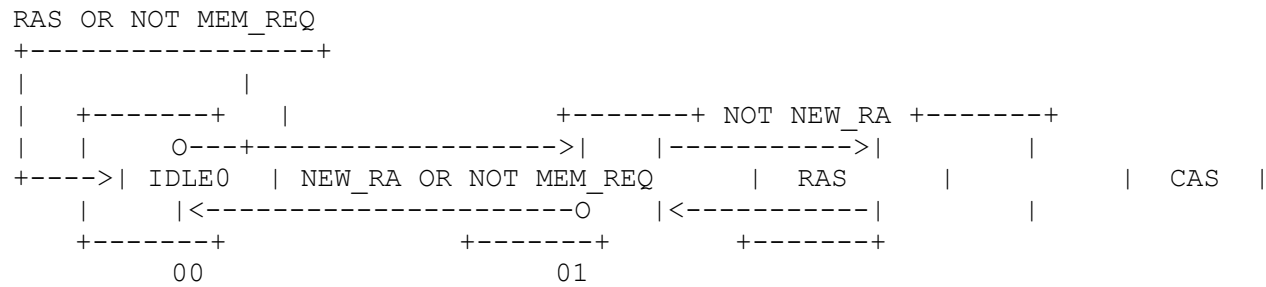
Read/Write BLOCK BLKADRS DATA MODE COMMENT

```

-----
R      %      %  %  %  Normal read cycle.
W      > 0      %  %  NORM Normal write cycle.
W      0      % mode %  Set mode.
W      l      %      p  MAP  Map LOGICAL_BLOCK(l) to
                        PHYSICAL_BLOCK(p).
W      c      %      p  CONF Configure PHYSICAL_BLOCK(p) per
                        configuration word c.
W      t      %      p  TIME Set timing for PHYSICAL_BLOCK(p)
                        per timing word t.
W      l      a      %  COPY DMA copy contents of READ_ONLY_BLOCK
                        (identified during CONF) address =
                        a to LOGICAL_BLOCK(l) address = a.
                        No other writes allowed while in
                        copy mode. CPU data bus tristated
                        and CPU acts as DMA controller.

```

# MEMORY FETCH MACHINE:



CLOCK = COLOR\_CLOCK\*2

STATE PROCESS

-----

IDLE RAS = FALSE

CAS = FALSE

WAIT = TRUE

IF RAS OR NOT MEM\_REQ,

GOTO IDLE0

RASPC RAS = TRUE ; RAS pre-charge time = 2 pixels

CAS = FALSE

WAIT = TRUE

ADR\_BUS = RA

TEMP\_REG = RA

CAS RAS = TRUE ; CAS width = 2 pixels

CAS = TRUE

WAIT = FALSE

ADR\_BUS = CA

CASPC RAS = TRUE ; CAS pre-charge time = 2 pixels

CAS = FALSE

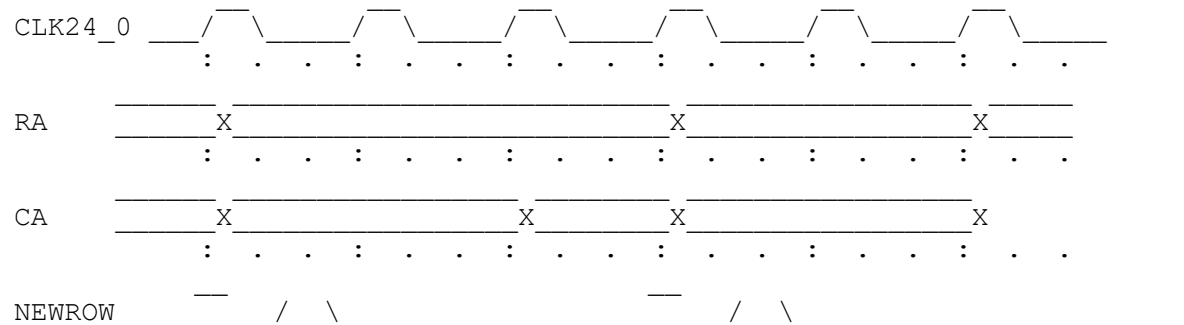
IF MEM\_REQ AND NOT NEW\_RA,

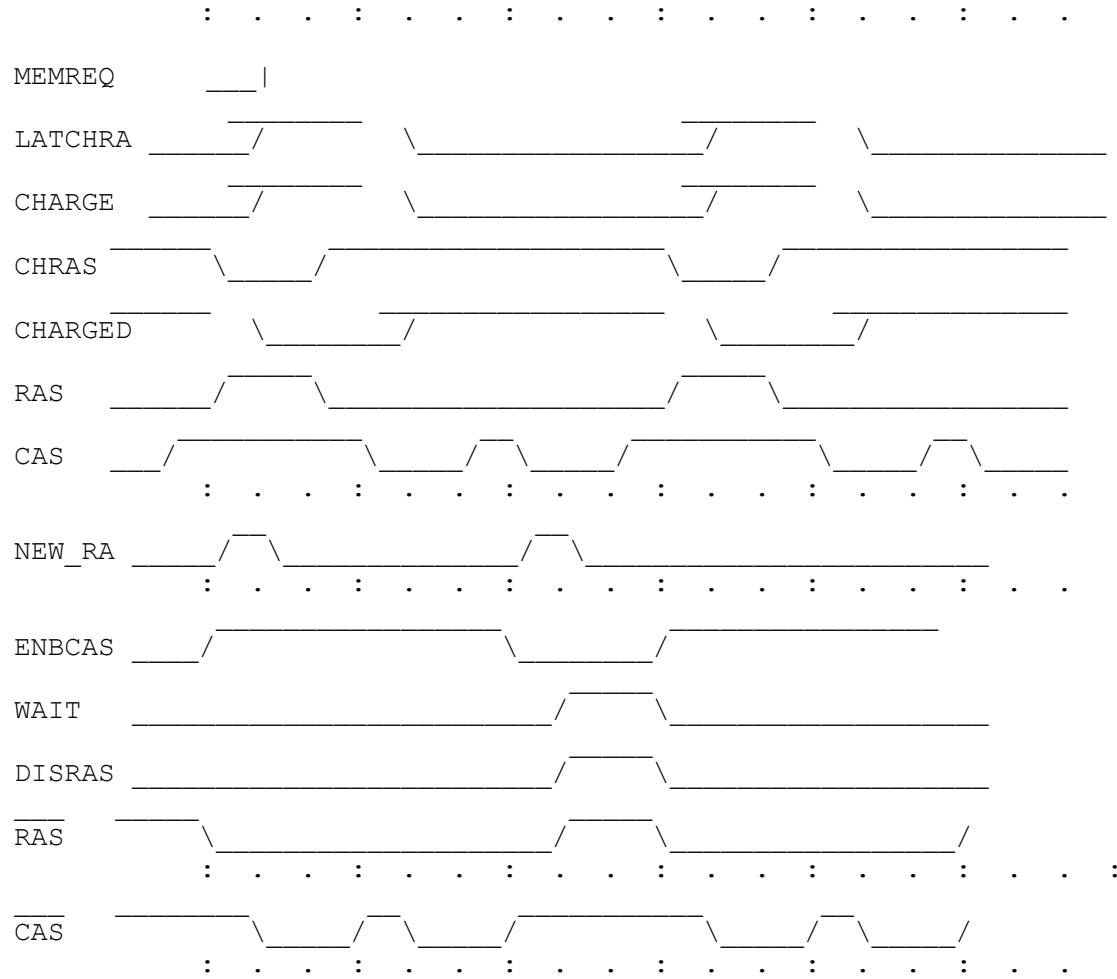
GOTO CAS

IF NOT MEM\_REQ OR NEW\_RA,

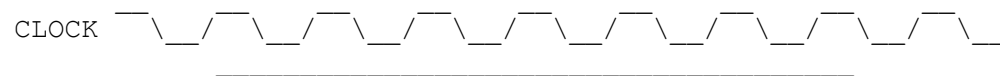
GOTO IDLE

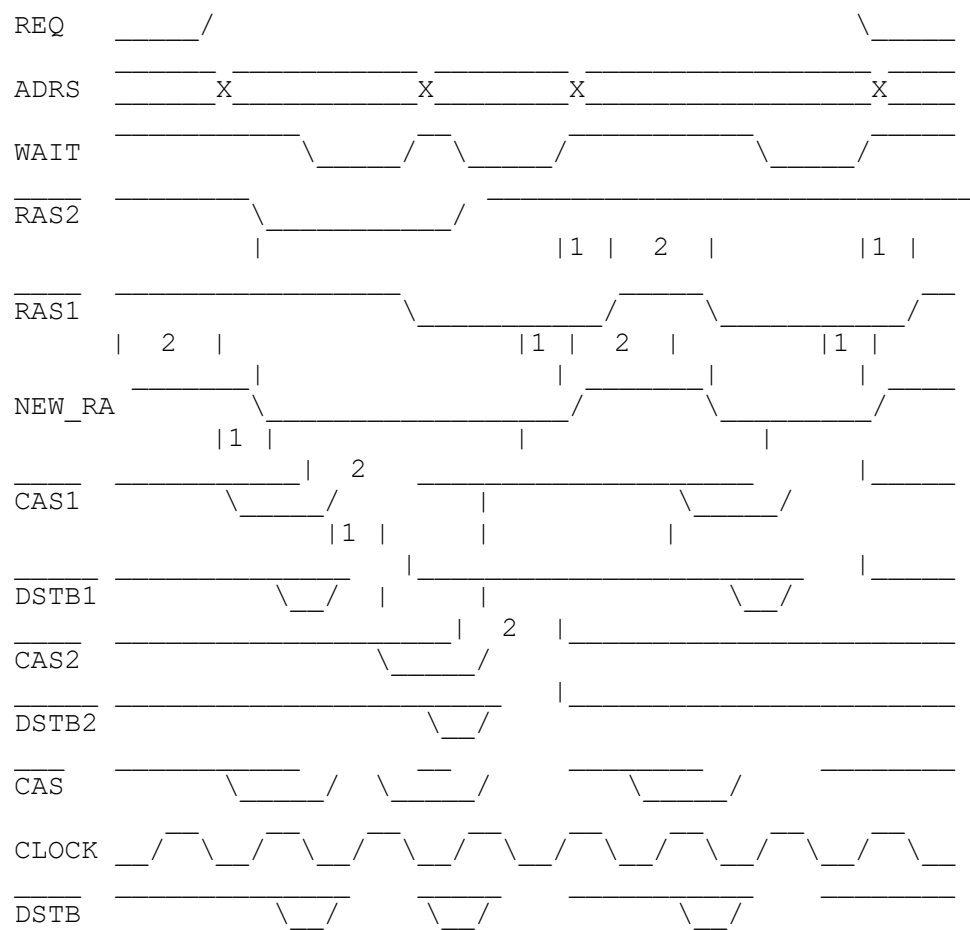
TIMING FOR INTERNAL MODE:



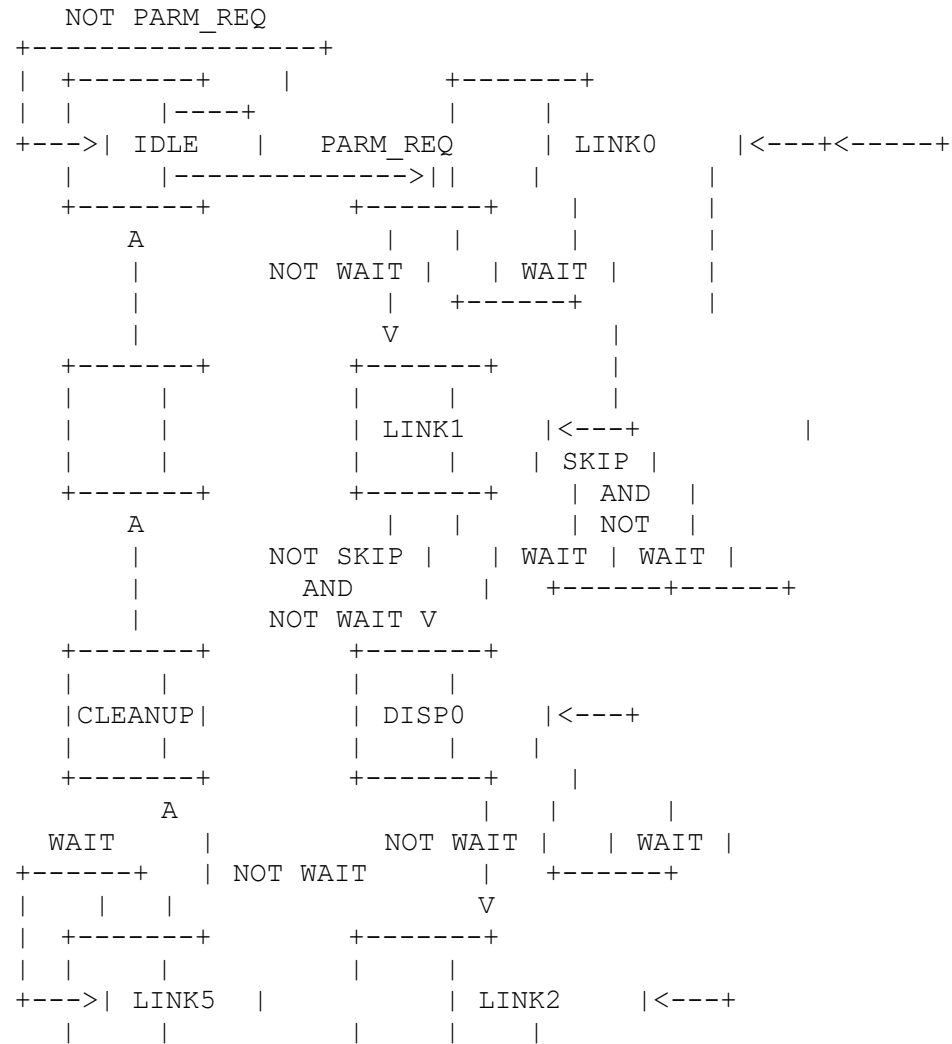


TIMING FOR ONE FETCH AND NO ROW ADDRESS CHANGE:





PERCEPT PARAMETER FETCH MACHINE:



```

      +-----+      +-----+      |
      A          |      |      |
    WAIT        |      NOT WAIT |      | WAIT |
+-----+      | NOT WAIT      | +-----+
|      |      |          V
| +-----+      +-----+
|      |      |      |
+--->| LINK4  |      | LINK3      |<---+
|      |      |      |
+-----+      +-----+      |
      A          |      |      |
    WAIT        |      NOT WAIT |      | WAIT |
+-----+      | NOT WAIT      | +-----+
|      |      |          V
| +-----+      NOT WAIT      +-----+
|      |      |<-----|
+--->| DISP2  |      WAIT      | DISP1      |<---+
|      |      +-----|      |
+-----+      | +-----+      |
              +-----+

```



EQUATES:

```
DXPOS EQU    DAT_BUS[9,0]
DXOFF EQU    DAT_BUS[8,0]
DYPOS EQU    DAT_BUS[9,0]
DYOFF EQU    DAT_BUS[8,0]
DZPOS EQU    DAT_BUS[8,0]
DZOFF EQU    DAT_BUS[7,0]
DHGT EQU    DAT_BUS[15,8]
DLINK EQU    DAT_BUS[13,0]
DDISP EQU    DAT_BUS[13,0]
DDATA EQU    DAT_BUS[13,0]
ALINK EQU    MEM_ADR[13,0]
PC EQU      MEM_ADR[13,0]
```

STATE	ADR_BUS	DAT_BUS	PROCESS	ADDER PERCEPT_DATA (STROBE)
-----	-----	-----	-----	-----
IDLE		MEM_REQ = PARM_REQ ALINK = LINK(P) IF NOT PARM_REQ, GOTO IDLE		
LINK0	ALINK	DYOFF	ALINK = LINK(P) Y = DYOFF FLAG = SKIP(P) ADDA = LINK(P) ; if last link this P ADDB[1,0] = 1 ; was skip, PC = LINK+5 ADDB[2] = SKIP(P) ; else, PC = LINK+1 ADDB[13,3] = 0 PC = ADD  IF WAIT, GOTO LINK0	

```

LINK1 PC      DDISP      ADDA = Y
              OR          ADDB = -LC
              DLINK      Y = ADD
                      SKIP(P) = DSKIP
                      LINK(P) = DLINK

                      IF WAIT,
                        GOTO LINK1

                      IF FLAG AND NOT WAIT,
                        GOTO LINK0

DISP0 DISP    DYPOS
              IF WAIT,
                GOTO DISP0

LINK2 LINK+2   DHGT      ADDA = Y
              DZOFF      ADDB = YPOS
                      Y = ADD
                      FLAG = SIGN(Y)

                      IF WAIT,
                        GOTO LINK2

LINK3 LINK+3   DDATA      ADDA = Y    PY (YS)
                      ADDB = HGT
                      IF FLAG,
                        HGT = ADD

                      IF FLAG,
                        PY = 0
                      IF NOT FLAG,
                        PY = NEGPOLY(Y)

                      IF WAIT,
                        GOTO LINK3

DISP1 DISP+1   DZPOS      ADDB = DATA PHGT (HGTS)
                      IF FLAG,

```

```

        ADDA = 2|Y|
        IF NOT FLAG,
            ADDA = 0
        DATA(P) = ADD
    IF SIGN(HGT)
        PHGT = 0
    IF NOT SIGN(HGT)
        PHGT = NEGPOLY (ABS (HGT) )

    IF WAIT,
        GOTO DISP1

DISP2 DISP+2    DXPOS        ADDA = ZPOS
                  ADDB = ZOFF
                  Z = ADD
                  FLAG = SIGN(ADD)

    IF WAIT,
        GOTO DISP2

LINK4 LINK+4    DXOFF        ADDA = XPOS ; x=0 is X=-128 in space
                  ADDB = 488
                  X = ADD
                  FLAG = SIGN(ADD)
    IF FLAG,          PZ (ZS)
        PZ = 0
    IF NOT FLAG,
        PZ = ABS(Z)

    IF WAIT,
        GOTO LINK4

LINK5 LINK+5    DLINK        ADDA = X
                  ADDB = XOFF
                  X = ADD
                  FLAG = SIGN(ADD)
    SKIP(P) = DSKIP
    LINK(P) = DLINK

```

```
IF WAIT,  
    GOTO LINK5  
  
CLEANUP          IF FLAG,          PX (XS)  
                PX = 0  
IF NOT FLAG,  
    PX = NEGPOLY(X)  
  
GOTO IDLE
```

# ALGORITHM FOR GENERATING YAW:

YAW	FACT1	FACT2	FACT3	FACT4	dz	dx
---	---	---	---	---	--	--
0	-	-	-	-	0	1
1	4	5	4	4	4	80
2	2	3	2	3	4	24
3	2	2	2	2	4	16
4	1	2	1	1	4	10
5	1	1	1	1	4	8
6	0	1	0	1	4	6
7	0	1	0	0	4	5
8	0	0	0	0	4	4
9	0	1	0	0	5	4
A	0	1	0	1	6	4
B	1	1	1	1	8	4
C	1	2	1	1	10	4
D	2	2	2	2	16	4
E	2	3	2	3	24	4
F	4	5	4	4	80	4

$$\text{yaw angle} = \arctan(dz/dx)$$

The basic idea here is as follows:

For yaw < 45 degrees, when x has incremented through 2^FACTn pixels, z increments/decrements by 1. This happens for 'n' = 1, 2, 3, & 4 successively for as many repetitions as the length of the data will permit.

For yaw >= 45 degrees, as x increments by one for each pixel, z increments/decrements by 2^FACTn. This happens for 'n' = 1, 2, 3, & 4 successively for as many repetitions as the length of the data will permit.

During the generation of the present line, the initial z value is incremented/decremented PINC times to arrive at the initial z value for the next line.

The reasoning behind the ' $2^{\text{FACTn}}$ ' increment/decrement instead of simply adding or subtracting a binary is that a power-of-two up/down counter is smaller than an adder. As can be seen, it still yields reasonably usable and accurate values of angle if four intervals are used.

DERIVATION OF COLOR CHARTS:

1. I PLOTTED ALL POSSIBLE COMBINATIONS OF PHASERS (IE, 16X31)

2. I THREW OUT ALL COMBINATIONS WHICH EXCEEDED THE LIMITS

$$0 \leq R, G, B \leq 23.9$$

3. I FOUND THE MAXIMUM SATURATED R, G & B POINTS AND DREW LINES FROM THE ORIGIN TO THOSE POINTS TO DIVIDE THE GRAPH INTO THREE REGIONS

AN R TO G REGION,  
A G TO B REGION &  
A B TO R REGION.

4. I ASSIGNED MINIMUM REQUIRED LUMINANCE FOR EACH SURVIVING POINT

FOR R TO G REGION

$$\begin{aligned} P0 &= .493*(-.30*R-.59*G) \\ P1 &= .877*(.70*R-.59*G) \\ LUMmin &= .30*R+.59*G = -P0/.493 \end{aligned}$$

FOR G TO B REGION

$$\begin{aligned} P0 &= .493*(.89*B-.59*G) \\ P1 &= .877*(-.11*B-.59*G) \\ LUMmin &= .11*B+.59*G = -P1/.877 \end{aligned}$$

FOR B TO R REGION

$$\begin{aligned} P0 &= .493*(-.30*R+.89*B) \\ P1 &= .877*(.70*R-.11*B) \\ LUMmin &= .30*R+.11*B = .3782*P0+.5798*P1 \end{aligned}$$

IN THE ABOVE CALCULATIONS, 'LUMmin' IS THE LUMINANCE WHICH IS  
REQUIRED JUST TO SUPPORT THE CHROMINANCE VECTORS WITHOUT  
ANY ADDED LUMINANCE (IE, FULLY SATURATED CHROMINANCE).

5. I READJUSTED THE MINIMUM LUMINANCE UPWARD FOR POINTS WHICH WERE  
OVERSATURATED (IE, THE SIGNAL DIPPED BELOW -20IRE)

$$\begin{aligned} &\text{OVERSATURATED SIGNAL} < -20\text{IRE} \\ &\text{LUMmin}(+/-)P0, \text{LUMmin}(+/-)P1 < -20\text{IRE} = -5 \end{aligned}$$

6. I THREW OUT ALL POINTS WHICH WERE OVERMODUATED (IE, THE SIGNAL  
OVERSHOT 120IRE) WITH EVEN THE MINIMUM LUMINANCE

$$\begin{aligned} &120\text{IRE} < \text{OVERMODUATED SIGNAL} \\ 29 = &120\text{IRE} < \text{LUMmin}(+/-)P0, \text{LUMmin}(+/-)P1 \end{aligned}$$



DERIVATION OF COLOR CHARTS (CONTINUED):

7. I FOUND THE AMOUNT OF PRIMARY COLORS IN EACH POINT AT MINIMUM LUMINANCE

FOR R TO G REGION

$$\begin{aligned}P0 &= .493*(-.30*R-.59*G) \\P1 &= .877*(.70*R-.59*G) \\R &= -P0/.493+P1/.877 \\G &= -2.4066*P0-.5798*P1\end{aligned}$$

FOR G TO B REGION

$$\begin{aligned}P0 &= .493*(.89*B-.59*G) \\P1 &= .877*(-.11*B-.59*G) \\B &= P0/.493-P1/.877 \\G &= -.3782*P0-1.7200*P1\end{aligned}$$

FOR B TO R REGION

$$\begin{aligned}P0 &= .493*(-.30*R+.89*B) \\P1 &= .877*(.70*R-.11*B) \\R &= .3782*P0+1.7200*P1 \\B &= 2.4066*P0+.5798*P1\end{aligned}$$

8. I SOLVED FOR MAXIMUM LUMINANCE FOR EACH POINT BY ADDING DELTA TO ALL THREE COLORS UP TO THE MAXIMUM OF 24 FOR ANY ONE COLOR

FOR R TO G REGION

$$\begin{aligned}\text{DELTA} &= \text{SMALLEST}\{24-R, 24-G\} \\ \text{LUMmax} &= \text{LUMmin}+\text{DELTA}\end{aligned}$$

FOR G TO B REGION

DELTA = SMALLEST{24-G,24-B}  
LUMmax = LUMmin+DELTA

FOR B TO R REGION

DELTA = SMALLEST{24-R,24-B}  
LUMmax = LUMmin+DELTA

9. I READJUSTED THE MAXIMUM LUMINANCE DOWNWARD FOR POINTS WHICH WERE OVERMODULATED (IE, THE SIGNAL OVERSHOT 120IRE)

120IRE < OVERMODULATED SIGNAL  
29 = 120IRE < LUMmax (+/-) P0, LUMmax (+/-) P1

10. I ENTERED LUMmin AND LUMmax FOR EACH POINT ON THE CHARTS BESIDE EACH POINT.

# GENERATION OF VIDEO OUTPUT

```

+-----+
! MUX !
+-----+
'0000'====4=>!3      !
DIM[B,8]==4=>!2      !
DIM[7,4]==4=>!1      ! LUMI
DIM[3,0]==4=>!0 out!=4===== [3,0]=>H
!      !
SEL=====2=>!sel  !      +-----+
+-----+      !ADDER!
+-----+      +-----+
!16x5 ROM!      !      !
+-----+      LUM====5=>!a      !
! F ! 1F !      !      !
! E ! 1E ! P1=>H      +-----+      !      !
! D ! 1D !      H      ! MUX !      !      ! CI
! C ! 1C !      H      +-----+      !a+b+c!=6===== [9,4]=>H
! B ! 1B !      H====I>o==5=>!3      !      !
! A ! 1A !      H H=====5=>!2      !      !      CI      IRE
! 9 ! 19 !      H H==I>o==5=>!1      !      !      DEC HEX LEVEL
P0=4=>! 8 ! 18 !==>H=====5=>!0 out!=5=>!b      ! --- --- ---
! 7 ! 17 !      !      !      !      29 1D +120
! 6 ! 16 !      H=2=>!sel      !      !      24 18 +100
! 5 ! 0A !      H      +-----+      !      !      0 00      0
! 4 ! 08 !      H      !      !      -5 3B      -20
! 3 ! 06 !      H-[0]----->!c      !
! 2 ! 04 !      H      +-----+
! 1 ! 02 !      H
! 0 ! 00 !      H
+-----+      H
!      CTRL      !      H

```

```

+-----+
!      state!=2=====>H
! LUM+P0 = 00 !
! LUM-P1 = 01 !
! LUM-P0 = 11 !
! LUM+P1 = 10 !
+-----+
H<=====H
H      +-----+
H      !      1024x6  ROM      !
H      +-----+
H      !
H      ! A = CI*10**{-LUMI/16* !
H      !      LOG[ABS(CI)] }+10 !
H      !
H      ! B = INT[A-.49] ; CI<0 !
H      !      = 10          ; CI=0 !
H      !      = INT[A+.49] ; CI>0 !
H==10==>!
! C = 0 ; B<5
!      = B ; 4<B<40
!      = 39 ; B>39
!
+-----+
! C      +-----+ +-----+
! =6==>! 6 BIT D/A !-->!
!      +-----+ ! SUM !--> VIDEO
!      SYNC & BLANK----->!
+-----+

```

# VIDEO OUTPUT ROM PROGRAMMING TABLE

bits 9,4 in HEX (DEC)	bits 3,0 in HEX															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
00 ( 0)	06	05	05	04	04	03	03	03	02	02	02	02	01	01	01	01
01 ( 1)	07	06	05	05					02	02	02	02	01	01	01	01
02 ( 2)	08	07	06	05					03	02	02	02	02	01	01	01
03 ( 3)	09	08	07	06	TO BE REVISED				03	02	02	02	02	01	01	01
04 ( 4)	0A	09	07	06					03	03	02	02	02	01	01	01
05 ( 5)	0B	09	08	07					03	03	02	02	02	01	01	01
06 ( 6)	0C	0A	09	07	06	05	05	04	03	03	02	02	02	01	01	01
07 ( 7)	0D	0B	09	08	07	06	05	04	03	03	02	02	02	02	01	01
08 ( 8)	0E	0C	0A	08	07	06	05	04	04	03	03	02	02	02	01	01
09 ( 9)	0F	0D	0B	09	07	06	05	04	04	03	03	02	02	02	01	01
0A (10)	10	0D	0B	09	08	07	06	05	04	03	03	02	02	02	01	01
0B (11)	11	0E	0C	0A	08	07	06	05	04	03	03	02	02	02	01	01
0C (12)	12	0F	0C	0A	09	07	06	05	04	03	03	02	02	02	01	01
0D (13)	13	10	0D	0B	09	07	06	05	04	04	03	02	02	02	01	01
0E (14)	14	11	0D	0B	09	08	06	05	04	04	03	02	02	02	01	01
0F (15)	15	11	0E	0C	0A	08	07	05	04	04	03	02	02	02	01	01
10 (16)	16	12	0F	0C	0A	08	07	06	05	04	03	03	02	02	01	01
11 (17)	17	13	0F	0D	0A	09	07	06	05	04	03	03	02	02	01	01
12 (18)	18	14	10	0D	0B	09	07	06	05	04	03	03	02	02	01	01
13 (19)	19	14	11	0E	0B	09	07	06	05	04	03	03	02	02	01	01
14 (20)	1A	15	11	0E	0B	09	08	06	05	04	03	03	02	02	01	01
15 (21)	1B	16	12	0E	0C	0A	08	06	05	04	03	03	02	02	01	01
16 (22)	1C	17	12	0F	0C	0A	08	06	05	04	03	03	02	02	01	01

17 (23)	1D 17 13 0F	0C 0A 08 07	05 04 03 03	02 02 01 01
18 (24)	1E 18 14 10	0D 0A 08 07	05 04 03 03	02 02 01 01
19 (25)	1F 19 14 10	0D 0A 08 07	05 04 04 03	02 02 01 01
1A (26)	20 1A 15 11	0D 0B 09 07	06 04 04 03	02 02 01 01
1B (27)	21 1A 15 11	0D 0B 09 07	06 05 04 03	02 02 01 01
1C (28)	22 1B 16 11	0E 0B 09 07	06 05 04 03	02 02 02 01
1D (29)	23 1C 16 12	0E 0B 09 07	06 05 04 03	02 02 02 01
1E (30)	24 1D 17 12	0F 0C 09 07	06 05 04 03	02 02 02 01
1F (31)	25 1D 17 13	0F 0C 09 08	06 05 04 03	02 02 02 01

# VIDEO OUTPUT ROM PROGRAMMING TABLE (CONTINUED)

bits 9,4 in HEX (DEC)	bits 3,0 in HEX															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20 (32)	26	1E	18	13	0F	0C	0A	08	06	05	04	03	02	02	02	01
21 (33)	27	1F	19	14	0F	0C	0A	08	06	05	04	03	02	02	02	01
22 (34)	28	20	19	14	10	0D	0A	08	06	05	04	03	02	02	02	01
23 (35)	29	20	1A	14	10	0D	0A	08	06	05	04	03	02	02	02	01
24 (36)	2A	21	1A	15	10	0D	0A	08	06	05	04	03	02	02	02	01
25 (37)	2B	22	1B	15	11	0D	0A	08	06	05	04	03	03	02	02	01
26 (38)	2B	23	1B	16	11	0D	0B	08	07	05	04	03	03	02	02	01
27 (39)	2B	23	1C	16	11	0E	0B	08	07	05	04	03	03	02	02	01
28 (40)	2B	24	1C	16	12	0E	0B	09	07	05	04	03	03	02	02	01
29 (41)	2B	25	1D	17	12	0E	0B	09	07	05	04	03	03	02	02	01
2A (42)	2B	26	1D	17	12	0E	0B	09	07	05	04	03	03	02	02	01
2B (43)	2B	26	1E	18	12	0E	0B	09	07	05	04	03	03	02	02	01
2C (44)	2B	27	1F	18	13	0F	0B	09	07	05	04	03	03	02	02	01
2D (45)	2B	28	1F	18	13	0F	0C	09	07	05	04	03	03	02	02	01
2E (46)	2B	29	20	19	13	0F	0C	09	07	06	04	03	03	02	02	01
2F	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
30	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
31 (-15)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
32 (-14)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
33 (-13)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
34 (-12)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
35 (-11)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
36 (-10)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

37 (-9)	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
38 (-8)	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
39 (-7)	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
3A (-6)	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
3B (-5)	01 01 01 01	01 01 01 01	01 01 01 01	01 01 01 01
3C (-4)	02 02 02 02	02 01 01 01	01 01 01 01	01 01 01 01
3D (-3)	03 03 03 02	02 02 02 02	01 01 01 01	01 01 01 01
3E (-2)	04 04 03 03	03 02 02 02	02 02 01 01	01 01 01 01
3F (-1)	05 04 04 04	03 03 03 02	02 02 02 01	01 01 01 01













