Airfield Sequences

www.ledseq.com

Document last updated: March 22, 2013

Contents

Introduction

Airfield kits include 64 individually programmable LEDs as well as 8 LEDs operating together in a general illumination bank. We include all the cables needed (including two spares) plus a roll of strong adhesive tape to mount the LEDs anywhere on a pinball playfield.

The circuit board board is the core of our kit. It includes hardware and firmware capable of controlling these LEDs. Typically our kit is used to reproduce the same chase patterns seen on original pinball machines. Our kit can alternatively be used to light up any other object which it makes sense to attach LEDs to, such as a Christmas tree or a storefront window.

Airfield sequence files describe pattern and timing data for LEDs. They are plain ASCII text files consisting of pattern rows and control rows. Pattern rows determine LED state. Control rows perform actions such as changing brightness or speed. These files are stored on the SD memory card and are loaded and played when the device is turned on.

The sequence file is played one row at a time and turns the LEDs on and off. When the end of the sequence file is reached Airfield returns to the top of the file and starts over again, looping infinitely.

Sequences

Sequence files may be downloaded from our website, built with our online editor, or constructed manually with a text editor. If you simply want to play pre-built sequences all you need to do is download and save them to your SD card and Airfield will do the rest. However if you wish to understand the file format better so you can construct your own from scratch, debug a particularly challenging sequence, or design your own utilities to work with sequence files, all the details are included in this document.

Pattern Rows

Pattern rows are 64 characters wide, each column referring to the state of one LED. The value of each character in the row determines the state: 0 (off), 1 (on), or 2 (random). There are 64 individually controllable LEDs. In general there will be an image map showing the physical location of each LED and its corresponding index number. If you are using a pre-built sequence this map will already be available.

If you are designing a new sequence file you may need to create a map. We recommend taking a digital photo of the object you are attaching the LEDs to and annotating that photo with your own index numbers. We suggest using sequential indices for nearby LEDs. For example, one section of your object may have ten LEDs close together. Label those #1 through #10. Grouping nearby LEDs with nearby index numbers tends to simplify the sequence programming later on.

Each LED is indexed directly to its own column within pattern rows. Column #1 determines the state of LED #1, column #2 the state of LED #2, and so on. The following example illustrates pattern rows with different states for indexed LEDs. Read this sequence from top-to-bottom, not left-to-right:

```
        columns (each column is indexed against one LED)

row #1: 0000000000000000000000000000000000000000000000000000000000000000
```

```
row #2: 10000000000000000000000000000000000000000000000000000000000000000
row #3: 11111111000000000000000000000000000000000000000000000000000000000
row #4: 11111111111111111111111111111111000000000000000000000000000000000
row #5: 11111111111111111111111111111111111111111111111111111111100000000
row #6: 11111111111111111111111111111111111111111111111111111111111111111
```

In the first row all LEDs are off. In the second row we turn the LED in column #1 on. In the third row we keep LED #1 on, and also turn on #2 through #8. The rest of the rows continue to turn on more and more LEDs until by the final row they are all on.

Control Rows

Control rows consist of special codes and parameters. For example a control row might include a brightness code with a parameter specifying how bright to make the LEDs.

Here is an example of pattern rows intermixed with control rows:

```
b15
t00100
1010101010101010101010101010101010101010101010101010101010101010
0101010101010101010101010101010101010101010101010101010101010101
b08
1010101010101010101010101010101010101010101010101010101010101010
0101010101010101010101010101010101010101010101010101010101010101
t02000
1010101010101010101010101010101010101010101010101010101010101010
0101010101010101010101010101010101010101010101010101010101010101
```

At the beginning of this sequence we use **b15** to turn the brightness up to full. We also establish the tick time (how long each row is played for) with the **t00100** control code. This establishes 100 milliseconds as the duration each row should be active. Note that it is possible to use multiple control codes back-to-back on adjacent lines with no effect on the pattern row render time.

We play a couple of pattern rows and then change the brightness to about half with the **b08** code. Two rows later we use the tick control code again, this time using **t02000** to set each row's duration to a full two seconds.

In order to maximize compatibility between text editors across different operating systems

(Windows, Mac, Linux) line endings may be CR (carriage return, ASCII code 13) LF (line feed, ASCII code 10), or both (in that order, CR/LF).

Blank lines are ignored. All other lines are interpreted as either pattern rows or control rows.

The specifications for all control codes and parameters are below.

File Names

Airfield will attempt to load the sequence file named `airfield.led`

Control Codes

Tick

Example: `t12345`

Tick controls how long each row of pattern data is active. It is specified in milliseconds. Legal values range from 00000 (as fast as possible) to 99999 (99.999 seconds). A typical value is 00100 (which plays 10 rows per second). Due to hardware constraints the fastest usable tick value is about 00003.

Tick may be modified by the speed control code (see Speed).

Comment

Example: `// this is a comment`

Comments are ignored by Airfield but are useful to annotate pattern data. For example:

```
// Turn all lights on
1111111111111111111111111111111111111111111111111111111111111111
// Turn all lights off
0000000000000000000000000000000000000000000000000000000000000000
// Turn odd lights on and even lights off
1010101010101010101010101010101010101010101010101010101010101010
```

Brightness

Example: `b12`

Brightness determines the light intensity for all 64 LEDs in the main bank. To perform a brightness control operation, include the character 'b' followed by two digits. For example, 'b12'.

Legal values for brightness are 00 (dim) to 15 (bright).

Brightness may be scaled with the dim control code (see Dim).

General Illumination (GI)

Example: `g123`

Although Airfield includes 64 individually controllable LEDs there may be circumstances where additional LEDs are needed. The general illumination bank supplies an additional 8 LEDs with one limitation: they must all share the same brightness level.

The brightness of the general illumination bank is controlled with this code. Legal values range from 000 (dim) to 255 (full). For example, g128 would mean 50% brightness (128 out of 256 levels).

The GI may be modified by the dim control code (see Dim).

Speed

Example: `s123`

Speed controls the rate of playback. It is a percentage with legal values being 000 (halt), to 100 (play at 100% i.e. normal speed), to 999 (play at 999% i.e. about 10 times as fast).

Speed control is independent of tick time. Users generally should use tick time to control relative timing rates for pattern data, and use this speed control as a global mechanism for speeding up or slowing down the entire sequence without having to recode all individual tick values. Speed is typically only specified once at the beginning of a sequence file.

The default value for speed is 100 (%).

Dim

Example: `d012`

Dim controls the brightness of LEDs. It is a percentage with legal values being 000 (very dim but not off), to 100 (full brightness).

Note that dim control is independent of the brightness control. Use the brightness control code instead to set relative intensity levels of LEDs.

Much like the speed control code, dim is typically only specified once at the beginning of a sequence file. Used in this way it will darken everything uniformly without the need to go back and modify all individual brightness code values. This is useful for example if the playfield is mounted on a wall in a dark room where the LED brightness needs to be reduced overall, while still allowing there to be a noticeable difference between bright LEDs and dim LEDs.

The default value for dim is 100 (%).

Loop

Example: `ls`, `le123`

Loop allows for playing a section of the sequence file more than once. The easiest way to understand it is to see an example:

```
00000000000000000000000000000000000000000000000000000000
10000000000000000000000000000000000000000000000000000000
01000000000000000000000000000000000000000000000000000000
00100000000000000000000000000000000000000000000000000000
ls
00010000000000000000000000000000000000000000000000000000
00001000000000000000000000000000000000000000000000000000
00000100000000000000000000000000000000000000000000000000
00000010000000000000000000000000000000000000000000000000
le005
00000001000000000000000000000000000000000000000000000000
00000000100000000000000000000000000000000000000000000000
```

Here we have four rows of regular pattern data, followed by an 'ls' (loop start). This is a bookmarked location into the sequence file which will be looped back to.

Then we have 4 more rows of pattern data.

Then we reach the 'le005': loop end, repeat 5 times. When reaching this point Airfield will set up an internal counter to track how many times it has looped so far. It will jump back to the bookmark and play forward. Every time it reaches the loop end the counter is decremented. When the loop counter reaches 0 (which in this example is after 5 loops) Airfield escapes the loop and continues playing the rest of the pattern data.

Loops may not be "nested" (yet). Please contact us if you would like this feature to be implemented.

Reference

Control Code	**Function**	**Valid Parameters**
t00100	tick time	00000 (fast) to 99999 (slow)
//	comment	any string
b15	brightness	0 (dim) to 15 (full)
g255	GI brightness	0 (dim) to 255 (full)
s100	speed	0 (halt) to 999 (play 9.99 times faster)
d100	dim	0 (dim) to 100 (full)
ls	loop start	none
le123	loop end	0 (do not loop) to 999 (loop 999 times)