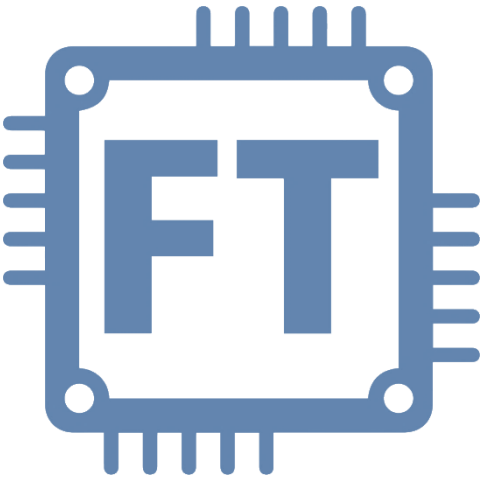
Pyroterra Lighttoys

presents

FUTURE CHIP TECHNOLOGY



Data format handbook

v1.0 (ENG)

Introduction

Dear customer, we are happy to share the revolutionary FT range of Lighttoys products with you. In this guide you will learn how to create a programmed sequence (show) for your Lighttoys FT product. You will understand syntax of the data format and we will also share some handy tips with you...

LSF file format

Each blinking sequence for your Lighttoys FT product is defined and stored in single lsf file (Lighttoys show file). Open a new blank document in your favorite plain text editor (Notepad, TextEdit, etc.…) and save it with ***.lsf** file extension. For example:

```
my_first_show.lsf
```

Sequence name

You can define the name of the sequence on the first line of your lsf file. It shouldn't start with number or # symbol (which is used for comments). Example:

```
MY_GREAT_SHOW_01
```

First 12 characters from the sequence name will also be stored into your FT device when the show is uploaded. In this example: "**MY_GREAT_SHO**"

Example code sequence

```
MY_GREAT_SHOW_01

0s: setrgb A 1m > 255,255,255   #slowly fade A to white

0s: loop 1m
     0s: setrgb B 2s > 255,0,0   #rainbow effect on B
     2s: setrgb B 2s > 0,255,0
     4s: setrgb B 2s > 0,0,255
     6s: end

1m: setrgb A 0s > 255,0,0   #set A to red color
1m: setrgb B 0s > 0,0,255   #set B to blue color

1m: setrgb AB 30s > 0,0,0   #slow fade out

90s: end
```

File structure

- On each line of your lsf file you can write one of the following commands: **`SETRGB`**, **`LOOP`**, **`END`**, **`FORK`** or comment text
- Commands are not case sensitive (SETRGB = setrgb = SetRGB)
- Command parameters are separated by “whitespace” characters (Space, Tab) in any arbitrary amount
- Comment text must begin with **#** symbol
- blank lines are ignored

Time units

Timing parameters are by default in milliseconds. You can define different time units by appending following letters to the parameters (without space):

- ‘**h**’ – hours
- ‘**m**’ – minutes
- ‘**s**’ – seconds
- ‘**ms**’ - milliseconds

Examples: **`17h, 62m, 32s, 643ms`**

Composite units are not supported (for example “1h20m”), you will need to write “80m” instead.

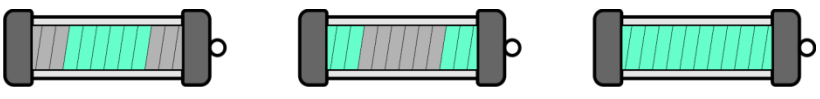
One millisecond is the shortest amount of time that can be used and the longest allowed show length is 2^{31}= 2,147,483,648ms. This should suffice for 25 days long show ;-)

Light segments

Each Lighttoys FT product has at least one controllable light segment (A), some have the lighting area split into two segments (AB).

For FT devices with 2 segments, you can program both segments independently:

Zebra Poi FT example:



A B A+B

single light segment:

LED Fans FT, LED Buugeng FT

2 light segments:

Zebra Poi FT, Fantastick FT, Zebra Stick FT

SETRGB command

The SETRGB command is used to set the next RGB color value for light segment A, B or both A and B. The color value can be set immediately or gradually faded over time.

The syntax is as follows:

time : SETRGB channel dTime > R,G,B

time - specific moment in time, when the command will be executed

channel - selects which light segment on your Lighttoys FT device should be affected by the command. Possible value: **A**, **B** or **AB**.

dTime - time value defining the speed of color change (linear fade) to the new RGB values. When you input **0**, the color change is instantaneous.

R,G,B - new Red Green Blue color values to which the current color setting will change. Possible values: **0-255**, **0-255**, **0-255**. 255 stands for full brightness, 0 stands for complete darkness of the specific color.

Examples:

`0: setrgb AB 0 > 255,0,0`	start with all light segments glowing red
`5s: setrgb A 2s > 255,255,0`	at 5 sec mark, start fading segment A to yellow color
`6s: setrgb B 100 > 0,0,0`	fade out (turn off) segment B in 100 milliseconds

END command

Every lsf file must finish with END command, indicating the show has ended. When the command is executed, all color outputs will turn off and the Lighttoys FT device will return to normal operation.

The syntax is as follows:

time : END

time - specific moment in time, when the END command will be executed

Example: `45s: end`

LOOP command

The LOOP command is used to easily create repeating light patterns - like pulses, flashes and strobe effects. Each LOOP must finish with END command.

The syntax is as follows:

time : LOOP LoopTime

time - specific moment in time, when the LOOP command will be executed

LoopTime – commands inside the LOOP sequence will be repeated for this time duration

Example sequence (at 10 second mark, start slow red color pulses for 1 minute):

```
RED_PULSES
10s: loop 1m
     0s: setrgb AB 1s > 255,0,0
     1s: setrgb AB 1s > 0,0,0
     2s: end
70s: end
```

Notice

- All time marks inside the LOOP command are written relative to the start time of the loop.
- The LOOP commands can be nested inside of existing loops, maximum 10 times (the body of the code itself counts as the top loop).
- If LOOP time and END time mark are not commensurable, the last repetition of the loop will stop midway and the color outputs will stay set to their respective values at that moment.
- Two LOOP commands can overlap each other in time, in that case FORK command must be used (see section ***Advanced stuff***).

Note: The commands in your lsf files must be in chronological order - time marks on each line must be greater than or equal to the most recent time mark (you cannot change something set already in the past).

This code snippet is wrong:

```
…
15s: setrgb A 0s > 0,255,0
10s: setrgb B 2s > 0,178,120
…
```

Example sequence

In this section we will show one complete lsf file example, utilizing all commands explained so far:

```
MY_SUPER_SHOW

  0: setrgb AB 0 > 0,0,0              #make sure everything is off
 3s: setrgb A 7s > 0,127,0            #fade channel A to middle green

10s: loop 25s                         #loop for 25 seconds
     0s: setrgb B 1s > 0,0,255        #start blue fade on channel B
     1s: loop 3s                      #loop green pulse 10 times
            0: setrgb A 50 > 0,255,0
           50: setrgb A 100 > 0,0,0
          150: setrgb A 100 > 0,255,0
          250: setrgb A 50 > 0,127,0
          300: end
     4s: setrgb B 0 > 0,0,0           #turn off channel B
     4s: end                          #close the LOOP sequence

35s: setrgb AB 0 > 0,0,0              #turn off lights
40s: end                              #after 5sec return to standby
```

The above sequence represents a valid show sequence. When saved as ***.lsf** file, it can be uploaded into your Lighttoys FT product with the FT loader application and enjoyed ☺

Please consult our ***Lighttoys FT - loader application*** guide to learn how to upload your sequences.

We wish you many great moments
with Lighttoys FT!

Advanced stuff

In this section we will explain some advanced topics that are not mandatory for normal usage. If you are interested, delve in but be prepared that your head might hurt ☺

FORK command

The FORK command is used to start multiple parallel code “threads” (independent set of commands). This is useful when you want to execute two different blinking patterns on segments A and B simultaneously.

The syntax is as follows:

time : FORK @ThreadName

time - specific moment in time, when the FORK command will be executed

@ThreadName – name of the code thread that will be run

Each code thread must start with a name prefaced with @ symbol and finish with the END command. Example:

```
@MY_THREAD
5s: setrgb B 1s > 117,0,28
10s: end
```

Code threads must be written below your main code part. You can start the threads from anywhere in the main code part, using the FORK command. Example:

```
0: setrgb A 50 > 0,255,0
2s: fork @MY_THREAD
12s: end
```

Notice

- You can run up to 8 eight parallel threads concurrently.
- Each thread can access any lighting segment A, B or AB.
- If more threads try to set the segment color value at given time mark, then the thread which was started last will set the value.
- All threads must finish before the main code part ends.

The *.lsf file below shows a code sequence utilizing FORK commands and threads. In this example we are running two different blinking patterns on A and B simultaneously – a slow green pulse on channel B and a quick pulsing on channel A:

```
FORK_test           #main code part

0s: fork THREAD1
5s: fork THREAD2
10s: fork THREAD3

15s: end            #main code part ends here

@THREAD1
0s: loop 15s
         0s: setrgb B 1s > 0,255,0
         1s: setrgb B 1s > 0,0,0
         2s: end
15s: end

@THREAD2
0s: loop 2s
         0ms: setrgb A 100ms > 127,0,0
       100ms: setrgb A 100ms > 0,0,0
       200ms: end
2s: end

@THREAD3
0s: loop 2s
         0ms: setrgb A 100ms > 0,0,127
       100ms: setrgb A 100ms > 0,0,0
       200ms: end
2s: end
```