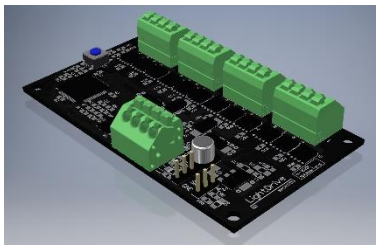
MACH ENGINEERING LLC
mach-engineering.com



LightDrive 12 – LED Controller *User Guide*

Thank you for purchasing the LightDrive 12 LED controller. The LightDrive is a very powerful controller built onto a tiny footprint. This guide is meant to give you a good idea of the LightDrive's capabilities and limitations. After that, it's up to you.

The LightDrive can be controlled several different ways. The user may choose between CANbus, TTL Serial, and R/C PWM inputs. These methods offer different advantages:

	R/C PWM	***CAN***	***TTL Serial***
Color Depth	'3 bit' – 8 Colors	Full 24bit – 8bit x 12ch	Full 24bit – 8bit x 12ch
Update Rate	0.8Hz – 1ch/100ms	10Hz – 12ch/100ms	10Hz – 12ch/100ms
Physical	2 R/C PWM Inputs	2-Wire CAN Input	TTL Serial Input
Signal Format	0.5-2.5ms width 50/125/250Hz	1Mbit CAN	115.2kBaud 8N1 Serial

For detailed information on these protocols, see Appendix A. The LightDrive will switch automatically to the proper protocol based on which signal it sees. If more than one is present, the LightDrive will choose a protocol in the following order: Serial -> CAN -> PWM. Example code, including an API, is available for the FRC RoboRIO controller in C++, Java, and LabView. Simply use the appropriate API for whichever control method you want to use (PWM, CAN, Serial).

Connection Diagram

CAN Port
- Connect H and L
- Connect anywhere in the chain with other devices

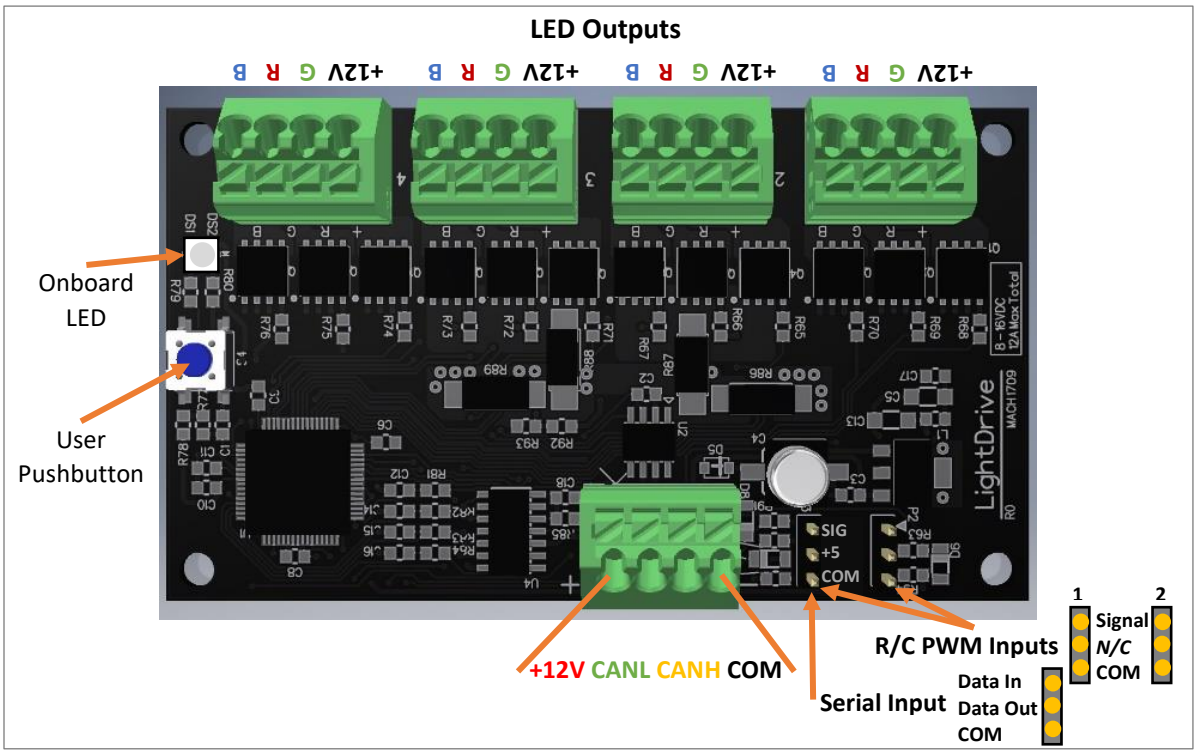
Serial Port
- Connect GND and TXD
- RXD Optional if feedback is not needed

R/C PWM Outputs
- Connect to Any 2

Choose One of These 3 Methods

Basic Operation

1. **Connect the LightDrive** as shown in the diagram.



2. **Apply Power**
   a. The LightDrive will display a test-pattern, then become ready after 1 second.
   b. If no signal is present, the LightDrive will enable all outputs at 100% *OR* display the last pattern that was active when the User Button was last pressed.
3. **Verify LED** Lights according to table below:

Color	***Meaning***
Yellow Flash	Idle – No signals detected. Displaying Last Output.
Yellow Steady	Standby – Press Button to Turn Outputs On/Off
Green Flash	Running on **PWM** Input.
Blue Flash	Running on **Serial** Input.
Purple Flash	Running on **CAN** Input.
Red Flash	Running at Limit. At least 1 channel >5A.
Red Solid	Tripped. Overload/Short on at least one channel. Auto-reset in 3 seconds.

4. **Begin Controlling** outputs using your chosen input type

Specifications and Notes

Specifications			
Size	2.0x3.5in (50.8x88.9mm)	Refresh Rate	100Hz for all channels
Input Voltage	8 – 16VDC	Power Connector	16-24AWG Screwless
Max Current	Lesser of 15A or 5A/output	RC Connector	3pos 0.1in Male HDR (**GND-6V*-SIG**) **6V Connection optional*
PWM Inputs	20-250Hz 0.5-2.5ms		
CAN Input	1Mbit/s		

Fault Tolerance		
Fault From/To	***Fault From/To***	***Duration***
12V Batt Pos	PWM Input	Indefinitely
	PWM GND	Intermittently
	CANL/H	Indefinitely
	TTL Serial Input	Indefinitely
12V Batt Neg	PWM Input	Indefinitely
	PWM GND	Indefinitely
	CAN L/H	Indefinitely
	TTL Serial Input	Indefinitely
Reverse Polarity		Intermittently
Shorted Output		Indefinitely
12V Batt Neg	12V LED Output	Fuse Blown

Note: The LightDrive is designed to be as resilient as possible against short-circuits and mis-wiring. However, some conditions, such as reverse polarity, may allow current to flow to the connected loads. This may damage them, even though the LightDrive survives.

Programming

An API is provided for the FRC WPILib. JavaDocs (also applicable to C++) can be found here. To use this library with VS Code follow the steps below:

1. From the WPI Control Palette, Type and Select "Manage Vendor Libraries"
2. Click "Install New Library (online)"
3. Paste the following URL:
   http://mach-engineering.com/products/maven/com/mach/LightDrive/LightDrive.json

C++/Java Programming

Once the above has been completed, add the needed import statements to your program:

```
//Java
import com.mach.LightDrive.*; //The LightDrive Library
import java.awt.Color;  //Predefined colors and routines

//C++
#include "LightDrive.h"
```

Now, create your desired LightDrive object:

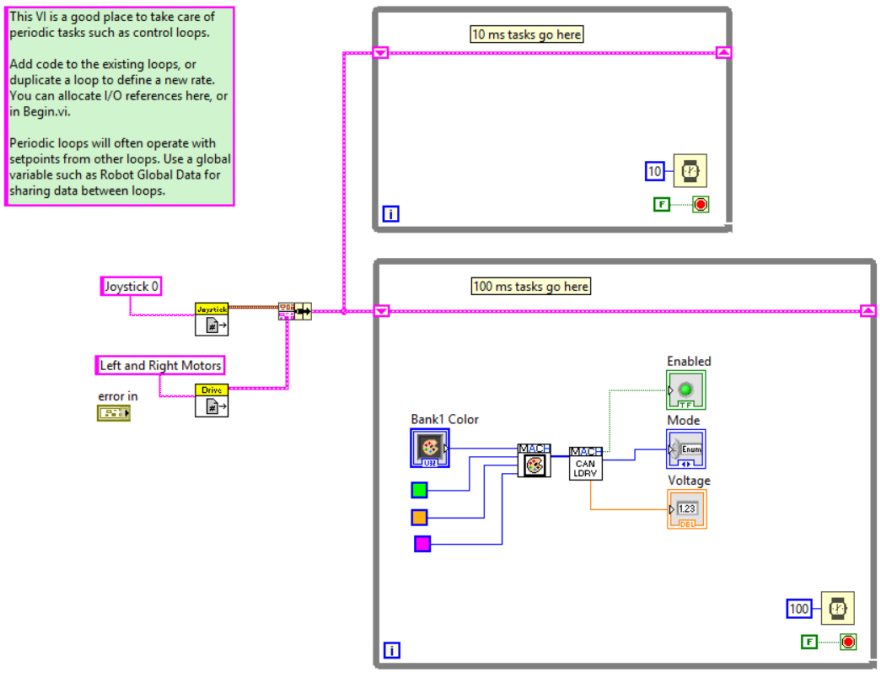
```
//C++/JAVA
//CAN-controlled LightDrive
ld_can = new LightDriveCAN();
//MXP Serial Port Controller LightDrive
ld_serial = new LightDriveSerial();
//PWM Servo Controlled LightDrive (specify 2 WPI Servo objects)
ld_pwm = new LightDrivePWM(Servo0,Servo1);
```

Ready to control LEDs!

```
//JAVA
//ld_object: ld_can, ld_serial, or ld_pwm.
//Set Bank 1 to Red, Bank 2 to yellow, Bank 3 to custom.
ld_object.SetColor(1, Color.blue);
ld_object.SetColor(2, Color.yellow);
ld_object.SetColor(3, new Color(1.0f,0.5f,0.0f));

//Send latest colors to LightDrive
ld_object.Update();


//C++
ld_object->SetColor(1, LIGHTDRIVE::COLORS::BLUE);
ld_object->SetColor(2, LIGHTDRIVE::COLORS::YELLOW);
ld_object->SetColor(3, ld_object->MakeColor(255,128,0));

ld_object->Update();
```

LabView Programming

Using the LabView VIs from mach-engineering.com, create a diagram as below. Each time the surrounding frame is called, the values will be pushed to the LightDrive.



Appendix A – Protocols

TTL Serial Protocol

Using the serial protocol, all 12 channels can be set to any level from 0-255. If desired, the RX line can be connected to get feedback from the LightDrive. The serial format is 8N1 (8-bit, no parity, 1 stop bit) at 115.2kBaud. The serial packets are constructed as below:

Start Byte	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10	CH11	CH12	Checksum
0xAA	0	0	0	0	0	0	0	0	0	0	0	0	8-bit sum*

*8bit sum of all bytes from 0xAA – CH12.

Example Packet:

0xAA 0xFF 0x00 0x00 0x00 0xFF 0x00 0x00 0x00 0xFF 0xC8 0xFF 0x00 0x6E

The feedback packet is structured as shown below. This packet is sent in response to the transmitted packet detailed above.

<table>
<tr><th>I1</th><th>I2</th><th>I3</th><th>I4</th><th>VIN</th><th>Status</th><th>Not Used</th><th>FW</th></tr>
<tr><td colspan="4">These 4 bytes represent the current on each bank in A*10 (ie 4.5A = 45)</td><td>Voltage input in V*10 (ie 12.6V = 126)</td><td><table><tr><th>Bit</th><th>Desc.</th></tr><tr><td>0</td><td>Enabled?</td></tr><tr><td>1-3</td><td>Mode</td></tr><tr><td>4-7</td><td>Trip Flags</td></tr></table></td><td></td><td>Firmware Version</td></tr>
</table>

CAN Protocol

Using the CANbus protocol, all 12 channels can be set to any level from 0-255. The CAN protocol used is designed to be compatible with the WPILib protocol used by the FRC RoboRIO. Two messages are used to control the outputs. Another message is sent by the LightDrive and provides feedback about the device.

PWM Protocol

The LightDrive is designed to use the same type of PWM signals as common hobby remote control servos and speed controllers. It has also been pre-calibrated to the signals generated by the RoboRIO. Each of the two input channels controls 6 of the 12 total channels. PWM1 controls channels 1-6 and PWM2 controls 7-12.