BriteBlox

Modular LED Marquee and Development System

Hardware and Protocol Guide

Version Draft, 9/16/2015

Table of Contents

Version Draft, 9/16/2015

BriteBlox Firmware 3.4

BriteBlox Displayboards as of the present time come shipped with firmware version 3.4. This is the fourth iteration of the firmware designed for Board Revision 3. The first two board revisions were made in very small prototyping runs and are last known to be in the hands of two elite Kickstarter backers.

Displayboard Hardware Design

The heart of a Displayboard is two Atmel ATmega168 microcontrollers. All digital I/O pins for both chips are exposed by headers on the board. However, as some pins are designated to be used as current sinks for the cathodes on the LED panels, and there are less cathode pins than anode pins, these pins have resistors in between the direct output from the chips. You will notice there are a total of 10 resistors on the board since there are 5 cathodes for red LEDs and 5 cathodes for green LEDs. This is defined by the design of the LED panels.

Here are the output pins that flow straight to the headers versus being impeded by resistors. These pins are the same whether considering the red or green chip.

No Resistor Before Header Pin	Resistor Present
PD0 (D0, RXD)	PD2 (D2)
PD1 (D1, TXD)	PD5 (D5)
PD3 (D3)	PD7 (D7)
PD4 (D4)	PB0 (D8)
PD6 (D6)	PB4 (D12)
PB1 (D9)	
PB2 (D10)	
PB3 (D11)	
PB5 (D13)	

Table 1: Pins on each chip that do or don't go through resistors.

In order to facilitate production, the resistors added by the factory are size 0805 surface-mount resistors of 33.2Ω value. There are also holes to allow tinkerers to add through-hole components as desired. Tinkerers are encouraged to configure the resistors to facilitate their application in any way that does not cause damage to the Displayboard. One good modification is to simply replace the resistor with a short or jumper if no resistance to/from the header pin is desired.

Due to limited space on the board, there are only two analog pins per chip exposed through our headers. The analog pins on the chip are routed to the headers where the LED panel has redundant pins. Since the LED panel has two places where one could provide it Row 4 [Red|Green] and Column 3 [Red|Green], analog inputs have been routed to the redundant locations instead.

The SMD edition of the ATmega168 also features an extra analog input pin compared to the DIP-socket version; this extra pin is used for the auto-addressing scheme, and the signal passing through this pin on all chips is called RIN. The RIN line, which is read by the

firmware to calculate the auto-addressing value, is supposed to be high (5V) on the left (or top left) side of the marquee, and low (0V or GND) on the right (or bottom right) side of the marquee. After passing through the right (red) chip, the RIN line goes through a 10 KΩ resistor before going to the output pin. This effectuates an even voltage drop between all boards in a BriteBlox marquee, and explains why the firmware (ideally) reads board addresses in the following manner:

Length	Addresses
2	0, 32
3	0, 21, 42
4	0, 16, 32, 48
Etc.	

Table 2: Values calculated by auto-addressing for various short marquees.

Of course, electronics rarely work as the mathematics dictates. As such, it is often observed that the board addresses will "sag" or "bulge" depending on how 5V and GND is wired through the RIN line. An example of this is when board addresses read {0, 15, 31, 47} rather than {0, 16, 32, 48}; in some cases, one color of one board might read lower than the other color on the same board. The software is set to make corrections to boards that are ±1 away from their expected value shortly before the text or animation begins to display. However, it is recommended to only use auto-addressing with a maximum of 12 boards in order to avoid situations where boards diverge more than ±1 away from their expected value.

Marquee Data Protocol

The proprietary BriteBlox protocol, utilizing the RS-232 serial protocol at TTL (5V/0V) logic levels, is designed to take advantage of the physical characteristics of the LED panels. The LED panels happen to have 7 rows and 5 columns. By propagating information about one column at a time in a single byte, the use of bandwidth is maximized without needing to split bytes into tinier pieces and possibly conjoin them using complex algorithms.

There are two types of bytes sent by the BriteBlox protocol: address bytes and data bytes. Despite both containing 8 bits, they contain very different data:

Each chip expects to receive one address bit and then five data bits. If the address propagated matches the chip's programmed or discovered address, it sets the internal variable `activeProc` to `true`. `activeProc` remains `true` until five more data bytes (the data bytes) have been read. In the unlikely event that an address byte or command is received before five data bytes are received, the board will process the address byte or command and will reset `activeProc` to `false`.

Bit	Purpose
8 (MSB)	1 = Address, 0 = Data
7	Address: 1 = Green, 0 = Red Data: Top (7^{th}) row
6	Address: MSB address bit (+32) Data: 6^{th} row
5	Address: 5th address bit (+16) Data: 5^{th} row
4	Address: 4th address bit (+8) Data: 4^{th} row
3	Address: 3rd address bit (+4) Data: 3^{rd} row
2	Address: 2nd address bit (+2) Data: 2^{nd} row
1 (LSB)	Address: LSB address bit (+1) Data: 1^{st} (bottom) row

Table 3: Encoding of the protocol.

Special Firmware Operations

Address bytes can also act like command bytes. The way to enter Command Mode is to satisfy a finite-state machine due to ground noise sometimes making communications unreliable, especially on long or multi-row marquees. Specifically, the finite-state machine requires that the following bytes are received over serial input in order before switching to Command Mode: {0xD0, 0xE1, 0xF3, 0xF3, 0xD7, 0xEF, 0xF2, 0xE4, 0xFF}. This sequence is the ASCII code for "PassWord" + 0x80 on each character, followed by 0xFF in order to maintain backward compatibility with firmware versions prior to 3.4. Once Command Mode is entered, the next byte expected is a number expressing a particular command.

Control Codes

All numbers in hexadecimal:

Control code ID	Argument(s)	Description
80	[00-39]	Display both red & green test patterns for given chip
80	[40-79]	Display test pattern on all chips
80	[80-BF]	Display only the prescribed red chip ID's test pattern
80	[C0-FF]	Display only the prescribed green chip ID's test pattern
81	[any value]	Reset chip ID for all chips on all boards
82	[80-FE]	Increment chip ID for specified chip
83	[81-FF]	Decrement chip ID for specified chip

84	[80-FF] [80-FF]	Set chip ID for <argument 1, old address> to <argument 2, new address>
85	[80-FF]	Program internal EEPROM to contain the current Chip ID reading
86	[80-FF]	Clear the current address (write 0's)
87-8E	N/A	reserved
8F	[any value]	Clear all EEPROM on all chips • Cleared address • Cleared baud rate
90	none	Reset serial to 9600 baud
91	none	Reset serial to 14400 baud
92	none	Reset serial to 19200 baud
93	none	Reset serial to 28800 baud
94	none	Reset serial to 38400 baud
95	none	Reset serial to 57600 baud
96	none	Reset serial to 76800 baud
97	none	Reset serial to 115200 baud
98	none	Reset serial to 200k baud
99	none	Reset serial to 250k baud
9A	none	Reset serial to 500k baud
9B	none	Reset serial to 1M baud
9C	none	Reset serial to 9600 baud
9D	none	Reset serial to 9600 baud
9E	none	Program internal EEPROM to contain the current baud rate
9F	none	Clear the current baud rate from the EEPROM (write 0's)
A0	[80-FF]	Broadcast the firmware revision number from given chip; otherwise temporarily disable serial

Additional Support

Questions? Comments? Email us at sales@ledgoes.com. Find us on Twitter at @LEDgoes_display, or on Facebook at https://www.facebook.com/LEDgoes.display.