NOTE: THESE WERE MOSTLY TESTED USING HANTEI AS REFERENCE, I RARELY EVER PUT THESE IN GAME

Thanks to @Sosfiro#9513 on discord for giving me a documentation file to reference along with my findings

Definitions: these aren't official definitions, these are just what I call them
Sprite: the frame number being called
Object: an item being called by a sprite
Effect: texture being used by an object
Shape: the type of effect, for example, “Ring”, “Sphere”, ect.
Page: the dds file used for texture mapping
Block: a section of the file where arguments goes

Theory on how interpolation works: My guess is that interpolation isn’t interpolating the drawing, but instead the object. When an object is created, the game will interpolate all its parameters into the next sprite, with the exceptions of PRID and PRFL(?), but it will only interpolate object to object, which is set in PRST, so the game won’t interpolate object 2 to object 3.

Things to note:

Bytes are put in reversed, for example 00 9A 1C E9 would go in as E9 1C 9A 00

_STR: start of the pat file

SPRITE/OBJECT ARGUMENTS (goes at top of file)

P_ST: starts a sprite block (int32)
Bytes 1-4 defines the sprite number

PANA: names a sprite block
First byte is the name length (uint8)
Rest of the bytes are the name, size must equal the length set by first byte (char)

PRST: starts an object block, goes inside sprite block, multiple can be there (uint32)
Note: For Uni, Dfci, and Bbtag, there is a limit of 40 blocks, and for MBTL there is a limit of 80.
Bytes 1-4 defines the object number

PRXY: where the object will be placed in the sprite (int32)
Bytes 1-4 is the X placement
Bytes 5-8 is the Y placement

PRFL: object filter(?)
First byte is the filter type(?), 1 is bilinear

PRA3: rotation values all as singles (float32)
Note: Rotation center is the effect center
Bytes 1-4 are always 00 00 00 00
Bytes 5-8 are the rotation along the X axis
Bytes 9-12 are the rotation along the Y axis
Bytes 13-16 are the rotation along the Z axis

PRZM: Scale values all as singles (float32)
Bytes 1-4 is the X scale
Bytes 5-8 is the Y scale

PRAL: Additive Blend (bool)
First byte is true or false

PRCL: color of object, FF is default color (uint8)
First byte is the blue color in hex
Second byte is the green color in hex
Third byte is the red color in hex
Fourth byte is the alpha in hex

PRSP: color overlay on object (uint8)
First byte is the blue overlay in hex
Second byte is the green overlay in hex
Third byte is the red overlay in hex
Fourth byte is unused

PRID: what effect the object will use (uint32)
Bytes 1-4 are the number set by PPST of the effect you want to use

PRED: ends an object block

P_ED: ends a sprite block

EFFECT ARGUMENTS (goes in middle of file)

PPST: starts a new effect block (uint32)
Bytes 1-4 is the effect number, cannot go above 1000, aka E8 03

PPNA: name of the effect
First byte is name length (uint8)
Rest is the name, size must equal the length set (char)

PPCC: the coordinates of where the center of the effect is based on uv (int32)

Bytes 1-4 is the y placement
Bytes 5-8 is the x placement

PPUV: the coordinates of the UV for the effect on the page (int32)
Note: The UV placement is the width/height of the page divided by 256, so the placement will scale when you change the page size. For example, 3A would be 464 if the page was 2048 pixels long/wide, but it would be 928 if the page was 4096 pixels long/wide. X and width use the width of the page while y and height use the height
Also Note: If the placement goes off the image, it will loop
Bytes 1-4 is the x start of the UV (0 is the top left corner)
Bytes 5-8 is the y start of the UV (0 is the top left corner, positive numbers go down)
Bytes 9-12 is the width of the UV
Bytes 13-16 is the height of the UV

PPSS: the width and height of how the effect will be placed in the sprite (int32)
Bytes 1-4 is the width
Bytes 5-8 is the height

PPTP: the page the effect will use (uint32)
Bytes 1-4 is the page number set in PGST

PPTE: no noticeable changes, the notes I was given say texture aspect ratio, but I'm not sure (int16)

PPPA: number on palette to overlay color (int32)
Bytes 1-4 is the number on the palette to overlay it onto
Note: if you go above 256, it will just loop back around

PPPP: Shape to use (int32)
Bytes 1-4 is the index of the shape to use. If it goes above the amount of shapes set it'll just use the default.

PPED: ends the current effect block

PAGE ARGUMENTS (goes at bottom of file)

PGST: starts a block for a page (int32)
Bytes 1-4 is the page number

PGNM: name of the page (string32)
The name must be 32 bytes long, any more or less will crash the game

PGTE: size of the page, but shortened (uint16)
Bytes 1 and 2 are the width, but only the first 2 bytes

Bytes 3 and 4 are the height, but only the first 2 bytes

PGT2: miscellaneous page setup
Bytes 1-4 are the file size + 16 bytes, so if the file was 00 00 43 83 bytes, I would put 93 43 00 00, note that if the file is uncompressed put the original file size here (uint32)
Bytes 5-8 are the width of the image, all 4 bytes this time (uint32)
Bytes 9-12 are the height of the image, all 4 bytes this time (uint32)
Bytes 13-16 is the dds type, DXT5 or DXT1 are some examples, I assume that it will work with any type, but FB only uses DXT5 and 1 from what I've seen (string4)
Bytes 17-24 are "DDPF_FOURCC flags" according to the notes I was given, I didn't mess with them though and it works fine
Bytes 25-28 are always gonna be FF 0B 10 01
Bytes 29-32 are always gonna be 00 00 00 00
Bytes 33-36 are the file size without the extra 16 bytes (uint32)
Bytes 37-40 is the aspect ratio of the image + 128 or 0x80
The rest of the bytes until you reach PGED is just the image. There is no limit on this

PGED: ends the block for a page

_END: end of the pat file