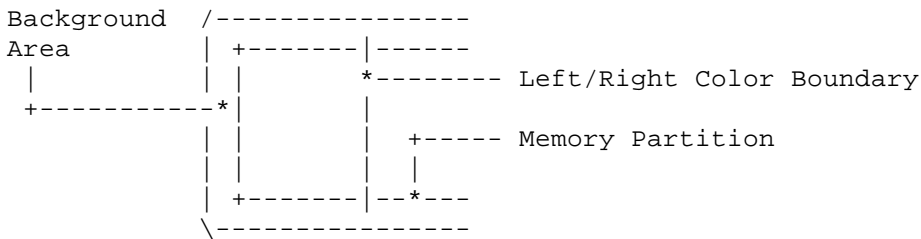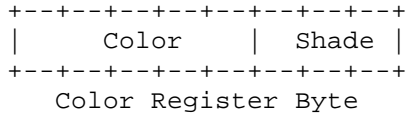Bally Astrocade Screen Operations
By John Perkins

The screen is divided into a left and a right side with a movable boundary. The
following outputs prevail:
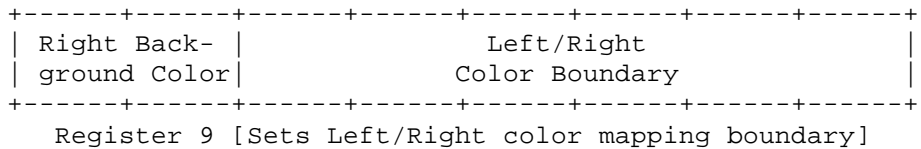
     &(0)=right side register 0   &(4)=left side register 0
     &(1)=right side register 1   &(5)=left side register 1
     &(2)=right side register 2   &(6)=left side register 2
     &(3)=right side register 3   &(7)=left side register 3

     +--+--+--+--+--+--+--+--+
     | Color | Shade |
     +--+--+--+--+--+--+--+--+
        Color Register Byte

     Background /----------------
     Area | +-------|------
      | | | *-------- Left/Right Color Boundary
      +-----------*| |
                  | | | +----- Memory Partition
                  | | | |
                  | | | |
                  | +-------|--*---
                  \----------------

Bally BASIC continually sets &(4) and &(5) to the color/shade defined by BC
[Background Color], and &(6) and &(7) to the color/shade defined by FC
[Foreground Color]. These are fixed while BASIC is in control. But by moving
the boundary so that the right side is visible, we can then control four
different color/shades by using the &(0) thru &(3). [Type this] example [into
BASIC]:

     &(0)=30;&(1)=85;&(2)=153;&(3)=125;&(9)=0

Three colors are displayed- listing, background, and "garbage" at the top (more
on this later). The fourth color should be visible as we scroll the text into
the upper border area. With &(9) at some other value, such as 10, the screen is
divided and the FC and BC [BASIC] commands allow two more colors on the screen.
Actually, &(9) has two functions:

     +------+------+------+------+------+------+------+------+
     | Right Back- | Left/Right |
     | ground Color| Color Boundary |
     +------+------+------+------+------+------+------+------+
         Register 9 [Sets Left/Right color mapping boundary]

The least significant six bits set the boundary position (four pixels or one
memory byte per unit). The most significant two bits choose the color register
associated with the left side background and the right side background. Try
[typing this example in BASIC after having tried the first example]:

     &(9)=135.

The "garbage" mentioned above is actually the stored program, in the screen
memory, using the even bit positions.

```
    +--------+---------+--------+---------+
    | Screen | Program | Screen | Program |
    +--------+---------+--------+---------+...
       Odd Bit  Even Bit  Odd Bit  Even Bit ...
```

Each pixel equates to two bits of memory- 4 pixels to an 8 bit byte. The two
bits of each pixel can have four representations:

    00 = &(4) left   or   &(0) right
    01 = &(5) left   or   &(l) right
    10 = &(6) left   or   &(2) right
    11 = &(7) left   or   &(3) right

When Bally BASIC sets the screen boundary (&(9)) all the way to the right,
then only the left registers &(4) to &(7) are used. Since it also sets &(4) and
&(5) to BC, and &(6) and &(7) to FC, only the odd bits of memory show on the
screen.

    00 is the same color as 01
    10 is the same color as 11

By storing the program in the even bits it can occupy screen memory (as every
other bit) and yet be invisible. However, by moving the boundary to the left,
the right-side registers are used, and since the program above set these to
different colors, the stored program becomes "visible" as the garbage at the top
of the screen.

&(10) controls how many raster lines are displayed from memory as opposed to
being part of the background. &(10)=204 displays all of memory (RAM) allowing
visual inspection of the running program.

Note: Original article informatiion:

- Perkins, John "Tutorial (1) Screen Operations," ARCADIAN, 1, no. 6 (May
1979): 40.

End of article
```