Pattern Table Documentation

1. Overview

This document defines the binary file (.dat) format used by the Pattern Table Pipeline.
The format is designed for efficient playback of time-based LED animations with optical fibers (OF) and LED strips.

The system consists of two binary files:

File	Purpose
control.dat	Global and hardware configuration
frame.dat	Sequential frame-based animation data

2. Data Format

Item	Specification
File type	Binary (.dat)
Byte order	Little endian
Numeric types	Unsigned integers only (uint8, uint32)
Color format	RGB888 (GRB order)

- Channel : we define channel as a single controllable lighting unit, which can be either an Optical Fiber or an LED strip
  - Optical Fiber Channel : one optical fiber corresponds to exactly one channel and contains a single color value defined by one GRB triplet
  - LED Strip Channel : one strip corresponds to one channel, but contains multiple color values, where each LED on the strip has its own GRB triplet

- Due to hardware data storage requirements, our color values are stored in GRB order instead of RGB

3. “control.dat” Specification

3.1 Purpose

The “control.dat” defines the global configuration parameters. All frames in “frame.dat” must follow this configuration.

3.2 File Layout

All fields are 1 byte/uint8 unless otherwise specified.

Offset	Element	Size	Description
0,1	Version	2 byte (uint8 * 2) little-endian byte order	The version of current pattern table file
2	OF_num	1 byte (uint8)	Number of optical fibers
3	Strip_num	1 byte (uint8)	Number of LED strips
4	LED_num[0]	1 byte (uint8)	LED count of strip 0
5	LED_num[1]	1 byte (uint8)	LED count of strip 1
…	…	…	…
4+Strip_num	LED_num[Strip_num]	1 byte (uint8)	LED count of last strip
4+Strip_num +1	Frame_num	4 byte (uint32) little-endian byte order	Number of total frame in this version
4+Strip_num +2	time_stamp[0]	4 byte (uint32) little-endian byte order	Start time of frame 0
4+Strip_num +3	time_stamp[1]	4 byte (uint32) little-endian byte order	Start time of frame 1
…	…	…	…
6+Strip_num +Frame_num	time_stamp[Frame_num]	4 byte (uint32) little-endian byte order	Start time of last frame

Version : The version is stored as two separate bytes (uint8) in little-endian order

- Byte 0 = major version (main version number)
- Byte 1 = minor version (sub-version number)

For example :

- version 0.1 is save as Byte 0 = 0, Byte 1 = 1
- version 2.5 is save as Byte 0 = 2, Byte 1 = 5

3.3 Field Constraints

Field	Constraint
OF_num	0~40
Strip_num	0~8
LED_num	0~100

4. “frame.dat” Specification

4.1 Overall Structure

“frame.dat” is a frame sequence with no global header

Offset	Element	Size	Description
0,1	Version	2 byte (uint8 * 2) little-endian byte order	The version of current pattern table file
2~2+frame_size	Frame 1	frame_size byte (uint8)	Data of first frame
2+frame_size ~2+2*frame_size	Frame 2	frame_size byte (uint8)	Data of second frame
…	…	…	…
2+(n-1)*Strip_num ~2+n*Strip_num	Frame n	frame_size byte (uint8)	Data of frame n

Frame count is determined by end-of-file, no padding or delimiter exists between frames

The size of frame_size consist all data below
Each frame consists of (in order) :

Field	Size	Type	Description
start_time	4 byte	uint32 little-endian byte order	Frame start timestamp
fade	1 byte	uint8	Fade enable flag (True=1 / False=0)
OF GRB data	OF_num * 3 byte	uint8	Number of optical fibers
LED GRB data	Σ(LED_num) * 3 byte	uint8	Number of LED strips
checksum	4 byte	uint32 little-endian byte order	Checkpoint of a total frame

Fade definition : we use 1 bytes to determine whether a color transition is applied between the current frame and the next frame

- If fade is set to true in frame i, the lighting output shall smoothly transition from the colors defined in frame i to the colors defined in frame i+1.
- If fade is set to false, the colors of frame i+1 shall be applied immediately without transition.

checksum definition : we use 1 bytes to save a checkpoint for a frame, define Checksum = (Σall bytes in frame) mod 2^32, where all bytes include (start_time, fade, OF GRB data, LED GRB data)

4.2 GRB Data Layout

OF GRB data

OF[0].G	OF[0].R	OF[0].B	OF[1].G	OF[1].R	OF[1].B	…	OF[n].G	OF[n].R	OF[n].B

- 1 byte (uint8, 0~255) for each R/G/B
- Total size = OF_num × 3 byte
- Stored sequentially for i = 0 … n = OF_num – 1

LED GRB Data

First ordered by strip index, then by LED index

LED[0][0].G	LED[0][0].R	LED[0][0].B	LED[0][1].G	LED[0][1].R	LED[0][1].B	…	LED[0][n_1].G	LED[0][n_1].R	LED[0][n_1].B
LED[1][0].G	LED[1][0].R	LED[1][0].B	LED[1][1].G	LED[1][1].R	LED[1][1].B	…	LED[1][n_2].G	LED[1][n_2].R	LED[1][n_2].B
…	…	…	…	…	…		…	…	…
LED[m][0].G	LED[m][0].R	LED[m][0].B	LED[m][1].G	LED[m][1].R	LED[m][1].B	…	LED[m][n_m].G	LED[m][n_m].R	LED[m][n_m].B

- 1 byte (uint8, 0~255) for each G, R, B
- Total size = Σ(LED_num) * 3 byte
- LED[i][j] Stored sequentially for i = 0 … m = Strip_num – 1, j = 0 … n_i = LED_num[i]
  - For LED[i][j], the index j starts from 0 at the LED physically closest to the ESP32 and increases along the strip away from ESP32
  - For example : LED[i][j] stand for the jth LED on ith strip

5. Consistency Rules

- “frame.dat” must be parsed using the corresponding “control.dat”
- OF_num, Strip_num, and LED_num must match exactly in each frame of frame.dat and control.dat
- Frames are expected to be ordered by non-decreasing start_time
- Behavior is undefined if frames are not sorted by time

6.Revision History

Revision	Date	descripition
0.1	1/19	testing version of the documentation
1.0	1/21	Same as v0.1. pass the API test on ESP32