Adafruit's 16x32 LED Matrix Guide for BeagleBone Green
Written by: Scott Plummer

Introduction

This guide will focus on how to wire up the 16x32 LED matrix sold by Adafruit, create a basic kernel driver to handle driving the matrix, and provide data collected about the boards flickering issue. The first part of this guide is a summary of the set up information in the guide "Adafruit's 16x32 LED Matrix Guide for BeagleBone" written by Janet Mardjuki with some additional information to reflect the BBG board.

Wiring

Wiring can be done on either the zen cape or the BBG itself. The zen cape provides pass through GPIO pins and as such if we use pins that are not used by other components the matrix will work without needing to disable anything.

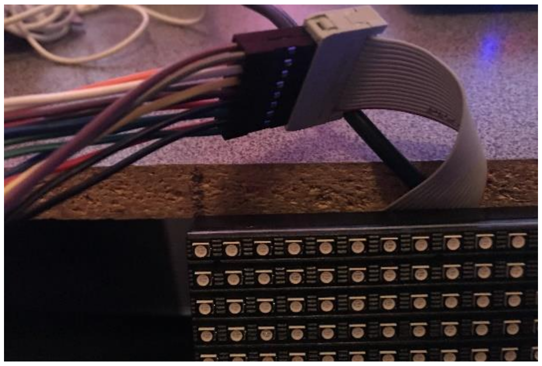
Figure 1

The ribbon wires and the LED matrix need to be wired up first. This link tells us that the ends on the ribbon cables are flipped. Using this information and this table from the "Adafruit's 16x32 LED Matrix Guide for BeagleBone" guide will allow us to properly wire up the ribbon cable with the matrix.

G1 (Green)	R1 (Red)
GND	B1 (Blue)
G2	R2
GND	B2
B	A
D	C
LAT	CLK
GND	OE

The D wire and potentially the OE wire are going to be unused, but still need to be wired for the board to function.

We'll be using pins 35-46, however, any non GPIO mapped pins can be used. You can find the zen cape GPIO diagram here, any of the unlabelled pins are usable. Below is a table from the "Adafruit's 16x32 LED Matrix Guide for BeagleBone" guide for the mapping the GPIO pins to the wires from the ribbon cable. It is important to note that both the D and OE wires are missing from this table as they should be run to GND pins.

Top row of pins on p8 header (even pin numbers from 36-44)	Bottom row of pins on p8 header (odd pin numbers from 35-45)
G1 [80]	R1 [8]
G2 [79]	B1 [78]
B [77]	R2 [76]
LATCH [75]	B2 [74]
CLK [73]	A [72]
	C [70]

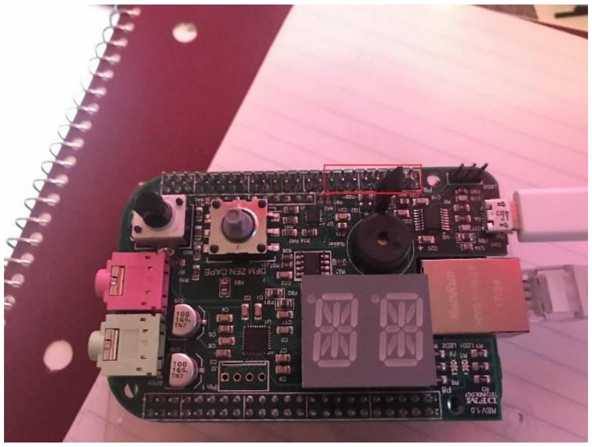
Figure 2

Highlighted in red is the P8 header pins which will be used to connect the wires from the ribbon cable to the board. They should be wired in the order of the table starting from the left most pin in the picture.

While wiring up the board there are 5 pins we want to ground: the three GND pins, the OE pin, and the D pin (only used for 64x32 matrices). The OE pin must be grounded or used in combination with a level shifter as the pin drives 7V to the board. Depending on which pin is used for the OE wire, the board may not enter the correct boot sequence, or the voltage regulator may be damaged. In the end the board with all the wires could look something like this

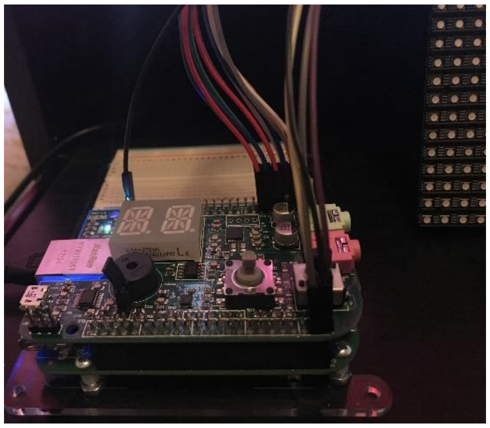
Figure 3

For a more in depth understanding of how to test and set up on the matrix please see this guide.

Flickering and 'Pixel' Bleed

One of the major issues with this LED matrix is the fact that the LEDs flicker when using a delay that is to long (4.5ms+) when trying to loop through and drive values to the registers on the board through GPIO. I'll be discussing what we've found while trying to solve this issue.

In general, there are two problems that we've encountered with regards to getting a nice-looking image displayed on the matrix. The first is the previously mentioned flickering, where when providing a delay in the ledMatrix_refresh function that is too long causes the refresh rate of the matrix to be visibly noticeable. The second issue is that if we reduce the delay (delay < 4ms) in the ledMatrix_refresh function it causes the LEDs to bleed over into unexpected positions. In this case trying to create a bar like image on the matrix caused multiple pixel to be activated incorrectly.

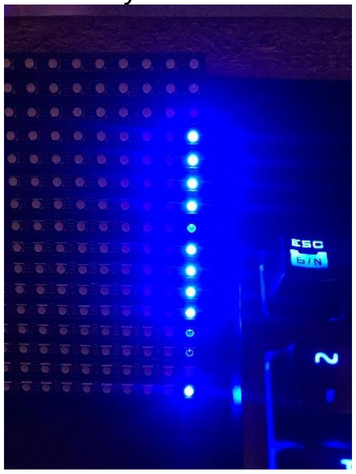
Figure 4

Here is a table of the data we collected through testing. Hopefully it will provided insight into how to improve or even remove the flickering/bleed-over problem in the future.

Table 1

Test Description	Test Data	Expected Output	Actual Output
Enable all LED 'pixels' in column 0 with colour red	32x16 array with all values in the first position of the arrays set to 1	All pixels in column 0 are enabled without bleed over into other columns/rows	All pixels in column 0 are enabled without bleed over into other columns/rows
Enable all LED 'pixels' in column 0 and rows 0-7 with colour red	32x16 array with all values in the first 8 position of the arrays set to 1	All pixels in column 0 and rows 0-7 are coloured red	All pixels in column 0 and rows 0-7 are coloured red
Enable 3 pixels in column 0 rows [2-4]	32x16 array with all values in rows [2-3] and column 0 set to 1	All 3 pixels in column 0 rows [2-3] are coloured red with no bleed over	Pixels bleed over into position (0,0), (0, 1), no bleed over in pixels below row 4

From this we can conclude a few interesting pieces of information that may be useful to future groups. The first is that LEDs only bleed over in their respective column, we don't ever see bleed over if all of LEDs are turned on in a column. The second is that LEDs do not bleed over the middle point of the matrix. While we were unable to resolve the problem, hopefully this data can be of use to future teams.

Kernel Driver

Attached with this guide is sample code for a simple misc kernel driver of the code provided from and ported the from the "Adafruit's 16x32 LED Matrix Guide for BeagleBone" guide into a kernel driver. The main difference between the sample code provided from their guide and ours, is that we implement a basic write function for interacting with the matrix code that can be passed a 2d array of data and copy it over into a buffer in kernel space. Along with this we are using the kernel GPIO interface, where more information can be found from this guide on how to program the LEDs using the kernel GPIO interface.

Our attempt at using a misc kernel driver and implementing the write function to interface between our application and the kernel proved to be slower than using just a user space program. This was because of the amount of data we needed to copy over from our user application to the driver. A more performant option would have been to implement the mmap function and use that to pass our data between the kernel and our application.

Troubleshooting

1. If the board isn't booting and the red LED on zen cape is on you've likely forgot to plug in the LED matrix to its power supply. Unplug the BBG from power, plug in the LED matrix into its power supply and turn on the BBG again.
2. If the board turns on but there's no LED activity on the BBG and no information being sent back from screen you've likely connected the OE pin to one of the GPIO pins without using a level shift, or instead of grounding the pin. Power off the board and either

attach a level shifter to the OE pin or place the OE pin on one of the GRND pins labeled above.

3. If the data size received by the driver code doesn’t match that of user space code you’ve likely used fwrite instead of write. Fwrite can cause padding of the data written to the driver. In this case, this shouldn’t cause any issues with the driver.
4. If the driver code is still too slow after implementing mmap you can try to run the matrix through the PRU (not covered in this guide)
5. For any issues with LED ‘pixels’ lighting up, or having in correct placement see this guide