BeagleBoard Docs

Release 1.0.20230308-wip

BeagleBoard.org Foundation

Mar 08, 2023

Table of contents

Chapter 1

Introduction

Welcome to the BeagleBoard documentation project. If you are looking for help with your Beagle open-hardware development platform, you've found the right place!

Important: This documentation is a work in progress. For the latest versions of this documentation, be sure to check the official release sites:

- https://docs.beagle.cc (cached with local proxies)
- https://docs.beagleboard.org (non-cached, without proxies)

For bleeding edge (development-stage) documentation:

- https://docs.beagleboard.io (straight from docs repo)

Please check out our *Support* page to find out how to get started, resolve issues, and engage with the developer community. Don't forget that this is an open-source project! Your contributions are welcome. Learn about how to contribute to the BeagleBoard documentation project and any of the many open-source Beagle projects ongoing on our *Contribution* page.

Warning: Make sure you thoroughly read and agree with our *Terms & Conditions* which covers warnings, restrictions, disclaimers, and warranty for all of our boards. Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

1.1 Support

1.1.1 Getting started

The starting experience for all Beagles has been made to be as consistent as is possible. For any of the Beagle Linux-based open hardware computers, visit *Getting Started Guide*.

Getting Started Guide

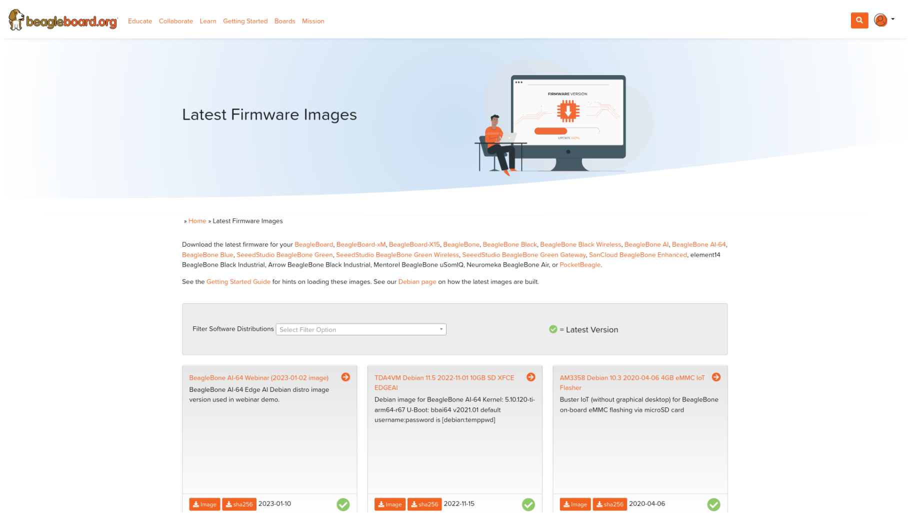
Beagles are tiny computers ideal for learning and prototyping with electronics. Read the step-by-step getting started tutorial below to begin developing with your Beagle in minutes.

Update board with latest software This step may or may not be necessary, depending on how old a software image you already have, but executing this step, the longest step, will ensure the rest will go as smooth as possible.

Download the latest software image Download the latest software image from beagleboard.org distros page. The "IoT" images provide more free disk space if you don't need to use a graphical user interface (GUI).

Note: Due to sizing necessities, this download may take 30 minutes or more.

The Debian/Ubuntu distribution is provided for the boards. The file you download will have an .img.xz extension. This is a compressed sector-by-sector image of the SD card.



Latest Firmware Images

» Home » Latest Firmware Images

Download the latest firmware for your BeagleBoard, BeagleBoard-xM, BeagleBoard-X15, BeagleBone, BeagleBone Black, BeagleBone Black Wireless, BeagleBone AI, BeagleBone AI-64, BeagleBone Blue, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SeeedStudio BeagleBone Green Gateway, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial, Mentorel BeagleBone uSomIQ, Neuromeka BeagleBone Air, or PocketBeagle.

See the Getting Started Guide for hints on loading these images. See our Debian page on how the latest images are built.

Filter Software Distributions: Select Filter Option

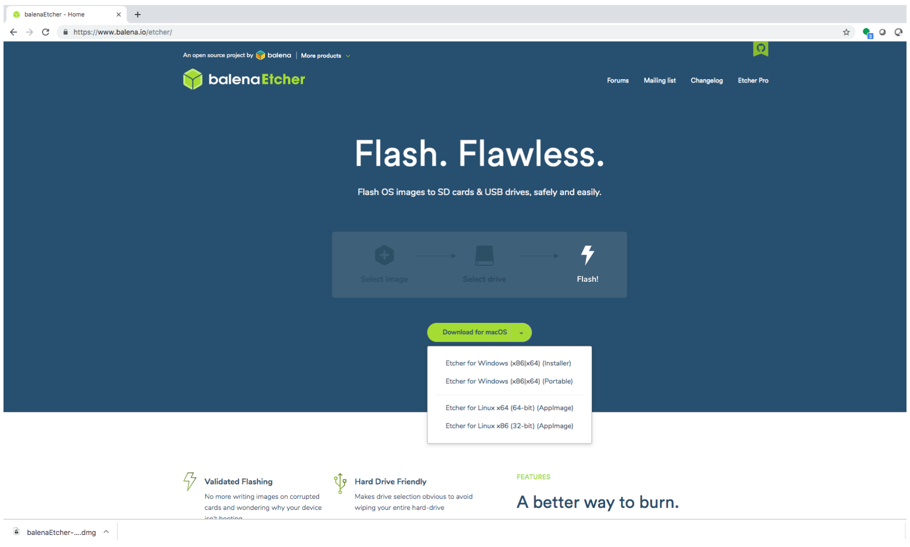
✓ = Latest Version

BeagleBone AI-64 Webinar (2023-01-02 image)
BeagleBone AI-64 Edge AI Debian distro image version used in webinar demo.
Image | sha256 | 2023-01-10 ✓

TDA4VM Debian 11.5 2022-11-01 10GB SD XFCE EDGEAI
Debian image for BeagleBone AI-64 Kernel: 5.10.120-ti-arm64-r67 U-Boot: bbai64 v2021.01 default username:password is [debian:temppwd]
Image | sha256 | 2022-11-15 ✓

AM3358 Debian 10.3 2020-04-06 4GB eMMC IoT Flasher
Buster IoT (without graphical desktop) for BeagleBone on-board eMMC flashing via microSD card
Image | sha256 | 2020-04-06 ✓

Install SD card programming utility Download and install balenaEtcher.


balenaEtcher - Home
https://www.balena.io/etcher/
An open source project by balena
More products
balenaEtcher
Forums
Mailing list
Changelog
Etcher Pro
Select image
Select drive
Flash!
Download for macOS
Etcher for Windows (x86|x64) (Installer)
Etcher for Windows (x86|x64) (Portable)
Etcher for Linux x64 (64-bit) (AppImage)
Etcher for Linux x86 (32-bit) (AppImage)
balenaEtcher-....dmg

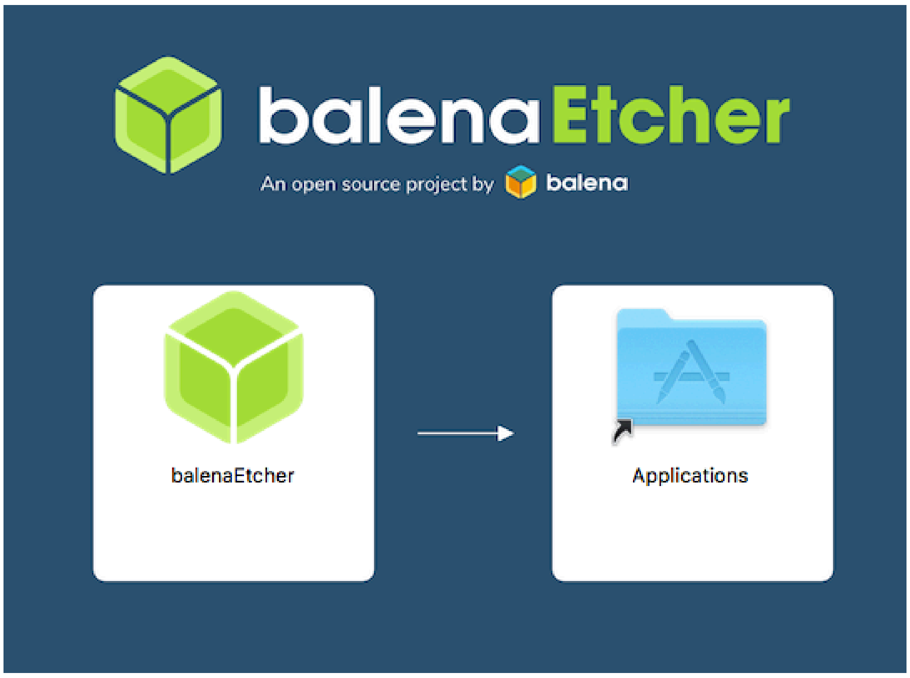
Flash. Flawless.

Flash OS images to SD cards & USB drives, safely and easily.

Validated Flashing
No more writing images on corrupted cards and wondering why your device isn't booting

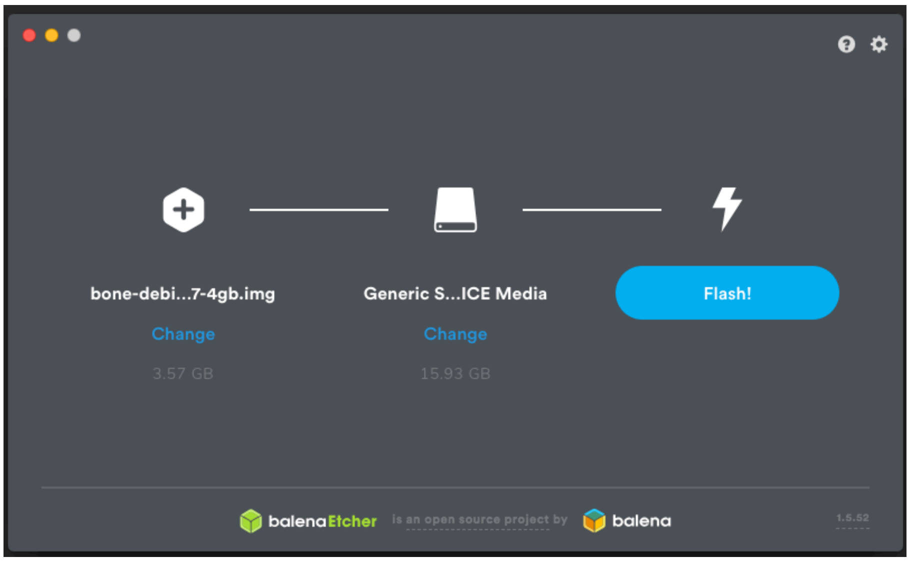
Hard Drive Friendly
Makes drive selection obvious to avoid wiping your entire hard-drive

FEATURES
A better way to burn.



Connect SD card to your computer Use your computer's SD slot or a USB adapter to connect the SD card to your computer.

Write the image to your SD card Use Etcher to write the image to your SD card. Etcher will transparently decompress the image on-the-fly before writing it to the SD card.



Eject the SD card Eject the newly programmed SD card.

Boot your board off of the SD card Insert SD card into your (powered-down) board, hold down the USER/BOOT button and apply power, either by the USB cable or 5V adapter.

If using an original BeagleBone or PocketBeagle, you are done.

Note: If using BeagleBone Black, BeagleBone Blue, BeagleBone AI, BeagleBone AI-64, BeaglePlay or other board with on-board eMMC flash and you desire to write the image to your on-board eMMC, you'll need to follow the instructions at http://elinux.org/Beagleboard:BeagleBoneBlack_Debian#Flashing_eMMC. When the flashing is complete, all 4 USRx LEDs will be steady off and possibly power down the board upon completion. This can take up to 45 minutes. Power-down your board, remove the SD card and apply power again to finish.

Start your Beagle If any step fails, it is recommended to update to the latest software image using the instructions above.

Power and boot Most Beagles can be powered via a USB cable, providing a convenient way to provide both power to your Beagle and connectivity to your computer. Be sure the cable is of good quality and your source can provide enough power.

Alternatively, your Beagle may have a barrel jack which can take power from a wall adapter. Checkout *Power supplies* to get the correct adapter for your Beagle.

> **Danger:** Make sure to use only a 5V center positive adapter for all Beagles except BeagleBone Blue and BeagleBoard-X15 (12V).

If you are using your Beagle with an SD (microSD) card, make sure it is inserted ahead of providing power. Most Beagles include programmed on-board flash and therefore do not require an SD card to be inserted.

You'll see the power (PWR or ON) LED lit steadily. Within a minute or so, you should see the other LEDs blinking in their default configurations. Consult your *Boards* documentation to locate these LEDs.

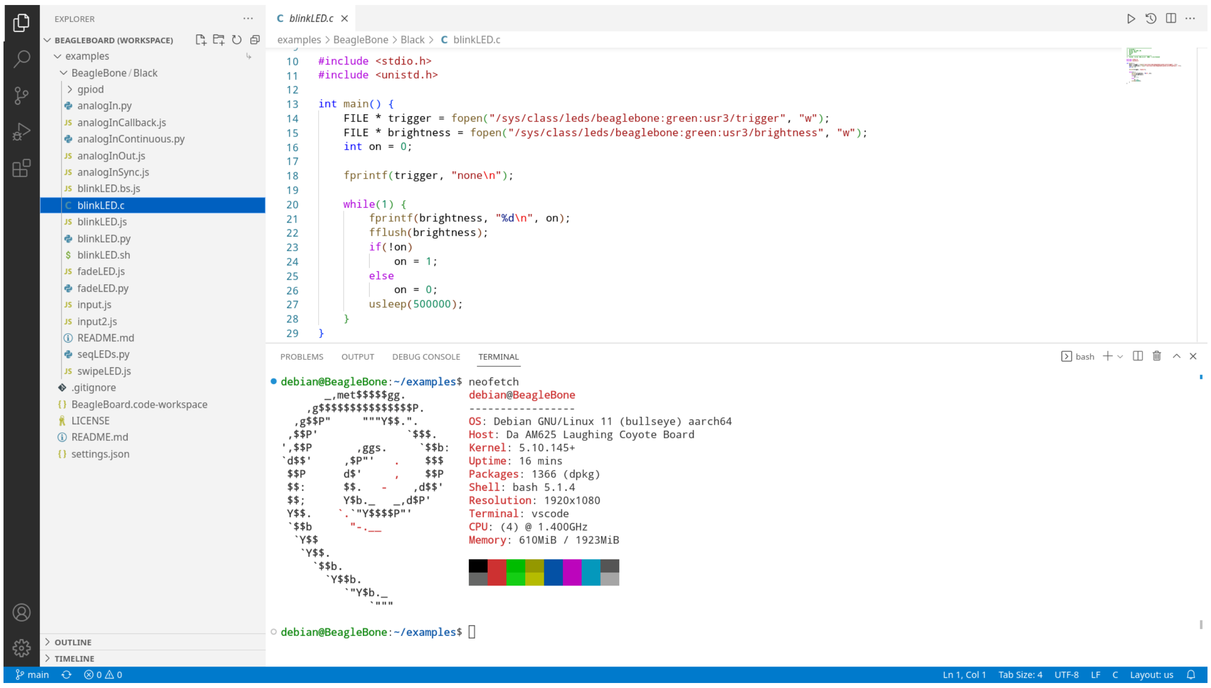
- USR0 is typically configured at boot to blink in a heartbeat pattern.
- USR1 is typically configured at boot to light during SD (microSD) card accesses.
- USR2 is typically configured at boot to light during CPU activity.
- USR3 is typically configured at boot to light during eMMC accesses.
- USR4/WIFI is typically configured at boot to light with WiFi (client) network association (Only on boards with built-in WiFi or M.2).

Enable a network connection If connected via USB, a network adapter should show up on your computer. Your Beagle should be running a DHCP server that will provide your computer with an IP address of either 192.168.7.1 or 192.168.6.1, depending on the type of USB network adapter supported by your computer's operating system. Your Beagle will reserve 192.168.7.2 or 192.168.6.2 for itself.

If your Beagle includes WiFi, an access point called "BeagleBone-XXXX" where "XXXX" varies between boards. The access point password defaults to "BeagleBone". Your Beagle should be running a DHCP server that will provide your computer with an IP address in the 192.168.8.x range and reserve 192.168.8.1 for itself.

If your Beagle is connected to your local area network (LAN) via either Ethernet or WiFi, it will utilize mDNS to broadcast itself to your computer. If your computer supports mDNS, you should see your Beagle as beaglebone.local. Non-BeagleBone boards will utilize alternate names. Multiple BeagleBone boards on the same network will add a suffix such as beaglebone-2.local.

Browse to your Beagle A web server with an Visual Studio Code (IDE) should be running on your Beagle. Point your browser to **http://192.168.7.2:3000** to begin development.



```c
#include <stdio.h>
#include <unistd.h>

int main() {
    FILE * trigger = fopen("/sys/class/leds/beaglebone:green:usr3/trigger", "w");
    FILE * brightness = fopen("/sys/class/leds/beaglebone:green:usr3/brightness", "w");
    int on = 0;

    fprintf(trigger, "none\n");

    while(1) {
        fprintf(brightness, "%d\n", on);
        fflush(brightness);
        if(!on)
            on = 1;
        else
            on = 0;
        usleep(500000);
    }
}
```

```
debian@BeagleBone:~/examples$ neofetch
       _,met$$$$$gg.          debian@BeagleBone
    ,g$$$$$$$$$$$$$$$P.       -----------------
  ,g$$P"     """Y$$.".        OS: Debian GNU/Linux 11 (bullseye) aarch64
 ,$$P'              `$$$.     Host: Da AM625 Laughing Coyote Board
',$$P       ,ggs.     `$$b:   Kernel: 5.10.145+
`d$$'     ,$P"'   .    $$$    Uptime: 16 mins
 $$P      d$'     ,    $$P    Packages: 1366 (dpkg)
 $$:      $$.   -    ,d$$'    Shell: bash 5.1.4
 $$;      Y$b._   _,d$P'      Resolution: 1920x1080
 Y$$.    `.`"Y$$$$P"'         Terminal: vscode
 `$$b      "-.__              CPU: (4) @ 1.400GHz
  `Y$$                        Memory: 610MiB / 1923MiB
   `Y$$.
     `$$b.
       `Y$$b.
          `"Y$b._
              `"""

debian@BeagleBone:~/examples$ 
```

Note: Use either Firefox or Chrome (Internet Explorer will NOT work), browse to the web server running on your board. It will load a presentation showing you the capabilities of the board. Use the arrow keys on your keyboard to navigate the presentation.

The below table summarizes the typical addresses.

Link	Connection type	Operating System(s)
http://192.168.7.2	USB	Windows
http://192.168.6.2	USB	Mac OS X, Linux
http://192.168.8.1	WiFi	all
http://beaglebone.local	all	mDNS enabled
http://beaglebone-2.local	all	mDNS enabled

Troubleshooting Do not use Internet Explorer.

Virtual machines are not recommended when using the direct USB connection. It is recommended you use only network connections to your board if you are using a virtual machine.

When using 'ssh' with the provided image, the username is 'debian' and the password is 'temppwd'.

With the latest images, it should no longer be necessary to install drivers for your operating system to give you network-over-USB access to your Beagle. In case you are running an older image, an older operating system or need additional drivers for serial access to older boards, links to the old drivers are below.

Operating system	USB Driver	Comments
Windows (64-bit)	64-bit installer	If in doubt, try the 64-bit installer first.
Windows (32-bit)	32-bit installer	
Mac OS X	Network Serial	Install both sets of drivers.
Linux	mkudevrules.sh	Driver installation isn't required, but you might find a few udev rules helpful.

Note: For Windows (64-bit):

1. Windows Driver Certification warning may pop up two or three times. Click "Ignore", "Install" or "Run".
2. To check if you're running 32 or 64-bit Windows see this.

3. On systems without the latest service release, you may get an error (0xc000007b). In that case, please perform the following and retry: https://answers.microsoft.com/en-us/windows/forum/all/windows-10-error-code-0xc000007b/02b74e7d-ce19-4ba4-90f0-e16e8d911866
4. You may need to reboot Windows.
5. These drivers have been tested to work up to Windows 10

Additional FTDI USB to serial/JTAG information and drivers are available from https://www.ftdichip.com/Drivers/VCP.htm

Additional USB to virtual Ethernet information and drivers are available from http://www.linux-usb.org/gadget/ and https://joshuawise.com/horndis

Visit https://docs.beagleboard.org/latest/intro/support/index.html for additional debugging tips.

Hardware documentation Be sure to check check the latest hardware documentation for your board at https://docs.beagleboard.org. Detailed design materials for various boards can be found at https://git.beagleboard.org/explore/projects/topics/boards.

Books For a complete list of books on BeagleBone, see 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.

1.1.2 Getting support

BeagleBoard.org products and open hardware designs are supported via the on-line community resources. We are very confident in our community's ability to provide useful answers in a timely manner. If you don't get a productive response within 24 hours, please escalate issues to Jason Kridner (contact info available on the About Page). In case it is needed, Jason will help escalate issues to suppliers, manufacturers or others. Be sure to provide a link to your questions on the community forums as answers will be provided there.

Be sure to ask smart questions that provide the following:

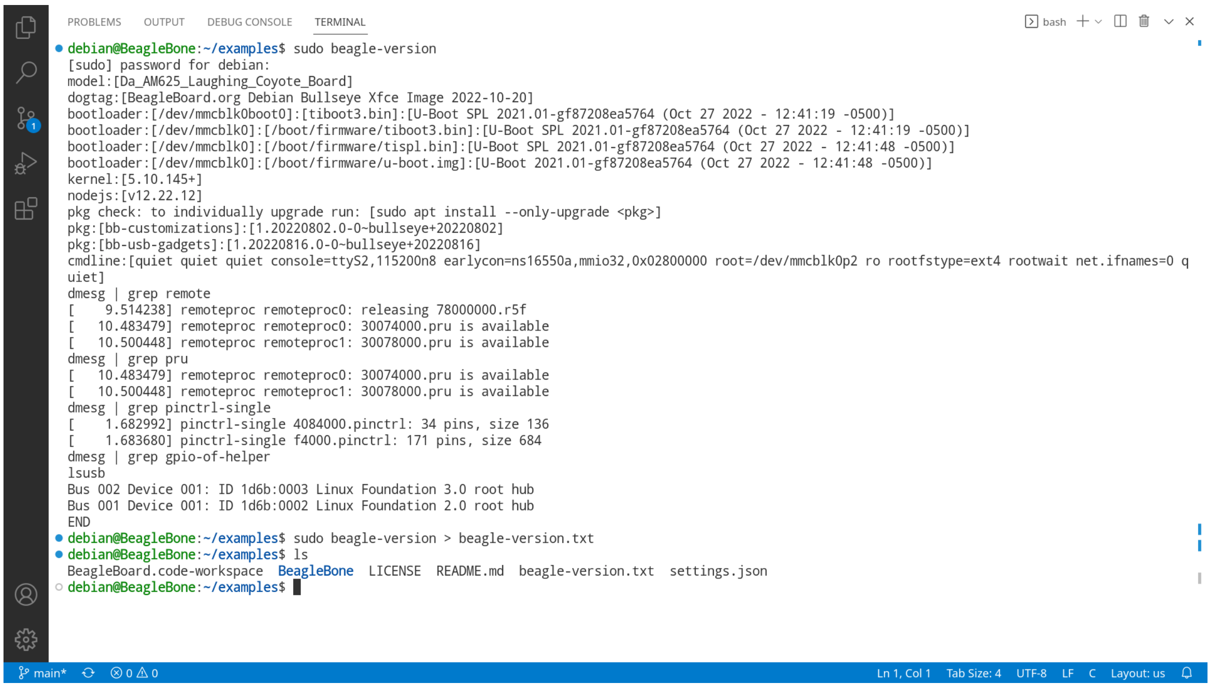
- What are you trying to accomplish?
- What did you find when researching how to accomplish it?
- What are the detailed results of what you tried?
- How did these results differ from what you expected?
- What would you consider to be a success?

Important: Remember that community developers are volunteering their expertise. Respect developers time and expertise and they might be happy to share with you. If you want paid support, there are *Consulting and other resources* options for that.

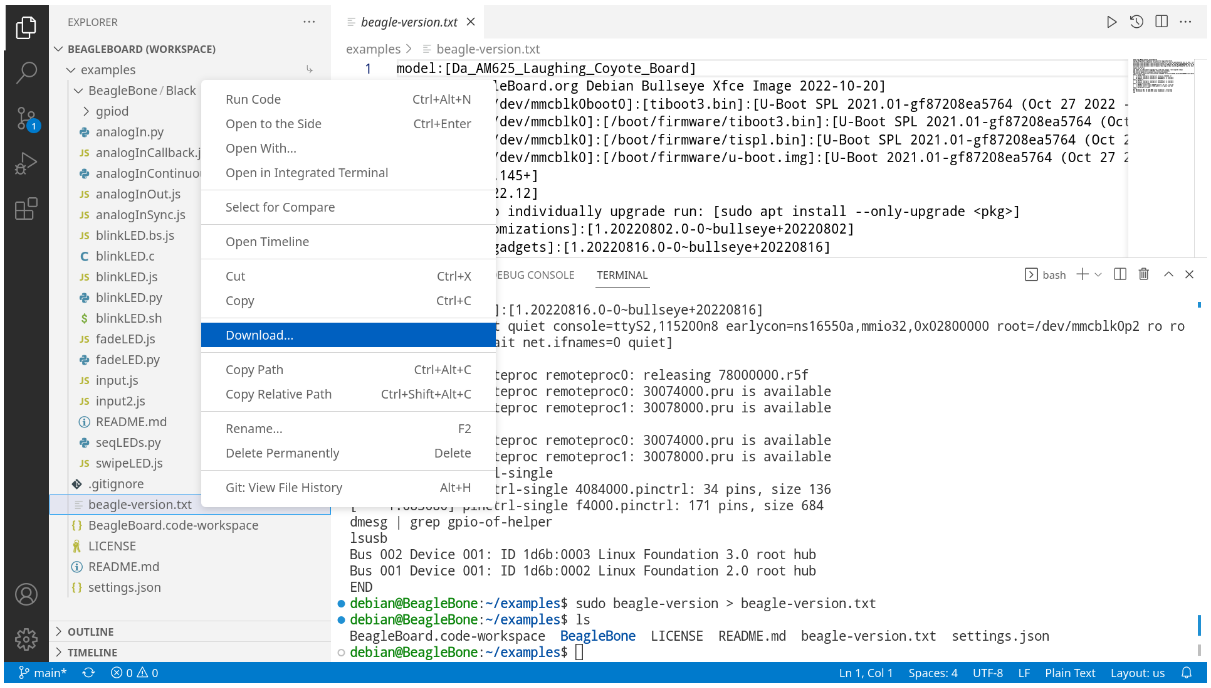
Diagnostic tools

Best to be prepared with good diagnostic information to aide with support.

- Output of *beagle-version* script needed for support requests
- Beagle Tester source



```
debian@BeagleBone:~/examples$ sudo beagle-version
[sudo] password for debian:
model:[Da_AM625_Laughing_Coyote_Board]
dogtag:[BeagleBoard.org Debian Bullseye Xfce Image 2022-10-20]
bootloader:[/dev/mmcblk0boot0]:[tiboot3.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct 27 2022 - 12:41:19 -0500)]
bootloader:[/dev/mmcblk0]:[/boot/firmware/tiboot3.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct 27 2022 - 12:41:19 -0500)]
bootloader:[/dev/mmcblk0]:[/boot/firmware/tispl.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct 27 2022 - 12:41:48 -0500)]
bootloader:[/dev/mmcblk0]:[/boot/firmware/u-boot.img]:[U-Boot 2021.01-gf87208ea5764 (Oct 27 2022 - 12:41:48 -0500)]
kernel:[5.10.145+]
nodejs:[v12.22.12]
pkg check: to individually upgrade run: [sudo apt install --only-upgrade <pkg>]
pkg:[bb-customizations]:[1.20220802.0-0~bullseye+20220802]
pkg:[bb-usb-gadgets]:[1.20220816.0-0~bullseye+20220816]
cmdline:[quiet quiet quiet console=ttyS2,115200n8 earlycon=ns16550a,mmio32,0x02800000 root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait net.ifnames=0 q
uiet]
dmesg | grep remote
[    9.514238] remoteproc remoteproc0: releasing 78000000.r5f
[   10.483479] remoteproc remoteproc0: 30074000.pru is available
[   10.500448] remoteproc remoteproc1: 30078000.pru is available
dmesg | grep pru
[   10.483479] remoteproc remoteproc0: 30074000.pru is available
[   10.500448] remoteproc remoteproc1: 30078000.pru is available
dmesg | grep pinctrl-single
[    1.682992] pinctrl-single 4084000.pinctrl: 34 pins, size 136
[    1.683680] pinctrl-single f4000.pinctrl: 171 pins, size 684
dmesg | grep gpio-of-helper
lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
END
debian@BeagleBone:~/examples$ sudo beagle-version > beagle-version.txt
debian@BeagleBone:~/examples$ ls
BeagleBoard.code-workspace  BeagleBone  LICENSE  README.md  beagle-version.txt  settings.json
debian@BeagleBone:~/examples$ 
```



```
model:[Da_AM625_Laughing_Coyote_Board]
leBoard.org Debian Bullseye Xfce Image 2022-10-20]
/dev/mmcblk0boot0]:[tiboot3.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct 27 2022 -
/dev/mmcblk0]:[/boot/firmware/tiboot3.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct
/dev/mmcblk0]:[/boot/firmware/tispl.bin]:[U-Boot SPL 2021.01-gf87208ea5764 (Oct 2
/dev/mmcblk0]:[/boot/firmware/u-boot.img]:[U-Boot 2021.01-gf87208ea5764 (Oct 27 2
.145+]
2.12]
 individually upgrade run: [sudo apt install --only-upgrade <pkg>]
omizations]:[1.20220802.0-0~bullseye+20220802]
gadgets]:[1.20220816.0-0~bullseye+20220816]
```

```
]:[1.20220816.0-0~bullseye+20220816]
t quiet console=ttyS2,115200n8 earlycon=ns16550a,mmio32,0x02800000 root=/dev/mmcblk0p2 ro ro
ait net.ifnames=0 quiet]

teproc remoteproc0: releasing 78000000.r5f
teproc remoteproc0: 30074000.pru is available
teproc remoteproc1: 30078000.pru is available

teproc remoteproc0: 30074000.pru is available
teproc remoteproc1: 30078000.pru is available
l-single
trl-single 4084000.pinctrl: 34 pins, size 136
trl-single f4000.pinctrl: 171 pins, size 684
dmesg | grep gpio-of-helper
lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
END
debian@BeagleBone:~/examples$ sudo beagle-version > beagle-version.txt
debian@BeagleBone:~/examples$ ls
BeagleBoard.code-workspace  BeagleBone  LICENSE  README.md  beagle-version.txt  settings.json
debian@BeagleBone:~/examples$ 
```

Tip: For debugging purposes you can either share the `beagle-version.txt` file you just downloaded using the steps shown in pictures above Or you can just paste the terminal output of `sudo beagle-version` to https://pastebin.com/ and send us the link.

Community resources

Please execute the board diagnostics, review the hardware documentation, and consult the mailing list and IRC channel for support. BeagleBoard.org is a "community" project with free support only given to those who are willing to discussing their issues openly for the benefit of the entire community.

- Frequently Asked Questions
- Mailing List
- Live Chat

Consulting and other resources

Need timely response or contract resources because you are building a product?

- Resources

Repairs

Repairs and replacements only provided on unmodified boards purchased via an authorized distributor within the first 90 days. All repaired board will have their flash reset to factory contents. For repairs and replacements, please contact `support` at BeagleBoard.org using the RMA form:

- RMA request

1.1.3 Understanding Your Beagle

- *Beagle 101*
- Hardware
- Software
- *Books*
  - *PRU Cookbook*
  - *BeagleBone Cookbook*
  - Exploring BeagleBone
  - Bad to the Bone

1.1.4 Working with Cape Add-on Boards

- *Capes*
- *BeagleBone cape interface spec*
- *Accessories*

1.2 Beagle 101

Note: This page is under construction. Most of the information here is drastically out of date.

This is a collection of articles to aide in quickly understanding how to make use of Beagles running Linux. Most of the useful information has moved to *BeagleBone Cookbook*, but some articles are being built here from a different perspective.

Articles under construction or to be imported and updated:

- *QWIIC, STEMMA and Grove Add-ons in Linux*
- https://beagleboard.github.io/bone101/Support/bone101/

1.2.1 QWIIC, STEMMA and Grove Add-ons in Linux

Note: This article is under construction.

I'm creating a place for me to start taking notes on how to load drivers for I2C devices (mostly), but also other Grove add-ons.

For simplicity sake, I'll use these definitions

- **add-on**: the QWIIC, STEMMA (QT) or Grove add-on separate from your Linux computer
- **device**: the "smart" IC on the add-on to which we will interface from your Linux computer
- **board**: the Linux single board computer with the embedded interface controller you are using
- **module**: a kernel module that might contain the driver

Using I2C with Linux drivers

Linux has a ton of drivers for I2C devices. We just need a few parameters to load them.

Using a Linux I2C kernel driver module can be super simple, like in the below example for monitoring a digital light sensor.

```
cd /dev/bone/i2c/2
echo tsl2561 0x29 > new_device
watch -n0 cat "2-0029/iio:device0/in_illuminance0_input"
```

Once you issue this, your screen continuously refresh with luminance values from the add-on sensor.

In the above example, */dev/bone/i2c/2* comes from which I2C controller we are using on the board and there are specific pins on the board where you can access it. On BeagleBone boards, there is often a symbolic link to the controller based upon the cape expansion header pins being used. See *I2C* for the cape expansion header pin assignments.

tsl2561 is the name of the driver we want to load and *0x29* is the address of the device on the I2C bus. If you want to know about I2C device addresses, the Sparkfun I2C tutorial isn't a bad place to start. The *new_device* virtual file is documented in the Linux kernel documentation on instantiating I2C devices.

On the last line, watch is a program that will repeatedly run the command that follows. The *-n0* sets the refresh rate. The program cat will share the contents of the file *2-0029/iio:device0/in_illuminance0_input*.

2-0029/iio:device0/in_illuminance0_input is not a file on a disk, but output directly from the driver. The leading 2 in *2-0029* represents the I2C controller index. The *0029* represents the device I2C address. Most small sensor and actuator drivers will show up as Industrial I/O (IIO) devices. New IIO devices get incrementing indexes. In this case, *iio:device0* is the first IIO device driver loaded. Finally, *in_illuminance0_input* comes from the SYSFS application binary interface for this type of device, a light sensor. The Linux kernel ABI documentation for sysfs-bus-iio provides the definition of available data often provided by light sensor drivers.

```
What:           /sys/.../iio:deviceX/in_illuminance_input
What:           /sys/.../iio:deviceX/in_illuminance_raw
What:           /sys/.../iio:deviceX/in_illuminanceY_input
What:           /sys/.../iio:deviceX/in_illuminanceY_raw
What:           /sys/.../iio:deviceX/in_illuminanceY_mean_raw
What:           /sys/.../iio:deviceX/in_illuminance_ir_raw
```

(continues on next page)

(continued from previous page)

```
What:          /sys/.../iio:deviceX/in_illuminance_clear_raw
KernelVersion:         3.4
Contact:       linux-iio@vger.kernel.org
Description:
               Illuminance measurement, units after application of scale
               and offset are lux.
```

Read further to discover how to find these bits of magic text used above.

The generic steps are fairly simple:

1. *Identify driver name and address*
2. *Ensure driver is enabled in kernel build*
3. *Identify I2C signals on board and controller in Linux*
4. *Ensure pinmux set to I2C*
5. *Ensure add-on connection is good*
6. *Issue Linux command to load driver*
7. *Identify and utilize interface provided by driver*

Driver name One resource that is very helpful is the list that Vaishnav put together for supporting Mikroelektronika Click add-ons. This list of Click add-ons with driver information can help a lot with matching a device to the driver name, device address, and kernel configuration setting.

Note: Documentation for your particular add-on might indicate a different device address than is configured on Click add-ons.

I'm not aware of a trivial way of discovering the mapping that Vaishnav created outside of looking at the kernel sources. As an example, let's look at the Grove Digital Light Sensor add-on which is documented to utilize a TSL2561.

Searching through the kernel sources, we can find the driver code at *drivers/iio/light/tsl2563.c*. There is a list of driver names in a i2c_device_id table:

```
static const struct i2c_device_id tsl2563_id[] = {
      { "tsl2560", 0 },
      { "tsl2561", 1 },
      { "tsl2562", 2 },
      { "tsl2563", 3 },
      {}
};
```

Important: Don't miss that the driver, *tsl2561* , is actually part of a a superset driver, *tsl2563* . This can make things a bit trickier to find, so you have to look within the text of the driver source, not just the filenames.

Kernel configuration

I2C signals and controller

Pinmuxing

Wiring

Load driver

Interface

Finding I2C add-on modules

Note: There are some great resources out there:

- Adafruit list of I2C devices
- Sparkfun list of QWIIC devices
- Adafruit STEMMA QT introduction

Pitfalls Not all I2C devices with drivers in the Linux kernel can be loaded this way. The most common reason is that the device driver expects an interrupt signal or other GPIO along with the I2C communication. In these cases, a device tree overlay or driver modification may be necessary.

1.3 Contribution

Note: This section is under developmement right now.

Important: First off, thanks for taking the time to think about contributing!

Note: For donations, see BeagleBoard.org - Donate.

The BeagleBoard.org Foundation maintains source for many open source projects.

Example projects suitable for first contributions:

- BeagleBoard project documentation
- Debian image bug repository
- Debian image builder

These guidelines are mostly suggestions, not hard-set rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

1.3.1 Code of Conduct

This project and everyone participating are governed by the same code of conduct.

Note: Check out https://forum.beagleboard.org/faq as a starting place for our code of conduct.

By participating, you are expected to uphold this code. Please report unacceptable behavior to contact one of our administrators or moderators on https://forum.beagleboard.org/about.

1.3.2 Frequently Asked Questions

Please refer to the technical and contribution frequently asked questions pages before posting any of your own questions. Please feel encouraged to ask follow-up questions if any of the answers are not clear enough.

- Frequently asked questions contribution category on the BeagleBoard.org Forum

1.3.3 What should I know before I get started?

The more you know about Linux and contributing to upstream projects, the better, but this knowledge isn't strictly required. Simply reading about contributing to Linux and upstream projects can help build your vocabulary in a meaningful way to help out. Learn about the skills required for Linux contributions in the *Upstream Kernel Contributions* section.

The most useful thing to know is how to ask smart questions. Read about this in the *Getting support* section. If you ask smart questions on the issue trackers and forum, you'll be doing a lot to help us improve the designs and documentation.

Upstream Kernel Contributions

Note: For detailed information on Kernel Developmement checkout the official kernel.org kernel docs.

For a person or company who wishes to submit a change to the Linux kernel, the process can sometimes be daunting if you're not familiar with "the system." This text is a collection of suggestions which can help you get started and greatly increase the chances of your change being accepted.

Note: This version is an unofficial draft and is subject to change.

Pre-requisites The following are the skills that are needed before you actually start to contribute to the linux kernel:

- *More Git!*
- *C-Programming*
- *Cross-arch Development*
- *Basics of embedded buses (I2C, UART, SPI, etc.)*
- *Device Drivers in Embedded Systems*
- *Device Trees*

For more guidance, check out the *Additional Resources*.

More Git! It is highly recommended that you go through *Git Usage* before starting to read and follow these guidelines. You will need to have a proper git setup on your computer in order to effectively follow these steps.

Creating your first patch When you first enter the world of Linux Kernel development from a background in contributing over gitlab or github, the terminologies slightly change.

Your Pull Requests (PRs) now become Patches or Patch Series. You no longer just go to some website and click on a "Create Pull Request" button. Whatever code/changes you want to add will have to be sent as patches via emails.

As an example, let's consider a commit to add the git section to these docs. I stage these changes first using `git add -p`.

```
diff --git a/contribution/contribute.rst b/contribution/contribute.rst
index def100b..0af08c5 100644
--- a/contribution/contribute.rst
+++ b/contribution/contribute.rst
```

Then, commit the above changes.

Note: Don't forget to make your commit message descriptive of the feature you are adding or the work that you have done in that commit. The commit has to be self explanatory in itself. Link any references if you have used and paste any logs to prove your code works or if there is a fix.

```
git commit -vs

[linux-contrib 3bc0821] contribute.rst: Add git section
 1 file changed, 27 insertions(+), 1 deletion(-)
```

Now, let's say we want to send this new feature to upstream kernel. You then have to create a patch file using the following command:

```
git format-patch -1 HEAD

0001-contribute.rst-Add-git-section.patch
```

This will generate one file that is generally referred to as the patch file. This is what you will now be sending upstream in order to get your patch merged. But wait, there are a few more things we need to setup for sending a patch via e-mail. That is, of course your email!

For configuring your email ID for sending patches refer to this excellent stackoverflow thread, configure git-send-email.

Finally, after you have configured you email properly, you can send out a patch using:

```
git send-email 0001-contribute.rst-Add-git-section.patch
```

replacing of course the above patchfile name with whatever was your own patch. This command will then ask you `To whom should the emails be sent (if anyone)?` Here, you have to write the email address of the list you want to send out the patch to.

`git send-email` also has command line options like `--to` and `--cc` that you can also use to add more email addresses of whoever you want to keep in CC. Generally it is a good idea to keep yourself in CC.

C-Programming It is highly recommended that you have proficiency in C-Programming, because well the kernel is mostly written in C! For starters, you can go through Dennis Ritchie's C Programming book to understand the language and also solve the exercises given there for getting hands on.

Cross-arch Development While working with the kernel, you'll most likely not be compiling it on the machine that you intend to actually boot it on. For example if you are compiling the Kernel for BeageBone Black it's probably not ideal for you to actually clone the entire kernel on BeagleBone Black and then compile it there. What you'd do instead is pick a much powerful machine like a Desktop PC or laptop and then use cross arch compilers like the arm-gcc for instance to compile the kernel for your target device.

Basics of embedded buses (I2C, UART, SPI, etc.) In the world of embedded, you often need to communicate with peripherals over very low level protocols. To name a few, I2C, UART, SPI, etc. are all serial protocols used to communicate with a variety of devices and peripherals.

It's recommended to understand at least the basics of each of the protocol so you know what's actually going on when you write for instance an I2C or SPI driver to communicate with let's say a sensor.

Device Drivers in Embedded Systems I used the term "Drivers" in the above section, but what does it really mean?

Why "device" drivers?

TODO

Why do we need drivers?

TODO

What do drivers look like?

TODO

Device Trees We just learned about drivers, and it's time that once you have written a driver in the kernel, you obviously want it to work! So how do we really tell the kernel which drivers to load? How do we, at boot time, instruct which devices are present on the board you are booting on?

The kernel does not contain the description of the hardware, it is located in a separate binary: the device tree blob.

What is a Device Tree?

A device tree is used to describe system hardware. A boot program loads a device tree into a client program's memory and passes a pointer to the device tree to the client.

A device tree is a tree data structure with nodes that describe the physical devices in a system.

Additional Resources

1. Device Trees for Dummies PDF
2. What are Device Drivers
3. Submitting your patches upstream

1.3.4 How can I contribute?

The most obvious way to contribute is using the git.beagleboard.org Gitlab server to report bugs, suggest enhancements and providing merge requests, also called pull requests, the provide fixes to software, hardware designs and documentation.

Reporting bugs

Suggesting enhancements

Submitting merge requests

1.3.5 Style and usage guidelines

- *Git Usage*
- Git commit messages
- *Documentation Style Guide*

Git Usage

Note: For detailed information on Git and Gitlab checkout the official Git and GitLab help page. Also, for good GitLab workflow you can checkout the Introduction to GitLab Flow (FREE) page.

These are (draft) general guidelines taken from BioPython project to be used for BeagleBoard development using git. We're still working on the finer details.

This document is meant as an outline of the way BeagleBoard projects are developed. It should include all essential technical information as well as typical procedures and usage scenarios. It should be helpful for core developers, potential code contributors, testers and everybody interested in BeagleBoard code.

Note: This version is an unofficial draft and is subject to change.

Relevance This page is about actually using git for tracking changes.

If you have found a problem with any BeagleBoard project, and think you know how to fix it, then we suggest following the simple route of filing a bug and describe your fix. Ideally, you would upload a patch file showing the differences between the latest version of BeagleBoard project (from our repository) and your modified version. Working with the command line tools *diff* and *patch* is a very useful skill to have, and is almost a precursor to working with a version control system.

Technicalities This section describes technical introduction into git usage including required software and integration with GitLab. If you want to start contributing to BeagleBoard, you definitely need to install git and learn how to obtain a branch of the BeagleBoard project you want to contribute. If you want to share your changes easily with others, you should also sign up for a BeagleBoard GitLab account and read the corresponding section of the manual. Finally, if you are engaged in one of the collaborations on experimental BeagleBoard modules, you should look also into code review and branch merging.

Installing Git You will need to install Git on your computer. Git is available for all major operating systems. Please use the appropriate installation method as described below.

Linux Git is now packaged in all major Linux distributions, you should find it in your package manager.

Ubuntu/Debian You can install Git from the *git-core* package. e.g.,

```
sudo apt-get install git-core
```

You'll probably also want to install the following packages: *gitk*, *git-gui*, and *git-doc*

Redhat/Fedora/Mandriva git is also packaged in rpm-based linux distributions.

```
dnf install gitk
```

should do the trick for you in any recent fedora/mandriva or derivatives

Mac OS X Download the *.dmg* disk image from http://code.google.com/p/git-osx-installer/

Windows Download the official installers from Windows installers

Testing your git installation If your installation succeeded, you should be able to run

```
$ git --help
```

in a console window to obtain information on git usage. If this fails, you should refer to git documentation for troubleshooting.

Creating a GitLab account (Optional) Once you have Git installed on your machine, you can obtain the code and start developing. Since the code is hosted at GitLab, however, you may wish to take advantage of the site's offered features by signing up for a GitLab account. While a GitLab account is completely optional and not required for obtaining the BeagleBoard code or participating in development, a GitLab account will enable all other BeagleBoard developers to track (and review) your changes to the code base, and will help you track other developers' contributions. This fosters a social, collaborative environment for the BeagleBoard community.

If you don't already have a GitLab account, you can create one here. Once you have created your account, upload an SSH public key by clicking on *SSH and GPG keys <https://git.beagleboard.org/-/profile/keys>* after logging in. For more information on generating and uploading an SSH public key, see this GitLab guide.

Working with the source code In order to start working with the BeagleBoard source code, you need to obtain a local clone of our git repository. In git, this means you will in fact obtain a complete clone of our git repository along with the full version history. Thanks to compression, this is not much bigger than a single copy of the tree, but you need to accept a small overhead in terms of disk space.

There are, roughly speaking, two ways of getting the source code tree onto your machine: by simply "cloning" the repository, or by "forking" the repository on GitLab. They're not that different, in fact both will result in a directory on your machine containing a full copy of the repository. However, if you have a GitLab account, you can make your repository a public branch of the project. If you do so, other people will be able to easily review your code, make their own branches from it or merge it back to the trunk.

Using branches on GitLab is the preferred way to work on new features for BeagleBoard, so it's useful to learn it and use it even if you think your changes are not for immediate inclusion into the main trunk of BeagleBoard. But even if you decide not to use GitLab, you can always change this later (using the .git/config file in your branch.) For simplicity, we describe these two possibilities separately.

Cloning BeagleBoard directly Getting a copy of the repository (called "cloning" in Git terminology) without GitLab account is very simple:

```
git clone https://git.beagleboard.org/docs/docs.beagleboard.io.git
```

This command creates a local copy of the entire BeagleBoard repository on your machine (your own personal copy of the official repository with its complete history). You can now make local changes and commit them to this local copy (although we advise you to use named branches for this, and keep the main branch in sync with the official BeagleBoard code).

If you want other people to see your changes, however, you must publish your repository to a public server yourself (e.g. on GitLab).

Forking BeagleBoard with your GitLab account If you are logged in to GitLab, you can go to the BeagleBoard Docs repository page:

https://git.beagleboard.org/docs/docs.beagleboard.io/-/tree/main

and click on a button named 'Fork'. This will create a fork (basically a copy) of the official BeagleBoard repository, publicly viewable on GitLab, but listed under your personal account. It should be visible under a URL that looks like this:

https://git.beagleboard.org/yourusername/docs.beagleboard.io/

Since your new BeagleBoard repository is publicly visible, it's considered good practice to change the description and homepage fields to something meaningful (i.e. different from the ones copied from the official repository).

If you haven't done so already, setup an SSH key and upload it to gitlab for authentication.

Now, assuming that you have git installed on your computer, execute the following commands locally on your machine. This "url" is given on the GitLab page for your repository (if you are logged in):

```
git clone https://git.beagleboard.org/yourusername/docs.beagleboard.io.git
```

Where *yourusername*, not surprisingly, stands for your GitLab username. You have just created a local copy of the BeagleBoard Docs repository on your machine.

You may want to also link your branch with the official distribution (see below on how to keep your copy in sync):

```
git remote add upstream https://git.beagleboard.org/docs/docs.beagleboard.io/
```

If you haven't already done so, tell git your name and the email address you are using on GitLab (so that your commits get matched up to your GitLab account). For example,

```
git config --global user.name "David Jones" config --global user.email "d.
↪jones@example.com"
```

Making changes locally Now you can make changes to your local repository - you can do this offline, and you can commit your changes as often as you like. In fact, you should commit as often as possible, because smaller commits are much better to manage and document.

First of all, create a new branch to make some changes in, and switch to it:

```
git branch demo-branch checkout demo-branch
```

To check which branch you are on, use:

```
git branch
```

Let us assume you've made changes to the file beaglebone-black/ch01.rst Try this:

```
git status
```

So commit this change you first need to explicitly add this file to your change-set:

```
git add beaglebone-black/ch01.rst
```

and now you commit:

```
git commit -m "added updates X in BeagleBone Black ch01"
```

Your commits in Git are local, i.e. they affect only your working branch on your computer, and not the whole BeagleBoard tree or even your fork on GitLab. You don't need an internet connection to commit, so you can do it very often.

Pushing changes to GitLab If you are using GitLab, and you are working on a clone of your own branch, you can very easily make your changes available for others.

Once you think your changes are stable and should be reviewed by others, you can push your changes back to the GitLab server:

```
git push origin demo-branch
```

This will not work if you have cloned directly from the official BeagleBoard branch, since only the core developers will have write access to the main repository.

Merging upstream changes We recommend that you don't actually make any changes to the **main** branch in your local repository (or your fork onGitLab). Instead, use named branches to do any of your own work. The advantage of this approach it is the trivial to pull the upstream **main** (i.e. the official BeagleBoard branch) to your repository.

Assuming you have issued this command (you only need to do this once):

```
git remote add upstream https://git.beagleboard.org/docs/docs.beagleboard.io/
```

Then all you need to do is:

```
git checkout main pull upstream main
```

Provided you never commit any change to your local **main** branch, this should always be a simple *fast forward* merge without any conflicts. You can then deal with merging the upstream changes from your local main branch into your local branches (and you can do that offline).

If you have your repository hosted online (e.g. at GitLab), then push the updated main branch there:

```
git push origin main
```

Submitting changes for inclusion in BeagleBoard If you think you changes are worth including in the main BeagleBoard distribution, then file an (enhancement) bug on our bug tracker, and include a link to your updated branch (i.e. your branch on GitLab, or another public Git server). You could also attach a patch to the bug. If the changes are accepted, one of the BeagleBoard developers will have to check this code into our main repository.

On GitLab itself, you can inform keepers of the main branch of your changes by sending a 'pull request' from the main page of your branch. Once the file has been committed to the main branch, you may want to delete your now redundant bug fix branch on GitLab.

If other things have happened since you began your work, it may require merging when applied to the official repository's main branch. In this case we might ask you to help by rebasing your work:

```
git fetch upstream checkout demo-branch

git rebase upstream/main
```

Hopefully the only changes between your branch and the official repository's main branch are trivial and git will handle everything automatically. If not, you would have to deal with the clashes manually. If this works, you can update the pull request by replacing the existing (pre-rebase) branch:

```
git push origin demo-branch --force
```

If however the rebase does not go smoothly, give up with the following command (and hopefully the BeagleBoard developers can sort out the rebase or merge for you):

```
git rebase --abort
```

Evaluating changes Since git is a fully distributed version control system, anyone can integrate changes from other people, assuming that they are using branches derived from a common root. This is especially useful for people working on new features who want to accept contributions from other people.

This section is going to be of particular interest for the BeagleBoard core developers, or anyone accepting changes on a branch.

For example, suppose Jason has some interesting changes on his public repository:

https://git.beagleboard.org/jkridner/docs.beagleboard.io

You must tell git about this by creating a reference to this remote repository:

```
git remote add jkridner https://git.beagleboard.org/jkridner/BeagleBoard.git
```

Now we can fetch *all* of Jason's public repository with one line:

```
git fetch jkridner
```

Now we can run a diff between any of our own branches and any of Jason's branches. You can list your own branches with:

```
git branch
```

Remember the asterisk shows which branch is currently checked out.

To list the remote branches you have setup:

```
git branch -r
```

For example, to show the difference between your **main** branch and Jason's **main** branch:

```
git diff main jkridner/main
```

If you are both keeping your **main** branch in sync with the upstream BeagleBoard repository, then his **main** branch won't be very interesting. Instead, try:

```
git diff main jkridner/awesomebranch
```

You might now want to merge in (some) of Jason's changes to a new branch on your local repository. To make a copy of the branch (e.g. awesomebranch) in your local repository, type:

```
git checkout --track jkridner/awesomebranch
```

If Jason is adding more commits to his remote branch and you want to update your local copy, just do:

```
git checkout awesomebranch  # if you are not already in branch awesomebranch␣
↪pull
```

If you later want to remove the reference to this particular branch:

```
git branch -r -d jkridner/awesomebranch
Deleted remote branch jkridner/awesomebranch (#######)
```

Or, to delete the references to all of Jason's branches:

```
git remote rm jkridner

git branch -r
    upstream/main
    origin/HEAD
    origin/main
```

Alternatively, from within GitLab you can use the fork-queue to cherry pick commits from other people's forked branches. While this defaults to applying the changes to your current branch, you would typically do this using a new integration branch, then fetch it to your local machine to test everything, before merging it to your main branch.

Committing changes to main branch This section is intended for BeagleBoard developers, who are allowed to commit changes to the BeagleBoard main "official" branch. It describes the typical activities, such as merging contributed code changes both from git branches and patch files.

Prerequisites Currently, the main BeagleBoard branch is hosted on GitLab. In order to make changes to the main branch you need a GitLab account and you need to be added as a collaborator/Maintainer to the BeagleBoard account. This needs to be done only once. If you have a GitLab account, but you are not yet a collaborator/Maintainer and you think you should be ask Jason to be added (this is meant for regular contributors, so in case you have only a single change to make, please consider submitting your changes through one of developers).

Once you are a collaborator/Maintainer, you can pull BeagleBoard official branch using the private url. If you want to make a new repository (linked to the main branch), you can just clone it:

```
git clone https://git.beagleboard.org/lorforlinux/docs.beagleboard.io.git
```

It creates a new directory "BeagleBoard" with a local copy of the official branch. It also sets the "origin" to the GitLab copy This is the recommended way (at least for the beginning) as it minimizes the risk of accidentally pushing changes to the official GitLab branch.

Alternatively, if you already have a working git repo (containing your branch and your own changes), you can add a link to the official branch with the git "remote command"… but we'll not cover that here.

In the following sections, we assume you have followed the recommended scenario and you have the following entries in your .git/config file:

```
[remote "origin"]
    url = https://git.beagleboard.org/lorforlinux/docs.beagleboard.io.git

[branch "main"]
    remote = origin
```

Committing a patch If you are committing from a patch, it's also quite easy. First make sure you are up to date with official branch:

```
git checkout main pull origin
```

Then do your changes, i.e. apply the patch:

```
patch -r someones_cool_feature.diff
```

If you see that there were some files added to the tree, please add them to git:

```
git add beaglebone-black/some_new_file
```

Then make a commit (after adding files):

```
git commit -a -m "committed a patch from a kind contributor adding feature X"
```

After your changes are committed, you can push toGitLab:

```
git push origin
```

Tagging the official branch If you want to put tag on the current BeagleBoard official branch (this is usually done to mark a new release), you need to follow these steps:

First make sure you are up to date with official branch:

```
git checkout main pull origin
```

Then add the actual tag:

```
git tag new_release
```

And push it to GitLab:

```
git push --tags origin main
```

Additional Resources

There are a lot of different nice guides to using Git on the web:

- Understanding Git Conceptually
- git ready: git tips
- https://web.archive.org/web/20121115132047/http://cheat.errtheblog.com/s/git
- https://docs.scipy.org/doc/numpy-1.15.1/dev/gitwash/development_workflow.html Numpy is also evaluating git
- https://github.github.com/training-kit/downloads/github-git-cheat-sheet
- https://skills.github.com/
- Pro Git

Documentation Style Guide

Note: This is currently a work-in-progress placeholder for some notes on how to style the BeagleBoard Documentation Project.

See the Zephyr Project Documentation Guidelines as a starting point.

ReStructuredText Cheat Sheet

BeagleBoard.org docs site uses ReStructuredText (rst) which is a file format[1] for textual data used primarily in the Python programming language community for technical documentation. It is part of the Docutils project of the Python Doc-SIG, aimed at creating a set of tools for Python similar to Javadoc for Java or Plain Old Documentation for Perl. If you are new with rst you may go through this rst cheat sheet[2][3][4] chapter to gain enough skills to edit and update any page on the BeagleBoard.org docs site. some things you should keep in mind while working with rst,

1. like Python, RST syntax is sensitive to indentation !
2. RST requires blank lines between paragraphs

Text formatting

With asterisk you can format the text as italic & bold,

1. Single asterisk (`*`) like `*emphasis*` gives you *italic text*
2. Double asterisk (`**`) like `**strong emphasis**` gives you **bold text**

With backquote character (‘) you can format the text as link & inline literal.

1. See *Links* section on how single backquote can be used to create a link like this.
2. With double back quotes before and after text you can easily create `inline lierals`.

Note: backquote can be found below escape key on most keyboards.

[1] reStructuredText wiki page
[2] Sphinx and RST syntax guide (0.9.3)
[3] Quick reStructuredText (sourceforge)
[4] A two-page cheatsheet for restructured text

Headings For each document we divide sections with headings and in ReStructuredText we can use matching overline and underline to indicate a heading.

1. Document heading (H1) use #.
2. First heading (H2) use *.
3. Second heading (H3) use =.
4. Third heading (H4) use -.
5. Fourth heading (H5) use ~.

Note: You can include only one (H1) # in a single documentation page.

Make sure the length of your heading symbol is at least (or more) the at least of the heading text, for example:

```
incorrect H1
##### ①

correct H1
############ ②
```

① Length of heading symbol # is smaller than the content above.

② Shows the correct way of setting the document title (H1) with #.

Code For adding a code snippet you can use tab indentation to start. For more refined code snippet display we have the `code-block` and `literalinclude` directives as shown below.

Indentation This the simplest way of adding code snippet in ReStructuredText.

```
This is python code:: ①
    ②
    import numpy as np ③
    import math
```

① Provide title of your code snippet and add `::` after the text.

② Empty line after the title is required for this to work.

③ Start adding your code.

Output This is python code:

```
import numpy as np
import math
```

Code block Simple indentation only supports python program highlighting but, with code block you can specify which language is your code written in. `code-block` also provides better readability and line numbers support you can useas shown below.

```
.. code-block:: python ①
    :linenos: ②

    import numpy as np ③
    import math
```

① Start with adding `.. code-block::` and then add language of code like python, bash, javascript, etc.

② Optionally, you can enable line numbers for your code.

③ Start adding your code.

Output

```
import numpy as np
import math
```

Literal include To include the entire code or a code snippet from a program file you can use this directive.

```
.. literalinclude:: filename.cpp ①
    :caption: ②
    :linenos: ③
    :language: C++ ④
    :lines: 2, 4-7 ⑤
    :start-after: 4 ⑥
    :end-before: 7 ⑦
    :lineno-start: 113 ⑧
```

① Provide the code file destination.

② Provide caption for the code.

③ Enable line numbers.

④ Set programming language.

⑤ Cherry pick some lines from a big program file.

⑥ Display the code snippet from a specific line number.

⑦ Set a specific line number as end of code snippet.

⑧ Instead of starting line number from 1 start it with some other number. It's useful when you use :lines:, :start-after:, and :end-before:.

Links We have three types of links to use in sphinx,

1. External links (http(s) links).
2. Implicit links to title (within same rst file).
3. Explicit links (labels that can be used anywhere in the project).

External links For a simple link to a site the format is

```
`<www.beagleboard.org>`_
```

this will be rendered as www.beagleboard.org.

You can also include a label to the link as shown below.

```
`BeagleBoard.org <www.beagleboard.org>`_
```

this will be rendered as BeagleBoard.org.

Implicit Links These are basically the headings inside the rst page which can be used as a link to that section within document.

```
`Links`_
```

when rendered it becomes *Links*

Explicit link These are special links you can assign to a specific part of the document and reference anywhere in the project unlike implicit links which can be used only within the document they are defined. On top of each page you'll see some text like `.. _rst-cheat-sheet:` is used to create a label for this chapter. These are called the explicit links amd you can reference these using two methods.

```
rst-cheat-sheet_
```

```
:ref:`<rst-cheat-sheet>`_
```

Both can be used inside/outside of the document and the rendered link will take you directly to that specific segment.

Annotations

```
.. callout:: ①

    .. code-block:: python ⍰

        import numpy as np # <1> ③
        import math # <2>

    .. annotations:: ④

        <1> Comment #1 ⑤

        <2> Comment #2

.. annotations::

    ① Indent everything under a `callout`

    ② Create a normal block for what you want to annotate

    ③ Add ``<number>`` everywhere you want to annotate. Put it under a␣
↪comment block if you want the code to run when copied directly.

    ④ Create an `annotations` block to hold your callout comments

    ⑤ Create an entry, separating each with a blank line and prefixing them␣
↪with ``<number>``
```

```python
import numpy as np # ⍰
import math # ⍰
```

① Comment #1

② Comment #2

Important: In the example, I inserted the invisible UTF character U+FEFF after the opening < to avoid it being interpreted as a callout symbol. Be sure to remove that character if you attempt to copy-and-paste the example.

More

footnotes

Chapter 2

Boards

BeaglePlay is designed to make embedded Linux simpler with boundless options for connectivity, sensors, actuators and indicators without breadboarding or other complex wiring solutions.

BeagleBone is a family of ARM-based, Linux-capable boards intended to be bare-bones, with a balance of features to enable rapid prototyping and provide a solid reference for building end products.

PocketBeagle boards are ultra-tiny ARM-based, Linux-capable boards intended to be very low cost, with minimal features suitable for beginners and attractive to professionals looking for a more minimal starting point.

BeagleBone and PocketBeagle *Capes* are add-on boards for BeagleBone and PocketBeagle boards.

BeagleConnect boards are ARM microcontroller-based, Zephyr-capable boards meant to act as ultra low cost smart peripherals to their Linux-capable counterparts, with connectivity options that enable almost endless sensing and actuation expansion.

BeagleBoard is a family of ARM-based, Linux-capable boards where this project started.

Contributors

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

2.1 BeagleBone (all)

BeagleBone boards are intended to be bare-bones, with a balance of features to enable rapid prototyping and provide a solid reference for building end products.

The most popular design is *BeagleBone Black*, a staple reference for an open hardware embedded Linux single board computer.

BeagleBone AI-64 is our most powerful design with tremendous machine learning inference performance, 64-bit processing and a mixture of microcontrollers for various types of highly-reliable and low-latency control.

For simplicity of developing small, mobile robotics, check out *BeagleBone Blue*, a highly integrated board with motor drivers, battery support, altimeter, gyroscope, accelerometer, and much more to get started developing quickly.

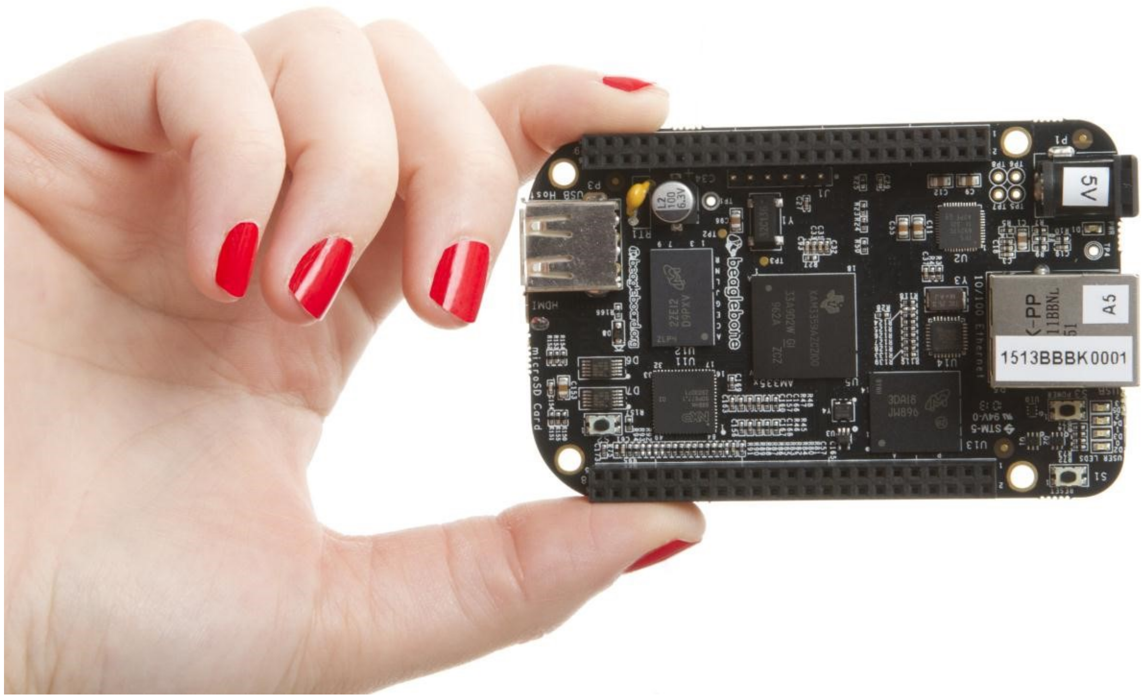
The System Reference Manual for each BeagleBone board is below. Older boards are supported with links to their latest PDF-formatted System Reference Manual and the latest boards are included both here and in the downloadable beagleboard-docs.pdf linked on the bottom-left of your screen.

All boards received without RMA approval will not be worked on.

- BeagleBone (original)
- *BeagleBone Black*
- *BeagleBone Blue*
- *BeagleBone AI*
- *BeagleBone AI-64*

2.2 BeagleBone Black

BeagleBone Black is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.



2.2.1 Introduction

This document is the *System Reference Manual* for the BeagleBone Black and covers its use and design. The board will primarily be referred to in the remainder of this document simply as the board, although it may also be referred to as the BeagleBone Black as a reminder. There are also references to the original BeagleBone as well, and will be referenced as simply BeagleBone.

This design is subject to change without notice as we will work to keep improving the design as the product matures based on feedback and experience. Software updates will be frequent and will be independent of the hardware revisions and as such not result in a change in the revision number.

Make sure you check the docs repository frequently for the most up to date information.

https://git.beagleboard.org/docs/docs.beagleboard.io/-/tree/main/beaglebone-black

2.2.2 Change History

This section describes the change history of this document and board. Document changes are not always a result of a board change. A board change will always result in a document change.

Document Change History

Table 2.1: AsciiDoc Change History

Rev	Changes	Date	By
A4	Preliminary	January 4, 2013	GC
A5	Production release	January 8.2013	GC

continues on next page

Table 2.1 – continued from previous page

Rev	Changes	Date	By
A5.1	1. Added information on Power button and the battery access points. 2. Final production released version.	April 1 2013	GC
A5.2	1. Edited version. 2. Added numerous pictures of the Rev A5A board.	April 23 2013	GC
A5.3	1. Updated serial number locations. 2. Corrected the feature table for 4 UARTS 3. Corrected eMMC pin table to match other tables in the manual.	April 30, 2013	GC
A5.4	1. Corrected revision listed in section 2. Rev A5A is the initial production release. 2. Added all the locations of the serial numbers 3. Made additions to the compatibility list. 4. Corrected «table-7» for LED GPIO pins. 5. Fixed several typos. 6. Added some additional information about LDOs and Step-Down converters. 7. Added short section on HDMI.	May 12, 2013	GC
A5.5	1. Release of the A5B version. 2. The LEDS were dimmed by changing the resistors. 3. The serial termination mode was incorporated into the PCB.	May 20, 2013	GC
A5.6	1. Added information on Rev A5C 2. Added PRU/ICSS options to tables for P8 and P9. 3. Added section on USB Host Correct modes on «table-15». 4. Fixed a few typos	June 16, 2013	GC
A5.7	1. Updated assembly revision to A6. 2. PCB change to add buffer to the reset line and ground the oscillator GND pin. 3. Added resistor on PCB for connection of OSC_GND to board GND.	August 9, 2013	GC
A6	1. Added Rev A6 changes.	October 11, 2013	GC
A6A	1. Added Rev A6A changes	December 17, 2013	GC
B	1. Changed the processor to the AM3358BZCZ	January 20, 2013	GC

continues on next page

Table 2.1 – continued from previous page

Rev	Changes	Date	By
C	1. Changed the eMMC from 2GB to 4GB. 2. Added additional supplier to DDR2 and eMMC.	March 21,2014	GC
C.1	1. Added note to recommend powering off the board with the power	March 22, 2014	GC
C.2	Numerous community edits and format changes to asciidoc.	May 6, 2020	JK
C.3	Added information for board rev C3.	August 24, 2021	JK

Board Changes

Rev C3 PCB revision C.

- Updated microSD card cage due to availability. See https://git.beagleboard.org/beagleboard/beaglebone-black/-/issues/6. Added series resistors and depopulated C5.
- Added reset option (GPIO1_8) for Ethernet PHY to avoid possible start-up issue. See https://git.beagleboard.org/beagleboard/beaglebone-black/-/issues/4.
- Added series resistors to MMC1 lines and depopulated C24.
- Connected pin A6 of J5 on U13 (eMMC IC) to DGND.
- Changed USB1_VBUS series resistor to 0 ohm.
- Change required PCB revision to C.

Initial boxes mistakenly say rev C1.

Rev C2 PCB revision B6.

- Update memories based on availability. See https://github.com/beagleboard/beaglebone-black/commit/74914bd01efeb61376ec3dda4bf9143ad2bb635c.
  - DDR3:
    - Kingston D2516EC4BXGGB-U
  - eMMC:
    - Kingston MMC04G-M627-X02U

Rev C1 PCB revision B6.

- Update memories based on availability. See https://github.com/beagleboard/beaglebone-black/commit/5787736d816832cc8cc9629d19f334b6a12e67f9.
  - DDR3:
    - Micron MT41K256M16TW-107:P
  - eMMC:
    - Micron MTFC4GACAJCN-1M WT
    - Kingston EMMC04G-S100-A08U

Rev C

- Changed the eMMC from 2GB to 4GB.

2GB devices are getting harder to get as they are being phased out. This required us to move to 4GB. We now have two sources for the device. This will however, require an increase in the price of the board.

Rev B

- Changed the processor to the AM3358BZCZ100.

Rev A6A

- Added connection from 32KHz OSC_GND to system ground and changed C106 to 1uF.
- Changes C25 to 2.2uF. This resolved an issue we were seeing in a few boards where the board would not boot in 1 in 20 tries.
- Change required PCB revision to B6.

Rev A6

- In random instances there could be a glitch in the SYS_RESETn signal from the processor where the SYS_RESETn signal was taken high for a momentary amount of time before it was supposed to. To prevent this, the signal was ORed with the PORZn (Power On reset).
- Noise issues were observed in other design where the clock oscillator was getting hit due to a suspected issue in ground bounce. A zero ohm resistor was added to connect the OSC_GND to the system ground.

There are no new features added as a result of these changes.

Rev A5C We were seeing some fallout in production test where we were seeing some jitter on the HDMI display test. It started showing up on our second production run. R46, R47, R48 were changed to 0 ohm from 33 ohm. R45 was taken from 330 ohm to 22 ohm.

We do not know of any boards that were shipped with this issue as this issue was caught in production test. No impact on features or functionality resulted from this change.

Rev A5B There is no operational difference between the Rev A5A and the Rev A5B. There were two changes made to the A5B version.

- Due to complaints about the brightness of the LEDs keeping people awake at night, the LEDs were dimmed. Resistors were changed from 820 ohms to 4.75K ohms.
- The PCB revision was updated to incorporate the hand mod that was being done on the board during manufacturing. The resistor was incorporated into the next revision of the PCB.

The highest supported resolution is now listed as 1920x1080@24Hz. This was not a result of any hardware changes but only updated software. The A5A version also supports this resolution.

Rev A5A This is the initial production release of the board. We will be tracking changes from this point forward.

2.2.3 Connecting Up Your BeagleBone Black

This section provides instructions on how to hook up your board. Two scenarios will be discussed:

1. Tethered to a PC and
2. As a standalone development platform in a desktop PC configuration.

What's In the Box

In the box you will find three main items as shown in «figure-1».

- BeagleBone Black
- miniUSB to USB Type A Cable
- Instruction card with link to the support WIKI address.

This is sufficient for the tethered scenario and creates an out of box experience where the board can be used immediately with no other equipment needed.

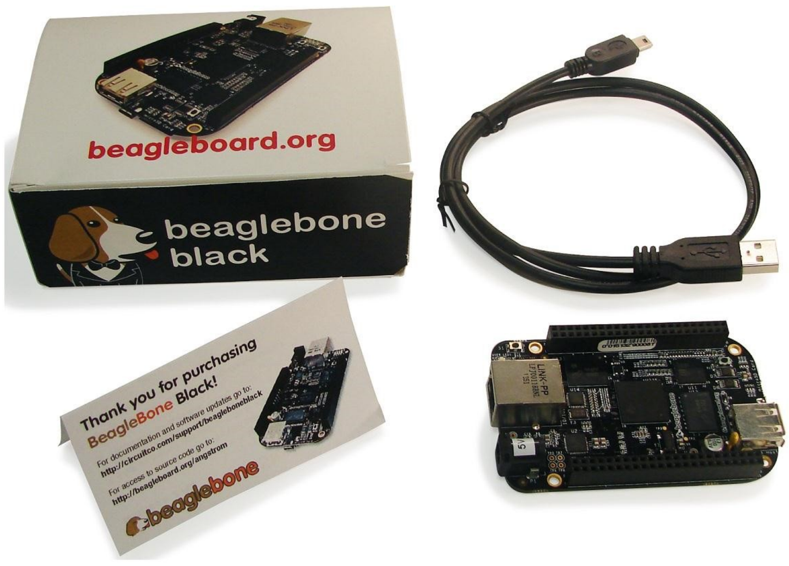


Fig. 2.1: In the Box

Main Connection Scenarios

This section will describe how to connect the board for use. This section is basically a slightly more detailed description of the Quick Start Guide that came in the box. There is also a Quick Start Guide document on the board that should also be referred to. The intent here is that someone looking to purchase the board will be able to read this section and get a good idea as to what the initial set up will be like.

The board can be configured in several different ways, but we will discuss the two most common scenarios as described in the Quick Start Guide card that comes in the box.

- Tethered to a PC via the USB cable
  - Board is accessed as a storage drive
  - Or a RNDIS Ethernet connection.
- Standalone desktop
  - Display
  - Keyboard and mouse
  - External 5V power supply

Each of these configurations is discussed in general terms in the following sections.

For an up-to-date list of confirmed working accessories please go to BeagleBone_Black_Accessories

Tethered To A PC

In this configuration, the board is powered by the PC via the provided USB cable–no other cables are required. The board is accessed either as a USB storage drive or via the browser on the PC. You need to use either Firefox or Chrome on the PC, Internet Explorer will not work properly. «figure-2» shows this configuration.

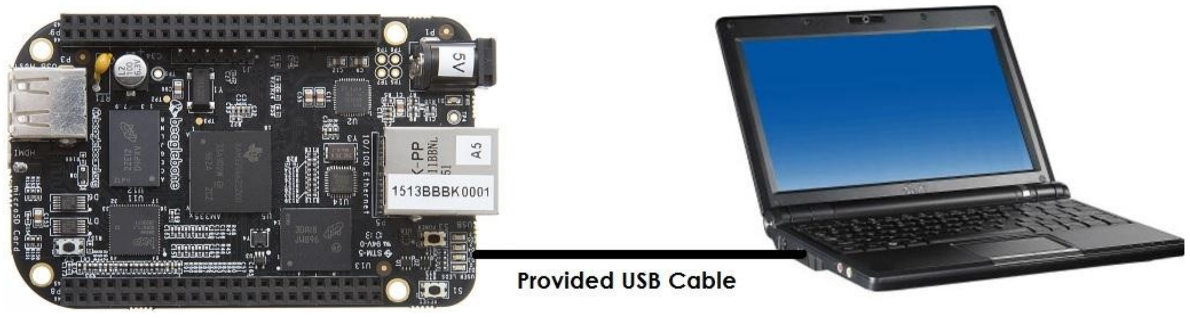


Fig. 2.2: Tethered Configuration

All the power for the board is provided by the PC via the USB cable. In some instances, the PC may not be able to supply sufficient power for the board. In that case, an external 5VDC power supply can be used, but this should rarely be necessary.

Connect the Cable to the Board

1. Connect the small connector on the USB cable to the board as shown in *figure-3*. The connector is on the bottom side of the board.

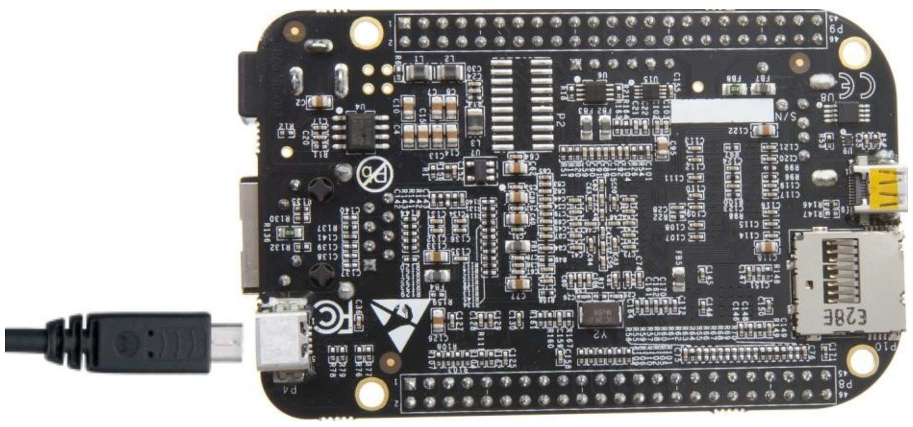


Fig. 2.3: USB Connection to the Board

2. Connect the large connector of the USB cable to your PC or laptop USB port.
3. The board will power on and the power LED will be on as shown in figure below.
4. When the board starts to the booting process started by the process of applying power, the LEDs will come on in sequence as shown in *figure-5* below. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it begins to boot the Linux kernel.

Accessing the Board as a Storage Drive The board will appear around a USB Storage drive on your PC after the kernel has booted, which will take around 10 seconds. The kernel on the board needs to boot before the port gets enumerated. Once the board appears as a storage drive, do the following:

1. Open the USB Drive folder.
2. Click on the file named *start.htm*

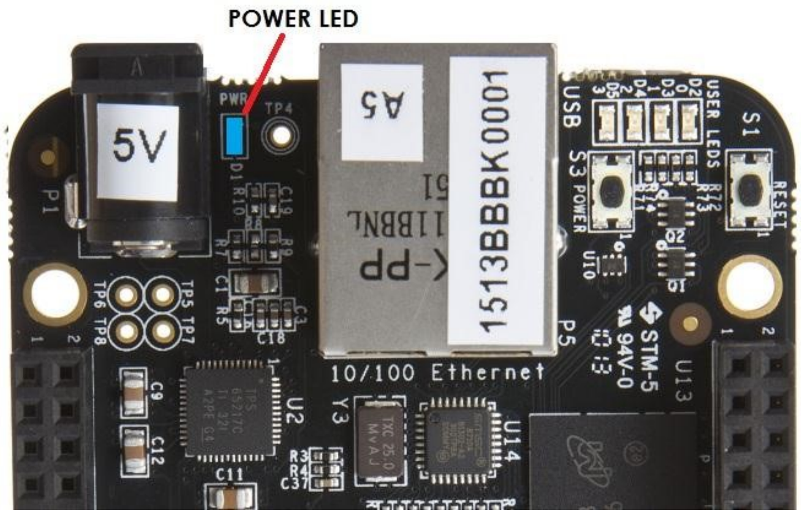


Fig. 2.4: Board Power LED

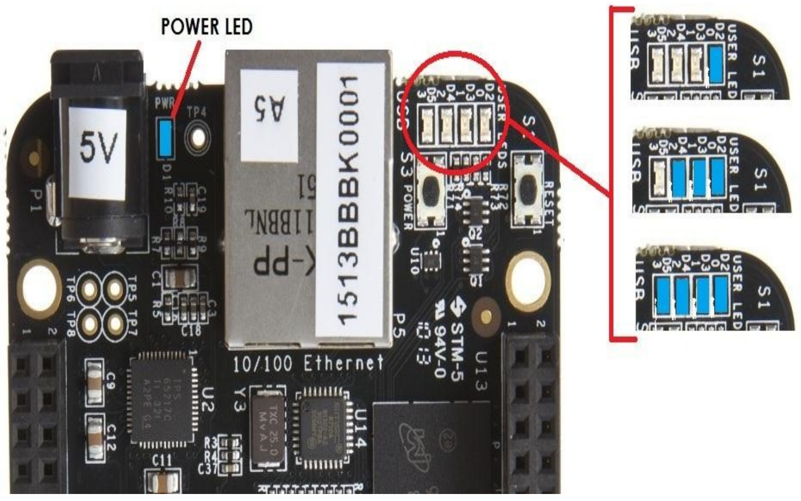


Fig. 2.5: Board Boot Status

3. The file will be opened by your browser on the PC and you should get a display showing the Quick Start Guide.
4. Your board is now operational! Follow the instructions on your PC screen.

Standalone w/Display and Keyboard/Mouse

In this configuration, the board works more like a PC, totally free from any connection to a PC as shown in «figure-6». It allows you to create your code to make the board do whatever you need it to do. It will however require certain common PC accessories. These accessories and instructions are described in the following section.

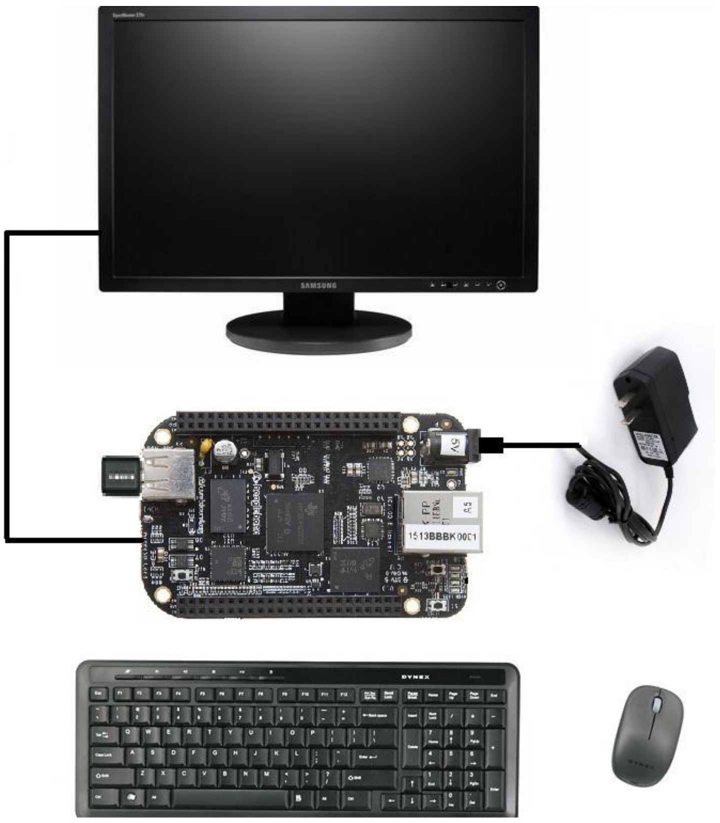


Fig. 2.6: Desktop Configuration

Optionally an Ethernet cable can also be used for network access.

Required Accessories In order to use the board in this configuration, you will need the following accessories:

- 1 x 5VDC 1A power supply
- 1 x HDMI monitor or a DVI-D monitor. (*NOTE:* Only HDMI will give you audio capability).
- 1 x Micro HDMI to HDMI cable or a Micro HDMI to DVI-D adapter.
- 1 x USB wireless keyboard and mouse combo.
- 1 x USB HUB (OPTIONAL). The board has only one USB host port, so you may need to use a USB Hub if your keyboard and mouse requires two ports.

For an up-to-date list of confirmed working accessories please go to BeagleBone_Black_Accessories

Connecting Up the Board

1. Connect the big end of the HDMI cable as shown in *figure-7* to your HDMI monitor. Refer to your monitor Owner's Manual for the location of your HDMI port. If you have a DVI-D Monitor go to *Step 3*, otherwise proceed to *Step 4* .

Fig. 2.7: Connect microHDMI Cable to the Monitor

2. If you have a DVI-D monitor you must use a DVI-D to HDMI adapter in addition to your HDMI cable. An example is shown in *figure-8* below from two perspectives. If you use this configuration, you will not have audio support.

Fig. 2.8: DVI-D to HDMI Adapter

3. If you have a single wireless keyboard and mouse combination such as seen in *figure-9* below, you need to plug the receiver in the USB host port of the board as shown in *figure-10* .

Fig. 2.9: Wireless Keyboard and Mouse Combo

If you have a wired USB keyboard requiring two USB ports, you will need a HUB similar to the ones shown in figure below . You may want to have more than one port for other devices. Note that the board can only supply up to 500mA, so if you plan to load it down, it will need to be externally powered.

4. Connect the Ethernet Cable

If you decide you want to connect to your local area network, an Ethernet cable can be used. Connect the Ethernet Cable to the Ethernet port as shown in figure below . Any standard 100M Ethernet cable should work.

5. The final step is to plug in the DC power supply to the DC power jack as shown in figure below.
6. The cable needed to connect to your display is a microHDMI to HDMI. Connect the microHDMI connector end to the board at this time. The connector is on the bottom side of the board as shown in *figure-14*

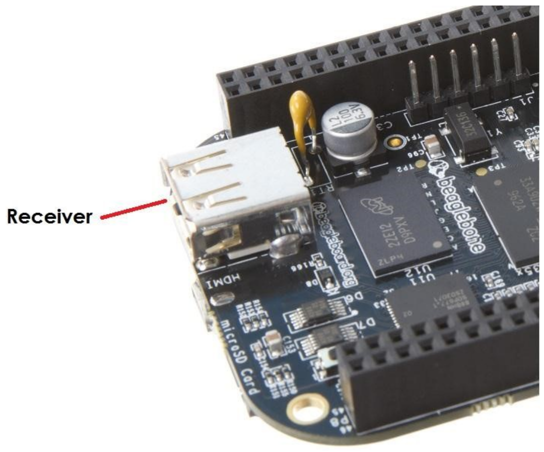


Fig. 2.10: Connect Keyboard and Mouse Receiver to the Board

Fig. 2.11: Keyboard and Mouse Hubs

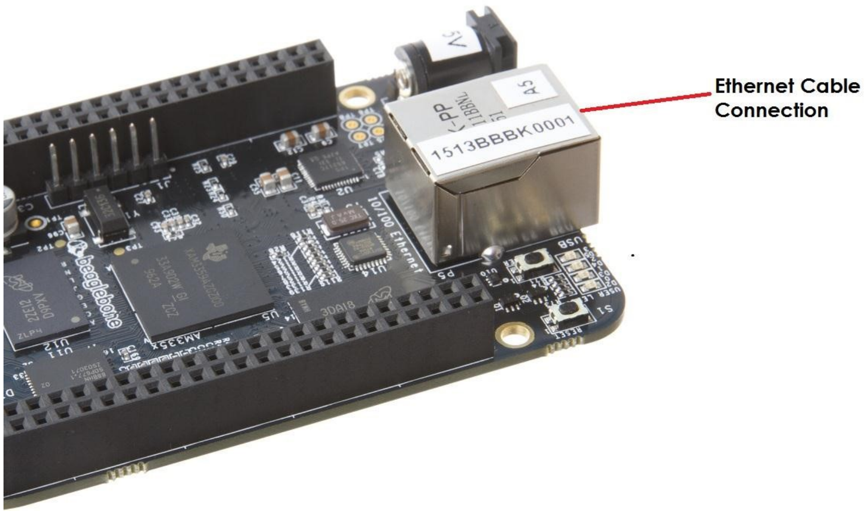


Fig. 2.12: Ethernet Cable Connection

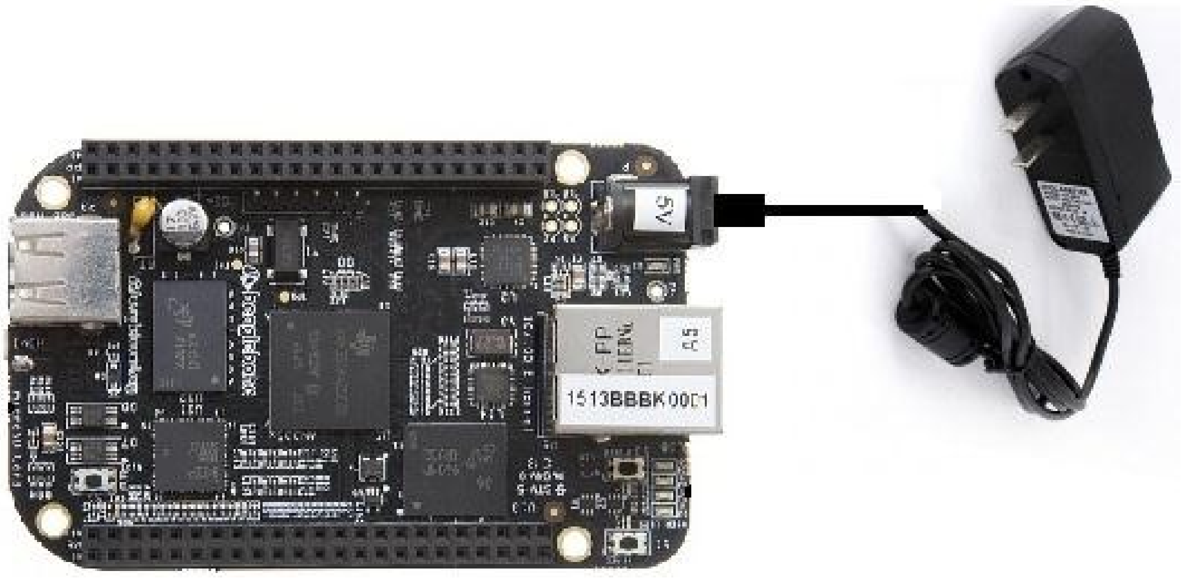


Fig. 2.13: External DC Power

below.

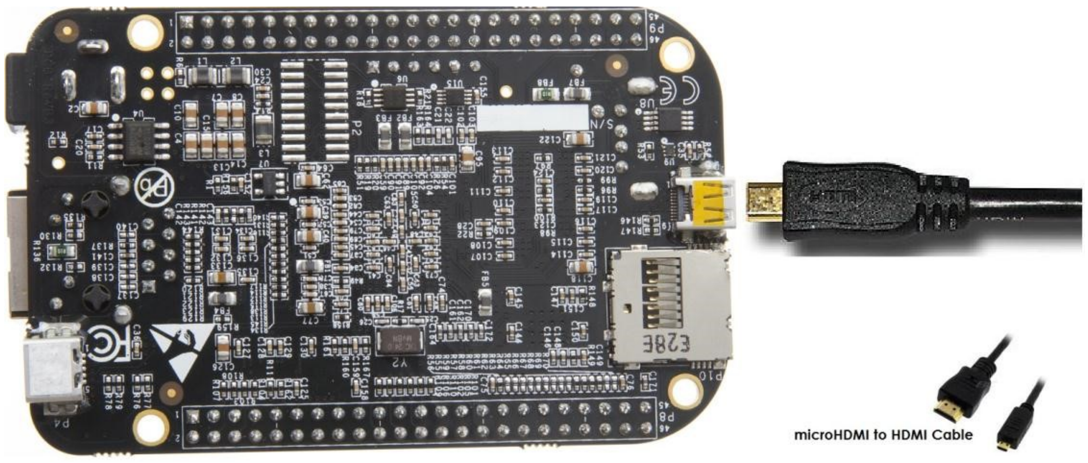


Fig. 2.14: Connect microHDMI Cable to the Board

The connector is fairly robust, but we suggest that you not use the cable as a leash for your Beagle. Take proper care not to put too much stress on the connector or cable.

7. Booting the Board

As soon as the power is applied to the board, it will start the booting up process. When the board starts to boot the LEDs will come on in sequence as shown in *figure-15* below. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it boots the Linux kernel.

While the four user LEDs can be overwritten and used as desired, they do have specific meanings in the image that is shipped with the board once the Linux kernel has booted.

- *USER0* is the heartbeat indicator from the Linux kernel.
- *USER1* turns on when the microSD card is being accessed
- *USER2* is an activity indicator. It turns on when the kernel is not in the idle loop.
- *USER3* turns on when the onboard eMMC is being accessed.

8. A Booted System

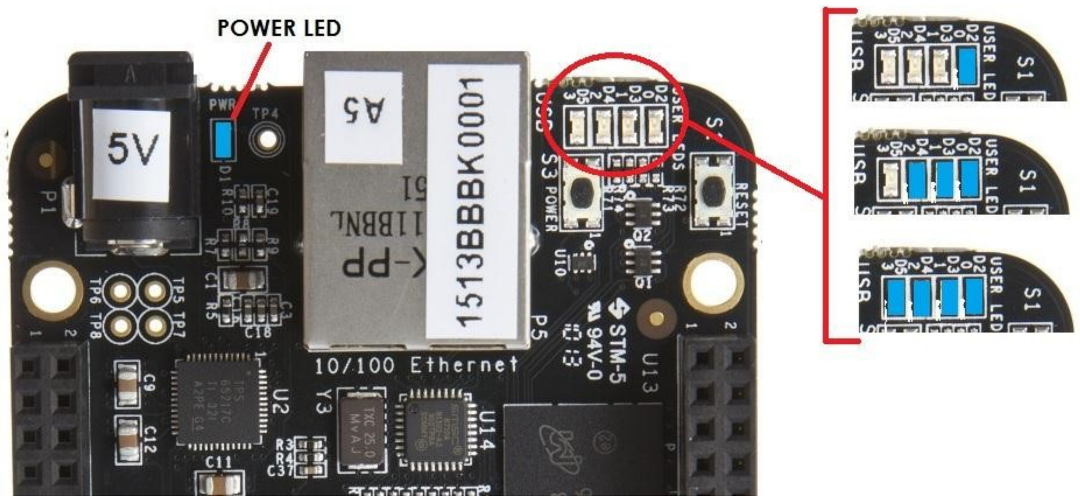


Fig. 2.15: Board Boot Status

a. The board will have a mouse pointer appear on the screen as it enters the Linux boot step. You may have to move the physical mouse to get the mouse pointer to appear. The system can come up in the suspend mode with the HDMI port in a sleep mode.

b. After a minute or two a login screen will appear. You do not have to do anything at this point.

c. After a minute or two the desktop will appear. It should be similar to the one shown in figure-1. HOWEVER, it will change from one release to the next, so do not expect your system to look exactly like the one in the figure, but it will be very similar.

d. And at this point you are ready to go! *figure-16* shows the desktop after booting.

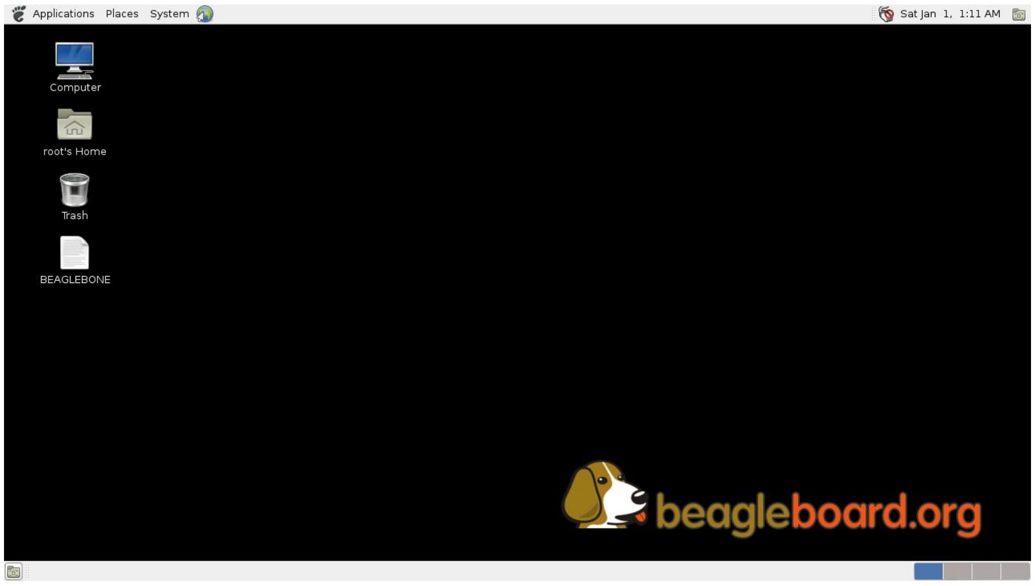


Fig. 2.16: Desktop Screen

9. Powering Down

   A. Press the power button momentarily.

   B. The system will power down automatically.

   C. Remove the power jack.

2.2.4 BeagleBone Black Overview

The BeagleBone Black is the latest addition to the BeagleBoard.org family and like its predecessors, is designed to address the Open Source Community, early adopters, and anyone interested in a low cost ARM Cortex-A8 based processor.

It has been equipped with a minimum set of features to allow the user to experience the power of the processor and is not intended as a full development platform as many of the features and interfaces supplied by the processor are not accessible from the BeagleBone Black via onboard support of some interfaces. It is not a complete product designed to do any particular function. It is a foundation for experimentation and learning how to program the processor and to access the peripherals by the creation of your own software and hardware.

It also offers access to many of the interfaces and allows for the use of add-on boards called capes, to add many different combinations of features. A user may also develop their own board or add their own circuitry.

BeagleBone Black is manufactured and warranted by partners listed at https://beagleboard.org/logo for the benefit of the community and its supporters.

Jason Kridner of Texas Instruments handles the community promotions and is the spokesman for BeagleBoard.org.

The board is designed by Gerald Coley of EmProDesign, a charter member of the BeagleBoard.org community.

The PCB layout up through PCB revision B was done by Circuitco and Circuitco is the sole funder of its development and transition to production. Later PCB revisions have been made by Embest, a subsidiary of Avent.

The Software is written and supported by the thousands of community members, including Jason Kridner, employee of Texas Instruments, and Robert Nelson, employee of DigiKey.

BeagleBone Compatibility

The board is intended to be compatible with the original BeagleBone as much as possible. There are several areas where there are differences between the two designs. These differences are listed below, along with the reasons for the differences.

- Sitara AM3358BZCZ100, 1GHZ, processor.
  - Sorry, we just had to make it faster.
- 512MB DDR3L
  - *Cost reduction*
  - Performance boost
  - Memory size increase
  - Lower power
- No Serial port by default
  - *Cost reduction*
  - Can be added by buying a TTL to USB Cable that is widely available
  - Single largest cost reduction action taken
- No JTAG emulation over USB
  - *Cost reduction* JTAG header is not populated, but can easily be mounted.
  - EEPROM Reduced from 32KB to 4KB
  - *Cost Reduction*
- Onboard Managed NAND (eMMC)
  - 4GB
  - *Cost reduction*

    - Performance boost x8 vs. x4 bits
    - Performance boost due to deterministic properties vs. microSD card
- GPMC bus may not be accessible from the expansion headers in some cases
    - Result of eMMC on the main board
    - Signals are still routed to the expansion connector
    - If eMMC is not used, signals can be used via expansion if eMMC is held in reset
- There may be 10 less GPIO pins available
    - Result of eMMC
    - If eMMC is not used, could still be used
- The power expansion header, for battery and backlight, has been removed
    - _*Cost reduction* , space reduction
    - Four pins were added to provide access to the battery charger function.
- HDMI interface onboard
    - Feature addition
    - Audio and video capable
    - Micro HDMI
- No three function USB cable
    - *Cost reduction*
- GPIO3_21 has a 24.576 MHZ clock on it.
    - This is required by the HDMI Framer for Audio purposes. We needed to run a clock into the processor to generate the correct clock frequency. The pin on the processor was already routed to the expansion header. In order not to remove this feature on the expansion header, it was left connected. In order to use the pin as a GPIO pin, you need to disable the clock. While this disables audio to the HDMI, the fact that you want to use this pin for something else, does the same thing.

BeagleBone Black Features and Specification

This section covers the specifications and features of the board and provides a high level description of the major components and interfaces that make up the board. table below provides a list of the features.

Table 2.2: BeagleBone Black Features

	Feature
Processor	Sitara AM3358BZCZ100 1GHz, 2000 MIPS
Graphics Engine	SGX530 3D, 20M Polygons/S
SDRAM Memory	512MB DDR3L 800MHZ
Onboard Flash	4GB, 8bit Embedded MMC
PMIC	TPS65217C PMIC regulator and one additional LDO.
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header
Power Source	miniUSB USB or DC Jack
PCB	3.4" x 2.1"
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated

continues on next page

Table 2.2 – continued from previous page

	Feature
Ethernet	10/100, RJ45
SD/MMC Connector	microSD , 3.3V
User Input	1. Reset Button 2. Boot Button 3. Power Button
Video Out	1. 16b HDMI, 1280x1024 (MAX) 2. 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support
Audio	Via HDMI Interface, Stereo
Expansion Connectors	1. Power 5V, 3.3V , VDD_ADC(1.8V) 2. 3.3V I/O on all signals 3. McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 4. AIN _(1.8V MAX)_, 4 Timers, 4 Serial Ports, CAN0, 5. EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)
Weight	1.4 oz (39.68 grams)
Power	Refer to *section-6-1-7*

Board Component Locations

This section describes the key components on the board. It provides information on their location and function. Familiarize yourself with the various components on the board.

Connectors, LEDs, and Switches figure below shows the locations of the connectors, LEDs, and switches on the PCB layout of the board.

- *DC Power* is the main DC input that accepts 5V power.
- *Power Button* alerts the processor to initiate the power down sequence and is used to power down the board.
- *10/100 Ethernet* is the connection to the LAN.
- *Serial Debug* is the serial debug port.
- *USB Client* is a miniUSB connection to a PC that can also power the board.
- *BOOT switch* can be used to force a boot from the microSD card if the power is cycled on the board, removing power and reapplying the power to the board..
- There are four blue **LED**'s that can be used by the user.
- *Reset Button* allows the user to reset the processor.
- *microSD* slot is where a microSD card can be installed.
- *microHDMI* connector is where the display is connected to.
- *USB Host* can be connected different USB interfaces such as Wi-Fi, BT, Keyboard, etc.

Key Components figure below shows the locations of the key components on the PCB layout of the board.

- *Sitara AM3358BZCZ100* is the processor for the board.

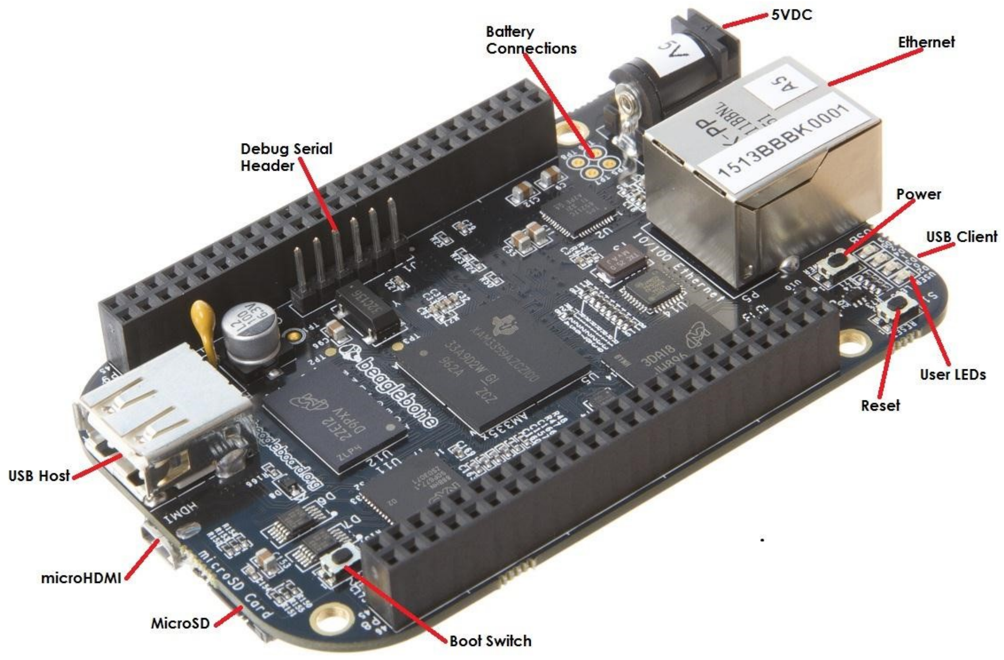


Fig. 2.17: Connectors, LEDs and Switches

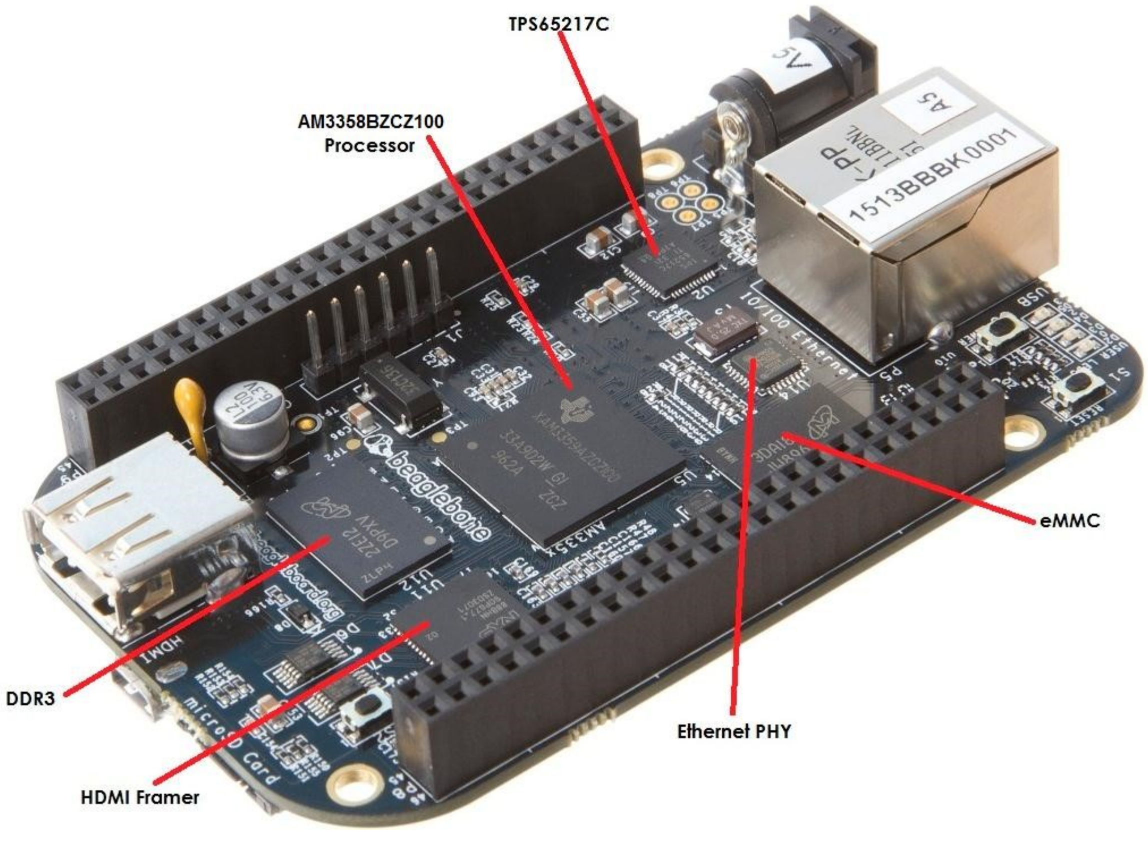


Fig. 2.18: Key Components

- *Micron 512MB DDR3L* or**Kingston 512mB DDR3** is the Dual Data Rate RAM memory.
- *TPS65217C PMIC* provides the power rails to the various components on the board.
- *SMSC Ethernet PHY* is the physical interface to the network.
- *Micron eMMC* is an onboard MMC chip that holds up to 4GB of data.
- *HDMI* Framer provides control for an HDMI or DVI-D display with an adapter.

2.2.5 BeagleBone Black High Level Specification

This section provides the high level specification of the BeagleBone Black.

Block Diagram

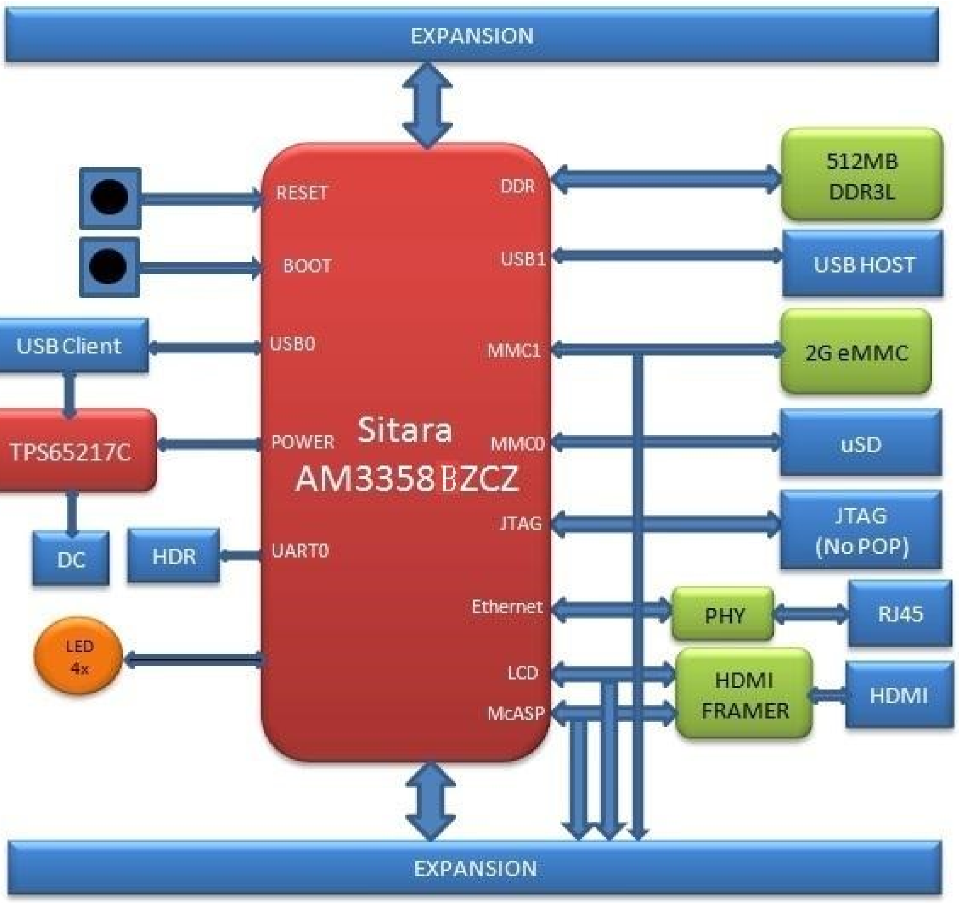


Fig. 2.19: BeagleBone Black Key Components

Processor

The revision B and later boards have moved to the Sitara AM3358BZCZ100 device.

Memory

Described in the following sections are the three memory devices found on the board.

512MB DDR3L A single 256Mb x16 DDR3L 4Gb (512MB) memory device is used. The memory used is one of two devices:

- MT41K256M16HA-125 from Micron
- D2516EC4BXGGB from Kingston

It will operate at a clock frequency of 400MHz yielding an effective rate of 800MHZ on the DDR3L bus allowing for 1.6GB/S of DDR3L memory bandwidth.

4KB EEPROM A single 4KB EEPROM is provided on I2C0 that holds the board information. This information includes board name, serial number, and revision information. This is the not the same as the one used on the original BeagleBone. The device was changed for cost reduction reasons. It has a test point to allow the device to be programmed and otherwise to provide write protection when not grounded.

4GB Embedded MMC A single 4GB embedded MMC (eMMC) device is on the board. The device connects to the MMC1 port of the processor, allowing for 8bit wide access. Default boot mode for the board will be MMC1 with an option to change it to MMC0, the SD card slot, for booting from the SD card as a result of removing and reapplying the power to the board. Simply pressing the reset button will not change the boot mode. MMC0 cannot be used in 8Bit mode because the lower data pins are located on the pins used by the Ethernet port. This does not interfere with SD card operation but it does make it unsuitable for use as an eMMC port if the 8 bit feature is needed.

MicroSD Connector The board is equipped with a single microSD connector to act as the secondary boot source for the board and, if selected as such, can be the primary boot source. The connector will support larger capacity microSD cards. The microSD card is not provided with the board. Booting from MMC0 will be used to flash the eMMC in the production environment or can be used by the user to update the SW as needed.

Boot Modes As mentioned earlier, there are four boot modes:

- **eMMC Boot:** This is the default boot mode and will allow for the fastest boot time and will enable the board to boot out of the box using the pre-flashed OS image without having to purchase an microSD card or an microSD card writer.
- **SD Boot:** This mode will boot from the microSD slot. This mode can be used to override what is on the eMMC device and can be used to program the eMMC when used in the manufacturing process or for field updates.
- **Serial BooT:** This mode will use the serial port to allow downloading of the software direct. A separate USB to serial cable is required to use this port.
- **USB Boot:** This mode supports booting over the USB port.

Software to support USB and serial boot modes is not provided by beagleboard.org.Please contact TI for support of this feature.

A switch is provided to allow switching between the modes.

- Holding the boot switch down during a removal and reapplication of power without a microSD card inserted will force the boot source to be the USB port and if nothing is detected on the USB client port, it will go to the serial port for download.
- Without holding the switch, the board will boot try to boot from the eMMC. If it is empty, then it will try booting from the microSD slot, followed by the serial port, and then the USB port.
- If you hold the boot switch down during the removal and reapplication of power to the board, and you have a microSD card inserted with a bootable image, the board will boot from the microSD card.

NOTE: Pressing the RESET button on the board will NOT result in a change of the_ _boot mode. You MUST remove power and reapply power to change the boot mode.The boot pins are sampled during power on reset from the PMIC to the processor.The reset button on the board is a warm reset only and will not force a boot mode change.

Power Management

The *TPS65217C* power management device is used along with a separate LDO to provide power to the system. The**TPS65217C** version provides for the proper voltages required for the DDR3L. This is the same device as used on the original BeagleBone with the exception of the power rail configuration settings which will be changed in the internal EEPROM to the *TPS65217C* to support the new voltages.

DDR3L requires 1.5V instead of 1.8V on the DDR2 as is the case on the original BeagleBone. The 1.8V regulator setting has been changed to 1.5V for the DDR3L. The LDO3 3.3V rail has been changed to 1.8V to support those rails on the processor. LDO4 is still 3.3V for the 3.3V rails on the processor. An external *LDOTLV70233* provides the 3.3V rail for the rest of the board.

PC USB Interface

The board has a miniUSB connector that connects the USB0 port to the processor. This is the same connector as used on the original BeagleBone.

Serial Debug Port

Serial debug is provided via UART0 on the processor via a single 1x6 pin header. In order to use the interface a USB to TTL adapter will be required. The header is compatible with the one provided by FTDI and can be purchased for about $$12 to $$20 from various sources. Signals supported are TX and RX. None of the handshake signals are supported.

USB1 Host Port

On the board is a single USB Type A female connector with full LS/FS/HS Host support that connects to USB1 on the processor. The port can provide power on/off control and up to 500mA of current at 5V. Under USB power, the board will not be able to supply the full 500mA, but should be sufficient to supply enough current for a lower power USB device supplying power between 50 to 100mA.

You can use a wireless keyboard/mouse configuration or you can add a HUB for standard keyboard and mouse interfacing.

Power Sources

The board can be powered from four different sources:

- A USB port on a PC
- A 5VDC 1A power supply plugged into the DC connector.
- A power supply with a USB connector.
- Expansion connectors

The USB cable is shipped with each board. This port is limited to 500mA by the Power Management IC. It is possible to change the settings in the *TPS65217C* to increase this current, but only after the initial boot. And, at that point the PC most likely will complain, but you can also use a dual connector USB cable to the PC to get to 1A.

The power supply is not provided with the board but can be easily obtained from numerous sources. A 1A supply is sufficient to power the board, but if there is a cape plugged into the board or you have a power hungry device or hub plugged into the host port, then more current may needed from the DC supply.

Power routed to the board via the expansion header could be provided from power derived on a cape. The DC supply should be well regulated and 5V +/-.25V.

Reset Button

When pressed and released, causes a reset of the board. The reset button used on the BeagleBone Black is a little larger than the one used on the original BeagleBone. It has also been moved out to the edge of the board so that it is more accessible.

Power Button

A power button is provided near the reset button close to the Ethernet connector. This button takes advantage of the input to the PMIC for power down features. While a lot of capes have a button, it was decided to add this feature to the board to ensure everyone had access to some new features. These features include:

- Interrupt is sent to the processor to facilitate an orderly shutdown to save files and to un-mount drives.
- Provides ability to let processor put board into a sleep mode to save power.
- Can alert processor to wake up from sleep mode and restore state before sleep was entered.

If you hold the button down longer than 8 seconds, the board will power off if you release the button when the power LED turns off. If you continue to hold it, the board will power back up completing a power cycle.

We recommend that you use this method to power down the board. It will also help prevent contamination of the SD card or the eMMC.

If you do not remove the power jack, you can press the button again and the board will power up.

Indicators

There are a total of five blue LEDs on the board.

- One blue power LED indicates that power is applied and the power management IC is up. If this LED flashes when applying power, it means that an excess current flow was detected and the PMIC has shut down.
- Four blue LEDs that can be controlled via the SW by setting GPIO pins.

In addition, there are two LEDs on the RJ45 to provide Ethernet status indication. One is yellow (100M Link up if on) and the other is green (Indicating traffic when flashing).

CTI JTAG Header

A place for an optional 20 pin CTI JTAG header is provided on the board to facilitate the SW development and debugging of the board by using various JTAG emulators. This header is not supplied standard on the board. To use this, a connector will need to be soldered onto the board.

If you need the JTAG connector you can solder it on yourself. No other components are needed. The connector is made by Samtec and the part number is FTR-110-03-G-D-06. You can purchase it from http://www.digikey.com/

HDMI Interface

A single HDMI interface is connected to the 16 bit LCD interface on the processor. The 16b interface was used to preserve as many expansion pins as possible to allow for use by the user. The NXP TDA19988BHN is used to convert the LCD interface to HDMI and convert the audio as well. The signals are still connected to the expansion headers to enable the use of LCD expansion boards or access to other functions on the board as needed.

The HDMI device does not support HDCP copy protection. Support is provided via EDID to allow the SW to identify the compatible resolutions. Currently the following resolutions are supported via the software:

- 1280 x 1024

- 1440 x 900
- 1024 x 768
- 1280 x 720

Cape Board Support

The BeagleBone Black has the ability to accept up to four expansion boards or capes that can be stacked onto the expansion headers. The word cape comes from the shape of the board as it is fitted around the Ethernet connector on the main board. This notch acts as a key to ensure proper orientation of the cape.

The majority of capes designed for the original BeagleBone will work on the BeagleBone Black. The two main expansion headers will be populated on the board. There are a few exceptions where certain capabilities may not be present or are limited to the BeagleBone Black. These include:

- GPMC bus may NOT be available due to the use of those signals by the eMMC. If the eMMC is used for booting only and the file system is on the microSD card, then these signals could be used.
- Another option is to use the microSD or serial boot modes and not use the eMMC.
- The power expansion header is not on the BeagleBone Black so those functions are not supported.

For more information on cape support refer to *BeagleBone Black Mechanical* section.

2.2.6 Detailed Hardware Design

This section provides a detailed description of the Hardware design. This can be useful for interfacing, writing drivers, or using it to help modify specifics of your own design.

Power Section

This section describes the power section of the design and all the functions performed by the *TPS65217C*.

TPS65217C PMIC The main Power Management IC (PMIC) in the system is the *TPS65217C* which is a single chip power management IC consisting of a linear dual-input power path, three step-down converters, and four LDOs. LDO stands for Low Drop Out. If you want to know more about an LDO, you can go to http://en.wikipedia.org/wiki/Low-dropout_regulator .If you want to learn more about step-down converters, you can go to

http://en.wikipedia.org/wiki/DC-to-DC_converter

The system is supplied by a USB port or DC adapter. Three high-efficiency 2.25MHz step-down converters are targeted at providing the core voltage, MPU, and memory voltage for the board.

The step-down converters enter a low power mode at light load for maximum efficiency across the widest possible range of load currents. For low-noise applications the devices can be forced into fixed frequency PWM using the I2C interface. The step-down converters allow the use of small inductors and capacitors to achieve a small footprint solution size.

LDO1 and LDO2 are intended to support system standby mode. In normal operation, they can support up to 100mA each. LDO3 and LDO4 can support up to 285mA each.

By default only LDO1 is always ON but any rail can be configured to remain up in SLEEP state. In particular the DCDC converters can remain up in a low-power PFM mode to support processor suspend mode. The *TPS65217C* offers flexible power-up and power-down sequencing and several house-keeping functions such as power-good output, pushbutton monitor, hardware reset function and temperature sensor to protect the battery.

For more information on the *TPS65217C*, refer to http://www.ti.com/product/tps65217C

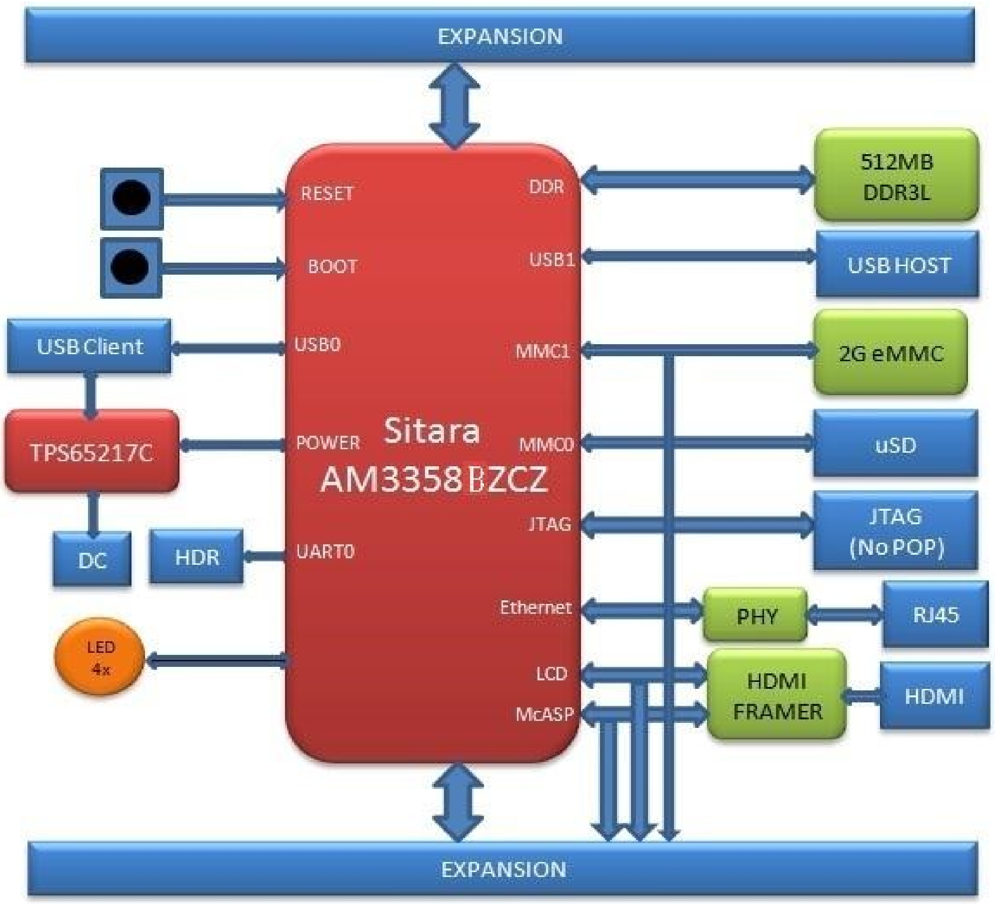


Fig. 2.20: BeagleBone Black Block Diagram

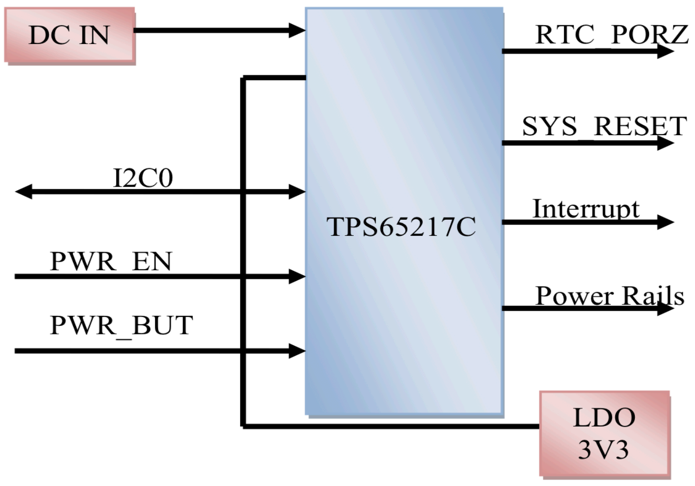


Fig. 2.21: High Level Power Block Diagram

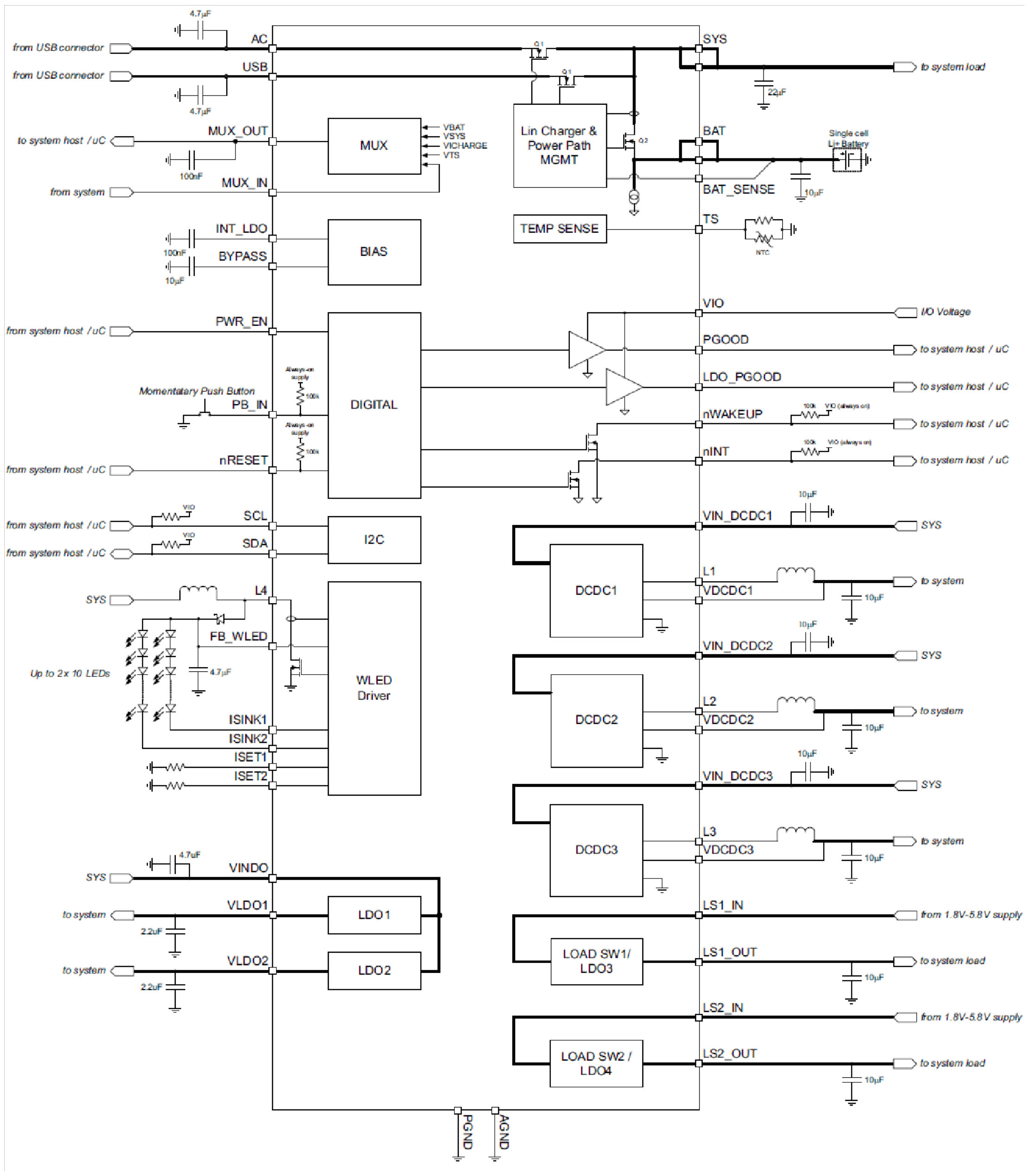


Fig. 2.22: TPS65217C Block Diagram

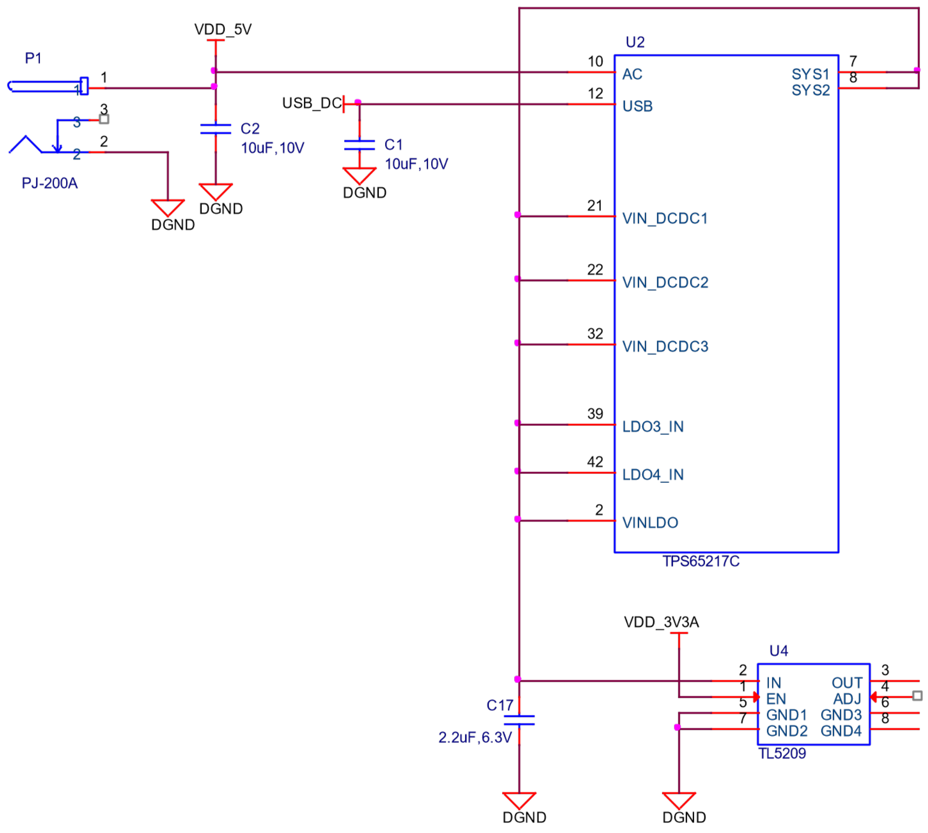


Fig. 2.23: TPS65217 DC Connection

DC Input A 5VDC supply can be used to provide power to the board. The power supply current depends on how many and what type of add-on boards are connected to the board. For typical use, a 5VDC supply rated at 1A should be sufficient. If heavier use of the expansion headers or USB host port is expected, then a higher current supply will be required.

The connector used is a 2.1MM center positive x 5.5mm outer barrel. The 5VDC rail is connected to the expansion header. It is possible to power the board via the expansion headers from an add-on card. The 5VDC is also available for use by the add-on cards when the power is supplied by the 5VDC jack on the board.

USB Power The board can also be powered from the USB port. A typical USB port is limited to 500mA max. When powering from the USB port, the VDD_5V rail is not provided to the expansion headers, so capes that require the 5V rail to supply the cape direct, bypassing the *TPS65217C*, will not have that rail available for use. The 5VDC supply from the USB port is provided on the SYS_5V, the one that comes from the**TPS65217C**, rail of the expansion header for use by a cape. *Figure 24* is the connection of the USB power input on the PMIC.

Power Selection The selection of either the 5VDC or the USB as the power source is handled internally to the *TPS65217C* and automatically switches to 5VDC power if both are connected. SW can change the power configuration via the I2C interface from the processor. In addition, the SW can read the**TPS65217C** and determine if the board is running on the 5VDC input or the USB input. This can be beneficial to know the capability of the board to supply current for things like operating frequency and expansion cards.

It is possible to power the board from the USB input and then connect the DC power supply. The board will switch over automatically to the DC input.

Power Button A power button is connected to the input of the *TPS65217C*. This is a momentary switch, the same type of switch used for reset and boot selection on the board.

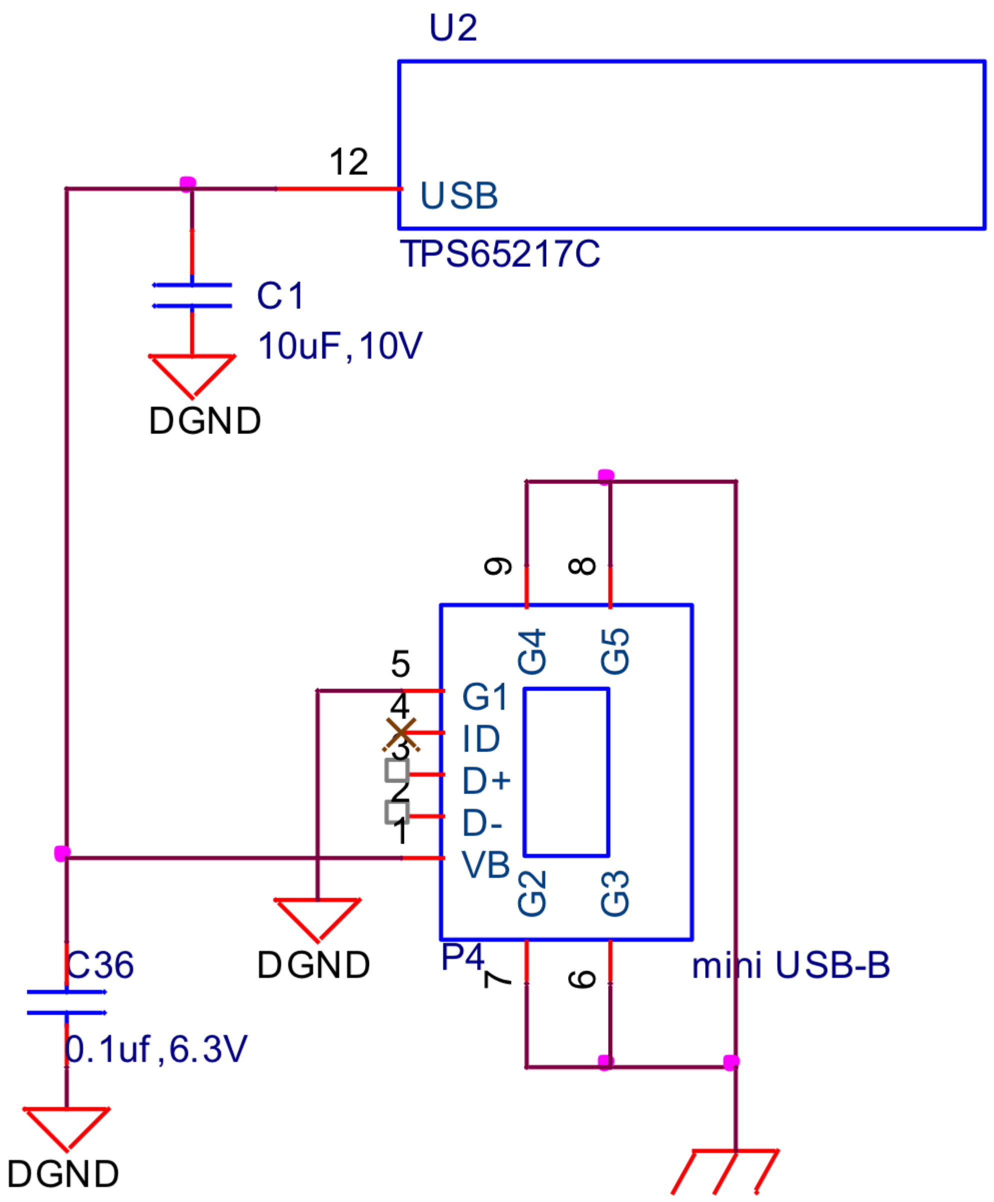


Fig. 2.24: USB Power Connections

If you push the button the *TPS65217C* will send an interrupt to the processor. It is up to the processor to then pull the**PMIC_POWER_EN** pin low at the correct time to power down the board. At this point, the PMIC is still active, assuming that the power input was not removed. Pressing the power button will cause the board to power up again if the processor puts the board in the power off mode.

In power off mode, the RTC rail is still active, keeping the RTC powered and running off the main power input. If you remove that power, then the RTC will not be powered. You also have the option of using the battery holes on the board to connect a battery if desired as discussed in the next section.

If you push and hold the button for greater than 8 seconds, the PMIC will power down. But you must release the button when the power LED turns off. Holding the button past that point will cause the board to power cycle.

Battery Access Pads Four pads are provided on the board to allow access to the battery pins on the *TPS65217C*. The pads can be loaded with a 4x4 header or you may just wire a battery into the pads. In addition they could provide access via a cape if desired. The four signals are listed below in *table-3* .

Table 2.3: BeagleBone Black Battery Pins

PIN	DESIGNATION	FUNCTION
BAT	TP5	Battery connection point
SENSE	TP6	Battery voltage sense input, connect to BAT directly at the battery terminal.
TS	TP7	Temperature sense input. Connect to NTC thermistor to sense battery temperature.
GND	TP8	System ground.

There is no fuel gauge function provided by the *TPS65217C*. That would need to be added if that function was required. If you want to add a fuel gauge, an option is to use 1-wire SPI or I2C device. You will need to add this using the expansion headers and place it on an expansion board.

NOTE: Refer to the TPS65217C documentation + before connecting anything to these pins.

Power Consumption The power consumption of the board varies based on power scenarios and the board boot processes. Measurements were taken with the board in the following configuration:

- DC powered and USB powered
- HDMI monitor connected
- USB HUB
- 4GB USB flash drive
- Ethernet connected @ 100M
- Serial debug cable connected

Table 2.4: BeagleBone Black Power Consumption(mA@5V)

MODE	USB	DC	DC+USB
Reset	TBD	TBD	TBD
Idling @ UBoot	210	210	210
Kernel Booting (Peak)	460	460	460
Kernel Idling	350	350	350
Kernel Idling Display Blank	280	280	280
Loading a Webpage	430	430	430

The current will fluctuate as various activates occur, such as the LEDs on and microSD/eMMC accesses.

Processor Interfaces The processor interacts with the *TPS65217C* via several different signals. Each of these signals is described below.

I2C0

I2C0 is the control interface between the processor and the *TPS65217C*. It allows the processor to control the registers inside the **TPS65217C** for such things as voltage scaling and switching of the input rails.

PMIC_POWR_EN

On power up the *VDD_RTC* rail activates first. After the RTC circuitry in the processor has activated it instructs the**TPS65217C** to initiate a full power up cycle by activating the *PMIC_POWR_EN* signal by taking it HI. When powering down, the processor can take this pin low to start the power down process.

LDO_GOOD

This signal connects to the *RTC_PORZn* signal, RTC power on reset. The small *n indicates that the signal is an active low signal. Word processors seem to be unable to put a bar over a word so the **n** is commonly used in electronics. As the RTC circuitry comes up first, this signal indicates that the LDOs, the 1.8V VRTC rail, is up and stable. This starts the power up process.

PMIC_PGOOD

Once all the rails are up, the *PMIC_PGOOD* signal goes high. This releases the **PORZn** signal on the processor which was holding the processor reset.

WAKEUP

The WAKEUP signal from the *TPS65217C* is connected to the **EXT_WAKEUP** signal on the processor. This is used to wake up the processor when it is in a sleep mode. When an event is detected by the *TPS65217C*, such as the power button being pressed, it generates this signal.

PMIC_INT

The *PMIC_INT* signal is an interrupt signal to the processor. Pressing the power button will send an interrupt to the processor allowing it to implement a power down mode in an orderly fashion, go into sleep mode, or cause it to wake up from a sleep mode. All of these require SW support.

Power Rails **VRTC Rail**

The *VRTC* rail is a 1.8V rail that is the first rail to come up in the power sequencing. It provides power to the RTC domain on the processor and the I/O rail of the **TPS65217C**. It can deliver up to 250mA maximum.

VDD_3V3A Rail

The *VDD_3V3A* rail is supplied by the **TPS65217C** and provides the 3.3V for the processor rails and can provide up to 400mA.

VDD_3V3B Rail

The current supplied by the *VDD_3V3A* rail is not sufficient to power all of the 3.3V rails on the board. So a second LDO is supplied, U4, a **TL5209A**, which sources the *VDD_3V3B* rail. It is powered up just after the *VDD_3V3A* rail.

VDD_1V8 Rail

The *VDD_1V8* rail can deliver up to 400mA and provides the power required for the 1.8V rails on the processor and the HDMI framer. This rail is not accessible for use anywhere else on the board.

VDD_CORE Rail

The *VDD_CORE* rail can deliver up to 1.2A at 1.1V. This rail is not accessible for use anywhere else on the board and connects only to the processor. This rail is fixed at 1.1V and should not be adjusted by SW using the PMIC. If you do, then the processor will no longer work.

VDD_MPU Rail

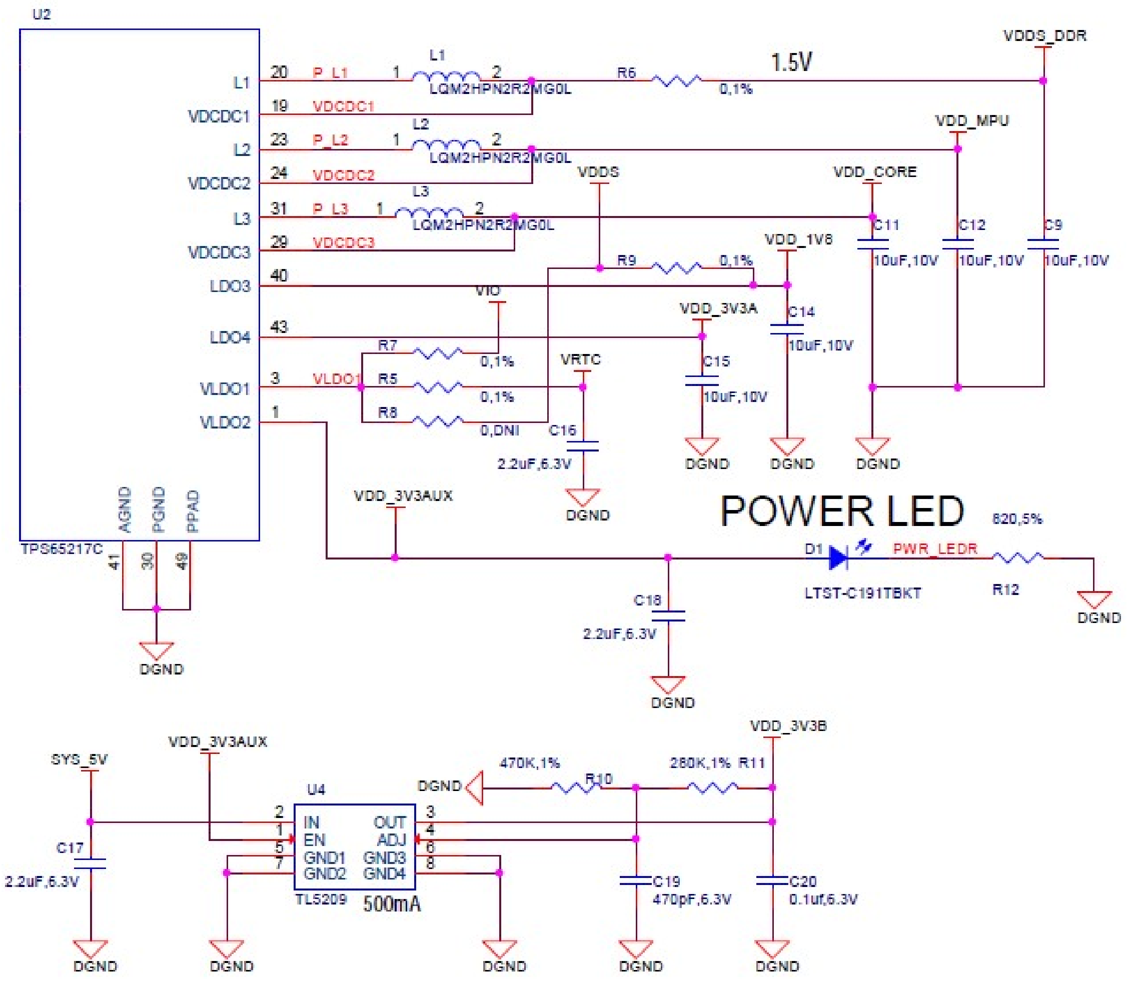


Fig. 2.25: Power Rails

The *VDD_MPU* rail can deliver up to 1.2A. This rail is not accessible for use anywhere else on the board and connects only to the processor. This rail defaults to 1.1V and can be scaled up to allow for higher frequency operation. Changing of the voltage is set via the I2C interface from the processor.

VDDS_DDR Rail

The *VDDS_DDR* rail defaults to**1.5V** to support the DDR3L rails and can deliver up to 1.2A. It is possible to adjust this voltage rail down to *1.35V* for lower power operation of the DDR3L device. Only DDR3L devices can support this voltage setting of 1.35V.

Power Sequencing

The power up process is consists of several stages and events. *figure-26* describes the events that make up the power up process for the processor from the PMIC. This diagram is used elsewhere to convey additional information. I saw no need to bust it up into smaller diagrams. It is from the processor datasheet supplied by Texas Instruments.

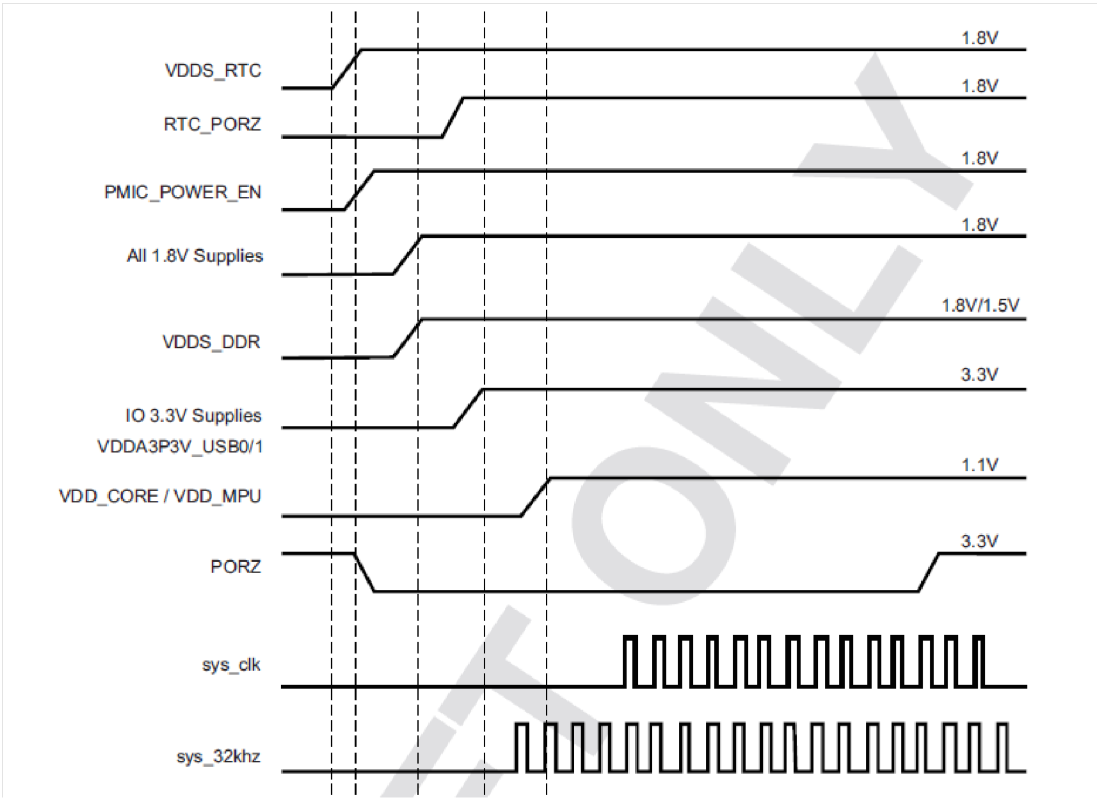


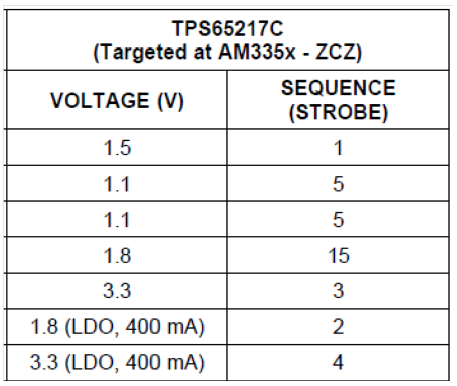
Fig. 2.26: Power Rail Power Up Sequencing

figure-27 the voltage rail sequencing for the **TPS65217C** as it powers up and the voltages on each rail. The power sequencing starts at 15 and then goes to one. That is the way the *TPS65217C* is configured. You can refer to the TPS65217C datasheet for more information.

Power LED The power LED is a blue LED that will turn on once the *TPS65217C* has finished the power up procedure. If you ever see the LED flash once, that means that the**TPS65217C** started the process and encountered an issue that caused it to shut down. The connection of the LED is shown in *figure-25*.

TPS65217C Power Up Process Figure below shows the interface between the **TPS65217C** and the processor. It is a cut from the PDF form of the schematic and reflects what is on the schematic.

When voltage is applied, DC or USB, the *TPS65217C* connects the power to the SYS output pin which drives the switchers and LDOs in the **TPS65217C**.



TPS65217C (Targeted at AM335x - ZCZ)	
VOLTAGE (V)	SEQUENCE (STROBE)
1.5	1
1.1	5
1.1	5
1.8	15
3.3	3
1.8 (LDO, 400 mA)	2
3.3 (LDO, 400 mA)	4

Fig. 2.27: TPS65217C Power Sequencing Timing

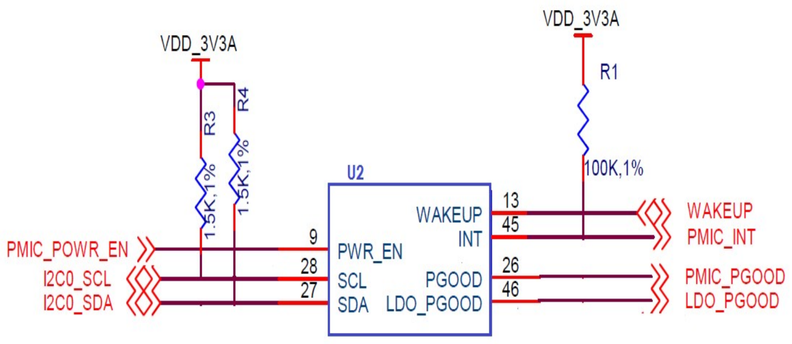


Fig. 2.28: Power Processor Interfaces

At power up all switchers and LDOs are off except for the *VRTC LDO* (1.8V), which provides power to the VRTC rail and controls the **RTC_PORZn** input pin to the processor, which starts the power up process of the processor. Once the RTC rail powers up, the *RTC_PORZn* pin, driven by the *LDO_PGOOD* signal from the *TPS65217C*, of the processor is released.

Once the *RTC_PORZn* reset is released, the processor starts the initialization process. After the RTC stabilizes, the processor launches the rest of the power up process by activating the **PMIC_POWER_EN** signal that is connected to the *TPS65217C* which starts the *TPS65217C* power up process.

The *LDO_PGOOD* signal is provided by the**TPS65217C** to the processor. As this signal is 1.8V from the *TPS65217C* by virtue of the *TPS65217C* VIO rail being set to 1.8V, and the *RTC_PORZ* signal on the processor is 3.3V, a voltage level shifter, *U4*, is used. Once the LDOs and switchers are up on the *TPS65217C*, this signal goes active releasing the processor. The LDOs on the *TPS65217C* are used to power the VRTC rail on the processor.

Processor Control Interface *figure-28* above shows two interfaces between the processor and the **TPS65217C** used for control after the power up sequence has completed.

The first is the *I2C0* bus. This allows the processor to turn on and off rails and to set the voltage levels of each regulator to supports such things as voltage scaling.

The second is the interrupt signal. This allows the *TPS65217C* to alert the processor when there is an event, such as when the power button is pressed. The interrupt is an open drain output which makes it easy to interface to 3.3V of the processor.

Low Power Mode Support This section covers three general power down modes that are available. These modes are only described from a Hardware perspective as it relates to the HW design.

RTC Only

In this mode all rails are turned off except the *VDD_RTC*. The processor will need to turn off all the rails to enter this mode. The **VDD_RTC** staying on will keep the RTC active and provide for the wakeup interfaces to be active to respond to a wake up event.

RTC Plus DDR

In this mode all rails are turned off except the *VDD_RTC* and the **VDDS_DDR**, which powers the DDR3L memory. The processor will need to turn off all the rails to enter this mode. The *VDD_RTC* staying on will keep the RTC active and provide for the wakeup interfaces to be active to respond to a wake up event.

The *VDDS_DDR* rail to the DDR3L is provided by the 1.5V rail of the **TPS65217C** and with *VDDS_DDR* active, the DDR3L can be placed in a self refresh mode by the processor prior to power down which allows the memory data to be saved.

Currently, this feature is not included in the standard software release. The plan is to include it in future releases.

Voltage Scaling

For a mode where the lowest power is possible without going to sleep, this mode allows the voltage on the ARM processor to be lowered along with slowing the processor frequency down. The I2C0 bus is used to control the voltage scaling function in the *TPS65217C*.

Sitara AM3358BZCZ100 Processor

The board is designed to use the Sitara AM3358BZCZ100 processor in the 15 x 15 package. Earlier revisions of the board used the XM3359AZCZ100 processor.

Description Figure below shows is a high level block diagram of the processor. For more information on the processor, go to http://www.ti.com/product/am3358

High Level Features

Table 2.5: Processor Features

Operating Systems	Linux, Android, Windows Embedded CE,QNX,ThreadX	**MMC/SD**	3
Standby Power	7 mW	**CAN**	2
ARM CPU	1 ARM Cortex-A8	**UART (SCI)**	6
ARM MHz (Max.)	275,500,600,800,1000	**ADC**	8-ch 12-bit
ARM MIPS (Max.)	1000,1200,2000	**PWM (Ch)**	3
Graphics Acceleration	1 3D	**eCAP**	3
Other Hardware Acceleration	2 PRU-ICSS,Crypto Accelerator	**eQEP**	3
On-Chip L1 Cache	64 KB (ARM Cortex-A8)	**RTC**	1
On-Chip L2 Cache	256 KB (ARM Cortex- A8)	**I2C**	3
Other On-Chip Memory	128 KB	**McASP**	2
Display Options	LCD	**SPI**	2
General Purpose Memory	1 16-bit (GPMC, NAND flash, NOR Flash, SRAM)	**DMA (Ch)**	64-Ch EDMA
DRAM	1 16-bit (LPDDR-400,DDR2-532, DDR3-400)	**IO Supply (V)**	1.8V(ADC), 3.3V
USB Ports	2	**Operating Temperature Range (C)**	40 to 90

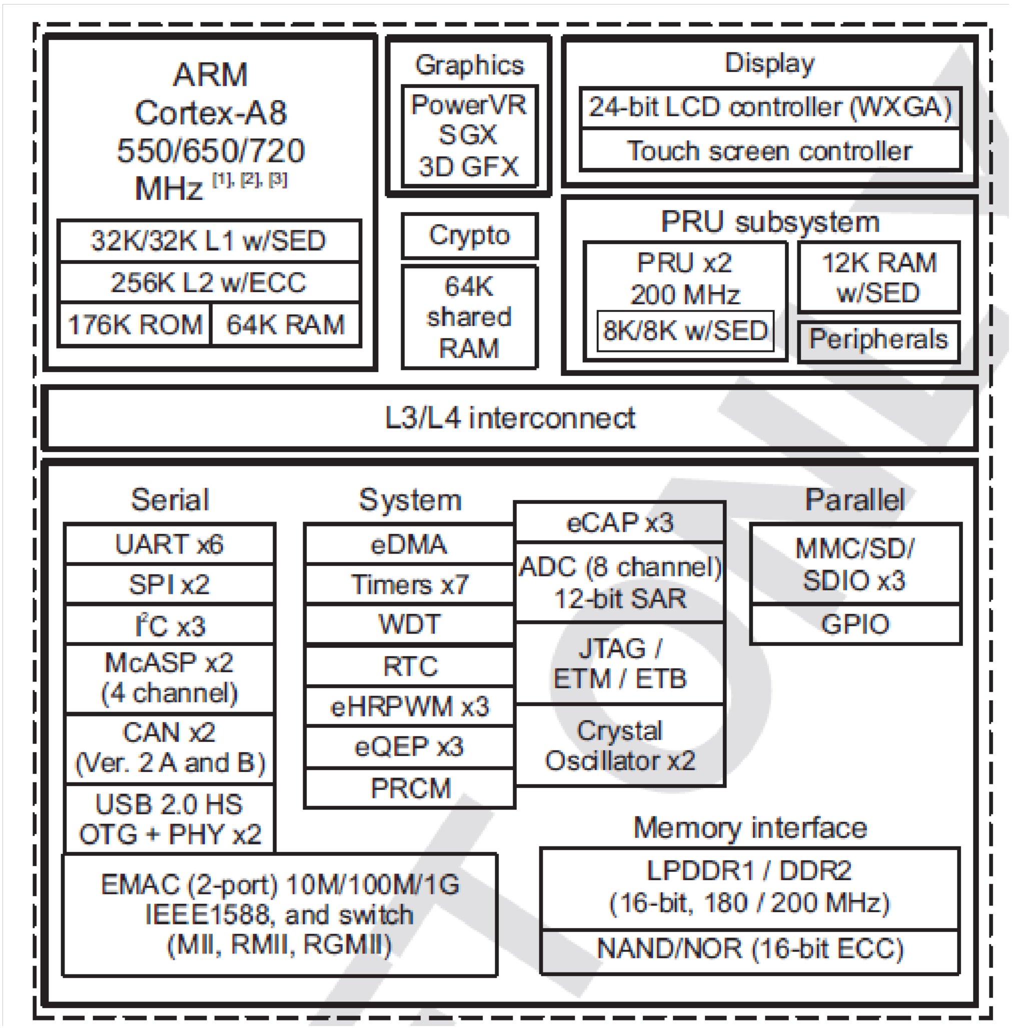


Fig. 2.29: Sitara AM3358BZCZ Block Diagram

Documentation Full documentation for the processor can be found on the TI website at http://www.ti.com/product/am3358 for the current processor used on the board. Make sure that you always use the latest datasheets and Technical Reference Manuals (TRM).

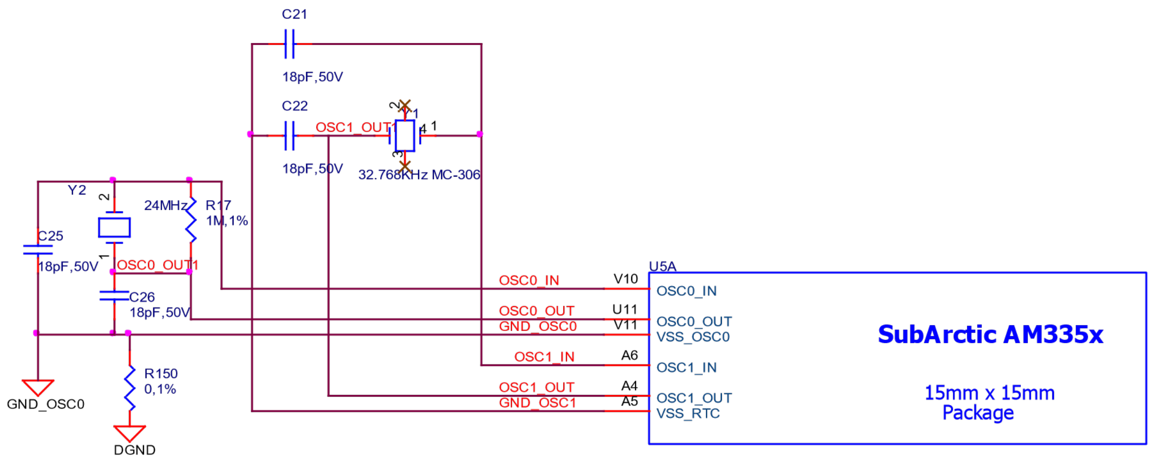


Fig. 2.30: Processor Crystals

Crystal Circuitry

Reset Circuitry *figure-31* is the board reset circuitry. The initial power on reset is generated by the **TPS65217C** power management IC. It also handles the reset for the Real Time Clock.

The board reset is the SYS_RESETn signal. This is connected to the NRESET_INOUT pin of the processor. This pin can act as an input or an output. When the reset button is pressed, it sends a warm reset to the processor and to the system.

On the revision A5D board, a change was made. On power up, the NRESET_INOUT signal can act as an output. In this instance it can cause the SYS_RESETn line to go high prematurely. In order to prevent this, the PORZn signal from the TPS65217C is connected to the SYS_RESETn line using an open drain buffer. These ensure that the line does not momentarily go high on power up.

This change is also in all revisions after A5D.

DDR3L Memory

The BeagleBone Black uses a single MT41K256M16HA-125 512MB DDR3L device from Micron that interfaces to the processor over 16 data lines, 16 address lines, and 14 control lines. On rev C we added the Kingston *KE4CN2H5A-A58* device as a source for the DDR3L device**.**

The following sections provide more details on the design.

Memory Device The design supports the standard DDR3 and DDR3L x16 devices and is built using the DDR3L. A single x16 device is used on the board and there is no support for two x8 devices. The DDR3 devices work at 1.5V and the DDR3L devices can work down to

1.35V to achieve lower power. The DDR3L comes in a 96-BALL FBGA package with 0.8 mil pitch. Other standard DDR3 devices can also be supported, but the DDR3L is the lower power device and was chosen for its ability to work at 1.5V or 1.35V. The standard frequency that the DDR3L is run at on the board is 400MHZ.

DDR3L Memory Design *figure-32* is the schematic for the DDR3L memory device. Each of the groups of signals is described in the following lines.

Address Lines: Provide the row address for ACTIVATE commands, and the column address and auto pre-charge bit (A10) for READ/WRITE commands, to select one location out of the memory array in the respective bank. A10 sampled during a PRECHARGE command determines whether the PRECHARGE applies to one bank

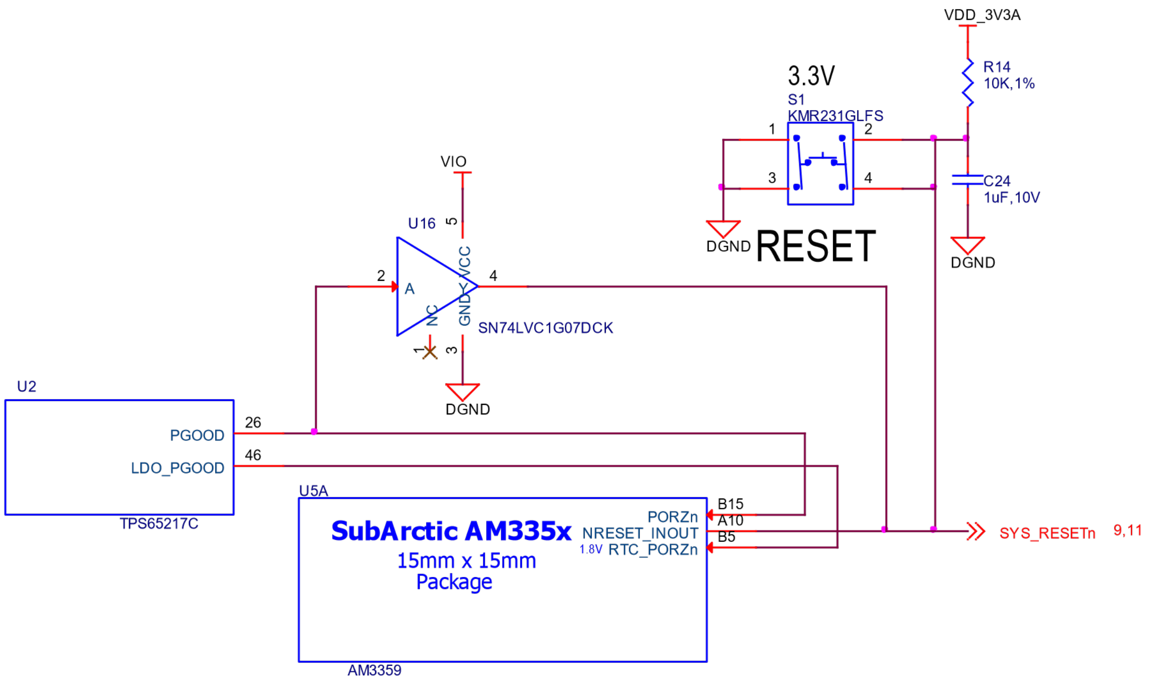


Fig. 2.31: Board Reset Circuitry

(A10 LOW, bank selected by BA[2:0]) or all banks (A10 HIGH). The address inputs also provide the op-code during a LOAD MODE command. Address inputs are referenced to VREFCA. A12/BC#: When enabled in the mode register (MR), A12 is sampled during READ and WRITE commands to determine whether burst chop (on-the-fly) will be performed (HIGH = BL8 or no burst chop, LOW = BC4 burst chop).

Bank Address Lines: BA[2:0] define the bank to which an ACTIVATE, READ, WRITE, or PRECHARGE command is being applied. BA[2:0] define which mode register (MR0, MR1, MR2, or MR3) is loaded during the LOAD MODE command. BA[2:0] are referenced to VREFCA.

CK and CK# Lines: are differential clock inputs. All address and control input signals are sampled on the crossing of the positive edge of CK and the negative edge of CK#. Output data strobe (DQS, DQS#) is referenced to the crossings of CK and CK#.

Clock Enable Line: CKE enables (registered HIGH) and disables (registered LOW) internal circuitry and clocks on the DRAM. The specific circuitry that is enabled/disabled is dependent upon the DDR3 SDRAM configuration and operating mode. Taking CKE LOW provides PRECHARGE power-down and SELF REFRESH operations (all banks idle) or active power-down (row active in any bank). CKE is synchronous for powerdown entry and exit and for self refresh entry. CKE is asynchronous for self refresh exit. Input buffers (excluding CK, CK#, CKE, RESET#, and ODT) are disabled during powerdown. Input buffers (excluding CKE and RESET#) are disabled during SELF REFRESH. CKE is referenced to VREFCA.

Chip Select Line: CS# enables (registered LOW) and disables (registered HIGH) the command decoder. All commands are masked when CS# is registered HIGH. CS# provides for external rank selection on systems with multiple ranks. CS# is considered part of the command code. CS# is referenced to VREFCA.

Input Data Mask Line: DM is an input mask signal for write data. Input data is masked when DM is sampled HIGH along with the input data during a write access. Although the DM ball is input-only, the DM loading is designed to match that of the DQ and DQS balls. DM is referenced to VREFDQ.

On-die Termination Line: ODT enables (registered HIGH) and disables (registered LOW) termination resistance internal to the DDR3L SDRAM. When enabled in normal operation, ODT is only applied to each of the following balls: DQ[7:0], DQS, DQS#, and DM for the x8; DQ[3:0], DQS, DQS#, and DM for the x4. The ODT input is ignored if disabled via the LOAD MODE command. ODT is referenced to VREFCA.

Power Rails The *DDR3L* memory device and the DDR3 rails on the processor are supplied by the**TPS65217C**. Default voltage is 1.5V but can be scaled down to 1.35V if desired.

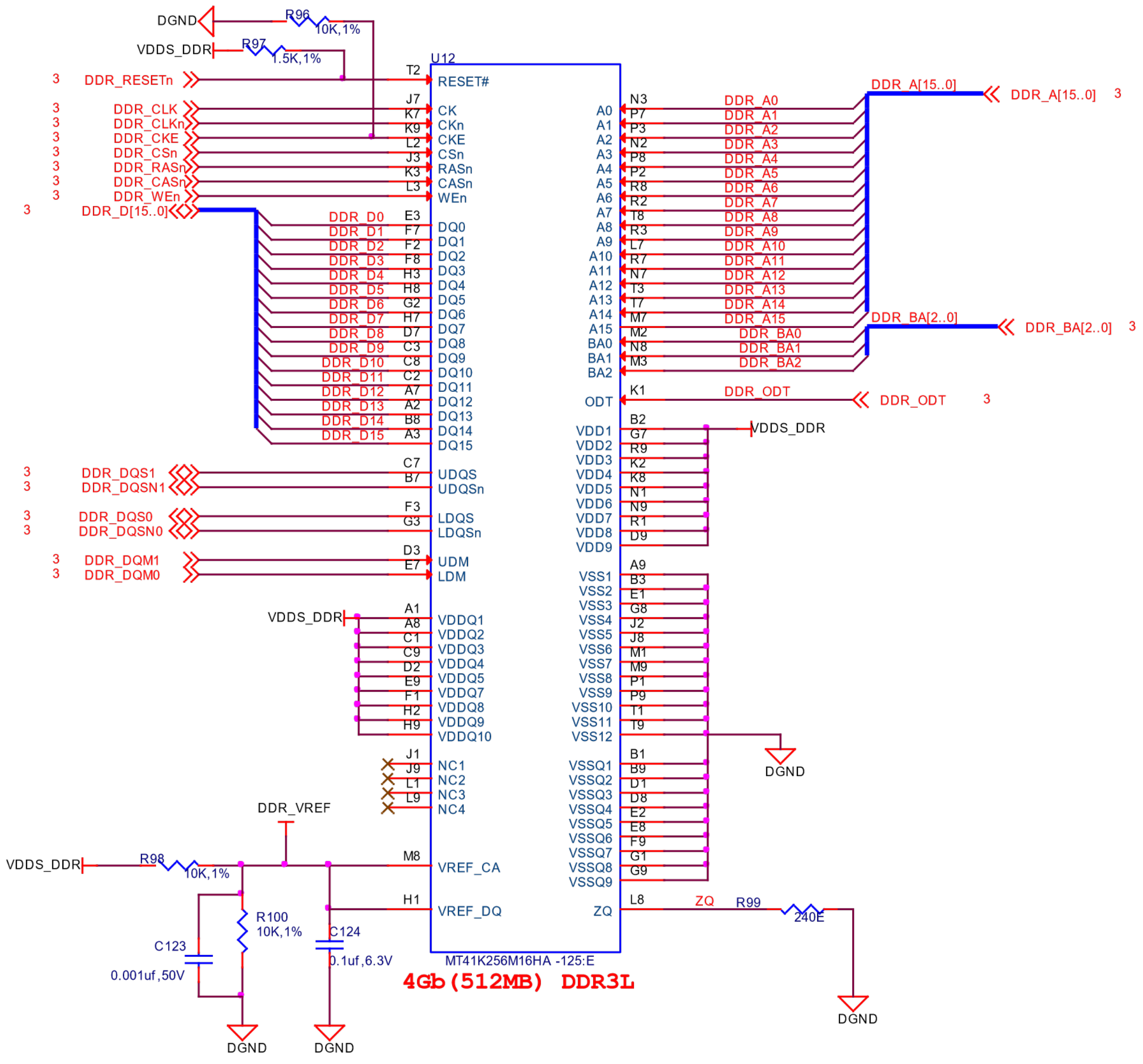


Fig. 2.32: DDR3L Memory Design

VREF The *VREF* signal is generated from a voltage divider on the**VDDS_DDR** rail that powers the processor DDR rail and the DDR3L device itself. *Figure 33* below shows the configuration of this signal and the connection to the DDR3L memory device and the processor.

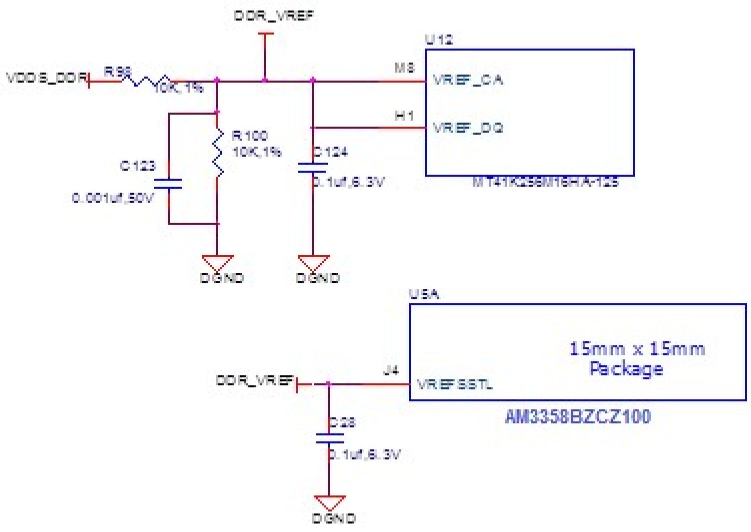


Fig. 2.33: DDR3L VREF Design

4GB eMMC Memory

The eMMC is a communication and mass data storage device that includes a Multi-MediaCard (MMC) interface, a NAND Flash component, and a controller on an advanced 11-signal bus, which is compliant with the MMC system specification. The nonvolatile eMMC draws no power to maintain stored data, delivers high performance across a wide range of operating temperatures, and resists shock and vibration disruption.

One of the issues faced with SD cards is that across the different brands and even within the same brand, performance can vary. Cards use different controllers and different memories, all of which can have bad locations that the controller handles. But the controllers may be optimized for reads or writes. You never know what you will be getting. This can lead to varying rates of performance. The eMMC card is a known controller and when coupled with the 8bit mode, 8 bits of data instead of 4, you get double the performance which should result in quicker boot times.

The following sections describe the design and device that is used on the board to implement this interface.

eMMC Device The device used is one of two different devices:

- Micron *MTFC4GLDEA 0M WT*
- Kingston *KE4CN2H5A-A58*

The package is a 153 ball WFBGA device on both devices.

eMMC Circuit Design *figure-34* is the design of the eMMC circuitry. The eMMC device is connected to the MMC1 port on the processor. MMC0 is still used for the microSD card as is currently done on the original BeagleBone. The size of the eMMC supplied is now 4GB.

The device runs at 3.3V both internally and the external I/O rails. The VCCI is an internal voltage rail to the device. The manufacturer recommends that a 1uF capacitor be attached to this rail, but a 2.2uF was chosen to provide a little margin.

Pullup resistors are used to increase the rise time on the signals to compensate for any capacitance on the board.

The pins used by the eMMC1 in the boot mode are listed below in *Table 6*.

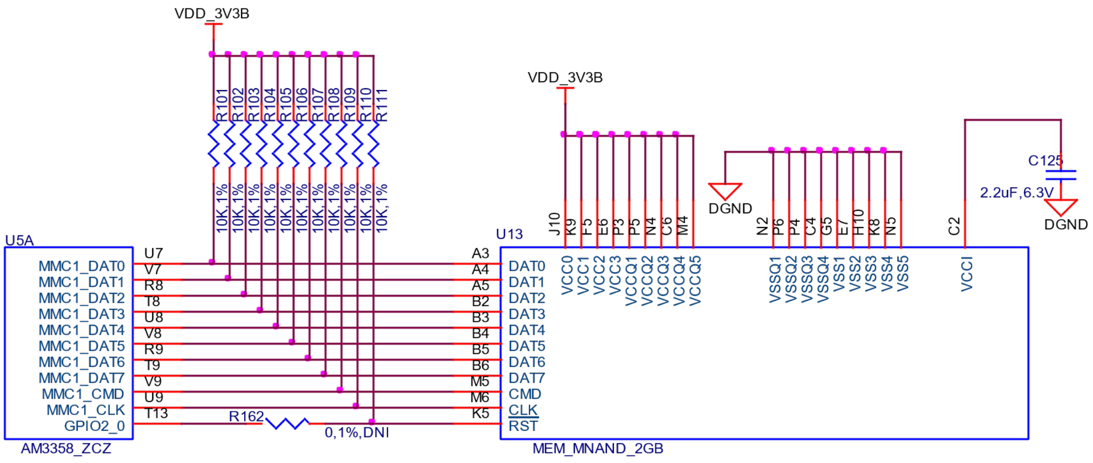
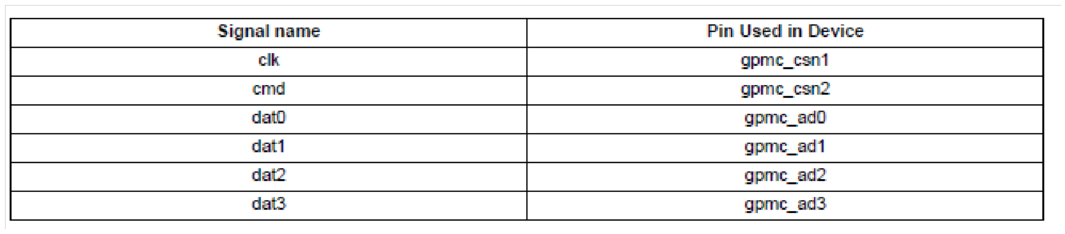


Fig. 2.34: eMMC Memory Design



Signal name	Pin Used in Device
clk	gpmc_csn1
cmd	gpmc_csn2
dat0	gpmc_ad0
dat1	gpmc_ad1
dat2	gpmc_ad2
dat3	gpmc_ad3

Fig. 2.35: eMMC Boot Pins

For eMMC devices the ROM will only support raw mode. The ROM Code reads out raw sectors from image or the booting file within the file system and boots from it. In raw mode the booting image can be located at one of the four consecutive locations in the main area: offset 0x0 / 0x20000 (128 KB) / 0x40000 (256 KB) / 0x60000 (384 KB). For this reason, a booting image shall not exceed 128KB in size. However it is possible to flash a device with an image greater than 128KB starting at one of the aforementioned locations. Therefore the ROM Code does not check the image size. The only drawback is that the image will cross the subsequent image boundary. The raw mode is detected by reading sectors #0, #256, #512, #768. The content of these sectors is then verified for presence of a TOC structure. In the case of a *GP Device*, a Configuration Header (CH)*must* be located in the first sector followed by a *GP header*. The CH might be void (only containing a CHSETTINGS item for which the Valid field is zero).

The ROM only supports the 4-bit mode. After the initial boot, the switch can be made to 8-bit mode for increasing the overall performance of the eMMC interface.

Board ID EEPROM

The BeagleBone is equipped with a single 32Kbit(4KB) 24LC32AT-I/OT EEPROM to allow the SW to identify the board. *Table 7* below defined the contents of the EEPROM.

Table 2.6: EEPROM Contents

Name	Size (bytes)	Contents
Header	4	0xAA, 0x55, 0x33, EE
Board Name	8	Name for board in ASCII: **A335BNLT**
Version	4	Hardware version code for board in ASCII: **00A3 for Rev A3, 00A4 for Rev A4, 00A5 for Rev A5, 00A6 for Rev A6,00B0 for Rev B, and 00C0 for Rev C.**

continues on next page

Table 2.6 – continued from previous page

Name	Size (bytes)	Contents
Serial Number	12	Serial number of the board. This is a 12 character string which is: **WWYY4P16nnnn** where, WW = 2 digit week of the year of production YY = 2 digit year of production BBBK = BeagleBone Black nnnn = incrementing board number
Configuration Option	32	Codes to show the configuration setup on this board. **All FF**
RSVD	6	FF FF FF FF FF FF
RSVD	6	FF FF FF FF FF FF
RSVD	6	FF FF FF FF FF FF
Available	4018	Available space for other non-volatile codes/data

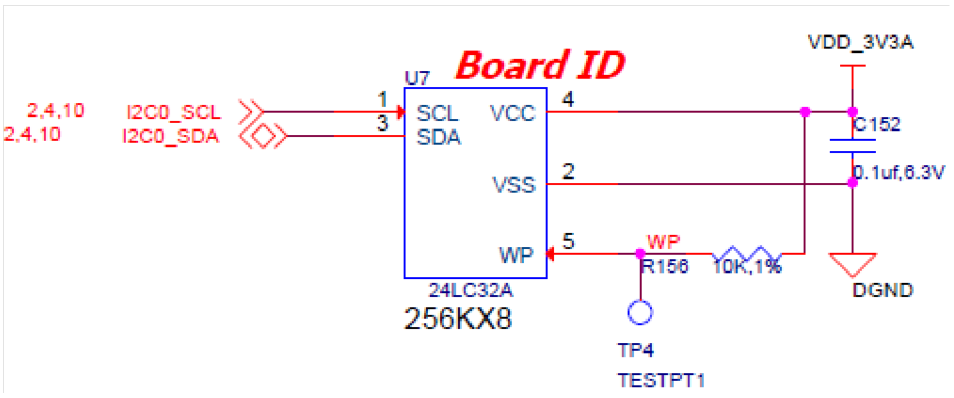


Fig. 2.36: EEPROM Design Rev A5

The EEPROM is accessed by the processor using the I2C 0 bus. The *WP* pin is enabled by default. By grounding the test point, the write protection is removed.

The first 48 locations should not be written to if you choose to use the extras storage space in the EEPROM for other purposes. If you do, it could prevent the board from booting properly as the SW uses this information to determine how to set up the board.

Micro Secure Digital

The microSD connector on the board will support a microSD card that can be used for booting or file storage on the BeagleBone Black.

microSD Design The signals *MMC0-3* are the data lines for the transfer of data between the processor and the microSD connector.

The *MMC0_CLK* signal clocks the data in and out of the microSD card.

The *MMCO_CMD* signal indicates that a command versus data is being sent.

There is no separate card detect pin in the microSD specification. It uses *MMCO_DAT3* for that function. However, most microSD connectors still supply a CD function on the connectors. In the BeagleBone Black design, this pin is connected to the **MMC0_SDCD** pin for use by the processor. You can also change the pin to *GPIO0_6*, which is able to wake up the processor from a sleep mode when an microSD card is inserted into the connector.

Pullup resistors are provided on the signals to increase the rise times of the signals to overcome PCB capacitance.

Power is provided from the *VDD_3V3B* rail and a 10uF capacitor is provided for filtering.

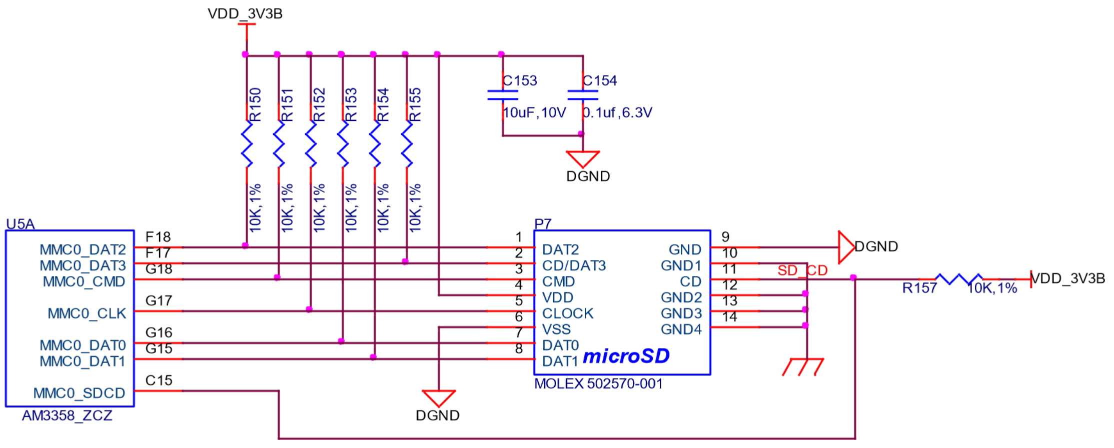


Fig. 2.37: microSD Design

6.6 User LEDs

There are four user LEDs on the BeagleBone Black. These are connected to GPIO pins on the processor. *Figure 37* shows the interfaces for the user LEDs.

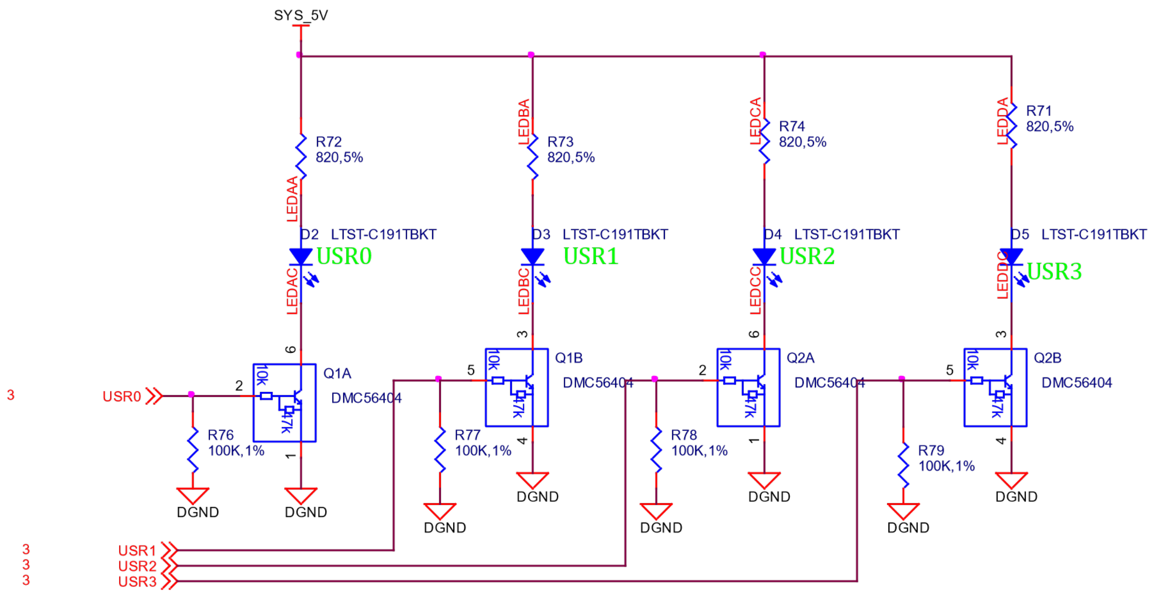


Fig. 2.38: User LEDs

Resistors R71-R74 were changed to 4.75K on the revision A5B and later boards.

Table 2.7: User LED Control Signals/Pins

LED	GPIO SIGNAL	PROC PIN
USR0	GPIO1_21	V15
USR1	GPIO1_22	U15
USR2	GPIO1_23	T15
USR3	GPIO1_24	V16

A logic level of "1" will cause the LEDs to turn on.

Boot Configuration

The design supports two groups of boot options on the board. The user can switch between these modes via the Boot button. The primary boot source is the onboard eMMC device. By holding the Boot button, the user can force the board to boot from the microSD slot. This enables the eMMC to be overwritten when needed or to just boot an alternate image. The following sections describe how the boot configuration works.

In most applications, including those that use the provided demo distributions available from beagleboard.org the processor-external boot code is composed of two stages. After the primary boot code in the processor ROM passes control, a secondary stage (secondary program loader – "SPL" or "MLO") takes over. The SPL stage initializes only the required devices to continue the boot process, and then control is transferred to the third stage "U-boot". Based on the settings of the boot pins, the ROM knows where to go and get the SPL and UBoot code. In the case of the BeagleBone Black, that is either eMMC or microSD based on the position of the boot switch.

Boot Configuration Design *figure-38* shows the circuitry that is involved in the boot configuration process. On power up, these pins are read by the processor to determine the boot order. S2 is used to change the level of one bit from HI to LO which changes the boot order.

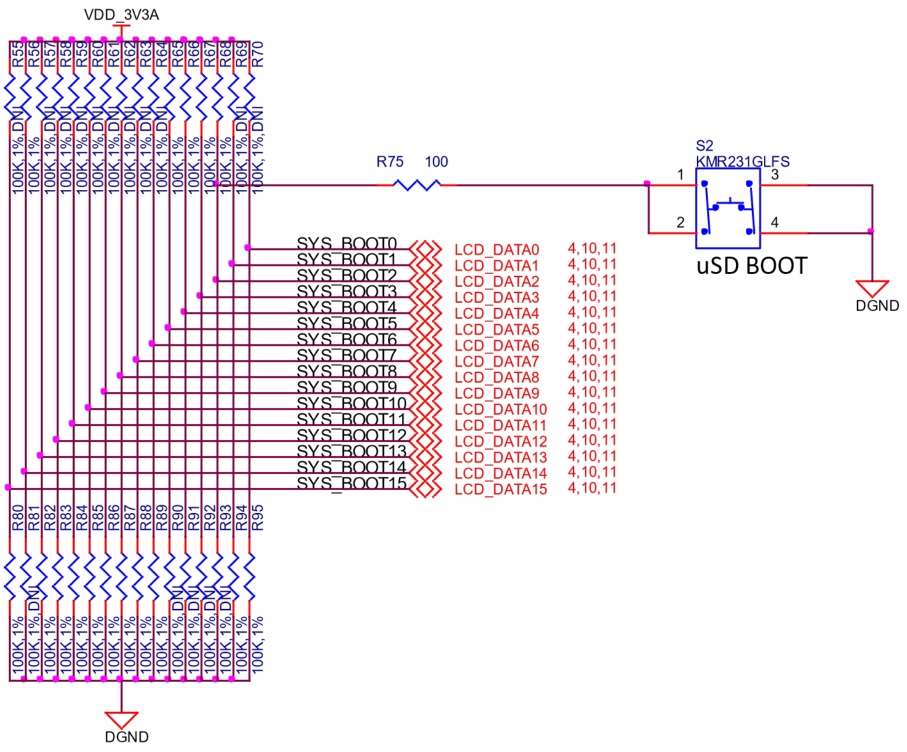

VDD_3V3A
R55 R56 R57 R58 R59 R60 R61 R62 R63 R64 R65 R66 R67 R68 R69 R70
100K,1%,DNI
100K,1%
R75 100
S2
KMR231GLFS
uSD BOOT
DGND
SYS_BOOT0 … SYS_BOOT15
LCD_DATA0 … LCD_DATA15
4,10,11
R80 R81 R82 R83 R84 R85 R86 R87 R88 R89 R90 R91 R92 R93 R94 R95
DGND

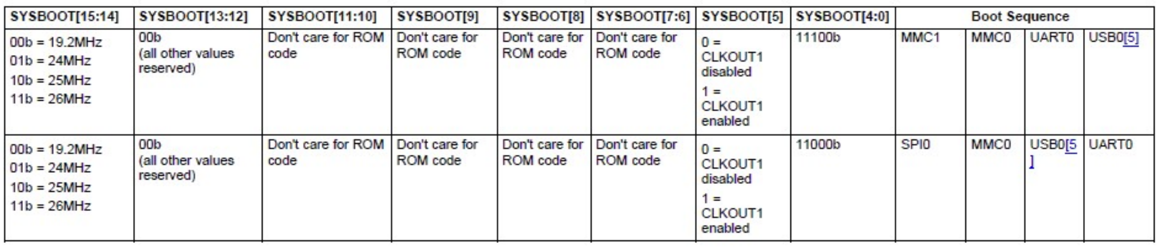
Fig. 2.39: Processor Boot Configuration Design

It is possible to override these setting via the expansion headers. But be careful not to add too much load such that it could interfere with the operation of the HDMI interface or LCD panels. If you choose to override these settings, it is strongly recommended that you gate these signals with the *SYS_RESETn* signal. This ensures that after coming out of reset these signals are removed from the expansion pins.

Default Boot Options

Based on the selected option found in *figure-39* below, each of the boot sequences for each of the two settings is shown.

The first row in «figure-39» is the default setting. On boot, the processor will look for the eMMC on the MMC1 port first, followed by the microSD slot on MMC0, USB0 and UART0. In the event there is no microSD card and the eMMC is empty, UART0 or USB0 could be used as the board source.



SYSBOOT[15:14]	SYSBOOT[13:12]	SYSBOOT[11:10]	SYSBOOT[9]	SYSBOOT[8]	SYSBOOT[7:6]	SYSBOOT[5]	SYSBOOT[4:0]	Boot Sequence			
00b = 19.2MHz 01b = 24MHz 10b = 25MHz 11b = 26MHz	00b (all other values reserved)	Don't care for ROM code	Don't care for ROM code	Don't care for ROM code	Don't care for ROM code	0 = CLKOUT1 disabled 1 = CLKOUT1 enabled	11100b	MMC1	MMC0	UART0	USB0[5]
00b = 19.2MHz 01b = 24MHz 10b = 25MHz 11b = 26MHz	00b (all other values reserved)	Don't care for ROM code	Don't care for ROM code	Don't care for ROM code	Don't care for ROM code	0 = CLKOUT1 disabled 1 = CLKOUT1 enabled	11000b	SPI0	MMC0	USB0[5]	UART0

Fig. 2.40: Processor Boot Configuration

If you have a microSD card from which you need to boot from, hold the boot button down. On boot, the processor will look for the SPIO0 port first, then microSD on the MMC0 port, followed by USB0 and UART0. In the event there is no microSD card and the eMMC is empty, USB0 or UART0 could be used as the board source.

10/100 Ethernet

The BeagleBone Black is equipped with a 10/100 Ethernet interface. It uses the same PHY as is used on the original BeagleBone. The design is described in the following sections.

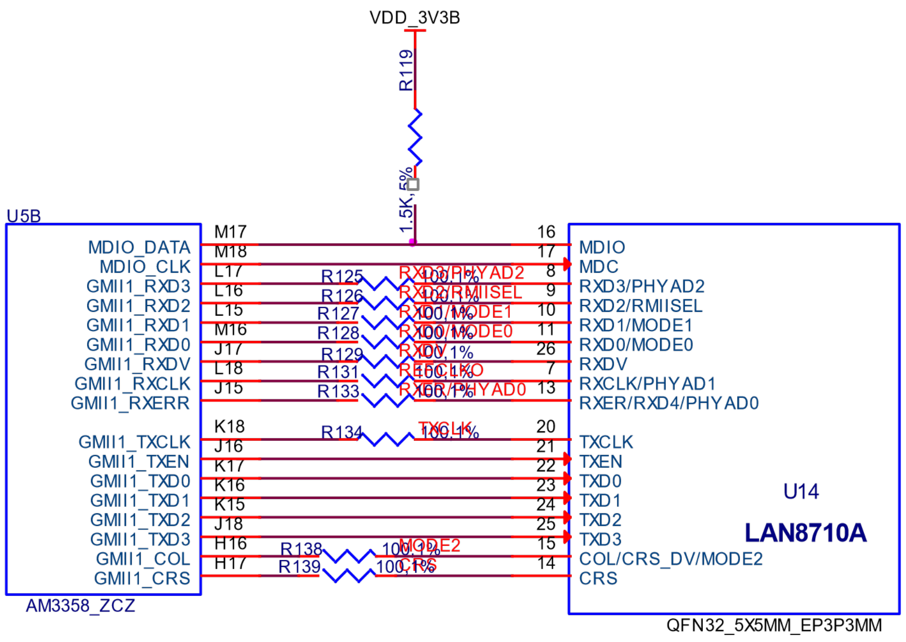


Fig. 2.41: Ethernet Processor Interface

6.9.1 Ethernet Processor Interface This is the same interface as is used on the BeagleBone. No changes were made in this design for the board.

Ethernet Connector Interface The off board side of the PHY connections are shown in *Figure 41* below.

This is the same interface as is used on the BeagleBone. No changes were made in this design for the board.

Ethernet PHY Power, Reset, and Clocks **VDD_3V3B Rail**

The VDD_3V3B rail is the main power rail for the *LAN8710A*. It originates at the VD_3V3B regulator and is the primary rail that supports all of the peripherals on the board. This rail also supplies the VDDIO rails which set the voltage levels for all of the I/O signals between the processor and the**LAN8710A**.

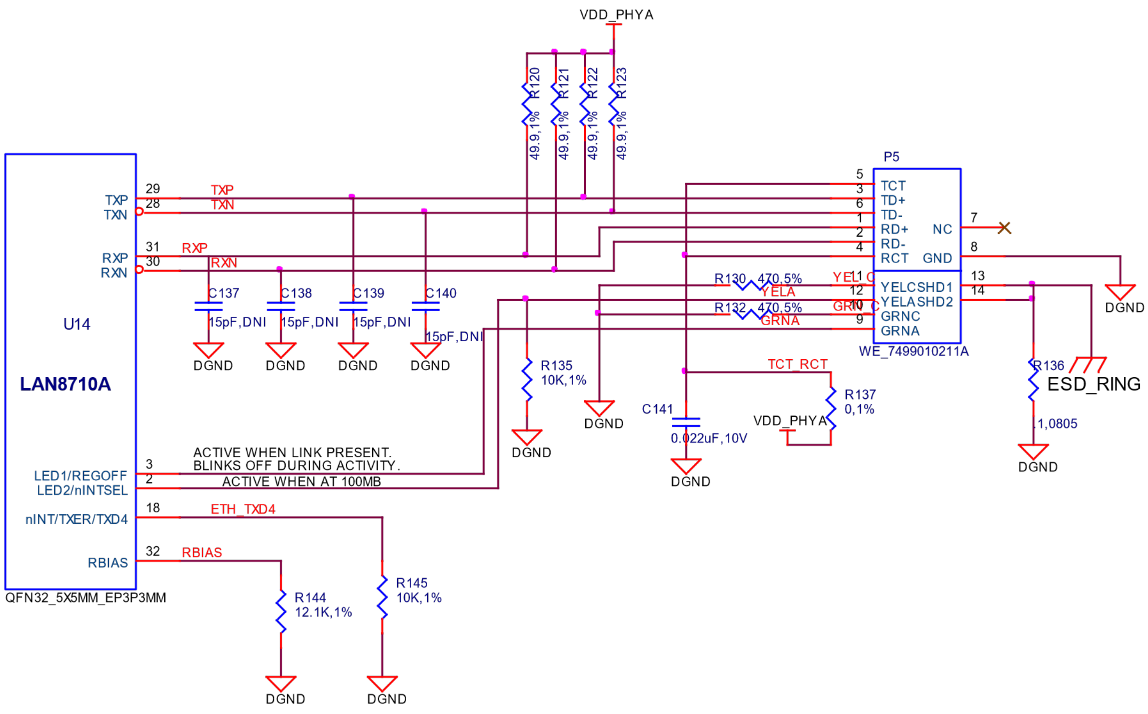


Fig. 2.42: Ethernet Connector Interface

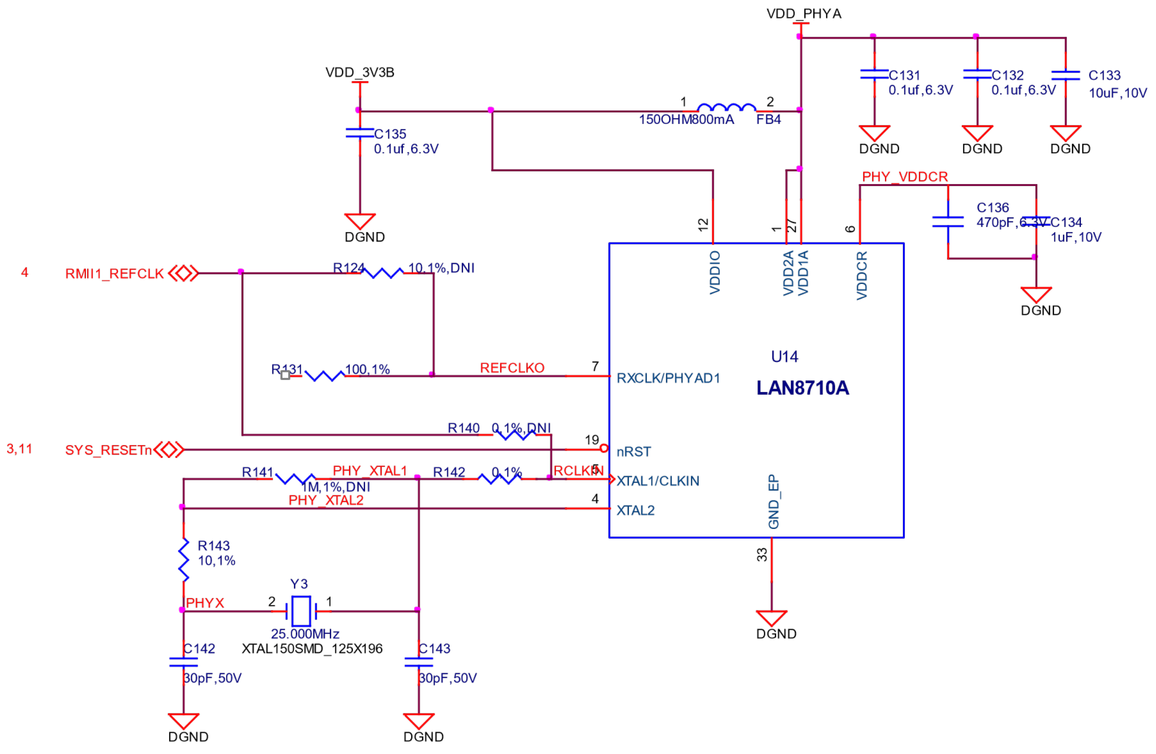


Fig. 2.43: Ethernet PHY, Power, Reset, and Clocks

VDD_PHYA Rail

A filtered version of VDD_3V3B rail is connected to the VDD rails of the LAN8710 and the termination resistors on the Ethernet signals. It is labeled as *VDD_PHYA*. The filtering inductor helps block transients that may be seen on the VDD_3V3B rail.

PHY_VDDCR Rail

The *PHY_VDDCR* rail originates inside the LAN8710A. Filter and bypass capacitors are used to filter the rail. Only circuitry inside the LAN8710A uses this rail.

SYS_RESET

The reset of the LAN8710A is controlled via the *SYS_RESETn* signal, the main board reset line.

Clock Signals

A crystal is used to create the clock for the LAN8710A. The processor uses the *RMII_RXCLK* signal to provide the clocking for the data between the processor and the LAN8710A.

LAN8710A Mode Pins

There are mode pins on the LAN8710A that sets the operational mode for the PHY when coming out of reset. These signals are also used to communicate between the processor and the LAN8710A. As a result, these signals can be driven by the processor which can cause the PHY not to be initialized correctly. To ensure that this does not happen, three low value pull up resistors are used. *Figure 43* below shows the three mode pin resistors.

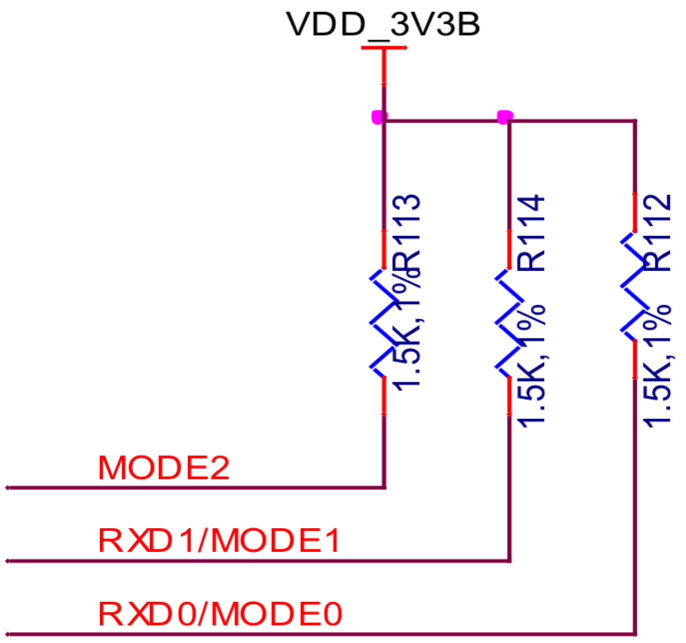


Fig. 2.44: Ethernet PHY Mode Pins

This will set the mode to be 111, which enables all modes and enables auto-negotiation.

HDMI Interface

The BeagleBone Black has an onboard HDMI framer that converts the LCD signals and audio signals to drive a HDMI monitor. The design uses an NXP *TDA19988* HDMI Framer.

The following sections provide more detail into the design of this interface.

Supported Resolutions The maximum resolution supported by the BeagleBone Black is 1280x1024 @ 60Hz. *Table 9* below shows the supported resolutions. Not all resolutions may work on all monitors, but these have been tested and shown to work on at least one monitor. EDID is supported on the BeagleBone Black. Based on the EDID reading from the connected monitor, the highest compatible resolution is selected.

Table 2.8: HDMI Supported Monitor Resolutions

RESOLUTION	AUDIO
800 x 600 @60Hz	
800 x 600 @56Hz	
640 x 480 @75Hz	
640 x 480 @60Hz	YES
720 x 400 @70Hz	
1280 x 1024 @75Hz	
1024 x 768 @75Hz	
1024 x 768 @70Hz	
1024 x 768 @60Hz	
800 x 600 @75Hz	
800 x 600 @72Hz	
720 x 480 @60Hz	YES
1280 x 720 @60Hz	YES
1920 x 1080 @24Hz	YES

NOTE: The updated software image used on the Rev A5B and later boards added support for 1920x1080@24HZ.

Audio is limited to CEA supported resolutions. LCD panels only activate the audio in CEA modes. This is a function of the specification and is not something that can be fixed on the board via a hardware change or a software change.

HDMI Framer The *TDA19988* is a High-Definition Multimedia Interface (HDMI) 1.4a transmitter. It is backward compatible with DVI 1.0 and can be connected to any DVI 1.0 or HDMI sink. The HDCP mode is not used in the design. The non-HDCP version of the device is used in the BeagleBone Black design.

This device provides additional embedded features like CEC (Consumer Electronic Control). CEC is a single bidirectional bus that transmits CEC over the home appliance network connected through this bus. This eliminates the need of any additional device to handle this feature. While this feature is supported in this device, as of this point, the SW to support this feature has not been implemented and is not a feature that is considered critical. It can be switched to very low power Standby or Sleep modes to save power when HDMI is not used. *TDA19988* embeds I~2~C-bus master interface for DDC-bus communication to read EDID. This device can be controlled or configured via I~2~C-bus interface.

HDMI Video Processor Interface The *Figure 44* shows the connections between the processor and the HDMI framer device. There are 16 bits of display data, 5-6-5 that is used to drive the framer. The reason for 16 bits is that allows for compatibility with display and LCD capes already available on the original BeagleBone. The unused bits on the **TDA19988** are tied low. In addition to the data signals are the VSYNC, HSYNC, DE, and PCLK signals that round out the video interface from the processor.

HDMI Control Processor Interface In order to use the *TDA19988*, the processor needs to setup the device. This is done via the I2C interface between the processor and the **TDA19988**. There are two signals on the *TDA19988* that could be used to set the address of the *TDA19988*. In this design they are both tied low. The I2C interface supports both 400kHz and 100KhZ operation. *Table 10* shows the I2C address.

Interrupt Signal There is a HDMI_INT signal that connects from the TDA19988 to the processor. This signal can be used to alert the processor in a state change on the HDMI interface.

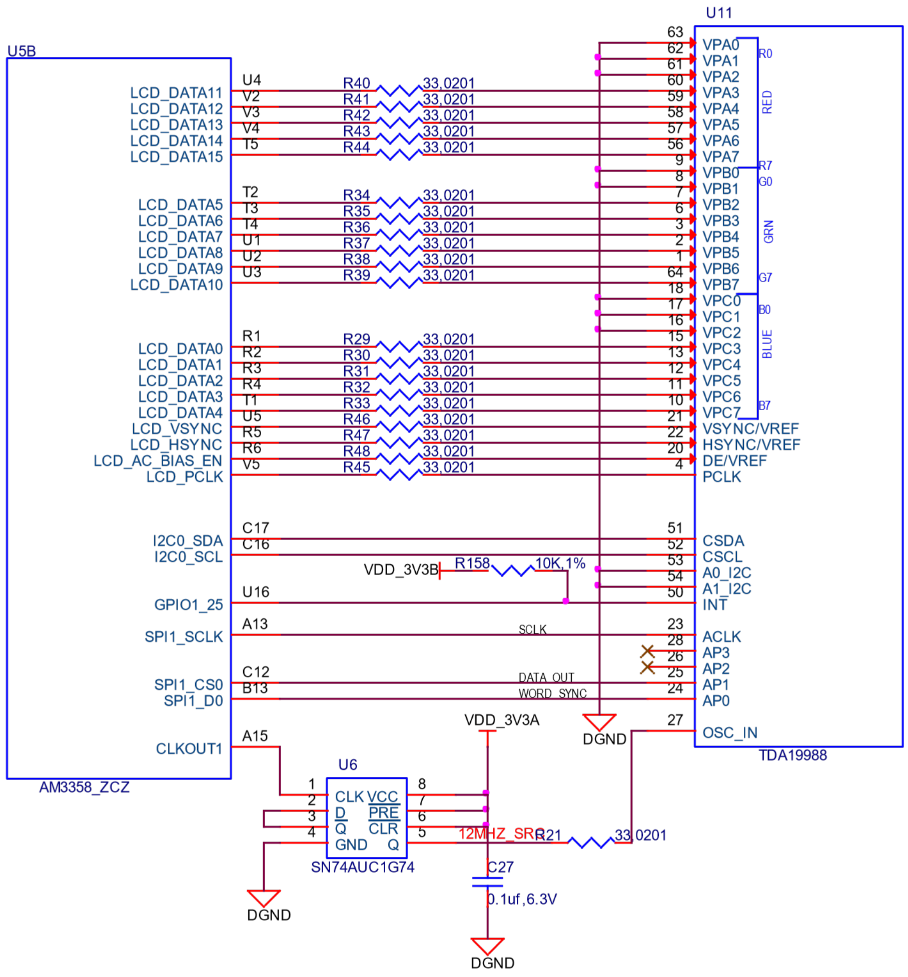
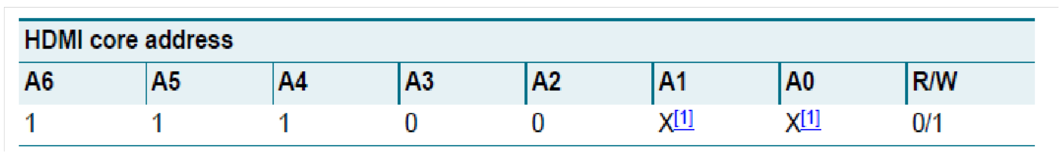


Fig. 2.45: HDMI Framer Processor Interface



HDMI core address							
A6	A5	A4	A3	A2	A1	A0	R/W
1	1	1	0	0	X[1]	X[1]	0/1

Fig. 2.46: TDA19988 I2C Address

Audio Interface There is an I2S audio interface between the processor and the *TDA19988*. Stereo audio can be transported over the HDMI interface to an audio equipped display. In order to create the required clock frequencies, an external 24.576MHz oscillator,*Y4*, is used. From this clock, the processor generates the required clock frequencies for the *TDA19988*.

There are three signals used to pass data from the processor to the *TDA19988*. SCLK is the serial clock. SPI1_CS0 is the data pin to the **TDA199888**. SPI1_D0 is the word sync pin. These signals are configured as I2S interfaces.

Audio is limited to CEA supported resolutions. LCD panels only activate the audio in CEA modes. This is a function of the specification and is not something that can be fixed on the board via a hardware change or a software change.

In order to create the correct clock frequencies, we had to add an external *24.576MHz* oscillator. Unfortunately this had to be input into the processor using the pin previously used for **GPIO3_21**. In order to keep GPIO3_21 functionality, we provided a way to disable the oscillator if the need was there to use the pin on the expansion header. *Figure 45* shows the oscillator circuitry.

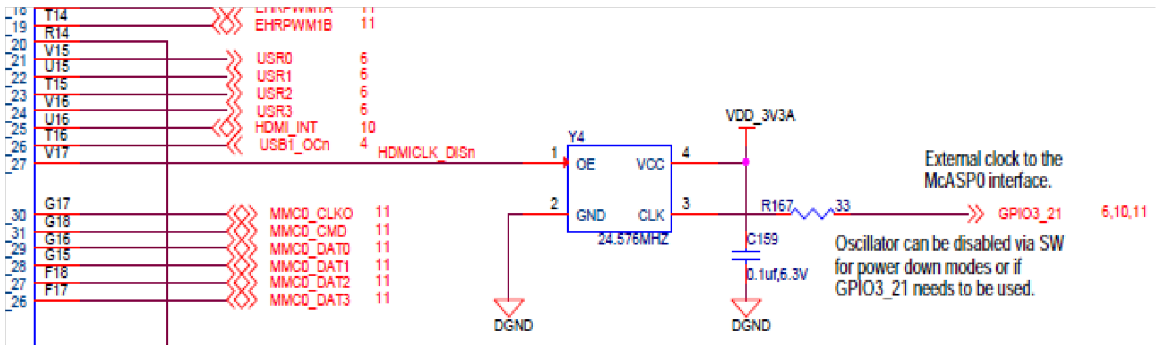


Fig. 2.47: 24.576MHZ Oscillator

Power Connections *figure-46* shows the power connections to the **TDA19988** device. All voltage rails for the device are at 1.8V. A filter is provided to minimize any noise from the 1.8V rail getting back into the device.

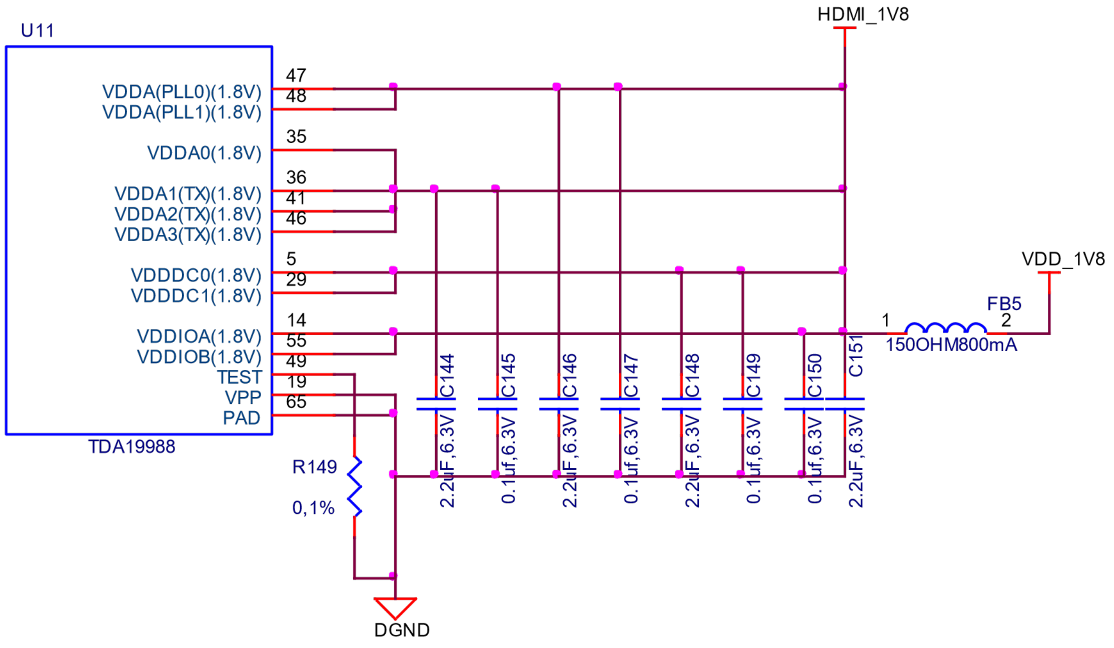


Fig. 2.48: HDMI Power Connections

All of the interfaces between the processor and the *TDA19988* are 3.3V tolerant allowing for direct connection.

HDMI Connector Interface *figure-47* shows the design of the interface between the HDMI Framer and the connector.

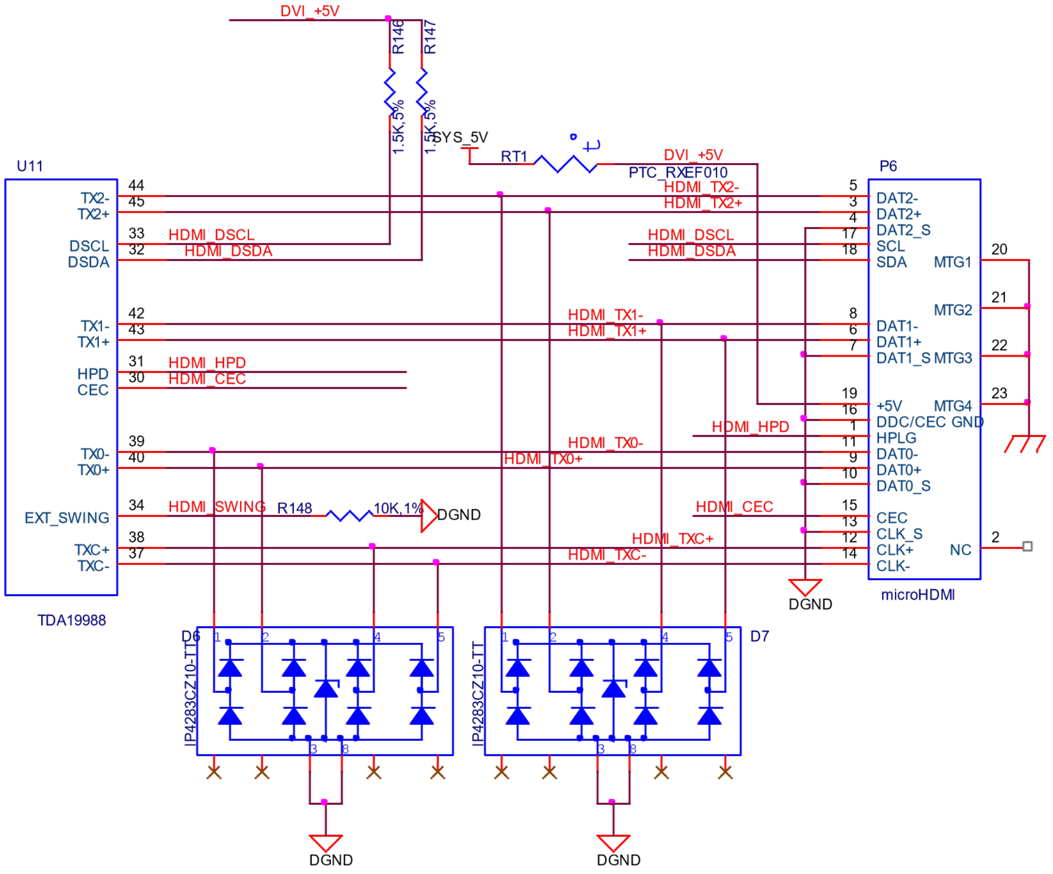


Fig. 2.49: Connector Interface Circuitry

The connector for the HDMI interface is a microHDMI. It should be noted that this connector has a different pinout than the standard or mini HDMI connectors. D6 and D7 are ESD protection devices.

USB Host

The board is equipped with a single USB host interface accessible from a single USB Type A female connector. «figure-48» is the design of the USB Host circuitry.

Power Switch *U8* is a switch that allows the power to the connector to be turned on or off by the processor. It also has an over current detection that can alert the processor if the current gets too high via the**USB1_OC** signal. The power is controlled by the *USB1_DRVBUS* signal from the processor.

ESD Protection *U9* is the ESD protection for the signals that go to the connector.

Filter Options *FB7* and**FB8** were added to assist in passing the FCC emissions test. The *USB1_VBUS* signal is used by the processor to detect that the 5V is present on the connector. *FB7* is populated and *FB8* is replaced with a .1 ohm resistor.

PRU-ICSS

The PRU-ICSS module is located inside the AM3358 processor. Access to these pins is provided by the expansion headers and is multiplexed with other functions on the board. Access is not provided to all of the available

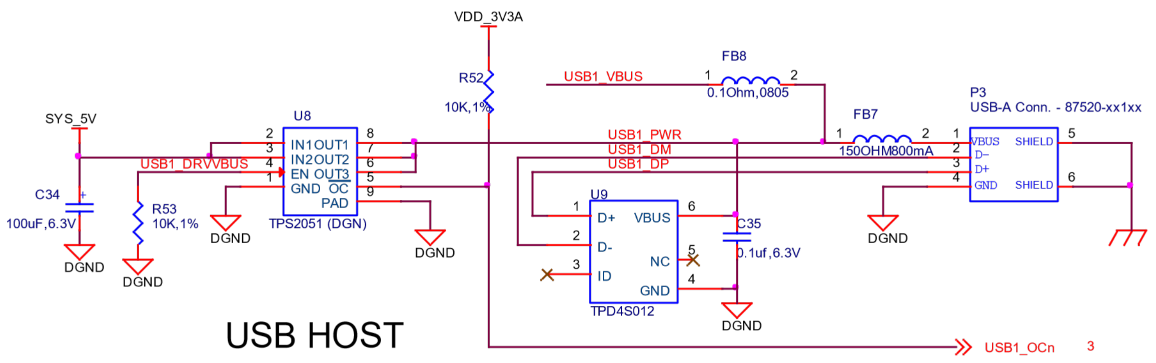


Figure 48. USB Host Circuitry

Fig. 2.50: USB Host circuit

pins.

All documentation is located at http://github.com/beagleboard/am335x_pru_package_

This feature is not supported by Texas Instruments.

PRU-ICSS Features The features of the PRU-ICSS include:

Two independent programmable real-time (PRU) cores:

- 32-Bit Load/Store RISC architecture
- 8K Byte instruction RAM (2K instructions) per core
- 8K Bytes data RAM per core
- 12K Bytes shared RAM
- Operating frequency of 200 MHz
- PRU operation is little endian similar to ARM processor
- All memories within PRU-ICSS support parity
- Includes Interrupt Controller for system event handling
- Fast I/O interface

16 input pins and 16 output pins per PRU core. *(Not all of these are accessible on the BeagleBone Black).*

PRU-ICSS Block Diagram

PRU-ICSS Pin Access Both PRU 0 and PRU1 are accessible from the expansion headers. Some may not be usable without first disabling functions on the board like LCD for example. Listed below is what ports can be accessed on each PRU.

- 8 outputs or 9 inputs
- 13 outputs or 14 inputs
- UART0_TXD, UART0_RXD, UART0_CTS, UART0_RTS

Table 2.9: P8 PRU0 and PRU1 Access

PIN	PROC	NAME			
11	R12	GPIO1_13		pr1_pru0_pru_r30_15 (Output)	

continues on next page

Table 2.9 – continued from previous page

PIN	PROC	NAME			
12	T12	GPIO1_12		pr1_pru0_pru_r30_14 (Output)	
15	U13	GPIO1_15		pr1_pru0_pru_r31_15 (Input)	
16	V13	GPIO1_14		pr1_pru0_pru_r31_14 (Input)	
20	V9	GPIO1_31	pr1_pru1_pru_r30_13 (Output)	pr1_pru1_pru_r31_13 (INPUT)	
21	U9	GPIO1_30	pr1_pru1_pru_r30_12 (Output)	pr1_pru1_pru_r31_12 (INPUT)	
27	U5	GPIO2_22	pr1_pru1_pru_r30_8 (Output)	pr1_pru1_pru_r31_8 (INPUT)	
28	V5	GPIO2_24	pr1_pru1_pru_r30_10 (Output)	pr1_pru1_pru_r31_10 (INPUT)	
29	R5	GPIO2_23	pr1_pru1_pru_r30_9 (Output)	pr1_pru1_pru_r31_9 (INPUT)	
39	T3	GPIO2_12	pr1_pru1_pru_r30_6 (Output)	pr1_pru1_pru_r31_6 (INPUT)	
40	T4	GPIO2_13	pr1_pru1_pru_r30_7 (Output)	pr1_pru1_pru_r31_7 (INPUT)	
41	T1	GPIO2_10	pr1_pru1_pru_r30_4 (Output)	pr1_pru1_pru_r31_4 (INPUT)	
42	T2	GPIO2_11	pr1_pru1_pru_r30_5 (Output)	pr1_pru1_pru_r31_5 (INPUT)	
43	R3	GPIO2_8	pr1_pru1_pru_r30_2 (Output)	pr1_pru1_pru_r31_2 (INPUT)	
44	R4	GPIO2_9	pr1_pru1_pru_r30_3 (Output)	pr1_pru1_pru_r31_3 (INPUT)	
45	R1	GPIO2_6	pr1_pru1_pru_r30_0 (Output)	pr1_pru1_pru_r31_0 (INPUT)	
46	R2	GPIO2_7	pr1_pru1_pru_r30_1 (Output)	pr1_pru1_pru_r31_1 (INPUT)	

Table 2.10: P9 PRU0 and PRU1 Access

PIN	PROC	NAME			
17	A16	I2C1_SCL	pr1_uart0_txd		
18	B16	I2C1_SDA	pr1_uart0_rxd		
19	D17	I2C2_SCL	pr1_uart0_rts_n		
20	D18	I2C2_SDA	pr1_uart0_cts_n		
21	B17	UART2_TXD	pr1_uart0_rts_n		
22	A17	UART2_RXD	pr1_uart0_cts_n		
24	D15	UART1_TXD	pr1_uart0_txd	pr1_pru0_pru_r31_16 (Input)	
25	A14	GPIO3_21	pr1_pru0_pru_r30_5 (Output)	pr1_pru0_pru_r31_5 (Input)	
26	D16	UART1_RXD	pr1_uart0_rxd	pr1_pru1_pru_r31_16	
27	C13	GPIO3_19	pr1_pru0_pru_r30_7 (Output)	pr1_pru0_pru_r31_7 (Input)	
28	C12	SPI1_CS0	eCAP2_in_PWM2_out	pr1_pru0_pru_r30_3 (Output)	pr1_pru0_pru_r31_3 (Input)
29	B13	SPI1_D0	pr1_pru0_pru_r30_1 (Output)	pr1_pru0_pru_r31_1 (Input)	

continues on next page

Table 2.10 – continued from previous page

PIN	PROC	NAME			
30	D12	SPI1_D1	pr1_pru0_pru_r30_2 (Output)	pr1_pru0_pru_r31_2 (Input)	
31	A13	SPI1_SCLK	pr1_pru0_pru_r30_0 (Output)	pr1_pru0_pru_r31_0 (Input)	

Note: GPIO3_21 is also the 24.576MHZ clock input to the processor to enable HDMI audio. To use this pin the oscillator must be disabled.

2.2.7 Connectors

This section describes each of the connectors on the board.

Expansion Connectors

The expansion interface on the board is comprised of two 46 pin connectors. All signals on the expansion headers are _3.3V_ unless otherwise indicated.

NOTE: Do not connect 5V logic level signals to these pins or the board will be damaged.

NOTE: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

The location and spacing of the expansion headers are the same as on the original BeagleBone.

Connector P8 *table-12* shows the pinout of the **P8** expansion header. Other signals can be connected to this connector based on setting the pin mux on the processor, but this is the default settings on power up. The SW is responsible for setting the default function of each pin. There are some signals that have not been listed here. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The *PROC* column is the pin number on the processor.

The *PIN* column is the pin number on the expansion header.

The *MODE* columns are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

NOTE: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

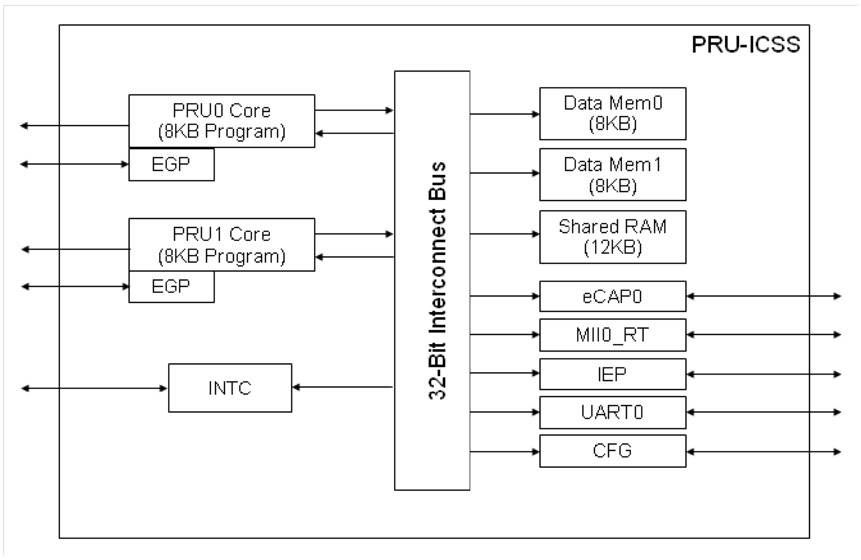


Fig. 2.51: PRU-ICSS Block Diagram

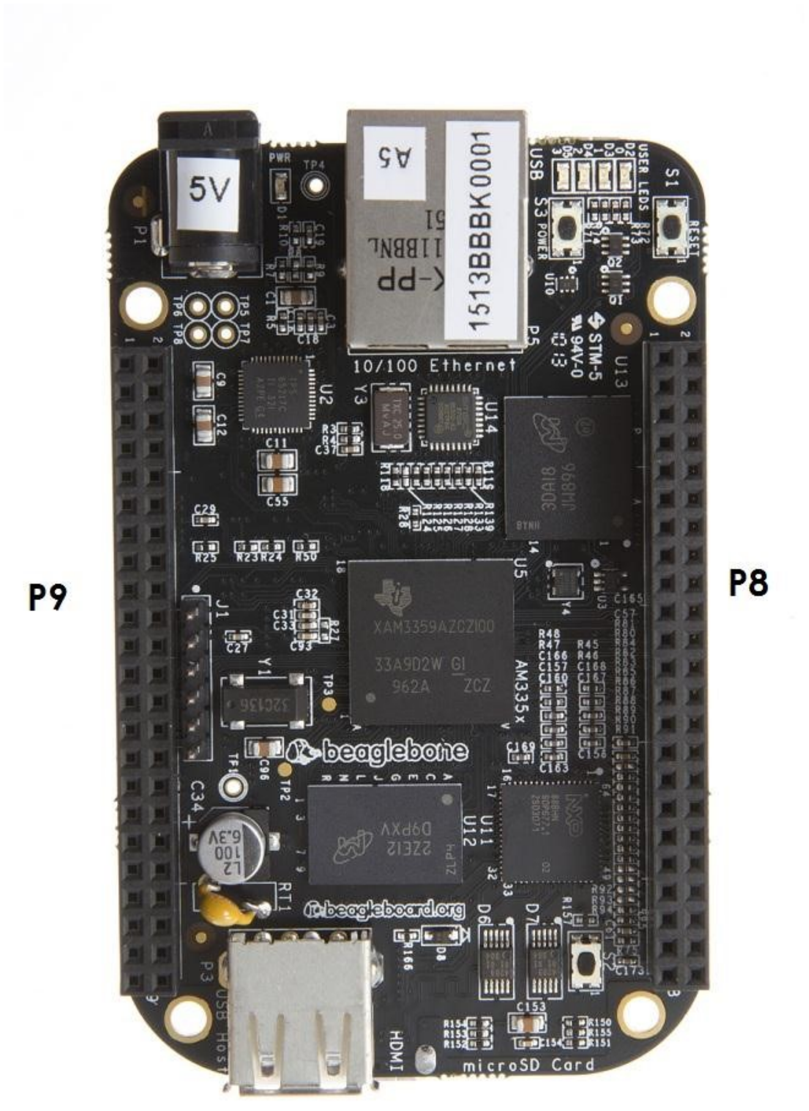


Fig. 2.52: Expansion Connector Location

Table 2.11: Expansion Header P8 Pinout

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6	MODE7
1,2		GND								
3	R9	GPIO1_6	gpmc_ad6	mmc1_dat6						gpio1[6]
4	T9	GPIO1_7	gpmc_ad7	mmc1_dat7						gpio1[7]
5	R8	GPIO1_2	gpmc_ad2	mmc1_dat2						gpio1[2]
6	T8	GPIO1_3	gpmc_ad3	mmc1_dat3						gpio1[3]
7	R7	TIMER4	gpmc_advn_ale		timer4					gpio2[2]
8	T7	TIMER7	gpmc_oen_ren		timer7					gpio2[3]
9	T6	TIMER5	gpmc_be0n_cle		timer5					gpio2[5]
10	U6	TIMER6	gpmc_wen		timer6					gpio2[4]
11	R12	GPIO1_13	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1	eQEP2B_in		pr1_pru0_pru_r30_15	gpio1[13]
12	T12	GPIO1_12	gpmc_ad12	lcd_data19	mmc1_dat4	mmc2_dat0	eQEP2a_in		pr1_pru0_pru_r30_14	gpio1[12]
13	T10	EHRPWM2B	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5	ehrpwm2B			gpio0[23]
14	T11	GPIO0_26	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6	ehrpwm2_tripzone_in			gpio0[26]
15	U13	GPIO1_15	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3	eQEP2_strobe		pr1_pru0_pru_r31_15	gpio1[15]
16	V13	GPIO1_14	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2	eQEP2_index		pr1_pru0_pru_r31_14	gpio1[14]
17	U12	GPIO0_27	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7	ehrpwm0_synco			gpio0[27]
18	V12	GPIO2_1	gpmc_clk_mux0	lcd_memory_clk	gpmc_wait1	mmc2_clk			mcasp0_fsr	gpio2[1]
19	U10	EHRPWM2A	gpmc_ad8	lcd_data23	mmc1_dat0	mmc2_dat4	ehrpwm2A			gpio0[22]
20	V9	GPIO1_31	gpmc_csn2	gpmc_be1n	mmc1_cmd			pr1_pru1_pru_r30_13	pr1_pru1_pru_r31_13	gpio1[31]
21	U9	GPIO1_30	gpmc_csn1	gpmc_clk	mmc1_clk			pr1_pru1_pru_r30_12	pr1_pru1_pru_r31_12	gpio1[30]
22	V8	GPIO1_5	gpmc_ad5	mmc1_dat5						gpio1[5]
23	U8	GPIO1_4	gpmc_ad4	mmc1_dat4						gpio1[4]
24	V7	GPIO1_1	gpmc_ad1	mmc1_dat1						gpio1[1]
25	U7	GPIO1_0	gpmc_ad0	mmc1_dat0						gpio1[0]
26	V6	GPIO1_29	gpmc_csn0							gpio1[29]
27	U5	GPIO2_22	lcd_vsync	gpmc_a8				pr1_pru1_pru_r30_8	pr1_pru1_pru_r31_8	gpio2[22]
28	V5	GPIO2_24	lcd_pclk	gpmc_a10				pr1_pru1_pru_r30_10	pr1_pru1_pru_r31_10	gpio2[24]
29	R5	GPIO2_23	lcd_hsync	gpmc_a9				pr1_pru1_pru_r30_9	pr1_pru1_pru_r31_9	gpio2[23]
30	R6	GPIO2_25	lcd_ac_bias_en	gpmc_a11						gpio2[25]
31	V4	UART5_CTSN	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1	uart5_rxd		uart5_ctsn	gpio0[10]
32	T5	UART5_RTSN	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx	mcasp0_axr3		uart5_rtsn	gpio0[11]
33	V3	UART4_RTSN	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr	mcasp0_axr3		uart4_rtsn	gpio0[9]
34	U4	UART3_RTSN	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr	mcasp0_axr2		uart3_rtsn	gpio2[17]
35	V2	UART4_CTSN	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr	mcasp0_axr2		uart4_ctsn	gpio0[8]
36	U3	UART3_CTSN	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0			uart3_ctsn	gpio2[16]
37	U1	UART5_TXD	lcd_data8	gpmc_a12	ehrpwm1_tripzone_in	mcasp0_aclkx	uart5_txd		uart2_ctsn	gpio2[14]
38	U2	UART5_RXD	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx	uart5_rxd		uart2_rtsn	gpio2[15]
39	T3	GPIO2_12	lcd_data6	gpmc_a6		eQEP2_index		pr1_pru1_pru_r30_6	pr1_pru1_pru_r31_6	gpio2[12]
40	T4	GPIO2_13	lcd_data7	gpmc_a7		eQEP2_strobe	pr1_edio_data_out7	pr1_pru1_pru_r30_7	pr1_pru1_pru_r31_7	gpio2[13]
41	T1	GPIO2_10	lcd_data4	gpmc_a4		eQEP2A_in		pr1_pru1_pru_r30_4	pr1_pru1_pru_r31_4	gpio2[10]
42	T2	GPIO2_11	lcd_data5	gpmc_a5		eQEP2B_in		pr1_pru1_pru_r30_5	pr1_pru1_pru_r31_5	gpio2[11]
43	R3	GPIO2_8	lcd_data2	gpmc_a2		ehrpwm2_tripzone_in		pr1_pru1_pru_r30_2	pr1_pru1_pru_r31_2	gpio2[8]
44	R4	GPIO2_9	lcd_data3	gpmc_a3		ehrpwm0_synco		pr1_pru1_pru_r30_3	pr1_pru1_pru_r31_3	gpio2[9]
45	R1	GPIO2_6	lcd_data0	gpmc_a0		ehrpwm2A		pr1_pru1_pru_r30_0	pr1_pru1_pru_r31_0	gpio2[6]
46	R2	GPIO2_7	lcd_data1	gpmc_a1		ehrpwm2B		pr1_pru1_pru_r30_1	pr1_pru1_pru_r31_1	gpio2[7]

Connector P9 Table-13 lists the signals on connector **P9**. Other signals can be connected to this connector based on setting the pin mux on the processor, but this is the default settings on power up.

There are some signals that have not been listed here. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The *PROC* column is the pin number on the processor.

The *PIN* column is the pin number on the expansion header.

The *MODE* columns are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

NOTES:

In the table are the following notations:

PWR_BUT is a 5V level as pulled up internally by the TPS65217C. It is activated by pulling the signal to GND.

NOTE: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

- Both of these signals connect to pin 41 of P11. Resistors are installed that allow for the GPIO3_20 connection to be removed by removing R221. The intent is to allow the SW to use either of these signals, one or the other, on pin 41. SW should set the unused pin in input mode when using the other pin. This allowed us to get an extra signal out to the expansion header.
- Both of these signals connect to pin 42 of P11. Resistors are installed that allow for the GPIO3_18 connection to be removed by removing R202. The intent is to allow the SW to use either of these signals, on pin 42. SW should set the unused pin in input mode when using the other pin. This allowed us to get an extra signal out to the expansion header.

Table 2.12: Expansion Header P9 Pinout

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6	MODE7
1,2	**GND**									
3,4	**DC_3.3V**									
5,6	**VDD_5V**									
7,8	**SYS_5V**									
9	**PWR_BUT**									
10	**A10**	**SYS_RESETn**								
11	**T17**	**UART4_RXD**	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd		uart4_rxd_mux2	gpio0[30]
12	**U18**	**GPIO1_28**	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir		mcasp0_aclkr_mux3	gpio1[28]
13	**U17**	**UART4_TXD**	gpmc_wpn	mii_rxerr	gpmc_csn5	rmii2_rxerr	mmc2_sdcd		uart4_txd_mux2	gpio0[31]
14	**U14**	**EHRPWM1A**	gpmc_a2	mii2_txd3	rgmii2_td3	mmc2_dat1	gpmc_a18		ehrpwm1A_mux1	gpio1[18]
15	**R13**	**GPIO1_16**	gpmc_a0	gmii2_txen	rmii2_tctl	mii2_txen	gpmc_a16		ehrpwm1_tripzone_input	gpio1[16]
16	**T14**	**EHRPWM1B**	gpmc_a3	mii2_txd2	rgmii2_td2	mmc2_dat2	gpmc_a19		ehrpwm1B_mux1	gpio1[19]
17	**A16**	**I2C1_SCL**	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci	pr1_uart0_txd			gpio0[5]
18	**B16**	**I2C1_SDA**	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone	pr1_uart0_rxd			gpio0[4]
19	**D17**	**I2C2_SCL**	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1	pr1_uart0_rts_n		gpio0[13]
20	**D18**	**I2C2_SDA**	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0	pr1_uart0_cts_n		gpio0[12]
21	**B17**	**UART2_TXD**	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B	pr1_uart0_rts_n		EMU3_mux1	gpio0[3]
22	**A17**	**UART2_RXD**	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A	pr1_uart0_cts_n		EMU2_mux1	gpio0[2]
23	**V14**	**GPIO1_17**	gpmc_a1	gmii2_rxdv	rgmii2_rxdv	mmc2_dat0	gpmc_a17		ehrpwm0_synco	gpio1[17]
24	**D15**	**UART1_TXD**	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL		pr1_uart0_txd	pr1_pru0_pru_r31_16	gpio0[15]
25	**A14**	**GPIO3_21**	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4_mux2	pr1_pru0_pru_r30_7	pr1_pru0_pru_r31_7	gpio3[21]
26	**D16**	**UART1_RXD**	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA		pr1_uart0_rxd	pr1_pru1_pru_r31_16	gpio0[14]
27	**C13**	**GPIO3_19**	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2_mux2	pr1_pru0_pru_r30_5	pr1_pru0_pru_r31_5	gpio3[19]
28	**C12**	**SPI1_CS0**	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out	pr1_pru0_pru_r30_3	pr1_pru0_pru_r31_3	gpio3[17]
29	**B13**	**SPI1_D0**	mcasp0_fsx	ehrpwm0B		spi1_d0	mmc1_sdcd_mux1	pr1_pru0_pru_r30_1	pr1_pru0_pru_r31_1	gpio3[15]
30	**D12**	**SPI1_D1**	mcasp0_axr0	ehrpwm0_tripzone		spi1_d1	mmc2_sdcd_mux1	pr1_pru0_pru_r30_2	pr1_pru0_pru_r31_2	gpio3[16]
31	**A13**	**SPI1_SCLK**	mcasp0_aclkx	ehrpwm0A		spi1_sclk	mmc0_sdcd_mux1	pr1_pru0_pru_r30_0	pr1_pru0_pru_r31_0	gpio3[14]
32	**VADC**									
33	**C8**	**AIN4**								
34	**AGND**									
35	**A8**	**AIN6**								
36	**B8**	**AIN5**								
37	**B7**	**AIN2**								
38	**A7**	**AIN3**								
39	**B6**	**AIN0**								
40	**C7**	**AIN1**								
41	**D14**	**CLKOUT2**	xdma_event_intr1		tclkin	clkout2	timer7_mux1	pr1_pru0_pru_r31_16	EMU3_mux0	gpio0[20]
	D13	**GPIO3_20**	mcasp0_axr1	eQEP0_index		mcasp1_axr0	emu3	pr1_pru0_pru_r30_6	pr1_pru0_pru_r31_6	gpio3[20]
42	**C18**	**GPIO0_7**	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	gpio0[7]
	B12	**GPIO3_18**	mcasp0_aclkr	eQEP0A_in	mcasp0_axr2	mcasp1_aclkx		pr1_pru0_pru_r30_4	pr1_pru0_pru_r31_4	gpio3[18]
43-46	**GND**									

Power Jack

The DC power jack is located next to the RJ45 Ethernet connector as shown in «figure-51». This uses the same power connector as is used on the original BeagleBone. The connector has a 2.1mm diameter center post (5VDC) and a 5.5mm diameter outer dimension on the barrel (GND).

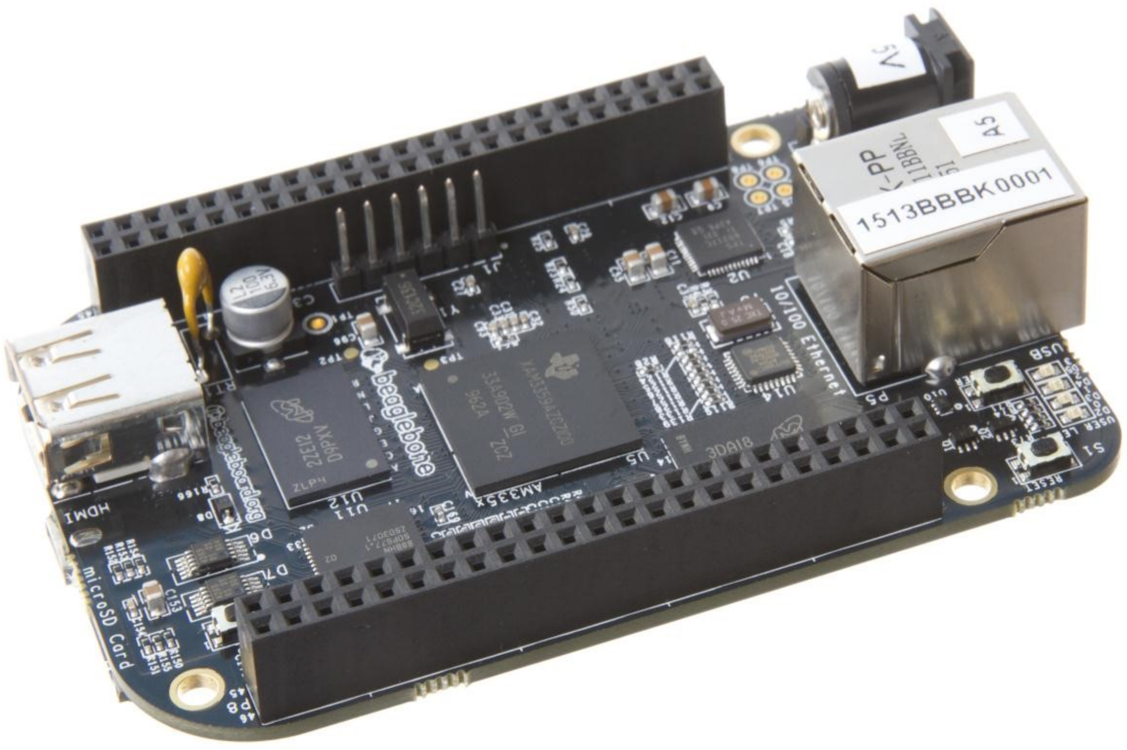


Fig. 2.53: 5VDC Power Jack

The board requires a regulated 5VDC +/-.25V supply at 1A. A higher current rating may be needed if capes are plugged into the expansion headers. Using a higher current power supply will not damage the board.

USB Client

The USB Client connector is accessible on the bottom side of the board under the row of four LEDs as shown in «figure-52». It uses a 5 pin miniUSB cable, the same as is used on the original BeagleBone. The cable is provided with the board. The cable can also be used to power the board.

This port is a USB Client only interface and is intended for connection to a PC.

USB Host

There is a single USB Host connector on the board and is shown in *Figure 53* below.

The port is USB 2.0 HS compatible and can supply up to 500mA of current. If more current or ports is needed, then a HUB can be used.

Serial Header

Each board has a debug serial interface that can be accessed by using a special serial cable that is plugged into the serial header as shown in *Figure 54* below.

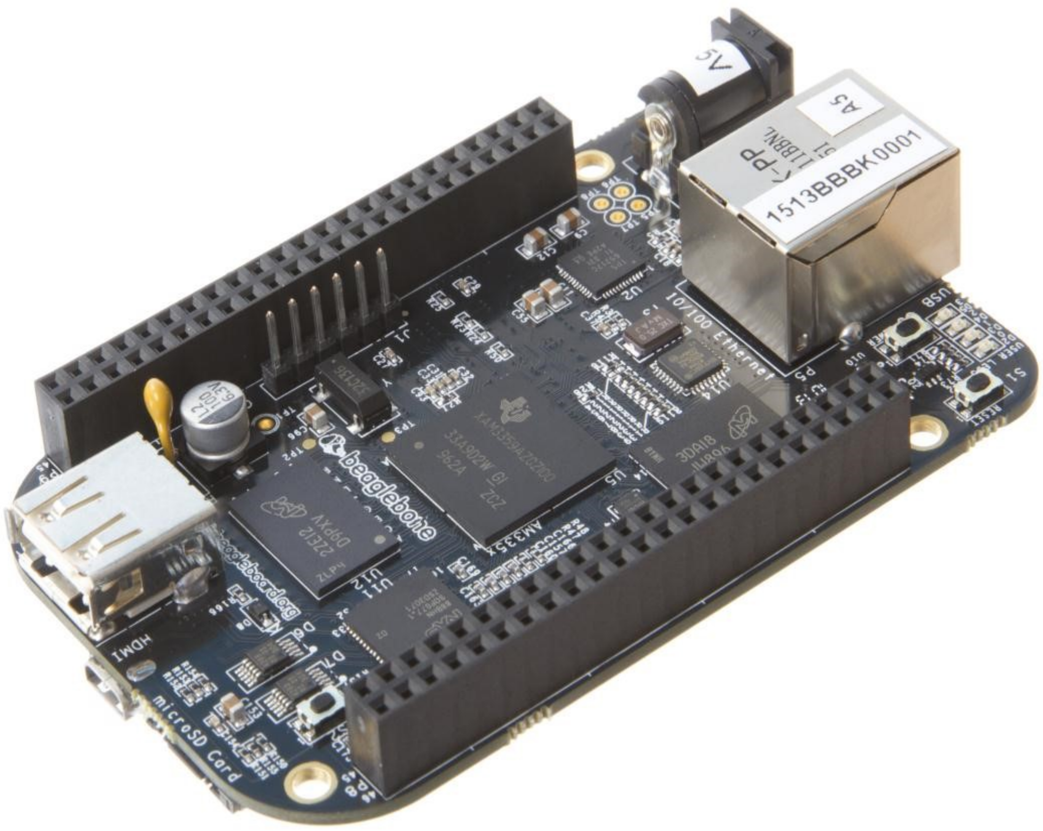


Fig. 2.54: USB Client

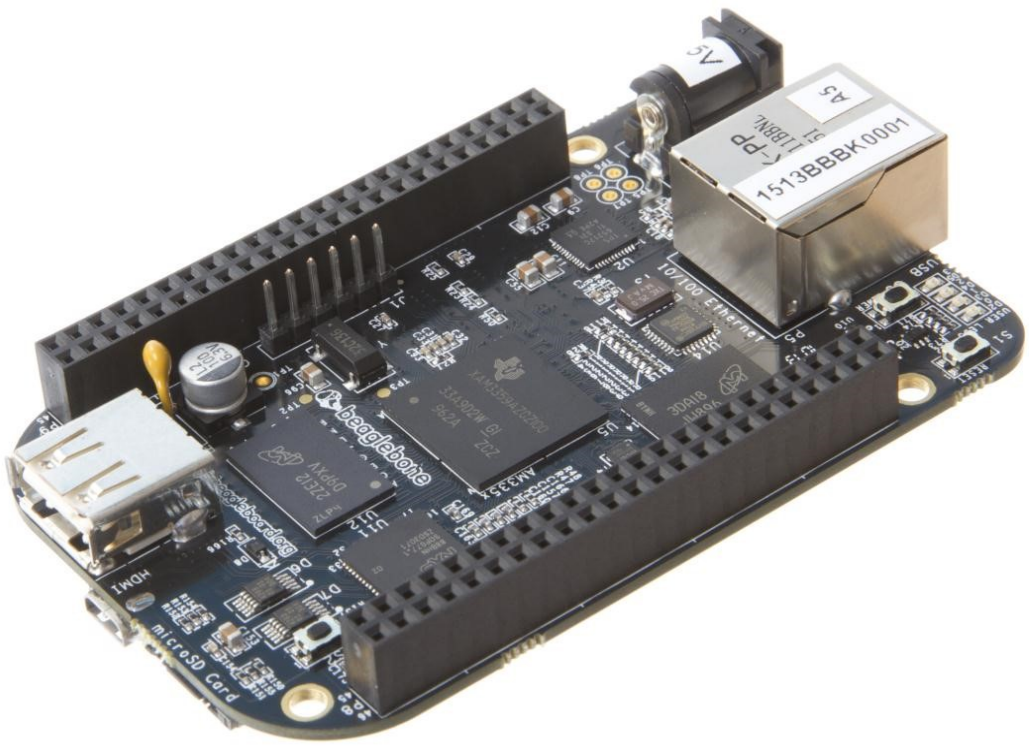


Fig. 2.55: USB Host Connector

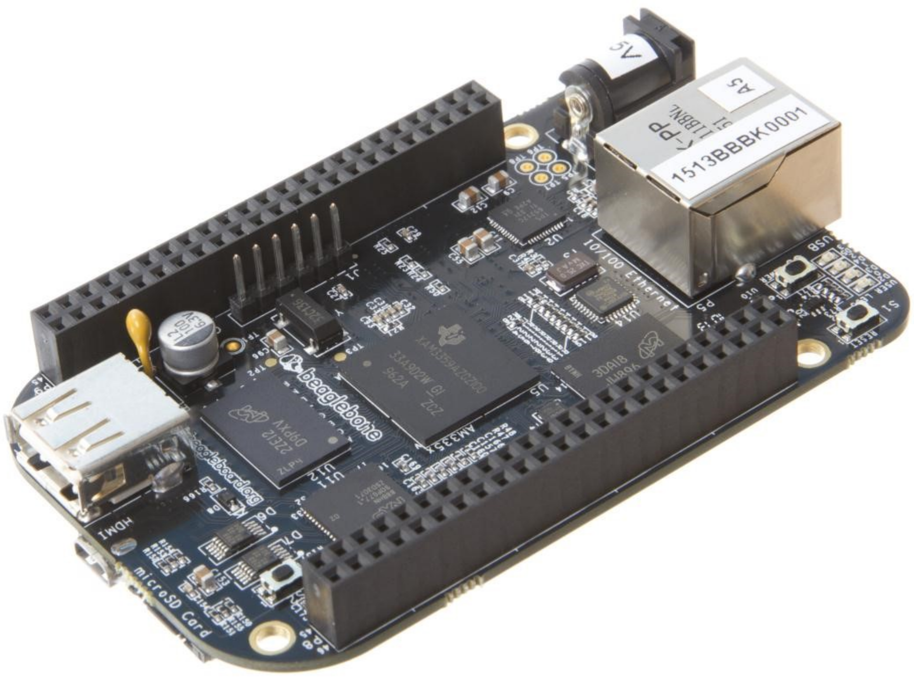


Fig. 2.56: Serial Debug Header

Two signals are provided, TX and RX on this connector. The levels on these signals are 3.3V. In order to access these signals, a FTDI USB to Serial cable is recommended as shown in *Figure 55* below.

Fig. 2.57: PRU-ICSS Block Diagram

The cable can be purchased from several different places and must be the 3.3V version TTL-232R-3V3. Information on the cable itself can be found direct from FTDI at: pdf

Pin 1 of the cable is the black wire. That must align with the pin 1 on the board which is designated by the white dot next to the connector on the board.

Refer to the support WIKI http://elinux.org/BeagleBoneBlack for more sources of this cable and other options that will work.

Table is the pinout of the connector as reflected in the schematic. It is the same as the FTDI cable which can be found at https://ftdichip.com/wp-content/uploads/2020/07/DS_USB_RS232_CABLES.pdf with the exception that only three pins are used on the board. The pin numbers are defined in *Table 14*. The signals are from the perspective of the board.

Table 2.13: J1 Serial Header Pins

PIN NUMBER	SIGNAL
1	Ground
4	Receive
5	Transmit

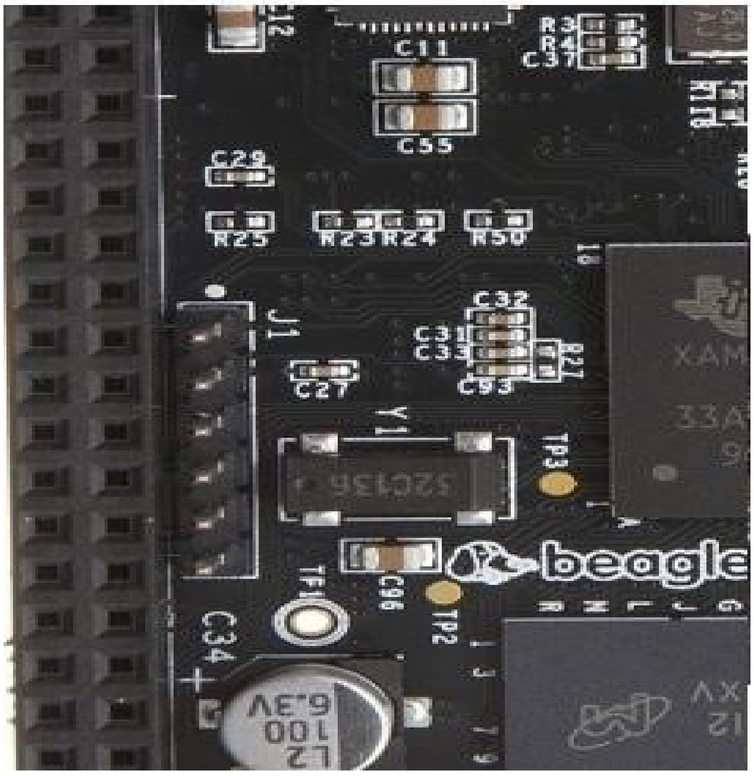


Fig. 2.58: Serial Header

HDMI

Access to the HDMI interface is through the HDMI connector that is located on the bottom side of the board as shown in *Figure 57* below.

The connector is microHDMI connector. This was done due to the space limitations we had in finding a place to fit the connector. It requires a microHDMI to HDMI cable as shown in *Figure 58* below. The cable can be purchased from several different sources.

microSD

A microSD connector is located on the back or bottom side of the board as shown in *Figure 59* below. The microSD card is not supplied with the board.

When plugging in the SD card, the writing on the card should be up. Align the card with the connector and push to insert. Then release. There should be a click and the card will start to eject slightly, but it then should latch into the connector. To eject the card, push the SD card in and then remove your finger. The SD card will be ejected from the connector.

Do not pull the SD card out or you could damage the connector.

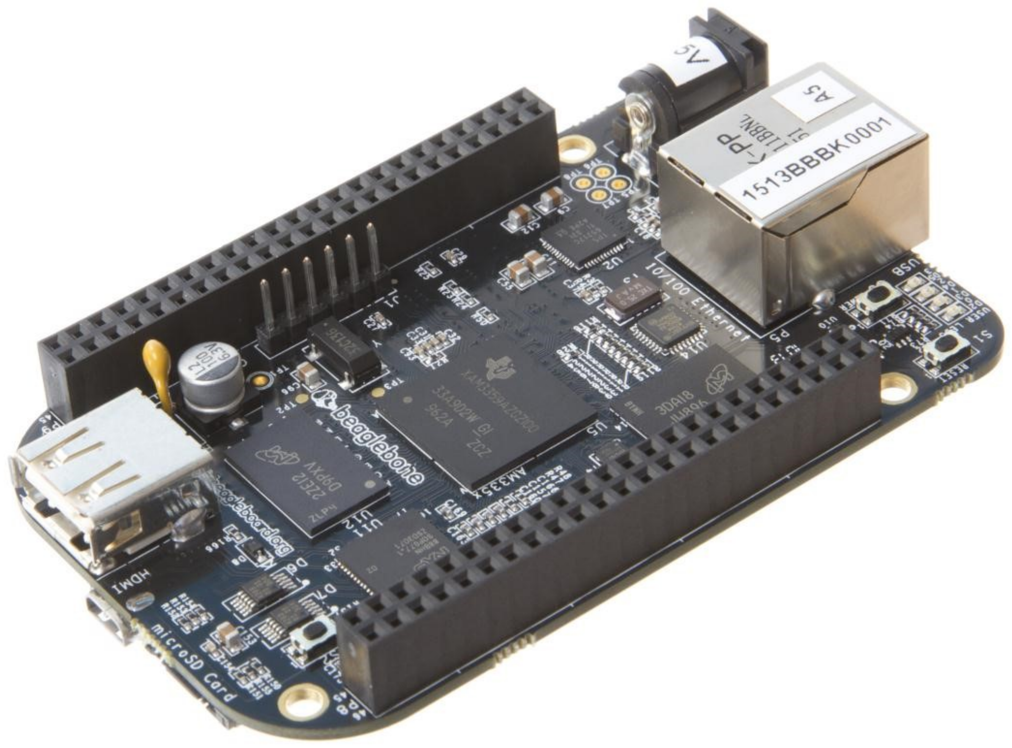


Fig. 2.59: HDMI Connector

Fig. 2.60: HDMI Cable

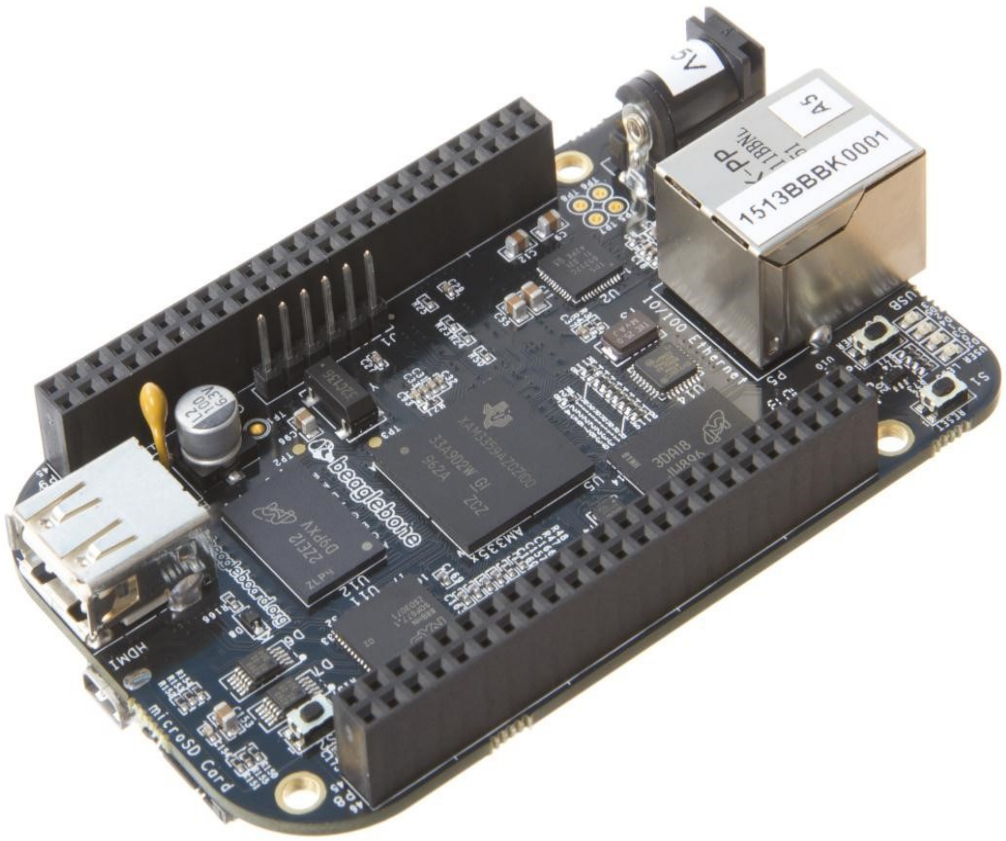


Fig. 2.61: microSD Connector

Ethernet

The board comes with a single 10/100 Ethernet interface located next to the power jack as shown in Figure below.

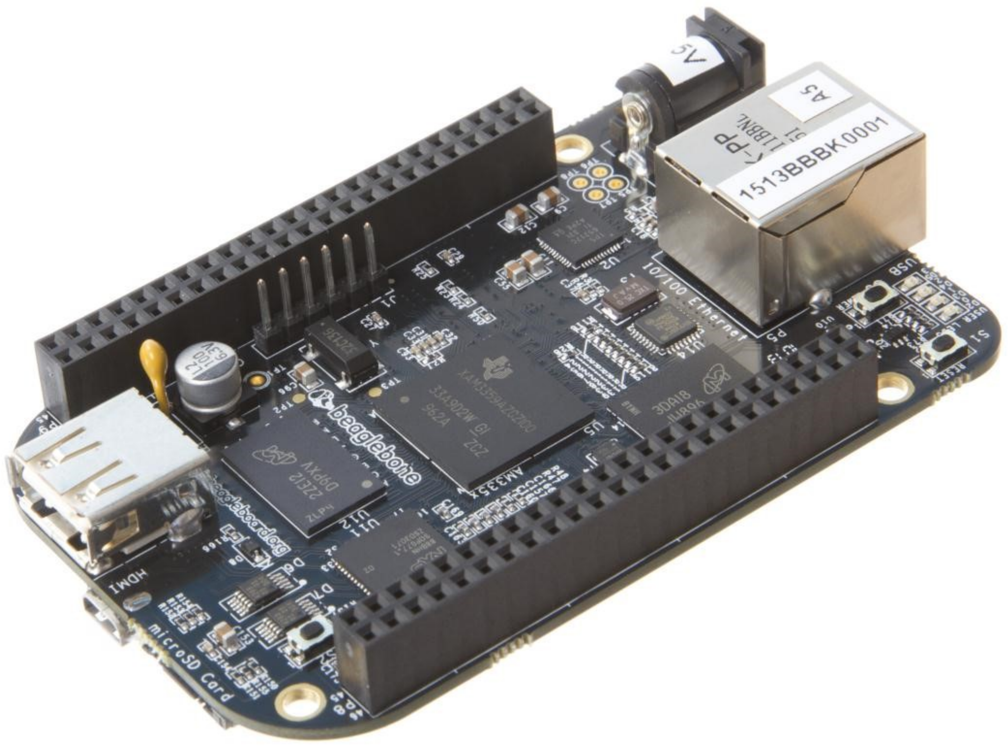


Fig. 2.62: Ethernet Connector

The PHY supports AutoMDX which means either a straight or a swap cable can be used.

JTAG Connector

A place for an optional 20 pin CTI JTAG header is provided on the board to facilitate the SW development and debugging of the board by using various JTAG emulators. This header is not supplied standard on the board. To use this, a connector will need to be soldered onto the board.

If you need the JTAG connector you can solder it on yourself. No other components are needed. The connector is made by Samtec and the part number is FTR-110-03-G-D-06. You can purchase it from http://www.digikey.com/

2.2.8 Cape Board Support

The BeagleBone Black has the ability to accept up to four expansion boards or capes that can be stacked onto the expansion headers. The word cape comes from the shape of the board as it is fitted around the Ethernet connector on the main board. This notch acts as a key to ensure proper orientation of the cape.

This section describes the rules for creating capes to ensure proper operation with the BeagleBone Black and proper interoperability with other capes that are intended to coexist with each other. Co-existence is not a requirement and is in itself, something that is impossible to control or administer. But, people will be able to create capes that operate with other capes that are already available based on public information as it pertains to what pins and features each cape uses. This information will be able to be read from the EEPROM on each cape.

This section is intended as a guideline for those wanting to create their own capes. Its intent is not to put limits on the creation of capes and what they can do, but to set a few basic rules that will allow the SW to administer their operation with the BeagleBone Black. For this reason there is a lot of flexibility in the specification that we hope most people will find liberating and in the spirit of Open Source Hardware. I am sure there are others that would like to see tighter control, more details, more rules and much more order to the way capes are handled.

Over time, this specification will change and be updated, so please refer to the latest version of this manual prior to designing your own capes to get the latest information.

DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN

POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

BeagleBone Black Cape Compatibility

The main expansion headers are the same between the BeagleBone and BeagleBone Black. While the pins are the same, some of these pins are now used on the BeagleBone Black. The following sections discuss these pins.

The Power Expansion header was removed from the BeagleBone Black and is not available.

PAY VERY CLOSE ATTENTION TO THIS SECTION AND READ CAREFULLY!!

LCD Pins The LCD pins are used on the BeagleBone Black to drive the HDMI framer. These signals are listed in *Table 15* below.

Table 2.14: P8 LCD Conflict Pins

PIN	PROC	NAME	MODE0
27	U5	GPIO2_22	lcd_vsync
28	V5	GPIO2_24	lcd_pclk
29	R5	GPIO2_23	lcd_hsync
30	R6	GPIO2_25	lcd_ac_bias_en
31	V4	UART5_CTSN	lcd_data14
32	T5	UART5_RTSN	lcd_data15
33	V3	UART4_RTSN	lcd_data13
34	U4	UART3_RTSN	lcd_data11
35	V2	UART4_CTSN	lcd_data12
36	U3	UART3_CTSN	lcd_data10
37	U1	UART5_TXD	lcd_data8
38	U2	UART5_RXD	lcd_data9
39	T3	GPIO2_12	lcd_data6
40	T4	GPIO2_13	lcd_data7
41	T1	GPIO2_10	lcd_data4
42	T2	GPIO2_11	lcd_data5
43	R3	GPIO2_8	lcd_data2
44	R4	GPIO2_9	lcd_data3
45	R1	GPIO2_6	lcd_data0
46	R2	GPIO2_7	lcd_data1

If you are using these pins for other functions, there are a few things to keep in mind:

- On the HDMI Framer, these signals are all inputs so the framer will not be driving these pins.
- The HDMI framer will add a load onto these pins.
- There are small filter caps on these signals which could also change the operation of these pins if used for other functions.
- When used for other functions, the HDMI framer cannot be used.

- There is no way to power off the framer as this would result in the framer being powered through these input pins which would not a be a good idea.
- These pins are also the *SYSBOOT* pins. DO NOT drive them before the **SYS_RESETN** signal goes high. If you do, the board may not boot because you would be changing the boot order of the processor.

In order to use these pins, the SW will need to reconfigure them to whatever function you need the pins to do. To keep power low, the HDMI framer should be put in a low power mode via the SW using the *I2C0* interface.

eMMC Pins The BeagleBone Black uses 10 pins to connect to the processor that also connect to the P8 expansion connector. These signals are listed below in *Table 16*. The proper mode is MODE2.

Table 2.15: P8 eMMC Conflict Pins

PIN	PROC	SIGNAL	MODE
22	V8	MMC1_DAT5	1
23	U8	MMC1_DAT4	1
24	V7	MMC1_DAT1	1
5	R8	MMC1_DAT2	1
4	T9	MMC1_DAT7	1
3	R9	MMC1_DAT6	1
6	T8	MMC1_DAT3	1
25	U7	MMC1_DAT0	1
20	V9	MMC1_CMD	2
21	U9	MMC1_CLK	2

If using these pins, several things need to be kept in mind when doing so:

- On the eMMC device, these signals are inputs and outputs.
- The eMMC device will add a load onto these pins.
- When used for other functions, the eMMC cannot be used. This means you must boot from the microSD slot.
- If using these pins, you need to put the eMMC into reset. This requires that the eMMC be accessible from the processor in order to set the eMMC to accept the eMMC pins.
- DO NOT drive the eMMC pins until the eMMC has been put into reset. This means that if you choose to use these pins, they must not drive any signal until enabled via Software. This requires a buffer or some other form of hold off function enabled by a GPIO pin on the expansion header.

On power up, the eMMC is NOT reset. If you hold the Boot button down, this will force a boot from the microSD. This is not convenient when a cape is plugged into the board. There are two solutions to this issue:

1. Wipe the eMMC clean. This will cause the board to default to microSD boot. If you want to use the eMMC later, it can be reprogrammed. 2. You can also tie LCD_DATA2 low on the cape during boot. This will be the same as if you were holding the boot button. However, in order to prevent unforeseen issues, you need to gate this signal with RESET, when the data is sampled. After set goes high, the signal should be removed from the pin.

BEFORE the SW reinitializes the pins, it **MUST** put the eMMC in reset. This is done by taking eMMC_RSTn (GPIO1_20) LOW **after** the eMMC has been put into a mode to enable the reset line. This pin does not connect to the expansion header and is accessible only on the board.

DO NOT automatically drive any conflicting pins until the SW enables it. This puts the SW in control to ensure that the eMMC is in reset before the signals are used from the cape. You can use a GPIO pin for this. No, we will not designate a pin for this function. It will be determined on a cape by cape basis by the designer of the respective cape.

EEPROM

Each cape must have its own EEPROM containing information that will allow the SW to identify the board and to configure the expansion headers pins as needed. The one exception is proto boards intended for prototyping. They may or may not have an EEPROM on them. An EEPROM is required for all capes sold in order for them operate correctly when plugged into the BeagleBone Black.

The address of the EEPROM will be set via either jumpers or a dipswitch on each expansion board. *Figure 61* below is the design of the EEPROM circuit.

The EEPROM used is the same one as is used on the BeagleBone and the BeagleBone Black, a CAT24C256. The CAT24C256 is a 256 kb Serial CMOS EEPROM, internally organized as 32,768 words of 8 bits each. It features a 64-byte page write buffer and supports the Standard (100 kHz), Fast (400 kHz) and Fast-Plus (1 MHz) I2C protocol.

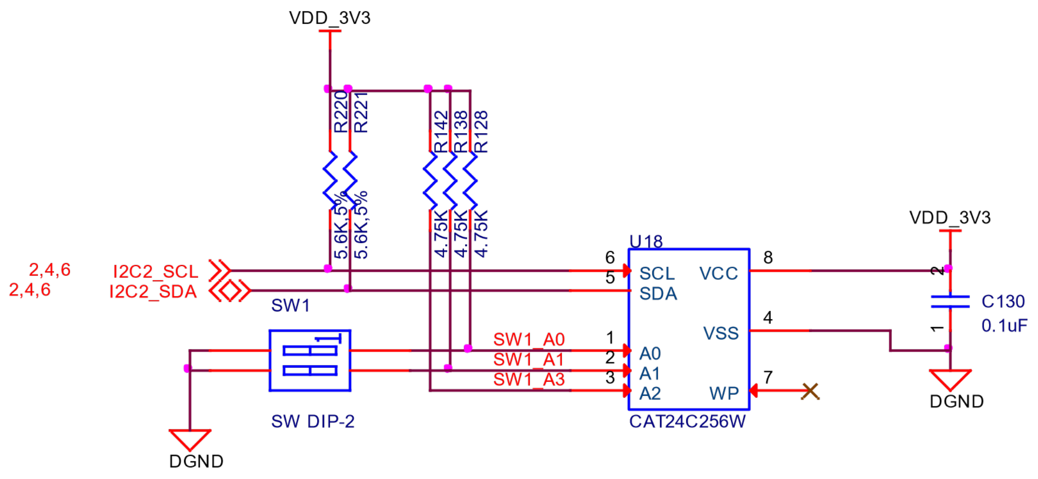


Fig. 2.63: Expansion Board EEPROM Without Write Protect

The addressing of this device requires two bytes for the address which is not used on smaller size EEPROMs, which only require only one byte. Other compatible devices may be used as well. Make sure the device you select supports 16 bit addressing. The part package used is at the discretion of the cape designer.

EEPROM Address In order for each cape to have a unique address, a board ID scheme is used that sets the address to be different depending on the setting of the dipswitch or jumpers on the capes. A two position dipswitch or jumpers is used to set the address pins of the EEPROM.

It is the responsibility of the user to set the proper address for each board and the position in the stack that the board occupies has nothing to do with which board gets first choice on the usage of the expansion bus signals. The process for making that determination and resolving conflicts is left up to the SW and, as of this moment in time, this method is a something of a mystery due to the new Device Tree methodology introduced in the 3.8 kernel.

Address line A2 is always tied high. This sets the allowable address range for the expansion cards to *0x54* to**0x57**. All other I2C addresses can be used by the user in the design of their capes. But, these addresses must not be used other than for the board EEPROM information. This also allows for the inclusion of EEPROM devices on the cape if needed without interfering with this EEPROM. It requires that A2 be grounded on the EEPROM not used for cape identification.

I2C Bus The EEPROMs on each expansion board are connected to I2C2 on connector P9 pins 19 and 20. For this reason I2C2 must always be left connected and should not be changed by SW to remove it from the expansion header pin mux settings. If this is done, the system will be unable to detect the capes.

The I2C signals require pullup resistors. Each board must have a 5.6K resistor on these signals. With four capes installed this will result in an effective resistance of 1.4K if all capes were installed and all the resistors used were exactly 5.6K. As more capes are added the resistance is reduced to overcome capacitance added to the signals. When no capes are installed the internal pullup resistors must be activated inside the processor to prevent I2C timeouts on the I2C bus.

The I2C2 bus may also be used by capes for other functions such as I/O expansion or other I2C compatible devices that do not share the same address as the cape EEPROM.

EEPROM ********************

The design in *Figure 62* has the write protect disabled. If the write protect is not enabled, this does expose the EEPROM to being corrupted if the I2C2 bus is used on the cape and the wrong address written to. It is recommended that a write protection function be implemented and a Test Point be added that when grounded, will allow the EEPROM to be written to. To enable write operation, Pin 7 of the EEPROM must be tied to ground.

When not grounded, the pin is HI via pullup resistor R210 and therefore write protected. Whether or not Write Protect is provided is at the discretion of the cape designer.

Variable & MAC Memory VDD_3V3B

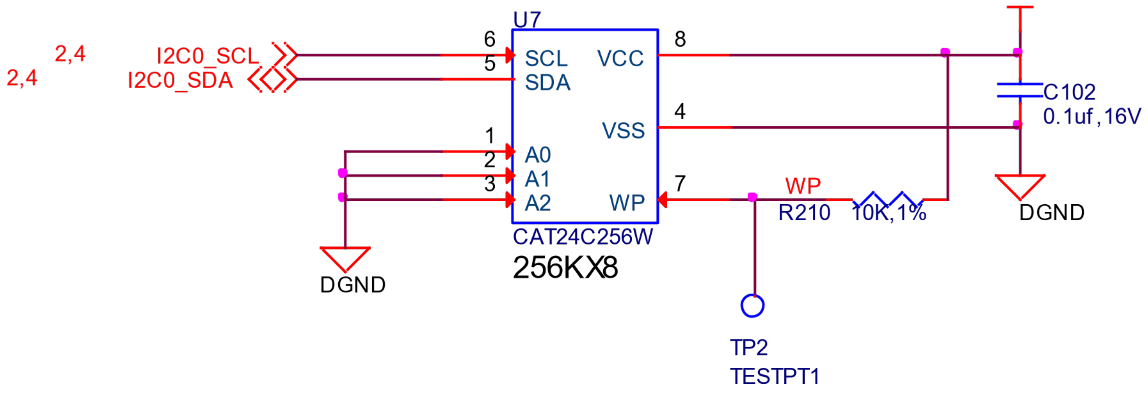


Fig. 2.64: Expansion Board EEPROM Write Protect

EEPROM Data Format Table below shows the format of the contents of the expansion board EEPROM. Data is stored in Big Endian with the least significant value on the right. All addresses read as a single byte data from the EEPROM, but two byte addressing is used. ASCII values are intended to be easily read by the user when the EEPROM contents are dumped.

Table 2.16: Expansion Board EEPROM

Name	Offset	Size (bytes)	Contents
Header	0	4	0xAA, 0x55, 0x33, 0xEE
EEPROM Revision	4	2	Revision number of the overall format of this EEPROM in ASCII =A1
Board Name	6	32	Name of board in ASCII so user can read it when the EEPROM is dumped. Up to
Version	38	4	Hardware version code for board in ASCII.Version format is up to the developer.
Manufacturer	42	16	ASCII name of the manufacturer. Company or individual's name.
Part Number	58	16	ASCII Characters for the part number. Up to maker of the board.
Number of Pins	74	2	Number of pins used by the daughter board including the power pins used. De
Serial Number	76	12	Serial number of the board. This is a 12 character string which is: **WWYY&&&**
Pin Usage	88	148	Two bytes for each configurable pins of the 74 pins on the expansion connector
VDD_3V3B Current	236	2	Maximum current in milliamps. This is HEX value of the current in decimal 150
VDD_5V Current	238	2	Maximum current in milliamps. This is HEX value of the current in decimal 150
SYS_5V Current	240	2	Maximum current in milliamps. This is HEX value of the current in decimal 150
DC Supplied	242	2	Indicates whether or not the board is supplying voltage on the VDD_5V rail and
Available	244	32543	Available space for other non-volatile codes/data to be used as needed by the

Pin Usage *Table 18* is the locations in the EEPROM to set the I/O pin usage for the cape. It contains the value to be written to the Pad Control Registers. Details on this can be found in section **9.2.2** of the *AM3358 Technical Reference Manual*, The table is left blank as a convenience and can be printed out and used as a template for creating a custom setting for each cape. The 16 bit integers and all 16 bit fields are to be stored in Big Endian format.

Bit 15 PIN USAGE is an indicator and should be a 1 if the pin is used or 0 if it is unused.

Bits 14-7 RESERVED is not to be used and left as 0.

Bit 6 SLEW CONTROL 0=Fast 1=Slow

Bit 5 RX Enabled 0=Disabled 1=Enabled

Bit 4 PU/PD 0=Pulldown 1=Pullup.

Bit 3 PULLUP/DN 0=Pullup/pulldown enabled 1= Pullup/pulldown disabled

Bit 2-0 MUX MODE SELECT Mode 0-7. (refer to TRM)

Refer to the TRM for proper settings of the pin MUX mode based on the signal selection to be used.

The *AIN0-6* pins do not have a pin mux setting, but they need to be set to indicate if each of the pins is used on the cape. Only bit 15 is used for the AIN signals.

Table 2.17: EEPROM Pin Usage

+	+	+	15	14	13	12	11	10	9	8	7	6	5
Off set	**Conn**	**Name**	**Pin Usage**	**Type**	+	**Reserved**	+	+	**S L E W**	**R X**	**P U - P D**	**P U / D E N**	**Mux Mode**
88	**P9-22**	**UART2_RXD**	+	+	+	+	+	+	+	+	+	+	+
90	**P9-21**	**UART2_TXD**	+	+	+	+	+	+	+	+	+	+	+
92	**P9-18**	**I2C1_SDA**	+	+	+	+	+	+	+	+	+	+	+
94	**P9-17**	**I2C1_SCL**	+	+	+	+	+	+	+	+	+	+	+
96	**P9-42**	**GPIO0_7**	+	+	+	+	+	+	+	+	+	+	+
98	**P8-35**	**UART4_CTSN**	+	+	+	+	+	+	+	+	+	+	+
100	**P8-33**	**UART4_RTSN**	+	+	+	+	+	+	+	+	+	+	+
102	**P8-31**	**UART5_CTSN**	+	+	+	+	+	+	+	+	+	+	+
104	**P8-32**	**UART5_RTSN**	+	+	+	+	+	+	+	+	+	+	+
106	**P9-19**	**I2C2_SCL**	+	+	+	+	+	+	+	+	+	+	+
108	**P9-20**	**I2C2_SDA**	+	+	+	+	+	+	+	+	+	+	+
110	**P9-26**	**UAR*T1_RXD**	+	+	+	+	+	+	+	+	+	+	+
112	**P9-24**	**UART1_TXD**	+	+	+	+	+	+	+	+	+	+	+
114	**P9-41**	**CLKOUT2**	+	+	+	+	+	+	+	+	+	+	+
116	**P8-19**	**EHRPWM2A**	+	+	+	+	+	+	+	+	+	+	+
118	**P8-13**	**EHRPWM2B**	+	+	+	+	+	+	+	+	+	+	+
120	**P8-14**	**GPIO0_26**	+	+	+	+	+	+	+	+	+	+	+
122	**P8-17**	**GPIO0_27**	+	+	+	+	+	+	+	+	+	+	+
124	**P9-11**	**UART4_RXD**	+	+	+	+	+	+	+	+	+	+	+
126	**P9-13**	**UART4_TXD**	+	+	+	+	+	+	+	+	+	+	+
128	**P8-25**	**GPIO1_0**	+	+	+	+	+	+	+	+	+	+	+
130	**P8-24**	**GPIO1_1**	+	+	+	+	+	+	+	+	+	+	+
132	**P8-5**	**GPIO1_2**	+	+	+	+	+	+	+	+	+	+	+
134	**P8-6**	**GPIO1_3**	+	+	+	+	+	+	+	+	+	+	+
136	**P8-23**	**GPIO1_4**	+	+	+	+	+	+	+	+	+	+	+
138	**P8-22**	**GPIO1_5**	+	+	+	+	+	+	+	+	+	+	+
140	**P8-3**	**GPIO1_6**	+	+	+	+	+	+	+	+	+	+	+
142	**P8-4**	**GPIO1_7**	+	+	+	+	+	+	+	+	+	+	+
144	**P8-12**	**GPIO1_12**	+	+	+	+	+	+	+	+	+	+	+
146	**P8-11**	**GPIO1_13**	+	+	+	+	+	+	+	+	+	+	+
148	**P8-16**	**GPIO1_14**	+	+	+	+	+	+	+	+	+	+	+
150	**P8-15**	**GPIO1_15**	+	+	+	+	+	+	+	+	+	+	+
152	**P9-15**	**GPIO1_16**	+	+	+	+	+	+	+	+	+	+	+

			15	14	13	12	11	10	9	8	7	6	5

Off set	Conn	Name	Pin Usage	Type	+	Reserve	+	+	S L E W	R X	P U - P D	P U / DE N	Mux Mode
154	P9-23	GPIO1_17											
156	P9-14	EHRPWM1A											
158	P9-16	EHRPWM1B											
160	P9-12	GPIO1_28											
162	P8-26	GPIO1_29											
164	P8-21	GPIO1_30											
166	P8-20	GPIO1_31											
168	P8-18	GPIO2_1											
170	P8-7	TIMER4											
172	P8-9	TIMER5	+	+	+	+	+	+	+	+	+	+	+
174	P8-10	TIMER6	+	+	+	+	+	+	+	+	+	+	+
176	P8-8	TIMER7	+	+	+	+	+	+	+	+	+	+	+
178	P8-45	GPIO2_6	+	+	+	+	+	+	+	+	+	+	+
180	P8-46	GPIO2_7	+	+	+	+	+	+	+	+	+	+	+
182	P8-43	GPIO2_8	+	+	+	+	+	+	+	+	+	+	+
184	P8-44	GPIO2_9	+	+	+	+	+	+	+	+	+	+	+
186	P8-41	GPIO2_10	+	+	+	+	+	+	+	+	+	+	+
188	P8-42	GPIO2_11	+	+	+	+	+	+	+	+	+	+	+
190	P8-39	GPIO2_12	+	+	+	+	+	+	+	+	+	+	+
192	P8-40	GPIO2_13	+	+	+	+	+	+	+	+	+	+	+
194	P8-37	UART5_TX'+'	+	+	+	+	+	+	+	+	+	+	+
196	P8-38	UART5_RX'+'	+	+	+	+	+	+	+	+	+	+	+
198	P8-36	UART3_CTSN	+	+	+	+	+	+	+	+	+	+	+
200	P8-34	UART3_RTSN	+	+	+	+	+	+	+	+	+	+	+
202	P8-27	GPIO2_22	+	+	+	+	+	+	+	+	+	+	+
204	P8-29	GPIO2_23	+	+	+	+	+	+	+	+	+	+	+
206	P8-28	GPIO2_24	+	+	+	+	+	+	+	+	+	+	+
208	P8-30	GPIO2_25	+	+	+	+	+	+	+	+	+	+	+
210	P9-29	SPI1_D0	+	+	+	+	+	+	+	+	+	+	+
212	P9-30	SPI1_D1	+	+	+	+	+	+	+	+	+	+	+
214	P9-28	SPI1_CS0	+	+	+	+	+	+	+	+	+	+	+
216	P9-27	GPIO3_19	+	+	+	+	+	+	+	+	+	+	+
218	P9-31	SPI1_SCLK	+	+	+	+	+	+	+	+	+	+	+
220	P9-25	GPIO3_21	+	+	+	+	+	+	+	+	+	+	+
+	+	+	15	14	13	12	11	10	9	8	7	6	5
Off set	Conn	Name	Pin Usage	Type		Reserve			S L E W	R X	P U - P D	P U / DE N	Mux Mode
+	+	+	+	0	0	0	0	0	0	0	0	0	0
222	P9-39	AIN0	+	+	+	+	+	+	+	+	+	+	+
224	P9-40	AIN1	+	+	+	+	+	+	+	+	+	+	+
226	P9-37	AIN2	+	+	+	+	+	+	+	+	+	+	+
228	P9-38	AIN3	+	+	+	+	+	+	+	+	+	+	+
230	P9-33	AIN4	+	+	+	+	+	+	+	+	+	+	+
232	P9-36	AIN5	+	+	+	+	+	+	+	+	+	+	+
234	P9-35	AIN6	+	+	+	+	+	+	+	+	+	+	+

Pin Usage Consideration

This section covers things to watch for when hooking up to certain pins on the expansion headers.

Boot PIN There are 16 pins that control the boot mode of the processor that are exposed on the expansion headers. *Figure 63* below shows those signals as they appear on the BeagleBone Black.:

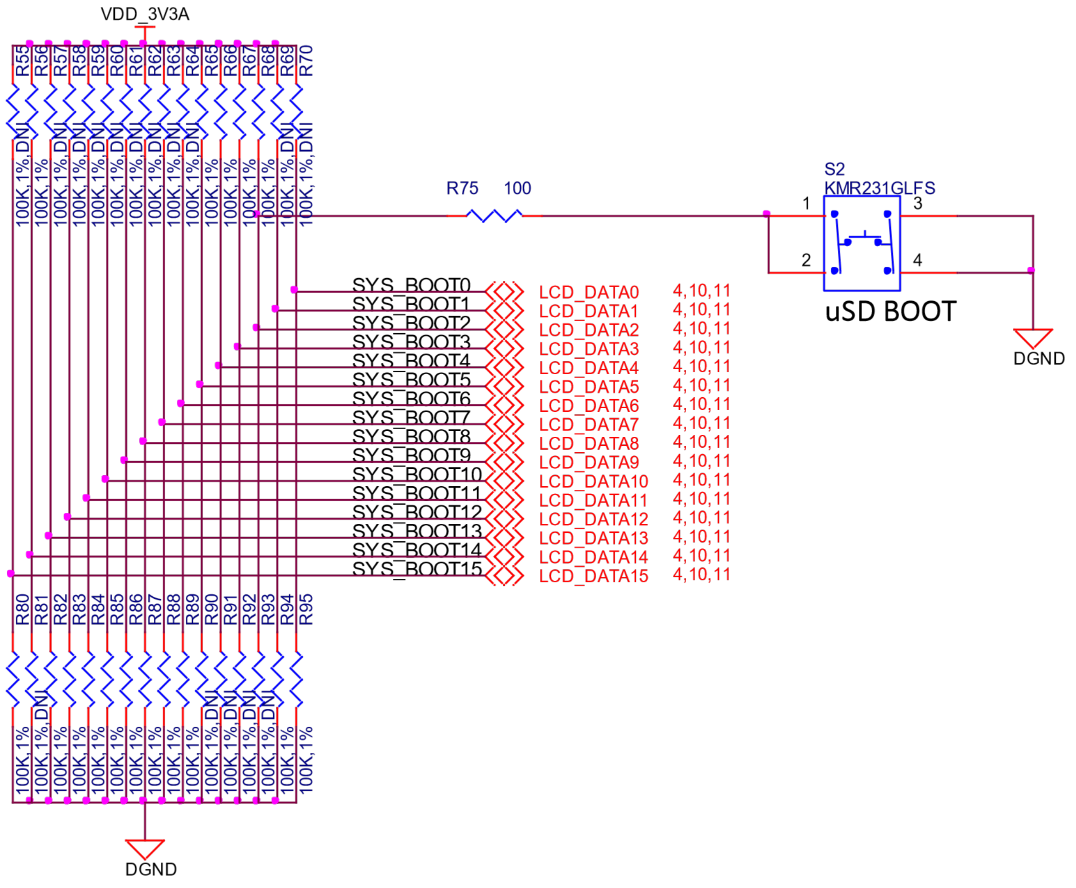

VDD_3V3A
R55 R56 R57 R58 R59 R60 R61 R62 R63 R64 R65 R66 R67 R68 R69 R70
100K,1%,DNI
100K,1%
R75 100
S2
KMR231GLFS
uSD BOOT
DGND
SYS_BOOT0 … SYS_BOOT15
LCD_DATA0 … LCD_DATA15
4,10,11
R80 R81 R82 R83 R84 R85 R86 R87 R88 R89 R90 R91 R92 R93 R94 R95
DGND

Figure 63. Expansion Boot Pins

Fig. 2.65: Boot signals

If you plan to use any of these signals, then on power up, these pins should not be driven. If you do, it can affect the boot mode of the processor and could keep the processor from booting or working correctly.

If you are designing a cape that is intended to be used as a boot source, such as a NAND board, then you should drive the pins to reconfigure the boot mode, but only at reset. After the reset phase, the signals should not be driven to allow them to be used for the other functions found on those pins. You will need to override the resistor values in order to change the settings. The DC pull-up requirement should be based on the AM3358 Vih min voltage of 2 volts and AM3358 maximum input leakage current of 18uA. Also take into account any other current leakage paths on these signals which could be caused by your specific cape design.

The DC pull-down requirement should be based on the AM3358 Vil max voltage of 0.8 volts and AM3358 maximum input leakage current of 18uA plus any other current leakage paths on these signals.

Expansion Connectors

A combination of male and female headers is used for access to the expansion headers on the main board. There are three possible mounting configurations for the expansion headers:

- *Single* no board stacking but can be used on the top of the stack.

- *Stacking-up* to four boards can be stacked on top of each other.
- *Stacking* with signal *stealing-up* to three boards can be stacked on top of each other, but certain boards will not pass on the signals they are using to prevent signal loading or use by other cards in the stack.

The following sections describe how the connectors are to be implemented and used for each of the different configurations.

Non-Stacking Headers-Single Cape For non-stacking capes single configurations or where the cape can be the last board on the stack, the two 46 pin expansion headers use the same connectors. *Figure 64* is a picture of the connector. These are dual row 23 position 2.54mm x 2.54mm connectors.

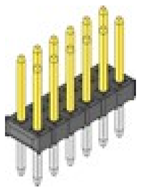


Fig. 2.66: Single Expansion Connector

The connector is typically mounted on the bottom side of the board as shown in *Figure 65*. These are very common connectors and should be easily located. You can also use two single row 23 pin headers for each of the dual row headers.

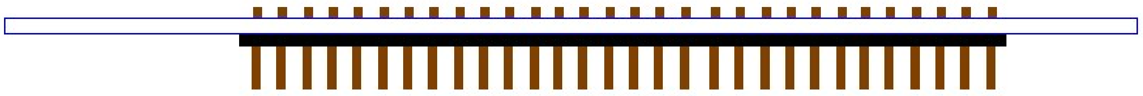


Fig. 2.67: Single Cape Expansion Connector

It is allowed to only populate the pins you need. As this is a non-stacking configuration, there is no need for all headers to be populated. This can also reduce the overall cost of the cape. This decision is up to the cape designer.

For convenience listed in *Table 19* are some possible choices for part numbers on this connector. They have varying pin lengths and some may be more suitable than others for your use. It should be noted, that the longer the pin and the further it is inserted into the BeagleBone Black connector, the harder it will be to remove due to the tension on 92 pins. This can be minimized by using shorter pins or removing those pins that are not used by your particular design. The first item in**Table 18** is on the edge and may not be the best solution. Overhang is the amount of the pin that goes past the contact point of the connector on the BeagleBone Black

Table 2.19: Single Cape Connectors

SUPPLIER	PARTNUMBER	LENGTH(in)	OVERHANG(in)
Major League	TSHC-123-D-03-145-G-LF	.145	.004
Major League	TSHC-123-D-03-240-G-LF	.240	.099
Major League	TSHC-123-D-03-255-G-LF	.255	.114

The G in the part number is a plating option. Other options may be used as well as long as the contact area is gold. Other possible sources are Sullins and Samtec for these connectors. You will need to ensure the depth into the connector is sufficient

Main Expansion Headers-Stacking For stacking configuration, the two 46 pin expansion headers use the same connectors. *Figure 66* is a picture of the connector. These are dual row 23 position 2.54mm x 2.54mm connectors.

The connector is mounted on the top side of the board with longer tails to allow insertion into the BeagleBone Black. *Figure 67* is the connector configuration for the connector.

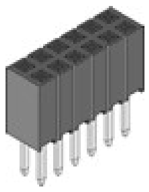


Fig. 2.68: Expansion Connector

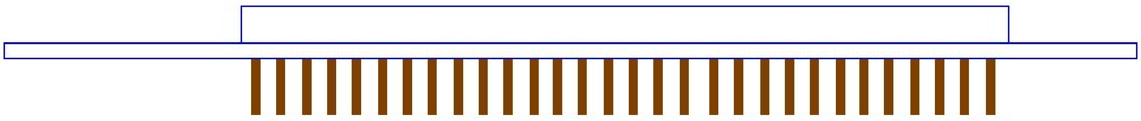


Fig. 2.69: Stacked Cape Expansion Connector

For convenience listed in *Table 18* are some possible choices for part numbers on this connector. They have varying pin lengths and some may be more suitable than others for your use. It should be noted, that the longer the pin and the further it is inserted into the BeagleBone Black connector, the harder it will be to remove due to the tension on 92 pins. This can be minimized by using shorter pins. There are most likely other suppliers out there that will work for this connector as well. If anyone finds other suppliers of compatible connectors that work, let us know and they will be added to this document. The first item in**Table 19** is on the edge and may not be the best solution. Overhang is the amount of the pin that goes past the contact point of the connector on the BeagleBone Black.

The third part listed in *Table 20* will have insertion force issues.

Table 2.20: Stacked Cape Connectors

SUPPLIER	PARTNUMBER	TAIL LENGTH(in)	OVERHANG(in)
Major League	SSHQ-123-D-06-G-LF	.190	0.049
Major League	SSHQ-123-D-08-G-LF	.390	0.249
Major League	SSHQ-123-D-10-G-LF	.560	0.419

There are also different plating options on each of the connectors above. Gold plating on the contacts is the minimum requirement. If you choose to use a different part number for plating or availability purposes, make sure you do not select the "LT" option.

Other possible sources are Sullins and Samtec but make sure you select one that has the correct mating depth.

StackedStealing *Figure 68* is the connector configuration for stackable capes that does not provide all of the signals upwards for use by other boards. This is useful if there is an expectation that other boards could interfere with the operation of your board by exposing those signals for expansion. This configuration consists of a combination of the stacking and nonstacking style connectors.

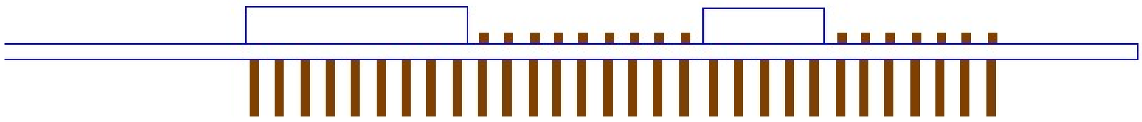


Fig. 2.70: Stacked w/Signal Stealing Expansion Connector

Retention Force The length of the pins on the expansion header has a direct relationship to the amount of force that is used to remove a cape from the BeagleBone Black. The longer the pins extend into the connector the harder it is to remove. There is no rule that says that if longer pins are used, that the connector pins have to extend all the way into the mating connector on the BeagleBone Black, but this is controlled by the user and therefore is hard to control. We have also found that if you use gold pins, while more expensive, it makes for a smoother finish which reduces the friction.

This section will attempt to describe the tradeoffs and things to consider when selecting a connector and its pin length.

Figure 69 shows the key measurements used in calculating how much the pin extends past the contact point on the connector, what we call overhang.

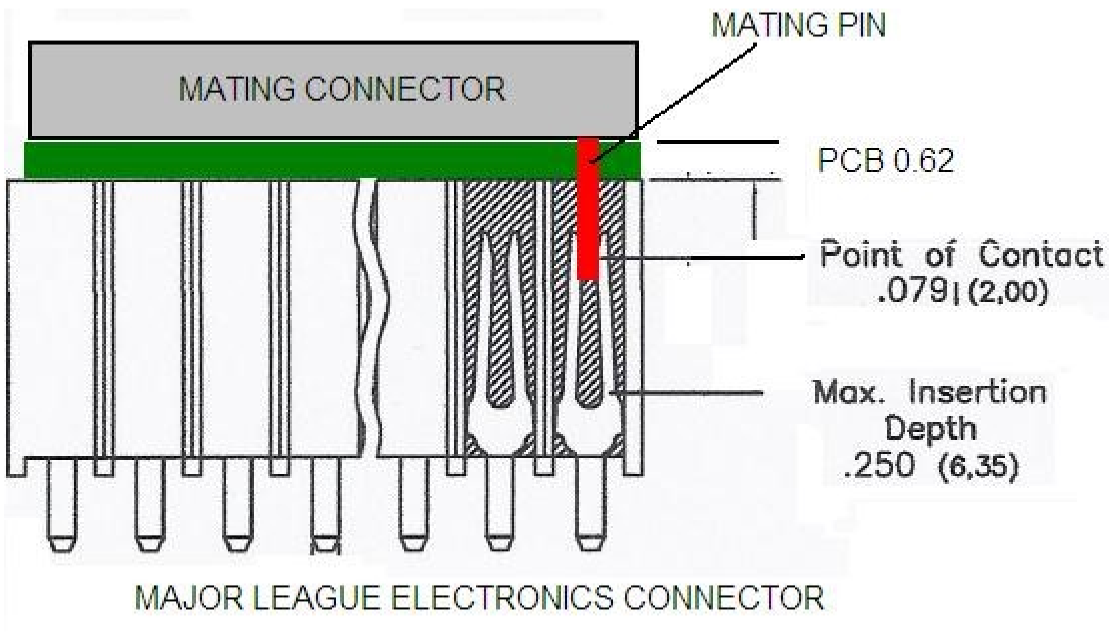


Fig. 2.71: Connector Pin Insertion Depth

8.5 Signal Usage

Based on the pin muxing capabilities of the processor, each expansion pin can be configured for different functions. When in the stacking mode, it will be up to the user to ensure that any conflicts are resolved between multiple stacked cards. When stacked, the first card detected will be used to set the pin muxing of each pin. This will prevent other modes from being supported on stacked cards and may result in them being inoperative.

In «section-7-1» of this document, the functions of the pins are defined as well as the pin muxing options. Refer to this section for more information on what each pin is. To simplify things, if you use the default name as the function for each pin and use those functions, it will simplify board design and reduce conflicts with other boards.

Interoperability is up to the board suppliers and the user. This specification does not specify a fixed function on any pin and any pin can be used to the full extent of the functionality of that pin as enabled by the processor.

DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

8.6 Cape Power

This section describes the power rails for the capes and their usage.

Main Board Power The *Table 1* describes the voltages from the main board that are available on the expansion connectors and their ratings. All voltages are supplied by connector**P9**. The current ratings listed are per pin.

Table 2.21: Expansion Voltages

Current	Name	P9	P9	Name	Current
250mA	VDD_3V3B	3	4	VDD_3V3B	250mA
1000mA	VDD_5V	5	6	VDD_5V	1000mA
250mA	SYS_5V	7	8	SYS_5V	250mA

The *VDD_3V3B* rail is supplied by the LDO on the BeagleBone Black and is the primary power rail for expansion boards. If the power requirement for the capes exceeds the current rating, then locally generated voltage rail can be used. It is recommended that this rail be used to power any buffers or level translators that may be used.

VDD_5V is the main power supply from the DC input jack. This voltage is not present when the board is powered via USB. The amount of current supplied by this rail is dependent upon the amount of current available. Based on the board design, this rail is limited to 1A per pin from the main board.

The *SYS_5V* rail is the main rail for the regulators on the main board. When powered from a DC supply or USB, this rail will be 5V. The available current from this rail depends on the current available from the USB and DC external supplies.

Power A cape can have a jack or terminals to bring in whatever voltages may be needed by that board. Care should be taken not to let this voltage be fed back into any of the expansion header pins.

It is possible to provide 5V to the main board from an expansion board. By supplying a 5V signal into the *VDD_5V* rail, the main board can be supplied. This voltage must not exceed 5V. You should not supply any voltage into any other pin of the expansion connectors. Based on the board design, this rail is limited to 1A per pin to the BeagleBone Black.

There are several precautions that need to be taken when working with the expansion headers to prevent damage to the board.

1. *Do not apply any voltages to any I/O pins when the board is not powered on.* 2. *Do not drive any external signals into the I/O pins until after the VDD_3V3B rail is up.* 3. *Do not apply any voltages that are generated from external sources.* 4. *If voltages are generated from the VDD_5V signal, those supplies must not become active until after the VDD_3V3B rail is up.* 5. *If you are applying signals from other boards into the expansion headers, make sure you power the board up after you power up the BeagleBone Black or make the connections after power is applied on both boards.*

Powering the processor via its I/O pins can cause damage to the processor.

8.7 Mechanical

This section provides the guidelines for the creation of expansion boards from a mechanical standpoint. Defined is a standard board size that is the same profile as the BeagleBone Black. It is expected that the majority of expansion boards created will be of standard size. It is possible to create boards of other sizes and in some cases this is required, as in the case of an LCD larger than the BeagleBone Black board.

Standard Cape Size A slot is provided for the Ethernet connector to stick up higher than the cape when mounted. This also acts as a key function to ensure that the cape is oriented correctly. Space is also provided to allow access to the user LEDs and reset button on the main board.

Some people have inquired as to the difference in the radius of the corners of the BeagleBone Black and why they are different. This is a result of having the BeagleBone fit into the Altoids style tin.

It is not required that the cape be exactly like the BeagleBone Black board in this respect.

Extended Cape Size Capes larger than the standard board size are also allowed. A good example would be an LCD panel. There is no practical limit to the sizes of these types of boards. The notch for the key is also not required, but it is up to the supplier of these boards to ensure that the BeagleBone Black is not plugged in incorrectly in such a manner that damage would be caused to the BeagleBone Black or any other capes that may be installed. Any such damage will be the responsibility of the supplier of such a cape to repair.

As with all capes, the EEPROM is required and compliance with the power requirements must be adhered to.

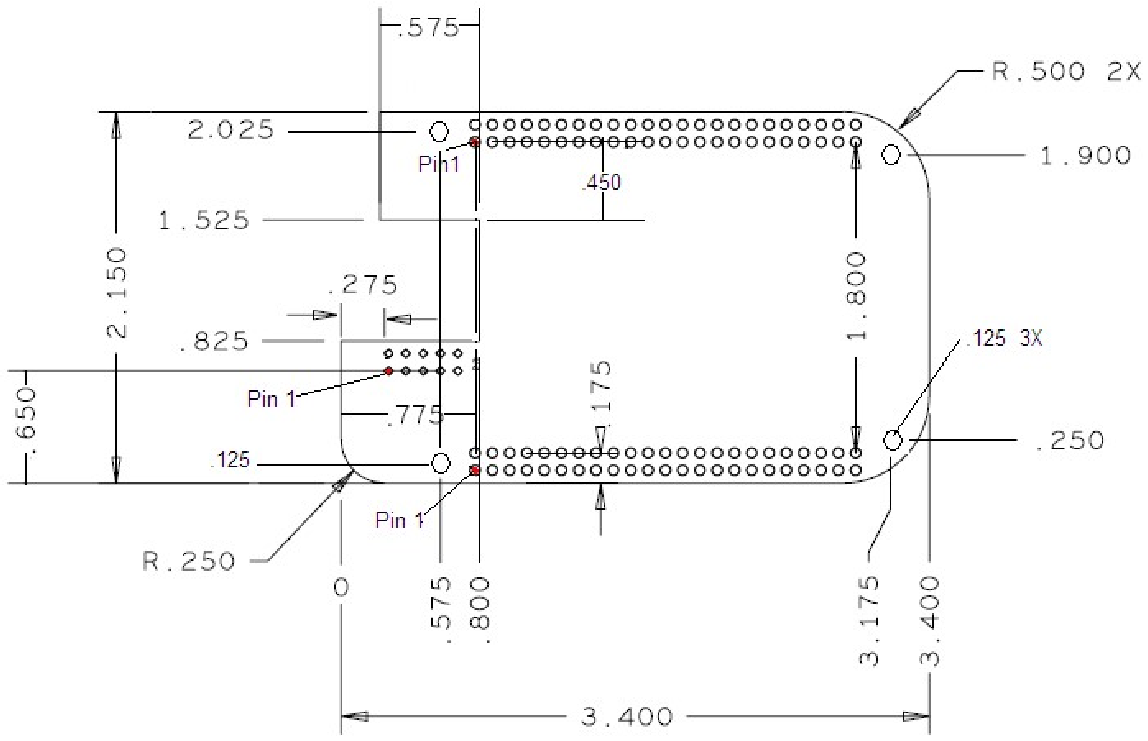


Fig. 2.72: Cape Board Dimensions

Enclosures There are numerous enclosures being created in all different sizes and styles. The mechanical design of these enclosures is not being defined by this specification.

The ability of these designs to handle all shapes and sizes of capes, especially when you consider up to four can be mounted with all sorts of interface connectors, it is difficult to define a standard enclosure that will handle all capes already made and those yet to be defined.

If cape designers want to work together and align with one enclosure and work around it that is certainly acceptable. But we will not pick winners and we will not do anything that impedes the openness of the platform and the ability of enclosure designers and cape designers to innovate and create new concepts.

2.2.9 BeagleBone Black Mechanical

Dimensions and Weight

Size: 3.5” x 2.15” (86.36mm x 53.34mm)

Max height: .187” (4.76mm)

PCB Layers: 6

PCB thickness: .062”

RoHS Compliant: Yes

Weight: 1.4 oz

Silkscreen and Component Locations

2.2.10 Pictures

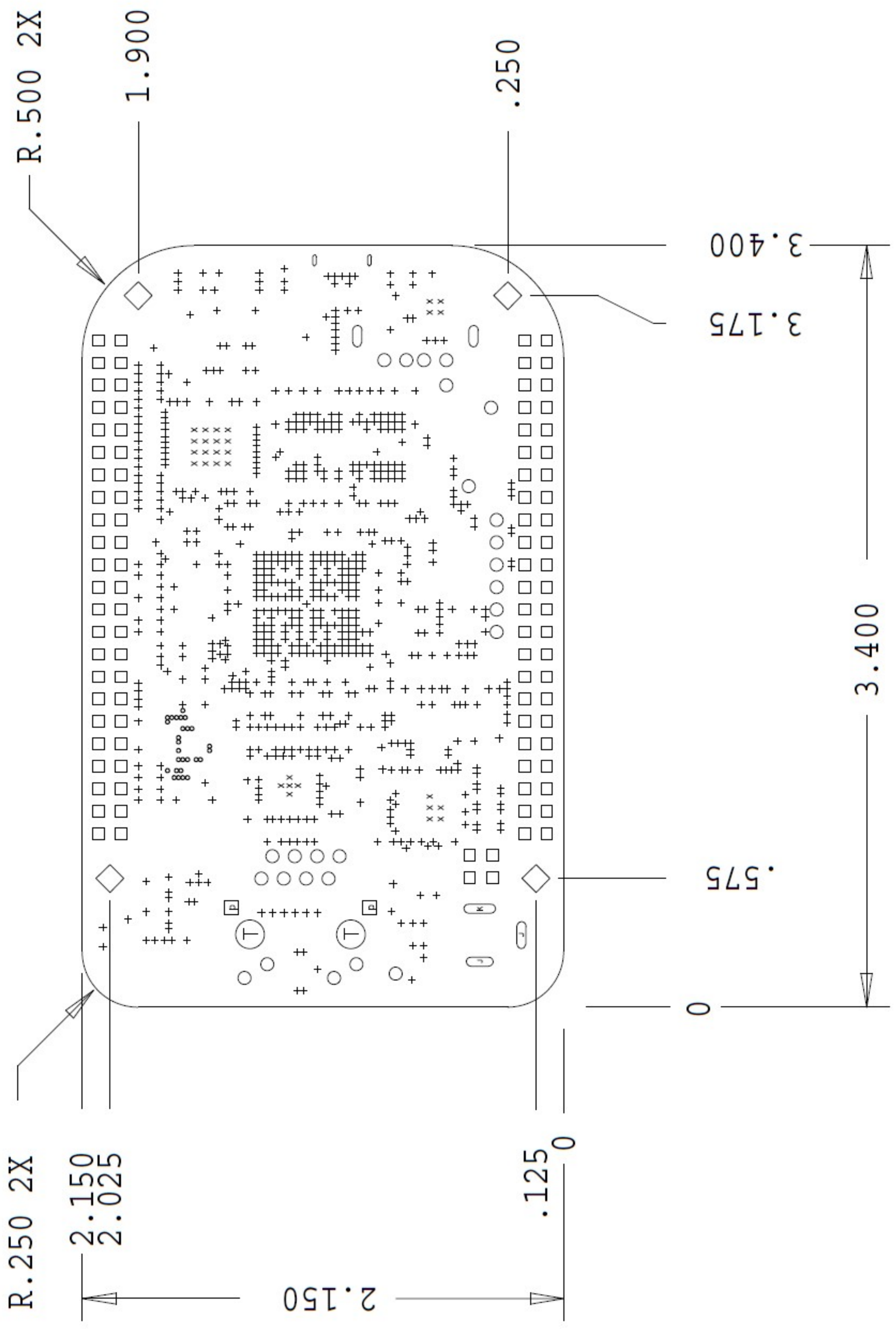


Fig. 2.73: Board Dimensions

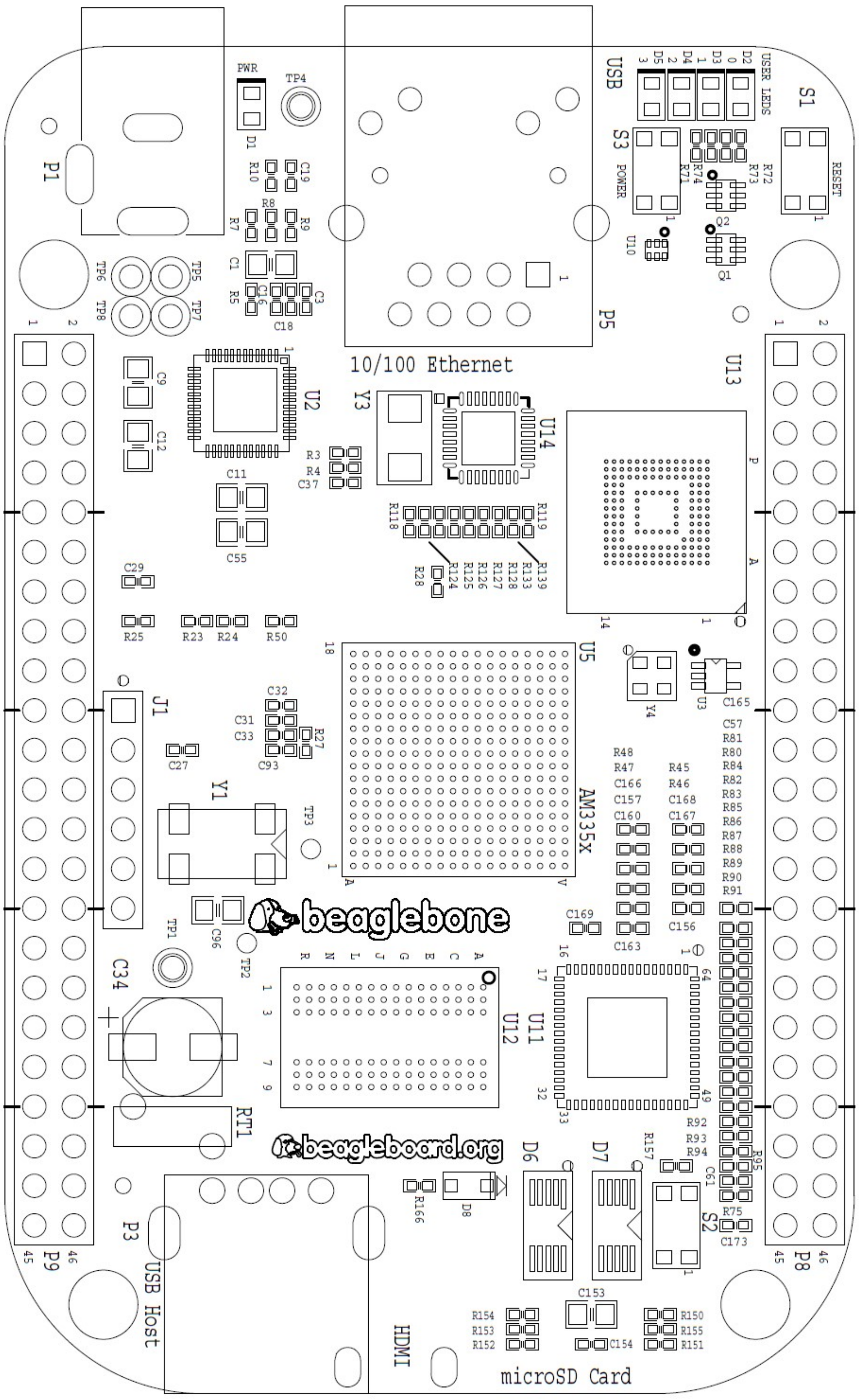


Fig. 2.74: Component Side Silkscreen

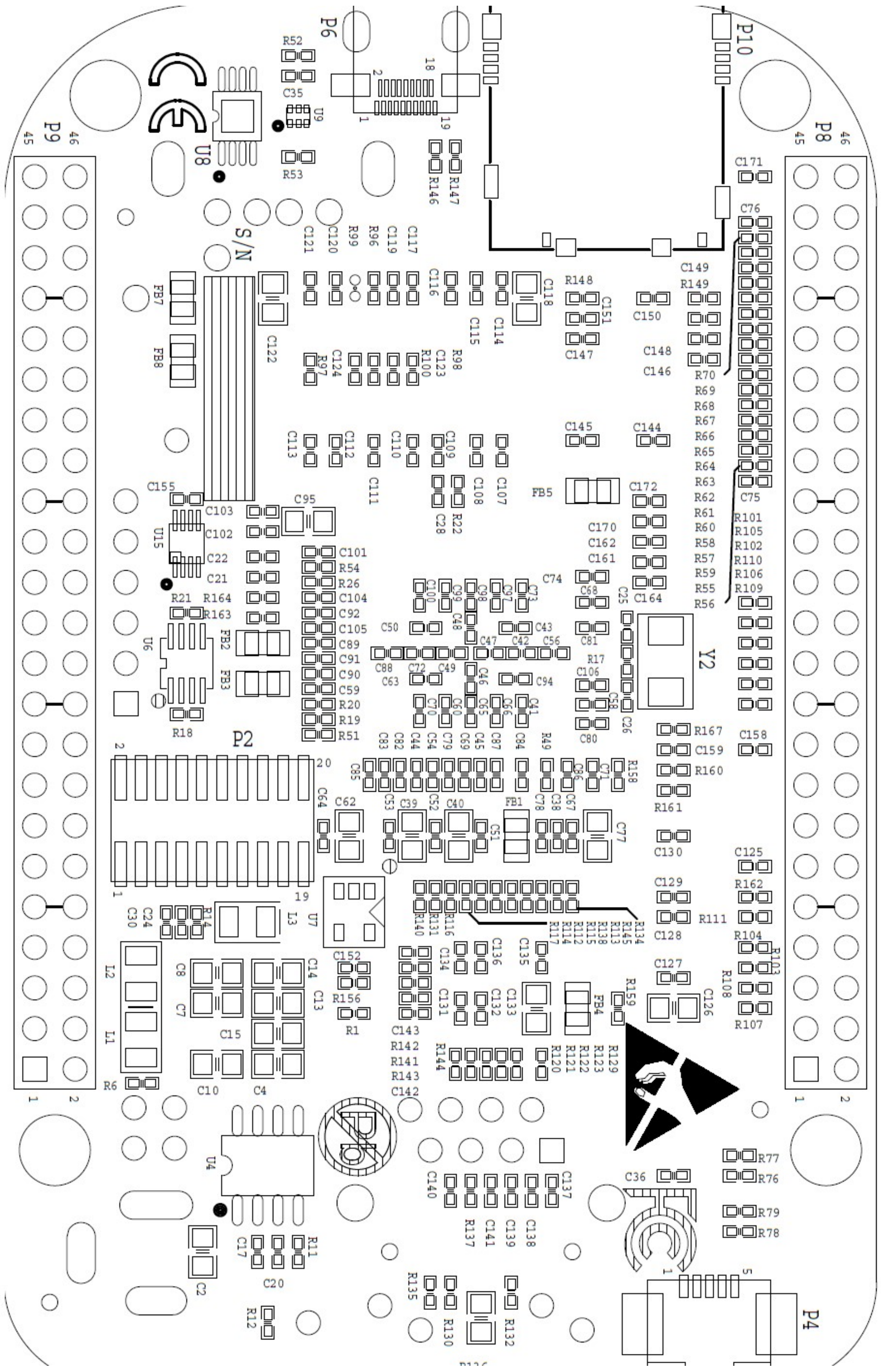


Fig. 2.75: Circuit Side Silkscreen

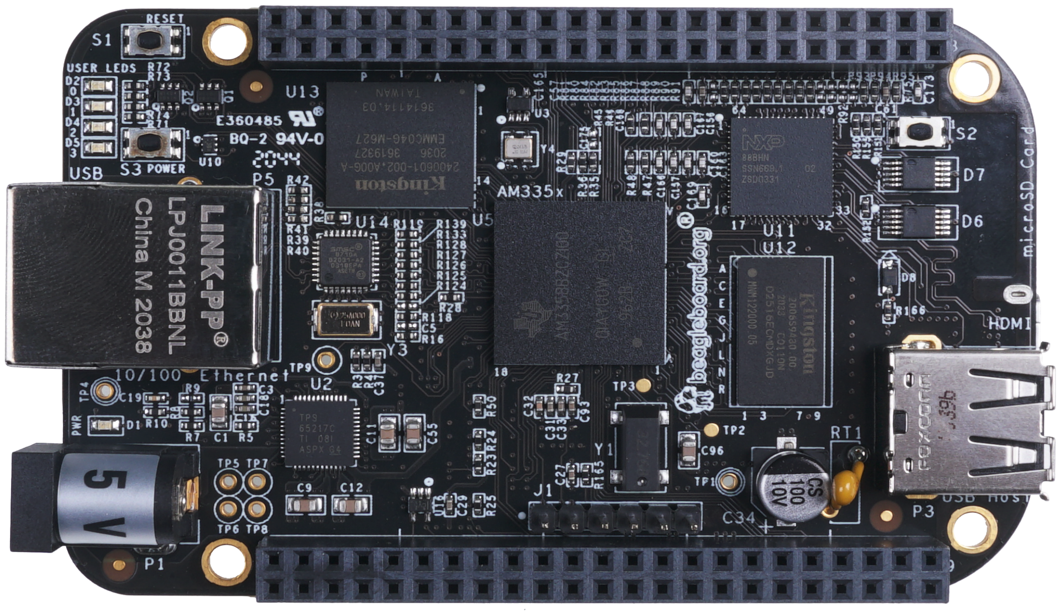


Fig. 2.76: Top Side

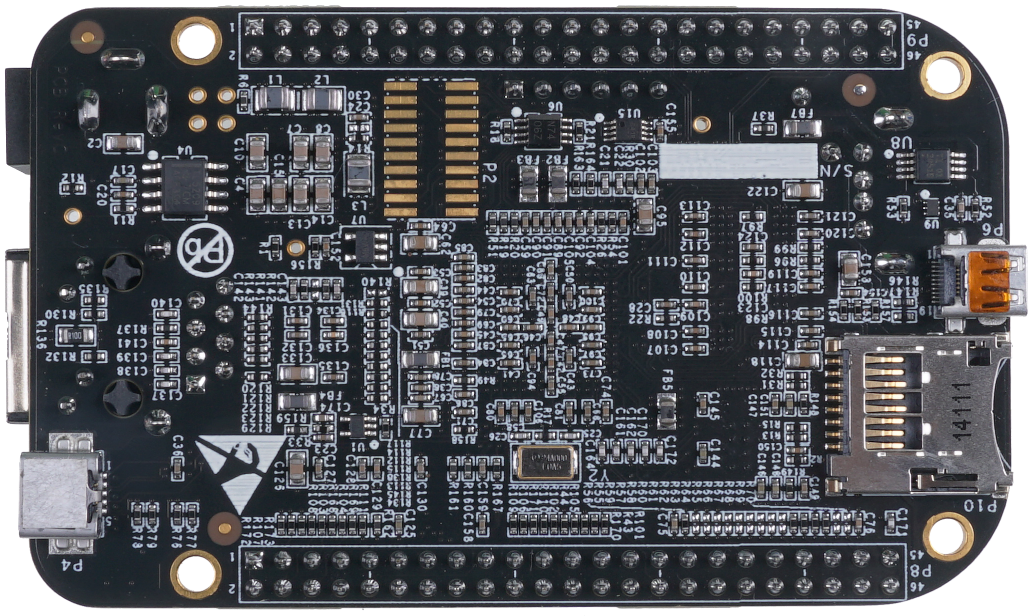


Fig. 2.77: Bottom Side

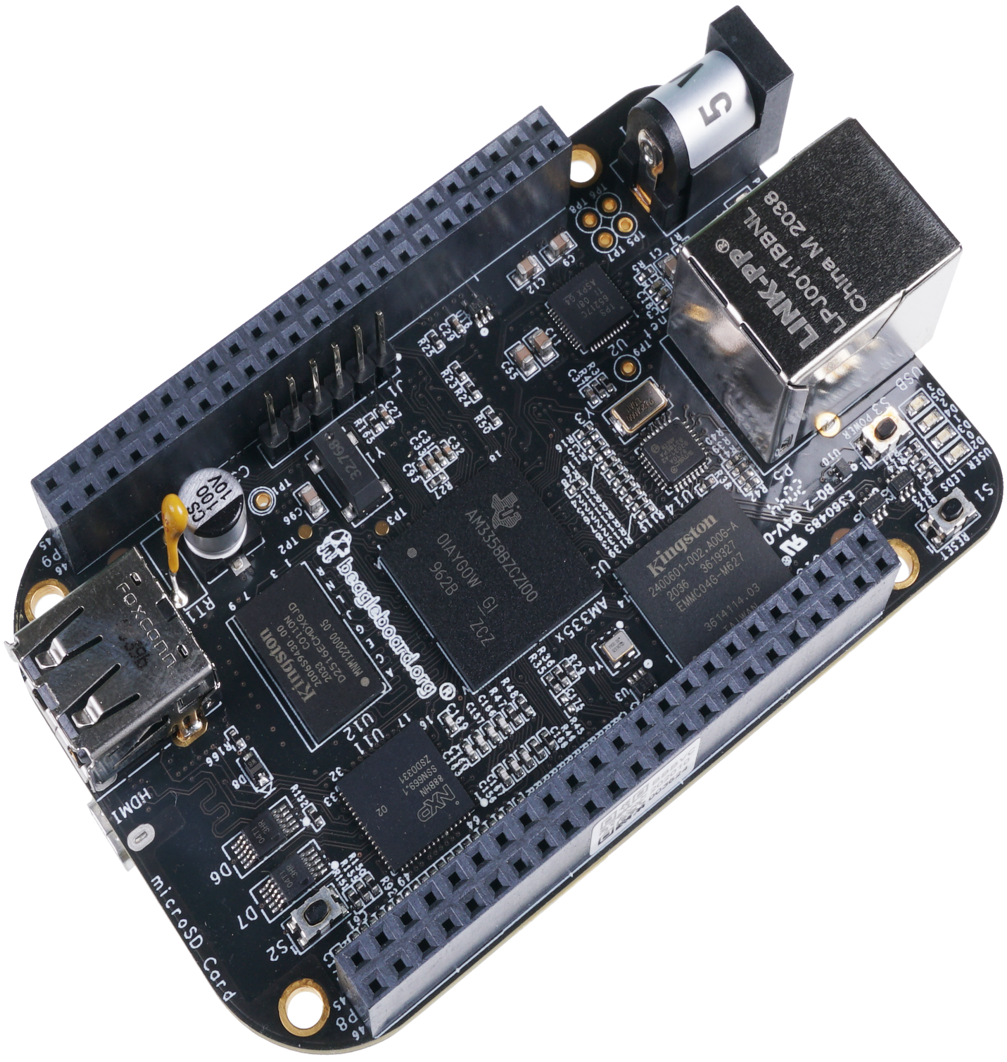


Fig. 2.78: 45 Degree Top

2.2.11 Support Information

All support for this design is through the BeagleBoard.org community at: beagleboard@googlegroups.com or http://beagleboard.org/discuss

Hardware Design

Design documentation can be found on the eMMC of the board under the documents/hardware directory when connected using the USB cable. Provided there is:

- Schematic in PDF
- Schematic in OrCAD (Cadence Design Entry CIS 16.3)
- PCB Gerber
- PCB Layout (Allegro)
- Bill of Material
- System Reference Manual (This document).

This directory is not always kept up to date in every SW release due to the frequency of changes of the SW. The best solution is to download the files from http://www.beagleboard.org/distros

We do not track SW revision of what is in the eMMC. SW is tracked separately from the HW due to the frequency of changes which would require massive relabeling of boards due to the frequent SW changes. You should always use the latest SW revision.

To see what SW revision is loaded into the eMMC follow the instructions at https://elinux.org/Beagleboard:Updating_The_Software#Checking_The_Angstrom_Image_Version

Software Updates

It is a good idea to always use the latest software. Instructions for how to update your software to the latest version can be found at:

http://elinux.org/BeagleBoneBlack#Updating_the_eMMC_Software

RMA Support

If you feel your board is defective or has issues, request an RMA by filling out the form at http://beagleboard.org/support/rma . You will need the serial number and revision of the board. The serial numbers and revisions keep moving. Different boards can have different locations depending on when they were made. The following figures show the three locations of the serial and revision number.

Trouble Shooting HDMI Issues

Many people are having issues with getting HDMI to work on their TV/Display. Unfortunately, we do not have the resources to buy all the TVs and Monitors on the market today nor go to eBay and buy all of the TVs and monitors made over the last five years to thoroughly test each and every one. We are depending on community members to help us get these tested and information provided on how to get them to work.

One would think that if it worked on a lot of different TVs and monitors it would work on most if not all of them, assuming they meet the specification. However, there are other issues that could also result in these various TVs and monitors not working. The intent is that this page will be useful in navigating some of these issues. As others also find solutions, as long as we know about them, they will be added here as well. For access to the most up to date troubleshooting capabilities, go to the support wiki at http://www.elinux.org/Beagleboard:BeagleBoneBlack_HDMI

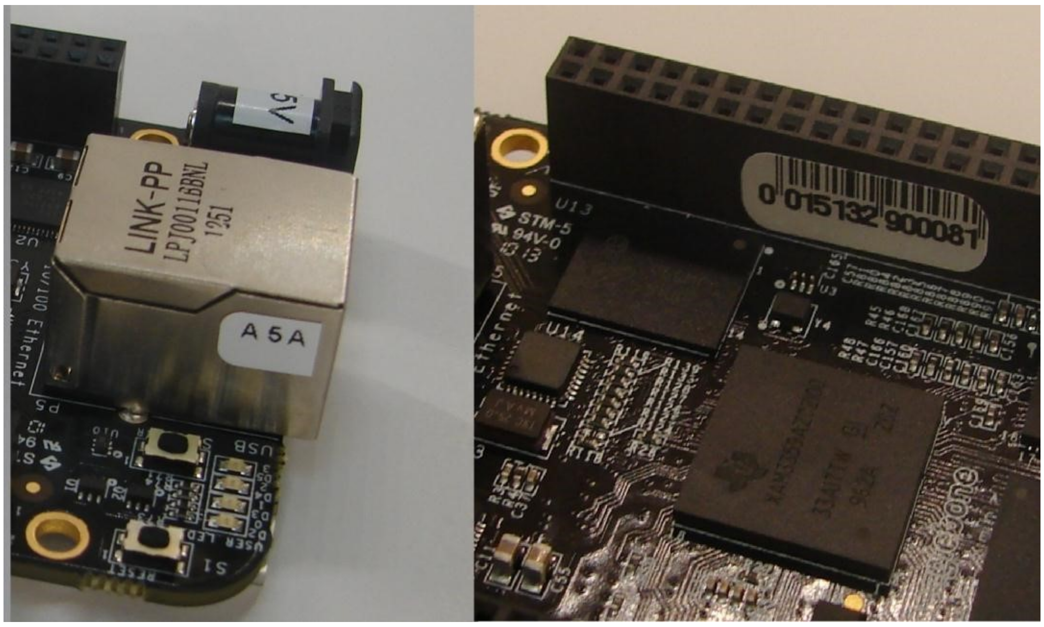


Fig. 2.79: Initial Serial Number and Revision Locations

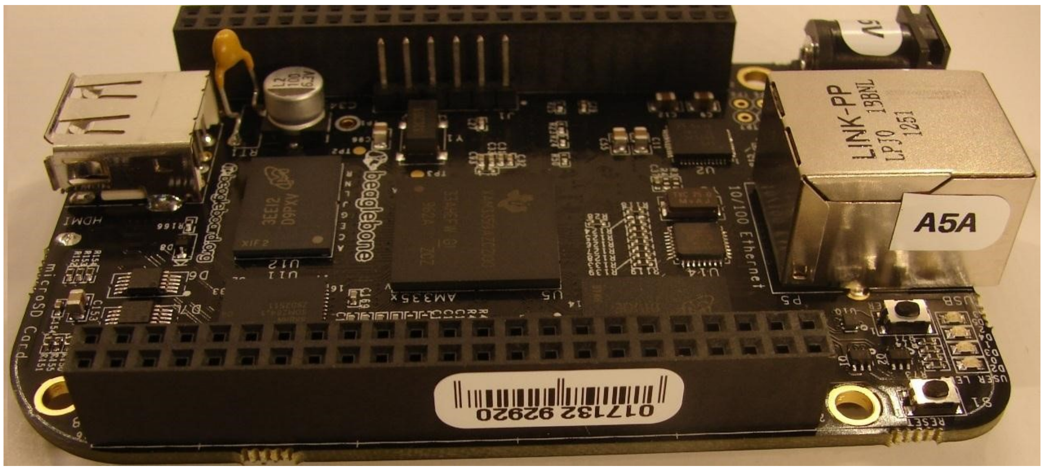


Fig. 2.80: Second Phase Serial Number and Revision Location

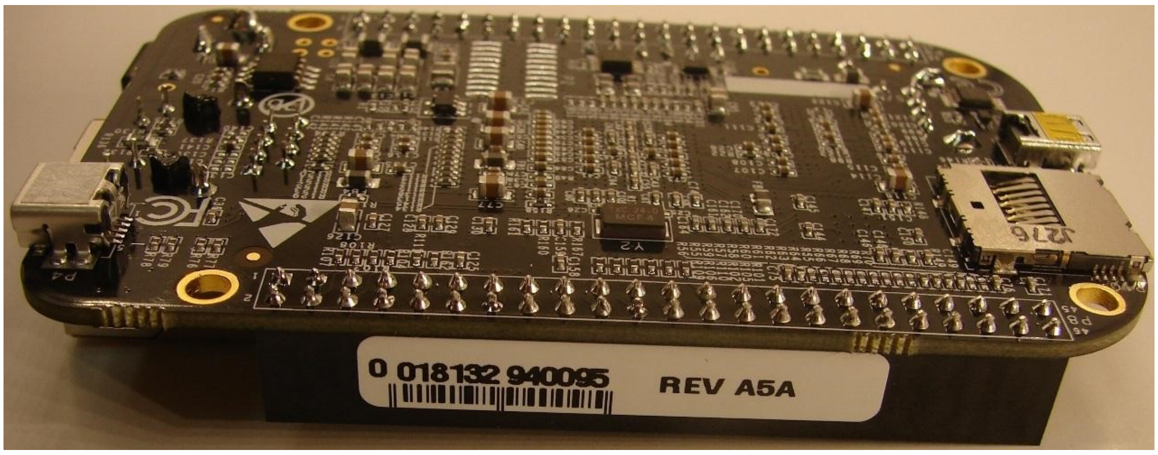


Fig. 2.81: Third Phase Serial Number and Revision Location

The early release of the Software had some issues in the HDMI driver. Be sure and use the latest SW to take advantage of the improvements.

http://www.elinux.org/Beagleboard:BeagleBoneBlack#Software_Resources

EDID EDID is the way the board requests information from the display and determines all the resolutions that it can support. The driver on the board will then look at these timings and find the highest resolution that is compatible with the board and uses that resolution for the display. For more information on EDID, you can take a look at http://en.wikipedia.org/wiki/Extended_display_identification_data

If the board is not able to read the EDID, for whatever reason, it does not have this information. A few possible reasons for this are:

- Bad cable
- Cable not plugged in all the way on both ends
- Display not powered on. (It should still work powered off, but some displays do not).

DISPLAY SOURCE SELECTION One easy thing to overlook is that you need to select the display source that matches the port you are using on the TV. Some displays may auto select, so you may need to disconnect the other inputs until you are sure the display works with the board.

OUT OF SEQUENCE Sometimes the display and the board can get confused. One way to prevent this is after everything is cabled up and running, you can power cycle the display, with the board still running. You can also try resetting the board and let it reboot to resync with the TV.

OVERSCAN Some displays use what is called overscan. This can be seen in TVs and not so much on Monitors. It causes the image to be missing on the edges, such that you cannot see them displayed. Some higher end displays allow you to disable overscan.

Most TVs have a mode that allows you to adjust the image. These are options like Normal, Wide, Zoom, or Fit. Normal seems to be the best option as it does not chop of the edges. The other ones will crop of the edges.

Taking a Nap In some cases the board can come up in a power down/screen save mode. No display will be present. This is due to the board believing that it is asleep. To come out of this, you will need to hit the keyboard or move the mouse.

Once working, the board will time out and go back to sleep again. This can cause the display to go into a power down mode as well. You may need to turn the display back on again. Sometimes, it may take a minute or so for the display to catch up and show the image.

AUDIO Audio will only work on TV resolutions. This is due to the way the specification was written. Some displays have built in speakers and others require external. Make sure you have a TV resolution and speakers are connected if they are not built in. The SW should default to a TV resolution giving audio support. The HDMI driver should default to the highest audio supported resolution.

Getting Help If you need some up to date troubleshooting techniques, we have a Wiki set up at http://elinux.org/Beagleboard:BeagleBoneBlack_HDMI

2.3 BeagleBone Blue

To optimize BeagleBone for education, BeagleBone Blue was created that integrates many components for robotics and machine control, including connectors for off-the-shelf robotic components. For education, this means you can quickly start talking about topics such as programming and control theory, without needing to spend so much time on electronics. The goal is to still be very hackable for learning electronics as well, including being fully open hardware.

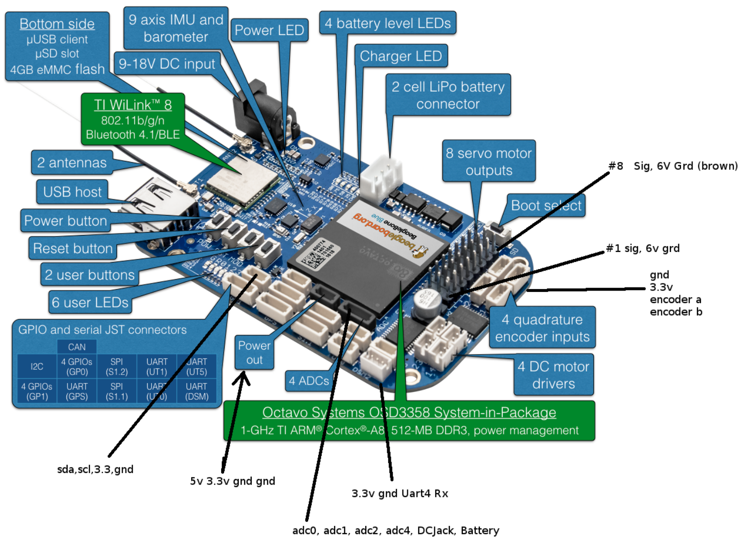
BeagleBone Blue's legacy is primarily from contributions to BeagleBone Black robotics by UCSD Flow Control and Coordinated Robotics Lab, Strawson Design, Octavo Systems, WowWee, National Instruments LabVIEW and of course the BeagleBoard.org Foundation.

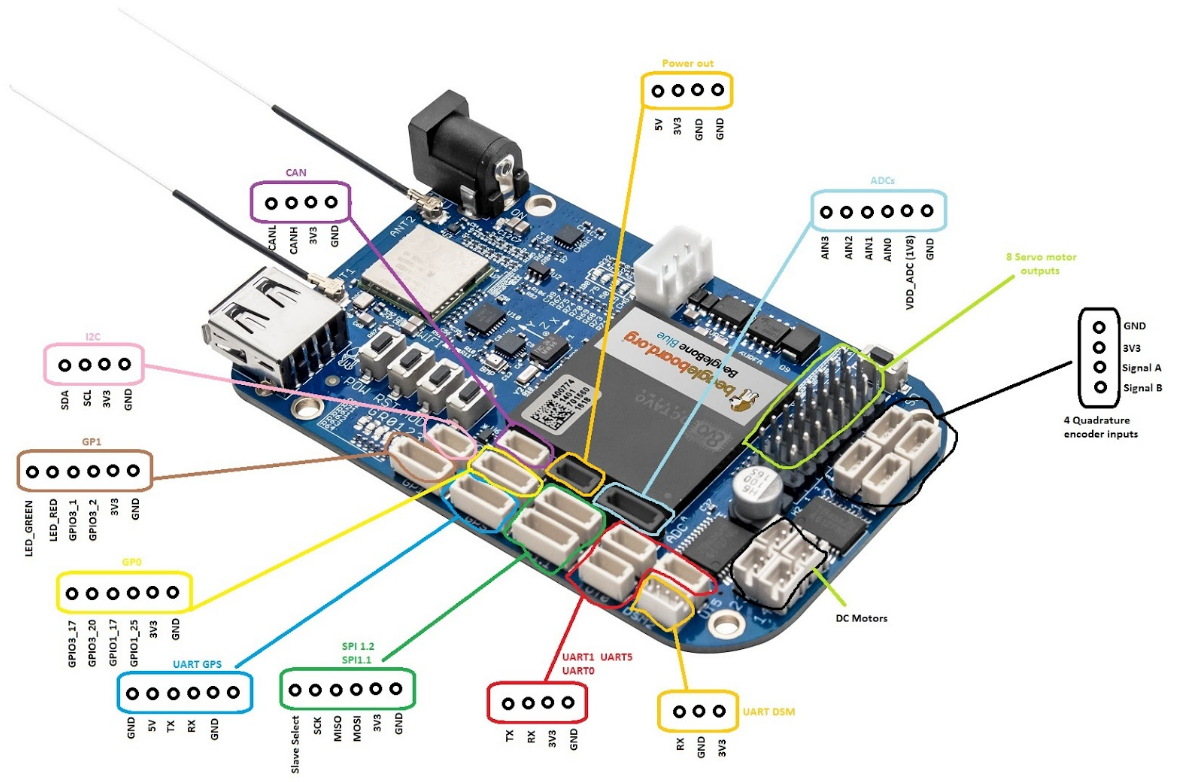
Contributors

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

2.3.1 BeagleBone Blue Pinouts





- Connector pinout details from schematic(s)
- Pin Table with some Blue : Black corelation.

UT1

UART (/dev/ttyS1)

```
config-pin P9.24 uart
config-pin P9.26 uart
```

GPS

UART (/dev/ttyS2)

```
config-pin P9.21 uart
config-pin P9.22 uart
```

2.3.2 SSH

If you don't have ssh installed, install it. (google is your friend) Then *ssh debian@192.168.7.2* The board will tell you what the password is, on my it was *temppwd*.

To change your password use the command password it will ask you what your current password is, then ask for the replacement. Then it will say it was too simple and you have to do it again. Normal stuff.

If you want to insist on using your simple password, try this.

```
sudo -s
(become superuser/root)
enter your password
password debian
 (put your simple password in)
```

(continues on next page)

(continued from previous page)

```
exit
(exit from superuser/root)
```

When you are running as root, password is more compliant and will accept simple password

2.3.3 WiFi Setup

On my network, I'm set up as ip 192.168.1.*. To turn your wifi on, do the following.

```
sudo -s
(become superuser/root)
cd /etc/network/
ifconfig
(Note the wifi inet address, if it is already set, you are done!)
connmanctl
tether wifi off
enable wifi
scan wifi
services
(at this point you should see your network appear along with other stuff, in␣
↪my case it was "AR Crystal wifi_f45eab2f1ee1_6372797774616c_managed_psk")
nano interfaces
(or whatever editor you like)
remove the comment # from the wifi lines so it now appears like
##connman: WiFi
#
connmanctl
connmanctl> tether wifi off
connmanctl> enable wifi
connmanctl> scan wifi
connmanctl> services
connmanctl> agent on
connmanctl> connect wifi_f45eab2f1ee1_6372797774616c_managed_psk
connmanctl> quit
exit
note that you will need to fill in your own network data
```

2.3.4 IP settings

You will usually want to have a fixed ip if you are doing robotics, so you have a standard ip to connect to. If you are already connected in dhcp you can borrow some of the settings from that to use in your new configurations.

```
route
```

make a note of the default one, (in the example below 192.168.1.1)

```
cat /etc/resolv.conf
```

make a note of the nameserver, (in the example below 8.8.8.8)

In my case I wanted 192.168.1.7 to do this,

```
sudo -s
connmanctl config wifi_f45eab2f1ee1_6372797774616c_managed_psk --ipv4 manual␣
↪192.168.1.7 255.255.255.0 192.168.1.1 --nameservers 8.8.8.8
exit
```

the –ipv4 says to use ipv4 settings (as opposed to ipv6), the manual means we are setting the values. 192.168.1.7 is the ip address we want. (use your own of course). 255.255.255.0 is the network mask

192.168.1.1 is the route to the internet. (You're might be different, but this is common). –nameservers 8.8.8.8 says where to find the ip address for a given domain name. the 8.8.8.8 says use's googles

2.3.5 Flashing Firmware

Overview

Most Beaglebones have a built in 4 GB SD card known as a eMMC (embedded MMC). When the boards are made the eMMC is "flashed" with some version of the BeagleBone OS that is usually outdated. Therefore, whenever receiving the BeagleBone it is recommend that you update the eMMC with the last version of the BeagleBone OS or a specific version of it if someone tells you otherwise.

Required Items

1. Micro sd card. 4 GB minimum
2. Micro sd card reader or a built in sd card reader for your PC
3. BeagleBone image you want to flash.
4. Etcher utility for your PC's OS.

Steps Overview

1. Burn the image you want to flash onto a micro sd card using the Etcher utility.
2. Boot the BeagleBone like normal and place the micro sd card into the board once booted.
3. Update the micro sd card image so its in "flashing" mode.
4. Insert micro sd card, remove power from the BeagleBone, hold sd card select button, power up board
5. Let the board flash

Windows PCs

1. Download the BeagleBone OS image you want to use.
2. Use the Etcher utility to burn the BeagleBone image you want to use on the micro sd card you plan on using.
3. Make sure you don't have the micro sd card plugged into your board.
4. Boot the board
5. Connect to the board via serial or ssh so that your on the command prompt.
6. Plug the micro sd card into the board.
7. Type dmesg in the terminal window
8. The last line from the output should say something like (the numbering may differ slightly):
   - "`[ 2805.442940] mmcblk0: p1`"
9. You want to take the above and combine it together by removing the : and space. For the above example it will change to "mmcblk0p1"
10. In the terminal window enter the following commands:

```
mkdir sd_tmp
sudo mount /dev/mmcblk0p1 sd_tmp
sudo su
echo "cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh" >> sd_
↪tmp/boot/uEnv.txt
exit
sudo umount sd_tmp
```

11. Now power off your board
12. Hold the update button labeled SD (the one by itself) to boot off the sdcard.
13. Restart (RST button) or power up (while still pushing SD button).

Flashing can take some minutes. ## Linux/Mac PCs 1. Download the BeagleBone OS image you want to use. 1. Use the Etcher utility to burn the BeagleBone image you want to use on the micro sd card you plan on using. 1. On the SD card edit the file `/boot/uEnv.txt` in order for the SD card contents to be flashed onto the firmware eMMC. (Otherwise the BBBL will do no more than boot the SD image.) Uncomment the line containing `init-eMMC-flasher-v<number>.sh` either manually or using these commands substituting X with what your SD card shows in `/dev/`: * `sudo mount /dev/emmcblkXp1 /mnt` * `cd /mnt` * `sed -i 's_#[ ]*\(cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v[0-9]\+.*\.sh\)_\1_' boot/uEnv.txt`

1. Eject the sdcard from your computer.
2. Put it into your BeagleBoneBlue.
3. If your board was already powered on then power it off
4. Hold the update button labeled SD (the one by itself) to boot off the sdcard.
5. Restart (RST button) or power up (while still pushing SD button).

Flashing can take some minutes.

How to tell if it is flashing? At first a blue heartbeat is shown indicating the image is booted. On flash procedure start, the blue user LEDs light up in a "larson scanner" or "cylon" pattern (back and forth).

When finished, either all blue LEDs are on or the board is already switched off.

If the LEDs are on for a long time then it may indicate failure e.g. wrong image. Can be verified if boot fails, i.e. board turns off again shortly after power up.

2.3.6 Play with the code

The board has some code built in to the system that can allow you to try out the various options. They all start with rc

```
rc_balance              rc_dsm_passthrough      rc_test_encoders
rc_battery_monitor      rc_kill                 rc_test_filters
rc_benchmark_algebra    rc_spi_loopback         rc_test_imu
rc_bind_dsm             rc_startup_routine      rc_test_motors
rc_blink                rc_test_adc             rc_test_polynomial
rc_calibrate_dsm        rc_test_algebra         rc_test_servos
rc_calibrate_escs       rc_test_barometer       rc_test_time
rc_calibrate_gyro       rc_test_buttons         rc_test_vector
rc_calibrate_mag        rc_test_cape            rc_uart_loopback
rc_check_battery        rc_test_dmp             rc_version
rc_check_model          rc_test_drivers
rc_cpu_freq             rc_test_dsm
```

Try them out to try out the various functions of the board. The source code for these tests and demos is at Robotics cape installer at github

2.3.7 BeagleBone Blue tests

ADC

- Grove Rotary Angle Sensor See output on adc_1 source

```
rc_test_adc
```

GP0

- Grove single GPIO output modules like LED Socket Kit

```
cd /sys/class/gpio;echo 49 >export;cd gpio49;echo out >direction;while sleep␣
↪1;do echo 0 >value;sleep 1;echo 1 >value;done
```

- Grove single GPIO input modules like IR Distance Interrupter or Touch Sensor

```
cd /sys/class/gpio;echo 49 >export;cd gpio49;echo in >direction;watch -n0␣
↪cat value
```

GP1

- Grove single GPIO output modules like LED Socket Kit

```
cd /sys/class/gpio;echo 97 >export;cd gpio97;echo out >direction;while sleep␣
↪1;do echo 0 >value;sleep 1;echo 1 >value;done
```

- Grove single GPIO input modules like IR Distance Interrupter or Touch Sensor

```
cd /sys/class/gpio;echo 97 >export;cd gpio97;echo in >direction;watch -n0␣
↪cat value
```

UT1

- Grove GPS

```
tio /dev/ttyO1 -b 9600
```

GPS

- GPS Receiver - EM-506

```
tio /dev/ttyO2 -b 4800
```

I2C

Grove I2C modules The Linux kernel source has some basic IIO SYSFS interface documentation which might provide a little help for understanding reading these entries. The ELC2017 conference also had an IIO presentation.

- Digital Light Sensor

```
cd /sys/bus/i2c/devices/i2c-1;echo tsl2561 0x29 >new_device;watch -n0 cat 1-
↪0029/iio\:device0/in_illuminance0_input
```

- Temperature & Humidity Sensor

```
cd /sys/bus/i2c/devices/i2c-1;echo th02 0x40 >new_device;watch -n0 cat 1-
↪0040/iio\:device0/in_temp_raw
```

Motors

```
rc_test_motors
```

2.3.8 Accessories

Note: #TODO#: We are going to work on a unified accessories page for all the boards and it should replace this.

Chassis and kits

- EduMIP
- Pololu Romi Chassis with geared motors
  - Wheel encoders
  - Chassis - Black
- Sprout Runt Rover

Cases

Cable assemblies and sub-assemblies

Beware; purchased pre-made connector assembly wire colors may not reflect true pin designations. These assemblies are readily available from Digi-Key, SparkFun, Hobby King, Pololu and Cables and Connectors.

JST Connector Bundle

Renaissance Robotics JST Jumper Bundle

Four of the 2-pin JST ZH (1.5mm pitch) connectors, with 150mm 28AWG wires, for motors,
Eight of the 4-pin JST SH (1mm pitch) connectors, with 150mm 28AWG wires, for encoders, UART, I2C, CAN, PWR, and
Four of the 6-pin JST SH (1mm pitch) connectors, with 150mm 28AWG wires, for SPI, GPS, GPIO, ADC.
Renaissance Robotics JST Jumper Bundle

Conrad BeagleBoard Kabel BB-Blue-Kabelset

10x 4-Pin JST-SH
6x 6-Pin JST-SH
4x 2-Pin JST-ZH
1x 3-Pin JST-ZH
BeagleBoard Kabel BB-Blue-Kabelset (Conrad.de)

UART, I2C, CAN, Quadrature encoders, PWR

4-wire JST-SH (1mm pitch)

- 4-wire Grove cable (Digi-Key)
- Hobby King SKU 258000190-0
- SparkFun PN 10359
- Cables and Connectors 4" ribbon PN #4904
- Digi-Key wires
- Digi-Key housings

SPI, GPIO, ADC

6-wire JST-SH (1mm pitch)

- Hobby King SKU 258000192-0
- SparkFun PN 10361
- Cables and Connectors 50cm length PN #49406
- Digi-Key wires
- Digi-Key housings
- 6-wire Grove cable (4 populated) (Digi-Key)

Motors

2-wire JST-ZH (1.5mm pitch)

- Digi-Key wires
- Digi-Key receptacle

DSM

3-wire JST-ZH (1.5mm pitch)

- Pololu PN# 2411

microUSB standard

Batteries 2S1P LiPo with 3-wire JST-XH (2.5mm pitch) charge connection

- Hobby King 1000mAh 2S 20C LiPo
- Hobby King 1600mAh 2S 20C LiPo

Power supplies

12V with 5.5mm/2.1mm center positive

- Jameco: supply and power cord
- Hobby King 12V 3A supply

Motors

Servo motors

6V DC

- Parallax Inc. 900-00005 Standard Servo
- Hobby King SKU HD-1900A
- TowerPro SG92R-7

DC motors

6V, typically geared

- SparkFun Hobby Gearmotor - 200 RPM (Pair)
- SparkFun Hobby Motor - Gear

Radio remotes

- Hobby King OrangeRX satellite receiver
- Spektrum DSM2 Remote Receiver

GPS

- Sparkfun GPS Receiver - EM-506 (48 Channel)
- Adafruit Ultimate GPS breakout
- Ublox Neo-M8N GPS with Compass
- SeeedStudio Grove - GPS

Replacement antennas

- LSR PIFA
- LSR Dipole: antenna and cable
- Anaren U.FL 2.4GHz 6MM Antenna
- TI approved antennas

USB devices

USB cameras

- Logitech C270
- Logitech C920

SPI devices

SPI TFT displays

- Adafruit 2.4" LCD breakout

I2C devices

- See **:ref:'One-Liner-Module-Tests#i2c <beaglebone-blue-one-liner-tests>'__**
- See *Using I2C with Linux drivers*.

UART devices

Computer serial adapters

- Sparkfun FTDI Cable 5V VCC-3.3V I/O
- Adafruit FTDI Serial TTL-232 USB Cable

Bluetooth devices

- WowWee Groove Cube Speaker

2.3.9 Frequently Asked Questions (FAQs)

Are there any books to help me get started?

The book BeagleBone Robotic Projects, Second Edition specifically covers how to get started building robots with BeagleBone Blue.

For more general books on BeagleBone, Linux and other related topics, see https://beagleboard.org/books.

What system firmware should I use for starting to explore my BeagleBone Blue?

Download the latest 'IoT' image from https://www.beagleboard.org/distros. As of this writing, that image is https://debian.beagleboard.org/images/bone-debian-9.5-iot-armhf-2018-10-07-4gb.img.xz.

Use http://etcher.io for writing that image to a 4GB or larger microSD card.

Power-up your BeagleBone Blue with the newly created microSD card to run this firmware image.

What is the name of the access point SSID and password default on BeagleBone Blue?

SSID: BeagleBone-XXXX where XXXX is based upon the board's assigned unique hardware address

Password: BeagleBone

I've connected to BeagleBone Blue's access point. How do I get logged into the board?

Browse to http://192.168.8.1:3000 to open the Cloud9 IDE and get access to the Linux command prompt.

If you've connected via USB instead, the address will be either http://192.168.6.2:3000 or http://192.168.7.2:3000, depending on the USB networking drivers provided by your operating system.

How do I connect BeagleBone Blue to my own WiFi network?

From the bash command prompt in Linux:

```
sudo -s (become superuser/root)

connmanctl
   connmanctl> tether wifi off (not really necessary on latest images)
   connmanctl> enable wifi (not really necessary)
   connmanctl> scan wifi
   connmanctl> services (at this point you should see your network
   appear along with other stuff, in my case it was "AR Crystal wifi_
↪f45eab2f1ee1_6372797774616c_managed_psk")
   connmanctl> agent on
```

(continues on next page)

(continued from previous page)

```
    connmanctl> connect wifi_f45eab2f1ee1_6372797774616c_managed_psk
    connmanctl> quit
```

Where can I find examples and APIs for programming BeagleBone Blue?

Programming in C: http://www.strawsondesign.com/#!manual-install

Programming in Python: https://github.com/mcdeoliveira/rcpy

Programming in Simulink: https://www.mathworks.com/hardware-support/beaglebone-blue.html

My BeagleBone Blue fails to run successful tests

You've tried to run rc_test_drivers to ensure your board is working for DOA warranty tests, but it errors. You should first look to fixing your bootloader as described http://strawsondesign.com/docs/librobotcontrol/installation.html#installation_s5

I'm running an image off of a microSD card. How do I write it to the on-board eMMC flash?

Refer to the "Flashing Firmware" page: https://git.beagleboard.org/beagleboard/beaglebone-blue/-/wikis/Flashing-firmware

Meanwhile, as root, run the /opt/scripts/tools/eMMC/bbb-eMMC-flasher-eewiki-ext4.sh script which will create a copy of the system in your microSD to a new single ext4 partition on the on-board eMMC.

I've written the latest image to a uSD card, but some features aren't working. How do I make it run properly?

It is possible you are running an old bootloader off of the eMMC. While power is completely off, hold the SD button (near the servo headers) while applying power. You can release the button as soon the power LED comes on. This will make sure the bootloader is loaded from microSD and not eMMC.

Verify the running image using version.sh via:

```
sudo /opt/scripts/tools/version.sh
```

The version.sh output will tell you which version of bootloader is on the eMMC or microSD. Future versions of version.sh might further inform you if the SD button was properly asserted on power-up.

One you've booted the latest image, you can update the bootloader on the eMMC using /opt/scripts/tools/developers/update_bootloader.sh. Better yet, read the above FAQ on flashing firmware.

I've got my on-board eMMC flash configured in a nice way. How do I copy that to other BeagleBone Blue boards?

As root, run the /opt/scripts/tools/eMMC/beaglebone-black-make-microSD-flasher-from-eMMC.sh script with a blank 4GB or larger microSD card installed and wait for the script to complete execution.

Remove the microSD card.

Boot your other BeagleBone Blue boards off of this newly updated microSD card and wait for the flashing process to complete. You'll know it successfully started when you see the "larson scanner" running on the LEDs. You'll know it successfully completed when it shuts off the board.

Remove the microSD card.

Reboot your newly flashed board.

I have some low-latency I/O tasks. How do I get started programming the BeagleBone PRUs?

There is a "Hello, World" app at https://gist.github.com/jadonk/2ecf864e1b3f250bad82c0eae12b7b64 that will get you blinking the USRx LEDS.

The libroboticscape software provides examples that are pre-built and included in the BeagleBone Blue software images for running the servo/ESC outputs and fourth quadrature encoder input. You can use those firmware images as a basis for building your own: https://github.com/StrawsonDesign/Robotics_Cape_Installer/tree/master/pru_firmware

You can find some more at https://beagleboard.org/pru

Are there available mechanical models?

A community contributed model is available at https://grabcad.com/library/beaglebone-blue-1

What is the operating temperature range?

'0..70' due to processor, else '-20..70'

What is the DC motor drive strength?

This is dictated by the 2 cell LiPo battery input, the TB6612FNG motor drivers and the JST-ZH connectors

- Voltage: 6V-8.4V (typical)
- Current: 1A (maximum for connectors) / 1.2A (maximum average from drivers) / 3.2A (peak from drivers) per channel

2.4 BeagleBone AI

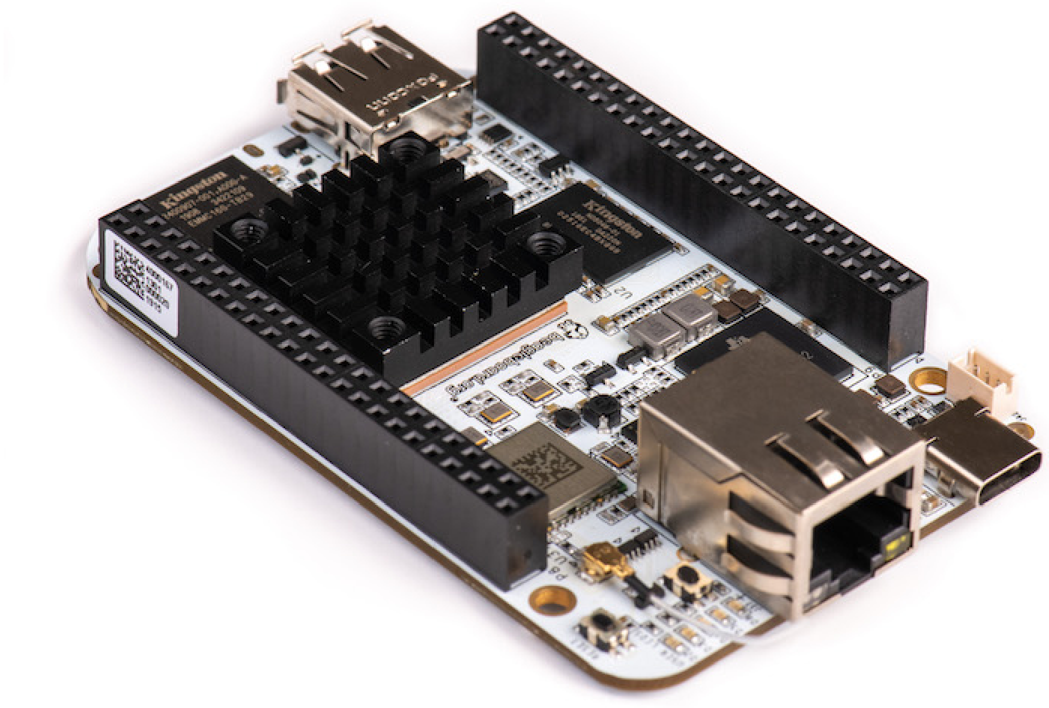
Contributors

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

2.4.1 Introduction

Built on the proven BeagleBoard.org® open source Linux approach, BeagleBone® AI fills the gap between small SBCs and more powerful industrial computers. Based on the Texas Instruments AM5729, developers have access to the powerful SoC with the ease of BeagleBone® Black header and mechanical compatibility. BeagleBone® AI makes it easy to explore how artificial intelligence (AI) can be used in everyday life via TI C66x digital-signal-processor (DSP) cores and embedded-vision-engine (EVE) cores supported through an optimized TIDL machine learning OpenCL API with pre-installed tools. Focused on everyday automation in industrial, commercial and home applications.



2.4.2 Change History

Rev A0

Initial prototype revision. Not taken to production. eMMC flash image provided by Embest.

Rev A1

Second round prototype.

- Fixed size of mounting holes.
- Added LED for WiFi status.
- Added microHDMI.
- Changed eMMC voltage from 3.3V to 1.8V to support HS200.
- Changed eMMC from 4GB to 16GB.
- Changed serial debug header from 6-pin 100mil pitch to 3-pin 1.5mm pitch.
- Switched expansion header from UART4 to UART5. The UART4 pins were used for the microHDMI.

eMMC flash image provided by Embest.

Rev A1a

Alpha pilot-run units and initial production.

- Added pull-down resistor on serial debug header RX line.

Alpha pilot-run eMMC flash image: https://debian.beagleboard.org/images/bbai-pilot-20190408.img.xz

Production eMMC flash image: http://debian.beagleboard.org/images/am57xx-eMMC-flasher-debian-9.9-lxqt-armhf-2019-08-03-4gb.img.xz

Rev A2

Proposed changes.

- HW: need pull-down on console uart RX line.
- HW: position of microSD may impact existing case designs.
- HW: P9.13 does not have a GPIO.
- HW: HDMI hotplug detection not working.
- HW: add extra DCAN.
- HW: wire mods required to enable JTAG.
- HW: Small I2C nvmem/eeprom for board identifier.

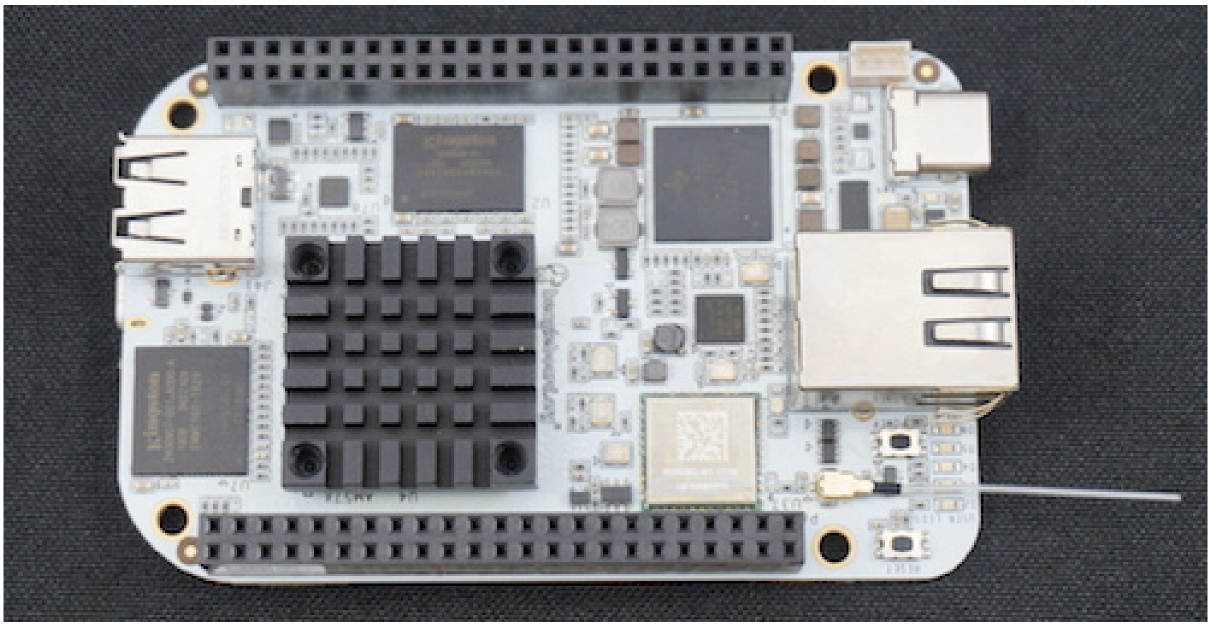
2.4.3 Connecting Up Your BeagleBone AI

What's In the Box

BeagleBone® AI comes in the box with the heat sink and antenna already attached. Developers can get up and running in five minutes with no microSD card needed. BeagleBone® AI comes preloaded with a Linux distribution. In the box you will find:

- BeagleBone® AI
- Quick Start Guide

TODO: Add links to the design materials for both



What's Not in the Box

You will need to purchase:

- USB C cable or USB C to USB A cable
- MicroSD Card (optional)
- Serial cable (optional)

More information or to purchase a replacement heat sink or antenna, please go to these websites:

- Antenna
- Heat Sink

Fans

The pre-attached heat sink has M3 holes spaced 20x20 mm. The height of the heat sink clears the USB type A socket, and all other components on the board except the 46-way header sockets and the Ethernet socket.

If you run all of the accelerators or have an older software image, you'll likely need fan. To find a fan, visit the link to fans in the FAQ.

Caution: BeagleBone AI can run **HOT**! Even without running the accelerators, getting up to 70C is not uncommon.

Official BeagleBone Fan Cape: https://www.newark.com/element14/6100310/beaglebone-ai-fan-cape/dp/50AH3704

TODO: create short-links for any long URLs so that text works.

Main Connection Scenarios

This section will describe how to connect the board for use. The board can be configured in several different ways. Below we will walk through the most common scenarios. NOTE: These connection scenarios are dependent on the software image presently on your BeagleBone® AI. When all else fails, follow the instructions at https://beagleboard.org/upgrade

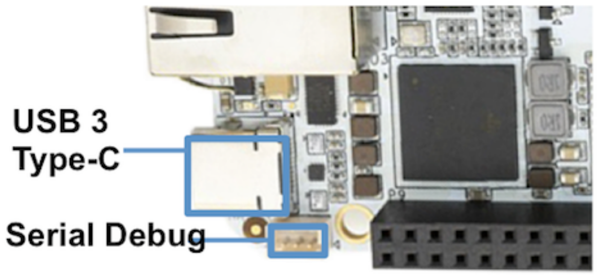
- *Tethered to a PC via USB C cable*
- *Standalone Desktop with powered USB hub, display, keyboard and mouse*
- *Wireless Connection to BeagleBone® AI*

Tethered to a PC

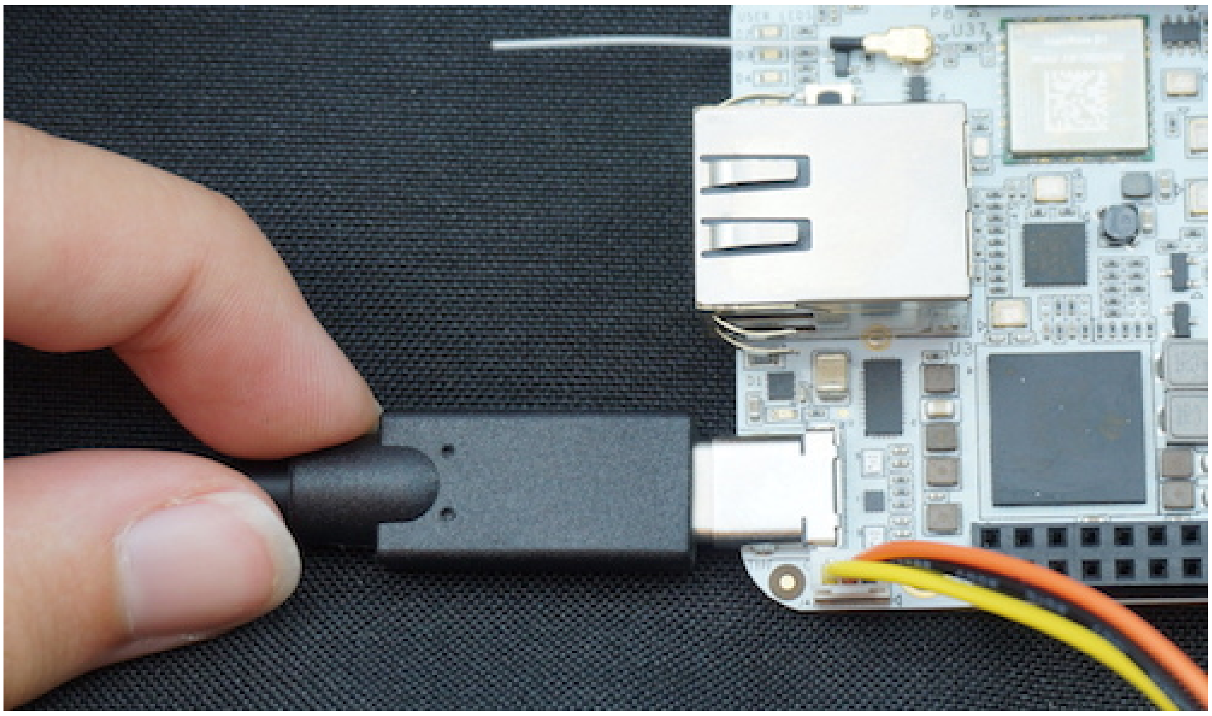
The most common way to program BeagleBone® AI is via a USB connection to a PC. If your computer has a USB C type port, BeagleBone® AI will both communicate and receive power directly from the PC. If your computer does not support USB C type, you can utilize a powered USB C hub to power and connect to BeagleBone® AI which in turn will connect to your PC. You can also use a powered USB C hub to power and connect peripheral devices such as a USB camera. After booting, the board is accessed either as a USB storage device or via the browser on the PC. You will need Chrome or Firefox on the PC.

NOTE:Start with this image "am57xx-eMMC-flasher-debian-10.3-iot-tidl-armhf-2020-04-06-6gb.img.xz" loaded on your BeagleBone® AI.

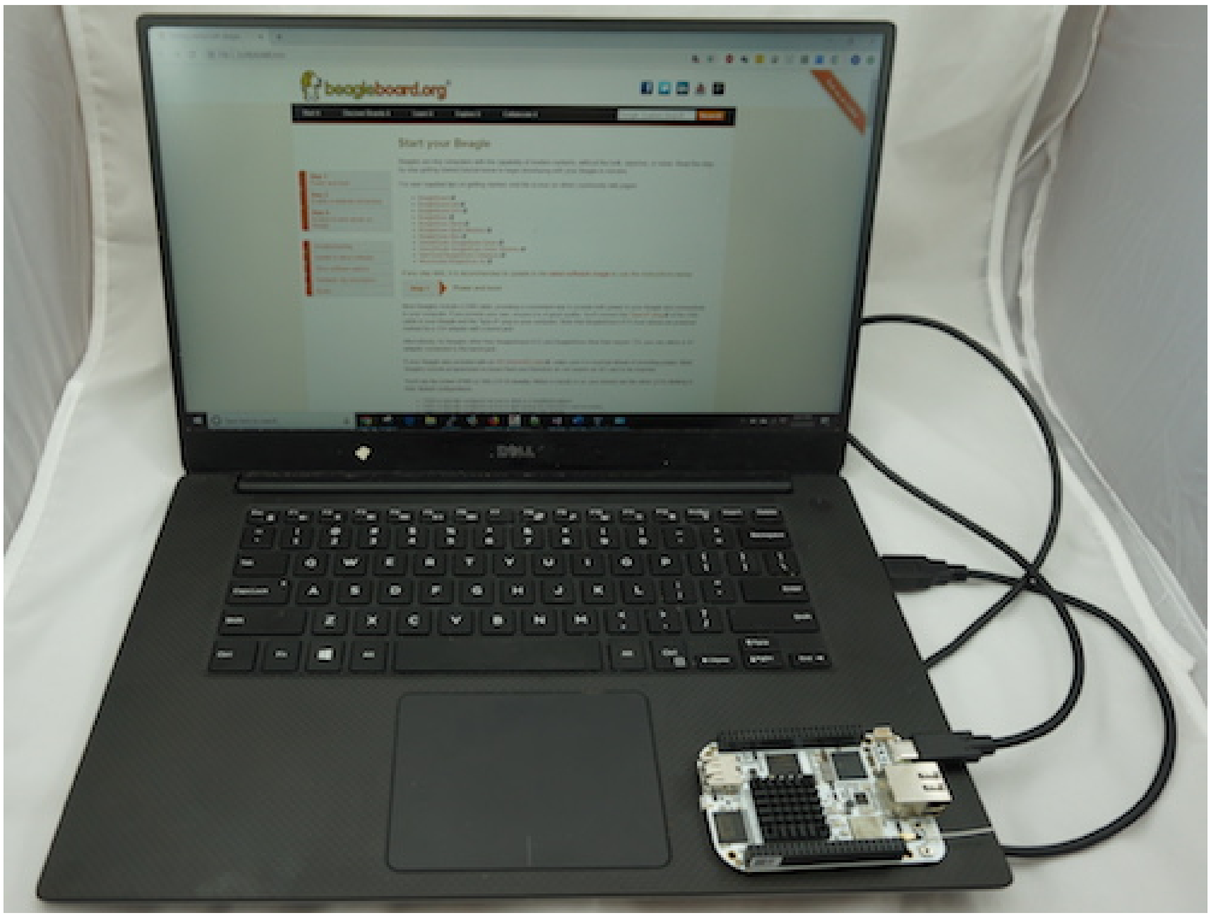
1. Locate the USB Type-C connector on BeagleBone® AI



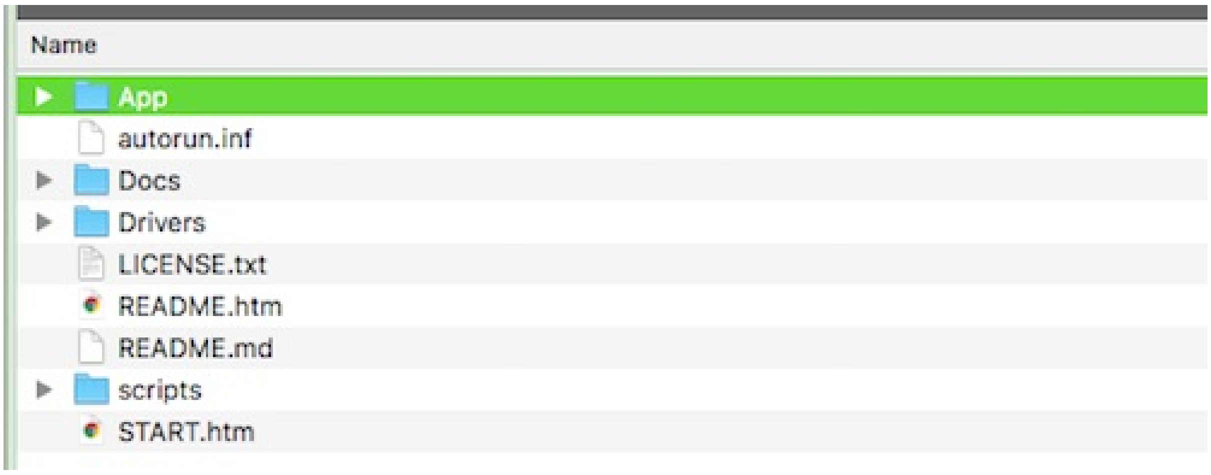
2. Connect a USB type-C cable to BeagleBone® AI USB type-C port.



3. Connect the other end of the USB cable to the PC USB 3 port.

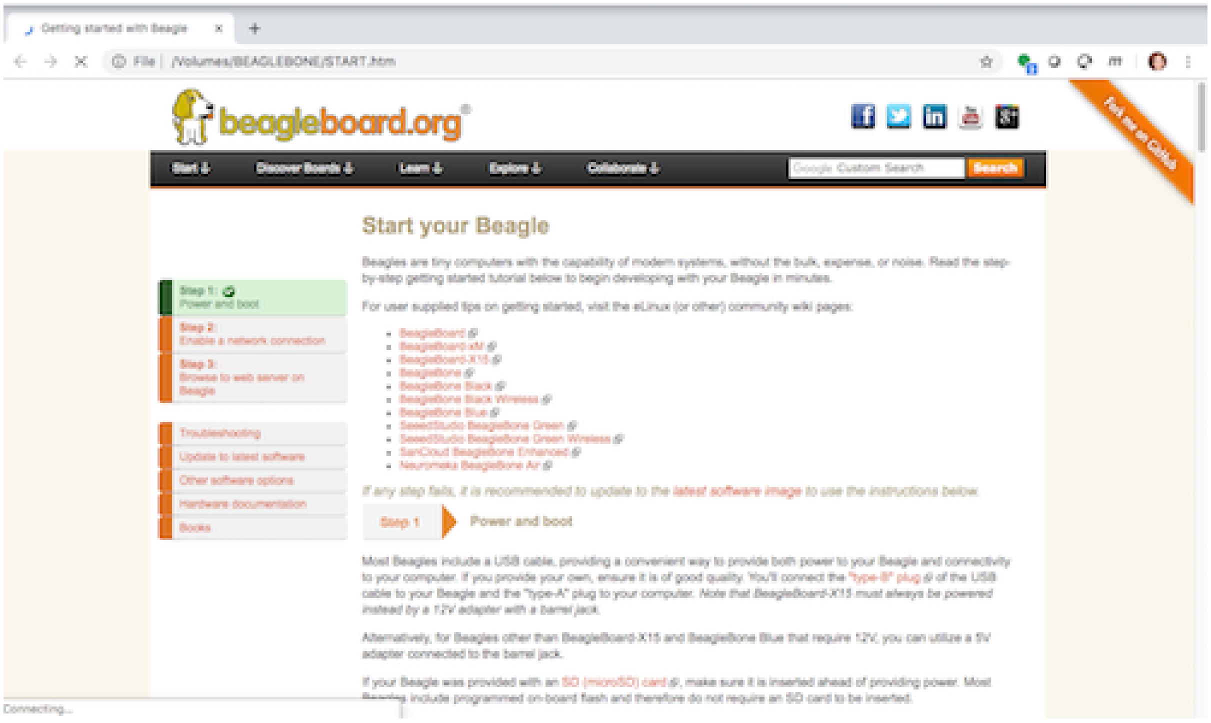


4. BeagleBone® AI will boot.
5. You will notice some of the 5 user LEDs flashing
6. Look for a new mass storage drive to appear on the PC.



```
Name
▶ App
  autorun.inf
▶ Docs
▶ Drivers
  LICENSE.txt
  README.htm
  README.md
▶ scripts
  START.htm
```

7. Open the drive and open START.HTM with your web browser.



Start your Beagle

Beagles are tiny computers with the capability of modern systems, without the bulk, expense, or noise. Read the step-by-step getting started tutorial below to begin developing with your Beagle in minutes.

For user supplied tips on getting started, visit the eLinux (or other) community wiki pages:

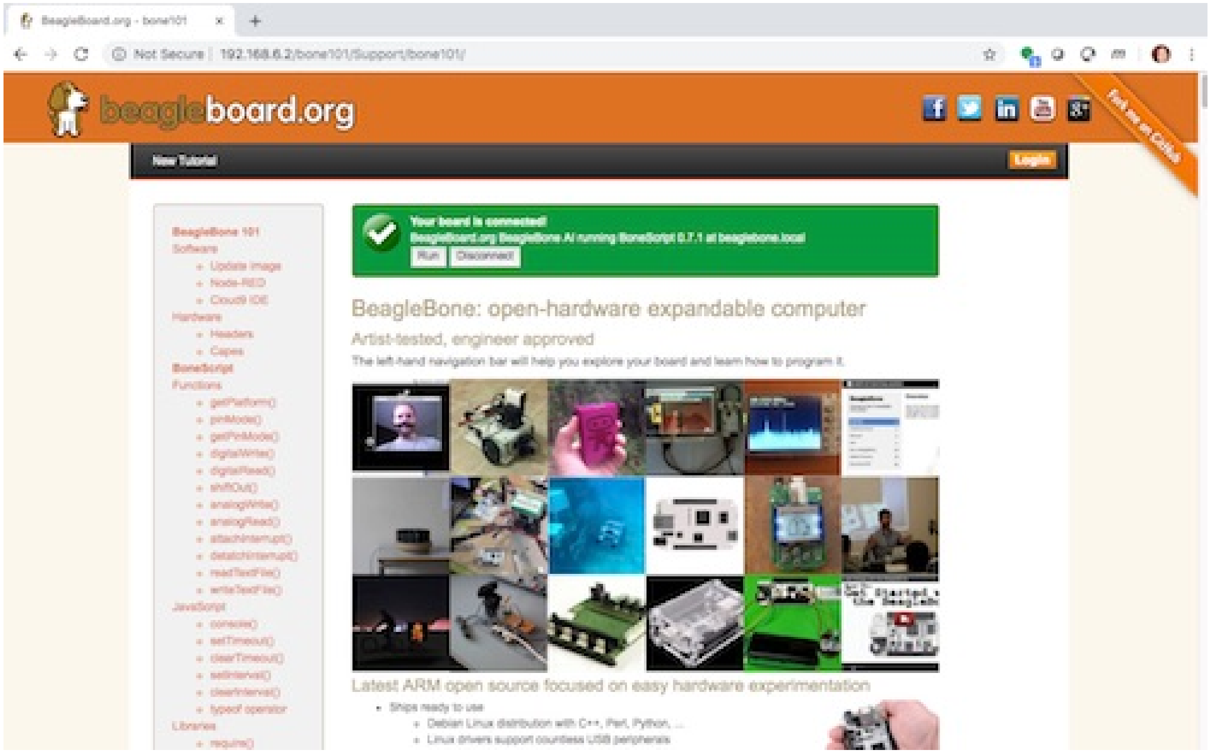
- BeagleBoard
- BeagleBoard-xM
- BeagleBoard-X15
- BeagleBone
- BeagleBone Black
- BeagleBone Black Wireless
- BeagleBone Blue
- SeeedStudio BeagleBone Green
- SeeedStudio BeagleBone Green Wireless
- SanCloud BeagleBone Enhanced
- Neuromeka BeagleBone Air

If any step fails, it is recommended to update to the latest software image to use the instructions below.

Most Beagles include a USB cable, providing a convenient way to provide both power to your Beagle and connectivity to your computer. If you provide your own, ensure it is of good quality. You'll connect the "type-B" plug of the USB cable to your Beagle and the "type-A" plug to your computer. *Note that BeagleBoard-X15 must always be powered instead by a 12V adapter with a barrel jack.*

Alternatively, for Beagles other than BeagleBoard-X15 and BeagleBone Blue that require 12V, you can utilize a 5V adapter connected to the barrel jack.

If your Beagle was provided with an SD (microSD) card, make sure it is inserted ahead of providing power. Most Beagles include programmed on-board flash and therefore do not require an SD card to be inserted.



Your board is connected!
BeagleBoard.org BeagleBone AI running BoneScript 0.7.1 at beaglebone.local

BeagleBone: open-hardware expandable computer

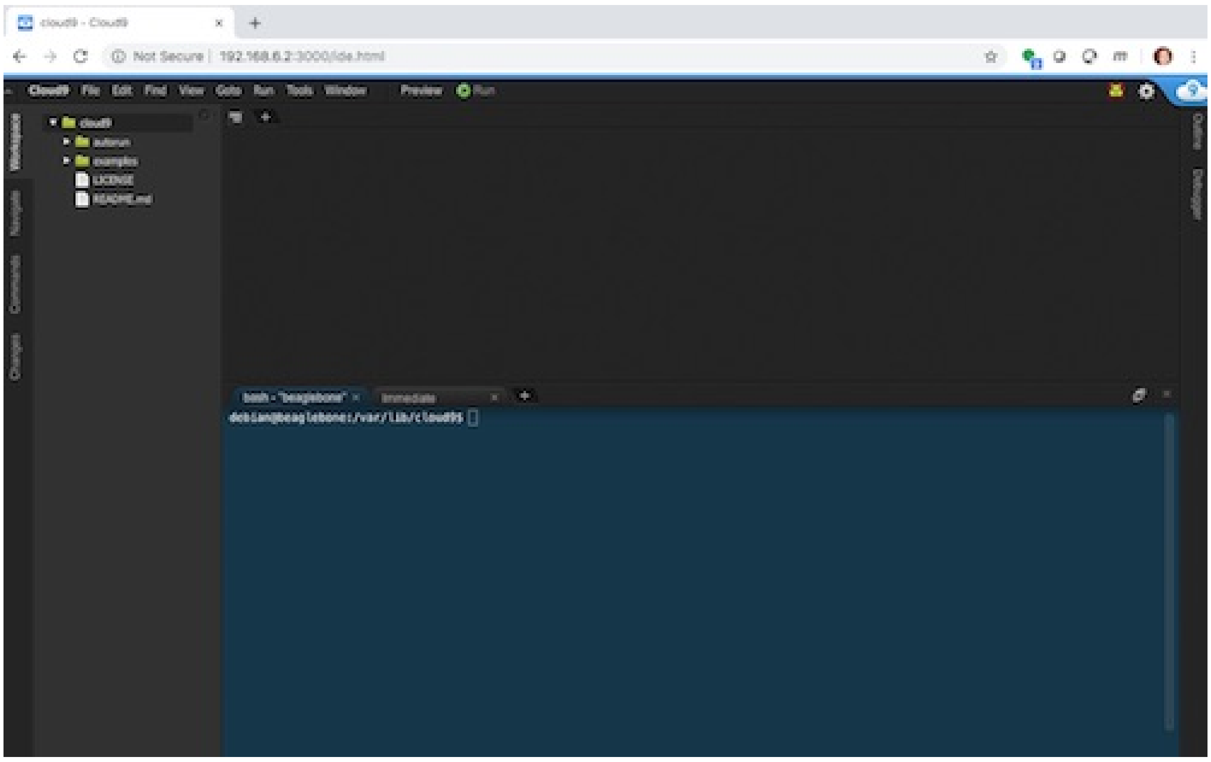
Artist-tested, engineer approved

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

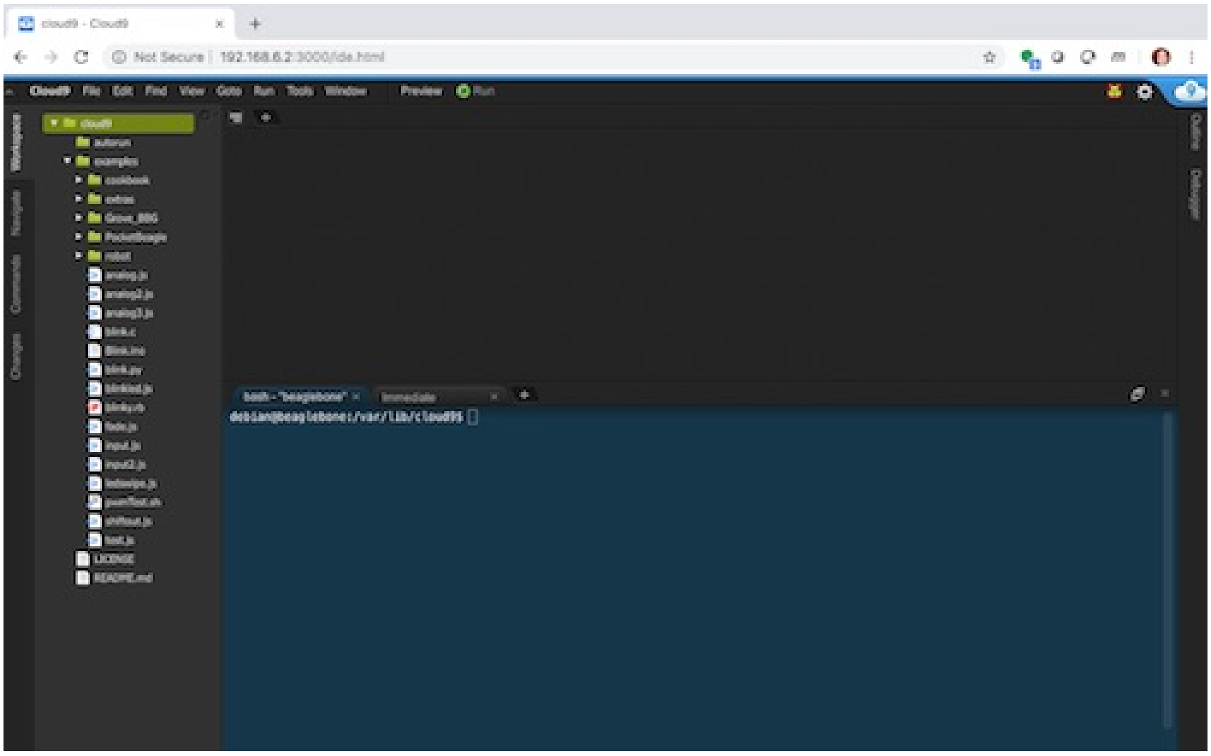
- Ships ready to use
  - Debian Linux distribution with C++, Perl, Python, ...
  - Linux drivers support countless USB peripherals

8. Follow the instructions in the browser window.



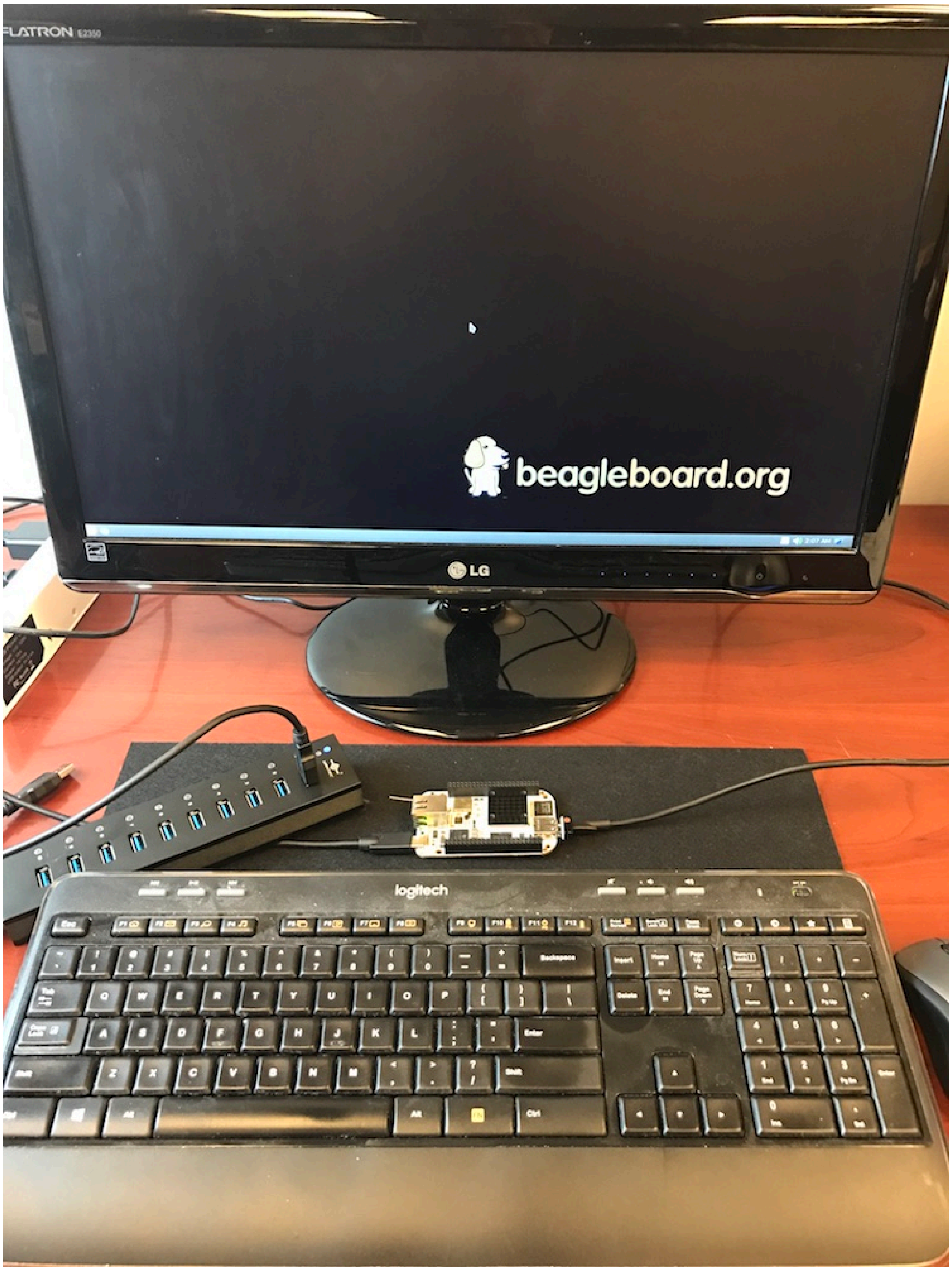
```
debian@beaglebone:/var/lib/cloud9$
```

9. Go to Cloud9 IDE.
10. Open the directories in the left navigation of Cloud9.



```
debian@beaglebone:/var/lib/cloud9$
```

Standalone w/Display and Keyboard/Mouse



Note: This configuration requires loading the latest debian 9 image from https://elinux.org/Beagleboard:Latest-images-testing

Load "am57xx-eMMC-flasher-debian-9.13-lxqt-tidl-armhf-2020-08-25-6gb.img.xz" image on the BeagleBone® AI

Presently, the "Cloud 9" application is broken in debian 10 only for this configuration. We re working on a better solution.

1. Connect a combo keyboard and mouse to BeagleBone® AI's USB host port.
2. Connect a microHDMI-to-HDMI cable to BeagleBone® AI's microHDMI port.
3. Connect the microHDMI-to-HDMI cable to an HDMI monitor.
4. Plug a 5V 3A USB type-C power supply into BeagleBone® AI's USB type-C port.
5. BeagleBone® AI will boot. No need to enter any passwords.
6. Depending on which software image is loaded, either a Desktop or a login shell will appear on the monitor.
7. Follow the instructions at https://beagleboard.org/upgrade

Wireless Connection

NOTE:Start with this image "am57xx-eMMC-flasher-debian-10.3-iot-tidl-armhf-2020-04-06-6gb.img.xz" loaded on your BeagleBone® AI.

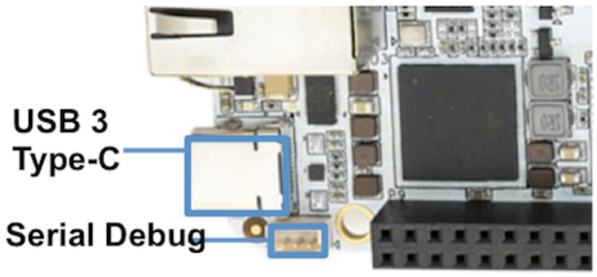
1. Plug a 5V 3A USB type-C power supply into BeagleBone® AI's USB type-C port.
2. BeagleBone® AI will boot.
3. Connect your PC's WiFi to SSID "BeagleBone-XXXX" where XXXX varies for your BeagleBone® AI.
4. Use password "BeagleBone" to complete the WiFi connection.
5. Open http://192.168.8.1 in your web browser.
6. Follow the instructions in the browser window.

Connecting a 3 PIN Serial Debug Cable

A 3 PIN serial debug cable can be helpful to debug when you need to view the boot messages through a terminal program such as putty on your host PC. This cable is not needed for most BeagleBone® AI boot up scenarios.

Cables: https://git.beagleboard.org/beagleboard/beaglebone-ai/-/wikis/Frequently-Asked-Questions#serial-cable

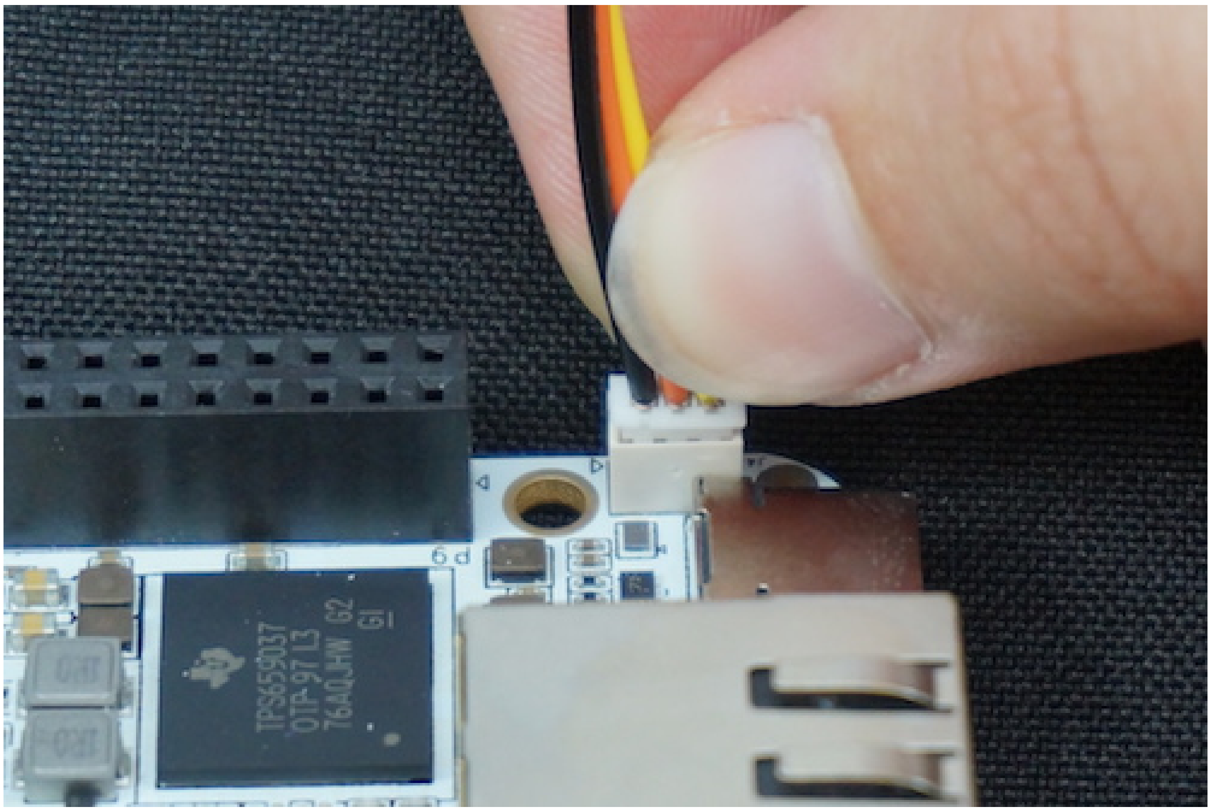
Locate the 3 PIN debug header on BeagleBone® AI, near the USB C connection.



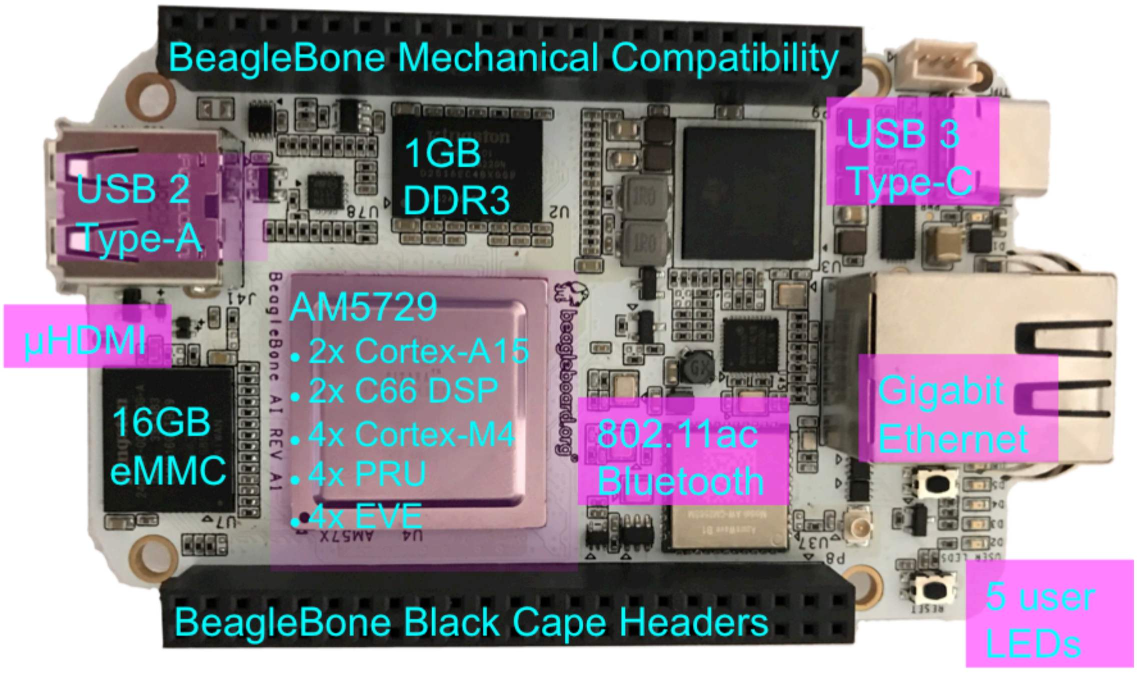
Press the small white connector into the 3 PIN debug header. The pinout is:

- Pin 1 (the pin closest to the screw-hole in the board. It is also marked with a shape on the silkscreen): GND
- Pin 2: UART1_RX (i.e. this is a BB-AI input pin)

- Pin 3: UART1_TX (i.e. BB-AI transmits out on this pin)



2.4.4 BeagleBone AI Overview



BeagleBone® AI Features

Main Processor Features of the AM5729 Within BeagleBone® AI

- Dual 1.5GHz ARM® Cortex®-A15 with out-of-order speculative issue 3-way superscalar execution pipeline for the fastest execution of existing 32-bit code
- 2 C66x Floating-Point VLIW DSP supported by OpenCL
- 4 Embedded Vision Engines (EVEs) supported by TIDL machine learning library
- 2x Dual-Core Programmable Real-Time Unit (PRU) subsystems (4 PRUs total) for ultra low-latency control and software generated peripherals
- 2x Dual ARM® Cortex®-M4 co-processors for real-time control
- IVA-HD subsystem with support for 4K @ 15fps H.264 encode/decode and other codecs @ 1080p60
- Vivante® GC320 2D graphics accelerator
- Dual-Core PowerVR® SGX544™ 3D GPU

Communications

- BeagleBone Black header and mechanical compatibility
- 16-bit LCD interfaces
- 4+ UARTs
- 2 I2C ports
- 2 SPI ports
- Lots of PRU I/O pins

Memory

- 1GB DDR3L
- 16GB on-board eMMC flash

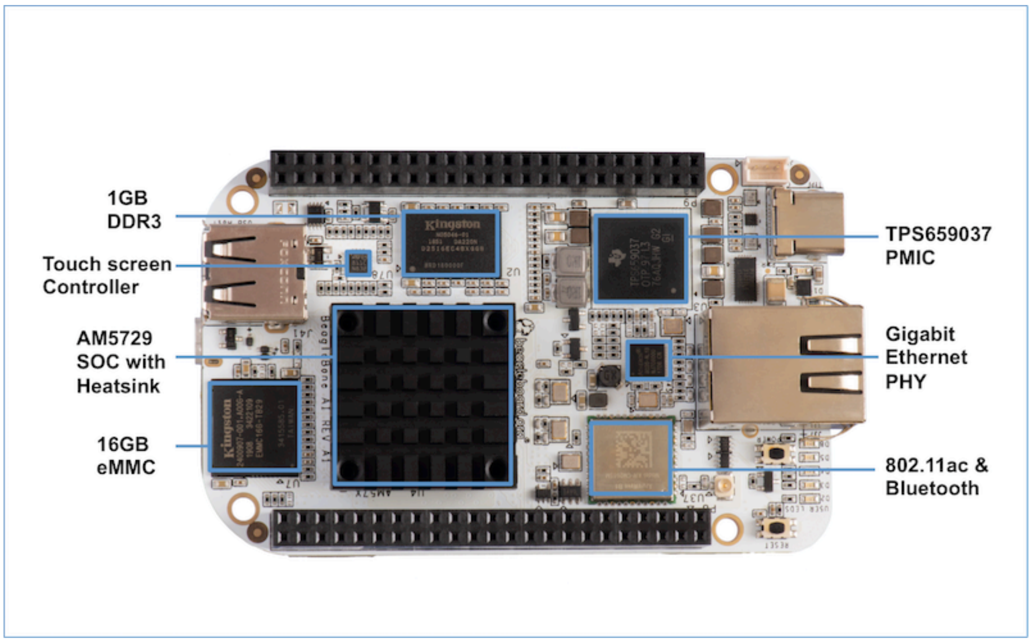
Connectors

- USB Type-C connector for power and SuperSpeed dual-role controller
- Gigabit Ethernet
- 802.11ac 2.4/5GHz WiFi via the AzureWave AW-CM256SM

Out of Box Software

- Zero-download out of box software environment

Board Component Locations

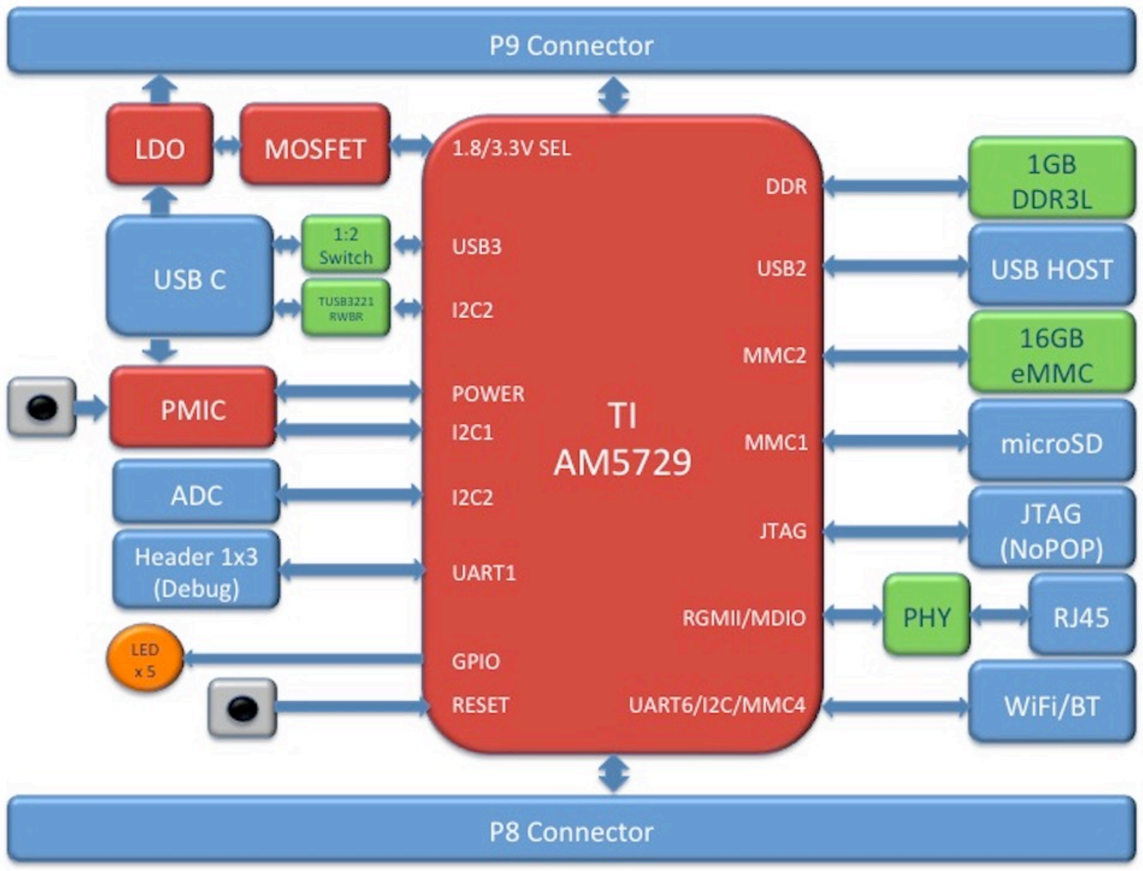


2.4.5 BeagleBone AI High Level Specification

This section provides the high level specification of BeagleBone® AI

Block Diagram

The figure below is the high level block diagram of BeagleBone® AI. For detailed layout information please check the schematics.

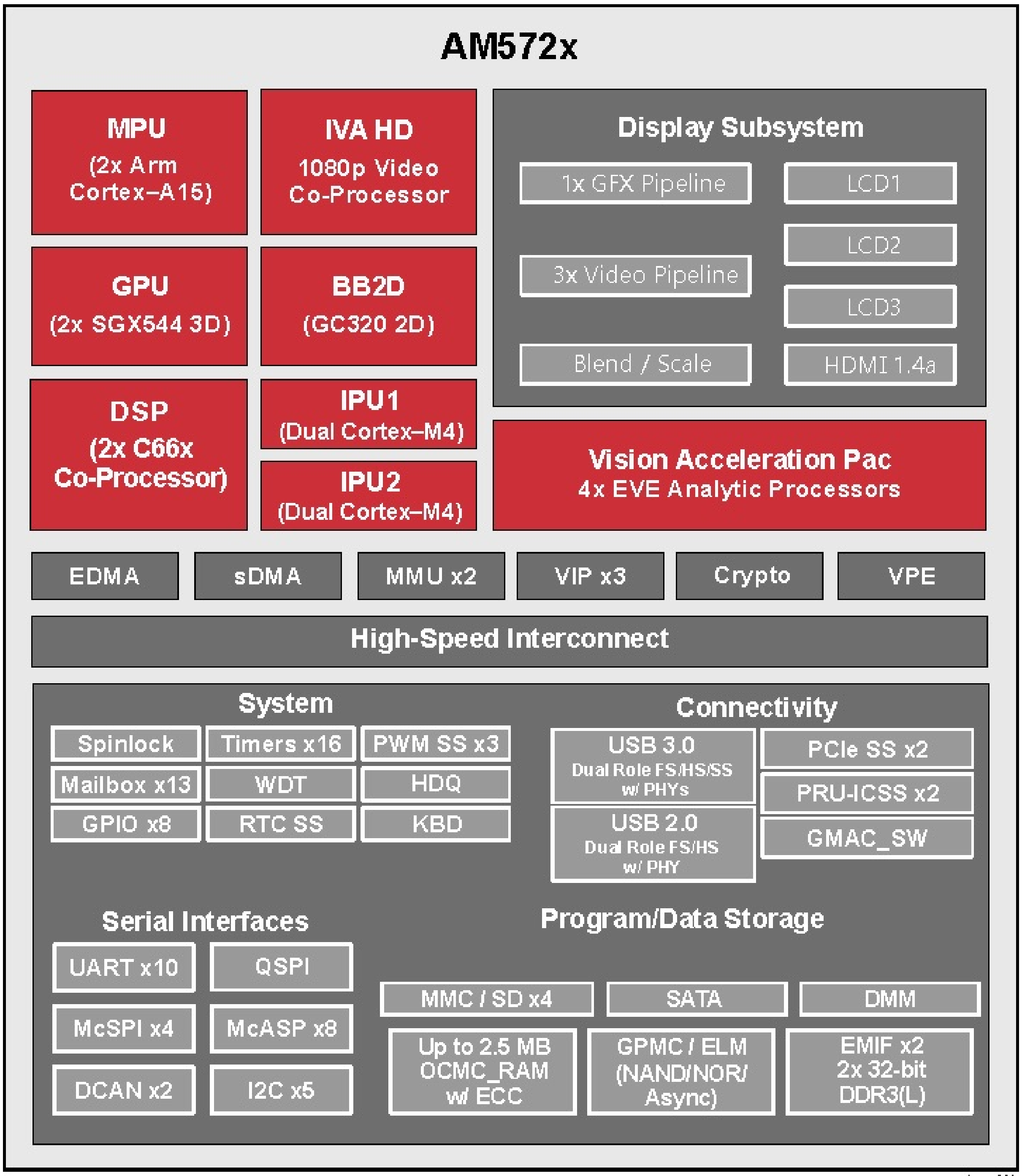


AM572x Sitara™ Processor

The Texas Instruments AM572x Sitara™ processor family of SOC devices brings high processing performance through the maximum flexibility of a fully integrated mixed processor solution. The devices also combine programmable video processing with a highly integrated peripheral set ideal for AI applications. The AM5729 used on BeagleBone® AI is the super-set device of the family.

Programmability is provided by dual-core ARM® Cortex®-A15 RISC CPUs with Arm® Neon™ extension, and two TI C66x VLIW floating-point DSP core, and Vision AccelerationPac (with 4x EVEs). The Arm allows developers to keep control functions separate from other algorithms programmed on the DSPs and coprocessors, thus reducing the complexity of the system software.

Texas Instruments AM572x Sitara™ Processor Family Block Diagram*



MPU Subsystem The Dual Cortex-A15 MPU subsystem integrates the following submodules:

- ARM Cortex-A15 MPCore
  - Two central processing units (CPUs)
  - ARM Version 7 ISA: Standard ARM instruction set plus Thumb®-2, Jazelle® RCT Java™ accelerator, hardware virtualization support, and large physical address extensions (LPAE)
  - Neon™ SIMD coprocessor and VFPv4 per CPU
  - Interrupt controller with up to 160 interrupt requests
  - One general-purpose timer and one watchdog timer per CPU – Debug and trace features
  - 32-KiB instruction and 32-KiB data level 1 (L1) cache per CPU
- Shared 2-MiB level 2 (L2) cache

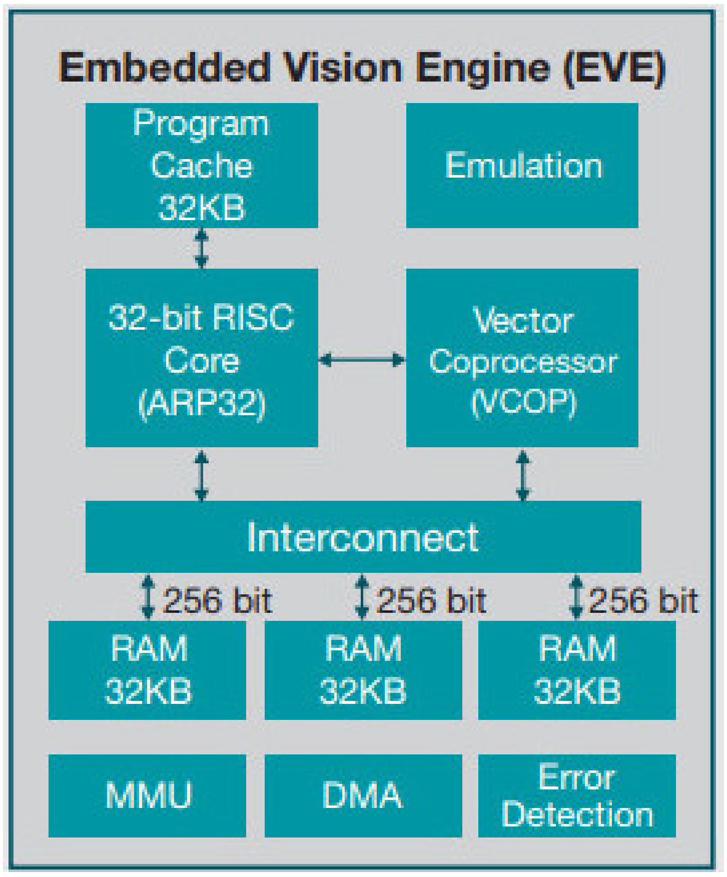
- 48-KiB bootable ROM
- Local power, reset, and clock management (PRCM) module
- Emulation features
- Digital phase-locked loop (DPLL)

DSP Subsystems There are two DSP subsystems in the device. Each DSP subsystem contains the following submodules:

- TMS320C66x™ Floating-Point VLIW DSP core for audio processing, and general-purpose imaging and video processing. It extends the performance of existing C64x+™ and C647x™ DSPs through enhancements and new features.
  - 32-KiB L1D and 32-KiB L1P cache or addressable SRAM
  - 288-KiB L2 cache
- 256-KiB configurable as cache or SRAM
- 32-KiB SRAM
- Enhanced direct memory access (EDMA) engine for video and audio data transfer
- Memory management units (MMU) for address management.
- Interrupt controller (INTC)
- Emulation capabilities
- Supported by OpenCL

EVE Subsystems

- 4 Embedded Vision Engines (EVEs) supported by TIDL machine learning library



The Embedded Vision Engine (EVE) module is a programmable imaging and vision processing engine. Software support for the EVE module is available through OpenCL Custom Device model with fixed set of functions. More information is available http://www.ti.com/lit/wp/spry251/spry251.pdf

PRU-ICSS Subsystems

- 2x Dual-Core Programmable Real-Time Unit (PRU) subsystems (4 PRUs total) for ultra low-latency control and software generated peripherals. Access to these powerful subsystems is available through through the P8 and P9 headers. These are detailed in Section 7.

IPU Subsystems There are two Dual Cortex-M4 IPU subsystems in the device available for general purpose usage, particularly real-time control. Each IPU subsystem includes the following components:

- Two Cortex-M4 CPUs
- ARMv7E-M and Thumb-2 instruction set architectures
- Hardware division and single-cycle multiplication acceleration
- Dedicated INTC with up to 63 physical interrupt events with 16-level priority
- Two-level memory subsystem hierarchy
  - L1 (32-KiB shared cache memory)
  - L2 ROM + RAM
- 64-KiB RAM
- 16-KiB bootable ROM
- MMU for address translation
- Integrated power management
- Emulation feature embedded in the Cortex-M4

IVA-HD Subsystem

- IVA-HD subsystem with support for 4K @ 15fps H.264 encode/decode and other codecs @ 1080p60 The IVA-HD subsystem is a set of video encoder and decoder hardware accelerators. The list of supported codecs can be found in the software development kit (SDK) documentation.

BB2D Graphics Accelerator Subsystem The Vivante® GC320 2D graphics accelerator is the 2D BitBlt (BB2D) graphics accelerator subsystem on the device with the following features:

- API support:
  - OpenWF™, DirectFB
  - GDI/DirectDraw
- BB2D architecture:
  - BitBlt and StretchBlt
  - DirectFB hardware acceleration
  - ROP2, ROP3, ROP4 full alpha blending and transparency
  - Clipping rectangle support
  - Alpha blending includes Java 2 Porter-Duff compositing rules
  - 90-, 180-, 270-degree rotation on every primitive
  - YUV-to-RGB color space conversion
  - Programmable display format conversion with 14 source and 7 destination formats
  - High-quality, 9-tap, 32-phase filter for image and video scaling at 1080p
  - Monochrome expansion for text rendering
  - 32K × 32K coordinate system

Dual-Core PowerVR® SGX544™ 3D GPU The 3D graphics processing unit (GPU) subsystem is based on POWERVR® SGX544 subsystem from Imagination Technologies. It supports general embedded applications. The GPU can process different data types simultaneously, such as: pixel data, vertex data, video data, and general-purpose data. The GPU subsystem has the following features:

- Multicore GPU architecture: two SGX544 cores.
- Shared system level cache of 128 KiB
- Tile-based deferred rendering architecture
- Second-generation universal scalable shader engines (USSE2), multithreaded engines incorporating pixel and vertex shader functionality
- Present and texture load accelerators
  - Enables to move, rotate, twiddle, and scale texture surfaces.
  - Supports RGB, ARGB, YUV422, and YUV420 surface formats.
  - Supports bilinear upscale.
  - Supports source colorkey.
- Fine-grained task switching, load balancing, and power management
- Programmable high-quality image antialiasing
- Bilinear, trilinear, anisotropic texture filtering
- Advanced geometry DMA driven operation for minimum CPU interaction
- Fully virtualized memory addressing for OS operation in a unified memory architecture (MMU)

Memory

1GB DDR3L Dual 256M x 16 DDR3L memory devices are used, one on each side of the board, for a total of 1 GB. They will each operate at a clock frequency of up to 533 MHz yielding an effective rate of 1066Mb/s on the DDR3L bus allowing for 4GB/s of DDR3L memory bandwidth.

16GB Embedded MMC A single 16GB embedded MMC (eMMC) device is on the board.

microSD Connector The board is equipped with a single microSD connector to act as a secondary boot source for the board and, if selected as such, can be the primary booth source. The connector will support larger capacity microSD cards. The microSD card is not provided with the board.

Boot Modes

Power Management

Connectivity

BeagleBone® AI supports the majority of the functions of the AM5729 SOC through connectors or expansion header pin accessibility. See section 7 for more information on expansion header pinouts. There are a few functions that are not accessible which are: (TBD)

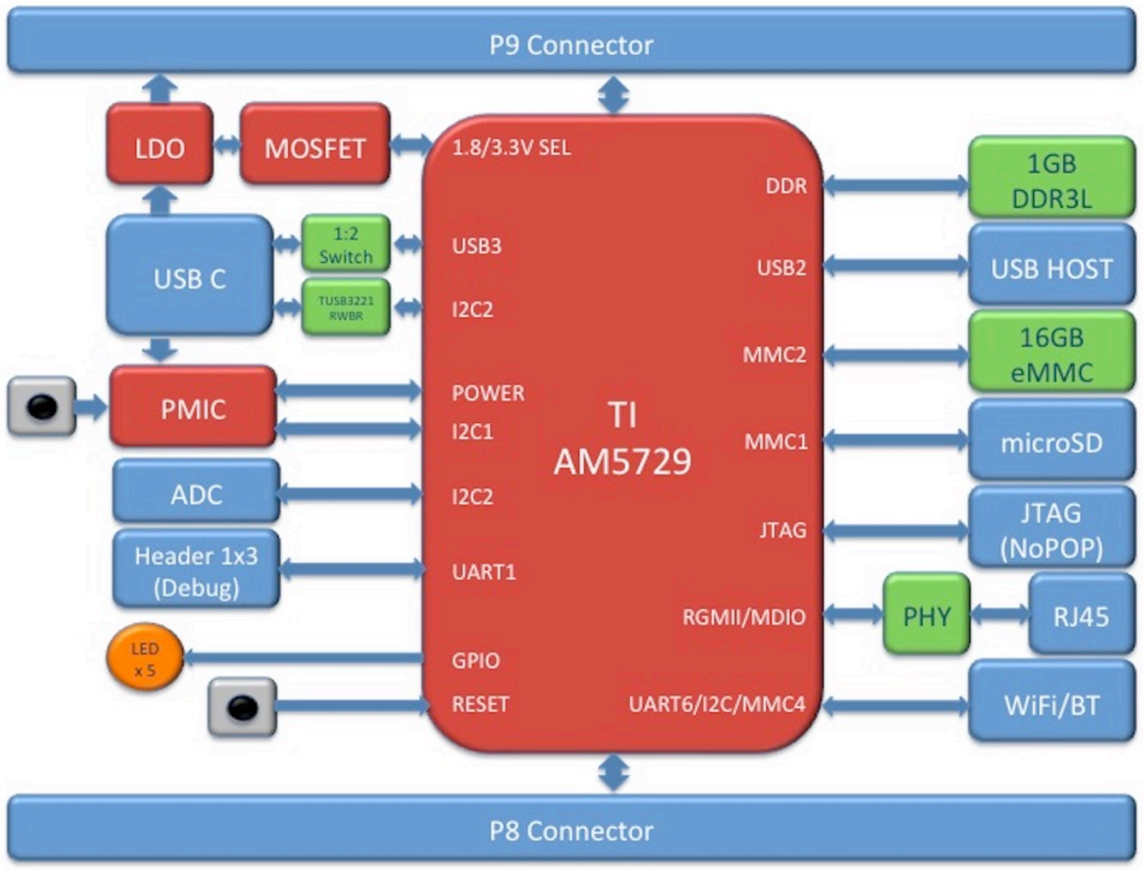
Table 2.22: On-board I2C Devices

Address	Identifier	Description
0x12	U3	TPS6590379 PMIC DVS
0x41	U78	STMPE811Q ADC and GPIO expander
0x47	U13	HD3SS3220 USB Type-C DRP port controller
0x50	U9	24LC32 board ID EEPROM
0x58	U3	TPS6590379 PMIC power registers
0x5a	U3	TPS6590379 PMIC interfaces and auxiliaries
0x5c	U3	TPS6590379 PMIC trimming and test
0x5e	U3	TPS6590379 PMIC OTP

2.4.6 Detailed Hardware Design

This section provides a detailed description of the Hardware design. This can be useful for interfacing, writing drivers, or using it to help modify specifics of your own design.

The figure below is the high level block diagram of BeagleBone® AI. For those who may be concerned, this is the same figure found in section 5. It is placed here again for convenience so it is closer to the topics to follow.



Power Section

Figure ? is the high level block diagram of the power section of the board.

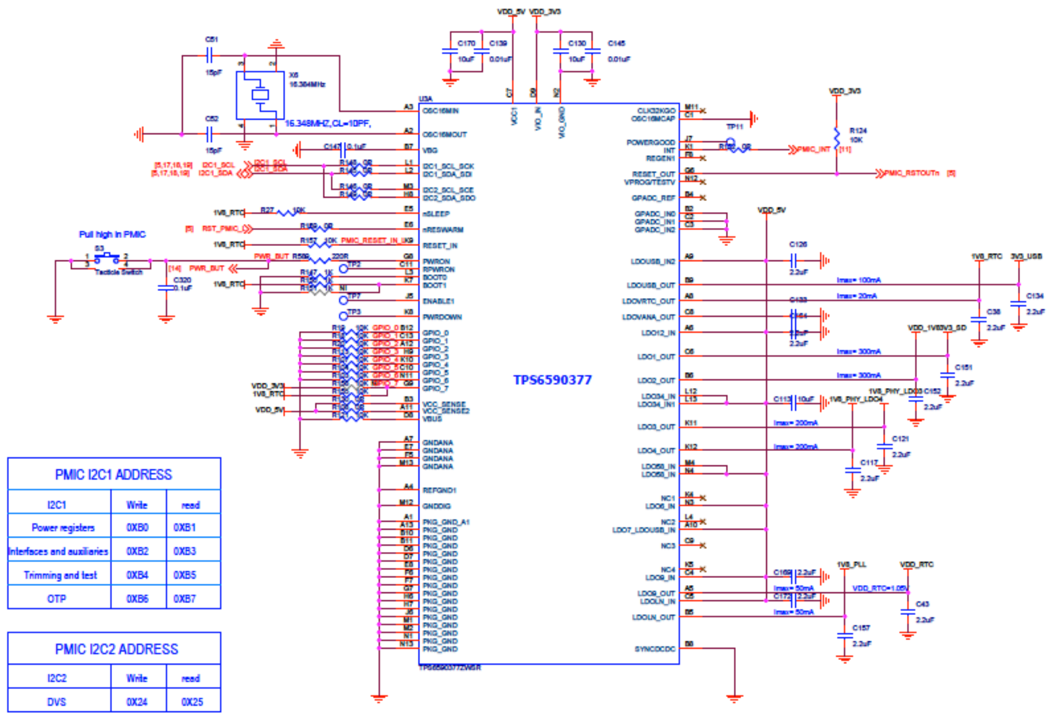
(Block Diagram for Power)

TPS6590379 PMIC The Texas Instruments TPS6590379ZWSR device is an integrated power-management IC (PMIC) specifically designed to work well ARM Cortex A15 Processors, such as the AM5729 used on BeagleBone® AI. The datasheet is located here https://www.ti.com/lit/ds/symlink/tps659037.pdf

The device provides seven configurable step-down converters with up to 6 A of output current for memory, processor core, input-output (I/O), or preregulation of LDOs. One of these configurable step-down converters can be combined with another 3-A regulator to allow up to 9 A of output current. All of the step-down converters can synchronize to an external clock source between 1.7 MHz and 2.7 MHz, or an internal fallback clock at 2.2 MHz.

The TPS659037 device contains seven LDO regulators for external use. These LDO regulators can be supplied from either a system supply or a preregulated supply. The power-up and power-down controller is configurable and supports any power-up and power-down sequences (OTP based). The TPS659037 device includes a 32-kHz RC oscillator to sequence all resources during power up and power down. In cases where a fast start up is needed, a 16-MHz crystal oscillator is also included to quickly generate a stable 32-kHz for the system. All LDOs and SMPS converters can be controlled by the SPI or I2C interface, or by power request signals. In addition, voltage scaling registers allow transitioning the SMPS to different voltages by SPI, I2C, or roof and floor control.

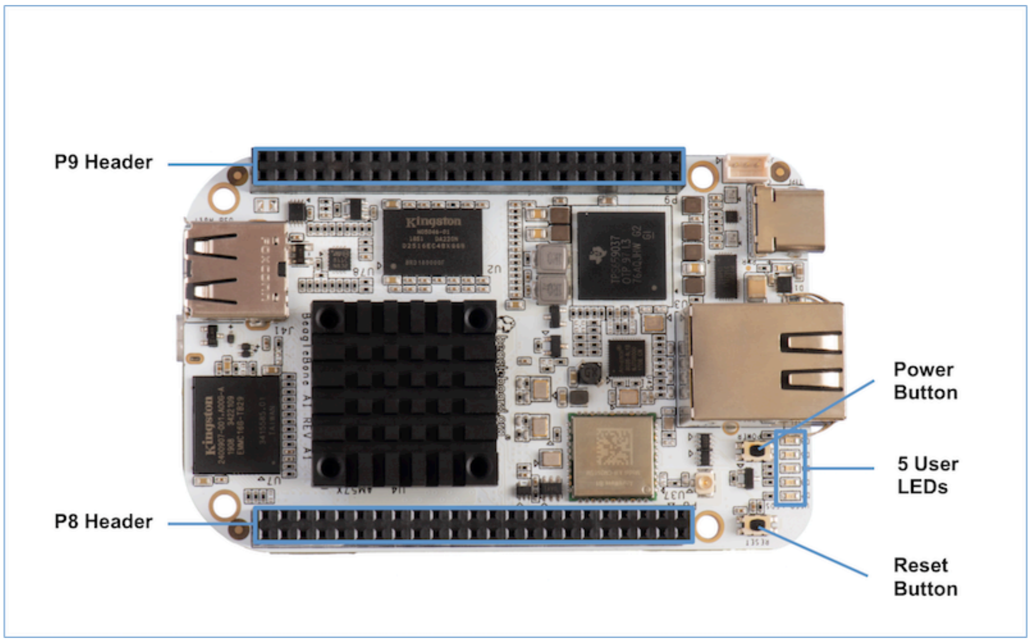
One dedicated pin in each package can be configured as part of the power-up sequence to control external resources. General-purpose input-output (GPIO) functionality is available and two GPIOs can be configured as part of the power-up sequence to control external resources. Power request signals enable power mode control for power optimization. The device includes a general-purpose sigma-delta analog-to-digital converter (GPADC) with three external input channels.



PMIC I2C1 ADDRESS		
I2C1	Write	read
Power registers	0XB0	0XB1
Interfaces and auxiliaries	0XB2	0XB3
Trimming and test	0XB4	0XB5
OTP	0XB6	0XB7

PMIC I2C2 ADDRESS		
I2C2	Write	read
DVS	0X24	0X25

USB-C Power Below image shows how the USB-C power input is connected to the **TPS6590379**.



Power Button

eMMC Flash Memory (16GB)

eMMC Device

eMMC Circuit Design

Board ID A board identifier is placed on the eMMC in the second linear boot partition (/dev/mmcblk1boot1). Reserved bytes up to 32k (0x8000) are filled with "FF".

Table 2.23: Board ID

Name	Size (bytes)	Contents
Header	4	MSB 0xEE3355AA LSB (stored LSB first)
Board Name	8	Name for board in ASCII "BBONE-AI" = BeagleBone AI
Version	4	Hardware version code for board in ASCII "00A1" = rev. A1
Serial Number	14	Serial number of the board. This is a 14 character string which is: WWYYEMAInnnnnn where: • WW = 2 digit week of the year of production • YY = 2 digit year of production • EM = Embest • AI = BeagleBone AI • nnnnnn = incrementing board number

```
debian@beaglebone:/var/lib/cloud9$ sudo hexdump -C /dev/mmcblk1boot1
00000000  aa 55 33 ee 42 42 4f 4e  45 2d 41 49 30 30 41 31  |.U3.BBONE-
↪AI00A1|
00000010  31 39 33 33 45 4d 41 49  30 30 30 38 30 33 ff ff  |1933EMAI000803..
↪|
00000020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................
↪|
```

(continues on next page)

(continued from previous page)

```
*
00008000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................
↪|
*
00400000
```

Wireless Communication: 802.11 ac & Bluetooth: AzureWave AW-CM256SM

Datasheet https://storage.googleapis.com/wzukusers/user-26561200/documents/5b7d0fe3c3f29Ct6k0QI/AW-CM256SM_DS_Rev%2015_CYW.pdf Wireless connectivity is provided on BeagleBone® AI via the AzureWave Technologies AW-CM256SM IEEE 802.11a/b/g/n/ac Wi-Fi with Bluetooth 4.2 Combo Stamp Module.

This highly integrated wireless local area network (WLAN) solution combines Bluetooth 4.2 and provides a complete 2.4GHz Bluetooth system which is fully compliant to Bluetooth 4.2 and v2.1 that supports EDR of 2Mbps and 3Mbps for data and audio communications. It enables a high performance, cost effective, low power, compact solution that easily fits onto the SDIO and UART combo stamp module.

Compliant with the IEEE 802.11a/b/g/n/ac standard, AW-CM256SM uses Direct Sequence Spread Spectrum (DSSS), Orthogonal Frequency Division Multiplexing (OFDM), BPSK, QPSK, CCK and QAM baseband modulation technologies. Compare to 802.11n technology, 802.11ac provides a big improvement on speed and range.

The AW-CM256SM module adopts a Cypress solution. The module design is based on the Cypress CYP43455 single chip.

WLAN on the AzureWave AW-CM256SM High speed wireless connection up to 433.3Mbps transmit/receive PHY rate using 80MHz bandwidth,

- 1 antennas to support 1(Transmit) and 1(Receive) technology and Bluetooth
- WCS (Wireless Coexistence System)
- Low power consumption and high performance
- Enhanced wireless security
- Fully speed operation with Piconet and Scatternet support
- 12mm(L) x 12mm(W) x1.65mm(H) LGA package
- Dual - band 2.4 GHz and 5GHz 802.11 a/b/g/n/ac
- External Crystal

Bluetooth on the AzureWave AW-CM256S

- 1 antennas to support 1(Transmit) and 1(Receive) technology and Bluetooth
- Fully qualified Bluetooth BT4.2
- Enhanced Data Rate(EDR) compliant for both 2Mbps and 3Mbps supported
- High speed UART and PCM for Bluetooth

HDMI

The HDMI interface is aligned with the HDMI TMDS single stream standard v1.4a (720p @60Hz to 1080p @24Hz) and the HDMI v1.3 (1080p @60Hz): 3 data channels, plus 1 clock channel is supported (differential).

TODO: Verify it isn't better than this. Doesn't seem right.

PRU-ICSS

The Texas Instruments AM5729 Sitara™ provides 2 Programmable Real-Time Unit Subsystem and Industrial Communciation Subsystems. (PRU-ICSS1 and PRU-ICSS2).

Within each PRU-ICSS are dual 32-bit Load / Store RISC CPU cores: Programmable Real-Time Units (PRU0 and PRU1), shared data and instruction memories, internal peripheral modules and an interrupt controller. Therefore the SoC is providing a total of 4 PRU 32-bit RISC CPU's:

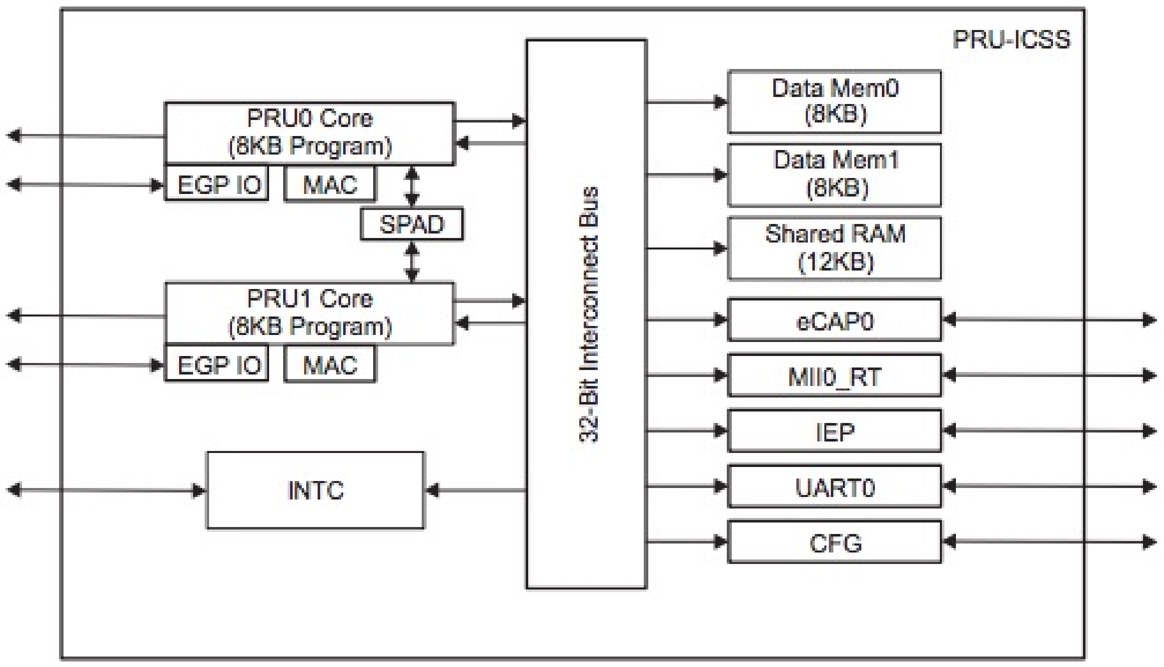
- PRU-ICSS1 PRU0
- PRU-ICSS1 PRU1
- PRU-ICSS2 PRU0
- PRU-ICSS2 PRU1

The programmable nature of the PRUs, along with their access to pins, events and all SoC resources, provides flexibility in implementing fast real-time responses, specialized data handling operations, peripheral interfaces and in off-loading tasks from the other processor cores of the SoC.

PRU-ICSS Features Each of the 2 PRU-ICSS (PRU-ICSS1 and PRU-ICSS2) includes the following main features:

- 2 Independent programmable real-time (PRU) cores (PRU0 and PRU1)
- 21x Enhanced GPIs (EGPIs) and 21x Enhanced GPOs (EGPOs) with asynchronous capture and serial support per each PRU CPU core
- One Ethernet MII_RT module (PRU-ICSS_MII_RT) with two MII ports and configurable connections to PRUs
- 1 MDIO Port (PRU-ICSS_MII_MDIO)
- One Industrial Ethernet Peripheral (IEP) to manage/generate Industrial Ethernet functions
- 1 x 16550-compatible UART with a dedicated 192 MHz clock to support 12Mbps Profibus
- 1 Industrial Ethernet timer with 7/9 capture and 8 compare events
- 1 Enhanced Capture Module (ECAP)
- 1 Interrupt Controller (PRU-ICSS_INTC)
- A flexible power management support
- Integrated switched central resource with programmable priority
- Parity control supported by all memories

PRU-ICSS Block Diagram Below is a high level block diagram of one of the PRU-ICSS Subsystems



PRU-ICSS Resources and FAQ's

Resources

- Great resources for PRU and BeagleBone® has been compiled here https://beagleboard.org/pru
- The PRU Cookbook provides examples and getting started information *PRU Cookbook*
- Detailed specification is available at http://processors.wiki.ti.com/index.php/PRU-ICSS

FAQ

- Q: Is it possible to configure the Ethernet MII to be accessed via a PRU MII?
- A: TBD

PRU-ICSS1 Pin Access The table below shows which PRU-ICSS1 signals can be accessed on BeagleBone® AI and on which connector and pins they are accessible from. Some signals are accessible on the same pins. Signal Names reveal which PRU-ICSS Subsystem is being addressed. pr1 is PRU-ICSS1 and pr2 is PRU-ICSS2

Table 2.24: PRU-ICSS1 Pin Access

SIGNAL NAME	DESCRIPTION	T Y P E	P R O C	HE ADER _PIN	M O D E	HE ADER _PIN	M O D E
pr1_pru0_gpo0	PRU0 G eneral-Purpose Output	O	A H 6	NA			
pr1_pru0_gpo1	PRU0 G eneral-Purpose Output	O	A H 3	NA			
pr1_pru0_gpo2	PRU0 G eneral-Purpose Output	O	A H 5	NA			
pr1_pru0_gpo3	PRU0 G eneral-Purpose Output	O	A G 6	P 8_12	M O D E 1 3		
pr1_pru0_gpo4	PRU0 G eneral-Purpose Output	O	A H 4	P 8_11	M O D E 1 3		
pr1_pru0_gpo5	PRU0 G eneral-Purpose Output	O	A G 4	P 9_15	M O D E 1 3		
pr1_pru0_gpo6	PRU0 G eneral-Purpose Output	O	A G 2	NA			
pr1_pru0_gpo7	PRU0 G eneral-Purpose Output	O	A G 3	NA			
pr1_pru0_gpo8	PRU0 G eneral-Purpose Output	O	A G 5	NA			
pr1_pru0_gpo9	PRU0 G eneral-Purpose Output	O	A F 2	NA			
pr1_pru0_gpo10	PRU0 G eneral-Purpose Output	O	A F 6	NA			
pr1_pru0_gpo11	PRU0 G eneral-Purpose Output	O	A F 3	NA			
pr1_pru0_gpo12	PRU0 G eneral-Purpose Output	O	A F 4	NA			
pr1_pru0_gpo13	PRU0 G eneral-Purpose Output	O	A F 1	NA			
pr1_pru0_gpo14	PRU0 G eneral-Purpose Output	O	A E 3	NA			
pr1_pru0_gpo15	PRU0 G eneral-Purpose Output	O	A E 5	NA			
pr1_pru0_gpo16	PRU0 G eneral-Purpose Output	O	A E 1	NA			
pr1_pru0_gpo17	PRU0 G eneral-Purpose Output	O	A E 2	P 9_26	M O D E 1 3		
pr1_pru0_gpo18	PRU0 G eneral-Purpose Output	O	A E 6	NA			
pr1_pru0_gpo19	PRU0 G eneral-Purpose Output	O	A D 2	NA			
pr1_pru0_gpo20	PRU0 G eneral-Purpose Output	O	A D 3	NA			
pr1_pru0_gpi0	PRU0 G eneral-Purpose Input	I	A H 6	NA			
pr1_pru0_gpi1	PRU0 G eneral-Purpose Input	I	A H 3	NA			
pr1_pru0_gpi2	PRU0 G eneral-Purpose Input	I	A H 5	NA			
pr1_pru0_gpi3	PRU0 G eneral-Purpose Input	I	A G 6	P 8_12	M O D E 1 2		
pr1_pru0_gpi4	PRU0 G eneral-Purpose Input	I	A H 4	P 8_11	M O D E 1 2		
pr1_pru0_gpi5	PRU0 G eneral-Purpose Input	I	A G 4	P 9_15	M O D E 1 2		
pr1_pru0_gpi6	PRU0 G eneral-Purpose Input	I	A G 2	NA			
pr1_pru0_gpi7	PRU0 G eneral-Purpose Input	I	A G 3	NA			
pr1_pru0_gpi8	PRU0 G eneral-Purpose Input	I	A G 5	NA			
pr1_pru0_gpi9	PRU0 G eneral-Purpose Input	I	A F 2	NA			
pr1_pru0_gpi10	PRU0 G eneral-Purpose Input	I	A F 6	NA			
pr1_pru0_gpi11	PRU0 G eneral-Purpose Input	I	A F 3	NA			
pr1_pru0_gpi12	PRU0 G eneral-Purpose Input	I	A F 4	NA			
pr1_pru0_gpi13	PRU0 G eneral-Purpose Input	I	A F 1	NA			

continues on next page

Table 2.24 – continued from previous page

SIGNAL NAME	DESCRIPTION	T Y P E	P R O C	HE ADER _PIN	M O D E	HE ADER _PIN	M O D E
pr1_pru0_gpi14	PRU0 G eneral-Purpose Input	I	A E 3	NA			
pr1_pru0_gpi15	PRU0 G eneral-Purpose Input	I	A E 5	NA			
pr1_pru0_gpi16	PRU0 G eneral-Purpose Input	I	A E 1	NA			
pr1_pru0_gpi17	PRU0 G eneral-Purpose Input	I	A E 2	P 9_26	M O D E 1 2		
pr1_pru0_gpi18	PRU0 G eneral-Purpose Input	I	A E 6	NA			
pr1_pru0_gpi19	PRU0 G eneral-Purpose Input	I	A D 2	NA			
pr1_pru0_gpi20	PRU0 G eneral-Purpose Input	I	A D 3	NA			
pr1_pru1_gpo0	PRU1 G eneral-Purpose Output	O	E 2	NA			
pr1_pru1_gpo1	PRU1 G eneral-Purpose Output	O	D 2	P 9_20	M O D E 1 3		
pr1_pru1_gpo2	PRU1 G eneral-Purpose Output	O	F 4	P 9_19	M O D E 1 3		
pr1_pru1_gpo3	PRU1 G eneral-Purpose Output	O	C 1	P 9_41	M O D E 1 3		
pr1_pru1_gpo4	PRU1 G eneral-Purpose Output	O	E 4	NA			
pr1_pru1_gpo5	PRU1 G eneral-Purpose Output	O	F 5	P 8_18	M O D E 1 3		
pr1_pru1_gpo6	PRU1 G eneral-Purpose Output	O	E 6	P 8_19	M O D E 1 3		
pr1_pru1_gpo7	PRU1 G eneral-Purpose Output	O	D 3	P 8_13	M O D E 1 3		
pr1_pru1_gpo8	PRU1 G eneral-Purpose Output	O	F 6	NA			
pr1_pru1_gpo9	PRU1 G eneral-Purpose Output	O	D 5	P 8_14	M O D E 1 3		
pr1_pru1_gpo10	PRU1 G eneral-Purpose Output	O	C 2	P 9_42	M O D E 1 3		
pr1_pru1_gpo11	PRU1 G eneral-Purpose Output	O	C 3	P 9_27	M O D E 1 3		
pr1_pru1_gpo12	PRU1 G eneral-Purpose Output	O	C 4	NA			
pr1_pru1_gpo13	PRU1 G eneral-Purpose Output	O	B 2	NA			
pr1_pru1_gpo14	PRU1 G eneral-Purpose Output	O	D 6	P 9_14	M O D E 1 3		
pr1_pru1_gpo15	PRU1 G eneral-Purpose Output	O	C 5	P 9_16	M O D E 1 3		
pr1_pru1_gpo16	PRU1 G eneral-Purpose Output	O	A 3	P 8_15	M O D E 1 3		
pr1_pru1_gpo17	PRU1 G eneral-Purpose Output	O	B 3	P 8_26	M O D E 1 3		
pr1_pru1_gpo18	PRU1 G eneral-Purpose Output	O	B 4	P 8_16	M O D E 1 3		
pr1_pru1_gpo19	PRU1 G eneral-Purpose Output	O	B 5	NA			
pr1_pru1_gpo20	PRU1 G eneral-Purpose Output	O	A 4	NA			
pr1_pru1_gpi0	PRU1 G eneral-Purpose Input	I	E 2	NA			
pr1_pru1_gpi1	PRU1 G eneral-Purpose Input	I	D 2	P 9_20	M O D E 1 2		
pr1_pru1_gpi2	PRU1 G eneral-Purpose Input	I	F 4	P 9_19	M O D E 1 2		
pr1_pru1_gpi3	PRU1 G eneral-Purpose Input	I	C 1	P 9_41	M O D E 1 2		
pr1_pru1_gpi4	PRU1 G eneral-Purpose Input	I	E 4	NA			
pr1_pru1_gpi5	PRU1 G eneral-Purpose Input	I	F 5	P 8_18	M O D E 1 2		
pr1_pru1_gpi6	PRU1 G eneral-Purpose Input	I	E 6	P 8_19	M O D E 1 2		
pr1_pru1_gpi7	PRU1 G eneral-Purpose Input	I	D 3	P 8_13	M O D E 1 2		
pr1_pru1_gpi8	PRU1 G eneral-Purpose Input	I	F 6	NA			

continues on next page

Table 2.24 – continued from previous page

SIGNAL NAME	DESCRIPTION	T Y P E	P R O C	HE ADER _PIN	M O D E	HE ADER _PIN	M O D E
pr1_pru1_gpi9	PRU1 G eneral-Purpose Input	I	D 5	P 8_14	M O D E 1 2		
pr1_pru1_gpi10	PRU1 G eneral-Purpose Input	I	C 2	P 9_42	M O D E 1 2		
pr1_pru1_gpi11	PRU1 G eneral-Purpose Input	I	C 3	P 9_27	M O D E 1 2		
pr1_pru1_gpi12	PRU1 G eneral-Purpose Input	I	C 4	NA			
pr1_pru1_gpi13	PRU1 G eneral-Purpose Input	I	B 2	NA			
pr1_pru1_gpi14	PRU1 G eneral-Purpose Input	I	D 6	P 9_14	M O D E 1 2		
pr1_pru1_gpi15	PRU1 G eneral-Purpose Input	I	C 5	P 9_16	M O D E 1 2		
pr1_pru1_gpi16	PRU1 G eneral-Purpose Input	I	A 3	P 8_15	M O D E 1 2		
pr1_pru1_gpi17	PRU1 G eneral-Purpose Input	I	B 3	P 8_26	M O D E 1 2		
pr1_pru1_gpi18	PRU1 G eneral-Purpose Input	I	B 4	P 8_16	M O D E 1 2		
pr1_pru1_gpi19	PRU1 G eneral-Purpose Input	I	B 5	NA			
pr1_pru1_gpi20	PRU1 G eneral-Purpose Input	I	A 4	NA			
pr1_mii_mt0_clk	MII0 Transmit Clock	I	U 5	NA			
pr1_mii0_txen	MII0 Transmit Enable	O	V 3	NA			
pr1_mii0_txd3	MII0 Transmit Data	O	V 5	NA			
pr1_mii0_txd2	MII0 Transmit Data	O	V 4	NA			
pr1_mii0_txd1	MII0 Transmit Data	O	Y 2	NA			
pr1_mii0_txd0	MII0 Transmit Data	O	W 2	NA			
pr1_mii0_rxdv	MII0 Data Valid	I	V 2	NA			
pr1_mii_mr0_clk	MII0 Receive Clock	I	Y 1	NA			
pr1_mii0_rxd3	MII0 Receive Data	I	W 9	NA			
pr1_mii0_rxd2	MII0 Receive Data	I	V 9	NA			
pr1_mii0_crs	MII0 Carrier Sense	I	V 7	NA			
pr1_mii0_rxer	MII0 Receive Error	I	U 7	NA			
pr1_mii0_rxd1	MII0 Receive Data	I	V 6	NA			
pr1_mii0_rxd0	MII0 Receive Data	I	U 6	NA			
pr1_mii0_col	MII0 Collision Detect	I	V 1	NA			
pr1_mii0_rxlink	MII0 Receive Link	I	U 4	NA			
pr1_mii_mt1_clk	MII1 Transmit Clock	I	C 1	P 9_41	M O D E 1 1		
pr1_mii1_txen	MII1 Transmit Enable	O	E 4	NA			
pr1_mii1_txd3	MII1 Transmit Data	O	F 5	P 8_18	M O D E 1 1		
pr1_mii1_txd2	MII1 Transmit Data	O	E 6	P 8_19	M O D E 1 1		
pr1_mii1_txd1	MII1 Transmit Data	O	D 5	P 8_14	M O D E 1 1		
pr1_mii1_txd0	MII1 Transmit Data	O	C 2	P 9_42	M O D E 1 1		
pr1_mii_mr1_clk	MII1 Receive Clock	I	C 3	P 9_27	M O D E 1 1		
pr1_mii1_rxdv	MII1 Data Valid	I	C 4	NA			
pr1_mii1_rxd3	MII1 Receive Data	I	B 2	NA			

continues on next page

Table 2.24 – continued from previous page

SIGNAL NAME	DESCRIPTION	TYPE	PROC	HEADER_PIN	MODE	HEADER_PIN	MODE
pr1_mii1_rxd2	MII1 Receive Data	I	D6	P9_14	MODE11		
pr1_mii1_rxd1	MII1 Receive Data	I	C5	P9_16	MODE11		
pr1_mii1_rxd0	MII1 Receive Data	I	A3	P8_15	MODE11		
pr1_mii1_rxer	MII1 Receive Error	I	B3	P8_26	MODE11		
pr1_mii1_rxlink	MII1 Receive Link	I	B4	P8_16	MODE11		
pr1_mii1_col	MII1 Collision Detect	I	B5	NA			
pr1_mii1_crs	MII1 Carrier Sense	I	A4	NA			
pr1_mdio_mdclk	MDIO Clock	O	D3	P8_13	MODE11		
pr1_mdio_data	MDIO Data	IO	F6	NA			
pr1_edc_latch0_in	Latch Input 0	I	AG3/E2	NA			
pr1_edc_latch1_in	Latch Input 1	I	AG5	NA			
pr1_edc_sync0_out	SYNC0 Output	O	AF2/D2	P9_20	MODE11		
pr1_edc_sync1_out	SYNC1 Output	O	AF6	NA			
pr1_edio_latch_in	Latch Input	I	AF3	NA			
pr1_edio_sof	Start Of Frame	O	AF4/F4	P9_19	MODE11		
pr1_edio_data_in0	Ethernet Digital Input	I	AF1/E1	NA			
pr1_edio_data_in1	Ethernet Digital Input	I	AE3/G2	NA			
pr1_edio_data_in2	Ethernet Digital Input	I	AE5/H7	NA			
pr1_edio_data_in3	Ethernet Digital Input	I	AE1/G1	NA			
pr1_edio_data_in4	Ethernet Digital Input	I	AE2/G6	P9_26	MODE10	P8_34	MODE12
pr1_edio_data_in5	Ethernet Digital Input	I	AE6/F2	P8_36	MODE12		
pr1_edio_data_in6	Ethernet Digital Input	I	AD2/F3	NA			
pr1_edio_data_in7	Ethernet Digital Input	I	AD3/D1	P8_15	MODE12		
p r1_edio_data_out0	Ethernet Digital Output	O	AF1/E1	NA			
p r1_edio_data_out1	Ethernet Digital Output	O	AE3/G2	NA			
p r1_edio_data_out2	Ethernet Digital Output	O	AE5/H7	NA			
p r1_edio_data_out3	Ethernet Digital Output	O	AE1/G1	NA			
p r1_edio_data_out4	Ethernet Digital Output	O	AE2/G6	P9_26	MODE11	P8_34	MODE13
p r1_edio_data_out5	Ethernet Digital Output	O	AE6/F2	P8_36	MODE13		
p r1_edio_data_out6	Ethernet Digital Output	O	AD2/F3	NA			
p r1_edio_data_out7	Ethernet Digital Output	O	AD3/D1	P8_15	MODE13		
pr1_uart0_cts_n	UART Clear-To-Send	I	G1/F11	P8_45	MODE10		
pr1_uart0_rts_n	UART Ready-To-Send	O	G6/G10	P8_34	MODE11	P8_46	MODE10
pr1_uart0_rxd	UART Receive Data	I	F2/F10	P8_36	MODE11	P8_43	MODE10
pr1_uart0_txd	UART Transmit Data	O	F3/G11	P8_44	MODE10		
pr1_ecap0_ ecap_capin_apwm_o	Capture Input/PWM Output	IO	D1/E9	P8_15	MODE11	P8_41	MODE10

PRU-ICSS2 Pin Access The table below shows which PRU-ICSS2 signals can be accessed on BeagleBone® AI and on which connector and pins they are accessible from. Some signals are accessible on the same pins. Signal Names reveal which PRU-ICSS Subsystem is being addressed. pr1 is PRU-ICSS1 and pr2 is PRU-ICSS2

Table 2.25: PRU-ICSS2 Pin Access

SIGNAL NAME	DESCR IPTION	TYPE	PROC	HEAD ER_PIN	MODE	HEAD ER_PIN	MODE
p r2_pru 0_gpo0	PRU0 Gen eral-P urpose Output	O	G 11/AC5	P8_44	MODE13		
p r2_pru 0_gpo1	PRU0 Gen eral-P urpose Output	O	E9/AB4	P8_41	MODE13		
p r2_pru 0_gpo2	PRU0 Gen eral-P urpose Output	O	F9/AD4	P8_42	MODE13	P8_21	MODE13
p r2_pru 0_gpo3	PRU0 Gen eral-P urpose Output	O	F8/AC4	P8_39	MODE13	P8_20	MODE13
p r2_pru 0_gpo4	PRU0 Gen eral-P urpose Output	O	E7/AC7	P8_40	MODE13	P8_25	MODE13
p r2_pru 0_gpo5	PRU0 Gen eral-P urpose Output	O	E8/AC6	P8_37	MODE13	P8_24	MODE13
p r2_pru 0_gpo6	PRU0 Gen eral-P urpose Output	O	D9/AC9	P8_38	MODE13	P8_5	MODE13
p r2_pru 0_gpo7	PRU0 Gen eral-P urpose Output	O	D7/AC3	P8_36	MODE13	P8_6	MODE13
p r2_pru 0_gpo8	PRU0 Gen eral-P urpose Output	O	D8/AC8	P8_34	MODE13	P8_23	MODE13
p r2_pru 0_gpo9	PRU0 Gen eral-P urpose Output	O	A5/AD6	P8_35	MODE13	P8_22	MODE13
pr 2_pru0 _gpo10	PRU0 Gen eral-P urpose Output	O	C6/AB8	P8_33	MODE13	P8_3	MODE13
pr 2_pru0 _gpo11	PRU0 Gen eral-P urpose Output	O	C8/AB5	P8_31	MODE13	P8_4	MODE13
pr 2_pru0 _gpo12	PRU0 Gen eral-P urpose Output	O	C7/B18	P8_32	MODE13		
pr 2_pru0 _gpo13	PRU0 Gen eral-P urpose Output	O	B7/F15	P8_45	MODE13		
pr 2_pru0 _gpo14	PRU0 Gen eral-P urpose Output	O	B8/B19	P9_11	MODE13	P9_11	MODE13
pr 2_pru0 _gpo15	PRU0 Gen eral-P urpose Output	O	A7/C17	P8_17	MODE13	P9_13	MODE13
pr 2_pru0 _gpo16	PRU0 Gen eral-P urpose Output	O	A8/C15	P8_27	MODE13		
pr 2_pru0 _gpo17	PRU0 Gen eral-P urpose Output	O	C9/A16	P8_28	MODE13		
pr 2_pru0 _gpo18	PRU0 Gen eral-P urpose Output	O	A9/A19	P8_29	MODE13		
pr 2_pru0 _gpo19	PRU0 Gen eral-P urpose Output	O	B9/A18	P8_30	MODE13		
pr 2_pru0 _gpo20	PRU0 Gen eral-P urpose Output	O	A 10/F14	P8_46	MODE13	P8_8	MODE13
p r2_pru 0_gpi0	PRU0 Gen eral-P urpose Input	I	G 11/AC5	P8_44	MODE12		
p r2_pru 0_gpi1	PRU0 Gen eral-P urpose Input	I	E9/AB4	P8_41	MODE12		
p r2_pru 0_gpi2	PRU0 Gen eral-P urpose Input	I	F9/AD4	P8_42	MODE12	P8_21	MODE12
p r2_pru 0_gpi3	PRU0 Gen eral-P urpose Input	I	F8/AC4	P8_39	MODE12	P8_20	MODE12
p r2_pru 0_gpi4	PRU0 Gen eral-P urpose Input	I	E7/AC7	P8_40	MODE12	P8_25	MODE12
p r2_pru 0_gpi5	PRU0 Gen eral-P urpose Input	I	E8/AC6	P8_37	MODE12	P8_24	MODE12
p r2_pru 0_gpi6	PRU0 Gen eral-P urpose Input	I	D9/AC9	P8_38	MODE12	P8_5	MODE12
p r2_pru 0_gpi7	PRU0 Gen eral-P urpose Input	I	D7/AC3	P8_36	MODE12	P8_6	MODE12
p r2_pru 0_gpi8	PRU0 Gen eral-P urpose Input	I	D8/AC8	P8_34	MODE12	P8_23	MODE12
p r2_pru 0_gpi9	PRU0 Gen eral-P urpose Input	I	A5/AD6	P8_35	MODE12	P8_22	MODE12
pr 2_pru0 _gpi10	PRU0 Gen eral-P urpose Input	I	C6/AB8	P8_33	MODE12	P8_3	MODE12
pr 2_pru0 _gpi11	PRU0 Gen eral-P urpose Input	I	C8/AB5	P8_31	MODE12	P8_4	MODE12
pr 2_pru0 _gpi12	PRU0 Gen eral-P urpose Input	I	C7/B18	P8_32	MODE12		
pr 2_pru0 _gpi13	PRU0 Gen eral-P urpose Input	I	B7/F15	P8_45	MODE12		

continues on next page

Table 2.25 – continued from previous page

SIGNAL NAME	DESCR IPTION	TYPE	PROC	HEAD ER_PIN	MODE	HEAD ER_PIN	MODE
pr 2_pru0 _gpi14	PRU0 Gen eral-P urpose Input	I	B8/B19	P9_11	MODE12	P9_11	MODE12
pr 2_pru0 _gpi15	PRU0 Gen eral-P urpose Input	I	A7/C17	P8_17	MODE12	P9_13	MODE12
pr 2_pru0 _gpi16	PRU0 Gen eral-P urpose Input	I	A8/C15	P8_27	MODE12		
pr 2_pru0 _gpi17	PRU0 Gen eral-P urpose Input	I	C9/A16	P8_28	MODE12		
pr 2_pru0 _gpi18	PRU0 Gen eral-P urpose Input	I	A9/A19	P8_29	MODE12		
pr 2_pru0 _gpi19	PRU0 Gen eral-P urpose Input	I	B9/A18	P8_30	MODE12		
pr 2_pru0 _gpi20	PRU0 Gen eral-P urpose Input	I	A 10/F14	P8_46	MODE12	P8_8	MODE12
p r2_pru 1_gpo0	PRU1 Gen eral-P urpose Output	O	V1/D17	P8_32	MODE13		
p r2_pru 1_gpo1	PRU1 Gen eral-P urpose Output	O	U4/AA3	NA			
p r2_pru 1_gpo2	PRU1 Gen eral-P urpose Output	O	U3/AB9	NA			
p r2_pru 1_gpo3	PRU1 Gen eral-P urpose Output	O	V2/AB3	NA			
p r2_pru 1_gpo4	PRU1 Gen eral-P urpose Output	O	Y1/AA4	NA			
p r2_pru 1_gpo5	PRU1 Gen eral-P urpose Output	O	W9/D18	P9_25	MODE13		
p r2_pru 1_gpo6	PRU1 Gen eral-P urpose Output	O	V9/E17	P8_9	MODE13		
p r2_pru 1_gpo7	PRU1 Gen eral-P urpose Output	O	V7/C14	P9_31	MODE13		
p r2_pru 1_gpo8	PRU1 Gen eral-P urpose Output	O	U7/G12	P9_18	MODE13		
p r2_pru 1_gpo9	PRU1 Gen eral-P urpose Output	O	V6/F12	P9_17	MODE13		
pr 2_pru1 _gpo10	PRU1 Gen eral-P urpose Output	O	U6/B12	P9_31	MODE13		
pr 2_pru1 _gpo11	PRU1 Gen eral-P urpose Output	O	U5/A11	P9_29	MODE13		
pr 2_pru1 _gpo12	PRU1 Gen eral-P urpose Output	O	V5/B13	P9_30	MODE13		
pr 2_pru1 _gpo13	PRU1 Gen eral-P urpose Output	O	V4/A12	P9_26	MODE13		
pr 2_pru1 _gpo14	PRU1 Gen eral-P urpose Output	O	V3/E14	P9_42	MODE13		
pr 2_pru1 _gpo15	PRU1 Gen eral-P urpose Output	O	Y2/A13	P8_10	MODE13		
pr 2_pru1 _gpo16	PRU1 Gen eral-P urpose Output	O	W2/G14	P8_7	MODE13		
pr 2_pru1 _gpo17	PRU1 Gen eral-P urpose Output	O	E11	P8_27	MODE13		
pr 2_pru1 _gpo18	PRU1 Gen eral-P urpose Output	O	F11	P8_45	MODE13		
pr 2_pru1 _gpo19	PRU1 Gen eral-P urpose Output	O	G10	P8_46	MODE13		
pr 2_pru1 _gpo20	PRU1 Gen eral-P urpose Output	O	F10	P8_43	MODE13		
p r2_pru 1_gpi0	PRU1 Gen eral-P urpose Input	I	V1/D17	P8_32	MODE12		
p r2_pru 1_gpi1	PRU1 Gen eral-P urpose Input	I	U4/AA3	NA			
p r2_pru 1_gpi2	PRU1 Gen eral-P urpose Input	I	U3/AB9	NA			
p r2_pru 1_gpi3	PRU1 Gen eral-P urpose Input	I	V2/AB3	NA			
p r2_pru 1_gpi4	PRU1 Gen eral-P urpose Input	I	Y1/AA4	NA			
p r2_pru 1_gpi5	PRU1 Gen eral-P urpose Input	I	W9/D18	P9_25	MODE12		
p r2_pru 1_gpi6	PRU1 Gen eral-P urpose Input	I	V9/E17	P8_9	MODE12		
p r2_pru 1_gpi7	PRU1 Gen eral-P urpose Input	I	V7/C14	P9_31	MODE12		
p r2_pru 1_gpi8	PRU1 Gen eral-P urpose Input	I	U7/G12	P9_18	MODE12		

continues on next page

Table 2.25 – continued from previous page

SIGNAL NAME	DESCR IPTION	TYPE	PROC	HEAD ER_PIN	MODE	HEAD ER_PIN	MODE
p r2_pru 1_gpi9	PRU1 Gen eral-P urpose Input	I	V6/F12	P9_17	MODE12		
pr 2_pru1 _gpi10	PRU1 Gen eral-P urpose Input	I	U6/B12	P9_31	MODE12		
pr 2_pru1 _gpi11	PRU1 Gen eral-P urpose Input	I	U5/A11	P9_29	MODE12		
pr 2_pru1 _gpi12	PRU1 Gen eral-P urpose Input	I	V5/B13	P9_30	MODE12		
pr 2_pru1 _gpi13	PRU1 Gen eral-P urpose Input	I	V4/A12	P9_28	MODE12		
pr 2_pru1 _gpi14	PRU1 Gen eral-P urpose Input	I	V3/E14	P9_42	MODE12		
pr 2_pru1 _gpi15	PRU1 Gen eral-P urpose Input	I	Y2/A13	P8_10	MODE12		
pr 2_pru1 _gpi16	PRU1 Gen eral-P urpose Input	I	W2/G14	P8_7	MODE12		
pr 2_pru1 _gpi17	PRU1 Gen eral-P urpose Input	I	E11	P8_27	MODE12		
pr 2_pru1 _gpi18	PRU1 Gen eral-P urpose Input	I	F11	P8_45	MODE12		
pr 2_pru1 _gpi19	PRU1 Gen eral-P urpose Input	I	G10	P8_46	MODE12		
pr 2_pru1 _gpi20	PRU1 Gen eral-P urpose Input	I	F10	P8_43	MODE12		
pr2_e dc_lat ch0_in	Latch Input 0	I	F9	P8_42	MODE10		
pr2_e dc_lat ch1_in	Latch Input 1	I	F8	P8_39	MODE10		
pr2_e dc_syn c0_out	SYNC0 Output	O	E7	P8_40	MODE10		
pr2_e dc_syn c1_out	SYNC1 Output	O	E8	P8_37	MODE10		
pr2_e dio_la tch_in	Latch Input	I	D9	P8_38	MODE10		
pr2_ed io_sof	Start Of Frame	O	D7	P8_36	MODE10		
pr2 _uart0 _cts_n	UART C lear-T o-Send	I	D8	P8_34	MODE10		
pr2 _uart0 _rts_n	UART R eady-T o-Send	O	A5	P8_35	MODE10		
p r2_uar t0_rxd	UART R eceive Data	I	C6	P8_33	MODE10		
p r2_uar t0_txd	UART Tr ansmit Data	O	C8	P8_31	MODE10		
pr2 _ecap0 _ecap_ **capin_** apwm_o	C apture Inp ut/PWM output	IO	C7	P8_32	MODE10		
pr2_e dio_da ta_in0	Et hernet D igital Input	I	B7	P8_45	MODE10		
pr2_e dio_da ta_in1	Et hernet D igital Input	I	B8	P9_11	MODE10		
pr2_e dio_da ta_in2	Et hernet D igital Input	I	A7	P8_17	MODE10		
pr2_e dio_da ta_in3	Et hernet D igital Input	I	A8	P8_27	MODE10		
pr2_e dio_da ta_in4	Et hernet D igital Input	I	C9	P8_28	MODE10		
pr2_e dio_da ta_in5	Et hernet D igital Input	I	A9	P8_29	MODE10		
pr2_e dio_da ta_in6	Et hernet D igital Input	I	B9	P8_30	MODE10		
pr2_e dio_da ta_in7	Et hernet D igital Input	I	A10	P8_46	MODE10		
pr2_ed io_dat a_out0	Et hernet D igital Output	O	B7	P8_45	MODE11		
pr2_ed io_dat a_out1	Et hernet D igital Output	O	B8	P9_11	MODE11		
pr2_ed io_dat a_out2	Et hernet D igital Output	O	A7	P8_17	MODE11		
pr2_ed io_dat a_out3	Et hernet D igital Output	O	A8	P8_27	MODE11		
pr2_ed io_dat a_out4	Et hernet D igital Output	O	C9	P8_28	MODE11		
pr2_ed io_dat a_out5	Et hernet D igital Output	O	A9	P8_29	MODE11		

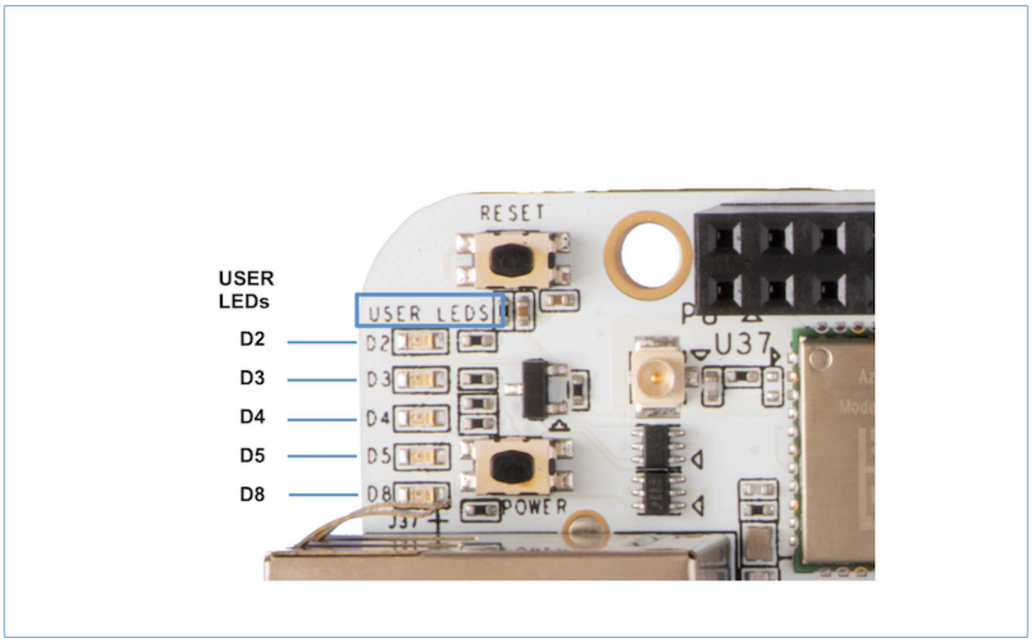
continues on next page

Table 2.25 – continued from previous page

SIGNAL NAME	DESCR IPTION	TYPE	PROC	HEAD ER_PIN	MODE	HEAD ER_PIN	MODE
pr2_ed io_dat a_out6	Et hernet D igital Output	O	B9	P8_30	MODE11		
pr2_ed io_dat a_out7	Et hernet D igital Output	O	A10	P8_46	MODE11		
pr2_mi i1_col	MII1 Col lision Detect	I	D18	P9_25	MODE11		
pr2_mi i1_crs	MII1 C arrier Sense	I	E17	P8_9	MODE11		
pr 2_mdio _mdclk	MDIO Clock	O	C 14/AB3	P9_31	MODE11		
p r2_mdi o_data	MDIO Data	IO	D 14/AA4	P9_29	MODE11		
p r2_mii 0_rxer	MII0 R eceive Error	I	G12	P9_18	MODE11		
pr2 _mii_m t0_clk	MII0 Tr ansmit Clock	I	F12	P9_17	MODE11		
p r2_mii 0_txen	MII0 Tr ansmit Enable	O	B12	P9_31	MODE11		
p r2_mii 0_txd3	MII0 Tr ansmit Data	O	A11	P9_29	MODE11		
p r2_mii 0_txd2	MII0 Tr ansmit Data	O	B13	P9_30	MODE11		
p r2_mii 0_txd1	MII0 Tr ansmit Data	O	A12	P9_28	MODE11		
p r2_mii 0_txd0	MII0 Tr ansmit Data	O	E14	P9_42	MODE11		
pr2 _mii_m r0_clk	MII0 R eceive Clock	I	A13	P8_10	MODE11		
p r2_mii 0_rxdv	MII0 Data Valid	I	G14	P8_7	MODE11		
p r2_mii 0_rxd3	MII0 R eceive Data	I	F14	P8_8	MODE11		
p r2_mii 0_rxd2	MII0 R eceive Data	I	A19	NA			
p r2_mii 0_rxd1	MII0 R eceive Data	I	A18	NA			
p r2_mii 0_rxd0	MII0 R eceive Data	I	C15	NA			
pr2 _mii0_ rxlink	MII0 R eceive Link	I	A16	NA			
pr2_mi i0_crs	MII0 C arrier Sense	I	B18	NA			
pr2_mi i0_col	MII0 Col lision Detect	I	F15	NA			
p r2_mii 1_rxer	MII1 R eceive Error	I	B19	P9_11	MODE11		
pr2 _mii1_ rxlink	MII1 R eceive Link	I	C17	P9_13	MODE11		
pr2 _mii_m t1_clk	MII1 Tr ansmit Clock	I	AC5	NA			
p r2_mii 1_txen	MII1 Tr ansmit Enable	O	AB4	NA			
p r2_mii 1_txd3	MII1 Tr ansmit Data	O	AD4	P8_21	MODE11		
p r2_mii 1_txd2	MII1 Tr ansmit Data	O	AC4	P8_20	MODE11		
p r2_mii 1_txd1	MII1 Tr ansmit Data	O	AC7	P8_25	MODE11		
p r2_mii 1_txd0	MII1 Tr ansmit Data	O	AC6	P8_24	MODE11		
pr2 _mii_m r1_clk	MII1 R eceive Clock	I	AC9	P8_5	MODE11		
p r2_mii 1_rxdv	MII1 Data Valid	I	AC3	P8_6	MODE11		
p r2_mii 1_rxd3	MII1 R eceive Data	I	AC8	P8_23	MODE11		
p r2_mii 1_rxd2	MII1 R eceive Data	I	AD6	P8_22	MODE11		
p r2_mii 1_rxd1	MII1 R eceive Data	I	AB8	P8_3	MODE11		
p r2_mii 1_rxd0	MII1 R eceive Data	I	AB5	P8_4	MODE11		
end	end	end	end	end	end	end	end

User LEDs

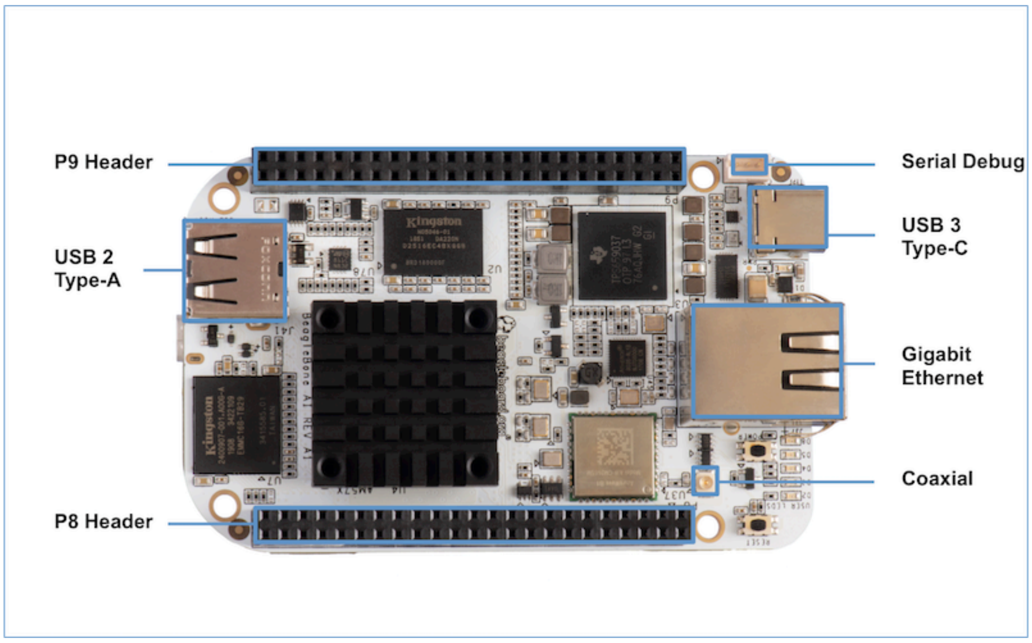
There are 5 User Programmable LEDs on BeagleBone® AI. These are connected to GPIO pins on the processor.

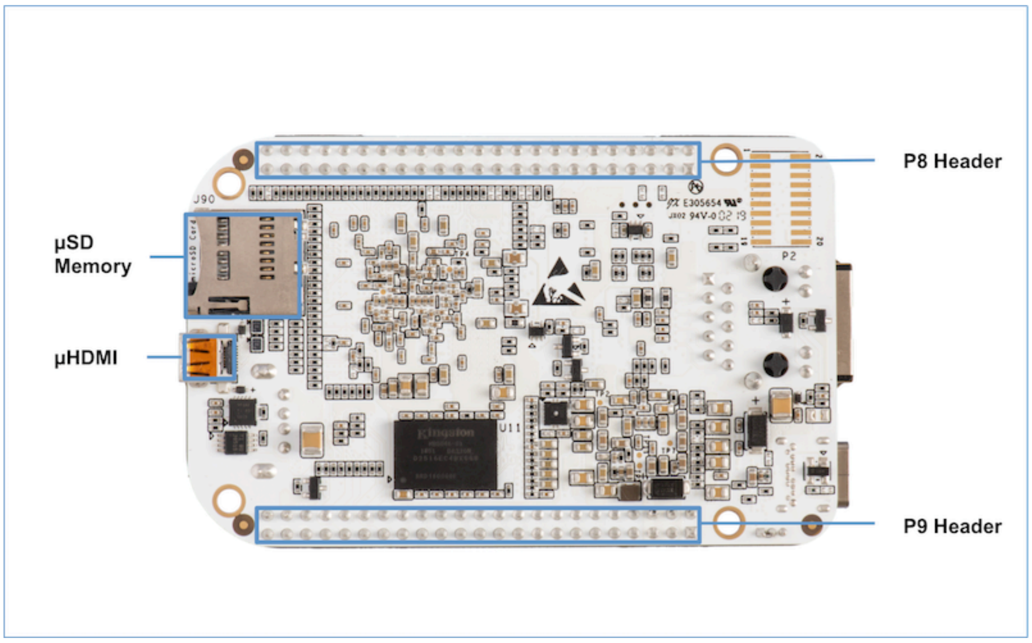


The table shows the signals used to control the LEDs from the processor. Each LED is user programmable. However, there is a Default Functions assigned in the device tree for BeagleBone® AI:

LED	GPIO SIGNAL	DEFAULT FUNCTION
D2	GPIO3_17	Heartbeat When Linux is Running
D3	GPIO5_5	microSD Activity
D4	GPIO3_15	CPU Activity
D5	GPIO3_14	eMMC Activity
D8	GPIO3_7	WiFi/Bluetooth Activity

2.4.7 Connectors





Expansion Connectors

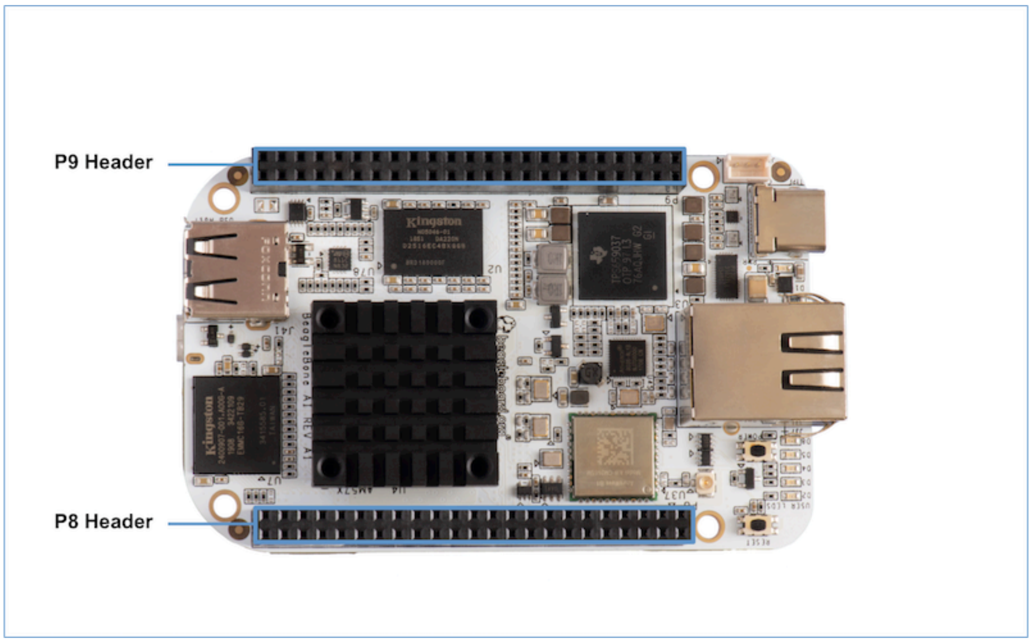
The expansion interface on the board is comprised of two 46 pin connectors, the P8 and P9 Headers. All signals on the expansion headers are **3.3V** unless otherwise indicated.

Note: Do not connect 5V logic level signals to these pins or the board will be damaged.

Note: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

Figure ? shows the location of the expansion connectors.



The location and spacing of the expansion headers are the same as on BeagleBone Black.

Connector P8 The following tables show the pinout of the **P8** expansion header. The SW is responsible for setting the default function of each pin. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The column heading is the pin number on the expansion header.

The **GPIO** row is the expected gpio identifier number in the Linux kernel.

The **BALL** row is the pin number on the processor.

The **REG** row is the offset of the control register for the processor pin.

The **MODE #** rows are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

If included, the **2nd BALL** row is the pin number on the processor for a second processor pin connected to the same pin on the expansion header. Similarly, all row headings starting with **2nd** refer to data for this second processor pin.

Note: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

Table 2.26: P8.01-P8.02

P8.01	P8.02
GND	GND

Table 2.27: P8.03-P8.05

	P8.03	P8.04	P8.05
GPIO	24	25	193
BALL	AB8	AB5	AC9
REG	0x179C	0x17A0	0x178C
MODE 0	mmc3_dat6	mmc3_dat7	mmc3_dat2
1	spi4_d0	spi4_cs0	spi3_cs0
2	uart10_ctsn	uart10_rtsn	uart5_ctsn
3			
4	vin2b_de1	vin2b_clk1	vin2b_d3
5			
6			
7			
8			
9	vin5a_hsync0	vin5a_vsync0	vin5a_d3
10	ehrpwm3_tripzone_input	eCAP3_in_PWM3_out	eQEP3_index
11	pr2_mii1_rxd1	pr2_mii1_rxd0	pr2_mii_mr1_clk
12	pr2_pru0_gpi10	pr2_pru0_gpi11	pr2_pru0_gpi6
13	pr2_pru0_gpo10	pr2_pru0_gpo11	pr2_pru0_gpo6
14	gpio1_24	gpio1_25	gpio7_1
15	Driver off	Driver off	Driver off

Table 2.28: P8.06-P8.09

	P8.06	P8.07	P8.08	P8.09
GPIO	194	165	166	178
BALL	AC3	G14	F14	E17
REG	0x1790	0x16EC	0x16F0	0x1698
MODE0	mmc3_dat3	mcasp1_axr14	mcasp1_axr15	xref_clk1
1	spi3_cs1	mcasp7_aclkx	mcasp7_fsx	mcasp2_axr9
2	uart5_rtsn	mcasp7_aclkr	mcasp7_fsr	mcasp1_axr5
3				mcasp2_ahclkx
4	vin2b_d2			mcasp6_ahclkx
5				
6				
7		vin6a_d9	vin6a_d8	vin6a_clk0
8				
9	vin5a_d2			
10	eQEP3_strobe	timer11	timer12	timer14
11	pr2_mii1_rxdv	pr2_mii0_rxdv	pr2_mii0_rxd3	pr2_mii1_crs
12	pr2_pru0_gpi7	pr2_pru1_gpi16	pr2_pru0_gpi20	pr2_pru1_gpi6
13	pr2_pru0_gpo7	pr2_pru1_gpo16	pr2_pru0_gpo20	pr2_pru1_gpo6
14	gpio7_2	gpio6_5	gpio6_6	gpio6_18
15	Driver off	Driver off	Driver off	Driver off

Table 2.29: P8.10-P8.13

	P8.10	P8.11	P8.12	P8.13
GPIO	164	75	74	107
BALL	A13	AH4	AG6	D3
REG	0x16E8	0x1510	0x150C	0x1590
MODE 0	mcasp1_axr13	vin1a_d7	vin1a_d6	vin2a_d10
1	mcasp7_axr1			
2				
3		vout3_d0	vout3_d1	mdio_mclk
4		vout3_d16	vout3_d17	vout2_d13
5				
6				
7	vin6a_d10			
8				
9				kbd_col7
10	timer10	eQEP2B_in	eQEP2A_in	ehrpwm2B
11	pr2_mii_mr0_clk			pr1_mdio_mdclk
12	pr2_pru1_gpi15	pr1_pru0_gpi4	pr1_pru0_gpi3	pr1_pru1_gpi7
13	pr2_pru1_gpo15	pr1_pru0_gpo4	pr1_pru0_gpo3	pr1_pru1_gpo7
14	gpio6_4	gpio3_11	gpio3_10	gpio4_11
15	Driver off	Driver off	Driver off	Driver off

Table 2.30: P8.14-P8.16

	P8.14	P8.15	P8.16
GPIO	109	99	125
BALL	D5	D1	B4
REG	0x1598	0x1570	0x15BC
MODE 0	vin2a_d12	vin2a_d2	vin2a_d21
1			
2			vin2b_d2
3	rgmii1_txc		rgmii1_rxd2
4	vout2_d11	vout2_d21	vout2_d2
5		emu12	vin3a_fld0
6			vin3a_d13
7			
8	mii1_rxclk	uart10_rxd	mii1_col
9	kbd_col8	kbd_row6	
10	eCAP2_in_PWM2_out	eCAP1_in_PWM1_out	

continues on next page

Table 2.30 – continued from previous page

	P8.14	P8.15	P8.16
11	pr1_mii1_txd1	pr1_ecap0_ecap_capin_apwm_o	pr1_mii1_rxlink
12	pr1_pru1_gpi9	pr1_edio_data_in7	pr1_pru1_gpi18
13	pr1_pru1_gpo9	pr1_edio_data_out7	pr1_pru1_gpo18
14	gpio4_13	gpio4_3	gpio4_29
15	Driver off	Driver off	Driver off
2nd BALL		A3	
2nd REG		0x15B4	
2nd MODE 0		vin2a_d19	
2nd 1			
2nd 2		vin2b_d4	
2nd 3		rgmii1_rxctl	
2nd 4		vout2_d4	
2nd 5			
2nd 6		vin3a_d11	
2nd 7			
2nd 8		mii1_txer	
2nd 9			
2nd 10		ehrpwm3_tripzone_input	
2nd 11		pr1_mii1_rxd0	
2nd 12		pr1_pru1_gpi16	
2nd 13		pr1_pru1_gpo16	
2nd 14		gpio4_27	
2nd 15		Driver off	

Table 2.31: P8.17-P8.19

	P8.17	P8.18	P8.19
GPIO	242	105	106
BALL	A7	F5	E6
REG	0x1624	0x1588	0x158C
MODE 0	vout1_d18	vin2a_d8	vin2a_d9
1			
2	emu4		
3	vin4a_d2		
4	vin3a_d2	vout2_d15	vout2_d14
5	obs11	emu18	emu19
6	obs27		
7			
8		mii1_rxd3	mii1_rxd0
9		kbd_col5	kbd_col6
10	pr2_edio_data_in2	eQEP2_strobe	ehrpwm2A
11	pr2_edio_data_out2	pr1_mii1_txd3	pr1_mii1_txd2
12	pr2_pru0_gpi15	pr1_pru1_gpi5	pr1_pru1_gpi6
13	pr2_pru0_gpo15	pr1_pru1_gpo5	pr1_pru1_gpo6
14	gpio8_18	gpio4_9	gpio4_10
15	Driver off	Driver off	Driver off

Table 2.32: P8.20-P8.22

	P8.20	P8.21	P8.22
GPIO	190	189	23
BALL	AC4	AD4	AD6
REG	0x1780	0x177C	0x1798
MODE 0	mmc3_cmd	mmc3_clk	mmc3_dat5
1	spi3_sclk		spi4_d1
2			uart10_txd
3			
4	vin2b_d6	vin2b_d7	vin2b_d0
5			
6			
7			
8			
9	vin5a_d6	vin5a_d7	vin5a_d0
10	eCAP2_in_PWM2_out	ehrpwm2_tripzone_input	ehrpwm3B
11	pr2_mii1_txd2	pr2_mii1_txd3	pr2_mii1_rxd2
12	pr2_pru0_gpi3	pr2_pru0_gpi2	pr2_pru0_gpi9
13	pr2_pru0_gpo3	pr2_pru0_gpo2	pr2_pru0_gpo9
14	gpio6_30	gpio6_29	gpio1_23
15	Driver off	Driver off	Driver off

Table 2.33: P8.23-P8.26

	P8.23	P8.24	P8.25	P8.26
GPIO	22	192	191	124
BALL	AC8	AC6	AC7	B3
REG	0x1794	0x1788	0x1784	0x15B8
MODE 0	mmc3_dat4	mmc3_dat1	mmc3_dat0	vin2a_d20
1	spi4_sclk	spi3_d0	spi3_d1	
2	uart10_rxd	uart5_txd	uart5_rxd	vin2b_d3
3				rgmii1_rxd3
4	vin2b_d1	vin2b_d4	vin2b_d5	vout2_d3
5				vin3a_de0
6				vin3a_d12
7				
8				mii1_rxer
9	vin5a_d1	vin5a_d4	vin5a_d5	
10	ehrpwm3A	eQEP3B_in	eQEP3A_in	eCAP3_in_PWM3_out
11	pr2_mii1_rxd3	pr2_mii1_txd0	pr2_mii1_txd1	pr1_mii1_rxer
12	pr2_pru0_gpi8	pr2_pru0_gpi5	pr2_pru0_gpi4	pr1_pru1_gpi17
13	pr2_pru0_gpo8	pr2_pru0_gpo5	pr2_pru0_gpo4	pr1_pru1_gpo17
14	gpio1_22	gpio7_0	gpio6_31	gpio4_28
15	Driver off	Driver off	Driver off	Driver off

Table 2.34: P8.27-P8.29

	P8.27	P8.28	P8.29
GPIO	119	115	118
BALL	E11	D11	C11
REG	0x15D8	0x15C8	0x15D4
MODE 0	vout1_vsync	vout1_clk	vout1_hsync
1			
2			
3	vin4a_vsync0	vin4a_fld0	vin4a_hsync0
4	vin3a_vsync0	vin3a_fld0	vin3a_hsync0
5			
6			
7			
8	spi3_sclk	spi3_cs0	spi3_d0
9			
10			

continues on next page

Table 2.34 – continued from previous page

	P8.27	P8.28	P8.29
11			
12	pr2_pru1_gpi17		
13	pr2_pru1_gpo17		
14	gpio4_23	gpio4_19	gpio4_22
15	Driver off	Driver off	Driver off
2nd BALL	A8	C9	A9
2nd REG	0x1628	0x162C	0x1630
2nd MODE0	vout1_d19	vout1_d20	vout1_d21
2nd 1			
2nd 2	emu15	emu16	emu17
2nd 3	vin4a_d3	vin4a_d4	vin4a_d5
2nd 4	vin3a_d3	vin3a_d4	vin3a_d5
2nd 5	obs12	obs13	obs14
2nd 6	obs28	obs29	obs30
2nd 7			
2nd 8			
2nd 9			
2nd 10	pr2_edio_data_in3	pr2_edio_data_in4	pr2_edio_data_in5
2nd 11	pr2_edio_data_out3	pr2_edio_data_out4	pr2_edio_data_out5
2nd 12	pr2_pru0_gpi16	pr2_pru0_gpi17	pr2_pru0_gpi18
2nd 13	pr2_pru0_gpo16	pr2_pru0_gpo17	pr2_pru0_gpo18
2nd 14	gpio8_19	gpio8_20	gpio8_21
2nd 15	Driver off	Driver off	Driver off

Table 2.35: P8.30-P8.32

	P8.30	P8.31	P8.32
GPIO	116	238	239
BALL	B10	C8	C7
REG	0x15CC	0x1614	0x1618
MODE 0	vout1_de	vout1_d14	vout1_d15
1			
2		emu13	emu14
3	vin4a_de0	vin4a_d14	vin4a_d15
4	vin3a_de0	vin3a_d14	vin3a_d15
5		obs9	obs10
6		obs25	obs26
7			
8	spi3_d1		
9			
10		pr2_uart0_txd	pr2_ecap0_ecap_capin_apwm_o
11			
12		pr2_pru0_gpi11	pr2_pru0_gpi12
13		pr2_pru0_gpo11	pr2_pru0_gpo12
14	gpio4_20	gpio8_14	gpio8_15
15	Driver off	Driver off	Driver off
2nd BALL	B9	G16	D17
2nd REG	0x1634	0x173C	0x1740
2nd MODE 0	vout1_d22	mcasp4_axr0	mcasp4_axr1
2nd 1			
2nd 2	emu18	spi3_d0	spi3_cs0
2nd 3	vin4a_d6	uart8_ctsn	uart8_rtsn
2nd 4	vin3a_d6	uart4_rxd	uart4_txd

continues on next page

Table 2.35 – continued from previous page

	P8.30	P8.31	P8.32
2nd 5	obs15		
2nd 6	obs31	vout2_d18	vout2_d19
2nd 7			
2nd 8		vin4a_d18	vin4a_d19
2nd 9		vin5a_d13	vin5a_d12
2nd 10	pr2_edio_data_in6		
2nd 11	pr2_edio_data_out6		
2nd 12	pr2_pru0_gpi19		pr2_pru1_gpi0
2nd 13	pr2_pru0_gpo19		pr2_pru1_gpo0
2nd 14	gpio8_22		
2nd 15	Driver off	Driver off	Driver off

Table 2.36: P8.33-P8.35

	P8.33	P8.34	P8.35
GPIO	237	235	236
BALL	C6	D8	A5
REG	0x1610	0x1608	0x160C
MODE 0	vout1_d13	vout1_d11	vout1_d12
1			
2	emu12	emu10	emu11
3	vin4a_d13	vin4a_d11	vin4a_d12
4	vin3a_d13	vin3a_d11	vin3a_d12
5	obs8	obs6	obs7
6	obs24	obs22	obs23
7		obs_dmarq2	
8			
9			
10	pr2_uart0_rxd	pr2_uart0_cts_n	pr2_uart0_rts_n
11			
12	pr2_pru0_gpi10	pr2_pru0_gpi8	pr2_pru0_gpi9
13	pr2_pru0_gpo10	pr2_pru0_gpo8	pr2_pru0_gpo9
14	gpio8_13	gpio8_11	gpio8_12
15	Driver off	Driver off	Driver off
2nd BALL	AF9	G6	AD9
2nd REG	0x14E8	0x1564	0x14E4
2nd MODE0	vin1a_fld0	vin2a_vsync0	vin1a_de0
2nd 1	vin1b_vsync1		vin1b_hsync1
2nd 2			
2nd 3		vin2b_vsync1	vout3_d17
2nd 4	vout3_clk	vout2_vsync	vout3_de
2nd 5	uart7_txd	emu9	uart7_rxd
2nd 6			
2nd 7	timer15	uart9_txd	timer16
2nd 8	spi3_d1	spi4_d1	spi3_sclk
2nd 9	kbd_row1	kbd_row3	kbd_row0
2nd 10	eQEP1B_in	ehrpwm1A	eQEP1A_in
2nd 11		pr1_uart0_rts_n	
2nd 12		pr1_edio_data_in4	
2nd 13		pr1_edio_data_out4	
2nd 14	gpio3_1	gpio4_0	gpio3_0
2nd 15	Driver off	Driver off	Driver off

Table 2.37: P8.36-P8.38

	P8.36	P8.37	P8.38
GPIO	234	232	233
BALL	D7	E8	D9
REG	0x1604	0x15FC	0x1600
MODE 0	vout1_d10	vout1_d8	vout1_d9
1			
2	emu3	uart6_rxd	uart6_txd
3	vin4a_d10	vin4a_d8	vin4a_d9
4	vin3a_d10	vin3a_d8	vin3a_d9
5	obs5		
6	obs21		
7	obs_irq2		
8			
9			
10	pr2_edio_sof	pr2_edc_sync1_out	pr2_edio_latch_in
11			
12	pr2_pru0_gpi7	pr2_pru0_gpi5	pr2_pru0_gpi6
13	pr2_pru0_gpo7	pr2_pru0_gpo5	pr2_pru0_gpo6
14	gpio8_10	gpio8_8	gpio8_9
15	Driver off	Driver off	Driver off
2nd BALL	F2	A21	C18
2nd REG	0x1568	0x1738	0x1734
2nd MODE 0	vin2a_d0	mcasp4_fsx	mcasp4_aclkx
2nd 1		mcasp4_fsr	mcasp4_aclkr
2nd 2		spi3_d1	spi3_sclk
2nd 3		uart8_txd	uart8_rxd
2nd 4	vout2_d23	i2c4_scl	i2c4_sda
2nd 5	emu10		
2nd 6		vout2_d17	vout2_d16
2nd 7	uart9_ctsn		
2nd 8	spi4_d0	vin4a_d17	vin4a_d16
2nd 9	kbd_row4	vin5a_d14	vin5a_d15
2nd 10	ehrpwm1B		
2nd 11	pr1_uart0_rxd		
2nd 12	pr1_edio_data_in5		
2nd 13	pr1_edio_data_out5		
2nd 14	gpio4_1		
2nd 15	Driver off	Driver off	Driver off

Table 2.38: P8.39-P8.41

	P8.39	P8.40	P8.41
GPIO	230	231	228
BALL	F8	E7	E9
REG	0x15F4	0x15F8	0x15EC
MODE 0	vout1_d6	vout1_d7	vout1_d4
1			
2	emu8	emu9	emu6
3	vin4a_d22	vin4a_d23	vin4a_d20
4	vin3a_d22	vin3a_d23	vin3a_d20
5	obs4		obs2
6	obs20		obs18
7			
8			
9			
10	pr2_edc_latch1_in	pr2_edc_sync0_out	pr1_ecap0_ecap_capin_apwm_o
11			
12	pr2_pru0_gpi3	pr2_pru0_gpi4	pr2_pru0_gpi1
13	pr2_pru0_gpo3	pr2_pru0_gpo4	pr2_pru0_gpo1
14	gpio8_6	gpio8_7	gpio8_4
15	Driver off	Driver off	Driver off

Table 2.39: P8.42-P8.44

	P8.42	P8.43	P8.44
GPIO	229	226	227
BALL	F9	F10	G11
REG	0x15F0	0x15E4	0x15E8
MODE 0	vout1_d5	vout1_d2	vout1_d3
1			
2	emu7	emu2	emu5
3	vin4a_d21	vin4a_d18	vin4a_d19
4	vin3a_d21	vin3a_d18	vin3a_d19
5	obs3	obs0	obs1
6	obs19	obs16	obs17
7		obs_irq1	obs_dmarq1
8			
9			
10	pr2_edc_latch0_in	pr1_uart0_rxd	pr1_uart0_txd
11			
12	pr2_pru0_gpi2	pr2_pru1_gpi20	pr2_pru0_gpi0
13	pr2_pru0_gpo2	pr2_pru1_gpo20	pr2_pru0_gpo0
14	gpio8_5	gpio8_2	gpio8_3
15	Driver off	Driver off	Driver off

Table 2.40: P8.45-P8.46

	P8.45	P8.46
GPIO	224	225
BALL	F11	G10
REG	0x15DC	0x15E0
MODE 0	vout1_d0	vout1_d1
1		
2	uart5_rxd	uart5_txd
3	vin4a_d16	vin4a_d17
4	vin3a_d16	vin3a_d17
5		
6		
7		
8	spi3_cs2	
9		
10	pr1_uart0_cts_n	pr1_uart0_rts_n

continues on next page

Table 2.40 – continued from previous page

	P8.45	P8.46
11		
12	pr2_pru1_gpi18	pr2_pru1_gpi19
13	pr2_pru1_gpo18	pr2_pru1_gpo19
14	gpio8_0	gpio8_1
15	Driver off	Driver off
2nd BALL	B7	A10
2nd REG	0x161C	0x1638
2nd MODE 0	vout1_d16	vout1_d23
2nd 1		
2nd 2	uart7_rxd	emu19
2nd 3	vin4a_d0	vin4a_d7
2nd 4	vin3a_d0	vin3a_d7
2nd 5		
2nd 6		
2nd 7		
2nd 8		spi3_cs3
2nd 9		
2nd 10	pr2_edio_data_in0	pr2_edio_data_in7
2nd 11	pr2_edio_data_out0	pr2_edio_data_out7
2nd 12	pr2_pru0_gpi13	pr2_pru0_gpi20
2nd 13	pr2_pru0_gpo13	pr2_pru0_gpo20
2nd 14	gpio8_16	gpio8_23
2nd 15	Driver off	Driver off

TODO: Notes regarding the resistors on muxed pins.

Connector P9 The following tables show the pinout of the **P9** expansion header. The SW is responsible for setting the default function of each pin. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The column heading is the pin number on the expansion header.

The **GPIO** row is the expected gpio identifier number in the Linux kernel.

The **BALL** row is the pin number on the processor.

The **REG** row is the offset of the control register for the processor pin.

The **MODE #** rows are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

If included, the **2nd BALL** row is the pin number on the processor for a second processor pin connected to the same pin on the expansion header. Similarly, all row headings starting with **2nd** refer to data for this second processor pin.

NOTES:

DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

In the table are the following notations:

PWR_BUT is a 5V level as pulled up internally by the TPS6590379. It is activated by pulling the signal to GND.

TODO: (Actually, on BeagleBone AI, I believe PWR_BUT is pulled to 3.3V, but activation is still done by pulling the signal to GND. Also, a quick grounding of PWR_BUT will trigger a system event where shutdown can occur, but there is no hardware power-off function like on BeagleBone Black via this signal. It does, however, act as a hardware power-on.)

TODO: (On BeagleBone Black, SYS_RESET was a bi-directional signal, but it is only an output from BeagleBone AI to capes on BeagleBone AI.)

Table 2.41: P9.11-P9.13

	P9.11	P9.12	P9.13
GPIO	241	128	172
BALL	B19	B14	C17
REG	0x172C	0x16AC	0x1730
MODE 0	mcasp3_axr0	mcasp1_aclkr	mcasp3_axr1
1		mcasp7_axr2	
2	mcasp2_axr14		mcasp2_axr15
3	uart7_ctsn		uart7_rtsn
4	uart5_rxd		uart5_txd
5			
6		vout2_d0	
7	vin6a_d1		vin6a_d0
8		vin4a_d0	
9			vin5a_fld0
10		i2c4_sda	
11	pr2_mii1_rxer		pr2_mii1_rxlink
12	pr2_pru0_gpi14		pr2_pru0_gpi15
13	pr2_pru0_gpo14		pr2_pru0_gpo15
14		gpio5_0	
15	Driver off	Driver off	Driver off
2nd BALL	B8		AB10**
2nd REG	0x1620		0x1680
2nd MODE 0	vout1_d17		usb1_drvvbus
2nd 1			
2nd 2	uart7_txd		
2nd 3	vin4a_d1		
2nd 4	vin3a_d1		
2nd 5			
2nd 6			
2nd 7			timer16
2nd 8			
2nd 9			
2nd 10	pr2_edio_data_in1		
2nd 11	pr2_edio_data_out1		
2nd 12	pr2_pru0_gpi14		
2nd 13	pr2_pru0_gpo14		
2nd 14	gpio8_17		gpio6_12
2nd 15	Driver off		Driver off

Table 2.42: P9.14-P9.16

	P9.14	P9.15	P9.16
GPIO	121	76	122
BALL	D6	AG4	C5
REG	0x15AC	0x1514	0x15B0
MODE 0	vin2a_d17	vin1a_d8	vin2a_d18
1		vin1b_d7	
2	vin2b_d6		vin2b_d5
3	rgmii1_txd0		rgmii1_rxc
4	vout2_d6	vout3_d15	vout2_d5
5			
6	vin3a_d9		vin3a_d10
7			
8	mii1_txd2		mii1_txd3
9		kbd_row2	
10	ehrpwm3A	eQEP2_index	ehrpwm3B
11	pr1_mii1_rxd2		pr1_mii1_rxd1
12	pr1_pru1_gpi14	pr1_pru0_gpi5	pr1_pru1_gpi15
13	pr1_pru1_gpo14	pr1_pru0_gpo5	pr1_pru1_gpo15
14	gpio4_25	gpio3_12	gpio4_26
15	Driver off	Driver off	Driver off

Table 2.43: P9.17-P9.19

	P9.17	P9.18	P9.19
GPIO	209	208	195
BALL	B24	G17	R6
REG	0x17CC	0x17C8	0x1440
MODE 0	spi2_cs0	spi2_d0	gpmc_a0
1	uart3_rtsn	uart3_ctsn	
2	uart5_txd	uart5_rxd	vin3a_d16
3			vout3_d16
4			vin4a_d0
5			
6			vin4b_d0
7			i2c4_scl
8			uart5_rxd
9			
10			
11			
12			
13			
14	gpio7_17	gpio7_16	gpio7_3
15	Driver off	Driver off	Driver off
2nd BALL	F12	G12	F4
2nd REG	0x16B8	0x16B4	0x157C
2nd MODE 0	mcasp1_axr1	mcasp1_axr0	vin2a_d5
2nd 1			
2nd 2			
2nd 3	uart6_txd	uart6_rxd	
2nd 4			vout2_d18
2nd 5			emu15
2nd 6			
2nd 7	vin6a_hsync0	vin6a_vsync0	
2nd 8			uart10_rtsn
2nd 9			kbd_col2
2nd 10	i2c5_scl	i2c5_sda	eQEP2A_in
2nd 11	pr2_mii_mt0_clk	pr2_mii0_rxer	pr1_edio_sof

continues on next page

Table 2.43 – continued from previous page

	P9.17	P9.18	P9.19
2nd 12	pr2_pru1_gpi9	pr2_pru1_gpi8	pr1_pru1_gpi2
2nd 13	pr2_pru1_gpo9	pr2_pru1_gpo8	pr1_pru1_gpo2
2nd 14	gpio5_3	gpio5_2	gpio4_6
2nd 15	Driver off	Driver off	Driver off

Table 2.44: P9.20-P9.22

	P9.20	P9.21	P9.22
GPIO	196	67	179
BALL	T9	AF8	B26
REG	0x1444	0x14F0	0x169C
MODE 0	gpmc_a1	vin1a_vsync0	xref_clk2
1		vin1b_de1	mcasp2_axr10
2	vin3a_d17		mcasp1_axr6
3	vout3_d17		mcasp3_ahclkx
4	vin4a_d1	vout3_vsync	mcasp7_ahclkx
5		uart7_rtsn	
6	vin4b_d1		vout2_clk
7	i2c4_sda	timer13	
8	uart5_txd	spi3_cs0	vin4a_clk0
9			
10		eQEP1_strobe	timer15
11			
12			
13			
14	gpio7_4	gpio3_3	gpio6_19
15	Driver off	Driver off	Driver off
2nd BALL	D2	B22	A26
2nd REG	0x1578	0x17C4	0x17C0
2nd MODE 0	vin2a_d4	spi2_d1	spi2_sclk
2nd 1		uart3_txd	uart3_rxd
2nd 2			
2nd 3			
2nd 4	vout2_d19		
2nd 5	emu14		
2nd 6			
2nd 7			
2nd 8	uart10_ctsn		
2nd 9	kbd_col1		
2nd 10	ehrpwm1_synco		
2nd 11	pr1_edc_sync0_out		
2nd 12	pr1_pru1_gpi1		
2nd 13	pr1_pru1_gpo1		
2nd 14	gpio4_5	gpio7_15	gpio7_14
2nd 15	Driver off	Driver off	Driver off

Table 2.45: P9.23-P9.25

	P9.23	P9.24	P9.25
GPIO	203	175	177
BALL	A22	F20	D18
REG	0x17B4	0x168C	0x1694
MODE 0	spi1_cs1	gpio6_15	xref_clk0
1		mcasp1_axr9	mcasp2_axr8
2	sata1_led	dcan2_rx	mcasp1_axr4
3	spi2_cs1	uart10_txd	mcasp1_ahclkx
4			mcasp5_ahclkx
5			
6		vout2_vsync	
7			vin6a_d0
8		vin4a_vsync0	hdq0
9		i2c3_scl	clkout2
10		timer2	timer13
11			pr2_mii1_col
12			pr2_pru1_gpi5
13			pr2_pru1_gpo5
14	gpio7_11	gpio6_15	gpio6_17
15	Driver off	Driver off	Driver off

Table 2.46: P9.26-P9.29

	P9.26	P9.27	P9.28	P9.29
GPIO	174	111	113	139
BALL	E21	C3	A12	A11
REG	0x1688	0x15A0	0x16E0	0x16D8
MODE 0	gpio6_14	vin2a_d14	mcasp1_axr11	mcasp1_axr9
1	mcasp1_axr8		mcasp6_fsx	mcasp6_axr1
2	dcan2_tx		mcasp6_fsr	
3	uart10_rxd	rgmii1_txd3	spi3_cs0	spi3_d1
4		vout2_d9		
5				
6	vout2_hsync			
7			vin6a_d12	vin6a_d14
8	vin4a_hsync0	mii1_txclk		
9	i2c3_sda			
10	timer1	eQEP3B_in	timer8	timer6
11		pr1_mii_mr1_clk	pr2_mii0_txd1	pr2_mii0_txd3
12		pr1_pru1_gpi11	pr2_pru1_gpi13	pr2_pru1_gpi11
13		pr1_pru1_gpo11	pr2_pru1_gpo13	pr2_pru1_gpo11
14	gpio6_14	gpio4_15	gpio4_17	gpio5_11
15	Driver off	Driver off	Driver off	Driver off
2nd BALL	AE2	J14		D14
2nd REG	0x1544	0x16B0		0x16A8
2nd MODE 0	vin1a_d20	mcasp1_fsr		mcasp1_fsx
2nd 1	vin1b_d3	mcasp7_axr3		
2nd 2				
2nd 3				
2nd 4	vout3_d3			
2nd 5				
2nd 6	vin3a_d4	vout2_d1		
2nd 7				vin6a_de0
2nd 8		vin4a_d1		
2nd 9	kbd_col5			
2nd 10	pr1_edio_data_in4	i2c4_scl		i2c3_scl
2nd 11	pr1_edio_data_out4			pr2_mdio_data

continues on next page

Table 2.46 – continued from previous page

	P9.26	P9.27	P9.28	P9.29
2nd 12	pr1_pru0_gpi17			
2nd 13	pr1_pru0_gpo17			
2nd 14	gpio3_24	gpio5_1		gpio7_30
2nd 15	Driver off	Driver off		Driveroff

Table 2.47: P9.30-P9.31

	P9.30	P9.31
GPIO	140	138
BALL	B13	B12
REG	0x16DC	0x16D4
MODE 0	mcasp1_axr10	mcasp1_axr8
1	mcasp6_aclkx	mcasp6_axr0
2	mcasp6_aclkr	
3	spi3_d0	spi3_sclk
4		
5		
6		
7	vin6a_d13	vin6a_d15
8		
9		
10	timer7	timer5
11	pr2_mii0_txd2	pr2_mii0_txen
12	pr2_pru1_gpi12	pr2_pru1_gpi10
13	pr2_pru1_gpo12	pr2_pru1_gpo10
14	gpio5_12	gpio5_10
15	Driver off	Driver off
2nd BALL		C14
2nd REG		0x16A4
2nd MODE 0		mcasp1_aclkx
2nd 1		
2nd 2		
2nd 3		
2nd 4		
2nd 5		
2nd 6		
2nd 7		vin6a_fld0
2nd 8		
2nd 9		
2nd 10		i2c3_sda
2nd 11		pr2_mdio_mdclk
2nd 12		pr2_pru1_gpi7
2nd 13		pr2_pru1_gpo7
2nd 14		gpio7_31
2nd 15		Driver off

Table 2.48: P9.41-P9.42

	P9.41	P9.42
GPIO	180	114
BALL	C23	E14
REG	0x16A0	0x16E4
MODE 0	xref_clk3	mcasp1_axr12

continues on next page

Table 2.48 – continued from previous page

	P9.41	P9.42
1	mcasp2_axr11	mcasp7_axr0
2	mcasp1_axr7	
3	mcasp4_ahclkx	spi3_cs1
4	mcasp8_ahclkx	
5		
6	vout2_de	
7	hdq0	vin6a_d11
8	vin4a_de0	
9	clkout3	
10	timer16	timer9
11		pr2_mii0_txd0
12		pr2_pru1_gpi14
13		pr2_pru1_gpo14
14	gpio6_20	gpio4_18
15	Driver off	Driver off
2nd BALL	C1	C2
2nd REG	0x1580	0x159C
2nd MODE 0	vin2a_d6	vin2a_d13
2nd 1		
2nd 2		
2nd 3		rgmii1_txctl
2nd 4	vout2_d17	vout2_d10
2nd 5	emu16	
2nd 6		
2nd 7		
2nd 8	mii1_rxd1	mii1_rxdv
2nd 9	kbd_col3	kbd_row8
2nd 10	eQEP2B_in	eQEP3A_in
2nd 11	pr1_mii_mt1_clk	pr1_mii1_txd0
2nd 12	pr1_pru1_gpi3	pr1_pru1_gpi10
2nd 13	pr1_pru1_gpo3	pr1_pru1_gpo10
2nd 14	gpio4_7	gpio4_14
2nd 15	Driver off	Driver off

TODO

Serial Debug

TODO

USB 3 Type-C

TODO

USB 2 Type-A

TODO

Gigabit Ethernet

TODO

Coaxial

TODO

microSD Memory

TODO

microHDMI

TODO

2.4.8 Cape Board Support

There is a Cape Headers Google Spreadsheet which has a lot of detail regarding various boards and cape add-on boards.

See also https://elinux.org/Beagleboard:BeagleBone_cape_interface_spec

TODO

BeagleBone® Black Cape Compatibility

TODO

See https://elinux.org/Beagleboard:BeagleBone_cape_interface_spec for now.

EEPROM

TODO

Pin Usage Consideration

TODO

GPIO

TODO

I2C

TODO

UART or PRU UART

This section is about both UART pins on the header and PRU UART pins on the headers we will include a chart and later some code

Table 2.49: UART

Function	Pin	ABC Ball	Pinctrl Register	Mode
uart3_txd	P9.21	B22	0x17C4	1
uart3_rxd	P9.22	A26	0x17C0	1
uart5_txd	P9.13	C17	0x1730	4
uart5_rxd	P9.11	B19	0x172C	4
uart5_ctsn	P8.05	AC9	0x178C	2
uart5_rtsn	P8.06	AC3	0x1790	2
uart8_txd	P8.37	A21	0x1738	3
uart8_rxd	P8.38	C18	0x1734	3
uart8_ctsn	P8.31	G16	0x173C	3
uart8_rtsn	P8.32	D17	0x1740	3
uart10_txd	P9.24	F20	0x168C	3
uart10_rxd	P9.26	E21	0x1688	3
uart10_ctsn	P8.03	AB8	0x179C	2
uart10_rtsn	P8.04	AB5	0x17A0	2
uart10_txd	P9.24	F20	0x168C	3
uart10_rxd	P9.26	E21	0x1688	3
uart10_ctsn	P9.20	D2	0x1578	8
uart10_rtsn	P9.19	F4	0x157C	8

Table 2.50: PRU UART

Function	Pin	ABC Ball	Pinctrl Register	Mode
pr2_uart0_txd	P8.31	C8	0x1614	10
pr2_uart0_rxd	P8.33	C6	0x1610	10
pr2_uart0_cts_n	P8.34	D8	0x1608	10
pr2_uart0_rts_n	P8.35	A5	0x160C	10
pr1_uart0_rxd	P8.43	F10	0x15E4	10
pr1_uart0_txd	P8.44	G11	0x15E8	10
pr1_uart0_cts_n	P8.45	F11	0x15DC	10
pr1_uart0_rts_n	P8.46	G10	0x15E0	10

TODO

SPI

TODO

Analog

TODO

PWM, TIMER, eCAP or PRU PWM/eCAP

TODO

eQEP

TODO

CAN

TODO

McASP (audio serial like I2S and AC97)

TODO

MMC

TODO

LCD

TODO

PRU GPIO

TODO

CLKOUT

TODO

Expansion Connector Headers

TODO: discuss header options for working with the expansion connectors per https://git.beagleboard.org/beagleboard/beaglebone-black/-/wikis/System-Reference-Manual#section-7-1

Signal Usage

TODO

Cape Power

TODO

Mechanical

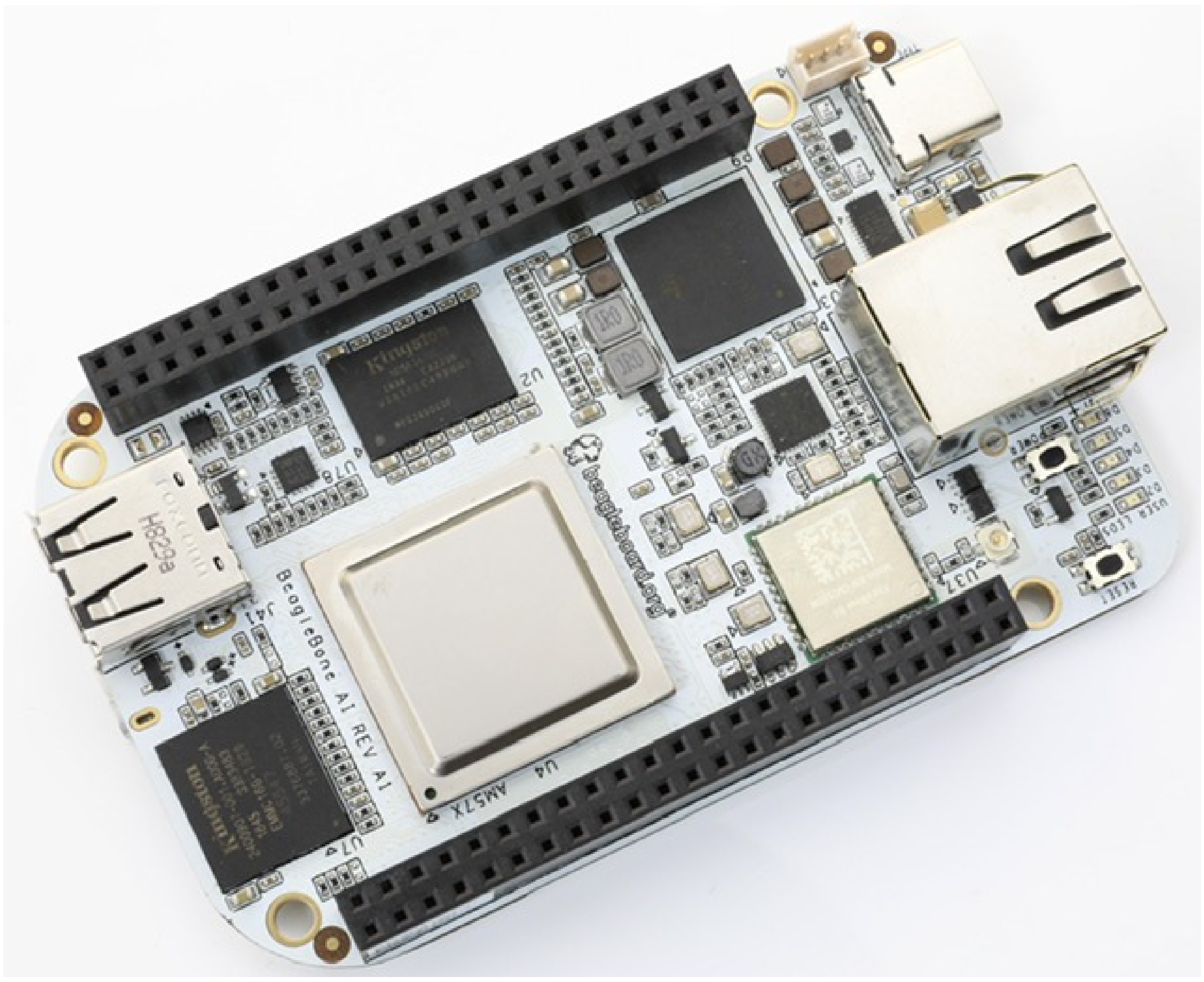
TODO

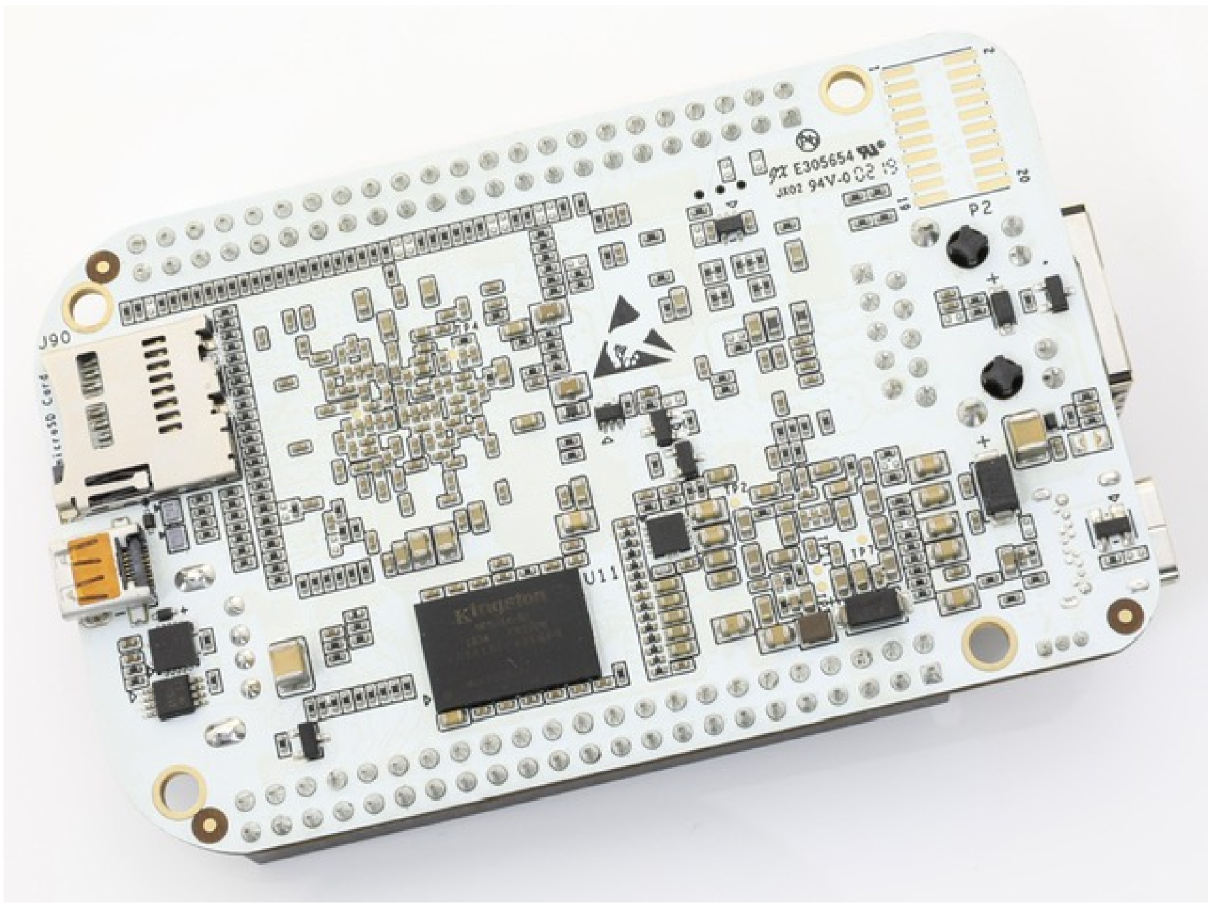
2.4.9 Mechanical Information

- Board Dimensions: 8.9cm x 5.4cm x 1.5cm
- Board Net Weight 48g
- Packaging Dimensions: 13.8cm x 10cm x 4cm
- Gross Weight (including packaging): 110g
- 3D STEP model: https://git.beagleboard.org/beagleboard/beaglebone-ai/-/tree/master/Mechanical

2.4.10 Pictures

BeagleBone AI Back of Board Image





2.4.11 Support Information

TODO: Reference https://docs.beagleboard.org/latest/intro/support/index.html and https://beagleboard.org/resources

Related TI documentation: http://www.ti.com/tool/BEAGLE-3P-BBONE-AI

2.4.12 Terms and Conditions

REGULATORY, COMPLIANCE, AND EXPORT INFORMATION

- Country of origin: PRC
- FCC: 2ATUT-BBONE-AI
- CE: TBD
- CNHTS: 8543909000
- USHTS: 8473301180
- MXHTS: 84733001
- TARIC: 8473302000
- ECCN: 5A992.C
- CCATS: Z1613110/G180570
- RoHS/REACH: TBD
- Volatility: TBD

BeagleBone AI is annotated to comply with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This Class A or B digital apparatus complies with Canadian ICES-003. Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Cet appareil numérique de la classe A ou B est conforme à la norme NMB-003 du Canada. Les changements ou les modifications pas expressément approuvés par la partie responsible de la conformité ont pu vider l'autorité de l'utilisateur pour actionner l'équipement.

WARRANTY AND DISCLAIMERS

The design materials referred to in this document are **NOT SUPPORTED** and **DO NOT** constitute a reference design. Support of the open source developer community is provided through the resources defined at https://docs.beagleboard.org/latest/intro/support/index.html.

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.

This board was designed as an evaluation and development tool. It was not designed with any other application in mind. As such, the design materials that are provided which include schematic, BOM, and PCB files, may or may not be suitable for any other purposes. If used, the design material becomes your responsibility as to whether or not it meets your specific needs or your specific applications and may require changes to meet your requirements.

Additional terms BeagleBoard.org Foundation and logo-licensed manufacturers (together, henceforth identified as "Supplier") provide BeagleBone AI under the following conditions:

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies Supplier from all claims arising from the handling or use of the goods.

Should BeagleBone AI not meet the specifications indicated in the System Reference Manual, BeagleBone AI may be returned within 90 days from the date of delivery to the distributor of purchase for a full refund. THE FOREGOING LIMITED WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

Please read the System Reference Manual and, specifically, the Warnings and Restrictions notice in the Systems Reference Manual prior to handling the product. This notice contains important safety information about temperatures and voltages.

No license is granted under any patent right or other intellectual property right of Supplier covering or relating to any machine, process, or combination in which such Supplier products or services might be or are used. The Supplier currently deals with a variety of customers for products, and therefore our arrangement with the user is not exclusive. The Supplier assume no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.

Warnings and Restrictions

For Feasibility Evaluation Only, in Laboratory/Development Environments BeagleBone AI is not a complete product. It is intended solely for use for preliminary feasibility evaluation in laboratory/development environments by technically qualified electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems and subsystems. It should not be used as all or part of a finished end product.

Your Sole Responsibility and Risk You acknowledge, represent, and agree that:

1. You have unique knowledge concerning Federal, State and local regulatory requirements (including but not limited to Food and Drug Administration regulations, if applicable) which relate to your products and which relate to your use (and/or that of your employees, affiliates, contractors or designees) of BeagleBone AI for evaluation, testing and other purposes.
2. You have full and exclusive responsibility to assure the safety and compliance of your products with all such laws and other applicable regulatory requirements, and also to assure the safety of any activities to be conducted by you and/or your employees, affiliates, contractors or designees, using BeagleBone AI. Further, you are responsible to assure that any interfaces (electronic and/or mechanical) between BeagleBone AI and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard.
3. Since BeagleBone AI is not a completed product, it may not meet all applicable regulatory and safety compliance standards which may normally be associated with similar items. You assume full responsibility to determine and/or assure compliance with any such standards and related certifications as may be applicable. You will employ reasonable safeguards to ensure that your use of BeagleBone AI will not result in any property damage, injury or death, even if BeagleBone AI should fail to perform as described or expected.

Certain Instructions It is important to operate BeagleBone AI within Supplier's recommended specifications and environmental considerations per the user guidelines. Exceeding the specified BeagleBone AI ratings (including but not limited to input and output voltage, current, power, and environmental ranges) may cause property damage, personal injury or death. If there are questions concerning these ratings please contact the Supplier representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may result in unintended and/or inaccurate operation and/or possible permanent damage to BeagleBone AI and/or interface electronics. Please consult the System Reference Manual prior to connecting any load to BeagleBone AI output. If there is uncertainty as to the load specification, please contact the Supplier representative. During normal operation, some circuit components may have case temperatures greater than 60 C as long as the input and output are maintained at a normal ambient operating temperature. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors which can be identified using BeagleBone AI's schematic located at the link in BeagleBone AI's System Reference Manual. When placing measurement probes near these devices during normal operation, please be aware that these devices may be very warm to the touch. As with all electronic evaluation tools, only qualified personnel knowledgeable in electronic measurement and diagnostics normally found in development environments should use BeagleBone AI.

Agreement to Defend, Indemnify and Hold Harmless You agree to defend, indemnify and hold Supplier, its licensors and their representatives harmless from and against any and all claims, damages, losses, expenses, costs and liabilities (collectively, "Claims") arising out of or in connection with any use of BeagleBone AI that is not in accordance with the terms of the agreement. This obligation shall apply whether Claims arise under law of tort or contract or any other legal theory, and even if BeagleBone AI fails to perform as described or expected.

Safety-Critical or Life-Critical Applications If you intend to evaluate the components for possible use in safety critical applications (such as life support) where a failure of the Supplier's product would reasonably be expected to cause severe personal injury or death, such as devices which are classified as FDA Class III or similar classification, then you must specifically notify Supplier of such intent and enter into a separate Assurance and Indemnity Agreement.

2.5 BeagleBone AI-64

BeagleBone® AI-64 brings a complete system for developing artificial intelligence (AI) and machine learning solutions with the convenience and expandability of the BeagleBone® platform and the peripherals on board to get started right away learning and building applications. With locally hosted, ready-to-use, open-source focused tool chains and development environment, a simple web browser, power source and network connection are all that need to be added to start building performance-optimized embedded applications. Industry-leading expansion possibilities are enabled through familiar BeagleBone® cape headers, with hundreds of open-source hardware examples and dozens of readily available embedded expansion options available off-the-shelf.

Contributors

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

2.5.1 Introduction

This document is the *System Reference Manual* for BeagleBone AI-64 and covers its use and design. The board will primarily be referred to in the remainder of this document simply as the board, although it may also be referred to as AI-64 or BeagleBone AI-64 as a reminder.

This design is subject to change without notice as we will work to keep improving the design as the product matures based on feedback and experience. Software updates will be frequent and will be independent of the hardware revisions and as such not result in a change in the revision number.

Make sure you frequently check the BeagleBone AI-64 git repository for the most up to date support documents.

2.5.2 Change History

This section describes the change history of this document and board. Document changes are not always a result of a board change. A board change will always result in a document change.

Document Change History

This table seeks to keep track of major revision cycles in the documentation. Moving forward, we'll seek to align these version numbers across all of the various documentation.

Table 2.51: Table 1: Change History

Rev	Changes	Date	By
0.0.1	AI-64 initial prototype	September 2021	James Anderson
0.0.2	AI-64 final prototype	December 2021	James Anderson
0.0.3	AI-64 initial production release	June 9, 2022	Deepak Khatri and Jason Kridner

Board Changes

Be sure to check the board revision history in the schematic file in the BeagleBone AI-64 git repository . Also check the issues list .

Rev B We are starting with revision B based on this being an update to the BeagleBone Black AI. However, because this board ended up being so different, we've decided to name it BeagleBone AI-64, rather than simply a new revision. This refers to the Seeed release on 21 Dec 2021 of "BeagleBone AI-64_SCH_Rev B_211221". This is the initial production release.

2.5.3 Connecting up your BeagleBone AI-64

This section provides instructions on how to hook up your board. This beagle requires a 5V > 3A power supply to work properly via either USB Type-C power adapter or a barrel jack power adapter.

Recommended adapters:

- 5V @ 3A USB C power supply adapter for SBCs.
- 5V > 3A laptop/mobile adapter with USB-C cable.

All the *Fig 3.1 BeagleBone AI-64 connections ports* we will use in this chapter are shown in the figure below.

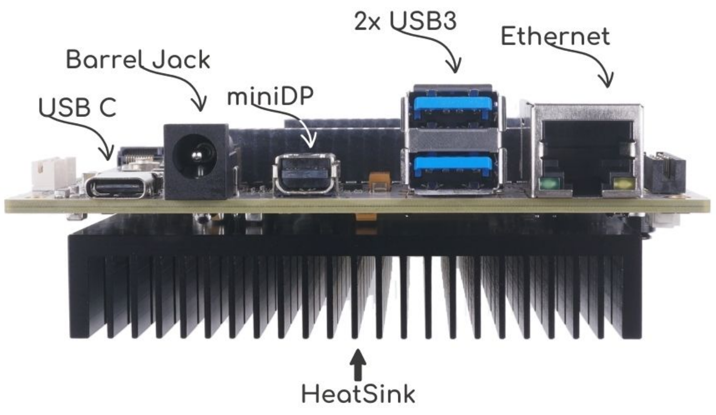


Fig. 2.82: Fig 3.1 BeagleBone AI-64 connections ports

Methods of operation

1. Tethered to a PC
2. Standalone development platform in a PC configuration using external peripherals

What's In the Box

In the box you will find three main items as shown in *Fig: BeagleBone AI-64 box content*.

- BeagleBone AI-64.
- Instruction card.

A USB-C to USB-C cable is not included bot recommended for the tethered scenario and creates an out of box experience where the board can be used immediately with no other equipment needed.

Fig. 2.83: Fig: BeagleBone AI-64 box content

Main Connection Scenarios

This section describes how to connect and power the board and serves as a slightly more detailed description of the Quick Start Guide included in the box.

The board can be configured in several different ways, but we will discuss the two most common scenarios.

- Tethered to a PC via the USB cable
    - `Board is accessed as a storage drive and virtual Ethernet connection.`
- Standalone Desktop
    - `Display`
    - `Keyboard and Mouse`
    - `External 5V > 3A power supply`

Each of these configurations is discussed in general terms in the following sections.

Tethered To A PC

In this configuration, the board is powered by the PC via a single USB cable. The board is accessed either as a USB storage drive or via the browser on the connected PC. You need to use either Firefox or Chrome on the PC, Internet Explorer will not work properly.

At least 5V @ 3A is required to power the board, In most cases the PC may not be able to supply sufficient power for the board unless the connection is made over a Type-C to Type-C cable. You should always use an external 5V > 3A DC power supply connected to the barrel jack if you are unsure that the system can provide the required power or are otherwise using a USB-A to Type-C cable which will always require power from the DC barrel jack.

Connect the Cable to the Board

1. Connect the type C USB cable to the board as shown in *Fig: USB Connection to the Board*. The connector is on the top side of the board near barrel jack.
2. Connect the USB-A end of the cable to your PC or laptop USB port as shown in the usb-a-connect-figure below.
3. The board will power on and the power LED will be on as shown in *Fig: Board Power LED* below.

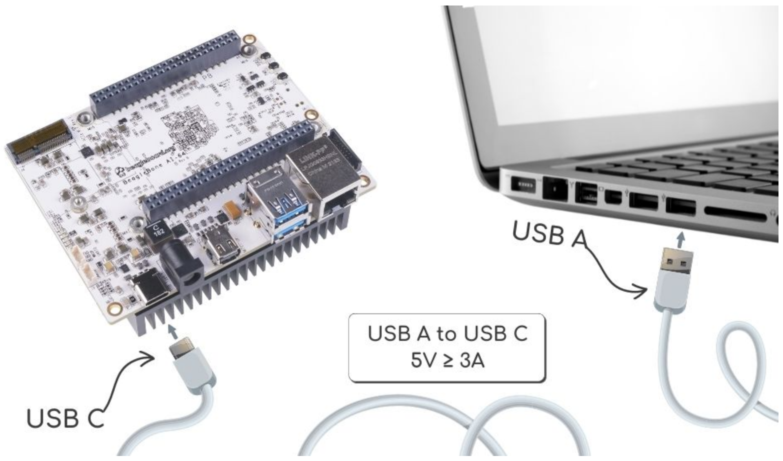


Fig. 2.84: Fig: Tethered Configuration

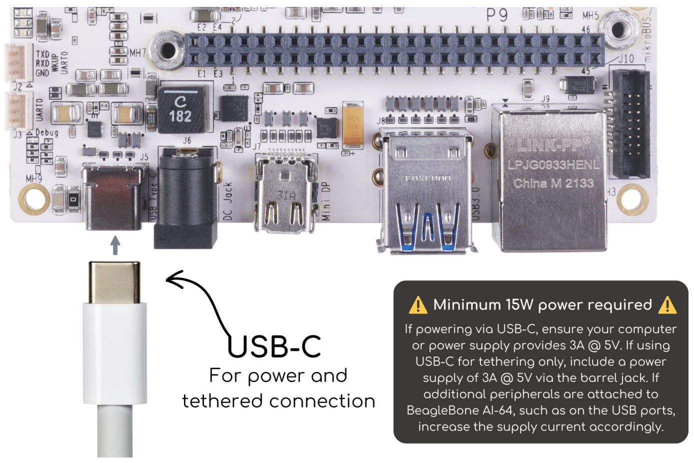

USB-C
For power and
tethered connection
⚠ Minimum 15W power required ⚠
If powering via USB-C, ensure your computer or power supply provides 3A @ 5V. If using USB-C for tethering only, include a power supply of 3A @ 5V via the barrel jack. If additional peripherals are attached to BeagleBone AI-64, such as on the USB ports, increase the supply current accordingly.

Fig. 2.85: Fig: USB Connection to the Board

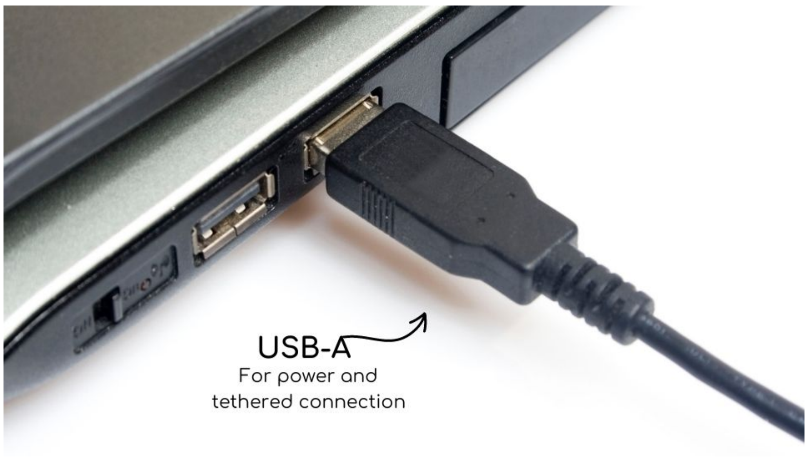


Fig. 2.86: Fig: USB Connection to the PC/Laptop

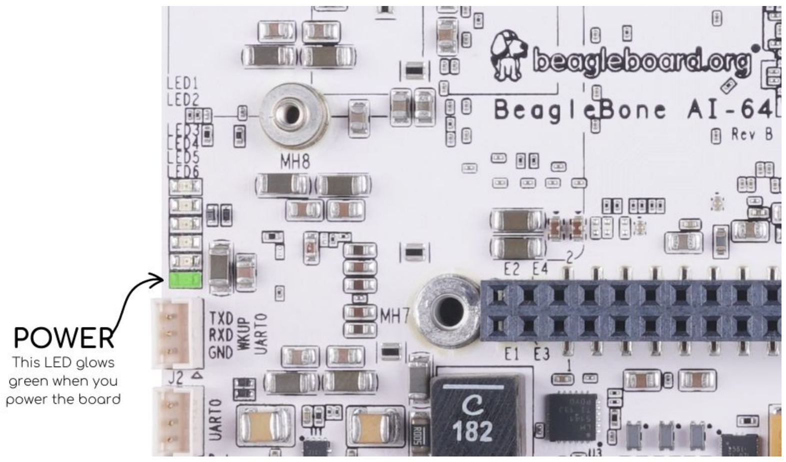


Fig. 2.87: Fig: Board Power LED

4. When the board starts to the booting process started by the process of applying power, the LEDs will come on in sequence as shown in boot-status-figure below. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it begins to boot the Linux kernel.

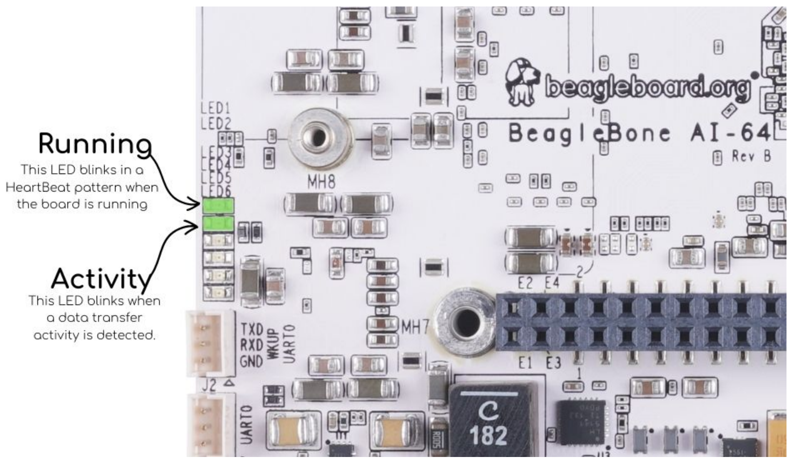

Running
This LED blinks in a HeartBeat pattern when the board is running
Activity
This LED blinks when a data transfer activity is detected.

Fig. 2.88: Fig: Board Boot Status

Accessing the Board as a Storage Drive The board will appear around a USB Storage drive on your PC after thekernel has booted, which will take a round 10 seconds. The kernel on the board needs to boot before the port gets enumerated. Once the board appears as a storage drive, do the following:

1. Open the USB Drive folder.
2. Click on the file named **start.htm**
3. The file will be opened by your browser on the PC and you should get a display showing the Quick Start Guide.
4. Your board is now operational! Follow the instructions on your PC screen.

Standalone w/Display and Keyboard/Mouse

In this configuration, the board works more like a PC, totally free from any connection to a PC as shown in desktop-config-figure. It allows you to create your code to make the board do whatever you need it to do. It will however require certain common PC accessories. These accessories and instructions are described in the following section.

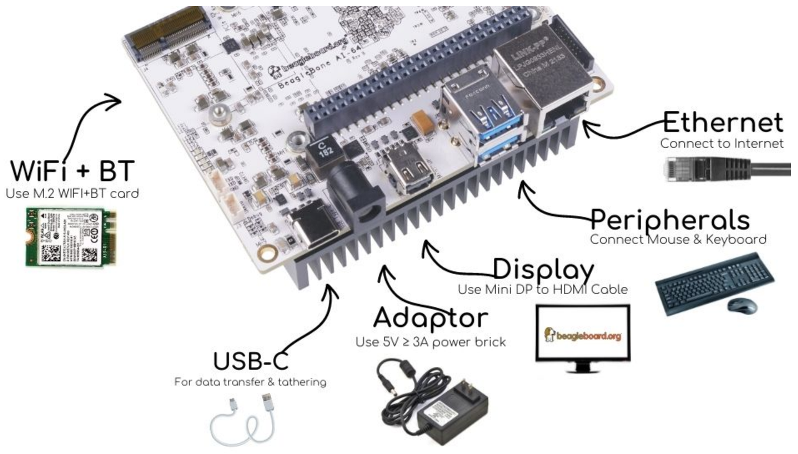


Fig. 2.89: Fig: Desktop Configuration

Ethernet cable and M.2 WiFi + Bluetooth card are optional. They can be used if network access required.

Required Accessories In order to use the board in this configuration, you will need the following accessories:

- 5V > 3A power supply.
- Display Port or HDMI monitor.
- miniDP-DP or active miniDP-HDMI cable (or a recommended **miniDP-DP or active miniDP-HDMI adapter** https://www.amazon.com/dp/B089GF8M87 has been tested and worked beautifully).
- USB wired/wireless keyboard and mouse.
- powered USB HUB (OPTIONAL). The board has only two USB Type-A host ports, so you may need to use a powered USB Hub if you wish to add additional USB devices, such as a USB WiFi adapter.
- M.2 Bluetooth & WiFi module (OPTIONAL). For wireless connections, a USB WiFi adapter or a recommended M.2 WiFi module can provide wireless networking.

Connecting Up the Board

1. Connect the miniDP to DP or active miniDP to HDMI cable from your BeagleBone AI-64 to your monitor.

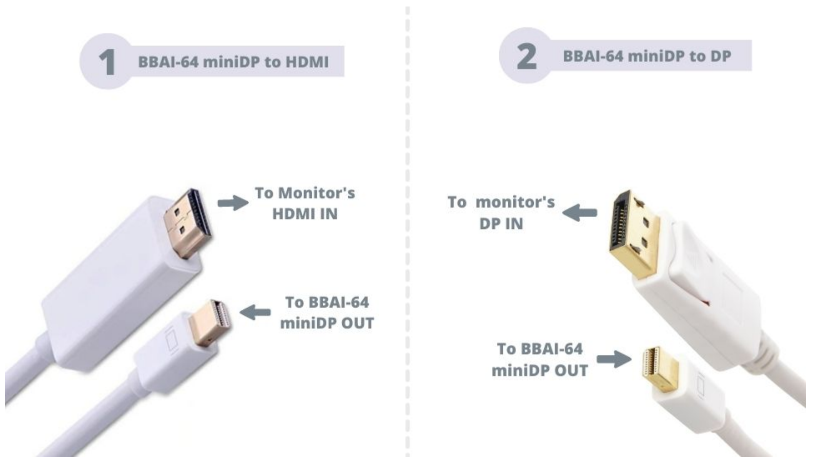


Fig. 2.90: Fig: Connect miniDP-DP or active miniDP-HDMI cable to BeagleBone AI-64

2. If you have an Display Port or HDMI monitor with HDMI-HDMI or DP-DP cable you can use adapters as shown in. *Fig: Display adapters*.

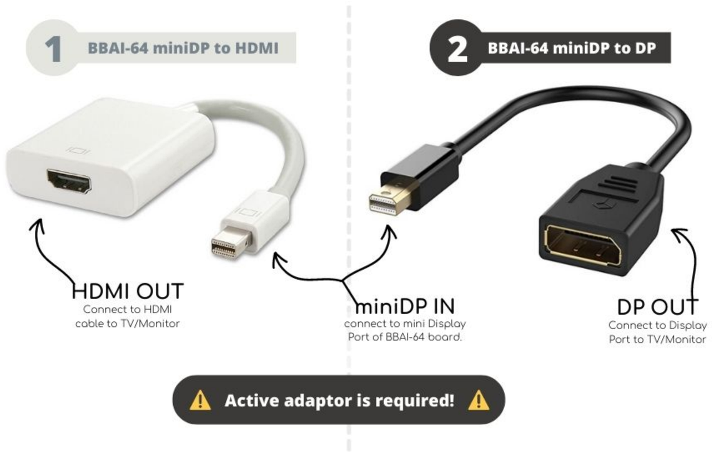

1 BBAI-64 miniDP to HDMI
2 BBAI-64 miniDP to DP
HDMI OUT
Connect to HDMI cable to TV/Monitor
miniDP IN
connect to mini Display Port of BBAI-64 board.
DP OUT
Connect to Display Port to TV/Monitor
Active adaptor is required!

Fig. 2.91: Fig: Display adapters

3. If you have wired/wireless USB keyboard and mouse such as

   seen in *FigKeyboard and Mouse* below, you need to plug the receiver in the USB host port of the board as shown in *FigKeyboard and Mouse*.

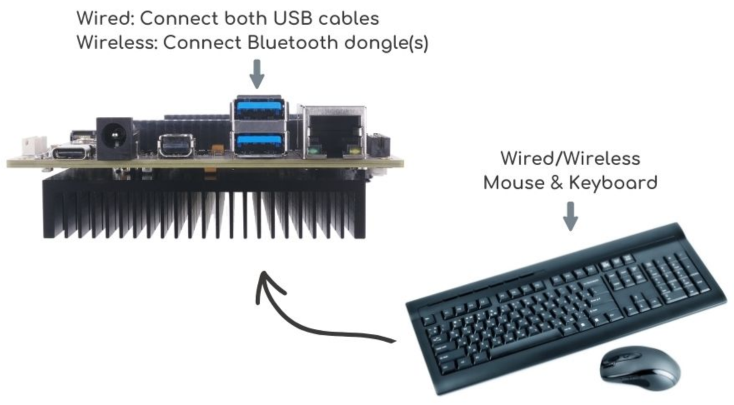

Wired: Connect both USB cables
Wireless: Connect Bluetooth dongle(s)
Wired/Wireless
Mouse & Keyboard

Fig. 2.92: FigKeyboard and Mouse

4. Connect the Ethernet Cable

If you decide you want to connect to your local area network, an Ethernet cable can be used. Connect the Ethernet Cable to the Ethernet port as shown in *Fig: Ethernet Cable Connection*. Any standard 100M Ethernet cable should work.

5. The final step is to plug in the DC power supply to the DC power jack as shown in barrel-jack-figure below.
6. The cable needed to connect to your display is a miniDP-DP or active miniDP-HDMI. Connect the miniDP connector end to the board at this time. The connector is on the top side of the board as shown in miniDP-figure below.

The connector is fairly robust, but we suggest that you not use the cable as a leash for your Beagle. Take proper care not to put too much stress on the connector or cable.

7. Booting the Board

As soon as the power is applied to the board, it will start the booting up process. When the board starts to boot the LEDs will come on. It will take a few seconds for the status LEDs to come on, so be patient. The LEDs will be flashing in an erratic manner as it boots the Linux kernel.

While the four user LEDS can be over written and used as desired, they do have specific meanings in the image that is shipped with the board once the Linux kernel has booted.

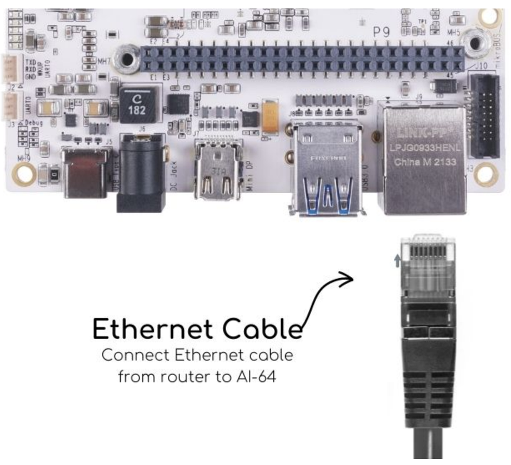


Fig. 2.93: Fig: Ethernet Cable Connection

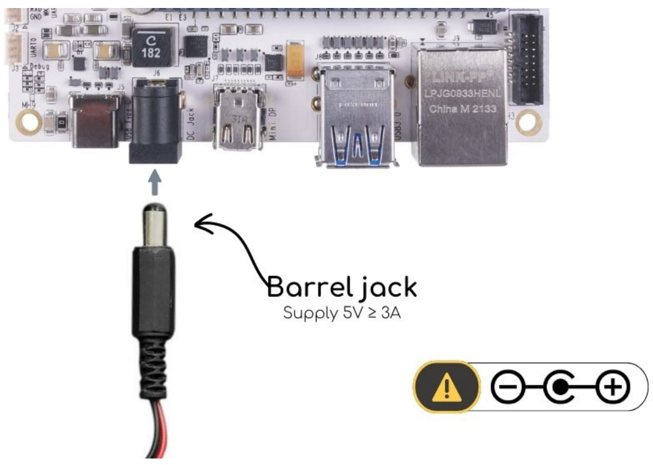


Fig. 2.94: Fig: External DC Power

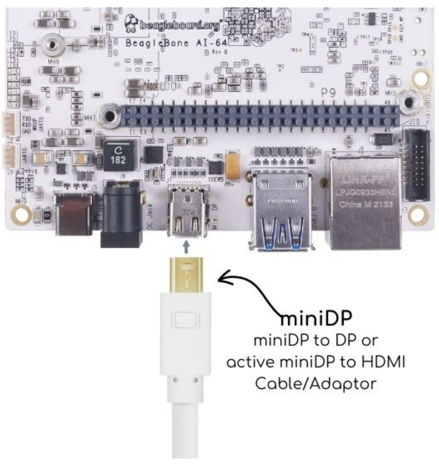


Fig. 2.95: Fig: Connect miniDP to DP or active miniDP to HDMI Cable to the Board

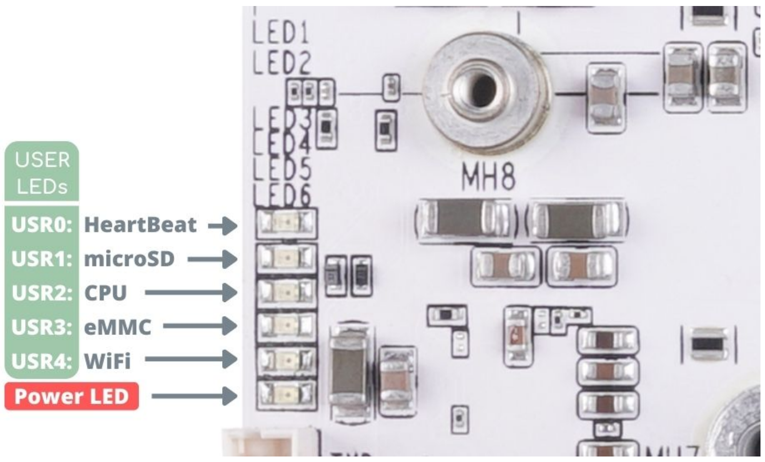


Fig. 2.96: Fig: BeagleBone AI-64 LEDs

- **USR0** is the heartbeat indicator from the Linux kernel.
- **USR1** turns on when the microSD card is being accessed
- **USR2** is an activity indicator. It turns on when the kernel is not in the idle loop.
- **USR3** turns on when the onboard eMMC is being accessed.
- **USR4** is an activity indicator for WiFi.

8. A Booted System

   a. The board will have a mouse pointer appear on the screen as it enters the Linux boot step. You may have to move the physical mouse to get the mouse pointer to appear. The system can come up in the suspend mode with the monitor in a sleep mode.

   b. After a minute or two a login screen will appear. You do not have to do anything at this point.

   c. After a minute or two the desktop will appear. It should be similar to the one shown in *Fig: BeagleBone XFCE Desktop Screen*. HOWEVER, it will change from one release to the next, so do not expect your system to look exactly like the one in the figure, but it will be very similar.

   d. And at this point you are ready to go! *Fig: BeagleBone XFCE Desktop Screen* shows the desktop after booting.

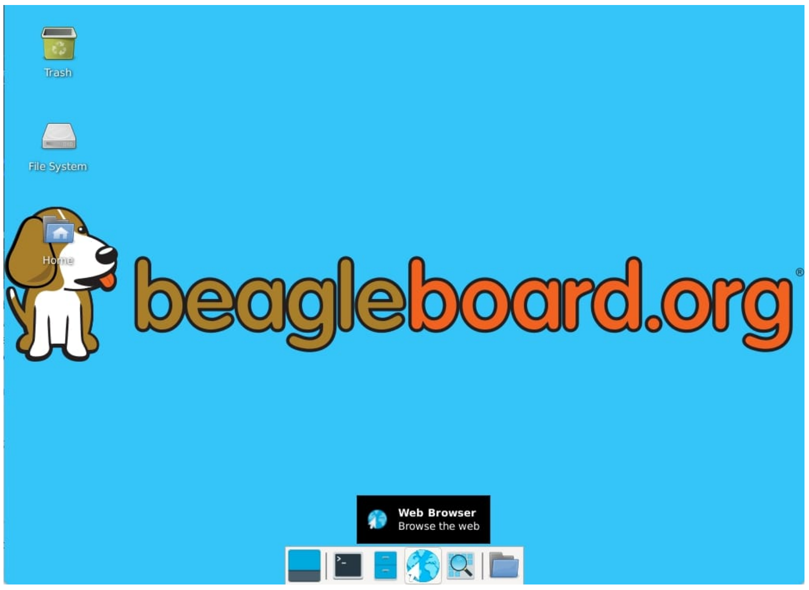


Fig. 2.97: Fig: BeagleBone XFCE Desktop Screen

2.5.4 BeagleBone AI-64 Overview

BeagleBone AI-64 is the latest addition to BeagleBoard.org family and like its predecessors, is designed to address the open-source Community, early adopters, and anyone interested in a low cost 64-bit Dual Arm® Cortex®-A72 processor based Single Board Computer (SBC).

It has been equipped with a minimum set of features to allow the user to experience the power of the processor and is not intended as a full development platform as many of the features and interfaces supplied by the processor are not accessible from BeagleBone AI-64 via onboard support of some interfaces. It is not a complete product designed to do any particular function. It is a foundation for experimentation and learning how to program the processor and to access the peripherals by the creation of your own software and hardware.

It also offers access to many of the interfaces and allows for the use of add-on boards called capes, to add many different combinations of features. A user may also develop their own board or add their own circuitry.

BeagleBone AI-64 is manufactured and warranted by partners listed at https://beagleboard.org/logo for the benefit of the community and its supporters including the current BeagleBoard.org Foundation board members

- Jason Kridner, principal of JK Embedded Consulting an independent contractor and architect for new Beagle designs.
- Drew Fustini, independent Linux developer
- Robert Nelson, applications engineer at Digi-Key
- Mark Yoder, professor at Rose-Hulman Institute of Technology
- Kathy Giori, product engineer at ZEDEDA

See bbb.io/about

BeagleBone AI-64 has been designed by Seeed Studio (Seeed Development Limited) under guidance from BeagleBoard.org Foundation.

BeagleBone Compatibility

The board is intended to provide functionality well beyond BeagleBone Black or BeagleBone AI, while still providing compatibility with BeagleBone Black's expansion headers as much as possible. There are several significant differences between the three designs.

Table 2.52: Table: BeagleBone Compatibility

Feature	AI-64	AI	Black
SoC	TDA4VM	AM5729	AM3358
Arm CPU	Cortex-A72 (64-bit)	Cortex-A15 (32-bit)	Cortex-A8 (32-bit)
Arm cores/MHz	2x 2GHz	2x 1.5GHz	1x 1GHz
RAM	4GB	1GB	512MB
eMMC flash	16GB	16GB	4GB
Size	4" x 3.1"	3.4" x 2.1"	.4" x 2.1"
Display	miniDP + DSI	microHDMI	microHDMI
USB host (Type-A)	2x 5Gbps	1x 480Mbps	1x 480Mbps
USB dual-role	Type-C 5Gbps	Type-C 5Gbps	mini-AB 480Mbps
Ethernet	10/100/1000M	10/100/1000M	10/100M
M.2	E-key	-	-
WiFi/ Bluetooth	-	AzureWave AW&#8209;CM256SM	-

Note: TODO: add cape compatibility details

BeagleBone AI-64 Features and Specification

This section covers the specifications and features of the board and provides a high level description of the major components and interfaces that make up the board.

Table 2.53: Table: BeagleBone AI-64 Features and Specification

	Feature
Processor	Texas Instruments TDA4VM
Graphics Engine	PowerVR® Series8XE GE8430
SDRAM Memory	LPDDR4 3.2GHz (4GB) Kingston Q3222PM1WDGTK-U
Onboard Flash	eMMC (16GB) Kingston EMMC16G-TB29-PZ90
PMIC	TPS65941213 and TPS65941111 PMICs regulator and one additional LDO.
Debug Support	**2x 3 pin 3.3V TTL header** 1. WKUP_UART0: Wake-up domain serial port 2. UART0: Main domain serial port 10-pin JTAG TAG-CONNECT footprint
Power Source	USB C or DC Jack (5V, >3A)
PCB	4” x 3.1”
Indicators	1-Power, 5-User Controllable LEDs
USB-3.0 Client Port	Access to USB0, SuperSpeed, dual-role mode via USB-C (no power output)
USB-3.0 Host Port	TUSB8041 4-port SuperSpeed hub on USB1, 2xType A Socket, up-to 2.8A total, depending on power input
Ethernet	Gigabit, RJ45, link indicator, speed indicator
SD/MMC Connector	microSD , 1.8/3.3V
User Input	1. Reset Button 2. Boot Button 3. Power Button
Video Out	miniDP
Audio	via miniDP (stereo)
Weight	192gm (with heatsink)
Power	Refer to main-board-power section

Board Component Locations

This section describes the key components on the board. It provides information on their location and function. Familiarize yourself with the various components on the board.

Board components

Fig: BeagleBone AI-64 board components below shows the locations of the connectors, LEDs, and switches on the PCB layout of the board.

- **DC Power** is the main DC input that accepts 5V power.
- **Power Button** alerts the processor to initiate the power down sequence and is used to power down the board.
- **GigaBit Ethernet** is the connection to the LAN.
- **Serial Debug ports** WKUP_UART0 for early boot from the management MCU and UART0 is for the main processor.
- **USB Client** is a USB-C connection to a PC that can also power the board.
- **BOOT switch** can be used to force a boot from the microSD card if the power is cycled on the board, removing power and reapplying the power to the board.
- There are five green **LEDs** that can be used by the user.
- **Reset Button** allows the user to reset the processor.
- **microSD** slot is where a microSD card can be installed.
- **miniDP** connector is where the display is connected to.

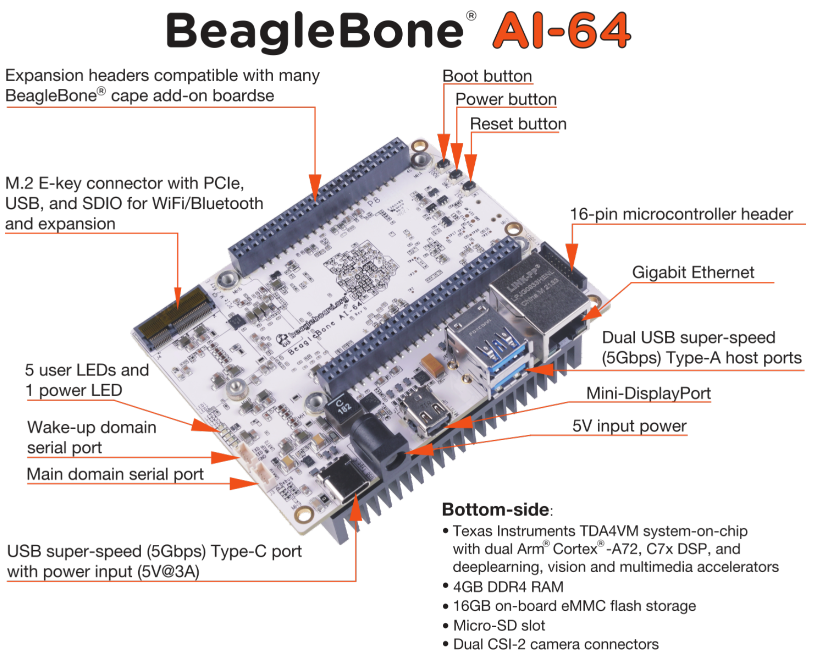

BeagleBone® AI-64
Expansion headers compatible with many BeagleBone® cape add-on boardse
Boot button
Power button
Reset button
M.2 E-key connector with PCIe, USB, and SDIO for WiFi/Bluetooth and expansion
16-pin microcontroller header
Gigabit Ethernet
Dual USB super-speed (5Gbps) Type-A host ports
5 user LEDs and 1 power LED
Mini-DisplayPort
5V input power
Wake-up domain serial port
Main domain serial port
Bottom-side:
• Texas Instruments TDA4VM system-on-chip with dual Arm® Cortex®-A72, C7x DSP, and deeplearning, vision and multimedia accelerators
• 4GB DDR4 RAM
• 16GB on-board eMMC flash storage
• Micro-SD slot
• Dual CSI-2 camera connectors
USB super-speed (5Gbps) Type-C port with power input (5V@3A)

Fig. 2.98: Fig: BeagleBone AI-64 board components

- **USB Host** can be connected different USB interfaces such as Wi-Fi, Bluetooth, Keyboard, etc.

On bottom side we have,

- **TI TDA4VM** processor.
- **4GB LPDDR4** Dual Data Rate RAM memory.
- **Ethernet PHY** physical interface to the network.
- **eMMC** onboard MMC chip that holds up to 16GB of data.

2.5.5 BeagleBone AI-64 High Level Specification

Fig: BeagleBone AI-64 Key Components below shows the high level block diagram of BeagleBone AI-64 board surrounding TDA4VM SoC.

Processor

BeagleBone AI-64 uses TI J721E-family TDA4VM system-on-chip (SoC) which is part of the K3 Multicore SoC architecture platform and it is targeted for the reliability and low-latency needs of the automotive market provide for a great general purpose platform suitable for industrial automation, mobile robotics, building automation and numerous hobby projects.

The SoC designed as a low power, high performance and highly integrated device architecture, adding significant enhancement on processing power, graphics capability, video and imaging processing, virtualization and coherent memory support. In addition, these SoCs support state of the art security and functional safety features. For the remaining of this section device, SoC, and processor will be used interchangeably.

Some of the main distinguished characteristics of the device are:

- 64-bit architecture with virtualization and coherent memory support, which leverages full processing capability of 64-bit Arm® Cortex®-A72
- Fully programmable industrial communication subsystems to enable future-proof designs for customers that need to adopt the new Gigabit Time-sensitive Networks (TSN) standards, but still need full support on legacy protocols and continuous system optimization over the product deployment

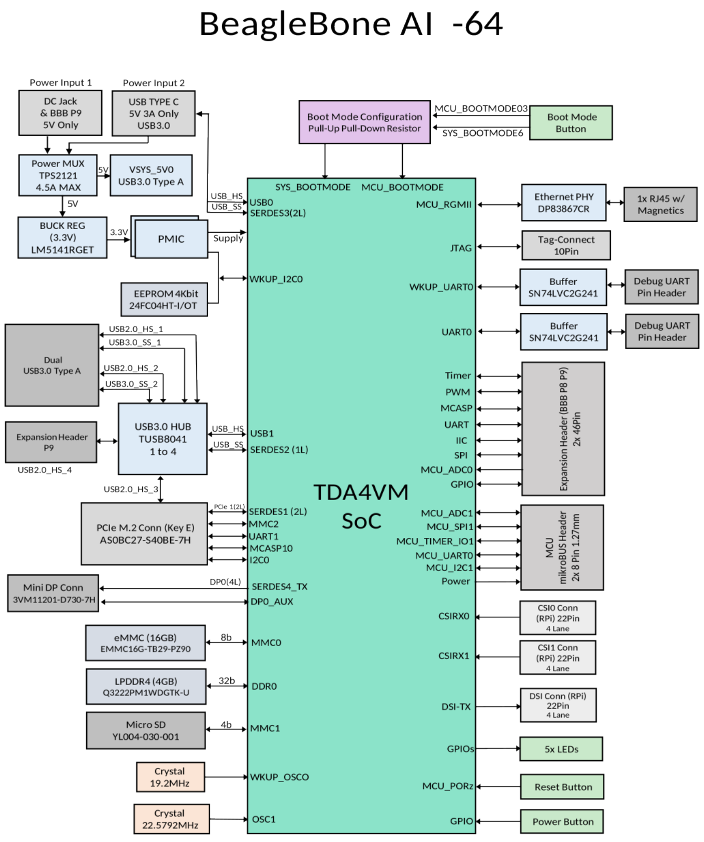


Fig. 2.99: Fig: BeagleBone AI-64 Key Components

- Integration of vision hardware processing accelerators to facilitate extensive processing requirements in low power budget for automotive ADAS and machine vision applications
- Integration of a general-purpose microcontroller unit (MCU) with a dual Arm® Cortex®-R5F MCU subsystem, available for general purpose use as two cores or in lockstep, intended to help customers achieve functional safety goals for their end products
- Integration of a next-generation fixed and floating-point C71x Digital Signal Processor (DSP) that significantly boosts power over a broad range of general signal processing tasks for both general applications and automotive functions which also incorporates advanced techniques to improve control code efficiency and ease of programming such as branch prediction, protected pipeline, precise exception and virtual memory management
- Tightly coupled Matrix Multiplication Accelerator (MMA) that extends the C71x DSP architecture's scalar and vector facilities enabling deep learning and enhance vision, analytics and wide range of general applications. The achieved total TOPS (Tera Operations Per Second) performance significantly differentiates the device for single board computer in machine vision and deep learning applications
- Key display features including flexibility to interface with different panel types (eDP, DSI, DPI) with multi-layer hardware composition
- Integration of hardware features that help applications to achieve functional safety mechanisms
- Robust security architecture with sandboxed DMSC controller managing all secure configurations with high performance client-server messaging scheme between secure DMSC and all cores
- Simplified solution for power supply management, enabling lower cost system solution (on-die bias LDOs and power good comparators for minimal power sequencing requirements consistent with low cost supply design)

The device is composed of the following main subsystems, across different domains of the SoC, among others:

- One dual-core 64-bit Arm Cortex-A72 microprocessor subsystem at up to 2.0 GHz and up to 24K DMIPS (Dhrystone Million Instructions per Second)
- Up to three Microcontroller Units (MCU), based on dual-core Arm Cortex-R5F processor running at up to 1.0 GHz, up to 12K DMIPS
- Up to two TMS320C66x DSP CorePac modules running at up to 1.35 GHz, up to 40 GFLOPS
- One C71x floating point, vector DSP running at up to 1.0 GHz, up to 80 GFLOPS
- One deep-learning MMA, up to 8 TOPS (8b) at 1.0 GHz
- Up to two gigabit dual-core Programmable Real-Time Unit and Industrial Communication Subsystems (PRU_ICSSG)
- Two Navigator Subsystems (NAVSS) for data movement and control
- One multi-pipeline Display Subsystem (DSS) with one MIPI® Display Serial Interface Controller (DSI) and shared MIPI D-PHY Transmitter (DPHY_TX), one Embedded DisplayPort Transmitter (EDP) with shared Serializer/Deserializer (SERDES), and two MIPI Display Pixel Interface (DPI) ports
- Two Camera Streaming Interface Receivers (CSI_RX_IF) with dedicated MIPI D-PHYs (DPHY_RX)
- One Camera Streaming Interface Transmitter (CSI_TX_IF) with MIPI D-PHY Transmitter (DPHY_TX) shared with DSI
- One Vision Processing Accelerator (VPAC) with image signal processor
- One Depth and Motion Processing Accelerator (DMPAC)
- One dual-core multi-standard HD Video Decoder (DECODER)
- One dual-core multi-standard HD Video Encoder (ENCODER)
- One Graphics Processing Unit (GPU)
- One Device Management and Security Controller (DMSC)

The device provides a rich set of peripherals such as:

- General connectivity peripherals, including:
  - Two 12-bit general purpose Analog-to-Digital Converters (ADC)
  - Ten Inter-Integrated Circuit (I2C) interfaces
  - Three Improved Inter-Integrated Circuit (I3C) controllers
  - Eleven master/slave Multichannel Serial Peripheral Interfaces (MCSPI)
  - Twelve configurable Universal Asynchronous Receiver/Transmitter (UART) interfaces
  - Ten General-Purpose Input/Output (GPIO) modules
- High-speed interfaces, including:
  - Two Gigabit Ethernet Switch (CPSW) modules
  - Two Dual-Role-Device (DRD) Universal Serial Bus Subsystems (USBSS) with integrated PHY
  - Four Peripheral Component Interconnect express (PCIe) Gen3 subsystems
- Flash memory interfaces, including:
  - One Octal SPI (OSPI) interface and one Quad SPI (QSPI) or one QSPI and one HyperBus^TM^
  - One General Purpose Memory Controller (GPMC) with Error Location Module (ELM) and 8- or 16-bit-wide data bus width (supports parallel NOR or NAND FLASH devices)
  - Three Multimedia Card/Secure Digital (MMCSD) controllers
  - One Universal Flash Storage (UFS) interface
- Industrial and control interfaces, including:
  - Sixteen Controller Area Network (MCAN) interfaces with flexible data rate support
  - Three Enhanced Capture (ECAP) modules
  - Six Enhanced Pulse-Width Modulation (EPWM) subsystems
  - Three Enhanced Quadrature Encoder Pulse (EQEP) modules
- Audio peripherals, including:
  - One Audio Tracking Logic (ATL)
  - Twelve Multichannel Audio Serial Port (MCASP) modules supporting up to 16 channels with independent TX/RX clock/sync domain
- One Video Processing Front End (VPFE) interface module

The device also integrates:

- Power distribution, reset controls and clock management components
- Power-management techniques for device power consumption minimization:
  - Adaptive Voltage Scaling (AVS)
  - Dynamic Frequency Scaling (DFS)
  - Gated clocks
  - Multiple voltage domains

    - Independently controlled power domains for major modules
    - Voltage and Temperature Management (VTM) module
    - Power-on Reset Generators (PRG)
    - Power Sleep Controllers (PSC)

- Optimized interconnect (CBASS) architecture to enable latency-critical real time network and IO applications
- Control modules (CTRL_MMRs) mainly associated with device top-level configurations such as:

    - IO Pad and pin multiplexing configuration
    - PLL control and associated High-Speed Dividers (HSDIV)
    - Clock selection
    - Analog function controls

- Multicore Shared Memory Controller (MSMC)
- DDR Subsystem (DDRSS) with Error Correcting Code (ECC), supporting LPDDR4
- 1KB RAM with ECC support for C71x boot vectors
- 2KB RAM with ECC support for A72 and R5F boot vectors
- 512KB On-Chip SRAM protected by ECC
- One Global Time Counter (GTC) module
- Thirty 32-bit counter timers with compare and capture modes
- Debug and trace capabilities

The device includes different modules for functional safety requirements support:

- MCU island with dual lock step Arm Cortex-R5F
- Safety enabled interconnect with implemented features to help with Freedom From Interference (FFI)
- Twelve Real Time Interrupt (RTI) modules with Windowed Watchdog Timer (WWDT) functionality to monitor processor cores
- Sixteen Dual-Clock Comparators (DCC) to monitor clocking sources during run-time
- Three Error Signaling Modules (ESM) to enable error monitoring
- Temperature monitoring sensors
- ECC on all critical memories
- Dedicated hardware Memory Cyclic Redundancy Check (MCRC) blocks

The device supports the following main security functionalities among others:

- Secure Boot Management
- Public Key Accelerator (PKA) for large vector math operation
- Cryptographic acceleration (AES, 3DES, MD5, SHA1, SHA2-224, 256, 512 operation)
- Trusted Execution Environment (TEE)
- Secure storage support
- On-the-fly encryption and authentication support for OSPI interface

The device is partitioned into three functional domains as shown in *Fig: Device Top-level Block Diagram*, each containing specific processing cores and peripherals:

- Wake-up (WKUP) domain
- Microcontroller (MCU) domain with one of the dual Cortex-R5 cluster

- MAIN domain

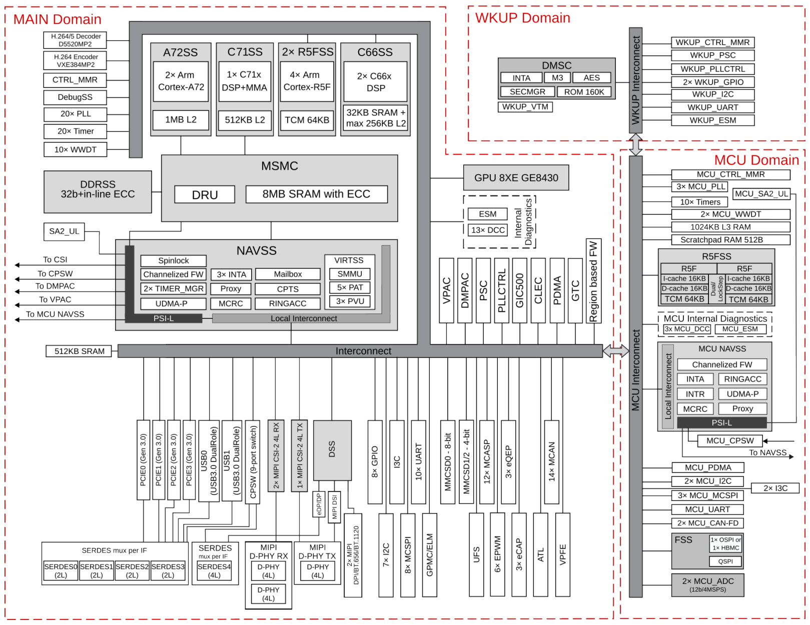


Fig. 2.100: Fig: Device Top-level Block Diagram

Memory

Described in the following sections are the three memory devices found on the board.

4GB LPDDR4 A single (1024M x 16bits x 2channels) LPDDR4 4Gb memory device is used. The memory used is:

- Kingston Q3222PM1WDGTK-U

4Kb EEPROM A single 4Kb EEPROM (24FC04HT-I/OT) is provided on I2C0 that holds the board information. This information includes board name, serial number, and revision information.

16GB Embedded MMC A single 16GB embedded MMC (eMMC) device is on the board. The device connects to the MMC1 port of the processor, allowing for 8bit wide access. Default boot mode for the board will be MMC1 with an option to change it to MMC0, the SD card slot, for booting from the SD card as a result of removing and reapplying the power to the board. Simply pressing the reset button will not change the boot mode. MMC0 cannot be used in 8Bit mode because the lower data pins are located on the pins used by the Ethernet port. This does not interfere with SD card operation but it does make it unsuitable for use as an eMMC port if the 8 bit feature is needed.

MicroSD Connector The board is equipped with a single microSD connector to act as the secondary boot source for the board and, if selected as such, can be the primary boot source. The connector will support larger capacity microSD cards. The microSD card is not provided with the board. Booting from MMC0 will be used to flash the eMMC in the production environment or can be used by the user to update the SW as needed.

Boot Modes As mentioned earlier, there are two boot modes:

- **eMMC Boot:** This is the default boot mode and will allow for the fastest boot time and will enable the board to boot out of the box using the pre-flashed OS image without having to purchase an microSD card or an microSD card writer.

- **SD Boot:** This mode will boot from the microSD slot. This mode can be used to override what is on the eMMC device and can be used to program the eMMC when used in the manufacturing process or for field updates.

Note: TODO: This section needs more work and references to greater detail. Other boot modes are possible. Software to support USB and serial boot modes is not provided by beagleboard.org._Please contact TI for support of this feature.

A switch is provided to allow switching between the modes.

- Holding the boot switch down during a removal and reapplication of power without a microSD card inserted will force the boot source to be the USB port and if nothing is detected on the USB client port, it will go to the serial port for download.
- Without holding the switch, the board will boot try to boot from the eMMC. If it is empty, then it will try booting from the microSD slot, followed by the serial port, and then the USB port.
- If you hold the boot switch down during the removal and reapplication of power to the board, and you have a microSD card inserted with a bootable image, the board will boot from the microSD card.

Note: Pressing the RESET button on the board will NOT result in a change of the boot mode. You MUST remove power and reapply power to change the boot mode. The boot pins are sampled during power on reset from the PMIC to the processor.The reset button on the board is a warm reset only and will not force a boot mode change.

Power Management

The *TPS65941213 and TPS65941111* power management device is used along with a separate LDO to provide power to the system.

PC USB Interface

The board has a USB type-C connector that connects to USB0 port of the processor.

Serial Debug Ports

Two serial debug ports are provided on board via 3pin micro headers,

1. WKUP_UART0: Wake-up domain serial port
2. UART0: Main domain serial port

In order to use the interfaces a 3pin micro to 6pin dupont adaptor header is required with a 6 pin USB to TTL adapter. The header is compatible with the one provided by FTDI and can be purchased for about $$12 to $$20 from various sources. Signals supported are TX and RX. None of the handshake signals are supported.

USB1 Host Port

On the board is a single USB Type A female connector with full LS/FS/HS Host support that connects to USB1 on the processor. The port can provide power on/off control and up to 1.5A of current at 5V. Under USB power, the board will not be able to supply the full 1.5A, but should be sufficient to supply enough current for a lower power USB device supplying power between 50 to 100mA.

Power Sources

The board can be powered from two different sources:

- A 5V > 3A power supply plugged into the barrel jack.
- A wall adaptor with 5V > 3A output power.

The power supply is not provided with the board but can be easily obtained from numerous sources. A 5V > 3A supply is mandatory to have with the board, but if there is a cape plugged into the board or you have a power hungry device or hub plugged into the host port, then more current may needed from the DC supply.

Reset Button

When pressed and released, causes a reset of the board.

Power Button

This button takes advantage of the input to the PMIC for power down features.

Indicators

There are a total of six green LEDs on the board.

- One green power LED indicates that power is applied and the power management IC is up.
- Five blue LEDs that can be controlled via the SW by setting GPIO pins.

2.5.6 Connectors

Expansion Connectors

The expansion interface on the board is comprised of two headers P8 (46 pin) & P9 (50 pin). All signals on the expansion headers are **3.3V** unless otherwise indicated.

Note: Do not connect 5V logic level signals to these pins or the board will be damaged.

Note: DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

Connector P8 The following tables show the pinout of the **P8** expansion header. The SW is responsible for setting the default function of each pin. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The column heading is the pin number on the expansion header.

The **GPIO** row is the expected gpio identifier number in the Linux kernel.

Each row includes the gpiochipX and pinY in the format of *X Y*. You can use these values to directly control the GPIO pins with the commands shown below.

```
# to set the GPIO pin state to HIGH
debian@BeagleBone:~$ gpioset X Y=1

# to set the GPIO pin state to LOW
debian@BeagleBone:~$ gpioset X Y=0

For Example:

+----------+----------+
| Pin      | P8.03    |
+==========+==========+
| GPIO     | 1 20     |
+----------+----------+

Use the commands below for controlling this pin (P8.03) where X = 1 and Y =␣
↪20

# to set the GPIO pin state to HIGH
debian@BeagleBone:~$ gpioset 1 20=1

# to set the GPIO pin state to LOW
debian@BeagleBone:~$ gpioset 1 20=0
```

The **BALL** row is the pin number on the processor.

The **REG** row is the offset of the control register for the processor pin.

The **MODE #** rows are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

NOTES:

DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

P8.E1-P8.E4

E1	E2	E3	E4
USB1 DP	USB1 DN	VSYS_5V0	GND

P8.01-P8.02

P8.01	P8.02
GND	GND

P8.03-P8.05

Pin	P8.03	P8.04	P8.05
GPIO	1 20	1 48	1 33
BALL	AH21	AC29	AH25
REG	0x00011C054	0x00011C0C4	0x00011C088
Page	46	30	50
MODE 0	PRG1_PRU0_GPO19	PRG0_PRU0_GPO5	PRG1_PRU1_GPO12
1	PRG1_PRU0_GPI19	PRG0_PRU0_GPI5	PRG1_PRU1_GPI12
2	PRG1_IEP0_EDC_SYNC_OUT0	~	PRG1_RGMII2_TD1
3	PRG1_PWM0_TZ_OUT	PRG0_PWM3_B2	PRG1_PWM1_A0
4	~	~	RGMII2_TD1
5	RMII5_TXD0	RMII3_TXD0	~
6	MCAN6_TX	~	MCAN7_TX
7	GPIO0_20	GPIO0_48	GPIO0_33
8	~	GPMC0_AD0	RGMII8_TD1
9	~	~	~
10	VOUT0_EXTPCLKIN	~	VOUT0_DATA12
11	VPFE0_PCLK	~	~
12	MCASP4_AFSX	MCASP0_AXR3	MCASP9_AFSX
13	~	~	~
14	~	~	~
Bootstrap	~	BOOTMODE2	~

P8.06-P8.09

Pin	P8.06	P8.07	P8.08	P8.09
GPIO	1 34	1 15	1 14	1 17
BALL	AG25	AD24	AG24	AE24
REG	0x00011C08C	0x00011C03C	0x00011C038	0x00011C044
Page	51	44	44	45
MODE 0	PRG1_PRU1_GPO13	PRG1_PRU0_GPO14	PRG1_PRU0_GPO13	PRG1_PRU0_GPO16
1	PRG1_PRU1_GPI13	PRG1_PRU0_GPI14	PRG1_PRU0_GPI13	PRG1_PRU0_GPI16
2	PRG1_RGMII2_TD2	PRG1_RGMII1_TD3	PRG1_RGMII1_TD2	PRG1_RGMII1_TXC
3	PRG1_PWM1_B0	PRG1_PWM0_A1	PRG1_PWM0_B0	PRG1_PWM0_A2
4	RGMII2_TD2	RGMII1_TD3	RGMII1_TD2	RGMII1_TXC
5	~	~	~	~
6	MCAN7_RX	MCAN5_RX	MCAN5_TX	MCAN6_RX
7	GPIO0_34	GPIO0_15	GPIO0_14	GPIO0_17
8	RGMII8_TD2	~	~	~
9	~	RGMII7_TD3	RGMII7_TD2	RGMII7_TXC
10	VOUT0_DATA13	VOUT0_DATA19	VOUT0_DATA18	VOUT0_DATA21
11	VPFE0_DATA8	VPFE0_DATA3	VPFE0_DATA2	VPFE0_DATA5
12	MCASP9_AXR0	MCASP7_AXR1	MCASP7_AXR0	MCASP7_AXR3
13	MCASP4_ACLKR	~	~	MCASP7_AFSR
14	~	~	~	~
Bootstrap	~	~	~	~

P8.10-P8.13

Pin	P8.10	P8.11	P8.12	P8.13
GPIO	1 16	1 60	1 59	1 89
BALL	AC24	AB24	AH28	V27
REG	0x00011C040	0x00011C0F4	0x00011C0F0	0x00011C168
Page	44	33	33	56
MODE 0	PRG1_PRU0_GPO15	PRG0_PRU0_GPO17	PRG0_PRU0_GPO16	RGMII5_TD1
1	PRG1_PRU0_GPI15	PRG0_PRU0_GPI17	PRG0_PRU0_GPI16	RMII7_TXD1
2	PRG1_RGMII1_TX_CTL	PRG0_IEP0_EDC_SYNC_OUT1	PRG0_RGMII1_TXC	I2C3_SCL
3	PRG1_PWM0_B1	PRG0_PWM0_B2	PRG0_PWM0_A2	~
4	RGMII1_TX_CTL	PRG0_ECAP0_SYNC_OUT	RGMII3_TXC	VOUT1_DATA4
5	~	~	~	TRC_DATA2
6	MCAN6_TX	~	~	EHRPWM0_B
7	GPIO0_16	GPIO0_60	GPIO0_59	GPIO0_89
8	~	GPMC0_AD5	~	GPMC0_A5
9	RGMII7_TX_CTL	OBSCLK1	~	~
10	VOUT0_DATA20	~	DSS_FSYNC1	~
11	VPFE0_DATA4	~	~	~
12	MCASP7_AXR2	MCASP0_AXR13	MCASP0_AXR12	MCASP11_ACLKX
13	MCASP7_ACLKR	~	~	~
14	~	~	~	~
Bootstrap	~	BOOTMODE7	~	~

P8.14-P8.16

Pin	P8.14	P8.15	P8.16
GPIO	1 75	1 61	1 62
BALL	AF27	AB29	AB28
REG	0x00011C130	0x00011C0F8	0x00011C0FC
Page	37	33	34
MODE 0	PRG0_PRU1_GPO12	PRG0_PRU0_GPO18	PRG0_PRU0_GPO19
1	PRG0_PRU1_GPI12	PRG0_PRU0_GPI18	PRG0_PRU0_GPI19
2	PRG0_RGMII2_TD1	PRG0_IEP0_EDC_LATCH_IN0	PRG0_IEP0_EDC_SYNC_OUT0
3	PRG0_PWM1_A0	PRG0_PWM0_TZ_IN	PRG0_PWM0_TZ_OUT
4	RGMII4_TD1	PRG0_ECAP0_IN_APWM_OUT	~
5	~	~	~
6	~	~	~
7	GPIO0_75	GPIO0_61	GPIO0_62
8	~	GPMC0_AD6	GPMC0_AD7
9	~	~	~
10	~	~	~
11	~	~	~
12	MCASP1_AXR8	MCASP0_AXR14	MCASP0_AXR15
13	~	~	~
14	UART8_CTSn	~	~
Bootstrap	~	~	~

P8.17-P8.19

Pin	P8.17	P8.18	P8.19
GPIO	1 3	1 4	1 88
BALL	AF22	AJ23	V29
REG	0x00011C00C	0x00011C010	0x00011C164
Page	40	40	57
MODE 0	PRG1_PRU0_GPO2	PRG1_PRU0_GPO3	RGMII5_TD2
1	PRG1_PRU0_GPI2	PRG1_PRU0_GPI3	UART3_TXD
2	PRG1_RGMII1_RD2	PRG1_RGMII1_RD3	~
3	PRG1_PWM2_A0	PRG1_PWM3_A2	SYNC3_OUT
4	RGMII1_RD2	RGMII1_RD3	VOUT1_DATA3
5	RMII1_CRS_DV	RMII1_RX_ER	TRC_DATA1
6	~	~	EHRPWM0_A
7	GPIO0_3	GPIO0_4	GPIO0_88
8	GPMC0_WAIT1	GPMC0_DIR	GPMC0_A4
9	RGMII7_RD2	RGMII7_RD3	~
10	~	~	~
11	~	~	~
12	MCASP6_AXR0	MCASP6_AXR1	MCASP10_AXR1
13	~	~	~
14	UART1_RXD	UART1_TXD	~
Bootstrap	~	~	~

P8.20-P8.22

Pin	P8.20	P8.21	P8.22
GPIO	1 76	1 30	1 5
BALL	AF26	AF21	AH23
REG	0x00011C134	0x00011C07C	0x00011C014
Page	37	49	41
MODE 0	PRG0_PRU1_GPO13	PRG1_PRU1_GPO9	PRG1_PRU0_GPO4
1	PRG0_PRU1_GPI13	PRG1_PRU1_GPI9	PRG1_PRU0_GPI4
2	PRG0_RGMII2_TD2	PRG1_UART0_RXD	PRG1_RGMII1_RX_CTL
3	PRG0_PWM1_B0	~	PRG1_PWM2_B0
4	RGMII4_TD2	SPI6_CS3	RGMII1_RX_CTL
5	~	RMII6_RXD1	RMII1_TXD0
6	~	MCAN8_TX	~
7	GPIO0_76	GPIO0_30	GPIO0_5
8	~	GPMC0_CSn0	GPMC0_CSn2
9	~	PRG1_IEP0_EDIO_DATA_IN_OUT30	RGMII7_RX_CTL
10	~	VOUT0_DATA9	~
11	~	~	~
12	MCASP1_AXR9	MCASP4_AXR3	MCASP6_AXR2
13	~	~	MCASP6_ACLKR
14	UART8_RTSn	~	UART2_RXD
Bootstrap	~	~	~

P8.23-P8.26

Pin	P8.23	P8.24	P8.25	P8.26
GPIO	1 31	1 6	1 35	1 51
BALL	AB23	AD20	AH26	AC27
REG	0x00011C080	0x00011C018	0x00011C090	0x00011C0D0
Page	50	41	51	31
MODE 0	PRG1_PRU1_GPO10	PRG1_PRU0_GPO5	PRG1_PRU1_GPO14	PRG0_PRU0_GPO8
1	PRG1_PRU1_GPI10	PRG1_PRU0_GPI5	PRG1_PRU1_GPI14	PRG0_PRU0_GPI8
2	PRG1_UART0_TXD	~	PRG1_RGMII2_TD3	~
3	PRG1_PWM2_TZ_IN	PRG1_PWM3_B2	PRG1_PWM1_A1	PRG0_PWM2_A1
4	~	~	RGMII2_TD3	~
5	RMII6_CRS_DV	RMII1_TX_EN	~	~
6	MCAN8_RX	~	MCAN8_TX	MCAN9_RX
7	GPIO0_31	GPIO0_6	GPIO0_35	GPIO0_51
8	GPMC0_CLKOUT	GPMC0_WEn	RGMII8_TD3	GPMC0_AD2
9	PRG1_IEP0_EDIO_DATA_IN_OUT31	~	~	~
10	VOUT0_DATA10	~	VOUT0_DATA14	~
11	GPMC0_FCLK_MUX	~	~	~
12	MCASP5_ACLKX	MCASP3_AXR0	MCASP9_AXR1	MCASP0_AXR6
13	~	~	MCASP4_AFSR	~
14	~	~	~	UART6_RXD
Bootstrap	~	BOOTMODE0	~	~

P8.27-P8.29

Pin	P8.27	P8.28	P8.29
GPIO	1 71	1 72	1 73
BALL	AA28	Y24	AA25
REG	0x00011C120	0x00011C124	0x00011C128
Page	36	36	36
MODE 0	PRG0_PRU1_GPO8	PRG0_PRU1_GPO9	PRG0_PRU1_GPO10
1	PRG0_PRU1_GPI8	PRG0_PRU1_GPI9	PRG0_PRU1_GPI10
2	~	PRG0_UART0_RXD	PRG0_UART0_TXD
3	PRG0_PWM2_TZ_OUT	~	PRG0_PWM2_TZ_IN
4	~	SPI3_CS3	~
5	~	~	~
6	MCAN11_RX	PRG0_IEP0_EDIO_DATA_IN_OUT30	PRG0_IEP0_EDIO_DATA_IN_OUT31
7	GPIO0_71	GPIO0_72	GPIO0_73
8	GPMC0_AD10	GPMC0_AD11	GPMC0_AD12
9	~	~	CLKOUT
10	~	DSS_FSYNC3	~
11	~	~	~
12	MCASP1_AFSX	MCASP1_AXR5	MCASP1_AXR6
13	~	~	~
14	~	UART8_RXD	UART8_TXD
Bootstrap	~	~	~

P8.30-P8.32

Pin	P8.30	P8.31	~	P8.32	~
GPIO	1 74	1 32	1 63	1 26	1 64
BALL	AG26	AJ25	AE29	AG21	AD28
REG	0x00011C12C	0x00011C084	0x00011C100	0x00011C06C	0x00011C104
Page	37	50	34	48	34
MODE 0	PRG0_PRU1_GPO11	PRG1_PRU1_GPO11	PRG0_PRU1_GPO0	PRG1_PRU1_GPO5	PRG0_PRU1_GPO1
1	PRG0_PRU1_GPI11	PRG1_PRU1_GPI11	PRG0_PRU1_GPI0	PRG1_PRU1_GPI5	PRG0_PRU1_GPI1
2	PRG0_RGMII2_TD0	PRG1_RGMII2_TD0	PRG0_RGMII2_RD0	~	PRG0_RGMII2_RD1
3	~	~	~	~	~
4	RGMII4_TD0	RGMII2_TD0	RGMII4_RD0	~	RGMII4_RD1
5	RMII4_TX_EN	RMII2_TX_EN	RMII4_RXD0	RMII5_TX_EN	RMII4_RXD1
6	~	~	~	MCAN6_RX	~
7	GPIO0_74	GPIO0_32	GPIO0_63	GPIO0_26	GPIO0_64
8	GPMC0_A26	RGMII8_TD0	UART4_CTSn	GPMC0_WPn	UART4_RTSn
9	~	EQEP1_I	~	EQEP1_S	~
10	~	VOUT0_DATA11	~	VOUT0_DATA5	~
11	~	~	~	~	~
12	MCASP1_AXR7	MCASP9_ACLKX	MCASP1_AXR0	MCASP4_AXR0	MCASP1_AXR1
13	~	~	~	~	~
14	~	~	UART5_RXD	TIMER_IO4	UART5_TXD
Bootstrap	~	~	~	~	~

P8.33-P8.35

Pin	P8.33	~	P8.34	P8.35 | ~	
GPIO	1 25	1 111	1 7	1 24	1 116
BALL	AH24	AA2	AD22	AD23	Y3
REG	0x00011C068	0x00011C1C0	0x00011C01C	0x00011C064	0x00011C1D4
Page	48	67	41	47	67
MODE 0	PRG1_PRU1_GPO4	SPI0_CS0	PRG1_PRU0_GPO6	PRG1_PRU1_GPO3	SPI1_CS0
1	PRG1_PRU1_GPI4	UART0_RTSn	PRG1_PRU0_GPI6	PRG1_PRU1_GPI3	UART0_CTSn
2	PRG1_RGMII2_RX_CTL	~	PRG1_RGMII1_RXC	PRG1_RGMII2_RD3	~
3	PRG1_PWM2_B2	~	PRG1_PWM3_A1	~	UART5_RXD
4	RGMII2_RX_CTL	~	RGMII1_RXC	RGMII2_RD3	~
5	RMII2_TXD0	~	RMII1_TXD1	RMII2_RX_ER	~
6	~	~	AUDIO_EXT_REFCLK0	~	PRG0_IEP0_EDIO_OUTVALID
7	GPIO0_25	GPIO0_111	GPIO0_7	GPIO0_24	GPIO0_116
8	RGMII8_RX_CTL	~	GPMC0_CSn3	RGMII8_RD3	PRG0_IEP0_EDC_LATCH_IN0
9	EQEP1_B	~	RGMII7_RXC	EQEP1_A	~
10	VOUT0_DATA4	~	~	VOUT0_DATA3	~
11	VPFE0_DATA13	~	~	VPFE0_WEN	~
12	MCASP8_AXR2	~	MCASP6_AXR3	MCASP8_AXR1	~
13	MCASP8_ACLKR	~	MCASP6_AFSR	MCASP3_AFSR	~
14	TIMER_IO3	~	UART2_TXD	TIMER_IO2	~
Bootstrap	~	~	~	~	~

P8.36-P8.38

Pin	P8.36	P8.37	~	P8.38	~
GPIO	1 8	1 106	1 11	1 105	1 9
BALL	AE20	Y27	AD21	Y29	AJ20
REG	0x00011C020	0x00011C1AC	0x00011C02C	0x00011C1A8	0x00011C024
Page	42	58	43	58	42
MODE 0	PRG1_PRU0_GPO7	RGMII6_RD2	PRG1_PRU0_GPO10	RGMII6_RD3	PRG1_PRU0_GPO8
1	PRG1_PRU0_GPI7	UART4_RTSn	PRG1_PRU0_GPI10	UART4_CTSn	PRG1_PRU0_GPI8
2	PRG1_IEP0_EDC_LATCH_IN1	~	PRG1_UART0_RTSn	~	~
3	PRG1_PWM3_B1	UART5_TXD	PRG1_PWM2_B1	UART5_RXD	PRG1_PWM2_A1
4	~	~	SPI6_CS2	CLKOUT	~
5	AUDIO_EXT_REFCLK1	TRC_DATA19	RMII5_CRS_DV	TRC_DATA18	RMII5_RXD0
6	MCAN4_TX	EHRPWM5_A	~	EHRPWM_TZn_IN4	MCAN4_RX
7	GPIO0_8	GPIO0_106	GPIO0_11	GPIO0_105	GPIO0_9
8	~	GPMC0_A22	GPMC0_BE0n_CLE	GPMC0_A21	GPMC0_OEn_REn
9	~	~	PRG1_IEP0_EDIO_DATA_IN_OUT29	~	~
10	~	~	OBSCLK2	~	VOUT0_DATA22
11	~	~	~	~	~
12	MCASP3_AXR1	MCASP11_AXR5	MCASP3_AFSX	MCASP11_AXR4	MCASP3_AXR2
13	~	~	~	~	~
14	~	~	~	~	~
Boot-strap	~	~	~	~	~

P8.39-P8.41

Pin	P8.39	P8.40	P8.41
GPIO	1 69	1 70	1 67
BALL	AC26	AA24	AD29
REG	0x00011C118	0x00011C11C	0x00011C110
Page	35	36	35
MODE 0	PRG0_PRU1_GPO6	PRG0_PRU1_GPO7	PRG0_PRU1_GPO4
1	PRG0_PRU1_GPI6	PRG0_PRU1_GPI7	PRG0_PRU1_GPI4
2	PRG0_RGMII2_RXC	PRG0_IEP1_EDC_LATCH_IN1	PRG0_RGMII2_RX_CTL
3	~	~	PRG0_PWM2_B2
4	RGMII4_RXC	SPI3_CS0	RGMII4_RX_CTL
5	RMII4_TXD0	~	RMII4_TXD1
6	~	MCAN11_TX	~
7	GPIO0_69	GPIO0_70	GPIO0_67
8	GPMC0_A25	GPMC0_AD9	GPMC0_A24
9	~	~	~
10	~	~	~
11	~	~	~
12	MCASP1_AXR3	MCASP1_AXR4	MCASP1_AXR2
13	~	~	~
14	~	UART2_TXD	~
Bootstrap	~	~	~

P8.42-P8.44

Pin	P8.42	P8.43	P8.44
GPIO	1 68	1 65	1 66
BALL	AB27	AD27	AC25
REG	0x00011C114	0x00011C108	0x00011C10C
Page	35	34	35
MODE 0	PRG0_PRU1_GPO5	PRG0_PRU1_GPO2	PRG0_PRU1_GPO3
1	PRG0_PRU1_GPI5	PRG0_PRU1_GPI2	PRG0_PRU1_GPI3
2	~	PRG0_RGMII2_RD2	PRG0_RGMII2_RD3
3	~	PRG0_PWM2_A2	~
4	~	RGMII4_RD2	RGMII4_RD3
5	~	RMII4_CRS_DV	RMII4_RX_ER
6	~	~	~
7	GPIO0_68	GPIO0_65	GPIO0_66
8	GPMC0_AD8	GPMC0_A23	~
9	~	~	~
10	~	~	~
11	~	~	~
12	MCASP1_ACLKX	MCASP1_ACLKR	MCASP1_AFSR
13	~	MCASP1_AXR10	MCASP1_AXR11
14	~	~	~
Bootstrap	BOOTMODE6	~	~

P8.45-P8.46

Pin	P8.45	P8.46
GPIO	1 79	1 80
BALL	AG29	Y25
REG	0x00011C140	0x00011C144
Page	38	38
MODE 0	PRG0_PRU1_GPO16	PRG0_PRU1_GPO17
1	PRG0_PRU1_GPI16	PRG0_PRU1_GPI17
2	PRG0_RGMII2_TXC	PRG0_IEP1_EDC_SYNC_OUT1
3	PRG0_PWM1_A2	PRG0_PWM1_B2
4	RGMII4_TXC	SPI3_CLK
5	~	~
6	~	~
7	GPIO0_79	GPIO0_80
8	~	GPMC0_AD13
9	~	~
10	~	~
11	~	~
12	MCASP2_AXR2	MCASP2_AXR3
13	~	~
14	~	~
Bootstrap	~	BOOTMODE3

Connector P9 The following tables show the pinout of the **P9** expansion header. The SW is responsible for setting the default function of each pin. Refer to the processor documentation for more information on these pins and detailed descriptions of all of the pins listed. In some cases there may not be enough signals to complete a group of signals that may be required to implement a total interface.

The column heading is the pin number on the expansion header.

The **GPIO** row is the expected gpio identifier number in the Linux kernel.

Each row includes the gpiochipX and pinY in the format of *X Y*. You can use these values to directly control the GPIO pins with the commands shown below.

```
# to set the GPIO pin state to HIGH
debian@BeagleBone:~$ gpioset X Y=1

# to set the GPIO pin state to LOW
debian@BeagleBone:~$ gpioset X Y=0

For Example:
```

(continues on next page)

(continued from previous page)

```
+----------+----------+
| Pin      | P9.11    |
+==========+==========+
| GPIO     | 1 1      |
+----------+----------+

Use the commands below for controlling this pin (P9.11) where X = 1 and Y = 1

# to set the GPIO pin state to HIGH
debian@BeagleBone:~$ gpioset 1 20=1

# to set the GPIO pin state to LOW
debian@BeagleBone:~$ gpioset 1 20=0
```

The **BALL** row is the pin number on the processor.

The **REG** row is the offset of the control register for the processor pin.

The **MODE #** rows are the mode setting for each pin. Setting each mode to align with the mode column will give that function on that pin.

If included, the **2nd BALL** row is the pin number on the processor for a second processor pin connected to the same pin on the expansion header. Similarly, all row headings starting with **2nd** refer to data for this second processor pin.

NOTES:

DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.

NO PINS ARE TO BE DRIVEN UNTIL AFTER THE SYS_RESET LINE GOES HIGH.

P9.01-P9.05

P9.01	P9.02	P9.03	P9.04	P9.05
GND	GND	VOUT_3V3	VOUT_3V3	VIN

P9.06-P9.10

P9.06	P9.07	P9.08	P9.09	P9.10
VIN	VOUT_SYS	VOUT_SYS	RESET#	RESET#

P9.11-P9.13

Pin	P9.11	P9.12	P9.13
GPIO	1 1	1 45	1 2
BALL	AC23	AE27	AG22
REG	0x00011C004	0x00011C0B8	0x00011C008
Page	39	29	40
MODE 0	PRG1_PRU0_GPO0	PRG0_PRU0_GPO2	PRG1_PRU0_GPO1
1	PRG1_PRU0_GPI0	PRG0_PRU0_GPI2	PRG1_PRU0_GPI1
2	PRG1_RGMII1_RD0	PRG0_RGMII1_RD2	PRG1_RGMII1_RD1
3	PRG1_PWM3_A0	PRG0_PWM2_A0	PRG1_PWM3_B0
4	RGMII1_RD0	RGMII3_RD2	RGMII1_RD1
5	RMII1_RXD0	RMII3_CRS_DV	RMII1_RXD1
6	~	~	~
7	GPIO0_1	GPIO0_45	GPIO0_2
8	GPMC0_BE1n	UART3_RXD	GPMC0_WAIT0
9	RGMII7_RD0	~	RGMII7_RD1
10	~	~	~
11	~	~	~
12	MCASP6_ACLKX	MCASP0_ACLKR	MCASP6_AFSX
13	~	~	~
14	UART0_RXD	~	UART0_TXD
Bootstrap	~	~	~

P9.14-P9.16

Pin	P9.14	P9.15	P9.16
GPIO	1 93	1 47	1 94
BALL	U27	AD25	U24
REG	0x00011C178	0x00011C0C0	0x00011C17C
Page	56	30	56
MODE 0	RGMII5_RD3	PRG0_PRU0_GPO4	RGMII5_RD2
1	UART3_CTSn	PRG0_PRU0_GPI4	UART3_RTSn
2	~	PRG0_RGMII1_RX_CTL	~
3	UART6_RXD	PRG0_PWM2_B0	UART6_TXD
4	VOUT1_DATA8	RGMII3_RX_CTL	VOUT1_DATA9
5	TRC_DATA6	RMII3_TXD1	TRC_DATA7
6	EHRPWM2_A	~	EHRPWM2_B
7	GPIO0_93	GPIO0_47	GPIO0_94
8	GPMC0_A9	~	GPMC0_A10
9	~	~	~
10	~	~	~
11	~	~	~
12	MCASP11_AXR0	MCASP0_AXR2	MCASP11_AXR1
13	~	~	~
14	~	~	~
Bootstrap	~	~	~

P9.17-P9.18

Pin	P9.17	~	P9.18	~
GPIO	1 28	1 115	1 40	1 120
BALL	AC21	AA3	AH22	Y2
REG	0x00011C074	0x00011C1D0	0x00011C0A4	0x00011C1E4
Page	49	67	53	68
MODE 0	PRG1_PRU1_GPO7	SPI0_D1	PRG1_PRU1_GPO19	SPI1_D1
1	PRG1_PRU1_GPI7	~	PRG1_PRU1_GPI19	~
2	PRG1_IEP1_EDC_LATCH_IN1	I2C6_SCL	PRG1_IEP1_EDC_SYNC_OUT0	I2C6_SDA
3	~	~	PRG1_PWM1_TZ_OUT	~
4	SPI6_CS0	~	SPI6_D1	~
5	RMII6_RX_ER	~	RMII6_TXD1	~
6	MCAN7_TX	~	PRG1_ECAP0_IN_APWM_OUT	~
7	GPIO0_28	GPIO0_115	GPIO0_40	GPIO0_120
8	~	~	~	PRG0_IEP1_EDC_SYNC_OUT0
9	~	~	~	~
10	VOUT0_DATA7	~	VOUT0_PCLK	~
11	VPFE0_DATA15	~	~	~
12	MCASP4_AXR1	~	MCASP5_AXR1	~
13	~	~	~	~
14	UART3_TXD	~	~	~
Bootstrap	~	~	~	~

P9.19-P9.20

Pin	P9.19	~	P9.20	~
GPIO	2 1	1 78	2 2	1 77
BALL	W5	AF29	W6	AE25
REG	0x00011C208	0x00011C13C	0x00011C20C	0x00011C138
Page	19	38	19	37
MODE 0	MCAN0_RX	PRG0_PRU1_GPO15	MCAN0_TX	PRG0_PRU1_GPO14
1	~	PRG0_PRU1_GPI15	~	PRG0_PRU1_GPI14
2	~	PRG0_RGMII2_TX_CTL	~	PRG0_RGMII2_TD3
3	~	PRG0_PWM1_B1	~	PRG0_PWM1_A1
4	I2C2_SCL	RGMII4_TX_CTL	I2C2_SDA	RGMII4_TD3
5	~	~	~	~
6	~	~	~	~
7	GPIO1_1	GPIO0_78	GPIO1_2	GPIO0_77
8	~	~	~	~
9	~	~	~	~
10	~	~	~	~
11	~	~	~	~
12	~	MCASP2_AXR1	~	MCASP2_AXR0
13	~	~	~	~
14	~	UART2_RTSn	~	UART2_CTSn
Bootstrap	~	~	~	~

P9.21-P9.22

Pin	P9.21	~	P9.22	~
GPIO	1 39	1 90	1 38	1 91
BALL	AJ22	U28	AC22	U29
REG	0x00011C0A0	0x00011C16C	0x00011C09C	0x00011C170
Page	52	56	52	54
MODE 0	PRG1_PRU1_GPO18	RGMII5_TD0	PRG1_PRU1_GPO17	RGMII5_TXC
1	PRG1_PRU1_GPI18	RMII7_TXD0	PRG1_PRU1_GPI17	RMII7_TX_EN
2	PRG1_IEP1_EDC_LATCH_IN0	I2C3_SDA	PRG1_IEP1_EDC_SYNC_OUT1	I2C6_SCL
3	PRG1_PWM1_TZ_IN	~	PRG1_PWM1_B2	~
4	SPI6_D0	VOUT1_DATA5	SPI6_CLK	VOUT1_DATA6
5	RMII6_TXD0	TRC_DATA3	RMII6_TX_EN	TRC_DATA4
6	PRG1_ECAP0_SYNC_IN	EHRPWM1_A	PRG1_ECAP0_SYNC_OUT	EHRPWM1_B
7	GPIO0_39	GPIO0_90	GPIO0_38	GPIO0_91
8	~	GPMC0_A6	~	GPMC0_A7
9	VOUT0_VP2_VSYNC	~	VOUT0_VP2_DE	~
10	VOUT0_VSYNC	~	VOUT0_DE	~
11	~	~	VPFE0_DATA10	~
12	MCASP5_AXR0	MCASP11_AFSX	MCASP5_AFSX	MCASP10_AXR2
13	~	~	~	~
14	VOUT0_VP0_VSYNC	~	VOUT0_VP0_DE	~
Bootstrap	~	~	BOOTMODE1	~

P9.23-P9.25

Pin	P9.23	P9.24	~	P9.25	~
GPIO	1 10	1 119	1 13	1 127	1 104
BALL	AG20	Y5	AJ24	AC4	W26
REG	0x00011C028	0x00011C1E0	0x00011C034	0x00011C200	0x00011C1A4
Page	42	68	43	69	54
MODE 0	PRG1_PRU0_GPO9	SPI1_D0	PRG1_PRU0_GPO12	UART1_CTSn	RGMII6_RXC
1	PRG1_PRU0_GPI9	UART5_RTSn	PRG1_PRU0_GPI12	MCAN3_RX	~
2	PRG1_UART0_CTSn	I2C4_SCL	PRG1_RGMII1_TD1	~	~
3	PRG1_PWM3_TZ_IN	UART2_TXD	PRG1_PWM0_A0	~	AU-DIO_EXT_REFCLK2
4	SPI6_CS1	~	RGMII1_TD1	SPI2_D0	VOUT1_DE
5	RMII5_RXD1	~	~	EQEP0_S	TRC_DATA17
6	~	~	MCAN4_RX	~	EHRPWM4_B
7	GPIO0_10	GPIO0_119	GPIO0_13	GPIO0_127	GPIO0_104
8	GPMC0_ADVn_ALE	PRG0_IEP1_EDC_LATCH_IN0	~	~	GPMC0_A20
9	PRG1_IEP0_EDIO_DATA_IN_OUT28	~	RGMII7_TD1	~	VOUT1_VP0_DE
10	VOUT0_DATA23	~	VOUT0_DATA17	~	~
11	~	~	VPFE0_DATA1	~	~
12	MCASP3_ACLKX	~	MCASP7_AFSX	~	MCASP10_AXR7
13	~	~	~	~	~
14	~	~	~	~	~
Boot-strap	~	~	~	~	~

P9.26-P9.27

Pin	P9.26	~	P9.27	~
GPIO	1 118	1 12	1 46	1 124
BALL	Y1	AF24	AD26	AB1
REG	0x00011C1DC	0x00011C030	0x00011C0BC	0x00011C1F4
Page	67	43	30	69
MODE 0	SPI1_CLK	PRG1_PRU0_GPO11	PRG0_PRU0_GPO3	UART0_RTSn
1	UART5_CTSn	PRG1_PRU0_GPI11	PRG0_PRU0_GPI3	TIMER_IO7
2	I2C4_SDA	PRG1_RGMII1_TD0	PRG0_RGMII1_RD3	SPI0_CS3
3	UART2_RXD	PRG1_PWM3_TZ_OUT	PRG0_PWM3_A2	MCAN2_TX
4	~	RGMII1_TD0	RGMII3_RD3	SPI2_CLK
5	~	~	RMII3_RX_ER	EQEP0_B
6	~	MCAN4_TX	~	~
7	GPIO0_118	GPIO0_12	GPIO0_46	GPIO0_124
8	PRG0_IEP0_EDC_SYNC_OUT0	~	UART3_TXD	~
9	~	RGMII7_TD0	~	~
10	~	VOUT0_DATA16	~	~
11	~	VPFE0_DATA0	~	~
12	~	MCASP7_ACLKX	MCASP0_AFSR	~
13	~	~	~	~
14	~	~	~	~
Bootstrap	~	~	~	~

P9.28-P9.29

Pin	P9.28	~	P9.29	~
GPIO	2 11	1 43	2 14	1 53
BALL	U2	AF28	V5	AB25
REG	0x00011C230	0x00011C0B0	0x00011C23C	0x00011C0D8
Page	18	29	68	31
MODE 0	ECAP0_IN_APWM_OUT	PRG0_PRU0_GPO0	TIMER_IO1	PRG0_PRU0_GPO10
1	SYNC0_OUT	PRG0_PRU0_GPI0	ECAP2_IN_APWM_OUT	PRG0_PRU0_GPI10
2	CPTS0_RFT_CLK	PRG0_RGMII1_RD0	OBSCLK0	PRG0_UART0_RTSn
3	~	PRG0_PWM3_A0	~	PRG0_PWM2_B1
4	SPI2_CS3	RGMII3_RD0	~	SPI3_CS2
5	I3C0_SDAPULLEN	RMII3_RXD1	~	PRG0_IEP0_EDIO_DATA_IN_OUT29
6	SPI7_CS0	~	SPI7_D1	MCAN10_RX
7	GPIO1_11	GPIO0_43	GPIO1_14	GPIO0_53
8	~	~	~	GPMC0_AD4
9	~	~	~	~
10	~	~	~	~
11	~	~	~	~
12	~	MCASP0_AXR0	~	MCASP0_AFSX
13	~	~	~	~
14	~	~	~	~
Bootstrap	~	~	BOOTMODE5	~

P9.30-P9.31

Pin	P9.30	~	P9.31	~
GPIO	2 13	1 44	2 12	1 52
BALL	V6	AE28	U3	AB26
REG	0x00011C238	0x00011C0B4	0x00011C234	0x00011C0D4
Page	68	29	18	31
MODE 0	TIMER_IO0	PRG0_PRU0_GPO1	EXT_REFCLK1	PRG0_PRU0_GPO9
1	ECAP1_IN_APWM_OUT	PRG0_PRU0_GPI1	SYNC1_OUT	PRG0_PRU0_GPI9
2	SYSCLKOUT0	PRG0_RGMII1_RD1	~	PRG0_UART0_CTSn
3	~	PRG0_PWM3_B0	~	PRG0_PWM3_TZ_IN
4	~	RGMII3_RD1	~	SPI3_CS1
5	~	RMII3_RXD0	~	PRG0_IEP0_EDIO_DATA_IN_OUT28
6	SPI7_D0	~	SPI7_CLK	MCAN10_TX
7	GPIO1_13	GPIO0_44	GPIO1_12	GPIO0_52
8	~	~	~	GPMC0_AD3
9	~	~	~	~
10	~	~	~	~
11	~	~	~	~
12	~	MCASP0_AXR1	~	MCASP0_ACLKX
13	~	~	~	~
14	~	~	~	UART6_TXD
Bootstrap	BOOTMODE4	~	~	~

P9.32-P9.35

P9.32	P9.34
VDD_ADC	GND

Pin	P9.33	~	P9.35	~
GPIO	~	1 50	~	1 55
BALL	K24	AC28	K29	AH27
REG	0x00011C140	0x00011C0CC	0x00011C148	0x00011C0E0
Page	20	31	20	32
MODE 0	MCU_ADC0_AIN4	PRG0_PRU0_GPO7	MCU_ADC0_AIN6	PRG0_PRU0_GPO12
1	~	PRG0_PRU0_GPI7	~	PRG0_PRU0_GPI12
2	~	PRG0_IEP0_EDC_LATCH_IN1	~	PRG0_RGMII1_TD1
3	~	PRG0_PWM3_B1	~	PRG0_PWM0_A0
4	~	PRG0_ECAP0_SYNC_IN	~	RGMII3_TD1
5	~	~	~	~
6	~	MCAN9_TX	~	~
7	~	GPIO0_50	~	GPIO0_55
8	~	GPMC0_AD1	~	~
9	~	~	~	~
10	~	~	~	DSS_FSYNC0
11	~	~	~	~
12	~	MCASP0_AXR5	~	MCASP0_AXR8
13	~	~	~	~
14	~	~	~	~
Bootstrap	~	~	~	~

P9.36-P9.37

Pin	P9.36	~	P9.37	~
GPIO	~	1 56	~	1 57
BALL	K27	AH29	K28	AG28
REG	0x00011C144	0x00011C0E4	0x00011C138	0x00011C0E8
Page	20	32	20	32
MODE 0	MCU_ADC0_AIN5	PRG0_PRU0_GPO13	MCU_ADC0_AIN2	PRG0_PRU0_GPO14
1	~	PRG0_PRU0_GPI13	~	PRG0_PRU0_GPI14
2	~	PRG0_RGMII1_TD2	~	PRG0_RGMII1_TD3
3	~	PRG0_PWM0_B0	~	PRG0_PWM0_A1
4	~	RGMII3_TD2	~	RGMII3_TD3
5	~	~	~	~
6	~	~	~	~
7	~	GPIO0_56	~	GPIO0_57
8	~	~	~	UART4_RXD
9	~	~	~	~
10	~	DSS_FSYNC2	~	~
11	~	~	~	~
12	~	MCASP0_AXR9	~	MCASP0_AXR10
13	~	~	~	~
14	~	~	~	~
Bootstrap	~	~	~	~

P9.38-P9.39

Pin	P9.38	~	P9.39	~
GPIO	~	1 58	~	1 54
BALL	L28	AG27	K25	AJ28
REG	0x00011C13C	0x00011C0EC	0x00011C130	0x00011C0DC
Page	~	33	20	32
MODE 0	MCU_ADC0_AIN3	PRG0_PRU0_GPO15	MCU_ADC0_AIN0	PRG0_PRU0_GPO11
1	~	PRG0_PRU0_GPI15	~	PRG0_PRU0_GPI11
2	~	PRG0_RGMII1_TX_CTL	~	PRG0_RGMII1_TD0
3	~	PRG0_PWM0_B1	~	PRG0_PWM3_TZ_OUT
4	~	RGMII3_TX_CTL	~	RGMII3_TD0
5	~	~	~	~
6	~	~	~	~
7	~	GPIO0_58	~	GPIO0_54
8	~	UART4_TXD	~	~
9	~	~	~	CLKOUT
10	~	DSS_FSYNC3	~	~
11	~	~	~	~
12	~	MCASP0_AXR11	~	MCASP0_AXR7
13	~	~	~	~
14	~	~	~	~
Bootstrap	~	~	~	~

P9.40-P9.42

Pin	P9.40	~	P9.41	P9.42	~
GPIO	~	1 81	2 0	1 123	1 18
BALL	K26	AA26	AD5	AC2	AJ21
REG	0x00011C134	0x00011C148	0x00011C204	0x00011C1F0	0x00011C04C
Page	20	38	69	68	45
MODE 0	MCU_ADC0_AIN1	PRG0_PRU1_GPO18	UART1_RTSn	UART0_CTSn	PRG1_PRU0_GPO17
1	~	PRG0_PRU1_GPI18	MCAN3_TX	TIMER_IO6	PRG1_PRU0_GPI17
2	~	PRG0_IEP1_EDC_LATCH_IN0	~	SPI0_CS2	PRG1_IEP0_EDC_SYNC_OUT1
3	~	PRG0_PWM1_TZ_IN	~	MCAN2_RX	PRG1_PWM0_B2
4	~	SPI3_D0	SPI2_D1	SPI2_CS0	~
5	~	~	EQEP0_I	EQEP0_A	RMII5_TXD1
6	~	MCAN12_TX	~	~	MCAN5_TX
7	~	GPIO0_81	GPIO1_0	GPIO0_123	GPIO0_18
8	~	GPMC0_AD14	~	~	~
9	~	~	~	~	~
10	~	~	~	~	~
11	~	~	~	~	VPFE0_DATA6
12	~	MCASP2_AFSX	~	~	MCASP3_AXR3
13	~	~	~	~	~
14	~	UART2_RXD	~	~	~
Bootstrap	~	~	~	~	~

P9.43-P9.46

P9.43	P9.44	P9.45	P9.46
GND	GND	GND	GND

2.5.7 BeagleBone AI-64 Mechanical

Dimensions and Weight

Size: 102.5 x 80 (4” x 3.15”)

Max height: #TODO#

PCB Layers: #TODO#

PCB thickness: 2mm (0.08”)

RoHS Compliant: Yes

Weight: 192gm

Silkscreen and Component Locations

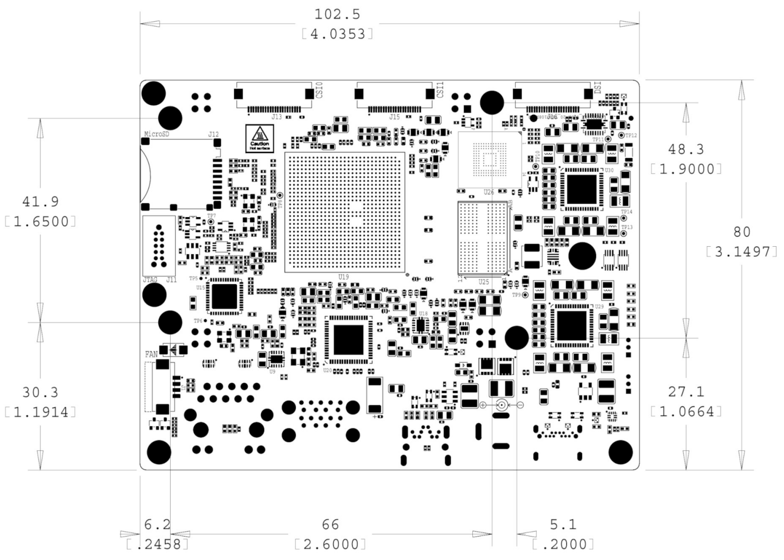


Fig. 2.101: Fig: Board Dimensions

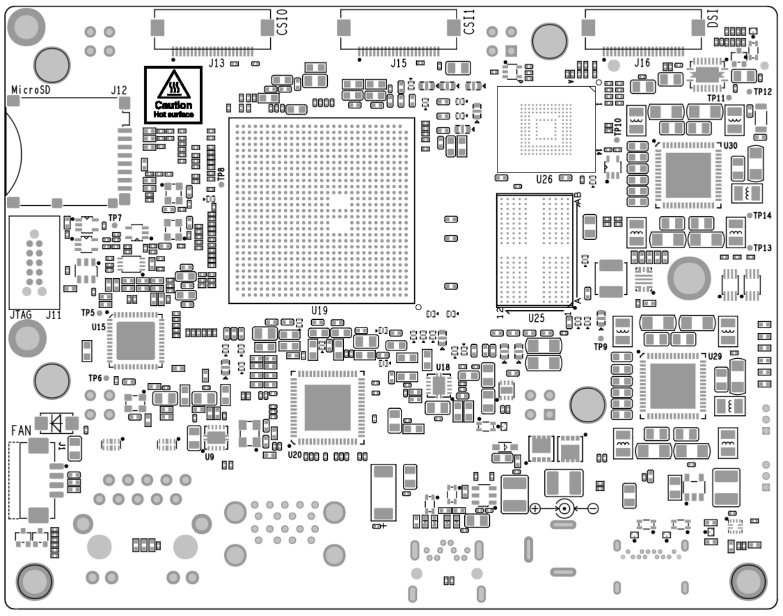


Fig. 2.102: Fig: Top silkscreen

2.5.8 Pictures

2.5.9 Support Information

All support for this design is through BeagleBoard.org community at: link: BeagleBoard.org forum .

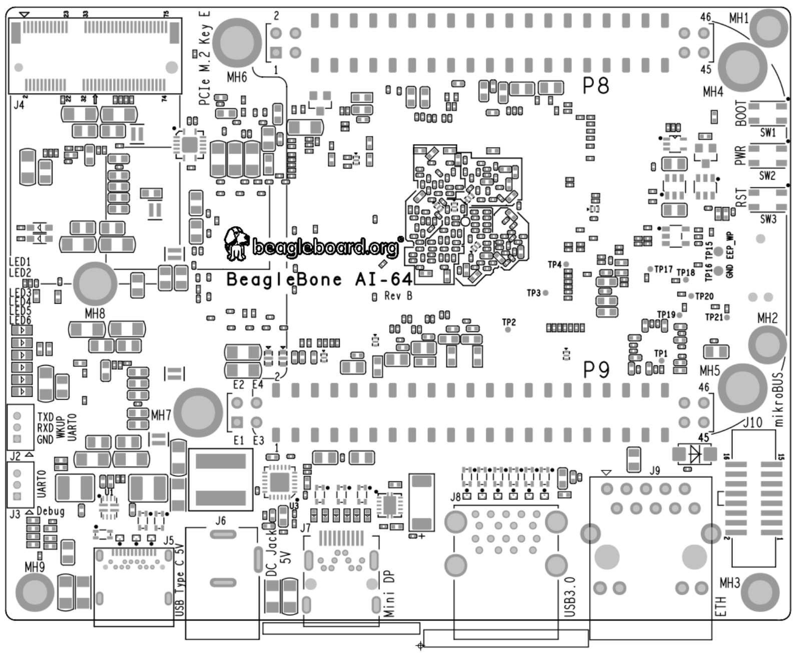


Fig. 2.103: Fig: Bottom silkscreen

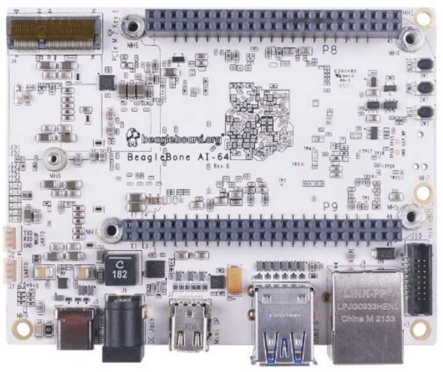


Fig. 2.104: Fig: BeagleBone AI-64 front

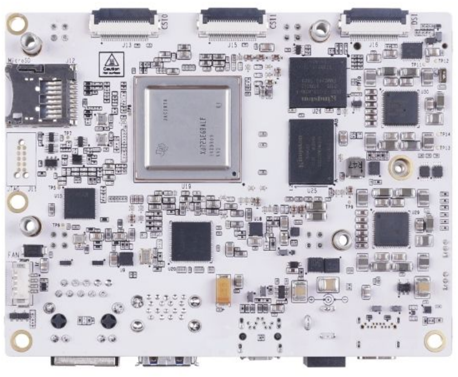


Fig. 2.105: Fig: BeagleBone AI-64 back

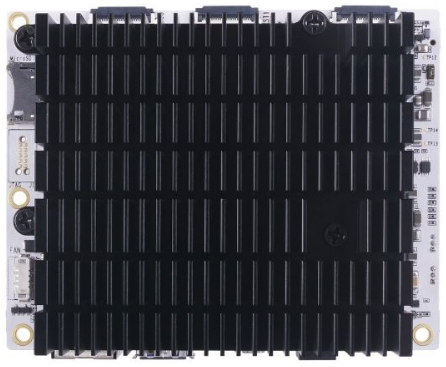


Fig. 2.106: Fig: BeagleBone AI-64 back with heatsink

Fig. 2.107: Fig: BeagleBone AI-64 front at 45° angle

Fig. 2.108: Fig: BeagleBone AI-64 back at 45° angle

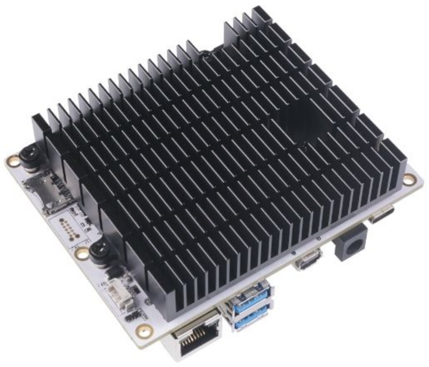


Fig. 2.109: Fig: BeagleBone AI-64 back with heatsink at 45° angle

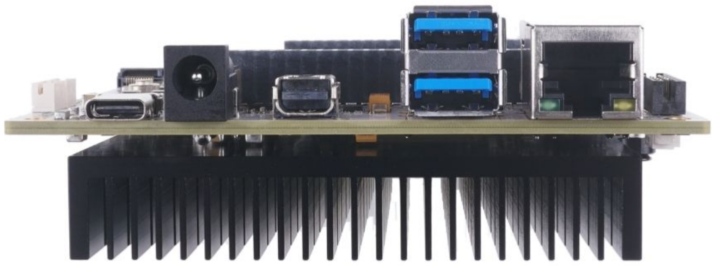


Fig. 2.110: Fig: BeagleBone AI-64 ports

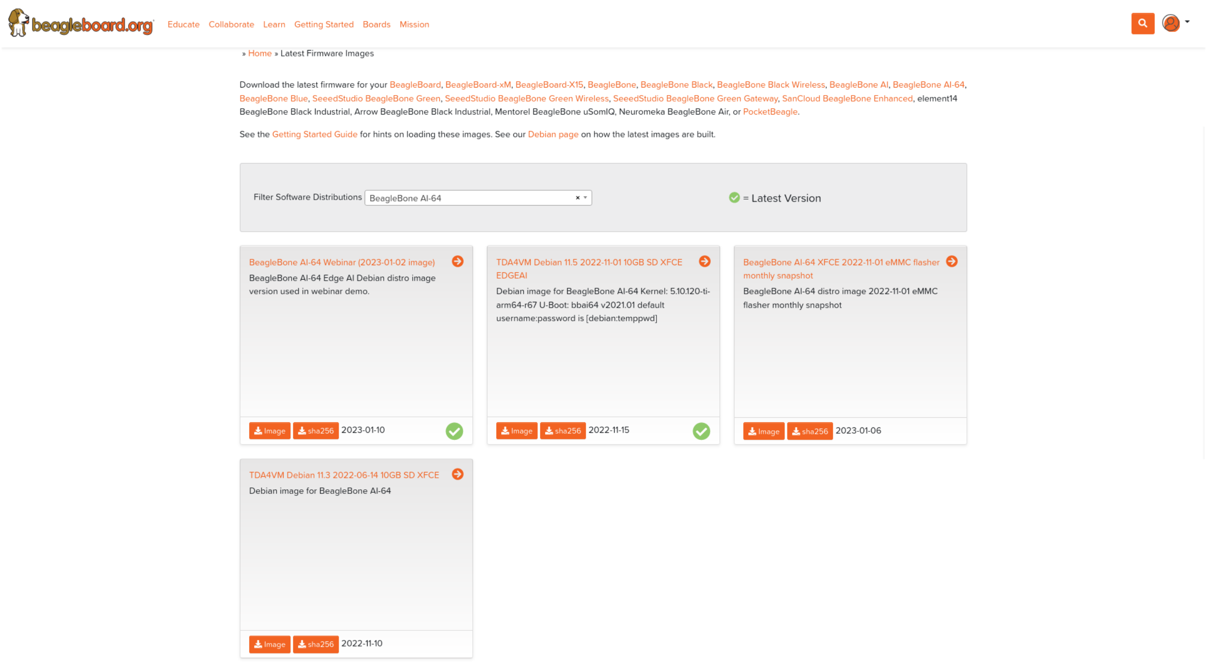
Hardware Design

You can find all BeagleBone AI-64 hardware files here under the *hw* folder.

Software Updates

Follow instructions below to download the latest image for your BeagleBone AI-64:

1. Go to BeagleBoard.org distro page.
2. *Filter Software Distributions for BeagleBone AI-64* from dropdown and download the image.



» Home » Latest Firmware Images

Download the latest firmware for your BeagleBoard, BeagleBoard-xM, BeagleBoard-X15, BeagleBone, BeagleBone Black, BeagleBone Black Wireless, BeagleBone AI, BeagleBone AI-64, BeagleBone Blue, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SeeedStudio BeagleBone Green Gateway, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial, Mentorel BeagleBone uSomIQ, Neuromeka BeagleBone Air, or PocketBeagle.

See the Getting Started Guide for hints on loading these images. See our Debian page on how the latest images are built.

Filter Software Distributions: BeagleBone AI-64

✓ = Latest Version

BeagleBone AI-64 Webinar (2023-01-02 image)
BeagleBone AI-64 Edge AI Debian distro image version used in webinar demo.
Image | sha256 | 2023-01-10 ✓

TDA4VM Debian 11.5 2022-11-01 10GB SD XFCE EDGEAI
Debian image for BeagleBone AI-64 Kernel: 5.10.120-ti-arm64-r67 U-Boot: bbai64 v2021.01 default username:password is [debian:temppwd]
Image | sha256 | 2022-11-15 ✓

BeagleBone AI-64 XFCE 2022-11-01 eMMC flasher monthly snapshot
BeagleBone AI-64 distro image 2022-11-01 eMMC flasher monthly snapshot
Image | sha256 | 2023-01-06

TDA4VM Debian 11.3 2022-06-14 10GB SD XFCE
Debian image for BeagleBone AI-64
Image | sha256 | 2022-11-10

Fig. 2.111: Filter Software Distributions for BeagleBone AI-64

Tip: You can follow the *Update board with latest software* guide for more information on flashing the downloaded image to your board.

To see what SW revision is loaded into the eMMC check */etc/dogtag*. It should look something like as shown below,

` root@BeagleBone:~# cat /etc/dogtag BeagleBoard.org Debian Bullseye Xfce Image 2022-01-14 `

RMA Support

If you feel your board is defective or has issues, request an Return Merchandise Application (RMA) by filling out the form at http://beagleboard.org/support/rma . You will need the serial number and revision of the board. The serial numbers and revisions keep moving. Different boards can have different locations depending on when they were made. The following figures show the three locations of the serial and revision number.

Troubleshooting video output issues

Warning: When connecting to an HDMI monitor, make sure your miniDP adapter is *active*. A *passive* adapter will not work. See *Fig: Display adapters*.

Getting Help If you need some up to date troubleshooting techniques, you can post your queries on link: BeagleBoard.org forum

2.5.10 Update software on BeagleBone AI-64

Production boards currently ship with the factory-installed 2022-01-14-8GB image. To upgrade from the software image on your BeagleBone AI-64 to the latest, you don't need to completely reflash the board. If you do want to reflash it, visit the flashing instructions on the getting started page. Factory Image update (without reflashing)…

```
sudo apt update
sudo apt install --only-upgrade bb-j721e-evm-firmware generic-sys-mods
sudo apt upgrade
```

Update U-Boot:

to ensure only tiboot3.bin is in boot0, the pre-production image we tried to do more in boot0, but failed…

```
sudo /opt/u-boot/bb-u-boot-beagleboneai64/install-emmc.sh
sudo /opt/u-boot/bb-u-boot-beagleboneai64/install-microsd.sh
sudo reboot
```

Update Kernel and SGX modules:

```
sudo apt install bbb.io-kernel-5.10-ti-k3-j721e
```

Update xfce:

```
sudo apt install bbb.io-xfce4-desktop
```

Update ti-edge-ai 8.2 examples

```
sudo apt install ti-edgeai-8.2-base ti-vision-apps-8.2 ti-vision-apps-eaik-
↪firmware-8.2
```

Cleanup:

```
sudo apt autoremove --purge
```

2.5.11 Edge AI

Getting Started

Hardware setup BeagleBone® AI-64 has TI's TDA4VM SoC which houses dual core A72, high performance vision accelerators, video codec accelerators, latest C71x and C66x DSP, high bandwidth realtime IPs for capture and display, GPU, dedicated safety island security accelerators. The SoC is power optimized to provide best in class performance for perception, sensor fusion, localization and path planning tasks in robotics, industrial and automotive applications.

For more details visit https://www.ti.com/product/TDA4VM

BeagleBone® AI-64 BeagleBone® AI-64 brings a complete system for developing artificial intelligence (AI) and machine learning solutions with the convenience and expandability of the BeagleBone® platform and the peripherals on board to get started right away learning and building applications. With locally hosted, ready-to-use, open-source focused tool chains and development environment, a simple web browser, power source and network connection are all that need to be added to start building performance-optimized embedded applications. Industry-leading expansion possibilities are enabled through familiar BeagleBone® cape headers, with hundreds of open-source hardware examples and dozens of readily available embedded expansion options available off-the-shelf.

To run the demos on BeagleBone® AI-64 you will require,

- BeagleBone® AI-64
- USB camera (Any V4L2 compliant 1MP/2MP camera, Eg. Logitech C270/C920/C922)
- Full HD eDP/HDMI display
- Minimum 16GB high performance SD card
- 100Base-T Ethernet cable connected to internet
- UART cable
- External Power Supply or Power Accessory Requirements
    a. Nominal Output Voltage: 5VDC
    b. Maximum Output Current: 5000 mA

Connect the components to the SK as shown in the image.

USB Camera UVC (USB video class) compliant USB cameras are supported on the BeagleBone® AI-64. The driver for the same is enabled in linux image. The linux image has been tested with C270/C920/C922 versions of Logitech USB cameras. Please refer to pub_edgeai_multiple_usb_cams to stream from multiple USB cameras simultaneously.

IMX219 Raw sensor **IMX219 camera module** from **Raspberry pi / Arducam** is supported by BeagleBone® AI-64. It is a 8MP sensor with no ISP, which can transmit raw SRGGB8 frames over CSI lanes at 1080p 60 fps. This camera module can be ordered from https://www.amazon.com/Raspberry-Pi-Camera-Module-Megapixel/dp/B01ER2SKFS The camera can be connected to any of the 2 RPi zero 22 pin camera headers on BB AI-64 as shown below

Note that the headers have to be lifted up to connect the cameras

Note: To be updated By default IMX219 is disabled. After connecting the camera you can enable it by specifying the dtb overlay file in `/run/media/mmcblk0p1/uenv.txt` as below,

```
name_overlays=k3-j721e-edgeai-apps.dtbo    k3-j721e-sk-rpi-cam-imx219.
dtbo
```

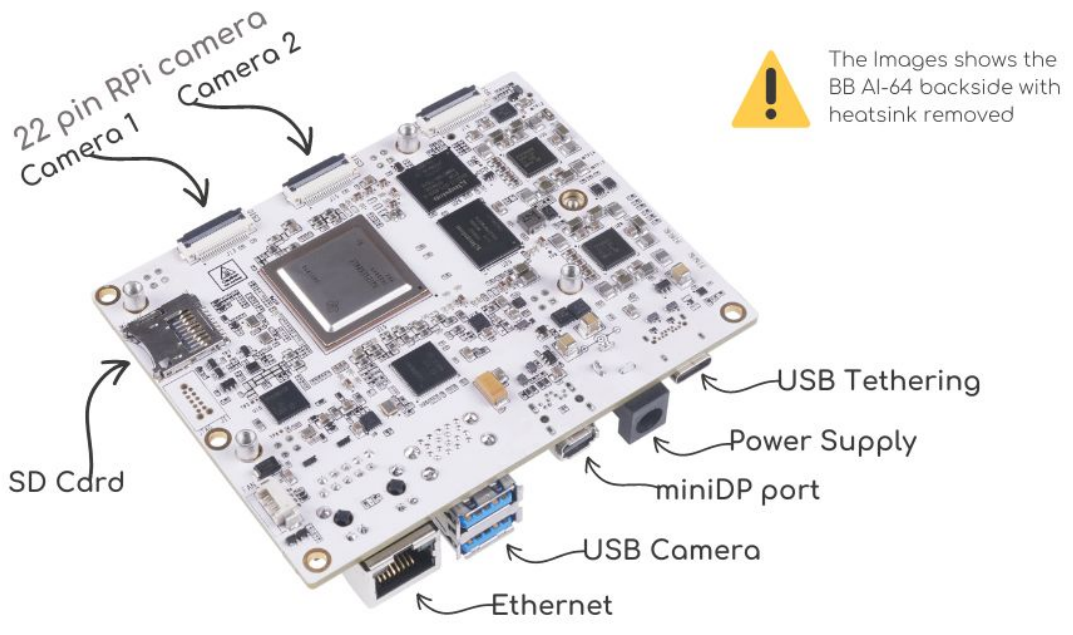


Fig. 2.112: BeagleBone® AI-64 for Edge AI connections

Reboot the board after editing and saving the file.

Two RPi cameras can be connected to 2 headers for multi camera use-cases

Please refer pub_edgeai_camera_sources to know how to list all the cameras connected and select which one to use for the demo.

By default imx219 will be configured to capture at 8 bit, but it also supports 10 bit capture in 16 bit container. To use it in 10 bit mode, below steps are required:

- Modify the `/opt/edge_ai_apps/scripts/setup_cameras.sh` to set the format to 10 bit like below

```
CSI_CAM_0_FMT='[fmt:SRGGB8_1X10/1920x1080]'
CSI_CAM_1_FMT='[fmt:SRGGB8_1X10/1920x1080]'
```

- Change the imaging binaries to use 10 bit versions

```
mv /opt/imaging/imx219/dcc_2a.bin /opt/imaging/imx219/dcc_2a_8b.bin
mv /opt/imaging/imx219/dcc_viss.bin /opt/imaging/imx219/dcc_viss_8b.
↪bin
mv /opt/imaging/imx219/dcc_2a_10b.bin /opt/imaging/imx219/dcc_2a.bin
mv /opt/imaging/imx219/dcc_viss_10b.bin /opt/imaging/imx219/dcc_viss.
↪bin
```

- Set the input format in the `/opt/edge_ai_apps/configs/rpiV2_cam_example.yaml` as `rggb10`

Software setup

Preparing SD card image Download the `bullseye-xfce-edgeai-arm64` image from the links below and flash it to SD card using Balena etcher tool.

- To use via SD card: bbai64-debian-11.4-xfce-edgeai-arm64-2022-08-02-10gb.img.xz

- To flash on eMMC: bbai64-emmc-flasher-debian-11.4-xfce-edgeai-arm64-2022-08-02-10gb.img.xz

The Balena etcher tool can be installed either on Windows/Linux. Just download the etcher image and follow the instructions to prepare the SD card.

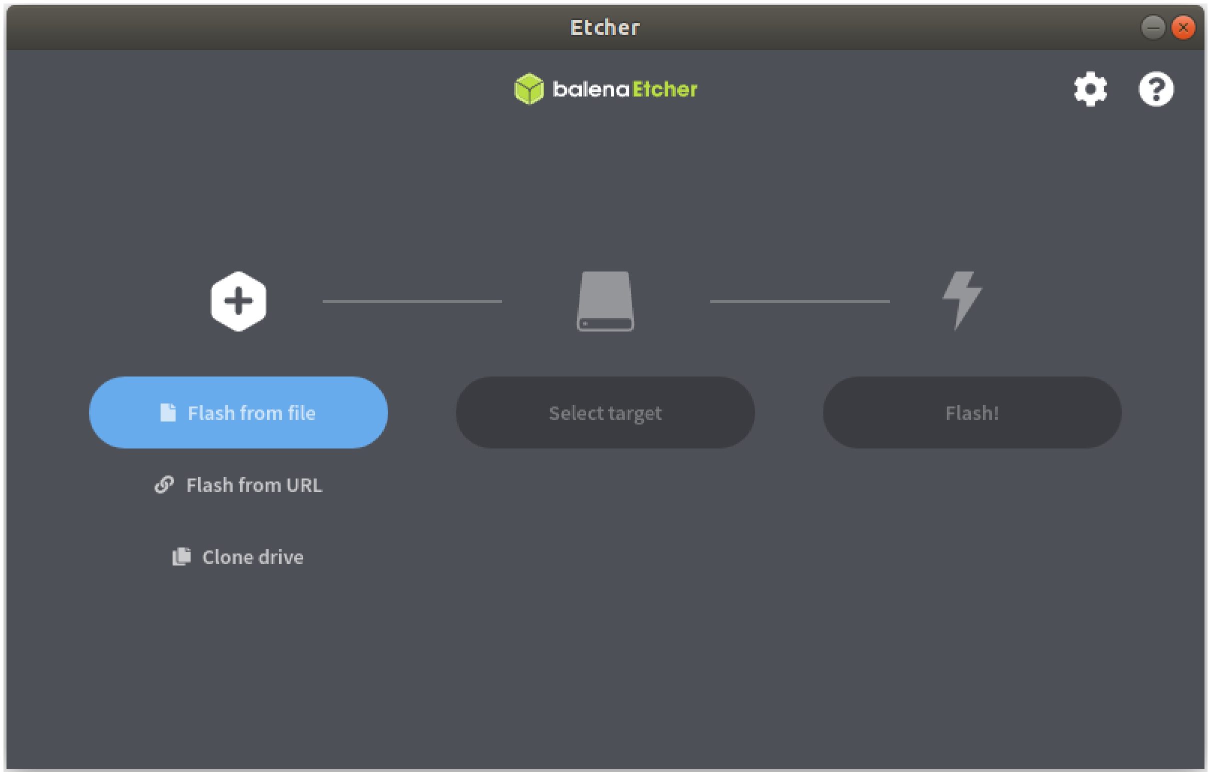


Fig. 2.113: Balena Etcher tool to flash SD card with Processor linux image Linux for Edge AI

The etcher image is created for 16 GB SD cards, if you are using larger SD card, it is possible to expand the root filesystem to use the full SD card capacity using below steps

```
#find the SD card device entry using lsblk (Eg: /dev/sdc)
#use the following commands to expand the filesystem
#Make sure you have write permission to SD card or run the commands as root

#Unmount the BOOT and rootfs partition before using parted tool
umount /dev/sdX1
umount /dev/sdX2

#Use parted tool to resize the rootfs partition to use
#the entire remaining space on the SD card
#You might require sudo permissions to execute these steps
parted -s /dev/sdX resizepart 2 '100%'
e2fsck -f /dev/sdX2
resize2fs /dev/sdX2

#replace /dev/sdX in above commands with SD card device entry
```

Power ON and Boot Ensure that the power supply is disconnected before inserting the SD card. Once the SD card is firmly inserted in its slot and the board is powered ON, the board will take less than 20sec to boot and display a wallpaper as shown in the image below.

You can also view the boot log by connecting the UART cable to your PC and use a serial port communications program.

For **Linux OS minicom** works well. Please refer to the below documentation on 'minicom' for more details.

https://help.ubuntu.com/community/Minicom

When starting minicom, turn on the colors options like below:

```
sudo minicom -D /dev/ttyUSB2 -c on
```

For **Windows OS Tera Term** works well. Please refer to the below documentation on 'TeraTerm' for more details

https://learn.sparkfun.com/tutorials/terminal-basics/tera-term-windows

---

Note: Baud rate should be configured to 115200 bps in serial port communication program. You may not see any log in the UART console if you connect to it after the booting is complete or login prompt may get lost in between boot logs, press ENTER to get login prompt

---

As part of the linux systemd `/opt/edge_ai_apps/init_script.sh` is executed which does the below,

- This kills weston compositor which holds the display pipe. This step will make the wallpaper showing on the display disappear and come back
- The display pipe can now be used by 'kmssink' GStreamer element while running the demo applications.
- The script can also be used to setup proxies if connected behind a firewall.

Once Linux boots login as `root` user with no password.

Connect remotely If you don't prefer the UART console, you can also access the device with the IP address that is shown on the display.

With the IP address one can ssh directly to the board, view the contents and run the demos.

For best experience we recommend using VSCode which can be downloaded from here.

https://code.visualstudio.com/download

You also require the "Remote development extension pack" installed in VSCode as mentioned here:

https://code.visualstudio.com/docs/remote/ssh

Running Simple demos

This chapter describes how to run Python and C++ demo applications in edge_ai_apps with live camera and display.

Note: Please note that the Python demos are useful for quick prototyping while C++ demos are similar by design but tuned for performance.

Running Python based demo applications Python based demos are simple executable scripts written for image classification, object detection and semantic segmentation. Demos are configured using a YAML file. Details on configuration file parameters can be found in pub_edgeai_configuration

Sample configuration files for out of the box demos can be found in `edge_ai_apps/configs` this folder also contains a template config file which has brief info on each configurable parameter `edge_ai_apps/configs/app_config_template.yaml`

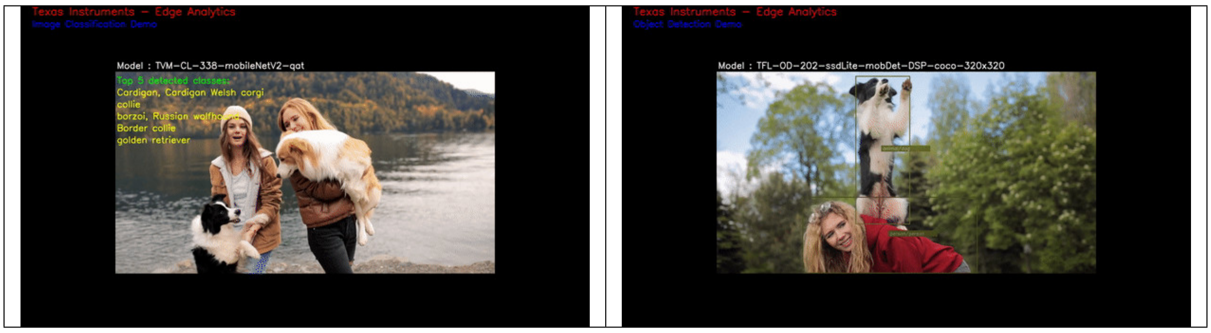
Here is how a Python based image classification demo can be run,

```
# go to edge-ai-apps folder
debian@beaglebone:~$ cd /opt/edge_ai_apps/apps_python

# enable root (password: temppwd)
debian@beaglebone:~$ sudo su
[sudo] password for beaglebone:

# use edge-ai-apps
debian@beaglebone:/opt/edge_ai_apps/apps_cpp# sudo ./app_edgeai.py ../
↪configs/image_classification.yaml
```

The demo captures the input frames from connected USB camera and passes through pre-processing, inference and post-processing before sent to display. Sample output for image classification and object detection demos are as below,


Texas Instruments - Edge Analytics
Image Classification Demo
Model : TVM-CL-338-mobileNetV2-qat
Top 5 detected classes:
Cardigan, Cardigan Welsh corgi
collie
borzoi, Russian wolfhound
Border collie
golden retriever
Texas Instruments - Edge Analytics
Object Detection Demo
Model : TFL-OD-202-ssdLite-mobDet-DSP-coco-320x320

To exit the demo press Ctrl+C.

Building and running C++ based demo applications C++ apps needs to be built directly on target and requires header files of different deep-learning runtime framework and its dependencies which are installed in the setup script. The setup script builds the C++ apps when executed. However one can also follow below steps to clean build C++ apps

```
debian@beaglebone:/opt/edge_ai_apps/apps_cpp# rm -rf build bin lib
debian@beaglebone:/opt/edge_ai_apps/apps_cpp# mkdir build
debian@beaglebone:/opt/edge_ai_apps/apps_cpp# cd build
debian@beaglebone:/opt/edge_ai_apps/apps_cpp/build# cmake ..
debian@beaglebone:/opt/edge_ai_apps/apps_cpp/build# make -j2
```

Run the demo once the application is successfully built

```
debian@beaglebone:/opt/edge_ai_apps/apps_cpp# ./bin/Release/app_edgeai ../
↪configs/image_classification.yaml
```

To exit the demo press Ctrl+C.

Note: Both Python and C++ applications are similar by construction and can accept the same config file and command line arguments

Note: The C++ apps built on Yocto Linux may not run in Docker as there could be a mismatch in Glib and other related tools. So its **highly recommended** to rebuild the C++ apps within the Docker environment.

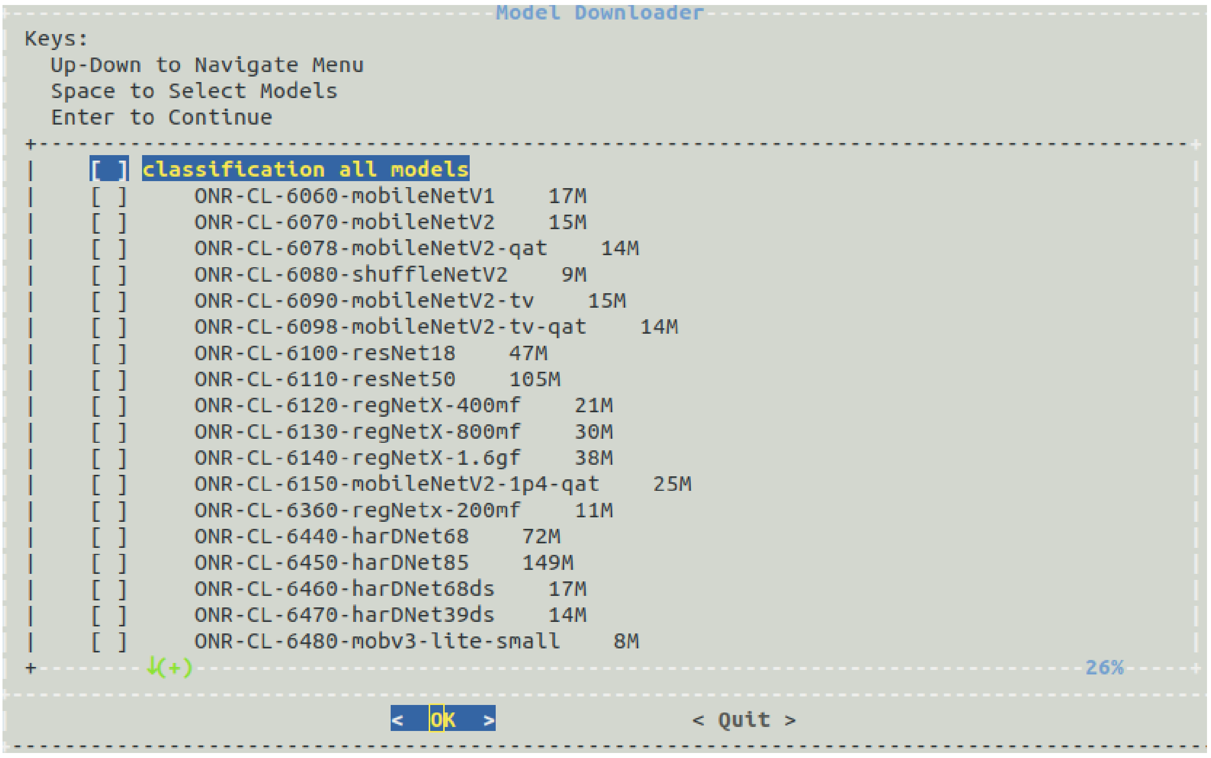
DL models for Edge Inference

Model Downloader Tool TI Model Zoo is a large collection of deep learning models validated to work on TI processors for edge AI. It hosts several pre-compiled model artifacts for TI hardware.

Use the **Model Downloader Tool** to download more models on target as shown,

```
debian@beaglebone:/opt/edge_ai_apps# ./download_models.sh
```

The script will launch an interactive menu showing the list of available, pre-imported models for download. The downloaded models will be placed under `/opt/model_zoo/` directory



```
Model Downloader
Keys:
  Up-Down to Navigate Menu
  Space to Select Models
  Enter to Continue
+------------------------------------------------------------------------+
|    [ ] classification all models
|    [ ]     ONR-CL-6060-mobileNetV1    17M
|    [ ]     ONR-CL-6070-mobileNetV2    15M
|    [ ]     ONR-CL-6078-mobileNetV2-qat    14M
|    [ ]     ONR-CL-6080-shuffleNetV2    9M
|    [ ]     ONR-CL-6090-mobileNetV2-tv    15M
|    [ ]     ONR-CL-6098-mobileNetV2-tv-qat    14M
|    [ ]     ONR-CL-6100-resNet18    47M
|    [ ]     ONR-CL-6110-resNet50    105M
|    [ ]     ONR-CL-6120-regNetX-400mf    21M
|    [ ]     ONR-CL-6130-regNetX-800mf    30M
|    [ ]     ONR-CL-6140-regNetX-1.6gf    38M
|    [ ]     ONR-CL-6150-mobileNetV2-1p4-qat    25M
|    [ ]     ONR-CL-6360-regNetx-200mf    11M
|    [ ]     ONR-CL-6440-harDNet68    72M
|    [ ]     ONR-CL-6450-harDNet85    149M
|    [ ]     ONR-CL-6460-harDNet68ds    17M
|    [ ]     ONR-CL-6470-harDNet39ds    14M
|    [ ]     ONR-CL-6480-mobv3-lite-small    8M
+--------↓(+)---------------------------------------------------26%-----+

                  <  OK  >                  < Quit >
```

Fig. 2.114: Model downloader tool menu option to download models

The script can also be used in a non-interactive way as shown below:

```
debian@beaglebone:/opt/edge_ai_apps# ./download_models.sh --help
```

Import Custom Models The BeagleBone® AI-64 Linux for Edge AI also supports importing pre-trained custom models to run inference on target.

The SDK makes use of pre-compiled DNN (Deep Neural Network) models and performs inference using various OSRT (open source runtime) such as TFLite runtime, ONNX runtime and Neo AI-DLR. In order to infer a DNN, SDK expects the DNN and associated artifacts in the below directory structure.

```
TFL-OD-2010-ssd-mobV2-coco-mlperf-300x300
│
├── param.yaml
│
├── artifacts
│    ├── 264_tidl_io_1.bin
│    ├── 264_tidl_net.bin
│    ├── 264_tidl_net.bin.layer_info.txt
│    ├── 264_tidl_net.bin_netLog.txt
│    ├── 264_tidl_net.bin.svg
```

(continues on next page)

(continued from previous page)

```
│    ├── allowedNode.txt
│    └── runtimes_visualization.svg
│
└── model
     └── ssd_mobilenet_v2_300_float.tflite
```

DNN directory structure Each DNN must have the following 3 components:

1. **model**: This directory contains the DNN being targeted to infer
2. **artifacts**: This directory contains the artifacts generated after the compilation of DNN for SDK, and described in pub_edgeai_compile_artifacts
3. **param.yaml**: A configuration file in yaml format to provide basic information about DNN, and associated pre and post processing parameters. More details can be find pub_edgeai_params

Param file format Each DNN has its own pre-process, inference and post-process parameters to get the correct output. This information is typically available in the training software that was used to train the model. In order to convey this information to the SDK in a standardized fashion, we have defined a set of parameters that describe these operations. These parameters are in the param.yaml file.

Please see sample yaml files for various tasks such as image classification, semantic segmentation and object detection in edgeai-benchmark examples. Descriptions of various parameters are also in the yaml files. If users want to bring their own model to the SDK, then they need to prepare this information offline and get to the SDK. In next section we explain how to prepare this information

DNN compilation for SDK – Basic Instructions The BeagleBone® AI-64 Linux for Edge AI supports three different runtimes to infer a DNN, and user can choose a run time depending on the format of DNN. We recommend users to use different run times and compare the performance and select the one which provides best performance. User can find the steps to generate the artifacts directory at Edge AI TIDL Tools

DNN compilation for SDK – Advanced Instructions For beginners who are trying to compile models for the SDK, we recommend the basic instructions given in the previous section. However, DNNs have lot of variety and some models may need a different kind of preprocessing or postprocessing operations. In order to help customers deal with different kinds of models, we have prepared a model zoo in the repository edgeai-modelzoo

For the DNNs which are part of TI's model zoo, one can find the compilation settings and pre-compiled model artifacts in edgeai-benchmark repository. Instructions are also given to compile custom models. When using edgeai-benchmark for model compilation, the yaml file is automatically generated and artifacts are packaged in the way SDK understands. Please follow the instructions in the repository to get started.

Demo Configuration file

The demo config file uses YAML format to define input sources, models, outputs and finally the flows which defines how everything is connected. Config files for out-of-box demos are kept in `edge_ai_apps/configs` folder. The folder contains config files for all the use cases and also multi-input and multi-inference case. The folder also has a template YAML file `app_config_template.yaml` which has detailed explanation of all the parameters supported in the config file.

Config file is divided in 4 sections:

1. Inputs
2. Models
3. Outputs
4. Flows

Inputs The input section defines a list of supported inputs like camera, filesrc etc. Their properties like shown below.

```
inputs:
    input0:                                            #Camera Input
        source: /dev/video2                            #Device file entry of the␣
↪camera
        format: jpeg                                   #Input data format␣
↪supported by camera
        width: 1280                                    #Width and Height of the␣
↪input
        height: 720
        framerate: 30                                  #Framerate of the source

    input1:                                            #Video Input
        source: ../data/videos/video_0000_h264.mp4     #Video file
        format: h264                                   #File encoding format
        width: 1280
        height: 720
        framerate: 25

    input2:                                            #Image Input
        source: ../data/images/%04d.jpg                #Sequence of Image files,␣
↪printf style formatting is used
        width: 1280
        height: 720
        index: 0                                       #Starting Index␣
↪(optional)
        framerate: 1
```

All supported inputs are listed in template config file. Below are the details of most commonly used inputs.

Camera sources (v4l2) **v4l2src** GStreamer element is used to capture frames from camera sources which are exposed as v4l2 devices. In Linux, there are many devices which are implemented as v4l2 devices. Not all of them will be camera devices. You need to make sure the correct device is configured for running the demo successfully.

`init_script.sh` is ran as part of systemd, which detects all cameras connected and prints the detail like below in the UART console:

```
debian@beaglebone:/opt/edge_ai_apps# ./init_script.sh
USB Camera detected
    device = /dev/video18
    format = jpeg
CSI Camera 0 detected
    device = /dev/video2
    name = imx219 8-0010
    format = [fmt:SRGGB8_1X8/1920x1080]
    subdev_id = 2
    isp_required = yes
IMX390 Camera 0 detected
    device = /dev/video18
    name = imx390 10-001a
    format = [fmt:SRGGB12_1X12/1936x1100 field: none]
    subdev_id = /dev/v4l-subdev7
    isp_required = yes
    ldc_required = yes
```

script can also be run manually later to get the camera details.

From the above log we can determine that 1 USB camera is connected (/dev/video18), and 1 CSI camera is connected (/dev/video2) which is imx219 raw sensor and needs ISP. IMX390 camera needs both ISP and LDC.

Using this method, you can configure correct device for camera capture in the input section of config file.

```
input0:
    source: /dev/video18  #USB Camera
    format: jpeg          #if connected USB camera supports jpeg
    width: 1280
    height: 720
    framerate: 30

input1:
    source: /dev/video2  #CSI Camera
    format: auto         #let the gstreamer negotiate the format
    width: 1280
    height: 720
    framerate: 30

input2:
    source: /dev/video2  #IMX219 raw sensor that needs ISP
    format: rggb         #ISP will be added in the pipeline
    width: 1920
    height: 1080
    framerate: 30
    subdev-id: 2         #needed by ISP to control sensor params via ioctls

input3:
    source: /dev/video2  #IMX390 raw sensor that needs ISP
    width: 1936
    height: 1100
    format: rggb12       #ISP will be added in the pipeline
    subdev-id: 2         #needed by ISP to control sensor params via ioctls
    framerate: 30
    sen-id: imx390
    ldc: True            #LDC will be added in the pipeline
```

Make sure to configure correct `format` for camera input. `jpeg` for USB camera that supports MJPEG (Ex. C270 logitech USB camera). `auto` for CSI camera to allow gstreamer to negotiate the format. `rggb` for sensor that needs ISP.

Video sources H.264 and H.265 encoded videos can be provided as input sources to the demos. Sample video files are provided under `/opt/edge_ai_apps/data/videos/video_0000_h264.mp4` and `/opt/edge_ai_apps/data/videos/video_000_h265.mp4`

```
input1:
    source: ../data/videos/video_0000_h264.mp4
    format: h264
    width: 1280
    height: 720
    framerate: 25

input2:
    source: ../data/videos/video_0000_h265.mp4
    format: h265
    width: 1280
    height: 720
    framerate: 25
```

Make sure to configure correct `format` for video input as shown above. By default the format is set to `auto` which will then use the GStreamer bin `decodebin` instead.

Image sources JPEG compressed images can be provided as inputs to the demos. A sample set of images are provided under `/opt/edge_ai_apps/data/images`. The names of the files are numbered sequentially and incrementally and the demo plays the files at the fps specified by the user.

```
input2:
    source: ../data/images/%04d.jpg
    width: 1280
    height: 720
    index: 0
    framerate: 1
```

RTSP sources H.264 encoded video streams either coming from a RTSP compliant IP camera or via RTSP server running on a remote PC can be provided as inputs to the demo.

```
input0:
    source: rtsp://172.24.145.220:8554/test # rtsp stream url, replace this␣
↪with correct url
    width: 1280
    height: 720
    framerate: 30
```

Note: Usually video streams from any IP camera will be encrypted and cannot be played back directly without a decryption key. We tested RTSP source by setting up an RTSP server on a Ubuntu 18.04 PC by referring to this writeup, Setting up RTSP server on PC

Models The model section defines a list of models that are used in the demo. Path to the model directory is a required argument for each model and rest are optional properties specific to given use cases like shown below.

```
models:
    model0:
        model_path: ../models/segmentation/ONR-SS-871-deeplabv3lite-mobv2-
↪cocoseg21-512x512    #Model Directory
        alpha: 0.4                                                              ␣
↪                   #alpha for blending segmentation mask (optional)
    model1:
        model_path: ../models/detection/TFL-OD-202-ssdLite-mobDet-DSP-coco-
↪320x320
        viz_threshold: 0.3                                                      ␣
↪                 #Visualization threshold for adding bounding boxes␣
↪(optional)
    model2:
        model_path: ../models/classification/TVM-CL-338-mobileNetV2-qat
        topN: 5                                                                 ␣
↪                   #Number of top N classes (optional)
```

Below are some of the use case specific properties:

1. **alpha**: This determines the weight of the mask for blending the semantic segmentation output with the input image `alpha * mask + (1 - alpha) * image`
2. **viz_threshold**: Score threshold to draw the bounding boxes for detected objects in object detection. This can be used to control the number of boxes in the output, increase if there are too many and decrease if there are very few
3. **topN**: Number of most probable classes to overlay on image classification output

The content of the model directory and its structure is discussed in detail in pub_edgeai_import_custom_models

Outputs The output section defines a list of supported outputs.

```
outputs:
    output0:                                                        #Display␣
↪Output
        sink: kmssink
        width: 1920                                                 #Width and␣
↪Height of the output
        height: 1080
        connector: 39                                               #Connector␣
↪ID for kmssink (optional)

    output1:                                                        #Video Output
        sink: ../data/output/videos/output_video.mkv                #Output␣
↪video file
        width: 1920
        height: 1080

    output2:                                                        #Image Output
        sink: ../data/output/images/output_image_%04d.jpg           #Image file␣
↪name, printf style formatting is used
        width: 1920
        height: 1080
```

All supported outputs are listed in template config file. Below are the details of most commonly used outputs

Display Sink (kmssink) When you have only one display connected to the SK, kmssink will try to use it for displaying the output buffers. In case you have connected multiple display monitors (e.g. Display Port and HDMI), you can select a specific display for kmssink by passing a specific connector ID number. Following command finds out the connected displays available to use.

Note: Run this command outside docker container. The first number in each line is the connector-id which we will use in next step.

```
debian@beaglebone:/opt/edge_ai_apps# modetest -M tidss -c | grep connected
39      38      connected       DP-1            530x300         12      38
48      0       disconnected    HDMI-A-1        0x0             0       47
```

From above output, we can see that connector ID 39 is connected. Configure the connector ID in the output section of the config file.

Video sinks The post-processed outputs can be encoded in H.264 format and stored on disk. Please specify the location of the video file in the configuration file.

```
output1:
    sink: ../data/output/videos/output_video.mkv
    width: 1920
    height: 1080
```

Image sinks The post-processed outputs can be stored as JPEG compressed images. Please specify the location of the image files in the configuration file. The images will be named sequentially and incrementally as shown.

```
output2:
    sink: ../data/output/images/output_image_%04d.jpg
    width: 1920
    height: 1080
```

Flows The flows section defines how inputs, models and outputs are connected. Multiple flows can be defined to achieve multi input, multi inference like below.

```
flows:
    flow0:                                      #First Flow
        input: input0                           #Input for the Flow
        models: [model1, model2]                #List of models to be used
        outputs: [output0, output0]             #Outputs to be used for each model␣
↪inference output
        mosaic:                                 #Positions to place the inference␣
↪outputs in the output frame
            mosaic0:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  90
            mosaic1:
                width:  800
                height: 450
                pos_x:  960
                pos_y:  90
    flow1:                                      #Second Flow
        input: input1
        models: [model0, model3]
        outputs: [output0, output0]
        mosaic:
            mosaic0:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  540
            mosaic1:
                width:  800
                height: 450
                pos_x:  960
                pos_y:  540
```

Each flow should have exactly **1 input**, **n models** to infer the given input and **n outputs** to render the output of each inference. Along with input, models and outputs it is required to define **n mosaics** which are the position of the inference output in the final output plane. This is needed because multiple inference outputs can be rendered to same output (Ex: Display).

Command line arguments Limited set of command line arguments can be provided, run with '-h' or '–help' option to list the supported parameters.

```
usage: Run : ./app_edgeai.py -h for help

positional arguments:
  config           Path to demo config file
                       ex: ./app_edgeai.py ../configs/app_config.yaml

optional arguments:
  -h, --help       show this help message and exit
  -n, --no-curses  Disable curses report
                   default: Disabled
  -v, --verbose    Verbose option to print profile info on stdout
                   default: Disabled
```

Running Advance demos

The same Python and C++ demo applications can be used to run multiple inference models and also work with multiple inputs with just simple changes in the config file.

From a repo of input sources, output sources and models one can define advance dataflows which connect them in various configurations. Details on configuration file parameters can be found in pub_edgeai_configuration

Single input multi inference demo Here is an example of a single-input, multi-inference demo which takes a camera input and run multiple networks on each of them.

```
debian@beaglebone:/opt/edge_ai_apps/apps_python# ./app_edgeai.py ../configs/
↪single_input_multi_infer.yaml
```

Sample output for single input, multi inference demo is as shown below,

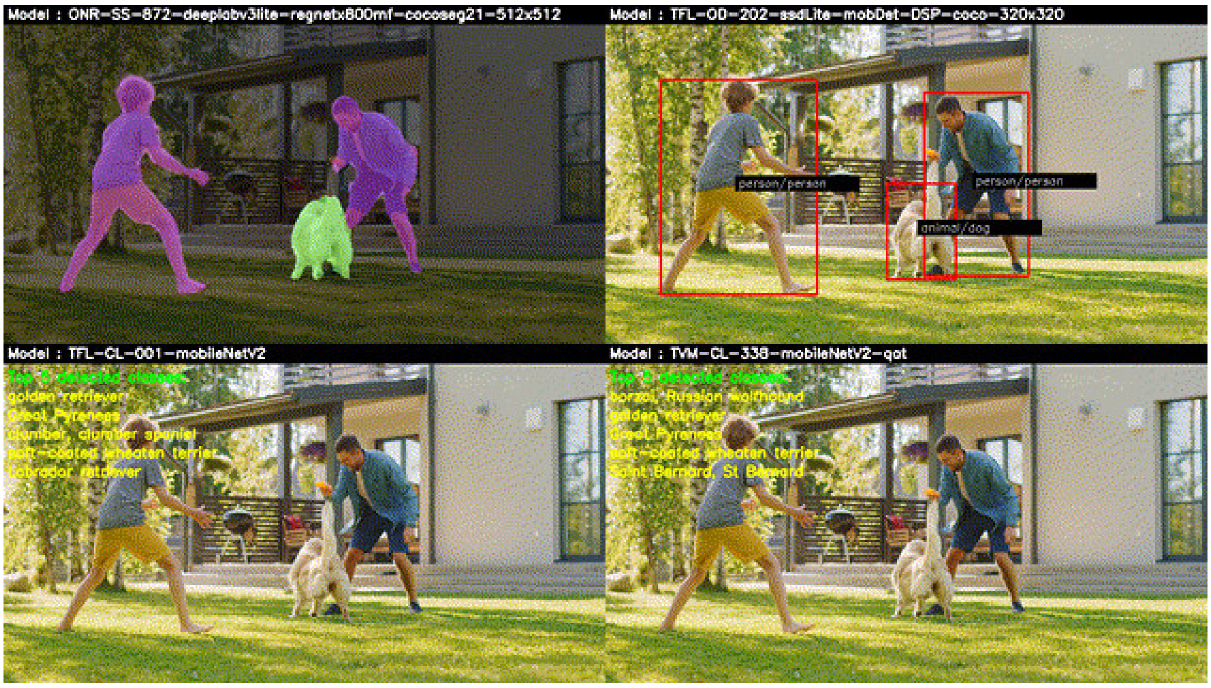

Model : ONR-SS-872-deeplabv3lite-regnetx800mf-cocoseg21-512x512
Model : TFL-OD-202-ssdLite-mobDet-DSP-coco-320x320
person/person
person/person
animal/dog
Model : TFL-CL-001-mobileNetV2
Top 5 detected classes:
golden retriever
Great Pyrenees
clumber, clumber spaniel
soft-coated wheaten terrier
Labrador retriever
Model : TVM-CL-338-mobileNetV2-qat
Top 5 detected classes:
borzoi, Russian wolfhound
golden retriever
Great Pyrenees
soft-coated wheaten terrier
Saint Bernard, St Bernard

Fig. 2.115: Sample output showing single input, mutli-inference output

We can specify the output window location and sizes as shown in the configuration file,

```
flows:
    flow0:
        input: input0
        models: [model0, model1, model2, model3]
        outputs: [output0, output0, output0, output0]
        mosaic:
            mosaic0:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  90
            mosaic1:
                width:  800
                height: 450
                pos_x:  960
                pos_y:  90
            mosaic2:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  540
            mosaic3:
```

(continues on next page)

(continued from previous page)

```
                width:  800
                height: 450
                pos_x:  960
                pos_y:  540
```

Multi input multi inference demo Here is an example of a multi-input, multi-inference demo which takes a camera input and video input and runs multiple networks on each of them.

```
debian@beaglebone:/opt/edge_ai_apps/apps_python# ./app_edgeai.py ../configs/
↪multi_input_multi_infer.yaml
```

Sample output for multi input, multi inference demo is as shown below,

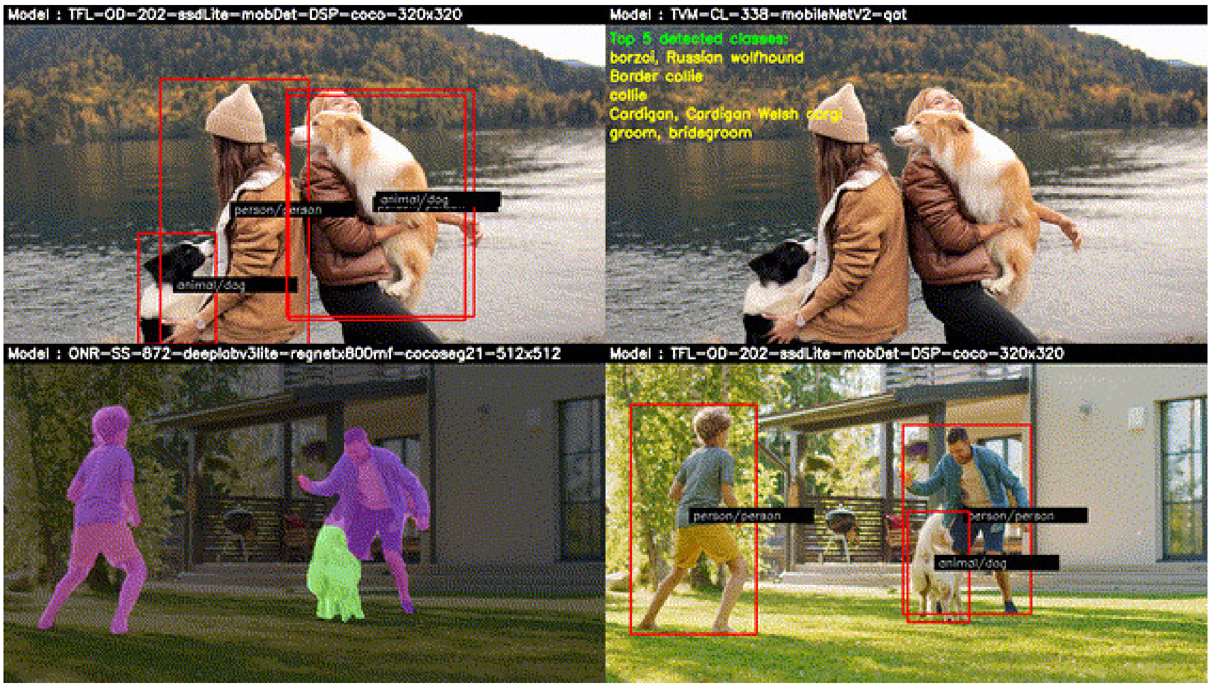

Model : TFL-OD-202-ssdLite-mobDet-DSP-coco-320x320
person/person
animal/dog
animal/dog
Model : TVM-CL-338-mobileNetV2-qat
Top 5 detected classes:
borzoi, Russian wolfhound
Border collie
collie
Cardigan, Cardigan Welsh corgi
groom, bridegroom
Model : ONR-SS-872-deeplabv3lite-regnetx800mf-cocoseg21-512x512
Model : TFL-OD-202-ssdLite-mobDet-DSP-coco-320x320
person/person
person/person
animal/dog

Fig. 2.116: Sample output showing multi-input, mutli-inference output

We can specify the output window location and sizes as shown in the configuration file,

```
flows:
    flow0:
        input: input0
        models: [model1, model2]
        outputs: [output0, output0]
        mosaic:
            mosaic0:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  90
            mosaic1:
                width:  800
                height: 450
                pos_x:  960
                pos_y:  90
    flow1:
        input: input1
        models: [model0, model3]
```

(continues on next page)

(continued from previous page)

```
        outputs: [output0, output0]
        mosaic:
            mosaic0:
                width:  800
                height: 450
                pos_x:  160
                pos_y:  540
            mosaic1:
                width:  800
                height: 450
                pos_x:  960
                pos_y:  540
```

Docker Environment

Docker is a set of "platform as a service" products that uses the OS-level virtualization to deliver software in packages called containers. Docker container provides a quick start environment to the developer to run the out of box demos and build applications.

The Docker image is based on Ubuntu 20.04.LTS and contains different open source components like OpenCV, GStreamer, Python and pip packages which are required to run the demos. The user can choose to install any additional 3rd party applications and packages as required.

Building Docker image The *docker/Dockerfile* in the edge_ai_apps repo describes the recipe for creating the Docker container image. Feel free to review and update it to include additional packages before building the image.

Note: Building Docker image on target using the provided Dockerfile will take about 15-20 minutes to complete with good internet connection. Building Docker containers on target can be slow and resource constrained. The Dockerfile provided will build on target without any issues but if you add more packages or build components from source, running out of memory can be a common problem. As an alternative we highly recommend trying QEMU builds for cross-compiling the images for arm64 architecture on a PC and then load the compiled image on target.

Initiate the Docker image build as shown,

```
debian@beaglebone:/opt/edge_ai_apps/docker#./docker_build.sh
```

Running the Docker container Enter the Docker session as shown,

```
debian@beaglebone:/opt/edge_ai_apps/docker#./docker_run.sh
```

This will start a Ubuntu 20.04.LTS image based Docker container and the prompt will change as below,

```
[docker] debian@beaglebone:/opt/edge_ai_apps#
```

The Docker container has been created in privilege mode, so that it has root capabilities to all devices on the target system like Network etc. The container file system also mounts the target file system of /dev, /opt to access camera, display and other hardware accelerators the SoC has to offer.

Note: It is highly recommended to use the docker_run.sh script to launch the Docker container because this script will take care of saving any changes made to the filesystem. This will make sure that any modifications to the Docker filesystem including new package installation, updates to some files and also command history is saved automatically and is available the next time you launch the container. The container will be committed only if you exit from the container explicitly. If you restart the board without exiting container, any changes done from last saved state will be lost.

Note: After building and running the docker container, one needs to run `setup_script.sh` before running any of the demo applications. Please refer to pub_edgeai_install_dependencies for more details.

Handling proxy settings If the board running the Docker container is behind a proxy server, the default settings for downloading files and installing packages via apt-get will not work. If you are running the board from TI network, docker build and run scripts will automatically detect and configure necessary proxy settings

For other cases, you need to modify the script `/usr/bin/setup_proxy.sh` to add the custom proxy settings required for your network.

Additional Docker commands

Note: This section is provided only for additional reference and not required to run out-of-box demos

Commit Docker container

Generally, containers have a short life cycle. If the container has any local changes it is good to save the changes on top of the existing Docker image. When re-running the Docker image, the local changes can be restored.

Following commands show how to save the changes made to the last container. Note that this is already done automatically by `docker_run.sh` when you exit the container.

```
cont_id=`docker ps -q -l`
docker commit $cont_id edge_ai_kit
docker container rm $cont_id
```

For more information refer: Commit Docker image

Save Docker Image

Docker image can be saved as tar file by using the command below:

```
docker save --output <pre_built_docker_image.tar>
```

For more information refer here. Save Docker image

Load Docker image

Load a previously saved Docker image using the command below:

```
docker load --input <pre_built_docker_image.tar>
```

For more information refer here. Load Docker image

Remove Docker image

Docker image can be removed by using the command below:

```
Remove selected image:
docker rmi <image_name/ID>

Remove all image:
docker image prune -a
```

For more information refer rmi reference and Image prune reference

Remove Docker container

Docker container can be removed by using the command below:

```
Remove selected container:
docker rm <container_ID>

Remove all container:
docker container prune
```

For more information refer here. rm reference and Container Prune reference

Relocating Docker Root Location The default location for Docker files is **/var/lib/docker**. Any Docker images created will be stored here. This will be a problem anytime the SD card is updated with a new targetfs. If a secondary storage (SSD or USB based storage) is available, then it is recommended to relocate the default Docker root location so as to preserve any existing Docker images. Once the relocation has been done, the Docker content will not be affected by any future targetfs updates or accidental corruptions of the SD card.

The following steps outline the process for Docker root directory relocation assuming that the current Docker root is not at the desired location. If the current location is the desired location then exit this procedure.

1. Run 'Docker info' command inspect the output. Locate the line with content **Docker Root Dir**. It will list the current location.
2. To preserve any existing images, export them to .tar files for importing later into the new location.
3. Inspect the content under /etc/docker to see if there is a file by name **daemon.json**. If the file is not present then create **/etc/docker/docker.json** and add the following content. Update the 'key:value' pair for the key "graph" to reflect the desired root location. If the file already exists, then make sure that the line with "graph" exists in the file and points to the desired target location.

```
{
  "graph": "/run/media/nvme0n1/docker_root",
  "storage-driver": "overlay",
  "live-restore": true
}
```

In the configuration above, the key/value pair **'"graph": "/run/media/nvme0n1/docker_root"'** defines the root location **'/run/media/nvme0n1/docker_root'.**

4. Once the daemon.json file has been copied and updated, run the following commands

```
$ systemctl restart docker
$ docker info
```

Make sure that the new Docker root appears under **Docker Root Dir** value.

5. If you exported the existing images in step (2) then import them and they will appear under the new Docker root.
6. Anytime the SD card is updated with a new targetfs, steps (1), (3), and (4) need to be followed.

Additional references

https://docs.docker.com/engine/reference/commandline/images/
https://docs.docker.com/engine/reference/commandline/ps/

Data Flows

The **app_edgeai** application at a high level can be split into 3 parts,

- Input pipeline - Grabs a frame from camera, video, image or RTSP source
- Output pipeline - Sends the output to display or a file

- Compute pipeline - Performs pre-processing, inference and post-processing

Here are the data flows for each reference demo and the corresponding GStreamer launch strings that **app_edgeai** application generates. User can interact with the application via the pub_edgeai_configuration

Image classification In this demo, a frame is grabbed from an input source and split into two paths. The "analytics" path resizes the input maintaining the aspect ratio and crops the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which overlays the detected classes. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 !␣
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=454, height=256 ! tiovxdlcolorconvert␣
↪out-pool-size=4 ! video/x-raw, format=RGB ! videobox left=115 right=115␣
↪top=16 bottom=16 ! tiovxdlpreproc data-type=10 channel-order=0 mean-0=123.
↪675000 mean-1=116.280000 mean-2=103.530000 scale-0=0.017125 scale-1=0.
↪017507 scale-2=0.017429 tensor-format=rgb out-pool-size=4 ! application/x-
↪tensor-tiovx ! appsink name=pre_0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

Object Detection In this demo, a frame is grabbed from an input source and split into two paths. The "analytics" path resizes the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which overlays rectangles around detected objects. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 !␣
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

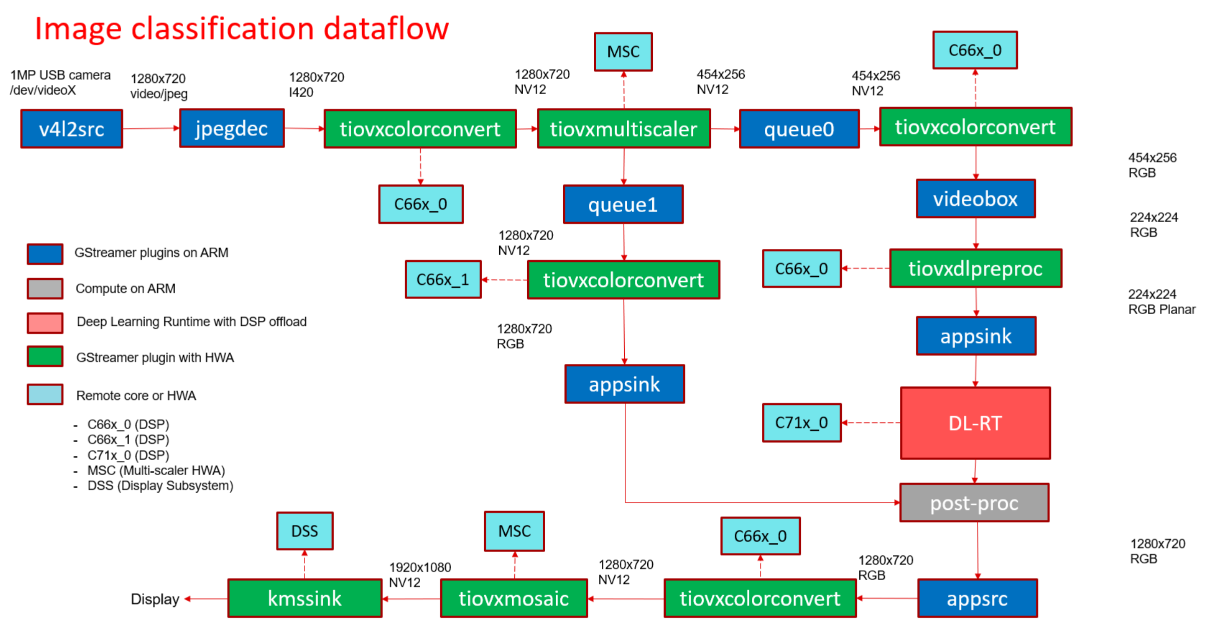


Fig. 2.117: GStreamer based data-flow pipeline for image classification demo with USB camera and display

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

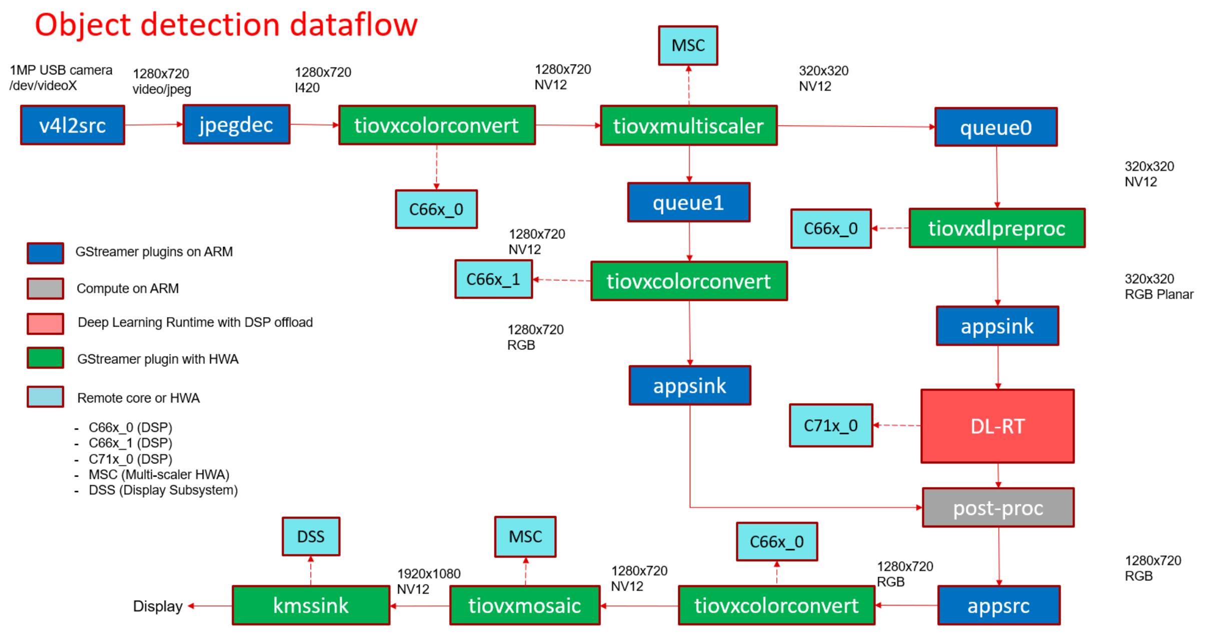


Fig. 2.118: GStreamer based data-flow pipeline for object detection demo with USB camera and display

Semantic Segmentation In this demo, a frame is grabbed from an input source and split into two paths. The "analytics" path resize the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which blends each segmented pixel to a color map. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 !␣
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=512, height=512 ! tiovxdlpreproc data-
↪type=10 channel-order=0 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.015625 scale-1=0.015625 scale-2=0.015625 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

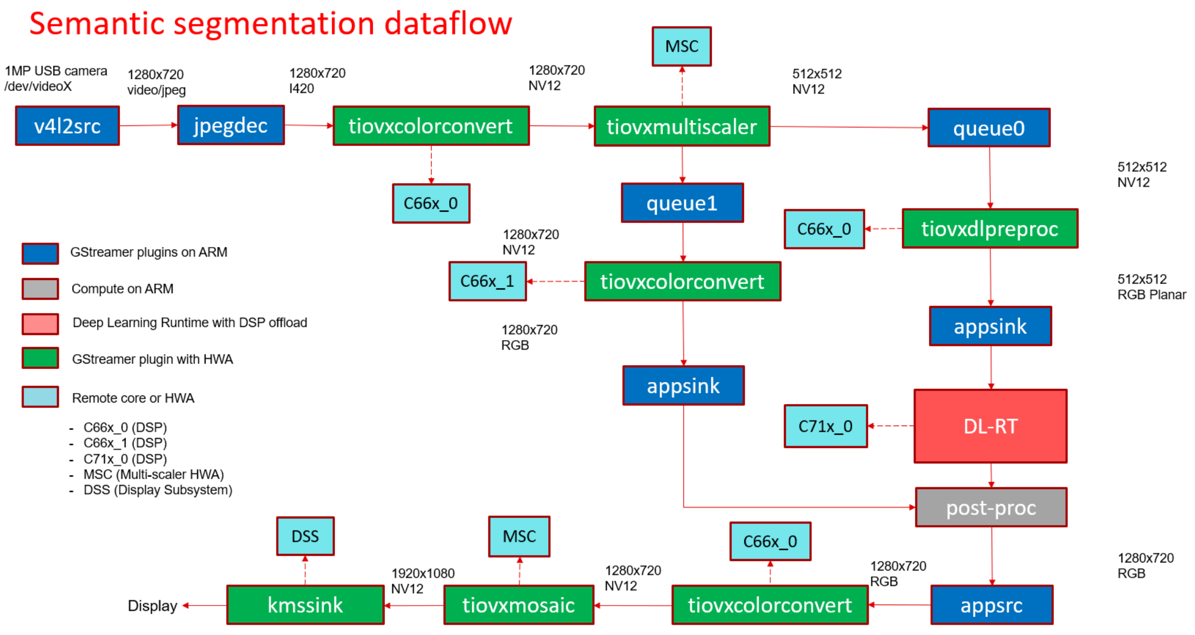


Fig. 2.119: GStreamer based data-flow pipeline for semantic segmentation demo with USB camera and display

Human Pose Estimation In this demo, a frame is grabbed from an input source and split into two paths. The "analytics" path resize the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which overlays the keypoints and lines to draw the pose. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video2 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 !␣
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=640, height=640 ! tiovxdlpreproc data-
↪type=10 target=0 channel-order=0 mean-0=0.000000 mean-1=0.000000 mean-2=0.
↪000000 scale-0=1.000000 scale-1=1.000000 scale-2=1.000000 tensor-
↪format=bgr out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
GStreamer output pipeline:
```

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
tiovxmosaic name=mosaic_0 background=/tmp/background_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

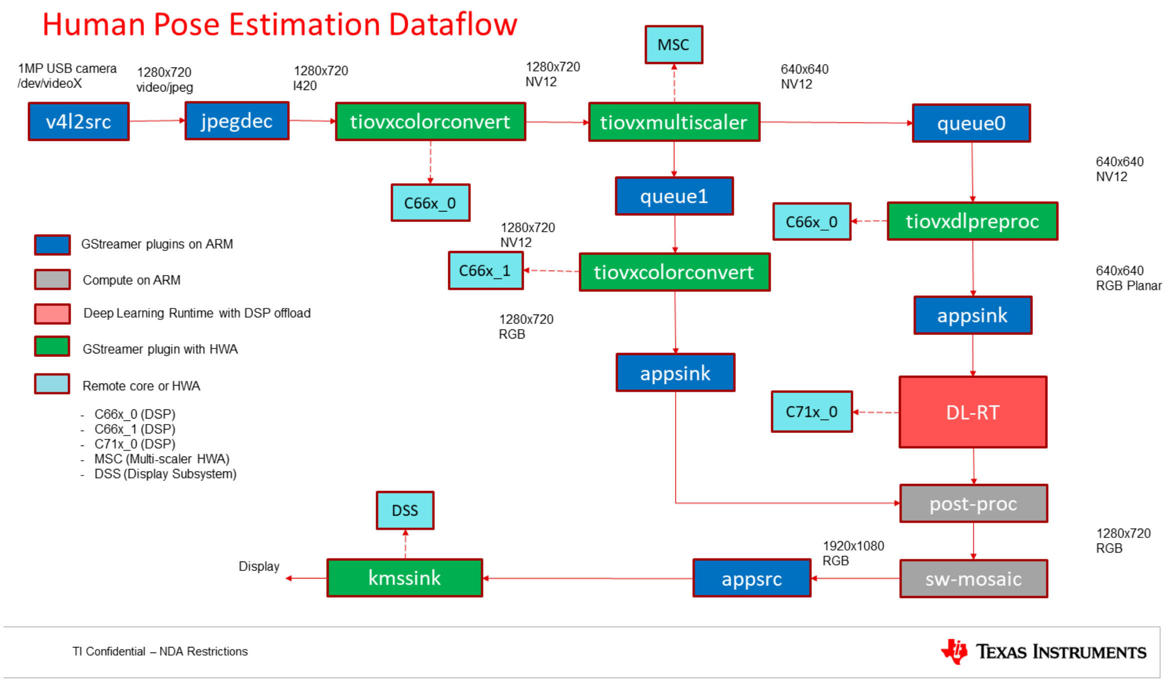


Fig. 2.120: GStreamer based data-flow pipeline for Human Pose Estimation demo with USB camera and display

Video source In this demo, a video file is read from a known location and passed to a de-muxer to extract audio and video streams, the video stream is parsed and raw encoded information is passed to a HW video decoder. Note that H.264 and H.265 encoded videos are supported, making use of the respective HW decoders. The resulting output is split into two paths. The "analytics" path resizes the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which does the required post process required by the model. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
filesrc location=/opt/edge_ai_apps/data/videos/video_0000_h264.mp4 ! qtdemux␣
↪! h264parse ! v4l2h264dec ! video/x-raw, format=NV12  ! tiovxmultiscaler␣
↪name=split_01
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

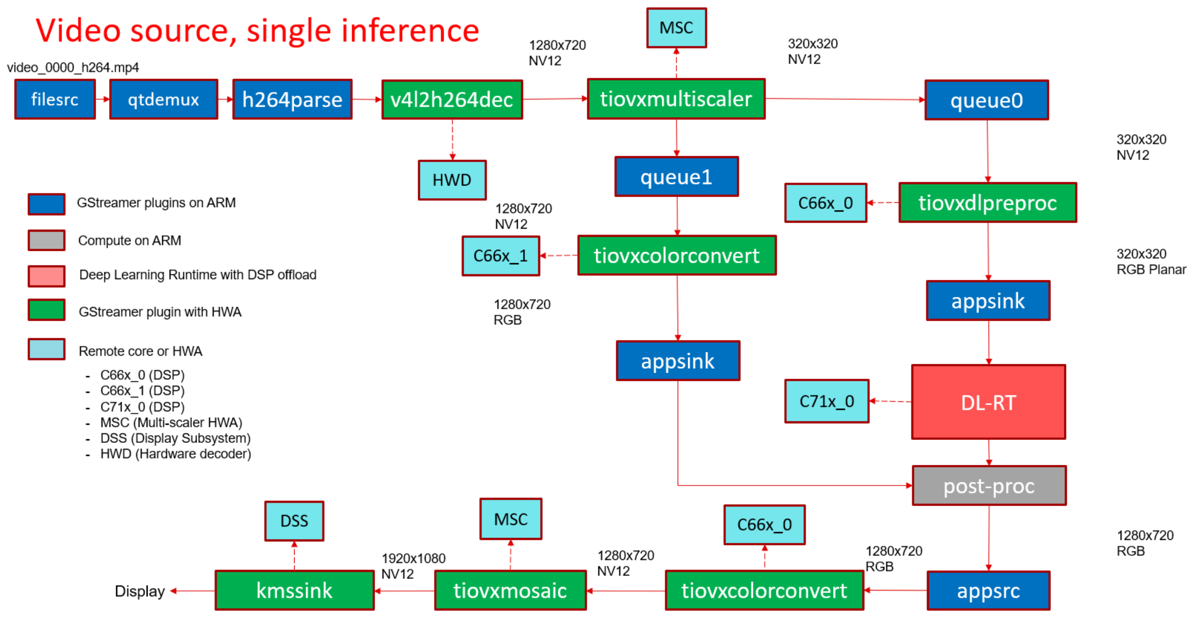


Fig. 2.121: GStreamer based data-flow pipeline with video file input source and display

RTSP source In this demo, a video file is read from a RTSP source and passed to a de-muxer to extract audio and video streams, the video stream is parsed and raw encoded information is passed to a video decoder and the resulting output is split into two paths. The “analytics” path resizes the input to match the resolution required to run the deep learning network. The “visualization” path is provided to the post-processing module which does the required post process required by the model. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
rtspsrc location=rtsp://172.24.145.220:8554/test latency=0 buffer-mode=auto !
↪ rtph264depay ! h264parse ! v4l2h264dec ! video/x-raw, format=NV12 !
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

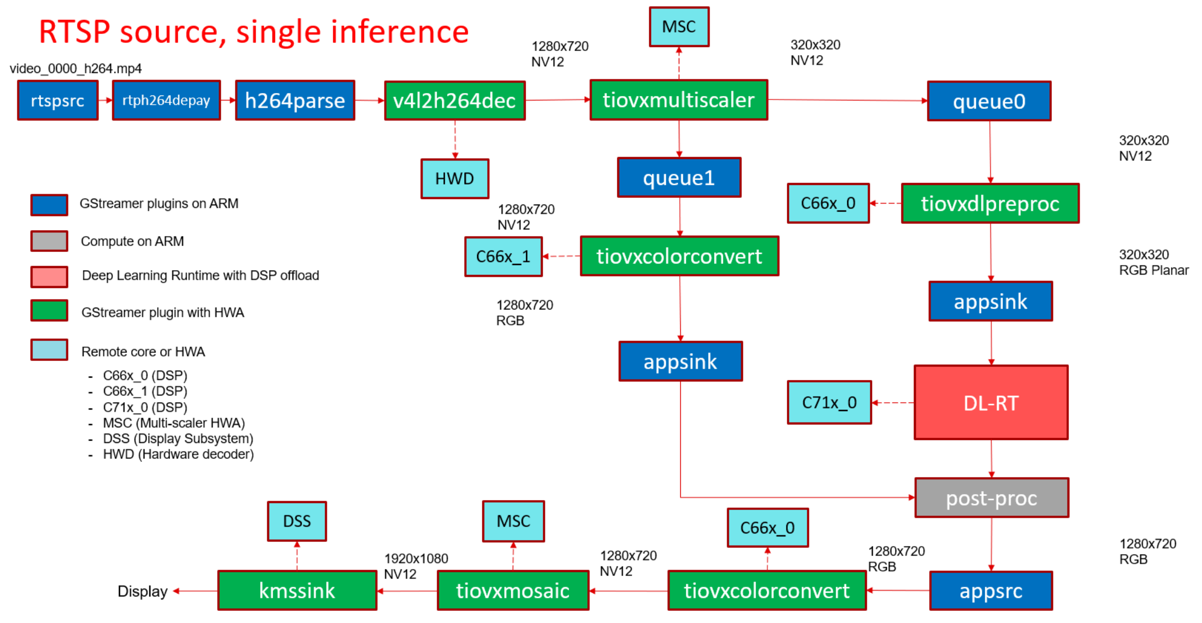


Fig. 2.122: GStreamer based data-flow pipeline with RTSP based video file source and display

RPiV2 Camera Sensor (IMX219) In this demo, raw frames in SRGGB8 format are captured form RPiV2 (imx219) camera sensor. VISS (Vision Imaging Subsystem) is used to process the raw frames and get the output in NV12, VISS also cotrols the sensor parameters like exposure, gain etc.. via v4l2 ioctls. The NV12 output is split into two paths. The "analytics" path resizes the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which does the required post process required by the model. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video2 io-mode=5 ! video/x-bayer, width=1920,␣
↪height=1080, format=rggb ! tiovxisp device=/dev/v4l-subdev2 dcc-isp-file=/
↪opt/imaging/imx219/dcc_viss.bin dcc-2a-file=/opt/imaging/imx219/dcc_2a.bin␣
↪format-msb=7 ! video/x-raw, format=NV12 ! tiovxmultiscaler ! video/x-raw,␣
↪width=1280, height=720 ! tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

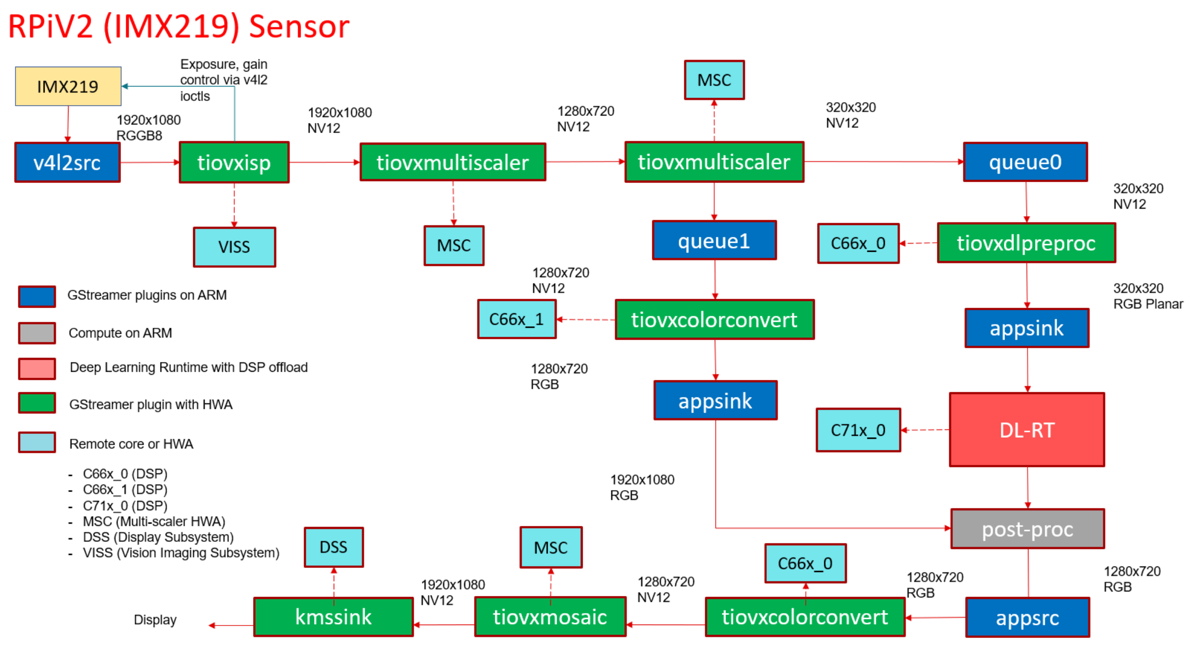


Fig. 2.123: GStreamer based data-flow pipeline with IMX219 sensor, ISP and display

IMX390 Camera Sensor In this demo, raw frames in SRGGB12 format are captured from IMX390 camera sensor. VISS (Vision Imaging Subsystem) is used to process the raw frames and get the output in NV12, VISS also controls the sensor parameters like exposure, gain etc.. via v4l2 ioctls. This is followed by LDC (Lens Distortion Correction) required due to the fisheye lens. The NV12 output is split into two paths. The "analytics" path resizes the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which does the required post process required by the model. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 ! queue leaky=2 ! video/x-bayer, width=1936,␣
↪height=1100, format=rggb12 ! tiovxisp sink_0::device=/dev/v4l-subdev7␣
↪sensor-name=IMX390-UB953_D3 dcc-isp-file=/opt/imaging/imx390/dcc_viss.bin␣
↪sink_0::dcc-2a-file=/opt/imaging/imx390/dcc_2a.bin format-msb=11 ! video/x-
↪raw, format=NV12 ! tiovxldc dcc-file=/opt/imaging/imx390/dcc_ldc.bin␣
↪sensor-name=IMX390-UB953_D3 ! video/x-raw, format=NV12, width=1920,␣
↪height=1080 !tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=512, height=512 ! tiovxdlpreproc data-
↪type=3 target=0 channel-order=0 tensor-format=bgr out-pool-size=4 !␣
↪application/x-tensor-tiovx ! appsink name=pre_0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
tiovxmosaic name=mosaic_0 background=/tmp/background_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

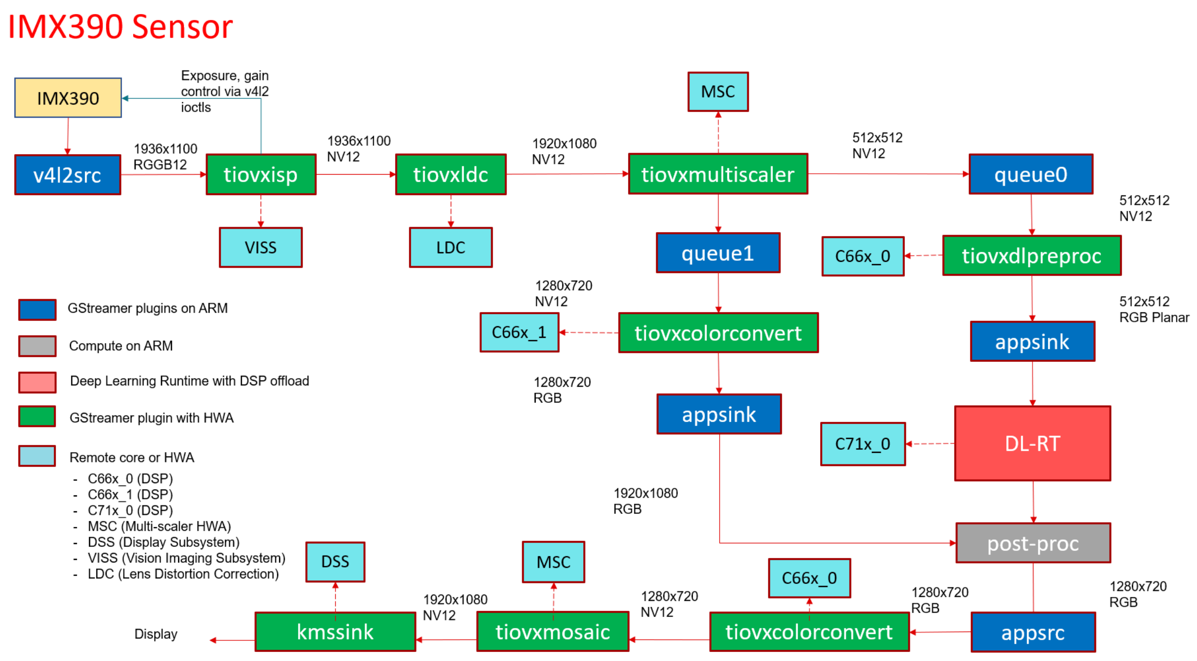


Fig. 2.124: GStreamer based data-flow pipeline with IMX390 sensor, ISP, LDC and display

Video output In this demo, a frame is grabbed from an input source and split into two paths. The "analytics" path resizes the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which does the required post process required by the model. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background. Finally the video is encoded using the H.264 HW encoder and written to a video file.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 !␣
↪tiovxmultiscaler name=split_01
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=1280, height=720 !␣
↪tiovxdlcolorconvert target=1 out-pool-size=4 ! video/x-raw, format=RGB !␣
↪appsink name=sen_0 max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1280,␣
↪height=720 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=1280   sink_
↪0::height=720
! video/x-raw,format=NV12, width=1920, height=1080 ! v4l2h264enc␣
↪bitrate=10000000 ! h264parse ! matroskamux !  filesink location=/opt/edge_
↪ai_apps/data/output/videos/output_video.mkv
```

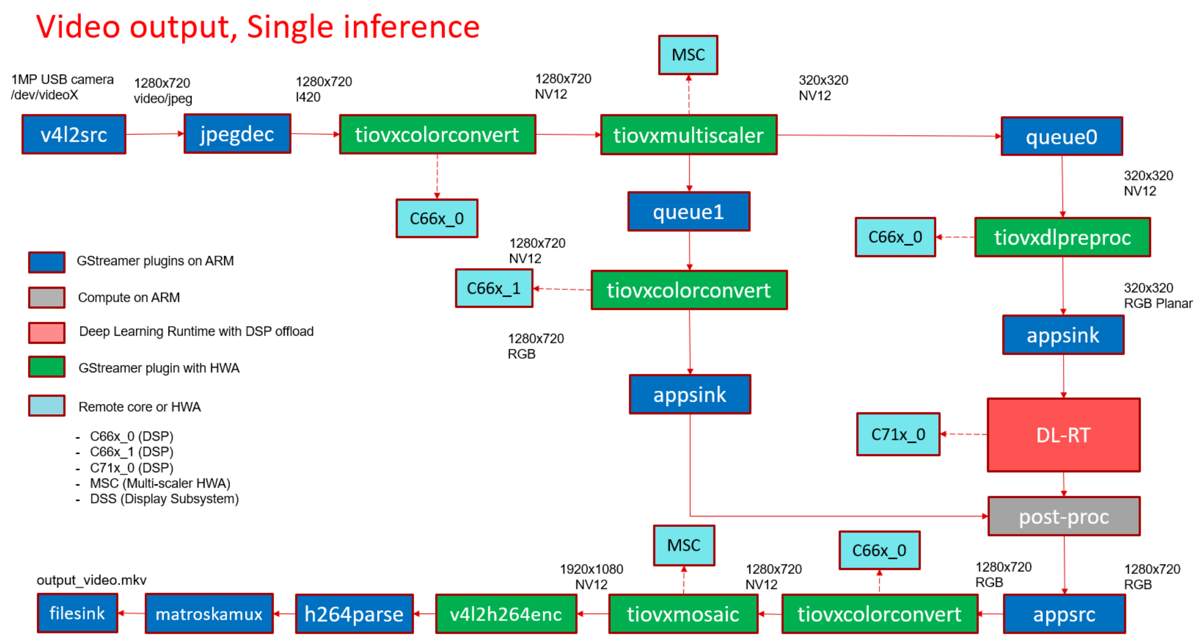


Fig. 2.125: GStreamer based data-flow pipeline with video file input source and display

Single Input Multi inference In this demo, a frame is grabbed from an input source and split into multiple paths. Each path is further split into two sub-paths one for analytics and another for visualization. Each path can run any type of network, image classification, object detection, semantic segmentation and using any supported run-time.

For example the below GStreamer pipeline splits the input into 4 paths for running 4 deep learning networks. First is a semantic segmentation network, followed by object detection network, followed by two image classification networks. If we look at the image classification path, the analytics sub-path resizes the input to maintain the aspect ratio and crops the input to match the resolution required to run the deep learning network. The visualization sub-path is provided to the post-processing module which overlays the detected classes. Post-processed output from all the 4 paths is given to HW mosaic plugin which positions and resizes the output windows on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 ! tee name=tee_
↪split0
tee_split0. ! queue ! tiovxmultiscaler name=split_01
tee_split0. ! queue ! tiovxmultiscaler name=split_02
tee_split0. ! queue ! tiovxmultiscaler name=split_03
tee_split0. ! queue ! tiovxmultiscaler name=split_04
split_01. ! queue ! video/x-raw, width=512, height=512 ! tiovxdlpreproc data-
↪type=10 channel-order=0 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.015625 scale-1=0.015625 scale-2=0.015625 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_0␣
↪max-buffers=2 drop=true
split_02. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪1 max-buffers=2 drop=true
split_02. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_1␣
↪max-buffers=2 drop=true
split_03. ! queue ! video/x-raw, width=454, height=256 ! tiovxdlcolorconvert␣
↪out-pool-size=4 ! video/x-raw, format=RGB ! videobox left=115 right=115␣
↪top=16 bottom=16 ! tiovxdlpreproc data-type=10 channel-order=1 mean-0=128.
↪000000 mean-1=128.000000 mean-2=128.000000 scale-0=0.007812 scale-1=0.
↪007812 scale-2=0.007812 tensor-format=rgb out-pool-size=4 ! application/x-
↪tensor-tiovx ! appsink name=pre_2 max-buffers=2 drop=true
split_03. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_2␣
↪max-buffers=2 drop=true
split_04. ! queue ! video/x-raw, width=454, height=256 ! tiovxdlcolorconvert␣
↪out-pool-size=4 ! video/x-raw, format=RGB ! videobox left=115 right=115␣
↪top=16 bottom=16 ! tiovxdlpreproc data-type=10 channel-order=0 mean-0=123.
↪675000 mean-1=116.280000 mean-2=103.530000 scale-0=0.017125 scale-1=0.
↪017507 scale-2=0.017429 tensor-format=rgb out-pool-size=4 ! application/x-
↪tensor-tiovx ! appsink name=pre_3 max-buffers=2 drop=true
split_04. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_3␣
↪max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_1 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_1
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_2 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_2
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_3 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_3
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
```

(continues on next page)

(continued from previous page)

```
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=640   sink_
↪0::height=360
sink_1::startx=960  sink_1::starty=180  sink_1::width=640   sink_
↪1::height=360
sink_2::startx=320  sink_2::starty=560  sink_2::width=640   sink_
↪2::height=360
sink_3::startx=960  sink_3::starty=560  sink_3::width=640   sink_
↪3::height=360
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

Multi Input Multi inference In this demo, a frame is grabbed from multiple input sources and split into multiple paths. The multiple input sources could be either multiple cameras or a combination of camera, video, image, RTSP source. Each path is further split into two sub-paths one for analytics and another for visualization. Each path can run any type of network, image classification, object detection, semantic segmentation and using any supported run-time.

For example the below GStreamer pipeline splits two inputs into 4 paths for running 2 deep learning networks. First is a object detection network, followed by image classification networks. If we look at the image classification path, the analytics sub-path resizes the input to maintain the aspect ratio and crops the input to match the resolution required to run the deep learning network. The visualization sub-path is provided to the post-processing module which overlays the detected classes. Post-processed output from all the 4 paths is given to HW mosaic plugin which positions and resizes the output windows on an empty background before sending to display.

GStreamer input pipeline:

```
v4l2src device=/dev/video18 io-mode=2 ! image/jpeg, width=1280, height=720 !␣
↪jpegdec ! tiovxdlcolorconvert ! video/x-raw, format=NV12 ! tee name=tee_
↪split0
tee_split0. ! queue ! tiovxmultiscaler name=split_01
tee_split0. ! queue ! tiovxmultiscaler name=split_02
split_01. ! queue ! video/x-raw, width=320, height=320 ! tiovxdlpreproc data-
↪type=10 channel-order=1 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.007812 tensor-
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪0 max-buffers=2 drop=true
split_01. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_0␣
↪max-buffers=2 drop=true
split_02. ! queue ! video/x-raw, width=454, height=256 ! tiovxdlcolorconvert␣
↪out-pool-size=4 ! video/x-raw, format=RGB ! videobox left=115 right=115␣
↪top=16 bottom=16 ! tiovxdlpreproc data-type=10 channel-order=1 mean-0=128.
↪000000 mean-1=128.000000 mean-2=128.000000 scale-0=0.007812 scale-1=0.
↪007812 scale-2=0.007812 tensor-format=rgb out-pool-size=4 ! application/x-
↪tensor-tiovx ! appsink name=pre_1 max-buffers=2 drop=true
split_02. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_1␣
↪max-buffers=2 drop=true

filesrc location=/opt/edge_ai_apps/data/videos/video_0000_h264.mp4 ! qtdemux␣
↪! h264parse ! v4l2h264dec ! video/x-raw, format=NV12  ! tee name=tee_split1
tee_split1. ! queue ! tiovxmultiscaler name=split_11
tee_split1. ! queue ! tiovxmultiscaler name=split_12
split_11. ! queue ! video/x-raw, width=512, height=512 ! tiovxdlpreproc data-
↪type=10 channel-order=0 mean-0=128.000000 mean-1=128.000000 mean-2=128.
↪000000 scale-0=0.015625 scale-1=0.015625 scale-2=0.015625 tensor-
```

(continues on next page)

(continued from previous page)

```
↪format=rgb out-pool-size=4 ! application/x-tensor-tiovx ! appsink name=pre_
↪2 max-buffers=2 drop=true
split_11. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_2␣
↪max-buffers=2 drop=true
split_12. ! queue ! video/x-raw, width=454, height=256 ! tiovxdlcolorconvert␣
↪out-pool-size=4 ! video/x-raw, format=RGB ! videobox left=115 right=115␣
↪top=16 bottom=16 ! tiovxdlpreproc data-type=10 channel-order=0 mean-0=123.
↪675000 mean-1=116.280000 mean-2=103.530000 scale-0=0.017125 scale-1=0.
↪017507 scale-2=0.017429 tensor-format=rgb out-pool-size=4 ! application/x-
↪tensor-tiovx ! appsink name=pre_3 max-buffers=2 drop=true
split_12. ! queue ! video/x-raw, width=640, height=360 ! tiovxdlcolorconvert␣
↪target=1 out-pool-size=4 ! video/x-raw, format=RGB ! appsink name=sen_3␣
↪max-buffers=2 drop=true
```

GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_0
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_1 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_1
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_2 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_2
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_3 ! tiovxdlcolorconvert ! video/x-raw,format=NV12, width=640,␣
↪height=360 ! queue ! mosaic_0.sink_3
appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 !␣
↪tiovxdlcolorconvert ! video/x-raw,format=NV12, width=1920, height=1080 !␣
↪queue ! mosaic_0.background
tiovxmosaic name=mosaic_0
sink_0::startx=320  sink_0::starty=180  sink_0::width=640   sink_
↪0::height=360
sink_1::startx=960  sink_1::starty=180  sink_1::width=640   sink_
↪1::height=360
sink_2::startx=320  sink_2::starty=560  sink_2::width=640   sink_
↪2::height=360
sink_3::startx=960  sink_3::starty=560  sink_3::width=640   sink_
↪3::height=360
! video/x-raw,format=NV12, width=1920, height=1080 ! kmssink sync=false␣
↪driver-name=tidss
```

Performance Visualization Tool

The performance visualization tool can be used to view all the performance statistics recorded when running the edge AI C++ demo application. This includes the CPU and HWA loading, DDR bandwidth, Junction Temperatures and FPS obtained. Refer to pub_edgeai_available_statistics for details on the performance metrics available to be plotted.

This tool works as follows:

- Logging: When running the application, the performance statistics can be recorded and stored in log files. This is done automatically when running the C++ application, but the Python application does not generate logs. However a standalone binary executable is provided that can be run in parallel with the Python application, which will generate these performance logs.
- Visualization: There is a Python script which parses these logs and plots graphs, which can be easily viewed by a visiting a URL in any browser. This script uses Streamlit package to update the graphs in real-time, as the Edge AI application runs in parallel. However, since Streamlit is not supported in the

SDK out of box, this script needs to run on docker. Please refer to pub_edgeai_docker_env for building and running a docker container.

Generating Performance Logs

Each log file contains real-time values for some performance metrics, averaged over a 2s window. The temperature sensor values are sampled in real time, every 2s. The performance visualization tool then parses these log files one by one based on the modification timestamps.

The edge AI C++ demo will automatically generate log files and store them in the directory `../perf_logs`, that is, one level up from where the C++ app is run. For example, if the app is run from `edge_ai_apps/apps_cpp`, the logs will be stored in `edge_ai_apps/perf_logs`.

Similarly, there is a binary executable that can be compiled that does the same logging standalone. The source for this is available under `edge_ai_apps/scripts/perf_stats/`. The README.md file has simple instructions to build and run this standalone logger binary. After building it, use following command to print the statistics on the terminal as well as save them in log files that can be parsed.

```
debian@beaglebone:/opt/edge_ai_apps/scripts/perf_stats/build# ../bin/Release/
↪ti_perfstats -l
```

Running the Visualization tool

To use this tool, simply start a docker session and then run the command given below. This script expects some log files to be present in the directory `edge_ai_apps/perf_logs` after running any C++ demo. One can also bring up this tool while running the demo but it might affect the performance of the demo itself as it consumes a bit of ARM cycles during launch but stabilizes over a certain duration.

```
[docker] debian@beaglebone:/opt/edge_ai_apps# streamlit run scripts/perf_vis.
↪py --theme.base="light"
```

This script also accepts the log directory as a command line argument as follows:

```
[docker] debian@beaglebone:/opt/edge_ai_apps# streamlit run scripts/perf_vis.
↪py --theme.base="light" -- -D <path/to/logs/directory/>
```

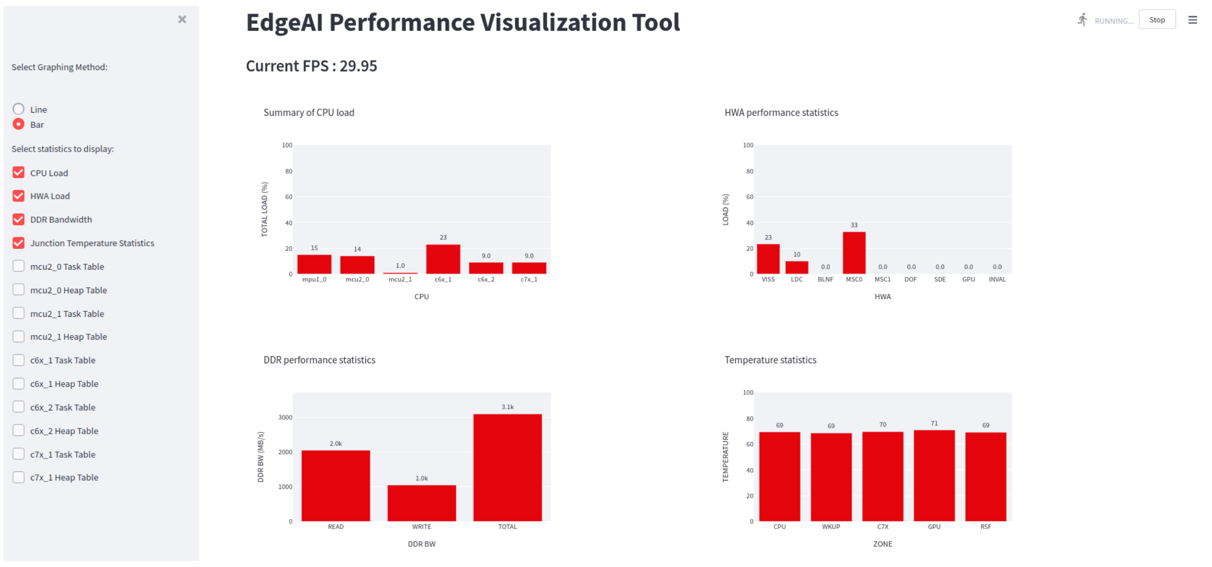
A network URL can be seen in the terminal output. The graphs can be viewed by visiting this URL in any browser. The plotted graphs will keep updating based on the available log files.

To exit press Ctrl+C in the terminal.

Available options Average frames per second (FPS) recorded by the application is displayed by default. Using the checkboxes in the sidebar, one can select which performance metrics to view. There are 14 metrics available to be plotted, as seen from the above image:

- CPU Load: Total loading for the A72(mpu1_0), R5F(mcu2_0/1), C66x(c6x_1/2) and C71x(c7x_1) DSPs.
- HWA Load: Loading (percentage) for the various available hardware accelerators.
- DDR Bandwidth: Average read, write and total bandwidth recorded in the previous 2s interval.
- Junction Temperatures: The live temperatures recorded at various junctions
- Task Table: A separate graph for each cpu showing the loading due to various tasks running on it.
- Heap Table: A separate graph for each cpu showing the heap memory usage statistics.

For the first three metrics, there is a choice to view line graphs with a 30s history or bar graphs with only the real-time values. The remaining eleven have real-time bar graphs as the only option.



EdgeAI Performance Visualization Tool

Current FPS : 29.95

Select Graphing Method:

( ) Line
(●) Bar

Select statistics to display:

- [x] CPU Load
- [x] HWA Load
- [x] DDR Bandwidth
- [x] Junction Temperature Statistics
- [ ] mcu2_0 Task Table
- [ ] mcu2_0 Heap Table
- [ ] mcu2_1 Task Table
- [ ] mcu2_1 Heap Table
- [ ] c6x_1 Task Table
- [ ] c6x_1 Heap Table
- [ ] c6x_2 Task Table
- [ ] c6x_2 Heap Table
- [ ] c7x_1 Task Table
- [ ] c7x_1 Heap Table

Fig. 2.126: Performance visualizer dashboard showing CPU and HWA loading, DDR bandwidth, Junction Temperatures and the FPS obtained

SDK Components

The BeagleBone® AI-64 Linux for Edge AI can be divided into 3 parts, Applications, BeagleBone® AI-64 Linux and Processor SDK RTOS. Users can get the latest application updates and bug fixes from the public repositories (GitHub and git.ti.com) which aligns with the SDK releases done quarterly. One can also build every component from source by following the steps here, pub_edgeai_sdk_development_flow

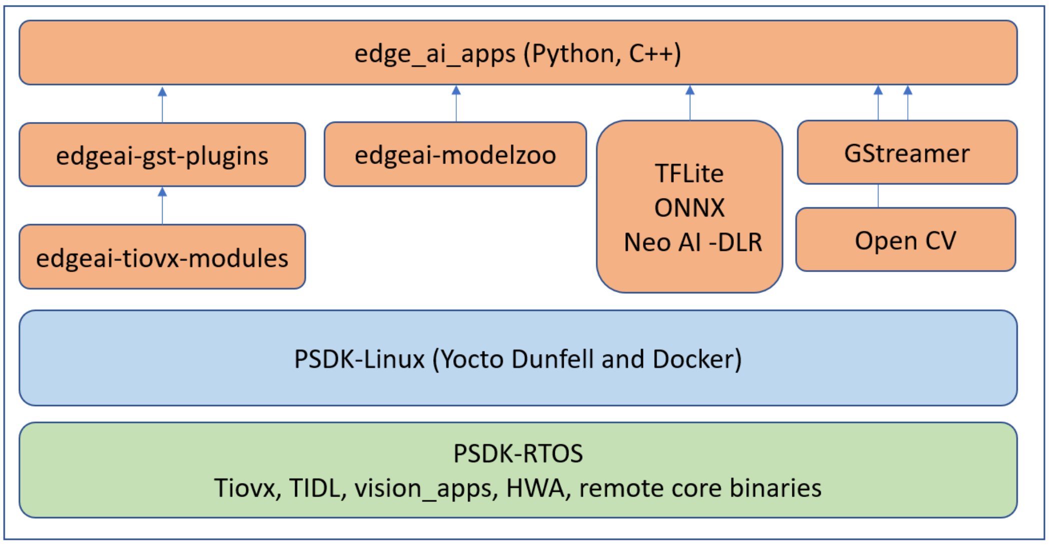

edge_ai_apps (Python, C++)
edgeai-gst-plugins
edgeai-modelzoo
TFLite
ONNX
Neo AI -DLR
GStreamer
Open CV
edgeai-tiovx-modules
PSDK-Linux (Yocto Dunfell and Docker)
PSDK-RTOS
Tiovx, TIDL, vision_apps, HWA, remote core binaries

Fig. 2.127: BeagleBone® AI-64 Linux for Edge AI components

Edge AI Applications The edge AI applications are designed for users to quickly evaluate various Deep Learning networks on TDA4 SoC. The user can run standalone examples and Jupyter notebook applications to evaluate inference models either from TI Edge AI Model Zoo or a custom network. Once a network is finalized for performance and accuracy it can also be easily integrated in a typical capture-inference-display usecase using example GStreamer based applications for rapid prototyping and deployment.

edgeai-tidl-tools This application repository provides standalone Python and C/C++ examples to quickly evaluate inference models using TFLite, ONNX and NeoAI-DLR runtime using file based inputs. It also houses the Jupyter notebooks similar to TI Edge AI Cloud which can be executed right on the TDA4VM Starter Kit.

For more details on using this application repo please refer to the documentation and source code found here: https://github.com/TexasInstruments/edgeai-tidl-tools

edgeai-modelzoo This repo provides collection of example Deep Neural Network (DNN) Models for various computer vision tasks. A few example models are packaged as part of the SDK to run out-of-box demos. More can be downloaded using a download script made available in the edge_ai_apps repo.

For more details on the pre-imported models and related documentation please visit: https://github.com/TexasInstruments/edgeai-modelzoo

edge_ai_apps These are plug-and-play Deep Learning applications which support running open source runtime frameworks such as TFLite, ONNX and NeoAI-DLR with a live camera and display. They help connect realtime camera, video or RTSP sources to DL inference to live display, bitstream or RTSP sinks.

The latest source code with fixes can be pulled from: https://git.ti.com/cgit/edgeai/edge_ai_apps

edgeai-gst-plugins This repo provides the source of custom GStreamer plugins which helps offload tasks to TDA4 hardware accelerators and advanced DSPs with the help of edgeai-tiovx-modules. The repo gets downloaded, built and installed as part of the pub_edgeai_install_dependencies step.

Source code and documentation: https://github.com/TexasInstruments/edgeai-gst-plugins

edgeai-tiovx-modules This repo provides OpenVx modules which help access underlying hardware accelerators in the TDA4 SoC and serves as a bridge between GStreamer custom elements and underlying OpenVx custom kernels. The repo gets downloaded, built and installed as part of the pub_edgeai_install_dependencies step.

Source code and documentation: https://github.com/TexasInstruments/edgeai-tiovx-modules

Processor SDK RTOS The BeagleBone® AI-64 Linux for Edge AI gets all the HWA drivers, optimized libraries, OpenVx framework and more from Processor SDK RTOS

For more information visit Processor SDK RTOS Getting Started Guide.

BeagleBone® AI-64 Linux The BeagleBone® AI-64 Linux for Edge AI gets all the Linux kernel, filesystem, device-drivers and more from BeagleBone® AI-64 Linux

For more information visit BeagleBone® AI-64 Linux Software Developer's Guide.

Datasheet

This chapter describes the performance measurements of the Edge AI Inference demos.

Performance data of the demos can be auto generated by running following command on target:

```
debian@beaglebone:/opt/edge_ai_apps/tests# ./gen_data_sheet.sh
```

The performance measurements includes the following

1. **FPS** : Effective framerate at which the application runs
2. **Total time** : Average time taken to process each frame, which includes pre-processing, inference and post-processing time
3. **Inference time** : Average time taken to infer each frame

4. **CPU loading** : Loading on different CPU cores present
5. **DDR BW** : DDR read and write BW used
6. **HWA Loading** : Loading on different Hardware accelerators present

Following are the latest performance numbers of the C++ demos:

Source : USB Camera Capture Framerate : **30 fps** Resolution : **720p** format : **JPEG**

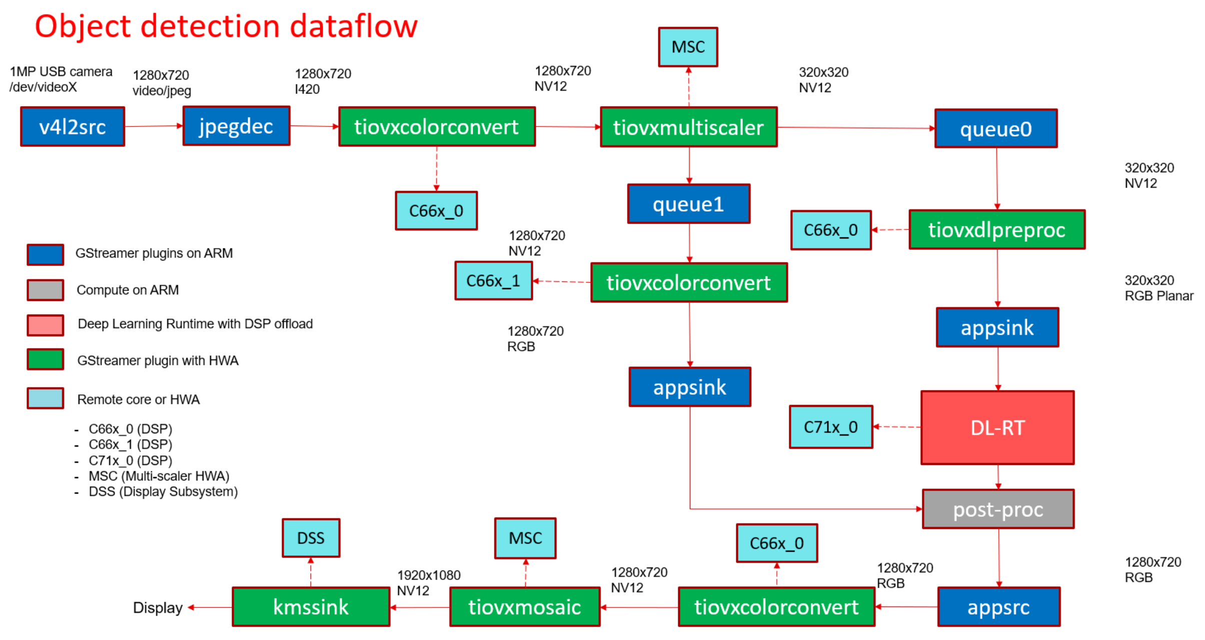


Fig. 2.128: GStreamer based data-flow pipeline with USB camera input and display output

Model	FPS	Total time (ms)	Inference time (ms)	A72 Load (%)	DDR Read BW (MB/s)	DDR Write BW (MB/s)	DDR Total BW (MB/s)	C71 Load (%)	C66_1 Load (%)	C66_2 Load (%)	MCU2_0 Load (%)	MCU2_1 Load (%)	MSC_0 (%)	MSC_1 (%)	VISS (%)	NF (%)	LDC (%)	SDE (%)	DOF (%)
ONR-CL-6150-mobileNetV2-1p4-qat	30.80	33.22	3.02	21.60	1596	619	2215	9.0	20.0	9.0	6.0	1.0	22.17	0	0	0	0	0	0
TFL-CL-0000-mobileNetV1-mlperf	30.69	33.19	1.04	15.93	1425	563	1988	5.0	22.0	9.0	6.0	1.0	21.90	0	0	0	0	0	0
TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320	30.69	33.25	5.00	10.24	1534	570	2104	15.0	29.0	9.0	6.0	1.0	22.67	0	0	0	0	0	0
TVM-CL-3410-gluoncv-mxnet-mobv2	30.58	33.21	2.02	22.80	1522	617	2139	6.0	20.0	9.0	6.0	1.0	21.84	0	0	0	0	0	0

Source : Video Video Framerate : **30 fps** Resolution : **720p** Encoding : **h264**

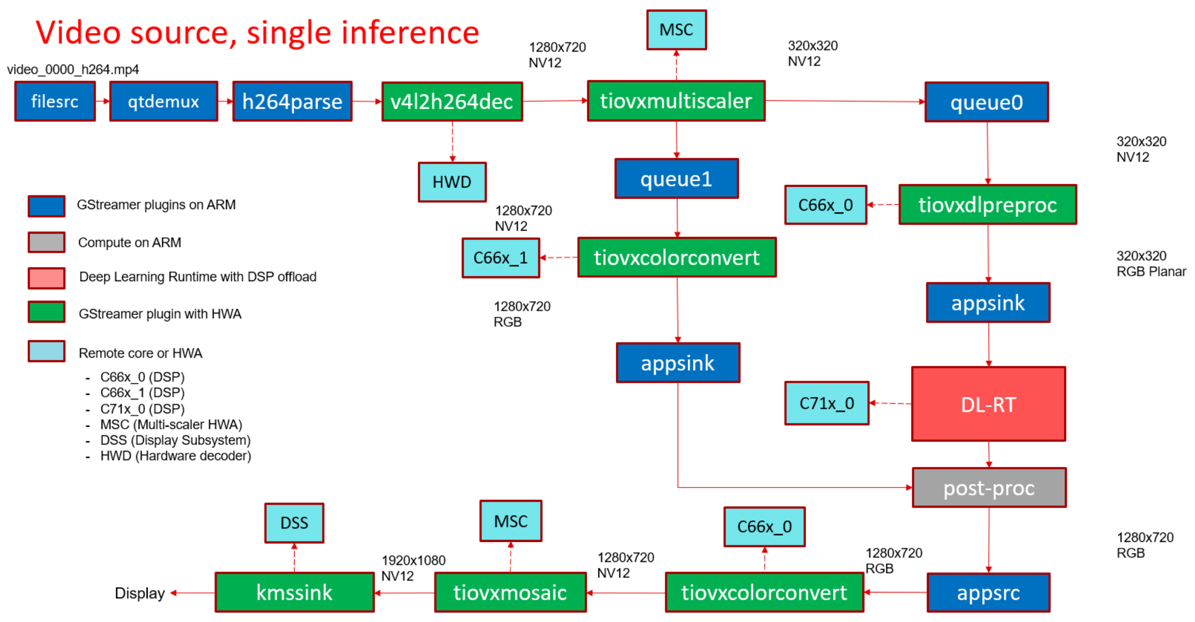


Fig. 2.129: GStreamer based data-flow pipeline with video file input source and display output

Model	FPS	Total time (ms)	Inference time (ms)	A72 Load (%)	DDR Read BW (MB/s)	DDR Write BW (MB/s)	DDR Total BW (MB/s)	C71 Load (%)	C66_1 Load (%)	C66_2 Load (%)	MCU2_0 Load (%)	MCU2_1 Load (%)	MSC_0 (%)	MSC_1 (%)	VISS (%)	NF (%)	LDC (%)	SDE (%)	DOF (%)
ONR-CL-6150-mobileNetV2-1p4-qat	30.52	33.46	3.03	14.28	990	403	1393	2.0	7.0	4.0	1.0	1.0	10.27	0	0	0	0	0	0
TFL-CL-0000-mobileNetV1-mlperf	30.77	33.47	1.07	30.76	746	97	843	2.0	2.0	1.0	1.0	1.0	15.76	0	0	0	0	0	0
TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320	30.56	33.54	5.06	22.58	736	92	828	2.0	2.0	1.0	1.0	1.0	16.9	0	0	0	0	0	0
TVM-CL-3410-gluoncv-mxnet-mobv2	30.64	33.47	2.01	33.33	712	110	822	1.0	1.0	0.0	1.0	1.0	15.3	0	0	0	0	0	0

Source : CSI Camera (ov5640) Capture Framerate : **30 fps** Resolution : **720p** format : **YUYV**

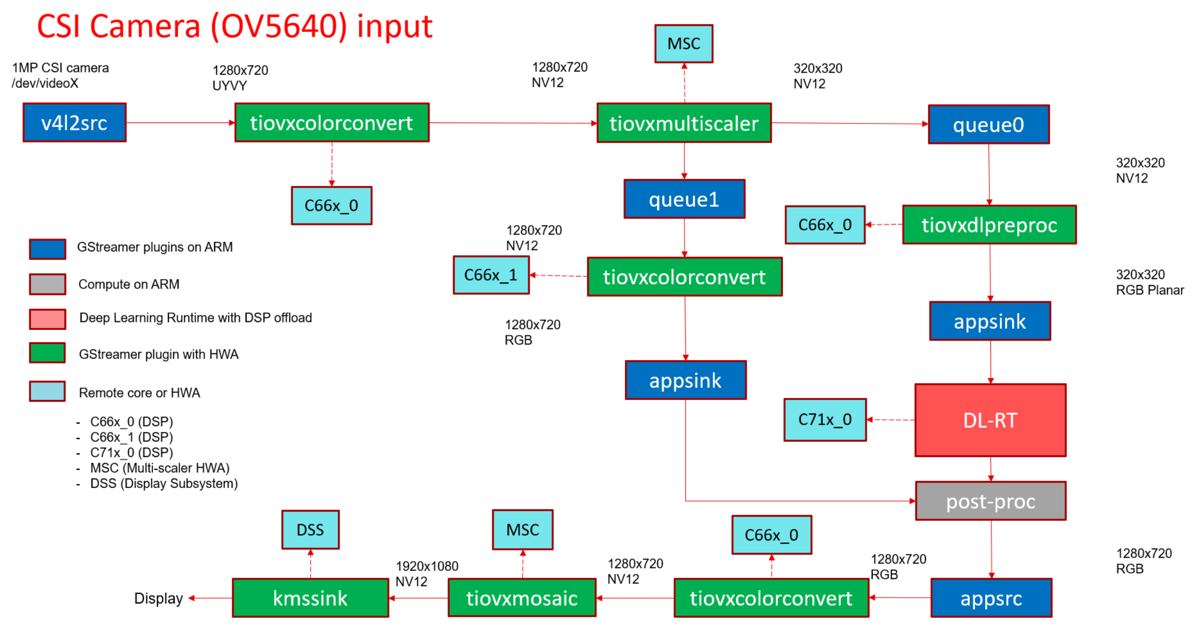


Fig. 2.130: GStreamer based data-flow pipeline for with CSI camera (OV5640) input and display output

Model	FPS	Total time (ms)	Inference time (ms)	A72 Load (%)	DDR Read BW (MB/s)	DDR Write BW (MB/s)	DDR Total BW (MB/s)	C71 Load (%)	C66_1 Load (%)	C66_2 Load (%)	MCU2_0 Load (%)	MCU2_1 Load (%)	MSC_0 (%)	MSC_1 (%)	VISS (%)	NF (%)	LDC (%)	SDE (%)	DOF (%)
ONR-CL-6150-mobileNetV2-1p4-qat	29.57	34.09	3.02	12.21	1671	699	2370	8.0	45.0	9.0	6.0	1.0	21.35	0	0	0	0	0	0
TFL-CL-0000-mobileNetV1-mlperf	29.41	34.15	1.01	10.27	1502	645	2147	5.0	47.0	9.0	6.0	1.0	20.96	0	0	0	0	0	0
TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320	29.36	34.65	5.00	10.5	1610	655	2265	14.0	53.0	9.0	6.0	1.0	21.47	0	0	0	0	0	0
TVM-CL-3410-gluoncv-mxnet-mobv2	29.38	34.17	2.01	11.66	1596	698	2294	6.0	45.0	9.0	5.0	1.0	21.10	0	0	0	0	0	0

Source : CSI Camera with VISS (imx219) Capture Framerate : **30 fps** Resolution : **1080p** format : **SRGGB8**

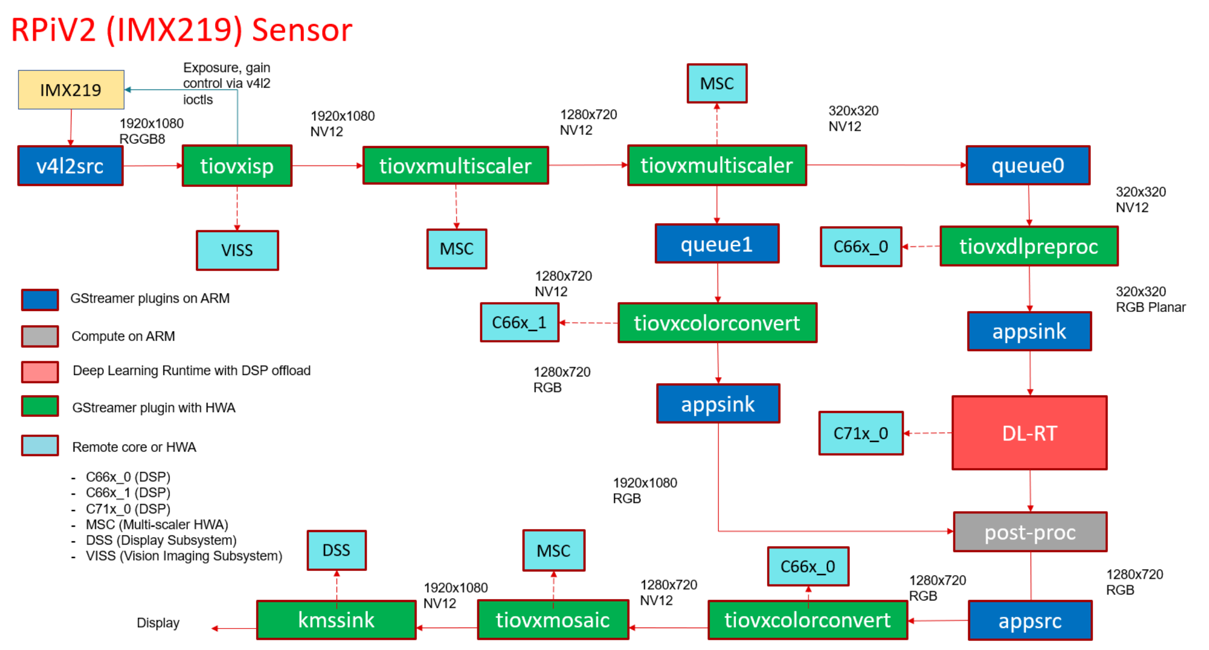


Fig. 2.131: GStreamer based data-flow pipeline with IMX219 sensor, ISP and display

Model	FPS	Total time (ms)	Inference time (ms)	A72 Load (%)	DDR Read BW (MB/s)	DDR Write BW (MB/s)	DDR Total BW (MB/s)	C71 Load (%)	C66_1 Load (%)	C66_2 Load (%)	MCU2_0 Load (%)	MCU2_1 Load (%)	MSC_0 (%)	MSC_1 (%)	VISS (%)	NF (%)	LDC (%)	SDE (%)	DOF (%)
ONR-CL-6150-mobileNetV2-1p4-qat	30.64	33.19	3.01	15.72	1781	853	2634	9.0	16.0	9.0	13.0	1.0	31.78	0	22.37	0	0	0	0
TFL-CL-0000-mobileNetV1-mlperf	30.59	33.14	1.04	12.78	1612	798	2410	5.0	18.0	9.0	13.0	1.0	31.65	0	22.31	0	0	0	0
TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320	30.56	33.07	5.00	13.30	1730	809	2539	15.0	25.0	9.0	13.0	1.0	32.6	0	22.19	0	0	0	0
TVM-CL-3410-gluoncv-mxnet-mobv2	30.48	33.14	2.01	12.91	1708	852	2560	7.0	16.0	9.0	13.0	1.0	31.83	0	22.26	0	0	0	0

Source : IMX390 over FPD-Link Capture Framerate : **30 fps** Resolution : **1080p** format : **SRGGB12**

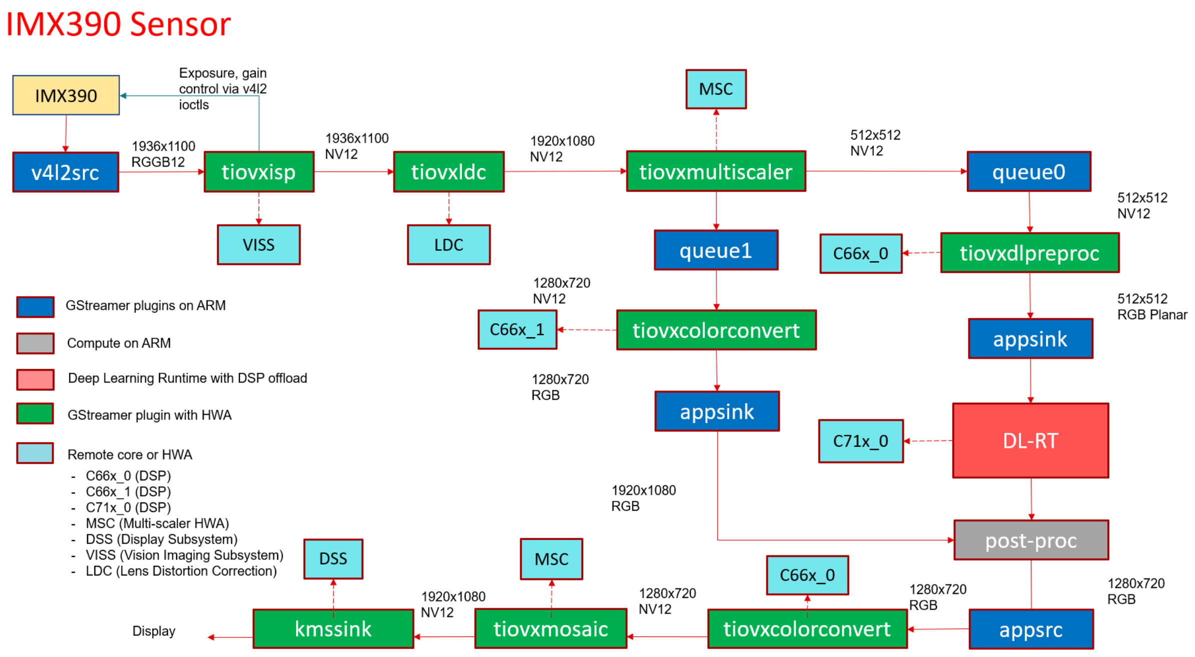


Fig. 2.132: GStreamer based data-flow pipeline with IMX390 sensor, ISP, LDC and display

Model	FPS	Total time (ms)	Inference time (ms)	A72 Load (%)	DDR Read BW (MB/s)	DDR Write BW (MB/s)	DDR Total BW (MB/s)	C71 Load (%)	C66_1 Load (%)	C66_2 Load (%)	MCU2_0 Load (%)	MCU2_1 Load (%)	MSC_0 (%)	MSC_1 (%)	VISS (%)	NF (%)	LDC (%)	SDE (%)	DOF (%)
ONR-CL-6150-mobileNetV2-1p4-qat	30.59	33.15	3.09	25.18	2207	1102	3309	10.0	16.0	9.0	14.0	1.0	31.73	0	22.94	0	10.8	0	0
TFL-CL-0000-mobileNetV1-mlperf	30.53	33.15	1.21	16.20	2019	1040	3059	5.0	18.0	9.0	15.0	1.0	32.80	0	23.34	0	10.10	0	0
TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320	30.43	33.13	5.02	23.7	2201	1067	3268	15.0	25.0	9.0	14.0	1.0	32.80	0	22.88	0	9.95	0	
TVM-CL-3410-gluoncv-mxnet-mobv2	30.44	33.16	2.12	21.50	2111	1100	3211	7.0	16.0	9.0	15.0	1.0	32.28	0	22.88	0	10.6	0	0

Test Report

Here is the summary of the sanity tests we ran with both Python and C++ demos. Test cases vary with different inputs, outputs, runtime, models, python/c++ apps.

1. Inputs:

   - Camera (Logitech C270, 1280x720, JPEG)

- Camera (Omnivision OV5640, 1280x720, YUV)
- Camera (Rpi v2 Sony IMX219, 1920x1080, RAW)
- Image files (30 images under edge_ai_apps/data/images)
- Video file (10s video 1 file under edge_ai_apps/data/videos)
- RSTP Video Server

2. Outputs:
   - Display (eDP or HDMI)
   - File write to SD card
3. Inference Type:
   - Image classification
   - Object detection
   - Semantic segmentation
4. Runtime/models:
   - DLR
   - TFLite
   - ONNX
5. Applications:
   - Python
   - C++
6. Platform:
   - Host OS
   - Docker

Demo Apps test report

Single Input Single Output

Category	# test case	Pass	Fail
Host OS - Python	99	99	0
Host OS - C++	99	99	0

S.No	Models	Input	Output	Host OS-C++	Host OS-Python	Docker-C++	Docker-Python	Comments
1	TVM-CL-3410-gluoncv-mxnet-mobv2	Image	Display	Pass				
2	TVM-CL-3410-gluoncv-mxnet-mobv2	Image	Video-Filewrite	Fail				
3	TVM-CL-3410-gluoncv-mxnet-mobv2	Image	Image-Filewrite	Pass				
4	TVM-CL-3410-gluoncv-mxnet-mobv2	Video	Display	Pass				
5	TVM-CL-3410-gluoncv-mxnet-mobv2	Video	Video-Filewrite	Pass				
6	TVM-CL-3410-gluoncv-mxnet-mobv2	USB Camera	Display	Pass				
7	TVM-CL-3410-gluoncv-mxnet-mobv2	USB Camera	Video-Filewrite	Pass				
8	TVM-CL-3410-gluoncv-mxnet-mobv2	CSI Camera	Display	Pass				
9	TVM-CL-3410-gluoncv-mxnet-mobv2	CSI Camera	Video-Filewrite	Pass				
10	TVM-CL-3410-gluoncv-mxnet-mobv2	RPI Camera	Display	Pass				
11	TVM-CL-3410-gluoncv-mxnet-mobv2	RPI Camera	Video-Filewrite	Pass				
12	TVM-CL-3410-gluoncv-mxnet-mobv2	RTSP - Video	Display	Pass				
13	TVM-CL-3410-gluoncv-mxnet-mobv2	RTSP - Video	Video-Filewrite	Pass				

continues on next page

Table 2.54 – continued from previous page

S.No	Models	Input	Output	Host OS-C++	Host OS-Python	Docker-C++	Docker-Python	Comments
14	TFL-CL-0000-mobileNetV1-mlperf					Image	Display	Pass
15	TFL-CL-0000-mobileNetV1-mlperf					Image	Video-Filewrite	Fail
16	TFL-CL-0000-mobileNetV1-mlperf					Image	Image-Filewrite	Pass
17	TFL-CL-0000-mobileNetV1-mlperf					Video	Display	Pass
18	TFL-CL-0000-mobileNetV1-mlperf					Video	Video-Filewrite	Pass
19	TFL-CL-0000-mobileNetV1-mlperf					USB Camera	Display	Pass
20	TFL-CL-0000-mobileNetV1-mlperf					USB Camera	Video-Filewrite	Pass
21	TFL-CL-0000-mobileNetV1-mlperf					CSI Camera	Display	Pass
22	TFL-CL-0000-mobileNetV1-mlperf					CSI Camera	Video-Filewrite	Pass
23	TFL-CL-0000-mobileNetV1-mlperf					RPI Camera	Display	Pass
24	TFL-CL-0000-mobileNetV1-mlperf					RPI Camera	Video-Filewrite	Pass
25	TFL-CL-0000-mobileNetV1-mlperf					RTSP - Video	Display	Pass
26	TFL-CL-0000-mobileNetV1-mlperf					RTSP - Video	Video-Filewrite	Pass
27	ONR-CL-6360-regNetx-200mf					Image	Display	Pass
28	ONR-CL-6360-regNetx-200mf					Image	Video-Filewrite	Fail
29	ONR-CL-6360-regNetx-200mf					Image	Image-Filewrite	Pass
30	ONR-CL-6360-regNetx-200mf					Video	Display	Pass
31	ONR-CL-6360-regNetx-200mf					Video	Video-Filewrite	Pass
32	ONR-CL-6360-regNetx-200mf					USB Camera	Display	Pass
33	ONR-CL-6360-regNetx-200mf					USB Camera	Video-Filewrite	Pass
34	ONR-CL-6360-regNetx-200mf					CSI Camera	Display	Pass
35	ONR-CL-6360-regNetx-200mf					CSI Camera	Video-Filewrite	Pass
36	ONR-CL-6360-regNetx-200mf					RPI Camera	Display	Pass
37	ONR-CL-6360-regNetx-200mf					RPI Camera	Video-Filewrite	Pass
38	ONR-CL-6360-regNetx-200mf					RTSP - Video	Display	Pass
39	ONR-CL-6360-regNetx-200mf					RTSP - Video	Video-Filewrite	Pass
40	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					Image	Display	Pass
41	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					Image	Video-Filewrite	Fail
42	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					Image	Image-Filewrite	Pass
43	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					Video	Display	Pass
44	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					Video	Video-Filewrite	Pass
45	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					USB Camera	Display	Pass
46	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					USB Camera	Video-Filewrite	Pass
47	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					CSI Camera	Display	Pass
48	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					CSI Camera	Video-Filewrite	Pass
49	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					RPI Camera	Display	Pass
50	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					RPI Camera	Video-Filewrite	Pass
51	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					RTSP - Video	Display	Pass
52	TVM-OD-5020-yolov3-mobv1-gluon-mxnet-coco-416x416					RTSP - Video	Video-Filewrite	Pass
53	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					Image	Display	Pass
54	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					Image	Video-Filewrite	Fail
55	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					Image	Image-Filewrite	Pass
56	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					Video	Display	Pass
57	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					Video	Video-Filewrite	Pass
58	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					USB Camera	Display	Pass
59	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					USB Camera	Video-Filewrite	Pass
60	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					CSI Camera	Display	Pass
61	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					CSI Camera	Video-Filewrite	Pass
62	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					RPI Camera	Display	Pass
63	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					RPI Camera	Video-Filewrite	Pass
64	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					RTSP - Video	Display	Pass
65	TFL-OD-2020-ssdLite-mobDet-DSP-coco-320x320					RTSP - Video	Video-Filewrite	Pass
66	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512					Image	Display	Pass

continues on next page

Table 2.54 – continued from previous page

S.No	Models	Input	Output	Host OS-C++	Host OS-Python	Docker-C++	Docker-Python	Comments
67	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	Image	Video-Filewrite	Fail				
68	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	Image	Image-Filewrite	Pass				
69	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	Video	Display	Pass				
70	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	Video	Video-Filewrite	Pass				
71	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	USB Camera	Display	Pass				
72	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	USB Camera	Video-Filewrite	Pass				
73	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	CSI Camera	Display	Pass				
74	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	CSI Camera	Video-Filewrite	Pass				
75	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	RPI Camera	Display	Pass				
76	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	RPI Camera	Video-Filewrite	Pass				
77	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	RTSP - Video	Display	Pass				
78	ONR-OD-8050-ssd-lite-regNetX-800mf-fpn-bgr-mmdet-coco-512x512	RTSP - Video	Video-Filewrite	Pass				
79	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	Image	Display	Pass				
80	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	Image	Video-Filewrite	Fail				
81	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	Image	Image-Filewrite	Pass				
82	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	Video	Display	Pass				
83	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	Video	Video-Filewrite	Pass				
84	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	USB Camera	Display	Pass				
85	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	USB Camera	Video-Filewrite	Pass				
86	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	CSI Camera	Display	Pass				
87	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	CSI Camera	Video-Filewrite	Pass				
88	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	RPI Camera	Display	Pass				
89	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	RPI Camera	Video-Filewrite	Pass				
90	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	RTSP - Video	Display	Pass				
91	TVM-SS-5720-deeplabv3lite-regnetx800mf-cocoseg21-512x512	RTSP - Video	Video-Filewrite	Pass				
92	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	Image	Display	Pass				
93	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	Image	Video-Filewrite	Fail				
94	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	Image	Image-Filewrite	Pass				
95	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	Video	Display	Pass				
96	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	Video	Video-Filewrite	Pass				
97	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	USB Camera	Display	Pass				
98	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	USB Camera	Video-Filewrite	Pass				
99	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	CSI Camera	Display	Pass				
100	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	CSI Camera	Video-Filewrite	Pass				
101	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	RPI Camera	Display	Pass				
102	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	RPI Camera	Video-Filewrite	Pass				
103	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	RTSP - Video	Display	Pass				
104	TFL-SS-2580-deeplabv3_mobv2-ade20k32-mlperf-512x512	RTSP - Video	Video-Filewrite	Pass				
105	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	Image	Display	Pass				
106	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	Image	Video-Filewrite	Fail				
107	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	Image	Image-Filewrite	Pass				
108	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	Video	Display	Pass				
109	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	Video	Video-Filewrite	Pass				
110	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	USB Camera	Display	Pass				
111	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	USB Camera	Video-Filewrite	Pass				
112	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	CSI Camera	Display	Pass				
113	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	CSI Camera	Video-Filewrite	Pass				
114	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	RPI Camera	Display	Pass				
115	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	RPI Camera	Video-Filewrite	Pass				
116	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	RTSP - Video	Display	Pass				
117	ONR-SS-8610-deeplabv3lite-mobv2-ade20k32-512x512	RTSP - Video	Video-Filewrite	Pass				

Single Input Multi Output

Category	# test case	Pass	Fail
Host OS - Python	15	15	0
docker - Python	15	15	0
Host OS - C++	15	15	0
Docker - C++	15	15	0

S.No	Models	Input	Output	Host OS-C++	Host OS-Python	Docker-C++	Docker-Python	Comments
1	2 Models (TFL-CL, ONR-SS)	%04d.jpg	Display	Pass	Pass	Pass	Pass	
2	3-Models (TVM-CL, TFL-OD, ONR-SS)	%04d.jpg	Display	Pass	Pass	Pass	Pass	
3	4-Models (TVM-SS, TFL-OD, ONR-SS, ONR-CL)	%04d.jpg	Display	Pass	Pass	Pass	Pass	
4	2 Models (TFL-CL, ONR-SS)	video_0000.mp4	Display	Pass	Pass	Pass	Pass	
5	3-Models (TVM-CL, TFL-OD, ONR-SS)	video_0000.mp4	Display	Pass	Pass	Pass	Pass	
6	4-Models (TVM-SS, TFL-OD, ONR-SS, ONR-CL)	video_0000.mp4	Display	Pass	Pass	Pass	Pass	
7	2 Models (TFL-CL, ONR-SS)	USB_camera	Display	Pass	Pass	Pass	Pass	
8	3-Models (TVM-CL, TFL-OD, ONR-SS)	USB_camera	Display	Pass	Pass	Pass	Pass	
9	4-Models (TVM-SS, TFL-OD, ONR-SS, ONR-CL)	USB_camera	Display	Pass	Pass	Pass	Pass	
10	2 Models (TFL-CL, ONR-SS)	CSI_camera	Display	Pass	Pass	Pass	Pass	
11	3-Models (TVM-CL, TFL-OD, ONR-SS)	CSI_camera	Display	Pass	Pass	Pass	Pass	
12	4-Models (TVM-SS, TFL-OD, ONR-SS, ONR-CL)	CSI_camera	Display	Pass	Pass	Pass	Pass	
13	2 Models (TFL-CL, ONR-SS)	rtsp	Display	Pass	Pass	Pass	Pass	
14	3-Models (TVM-CL, TFL-OD, ONR-SS)	rtsp	Display	Pass	Pass	Pass	Pass	
15	4-Models (TVM-SS, TFL-OD, ONR-SS, ONR-CL)	rtsp	Display	Pass	Pass	Pass	Pass	

Multi Input Multi Output

Category	# test case	Pass	Fail
Host OS - Python	8	8	0
docker - Python	8	8	0
Host OS - C++	8	8	0
Docker - C++	8	8	0

S.No	Models	Input	Output	Host OS-C++	Host OS-Python	Docker-C++	Docker-Python	Comments
1	2 Models (TVM-CL, TFL-OD)	%04d.jpg,video_0000.mp4	Display	Pass	Pass	Pass	Pass	
2	2 Models (TVM-OD, ONR-SS)	%04d.jpg,rtsp	Video-Filewrite	Pass	Pass	Pass	Pass	
3	2 Models (ONR-CL, TVM-SS)	%04d.jpg,USB_camera	Display	Pass	Pass	Pass	Pass	
4	3-Models (TVM-CL, TFL-OD, ONR-SS)	%04d.jpg,CSI_camera,rtsp	Video-Filewrite	Pass	Pass	Pass	Pass	
5	3-Models (TVM-CL, TFL-OD, ONR-SS)	video_0000.mp4,rtsp,%04d.jpg	Display	Pass	Pass	Pass	Pass	
6	3-Models (TFL-CL, ONR-CL, TVM-SS)	video_0000.mp4,USB_camera,CSI_camera	Video-Filewrite	Pass	Pass	Pass	Pass	
7	4-Models (TVM-CL, TFL-SS, ONR-OD, TFL-CL)	USB_camera,CSI_camera	Display	Pass	Pass	Pass	Pass	
8	4-Models (TVM-SS, TFL-SS, ONR-SS, ONR-OD)	USB_camera,video_0000.mp4	Video-Filewrite	Pass	Pass	Pass	Pass	

Note:

- Video file from RTSP server used for RTSP input test
- Please refer to the pub_edgeai_known_issues section for more details

2.6 PocketBeagle

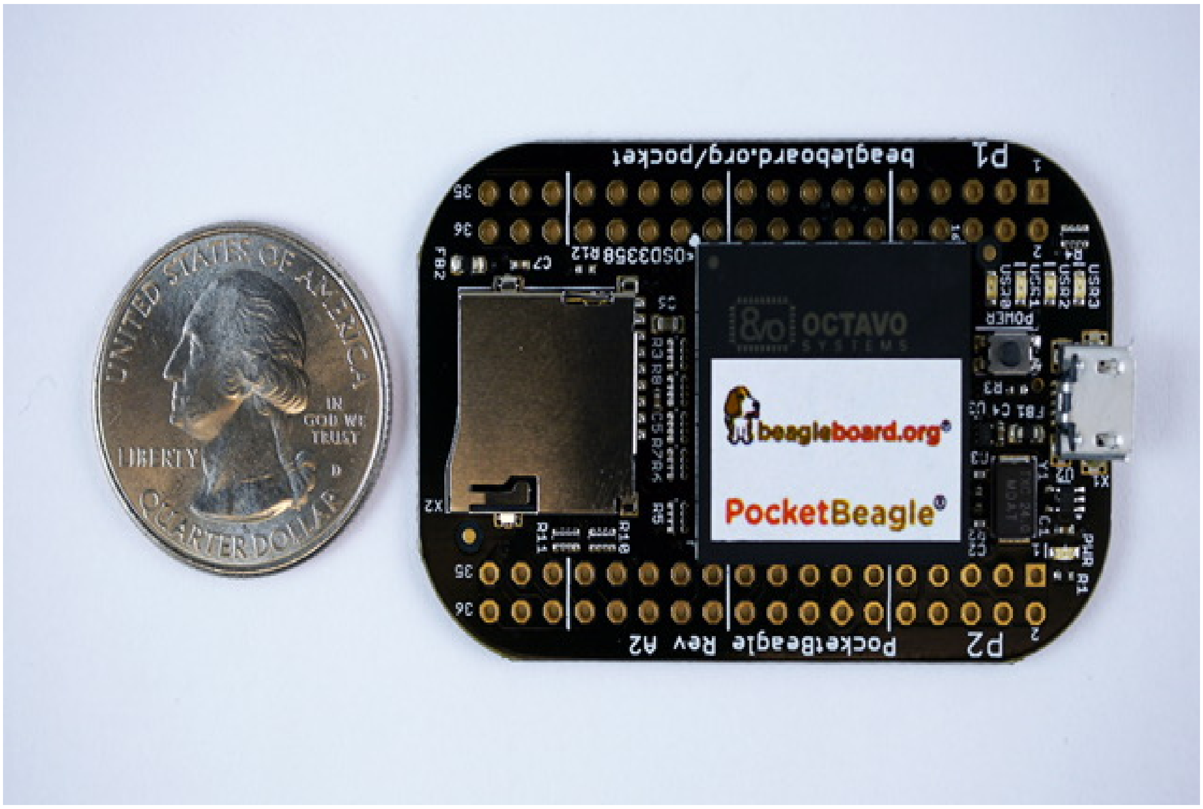
Contributors

- Maintaining author: Jason Kridner
- Contributing Editor: Cathy Wicks

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

PocketBeagle is an ultra-tiny-yet-complete open-source USB-key-fob computer. PocketBeagle features an incredible low cost, slick design and simple usage, making PocketBeagle the ideal development board for beginners and professionals alike.



2.6.1 Introduction

This document is the **System Reference Manual** for PocketBeagle and covers its use and design. PocketBeagle is an ultra-tiny-yet-complete Linux-enabled, community-supported, open-source USB-key-fob-computer. PocketBeagle features an incredible low cost, slick design and simple usage, making it the ideal development board for beginners and professionals alike. Simply develop directly in a web browser providing you with a playground for programming and electronics. Exploring is made easy with several available libraries and tutorials with many more coming.

PocketBeagle will boot directly from a microSD card. Load a Linux distribution onto your card, plug your board into your computer and get started. PocketBeagle runs GNU.Linux, so you can leverage many different high-level programming languages and a large body of drivers that prevent you from needing to write a lot of your own software.

This design will keep improving as the product matures based on feedback and experience. Software updates will be frequent and will be independent of the hardware revisions and as such not result in a change in the revision number of the board. A great place to find out the latest news and projects for PocketBeagle is on the home page beagleboard.org/pocket

Important: Make sure you check the BeagleBoard.org docs repository for the most up to date information.

PocketBeagle®

What is PocketBeagle?

PocketBeagle is an ultra-tiny-yet-complete open-source USB-key-fob computer. PocketBeagle features an incredible low cost, slick design and simple usage, making PocketBeagle the ideal development board for beginners and professionals alike.

Processor: Octavo Systems OSD3358 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM integrated
- Integrated power management
- 2×32-bit 200-MHz programmable real-time units (PRUs)
- ARM Cortex-M3

Summary of Key Features

- Low cost Linux computer with tremendous expansibility
- Opportunity to learn many programming aspects from educators on-line
- Openness and flexibility tear-down limits on your imagination

Summary of Technical Specifications

- Based on new Octavo Systems OSD3358-SM 21mm x 21mm system-in-package that includes 512MB DDR3 RAM, 1-GHz ARM Cortex-A8 CPU, 2x 200-MHz PRUs, ARM Cortex-M3, 3D accelerator, power/battery management and EEPROM
- 72 expansion pin headers with power and battery I/Os, high-speed USB, 8 analog inputs, 44 digital I/Os and numerous digital interface peripherals
- microUSB host/client and microSD connectors

Fig. 2.133: PocketBeagle Home Page

2.6.2 Change History

This section describes the change history of this document and board. Document changes are not always a result of a board change. A board change will always result in a document change.

Document Change History

Table 2.55: Change History

Rev	Changes	Date	By
A.x	Production Document	*December 7, 2017*	JK
0.0.5	Converted to .rst and gitlab hosting	*July 21, 2022*	DK

Board Changes

Table 2.56: Board History

Rev	Changes	Date	By
A1	Preliminary	*February 14, 2017*	JK
A2	Production. Fixed mikroBUS Click reset pins (made GPIO).	*September 22, 2017*	JK

PocketBone Upon the creation of the first, 27mm-by-27mm, Octavo Systems OSD3358 SIP, Jason did a hack two-layer board in EAGLE called "PocketBone" to drop the Beagle name as this was a totally unofficial effort not geared at being a BeagleBoard.org Foundation project. The board never worked because the 32kHz and 24MHz crystals were backwards and Michael Welling decided to pick it up and redo the design in KiCad as a four-layer board. Jason paid for some prototypes and this resulted in the first successful "PocketBone", a fully-open-source 1-GHz Linux computer in a fitting into a mini-mint tin.

Rev A1 The Rev A1 of PocketBeagle was a prototype not released to production. A few lines were wrong to be able to control mikroBUS Click add-on board reset lines and they were adjusted.

Rev A2 The Rev A2 of PocketBeagle was released to production and [https://www.prnewswire.com/news-releases/small-in-size–cost-meet-pocketbeagle-the-25-development-board-for-hobbyists-educators-and-professionals-300519950.html*launched at World MakerFaire 2017*].

Known issues in rev A2:

Issue	Link
GPIO44 is incorrectly labelled as GPIO48	github .com/beagleboard/pocketbeagle/is sues/4

2.6.3 Connecting Up PocketBeagle

This section provides instructions on how to hook up your board. The most common scenario is tethering PocketBeagle to your PC for local development.

What's In the Package

In the package you will find two items as shown in figures below.

- PocketBeagle
- Getting Started instruction card with link to the support URL.

Connecting the board

This section will describe how to connect to the board. Information can also be found on the Quick Start Guide that came in the box. Detailed information is also available at beagleboard.org/getting-started

The board can be configured in several different ways, but we will discuss the most common scenario. Future revisions of this document may include additional configurations.

Tethered to a PC using Debian Images

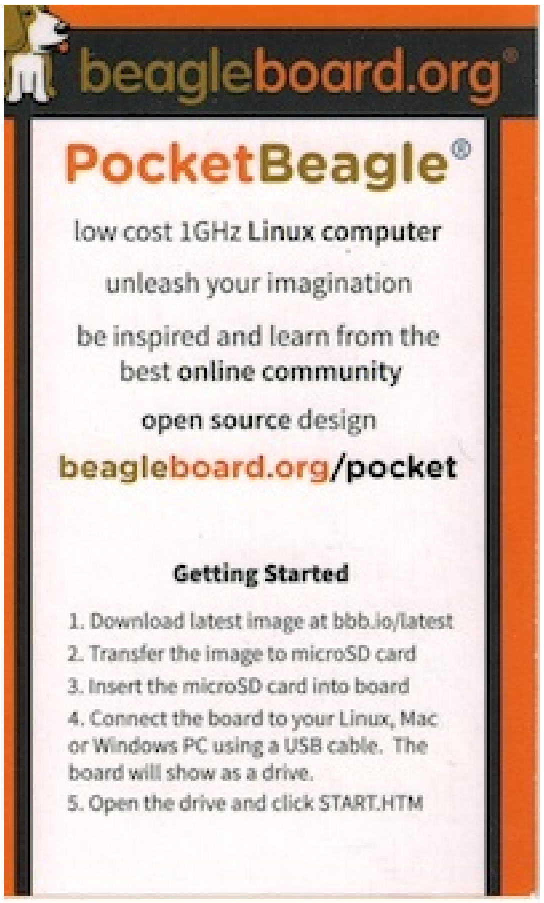
In this configuration, you will need the following additional items:

- microUSB to USB Type A Cable
- microSD card (>=4GB and <128GB)

The board is powered by the PC via the USB cable, no other cables are required. The board is accessed either as a USB storage drive or via a web browser on the PC. You need to use either Firefox or Chrome on the PC, IE will not work properly. Figure below shows this configuration.

In some instances, such as when additional add-on boards, or PocketCapes are connected, the PC may not be able to supply sufficient power for the full system. In that case, review the power requirements for the add-on board/cape; additional power may need to be supplied via the 5v input, but rarely is this the case.

Fig. 2.134: PocketBeagle Package



beagleboard.org

PocketBeagle®

low cost 1GHz Linux computer

unleash your imagination

be inspired and learn from the best online community

open source design

beagleboard.org/pocket

Getting Started

1. Download latest image at bbb.io/latest
2. Transfer the image to microSD card
3. Insert the microSD card into board
4. Connect the board to your Linux, Mac or Windows PC using a USB cable. The board will show as a drive.
5. Open the drive and click START.HTM

Fig. 2.135: PocketBeagle Package Insert front

Fig. 2.136: PocketBeagle Package Insert back

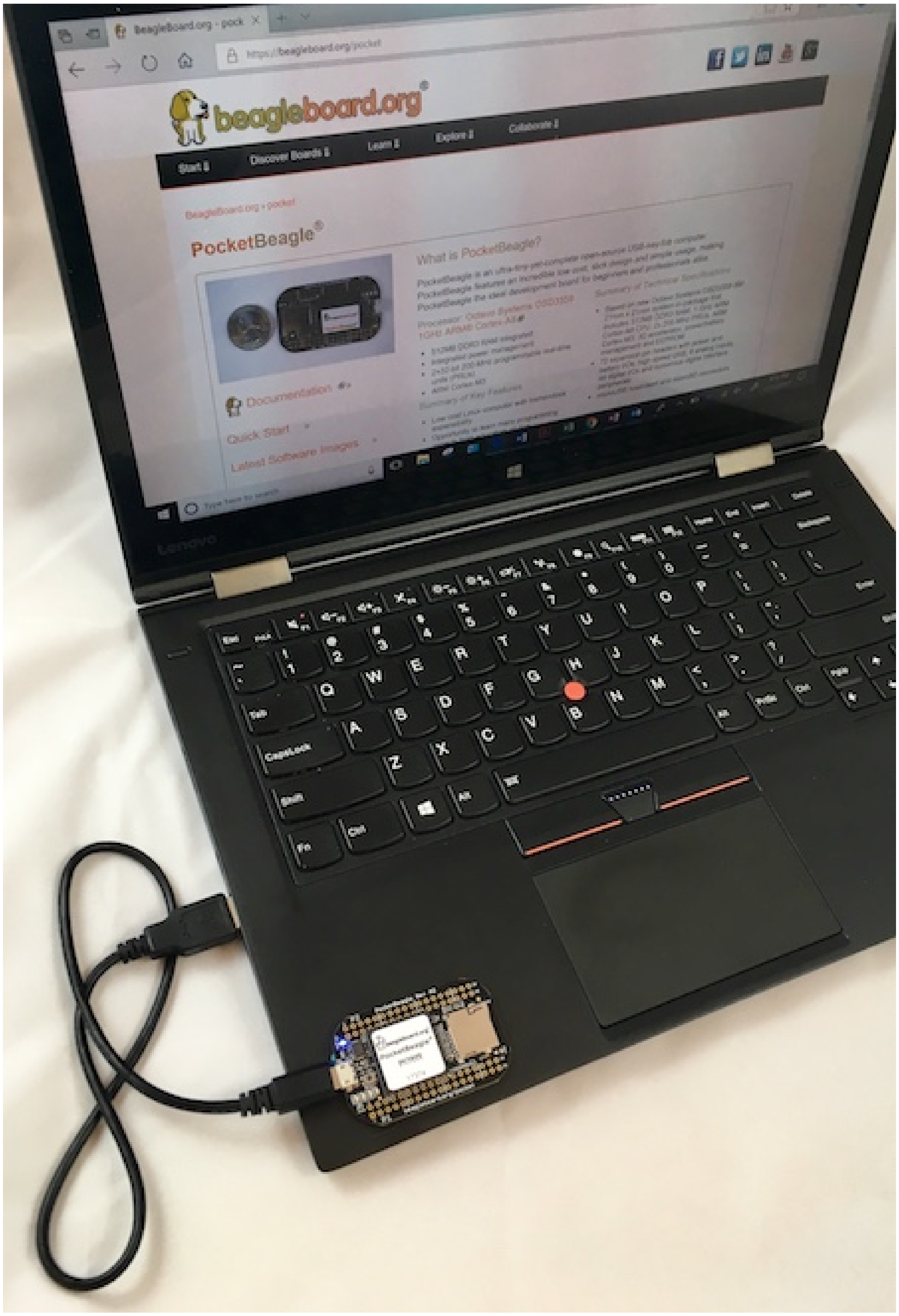
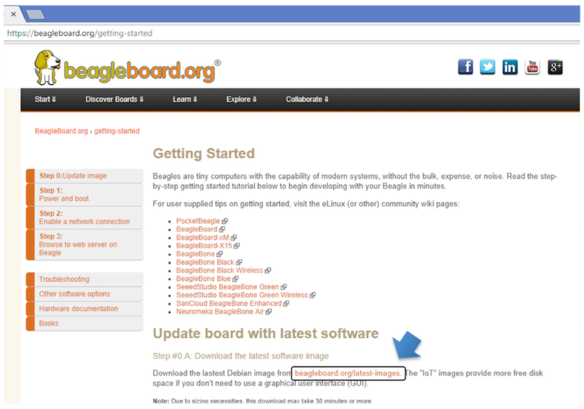


Fig. 2.137: Tethered Configuration

Getting Started The following steps will guide you to quickly download a PocketBeagle software image onto your microSD card and get started writing code.

1. Navigate to the Getting Started Page beagleboard.org/getting-started Follow along with the instructions and click on the link noted in Figure 5 below www.beagleboard.org/distros. You can also get to this page directly by going to bbb.io/latest



https://beagleboard.org/getting-started

Start ⇓ Discover Boards ⇓ Learn ⇓ Explore ⇓ Collaborate ⇓

BeagleBoard.org › getting-started

Step 0:Update image
Step 1: Power and boot
Step 2: Enable a network connection
Step 3: Browse to web server on Beagle
Troubleshooting
Other software options
Hardware documentation
Books

Getting Started

Beagles are tiny computers with the capability of modern systems, without the bulk, expense, or noise. Read the step-by-step getting started tutorial below to begin developing with your Beagle in minutes.

For user supplied tips on getting started, visit the eLinux (or other) community wiki pages:

- PocketBeagle
- BeagleBoard
- BeagleBoard-xM
- BeagleBoard-X15
- BeagleBone
- BeagleBone Black
- BeagleBone Black Wireless
- BeagleBone Blue
- SeeedStudio BeagleBone Green
- SeeedStudio BeagleBone Green Wireless
- SanCloud BeagleBone Enhanced
- Neuromeka BeagleBone Air

Update board with latest software

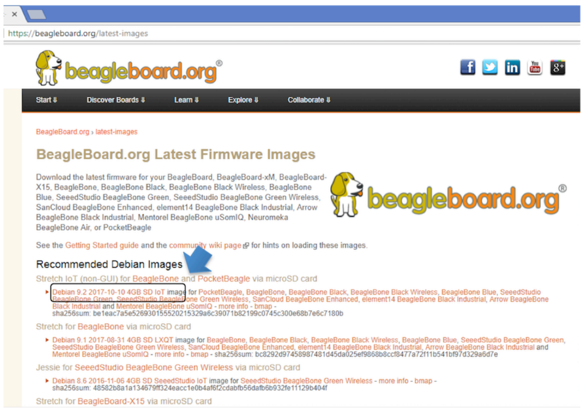
Step #0.A: Download the latest software image

Download the lastest Debian image from beagleboard.org/latest-images. The "IoT" images provide more free disk space if you don't need to use a graphical user interface (GUI).

Note: Due to sizing necessities, this download may take 30 minutes or more.

Fig. 2.138: Getting Started Page

1. Download the latest image onto your computer by following the link to the latest image and click on the Debian image for Stretch IoT (non-GUI) for BeagleBone and PocketBeagle via microSD card. See Figure 6 below. This will download a .img.xz file into the downloads folder of your computer.



https://beagleboard.org/latest-images

Start ⇓ Discover Boards ⇓ Learn ⇓ Explore ⇓ Collaborate ⇓

BeagleBoard.org › latest-images

BeagleBoard.org Latest Firmware Images

Download the latest firmware for your BeagleBoard, BeagleBoard-xM, BeagleBoard-X15, BeagleBone, BeagleBone Black, BeagleBone Black Wireless, BeagleBone Blue, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial, Mentorel BeagleBone uSomIQ, Neuromeka BeagleBone Air, or PocketBeagle

See the Getting Started guide and the community wiki page for hints on loading these images.

Recommended Debian Images

Stretch IoT (non-GUI) for BeagleBone and PocketBeagle via microSD card

- Debian 9.2 2017-10-10 4GB SD IoT image for PocketBeagle, BeagleBone, BeagleBone Black, BeagleBone Black Wireless, BeagleBone Blue, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial and Mentorel BeagleBone uSomIQ - more info - bmap - sha256sum: be1eac7a5e526930155520215329a6c39071b82199c0745c300e68b7e6c7180b

Stretch for BeagleBone via microSD card

- Debian 9.1 2017-08-31 4GB SD LXQT image for BeagleBone, BeagleBone Black, BeagleBone Black Wireless, BeagleBone Blue, SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SanCloud BeagleBone Enhanced, element14 BeagleBone Black Industrial, Arrow BeagleBone Black Industrial and Mentorel BeagleBone uSomIQ - more info - bmap - sha256sum: bc8292d97458987481d45da025ef9868b8ccf8477a72f11b541bf97d329a6d7e

Jessie for SeeedStudio BeagleBone Green Wireless via microSD card

- Debian 8.6 2016-11-06 4GB SD SeeedStudio IoT image for SeeedStudio BeagleBone Green Wireless - more info - bmap - sha256sum: 48582b8a1a134679ff324eacc1e0b4af6f2cdabfb56dafb6b932fe11129b404f

Stretch for BeagleBoard-X15 via microSD card

Fig. 2.139: Download Latest Software Image

1. Transfer the image to a microSD card.

Download and install an SD card programming utility if you do not already have one. We like https://etcher.io/ for new users and so we show that one in the steps below. Go to your downloads folder and doubleclick on the .exe file and follow the on-screen prompts. See figure 7.

Insert a new microSD card into a card reader/writer and attach it via the USB connection to your computer. Follow the instructions on the screen for selecting the .img file and burning the image from your computer to the microSD card. Eject the SD card reader when prompted and remove the card. See Figures 8 and 9.

1. Insert the microSD card into the board - you'll hear a satisfying click when it seats properly into the slot. It is important that your microSD card is fully inserted prior to powering the system.

1. Connect the micro USB connector on your cable to the board as shown in Figure 11. The microUSB connector is fairly robust, but we suggest that you not use the cable as a leash for your PocketBeagle. Take proper care not to put too much stress on the connector or cable.

1. Connect the large connector of the USB cable to your Linux, Mac or Windows PC USB port as shown in Figure 12. The board will power on and the power LED will be on as shown in Figure 13 below.

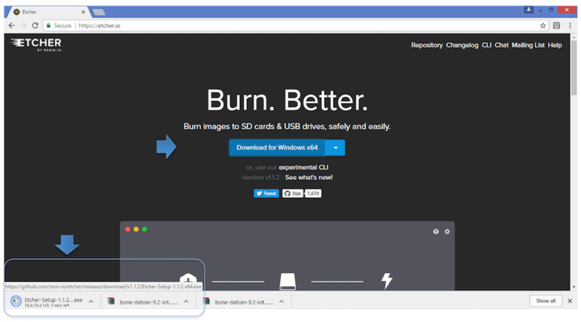

ETCHER
Repository Changelog CLI Chat Mailing List Help
Burn. Better.
Burn images to SD cards & USB drives, safely and easily.
Download for Windows x64
or use our experimental CLI
version v1.1.2 · See what's new!

Fig. 2.140: Download Etcher SD Card Utility

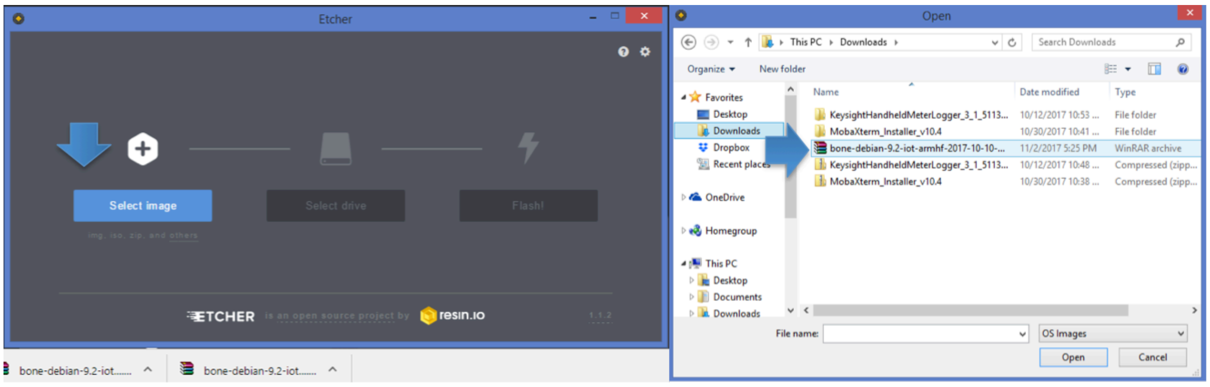


Fig. 2.141: Select the PocketBeagle Image

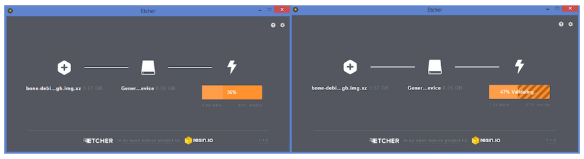


Fig. 2.142: Burn the Image to the SD Card

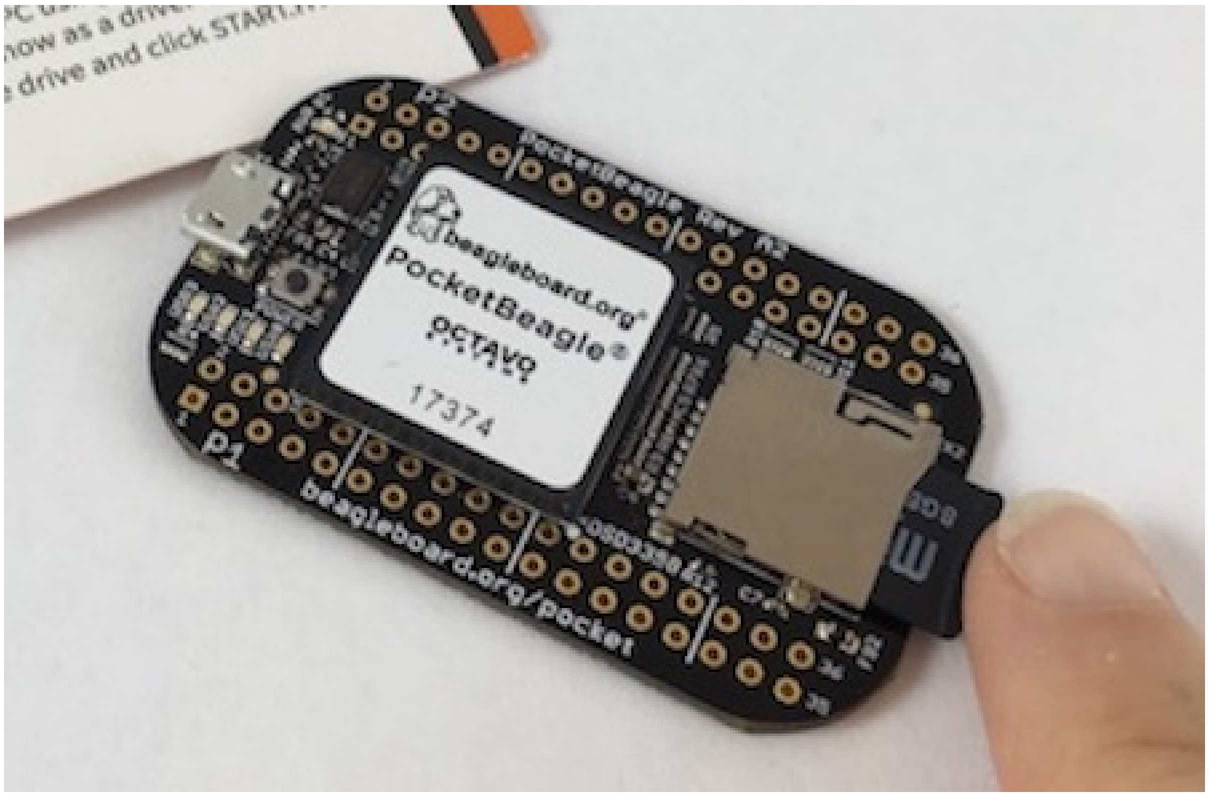


Fig. 2.143: Insert the microSD Card into PocketBeagle

1. As soon as you apply power, the board will begin the booting process and the userLEDs **Figure 14** will come on in sequence as shown below. It will take a few seconds for the status LEDs to come on, like teaching PocketBeagle to 'stay'. The LEDs will be flashing as it begins to boot the Linux kernel. While the four user LEDS can be over written and used as desired, they do have specific meanings in the image that you've initially placed on your microSD card once the Linux kernel has booted.

- **USER0** is the heartbeat indicator from the Linux kernel.
- **USER1** turns on when the microSD card is being accessed
- **USER2** is an activity indicator. It turns on when the kernel is not in the idle loop.
- **USER3** idle

Accessing the Board and Getting Started with Coding The board will appear as a USB Storage drive on your PC after the kernel has booted, which will take approximately 10 seconds. The kernel on the board needs to boot before the port gets enumerated. Once the board appears as a storage drive, do the following:

1. Open the USB Drive folder to view the files on your PocketBeagle.
2. Launch Interactive Quick Start Guide.

Right Click on the file named **START.HTM** and open it in Chrome or Firefox. This will use your browser to open a file running on PocketBeagle via the microSD card. You will see file:///Volumes/BEAGLEBONE/START.htm in the url bar of the browser. See Figure 15 below. This action displays an interactive Quick Start Guide from PocketBeagle.

1. Enable a Network Connection.

Click on 'Step 2' of the Interactive Quick Start Guide page to follow instructions to "Enable a Network Connection" (pointing to the DHCP server that is running on PocketBeagle). Copy the appropriate IP Address from the chart (according to your PC operating system type) and paste into your browser then add a **:3000** to the end of it. See example in Figure 16 below. This will launch from PocketBeagle one of it's favorite Web Based Development Environments, Cloud9 IDE, (Figure 17) so that you can teach your beagle new tricks!

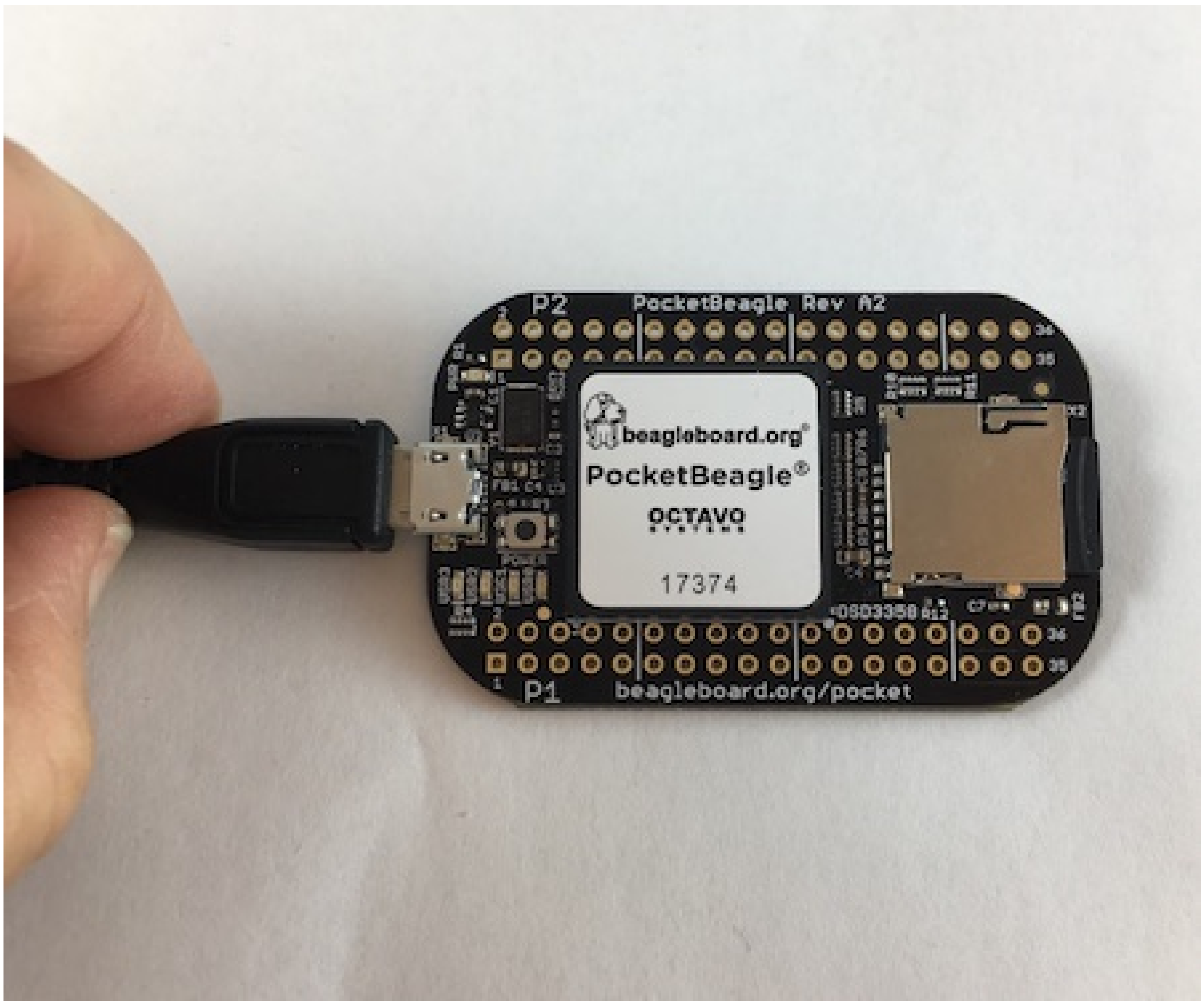


Fig. 2.144: Insert the micro USB Connector into PocketBeagle

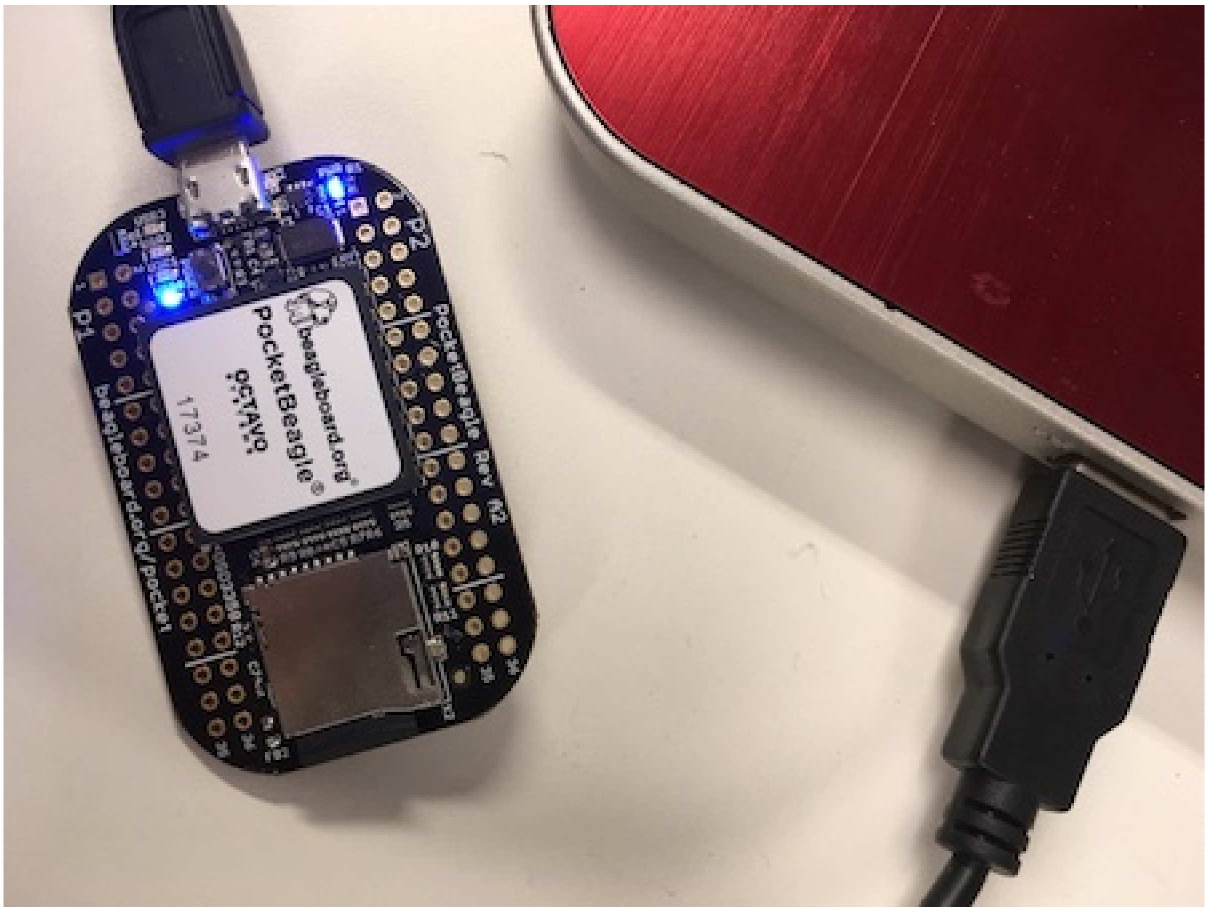


Fig. 2.145: Insert the USB connector into PC

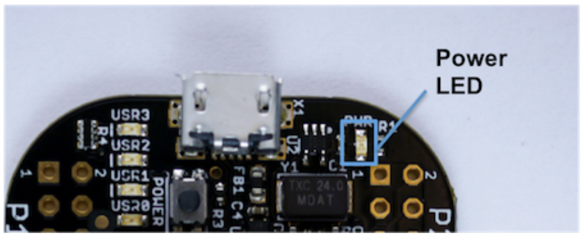


Fig. 2.146: Board Power LED

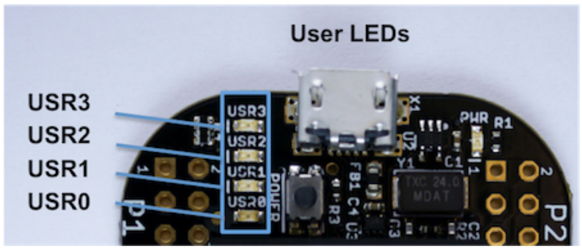
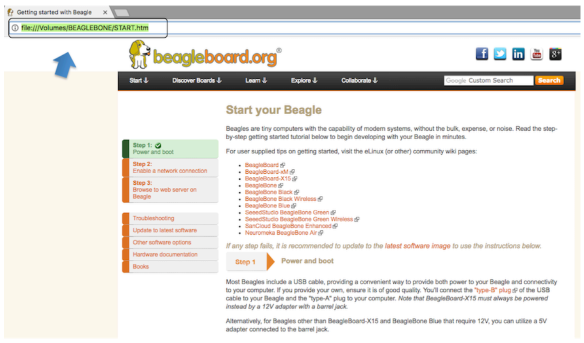


Fig. 2.147: User LEDs



Start your Beagle

Beagles are tiny computers with the capability of modern systems, without the bulk, expense, or noise. Read the step-by-step getting started tutorial below to begin developing with your Beagle in minutes.

For user supplied tips on getting started, visit the eLinux (or other) community wiki pages:

- BeagleBoard
- BeagleBoard-xM
- BeagleBoard-X15
- BeagleBone
- BeagleBone Black
- BeagleBone Black Wireless
- BeagleBone Blue
- SeeedStudio BeagleBone Green
- SeeedStudio BeagleBone Green Wireless
- SanCloud BeagleBone Enhanced
- Neuromeka BeagleBone Air

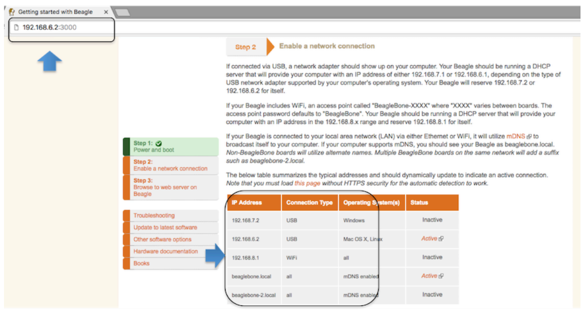
If any step fails, it is recommended to update to the latest software image to use the instructions below.

Step 1 Power and boot

Most Beagles include a USB cable, providing a convenient way to provide both power to your Beagle and connectivity to your computer. If you provide your own, ensure it is of good quality. You'll connect the "type-B" plug of the USB cable to your Beagle and the "type-A" plug to your computer. *Note that BeagleBoard-X15 must always be powered instead by a 12V adapter with a barrel jack.*

Alternatively, for Beagles other than BeagleBoard-X15 and BeagleBone Blue that require 12V, you can utilize a 5V adapter connected to the barrel jack.

Fig. 2.148: Interactive Quick Start Guide Launch


Getting started with Beagle
192.168.6.2:3000
Step 1: Power and boot
Step 2: Enable a network connection
Step 3: Browse to web server on Beagle
Troubleshooting
Update to latest software
Other software options
Hardware documentation
Books

Step 2 Enable a network connection

If connected via USB, a network adapter should show up on your computer. Your Beagle should be running a DHCP server that will provide your computer with an IP address of either 192.168.7.1 or 192.168.6.1, depending on the type of USB network adapter supported by your computer's operating system. Your Beagle will reserve 192.168.7.2 or 192.168.6.2 for itself.

If your Beagle includes WiFi, an access point called "BeagleBone-XXXX" where "XXXX" varies between boards. The access point password defaults to "BeagleBone". Your Beagle should be running a DHCP server that will provide your computer with an IP address in the 192.168.8.x range and reserve 192.168.8.1 for itself.

If your Beagle is connected to your local area network (LAN) via either Ethernet or WiFi, it will utilize mDNS to broadcast itself to your computer. If your computer supports mDNS, you should see your Beagle as beaglebone.local. *Non-BeagleBone boards will utilize alternate names. Multiple BeagleBone boards on the same network will add a suffix such as beaglebone-2.local.*

The below table summarizes the typical addresses and should dynamically update to indicate an active connection. *Note that you must load this page without HTTPS security for the automatic detection to work.*

IP Address	Connection Type	Operating System(s)	Status
192.168.7.2	USB	Windows	Inactive
192.168.6.2	USB	Mac OS X, Linux	*Active*
192.168.8.1	WiFi	all	Inactive
beaglebone.local	all	mDNS enabled	*Active*
beaglebone-2.local	all	mDNS enabled	Inactive

Fig. 2.149: Enable a Network Connection

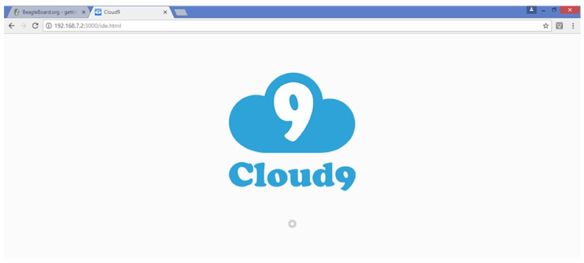
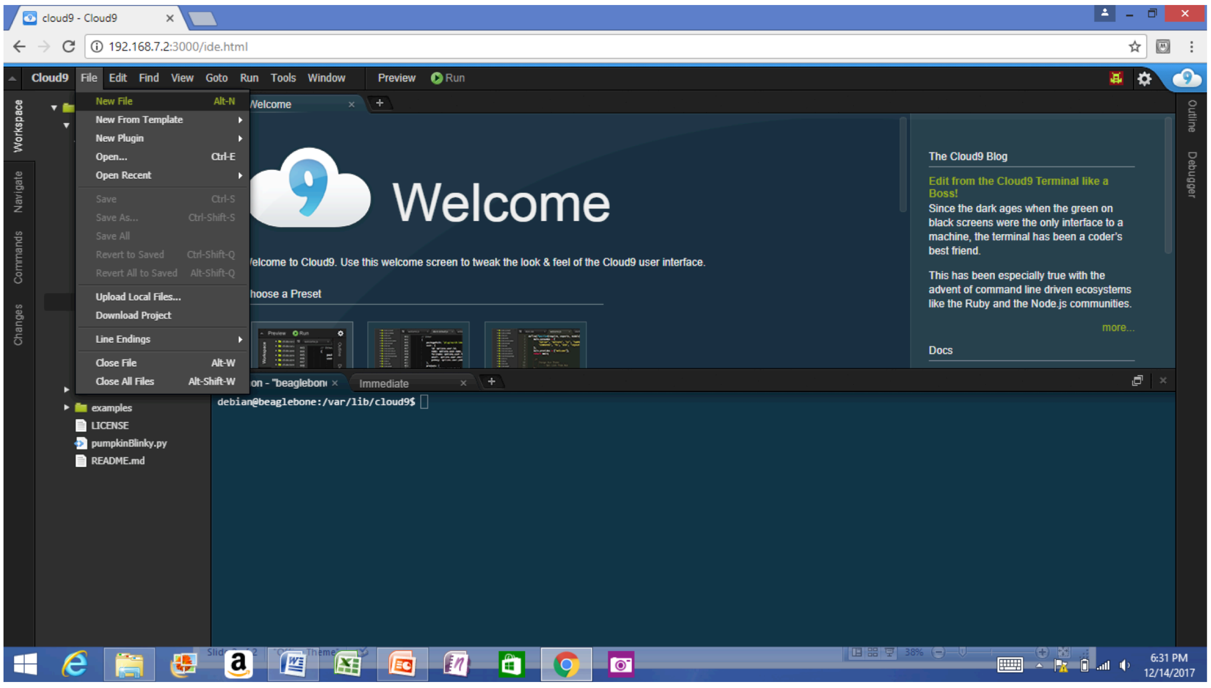


Fig. 2.150: Launch Cloud9 IDE

1. Get Started Coding with Cloud9 IDE - blinking USR3 LED in JavaScript using the BoneScript library example

   1. Create a new text file



Welcome

elcome to Cloud9. Use this welcome screen to tweak the look & feel of the Cloud9 user interface.

hoose a Preset

The Cloud9 Blog

Edit from the Cloud9 Terminal like a Boss!

Since the dark ages when the green on black screens were the only interface to a machine, the terminal has been a coder's best friend.

This has been especially true with the advent of command line driven ecosystems like the Ruby and the Node.js communities.

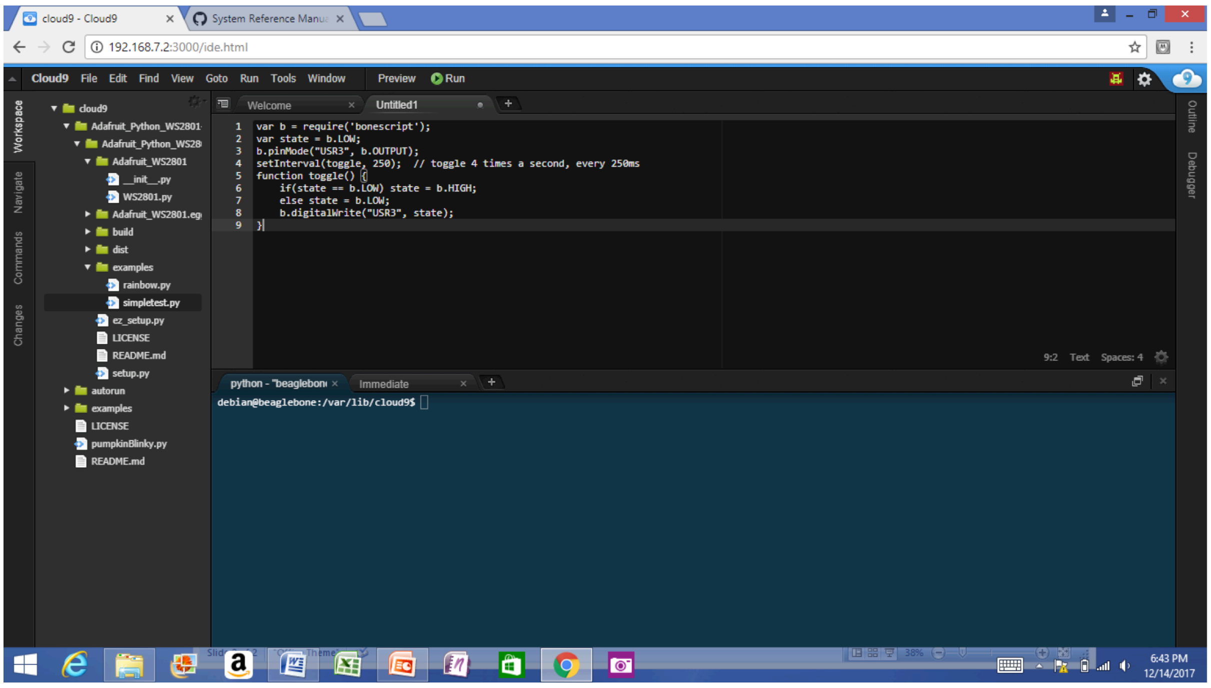
more...

Docs

```
debian@beaglebone:/var/lib/cloud9$
```

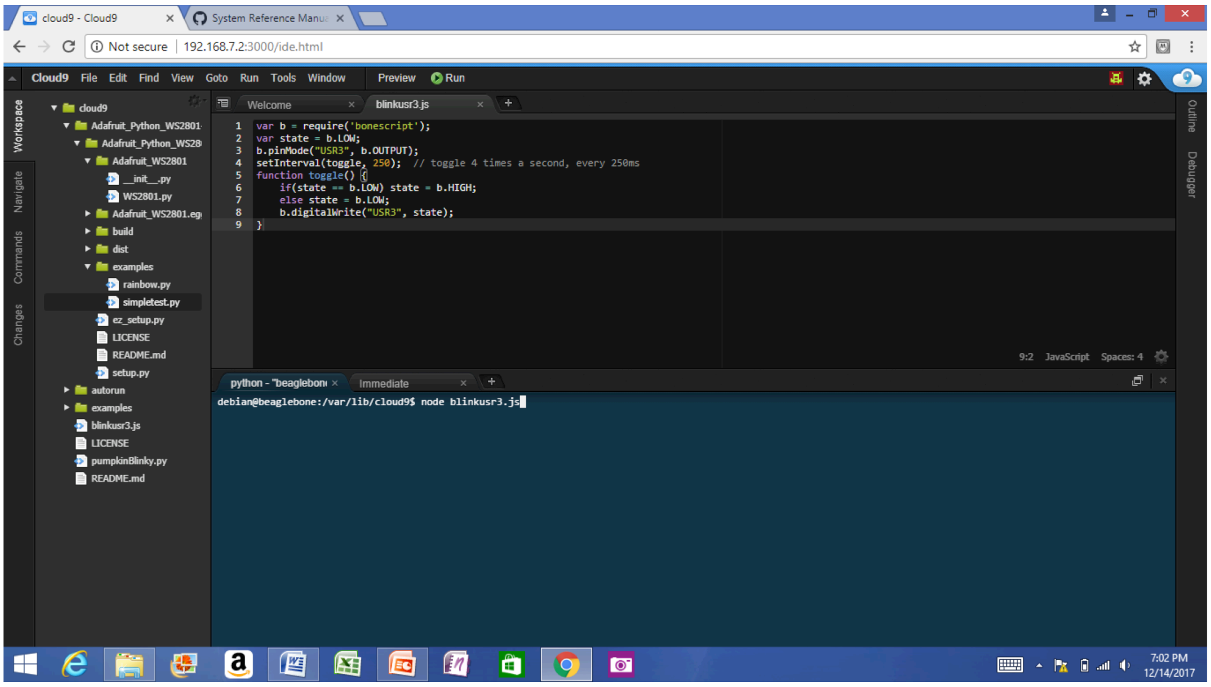
Copy and paste the below code into the editor

```
var b = require('bonescript');
var state = b.LOW;
b.pinMode("USR3", b.OUTPUT);
setInterval(toggle, 250);  // toggle 4 times a second, every 250ms
function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite("USR3", state);
}
```



```
var b = require('bonescript');
var state = b.LOW;
b.pinMode("USR3", b.OUTPUT);
setInterval(toggle, 250);  // toggle 4 times a second, every 250ms
function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite("USR3", state);
}
```

```
debian@beaglebone:/var/lib/cloud9$
```



```
var b = require('bonescript');
var state = b.LOW;
b.pinMode("USR3", b.OUTPUT);
setInterval(toggle, 250);  // toggle 4 times a second, every 250ms
function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite("USR3", state);
}
```

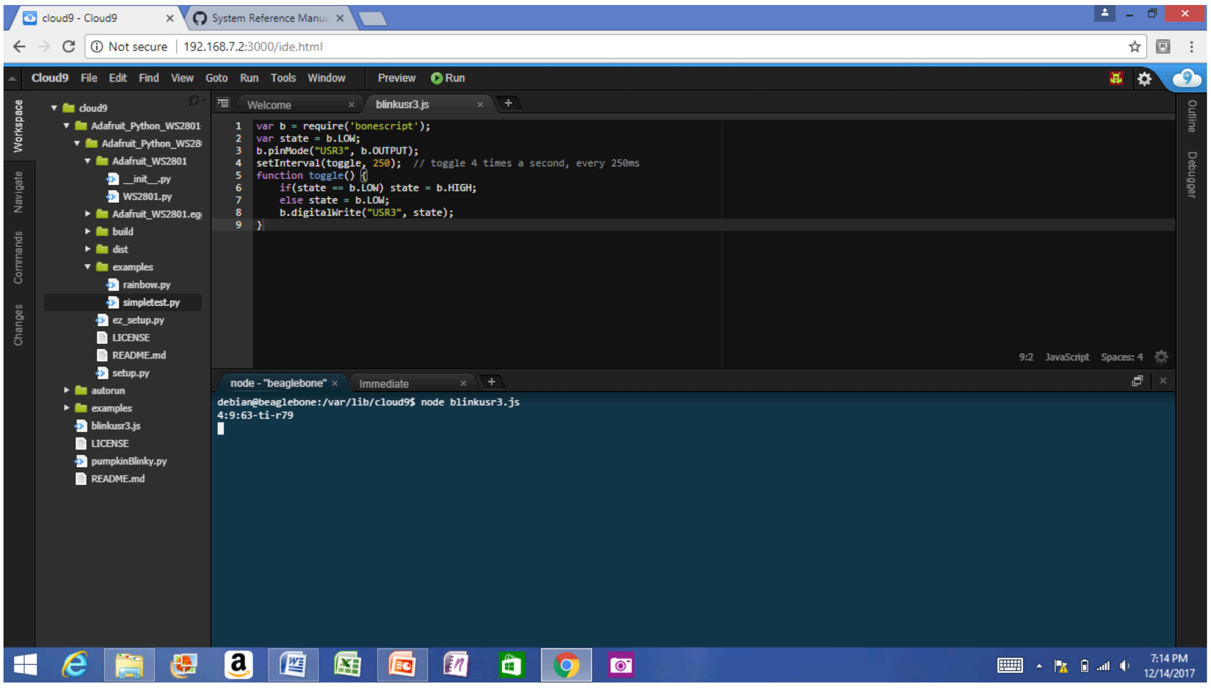
```
debian@beaglebone:/var/lib/cloud9$ node blinkusr3.js
```

Save the new text file as *blinkusr3.js* within the default directory

Execute .. code-block:

```
node blinkusr3.js
```

within the default (/var/lib/cloud9) directory



```
var b = require('bonescript');
var state = b.LOW;
b.pinMode("USR3", b.OUTPUT);
setInterval(toggle, 250);  // toggle 4 times a second, every 250ms
function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite("USR3", state);
}
```

```
debian@beaglebone:/var/lib/cloud9$ node blinkusr3.js
4:9:63-ti-r79
```

Type CTRL+C to stop the program running

Powering Down

1. Standard Power Down Press the power button momentarily with a tap. The system will power down automatically. This will shut down your software with grace. Software routines will run to completion.
The Standard Power Down can also be invoked from the Linux command shell via “sudo shutdown -h now”.

2. Hard Power Down Press the power button for 10 seconds. This will force an immediate shut down of the software. For example you may lose any items you have written to the memory. Holding the button longer than 10 seconds will perform a power reset and the system will power back on.

1. Remove the USB cable Remember to hold your board firmly at the USB connection while you remove the cable to prevent damage to the USB connector.

4. Powering up again. If you'd like to power up again without removing the USB cable follow these instructions:

   1. If you used Step 1 above to power down, to power back up, hold the power button for 10 seconds, release then tap it once and the system will boot normally.

   2. If you used Step 2 above to power down, to power back up, simply tap the power button and the system will boot normally.

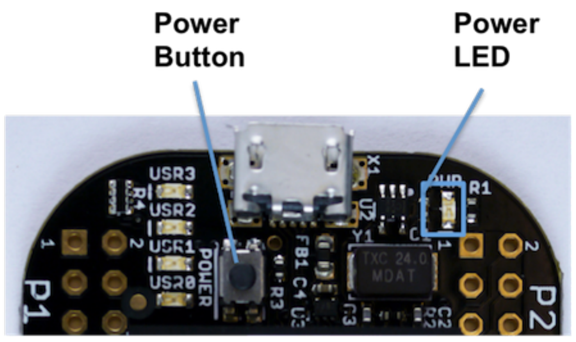


Fig. 2.151: Power Button

Other ways to Connect up to your PocketBeagle

The board can be configured in several different ways. Future revisions of this document may include additional configurations.

As other examples become documented, we'll update them on the Wiki for PocketBeagle PocketBeagle WiKi See also the on-line discussion.

2.6.4 PocketBeagle Overview

PocketBeagle is built around Octavo Systems' OSD335x-SM System-In-Package that integrates a high-performance Texas Instruments AM3358 processor, 512MB of DDR3, power management, nonvolatile serial memory and over 100 passive components into a single package. This integration saves board space by eliminating several packages that would otherwise need to be placed on the board, but more notably simplifies our board design so we can focus on the user experience.

The compact PocketBeagle design also offers access through the expansion headers to many of the interfaces and allows for the use of add-on boards called PocketCapes and Click Boards from MikroElektronika, to add many different combinations of features. A user may also develop their own board or add their own circuitry.

PocketBeagle Features and Specification

This section covers the specifications and features of the board in a chart and provides a high level description of the major components and interfaces that make up the board.

Table 2.57: PocketBeagle Features

Feature	
System-In-Package	Octavo Systems OSD335x-SM in 256 Ball BGA (21mm x 21mm)
SiP Incorporates	
Processor	Texas Instruments 1GHz Sitara™ AM3358 ARM® Cortex®-A8 with NEON floating-point accelerator
Graphics Engine	Imagination Technologies PowerVR SGX530 Graphics Accelerator
Real-Time Units	2x programmable real-time unit (PRU) 32-bit 200MHz microcontrollers with single-cycle I/O latency
Coprocessor	ARM® Cortex®-M3 for power management functions
SDRAM Memory	512MB DDR3 800MHz RAM
Non-Volatile Memory	4KB I2C EEPROM for board configuration information
Power Management	TPS65217C PMIC along with TL5209 LDO to provide power to the system with integrated 1-cell LiPo battery support
Connectivity	
SD/MMC	Bootable microSD card slot
USB	High speed USB 2.0 OTG (host/client) micro-B connector
Debug Support	JTAG test points and gdb/other monitor-mode debug possible
Power Source	microUSB connector, also expansion header options (battery, VIN or USB-VIN)
User I/O	Power Button with press detection interrupt via TPS65217C PMIC
Expansion Header	
USB	High speed USB 2.0 OTG (host/client) control signals
Analog Inputs	8 analog inputs with 6 @ 1.8V and 2 @ 3.3V along with 1.8V references
Digital I/O	44 digital GPIOs accessible with 18 enabled by default including 2 shared with the 3.3V analog input pins
UART	3 UARTs accessible with 2 enabled by default
I2C	2 I2C buses enabled by default
SPI	2 SPI buses with single chip selects enabled by default
PWM	4 Pulse Width Modulation outputs accessible with 2 enabled by default
QEP	2 Quadrature encoder inputs accessible
CAN	2 CAN bus controllers accessible

OSD3358-512M-BSM System in Package The Octavo Systems OSD3358-512M-BSM System-In-Package (SiP) is part of a family of products that are building blocks designed to allow easy and cost-effective implementation of systems based in Texas Instruments powerful Sitara AM335x line of processors. The OSD335x-SM integrates the AM335x along with the TI TPS65217C PMIC, the TI TL5209 LDO, up to 1 GB of DDR3 Memory, a 4 KB EEPROM for non-volatile configuration storage and resistors, capacitors and inductors into a single 21mm x 21mm design-in-ready package.

With this level of integration, the OSD335x-SM family of SiPs allows designers to focus on the key aspects of their system without spending time on the complicated high-speed design of the processor/DDR3 interface or the PMIC power distribution. It reduces size and complexity of design.

Full Datasheet and more information is available at octavosystems.com/octavo_products/osd335x-sm/

Board Component Locations

This section describes the key components on the board, their location and function.

Figure below shows the locations of the devices, connectors, LEDs, and switches on the PCB layout of the board.

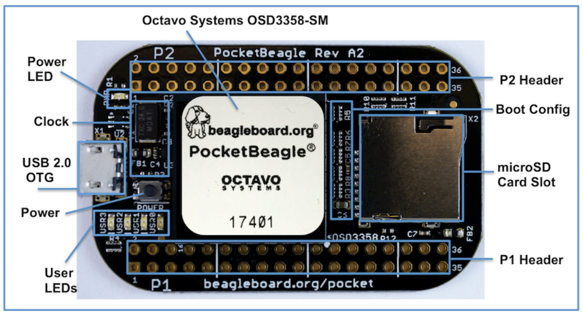


Fig. 2.152: Key Board Component Locations

Key Components

- **The Octavo Systems OSD3358-512M-BSM System-In-Package** is the processor system for the board
- **P1 and P2 Headers** come unpopulated so a user may choose their orientation
- **User LEDs** provides 4 programmable blue LEDs
- **Power BUTTON** can be used to power up or power down the board (see section 3.3.3 for details)
- **USB 2.0 OTG** is a microUSB connection to a PC that can also power the board
- **Power LED** provides communication regarding the power to the board
- **microSD** slot is where a microSD card can be installed.

2.6.5 PocketBeagle High Level Specification

This section provides the high level specification of PocketBeagle.

Block Diagram

Figure 22 below is the high level block diagram of PocketBeagle.

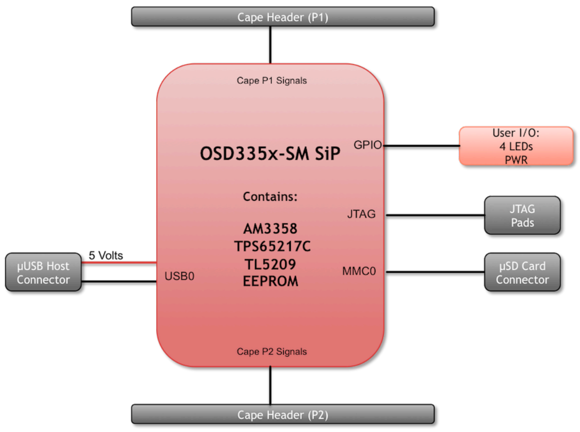


Fig. 2.153: PocketBeagle Key Components

System in Package (SiP)

The OSD335x-SM Block Diagram is detailed in Figure 23 below. More information, including design resources are available on the 'Octavo Systems Website'

Note: PocketBeagle utilizes the 512MB DDR3 memory size version of the OSD335x-SM A few of the features of the OSD335x-SM SiP may not be available on PocketBeagle headers. Please check Section 7 for the P1 and P2 header pin tables.

Connectivity

Expansion Headers PocketBeagle gives access to a large number of peripheral functions and GPIO via 2 dual rail expansion headers. With 36 pins each, the headers have been left unpopulated to enable users to choose the header connector orientation or add-on board / cape connector style. Pins are clearly marked on the bottom of the board with additional pin configurations available through software settings. Detailed information is available in Section 7.

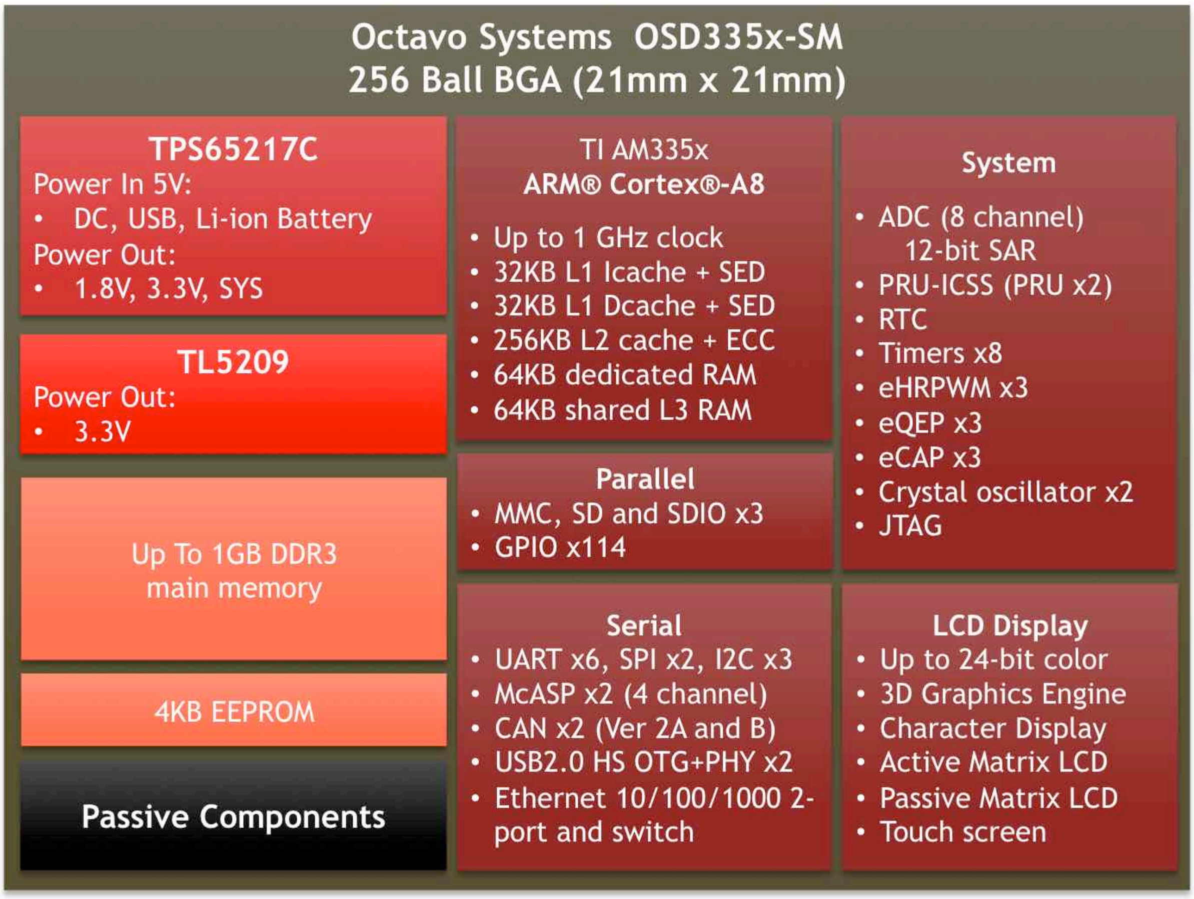


Fig. 2.154: OSD335x SIP Block Diagram

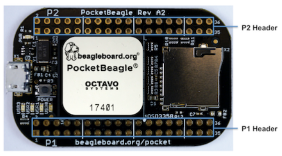


Fig. 2.155: PocketBeagle Expansion Headers

microSD Connector The board is equipped with a single microSD connector to act as the primary boot source for the board. Just about any microSD card you have will work, we commonly find 4G to be suitable.

When plugging in the SD card, the writing on the card should be up. Align the card with the connector and push to insert. Then release. There should be a click and the card will start to eject slightly, but it then should latch into the connector. To eject the card, push the SD card in and then remove your finger. The SD card will be ejected from the connector. Do not pull the SD card out or you could damage the connector.

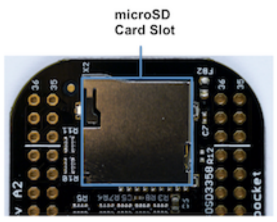


Fig. 2.156: microSD Connector

USB 2.0 Connector The board has a microUSB connector that is USB 2.0 HS compatible that connects the USB0 port to the SiP. Generally this port is used as a client USB port connected to a power source, such as your PC, to power the board. If you would like to use this port in host mode you will need to supply power for peripherals via Header P1 pin 7 (USB1.VIN) or through a powered USB Hub. Additionally, in the USB host configuration, you will need to power the board through Header P1 pin 1 (VIN) or Header P1 pin 7 (USB1.VIN) or Header P2 pin 14 (BAT.VIN)

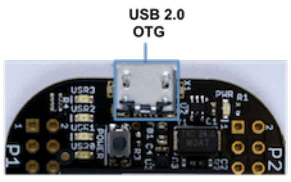


Fig. 2.157: USB 2.0 Connector

Boot Modes There are three boot modes:

- **SD Boot**: MicroSD connector acts as the primary boot source for the board. This is described in Section 3.
- **USB Boot**: This mode supports booting over the USB port. More information can be found in the project called "BeagleBoot" This project ported the BeagleBone bootloader server BBBlfs(currently written in c) to JavaScript(node.js) and make a cross platform GUI (using electron framework) flashing tool utilizing the etcher.io project. This will allow a single code base for a cross platform tool. For more information on BeagleBoot, see the BeagleBoot Project Page.
- **Serial Boot**: This mode will use the serial port to allow downloading of the software. A separate USB to TTL level serial UART converter cable is required or you can connect one of the Mikroelektronika FTDI Click Boards to use this method. The UART pins on PocketBeagle's expansion headers support the interface. For more information regarding the pins on the expansion headers and various modes, see Section 7.

Table 2.58: UART Pins on Expansion Headers for Serial Boot

H eader.Pin	S ilkscreen	Proc Ball	SiP Ball	Pin Name (Mode 0)
P1.22	GND			GND
P1.30	U0_TX	E16	B12	uart0_txd
P1.32	U0_RX	E15	A12	uart0_rxd

If the Serial Boot is not in use, the UART0 pins can be used for Serial Debug. See Section 5.6 for more information.

Software to support USB and serial boot modes is not provided by beagleboard.org. Please contact TI for support of this feature.

Power

The board can be powered from three different sources:

- A USB port on a PC.
- A power supply with a USB connector.
- Expansion Header pins.

Note: VIN-USB is directly shorted between the USB connector on PocketBeagle and USB1_VI on the expansion headers. You should only source power to the board over one of these and may optionally use the other as a power sink.

The tables below show the power related pins available on PocketBeagle's Expansion Headers.

Table 2.59: Power Inputs Available on Expansion Headers

H eader.Pin	S ilkscreen	Proc Ball	SiP Ball	Pin Name (Mode 0)
P1.01	VIN		P10, R10, T10	VIN
P1.07	USB1_VI		P9, R9, T9	VIN-USB
P2.14	BAT_+		P8, R8, T8	VIN-BAT

Table 2.60: Power Outputs Available on Expansion Headers

H eader.Pin	S ilkscreen	Proc Ball	SiP Ball	Pin Name (Mode 0)
P1.14	+3.3V		F6, F7, G6, G7	VOUT-3.3V
P1.24	VOUT		K6, K7, L6, L7	VOUT-5V
P2.13	VOUT		K6, K7, L6, L7	VOUT-5V
P2.23	+3.3V		F6, F7, G6, G7	VOUT-3.3V

Table 2.61: Ground Pins Available on Expansion Headers

H eader.Pin	S ilkscreen	Proc Ball	SiP Ball	Pin Name (Mode 0)
P1.15	USB1_GND			GND
P1.16	GND			GND
P1.22	GND			GND
P2.15	GND			GND
P2.21	GND			GND

Note: A comprehensive tutorial for Power Inputs and Outputs for the OSD335x System in Package is available in the 'Tutorial Series' on the Octavo Systems website.

JTAG Pads

Pads for an optional connection to a JTAG emulator has been provided on the back of PocketBeagle. More information about JTAG emulation can be found on the TI website - 'Entry-level debug through full-capability development'

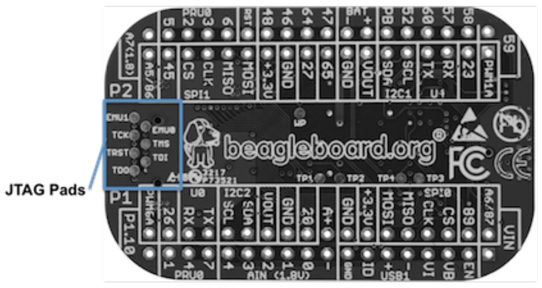
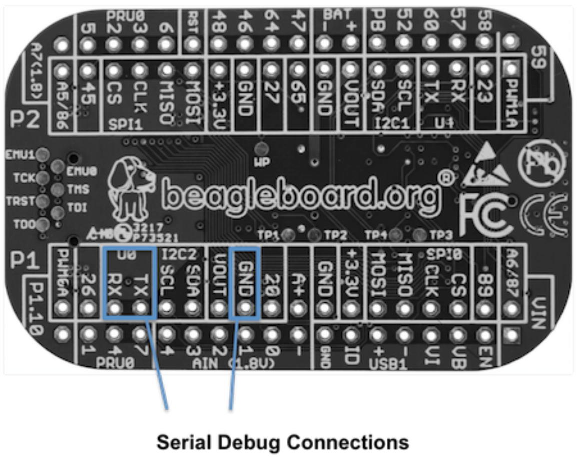


Fig. 2.158: JTAG Pad Connections

Serial Debug Port

Serial debug is provided via UART0 on the processor. See Section 5.3.4 for the Header Pin table. Signals supported are TX and RX. None of the handshake signals (CTS/RTS) are supported. A separate USB to TTL level serial UART converter cable is required or you can connect one of the Mikroelektronika FTDI Click Boards to use this method.



If serial boot is not used, the UART0 can be used to view boot messages during startup and can provide access to a console using a terminal access program like Putty. To view the boot messages or use the console the UART should be set to a baud rate of 115200 and use 8 bits for data, no parity bit and 1 stop bit (8N1).

2.6.6 Detailed Hardware Design

The following sections contain schematic references for PocketBeagle. Full schematics in both PDF and Eagle are available on the 'PocketBeagle Wiki'

OSD3358-SM SiP Design

Schematics for the OSD3358-SM SiP are divided into several diagrams.

SiP A OSD3358 SiP System and Power Signals

SiP B OSD3358 SiP JTAG, USB & Analog Signals

SiP C OSD3358 SiP Peripheral Signals

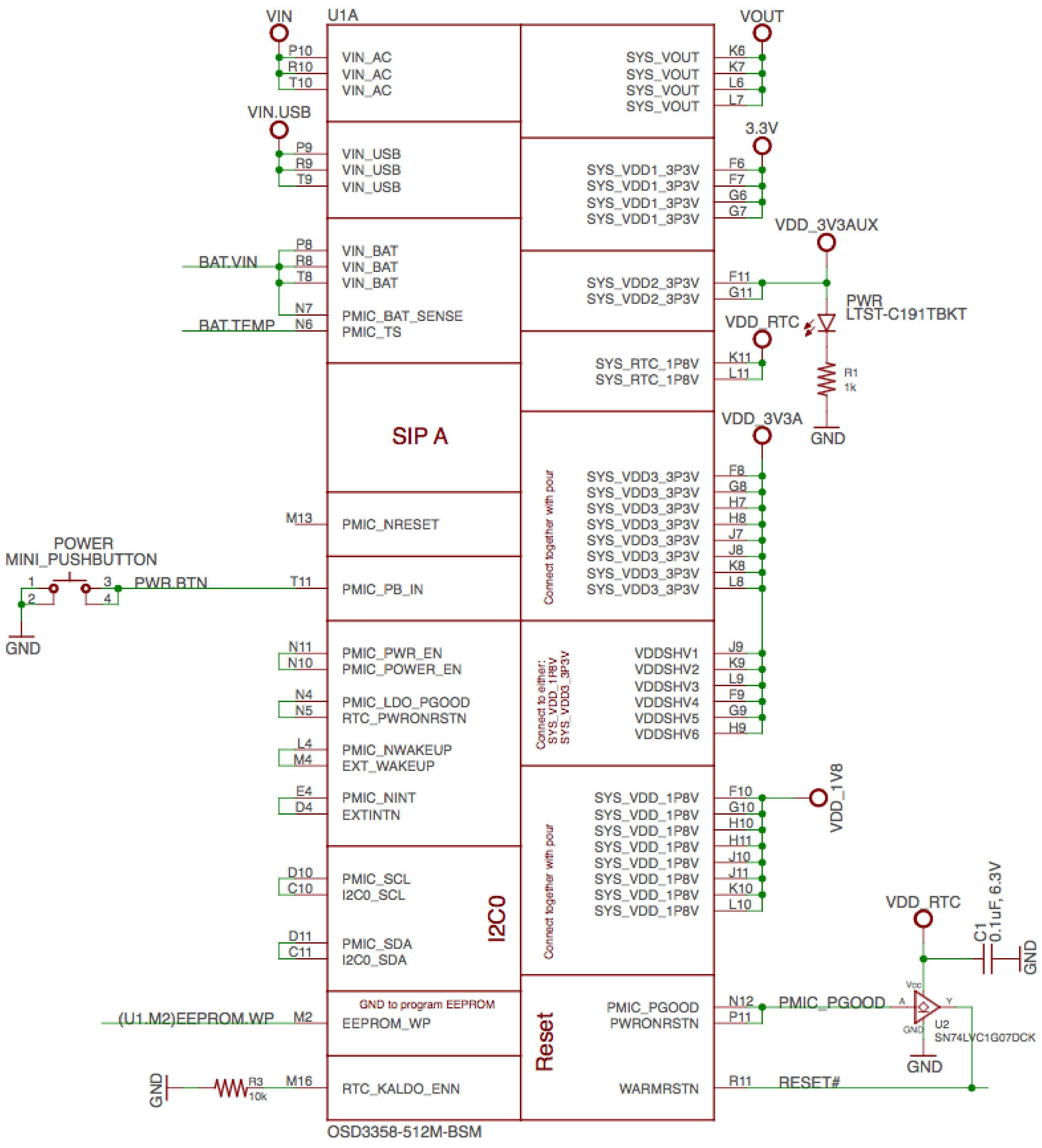


Fig. 2.159: SiP A OSD3358 SiP System and Power Signals

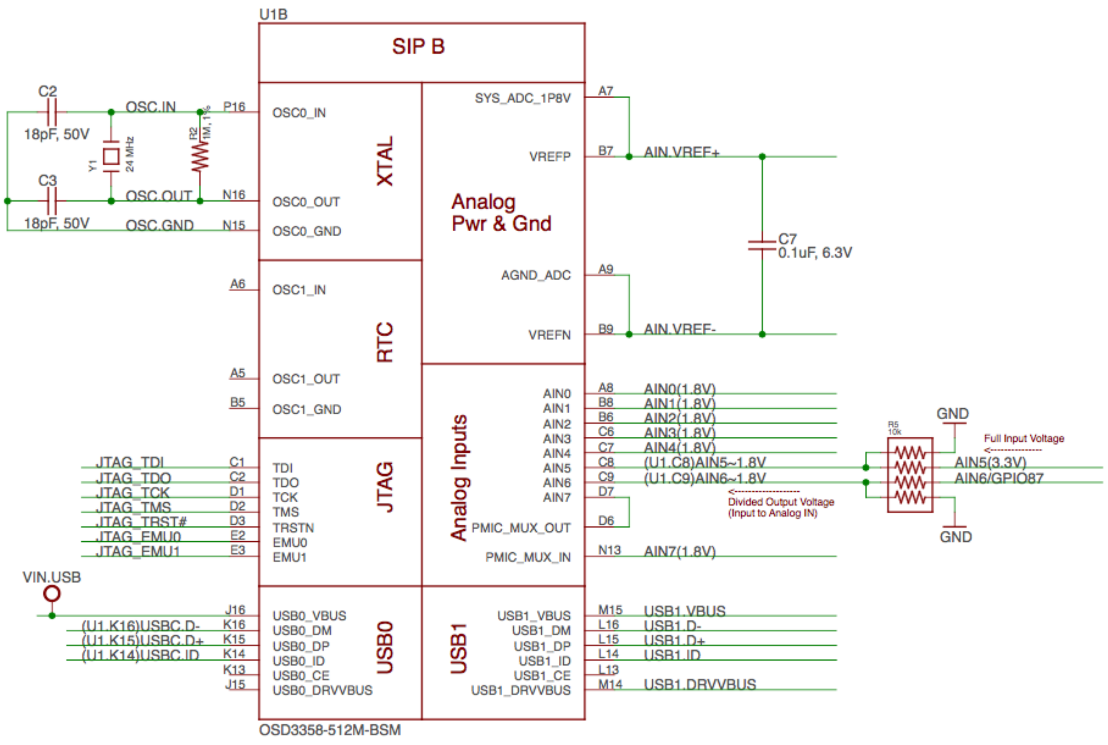


Fig. 2.160: SiP B OSD3358 SiP JTAG, USB & Analog Signals

SiP D OSD3358 SiP System Boot Configuration

SiP E OSD3358 SiP Power Signals

SiP F OSD3358 SiP Power Signals

MicroSD Connection

The Micro Secure Digital (microSD) connector design is highlighted in Figure 35.

USB Connector

The USB connector design is highlighted in Figure 36.

Note that there is an ID pin for dual-role (host/client) functionality. The hardware fully supports it, but care should be taken to ensure the kernel in use is either statically or dynamically configured to recognize and utilize the proper mode.

Power Button Design

The power button design is highlighted in Figure 37.

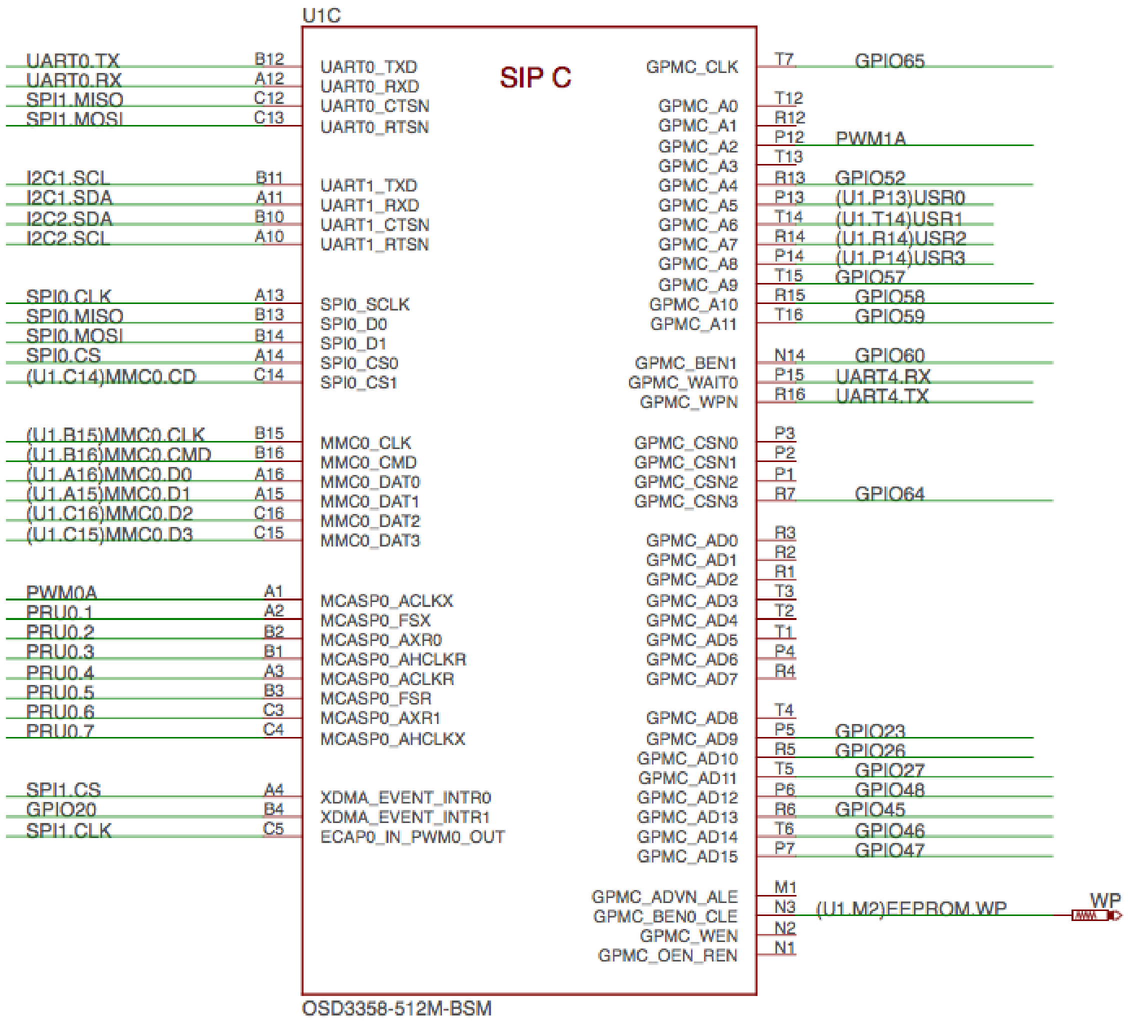


Fig. 2.161: SiP C OSD3358 SiP Peripheral Signals

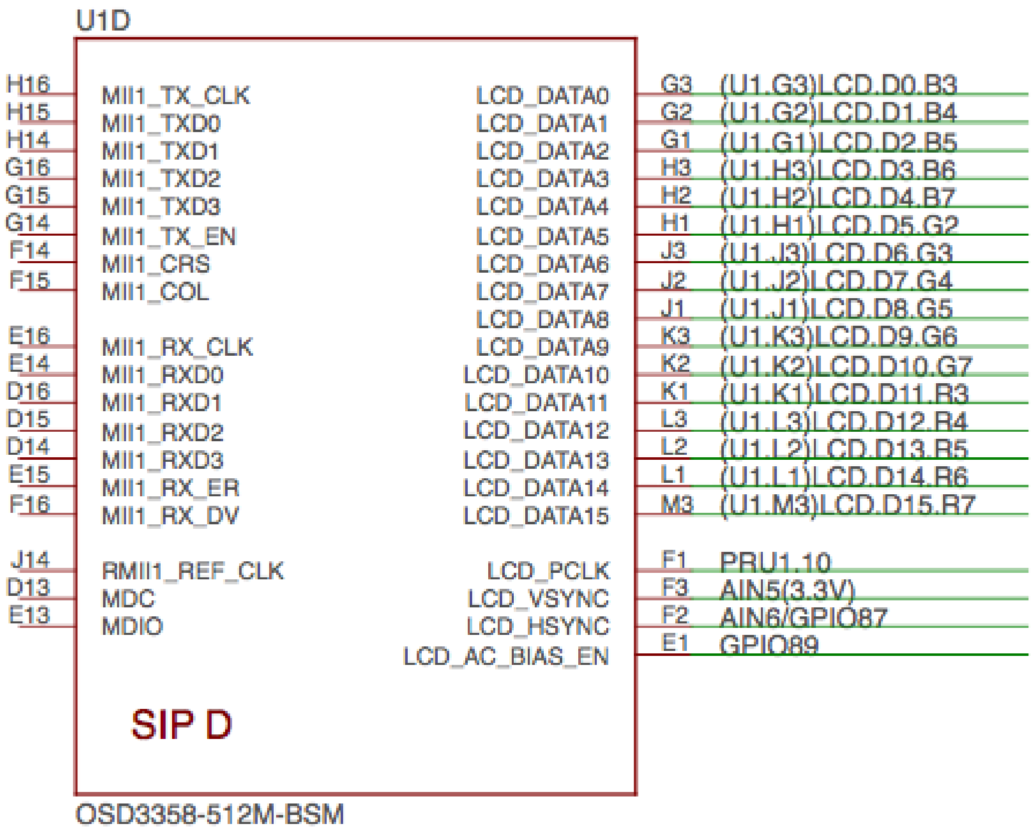

U1D
H16 MII1_TX_CLK
H15 MII1_TXD0
H14 MII1_TXD1
G16 MII1_TXD2
G15 MII1_TXD3
G14 MII1_TX_EN
F14 MII1_CRS
F15 MII1_COL
E16 MII1_RX_CLK
E14 MII1_RXD0
D16 MII1_RXD1
D15 MII1_RXD2
D14 MII1_RXD3
E15 MII1_RX_ER
F16 MII1_RX_DV
J14 RMII1_REF_CLK
D13 MDC
E13 MDIO
LCD_DATA0 G3 (U1.G3)LCD.D0.B3
LCD_DATA1 G2 (U1.G2)LCD.D1.B4
LCD_DATA2 G1 (U1.G1)LCD.D2.B5
LCD_DATA3 H3 (U1.H3)LCD.D3.B6
LCD_DATA4 H2 (U1.H2)LCD.D4.B7
LCD_DATA5 H1 (U1.H1)LCD.D5.G2
LCD_DATA6 J3 (U1.J3)LCD.D6.G3
LCD_DATA7 J2 (U1.J2)LCD.D7.G4
LCD_DATA8 J1 (U1.J1)LCD.D8.G5
LCD_DATA9 K3 (U1.K3)LCD.D9.G6
LCD_DATA10 K2 (U1.K2)LCD.D10.G7
LCD_DATA11 K1 (U1.K1)LCD.D11.R3
LCD_DATA12 L3 (U1.L3)LCD.D12.R4
LCD_DATA13 L2 (U1.L2)LCD.D13.R5
LCD_DATA14 L1 (U1.L1)LCD.D14.R6
LCD_DATA15 M3 (U1.M3)LCD.D15.R7
LCD_PCLK F1 PRU1.10
LCD_VSYNC F3 AIN5(3.3V)
LCD_HSYNC F2 AIN6/GPIO87
LCD_AC_BIAS_EN E1 GPIO89
SIP D
OSD3358-512M-BSM

Fig. 2.162: SiP D OSD3358 SiP System Boot Configuration

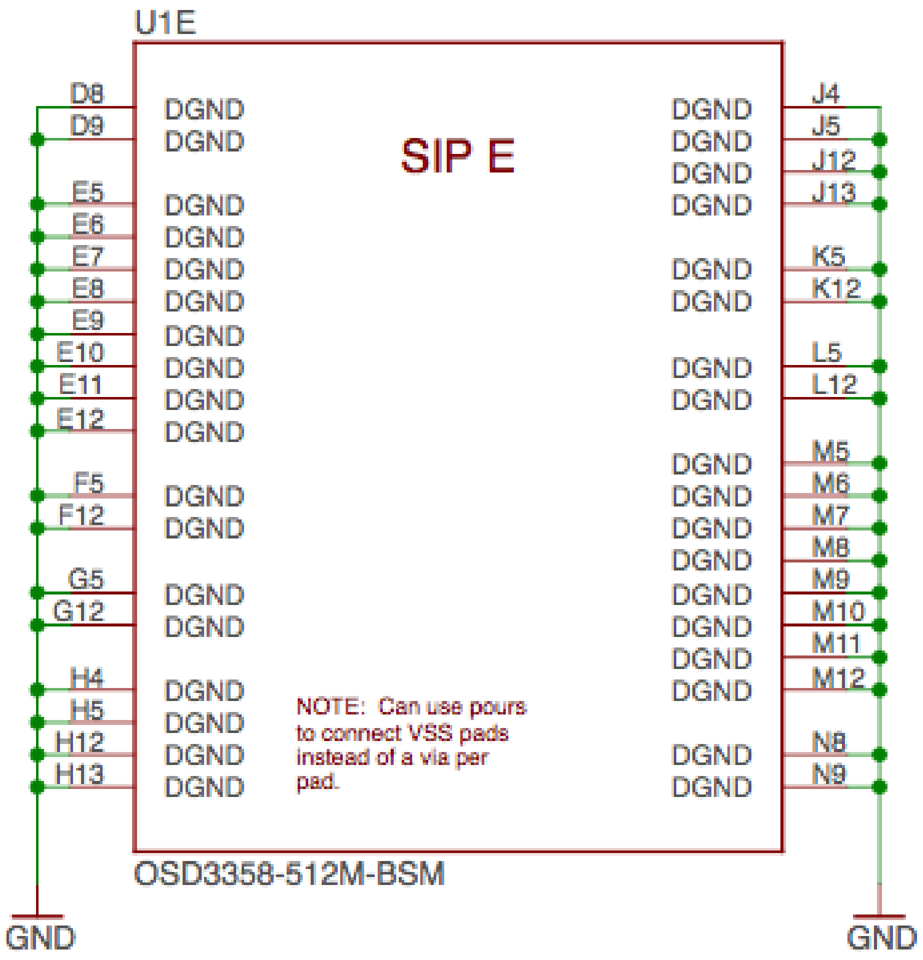
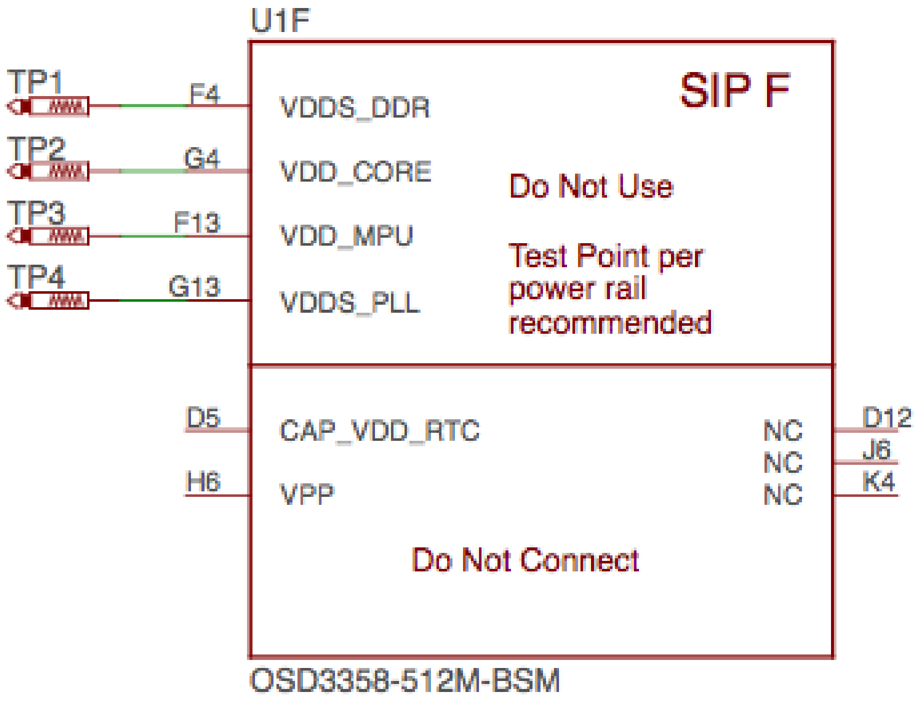


Fig. 2.163: SiP E OSD3358 SiP Power Signals



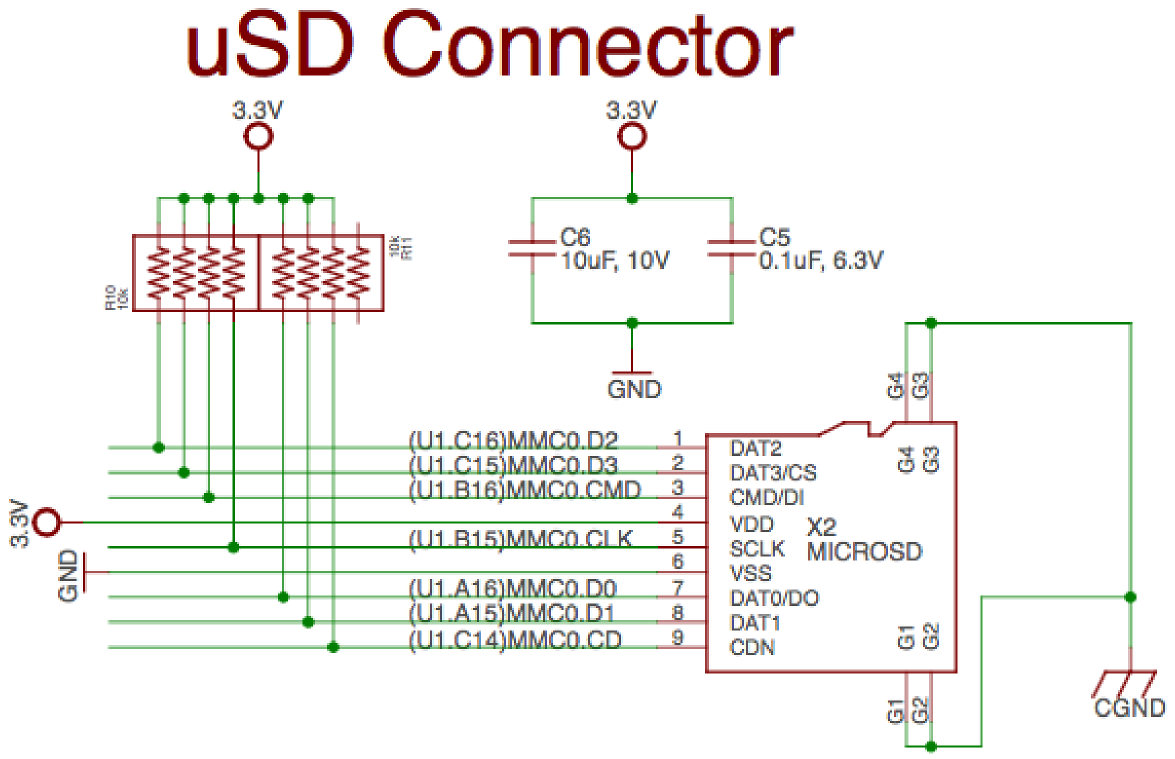


Fig. 2.164: microSD Connections

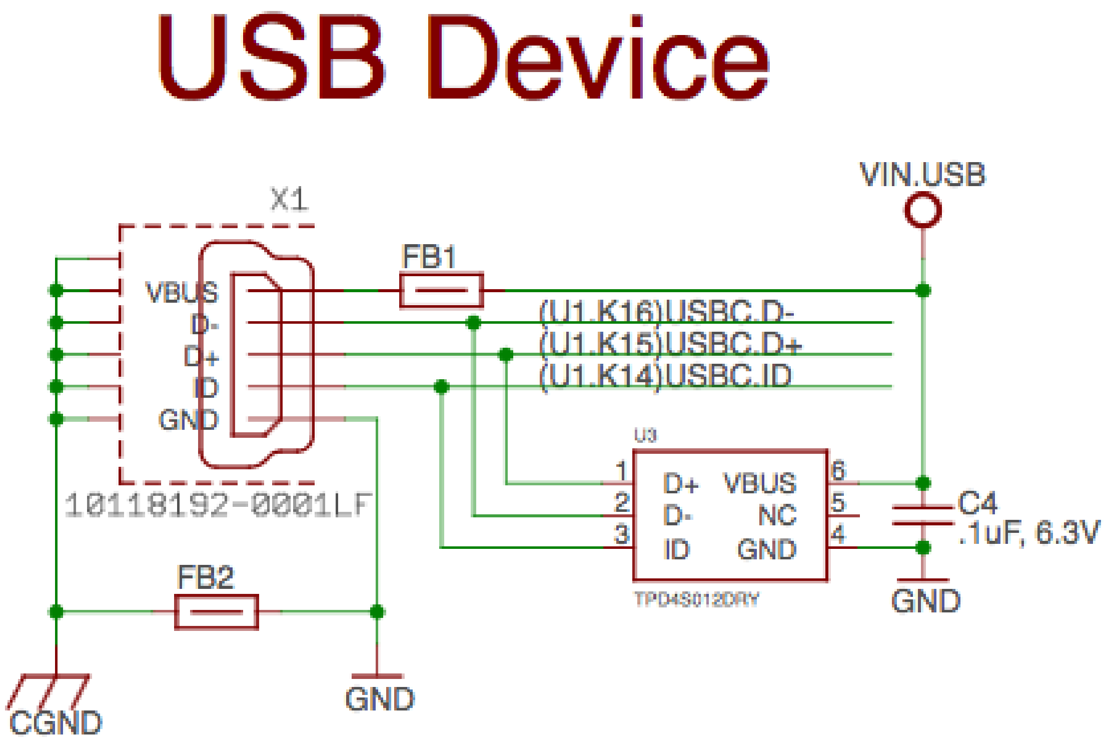


Fig. 2.165: USB Connection

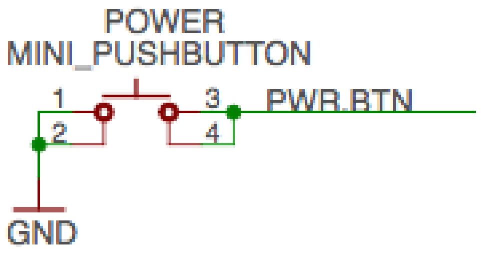


Fig. 2.166: Power Button

User LEDs

There are four user programmable LEDs on PocketBeagle. The design is highlighted in Figure 38. Table 6 Provides the LED control signals and pins. A logic level of "1" will cause the LEDs to turn on.

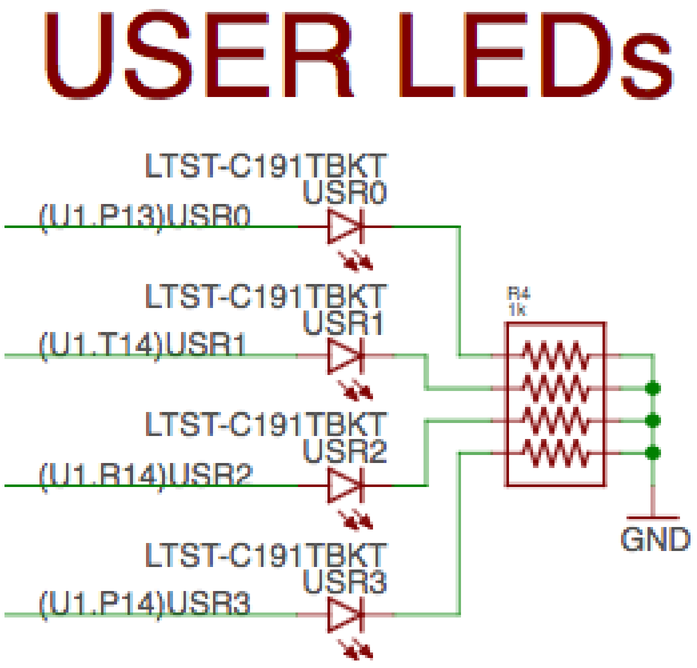


Fig. 2.167: User LEDs

Table 2.62: User LED Control Signals/Pins

LED	Signal Name	Proc Ball	SiP Ball
USR0	GPIO1_21	V15	P13
USR1	GPIO1_22	U15	T14
USR2	GPIO1_23	T15	R14
USR3	GPIO1_24	V16	P14

JTAG Pads

There are 7 pads on the bottom of PocketBeagle to connect JTAG for debugging. The design is highlighted in Figure 39. More information regarding JTAG debugging can be found at 'www.ti.com/jtag'

PRU-ICSS

The Programmable Real-Time Unit Subsystem and Industrial Communication SubSystem (PRU-ICSS) module is located inside the AM3358 processor, which is inside the Octavo Systems SiP. Commonly referred to as just the "PRU", this little subsystem will unleash a lot of performance for you to use in your application. Consisting of dual 32-bit RISC cores (Programmable Real-Time Units, or PRUs), data and instruction memories, internal peripheral modules, and an interrupt controller (INTC). The programmable nature of the PRU-ICSS, along with their access to pins, events and all SoC resources, provides flexibility in implementing fast real-time responses, specialized data handling operations, custom peripheral interfaces, and in offloading tasks from the other processor cores of the system-on-chip (SoC). Access to these pins is provided by PocketBeagle's expansion

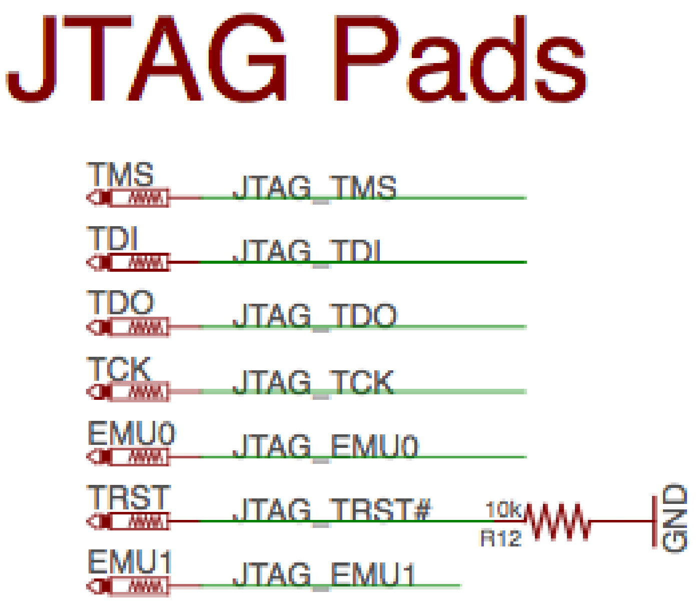


Fig. 2.168: JTAG Pads Design

headers and is multiplexed with other functions on the board. Access is not provided to all of the available pins.

Some getting started information can be found on https://beagleboard.org/pru.

Additional documentation is located on the Texas Instruments website at processors.wiki.ti.com/index.php/PRU-ICSS and also located at http://github.com/beagleboard/am335x_pru_package.

Example projects using the PRU-ICSS can be found in *PRU Cookbook*.

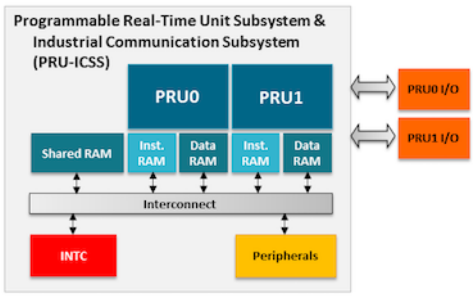
PRU-ICSS Features The features of the PRU-ICSS include:

Two independent programmable real-time (PRU) cores:

- 32-Bit Load/Store RISC architecture
- 8K Byte instruction RAM (2K instructions) per core
- 8K Bytes data RAM per core
- 12K Bytes shared RAM
- Operating frequency of 200 MHz
- PRU operation is little endian similar to ARM processor
- All memories within PRU-ICSS support parity
- Includes Interrupt Controller for system event handling
- Fast I/O interface

– 16 input pins and 16 output pins per PRU core. (Not all of these are accessible on the PocketBeagle. Please check the Pin Table below for PRU-ICSS features available through the P1 and P2 headers.)

PRU-ICSS Block Diagram Figure below is a high level block diagram of the PRU-ICSS.



PRU-ICSS Pin Access Both PRU 0 and PRU1 are accessible from the expansion headers. Listed below are the ports that can be accessed on each PRU.

Table 6. below shows which PRU-ICSS signals can be accessed on PocketBeagle and on which connector and pins on which they are accessible. Some signals are accessible on the same pins.

Use scroll bar at bottom of chart to see additional features in columns to the right. When printing this document, you will need to print this chart separately.

Table 2.63: PRU0 and PRU1 Access

Header.Pin	Silkscreen	Processor Ball	SiP Ball	Mode3	Mode4	Mode5	Mode6	Note
P1.02	A6/87	R5	F2			pr1_pru1_pru_r30_9 (Output)	pr1_pru1_pru_r31_9 (Input)	
P1.04	89	R6	E1			pr1_pru1_pru_r30_11 (Output)	pr1_pru1_pru_r31_11 (Input)	
P1.06	SPI0_CS	A16	A14		pr1_uart0_txd (Output)			UART Transmit Data
P1.08	SPI0_CLK	A17	A13		pr1_uart0_cts_n (Input)			UART Clear to Send
P1.10	SPI0_MISO	B17	B13		pr1_uart0_rts_n (Output)			UART Request to Send
P1.12	SPI0_MOSI	B16	B14		pr1_uart0_rxd (Input)			UART Receive Data
P1.20	20	D14	B4			pr1_pru0_pru_r31_16 (Input)		
P1.26	I2C2_SDA	D18	B10			pr1_uart0_cts_n (Input)		UART Clear to Send
P1.28	I2C2_SCL	D17	A10			pr1_uart0_rts_n (Output)		UART Request to Send
P1.29	PRU0_7	A14	C4			pr1_pru0_pru_r30_7 (Output)	pr1_pru0_pru_r31_7 (Input)	
P1.30	U0_TX	E16	B12			pr1_pru1_pru_r30_15 (Output)	pr1_pru1_pru_r31_15 (Input)	
P1.31	PRU0_4	B12	A3			pr1_pru0_pru_r30_4 (Output)	pr1_pru0_pru_r31_4 (Input)	
P1.32	U0_RX	E15	A12			pr1_pru1_pru_r30_14 (Output)	pr1_pru1_pru_r31_14 (Input)	
P1.33	PRU0_1	B13	A2			pr1_pru0_pru_r30_1 (Output)	pr1_pru0_pru_r31_1 (Input)	
P1.35	P1.10	V5	F1			pr1_pru1_pru_r30_10 (Output)	pr1_pru1_pru_r31_10 (Input)	
P1.36	PWM0A	A13	A1			pr1_pru0_pru_r30_0 (Output)	pr1_pru0_pru_r31_0 (Input)	
P2.09	I2C1_SCL	D15	B11			pr1_uart0_txd (Output)	pr1_pru0_pru_r31_16 (Input)	UART Transmit Data
P2.11	I2C1_SDA	D16	A11			pr1_uart0_rxd (Input)	pr1_pru1_pru_r31_16 (Input)	UART Receive Data
P2.17	65	V12	T7			pr1_mdio_mdclk		MDIO Clk
P2.18	47	U13	P7			pr1_ecap0_ecap_capin_apwm_o	pr1_pru0_pru_r31_15 (Input)	Enhanced capture input or Auxiliary PWM out
P2.20	64	T13	R7			pr1_mdio_data		MDIO Data
P2.22	46	V13	T6				pr1_pru0_pru_r31_14 (Input)	
P2.24	48	T12	P6				pr1_pru0_pru_r30_14 (Output)	
P2.28	PRU0_6	D13	C3			pr1_pru0_pru_r30_6 Output)	pr1_pru0_pru_r31_6 (Input)	
P2.29	SPI1_CLK	C18	C5	pr1_ecap0_ecap_capin_apwm_o				Enhanced capture input or Auxiliary PWM out
P2.30	PRU0_3	C12	B1			pr1_pru0_pru_r30_3 (Output)	pr1_pru0_pru_r31_3 (Input)	
P2.31	SPI1_CS	A15	A4			pr1_pru1_pru_r31_16 (Input)		
P2.32	PRU0_2	D12	B2			pr1_pru0_pru_r30_2 (Output)	pr1_pru0_pru_r31_2 (Input)	
P2.33	45	R12	R6				pr1_pru0_pru_r30_15 (Output)	
P2.34	PRU0_5	C13	B3			pr1_pru0_pru_r30_5 (Output)	pr1_pru0_pru_r31_5 (Input)	
P2.35	A5/86	U5	F3			pr1_pru1_pru_r30_8 (Output)	pr1_pru1_pru_r31_8 (Input)	

2.6.7 Connectors

This section describes each of the connectors on the board.

Expansion Header Connectors

The expansion interface on the board is comprised of two 36 pin connectors. The two Expansion Header Connectors on PocketBeagle are labeled P1 and P2. The connections are a standard 100 mil distance so that they can be compatible with many standard expansion items. The silkscreen for the headers on the bottom of the board provides the easiest way to identify them. See Figure 41.

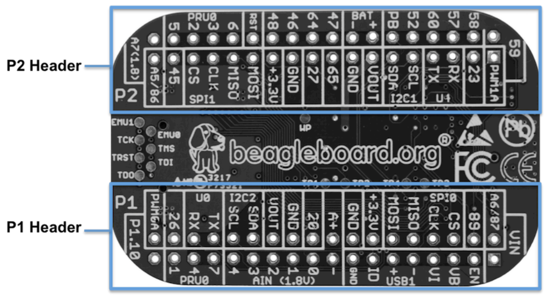


Fig. 2.169: Expansion Headers for PocketBeagle

All signals on the expansion headers are **3.3V** unless otherwise indicated.

Note:

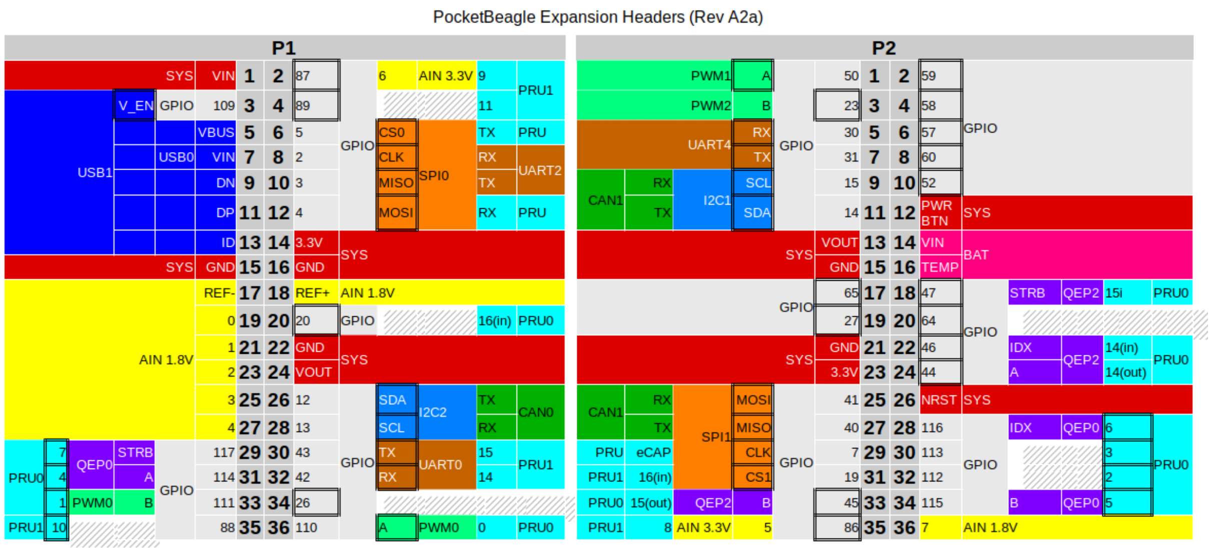
- Do not connect 5V logic level signals to these pins or the board will be damaged.
- DO NOT APPLY VOLTAGE TO ANY I/O PIN WHEN POWER IS NOT SUPPLIED TO THE BOARD. IT WILL DAMAGE THE PROCESSOR AND VOID THE WARRANTY.
- NO PINS ARE TO BE DRIVEN UNTIL AFTER THE NRESET LINE GOES HIGH.

Figure 42 shows a color coded chart with an overview of the most popular functions of PocketBeagle's Expansion Header pins. The Header Pin tables in Sections 7.1.1 and 7.1.2 show the full pin assignments for each header.

P1 Header

Figure 43 shows the schematic diagram for the P1 Header.

Use scroll bar at bottom of chart to see additional features in columns to the right. When printing this document you will need to print this chart separately.



☐	Mode enabled by default after kernel boot, if more than one is possible
SYS	Power and other system control signals
GPIO	General purpose inputs and outputs
AIN	Analog inputs, note that these are all enabled by default after kernel boot
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit bus
UART	Serial port
PWM	Pulse width modulator
QEP	Quadrature encoder peripheral
PRU	Programmable real-time unit input, output, or peripheral
CAN	Controller Area Network – requires external PHY
USB	Universal Serial Bus
BAT	Battery

Fig. 2.170: Expansion Header Popular Functions - Color Coded

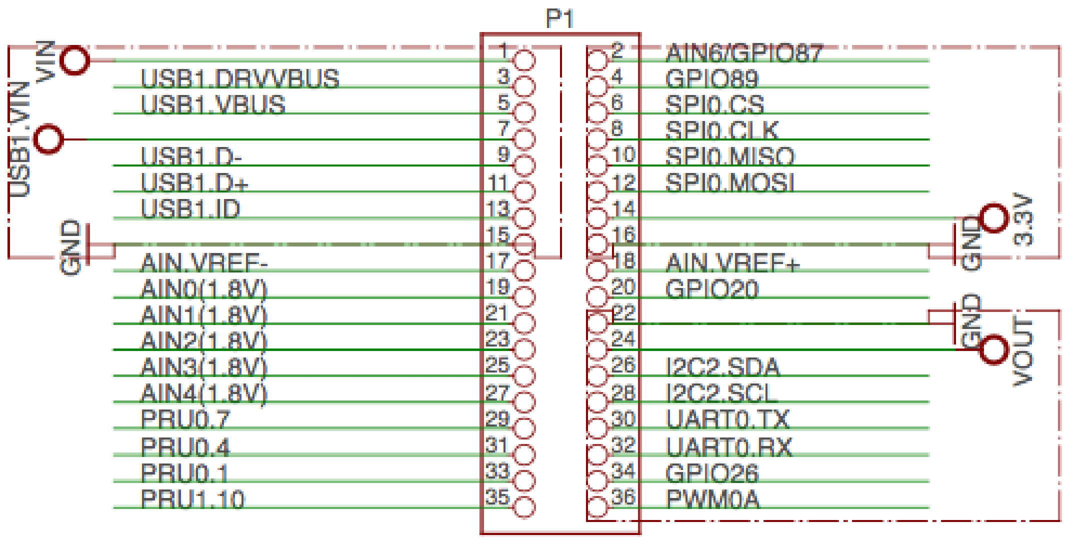


Table 2.64: P1 Header Pinout

Header.Pin	Silkscreen	PocketBeagle wiring	Proc Ball	SiP Ball	Mode0 (Name)	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
P1.01	VIN	P1.01 (VIN)		P10 & R10 & T10	VIN							
P1.02	A6/87	P1.02 (AIN6/GPIO87)	A8	C9	ain6							
P1.02	A6/87	P1.02 (AIN6/GPIO87)	R5	F2	lcd_hsync	gpmc_a9	gpmc_a2	pr1_edio_data_in3	pr1_edio_data_out3	pr1_pru1_pru_r30_9	pr1_pru1_pru_r31_9	gpio2_23
P1.03	USB1_EN	P1.03 (USB1-DRVVBUS)	F15	M14	USB1_DRVVBUS	•	•	•	•	•	•	gpio3_13
P1.04	89	P1.04 (PRU1.11)	R6	E1	lcd_ac_bias_en	gpmc_a11	pr1_mii1_crs	pr1_edio_data_in5	pr1_edio_data_out5	pr1_pru1_pru_r30_11	pr1_pru1_pru_r31_11	gpio2_25
P1.05	USB1_VB	P1.05 (USB1-VBUS)	T18	M15	USB1_VBUS	•	•	•	•	•	•	•
P1.06	SPI0_CS	P1.06 (SPI0-CS)	A16	A14	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci	pr1_uart0_txd	pr1_edio_data_in1	pr1_edio_data_out1	gpio0_5
P1.07	USB1_VI	P1.07 (VIN-USB)		P9 &R9 &T9	VIN-USB							
P1.08	SPI0_CLK	P1.08 (SPI0-CLK)	A17	A13	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A	pr1_uart0_cts_n	pr1_edio_sof	EMU2	gpio0_02
P1.09	USB1 -	P1.09 (USB1-DN)	R18	L16	USB1_DM	•	•	•	•	•	•	•
P1.10	SPI0_MISO	P1.10 (SPI0-MISO)	B17	B13	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B	pr1_uart0_rts_n	pr1_edio_latch_in	EMU3	gpio0_3
P1.11	USB1 +	P1.11 (USB1-DP)	R17	L15	USB1_DP	•	•	•	•	•	•	•
P1.12	SPI0_MOSI	P1.12 (SPI0-MOSI)	B16	B14	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone_input	pr1_uart0_rxd	pr1_edio_data_in0	pr1_edio_data_out0	gpio0_04
P1.13	USB1_ID	P1.13 (USB1-ID)	P17	L14	USB1_ID	•	•	•	•	•	•	•
P1.14	+3.3V	P1.14 (VOUT-3.3V)		F6 & F7 & G6 & G7	VOUT-3.3V							
P1.15	USB1_GND	P1.15 (GND)			GND							
P1.16	GND	P1.16 (GND)			GND							
P1.17	AIN(1.8V)-	P1.17 (VREFN)	A9	B9	VREFN							
P1.18	AIN(1.8V)A+	P1.18 (VREFP)	B9	B7	VREFP							
P1.19	AIN(1.8V)0	P1.19 (AIN0-1.8V)	B6	A8	ain0							
P1.20	20	P1.20 (PRU0.16)	D14	B4	xdma_event_intr1	•	tclkin	clkout2	timer7	pr1_pru0_pru_r31_16	EMU3	gpio0_20
P1.21	AIN(1.8V)1	P1.21 (AIN1-1.8V)	C7	B8	ain1							
P1.22	GND	P1.22 (GND)			GND							
P1.23	AIN(1.8V)2	P1.23 (AIN2-1.8V)	B7	B6	ain2							

continues on next page

Table 2.64 – continued from previous page

Header.Pin	Silkscreen	PocketBeagle wiring	Proc Ball	SiP Ball	Mode0 (Name)	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
P1.24	VOUT	P1.24 (VOUT-5V)		K6 & K7 & L6 & L7	VOUT-5V							
P1.25	AIN(1.8V)3	P1.25 (AIN3-1.8V)	A7	C6	ain3							
P1.26	I2C2_SDA	P1.26 (I2C2-SDA)	D18	B10	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0	pr1_uart0_cts_n	pr1_edc_latch0_in	gpio0_12
P1.27	AIN(1.8V)4	P1.27 (AIN4-1.8V)	C8	C7	ain4							
P1.28	I2C2_SCL	P1.28 (I2C2-SCL)	D17	A10	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1	pr1_uart0_rts_n	pr1_edc_latch1_in	gpio0_13
P1.29	PRU0_7	P1.29 (PRU0.7)	A14	C4	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4	pr1_pru0_pru_r30_7	pr1_pru0_pru_r31_7	gpio3_21
P1.30	U0_TX	P1.30 (UART0-TX)	E16	B12	uart0_txd	spi1_cs1	dcan0_rx	I2C2_SCL	eCAP1_in_PWM1_out	pr1_pru1_pru_r30_15	pr1_pru1_pru_r31_15	gpio1_11
P1.31	PRU0_4	P1.31 (PRU0.4)	B12	A3	mcasp0_aclkr	eQEP0A_in	mcasp0_axr2	mcasp1_aclkx	mmc0_sdwp	pr1_pru0_pru_r30_4	pr1_pru0_pru_r31_4	gpio3_18
P1.32	U0_RX	P1.32 (UART0-RX)	E15	A12	uart0_rxd	spi1_cs0	dcan0_tx	I2C2_SDA	eCAP2_in_PWM2_out	pr1_pru1_pru_r30_14	pr1_pru1_pru_r31_14	gpio1_10
P1.33	PRU0_1	P1.33 (PRU0.1)	B13	A2	mcasp0_fsx	ehrpwm0B	•	spi1_d0	mmc1_sdcd	pr1_pru0_pru_r30_1	pr1_pru0_pru_r31_1	gpio3_15
P1.34	26	P1.34 (GPIO0.26)	T11	R5	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6	ehrpwm2_tripzone_input	pr1_mii0_txen	•	gpio0_26
P1.35	P1.10	P1.35 (PRU1.10)	V5	F1	lcd_pclk	gpmc_a10	pru_mii0_crs	pr1_edio_data_in4	pr1_edio_data_out4	pr1_pru1_pru_r30_10	pr1_pru1_pru_r31_10	gpio2_24
P1.36	PWM0A	P1.36 (PWM0A)	A13	A1	mcasp0_aclkx	ehrpwm0A	•	spi1_sclk	mmc0_sdcd	pr1_pru0_pru_r30_0	pr1_pru0_pru_r31_0	gpio3_14

P2 Header

Figure 44 shows the schematic diagram for the P2 Header.

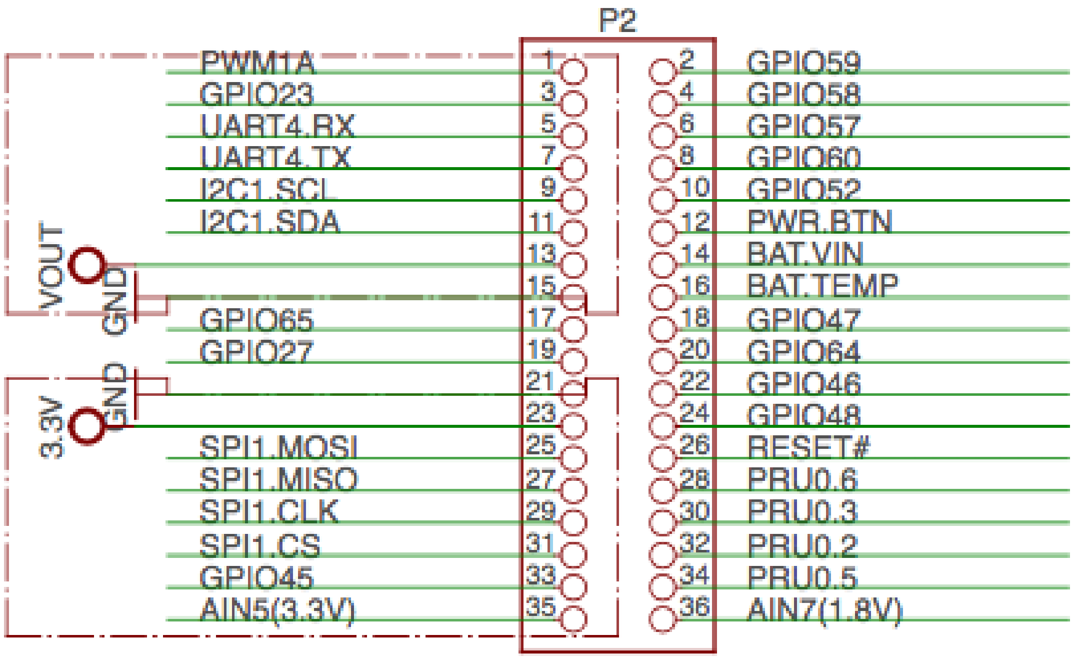


Fig. 2.171: P2 Header

Use scroll bar at bottom of chart to see additional features in columns to the right. When printing this document you will need to print this chart separately.

Table 2.65: P2 Header Pinout

Header.Pin	Silkscreen	PocketBeagle wiring	Proc Ball	SiP Ball	Mode0 (Name)	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
P2.01	PWM1A	P2.01 (PWM1A)	U14	P12	gpmc_a2	gmii2_txd3	rgmii2_td3	mmc2_dat1	gpmc_a18	pr1_mii1_txd2	ehrpwm1A	gpio1_18
P2.02	59	P2.02 (GPIO1.27)	V17	T16	gpmc_a11	gmii2_rxd0	rgmii2_rd0	rmii2_rxd0	gpmc_a27	pr1_mii1_rxer	mcasp0_axr1	gpio1_27
P2.03	23	P2.03 (GPIO0.23)	T10	P5	gpmc_d9	lcd_data22	mmc1_dat1	mmc2_dat5	ehrpwm2B	pr1_mii0_col	•	gpio0_23
P2.04	58	P2.04 (GPIO1.26)	T16	R15	gpmc_a10	gmii2_rxd1	rgmii2_rd1	rmii2_rxd1	gpmc_a26	pr1_mii1_rxdv	mcasp0_axr0	gpio1_26
P2.05	U1_RX	P2.05 (UART4-RX)	T17	P15	gpmc_wait0	gmii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd	pr1_mii1_col	uart4_rxd	gpio0_30
P2.06	57	P2.06 (GPIO1.25)	U16	T15	gpmc_a9	gmii2_rxd2	rgmii2_rd2	mmc2_dat7 / rmii2_crs_dv	gpmc_a25	pr1_mii_mr1_clk	mcasp0_fsx	gpio1_25
P2.07	U1_TX	P2.07 (UART4-TX)	U17	R16	gpmc_wp	gmii2_rxerr	gpmc_csn5	rmii2_rxerr	mmc2_sdcd	pr1_mii1_txen	uart4_txd	gpio0_31
P2.08	60	P2.08 (GPIO1.28)	U18	N14	gpmc_be1n	gmii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir	pr1_mii1_rxlink	mcasp0_aclkr	gpio1_28
P2.09	I2C1_SCL	P2.09 (I2C1-SCL)	D15	B11	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL	•	pr1_uart0_txd	pr1_pru0_pru_r31_16	gpio0_15
P2.10	52	P2.10 (GPIO1.20)	R14	R13	gpmc_a4	gmii2_txd1	rgmii2_td1	rmii2_txd1	gpmc_a20	pr1_mii1_txd0	eQEP1A_in	gpio1_20
P2.11	I2C1_SDA	P2.11 (I2C1-SDA)	D16	A11	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA	•	pr1_uart0_rxd	pr1_pru1_pru_r31_16	gpio0_14
P2.12	PB	P2.12 (POWER_BTN)		T11	POWER							
P2.13	VOUT	P2.13 (VOUT-5V)		K6, K7, L6, L7	VOUT-5V							
P2.14	BAT +	P2.14 (VIN-BAT)		P8, R8, T8	VIN-BAT							
P2.15	GND	P2.15 (GND)			GND							
P2.16	BAT -	P2.16 (BAT-TEMP)		N6	BAT-TEMP							
P2.17	65	P2.17 (GPIO2.1)	V12	T7	gpmc_clk	lcd_memory_clk	gpmc_wait1	mmc2_clk	pr1_mii1_crs	pr1_mdio_mdclk	mcasp0_fsr	gpio2_01
P2.18	47	P2.18 (PRU0.15i)	U13	P7	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3	eQEP2_strobe	pr1_ecap0_ecap_capin_apwm_o	pr1_pru0_pru_r31_15	gpio1_15P
P2.19	27	P2.19 (GPIO0.27)	U12	T5	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7	ehrpwm0_synco	pr1_mii0_txd3	•	gpio0_27
P2.20	64	P2.20 (GPIO2.0)	T13	R7	gpmc_csn3	gpmc_a3	rmii2_crs_dv	mmc2_cmd	pr1_mii0_crs	pr1_mdio_data	EMU4	gpio2_00
P2.21	GND	P2.21 (GND)			GND							
P2.22	46	P2.22 (GPIO1.14)	V13	T6	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2	eQEP2_index	pr1_mii0_txd0	pr1_pru0_pru_r31_14	gpio1_14
P2.23	+3.3V	P2.23 (VOUT-3.3V)		F6 & F7 & G6 & G7	VOUT-3.3V							
P2.24	48	P2.24 (GPIO1.12)	T12	P6	gpmc_ad12	lcd_data19	mmc1_dat4	mmc2_dat0	eQEP2A_in	pr1_mii0_txd2	pr1_pru0_pru_r30_14	gpio1_12

continues on next page

Table 2.65 – continued from previous page

Header.Pin	Silkscreen	PocketBeagle wiring	Proc Ball	SiP Ball	Mode0 (Name)	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
P2.25	SPI1_MOSI	P2.25 (SPI1-MOSI)	E17	C13	uart0_rtsn	uart4_txd	dcan1_rx	I2C1_SCL	spi1_d1	spi1_cs0	pr1_edc_sync1_out	gpio1_09
P2.26	RST	P2.26 (NRE-SET)	A10	R11	nRE-SETIN_OUT	•	•	•	•	•	•	•
P2.27	SPI1_MISO	P2.27 (SPI1-MISO)	E18	C12	uart0_ctsn	uart4_rxd	dcan1_tx	I2C1_SDA	spi1_d0	timer7	pr1_edc_sync0_out	gpio1_08
P2.28	PRU0_6	P2.28 (PRU0.6)	D13	C3	mcasp0_axr1	eQEP0_index	•	mcasp1_axr0	EMU3	pr1_pru0_pru_r30_6	pr1_pru0_pru_r31_6	gpio3_20
P2.29	SPI1_CLK	P2.29 (SPI1-CLK)	C18	C5	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	gpio0_7
P2.30	PRU0_3	P2.30 (PRU0.3)	C12	B1	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out	pr1_pru0_pru_r30_3	pr1_pru0_pru_r31_3	gpio3_17
P2.31	SPI1_CS	P2.31 (SPI1-CS1)	A15	A4	xdma_event_intr0	•	timer4	clkout1	spi1_cs1	pr1_pru1_pru_r31_16	EMU2	gpio0_19
P2.32	PRU0_2	P2.32 (PRU0.2)	D12	B2	mcasp0_axr0	ehrpwm0_tripzone_input	•	spi1_d1	mmc2_sdcd	pr1_pru0_pru_r30_2	pr1_pru0_pru_r31_2	gpio3_16
P2.33	45	P2.33 (GPIO1.13)	R12	R6	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1	eQEP2B_in	pr1_mii0_txd1	pr1_pru0_pru_r30_15	gpio1_13
P2.34	PRU0_5	P2.34 (PRU0.5)	C13	B3	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2	pr1_pru0_pru_r30_5	pr1_pru0_pru_r31_5	gpio3_19
P2.35	A5/86	P2.35 (AIN5/GPIO86)	B8	C8	ain5							
P2.35	A5/86	P2.35 (AIN5/GPIO86)	U5	F3	lcd_vsync	gpmc_a8	gpmc_a1	pr1_edio_data_in2	pr1_edio_data_out2	pr1_pru1_pru_r30_8	pr1_pru1_pru_r31_8	gpio2_22
P2.36	A7(1.8)	P2.36 (AIN7)		N13	ain7							

mikroBUS socket connections

mikroBUS and, by extension "mikroBUS Click boards", are trademarks of MikroElektronika. We do not make any claims of compatibility nor adherence to their specification. We've just seen that many of the Click boards "just work".

The Expansion Headers on PocketBeagle have been designed to accept up to two Click Boards added to the header pins at the same time. This provides an exciting opportunity to add functionality easily to PocketBeagle from 'hundreds of existing add-on Click Boards'.

The mikroBUS standard comprises a pair of 1×8 female headers with a standardized pin configuration. The pinout (always laid out in the same order) consists of three groups of communications pins (SPI, UART and I2C), six additional pins (PWM, Interrupt, Analog input, Reset and Chip select), and two power groups (+3.3V and 5V).

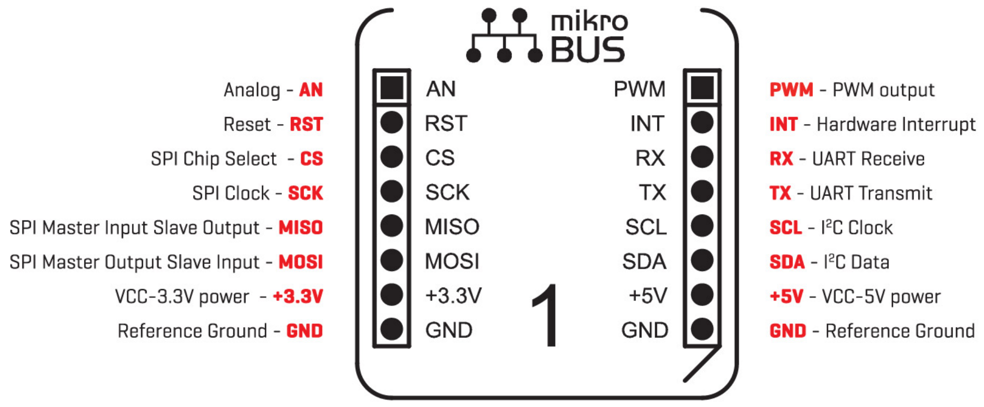


Fig. 2.172: mikroBUS

The Expansion Header pin alignment enables 2 Click Boards on the top side of PocketBeagle using the inside rails of the headers. This leaves the outside rails open to be accessed from either the top or the bottom of PocketBeagle. Place each Click Board into the position shown in Figure 46, with one Click Board facing each direction. When choosing Click boards, make sure you are checking that they meet the 3.3V requirements for PocketBeagle. A growing number of community members are trying out various Click Boards and posting results on the 'PocketBeagle Wiki mikroBus Click Boards page'.

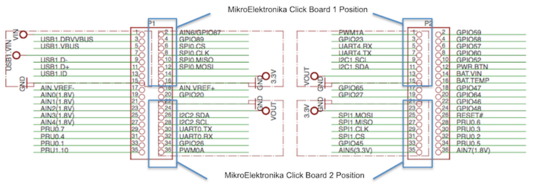
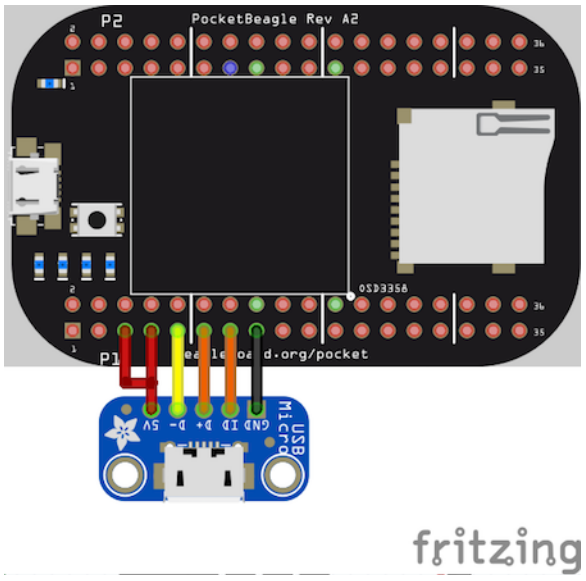


Fig. 2.173: PocketBeagle Both Headers

Setting up an additional USB Connection

You can add an additional USB connection to PocketBeagle easily by connecting a microUSB breakout. By default in the current software, the system should be configured to use this port as a host. Keep up to date on this project on the 'PocketBeagle Wiki FAQ'.



2.6.8 PocketBeagle Cape Support

This is a placeholder for recommendations for those building their own PocketBeagle Cape designs. If you'd like to join the conversation 'check out the discussion on the forum for PocketBeagle'

See also PocketBeagle under 'BeagleBoard Capes'

2.6.9 PocketBeagle Mechanical

9.1 Dimensions and Weight

Size: 2.21" x 1.38" (56mm x 35mm)

Max height: .197" (5mm)

PCB size: 55mm x 35mm

PCB Layers: 4

PCB thickness: 1.6mm

RoHS Compliant: Yes

Weight: 10g

Rough model can be found at PocketBeagle models

2.6.10 Additional Pictures

2.6.11 Support Information

All support for this design is through the BeagleBoard.org community at:

- beagleboard@googlegroups.com or
- beagleboard.org/discuss.

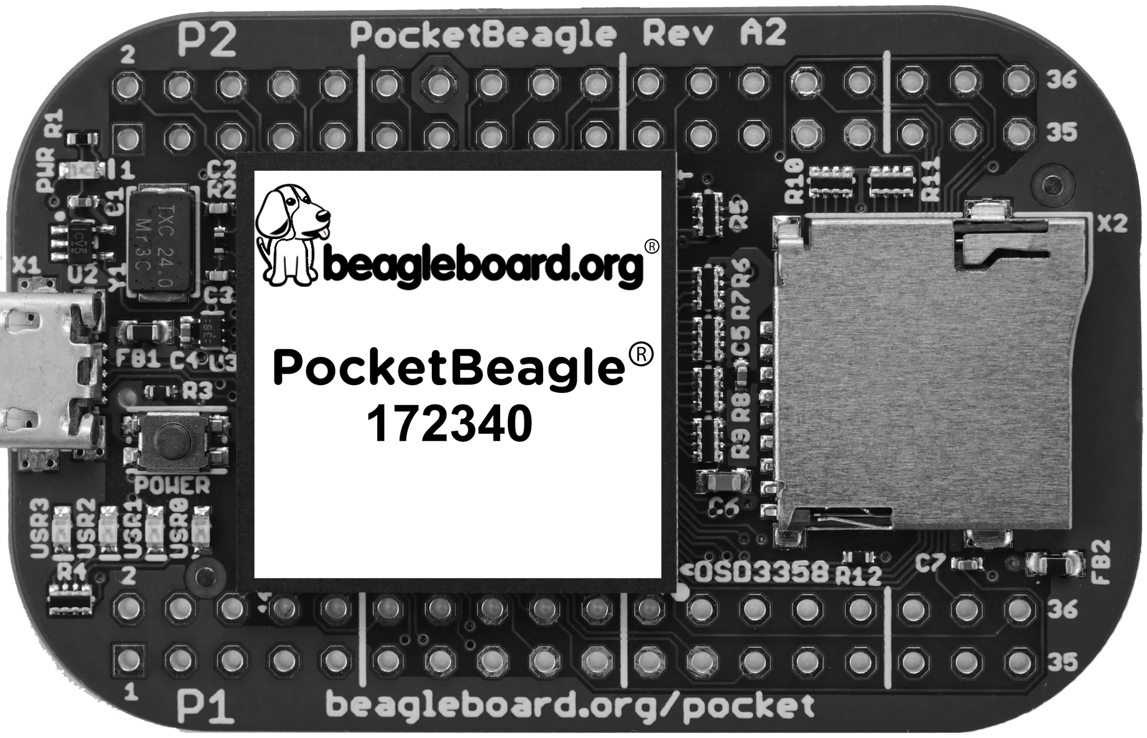


Fig. 2.174: PocketBeagle Front BW

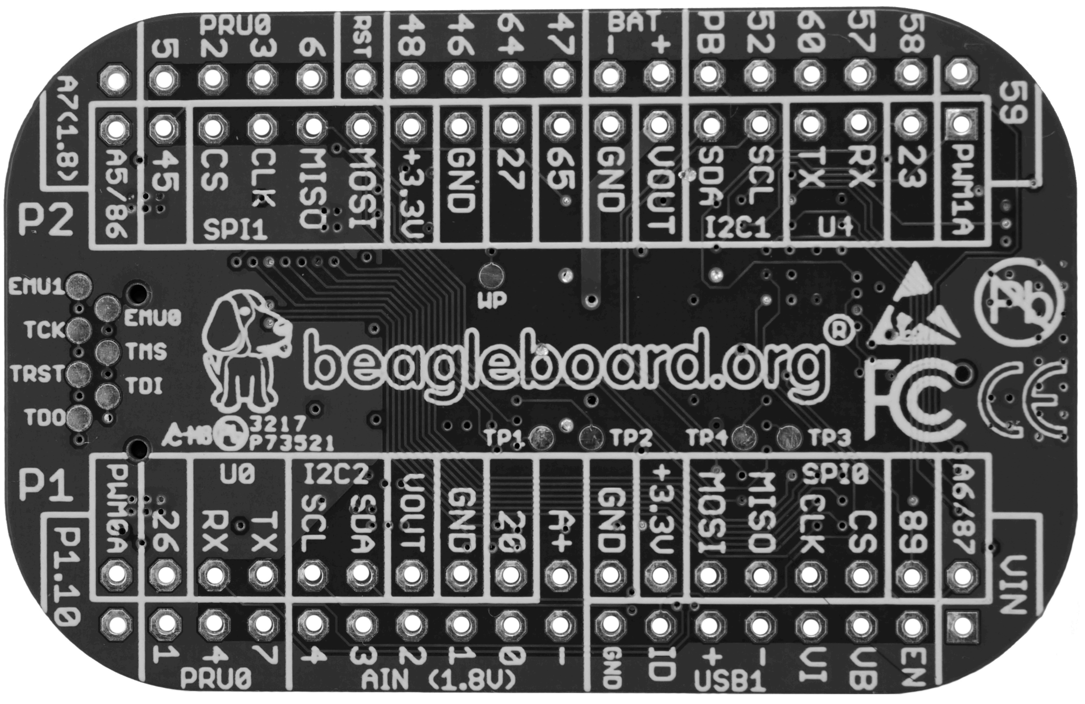


Fig. 2.175: PocketBeagle Back BW

Hardware Design

Design documentation can be found on the wiki. https://git.beagleboard.org/beagleboard/pocketbeagle/ Including:

- Schematic in PDF https://git.beagleboard.org/beagleboard/pocketbeagle/-/blob/master/PocketBeagle_sch.pdf
- Schematic and layout in EAGLE https://git.beagleboard.org/beagleboard/pocketbeagle/-/tree/master/EAGLE
- Schematic and layout in KiCAD https://git.beagleboard.org/beagleboard/pocketbeagle/-/tree/master/KiCAD
- Bill of Materials https://git.beagleboard.org/beagleboard/pocketbeagle/-/blob/master/PocketBeagle_BOM.csv
- *PocketBeagle* docs.

Software Updates

It is a good idea to always use the latest software. Instructions for how to update your software to the latest version can be found at:

Download the latest software files from www.beagleboard.org/distros

Export Information

- ECCN: EAR99
- CCATS: G173833
- Documentation: PocketBeagle_Export_Classification.pdf

RMA Support

If you feel your board is defective or has issues and before returning merchandise, please seek approval from the manufacturer using beagleboard.org/support/rma. You will need the manufacturer, model, revision and serial number of the board.

Getting Help

If you need some up to date troubleshooting techniques, the Wiki is a great place to start PocketBeagle wiki.

If you need professional support, check out beagleboard.org/resources.

2.7 Capes

Note: This page is under development.

Contributors

Note: Make sure to read and accept all the terms & condition provided in the *Terms & Conditions* page.

Use of either the boards or the design materials constitutes agreement to the T&C including any modifications done to the hardware or software solutions provided by beagleboard.org foundation.

Capes are add-on boards for BeagleBone or PocketBeagle families of boards. Using a Cape add-on board, you can easily add sensors, communication peripherals, and more.

Please visit BeagleBoard.org - Cape for the list of currently available Cape add-on boards.

In the BeagleBone board family, there are many variants, such as *BeagleBone Black*, *BeagleBone AI*, *BeagleBone AI-64* and compatibles such as SeeedStudio BeagleBone Green, SeeedStudio BeagleBone Green Wireless, SeeedStudio BeagleBone Green Gateway and more.

The *BeagleBone cape interface spec* enables a common set of device tree overlays and software to be utilized on each of these different BeagleBone boards.

Each hardware has different internal pin assignments and the number of peripherals in the SoC, but the device tree overlay absorbs these differences.

The user of the Cape add-on boards are essentially able to use it across the corresponding Boards without changing any code at all.

Find the instructions below on using each cape:

- *BeagleBoard.org BeagleBone Relay Cape*

2.7.1 BeagleBone cape interface spec

This page is a fork of BeagleBone cape interface spec page on elinux. This is the new official home.

Background and overview

Important: Resources

- See Device Tree: Supporting Similar Boards - The BeagleBone Example blog post on BeagleBoard.org
- See spreadsheet with pin header details
- See elinux.org Cape Expansion Headers for BeagleBone page
- See *BeagleBone Black System Reference Manual Connectors section*
- See *BeagleBone AI System Reference Manual Connectors section*
- See BeagleBone AI-64 System Reference Manual Connectors section

Note: Below, when mentioning "Black", this is true for all AM3358-based BeagleBone boards. "AI" is AM5729-based. "AI-64" is TDA4VM-based.

The device tree symbols for the BeagleBone Cape Compatibility Layer are provided in BeagleBoard-DeviceTrees at:

- Black: bbb-bone-buses.dtsi
- AI: bbai-bone-buses.dtsi
- AI-64: k3-j721e-beagleboneai-64-bone-buses.dtsi

The udev rules used to create the userspace symlinks for the BeagleBone Cape Compatibility Layer are provided in usr-customizations at:

More details can be found in *Methodology*.

Note: Legend

- *D*: Digital general purpose input and output (GPIO)
- *I*: Inter-integrated circuit bus (I^2C) ports
- *S*: Serial peripheral interface (SPI) ports
- *U*: Universal asynchronous reciever/transmitter (UART) serial ports
- *C*: CAN
- *A*: Analog inputs
- E: PWM
- *Q*: Capture/EQEP
- M: MMC/SD/SDIO
- B: I2S/audio serial ports
- L: LCD
- P: PRU
- Y: ECAP

Table 2.66: Overall

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
USB D+	E1	E2	USB D-					
5V OUT	E3	E4	GND					
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		*D* M	3	4	*D* M
5V IN	5	6	5V IN		*D* M *C*	5	6	*D* M *C*
5V OUT	7	8	5V OUT		*D C*	7	8	*D C*
PWR BUT	9	10	RESET		*D C*	9	10	*D C*
D U	11	12	*D*		*D* P	11	12	*D Q* P
D U	13	14	*D* E		*D* E	13	14	*D*
D	15	16	*D* E		*D* P	15	16	*D* P
D I S	17	18	*D I S*		*D*	17	18	*D*
D I C	19	20	*D I C*		*D* E	19	20	*D* M P
D E *S U*	21	22	*D* E *S U*		*D* M P	21	22	*D* M *Q*
D S	23	24	*D I U C*		*D* M	23	24	*D* M
D P	25	26	*D I U C*		*D* M	25	26	*D*
D P *Q*	27	28	*D S* P		*D* L P	27	28	*D* L P *U*
D E *S* P	29	30	*D S* P		*D* L P *U*	29	30	*D* L P
D E *S* P	31	32	ADC VDD REF OUT		*D* L	31	32	*D* L
A	33	34	ADC GND		*D* L *Q*	33	34	*D* E L
A	35	36	*A*		*D* L *Q*	35	36	*D* E L
A	37	38	*A*		*D* L *U*	37	38	*D* L *U*
A	39	40	*A*		*D* L P	39	40	*D* L P
D P	41	42	*D Q S U* P		*D* L P	41	42	*D* L P
GND	43	44	GND		*D* L P	43	44	*D* L P
GND	45	46	GND		*D* E L P	45	46	*D* E L P

Digital GPIO

The compatibility layer comes with simple reference nodes for attaching the Linuuux gpio-leds or gpio-keys to any cape header GPIO pin. This provides simple userspace general purpose input or output with various trigger modes.

The format followed for the gpio-leds nodes is **bone_led_P8_## / bone_led_P9_##**. The **gpio-leds** driver is used by these reference nodes internally and allows users to easily create compatible led nodes in overlays for Black, AI and AI-64.

Listing 2.1: Example device tree overlay to enable LED driver on header P8 pin 3

```
/dts-v1/;
/plugin/;

&bone_led_P8_03 {
    status = ”okay”;
}
```

In *Example device tree overlay to enable LED driver on header P8 pin 3*, it is possible to redefine the default label and other properties defined in the gpio-leds schema.

Table 2.67: GPIO pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		D M	3	4	D M
5V IN	5	6	5V IN		D M C4t	5	6	D M C4r
5V OUT	7	8	5V OUT		D C2r	7	8	D C2t
PWR BUT	9	10	RESET		D C3r	9	10	D C3t
D U4r	11	12	D		D P0o	11	12	D Q2a P0o
D U4t	13	14	D E1a		D E2b	13	14	D
D	15	16	D E1b		D P0i	15	16	D P0i
D I1c S00	17	18	D I1d S0o		D	17	18	D
C0r D I2c	19	20	C0t D I2d		D E2a	19	20	D M P1
D E0b S0i U2t	21	22	D E0a S0c U2r		D M P1	21	22	D M Q2b
D S01	23	24	C1r D I3c U1t		D M	23	24	D M
D P0	25	26	C1t D I3d U1r		D M	25	26	D
D P0 Q0b	27	28	D P0 S10		D L P1	27	28	D L P1 U6r
D E S1i P0	29	30	D P0 S1o		D L P1 U6t	29	30	D L P1
D E S1c P0	31	32	ADC VDD		D L	31	32	D L
A 4	33	34	ADC GND		D L Q1b	33	34	D E L
A 6	35	36	*A* 5		D L Q1a	35	36	D E L
A 2	37	38	*A* 3		D L U5t	37	38	D L U5r
A 0	39	40	*A* 1		D L P1	39	40	D L P1
D P0	41	42	D Q0a S11 U3t P0		D L P1	41	42	D L P1
GND	43	44	GND		D L P1	43	44	D L P1
GND	45	46	GND		D E L P1	45	46	D E L P1

Table 2.68: Bone GPIO LEDs interface

LED SYSFS	Header pin	Black	AI	AI-64
/sys/class/leds/P8_03	P8_03	gpio1_6	gpio1_24	gpio0_20
/sys/class/leds/P8_04	P8_04	gpio1_7	gpio1_25	gpio0_48
/sys/class/leds/P8_05	P8_05	gpio1_2	gpio7_1	gpio0_33
/sys/class/leds/P8_06	P8_06	gpio1_3	gpio7_2	gpio0_34
/sys/class/leds/P8_07	P8_07	gpio2_2	gpio6_5	gpio0_15
/sys/class/leds/P8_08	P8_08	gpio2_3	gpio6_6	gpio0_14
/sys/class/leds/P8_09	P8_09	gpio2_5	gpio6_18	gpio0_17
/sys/class/leds/P8_10	P8_10	gpio2_4	gpio6_4	gpio0_16
/sys/class/leds/P8_11	P8_11	gpio1_13	gpio3_11	gpio0_60
/sys/class/leds/P8_12	P8_12	gpio1_12	gpio3_10	gpio0_59

continues on next page

Table 2.68 – continued from previous page

LED SYSFS	Header pin	Black	AI	AI-64
/sys/class/leds/P8_13	P8_13	gpio0_23	gpio4_11	gpio0_89
/sys/class/leds/P8_14	P8_14	gpio0_26	gpio4_13	gpio0_75
/sys/class/leds/P8_15	P8_15	gpio1_15	gpio4_3	gpio0_61
/sys/class/leds/P8_16	P8_16	gpio1_14	gpio4_29	gpio0_62
/sys/class/leds/P8_17	P8_17	gpio0_27	gpio8_18	gpio0_3
/sys/class/leds/P8_18	P8_18	gpio2_1	gpio4_9	gpio0_4
/sys/class/leds/P8_19	P8_19	gpio0_22	gpio4_10	gpio0_88
/sys/class/leds/P8_20	P8_20	gpio1_31	gpio6_30	gpio0_76
/sys/class/leds/P8_21	P8_21	gpio1_30	gpio6_29	gpio0_30
/sys/class/leds/P8_22	P8_22	gpio1_5	gpio1_23	gpio0_5
/sys/class/leds/P8_23	P8_23	gpio1_4	gpio1_22	gpio0_31
/sys/class/leds/P8_24	P8_24	gpio1_1	gpio7_0	gpio0_6
/sys/class/leds/P8_25	P8_25	gpio1_0	gpio6_31	gpio0_35
/sys/class/leds/P8_26	P8_26	gpio1_29	gpio4_28	gpio0_51
/sys/class/leds/P8_27	P8_27	gpio2_22	gpio4_23	gpio0_71
/sys/class/leds/P8_28	P8_28	gpio2_24	gpio4_19	gpio0_72
/sys/class/leds/P8_29	P8_29	gpio2_23	gpio4_22	gpio0_73
/sys/class/leds/P8_30	P8_30	gpio2_25	gpio4_20	gpio0_74
/sys/class/leds/P8_31	P8_31	gpio0_10	gpio8_14	gpio0_32
/sys/class/leds/P8_32	P8_32	gpio0_11	gpio8_15	gpio0_26
/sys/class/leds/P8_33	P8_33	gpio0_9	gpio8_13	gpio0_25
/sys/class/leds/P8_34	P8_34	gpio2_17	gpio8_11	gpio0_7
/sys/class/leds/P8_35	P8_35	gpio0_8	gpio8_12	gpio0_24
/sys/class/leds/P8_36	P8_36	gpio2_16	gpio8_10	gpio0_8
/sys/class/leds/P8_37	P8_37	gpio2_14	gpio8_8	gpio0_106
/sys/class/leds/P8_38	P8_38	gpio2_15	gpio8_9	gpio0_105
/sys/class/leds/P8_39	P8_39	gpio2_12	gpio8_6	gpio0_69
/sys/class/leds/P8_40	P8_40	gpio2_13	gpio8_7	gpio0_70
/sys/class/leds/P8_41	P8_41	gpio2_10	gpio8_4	gpio0_67
/sys/class/leds/P8_42	P8_42	gpio2_11	gpio8_5	gpio0_68
/sys/class/leds/P8_43	P8_43	gpio2_8	gpio8_2	gpio0_65
/sys/class/leds/P8_44	P8_44	gpio2_9	gpio8_3	gpio0_66
/sys/class/leds/P8_45	P8_45	gpio2_6	gpio8_0	gpio0_79
/sys/class/leds/P8_46	P8_46	gpio2_7	gpio8_1	gpio0_80
/sys/class/leds/P9_11	P9_11	gpio0_30	gpio8_17	gpio0_1
/sys/class/leds/P9_12	P9_12	gpio1_28	gpio5_0	gpio0_45
/sys/class/leds/P9_13	P9_13	gpio0_31	gpio6_12	gpio0_2
/sys/class/leds/P9_14	P9_14	gpio1_18	gpio4_25	gpio0_93
/sys/class/leds/P9_15	P9_15	gpio1_16	gpio3_12	gpio0_47
/sys/class/leds/P9_16	P9_16	gpio1_19	gpio4_26	gpio0_94
/sys/class/leds/P9_17	P9_17	gpio0_5	gpio7_17	gpio0_28
/sys/class/leds/P9_18	P9_18	gpio0_4	gpio7_16	gpio0_40
/sys/class/leds/P9_19	P9_19	gpio0_13	gpio7_3	gpio0_78
/sys/class/leds/P9_20	P9_20	gpio0_12	gpio7_4	gpio0_77
/sys/class/leds/P9_21	P9_21	gpio0_3	gpio3_3	gpio0_39
/sys/class/leds/P9_22	P9_22	gpio0_2	gpio6_19	gpio0_38
/sys/class/leds/P9_23	P9_23	gpio1_17	gpio7_11	gpio0_10
/sys/class/leds/P9_24	P9_24	gpio0_15	gpio6_15	gpio0_13
/sys/class/leds/P9_25	P9_25	gpio3_21	gpio6_17	gpio0_127
/sys/class/leds/P9_26	P9_26	gpio0_14	gpio6_14	gpio0_12
/sys/class/leds/P9_27	P9_27	gpio3_19	gpio4_15	gpio0_46
/sys/class/leds/P9_28	P9_28	gpio3_17	gpio4_17	gpio1_11
/sys/class/leds/P9_29	P9_29	gpio3_15	gpio5_11	gpio0_53

continues on next page

Table 2.68 – continued from previous page

LED SYSFS	Header pin	Black	AI	AI-64
/sys/class/leds/P9_30	P9_30	gpio3_16	gpio5_12	gpio0_44
/sys/class/leds/P9_31	P9_31	gpio3_14	gpio5_10	gpio0_52
/sys/class/leds/P9_33	P9_33	*n/a*	*n/a*	gpio0_50
/sys/class/leds/P9_35	P9_35	*n/a*	*n/a*	gpio0_55
/sys/class/leds/P9_36	P9_36	*n/a*	*n/a*	gpio0_56
/sys/class/leds/P9_37	P9_37	*n/a*	*n/a*	gpio0_57
/sys/class/leds/P9_38	P9_38	*n/a*	*n/a*	gpio0_58
/sys/class/leds/P9_39	P9_39	*n/a*	*n/a*	gpio0_54
/sys/class/leds/P9_40	P9_40	*n/a*	*n/a*	gpio0_81
/sys/class/leds/P9_41	P9_41	gpio0_20	gpio6_20	gpio1_0
/sys/class/leds/P9_42	P9_42	gpio0_7	gpio4_18	gpio0_123
/sys/class/leds/A15	A15	gpio0_19	NA	NA

I^{2}C

Compatibility layer provides simple I2C bone bus nodes for creating compatible overlays for Black, AI and AI-64. The format followed for these nodes is **bone_i2c_#**.

Table 2.69: I2C pins

P9			
Functions	odd	even	Functions
1 SCL	17	18	1 SDA
2 SCL	19	20	2 SDA
4 SCL[4][5]	21	22	4 SDA[?][?]
	23	24	3 SCL[3]
	25	26	3 SDA[?]

Table 2.70: I2C port mapping

SYSFS	DT symbol	Black	AI	AI-64	SCL	SDA	Overlay
/dev/bone/i2c/0	bone_i2c_0	I2C0	I2C1	TBD	On-board		
/dev/bone/i2c/1	bone_i2c_1	I2C1	I2C5	MAIN_I2C6	P9.17	P9.18	BONE-I2C1
/dev/bone/i2c/2	bone_i2c_2	I2C2	I2C4	MAIN_I2C3	P9.19	P9.20	BONE-I2C2
/dev/bone/i2c/3	bone_i2c_3	I2C1	I2C3	MAIN_I2C4	P9.24	P9.26	BONE-I2C3
/dev/bone/i2c/4	bone_i2c_4	I2C2	*n/a*	MAIN_I2C3	P9.21	P9.22	BONE-I2C4

Important: In the case the same controller is used for 2 different bone bus nodes, usage of those nodes is mutually-exclusive.

Note: The provided pre-compiled overlays enable the I^{2}C bus driver only, not a specific device driver. Either a custom overlay is required to load the device driver or usermode device driver loading can be performed, depending on the driver. See *Using I2C with Linux drivers* for information on loading I^{2}C drivers from userspace.

Listing 2.2: Example device tree overlay to enable I2C driver

```
/dts-v1/;
/plugin/;

&bone_i2c_1 {
    status = "okay";
```

(continues on next page)

[4] Mutually exclusive with port 2 on Black
[5] On Black and AI-64 only
[3] Mutually exclusive with port 1 on Black

(continued from previous page)

```
    accel@1c {
        compatible = "fsl,mma8453";
        reg = <0x1c>;
    };
}
```

In *Example device tree overlay to enable I2C driver*, you can specify what driver you want to load and provide any properties it might need.

- https://www.kernel.org/doc/html/v5.10/i2c/summary.html
- https://www.kernel.org/doc/html/v5.10/i2c/instantiating-devices.html#method-1-declare-the-i2c-devices-statically
- https://www.kernel.org/doc/Documentation/devicetree/bindings/i2c/

SPI

SPI bone bus nodes allow creating compatible overlays for Black, AI and AI-64.

Table 2.71: SPI pins

P9			
Functions	odd	even	Functions
0 CS0	17	18	0 SDO
	19	20	
0 SDI	21	22	0 CLK
0 CS1	23	24	
	25	26	
	27	28	1 CS0
1 SDI	29	30	1 SDO
1 CLK	31	32	
	33	34	
	35	36	
	37	38	
	39	40	
	41	42	1 CS1[2]

Table 2.72: SPI port mapping

Bone bus	DT symbol	Black	AI	AI-64	SDO	SDI	CLK	CS	Overlay
/dev/bone/spi/0.0	bone_spi_0	SPI0	SPI2	MAIN_SPI6	P9.18	P9.21	P9.22	P9.17 (CS0)	BONE-SPI0_0
/dev/bone/spi/0.1								P9.23 (CS1)[?]	BONE-SPI0_1
/dev/bone/spi/1.0	bone_spi_1	SPI1	SPI3	MAIN_SPI7	P9.30	P9.29	P9.31	P9.28 (CS0)	BONE-SPI1_0
/dev/bone/spi/1.1								P9.42 (CS1)	BONE-SPI1_1

Note: The provided pre-compiled overlays enable the "spidev" driver using the "rohm,dh2228fv" compatible string. See https://stackoverflow.com/questions/53634892/linux-spidev-why-it-shouldnt-be-directly-in-devicetree for more background. A custom overlay is required to overload the compatible string to load a non-spidev driver.

Note: #TODO# figure out if BONE-SPI0_0 and BONE-SPI0_1 can be loaded at the same time

Listing 2.3: Example device tree overlay to enable SPI driver

```
/dts-v1/;
/plugin/;
```

(continues on next page)

[2] Only available on AI and AI-64

(continued from previous page)

```

&bone_spi_0 {
    status = "okay";
     pressure@0 {
         compatible = "bosch,bmp280";
         reg = <0>;      /* CS0 */
         spi-max-frequency = <5000000>;
     };
}
```

In *Example device tree overlay to enable SPI driver*, you can specify what driver you want to load and provide any properties it might need.

- https://www.kernel.org/doc/html/v5.10/spi/spi-summary.html
- https://www.kernel.org/doc/Documentation/devicetree/bindings/spi/

UART

UART bone bus nodes allow creating compatible overlays for Black, AI and AI-64.

Table 2.73: UART pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
4 RX[6]	11	12				11	12	
4 TX[?]	13	14				13	14	
	15	16				15	16	
	17	18				17	18	
	19	20				19	20	
2 TX	21	22	2 RX			21	22	
	23	24	1 TX			23	24	
	25	26	1 RX			25	26	
	27	28				27	28	6 RX
	29	30			6 TX	29	30	
	31	32				31	32	
	33	34				33	34	7 TX
	35	36				35	36	
	37	38			5 TX	37	38	5 RX
	39	40				39	40	
	41	42	3 TX			41	42	

Important: RTSn and CTSn mappings are not compatible across boards in the family and are therefore not part of the cape specification.

Table 2.74: UART port mapping

Bone bus	DT symbol	Black	AI	AI-64	TX	RX	Overlay
/dev/bone/uart/0	bone_uart_0	UART0	UART1	MAIN_UART0	Console debug header pins		
/dev/bone/uart/1	bone_uart_1	UART1	UART10	MAIN_UART2	P9.24	P9.26	BONE-UART1
/dev/bone/uart/2	bone_uart_2	UART2	UART3	*n/a*	P9.21	P9.22	BONE-UART2
/dev/bone/uart/3	bone_uart_3	UART3	*n/a*	*n/a*	P9.42	*n/a*	BONE-UART3
/dev/bone/uart/4	bone_uart_4	UART4	UART5	MAIN_UART0[?]	P9.13	P9.11	BONE-UART4
/dev/bone/uart/5	bone_uart_5	UART5	UART8	MAIN_UART5	P8.37	P8.38	BONE-UART5
/dev/bone/uart/6	bone_uart_6	*n/a*	*n/a*	MAIN_UART8	P8.29	P8.28	BONE-UART6
/dev/bone/uart/7	bone_uart_7	*n/a*	*n/a*	MAIN_UART2	P8.34	P8.22	BONE-UART7

Important: In the case the same controller is used for 2 different bone bus nodes, usage of those nodes is

[6] This port is shared with the console UART on AI-64

mutually-exclusive.

CAN

CAN bone bus nodes allow creating compatible overlays for Black, AI and AI-64.

Table 2.75: CAN pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
	5	6			4 TX	5	6	4 RX
	7	8			2 RX	7	8	2 TX
	9	10			3 RX	9	10	3 TX
	11	12				11	12	
	13	14				13	14	
	15	16				15	16	
	17	18				17	18	
0 RX	19	20	0 TX			19	20	
	21	22				21	22	
	23	24	1 RX			23	24	
	25	26	1 TX			25	26	

Table 2.76: CAN port mapping

Bone bus	Black	AI	AI-64	TX	RX	Overlays
/dev/bone/can/0	CAN0	*n/a*	MAIN_MCAN0	P9.20	P9.19	BONE-CAN0
/dev/bone/can/1	CAN1	CAN2	MAIN_MCAN4	P9.26	P9.24	BONE-CAN1
/dev/bone/can/2	*n/a*	CAN1[1]	MAIN_MCAN5	P8.08	P8.07	BONE-CAN2
/dev/bone/can/3	*n/a*	*n/a*	MAIN_MCAN6	P8.10	P8.09	BONE-CAN3
/dev/bone/can/4	*n/a*	*n/a*	MAIN_MCAN7	P8.05	P8.06	BONE-CAN4

ADC

- TODO: We need a udev rule to make sure the ADC shows up at /dev/bone/adc! There's nothing for sure that IIO devices will show up in the same place.
- TODO: I think we can also create symlinks for each channel based on which device is there, such that we can do /dev/bone/adc/Px_y

[1] BeagleBone AI rev A2 and later only

Table 2.77: ADC pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
USB D+	E1	E2	USB D-					
5V OUT	E3	E4	GND					
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		D M	3	4	D M
5V IN	5	6	5V IN		D M C4t	5	6	D M C4r
5V OUT	7	8	5V OUT		D C2r	7	8	D C2t
PWR BUT	9	10	RESET		D C3r	9	10	D C3t
D U4r	11	12	D		D P0o	11	12	D Q2a P0o
D U4t	13	14	D E1a		D E2b	13	14	D
D	15	16	D E1b		D P0i	15	16	D P0i
D I1c S00	17	18	D I1d S0o		D	17	18	D
C0r D I2c	19	20	C0t D I2d		D E2a	19	20	D M P1
D E0b S0i U2t	21	22	D E0a S0c U2r		D M P1	21	22	D M Q2b
D S01	23	24	C1r D I3c U1t		D M	23	24	D M
D P0	25	26	C1t D I3d U1r		D M	25	26	D
D P0 Q0b	27	28	D P0 S10		D L P1	27	28	D L P1 U6r
D E S1i P0	29	30	D P0 S1o		D L P1 U6t	29	30	D L P1
D E S1c P0	31	32	ADC VDD		D L	31	32	D L
A 4	33	34	ADC GND		D L Q1b	33	34	D E L
A 6	35	36	*A* 5		D L Q1a	35	36	D E L
A 2	37	38	*A* 3		D L U5t	37	38	D L U5r
A 0	39	40	*A* 1		D L P1	39	40	D L P1
D P0	41	42	D Q0a S11 U3t P0		D L P1	41	42	D L P1
GND	43	44	GND		D L P1	43	44	D L P1
GND	45	46	GND		D E L P1	45	46	D E L P1

Table 2.78: Bone ADC

Index	Header pin	Black/AI-64	AI
0	P9_39	in_voltage0_raw	in_voltage0_raw
1	P9_40	in_voltage1_raw	in_voltage1_raw
2	P9_37	in_voltage2_raw	in_voltage3_raw
3	P9_38	in_voltage3_raw	in_voltage2_raw
4	P9_33	in_voltage4_raw	in_voltage7_raw
5	P9_36	in_voltage5_raw	in_voltage6_raw
6	P9_35	in_voltage6_raw	in_voltage4_raw

Table 2.79: Bone ADC Overlay

Black	AI	AI-64	overlay
Internal	External (STMPE811)	TBD	BONE-ADC.dts

PWM PWM bone bus nodes allow creating compatible overlays for Black, AI and AI-64. For the definitions, you can see bbai-bone-buses.dtsi#L415 & bbb-bone-buses.dtsi#L432

Table 2.80: PWM pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
USB D+	E1	E2	USB D-					
5V OUT	E3	E4	GND					
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		D M	3	4	D M
5V IN	5	6	5V IN		D M C4t	5	6	D M C4r
5V OUT	7	8	5V OUT		D C2r	7	8	D C2t
PWR BUT	9	10	RESET		D C3r	9	10	D C3t
D U4r	11	12	D		D P0o	11	12	D Q2a P0o
D U4t	13	14	D E1a		D E2b	13	14	D
D	15	16	D E1b		D P0i	15	16	D P0i
D I1c S00	17	18	D I1d S0o		D	17	18	D
C0r D I2c	19	20	C0t D I2d		D E2a	19	20	D M P1
D E0b S0i U2t	21	22	D E0a S0c U2r		D M P1	21	22	D M Q2b
D S01	23	24	C1r D I3c U1t		D M	23	24	D M
D P0	25	26	C1t D I3d U1r		D M	25	26	D
D P0 Q0b	27	28	D P0 S10		D L P1	27	28	D L P1 U6r
D E S1i P0	29	30	D P0 S1o		D L P1 U6t	29	30	D L P1
D E S1c P0	31	32	ADC VDD		D L	31	32	D L
A 4	33	34	ADC GND		D L Q1b	33	34	D E L
A 6	35	36	*A* 5		D L Q1a	35	36	D E L
A 2	37	38	*A* 3		D L U5t	37	38	D L U5r
A 0	39	40	*A* 1		D L P1	39	40	D L P1
D P0	41	42	D Q0a S11 U3t P0		D L P1	41	42	D L P1
GND	43	44	GND		D L P1	43	44	D L P1
GND	45	46	GND		D E L P1	45	46	D E L P1

Table 2.81: Bone bus PWM

Bone bus	Black	AI	AI-64	A	B	Overlay
/dev/bone/pwm/0	PWM0	•	PWM1	P9.22	P9.21	BONE-PWM0.dts
/dev/bone/pwm/1	PWM1	PWM3	PWM2	P9.14	P9.16	BONE-PWM1.dts
/dev/bone/pwm/2	PWM2	PWM2	PWM0	P8.19	P8.13	BONE-PWM2.dts

TIMER PWM TIMER PWM bone bus uses ti,omap-dmtimer-pwm driver, and timer nodes that allow creating compatible overlays for Black, AI and AI-64. For the timer node definitions, you can see bbai-bone-buses.dtsi#L449 & bbb-bone-buses.dtsi#L466.

Table 2.82: Bone TIMER PWMs

Bone bus	Header pin	Black	AI	overlay
/sys/bus/platform/devices/bone_timer_pwm_0/	P8.10	timer6	timer10	BONE-TIMER_PWM_0.dts
/sys/bus/platform/devices/bone_timer_pwm_1/	P8.07	timer4	timer11	BONE-TIMER_PWM_1.dts
/sys/bus/platform/devices/bone_timer_pwm_2/	P8.08	timer7	timer12	BONE-TIMER_PWM_2.dts
/sys/bus/platform/devices/bone_timer_pwm_3/	P9.21	•	timer13	BONE-TIMER_PWM_3.dts
/sys/bus/platform/devices/bone_timer_pwm_4/	P8.09	timer5	timer14	BONE-TIMER_PWM_4.dts
/sys/bus/platform/devices/bone_timer_pwm_5/	P9.22	•	timer15	BONE-TIMER_PWM_5.dts

eQEP

Table 2.83: eQEP pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
USB D+	E1	E2	USB D-					
5V OUT	E3	E4	GND					
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		D M	3	4	D M
5V IN	5	6	5V IN		D M C4t	5	6	D M C4r
5V OUT	7	8	5V OUT		D C2r	7	8	D C2t
PWR BUT	9	10	RESET		D C3r	9	10	D C3t
D U4r	11	12	D		D P0o	11	12	D Q2a P0o
D U4t	13	14	D E1a		D E2b	13	14	D
D	15	16	D E1b		D P0i	15	16	D P0i
D I1c S00	17	18	D I1d S0o		D	17	18	D
C0r D I2c	19	20	C0t D I2d		D E2a	19	20	D M P1
D E0b S0i U2t	21	22	D E0a S0c U2r		D M P1	21	22	D M Q2b
D S01	23	24	C1r D I3c U1t		D M	23	24	D M
D P0	25	26	C1t D I3d U1r		D M	25	26	D
D P0 Q0b	27	28	D P0 S10		D L P1	27	28	D L P1 U6r
D E S1i P0	29	30	D P0 S1o		D L P1 U6t	29	30	D L P1
D E S1c P0	31	32	ADC VDD		D L	31	32	D L
A 4	33	34	ADC GND		D L Q1b	33	34	D E L
A 6	35	36	*A* 5		D L Q1a	35	36	D E L
A 2	37	38	*A* 3		D L U5t	37	38	D L U5r
A 0	39	40	*A* 1		D L P1	39	40	D L P1
D P0	41	42	D Q0a S11 U3t P0		D L P1	41	42	D L P1
GND	43	44	GND		D L P1	43	44	D L P1
GND	45	46	GND		D E L P1	45	46	D E L P1

On BeagleBone's without an eQEP on specific pins, consider using the PRU to perform a software counter function.

Table 2.84: Bone eQEP

Bone bus	Black	AI	AI-64	A	B	strobe	index	overlay
/dev/bone/counter/0	eQEP0	eQEP2	eQEP0	P9.42	P9.27	• Black/AI-64: P9.25 • AI: P8.06	• Black/AI-64: P9.41 • AI: P8.05	
/dev/bone/counter/1	eQEP1	eQEP0	eQEP1	P8.35	P8.33	• Black/AI-64: P8.32 • AI: P9.21	• Black/AI-64: P8.31 • AI: –	
/dev/bone/counter/2	eQEP2	eQEP1	–	P8.12	P8.22	• Black: P8.15 • AI: P8.18	• Black: P8.16 • AI: P9.15	

eCAP #TODO: This doesn't include any abstraction yet.

Table 2.85: ECAP pins

P9					P8			
Functions	odd	even	Functions		Functions	odd	even	Functions
USB D+	E1	E2	USB D-					
5V OUT	E3	E4	GND					
GND	1	2	GND		GND	1	2	GND
3V3 OUT	3	4	3V3 OUT		D M	3	4	D M
5V IN	5	6	5V IN		D M C4t	5	6	D M C4r
5V OUT	7	8	5V OUT		D C2r	7	8	D C2t
PWR BUT	9	10	RESET		D C3r	9	10	D C3t
D U4r	11	12	D		D P0o	11	12	D Q2a P0o
D U4t	13	14	D E1a		D E2b	13	14	D
D	15	16	D E1b		D P0i	15	16	D P0i
D I1c S00	17	18	D I1d S0o		D	17	18	D
C0r D I2c	19	20	C0t D I2d		D E2a	19	20	D M P1
D E0b S0i U2t	21	22	D E0a S0c U2r		D M P1	21	22	D M Q2b
D S01	23	24	C1r D I3c U1t		D M	23	24	D M
D P0	25	26	C1t D I3d U1r		D M	25	26	D
D P0 Q0b	27	28	D P0 S10		D L P1	27	28	D L P1 U6r
D E S1i P0	29	30	D P0 S1o		D L P1 U6t	29	30	D L P1
D E S1c P0	31	32	ADC VDD		D L	31	32	D L
A 4	33	34	ADC GND		D L Q1b	33	34	D E L
A 6	35	36	*A* 5		D L Q1a	35	36	D E L
A 2	37	38	*A* 3		D L U5t	37	38	D L U5r
A 0	39	40	*A* 1		D L P1	39	40	D L P1
D P0	41	42	D Q0a S11 U3t P0		D L P1	41	42	D L P1
GND	43	44	GND		D L P1	43	44	D L P1
GND	45	46	GND		D E L P1	45	46	D E L P1

Table 2.86: Black eCAP PWMs

Bone bus	Header pin	peripheral	overlay
/sys/bus/platform/drivers/ecap/48302100.ecap	P9.42	eCAP0_in_PWM0_out	BBB-ECAP0.dts
/sys/bus/platform/drivers/ecap/48304100.ecap	P9.28	eCAP2_in_PWM2_out	BBB-ECAP2.dts

Table 2.87: AI eCAP PWMs

Bone bus	Header pin	peripheral	overlay
/sys/bus/platform/drivers/ecap/4843e100.ecap	P8.15	eCAP1_in_PWM1_out	BBAI-ECAP1.dts
/sys/bus/platform/drivers/ecap/48440100.ecap	P8.14	eCAP2_in_PWM2_out	BBAI-ECAP2.dts
/sys/bus/platform/drivers/ecap/48440100.ecap	P8.20	eCAP2_in_PWM2_out	BBAI-ECAP2A.dts
/sys/bus/platform/drivers/ecap/48442100.ecap	P8.04	eCAP3_in_PWM3_out	BBAI-ECAP3.dts
/sys/bus/platform/drivers/ecap/48442100.ecap	P8.26	eCAP3_in_PWM3_out	BBAI-ECAP3A.dts

eMMC

Table 2.88: Bone eMMC

Header pin	Description
P8.3	DAT6
P8.4	DAT7
P8.5	DAT2
P8.6	DAT3
P8.20	CMD
P8.21	CLK
P8.22	DAT5
P8.23	DAT4
P8.24	DAT1
P8.25	DAT0

Table 2.89: Bone eMMC Overlay

Black	AI	overlay
MMC2	MMC3	BONE-eMMC.dts

LCD

Table 2.90: 16bit LCD interface

Header pin	Description
P8_45	lcd_data0
P8_46	lcd_data1
P8_43	lcd_data2
P8_44	lcd_data3
P8_41	lcd_data4
P8_42	lcd_data5
P8_39	lcd_data6
P8_40	lcd_data7
P8_37	lcd_data8
P8_38	lcd_data9
P8_36	lcd_data10
P8_34	lcd_data11
P8_35	lcd_data12
P8_33	lcd_data13
P8_31	lcd_data14
P8_32	lcd_data15
P8_27	lcd_vsync
P8_29	lcd_hsync
P8_28	lcd_pclk
P8_30	lcd_ac_bias_en

Table 2.91: 16bit LCD interface Overlay

Black	AI	overlay
lcdc	dss	

McASP

Table 2.92: Bone McASP0

Header pin	Description
P9.12	aclkr
P9.25	ahclkx
P9.27	fsr
P9.28	Black: axr2 AI: axr9
P9.29	fsx
P9.30	Black: axr0 AI: axr10
P9.31	aclkx

Table 2.93: Bone McASP0 Overlay

Black	AI	overlay
McASP0	McASP1	

PRU The overlay situation for PRUs is a bit more complex than with other peripherals. The mechanism for loading, starting and stopping the PRUs can go through either [https://www.kernel.org/doc/html/latest/driver-api/uio-howto.html UIO] or [https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/PRU-ICSS/Linux_Drivers/RemoteProc_and_RPMsg.html RemoteProc].

- /dev/remoteproc/prussX-coreY (AM3358 X = "", other x = "1|2")

Table 2.94: Bone PRU eCAP

Header Pin	Black	AI
P8.15	pr1_ecap0	pr1_ecap0
P8.32	•	pr2_ecap0
P9.42	pr1_ecap0	•

Table 2.95: AI PRU UART

UART	TX	RX	RTSn	CTSn	Overlays
PRU1 UART0	P8_31	P8_33	P8_34	P8_35	
PRU2 UART0	P8_43	P8_44	P8_45	P8_46	

Table 2.96: Bone PRU

Header Pin	Black	AI
P8.03	•	pr2_pru0 10
P8.04	•	pr2_pru0 11
P8.05	•	pr2_pru0 06
P8.06	•	pr2_pru0 07
P8.07	•	pr2_pru1 16
P8.08	•	pr2_pru0 20
P8.09	•	pr2_pru1 06
P8.10	•	pr2_pru1 15
P8.11	pr1_pru0 15 (Out)	pr1_pru0 04
P8.12	pr1_pru0 14 (Out)	pr1_pru0 03
P8.13	•	pr1_pru1 07
P8.14	•	pr1_pru1 09
P8.15	pr1_pru0 15 (In)	pr1_pru1 16
P8.16	pr1_pru0 14 (In)	pr1_pru1 18
P8.17	•	pr2_pru0 15

continues on next page

Table 2.96 – continued from previous page

Header Pin	Black	AI
P8.18	•	pr1_pru1 05
P8.19	•	pr1_pru1 06
P8.20	•	pr2_pru0 03
P8.21	•	pr2_pru0 02
P8.22	•	pr2_pru0 09
P8.23	•	pr2_pru0 08
P8.24	•	pr2_pru0 05
P8.25	•	pr2_pru0 04
P8.26	•	pr1_pru1 17
P8.27	•	pr2_pru1 17
P8.28	•	pr2_pru0 17
P8.29	•	pr2_pru0 18
P8.30	•	pr2_pru0 19
P8.31	•	pr2_pru0 11
P8.32	•	pr2_pru1 00
P8.33	•	pr2_pru0 10
P8.34	•	pr2_pru0 08
P8.35	•	pr2_pru0 09

continues on next page

Table 2.96 – continued from previous page

Header Pin	Black	AI
P8.36	•	pr2_pru0 07
P8.37	•	pr2_pru0 05
P8.38	•	pr2_pru0 06
P8.39	•	pr2_pru0 03
P8.40	•	pr2_pru0 04
P8.41	•	pr2_pru0 01
P8.42	•	pr2_pru0 02
P8.43	•	pr2_pru1 20
P8.44	•	pr2_pru0 00
P8.45	•	pr2_pru1 18
P8.46	•	pr2_pru1 19
P9.11	•	pr2_pru0 14
P9.13	•	pr2_pru0 15
P9.14	•	pr1_pru1 14
P9.15	•	pr1_pru0 5
P9.16	•	pr1_pru1 15
P9.17	•	pr2_pru1 09
P9.18	•	pr2_pru1 08

continues on next page

Table 2.96 – continued from previous page

Header Pin	Black	AI
P9.19	•	pr1_pru1 02
P9.20	•	pr1_pru1 01
P9.24	pr1_pru0 16 (In)	•
P9.25	pr1_pru0 07	pr2_pru1 05
P9.26	pr1_pru1 16 (In)	pr1_pru0 17
P9.27	pr1_pru0 05	pr1_pru1 11
P9.28	pr1_pru0 03	pr2_pru1 13
P9.29	pr1_pru0 01	pr2_pru1 11
P9.30	pr1_pru0 02	pr2_pru1 12
P9.31	pr1_pru0 00	pr2_pru1 10
P9.41	pr1_pru0 06	pr1_pru1 03
P9.42	pr1_pru0 04	pr1_pru1 10

GPIO TODO<br> For each of the pins with a GPIO, there should be a symlink that comes from the names *

Methodology

The methodology for applied in the kernel and software images to expose the software interfaces is to be documented here. The most fundamental elements are the device tree entries, including overlays, and udev rules.

10-of-symlink.rules

```
#From: https://github.com/mvduin/py-uio/blob/master/etc/udev/rules.d/10-of-
↪symlink.rules
# allow declaring a symlink for a device in DT
ATTR{device/of_node/symlink}!="", \
        ENV{OF_SYMLINK}="%s{device/of_node/symlink}"

ENV{OF_SYMLINK}!="", ENV{DEVNAME}!="", \
        SYMLINK+="%E{OF_SYMLINK}", \
        TAG+="systemd", ENV{SYSTEMD_ALIAS}+="/dev/%E{OF_SYMLINK}"
```

TBD

```
# Also courtesy of mvduin
# create symlinks for gpios exported to sysfs by DT
SUBSYSTEM=="gpio", ACTION=="add", TEST=="value", ATTR{label}!="sysfs", \
                RUN+="/bin/mkdir -p /dev/bone/gpio", \
                RUN+="/bin/ln -sT '/sys/class/gpio/%k' /dev/bone/gpio/%s
↪{label}"
```

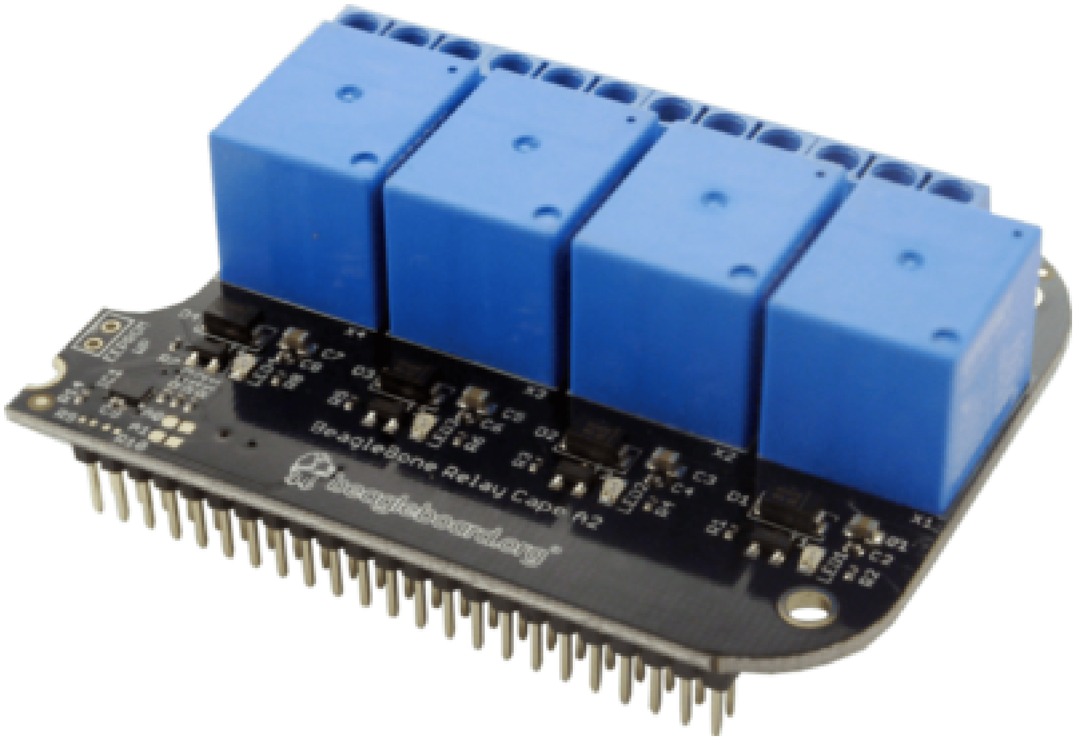
Verification TODO: The steps used to verify all of these configurations is to be documented here. It will serve to document what has been tested, how to reproduce the configurations, and how to verify each major triannual release. All faults will be documented in the issue tracker.

References

- Device Tree: Supporting Similar Boards - The BeagleBone Example
- Google drive with summary of expansion signals on various BeagleBoard.org designs
- Beagleboard:Cape Expansion Headers

2.7.2 BeagleBoard.org BeagleBone Relay Cape

Relay Cape, as the name suggests, is a simple Cape with relays on it. It contains four relays, each of which can be operated independently from the BeagleBone.



- Order page
- Schematic

Note: The following describes how to use the device tree overlay under development. The description may not be suitable for those using older firmware.

Installation

No special configuration is required. When you plug Cape into your BeagleBoard, it is automatically recognized by the Cape Universal function.

You can check to see if the Relay Cape is recognized with the following command.

```
ls /proc/device-tree/chosen/overlay
```

A list of currently loaded device tree overlays is displayed here. If you see *BBORG_RELAY-00A2.kernel* in this list, it has been loaded correctly.

If it is not loaded correctly, you can also load it directly by adding the following to the U-Boot options (which can be reflected by changing /boot/uEnv.txt).

```
uboot_overlay_addr0=BBORG_RELAY-00A2.dtbo
```

Usage

```
ls /sys/class/leds
```

The directory "relay1", for instance, exists in the following directory. The LEDs can be controlled by modifying the files in its directory.

```
echo 1 > relay1/brightness
```

This allows you to adjust the brightness; entering 1 for brightness turns it ON, and entering 0 for OFF.

The four relays can be changed individually by changing the number after "relay" in /sys/class/leds/relay.

Code to Get Started

Currently, using sysfs in .c files, libgpiod-dev/gpiod in .c files, and python3 files with the Relay Cape work well!

- For instance, a kernel that I found to work is kernel: *5.10.140-ti-r52*
- Another idea, an image I found that works is *BeagleBoard.org Debian Bullseye Minimal Image 2022-11-01*

There are newer images and kernels if you want to update and there are older ones in case you would like to go back in time to use older kernels and images for the Relay Cape. Please remember that older firmware will work differently on the BeagleBone Black or other related am335x SBC.

C Source with File Descriptors

You can name this file GPIO.c and use gcc to handle compiling the source into a binary like so:

gcc GPIO.c -o GPIO

```
/*

This is an example of programming GPIO from C using the sysfs interface on
a BeagleBone Black/BeagleBone Black Wireless or other am335x board with the␣
↪Relay Cape.

Use the Relay Cape attached to the BeagleBone Black for a change in seconds␣
↪and then exit with CTRL-C.

The original source can be found here by Mr. Tranter: https://github.com/
↪tranter/blogs/blob/master/gpio/part5/demo1.c

Jeff Tranter <jtranter@ics.com>

and...Seth. I changed the source a bit to fit the BeagleBone Black and Relay␣
↪Cape while using sysfs.

*/

#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
```

(continues on next page)

(continued from previous page)

```
int main()
{

// Export the desired pin by writing to /sys/class/leds/relay1/brightness

    int fd = open("/sys/class/leds/relay1/brightness", O_WRONLY);
    if (fd == -1) {
        perror("Unable to open /sys/class/leds/relay1/brightness");
        exit(1);
    }

    fd = open("/sys/class/leds/relay1/brightness", O_WRONLY);
    if (fd == -1) {
        perror("Unable to open /sys/class/leds/relay1/brightness");
        exit(1);
    }

// Toggle LED 50 ms on, 50ms off, 100 times (10 seconds)

    for (int i = 0; i < 100; i++) {
        if (write(fd, "1", 1) != 1) {
            perror("Error writing to /sys/class/leds/relay1/brightness");
            exit(1);
        }
        usleep(50000);

        if (write(fd, "0", 1) != 1) {
            perror("Error writing to /sys/class/leds/relay1/brightness");
            exit(1);
        }
        usleep(50000);
    }

    close(fd);

    // And exit
    return 0;
}
```

C Source with LibGPIOd-dev and File Descriptors

Also…if you are looking to dive into the new interface, libgpiod-dev/gpiod.h, here is another form of source that can toggle the same GPIO listed from the file descriptor.

One thing to note: *sudo apt install cmake*

1. mkdir GPIOd && cd GPIOd
2. nano LibGPIO.c
3. add the below source into the file LibGPIO.c

```
/*
Simple gpiod example of toggling a LED connected to a gpio line from
the BeagleBone Black Wireless and Relay Cape.
Exits with or without CTRL-C.
*/

// This source can be found here: https://github.com/tranter/blogs/blob/
↪master/gpio/part9/example.c
```

(continues on next page)

(continued from previous page)

```
// It has been changed by me, Seth, to handle the RelayCape and BBBW Linux␣
↪based SiP SBC.

// kernel: 5.10.140-ti-r52
// image : BeagleBoard.org Debian Bullseye Minimal Image 2022-11-01

// type gpioinfo and look for this line: line 20: "P9_41B" "relay1" output␣
↪active-high [used]
// That line shows us the info. we need to make an educated decision on what␣
↪fd we will use, i.e. relay1.
// We will also need to locate which chipname is being utilized. For␣
↪instance: gpiochip0 - 32 lines:

// #include <linux/gpio.h>
#include <gpiod.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    const char *chipname = "gpiochip0";
    struct gpiod_chip *chip;
    struct gpiod_line *lineLED;

int i, ret;

// Open GPIO chip
chip = gpiod_chip_open_by_name(chipname);
if (!chip) {
    perror("Open chip failed\n");
    return 1;
}

// Open GPIO lines
lineLED = gpiod_chip_get_line(chip, 20);
if (!lineLED) {
    perror("Get line failed\n");
    return 1;
}

// Open LED lines for output
ret = gpiod_line_request_output(lineLED, "relay1", 0);
if (ret < 0) {
    perror("Request line as output failed\n");
    return 1;
}

// Blink a LED
i = 0;
while (true) {
    ret = gpiod_line_set_value(lineLED, (i & 1) != 0);
    if (ret < 0) {
        perror("Set line output failed\n");
        return 1;
    }
    usleep(1000000);
    i++;
}

// Release lines and chip
gpiod_line_release(lineLED);
```

(continues on next page)