TL50 Pro Tower Light with USB Product Manual

Original Instructions
p/n: 218025 Rev. B
26-Feb-25

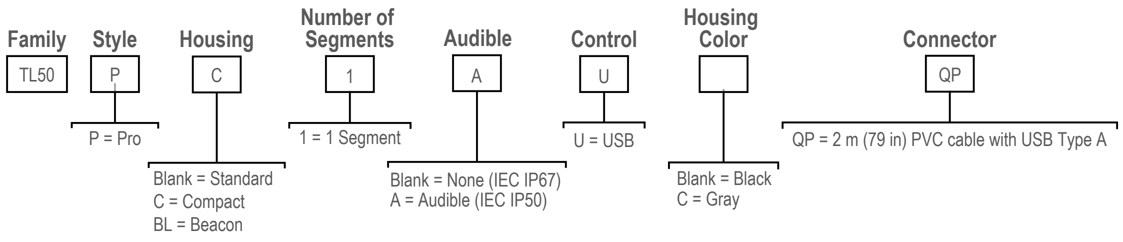
Contents

Chapter Contents

Chapter 1 Overview

The TL50 Pro Tower Light with USB is a PC-controlled device. The tower light is powered directly from the USB port and utilizes a software library to control all device functions. The device is compatible with a variety of software environments, such as C#, Python, VisualBasic, Visual C++, Labview, Windows, and Linux. Sample applications are included to show how to use the library.

Models



Chapter Contents

Chapter 2 Libraries

The functionality of the different libraries is similar, with the exception that the .NET Standard version of the library offers a few extra operating modes that are useful in niche applications. Each library comes with a help document that describes its API in greater detail.

- *Libraries/Windows/.NET/x86-32/Tl50UsbDotNetWin32.dll*
    - A .NET Framework 4.0 assembly built for 32-bit Windows platforms.
- *Libraries/Windows/.NET/x86-64/Tl50UsbDotNetx64.dll*
    - A .NET Framework 4.0 assembly built for 64-bit Windows platforms.
- *Libraries/Windows/native/dll/x86-32/Tl50UsbLibraryWin32.dll*
    - A native Windows (Win32, unmanaged) dynamic-link library for 32-bit Windows platforms. Includes supporting files, such as a header file that describes the available functions.
- *Libraries/Windows/native/dll/x86-64/Tl50UsbLibraryx64.dll*
    - A native Windows (unmanaged) dynamic-link library for 64-bit Windows platforms. Includes supporting files, such as a header file that describes the available functions.
- *Libraries/Windows/native/static_library/x86-32/Tl50UsbLibraryWin32.lib*
    - A native Windows (Win32, unmanaged) statically-linked library for 32-bit Windows platforms. Includes a supporting header file that describes the available functions.
- *Libraries/Windows/native/static_library/x86-64/Tl50UsbLibraryx64.lib*
    - A native Windows (unmanaged) statically-linked library for 64-bit Windows platforms. Includes a supporting header file that describes the available functions.
- *Libraries/.NET Standard/Banner.TL50.1.0.0.nupkg*
    - A library for .NET Standard 2.1 compatible platforms.
- *Libraries/Linux CLI/Tl50UsbCli*
    - A Linux 64-bit application that allows control of the device using a command line interface (REPL).

Chapter Contents

Chapter 3 Application Programming Interface (API)

The following gives a general description of the available functionality in the control library. The API varies slightly between the different versions of the library. Specific detailed API documentation is available alongside each library in the *Libraries/* folder.

Device Initialization

int Init()

The function of *Init* is used to find the correct COM port and initialize the device.

int InitByPort(int PortNumber)

The function *InitByPort* is used to initialize the device connected to the indicated port.

Parameters

PortNumber

The value indicates the number of the port to use.

Returns

On success, a positive value representing the port number chosen is returned. A negative value represents an error as described by *CommReturnValue*.

SetSegmentSolid

int SetSegmentSolid(int Segment, int Color)

The function sets a segment with the desired color. This setting is not persisted across power cycles.

Parameters

Segment

The configurable number of the segment on the tower light, starting from 0. For single-segment tower lights, use 0 as its value.

Color

A *ColorType*.

Returns
A *CommReturnValue*.

SetSegmentOff

int SetSegmentOff(int Segment)

The function turns off a segment.

Parameters

Segment

The configurable number of the segment on the tower light, starting from 0. For single-segment tower lights, use 0 as its value.

Returns
A *CommReturnValue*.

SetSegment

CommReturnValue SetSegment (int segment, enum SegmentAnimation animation, enum Color color1, enum Intensity intensity1, enum Speed speed, enum FlashPattern flashPattern, enum Color color2, enum Intensity intensity2, enum RotationalDirection direction)

The function changes the indication of a single segment. This setting is not persisted across power cycles.

Parameters

segment

The 0-based index of the segment to change (0-9).

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 Half-Half).

flashPattern

The manner in which flashing occurs (only applicable to Flash and Two Color Flash).

color2

The second color of the indication (not applicable to Off, Steady, Flash, or Intensity Sweep).

intensity2

The intensity of the second color (not applicable to Off, Steady, Flash, or Intensity Sweep).

direction

The direction that the animation progresses (only applicable to Half-Half Rotate, Chase, and Intensity Sweep).

Returns
The status of the command.

SetAudible

CommReturnValue SetAudible(enum Audible audible)

Change the state of the audible segment (if present). This setting is not persisted across power cycles.

Parameters

audible

The manner in which the audible segment is producing sound.

Returns

The status of the command.

SetCustomColor1

CommReturnValue SetCustomColor1(unsigned char red, unsigned green, unsigned char blue)

Change the value of Custom Color 1 (Color::CUSTOM_COLOR_1). This only controls the ratio of the colors; the intensity of indication (brightness) is controlled separately. This setting is persisted across power cycles.

Parameters

red

The proportion of red in the custom color.

green

The proportion of green in the custom color.

blue

The proportion of blue in the custom color.

Returns

The status of the command.

SetCustomColor2

CommReturnValue SetCustomColor2(unsigned char red, unsigned green, unsigned char blue)

Change the value of Custom Color 2 (Color::CUSTOM_COLOR_2). This only controls the ratio of the colors; the intensity of indication (brightness) is controlled separately. This setting is persisted across power cycles.

Parameters

red

The proportion of red in the custom color.

green

The proportion of green in the custom color.

blue

The proportion of blue in the custom color.

Returns

The status of the command.

SetCustomIntensity

CommReturnValue SetCustomIntensity(int percent)

The function changes the value used for Custom Intensity (Intensity::INTENSITY_CUSTOM). The perceived brightness is approximately logarithmic with respect to duty cycle. As the percent increases, the perceived brightness increases less and less. This setting is persisted across power cycles.

Parameters

percent

Change the value used for Custom Intensity (Intensity::INTENSITY_CUSTOM), 0-100.

Returns

The status of the command.

SetCustomSpeed

CommReturnValue SetCustomSpeed(int dHz)

Change the value used for Custom Speed (SPEED::SPEED_CUSTOM). This setting is persisted across power cycles.

Parameters

dHz

The speed in dHz, 5-200.

Returns

The status of the command.

SetSegmentAdvanced

int SetSegmentAdvanced(int Segment, char* Data)

The function turns on an individual segment with a variety of animations. It has the same functionality as *SetSegment()*, but uses a byte buffer instead of individual arguments.

Parameters

Segment

The configurable number of the segment on the tower light, starting from 0. For single-segment tower lights, use 0 as its value.

Data

An array of three bytes, whose bits mean the following (in order):

Attribute	Size	Value Type
Color 1	4	Color
Intensity 1	3	Intensity
Reserved	1	0
Animation	3	SegmentAnimation
Speed	2	Speed
Pattern	3	FlashPattern
Color 2	4	Color
Intensity 2	3	Intensity
Rotational direction	1	Rotational Direction

Returns

A *CommReturnValue*.

Deinit

int Deinit()

The function can be used to cycle the device in case of error. The USB takes time to re-enum.

Returns

A *CommReturnValue*. Always `SUCCESS` for now.

GetDllVersion

unsigned short GetDllVersion()

Returns

The version of the DLL. The upper byte is the major version and the lower byte is the minor version.

CommReturnValue

The function indicates the result of a serial command.

Values

enum CommReturnValue

Value	Numeric Code	Value	Numeric Code
SUCCESS	0	FAILED_READ	-4
FAILED_PORT_NOT_FOUND	-1	FAILED_CHECKSUM	-5
FAILED_PORT_OPEN	-2	FAILED_WITH_NACK	-6
FAILED_WRITE	-3	FAILED_NO_INIT	-7

Color

The available colors for indication.

Values

enum Color

Value	Numeric Code	Value	Numeric Code
GREEN	0	SKY_BLUE	8
RED	1	BLUE	9
ORANGE	2	VIOLET	10
AMBER	3	MAGENTA	11
YELLOW	4	ROSE	12
LIME_GREEN	5	WHITE	13
SPRING_GREEN	6	CUSTOM_COLOR_1	14
CYAN	7	CUSTOM_COLOR_2	15

SegmentAnimation

The styles of indication available for individual segments.

Values

enum SegmentAnimation

Value	Numeric Code	Value	Numeric Code
SEGMENT_OFF	0	SEGMENT_HALF_HALF	4
SEGMENT_STEADY	1	SEGMENT_HALF_HALF_ROTATE	5
SEGMENT_FLASH	2	SEGMENT_CHASE	6
SEGMENT_TWO_COLOR_FLASH	3	SEGMENT_INTENSITY_SWEEP	7

Intensity

The brightness of indication.

Values

enum Intensity

Value	Numeric Code	Value	Numeric Code
INTENSITY_HIGH	0	INTENSITY_OFF	3
INTENSITY_LOW	1	INTENSITY_CUSTOM	4
INTENSITY_MEDIUM	2		

Speed

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.

Values

enum Speed

Value	Numeric Code	Value	Numeric Code
SPEED_STANDARD	0	SPEED_SLOW	2
SPEED_FAST	1	SPEED_CUSTOM	3

FlashPattern

For flashing animations, the manner in which the flashing happens. Applicable to flash and two-color flash.

Values

enum FlashPattern

Value	Numeric Code	Value	Numerical Code
FLASH_NORMAL	0	FLASH_SOS	3
FLASH_STROBE	1	FLASH_RANDOM	4
FLASH_THREE_PULSE	2		

RotationalDirection

For dynamic animations, the direction that the animation progresses. Mostly for half-half rotate and chase, but also influences the other dynamic animations.

Values

enum RotationalDirection

Value	Numeric Code
DIRECTION_COUNTERCLOCKWISE	0
DIRECTION_CLOCKWISE	1

Audible

The value indicates the pattern of sound emitted from the audible segment (if present).

Values

enum Audible

Value	Numeric Code	Value	Numeric Code
AUDIBLE_OFF	0	AUDIBLE_PULSED	2
AUDIBLE_STEADY	1	AUDIBLE_SOS	3

Chapter Contents

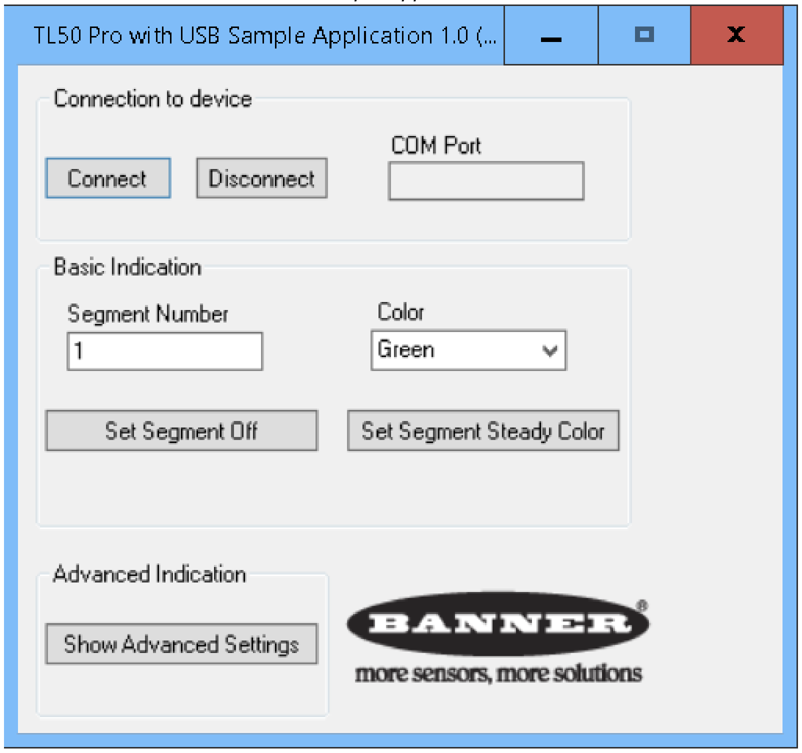
Chapter 4 Sample Applications

These are examples projects that show how to use the library.

C#WinForms Sample Application

An advanced example that offers a graphical user interface with the ability to configure the light. It uses the native x64 Windows DLL.

Sample Application



1. Connect the light to the PC.
2. Run the application.
3. Click **Initialize** and **Set Segment Steady Color** to turn the light on.

A Visual Studio project is provided at *Examples/Tl50UsbWinFormsExample/Tl50UsbWinFormsExample.csproj*.

C#WPF Sample Application

A basic example with a one-button GUI that turns on the light when pressed. It uses the x64 .NET Framework DLL.

A Visual Studio project is provided at *Examples/Tl50UsbWpfExample/ Tl50UsbWpfExample.csproj*.

C console Sample Application

A basic example that runs a script that turns on the light. It uses the native x64 static library.

A Visual Studio project is provided at *Examples/Tl50UsbCExample/ Tl50UsbCExample.vcxproj*.

C# .NET Core Sample Application

A basic example that turns on the light. It shows how to integrate the .NET Standard version of the control library.

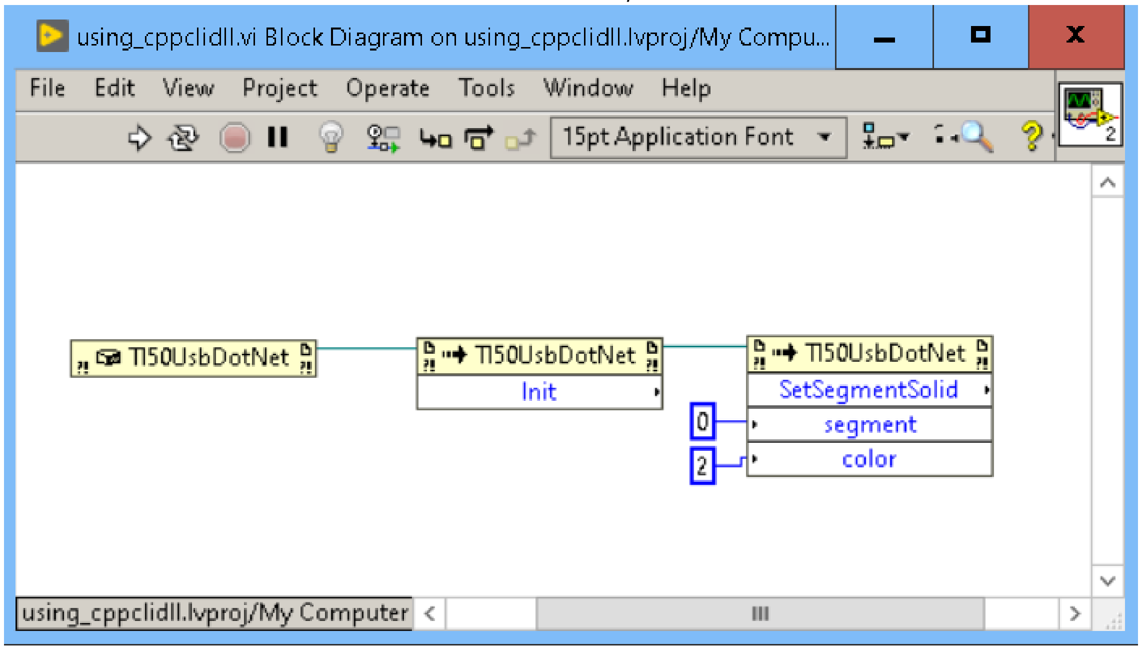
A Visual Studio project is provided at *Examples/Tl50UsbDotNetCoreExample/Tl50UsbDotNetCoreExample.csproj*.

LabVIEW Sample .vi

A basic example that turns on the light. It uses the x86 .NET DLL.

A LabVIEW 2019 project is provided at *Examples/Tl50UsbLabviewExample/using_cppclidll.lvproj*.

LabVIEW Sample



Chapter Contents

Chapter 5 Additional Dependencies

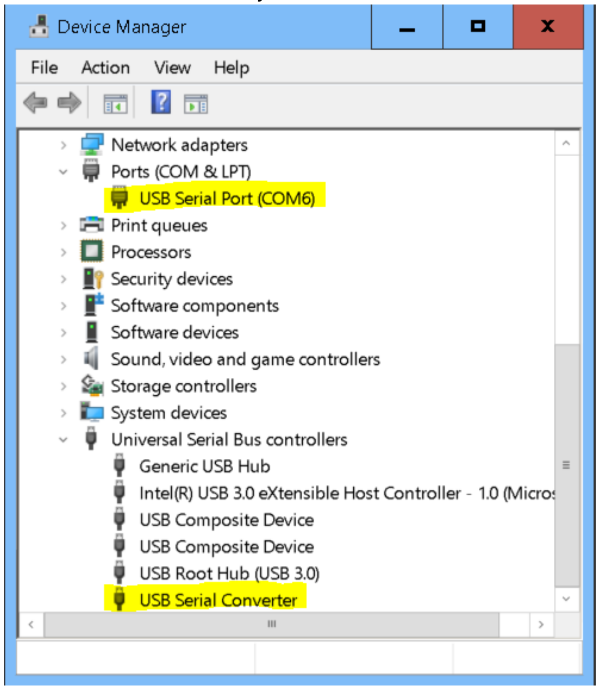
FTDI Drivers

A device driver is a piece of software that tells the computer how to interact with a connected device. The TL50 Pro Tower Light with USB Windows libraries (the non-.NET Standard versions) use a common USB device driver from FTDI.

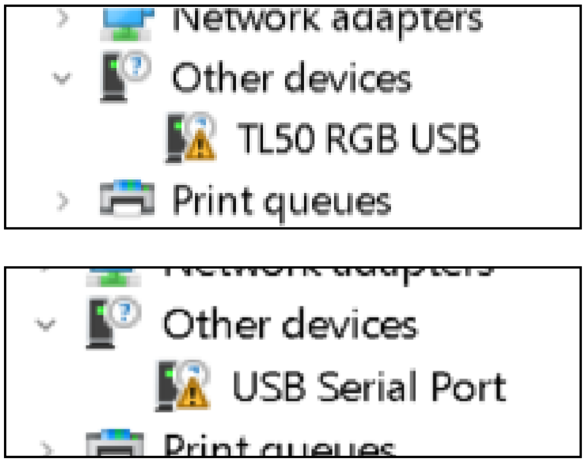
Automatic Installation

When the tower light is first plugged into the PC, Windows automatically detects and retrieves the necessary driver. A properly installed driver appears as a **USB Serial Port** in Windows' **Device Manager**, as shown in "Figure: Correctly Installed Driver" on page 15.

Correctly Installed Driver



Device Driver Not Installed



Explicit Installation

In the event that the driver is not automatically installed, it may appear in Windows' **Device Manager** as shown in "Figure: Device Driver Not Installed" on page 15. This commonly occurs if there is no internet access on the PC. In this scenario, the user needs to explicitly install the drivers.

To complete an explicit install of the device drivers:

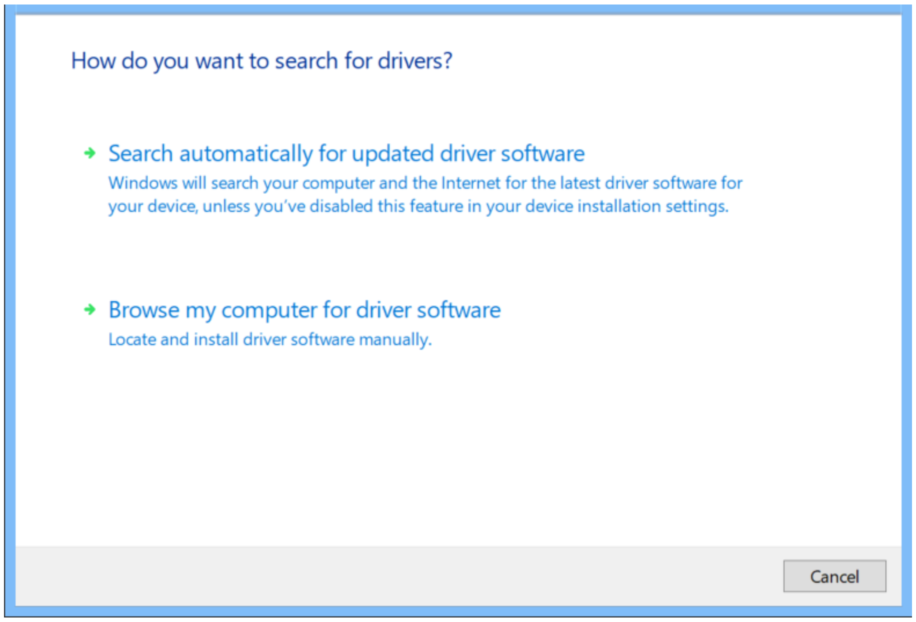
1. Run the included installer titled **FTDI drivers/CDM21228_Setup.exe**.
2. Use the default installer settings.
3. After installing, disconnect and reconnect the tower light to reestablish an accurate connection.

Manual Installation

A third installation option is to use **Device Manager** to manually choose the driver.

1. Right click the entry in **Device Manager** and choose **Update Device Driver** from the context menu.
2. Choose **Browse my computer for driver software**.

Driver Search Selection Window



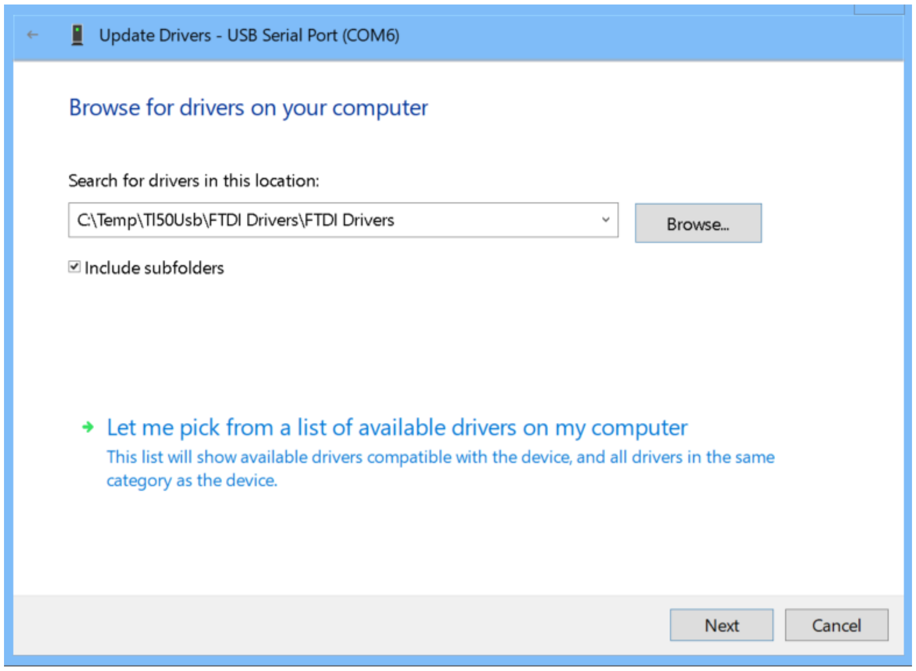
How do you want to search for drivers?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

3. Choose **Browse…** and navigate to the *FTDI Drivers\FTDI Drivers* directory at the location of this software collection.

Driver Browser Window



Update Drivers - USB Serial Port (COM6)

Browse for drivers on your computer

Search for drivers in this location:
C:\Temp\Tl50Usb\FTDI Drivers\FTDI Drivers

☑ Include subfolders

Let me pick from a list of available drivers on my computer
This list will show available drivers compatible with the device, and all drivers in the same category as the device.

4. Click **Next** and choose the driver to install.

Visual C++ Redistributable

The Windows libraries (non-.NET Standard versions) use common software routines provided by Microsoft. These are available from Microsoft (x86, x64), or are included at *Libraries/Windows/VC_redist.x86.exe* and *Libraries/Windows/VC_redist.x64.exe*.

Chapter Contents

Chapter 6 Additional Documentation

For more information about the TL50 Pro Tower Light with USB, please see additional documentation on the Banner website: www.bannerengineering.com.

- P/N 217569, TL50 Pro Tower Light with USB Datasheet: General information about the product
- P/N 205167, TL50 Pro Tower Light with IO-Link Datasheet: Alternate description of operating modes

Chapter Contents

Chapter 7 Appendix A: Serial Protocol

Banner Engineering's TL50 Pro Tower Light with USB can be controlled using a serial communication protocol. Most users opt to use one of the control libraries to simplify their interaction with the device. However, some users may prefer to implement the serial communications directly. This appendix describes the protocol.

Serial Communication Settings

Baud rate	19200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

Advanced Segment Mode

The Advanced Segment Mode controls individual segments.

To configure Advanced Segment Mode:

1. Send the Enable Advanced Segment Mode command
2. Send Change Advanced Segment Indication commands as needed

Enable Advanced Segment Mode Command

This command changes the device to Advanced Segment Mode and is persisted.

Bytes

Index	Value
1	0xF4
2	0x41
3	0xC7
4	0x01
5	0x00
6	0x01
7	0x01
8	0xFE

Example Byte String

F441C701000101FE

Expected Response

F441C7010006FCFD

Change Advanced Segment Indication Command

This command changes the appearance of the indication and is not persisted.

Bytes

Index	Bits	Value
1		0xF4
2		0x41
3		0xC1
4		0x1F
5		0x00
6	1-4	<color code 1>
	5-7	<intensity code 1>
	8	0x00
7	1-3	<animation code>
	4-5	<speed code>
	6-8	<pattern code>
8	1-4	<color code 2>
	5-7	<intensity code 2>
	8	<rotation direction code>
9-35		0x00
36		<audible code>
37		<checksum byte 1>
38		<checksum byte 2>

Color Codes

Color	Code	Color	Code	Color	Code
Green	0	Lime Greem	5	Violet	10
Red	1	Spring Green	6	Magenta	11
Orange	2	Cyan	7	Rose	12
Amber	3	Sky Blue	8	White	13
Yellow	4	Blue	9		

Animation codes set the style of indication

Animation style	Value	Animation style	Value
Off	0	Half/half	4
On	1	Half/half rotate	5
Flashing	2	Chasing	6
Two color flashing	3	Intensity sweep	7

Example Byte Strings

Red:

```
F441C11F0001010000000000000000000000000000000000000000000000000000000000E8FD
```

Yellow:

F441C11F00040100000000000000000000000000000000000000000000000000000000E5FD

Green:

F441C11F00000100000000000000000000000000000000000000000000000000000000E9FD

Expected Response

F441C101000602FE

Level Mode

The Level Mode works primarily with multi-segmented tower lights. It has limited use on single segment tower lights.

The indication varies with an input signal value. The tower can be split into up to three regions with their own indication behavior. Regions are defined by threshold values.

To configure Level Mode:

1. Define the style settings by sending the Level Mode Parameters command
2. Send the Enable Level Mode command
3. Send Change Level Indication commands to update the level with the latest signal input value

Level Mode Parameters Command

This command determines the style of indication and is persisted.

Bytes

Index	Value
Byte 1	0xF4
Byte 2	0x41
Byte 3	0xBA
Byte 4	0x16
Byte 5	0x00
Byte 6-7	<full scale value>
Byte 8	<color code, normal threshold region>
Byte 9	<intensity code, normal threshold region>
Byte 10	<state code, normal threshold region>
Byte 11	<level mode thresholds code>
Byte 12-13	<threshold value, low threshold region>
Byte 14	<color code, low threshold region>
Byte 15	<intensity code, low threshold region>
Byte 16	<state code, low threshold region>
Byte 17-18	<threshold value, high threshold region>
Byte 19	<color code, high threshold region>
Byte 20	<intensity code, high threshold region>
Byte 21	<state code, high threshold region>
Byte 22	<flashing speed code>
Byte 23	<dominance code>
Byte 24	<subsegment style code>
Byte 25	<color code, background>
Byte 26	<intensity code, background>

Continued on page 22

Continued from page 21

Index	Value
Byte 27	<upside down enabled>
Byte 28	<checksum byte 1>
Byte 29	<checksum byte 2>

Full-Scale Value

The signal input value that represents the maximum allowed. It is associated with the level of the full tower.

0-65535

Color Codes

Color Codes

Color	Code	Color	Code	Color	Codc
Green	0x00	Lime Green	0x05	Violet	0x0A
Red	0x01	Spring Green	0x06	Magenta	0x0B
Orange	0x02	Cyan	0x07	Rose	0x0C
Amber	0x03	Sky Blue	0x08	White	0x0D
Yellow	0x04	Blue	0x09		

Intensity Codes

Sets the brightness of the associated colors.

- High = 0x00
- Low = 0x01
- Medium = 0x02
- Off = 0x03

State Codes

Sets the animation used in the threshold region.

- Steady = 0x00
- Flashing = 0x01

Level Mode Threshold Codes

Configures which threshold regions are enabled. Normal is always enabled, but this can enable additional regions.

- None = 0x00
- Low = 0x01
- High = 0x02
- High and Low = 0x03

Threshold Value

A value used to separate threshold regions. For Low Threshold, this is the upper value of the low region. For High Threshold, this is the upper value of the normal region.

0-65535

Flashing Speed Code

Sets how fast the animation flashes, if a flashing state is configured.

- Standard = 0x00
- Fast = 0x01
- Slow = 0x02

Dominance Code

Sets a threshold's region indication settings only for the activated segments of the region (Disabled), or sets them for all active segments (Enabled).

NOTE: For single segment devices, the Dominance Code should be set to Enabled.

- Disabled = 0x00
- Enabled = 0x01

Subsegment Style Code

Specifies how a segment is indicated, if the input signal value is in the value range of a segment.

Steady = 0x00

Provides solid uniform indication once the signal reaches the range

Flashing = 0x01

Causes the indication to blink while the signal is in the segment's range, which becomes steady once the value moves to the next segment

Analog = 0x02

Specifies that the brightness is proportional to how close the signal input value is to the full-scale value

Upside Down Mode

Sets the value to start from the base of the device and proceed toward the top (Disabled), or sets the value to start at the top and proceed toward the base (Enabled).

- Disabled = 0x00
- Enabled = 0x01

Example Byte String

Full scale value of 100, high and low thresholds on, low of 20, high of 80:

F441BA160064000000000314000000005000000000000020003002AFD

Expected Response

F441BA01000609FE

Enable Level Mode Command

This command activates the Level Mode indication and is persisted.

Bytes

Index	Value
Byte 1	0xF4
Byte 2	0x41
Byte 3	0xC7
Byte 4	0x01
Byte 5	0x00
Byte 6	0x03
Byte 7	0xFF
Byte 8	0xFD

Example Byte String

F441C7010003FFFD

Expected Response

F441C7010006FCFD

Change Level Indication Command

This command changes the appearance of the indication and is not persisted.

Bytes

Index	Value
Byte 1	0xF4
Byte 2	0x41
Byte 3	0xC1
Byte 4	0x1F
Byte 5-32	0x00
Byte 33	<audible code>
Byte 34	0x00
Byte 35-36	<signal value>
Byte 37	<checksum byte 1>
Byte 38	<checksum byte 2>

Audible Code. Controls the audible segment, if present.
- Off = 0
- Steady = 1
- Pulsed = 2
- SOS = 3

Signal Value. Used to control the level.

0-65535

Example Byte String. Signal value of 50:

F441C11F0000000000000000000000000000000000000000000000000000000000003200B8FD

Expected Response. F441C101000602FE

Run Mode

The Run Mode controls all segments. It is used primarily for multi-segmented tower lights, but it can be used in certain situations for single-segment tower lights.

To configure Run Mode:
1. Send the Enable Run Mode command
2. Send the Change Run Indication commands as needed

Enable Run Mode Command

This command activates Run Mode and is persisted.

Bytes

Index	Value
Byte 1	0xF4
Byte 2	0x41
Byte 3	0xC7
Byte 4	0x01
Byte 5	0x00
Byte 6	0x02

Continued on page 25

Continued from page 24

Index	Value
Byte 7	0x00
Byte 8	0xFE

Example Byte String
F441C701000200FE

Expected Response
F441C7010006FCFD

Change Run Indication Command
This command changes the appearance of the indication and is not persisted.

Bytes

Index	Value
Byte 1	0xF4
Byte 2	0x41
Byte 3	0xC1
Byte 4	0x1F
Byte 5-26	0x00
Byte 27	<animation code>
Byte 28	<color code>
Byte 29	<intensity code>
Byte 30	<speed code>
Byte 31	<pattern code>
Byte 32	<color code 2>
Byte 33	<intensity code 2>
Byte 34	<shift code>
Byte 35	<rotation direction code>
Byte 36	<audible state>
Byte 37	<checksum byte 1>
Byte 38	<checksum byte 2>

Animation Codes
Sets the style of indication.

Animation codes

Animation	Code	Animation	Code	Animation	Code
Off	0x00	Half/half rotate	0x05	Rainbow	0x0A
On	0x01	Chase	0x06	Demo	0x0B
Flashing	0x02	Intensity sweep	0x07		
Two-color flashing	0x03	Scroll	0x08		
Half/half	0x04	Bounce	0x09		

Color Codes

Color Codes

Color	Code	Color	Code	Color	Codc
Green	0x00	Lime Green	0x05	Violet	0x0A
Red	0x01	Spring Green	0x06	Magenta	0x0B
Orange	0x02	Cyan	0x07	Rose	0x0C
Amber	0x03	Sky Blue	0x08	White	0x0D
Yellow	0x04	Blue	0x09		

Intensity Codes
Sets the brightness of the associated color.

- High = 0x00
- Low = 0x01
- Medium = 0x02
- Off = 0x03

Speed Codes
Sets how fast the animation progresses.

- Standard = 0x00
- Fast = 0x01
- Slow = 0x02

Pattern Codes
Sets the flashing pattern for Flash and Two-Color Flash.

- Normal = 0x00
- Strobe = 0x01
- Three Pulse = 0x02
- SOS = 0x03
- Random = 0x04

Shift Codes
Offsets each segment's animation from the previous segment.

- Disabled = 0x00
- Enabled = 0x01

Rotation Codes
Sets the direction of Half/Half Rotate and Chase.

- Counterclockwise = 0x00
- Clockwise = 0x01

Audible Codes
Controls the audible segment, if used.

- Off = 0
- Steady = 1
- Pulsed = 2
- SOS = 3

Example Byte String
Turn segments on steady green:

F441C11F0000000000000000000000000000000000000000000000001000000000000000000E9FD

Expected Response

F441C101000602FE

LinkedIn

X (formerly Twitter)

Facebook