PFx Brick
Scripting Language
Cheatsheet

Basics

As of ICD version **3.37** (and **PFx Brick** firmware versions **1.40**+), the ability to execute complex actions and behaviours defined in script files was added. Script files are simple, human readable text files stored in the **PFx Brick** file system. These files conform to a simple script language syntax summarized in this cheatsheet.

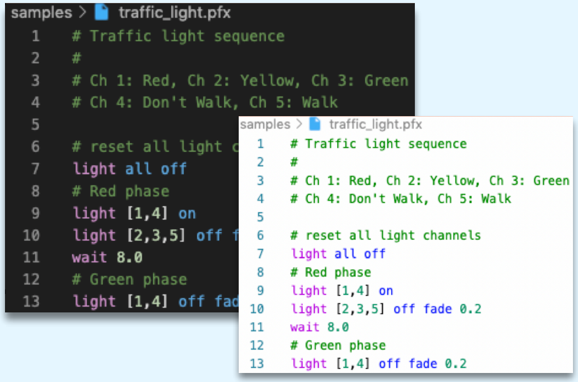
- Scripts are ASCII text files stored in the **PFx Brick** file system.
- Scripts execute one at a time. Executing another script will terminate the current script and start the new one.
- Script execution is sequential line-by-line from the start of the file to the end. At the end, the script will either stop or repeat if a repeat command is the last line.

Editing Scrips

Edit script files with the text editor of your choice (e.g. Notepad on Windows, TextEdit on macOS). PFx Language extensions with syntax highlighting for Visual Studio Code and Notepad++ can be found here:

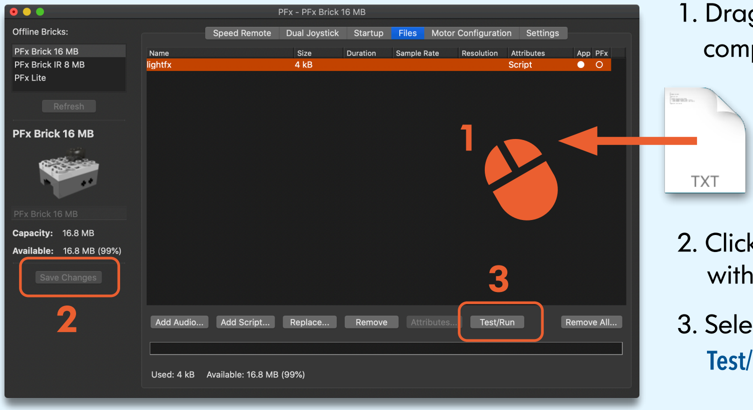
https://github.com/fx-bricks/pfx-brick-vscode

https://github.com/Brickelectronic/pfxbrick_notepadplusplus


samples > traffic_light.pfx

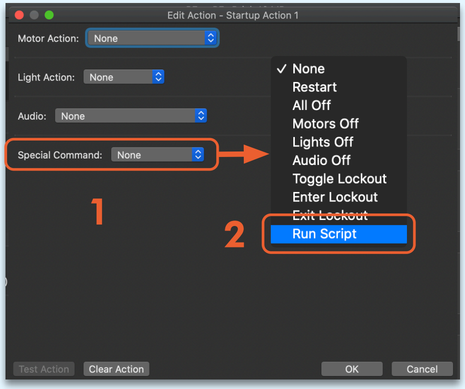
```
# Traffic light sequence
#
# Ch 1: Red, Ch 2: Yellow, Ch 3: Green
# Ch 4: Don't Walk, Ch 5: Walk

# reset all light channels
light all off
# Red phase
light [1,4] on
light [2,3,5] off fade 0.2
wait 8.0
# Green phase
light [1,4] off fade 0.2
```

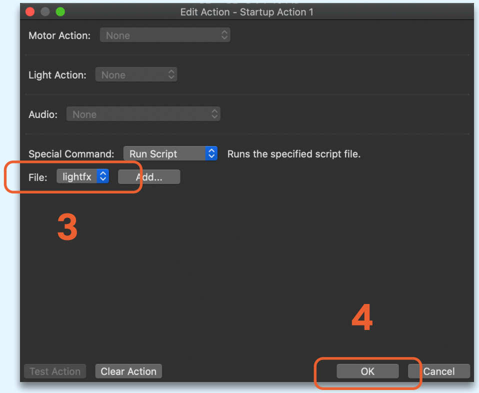
Loading Scripts



1. Drag and drop script file from computer to PFx App Files window
2. Click Save Changes to update **PFx Brick** with new script file
3. Select script file from file list and click Test/Run button to run script

Configuring Scripts as Actions





1. Select any Remote or Startup action to edit and select Special Command
2. Select Run Script from the drop down list
3. Select a script file from File drop down list
4. Click Ok to confirm

Automatic *startup.pfx* Script

As of ICD version **3.38** (and **PFx Brick** firmware versions **1.50**+), a special reserved filename "startup.pfx" can be used for a script that you want to automatically launch after power on or reset. This can be useful for setting special configuration or complex startup action sequences.

ICD refers to the Interface Control Document; a document intended for the developer community to build software Apps compatible with the PFx Brick visit our GitHub repo here for more details: https://github.com/fx-bricks/pfx-brick-dev

Syntax Basics

Comments

start with # or //

```
# Valid comment
// Another valid comment
light 1 on  # not a valid comment
```

⚠ cannot appear on a line with a command

Numbers

Decimal values

`0 127 -55 0.010 35.75 -90.5`

Hexadecimal values

`0x0 0xABCD 0x32`

Lists

`[0, 5, 8]` enclose in [ ] comma "," separated

Strings

`"This is a string"` enclose in " " double quotes

Keyword Commands

Keyword commands appear at the beginning of a line followed by other keywords and values depending on the command syntax.

```
event      motor      set       wait
ir         repeat     sound
light      run        stop
```

Other Keywords

Other keywords are used in support of commands as qualifiers or arguments.

```
acc        config      invert    servo
all        connect     joy       shutdown
bass       decel       left      speed
beep       disconnect  loop      startup
ble        down        long      thr
brake      fade        off       treble
bright     file        on        up
button     flash       play      usb
ch         fx          rate      vol
changedir  gated       right
```

⚠ all keywords must be separated by whitespace

Run/Stop Commands

execute another script

`run file` *file* : file ID number or string

```
run "traffic_light"
```

stop execution

`stop`

Wait Command

wait for time delay

`wait time` *time* : 0.05 – 30000 sec

```
# wait for 3.5 seconds
wait 3.5
```

wait for sound file playback end

`wait sound file` *file* : file ID number or string

```
# wait for sound file 2 to stop
wait sound 2
# wait for "Siren1.wav" to stop
wait sound "Siren1.wav"
```

wait for IR event

`wait ir parameters`

parameters : any combination of:
`joy` : joystick remote
`speed` : speed control remote
`up`, `down`, `left`, `right`, `button` : remote actions
`ch` *value* : IR channel (1, 2, 3 ,4)

```
# wait for the joystick left control
# pushed up on any IR channel
wait ir joy left up
# wait for the left button on speed remote
# on IR channel 4
wait ir speed ch 4 left button
```

wait for pushbutton event (from touchLAB)

`wait button`

```
# wait for button press
wait button
```

Repeat Command

repeat current script from start

`repeat` restarts the script from the beginning

repeat enclosed code block (with nesting)

```
repeat count {
. . .
}
```

repeats a block of code *count* number of times enclosed in braces { }

⚠ opening { must be on same line as **repeat**
closing } must be on a line by itself

```
# repeat this code 5 times
repeat 5 {
  light 1 on
  wait 2.0
  light 1 off
  wait 10
}
```

```
# nested repeats
repeat 5 {
  light 1 fx 1
  repeat 3 {
    light 2 fx 1
    wait 2.0
  }
}
```

IR Command

`ir on` activate IR sensor

`ir off` disables IR sensor

Sound Command

```
sound command
```

Commands

simple playback

```
play file
```

```
sound play 3
sound play “Siren1.wav”
```

continously repeated playback

```
play file repeat
```

```
sound play 3 repeat
sound play “Siren1.wav” repeat
```

playback a number of times

```
play file loop value
```

value : 1 – 255

```
sound play 3 loop 5
sound play “Siren1.wav” loop 5
```

random playback

```
play file random probability
```

0 – 3
often *rare*

```
sound play 3 random 0
sound play “Siren1.wav” random 2
```

stop playback

```
stop file
```

file : file ID number, string, or all

```
sound stop 3
sound stop “Siren1.wav”
sound stop all
```

set volume

```
vol value
```

value : 0 – 255

```
sound vol 0
```

set bass / treble

```
bass value   treble value
```

value : -20 – +20

```
sound bass 3
sound treble -10
```

short beep sound

```
beep
```

```
sound beep
```

Light Command

```
light channels commands
```

Channels

`light 1`	single channel
`light [2, 4, 5]`	multiple channels
`light all`	all channels

Commands

simple on/off

```
on  off
```

```
light 1 on
light [1, 2, 4] off
light all off
```

flashing light

```
flash ontime offtime
```

ontime, *offtime* : 0.05 – 60 seconds

offtime is optional; if omitted *ontime*=*offtime*

```
# on for 0.5 sec, off for 0.5 sec
light 1 flash 0.5
# on for 0.1 sec, off for 0.4 sec
light [1, 2, 4] flash 0.1 0.4
```

optional fade

```
fade time
```

time : 0 – 10 seconds

combine with `on`, `off`, & `flash` commands

```
# fade channel 1 on
light 1 on fade 0.5
# fade channels 3, 5 off
light [3, 5] off fade 1.0
# 1.5 sec flash with 0.2 sec fade
light 8 flash 1.5 fade 0.2
```

set brightness

```
bright value
```

value : 0 – 255

```
# channel brightness 10
light 1 bright 10
# channels 1,2,4 brightness 255
light [1, 2, 4] bright 255
```

Motor Command

```
motor channels commands
```

Channels

`motor a`	single channel
`motor [a, b]`	multiple channels
`motor all`	all channels

Commands

set speed

```
speed value
```

value : -255 – +255

```
motor a speed -50
motor all speed 100
```

stop

```
stop
```

```
motor b stop
motor all stop
```

servo angle

```
servo value
```

value : -90 – +90

```
motor a servo -90
```

optional acceleration/deceleration momentum

```
acc rate
```

rate : 0 – 15

combine with `speed` & `stop` commands

```
# accelate motor b to half speed gradually
motor b speed 128 acc 12
# stop motor a with gentle slow down
motor a stop acc 8
```

Event Command

```
event event {
. . .
}
```

stores actions enclosed in braces { } associated to specified events

⚠ opening { must be on same line as **event**
closing } must be on a line by itself

Actions specified between braces are **not** performed during script execution. They are stored in the PFx Brick's non-volatile memory and executed when the desired event occurs.

Multiple actions can be associated with an event. However, only **one** of each type can be stored, similar to how actions are configured in the PFx App.

⚠ The follow actions are not supported with **event**

```
sound bass
sound treble
sound beep
wait
set
repeat
light flash
```

⚠ **event** cannot be used inside a **repeat** block

Events

event can be specified in the following ways:

Event Numeric Address

`event address {` *address* : 0x0 – 0x7C

```
# run a script file with startup event 1
event 0x3C {
  run "traffic_lights.txt"
}
```

consult the ICD documentation for event LUT address details

Startup Events

`event startup id {` *id* : 1 – 8

```
# store sound action to startup event 3
event startup 3 {
  sound play "Siren1.wav"
}
```

IR Event

`event ir parameters {`

parameters : any combination of:
`joy` : joystick remote
`speed` : speed control remote
`up`, `down`, `left`, `right`, `button` : remote actions
`ch` *value* : IR channel (1, 2, 3 ,4)

```
# store sound action when joystick
# left stick is pushed up on ch 1
event ir joy left up ch 1 {
  sound play "Beep.wav"
}
```

Special Events

`event button {`	button press
`event button long {`	long button press
`event button down {`	button down
`event button up {`	button up
`event ble connect {`	Bluetooth host connect
`event ble disconnect {`	BLE host disconnected
`event usb connect {`	USB host connect
`event usb disconnect {`	USB host disconnected

```
# light action when button is pressed
event button {
  light all on
}
```

```
# define sound action and disable IR
# when a Bluetooth host establishes
# a connection
event ble connect {
  sound play "Welcome.wav"
  ir off
}
# define sound action and enable IR
# when a Bluetooth host disconnects
event ble disconnect {
  sound play "Goodbye.wav"
  ir on
}
```

The **event** command is useful for making configuration scripts which can be executed as a startup action. This is an alternative to using the PFx App to define the brick's configuration.

Set Command

variables

```
set var = value
```

There are 6x variable registers called:

`$A`, `$B`, `$C`, `$D`, `$E`, `$F` ⚠ must be uppercase

Both numeric and string values can be stored with the **set** command and can represent values in other commands

```
set $A = 50
motor a speed $A
```

```
set $A = 1.5
set $F = 0.1
light [1, 2] flash $A fade $F
set $B = “LongBeep.wav”
sound play $B
wait sound $B
```

configuration

```
set config var = value
```

⚠ *var* must be lowercase

values changed with **set config** replace existing configuration settings in the PFx Brick non-volatile memory

var	*value*	*ch*	**Description**
`set config bass = value`	−20 – +20		Audio bass
`set config treble = value`	−20 – +20		Audio treble
`set config vol = value`	0 – 255		Default volume
`set config bright ch = value`	0 – 255	1 – 8	Default brightness
`set config nc = value`	1 – 8		Notch count for indexed sound
`set config nb ch = value`	0 – 255	1 – 8	Speed between notch boundaries
`set config motor ch accel = value`	0 – 15	a, b	Motor acceleration
`set config motor ch decel = value`	0 – 15	a, b	Motor deceleration
`set config motor ch invert = value`	0 – 1	a, b	Invert motor polarity
`set config motor ch v0 = value`	0 – 255	a, b	Speed curve min speed
`set config motor ch v1 = value`	0 – 255	a, b	Speed curve mid speed
`set config motor ch v2 = value`	0 – 255	a, b	Speed curve max speed
`set config thr accel = value`	0 – 255		Thr for rapid accel sound
`set config thr decel = value`	0 – 255		Thr for rapid decel sound
`set config thr rate = value`	0 – 255		Thr accel braking sound
`set config thr speed = value`	0 – 255		Thr speed for braking sound

```
# set number of notch levels to 4
set config nc = 4
set config nb 1 = 60
# set default volume to 80
set config vol = 80
# set motor parameters
set config motor a accel = 4
set config motor a invert = 1
```

file attributes

```
set file type = file
```

Audio files used for motor indexed playback and gated playback based on motor speed can be assigned in a script.

⚠ *type* must be lowercase

file can be specified with either a numeric file ID or filename string.

type	*ch*	**Description**
`set file speed ch = file`	1 – 8	Sound loop at speed notch
`set file accel ch = file`	1 – 7	Sound loop for accel between notches
`set file decel ch = file`	1 – 7	Sound loop for decel between notches
`set file gated ch = file`	11–14, 21–24, 31–34, 41–44	Gated sound loops (4x for ea. notch 1–4)
`set file startup = file`		Sound played at startup
`set file shutdown = file`		Sound played for shutdown
`set file changedir = file`		Sound played due to direction change
`set file thr accel = file`		Sound played with rapid accel
`set file thr decel = file`		Sound played due to rapid decel
`set file brake on = file`		Sound played with brake application
`set file brake off = file`		Sound played at set-off from stop

file assignments made with **set file** will modify the file directory entry on PFx Brick file system

```
set file startup = “EngineStart.wav”
set file speed 1 = “IdleLoop1.wav”
set file accel 1 = 250
set file accel 12 = “ChuffLoop12.wav”
```

Fx Commands

Commands using the **fx** keyword are used for activating the advanced builtin effects

```
light channels fx id parameters
light all fx id parameters
motor channels fx id parameters
sound fx id file parameters
```

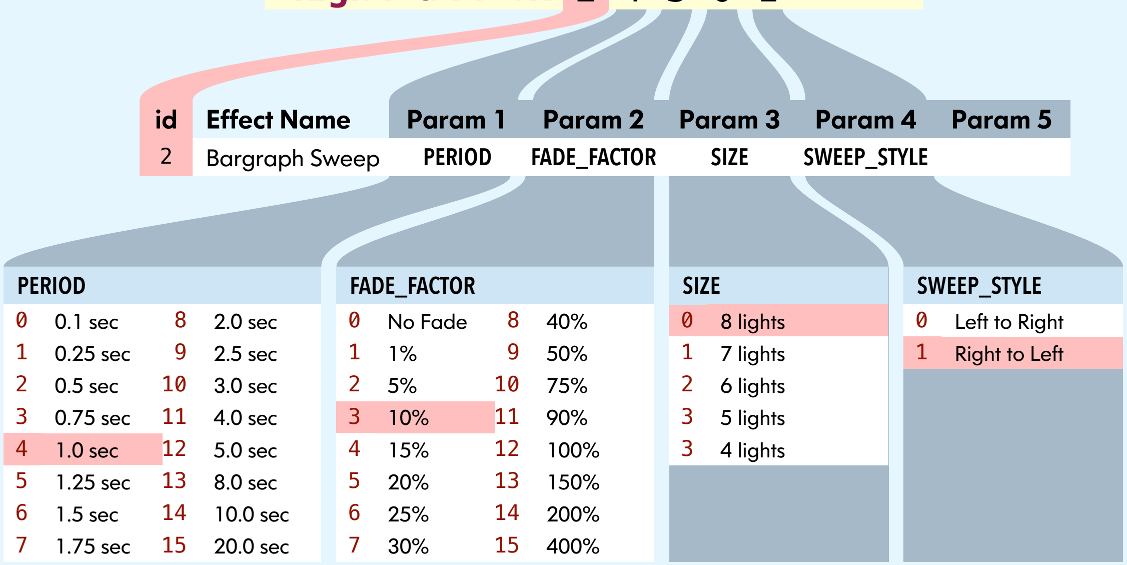
An effect is specified by a numeric *id* value. Zero of more *parameters* are appended to the command if required. Parameter values follow immediately separated by spaces or in a list.

```
# Equally valid syntaxes for fx parameters
# Parameters after
light all fx 2 4 3 0 1
# Parameters in a list
light all fx 2 [4, 3, 0, 1]
```

Each Fx type has its own number of parameters. The number of parameters and their respective definitions are shown in the diagrams over the next few pages.

This example shows a light Fx command and its corresponding details for each parameter

```
# Bargraph Sweep
# Right to Left
# 1 sec period, 10% fade, 8 lights
light all fx 2 4 3 0 1
```



id	Effect Name	Param 1	Param 2	Param 3	Param 4	Param 5
2	Bargraph Sweep	PERIOD	FADE_FACTOR	SIZE	SWEEP_STYLE	

PERIOD

0	0.1 sec	8	2.0 sec
1	0.25 sec	9	2.5 sec
2	0.5 sec	10	3.0 sec
3	0.75 sec	11	4.0 sec
4	1.0 sec	12	5.0 sec
5	1.25 sec	13	8.0 sec
6	1.5 sec	14	10.0 sec
7	1.75 sec	15	20.0 sec

FADE_FACTOR

0	No Fade	8	40%
1	1%	9	50%
2	5%	10	75%
3	10%	11	90%
4	15%	12	100%
5	20%	13	150%
6	25%	14	200%
7	30%	15	400%

SIZE

0	8 lights
1	7 lights
2	6 lights
3	5 lights
3	4 lights

SWEEP_STYLE

0	Left to Right
1	Right to Left

Light Fx Command

```
light channels fx id parameters      (up to 5 parameters)
```

id	Effect Name	Param 1	Param 2	Param 3
1	On/Off with options	DIR_OPTION	FADE_TIME	FLICKER_ON
2	Increase Brightness			
3	Decrease Brightness			
4	Set Brightness	BRIGHTNESS		
5	50% Flasher (positive)	PERIOD	FADE_FACTOR	
6	50% Flasher (negative)	PERIOD	FADE_FACTOR	
7	Strobe Flasher (positive)	PERIOD	DUTY_CYCLE	BURST_COUNT
8	Strobe Flasher (negative)	PERIOD	DUTY_CYCLE	BURST_COUNT
9	Gyralite Flasher (positive)	PERIOD	FADE_FACTOR	
10	Gyralite Flasher (negative)	PERIOD	FADE_FACTOR	
11	Flicker	PERIOD2	FADE_FACTOR	
12	Random Blinker	PERIOD2	FADE_FACTOR	
13	Photon Torpedo	PERIOD2		
14	Laser Pulse	PERIOD2		
15	Engine Glow	PERIOD		
16	Lighthouse	PERIOD		
17	Broken Light	FAULT_RATE	FADE_TIME	FAULT_INTENSITY
18	Status Indicator	SOURCE1	SOURCE2	INVERT
19	Sound Modulated	FADE_TIME		INVERT
20	Motor Speed Modulated	FADE_TIME	SOURCE2	INVERT

		Param 4	Param 5
XX	All Effects		RGB

7	6	5	4	3	2	1	0
				X	X		

DURATION			
0	0.5 sec	8	15 sec
1	1.0 sec	9	20 sec
2	1.5 sec	10	30 sec
3	2.0 sec	11	45 sec
4	3.0 sec	12	60 sec
5	4.0 sec	13	90 sec
6	5.0 sec	14	2 min
7	10 sec	15	5 min

TRANSITION	
0	Toggle
1	On
2	Off
3	On for **DURATION**

RGB	
0	White
1	Red
2	Green
3	Blue
4	Cyan
5	Magenta
6	Yellow
7	Orange
8	Turquoise
9	Violet

DIR_OPTION			
0	None	9	Odd if A ▲ Even if A ▼
1	On if A ▲	10	Odd if B ▲ Even if B ▼
2	On if A ▼	11	Odd if C ▲ Even if C ▼
3	On if B ▲	12	Odd if D ▲ Even if D ▼
4	On if B ▼	13	Odd if A ▼ Even if A ▲
5	On if C ▲	14	Odd if B ▼ Even if B ▲
6	On if C ▼	15	Odd if C ▼ Even if C ▲
7	On if D ▲	16	Odd if D ▼ Even if D ▲
8	On if D ▼		

SOURCE1	
0x01	USB Connected
0x02	USB Activity
0x04	IR Activity
0x08	IR Lockout active
0x10	Audio active
0x20	BLE Connected
0x40	BLE Activity
0x80	File system active
	logic OR combo

SOURCE2	
0x01	Motor A ▲
0x02	Motor A ▼
0x04	Motor B ▲
0x08	Motor B ▼
0x10	-
0x20	-
0x40	Button
0x80	-
	logic OR combo

PERIOD			
0	0.1 sec	8	2.0 sec
1	0.25 sec	9	2.5 sec
2	0.5 sec	10	3.0 sec
3	0.75 sec	11	4.0 sec
4	1.0 sec	12	5.0 sec
5	1.25 sec	13	8.0 sec
6	1.5 sec	14	10.0 sec
7	1.75 sec	15	20.0 sec

PERIOD2			
0	0.05 sec	8	0.8 sec
1	0.1 sec	9	0.9 sec
2	0.2 sec	10	1.0 sec
3	0.3 sec	11	1.25 sec
4	0.4 sec	12	1.5 sec
5	0.5 sec	13	1.75 sec
6	0.6 sec	14	2.0 sec
7	0.7 sec	15	3.0 sec

FLICKER_ON	
0	No
1	Yes

BURST_COUNT	
0	1 pulse
1	2 pulses
2	3 pulses
3	4 pulses

FADE_TIME			
0	No Fade	8	1.0 sec
1	50 ms	9	1.5 sec
2	0.1 sec	10	2.0 sec
3	0.2 sec	11	2.5 sec
4	0.4 sec	12	3.0 sec
5	0.5 sec	13	4.0 sec
6	0.6 sec	14	5.0 sec
7	0.8 sec	15	10.0 sec

FADE_FACTOR			
0	No Fade	8	40%
1	1%	9	50%
2	5%	10	75%
3	10%	11	90%
4	15%	12	100%
5	20%	13	150%
6	25%	14	200%
7	30%	15	400%

INVERT	
0	No
1	Yes

FAULT_RATE	
0	Rare
1	Occasionally
2	Often
3	Very Often

DUTY_CYCLE							
0	1%	5	20%	10	60%	15	90%
1	2%	6	25%	11	70%	16	95%
2	5%	7	30%	12	75%	17	98%
3	10%	8	40%	13	80%	18	99%
4	15%	9	50%	14	85%		

FAULT_INTENSITY	
0	Subtle
1	Moderate
2	Severe
3	Maximum

Light Combo Fx Command

`light all fx id parameters` *(up to 5 parameters)*

id	Effect Name	Param 1	Param 2	Param 3	Param 4	Param 5
1	Linear Sweep	PERIOD	FADE_FACTOR	SIZE	SWEEP_STYLE	
2	Bargraph Sweep	PERIOD	FADE_FACTOR	SIZE	SWEEP_STYLE	
3	Knight Rider	PERIOD	FADE_FACTOR			
4	Emergency Flasher (Twinsonic)	TWIN_STYLE	SEQ	FLASH_RATE		
5	Emergency Flasher (Whelen)	WHELEN_STYLE	SEQ	FLASH_RATE		
6	Times Square	PERIOD2	FADE_FACTOR			
7	Noise	PERIOD2	FADE_FACTOR			
8	Twinkling Stars	PERIOD	DUTY_CYCLE			
9	Traffic Lights	TRAFFIC_STYLE	FADE_FACTOR	SEQ_TIME		
10	Sound Bar	BAR_STYLE	FADE_FACTOR	SIZE		
11	Alternating Flashers	PERIOD	FADE_FACTOR	DUTY_CYCLE	OUT_MASK	TRANSITION
12	Lava Lamp	PERIOD	SIZE			
13	Laser Cannon	FLASH_RATE	FADE_FACTOR	SIZE	SWEEP_STYLE	
14	Dragster Starter	DRAGSTER_STYLE	FADE_FACTOR			
15	Airport Runway	RUNWAY_RATE	FADE_FACTOR	RUNWAY_BRIGHT		
16	Formula 1 Indicators	F1_STYLE	FADE_FACTOR	FLASH_RATE		

PERIOD

0	0.1 sec	8	2.0 sec
1	0.25 sec	9	2.5 sec
2	0.5 sec	10	3.0 sec
3	0.75 sec	11	4.0 sec
4	1.0 sec	12	5.0 sec
5	1.25 sec	13	8.0 sec
6	1.5 sec	14	10.0 sec
7	1.75 sec	15	20.0 sec

SWEEP_STYLE

0	Left to Right
1	Right to Left

FLASH_RATE

0	Slow (60 fpm)
1	Med (90 fpm)
2	Fast (120 fpm)
3	V. Fast (150 fpm)

PERIOD2

0	0.05 sec	8	0.8 sec
1	0.1 sec	9	0.9 sec
2	0.2 sec	10	1.0 sec
3	0.3 sec	11	1.25 sec
4	0.4 sec	12	1.5 sec
5	0.5 sec	13	1.75 sec
6	0.6 sec	14	2.0 sec
7	0.7 sec	15	3.0 sec

SIZE

0	8 lights
1	7 lights
2	6 lights
3	5 lights
3	4 lights

SEQ

0	Solid
1	Left/Right
1	In/Out

FADE_FACTOR

0	No Fade	8	40%
1	1%	9	50%
2	5%	10	75%
3	10%	11	90%
4	15%	12	100%
5	20%	13	150%
6	25%	14	200%
7	30%	15	400%

SEQ_TIME

0	Slow (60 sec)
1	Med (45 sec)
2	Fast (30 sec)
3	Very Fast (20 sec)

DUTY_CYCLE

0	1%	10	60%
1	2%	11	70%
2	5%	12	75%
3	10%	13	80%
4	15%	14	85%
5	20%	15	90%
6	25%	16	95%
7	30%	17	98%
8	40%	18	99%
9	50%		

TWINSONIC_STYLE

0	Single
1	Dual
2	Aero
3	Combo

BAR_STYLE

0	Left to Right
1	Right to Left
2	In to Out
3	Out to In

TRAFFIC_STYLE

0	Standard
1	Standard with flashing green
2	European
3	Flash red (NS), flash yellow (EW)
4	Standard w/ped crossing
5	European w/ped crossing
6	Flash red (EW), flash yellow (NS)
7	International
8	International w/ped crossing
9	International 2
10	International 2 w/ped crossing

WHELEN_STYLE

0	Signal Alert
1	Signal Alert Steady
2	Comet Flash
3	Action Flash 50
4	Action Flash 150
5	Modu Flash
6	Single Flash
7	Double Flash
8	Triple Flash
9	Warning
10	Random

DRAGSTER_STYLE

0	Standard countdown to green
1	Pro countdown (0.5 sec)
2	Pro countdown (0.4 sec)

F1_STYLE

0	Race start countdown
1	Training countdown
2	Race break/caution
3	Training start
4	Training break
5	Training end

RUNWAY_RATE

0	Steady (no flashing)
1	Slow
2	Med
3	Fast

RUNWAY_BRIGHT

0	Maximum
1	Med
2	Low
3	Minimum

OUT_MASK logic OR combo

0x01	Ch 1	0x10	Ch 5
0x02	Ch 2	0x20	Ch 6
0x04	Ch 3	0x40	Ch 7
0x08	Ch 4	0x80	Ch 8

TRANSITION

0	Toggle
1	to On
2	to Off
3	Duration

Motor Fx Command

`motor channels fx id parameters` *(up to 2 parameters)*

id	Effect Name	Param 1	Param 2
0	Emergency Stop		
1	Stop		
2	Increase Speed	MOTOR_STEP	
3	Decrease Speed	MOTOR_STEP	
4	Increase Speed (Bidirectional)	MOTOR_STEP	
5	Decrease Speed (Bidirectional)	MOTOR_STEP	
6	Change Direction		
7	Set Speed	MOTOR_SPEED	
8	Set Speed (Timed duration)	MOTOR_SPEED	DURATION
9	Oscillate	MOTOR_SPEED	MOTOR_PERIOD
10	Oscillate Bidirectional	MOTOR_SPEED	MOTOR_PERIOD
11	Oscillate Bidirectional with Wait	MOTOR_SPEED	MOTOR_PERIOD
12	Random	MOTOR_SPEED	MOTOR_PERIOD
13	Random Bidirectional	MOTOR_SPEED	MOTOR_PERIOD
14	Sound Modulated	MOTOR_SPEED	
15	Set Servo	MOTOR_POS	

MOTOR_STEP

0	±1 step (high res)
1	1% (100 steps)
2	2% (50 steps)
3	3% (33 steps)
4	5% (20 steps)
5	6% (16 steps)
6	10% (10 steps)
7	20% (5 steps)
8	25% (4 steps)
9	33% (3 steps)
10	LEGO compatible 7 steps
11	15 deg (servo increment)

MOTOR_PERIOD

0	0.25 sec	8	3.0 sec
1	0.5 sec	9	4.0 sec
2	0.75 sec	10	5.0 sec
3	1.0 sec	11	10 sec
4	1.25 sec	12	15 sec
5	1.5 sec	13	20 sec
6	2.0 sec	14	30 sec
7	2.5 sec	15	60 sec

MOTOR_POS

0	-90 deg	7	15 deg
1	-75 deg	8	30 deg
2	-60 deg	9	45 deg
3	-45 deg	10	60 deg
4	-30 deg	11	75 deg
5	-15 deg	12	90 deg
6	0 deg		

MOTOR_SPEED

Low Resolution 8 Speed Step Range

0	stopped		8	stopped	
1	10%	% of maximum speed in the forward direction	9	10%	% of maximum speed in the reverse direction
2	25%		10	25%	
3	33%		11	33%	
4	50%		12	50%	
5	67%		13	67%	
6	75%		14	75%	
7	100%		15	100%	

High Resolution 64 Speed Step Range

128	stopped	Forward Direction	192	stopped	Reverse Direction
129	1		193	1	
130	2		194	2	
⋮			⋮		
190	62		254	62	
191	63		255	63	

MOTOR_PERIOD

bit position	7	6	5	4	3	2	1	0
	Off Time				Off Time			

OFF TIME

0	0.25 sec	8	3.0 sec
1	0.5 sec	9	4.0 sec
2	0.75 sec	10	5.0 sec
3	1.0 sec	11	10 sec
4	1.25 sec	12	15 sec
5	1.5 sec	13	20 sec
6	2.0 sec	14	30 sec
7	2.5 sec	15	60 sec

ON TIME

0	0.25 sec	8	3.0 sec
1	0.5 sec	9	4.0 sec
2	0.75 sec	10	5.0 sec
3	1.0 sec	11	10 sec
4	1.25 sec	12	15 sec
5	1.5 sec	13	20 sec
6	2.0 sec	14	30 sec
7	2.5 sec	15	60 sec

DURATION

0	0.5 sec	4	3.0 sec	8	15 sec	12	60 sec
1	1.0 sec	5	4.0 sec	9	20 sec	13	90 sec
2	1.5 sec	6	5.0 sec	10	30 sec	14	2 min
3	2.0 sec	7	10 sec	11	45 sec	15	5 min

Sound Fx Command

`sound fx id file parameters` *(up to 2 parameters)*

id	file	Effect Name	Param 1	Param 2
1	○	Increase Volume		
2	○	Decrease Volume		
3	○	Set Volume		VOLUME
4	●	Play Once	RETRIGGER	RELVOLUME
5	●	Play Continuous		RELVOLUME
6	●	Play N Times	REPEAT_COUNT	RELVOLUME
7	●	Play for Duration	DURATION	RELVOLUME
8	●	Motor Pitch Modulated	MOTOR_OUTPUT	GAIN
9	○	Gated Motor Modulated	MOTOR_OUTPUT	GAIN
10	●	Amplitude Motor Modulated	MOTOR_OUTPUT	GAIN
11	●	Stop		
12	○	Play Indexed by Motor Speed	MOTOR_OUTPUT	IDX_OPTIONS
13	●	Random Playback	PROBABILITY	
14	●	Seek within file playback	TIME_MSB	TIME_LSB
15	●	Scrub file playback	TIME_MSB	TIME_LSB

Sound Fx which do not require a `file` specification should substitute a placeholder value of `0`:

```
# Set the volume to 60 with fx command
sound fx 3 0 0 60
```

`file` can be specified either as numeric file ID or string:

```
# Play a sound file randomly
sound fx 13 "Siren1.wav" 3 0
sound fx 13 50 3 0
```

VOLUME

0 ~ 255	Volume range

REPEAT_COUNT

1 ~ 100	Playback times

RETRIGGER

0	No, toggle playback
1	Yes, restart playback

REL_VOLUME

0	0 dB	8	-8 dB
1	1 dB	9	-7 dB
2	+2 dB	10	-6 dB
3	+3 dB	11	-5 dB
4	+4 dB	12	-4 dB
5	+5 dB	13	-3 dB
6	+6 dB	14	-2 dB
7	+7 dB	15	-1 dB

DURATION

0	0.5 sec	8	15 sec
1	1.0 sec	9	20 sec
2	1.5 sec	10	30 sec
3	2.0 sec	11	45 sec
4	3.0 sec	12	60 sec
5	4.0 sec	13	90 sec
6	5.0 sec	14	2 min
7	10 sec	15	5 min

MOTOR_OUTPUT

0	A Target Speed	4	A Current Speed
1	B Target Speed	5	B Current Speed
2	C Target Speed	6	C Current Speed
3	D Target Speed	7	D Current Speed

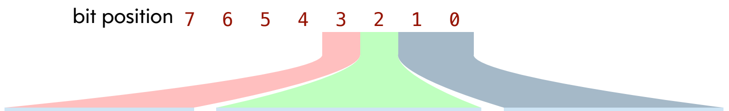
PROBABILITY

0	Rare
1	Occasionally
2	Often
3	Very Often

GAIN

-100 ~ +100	Gain range

IDX_OPTIONS



STARTUP OVERIDE	
0	No
1	Yes

PLAY STARTUP/SHUTDOWN	
0	No
1	Yes

VOL MODULATION	
0	None
1	Min
2	Med
3	Max

TIME_MSB TIME_LSB

Combined as a 16 bit two's complement value of 100 ms per LSB.

TIME_MSB	TIME_LSB		TIME_MSB	TIME_LSB	
0x7F	0xFF	3276.7 sec	0xFF	0xFF	-0.1 sec
…	…		0xFF	0xFE	-0.2 sec
0x00	0x01	0.1 sec	…	…	
0x00	0x00	0 sec	0x80	0x00	-3276.8 sec

Recipes

Directional Headlight for a Train

```
# Front light:  Ch 1
# Rear light: Ch 2
# odd numbered light channels ON when Fwd
# even numbered light channels ON when Rev

# method 1: use one startup action with
# Light On/Off toggle with DIR_OPTIONS

event startup 1 {
  light [1, 2] fx 1 9
}
```

```
# method 2: use two startup actions
# (one for each group of light channels)
# configured as Status Indicators with
# bits 0 and 1 of SOURCE2

event startup 1 {
  light 1 fx 18 0 1
}
event startup 2 {
  light 2 fx 18 0 2
}
```

Locomotive Ditch Lights

```
# Setup ditch lights which stay ON between
# flashing intervals
# 5 = 1.25 sec period
# 6 = 25% fade
# 8 = 40% duty cycle
# 0x03 = channels 1 & 2
# 1 = transition to stay ON

event startup 1 {
  light all fx 11 5 6 8 0x03 1
}
```

```
# Same as above, but flashing lasts 10 sec
# 0x73 = transition to timed 10sec duration

event startup 1 {
  light all fx 11 5 6 8 0x03 0x73
}
```

Bluetooth Connection Indicator Light

```
event ble connect {
  light 1 on
}
event ble disconnect {
  light 1 off
}
```

Bluetooth Connection Indicator with IR disable

```
event ble connect {
  light 1 on
  ir off
}
event ble disconnect {
  light 1 off
  ir on
}
```

Bluetooth Activity Indicator Light

```
# Setup Status Indicator light
# action using bit 6 of SOURCE1

light 1 fx 18 0x40 0x00

# Can also assign this to a startup action
event startup 1 {
  light 1 fx 18 0x40 0x00
}
```

USB Activity Indicator Light

```
# Setup Status Indicator light
# action using bit 1 of SOURCE1

light 1 fx 18 0x02 0x00
```

Sound Playback with Bargraph Lights

```
# Setup Status Indicator light
# action using bit 1 of SOURCE1
# 0 = L to R bar graph
# 6 = 25% fade factor
# 0 = use 8 light channels

light all fx 10 0 6 0
sound play “MusicClip.wav”
wait sound “MusicClip.wav”
light all off
```

Joystick activated Crossing Lights / Arm

```
# Configure crossing lights which flash
# ch 1 and 2
# activate a crossing arm attached to servo
# motor
# Use joystick Ch 1 left lever to toggle

event joy ch 1 left up {
  light all fx 11 5 6 8 0x03
  motor servo 90
}
event joy ch 1 left down {
  light [1, 2] off
  motor servo 0
}
```

Stop Motor when BLE Disconnects

```
event ble disconnect {
  motor off off
}
```

Toggle Lights with touchLAB

```
event button {
  light 1 fx 1
}
```

Show touchLAB button status with a Light

```
# Method 1: setup events to respond to
# up and down event separately

event button up {
  light 1 on
}
event button down {
  light 1 off
}
```

```
# Method 2: setup a Status Indicator light
# action using bit 6 of SOURCE2

light 1 fx 18 0x00 0x40

# Can also assign this to a startup action
event startup 1 {
  light 1 fx 18 0x00 0x40
}
```