chronome serial protocol

owen vallis – [contact@flipmu.com](mailto:contact@flipmu.com)

//based off of the monome serial protocol series 256/128/64

//by brian crabtree

revision: 004

from device:

message id: (1) pressure

bytes: 3

format: iiii.xxx .yyy..dd dddddddd

i (message id) = 1

x (x value) = 0-7 (three bits)

y (y value) = 0-7 (three bits)

d (data value) = 0 – 1024 (ten bits)

decode: id match: byte 0 & 0xf0 == 16

x: byte 0 & 0x0f

y: byte 1 >> 4

d: uint16\_t val = ((byte 1 & 0x0f) << 8) | byte 2

to device:

message id: (1) rgb\_led\_on

bytes: 2

format: 1...iiii 0xxx0yyy

i (message id) = 1

x (x value) = 0-7 (three bits)

y (y value) = 0-7 (three bits)

encode: byte 0 = id | 0x80 = 129

byte 1 = ((x << 4) | y) & 0x7f

message id: (2) rgb\_led\_off

bytes: 2

format: 1...iiii xxxxyyyy

i (message id) = 2

x (x value) = 0-7 (three bits)

y (y value) = 0-7 (three bits)

encode: byte 0 = id | 0x80 = 130

byte 1 = ((x << 4) | y) & 0x7f

message id: (3) rgb\_led\_color

bytes: 5

format: 1...iiii 0xxx0yyy 0rrrrrrr 0ggggggg 0bbbbbbb

i (message id) = 3

x (x value) = 0-7 (three bits)

y (y value) = 0-7 (three bits)

r (red value) = 0 - 127 (7 bits)

g (green value) = 0 - 127 (7 bits)

b (blue value) = 0 - 127 (7 bits)

encode: byte 0 = id | 0x80 = 131

byte 1 = ((x << 4) | y) & 0x7f

byte 2 = (r & 0x7f)

byte 3 = (g & 0x7f)

byte 4 = (b & 0x7f)

message id: (4) rgb\_led\_all\_state

bytes: 1

format: 1..siiii

i (message id) = 4

s (test state) = 0-1

encode: byte 0 = id | 0x80 | (s << 4) = 132 | (s << 4)

message id: (5) rgb\_row

bytes: 2

format: 1yyyiiii aaaaaaaa

i (message id) = 5

y (row to update) = 0-7 (three bits)

a (row data 0-7) = 0-255 (eight bits)

encode: byte 0 = id | 0x80 | (y << 4) = 133 | (y << 4)

byte 1 = a (row data 0-7)

message id: (6) rgb\_col

bytes: 2

format: 1xxxiiii aaaaaaaa

i (message id) = 6

x (col to update) = 0-7 (three bits)

a (row data 0-7) = 0-255 (eight bits)

encode: byte 0 = id | 0x80 | (x << 4) = 134 | (x << 4)

byte 1 = a (row data 0-7)