

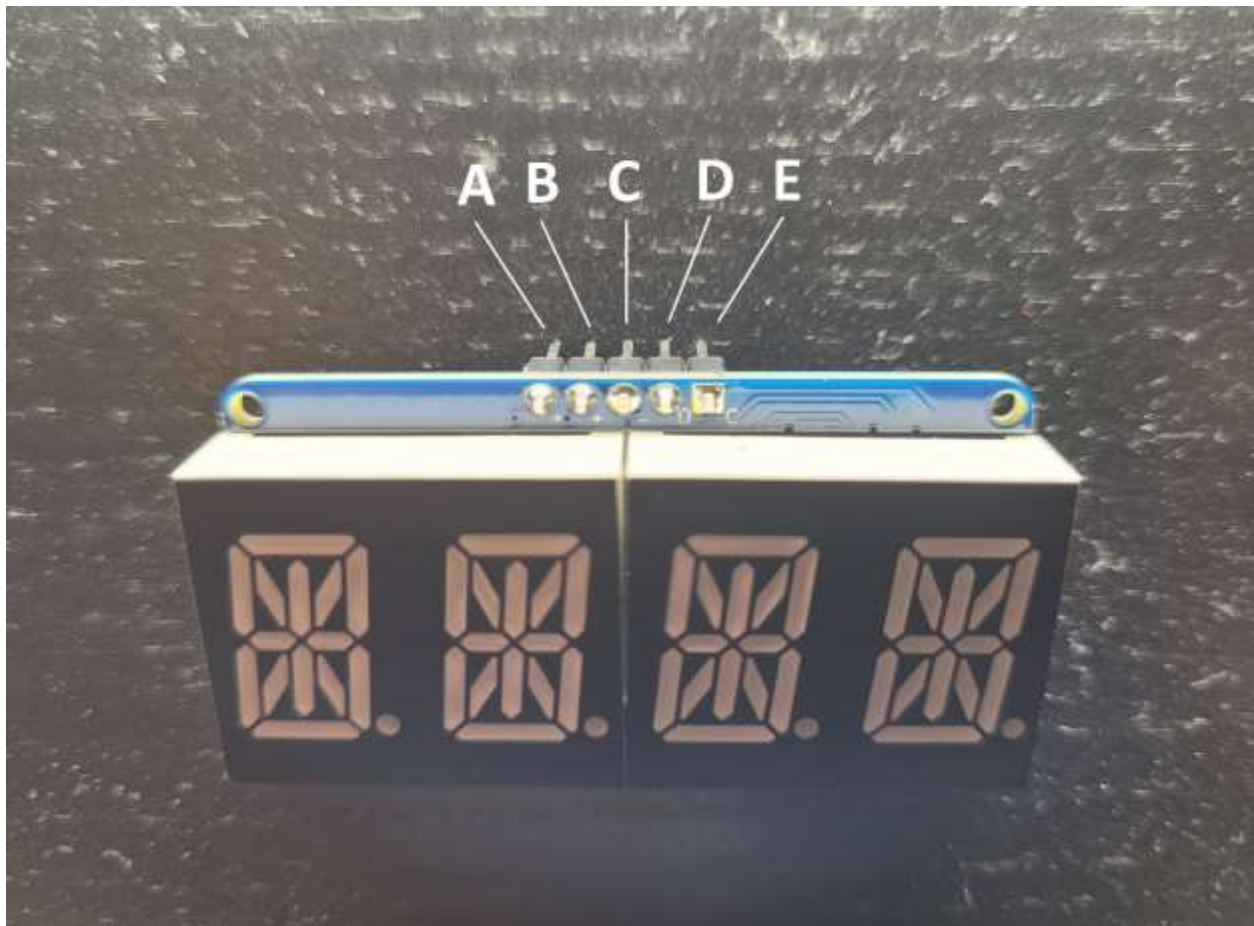
How - to Guide: Alphanumeric Display

By: Sami Ben Amara (Used commands from I²C guide)

Welcome to the How – to guide for the 0.54” Quad Alphanumeric display w/I²C backpack. You can find more information about the display at: <https://www.adafruit.com/product/1912>, and you can find more instructions on how to use it at: <https://learn.adafruit.com/adafruit-led-backpack/0-54-alphanumeric>.

Overview

This display has five pins that are used to connect to the Beaglebone, as shown in the diagram below, marked A – E.



The legend for the above diagram is as follows:

- **A: V²C Pin**
- **B: VCC Pin**
- **C: GND Pin**
- **D: SDA Pin**

How - to Guide: Alphanumeric Display

- **E: SCL Pin**

Wiring

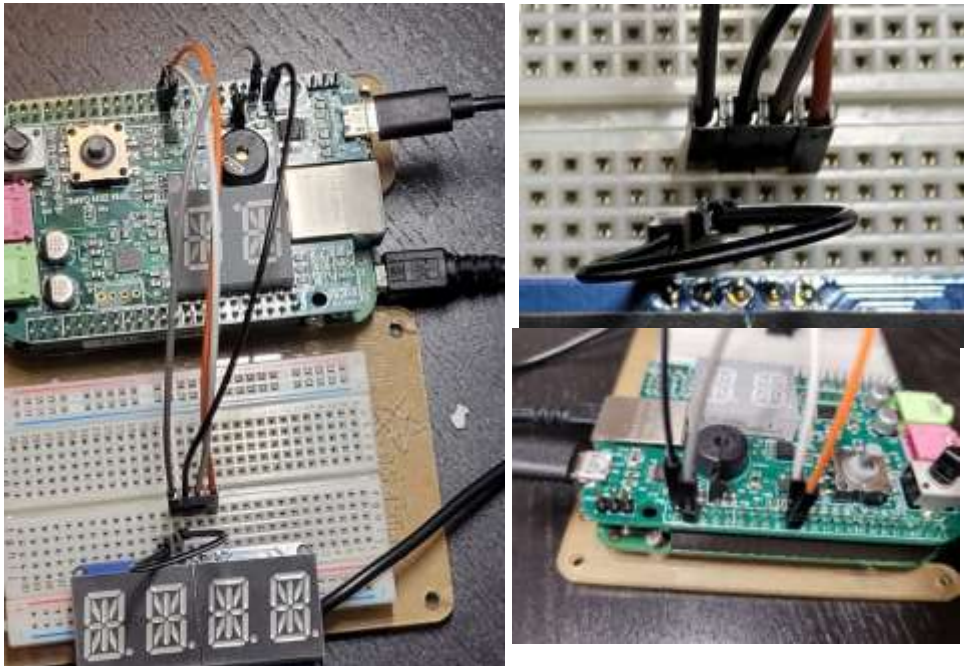
Rules for wiring any component:

- Make sure your board is powered off before placing, removing, or moving components around on the board
- Have someone double check your setup if you are unsure if it's correct

Connect the display to the Zen cape as follows:

- Alphanumeric SCL Pin (**E** in previous picture) to **I2C2_SCL**
 - If using I²C bus **1**, **I2C2_SCL** is P9_17
 - If using I²C bus **2**, **I2C2_SCL** is P9_19
- Alphanumeric SDA Pin (**D** in previous picture) to **I2C2_SDA**
 - If using I²C bus **1**, **I2C2_SDA** is P9_18
 - If using I²C bus **2**, **I2C2_SDA** is P9_20
- Alphanumeric GND (**C** in previous picture) Pin to **DGND**
 - **DGND** is either P9_01, P9_02, P9_43, P9_44, P9_45, P9_46
- Alphanumeric VCC Pin (**B** in previous picture) to **VDD_3V3**
 - **VDD_3V3** is either P9_03, or P9_04
- Alphanumeric **VI²C** Pin (**A** in previous picture) to Alphanumeric VCC Pin (**B** in previous picture)

Here is an example setup, where I am using I²C bus 2:



How - to Guide: Alphanumeric Display

Usage

Turning it on

Connect your beaglebone to your PC and log in via the **serial port** or **ssh**.

If using I²C bus 1, configure **P9_17** and **P9_18**:

- (bbg) \$ **config-pin P9_17 i2c**
- (bbg) \$ **config-pin P9_18 i2c**

If using I²C bus 2, configure **P9_19** and **P9_20**:

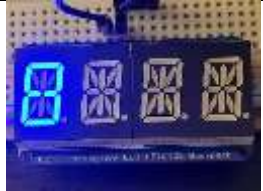

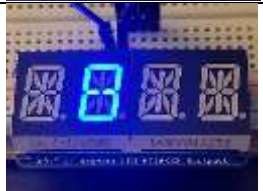



- (bbg) \$ **config-pin P9_19 i2c**
- (bbg) \$ **config-pin P9_20 i2c**

You can view the I²C devices on the bus using the following command:

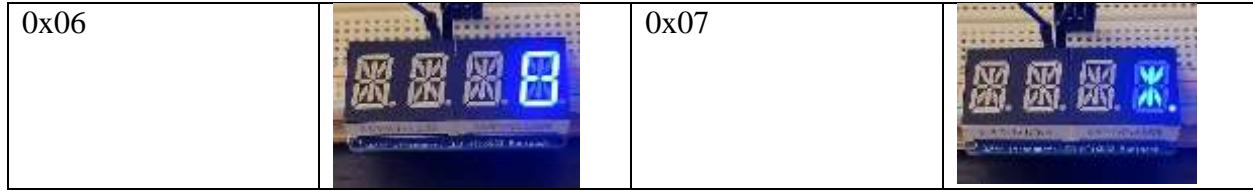
- For I²C bus 1:
 - (bbg) \$ **i2cdetect -y -r 1**
- For I²C bus 2:
 - (bbg) \$ **i2cdetect -y -r 2**

The address of the alphanumeric display is **0x70**.

We need to set the bits of certain registers to use the characters on the alphanumeric display. Each character on the display has two registers that control its inner and outer portions, as shown below.

| “Outer” Registers | Resulting display when all bits in register turned on | “Inner” Registers | Resulting display when all bits in register turned on |
|-------------------|---|-------------------|---|
| 0x00 |  | 0x01 |  |
| 0x02 |  | 0x03 |  |
| 0x04 |  | 0x05 |  |

How - to Guide: Alphanumeric Display



Steps for turning on the display:

1. We will use the **i2cset** command to control the display
 - a. The format of **i2cset** for the commands that I will show you is as follows:
 - i. **i2cset -y [bus number] [device address] [register] [value]**
 - b. I will be using I²C bus 2 for these steps
 - i. If you are using I²C bus 1, change the [bus number], as shown above, to **1** in the next steps

2. First, to initialize the display, set the bits of registers 0x00 – 0x07 to 0xFF:

- a. (bbg) \$ **i2cset -y 2 0x70 0x00 0xFF**
- b. (bbg) \$ **i2cset -y 2 0x70 0x01 0xFF**
- c. (bbg) \$ **i2cset -y 2 0x70 0x02 0xFF**
- d. (bbg) \$ **i2cset -y 2 0x70 0x03 0xFF**
- e. (bbg) \$ **i2cset -y 2 0x70 0x04 0xFF**
- f. (bbg) \$ **i2cset -y 2 0x70 0x05 0xFF**
- g. (bbg) \$ **i2cset -y 2 0x70 0x06 0xFF**
- h. (bbg) \$ **i2cset -y 2 0x70 0x07 0xFF**

3. Set the bits in the register 0x21 to 0xFF using the command:

- a. (bbg) \$ **i2cset -y 2 0x70 0x21 0xFF**

4. The register **0x81** turns on the display, and the register **0x80** turns off the display. Turn on the display using

- a. (bbg) \$ **i2cset -y 2 0x70 0x81 0xFF**

b. Expected display:



5. To clear all the bits in a register, drive the value **0x00** to a particular register. Example: clear the outer portion of the first character:

- a. (bbg) \$ **i2cset -y 2 0x70 0x00 0x00**

b. Expected display:



6. Command to turn off the display:

- a. (bbg) \$ **i2cset -y 2 0x70 0x80 0xFF**

How - to Guide: Alphanumeric Display

Displaying patterns

First, turn on the display:

- (bbg) \$ **i2cset -y 2 0x70 0x81 0xFF**

Clear all the registers in on the display:

- (bbg) \$ **i2cset -y 2 0x70 0x00 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x01 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x02 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x03 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x04 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x05 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x06 0x00**
- (bbg) \$ **i2cset -y 2 0x70 0x07 0x00**

As mentioned before, driving bits to registers 0x00 – 0x07 makes characters appear on the display.

As an example, let's display the number 7 on the first character of the display:

- (bbg) \$ **i2cset -y 2 0x70 0x00 0x07**

- Expected output:



You can drive any value from 0x00 to 0xFF to a register to display a pattern. It is very difficult for anybody to remember all of these different patterns, so here is an easier way of remembering them.:

When you drive a value such as 0xbd to a register, that value has a left number (0xb**d**), and a right number (0x**b**d).

The left number (0xb0) by itself produces this pattern:

- (bbg) \$ **i2cset -y 2 0x70 0x00 0xbd**



The right number (0x0d) by itself produces this pattern:

- (bbg) \$ **i2cset -y 2 0x70 0x00 0x0d**



How - to Guide: Alphanumeric Display

When we combine 0xb0 and 0x0d into 0xbd, we get this pattern, which is a combination of the previous two patterns :



- (bbg) \$ **i2cset -y 2 0x70 0x00 0xbd**

The resulting pattern is the letter G!

You can also create patterns by combining the patterns created by inner and outer registers. As an example, to display the letter I:

- (bbg) \$ **i2cset -y 2 0x70 0x00 0x09**
- (bbg) \$ **i2cset -y 2 0x70 0x01 0x12**

Result:



Here is a table containing all of the left / right values that you can combine to create patterns:

| Right number in outer register | Pattern | Left number in outer register | Pattern | Right number in inner register | Pattern | Left number in inner register | Pattern |
|--------------------------------|---------|-------------------------------|---------|--------------------------------|---------|-------------------------------|---------|
| 0x01 | | 0x10 | | 0x01 | | 0x10 | |
| 0x02 | | 0x20 | | 0x02 | | 0x20 | |
| 0x03 | | 0x30 | | 0x03 | | 0x30 | |
| 0x04 | | 0x40 | | 0x04 | | 0x40 | |
| 0x05 | | 0x50 | | 0x05 | | 0x50 | |
| 0x06 | | 0x60 | | 0x06 | | 0x60 | |

How - to Guide: Alphanumeric Display

| | | | | | | | |
|------|---|------|---|------|--|------|---|
| 0x07 |  | 0x70 |  | 0x07 |  | 0x70 |  |
| 0x08 |  | 0x80 |  | 0x08 |  | 0x80 |  |
| 0x09 |  | 0x90 |  | 0x09 |  | 0x90 |  |
| 0x0a |  | 0xa0 |  | 0x0a |  | 0xa0 |  |
| 0x0b |  | 0xb0 |  | 0x0b |  | 0xb0 |  |
| 0x0c |  | 0xc0 |  | 0x0c |  | 0xc0 |  |
| 0x0d |  | 0xd0 |  | 0x0d |  | 0xd0 |  |
| 0x0e |  | 0xe0 |  | 0x0e |  | 0xe0 |  |
| 0x0f |  | 0xf0 |  | 0x0f |  | 0xf0 |  |