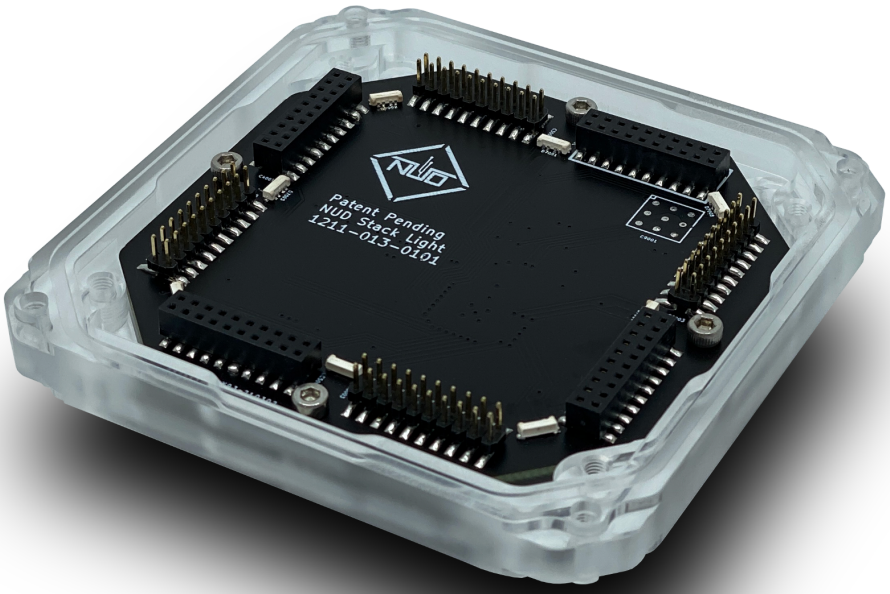
LED Peripheral Manual



https://www.nodeudesign.com
sales@nodeudesign.com

Table of Contents

1.1. Electrical Characteristics

Absolute Maximum Ratings	
	Max
VIN	36V
V(ESD) Electrostatic discharge	16000V
Temperature (Storage & Operating)	-40℃ - 80℃
Operating Temperature	-20℃ - 80℃
LED Power Supply Ripple Frequency	1kHz

Recommended Maximum Ratings			
	Min	Typical	Max
Operating Voltage	7.5V	-	36V

2. Overview

The LED Peripheral is designed to display status or event information about your NUD stack. Multiple programmable LED configurations exist, including Single LED control, simultaneous LED control, Blink Mode, Spin Mode, and State Mode. Each mode includes its own set of custom configurations, allowing the user to customize how the LED Peripheral indicates information.

2.1. Hardware

LEDs:

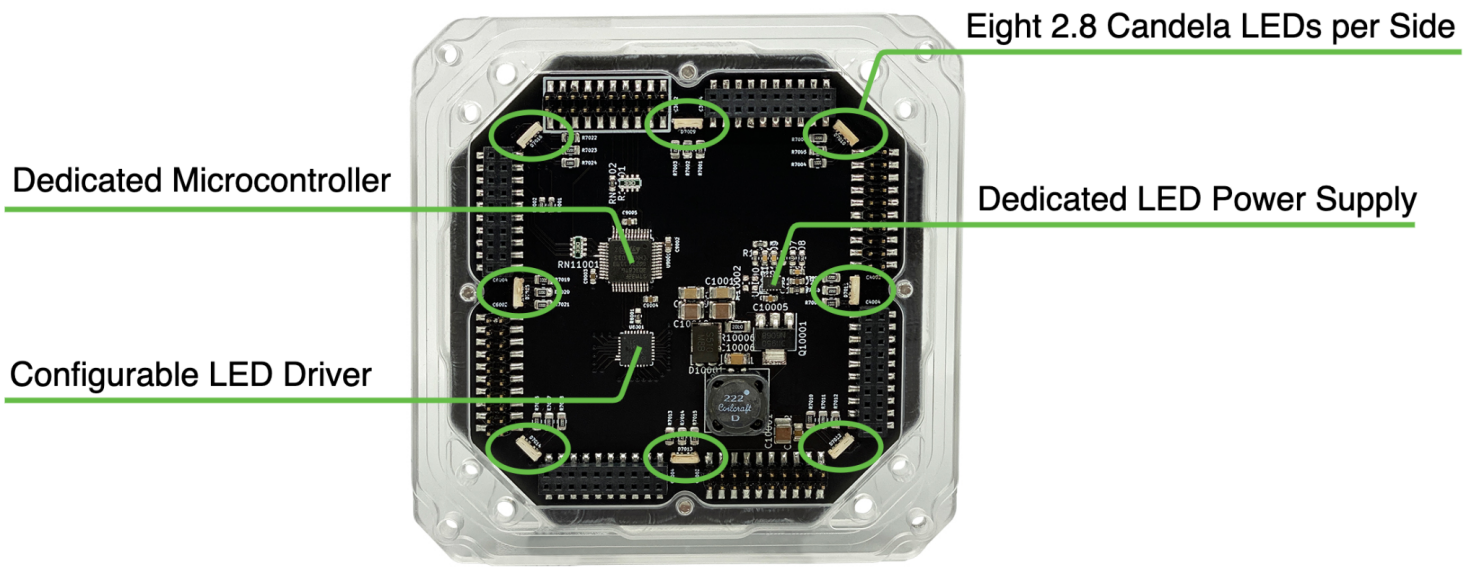
Eight high-performance RGB LEDs surrounding the perimeter of the LED Peripheral - each capable of creating up to 2.8 candles of luminous intensity. The LEDs high illumination ensures color and pattern identification for the user.

Dedicated Power Supply:

The LED Peripheral contains a dedicated power supply to maintain consistent power levels for the LEDs regardless of input voltage. The more LEDs that are enabled, the more power that will be drawn from VIN. The color white (255, 255, 255) draws the most power, requiring all outputs to be enabled for each tri-color LED package.

IP-67 Sealed Carrier:

The LED Peripheral's sealed acrylic carrier provides a translucent waterproof and dustproof enclosure for the LED stack. The translucent barrier allows ample and diffuse light through while protecting the inner electronics from hostile environments. The acrylic carrier is compatible with all NUD products.



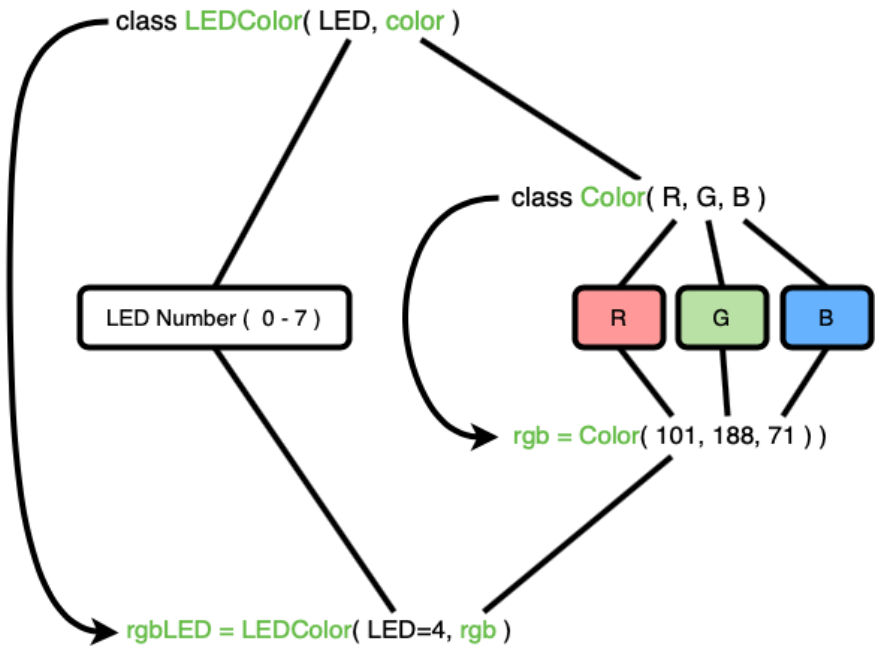
RGB Data Types:

`Color(R, G, B)`

`Color` is a class simply containing RGB values. An instance of Color may be declared and passed into functions which only require RGB values such as `AllColors()`. RGB members may be accessed individually, or reinitialized to new values, for easy color transitions.

`LEDColor(LED, Color)`

`LEDColor` is the parent class of `Color`. LEDColor contains both a color class instance as well as an associated LED number. It's LED Number and RGB Values are both declarable in a single line command, or accessible individually.



An instance of Color may be initialized within the LEDColor class:

```
rbgLED = LEDColor( LED=4, color=Color(101, 188, 71) )
```

Class members may be accessed individually:

```
rgb = Color(101, 188, 71)
print( rgb )

rgb.R = 255
rgb.G = 255
rgb.B = 255
print( rgb )
```

```
Output = 101, 188, 71
         255, 255, 255
```

Control Modes:

Normal:

When `ControlMode` is set to `NORMAL`, the LED Peripheral is capable of configuring RGB values to all 8 sides of it's LED banks individually, as well as simultaneously. Using the `OneColor()` function, the user may configure singular LED RGB values. If simultaneous configuration of all LEDs to a specific RGB color is desired, `AllColors()` will quickly and efficiently accomplish this.

State:

State Mode is available if specific LED patterns or timings are necessary for the application. Once `ControlMode` is set to `STATE`, RGB values with their LED numbers as well as `StateModeTime` may be assigned to any one of 8 states. Thus, each state becomes a user defined custom LED pattern with user controlled or timed display. When `StateModeEnable` is set to `STATE_ON`, state transitions are enabled. Two control methods are currently available to facilitate state transitions. Manually transitioning the LED Peripheral's states is accomplished first by configuring `StateMode` to `USER_UPDATE` and then setting `StateModeState` to the desired state. The LED Peripheral may be configured to automatically transition states based on a state time value by setting `StateMode` to `TIMED_UPDATE`. The LED Peripheral will begin to incrementally cycle through each of the 8 states at rate `StateModeTime`. StateModeEnable may be toggled on or off at any time during the LED Peripheral's operation to synchronize LED color preferences, or create pauses in state mode state incrementing.

Blink:

Blink Mode is an extension of State Mode with a single configurable state & blink time, as well as an added duty cycle feature to allow precise on / off times during blink periods. When `ControlMode` is set to `BLINK`, RGB values with their LED numbers and `BlinkTime` may be configured. `BlinkTime` describes the millisecond blinking period value of the blink state LED configuration. When `BlinkModeEnable` is set to `BLINK_ON`, the LED peripheral will begin blinking. `BlinkModeEnable` may be used to synchronize a blink state configuration by configuring the blink parameters first, then enabling blink modes output.

Spin:

Spin Mode is another extension of State Mode, where a configured single state of LEDs may rotate clockwise or counterclockwise around the LED Peripheral at a defined rate of rotation. `SpinRate` describes the millisecond time it takes for a state of LED's to rotate once around the perimeter of the LED Peripheral. `SpinDirection` controls the direction of LED rotation around the LED Peripheral. To configure Spin Mode, set `ControlMode` to `SPIN`. Then, a state of specified LED RGB values may then be configured using the `SpinColor()` method. When `SpinModeEnable` is set to `SPIN_ON`, the configured state of LEDs will then begin to rotate around the perimeter of the LED Peripheral. `SpinModeEnable` may also be used to pause LED rotation if necessary.

Note: In any `ControlMode` transition, state LED configurations and timings must be reinitialized.

4.1. ControlMode(Fetch, Store)

Description:

`ControlMode` configures the LED Peripheral's method of controlling LEDs.

`ChannelMode.fetch()`: Queues the control mode value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's control mode value. This function returns nothing.
`ChannelMode.value()`: Reads the most recently updated control mode value. This function must be called after a fetch(), and returns the LED Peripheral's control mode value.
`ChannelMode.store()`: Configures the LED Peripheral's control mode between Normal, Blink, Spin and State modes. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Control Mode: 0 : NORMAL 1 : BLINK 2 : SPIN 3 : STATE	Parameter 1: Control Mode: 0 : NORMAL 1 : BLINK 2 : SPIN 3 : STATE

Example:

```
LED.properties.ControlMode.store( ControlModes.SPIN )
LED.properties.ControlMode.fetch( )
ledControlMode = LED.properties.ControlMode.value( )
print( ledControlMode )
```

```
Output: 3
```

The code first configures the LED Peripheral's control mode to spin mode by calling `ControlMode.store( ControlModes.SPIN )`. The code requests the LED peripheral to reply back with the present control mode by calling `ControlMode.fetch( )`. Next, the LED Peripheral's control mode value is saved to the variable "`ledControlMode`" by calling `ControlMode.value( )`. Finally, the LED Peripheral's `ControlMode` value is printed in the terminal.

4.2. OneColor(Fetch, Store)

Description:

`OneColor` configures a specified LED's RGB profile on the LED Peripheral for use in Normal control mode. OneColor utilizes the data type `LEDColor()`.

`OneColor.fetch()`: Queues the selected LEDs RGB values to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the selected LEDs RGB values. This function returns nothing.
`OneColor.value()`: Reads the most recently updated selected LEDs RGB values. This function must be called after a fetch(), and returns the selected LEDs RGB values.
`OneColor.store()`: Sets the selected LED to the given RGB value.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: LED: (0 - 7)	Parameter 1: LED: (0 - 7)	Response 1: RGB Values: R : 0 - 255 G : 0 - 255 B : 0 - 255	Parameter 1: LEDColor( )

Example:

```
LED.properties.ControlMode.store( ControlModes.NORMAL )
LED.properties.OneColor.store( LEDColor( LED=4, color=Color( 101, 188, 71 ) ) )
LED.properties.OneColor.fetch( 4 )
ledColor = LED.properties.OneColor.value( 4 )
print( ledColor )
```

```
Output: 101, 188, 71
```

The code first configures the LED Peripheral's control mode to normal by calling `LEDPeripheral.properties.ControlMode.store( ControlModes.NORMAL )`. Then, LED 4 is set to the RGB value `101, 188, 71` by calling `OneColor.store( LED=4, color=Color( 101, 188, 71 ) )`. Next, the code requests the LED Peripheral to reply back with LED 4's RGB value by calling `OneColor.fetch( 4 )`. The RGB values are then saved to the variable "`ledColor`" by calling `OneColor.value( 4 )`. Finally, LED 4's RGB values are printed in the terminal.

4.3. AllColors(Store)

Description:

`AllColors` configures all LED's to an RGB color value, utilizing the `Color()` class. AllColors is intended as a quick way to set all LED's, saving the time of configuring each LED individually.

`AllColors.store()`: Sets all 8 sides of the LED Peripheral to the given RGB value.

Store:
Parameter 1: Color (R, G, B)

Example:

```
LEDPeripheral.properties.AllColors.store( color=Color( 101, 188, 71 ) )
```

The code simply sets all LED's to the RGB value `101, 188, 71` by calling `AllColors.store( color=Color( 101, 188, 71 ) )`.

5. Blink Mode

5.1. BlinkRate(Fetch, Store)

Description:

`BlinkRate` sets the millisecond time of the blink period (on + off).

`BlinkRate.fetch()`: Queues the blink rate value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's blink rate value. This function returns nothing.
`BlinkRate.value()`: Reads the most recently updated blink rate value. This function must be called after a fetch(), and returns the LED Peripheral's blink rate value.
`BlinkRate.store()`: Set's the blink rate of the LED Peripheral's LEDs. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Blink Rate: (0 - 4,294,967,295) [ms]	Parameter 1: Blink Rate: (0 - 4,294,967,295) [ms]

Example:

```
LEDPeripheral.properties.ControlMode.store( ControlModes.BLINK )
LEDPeripheral.properties.BlinkRate.store( 200 )
LEDPeripheral.properties.BlinkRate.fetch( )
ledBlinkRate = LED.properties.BlinkRate.value( )
print( ledBlinkRate )
```

```
Output: 200
```

The code first configures the LED Peripheral's control mode to blink mode by calling `ControlMode.store( ControlModes.BLINK )`. The code then sets the LED Peripheral's blink period to 200 milliseconds by calling `BlinkRate.store( 200 )`. Next, the code requests the LED peripheral to reply back the present blink rate value by calling `BlinkRate.fetch( )`. The value is then saved to the variable "`ledBlinkRate`" by calling `BlinkRate.value( )`. Finally, the LED Peripheral's `BlinkRate` value is printed in the terminal.

5.2. BlinkColor(Fetch, Store)

Description:

`BlinkColor` configures a specified LED's RGB profile on the LED Peripheral for use in Blink control mode. BlinkColor utilizes the data type `LEDColor()`.

`BlinkColor.fetch()`: Queues the selected LEDs RGB values to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the selected LEDs RGB values. This function returns nothing.
`BlinkColor.value()`: Reads the most recently updated selected LEDs RGB values. This function must be called after a fetch(), and returns the selected LEDs RGB values.
`BlinkColor.store()`: Sets the selected LED to the given RGB value.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: LED: (0 - 7)	Parameter 1: LED: (0 - 7)	Response 1: RGB Values: R : 0 - 255 G : 0 - 255 B : 0 - 255	Parameter 1: LEDColor( )

Example:

```
LED.properties.ControlMode.store( ControlModes.BLINK )
LED.properties.BlinkColor.store( LEDColor( LED=1, color=Color(0, 255, 0 ) ) )
LED.properties.BlinkColor.fetch( 1 )
blinkColor = LED.properties.BlinkColor.value( 1 )
print( blinkColor )
```

```
Output: 0, 255, 0
```

The code first configures the LED Peripheral's control mode to blink mode by calling `LEDPeripheral.properties.ControlMode.store( ControlModes.BLINK )`. Then, LED 1 is set to the RGB value `0, 255, 0` by calling `BlinkColor.store( LED=1, color=Color( 0, 255, 0 ) )`. Next, the code requests the LED Peripheral to reply back with LED 1's RGB value by calling `BlinkColor.fetch( 1 )`. The RGB values are then saved to the variable "`blinkColor`" by calling `BlinkColor.value( 1 )`. Finally, LED 1's RGB values are printed in the terminal.

5.3. BlinkDutyCycle(Fetch, Store)

Description:

`BlinkDutyCycle` configures the LED Peripheral's blinking on / off duty cycle when ControlMode is set to Blink Mode. The LED Peripheral's duty cycle is calculated as the thousandths percentage of on-time versus off-time of the `BlinkRate`.

`BlinkDutyCycle.fetch()`: Queues the LED Peripheral's blink duty cycle value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's blink duty cycle value. This function returns nothing.

`BlinkDutyCycle.value()`: Reads the most recently updated blink duty cycle value. This function must be called after a fetch(), and returns the LED Peripheral's blink duty cycle value.

`BlinkDutyCycle.store()`: Sets the duty cycle of the LED Peripheral's blinking.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Duty Cycle: (0 - 1000) [%]	Parameter 1: Duty Cycle: (0 - 1000) [%]

Example:

```
LED.properties.ControlMode.store( ControlModes.BLINK )
LED.properties.BlinkRate.store( 1000 )
LED.properties.BlinkDutyCycle.store( 500 )
LED.properties.BlinkDutyCycle.fetch( )
blinkDutyCycle = LED.properties.BlinkDutyCycle.value( )
print( blinkDutyCycle )
```

```
Output: 500
```

The code first configures the LED Peripheral's control mode to blink mode by calling `ControlMode.store( ControlModes.BLINK )`. The code then sets the LED Peripheral's blink period to 1000 milliseconds by calling `BlinkRate.store( 1000 )`. Next, the code sets the LED Peripheral's blink duty cycle to 50% by calling `BlinkDutyCycle.store( 500 )`. Next, the code requests the LED Peripheral to reply back with the LED Peripheral's duty cycle value by calling `BlinkDutyCycle.fetch( )`. The LED Peripheral's duty cycle value is then saved to the variable "`blinkDutyCycle`" by calling `BlinkDutyCycle.value( )`. Finally, the LED Peripheral's duty cycle value is printed in the terminal.

5.4. BlinkModeEnable(Fetch, Store)

Description:

`BlinkModeEnable` enables and disables the LED Peripheral's blinking output. This function must be enabled in order to activate the LED Peripheral's blink state transition.

`BlinkModeEnable.fetch()`: Queues the LED Peripheral's blink mode enable value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's blink mode enable value. This function returns nothing.
`BlinkModeEnable.value()`: Reads the most recently updated blink mode enable value. This function must be called after a fetch(), and returns the LED Peripheral's blink mode value.
`BlinkModeEnable.store()`: Enables or disables the LED Peripheral's blink mode operation.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Blink Mode Enable: 0 : Disable 1 : Enable	Parameter 1: Blink Mode Enable: 0 : Disable 1 : Enable

Example:

```
LED.properties.ControlMode.store( ControlModes.BLINK )
LED.properties.BlinkModeEnable.store( 1 )
LED.properties.BlinkModeEnable.fetch( )
blinkModeEnable = LED.properties.BlinkModeEnable.value( )
print( blinkModeEnable )
```

```
Output: 1
```

The code first configures the LED Peripheral's control mode to blink mode by calling `ControlMode.store( ControlModes.BLINK )`. Then, the LED Peripheral's blink mode is enabled by calling `BlinkModeEnable.store( 1 )`. Next, the code requests the LED Peripheral to reply back with it's blink mode enable value by calling `BlinkModeEnable.fetch( )`. The LED Peripheral's blink mode enable value is then saved to the variable "`blinkEnable`" by calling `BlinkModeEnable.value( )`. Finally, the LED Peripheral's blink mode enable value is printed in the terminal.

6. Spin Mode

6.1. SpinRate(Fetch, Store)

Description:

`SpinRate` sets the millisecond time of the period (on + off).

`SpinRate.fetch()`: Queues the spin rate value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's spin rate value. This function returns nothing.
`SpinRate.value()`: Reads the most recently updated spin rate value. This function must be called after a fetch(), and returns the LED Peripheral's spin rate value.
`BlinkRate.store()`: Sets the spin rate of the LED Peripheral's LEDs. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Spin Rate: (0 - 4,294,967,295) [ms]	Parameter 1: Spin Rate: (0 - 4,294,967,295) [ms]

Example:

```
LED.properties.ControlMode.store( ControlModes.SPIN )
LED.properties.SpinRate.store( 750 )
LED.properties.SpinRate.fetch( )
ledSpinRate = LED.properties.BlinkRate.value( )
print( ledSpinRate )
```

```
Output: 750
```

The code first configures the LED Peripheral's control mode to spin mode by calling `ControlMode.store( ControlModes.SPIN )`. The code then sets the LED Peripheral's spin period to 750 milliseconds by calling `SpinRate.store( 750 )`. Next, the code requests the LED peripheral to reply back the present spin rate value by calling `SpinRate.fetch( )`. The value is then saved to the variable "`ledSpinRate`" by calling `SpinRate.value( )`. Finally, the LED Peripheral's `SpinRate` value is printed in the terminal.

6.2. SpinColor(Fetch, Store)

Description:

SpinColor configures a specified LED's RGB profile on the LED Peripheral for use in Spin control mode. SpinColor utilizes the data type LEDColor().

SpinColor.fetch(): Queues the selected LEDs RGB values to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the selected LEDs RGB values. This function returns nothing.
SpinColor.value(): Reads the most recently updated selected LEDs RGB values. This function must be called after a fetch(), and returns the selected LEDs RGB values.
SpinColor.store(): Sets the selected LED to the given RGB value.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: LED: (0 - 7)	Parameter 1: LED: (0 - 7)	Response 1: RGB Values: R : 0 - 255 G : 0 - 255 B : 0 - 255	Parameter 1: LEDColor(LED, Color (R, G, B))

Example:

```
LED.properties.ControlMode.store( ControlModes.SPIN )
LED.properties.SpinColor.store( LEDColor( LED=2, color=Color(255, 255, 255 ) ) )
LED.properties.SpinColor.fetch( 2 )
spinColor = LED.properties.SpinColor.value( 2 )
print( spinColor )
```

```
Output: 255, 255, 255
```

The code first configures the LED Peripheral's control mode to spin mode by calling LEDPeripheral.properties.ControlMode.store(ControlModes.SPIN). Then, LED 2 is set to the RGB value 255, 255, 255 by calling SpinColor.store(LEDColor(LED=2, color=Color(255, 255, 255))). Next, the code requests the LED Peripheral to reply back with LED 2's RGB value by calling SpinColor.fetch(2). The RGB values are then saved to the variable "spinColor" by calling SpinColor.value(2). Finally, LED 1's RGB values are printed in the terminal.

6.3. SpinDirection(Fetch, Store)

Description:

SpinDirection configures the LED Peripheral's LED spinning direction.

SpinDirection.fetch(): Queues the LED Peripheral's spin direction value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's spin direction value. This function returns nothing.
SpinDirection.value(): Reads the most recently updated spin direction value. This function must be called after a fetch(), and returns the LED Peripheral's spin direction value.
SpinDirection.store(): Sets the spin direction of the LED Peripheral's blinking.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Spin Direction 0 : CCW 1 : CW	Parameter 1: Spin Direction 0 : CCW 1 : CW

Example:

```
LED.properties.ControlMode.store( ControlModes.SPIN )
LED.properties.SpinDirection.store( SpinDirections.COUNTER_CLOCKWISE )
LED.properties.SpinDirection.fetch( )
spinDirection = LED.properties.SpinDirection.value( )
print( spinDirection )
```

```
Output: 0
```

The code first configures the LED Peripheral's control mode to spin mode by calling ControlMode.store(ControlModes.SPIN). Then, the code sets the LED Peripheral's spin direction by calling SpinDirection.store(SpinDirections.COUNTER_CLOCKWISE). Next, the code requests the LED Peripheral to reply back with the LED Peripheral's spin direction value by calling SpinDirection.fetch( ). The LED Peripheral's spin direction value is then saved to the variable "spinDirection" by calling SpinDirection.value( ). Finally, the LED Peripheral's spin direction value is printed in the terminal.

6.4. SpinModeEnable(Fetch, Store)

Description:

`SpinModeEnable` enables and disables the LED Peripheral's spinning output. This function must be enabled in order to activate the LED Peripheral's spin state transition.

`SpinModeEnable.fetch()`: Queues the LED Peripheral's spin mode enable value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's spin mode enable value. This function returns nothing.
`SpinModeEnable.value()`: Reads the most recently updated spin mode enable value. This function must be called after a fetch(), and returns the LED Peripheral's spin mode enable value.
`SpinModeEnable.store()`: Enables or disables the LED Peripheral's spin mode operation.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Spin Mode Enable: 0 : Disable 1 : Enable	Parameter 1: Spin Mode Enable: 0 : Disable 1 : Enable

Example:

```
LED.properties.ControlMode.store( ControlModes.SPIN )
LED.properties.SpinModeEnable.store( 1 )
LED.properties.SpinModeEnable.fetch( )
spinModeEnable = LED.properties.SpinModeEnable.value( )
print( spinModeEnable )
```

```
Output: 1
```

The code first configures the LED Peripheral's control mode to spin mode by calling `ControlMode.store( ControlModes.SPIN )`. Then, the LED Peripheral's spin mode is enabled by calling `SpinModeEnable.store( 1 )`. Next, the code requests the LED Peripheral to reply back with it's spin mode enable value by calling `SpinModeEnable.fetch( )`. The LED Peripheral's spin mode enable value is then saved to the variable "`spinModeEnable`" by calling `SpinModeEnable.value( )`. Finally, the LED Peripheral's spin mode enable value is printed in the terminal.

7. State Mode

7.1. StateMode(Fetch, Store)

Description:

`StateMode` configures the LED Peripheral's state mode method of controlling LEDs.

`StateMode.fetch()`: Queues the state mode value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the state mode value. This function returns nothing.
`StateMode.value()`: Reads the most recently updated control mode value. This function must be called after a fetch(), and returns the LED Peripheral's state mode value.
`StateMode.store()`: Configures the LED Peripheral's state mode between User Update and Timed Update modes. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: State Mode: 0 : USER_UPDATE 1 : TIMED_UPDATE	Parameter 1: State Mode: 0 : USER_UPDATE 1 : TIMED_UPDATE

Example:

```
LED.properties.StateMode.store( ControlModes.STATE )
LED.properties.StateMode.store( StateModes.USER_UPDATE )
LED.properties.StateMode.fetch( )
stateMode = LED.properties.StateMode.value( )
print( stateMode )
```

```
Output: 0
```

The code first configures the LED Peripheral's control mode to state mode by calling `ControlMode.store( ControlModes.STATE )`. Then, the code configures the LED Peripheral's state mode to user control by calling `StateMode.store( StateModes.USER_UPDATE )`. Next, the code requests the LED peripheral to reply back with its present state mode value by calling `StateMode.fetch( )`. Next, the LED Peripheral's state mode value is saved to the variable "`stateMode`" by calling `StateMode.value( )`. Finally, the LED Peripheral's `StateMode` value is printed in the terminal.

7.2. StateModeState(Fetch, Store)

Description:

`StateModeState` is the LED Peripheral's current state for use in state ControlMode. This function may be used to force the LED Peripheral into a specified state at any time during state mode operation.

`StateModeState.fetch()`: Queues the current state value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with it's control mode value. This function returns nothing.
`StateModeState.value()`: Reads the most recently updated state value. This function must be called after a fetch(), and returns the LED Peripheral's current state value.
`StateModeState.store()`: Sets the LED Peripheral's current state to the value passed into the function. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: State: (0 - 7)	Parameter 1: State: (0 - 7)

Example:

```
LED.properties.ControlMode.store( ControlModes.STATE )
LED.properties.StateModeState.store( 7 )
LED.properties.StateModeState.fetch( )
LED = LED.properties.StateModeState.value( )
print( stateModeState )
```

```
Output: 7
```

The code first configures the LED Peripheral's control mode to state mode by calling `ControlMode.store( ControlModes.STATE )`. The code then sets the LED Peripheral's current state to state 7 by calling `StateModeState.store( 7 )`. Next, the code requests the LED peripheral to reply back with its present state by calling `StateModeState.fetch( )`. The state value is saved to the variable "`stateModeState`" by calling `StateModeState.value( )`. Finally, the LED Peripheral's `StateModeState` value is printed in the terminal.

7.3. StateModeTime(Fetch, Store)

Description:

StateModeTime is the millisecond time for which the selected state is active when StateMode is set to TIMED_UPDATE.

StateModeTime.fetch(): Queues the selected state's state mode time value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the selected state's state mode time value. This function returns nothing.
StateModeTime.value(): Reads the most recently updated selected state's state mode time value. This function must be called after a fetch(), and returns the selected states state mode time value.
StateModeTime.store(): Sets the selected state's state mode time. This function returns nothing.

Fetch:	**Value(Response):**		**Store:**	
Parameter 1: State: (0 - 7)	Parameter 1: State: (0 - 7)	Response 1: State Mode Time: (0 - 4,294,967,295) [ms]	Parameter 1: State: (0 - 7)	Parameter 2: State Mode Time: (0 - 4,294,967,295) [ms]

Example:

```
LED.properties.ControlMode.store( ControlModes.STATE )
LED.properties.StateModeTime.store( 1, 10000 )
LED.properties.StateModeTime.fetch( 1 )
stateModeTime = LED.properties.StateModeTime.value( 1 )
print( stateModeTime )
```

```
Output: 10000
```

The code first configures the LED Peripheral's control mode to state mode by calling ControlMode.store(ControlModes.STATE). The code then sets the LED Peripheral's state 1 state mode time to 10 seconds by calling StateModeTime.store(1, 10000). Next, the code requests the LED peripheral to reply back state 1's state mode time value by calling StateModeTime.fetch(1). The value is then saved to the variable "stateModeTime" by calling StateModeTime.value(1). Finally, the LED Peripheral's state 1 StateModeTime value is printed in the terminal.

7.4. StateModeLastState(Fetch)

Description:

StateModeLastState is the previous state before the last state transition.

StateModeLastState.fetch(): Queues the previous state mode's state value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the previous state mode's state value. This function returns nothing.
StateModeLastState.value(): Reads the previous state mode's state value. This function must be called after a fetch(), and returns the previous state mode's state value.

Fetch:	**Value(Response):**	
Parameter 1: (None)	Parameter 1: (None)	Response 1: State: (0 - 7)

Example:

```
LED.properties.ControlMode.store( ControlModes.STATE )
LED.properties.StateModeState.store( 1 )
time.sleep_ms( 100 )
LED.properties.StateModeState.store( 2 )
LED.properties.StateModeLastState.fetch( )
stateModeLastState = LED.properties.StateModeLastState.value( )
print( stateModeLastState )
```

```
Output: 1
```

The code first configures the LED Peripheral's control mode to state mode by calling ControlMode.store(ControlModes.STATE). The code then sequentially sets the LED Peripheral's states from 1 to 2. Next, the code requests the LED peripheral to reply back the previous state mode's state value by calling StateModeLastState.fetch( ). The value is then saved to the variable "stateModeLastState" by calling StateModeLastState.value( ). Finally, the LED Peripheral's StateModeLastState value is printed in the terminal.

7.5. StateColor(Fetch, Store)

Description:

`StateColor` configures the selected LEDs RGB profile on the LED Peripheral for use in State control mode. StateColor utilizes the data type `LEDColor()`.

`StateColor.fetch()`: Queues the specified states selected LED RGB values to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with the specified states selected LED RGB values. This function returns nothing.
`StateColor.value()`: Reads the most recently updated specified states selected LED RGB values. This function must be called after a fetch(), and returns the specified states selected LED RGB values.
`StateColor.store()`: Sets the specified states selected LED to the given RGB value.

Fetch:		**Value(Response):**			**Store:**	
Parameter 1: State: (0 - 7)	Parameter 2: LED: (0 - 7)	Parameter 1 State: (0 - 7)	Parameter 2: LED: (0 - 7)	Response 1: RGB Values: R : 0 - 255 G : 0 - 255 B : 0 - 255	Parameter 1: State: (0 - 7)	Parameter 2: LEDColor( )

Example:

```
LED.properties.ControlMode.store( ControlModes.STATE )
LED.properties.StateColor.store( 5, LEDColor( LED=1, color=Color( 101, 188, 71 ) ) )
LEDPeripheral.properties.StateColor.fetch( 5, 1 )
stateLEDColor = LED.properties.StateColor.value( 5, 1 )
print( stateColor )
```

```
Output: 0, 255, 0
```

The code first configures the LED Peripheral's control mode to state mode by calling `ControlMode.store( ControlModes.STATE )`. Then, state 5's LED 1 is set to the RGB value `101, 188, 71` by calling `StateColor.store( 5, LEDColor( LED=1, color=( 101, 188, 71 ) ) )`. Next, the code requests the LED Peripheral to reply back with state 5's LED 1 RGB value by calling `StateColor.fetch( 5, 1 )`. The RGB values are then saved to the variable "`stateLEDColor`" by calling `StateColor.value( 5, 1 )`. Finally, state 5's LED 1 RGB values are printed in the terminal.

7.6. StateModeEnable(Fetch, Store)

Description:

`StateModeEnable` enables and disables the LED Peripheral's state transitions in `TIMED_UPDATE` `StateMode`.

`StateModeEnable.fetch()`: Queues the LED Peripheral's state mode enable value to be transmitted to the brain. The LED Peripheral will then reply back to the Brain with its state mode enable value. This function returns nothing.
`StateModeEnable.value()`: Reads the most recently updated state mode enable value. This function must be called after a fetch(), and returns the LED Peripheral's state mode enable value.
`StateModeEnable.store()`: Enables or disables the LED Peripheral's spin mode operation.

Fetch:	**Value(Response):**		**Store:**
Parameter 1: (None)	Parameter 1: (None)	Response 1: Spin Mode Enable: 0 : Disable 1 : Enable	Parameter 1: Spin Mode Enable: 0 : Disable 1 : Enable

Example:

```
LED.properties.ControlMode.store( ControlModes.STATE )
LED.properties.StateModeEnable.store( 1 )
LED.properties.StateModeEnable.fetch( )
stateModeEnable = LED.properties.StateModeEnable.value( )
print( stateModeEnable )
```

```
Output: 1
```

The code first configures the LED Peripheral's control mode to state mode by calling `ControlMode.store( ControlModes.STATE )`. Then, the LED Peripheral's state mode is enabled by calling `StateModeEnable.store( 1 )`. Next, the code requests the LED Peripheral to reply back with its state mode enable value by calling `StateModeEnable.fetch( )`. The LED Peripheral's state mode enable value is then saved to the variable "`stateModeEnable`" by calling `StateModeEnable.value( )`. Finally, the LED Peripheral's state mode enable value is printed in the terminal.

1.1. Importing the LED Peripheral

```
import NUD.System as s

import NUD.Peripheral's.LED.LED as l
import NUD.Peripheral's.LED.Values as lv
import NUD.Peripheral's.LED.Types as lt

system = s.System()
LED = l.LED()

system.attachPeripheral( LED, s.SideSlot.SIDE_1_SLOT_A )
```

1.2. Configuring a Blinking LED in Normal Control Mode

```
# Import and Attach Setup

LED.Properties.ControlMode.store( lt.ControlModes.NORMAL )
for i in range( 10 )
   LED.Properties.OneColor.store( LEDColor( LED = 1, color = Color( 101, 188, 71 ) ) )
   time.sleep_ms( 500 )
   LED.Properties.OneColor.store( LEDColor( LED = 1, color = Color( 0, 0, 0 ) ) )
   time.sleep_ms( 500 )

# The above code configures LED 1 to blink at a 1s period for 10 seconds.
```

1.3. Configuring a Blinking LED in Blink Control Mode

```
# Import and Attach Setup

LED.Properties.ControlMode.store( lt.ControlModes.BLINK )
LED.Properties.BlinkColor.store( LEDColor( LED = 5, color = Color( 101, 188, 71 ) ) )
LED.Properties.BlinkRate.store( 500 )
LED.Properties.BlinkDutyCycle.store( 500 )
LED.Properties.BlinkModeEnable.store( 1 )

# The above code configures LED 5 to blink at a 500ms period at 50% Duty Cycle.
```

1.4. Configuring Multiple Spinning LEDs in Spin Control Mode

```
# Import and Attach Setup

LED.Properties.ControlMode.store( lt.ControlModes.SPIN )
LED.Properties.SpinColor.store( LEDColor( LED = 2, color = Color( 255, 255, 255 ) ) )
LED.Properties.SpinColor.store( LEDColor( LED = 3, color = Color( 101, 188, 71 ) ) )
LED.Properties.SpinRate.store( 2000 )
LED.Properties.SpinDirection.store( lt.SpinDirections.CLOCKWISE )
LED.Properties.SpinModeEnable.store( 1 )

# The above code configures LEDs 2 & 3 to spin clockwise at a 2s rotation period.
```

1.5. Configuring State Mode Transitions in Timed Update Mode

```
# Import and Attach Setup

LED.Properties.ControlMode.store( lt.ControlModes.STATE )
LED.Properties.StateMode.store( lt.StateModes.TIMED_UPDATE )

# Configure State 0
LED.Properties.StateColor.store( 0, LEDColor( LED = 1, color = Color( 101, 188, 71 ) ) )
LED.Properties.StateColor.store( 0, LEDColor( LED = 2, color = Color( 101, 188, 71 ) ) )
LED.Properties.SpinColor.store( 0, LEDColor( LED = 3, color = Color( 101, 188, 71 ) ) )
LED.Properties.StateModeTime.store( 0, 2000 )

# Configure State 1
LED.Properties.StateColor.store( 1, LEDColor( LED = 4, color = Color( 0, 0, 255 ) ) )
LED.Properties.StateColor.store( 1, LEDColor( LED = 5, color = Color( 0, 0, 255 ) ) )
LED.Properties.SpinColor.store( 1, LEDColor( LED = 6, color = Color( 0, 0, 255 ) ) )
LED.Properties.StateModeTime.store( 1, 1000 )

LED.Properties.StateModeEnable.store( 1 )

# The above code configures State 0 to enable LEDs 1, 2 & 3 to light for 2s, then switch
to State 1 after 2 seconds, where LED’s 4, 5, & 6 are enabled for 1 second.
```