TL50 USB API (.NET Standard version)

v1.0.0

Generated by Doxygen 1.8.16

Chapter 1

TL50 Pro Tower Light with USB: library API (.NET Standard version)

1.1 Introduction

Banner has created a software library for controlling the "TL50 Pro Tower Light with USB". This software is created as a library that runs on .NET Standard supported environments, including Windows, Linux, macOS, and more. The library is delivered as a NuGet package (.nupkg). This document describes the application programming interface (API).

1.2 Getting Started

In your host application, reference the NuGet package, and utilize the provided API to control the tower light. View the documentation for Banner.TL50.TL50Usb.

1.3 Dependency - .NET Standard

This software library uses some functionality covered by the .NET Standard 2.1 specification, or another version of .NET with the required functionality. It can be used in any application that runs on a platform that supports .N↩ ET Standard, for instance a .NET Core application. Individual components that the library uses are described as dependencies in the NuGet package. The choice depends on how you want to incorporate the library into your software, and what platform it will run on.

1.4 Example Project

A simple example of how to add the library to an application is shown. This application is a project in a Visual Studio solution. `Visual Studio Community Edition` is a free tool that can open these.

1.5 Troubleshooting - Linux permissions

On Linux systems, often the required USB port has access restrictions. The user account may need to be added the correct group, or use "sudo" to be able to access the port. Similarly, make sure that the application file being ran has execution privileges for the current user (e.g. "chmod 774").

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Banner Namespace Reference

Namespace for content related to the TL50 Pro Tower Light with USB.

4.1.1 Detailed Description

Namespace for content related to the TL50 Pro Tower Light with USB.

4.2 Banner.Binary Namespace Reference

4.3 Banner.Binary.Extensions Namespace Reference

Classes

- class **BinaryReaderExtensions**

  Extension methods for BinaryReader
- class **BinaryWriterExtensions**

  Extension methods for BinaryWriter
- class **BitwiseExtensions**

  Extension methods for bitwise operations

4.4 Banner.Binary.Util Namespace Reference

Classes

- class **BitwiseUtils**

  Static bitwise utility methods

4.5 Banner.Core Namespace Reference

Classes

- class **Specification**

  A generic specification used in domain-driven-design
- class **ValueEnum**

  Abstract base for class-based enumerations
- class **ValueObject**

  A immutable domain value object

4.6 Banner.Core.Extensions Namespace Reference

Classes

- class **OnesCompliment16Extensions**

  Extension methods for computing the 16-bit one's compliment value of a buffer
- class **ResourceManagerExtensions**

  Extension methods for ResourceManager
- class **SemaphoreSlimExtensions**

  Extension methods for SemaphoreSlim

4.7 Banner.Core.Reflection Namespace Reference

Classes

- class **ReflectionExtensions**

  Extension methods for reflection types

4.8 Banner.IO Namespace Reference

Classes

- class **BitMemoryStream**
- class **ClrStreamWrapper**

  Wraps a Stream instance as a IProtocolStream
- interface **IBinaryReadable**

  API for reading raw binary values from a stream
- interface **IProtocolStream**

  An asynchronous I/O stream for communicating with external devices
- class **LoopBackProtocolStream**

  A thread-safe loopback implementation of IProtocolStream
- class **NullModemStream**

  A null-modem that virtually connects two stream endpoints

4.9 Banner.IO.BannerBus Namespace Reference

4.10 Banner.IO.BannerBus.Abstractions Namespace Reference

Classes

- class **BannerBusAddress**

  A BannerBus address
- class **BannerBusConstants**

  Various BannerBus constant values
- class **BannerBusVersion**

  Enumeration of the different BannerBus versions

4.11 Banner.IO.BannerBus.Client Namespace Reference

Classes

- class **BannerBusClient**
- class **BannerBusDataExpectation**

  An data expectation for validating a IBannerBusResponse
- class BannerBusException

  An exception raised by the BannerBus protocol
- interface **IBannerBusClient**

  A BannerBus client that provides the ability to asynchronously issue requests to a remote device.
- interface **IBannerBusPacket**

  The raw frame data in a BannerBus transmission
- interface **IBannerBusRequest**

  A BannerBus request object
- interface **IBannerBusResponse**

  The response received during a BannerBus request
- interface **IBannerBusResult**

  A deserialized result from a IBannerBusRequest
- interface **IInterceptor**

  API for getting the request and response interceptors
- class **Interceptor**
- interface **IRequestInterceptor**

  API for adding and removing interceptors to the request pipeline which can modify the IBannerBusRequest instance before being sent
- interface **IRequestProcessor**

  Marker interface to join the IRequestTransport and IInterceptor interfaces
- interface **IResponseInterceptor**

  API for adding and removing interceptors to the response pipeline which can modify the IBannerBusResponse instance before being returned

4.12 Banner.IO.BannerBus.Client.Extensions Namespace Reference

Classes

- class **BannerBusClientExtensions**

  Extension methods for IBannerBusClient
- class **BannerBusRequestExtensions**

  Extension methods for a BannerBusRequest
- class **ResponseValidationFlagsExtensions**

  Extension methods for ResponseValidationFlags

4.13 Banner.IO.BannerBus.Client.Internal Namespace Reference

Classes

- class **BannerBusRequest**
- class **BannerBusResponse**
- class **BannerBusResult**
- interface **IRequestReader**
- interface **IRequestTransport**

  Transport object responsible for physically transmitting the request and receiving the returned data.
- interface **IRequestWriter**

  Transmits the contents of a BannerBus request
- class **ReceivedPacket**
- class **RequestLoop**
- class **RequestReader**
- class **RequestTransport**
- class **RequestWriter**

4.14 Banner.IO.BannerBus.Host Namespace Reference

Classes

- class **BannerBusHost**
- interface **IBannerBusContext**

  The BannerBus context for the middleware pipeline
- interface **IBannerBusHost**

  The BannerBus host
- interface **IBannerBusRequest**

  Request instance for the BannerBus middleware pipeline
- interface **IBannerBusResponse**

  The response instance for the BannerBus middleware pipeline
- interface **IHostBuilder**

  API for configuring a BannerBus Host's request pipeline.
- interface **IMiddleware**

  Interface for a middleware class implementation

4.15 Banner.IO.BannerBus.Host.Extensions Namespace Reference

Classes

- class **MapExtensions**

  Extension methods for the MapMiddleware.
- class **MapMiddleware**

  Represents a middleware that maps a request path to a sub-request pipeline.
- class **MapOptions**

  Options for the MapMiddleware.
- class **ProtocolExtensions**

  Extension methods for validating the BannerBus protocol
- class **ResponseWritingExtensions**

  Extension methods for writing BannerBus responses
- class **RunExtensions**

  Extension methods for adding terminal BannerBus middleware.
- class **UseExtensions**

  Extension methods for adding BannerBus middleware.

4.16 Banner.IO.BannerBus.Host.Internal Namespace Reference

Classes

- class **BannerBusContext**
- class **BannerBusRequest**
- class **BannerBusResponse**
- class **HostBuilder**

4.17 Banner.IO.Extensions Namespace Reference

Classes

- class **IProtocolStreamExtensions**

  Extension methods for IProtocolStream
- class **StreamExtensions**

  Extensions for extending the functionality of System.IO.Stream

4.18 Banner.IO.Ports Namespace Reference

Classes

- class **BannerSerialPort**

  A serial port driver that improves some issues with SerialPort and USB serial ports.
- class **BaudRate**

  A serial port baud rate
- interface **ISerialPort**

  API for a serial port instance
- class PortInUseException

  Indicates that a port was already in use when it was opened, either by this process or a different process.
- class **PortsExtensions**

  Extension methods for BannerSerialPort types
- class **SerialPortConfiguration**

  Configuration class for BannerSerialPort

4.19 Banner.IO.Serialization Namespace Reference

Classes

- class **BinaryDto**

  A data transfer object (DTO) that can be serialized into raw binary data
- class **BinaryFieldAttribute**

  Specifies that a DTO field should be serialized
- class **BinarySerializer**

  A serializer that serialize and deserialize objects to raw binary buffers
- class **BitReader**
- class **Bits**
- class **BitWriter**
- class **DeserializationOptions**

  Options to be used during deserialization
- class **DtoField**

  A single field that exists within a binary data transfer object (DTO)
- class **DynamicDtoField**

  A DTO field that can be dynamically added at runtime
- class **FixedSizeAttribute**

  Indicates that a BinaryDTO has a fixed size
- interface **IBinarySerializable**

  An object that can be serialized into raw binary content
- interface **IDeserializationContext**

  The context used when deserializing a binary object
- interface **ISerializationContext**

  The context used when serializing a binary object
- class **PropertyDtoField**

  A DTO field backed by a class property
- class **PropertyListField**

  A property field containing a list of serializable items

4.20 Banner.IO.Serialization.Extensions Namespace Reference

Classes

- class **SerializationExtensions**

4.21 Banner.TL50 Namespace Reference

Classes

- class AllSegmentSettings

  Contains settings used in Advanced Segment Mode.
- class AllSimpleStates

  For use in Simple Segment Mode, this is used to configure the active behavior of the segments.
- class **BannerBusDto**

  Tacks on read and write command codes to a BinaryDtO.
- class CustomColor

  A user defined color. This only controls the ratios of the individual components in the resulting color. The brightness of the color is controled by the indicator's intensity setting.
- class **CustomColorDto**
- class **LevelModeParametersDto**
- class LevelSettings

  All settings needed to configure Level Mode.
- class LevelState

  Given the current LevelSettings, this is used to control how much of the tower is active.
- class **OperationModeDto**
- class OtherCustomOptions

  Less commonly used ways to customize the indication.
- class **OtherCustomOptionsDto**
- class **ProcessDataAdvancedDto**
- class **ProcessDataLevelDto**
- class **ProcessDataRunDto**
- class **ProcessDataSimpleDto**
- class RunSettings

  All the available settings for Run Mode.
- class SegmentSettings

  Available settings for a single segment.
- class **SegmentSettingsDto**
- class SimpleIndicationSettings

  For use in Simple Segment Mode, this is used to determine what indication behavior a segment will show when its state is activated.
- class **SimpleSegmentParametersDto**
- class TL50Usb

  This class provides a way to control a Banner Engineering TL50 Pro Tower Light with USB.
- class Utilities

Enumerations

- enum Color : byte {
  Color.Green = 0, Color.Red = 1, Color.Orange = 2, Color.Amber = 3,
  Color.Yellow = 4, Color.LimeGreen = 5, Color.SpringGreen = 6, Color.Cyan = 7,
  Color.SkyBlue = 8, Color.Blue = 9, Color.Violet = 10, Color.Magenta = 11,
  Color.Rose = 12, Color.White = 13, Color.CustomColor1 = 14, Color.CustomColor2 = 15 }

  The available colors for indication.
- enum SegmentAnimation : byte {
  SegmentAnimation.Off = 0, SegmentAnimation.Steady = 1, SegmentAnimation.Flash = 2, SegmentAnimation.TwoColorFlash = 3,
  SegmentAnimation.HalfHalf = 4, SegmentAnimation.HalfHalfRotate = 5, SegmentAnimation.Chase = 6,
  SegmentAnimation.IntensitySweep = 7 }

  The available styles of indication available for individual segments.
- enum Intensity : byte {
  Intensity.High = 0, Intensity.Low = 1, Intensity.Medium = 2, Intensity.Off = 3,
  Intensity.Custom = 4 }

  The brightness of indication.
- enum Speed : byte { Speed.Standard = 0, Speed.Fast = 1, Speed.Slow = 2, Speed.Custom = 3 }

  For dynamic animations, the pace that the animation progresses.
- enum FlashPattern : byte {
  FlashPattern.Normal = 0, FlashPattern.Strobe = 1, FlashPattern.ThreePulse = 2, FlashPattern.SOS = 3,
  FlashPattern.Random = 4 }

  For flashing animations, the manner in which the flashing happens.
- enum RotationalDirection : byte { RotationalDirection.Counterclockwise = 0, RotationalDirection.CLockwise = 1 }

  For dynamic animations, the direction that the animation revovles.
- enum Audible : byte { Audible.Off = 0, Audible.Steady = 1, Audible.Pulsed = 2, Audible.SOS = 3 }

  Indicates the pattern of sound that will come out of the audible segment (if present).
- enum LevelAnimation : byte { LevelAnimation.StateSteady = 0, LevelAnimation.StateFlashing = 1 }

  The style of animation used in a threshold region.
- enum ThresholdType : byte { ThresholdType.None = 0, ThresholdType.Low = 1, ThresholdType.High = 2, ThresholdType.HighAndLow = 3 }

  Which thresholds are to be used.
- enum Dominance : byte { Dominance.Disabled = 0, Dominance.Enabled = 1 }

  If a threshold region controls only segments once their threshold is active, or all segments active so far.
- enum SubsegmentStyle : byte { SubsegmentStyle.Steady = 0, SubsegmentStyle.Flashing = 1, SubsegmentStyle.Analog = 2 }

  When the current value only partially fills a segment, this determines how that segment should be displayed.
- enum UpsideDownMode : byte { UpsideDownMode.RightsideUp = 0, UpsideDownMode.UpsideDown = 1 }

  Which direction the indication starts to fill from.
- enum Mode : byte { Mode.SimpleSegment = 0, Mode.AdvancedSegment = 1, Mode.Run = 2, Mode.Level = 3 }

  The different operating modes of the device. Each mode has its own set of methods which can be used to control the indication for the mode.
- enum RunAnimation : byte {
  RunAnimation.Off = 0, RunAnimation.Steady = 1, RunAnimation.Flash = 2, RunAnimation.TwoColorFlash = 3,
  RunAnimation.HalfHalf = 4, RunAnimation.HalfHalfRotate = 5, RunAnimation.Chase = 6, RunAnimation.IntensitySweep = 7,
  RunAnimation.Scroll = 8, RunAnimation.Bounce = 9, RunAnimation.Rainbow = 10, RunAnimation.Demo = 11 }

  Animations used in Run Mode. These affect the entire stack of light segments.
- enum ShiftAnimation : byte { ShiftAnimation.Disabled = 0, ShiftAnimation.Enabled = 1 }

Indicates if the animation of an individual segment should be offset slightly from the segment below it (like a barber pole).

- enum SimpleState : byte { SimpleState.Off = 0, SimpleState.Steady = 1, SimpleState.Flashing = 2, SimpleState.Animation = 3 }

  Sets what source will be used to determine how the associated segment should indicate.

4.21.1 Enumeration Type Documentation

4.21.1.1 Audible

`enum Banner.TL50.Audible : byte [strong]`

Indicates the pattern of sound that will come out of the audible segment (if present).

Enumerator

Off	No sound.
Steady	A constant sound.
Pulsed	Even bursts of sound.
SOS	Sounds in the morse code of SOS.

4.21.1.2 Color

`enum Banner.TL50.Color : byte [strong]`

The available colors for indication.

Enumerator

Green	Green
Red	Red
Orange	Orange
Amber	Amber
Yellow	Yellow
LimeGreen	Lime green
SpringGreen	Spring green
Cyan	Cyan
SkyBlue	Sky blue
Blue	Blue
Violet	Violet
Magenta	Magenta
Rose	Rose
White	White

Enumerator

CustomColor1	User defined color 1. **See also** TL50Usb.SetCustomColor1
CustomColor2	User defined color 1. **See also** TL50Usb.SetCustomColor1

4.21.1.3 Dominance

`enum Banner.TL50.Dominance : byte [strong]`

If a threshold region controls only segments once their threshold is active, or all segments active so far.

Enumerator

Disabled	One a threshold region becomes active, its indication configuration will affect only segments of that region.
Enabled	One a threshold region becomes active, its indication configuration will affect all active segments.

4.21.1.4 FlashPattern

`enum Banner.TL50.FlashPattern : byte [strong]`

For flashing animations, the manner in which the flashing happens.

Applicable to flash and two-color flash.

Enumerator

Normal	Equal parts on and off, or color 1 and color 2.
Strobe	Short blink then a pause.
ThreePulse	Three blinks then a pause.
SOS	Morse code pattern SOS.
Random	Blinks on and off in an unpredicatable manner.

4.21.1.5 Intensity

`enum Banner.TL50.Intensity : byte [strong]`

The brightness of indication.

Enumerator

High	Strong light
Low	Dim light
Medium	Medium brightness
Off	No indication
Custom	User defined brightness. **See also** TL50Usb.SetCustomIntensity

4.21.1.6 LevelAnimation

```
enum Banner.TL50.LevelAnimation : byte [strong]
```

The style of animation used in a threshold region.

Enumerator

StateSteady	When a segment becomes active, it will be on continiously.
StateFlashing	When a segment becomes active, it will blink off and on in an even manner.

4.21.1.7 Mode

```
enum Banner.TL50.Mode : byte [strong]
```

The different operating modes of the device. Each mode has its own set of methods which can be used to control the indication for the mode.

Enumerator

SimpleSegment	Segments have indication independent of each other. Useful if you infrequently change the style of indication. **See also** TL50Usb.SetSegmentsSimpleSettings(int, SimpleIndicationSettings), TL50Usb.SetSegmentsSimpleState(AllSimpleStates)
AdvancedSegment	Segments have indication indepedent of each other. Useful if you frequently change the style of indication. This is likely the most common mode used. **See also** TL50Usb.SetSegment(int, SegmentAnimation?, Color?, Intensity?, Speed?, FlashPattern?, Color?, Int TL50Usb.SetSegments(AllSegmentSettings)

Enumerator

Run	All segments are controled by a global style of indication. **See also** TL50Usb.SetRunSettings(RunSettings)
Level	The indiciation is visualy coupled to an input signal; as the input signal increases, the indication progresses along the segments of the device. Segments may be grouped into regions based on signal thresholds, and regions may have distinct indication styles. **See also** TL50Usb.SetLevelSettings(LevelSettings), TL50Usb.SetLevelState(LevelState)

4.21.1.8 RotationalDirection

`enum Banner.TL50.RotationalDirection : byte [strong]`

For dynamic animations, the direction that the animation revovles.

Mostly for half-half rotate and chase, but also has an effect on the other dynamic animations.

Enumerator

Counterclockwise	Progresses in a counterclockwise manner.
CLockwise	Progresses in a clockwise manner.

4.21.1.9 RunAnimation

`enum Banner.TL50.RunAnimation : byte [strong]`

Animations used in Run Mode. These affect the entire stack of light segments.

Enumerator

Off	No indication.
Steady	A single solid color.
Flash	A single color blinks off and on.
TwoColorFlash	Switches between two different colors.
HalfHalf	The indication is split between two colors.
HalfHalfRotate	The indication spins, showing two different colors.
Chase	A single colored spot travels around each segment, with another color as the background.
IntensitySweep	Indication gradually changes from off to bright and back to off again, repeatedly.
Scroll	A color travels from the top segment to the bottom segment, and then starts over again at the start.

Enumerator

Bounce	A color travels from the top segment to the bottom segment, and then moved back towards the start.
Rainbow	Many colors are shown.
Demo	Indication progresses through a variety of animation styles.

4.21.1.10 SegmentAnimation

`enum Banner.TL50.SegmentAnimation : byte [strong]`

The available styles of indication available for individual segments.

Enumerator

Off	No indication.
Steady	A single solid color.
Flash	A single color blinks off and on.
TwoColorFlash	Switches between two different colors.
HalfHalf	The indication is split between two colors.
HalfHalfRotate	The indication spins, showing two different colors.
Chase	A single colored spot travels around the segment, with another color as the background.
IntensitySweep	Indication gradually changes from off to bright and back to off again, repeatedly.

4.21.1.11 ShiftAnimation

`enum Banner.TL50.ShiftAnimation : byte [strong]`

Indicates if the animation of an individual segment should be offset slightly from the segment below it (like a barber pole).

Enumerator

Disabled	No shifting.
Enabled	Shifting is enabled.

4.21.1.12 SimpleState

`enum Banner.TL50.SimpleState : byte [strong]`

Sets what source will be used to determine how the associated segment should indicate.

Enumerator

Off	No indication is present.
Steady	Indication will show the set `SimpleIndicationSettings.SimpleColor` in a solid manner.
Flashing	Indication will show the set `SimpleIndicationSettings.SimpleColor` blink at the rate of `SimpleIndicationSettings.SimpleSpeed`.
Animation	Indication will show the style described by the non-simple settings (`SimpleIndicationSettings.Animation` through SimpleIndicationSettings.Direction).

4.21.1.13 Speed

`enum Banner.TL50.Speed : byte [strong]`

For dynamic animations, the pace that the animation progresses.

Applicable to flash, two-color flash, half-half rotate, chase, intensity sweep, scroll, bounce, rainbow, and demo.

Enumerator

Standard	Medium speed
Fast	High speed
Slow	Low speed
Custom	User defined speed. **See also** TL50Usb.SetCustomFlashRate

4.21.1.14 SubsegmentStyle

`enum Banner.TL50.SubsegmentStyle : byte [strong]`

When the current value only partially fills a segment, this determines how that segment should be displayed.

Enumerator

Steady	If the Level Value partially enters the threshold region of a segment, the segment will turn on fully.
Flashing	If the Level Value partially enters the threshold region of a segment, the segment will blink.
Analog	If the Level Value partially enters the threshold region of a segment, the segment's intensity will correspond to the current Level Value's place in the threshold region.

4.21.1.15 ThresholdType

```
enum Banner.TL50.ThresholdType : byte [strong]
```

Which thresholds are to be used.

Enumerator

None	No thresholding is used. The entire tower will be in the normal region.
Low	The low thresholds will be enabled. The tower will be split between the low and normal regions.
High	The high thresholds will be enabled. The tower will be split between the normal and high regions.
HighAndLow	Both low and high thresholds will be enabled. The tower will be split between the low, normal, and high regions.

4.21.1.16 UpsideDownMode

```
enum Banner.TL50.UpsideDownMode : byte [strong]
```

Which direction the indication starts to fill from.

Enumerator

RightsideUp	A Level Value 0 corresponds to the base of the tower. As the value increases towards the full scale value, indication proceeds away from the base of the tower.
UpsideDown	A Level Value 0 corresponds to the top of the tower. As the value increases towards the full scale value, indication proceeds towards the base of the tower.

4.22 Banner.TL50.DummyBannerBusResponse Namespace Reference

Classes

- class **DummyAck**
- class **DummyResult**
- class **DummyUtilities**

Chapter 5

Class Documentation

5.1 Banner.TL50.AllSegmentSettings Class Reference

Contains settings used in Advanced Segment Mode.

Static Public Attributes

- const int MaxNumberOfLightSegments = 10

  The largest number of indication segments on a single device supported by the software.

Properties

- List< SegmentSettings > LightSegments = new List<SegmentSettings>(Utilities.InitializeArray<SegmentSettings>(MaxNuml
  `[get]`

  The settings for each indicator segment.
- Audible? AudibleSegment `[get, set]`

  The settings for the audible segment (if present).

5.1.1 Detailed Description

Contains settings used in Advanced Segment Mode.

These values are not persisted to the device, they will reset between power cycles.

See also

Mode.AdvancedSegment, TL50Usb.SetSegments(AllSegmentSettings)

5.1.2 Member Data Documentation

5.1.2.1 MaxNumberOfLightSegments

```
const int Banner.TL50.AllSegmentSettings.MaxNumberOfLightSegments = 10  [static]
```

The largest number of indication segments on a single device supported by the software.

Currently only single indicator segment models are available, but Banner may make additional models avaiable in the future.

5.1.3 Property Documentation

5.1.3.1 AudibleSegment

```
Audible?  Banner.TL50.AllSegmentSettings.AudibleSegment  [get], [set]
```

The settings for the audible segment (if present).

5.1.3.2 LightSegments

```
List<SegmentSettings> Banner.TL50.AllSegmentSettings.LightSegments = new List<SegmentSettings>(Utilities.↩
InitializeArray<SegmentSettings>(MaxNumberOfLightSegments))  [get]
```

The settings for each indicator segment.

5.2 Banner.TL50.AllSimpleStates Class Reference

For use in Simple Segment Mode, this is used to configure the active behavior of the segments.

Properties

- Audible? **AudibleState** [get, set]
- SimpleState? **Segment1** [get, set]
- SimpleState? **Segment2** [get, set]
- SimpleState? **Segment3** [get, set]
- SimpleState? **Segment4** [get, set]
- SimpleState? **Segment5** [get, set]
- SimpleState? **Segment6** [get, set]
- SimpleState? **Segment7** [get, set]
- SimpleState? **Segment8** [get, set]
- SimpleState? **Segment9** [get, set]
- SimpleState? **Segment10** [get, set]

5.2.1 Detailed Description

For use in Simple Segment Mode, this is used to configure the active behavior of the segments.

Setting a property with a value of null means that you want to leave that value unchanged from what is currently configured on the device.

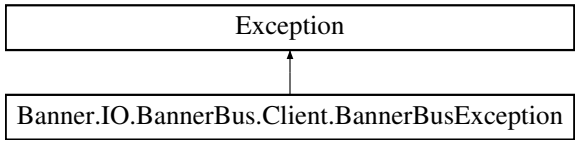
See also

Mode.SimpleSegment

5.3 Banner.IO.BannerBus.Client.BannerBusException Class Reference

An exception raised by the BannerBus protocol

Inheritance diagram for Banner.IO.BannerBus.Client.BannerBusException:



5.3.1 Detailed Description

An exception raised by the BannerBus protocol

5.4 Banner.TL50.CustomColor Class Reference

A user defined color. This only controls the ratios of the individual components in the resulting color. The brightness of the color is controled by the indicator's intensity setting.

Properties

- byte? Red `[get, set]`
  The proportion of red in the mixed color.
- byte? Green `[get, set]`
  The proportion of green in the mixed color.
- byte? Blue `[get, set]`
  The proportion of blue in the mixed color.

5.4.1 Detailed Description

A user defined color. This only controls the ratios of the individual components in the resulting color. The brightness of the color is controled by the indicator's intensity setting.

5.4.2 Property Documentation

5.4.2.1 Blue

```
byte?  Banner.TL50.CustomColor.Blue  [get], [set]
```

The proportion of blue in the mixed color.

5.4.2.2 Green

```
byte?  Banner.TL50.CustomColor.Green  [get], [set]
```

The proportion of green in the mixed color.

5.4.2.3 Red

```
byte?  Banner.TL50.CustomColor.Red  [get], [set]
```

The proportion of red in the mixed color.

5.5 Banner.Binary::Resources::Exceptions Class Reference

A strongly-typed resource class, for looking up localized strings, etc.

5.5.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

5.6 Banner.TL50.LevelSettings Class Reference

All settings needed to configure Level Mode.

Properties

- UInt16? FullScaleValue `[get, set]`

  The input signal LevelState.LevelValue that is to be considered "full", i.e. indication has reached the top of the tower light.
- Color? NormalColor `[get, set]`

  The color to be used in the normal threshold region.
- Intensity? NormalIntensity `[get, set]`

  The brightness to be used in the normal threshold region.
- LevelAnimation? NormalAnimation `[get, set]`

  The style of indication to use in the normal threshold region.
- ThresholdType? ThresholdsEnabled `[get, set]`

  Which thresholds are considered active.
- UInt16? LowThreshold `[get, set]`

  The value that marks the top of the low threshold region. Values below this are in the low threshold region, above this are in the normal region.
- Color? LowColor `[get, set]`

  The color to use in the low threshold region.
- Intensity? LowIntensity `[get, set]`

  The brightness to use in the low threshold region.
- LevelAnimation? LowAnimation `[get, set]`

  The style of indication to use in the low threshold region.
- UInt16? HighThreshold `[get, set]`

  The value that marks the start of the high threshold region. Values below this are in the normal threshold region, above this are in the high region.
- Color? HighColor `[get, set]`

  The color to use in the high threshold region.
- Intensity? HighIntensity `[get, set]`

  The brightness to use in the high threshold region.
- LevelAnimation? HighAnimation `[get, set]`

  The style of indication to use in the high threshold region.
- Speed? FlashingSpeed `[get, set]`

  For flashing segments, this controls how fast they flash.
- Dominance? Dominance `[get, set]`

  Whether threshold regions are dominant (affect their region and those below it) or not (just affect their region).
- SubsegmentStyle? SubsegStyle `[get, set]`

  What happens when the input signal is partially filling a segment.
- Color? BackgroundColor `[get, set]`

  The color used for segments when the input signal has not yet reached them.
- Intensity? BackgroundIntensity `[get, set]`

  The brightness used for segments when the input signal has not yet reached them.
- UpsideDownMode? UpsideDown `[get, set]`

  Which direction the indication starts to fill from.

5.6.1 Detailed Description

All settings needed to configure Level Mode.

Setting a property with a value of null means that you want to leave that value unchanged from what is currently configured on the device.

These values are persisted to the device.

See also

Mode.Level, LevelState, TL50Usb.SetLevelSettings(LevelSettings)

5.6.2 Property Documentation

5.6.2.1 BackgroundColor

```
Color?  Banner.TL50.LevelSettings.BackgroundColor  [get], [set]
```

The color used for segments when the input signal has not yet reached them.

5.6.2.2 BackgroundIntensity

```
Intensity?  Banner.TL50.LevelSettings.BackgroundIntensity  [get], [set]
```

The brightness used for segments when the input signal has not yet reached them.

5.6.2.3 Dominance

```
Dominance?  Banner.TL50.LevelSettings.Dominance  [get], [set]
```

Whether threshold regions are dominant (affect their region and those below it) or not (just affect their region).

5.6.2.4 FlashingSpeed

```
Speed?  Banner.TL50.LevelSettings.FlashingSpeed  [get], [set]
```

For flashing segments, this controls how fast they flash.

5.6.2.5 FullScaleValue

```
UInt16?  Banner.TL50.LevelSettings.FullScaleValue  [get], [set]
```

The input signal LevelState.LevelValue that is to be considered "full", i.e. indication has reached the top of the tower light.

5.6.2.6 HighAnimation

```
LevelAnimation?  Banner.TL50.LevelSettings.HighAnimation  [get], [set]
```

The style of indication to use in the high threshold region.

5.6.2.7 HighColor

```
Color?  Banner.TL50.LevelSettings.HighColor  [get], [set]
```

The color to use in the high threshold region.

5.6.2.8 HighIntensity

```
Intensity?  Banner.TL50.LevelSettings.HighIntensity  [get], [set]
```

The brightness to use in the high threshold region.

5.6.2.9 HighThreshold

```
UInt16?  Banner.TL50.LevelSettings.HighThreshold  [get], [set]
```

The value that marks the start of the high threshold region. Values below this are in the normal threshold region, above this are in the high region.

5.6.2.10 LowAnimation

```
LevelAnimation?  Banner.TL50.LevelSettings.LowAnimation  [get], [set]
```

The style of indication to use in the low threshold region.

5.6.2.11 LowColor

```
Color?  Banner.TL50.LevelSettings.LowColor  [get], [set]
```

The color to use in the low threshold region.

5.6.2.12 LowIntensity

```
Intensity?  Banner.TL50.LevelSettings.LowIntensity  [get], [set]
```

The brightness to use in the low threshold region.

5.6.2.13 LowThreshold

```
UInt16?  Banner.TL50.LevelSettings.LowThreshold  [get], [set]
```

The value that marks the top of the low threshold region. Values below this are in the low threshold region, above this are in the normal region.

5.6.2.14 NormalAnimation

```
LevelAnimation?  Banner.TL50.LevelSettings.NormalAnimation  [get], [set]
```

The style of indication to use in the normal threshold region.

5.6.2.15 NormalColor

```
Color?  Banner.TL50.LevelSettings.NormalColor  [get], [set]
```

The color to be used in the normal threshold region.

5.6.2.16 NormalIntensity

```
Intensity?  Banner.TL50.LevelSettings.NormalIntensity  [get], [set]
```

The brightness to be used in the normal threshold region.

5.6.2.17 SubsegStyle

```
SubsegmentStyle?  Banner.TL50.LevelSettings.SubsegStyle  [get], [set]
```

What happens when the input signal is partially filling a segment.

5.6.2.18 ThresholdsEnabled

```
ThresholdType?  Banner.TL50.LevelSettings.ThresholdsEnabled  [get], [set]
```

Which thresholds are considered active.

5.6.2.19 UpsideDown

```
UpsideDownMode?  Banner.TL50.LevelSettings.UpsideDown  [get], [set]
```

Which direction the indication starts to fill from.

5.7 Banner.TL50.LevelState Class Reference

Given the current LevelSettings, this is used to control how much of the tower is active.

Properties

- UInt16? LevelValue `[get, set]`
  This value represents the input signal.
- Audible? Audible `[get, set]`
  The current behavior of the audible segment can be controlled here.

5.7.1 Detailed Description

Given the current LevelSettings, this is used to control how much of the tower is active.

These value are not persisted; they will reset when the power is turned off

See also

Mode.Level, LevelSettings, TL50Usb.SetLevelState(LevelState)

5.7.2 Property Documentation

5.7.2.1 Audible

```
Audible?  Banner.TL50.LevelState.Audible  [get], [set]
```

The current behavior of the audible segment can be controlled here.

5.7.2.2 LevelValue

```
UInt16?  Banner.TL50.LevelState.LevelValue  [get], [set]
```

This value represents the input signal.

5.8 Banner.TL50.OtherCustomOptions Class Reference

Less commonly used ways to customize the indication.

Properties

- byte? CustomIntensityPercent `[get, set]`

  The brightness that will be used when indication has been configured to Intensity.Custom. 0-100.
- byte? CustomFlashRate_dHz `[get, set]`

  The flash rate that will be used when the indication has been configured to Speed.Custom. In dHz. 5-200.
- byte? NumberOfSegmentsForScrollAndBounce `[get, set]`

  For scroll and bounce animations, controlls the size of the moving element.
- bool? RestrictToGamut `[get, set]`

  Reduces the ranges of colors that can be produced. Enabling this will result in colors that appear a little washed out, but it will increase consistancy of colors between devices that have this enabled.

5.8.1 Detailed Description

Less commonly used ways to customize the indication.

5.8.2 Property Documentation

5.8.2.1 CustomFlashRate_dHz

```
byte?  Banner.TL50.OtherCustomOptions.CustomFlashRate_dHz  [get], [set]
```

The flash rate that will be used when the indication has been configured to Speed.Custom. In dHz. 5-200.

5.8.2.2 CustomIntensityPercent

```
byte?  Banner.TL50.OtherCustomOptions.CustomIntensityPercent  [get], [set]
```

The brightness that will be used when indication has been configured to Intensity.Custom. 0-100.

5.8.2.3 NumberOfSegmentsForScrollAndBounce

```
byte?  Banner.TL50.OtherCustomOptions.NumberOfSegmentsForScrollAndBounce  [get], [set]
```

For scroll and bounce animations, controlls the size of the moving element.

5.8.2.4 RestrictToGamut

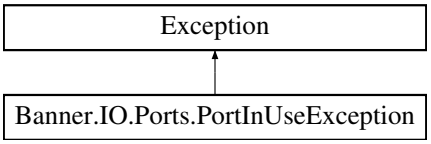
```
bool?  Banner.TL50.OtherCustomOptions.RestrictToGamut  [get], [set]
```

Reduces the ranges of colors that can be produced. Enabling this will result in colors that appear a little washed out, but it will increase consistancy of colors between devices that have this enabled.

5.9 Banner.IO.Ports.PortInUseException Class Reference

Indicates that a port was already in use when it was opened,
either by this process or a different process.

Inheritance diagram for Banner.IO.Ports.PortInUseException:



5.9.1 Detailed Description

Indicates that a port was already in use when it was opened,
either by this process or a different process.

5.10 Banner.TL50.RunSettings Class Reference

All the available settings for Run Mode.

Properties

- RunAnimation? Animation `[get, set]`

  The style of indication to use.
- Color? Color1 `[get, set]`

  The main color of the indication.
- Intensity? Intensity1 `[get, set]`

  The intensity of the main color.
- Speed? Speed `[get, set]`

  For dynamic animations, the pace that the animation progresses.
- FlashPattern? Pattern `[get, set]`

  For flashing animations, the manner in which the flashing happens.
- Color? Color2 `[get, set]`

  The second color of the indication.
- Intensity? Intensity2 `[get, set]`

  The intensity of the second color.
- ShiftAnimation? Shift `[get, set]`

  If the animation is shifted each segment from the previous segments orientation.
- RotationalDirection? Direction `[get, set]`

  The direction that the animation progresses.
- Audible? Audible `[get, set]`

  The settings for the audible segment (if present).

5.10.1 Detailed Description

All the available settings for Run Mode.

Setting a property with a value of null means that you want to leave that value unchanged from what is currently configured on the device.

These value are not persisted to the device, the will reset when power is turned off.

See also

Mode.Run

5.10.2 Property Documentation

5.10.2.1 Animation

```
RunAnimation?  Banner.TL50.RunSettings.Animation  [get], [set]
```

The style of indication to use.

5.10.2.2 Audible

```
Audible?  Banner.TL50.RunSettings.Audible  [get], [set]
```

The settings for the audible segment (if present).

5.10.2.3 Color1

```
Color?  Banner.TL50.RunSettings.Color1  [get], [set]
```

The main color of the indication.

5.10.2.4 Color2

```
Color?  Banner.TL50.RunSettings.Color2  [get], [set]
```

The second color of the indication.

Not applicable to Off, Steady, Flash, or IntensitySweep.

5.10.2.5 Direction

```
RotationalDirection?  Banner.TL50.RunSettings.Direction  [get], [set]
```

The direction that the animation progresses.

Primarily applicable to HalfHalfRotate, Chase, and IntensitySweep.

5.10.2.6 Intensity1

```
Intensity?  Banner.TL50.RunSettings.Intensity1  [get], [set]
```

The intensity of the main color.

5.10.2.7 Intensity2

```
Intensity?  Banner.TL50.RunSettings.Intensity2  [get], [set]
```

The intensity of the second color.

Not applicable to Off, Steady, Flash, or IntensitySweep.

5.10.2.8 Pattern

```
FlashPattern?  Banner.TL50.RunSettings.Pattern  [get], [set]
```

For flashing animations, the manner in which the flashing happens.

Applicable to Flash and TwoColorFlash.

5.10.2.9 Shift

```
ShiftAnimation?  Banner.TL50.RunSettings.Shift  [get], [set]
```

If the animation is shifted each segment from the previous segments orientation.

5.10.2.10 Speed

```
Speed?  Banner.TL50.RunSettings.Speed  [get], [set]
```

For dynamic animations, the pace that the animation progresses.

Applicable to Flash, TwoColorFlash, HalfHalfRotate, Chase, IntensitySweep, Scroll, and Bounce.

5.11 Banner.TL50.SegmentSettings Class Reference

Available settings for a single segment.

Properties

- Color? Color1 `[get, set]`
  The main color of the indication.
- Intensity? Intensity1 `[get, set]`
  The intensity of the main color.
- SegmentAnimation? Animation `[get, set]`
  The style of indication to use.
- Speed? Speed `[get, set]`
  For dynamic animations, the pace that the animation progresses.
- FlashPattern? Pattern `[get, set]`
  For flashing animations, the manner in which the flashing happens.
- Color? Color2 `[get, set]`
  The second color of the indication.
- Intensity? Intensity2 `[get, set]`
  The intensity of the second color.
- RotationalDirection? Direction `[get, set]`
  The direction that the animation progresses.

5.11.1 Detailed Description

Available settings for a single segment.

Setting a property with a value of null means that you want to leave that value unchanged from what is currently configured on the device.

See also

Mode.AdvancedSegment

5.11.2 Property Documentation

5.11.2.1 Animation

```
SegmentAnimation?  Banner.TL50.SegmentSettings.Animation  [get], [set]
```

The style of indication to use.

5.11.2.2 Color1

```
Color?  Banner.TL50.SegmentSettings.Color1  [get], [set]
```

The main color of the indication.

5.11.2.3 Color2

```
Color?  Banner.TL50.SegmentSettings.Color2  [get], [set]
```

The second color of the indication.

Not applicable to Off, Steady, Flash, or IntensitySweep.

5.11.2.4 Direction

```
RotationalDirection?  Banner.TL50.SegmentSettings.Direction  [get], [set]
```

The direction that the animation progresses.

Primarily applicable to HalfHalfRotate, Chase, and IntensitySweep.

5.11.2.5 Intensity1

`Intensity?` `Banner.TL50.SegmentSettings.Intensity1` `[get], [set]`

The intensity of the main color.

5.11.2.6 Intensity2

`Intensity?` `Banner.TL50.SegmentSettings.Intensity2` `[get], [set]`

The intensity of the second color.

Not applicable to Off, Steady, Flash, or IntensitySweep.

5.11.2.7 Pattern

`FlashPattern?` `Banner.TL50.SegmentSettings.Pattern` `[get], [set]`

For flashing animations, the manner in which the flashing happens.

Applicable to Flash and TwoColorFlash.

5.11.2.8 Speed

`Speed?` `Banner.TL50.SegmentSettings.Speed` `[get], [set]`

For dynamic animations, the pace that the animation progresses.

Applicable to Flash, TwoColorFlash, HalfHalfRotate, Chase, and IntensitySweep.

5.12 Banner.TL50.SimpleIndicationSettings Class Reference

For use in Simple Segment Mode, this is used to determine what indication behavior a segment will show when its state is activated.

Properties

- Color? **SimpleColor** `[get, set]`
- Speed? **SimpleSpeed** `[get, set]`
- SegmentAnimation? **Animation** `[get, set]`
- Color? **Color1** `[get, set]`
- Intensity? **Intensity1** `[get, set]`
- Speed? **Speed** `[get, set]`
- FlashPattern? **Pattern** `[get, set]`
- Color? **Color2** `[get, set]`
- Intensity? **Intensity2** `[get, set]`
- RotationalDirection? **Direction** `[get, set]`

5.12.1 Detailed Description

For use in Simple Segment Mode, this is used to determine what indication behavior a segment will show when its state is activated.

Setting a property with a value of null means that you want to leave that value unchanged from what is currently configured on the device.

See also

Mode.SimpleSegment

5.13 Banner.IO.Ports::Resources::Strings Class Reference

A strongly-typed resource class, for looking up localized strings, etc.

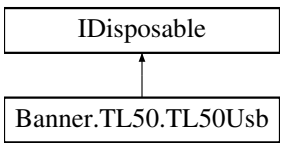
5.13.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

5.14 Banner.TL50.TL50Usb Class Reference

This class provides a way to control a Banner Engineering TL50 Pro Tower Light with USB.

Inheritance diagram for Banner.TL50.TL50Usb:



Public Member Functions

- void Init (string serialPortName)

  Initialize communications. Init is needed before communicating with the device.
- bool SetSegment (int segmentNumber=0, SegmentAnimation? animation=null, Color? color1=null, Intensity? intensity1=null, Speed? speed=null, FlashPattern? pattern=null, Color? color2=null, Intensity? intensity2=null, RotationalDirection? direction=null)

  Change indication of a single segment.
- bool SetSegments (AllSegmentSettings settings)

  Change indication of all segments.
- AllSegmentSettings GetSegments ()

  Get the current settings of the segments.
- bool SetRunSettings (RunSettings settings)

  Change the indication style used in Run mode, which has animations which span the entire tower.
- bool SetLevelSettings (LevelSettings settings)

Change the indication style used in Level mode, which dynamically fills increasing levels of the tower based on the level state.

- bool SetLevelState (LevelState state)

  Change the indication level.
- bool SetSegmentsSimpleSettings (int segmentNumber, SimpleIndicationSettings newSettings)

  Change the indication settings of a segment. This will change the device to Mode.SimpleSegment mode. This mode is similar in capability to Mode.AdvancedSegment mode, the main difference being that these indication configuration settings are persisted. If you aren't sure which you need, Mode.AdvancedSegment mode (e.g. SetSegment) is the suggested mode.
- bool SetSegmentsSimpleState (AllSimpleStates states)

  Enables or disables indication of each segment when the device is operating in Mode.SimpleSegment mode.
- bool SetCustomColor1 (CustomColor newColor)

  Change the value of Color.CustomColor1.
- bool SetCustomColor2 (CustomColor newColor)

  Change the value of Color.CustomColor2.
- bool SetCustomIntensity (int newIntensityPercent)

  Change the brightness used when indication intensity is set to Intensity.Custom.
- bool SetCustomFlashRate (int dHz)

  Change the value used when Speed.Custom is active.
- void Dispose ()

  Release the serial port and other resources.
- bool ResetToFactoryDefaults ()

  Restore the original settings of the device.

Protected Member Functions

- virtual void Dispose (bool includeManaged)

  Clean up.

Properties

- bool FullCommunication = false `[get, set]`

  Indicates if device should use read-modify-write each time a change is made. Otherwise the library will cache what it thinks the settings of the device are.

5.14.1 Detailed Description

This class provides a way to control a Banner Engineering TL50 Pro Tower Light with USB.

The tower light has different "operating modes" (see Mode).

Mode.AdvancedSegment mode is the most common mode, and uses SetSegment or SetSegments for control.

Mode.Run mode uses SetRunSettings.

Mode.Level mode uses SetLevelSettings combined with SetLevelState.

Any mode may use SetCustomColor1, SetCustomColor2, SetCustomIntensity, SetCustomFlashRate.

Create a device, initialize it with a desired serial/USB port, and turn the segment flashing two different colors.

```
using(TL50Usb device = new TL50Usb())
{
    device.Init("/dev/ttyUSB0");  // or something like "COM6" on Windows
    device.SetSegment(animation:SegmentAnimation.TwoColorFlash, color1:Color.Red, intensity1:Intensity.Low,
       color2:Color.Green);
}
```

5.14.2 Member Function Documentation

5.14.2.1 Dispose() [1/2]

```
void Banner.TL50.TL50Usb.Dispose ( )
```

Release the serial port and other resources.

5.14.2.2 Dispose() [2/2]

```
virtual void Banner.TL50.TL50Usb.Dispose (
            bool includeManaged )  [protected], [virtual]
```

Clean up.

Parameters

includeManaged	true then managed resources will be cleaned.

5.14.2.3 GetSegments()

```
AllSegmentSettings Banner.TL50.TL50Usb.GetSegments ( )
```

Get the current settings of the segments.

Returns

The current settings.

5.14.2.4 Init()

```
void Banner.TL50.TL50Usb.Init (
            string serialPortName )
```

Initialize communications. Init is needed before communicating with the device.

When this function returns, the communications will be ready.

Parameters

serialPortName	The serial port to use. On Windows systems this will be something like "COM6". On Linux systems it may be something like "/dev/ttyUSB0".

5.14.2.5 ResetToFactoryDefaults()

```
bool Banner.TL50.TL50Usb.ResetToFactoryDefaults ( )
```

Restore the original settings of the device.

Returns

true if the command was accepted. Otherwise false.

5.14.2.6 SetCustomColor1()

```
bool Banner.TL50.TL50Usb.SetCustomColor1 (
            CustomColor newColor )
```

Change the value of Color.CustomColor1.

Parameters

newColor	The color to change to.

Returns

true if command was accepted. Otherwise false.

5.14.2.7 SetCustomColor2()

```
bool Banner.TL50.TL50Usb.SetCustomColor2 (
            CustomColor newColor )
```

Change the value of Color.CustomColor2.

Parameters

newColor	The color to change to.

Returns

true if command was accepted. Otherwise false.

5.14.2.8 SetCustomFlashRate()

```
bool Banner.TL50.TL50Usb.SetCustomFlashRate (
            int dHz )
```

Change the value used when Speed.Custom is active.

Parameters

dHz	The speed in dHz, 5-200.

Returns

true if command was accepted. Otherwise false.

This setting is persisted across power cycles.

5.14.2.9 SetCustomIntensity()

```
bool Banner.TL50.TL50Usb.SetCustomIntensity (
            int newIntensityPercent )
```

Change the brightness used when indication intensity is set to Intensity.Custom.

Parameters

newIntensityPercent	The intensity to change to, 0-100.

Perceived brightness is approximately logarithmic with respect to duty cycle, i.e. as percent increases, perceived brightness increases less and less.

Returns

true if command was accepted. Otherwise false.

5.14.2.10 SetLevelSettings()

```
bool Banner.TL50.TL50Usb.SetLevelSettings (
            LevelSettings settings )
```

Change the indication style used in Level mode, which dynamically fills increasing levels of the tower based on the level state.

This will change the device to Mode.Level mode.

Parameters

settings	Describes the indication style.

Returns

true if the command was accepted. Otherwise false.

See also

SetLevelState

5.14.2.11 SetLevelState()

```
bool Banner.TL50.TL50Usb.SetLevelState (
            LevelState state )
```

Change the indication level.

This will not change the device to Mode.Level mode, that will happen with SetLevelSettings.

Parameters

state	Describes the current level. Higher value means high indication level.

Returns

true if the command was accepted. Otherwise false.

See also

SetLevelSettings

5.14.2.12 SetRunSettings()

```
bool Banner.TL50.TL50Usb.SetRunSettings (
            RunSettings settings )
```

Change the indication style used in Run mode, which has animations which span the entire tower.

This will change the device to Mode.Run mode.

Parameters

settings	Describes the indication style.

Returns

true if the command was accepted. Otherwise false.

5.14.2.13 SetSegment()

```
bool Banner.TL50.TL50Usb.SetSegment (
            int segmentNumber = 0,
            SegmentAnimation?  animation = null,
            Color?  color1 = null,
            Intensity?  intensity1 = null,
            Speed?  speed = null,
            FlashPattern?  pattern = null,
            Color?  color2 = null,
            Intensity?  intensity2 = null,
            RotationalDirection?  direction = null )
```

Change indication of a single segment.

This will change the device to Mode.AdvancedSegment mode.

You can set any parameter to null if you don't want to change that value.

Parameters

segmentNumber	The 0-based index of the segment to change (0-9). Note: on single segment devices, this value can be skipped because the default of 0 is the only relevant value.
animation	The style of indication to use.
color1	The main color of the indication.
intensity1	The intensity of the main color.
speed	The speed of the indication (not applicable to Off, Steady, or HalfHalf).
pattern	The manner in which flashing will happen (only applicable to Flash and TwoColorFlash).
color2	The second color of the indication (not applicable to Off, Steady, Flash, or IntensitySweep).
intensity2	The intensity of the second color (not applicable to Off, Steady, Flash, or IntensitySweep).
direction	The direction that the animation progresses (only applicable HalfHalfRotate, Chase, and IntensitySweep).

Returns

true if the command was accepted. Otherwise false.

5.14.2.14 SetSegments()

```
bool Banner.TL50.TL50Usb.SetSegments (
            AllSegmentSettings settings )
```

Change indication of all segments.

This will change the device to Mode.AdvancedSegment mode.

You can set any part of the AllSegmentsSettings object to null if you don't want to change that entity.

Parameters

settings	Describes desired indication for each light segment, as well as behavior of the audible segment. Content for segments that don't exist on the device is ignored.

Returns

true if the command was accepted. Otherwise false.

5.14.2.15 SetSegmentsSimpleSettings()

```
bool Banner.TL50.TL50Usb.SetSegmentsSimpleSettings (
            int segmentNumber,
            SimpleIndicationSettings newSettings )
```

Change the indication settings of a segment. This will change the device to Mode.SimpleSegment mode. This mode is similar in capability to Mode.AdvancedSegment mode, the main difference being that these indication configuration settings are persisted. If you aren't sure which you need, Mode.AdvancedSegment mode (e.g. SetSegment) is the suggested mode.

Parameters

segmentNumber	The 0-based index of the segment to change (0-9).
newSettings	The indication configuration to use for the segment.

Returns

true if the command was accepted. Otherwise false.

The settings are persisted.

See also

SetSegmentsSimpleState

5.14.2.16 SetSegmentsSimpleState()

```
bool Banner.TL50.TL50Usb.SetSegmentsSimpleState (
            AllSimpleStates states )
```

Enables or disables indication of each segment when the device is operating in Mode.SimpleSegment mode.

Parameters

states	Describes the new state of each segment.

Returns

true if the command was accepted. Otherwise false.

The simple states are not persisted.

See also

SetSegmentsSimpleSettings

5.14.3 Property Documentation

5.14.3.1 FullCommunication

```
bool Banner.TL50.TL50Usb.FullCommunication = false  [get], [set]
```

Indicates if device should use read-modify-write each time a change is made. Otherwise the library will cache what it thinks the settings of the device are.

true then full communication is forced (slower, but can resolve issues if the state is out of sync).

5.15 Banner.TL50.Utilities Class Reference

Index