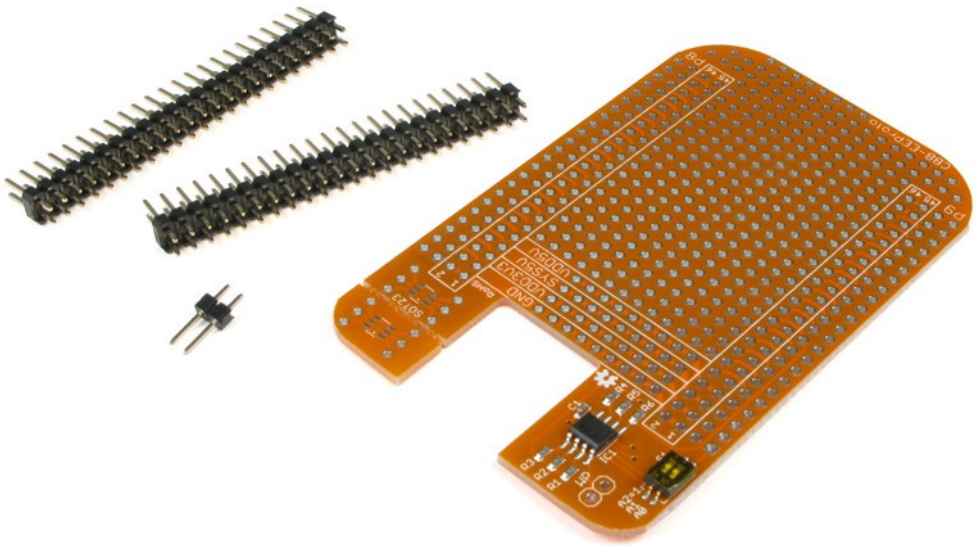


CBB-EEProto

Prototyping Cape for the BeagleBone and BeagleBone Black with Cape Manager compatible EEPROM circuit

Date: December 18, 2013
Revision: 1.0

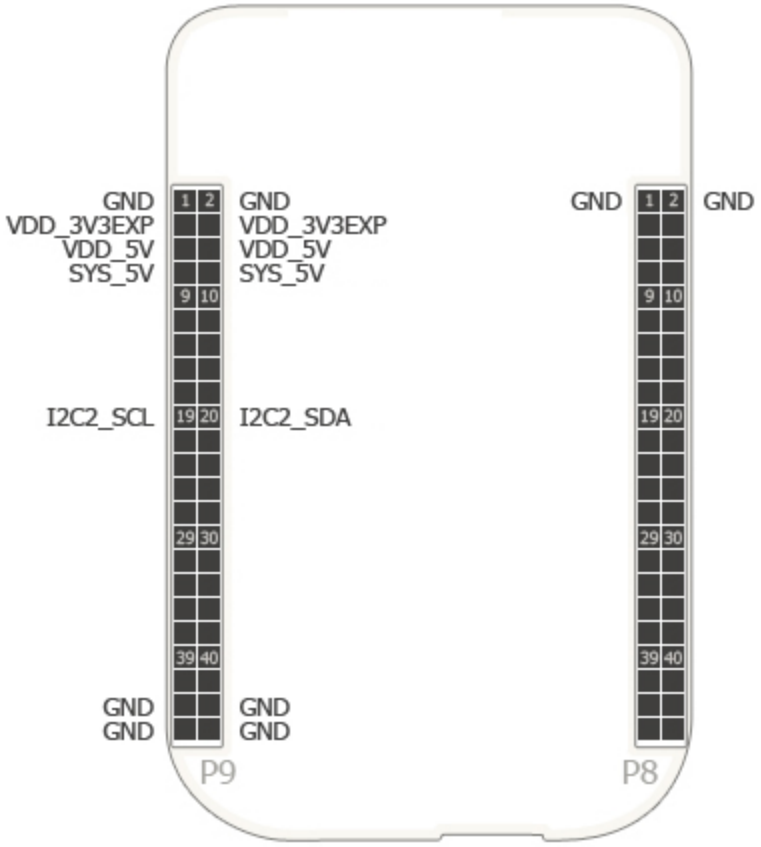
Description

The CBB-EEProto makes BeagleBone cape development simple by providing a large general purpose prototyping area as well as a pre-populated EEPROM circuit compatible with the BeagleBone's Capemgr (http://elinux.org/Capemgr), which automatically detects capes and loads their corrosponding Device Tree Overlay.

Features

- Large through-hole prototyping area
- Access to all IO pins on the P8 and P9 headers
- 256kbit EEPROM on I2C2 bus
- 2-position DIP switch for configuring EEPROM I2C address
- RoHS compliant

BeagleBone Pin Allocation



Requirements

The CBB-EEProto cape is designed to work with existing software on the BeagleBone Black.

Requirements for use (not included):

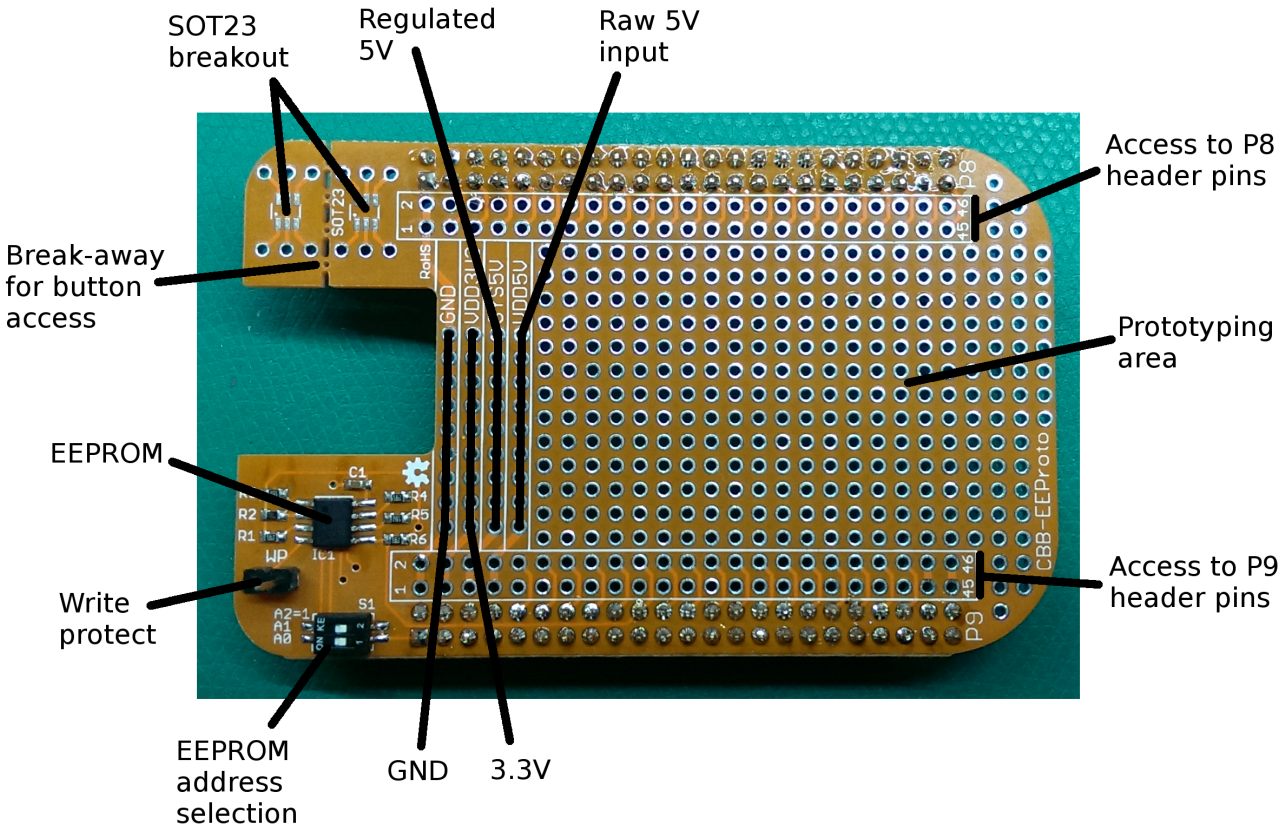
- BeagleBone Black with suitable distribution loaded
- 5V DC Power Supply for BeagleBone Black, 2A recommended

Getting Started

1. Make sure power is disconnected from the BeagleBone Black and connect the Cape to the BeagleBone Black by inserting the header pins in headers P8 and P9.
2. If using other capes, make sure the DIP switch on every Cape is set to a different EEPROM ID.
3. Ensure the BeagleBone Black is loaded with a suitable Linux distribution. Angstrom release 09.04.2013 or later is recommended. Other distributions may also be compatible with this Cape.
4. Connect a 5V DC Power Supply to the BeagleBone Black.
5. The Cape Manager will read the EEPROM, and if it points to a Device Tree overlay it will be loaded automatically.

Instructions to update the software image on the BeagleBone Black can be found at
http://circuitco.com/support/index.php?title=Updating_The_Software

Key Component Locations



WP header
Connect these pins to disable write protect when writing the EEPROM.

EEPROM Details

The EEPROM on the Cape is used to load the BeagleBone Black with the appropriate configuration for the Cape.

EEPROM Support: YES
Board Name: (User Programmable)
Manufacturer: (User Programmable)

Using the EEPROM with Cape Manager

When the BeagleBone boots with the cape attached the Cape Manager will see the EEPROM on the I2C bus and will create the special file:

```
/sys/bus/i2c/devices/1-005x/eeprom
```

Where `1-005x` will indicate the EEPROM address in hexadecimal. The switches, labeled **A0** and **A1**, configure the EEPROM address by setting the lower two bits of the EEPPROM's base address:

```
1 0 1 0 1 x x
        ↗   ↖
      A1     A0
```

This allows for the four possible addresses 0x54, 0x55, 0x56 and 0x57. Sliding a switch to the left sets the corresponding bit to 0, and to the right sets it to 1. When the cape arrives both **A0** and **A1** are set to 1, making the address 0x57, so the eeprom file will be in the `1-0057/` directory.

The cape data resides in the first 244 bytes of the EEPROM. To view it:

```
# hexdump -n 244 -C /sys/bus/i2c/devices/1-0057/eeprom
```

When your CBB-EEProto cape arrives the EEPROM will have all its bits set to on, so the output should be:

```
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
000000f0  ff ff ff ff                                      |....|
000000f4
```

First test that the EEPROM is working correctly by shorting the **WP** pins and writing some data to it:

```
# echo test > /sys/bus/i2c/devices/1-0057/eeprom
```

If the **WP** pins are not properly shorted you will receive the error:

```
-sh: echo: write error: Connection timed out
```

Once you've written the data, read the contents of the EEPROM again and verify it has been changed:

```
# hexdump -n 244 -C /sys/bus/i2c/devices/1-0057/eeprom
00000000  74 65 73 74 0a ff ff ff  ff ff ff ff ff ff ff ff  |test............|
00000010  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
000000f0  ff ff ff ff                                      |....|
000000f4
```

The required format of the EEPROM data is well documented in the BeagleBone Black System Reference Manual, and there are a number of tools available that can create the EEPROM data for you. One such tool is David Oliveira's BBCape_EEPROM, which can be found at https://github.com/picoflamingo/BBCape_EEPROM. First download and compile BCape_EEPROM:

```
# cd ~
# git clone https://github.com/picoflamingo/BBCape_EEPROM.git
# cd BBCape_EEPROM
# make
```

As a first test, create an EEPROM file with the cape name 'test' and version '00A0':

```
# ./bbcape_eeprom
BBCapeEEPROM-TOP> b
BBCapeEEPROM-BOARD> 1
test cape
BBCapeEEPROM-BOARD> 2
00A0
BBCapeEEPROM-BOARD> 4
test
BBCapeEEPROM-BOARD> u
BBCapeEEPROM-TOP> w test.eeprom
BBCapeEEPROM-TOP> q
```

This will create a file called `test.eeprom`. Connect the **WP** pins, then load the file on the EEPROM and confirm it has been written:

```
# cat test.eeprom > /sys/bus/i2c/devices/1-0057/eeprom
# hexdump -n 244 -C /sys/bus/i2c/devices/1-0057/eeprom
00000000  aa 55 33 ee 41 31 74 65  73 74 20 63 61 70 65 00  |.U3.A1test cape.|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 30 30  41 30 70 69 63 6f 46 6c  |......00A0picoFl|
00000030  61 6d 69 6e 67 6f 00 00  00 00 74 65 73 74 00 00  |amingo....test..|
00000040  00 00 00 00 00 00 00 00  00 00 30 30 32 39 31 32  |..........002912|
00000050  57 54 48 52 30 33 38 33  00 00 00 00 00 00 00 00  |WTHR0383........|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000f0  00 00 00 00                                       |....|
000000f4
```

You can enter a '?' at any time in the BBCape_EEPROM program to see a list of commands.

Next create a Device Tree overlay file for the Cape Manager to load:

```
# cat > test-00A0.dts
/dts-v1/;
/plugin/;

/{
  compatible = "ti,beaglebone", "ti,beaglebone-black";

  part-number = "test";
  version = "00A0";
  exclusive-use = "P9.16";

  fragment@0 {
    /* Sets pinmux for general purpose outputs. */
    target = <&am33xx_pinmux>;
    __overlay__ {
      test_pins: pinmux_test_pins {
        pinctrl-single,pins = <
          0x04c 0x7  /* gpmc_a3 - MODE7 (GPIO1_19) */
        >;
      };
    };
  };

  fragment@1 {
    /* Enable pinmux-helper driver for setting input pull configuration. */
    target = <&ocp>; /* On-Chip Peripherals */
    __overlay__ {
      test-pinmux {
        compatible = "bone-pinmux-helper"; /* Use the pinmux helper */
        status="okay";
        /* Define custom names for indexes in pinctrl array: */
        pinctrl-names = "default";
        /* Set the elements of the pinctrl array to the pinmux overlays
           defined above: */
        pinctrl-0 = <&test_pins>;
      };
    };
  };
};
```

Press `ctrl+D` to finish writing the file.

This overlay will mux **P9.18** as **GPIO1_19** with no pull and RX disabled. Compile the overlay and move it to the firmware directory:

```
# dtc -O dtb -o test-00A0.dtbo -b 0 -@ test-00A0.dts
# mv test-00A0.dtbo /lib/firmware
```

Note that the compiled overlay file must be named `[cape-name]-[cape-version].dtbo`, where `[cape-name]` and `[cape-version]` match the cape part number and version in the EEPROM file as well as the `part-number` and `version` values in the overlay file.

Once the EEPROM is written and the compiled overlay is in `/lib/firmware`, restart the BeagleBone and confirm that the overlay was loaded successfully:

```
# cat /sys/devices/bone_capemgr.9/slots
 0: 54:PF---
 1: 55:PF---
 2: 56:PF---
 3: 57:P---L test cape,00A0,picoFlamingo,test
 4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
 5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
```

Confirm the overlay is working:

```
# echo 51 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio51/direction
```

To set the pin high:

```
# echo 1 > /sys/class/gpio/gpio51/value
```

And to set it low:

```
# echo 0> /sys/class/gpio/gpio51/value
```

When finished using the pin it should be exported back to kernel control:

```
# echo 51 > /sys/class/gpio/unexport
```

For more info on creating and using Device Tree overlays for the BeagleBone cape manager see:

- http://elinux.org/Capemgr
- http://elinux.org/BeagleBone_and_the_3.8_Kernel
- http://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree/
- http://www.armhf.com/index.php/using-beaglebone-black-gpios/

Open Source Hardware

This product is Open Source Hardware. Design materials, schematics and source code is available on GitHub at https://github.com/lgxlogic/CBB-EEProto

Design materials are NOT SUPPORTED and DO NOT constitute a reference design. THERE IS NO WARRANTY FOR THE DESIGN MATERIALS, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE DESIGN MATERIALS "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE DESIGN MATERIALS IS WITH YOU. SHOULD THE DESIGN MATERIALS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
We mean it, these design materials may be totally unsuitable for any purposes.

License

Change History

[12/18/2013] Version 1.0 Initial Release

More Information

For more information, see *www.logicsupply.com*