The Prism RGB LED Driver is a coprocessor and voltage regulator meant to drive up to 256 WS2812 protocol, 5V LEDs. It has limited control options via a hobby servo PWM interface, and full-featured customization through an I²C interface.

A finite number of user-customizable animations can be layered on top of one another to create “Artboards” which the Prism saves to internal memory. This allows for short I²C transmissions while retaining options for unique designs.

Summary of Product Details:				
Input Voltage	6 - 30V		I²C Address	0x38
I²C Logic Level	3.3V		Max tested I²C Frequency	400khz
PWM Range	500-2500µsec		Input Power Connector	XT30 (FH-MC)
LED/PWM Power Output	5A @ 5V		LED Output Connector	3-Pos JST PH (FH-MC)
I²C Port Power Output	100mA @ 3.3V		I²C Connector Type	4-Pos JST PH [FH-MC]
Maximum number of LEDs	256		PWM Connector	Servo TJC8

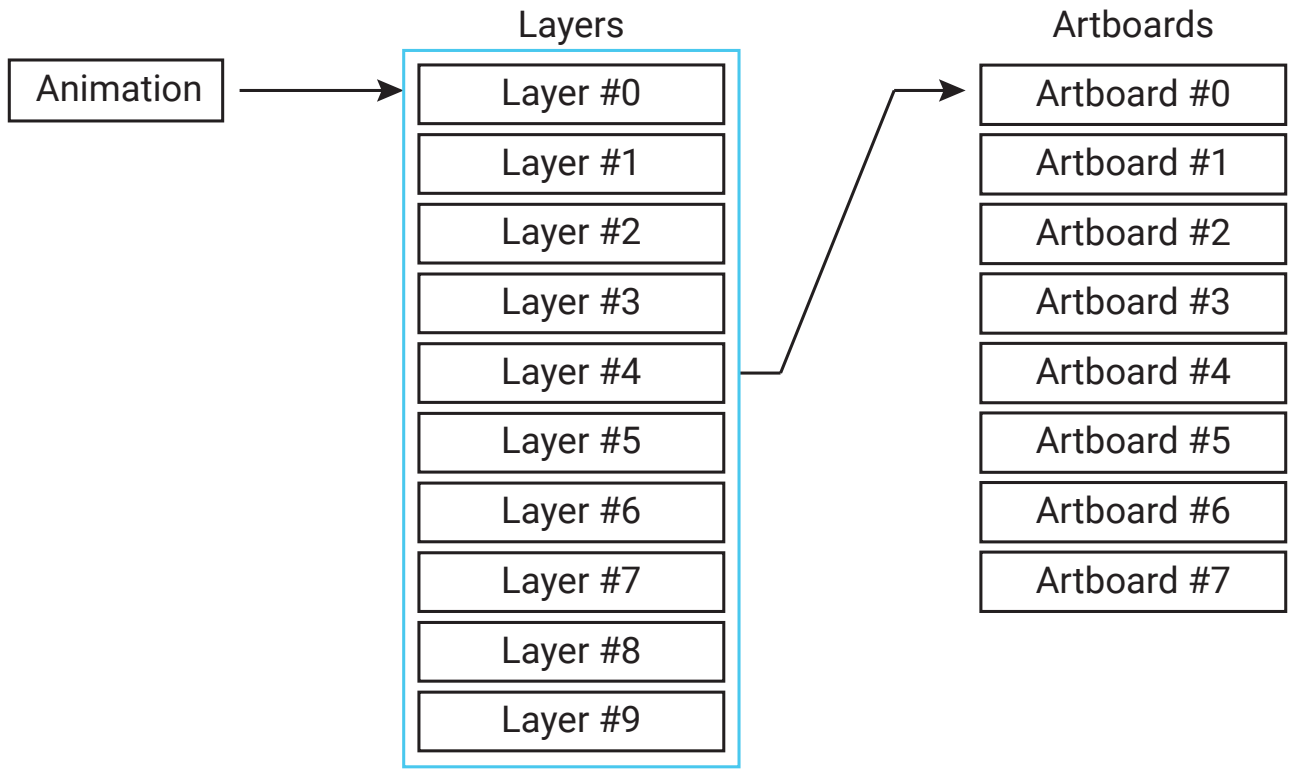
Table of Contents:

Animations, Layers, and Artboards:

When working with the Prism RGB LED Driver over I²C, it's important to understand three key concepts that we call "Animations," "Layers," and "Artboards."
There are 12 **Animations** which can be chosen from, and customized. These are things like solid colors, blinking, snakes, or police lights.

Layers can store an animation. Each animation can have its own start and stop point, and can have transparent elements. This allows you to layer multiple animations at different heights. For example, a full-color rainbow might be your base. On top of that you can layer a blink animation switching between a solid color, and transparent. You'd see the solid color for 1s, and the rainbow for another 1s. As you add layers your display can become more and more complex.

Artboards are stored combinations of up to 10 layers/animations. Artboards are stored in device EPROM so they are saved even when the devicepower cycles, and allow the user to save their creations and recall them. We intend for users to create Artboards through custom animations, and then in their robot programs simply switch between saved Artboards. This greatly shortens I²C transmission length compared to repeatedly sending animations alongside their customizations.

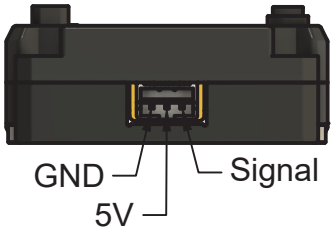


LED Compatibility:

The Prism Driver follows the WS2812 control protocol. It outputs 5V at up to 5A to power a large number of LEDs at a high brightness level. These combined allow it to drive a wide variety of commercially available LEDs in addition to the the goBILDA Prism LED Strips. The Prism Driver can drive up to 256 LEDs and is designed to shelter the input power supp from short circuit events on the output.

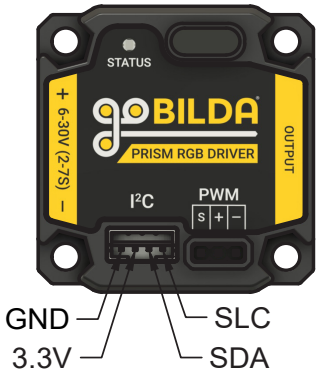
Output Connector Pinout:

The Prism LED Driver connects to LEDs via a 3-Pos JST PH connector. Pinout as shown:



I²C Connector Pinout:

The Prism LED Driver connects to an I²C Controller via a 4-Pos JST PH connector. This connector can supply 3.3V at up to 100mA to power a controller. This 3.3V source is designed to play well with other power supplies and it is not necessary to disconnect the voltage line when using 3.3V microcontrollers.



Device State Codes:

The Prism has several device “states” that it reports to the user through the Status LED on the device.

Status LED	Description
White	Device booting up.
Red	Saving data to EPROM (device permanent memory.)
Green	Valid PWM signal detected, showing animation from PWM.
Blue	Valid I²C communication detected.
Yellow	Showing default animation.

PWM Control:

While PWM is a more limited control option for the Prism compared to I²C, we did our best to pack in options to control the same RGB LEDs.

Much of the PWM Control range is split into 10µsec chunks, each selecting an Artboard. Many of the Artboards accessable via PWM are already configured, giving you access to tons of options for Snakes, Rainbow Snakes, Rainbow animations, and a few custom configured animations we put together. You can also select one of the 8 saved user-created Artboards which you may have configured via I²C.

Some of this PWM range instead adjusts spesifics of the selected animation. The largest example of this is the range from 1050-1950µsec which adjusts the hue of a solid color across the entire length of the strip - this matches the PWM/Color map on the goBILDA Indicator lights.
A similar range from 620-699µsec adjusts the hue of a Sparkle animation as you move through the range.

Some animations which are accessable via I²C are not accessible via PWM. These animations are either niche, or are tied very closely to the length of the LED strip, which is not configurable via PWM. These animations include: Blink, Droid Scan, Random, and Single Fill.

Using PWM with a REV Control Hub:

We recommend using I²C over PWM with a REV Control or Expansion Hub because of much increased flexibility and ease-of-use.

If you do choose to use PWM control of the Prism Driver with a REV Control or Expansion Hub, it is vital to not connect the power lead (the center/red wire on a standard servo connector) from the Prism Driver to the Control/Expansion hub. Both are sources of 5V power and while the Prism Driver is designed to tolerate this voltage difference, this can permanently damage your Control/Expanion hub servo ports. We strongly recommend either sourcing a male-to-male servo extension without the red wire, or removing the red wire from an off-the-shelf cable.

PWM Control Table

This table describes the LED output that coresponds to the PWM input to the Prism Driver.

PWM Range	Animation Selected	Description
500-519µsec	Custom Artboard Slot #0	User-Configured Artboard
510-519µsec	Custom Artboard Slot #1	User-Configured Artboard
520-529µsec	Custom Artboard Slot #2	User-Configured Artboard
530-539µsec	Custom Artboard Slot #3	User-Configured Artboard
540-549µsec	Custom Artboard Slot #4	User-Configured Artboard
550-559µsec	Custom Artboard Slot #5	User-Configured Artboard
560-569µsec	Custom Artboard Slot #6	User-Configured Artboard
570-579µsec	Custom Artboard Slot #7	User-Configured Artboard
580-589µsec	Emergency Lights	Police Lights animation with a period of 650ms.
590-599µsec	Aurora Borealis	Two layered Sine Wave animations below a Sparkle animaiton.
600-609µsec	Blink Counter	Three Blink animations layered so that you get a blue blink every second, a green blink every 5 seconds, and a red blink every 10 seconds.
610-619µsec	FTC Timer	Three Blink animations on top of a rainbow animation. It is green for 30s, then rainbow for 8s, then blue for 1:40, red for 40s, and finally rainbow.
620-699µsec	Sparkle	Continous adjustment of the hue of a Sparkle animation with a transparent background.
700-949µsec	Sine Wave	Continous adjustment of the hue of a Sine Wave animation on a transparent background.
950-959µsec	Rainbow - Red and Green	A rainbow animation with a hue from 0-180.
960-969µsec	Rainbow - Purple and Blue	A rainbow animation with a hue from 180-360.
970-979µsec	Rainbow - Blues	A rainbow animation with a hue from 170-275.
980-989µsec	Rainbow - Greens	A rainbow animation with a hue from 70-170.
990-999µsec	Rainbow - Reds	A rainbow animation with a hue from 295-40.
1000-1009µsec	Rainbow - Full Color	A full color rainbow animation.
1010-1019µsec	Rainbow - Party	A rainbow animation with a hue from 55-180.
1020-1029µsec	Rainbow - Ocean	A rainbow animation with a hue from 170-235.
1030-1039µsec	Rainbow - Lava	A rainbow animation with a hue from 0-40.
1040-1049µsec	Rainbow - Forest	A rainbow animation with a hue from 100-160.
1050-1949µsec	Solid Color	Continous adjustment of the hue of a Solid color.
950-2199µsec	Pulse	Continous adjustment of the hue of a Pulse animation on a transparent background.
2200-2349µsec	Snakes	15 unique Snakes spaced evenly at 10µsec intervals. Details on page 19.
2350-2500µsec	Rainbow Snakes	15 unique rainbow snakes animations spaced evenly at 10µsec intervals. Details on page 19.

I²C Register Map:

Register	Name	Access	Size (bits)	Description
0x00	Device ID	Read	8	Device identifier
0x01	Firmware Version	Read	24	Firmware version
0x02	Hardware Version	Read	16	Hardware version number
0x03	Power Cycle Count	Read	32	Number times device has been powered on/off
0x04	Runtime In Minutes	Read	32	Total device runtime in minutes
0x05	Status	Read	32	Device status information
0x06	Control	Write	32	Control device operation
0x07	Save/Load Animation Group	Write	32	Save/Load an Artboard to/from EPROM
0x08	Layer 0	R/W	x	Animation settings and parameters
0x09	Layer 1	R/W	x	Animation settings and parameters
0x0A	Layer 2	R/W	x	Animation settings and parameters
0x0B	Layer 3	R/W	x	Animation settings and parameters
0x0C	Layer 4	R/W	x	Animation settings and parameters
0x0D	Layer 5	R/W	x	Animation settings and parameters
0x0E	Layer 6	R/W	x	Animation settings and parameters
0x0F	Layer 7	R/W	x	Animation settings and parameters
0x10	Layer 8	R/W	x	Animation settings and parameters
0x11	Layer 9	R/W	x	Animation settings and parameters

0x00 - Device ID Register (Read-only):

Size: 8-bit (uint8_t)

Bits	Name	Description	Default
0:7	Device ID	Fixed device identifier	0x3

0x01 - Device Firmware Version Register (Read-only):

Size: 24-bit (uint32_t)

Bits	Name	Description	Data Type
0:7	Patch Version	Patch firmware version	uint8_t
8:15	Minor Version	Minor firmware version	uint8_t
16:23	Major Version	Major firmware version	uint8_t
24:31	Reserved	x	x

0x02 - Hardware Version Register (Read-only):

Size: 16-bit (uint16_t)

Bits	Name	Description	Data Type
0:7	Minor Revision	Minor Hardware version	uint8_t
8:15	Major Revision	Major Hardware version	uint8_t

0x03 - Power Cycle Count (Read-only):

Size: 32-bit (uint32_t)

Bits	Name	Description
0:31	Power Cycle Count	Number of Power Cycles

0x04 - Runtime in Minutes (Read-only):

Size: 32-bit (uint32_t)

Bits	Name	Description
0:31	Runtime In Minutes	Total minutes device has been powered on

0x05 - Status Register (Read Only):

Size: 32-bit (uint32_t)

Bits	Name	Description	Default	Values
0:7	Number of LEDs	Total number of LEDs in the chain	255	0-255
8:23	FPS	Current frame rate	60	0-33,333
24	Maximum FPS	If 1, then running at the Maximum FPS	0x1	0-1
25:28	Default Animation	Animation to run when the device is powered up	0	0-7
29	Default Animation Enabled	Determines if the Default Animation is shown on startup	0	0-1

0x06 - Control Register (Write-only):

Size: 32-bit (uint32_t)

Bits	Name	Description	Default	Values
0:14	Target FPS	Set desired frame rate	60	0-33,333
15	Change Target FPS	Setting to 1 Enables changing the Target FPS	x	0-1
16-23	Number of LEDs	Set number of LEDs (max 255)	255	0-255
24	Change Number of Leds	Setting to 1 Enables Changing the Number of LEDs	x	0-1
25	Clear Animations	Setting to 1 Removes all animations from the strip	x	0-1

0x07 - Save/Enable/Default Animation Slot (Write-only):

Size: 32-bit (uint32_t)

Bits	Name	Description	Values
0:7	Save Current Artboard to an Artboard slot	Bitwise opperation which saves current Artboard to a slot.	1<<0 to 1<<7
8:15	Load an Artboard from a slot and display it	Bitwise opperation which loads an Artboard from a slot	1<<8 to 1<<15
16:23	Set an Artboard Slot as Default Boot up Animation	Set the default Artboard to a slot.	1<<16 to 1<<23
24	Enable Boot Animation	Set high once to enable	single bit bitmask
25	Disable Boot Animation	Set high once to disable	single bit bitmask

Below is a table showing each bit in the register. Bits 0-23 fall into three groups, each bit corresponding to one of 8 Artboard slots. Artboard slots are 0 indexed starting at bit 0, 8, and 16. Within each group, selecting that Artboard slot performs a different action.
Pulling bit 6 high will save your currently displayed Artboard to Artboard slot #6. While pulling bit 18 high will select Artboard #2 as the default animation when you apply power to the device.

31-26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Show at Boot		Select this Artboard slot as default animation								Load Artboard saved at this slot and display it								Save currently displayed Artboard to this slot							

Bits 31-26: Reserved

Bit 25: Disable Default Artboard.
0: No change
1: Disable animation on boot up.

Bit 24: Enable Default Artboard.
0: No change
1: Enable animation on boot up.

Bits 23-16: Select default Artboard.
0: No change
1: Selected Artboard slot will show when device is powered on if enabled.

Bits 15-8: Load Artboard from slot
0: No change
1: Load Artboard saved at selected slot and display it.

Bits 7-0: Save Current Artboard to Slot
0: No change
1: Save current Artboard (all currently displayed animations) to selected Artboard Slot.

0x08-0x11 - layer x (Read/Write):

Size: 8-bit (uint8_t) for animation selection, variable for sub-registers

Sub-Register	Name	Size (C type)	Description	Default
0x00	Selected Animation	1 byte (uint8_t)	Selected Animation from 0 - 12	0 (Invalid/No Animation)
...	Animation Registers	Variable	Start index of the animation	0

Example I²C Transaction:

Below we detail an I²C transaction between a controller and the Prism Driver. At the start of an I²C transmission we send the address of the device we'd like to reach, here that's 0x38. Followed by the address of the register we'd like to write to.
When you write an animation to a slot on the Prism, we also require a sub-register. This allows you to configure a specific detail about an animation without needing to specify every detail of it every time. Only registers 0x8-0x11 have sub-registers. For writing to other registers you need only send the address, the register, and the data you'd like to write to that address.
Each animation's sub-registers are detailed on the following pages, but every animation has the following sub-registers:

Sub-Register	Name	Size (C type)	Description
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100
0x02	Start Index	1 byte (uint8_t)	Start index of the animation
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation

Brightness is an unsigned, 8 bit integer which is expecting a value between 0 and 100.

Start Index is the number of pixels past the Prism Driver that the animation should start. Increasing this number moves the start of the animation further from the Prism Driver.

Stop index is the end point of the animation. Decreasing this number moves the end of the animation closer to the Prism Driver.

Shown below is a transmission which inserts a Solid Color animation at layer 0. Then sets the start index, stop index, and color.

```
// Device Address: 0x38
// Layer 0 Address: 0x08
0x38, 0x08, 0x00, 0x01          // Device Address, Layer 0, Selected Animation, Solid Color Animation
0x38, 0x08, 0x02, 0x00          // Device Address, layer 0, Start Index Address, 0
0x38, 0x08, 0x03, 0x0A          // Device Address, Layer 0, Stop Index Address, 10
0x38, 0x08, 0x04, 0xFF, 0x00, 0x00 // Device Address, Slot 3, Primary Color Address, {255, 0, 0}(RGB)
```

Solid Color Animation:

Solid color is the simpliest animation, it just displays a single color. Like all animations it has a start point, and end point, and a brightness.

Sub-Register	Name	Value	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	SteelBlue (0x4682B4)

Blinking Animation:

Blink switches back and forth between two specified colors. The period of the animation defines the total time one loop of the animation takes, while the primary color period can be used to lengthen or shorten the primary color. This can be used to create timers or unique blinking patterns.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Red (0xFF0000)
0x05	Secondary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Blue (0x0000FF)
0x06	Period	4 bytes (uint32_t)	Animation period in ms	2000
0x07	Primary Color Period	4 bytes (uint32_t)	Period for the primary color	1000

Pulsing Animation:

This pulses between two colors, gradually increasing the transparency of one while decreasing the other. This creates a breathing effect.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Green (0x00FF00)
0x05	Secondary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x06	Period	4 bytes (uint32_t)	Animation period in ms	1000

Sine Wave Animation:

Similar to pulse, but with an extra layer of variability between both colors.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Yellow (0xFFFF00)
0x05	Secondary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x06	Period	4 bytes (uint32_t)	Animation period in ms	100ms
0x07	Direction	1 byte (bool)	0: Forward, 1: Backward	0 (Forward)
0x08	Offset	4 bytes (float)	Vertical Sine Wave offset	0.25
0x09	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.25

Droid Scan Animation:

Inspired by the Battlestar Galactica Cylons, this animation has a short string of LEDs which slide back and forth across the total length.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Red (0xFF0000)
0x05	Secondary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x06	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.4
0x07	Eye Width	1 byte (uint8_t)	Center Eye Size	3 (Pixels)
0x08	Trail Width	1 byte (uint8_t)	Size of front/back fading trail	3 (Pixels each way)
0x0B	Animation Style	1 bytes (float)	0: None, 1: Front, 2: Back, 3: Both	3 (Both)

Rainbow Animation:

A classic RGB Rainbow.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Start Hue	4 bytes (float)	Hue from 0.0 to 360.0	0.0 (Red)
0x05	End Hue	4 bytes (float)	Hue from 0.0 to 360.0	360.0 (Red)
0x06	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.50
0x07	Direction	1 byte (bool)	0: Forward, 1: Backward	1 (Backward)
0x09	Repeat After	1 byte (uint8_t)	Number of pixels before the rainbow repeats	25

Snakes Animation:

A flexible number of “snakes” of LEDs that zip along the length of the LED. Each snake can be a unique color.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Colors Used	1 byte (uint8_t)	Total number snake colors to show	3
0x05	Colors	30 bytes (uint8_t[3])[10]	0: Forward, 1: Backward	[Red, White, Blue]
0x06	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.50
0x07	Direction	1 byte (bool)	0: Forward, 1: Backward	0 (Forward)
0x08	Spacing Between	1 byte (uint8_t)	Spacing between consecutive snakes	2 Pixels
0x09	Repeat After	1 byte (uint8_t)	Spacing between the last and first	15 Pixels
0x0A	Background Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x0B	Snake Length	1 byte (uint8_t)	Length of each sake	5 Pixels

Random Animation:

Each pixel in this animation lights up a random color within the hue range specified.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Start Hue	4 bytes (float)	Hue from 0.0 to 360.0	0.0 (Red)
0x05	End Hue	4 bytes (float)	Hue from 0.0 to 360.0	360.0 (Red)
0x06	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.75

Sparkle Animation:

Random pixels within range light up the Primary Color.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Primary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Yellow (0xFFFF00)
0x05	Secondary Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x06	Period	4 bytes (uint32_t)	Animation period in ms	100ms
0x07	Sparkle Probability	1 byte (uint8_t)	Probability or Density of sparkles (higher is less dense)	16

Single Fill Animation:

Like Snakes, but instead of disappearing off the page these rows of fast moving pixels build upon each other until the entire animation length is the same color, before rows of a new color start replacing it.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Colors Used	1 byte (uint8_t)	Total number snake colors to show	3 Colors
0x05	Colors	30 bytes (uint8_t[3])[10]	0: Forward, 1: Backward	[Red, White, Blue]
0x06	Period	4 bytes (uint32_t)	Animation period in ms	750ms
0x07	Direction	1 byte (bool)	0: Forward, 1: Backward	0 (Forward)
0x08	Pixel Length	1 byte (uint8_t)	Length of each sake	5 Pixels
0x09	Speed	4 bytes (uin32_t)	Animation speed from 0-1	0.75
0x0B	Animation Style	1 byte (uint8_t)	Animations either fill-in or fill-out.	0 (Fill-in)

Rainbow Snakes Animation:

A blend of snakes with a classic rainbow RGB twist. Each of these snakes change color as they move down the animation.

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04	Start Hue	4 bytes (float)	Hue from 0.0 to 360.0	0.0 (Red)
0x05	End Hue	4 bytes (float)	Hue from 0.0 to 360.0	360.0 (Red)
0x06	Speed	4 bytes (float)	Animation speed from 0.0 to 1.0	0.50
0x07	Direction	1 byte (bool)	0: Forward, 1: Backward	1 (Backward)
0x08	Spacing Between	1 byte (uint8_t)	Spacing between consecutive snakes	3 Pixels
0x09	Repeat After	1 byte (uint8_t)	Spacing between the last and first	10 Pixels
0x0A	Background Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Transparent (0x000000)
0x0B	Snake Length	1 byte (uint8_t)	Length of each sake	6 Pixels
0x0C	Number of Snakes	1 byte (uint8_t)	Number of snakes before repeating	3

Police Lights Animation:

Bring a sense of urgency to your robot with this classic three-color animation. Three styles let you take inspiration from different emergency vehicles. Also consider changing the colors from their default blue/white/red to create your own style!

Sub-Register	Name	Size (C type)	Description	Default
0x01	Brightness	1 byte (uint8_t)	Value from 0 to 100	50
0x02	Start Index	1 byte (uint8_t)	Start index of the animation	0
0x03	Stop Index	1 byte (uint8_t)	Stop index of the animation	255
0x04-0x05	Reserved			
0x06	Period	4 bytes (uint32_t)	Animation period in ms	1000ms
0x07	Red Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Red (0xFF0000)
0x08	White Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	White (0x555555)
0x09	Blue Color	3 bytes (uint8_t[3])	RGB color (1 byte each)	Blue (0x0000FF)
0x0A	Reserved			
0x0B	Animation Style	1 byte (uint8_t)	Style of the Police Lights	0 (Standard Police Lights)

PWM Accessible Snakes Animations:

This table details the plethora of preconfigured snakes animations which are accesable via PWM, for more details on the PWM interface of the Prism Driver, see page #4.

PWM Range	Animation Selected	Description
2200-2209µsec	Snake 1	Red, white, and blue snakes on a transparent background.
2210-2219µsec	Snake 2	Red, green, and blue snakes on a transparent background.
2220-2229µsec	Snake 3	Lime, yellow-green, and green snakes on a transparent background.
2230-2239µsec	Snake 4	Black and yellow snakes on a transparent background.
2240-2249µsec	Snake 5	Red and white snakes on a transparent background.
2250-2259µsec	Snake 6	Pink, teal, yellow, light-green, and purple snakes on a dark gray background.
2260-2269µsec	Snake 7	Red, orange, yellow, green, blue, indigo, and violet snakes on a transparent background.
2270-2279µsec	Snake 8	Dark-red, orange-red, and yellow snakeson a transparent back-ground.
2280-2289µsec	Snake 9	White snakes on a navy background.
2290-2299µsec	Snake 10	Red snakes on a transparent background.
2300-2309µsec	Snake 11	Green snakes on a transparent background.
2310-2319µsec	Snake 12	Red, blue, and white snakes on a transparent background.
2320-2329µsec	Snake 13	Blue, magenta, and white snakes on a transparent background.
2330-2339µsec	Snake 14	White, purple, and blue snakes on a transparent background.
2340-2349µsec	Snake 15	White snakes on a transparent background.
2350-2359µsec	Rainbow Snake 1	A single, full-color rainbow snake.
2360-2369µsec	Rainbow Snake 2	Two, full-color rainbow snakes.
2370-2379µsec	Rainbow Snake 3	Three, full-color rainbow snakes.
2380-2389µsec	Rainbow Snake 4	Three snakes constrained to a hue of 170-235.
2390-2399µsec	Rainbow Snake 5	Three snakes constrained to a hue of 0-40.
2400-2409µsec	Rainbow Snake 6	Three snakes constrained to a hue of 100-160.
2410-2419µsec	Rainbow Snake 7	Three snakes constrained to a hue of 0-180.
2420-2429µsec	Rainbow Snake 8	Three snakes constrained to a hue of 180-360.
2430-2439µsec	Rainbow Snake 9	Same as Rainbow Snake 1 but direction is inverted.
2440-2449µsec	Rainbow Snake 10	Same as Rainbow Snake 2 but direction is inverted.
2450-2459µsec	Rainbow Snake 11	Same as Rainbow Snake 3 but direction is inverted.
2460-2469µsec	Rainbow Snake 12	Same as Rainbow Snake 4 but direction is inverted.
2470-2479µsec	Rainbow Snake 13	Same as Rainbow Snake 5 but direction is inverted.
2480-2489µsec	Rainbow Snake 14	Same as Rainbow Snake 6 but direction is inverted.
2490-2500µsec	Rainbow Snake 15	Same as Rainbow Snake 7 but direction is inverted.