Sandevices E68x Pixel Controller: Pixel String Configuration Tips

Nov 26, 2011 1st Revision

This document will define the various configuration options, setup commands, and limitations of the E68x controller series. This will not focus on command syntax as that is covered in the operating manual. Instead this document will explain how the architecture of the E68x controllers needs to be taken into account when planning your pixel configuration.

The E68x series can operate a wide range of pixel configurations. Because of this flexibility, it's important to understand the limits of the controller when planning your pixel system.

Hard limits (per controller):

No more than 16 total pixel strings.

No more than 680 unique pixels or pixel groups and no more than 4 DMX universes.

No more than 2 pixel voltages, and no more than 4 pixel chip types (e.g. 6803, 2801, GE, etc).

Sometimes these 'hard' limits can't be achieved. To understand why it's important to understand how the pixel outputs of the E68x controllers are organized. There are a total of 16 pixel string outputs, and they are arranged in 4 **CLUSTERS** of 4 strings. The important point here is that **most** string characteristics must be identical for all strings attached to a given cluster.

As an example, that's why there's a 'hard' limit of 4 different pixel types per controller, because all strings on a cluster must be of the same type; since there are only 4 clusters, there can be no more than 4 different pixel types.

Likewise the limitation of 2 different pixel voltages is because there are only 2 pixel power connectors; one of them powers clusters 1 and 2, the other powers clusters 3 and 4. This can create limitations also, for example you can't have 12 strings of 5-volt pixels and 4 strings of 12 volt pixels because of that power arrangement: Clusters 1 and 2 strings MUST be the same voltage, likewise pixels on clusters 3 and 4 MUST be the same voltage. So to achieve a total of 16 strings using 2 different voltages, they must be split 8 and 8.

Let's look at the list of characteristics that can be defined for pixel strings attached to the E68x:

Voltage	Typically will be either 5 volts or 12 volts
Chip Type	This is the type of integrated circuit that's built into each individual pixel. Examples are 6803, 2801, GE ColorEffects, 1804, 3001, etc.
Number of pixels per string	50 is common, some strings are 42 or 63, or 85 pixels, and strings may be made a custom length for some installations.
RGB Color Order	You would think this would be pretty standard, Red, then Green, then Blue, but it's not. You might run across any of the 6 possible color orders (meaning which color responds to which of the 3 DMX addresses used by each pixel).

Voltage is strictly dictated by how the pixel power supplies are wired to the controller, it's not a programmable option. The other 4 items listed above though are programmable parameters that must be configured using the E68x's web interface, as described in the operating manual. Each of these parameters applies to an entire cluster (up to 4 strings) of pixels.

There are some additional parameters not related specifically to the pixel strings themselves, but instead to how they are arranged or used, that also apply to entire clusters of strings. **Grouping** allows you to control the pixels in groups, rather than individually. This can be done either to conserve DMX addresses, or to simplify the control (sequencing) of the pixels. The **number of strings** in use in a particular cluster can be defined from 0 to 4. The **starting DMX address** (combined with other factors) determines the range of DMX addresses that will be used by each cluster.

It's important to understand that every string attached to a particular cluster must have the same: Operating voltage, Chip type, and RGB order. As mentioned earlier, voltage is determined by the power supply connections and isn't programmable; chip type is handled by the Chip command, and RGB order is handled by the RGB command.

Now let's discuss pixel string length and grouping. There are 3 items displayed on the web page for this: Pixels, Grp (Grouping), and Str Len (String Length):

Important: Pixels is often misunderstood. The term pixel is a contraction of the phrase "picture element". It means the smallest part of the display that can be individually controlled. Pixels, **as the term is used in the configuration of the E68x controllers**, is the smallest display element that can be controlled **by the device that's talking to the E68x**.

That probably deserves some additional explanation. The factor that makes this potentially confusing is grouping. When pixels are grouped it means that instead of being able to control each individual RGB pixel, we are going to control them in groups. We may want to use grouping to reduce the total number of DMX addresses needed, or we may use grouping to simplify the programming of the software that's going to control our pixels. Let's say we have a string of 50 RGB pixels, and we want to control them in groups of 5. Now, instead of using 150 DMX addresses (50x3) we only need 30 DMX addresses (10x3). And to our external control software this 50 pixel string now looks like a 10 pixel string.

The bottom line is, the value labeled PIXELS on the E68x configuration screen is the number of pixels as seen by the external software, so in the example given above PIXELS would be 10, even though the physical string is 50 pixels long.

So you will determine and enter your desired values for Pixels and Grouping. "***Str Len" is a calculated value (you can't change it directly) and it should match the physical number of pixels in each string.*** Let's look at a couple of examples:

Example 1, 50 pixel strings where each pixel is to be controlled individually:

GROUP=1 PIXELS=50 Str Len will be calculated and displayed as: 50

Example 2, 50 pixel string where pixels are to be controlled in groups of 5:

GROUP=5 PIXELS=10 Str Len will be calculated and displayed as: 50

Example 3, 50 pixel string where we will control the entire string as a single pixel:

GROUP=50 PIXELS=1 Str Len will be calculated and displayed as: 50

In the following discussion, we will assume group size of 1, and therefore pixels as displayed on the web page, and length of string in pixels are identical, and the terms will be used interchangeably.

Actual String Length vs Defined String Length: It has been shown that the string length is calculated based on pixels and grouping, and applies to all strings in the cluster. This length though should be thought of as the maximum length, and in fact it's perfectly fine for some of the strings making up the cluster to be shorter than this maximum.

There are 2 common scenarios where this would come into play. First, if you wanted to group strings of different sizes together in a cluster because they are part of a common display element, the second case would be if you have more different string sizes than there are clusters.

The only "down-side" to having one or more "short" strings in a cluster is that you might wind up wasting DMX addresses. This is because if you define a cluster as having 4 strings of 50 pixels, then that cluster is going to "use up" 600 DMX addresses, 150 per string. Now it's perfectly fine if some of those strings are actually shorter than 50 pixels, but you'll still use the full 600 addresses. If your configuration is such that you waste too many addresses, you might find that you don't have enough available addresses for all of your pixels.

For maximum efficiency in terms of utilizing DMX addresses, you should keep strings within a particular cluster as close to the same size as possible.

Now there is one special case. If you only have one "short" string in a cluster, and it's the last string of the cluster, you don't have to waste any DMX addresses at all. We'll explain that when we get to the section on DMX addresses.

There are some additional string parameters that are assigned on a per-string basis, meaning that even strings within the same cluster can have different values. These parameters are Reversed, Zigzag, and Null Pixels. The specific application of these parameters, and the syntax for the commands that set these values, are discussed in detail in the operating manual.

Assigning DMX addresses:

DMX addresses are grouped into Universes. By definition a Universe consists of 512 DMX addresses (1-512). The E68x controllers can receive data from no more than 4 different DMX universes. Because each pixel uses 3 addresses, and because we never allow a single pixel's 3 addresses to span across multiple universes, we only use addresses 1 through 510 odf each universe. 4 universes X 510 addresses per universe = 2040 total DMX addresses. Dividing by 3 (3 addresses per pixel) gives us 680, the maximum number of individual pixels that can be conreolled by a single E68x controller..

There are two stages to the assignment of DMX addresses on the E68x controllers. First we define which 4 universes the controller will recognize or "listen to". The E68x controllers have 4 DMX "sockets". Each socket can receive DMX data on only one particular universe. So the first step is to determine which 4 universes this board will listen to, and assign those 4 universe numbers to the 4 sockets using the UNIVERSE command, as detailed in the operating manual. **Important: Do not duplicate universe numbers, each socket's universe number MUST be unique. If you aren't using all 4 sockets, just assign the unused sockets to any universe number other than the universe numbers assigned to sockets that are being used. Socket numbers can range from 1 to 63999.**

Note that this is the only time we use actual universe numbers. When addresses are assigned to the pixel string clusters they will reference sockets, not universes.

Once we have assigned universe numbers to our sockets, we may assign a starting DMX address to each cluster. These addresses are assigned as a **socket number** (1-4) **(NOT an actual universe number!)** and a DMX channel number within the socket (1-508). Typically, in a single controller installation, the controller will be configured to respond to universes 1 through 4, and socket numbers, and universe numbers, will in fact be one and the same. When using multiple controllers, or if using a universe range other than 1 to 4, universe numbers and socket numbers will not be the same.

Typically (but not always) the first cluster will get a starting DMX address of S1-1, the first channel of the first socket. Note that you only assign the starting address, the ending address for the cluster is calculated and displayed based on the number of active strings in the cluster, and the number of pixels per string.

You may want to assign each cluster to a different socket. This is fine but not necessary. You can start each cluster at 1 more than the ending range of the previous cluster. It's also permissible for a cluster to span across multiple sockets, and it's permissible (but only proper for certain situations) for clusters to fully or partially overlap their DMX address range with another cluster.

Earlier we discussed the issue of "wasted" DMX addresses when strings on a given cluster are of different lengths, and we mentioned that it's possible to avoid wasting DMX addresses if there's only one short string. Here's how:

Assume our first cluster consists of 3 50-pixel strings and 1 20-pixel string. We would define the cluster as being 4 strings of 50 pixels since the longest string is 50. If we assign a starting address of S1-001 then this cluster will be assigned an address range of: S1-001 thru S2-090. (4 strings X 50 pixels X 3 addresses per pixel = 600 DMX addresses, so we use all 510 addresses of the first socket, remember 511 and 512 aren't used, and the first 90 addresses of the 2nd socket.)

Now, IF the short string is the last string, then those last 90 addresses won't actually be used because the pixels that would respond to those addresses don't exist. So, rather than starting our next cluster at S2-091 as we normally would, we can start the 2nd cluster at S2-001. Now those 90 pixels are no longer wasted. This is one scenario where it would make sense for two cluster's DMX address ranges to overlap. Another scenario would be if you wanted 2 strings, or 2 groups of strings, to light identically. Just assign them the same address range and it will happen automatically.