BeagleBone 101

- Joystick
- Ultrasonic sensor
- PIR motion sensor
- Accelerometer
- Temperature and pressure
- Servo motor

- Cape demos
  - Bacon Cape

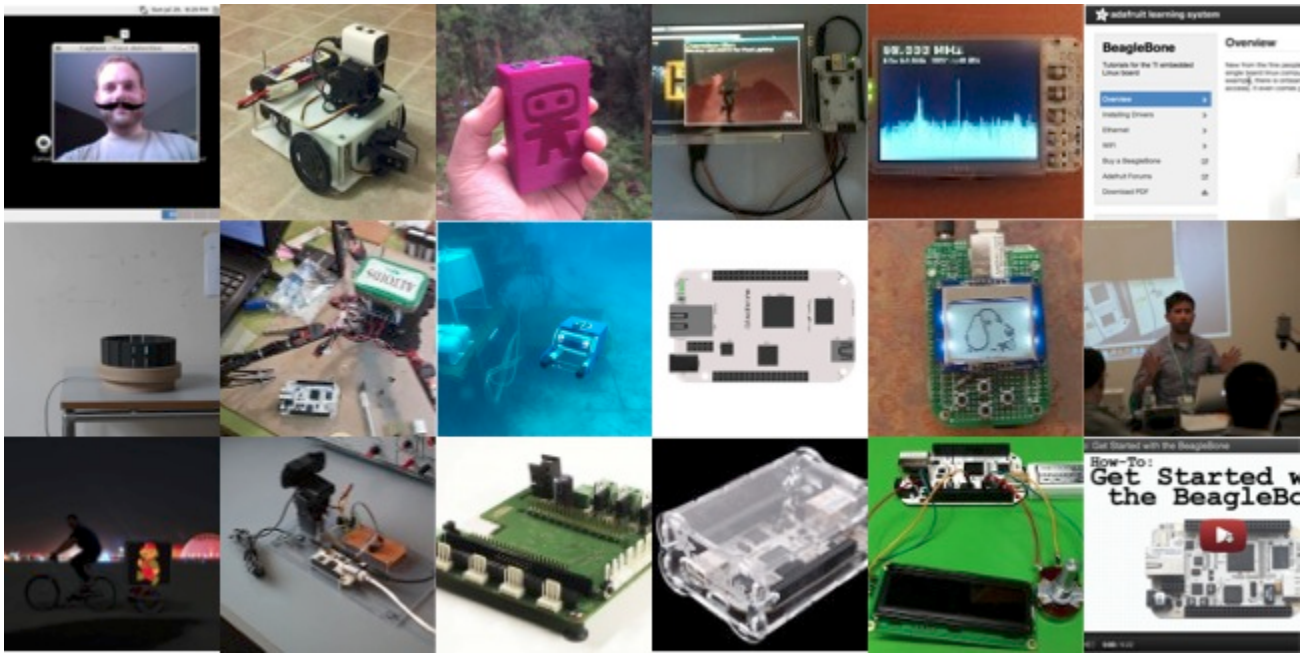
Did you know? This page can interact with your BeagleBone

Type in your BeagleBone's IP address here:

BeagleBone: open-hardware expandable computer

Artist-tested, engineer approved

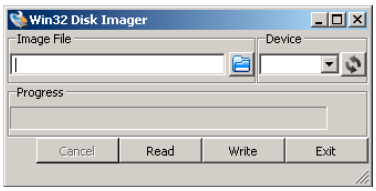
The left-hand navigation bar will help you explore your board and learn how to program it.



Latest ARM open source focused on easy

hardware experimentation

- Ships ready to use
  - Angstrom Distribution with C++, Perl, Python, ...
  - Linux drivers support countless USB peripherals
  - Interactive tutorial to start learning about capabilities
- Open source means options
  - Texas Instruments releases: Android, Linux, StarterWare (no OS)
  - Linux: Angstrom Distribution, Ubuntu, Debian, ArchLinux, Sabayon, Buildroot, Erlang, Fedora
  - Other: QNX, FreeBSD
  - Projects page
- SD card images like get-out-of-jail-free card




```
xzcat XXX.img.xz | sudo dd of=/dev/sdX
```

  - Can be used just as easily for backups
  - Board can be booted from SD using device ROM, so you can't "brick" it
  - 7-zip and Ubuntu Win32DiskImager enable programming cards from Windows

Update board with latest software

There are multiple ways to run initial software on your board, but it is likley that the simplest way to get an update is to create an exact replica of a bootable microSD card and boot off of it. The BeagleBone Black Rev C has 4GB of eMMC storage that can be initialized by a program booted off of a microSD card. If you want to update to the latest software image for your board, this is a way to do that.

See updates for the step-by-step guide.

Information about getting the source code for the image shipped with your board can be found at beagleboard.org/source, along with instructions for rebuilding it.

BoneScript interactive guide

BoneScript is a JavaScript library to simplify learning how to perform physical computing tasks using your embedded Linux. This web page is able to interact with your board to provide an interactive tutorial.

Example run restore

```
1  var b = require('bonescript');
2  b.pinMode('USR0', b.OUTPUT);
3  b.pinMode('USR1', b.OUTPUT);
4  b.pinMode('USR2', b.OUTPUT);
5  b.pinMode('USR3', b.OUTPUT);
6  b.digitalWrite('USR0', b.HIGH);
7  b.digitalWrite('USR1', b.HIGH);
8  b.digitalWrite('USR2', b.HIGH);
9  b.digitalWrite('USR3', b.HIGH);
10 setTimeout(restore, 2000);
11
```

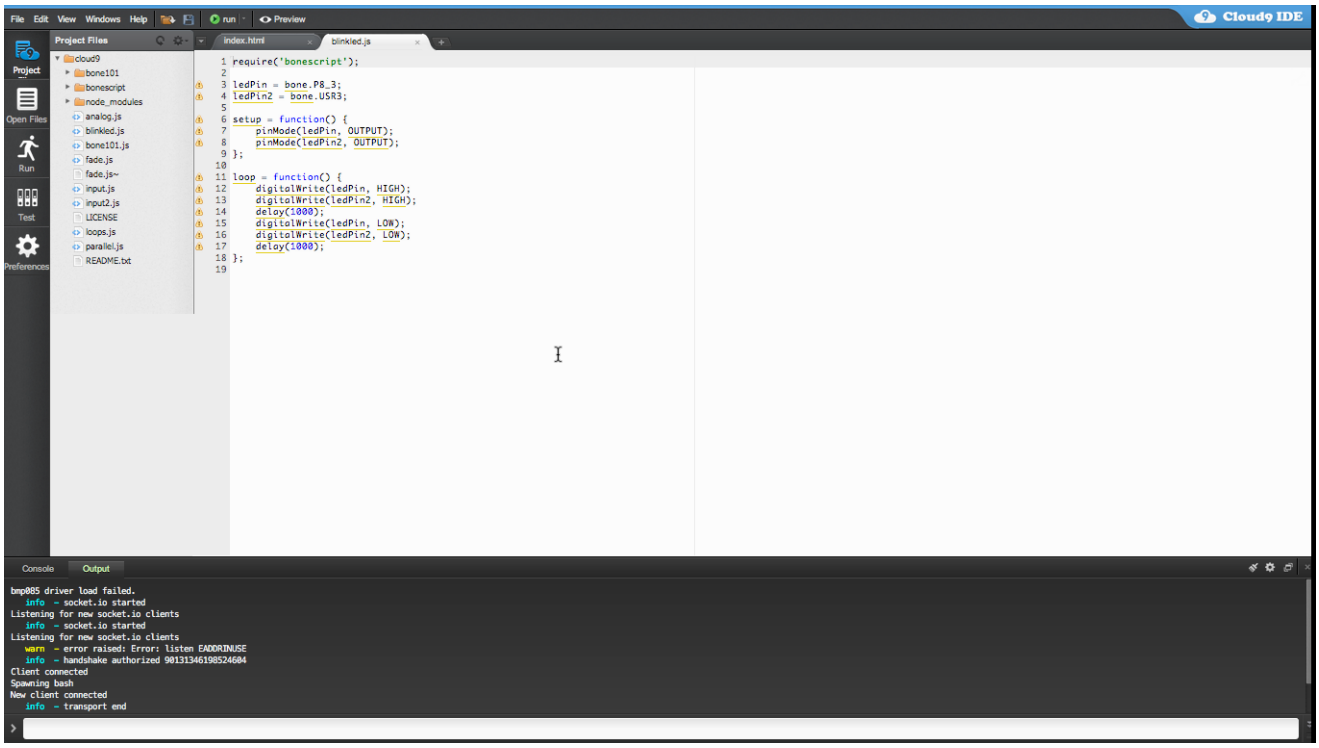
Running the above example will cause all of your LEDs to light up at once for a couple of seconds.

To learn more about Bonescript, please continue exploring this interactive guide.

Cloud9 IDE

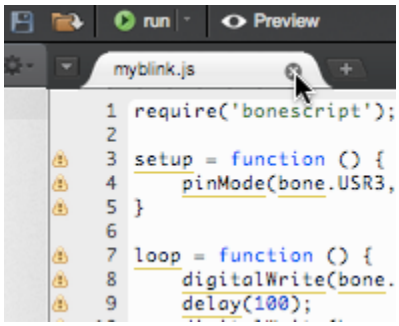
To begin editing programs that live on your board, you can use the Cloud9 IDE.

If your board is plugged into your USB port, click on the "Cloud9 IDE" link above to start the editor.

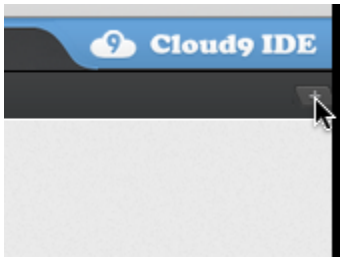


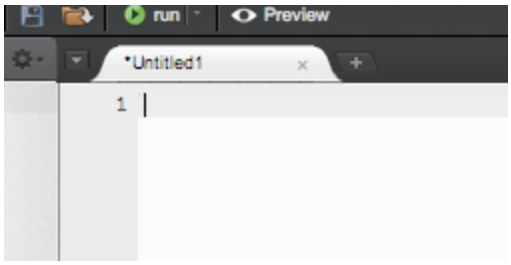
As a simple exercise to become familiar with Cloud9 IDE and the Bonescript JavaScript library, creating a simple application to blink one of the 4 user programmable LEDs on the BeagleBone is a good start.

- **Step A:** Close any open file tabs.



- **Step B:** Click the "+" in the top-right to create a new file.

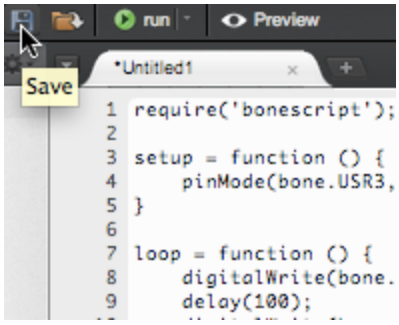


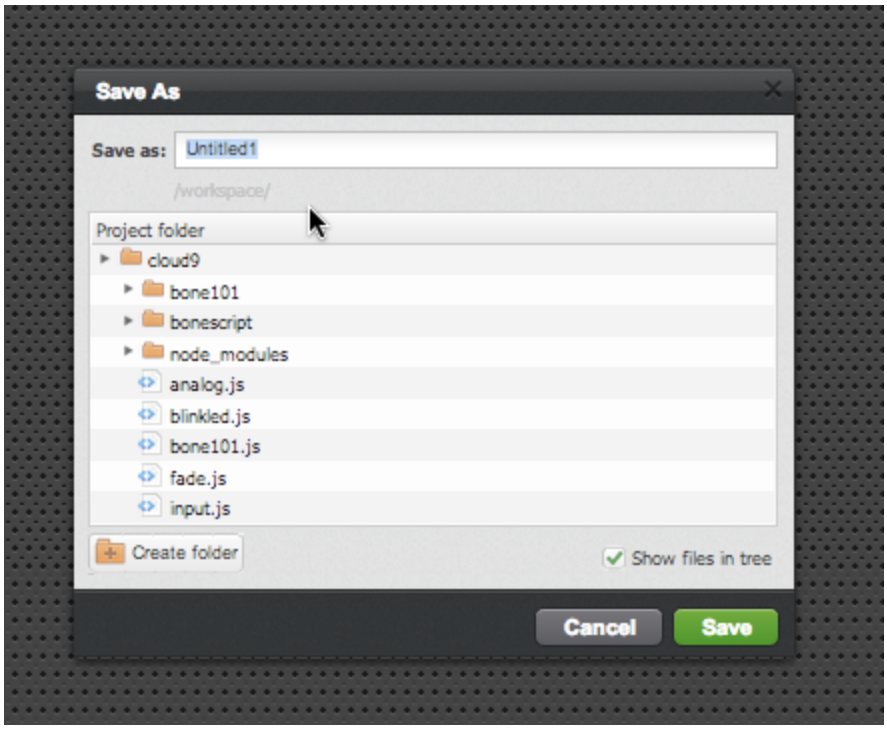


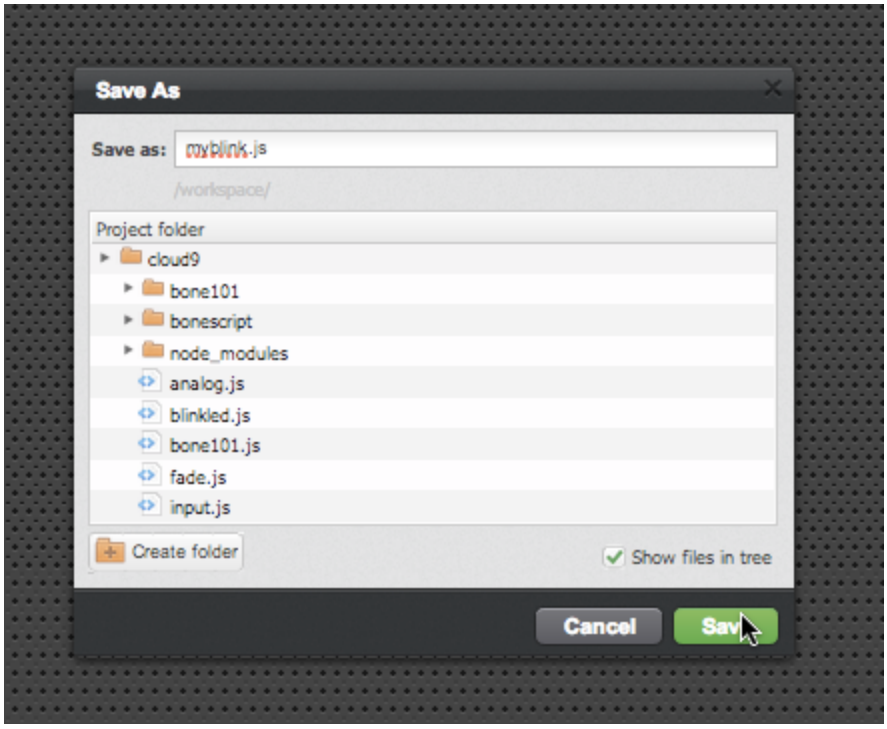
- **Step C:** Cut and paste the following code into the new tab:

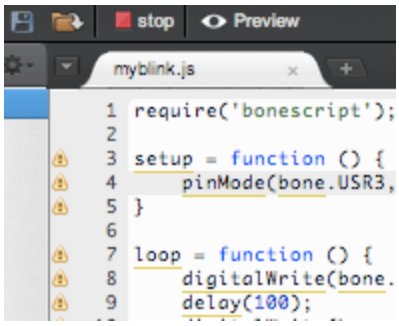
```
var b = require('bonescript');

var state = b.LOW;

b.pinMode("USR0", b.OUTPUT);
b.pinMode("USR1", b.OUTPUT);
b.pinMode("USR2", b.OUTPUT);
b.pinMode("USR3", b.OUTPUT);
setInterval(toggle, 1000);

function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite("USR3", state);
}
```

- **Step D:** Save the file by clicking the disk icon and giving the file a name with the .js extension.

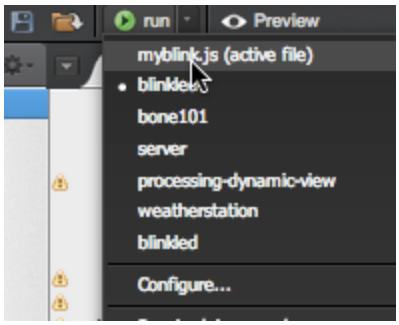



Save As
Save as: Untitled1
/workspace/
Project folder
cloud9
bone101
bonescript
node_modules
analog.js
blinkled.js
bone101.js
fade.js
input.js
Create folder
Show files in tree
Cancel
Save


Save As
Save as: myblink.js
/workspace/
Project folder
cloud9
bone101
bonescript
node_modules
analog.js
blinkled.js
bone101.js
fade.js
input.js
Create folder
Show files in tree
Cancel

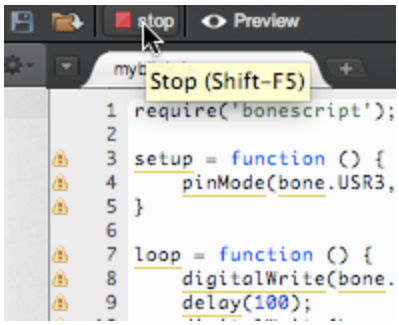


- **Step E:** Run the code by selecting the arrow to the right of "run" (or "debug") in the toolbar to pull down the list of files to run and select your new file.



- **Step F:** Observe the BeagleBone USR3 LED blinking steadily about 5 times a second.

- **Step G:** Stop the code by clicking "stop" in the toolbar.



Additional information about the Bonescript library is available in the presentation viewed in the next step and on-line at http://beagleboard.org/project/bonescript.

Autorun

Once you've finished developing your JavaScript application, you can have it start upon boot-up by simply dropping it into the 'autorun' subfolder (located at /var/lib/cloud9/autorun in the file system).

The systemd bonescript-autorun.service runs at start-up and uses the /usr/lib /node_modules/bonescript/autorun.js script to automatically detect when .js files are in this directory and invoke them as separate processes with node.js. When the files are changed or moved, the script will kill the processes.

Resources

To learn more about Cloud9 IDE and to synchronize the software on your board with cloud-hosted services, see www.c9.io.

For more information on Node.JS, the JavaScript interpreter, see www.nodejs.org. Note that version 0.10.25 is what is currently installed on the default image and you can find the api documentation at www.nodejs.org/docs /v0.10.25/api.

For more information about the Bonescript library, see www.beagleboard.org/bonescript.

List of common Linux commands

- *pwd* - show current directory
- *cd* - change current directory
- *ls* - list directory contents
- *chmod* - change file permissions
- *chown* - change file ownership
- *cp* - copy files
- *mv* - move files
- *rm* - remove files
- *mkdir* - make directory
- *rmdir* - remove directory
- *cat* - dump file contents
- *less* - progressively dump file
- *vi* - edit file (complex)
- *nano* - edit file (simple)
- *head* - trim dump to top
- *tail* - trim dump to bottom
- *echo* - print/dump value
- *env* - dump environment variables
- *export* - set environment variable
- *history* - dump command history
- *grep* - search dump for strings
- *man* - get help on command
- *apropos* - show list of man pages
- *find* - search for files
- *tar* - create/extract file archives
- *gzip* - compress a file
- *gunzip* - decompress a file
- *du* - show disk usage
- *df* - show disk free space
- *mount* - mount disks
- *tee* - write dump to file in parallel
- *hexdump* - readable binary dumps

Other programming environments

The board also ships with gcc, python and more that can be invoked directly from the command-line.

Hardware documentation

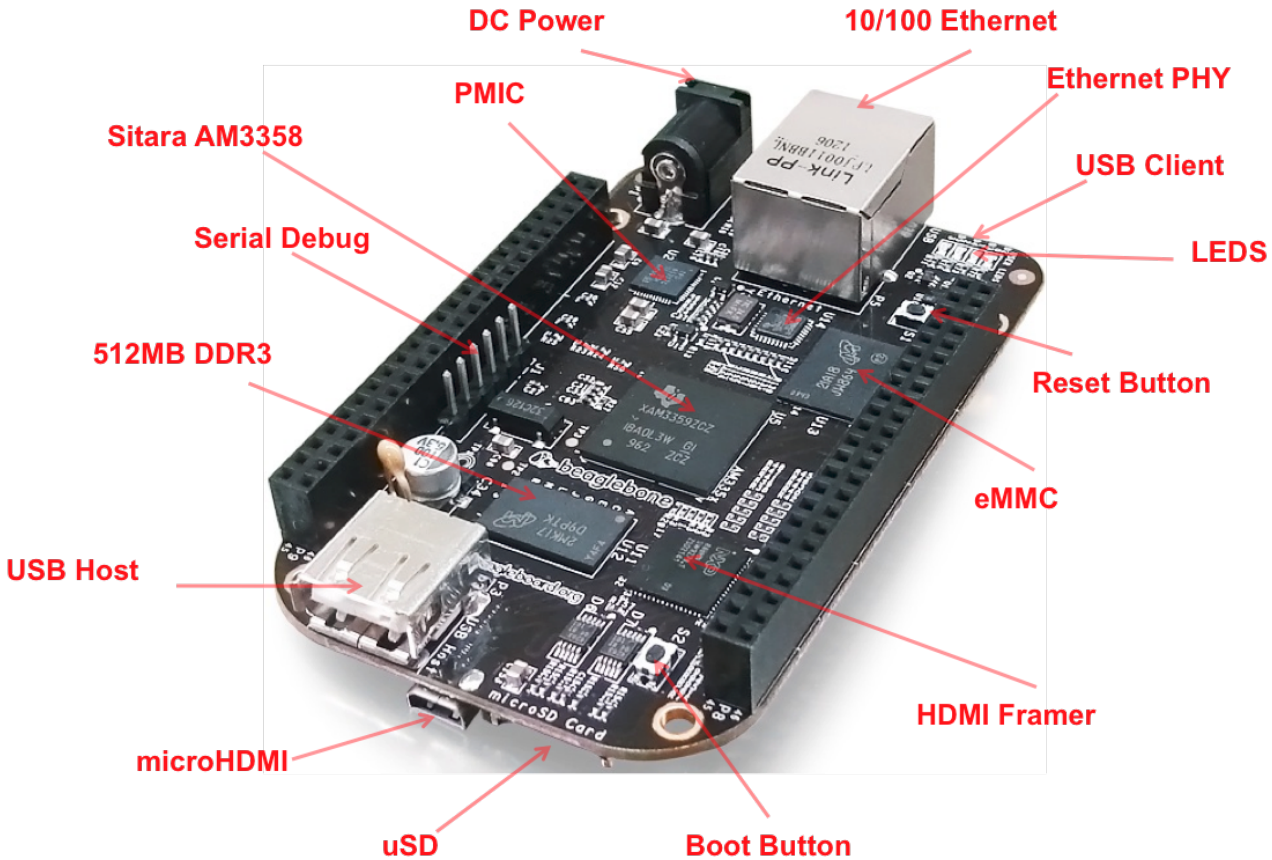
The BeagleBoneBlack wiki page documents all of the known hardware issues, as well as the latest available software, hardware documentation and design materials.

Always read the System Reference Manual!!!

Design materials

Design materials for creating your own customized version of the hardware or for better understanding the design are also linked from the traditional home of "http://beagleboard.org/hardware/design"

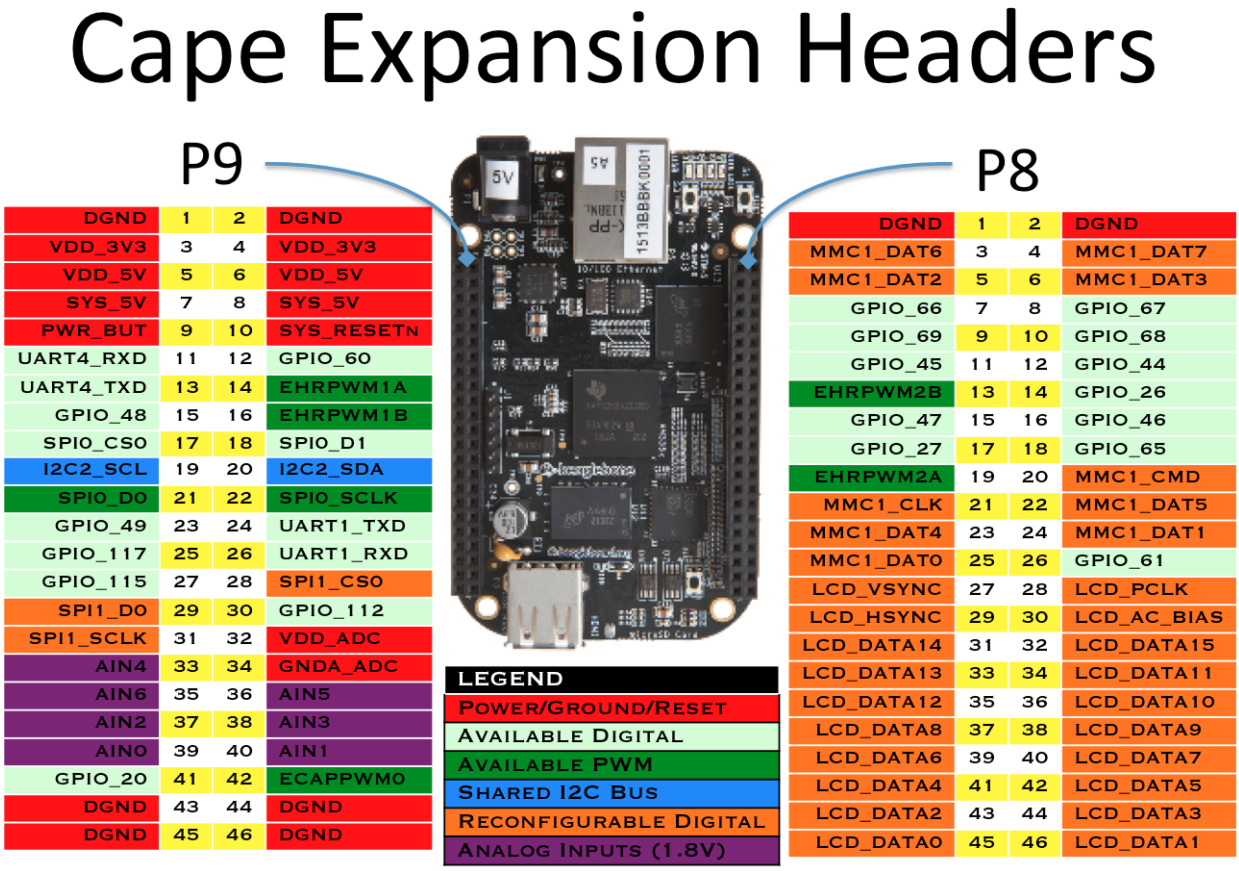
BeagleBone Black hardware details



Revision A5 also provides a POWER button that can be used to enter and exit hibernate modes once that feature is implemented in the software.

Headers

The expansion headers provide extensive I/O capabitilities.



P9			P9
DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

P8			P8
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

Each digital I/O pin has 8 different modes that can be selected, including GPIO.

65 possible digital I/Os

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

In GPIO mode, each digital I/O can produce interrupts.

8 PWMs and 4 timers

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
EHRPWM0B	21	22	EHRPWM0A
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	ECAPPWM2
EHRPWM0B	29	30	GPIO_112
EHRPWM0A	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
TIMER4	7	8	TIMER7
TIMER5	9	10	TIMER6
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	EHRPWM1B
GPIO_8	35	36	EHRPWM1A
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
EHRPWM2A	45	46	EHRPWM2B

Up to 8 digital I/O pins can be configured with pulse-width modulators (PWM) to produce signals to control motors or create analog voltage levels, without taking up any extra CPU cycles.

7 analog inputs (1.8V)

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

Make sure you don't input more than 1.8V to the analog input pins.

This is a single 12-bit analog-to-digital converter with 8 channels, 7 of which are made available on the headers.

4 UARTs and 1 TX only

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
UART1_RTSN	19	20	UART1_CTSN
UART2_TXD	21	22	UART2_RXD
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	UART3_TXD
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
UART5_CTSN+	31	32	UART5_RTSN
UART4_RTSN	33	34	UART3_RTSN
UART4_CTSN	35	36	UART3_CTSN
UARR5_TXD+	37	38	UART5_RXD+
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

There is a dedicated header for getting to the UART0 pins and connecting a debug cable. Five additional serial ports are brought to the expansion headers, but one of them only has a single direction brought to the headers.

2 I2C ports

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
I2C1_SCL	17	18	I2C1_SDA
I2C2_SCL	19	20	I2C2_SDA
I2C2_SCL	21	22	I2C2_SDA
GPIO_49	23	24	I2C1_SCL
GPIO_117	25	26	I2C1_SDA
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

The first I2C bus is utilized for reading EEPROMS on cape add-on boards and can't be used for other digital I/O operations without interfering with that function, but you can still use it to add other I2C devices at available addresses.

The second I2C bus is available for you to configure and use.

2 SPI ports

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
SPI0_CS0	17	18	SPI0_D1
SPI1_CS1	19	20	SPI1_CS0
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	SPI1_D1
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	SPI1_CS1
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

For shifting out data fast, you might consider using one of the SPI ports.

25 PRU low-latency I/Os

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
PRU0_7	25	26	PRU1_16 IN
PRU0_5	27	28	PRU0_3
PRU0_1	29	30	PRU0_2
PRU0_0	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
PRU0_6	41	42	PRU0_4
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
PRU0_15 OUT	11	12	PRU0_14 OUT
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	PRU1_13
PRU1_12	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
PRU1_8	27	28	PRU1_10
PRU1_9	29	30	PRU1_11
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
PRU1_6	39	40	PRU1_7
PRU1_4	41	42	PRU1_5
PRU1_2	43	44	PRU1_3
PRU1_0	45	46	PRU1_1

Advanced users can also make use of 2 built-in 32-bit 200-MHz microcontrollers called Programmable Real-time Units (PRUs) for performing real-time tasks. Each PRU has some pins associated with it tied directly to registers for super-low-latency access.

Capes

Capes are really just daughterboards for BeagleBones, but we refer to them enough that a short name makes sense. BeagleBone wears them, they typically have a cut-out around the Ethernet connector that gives them a cape-like shape and *Underdog is a Beagle*!

The beaglebonecapes.com site attempts to consolidate the currently available cape add-on boards for BeagleBone and BeagleBone Black.

What is a cape?

The types of capes available is quite diverse, from 3D printer capes and touchscreen displays to wireless communications and FPGA-based prototyping tools.

Cape EEPROM contents

Name	Offset	Size (bytes)	Contents (Big Endian)
Header	0	4	0xAA 0x55 0x33 0xEE
EEPROM rev	4	2	0x41 0x31 (ASCII "A1") – alphabetical string order
Board Name	6	32	ASCII readable board name - trailing null or space
Version	38	4	ASCII readable version – alphabetical string order
Manufacturer	42	16	ASCII readable company or individual's name
Part Number	58	16	ASCII readable part numbering for ordering
Number of Pins	74	2	16-bit integer number of pins used, up to 92
Serial Number	76	12	ASCII readable serial number: WWYY&&&&NNNN WW = week of manufacture YY = year of manufacture &&&& = assembly code determined by developer NNNN = incrementing board number for the week
Pin Usage	88	148	2 bytes per pin for 74 pins Bit 15: 0 = unused, 1 = used Bits 14-13: 10 = output, 01 = input, 11 = bidir Bits 12-7: reserved Bit 6 (Slew Rate): 0 = fast, 1 = slow Bit 5 (RX Enable): 0 = disabled, 1 = enabled Bit 4 (Pull-up Select): 0 = pull-down, 1 = pull-up Bit 3 (Pull-up Enable): 0 = disabled, 1 = enabled Bits 2-0 (Mux Mode): mode 0-7
VDD_3V3EXP Current	236	2	16-bit integer of maximum current in mA
VDD_5V Current	238	2	16-bit integer of maximum current in mA
SYS_5V Current	240	2	16-bit integer of maximum current in mA
DC Supplied	242	2	16-bit integer of current supplies to VDD_5V
Available	244	32543	For manufacturer's use

Every cape that utilizes pins on the expansion header is expected to notify software on the board by providing contents within EEPROM connected to I2C2 with an address from 0x54 to 0x57. To work with other capes, you can make your address selectable between those values to avoid conflicts. It is also possible to provide through connectors such that other capes can also get connections to the cape header pins

Headers on BeagleBone Black

Black eMMC and HDMI pins

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_ACBIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

BeagleBone Black ships with two virtual capes already on it, one for the on-board eMMC storage and one for the HDMI output. When configured for use these virtual capes consume actual resources.

If the eMMC is not placed in reset, the MMC1* signals may not be used without potentially corrupting the contents of your on-board eMMC---and possibly damaging the physical circuit as well.

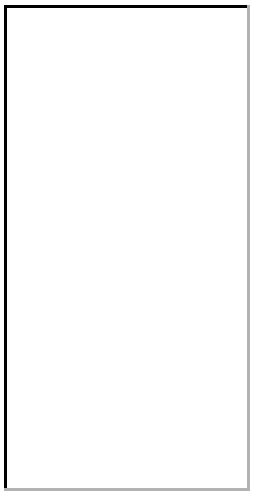
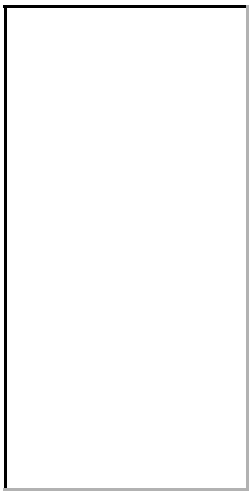
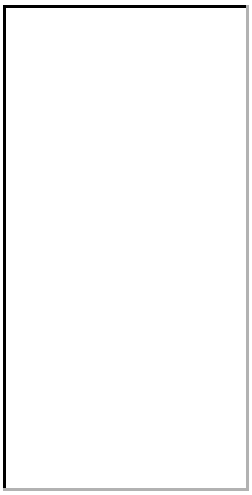
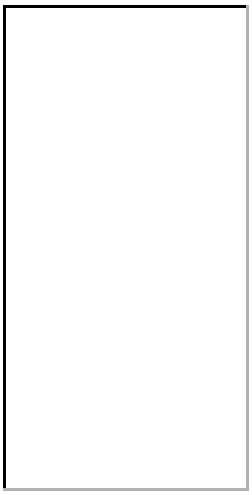
The HDMI signals are all inputs to the HDMI device, so there is no need to put the HDMI device into reset, but using those pins for other operations will cause the HDMI output to be disrupted. However, note that the Linux software typically allocates these for use by the HDMI driver, so your software might not be able to get access to them without unloading that driver.

Cape demos

- *Bacon Cape* - Teach basic interactions

Books

For a complete list of books on BeagleBone, see http://beagleboard.org/books.

Bad to the Bone

Perfect for high-school seniors or freshman univerisity level text, consider using "Bad to the Bone"

BeagleBone Cookbook

A lighter treatment suitable for a bit broader audience without the backgrounders on programming and electronics, consider "BeagleBone Cookbook"

Exploring BeagleBone and Embedded Linux Primer

To take things to the next level of detail, consider "Exploring BeagleBone" which can be considered the missing software manual and utilize "Embedded Linux Primer" as a companion textbook to provide a strong base on embedded Linux suitable for working with any hardware that will run Linux.

Last updated by default on Tue Jul 10 2018 19:39:54 GMT-0000 (UTC).

- Boards
- Getting Started

- Support

- Community
- Projects
- Videos

- About Us
- Privacy policy
- Terms of Use