PocketBeagle TechLab Workshop
PocketBeagle walk-through

Friendly to novices and experts alike, the Beagle experience tracks mainline u-boot, Linux and Debian development, while augmenting it to enable development to start as quickly as possible. Attendees will get started interacting with the hardware via the command-line, shell scripts, Python and JavaScript. Attendees will be walked through the configuration details for the boot configuration, pin multiplexing, USB networking and other helper scripts they should get to know. Support and development processes within the BeagleBoard.org community will be covered. Exercises will pave the way for the other workshops to dive into their topic without needing to backtrack excessively on PocketBeagle-specific details.
Author and license

• Author
  - Jason Kridner
    Co-founder BeagleBoard.org, Texas Instruments Sitara apps
    https://beagleboard.org/about

• License
  - Creative Commons Attribution – Share Alike 4.0
    https://creativecommons.org/licenses/by-sa/4.0/
Outline

• BeagleBoard.org, PocketBeagle and BaconBits

• Developer experience
  - Command-line and shell script
  - JavaScript and Python
  - C/C++
  - C on PRUs

• Project examples

• Labs
BeagleBoard.org’s objectives

• Education
  - Design and use of open source SW/HW
  - Embedded computing

• Collaboration
  - Physical computing
  - Robotics
  - Industrial/machine controls
Inspiration from early PCs

http://www.sandywalsh.com/2012_07_01_archive.html
Inspiration from early PCs

• How do people learn about embedded computers with so much ground to cover?

http://www.sandywalsh.com/2012_07_01_archive.html
Inspiration from early PCs

- How do people learn about embedded computers with so much ground to cover?
  - Linux keeps history
  - Affordable -> hackable
  - Open from boot
  - High-level languages
  - Motivate with hardware

http://www.sandywalsh.com/2012_07_01_archive.html
Vision
Vision

• Creating with electronics should be as easy as creating a web page
Vision

- Creating with electronics should be as easy as creating a web page
- Appliances are better than applications
Vision

- Creating with electronics should be as easy as creating a web page
- Appliances are better than applications
- Open source software and hardware enable
  - Collaboration on the problem
  - Ability to understand and improve the fundamentals
Reality

- Boot-to-browser feels too limiting → booting to Debian distro
- Collaborative programming still complex → collaborate at the kernel
- Many possible development environments
  - command-line/ssh, Cloud9 IDE, node-red, pureData, SuperCollider, LabView, Matlab, Eclipse, Visual Studio, Scratch, Blockly
- Domain specific approaches
  - Machinekit/LinuxCNC, PLC, many IoT toolkits
  - Many rapid sensor approaches: capes, mikroBus, Grove/Grove Zero, PMOD
  - Many rapid build approaches: LEGO, printing/milling, Makeblock, Vex, various other aluminum kits
Approach

• Don’t try to boil the ocean
  - We seek to engage the open source community

• Help where we can
  - Blue supports Grove cables
  - PocketBeagle supports mikroBus click pinout
  - Many “BeagleBoard Compatible” devices targeting specific application areas
Board history

Fanless open computer (BeagleBoard)

Mint tin sized with industrial peripherals (BeagleBone)

Application focused BeagleBones

Smalls mint tin sized with super-flexible design - PocketBeagle
PocketBeagle objectives

• Get simple
  - 4-layer PCB done in both Kicad and EAGLE
  - Every expansion header pin has a useful predefined mode

• Get flexible
  - USB to holes, no on-board pin consumption, no header soldered
  - Support for 2 mikroBus Click boards (over 300 already exist)

• Get small
  - Stick with mint-tin survival-kit theme, but go to “smalls” (35mm x 55mm)

• Get low cost
  - System-in-package approach has can lower build costs
  - Launched/sustainable at $25
PocketBeagle key features

• Processing
  - 1-GHz ARM Cortex-A8 processor
  - 2x200-MHz programmable real-time units (PRUs)
  - ARM Cortex-M3 microcontroller for power and security
  - SGX530 graphics processor (OpenGLES)

• Memory
  - 512-MB DDR3
  - 4-KB I2C EEPROM

• Interfaces
  - USB 2.0 OTG
  - microSD
  - 72 expansion header pins
  - 8 analog inputs (6@1.8V, 2@3.3V)
  - 44 digital I/Os (18 enabled)
  - 3 UARTs (2 enabled)
  - 2 I2C ports
  - 2 SPI ports
  - 2 quadrature encoders accessible
  - 2 CAN bus controllers accessible
  - USB, power/reset buttons, battery/DC
PocketBeagle top

Octavo Systems OSD3358-SM

- Power LED
- Clock
- USB 2.0 OTG
- Power
- User LEDs
- P2 Header
- Boot Config
- microSD Card Slot
- P1 Header
PocketBeagle bottom

P2 Header

P1 Header
### PocketBeagle Expansion Headers (Rev A2a)

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th></th>
<th>P2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V_E</td>
<td>GPIO 109</td>
<td>VBUS</td>
<td>GPIO</td>
<td></td>
</tr>
<tr>
<td>VIN</td>
<td>1287</td>
<td>5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>ref</td>
<td>18</td>
<td>4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>ref+</td>
<td>19</td>
<td>5</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>20</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>GND</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>VOUT</td>
<td>224</td>
<td>15</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>28</td>
<td>13</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>29</td>
<td>43</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>31</td>
<td>42</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>33</td>
<td>34</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>36</td>
<td>110</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

---

**Notes:**
- **SYS:** System power.
- **VIN:** Input voltage.
- **GPIO:** General-purpose input/output.
- **CAN:** Controller Area Network.
- **SCI:** Serial Communication Interface.
- **MISO:** MOSI: Master In Slave Out.
- **RX:** Receiver.
- **TX:** Transmitter.
- **PRU:** Programmable Real-Time Unit.
- **PWM:** Pulse Width Modulation.
- **VOUT:** Voltage output.
- **REF:** Reference voltage.
- **AIN1.8V:** Analog input 1.8V.
- **16:** Digital input 16.
- **15:** Digital input 15.
- **14:** Digital input 14.
- **13:** Digital input 13.
- **12:** Digital input 12.
- **11:** Digital input 11.
- **10:** Digital input 10.
- **9:** Digital input 9.
- **8:** Digital input 8.
- **7:** Digital input 7.
- **6:** Digital input 6.
- **5:** Digital input 5.
- **4:** Digital input 4.
- **3:** Digital input 3.
- **2:** Digital input 2.
- **1:** Digital input 1.

---

**BeagleBoard.org**

---

**Reference:**
- PocketBeagle Expansion Headers (Rev A2a) by BeagleBoard.org.
mikroBus Click

- Analog - AN
- Reset - RST
- SPI Chip Select - CS
- SPI Clock - SCK
- SPI Master Input Slave Output - MISO
- SPI Master Output Slave Input - MOSI
- VCC-3.3V power - +3.3V
- Reference Ground - GND

AN - PWM
RST - INT
CS - RX
SCK - TX
MISO - SCL
MOSI - SDA
+3.3V - +5V
GND - GND

PWM - PWM output
INT - Hardware Interrupt
RX - UART Receive
TX - UART Transmit
SCL - I²C Clock
SDA - I²C Data
+5V - VCC-5V power
GND - Reference Ground
Connecting mikroBus Clicks
TechLab objectives

- Designed specifically for e-ale training
- Inspired by “Bacon Cape” by Dave Anders
  - Designed for similar purpose on BeagleBone
- Migrated to PocketBeagle as “BaconBits” by Michael Welling
- Updated to have mikroBus header and made “pretty”
  - Extra button (with PRU option), light sensor rather than potentiometer
- Provides target for common embedded interfaces
  - SPI, I2C, GPIO, PWM, ADC, USB, serial
- Avoid users needing to buy several modules
TechLab features

- USB-to-Serial micro B
- USB Host A with power
- Reset button
- 2 GPIO push buttons (L and R)
- ADC light sensor
- PWM tri-color LED
- SPI 2-digit 7-segment display
- I2C accelerometer
- mikroBus header
TechLab GPIO inputs

Buttons

P1.29/PRU0.7(R BTN)
P2.33/GPIO45(L BTN)
TechLab ADC input

Light Sensor
TechLab PWM output

Multi-colored LED (Bulb)

- P1.35/PRU1.10(RGB.G.ALT)
- P2.1/PWM1A(RGB.G/EXP1.16)
- P1.33/PRU0.1(RGB.R)
- P1.36/PWM0A(RGB.B)

R7 (DNP(1.5k)) -> LD1
R1 (1.5k) -> G
R2 (1.5k) -> R
R3 (1k) -> B

GND
TechLab SPI 7-segment display
TechLab I2C sensor

Accelerometer

Slave Address SA0--
High: 0011101 (0x1D)
Low: 0011100 (0x1C)
TechLab mikroBus header

N6/GPIO8/(EXP1.1) 1
PIO89(EXP1.2) 2
PIO.CS(EXP1.3) 3
PIO.CLK(EXP1.4) 4
PIO.MISO(EXP1.5) 5
PIO.MOSI(EXP1.6) 6

3.3V 7
3V3 8
GND1

GND2 9

VOUT

AN
PWM 10
RST
INT 15
CS
RX 14
SCK
TX 13
MISO
SCL 12
MOSI
SDA 11

P2.1/PWM1A/RGB.G/EX
P2.3/GPIO23(EXP1.15)
P2.5/UART4.RX(EXP1.14)
UART4.TX(EXP1.13)
P2.9/I2C1.SCL(EXP1.12)
P2.11/I2C1.SDA(EXP1.11)
TechLab USB host
TechLab board image
Developer experience

• Customized Debian images – bbb.io/latest
• Self-hosted tools for ARMs (A8/M3) and PRUs
• Libraries for various high-level languages
• Scripts for common tasks
• Sources for bootloader, device tree, etc.
• Servers for network-based development
See labs in the Handouts

- No need to program the microSD card
  - The link is for your reference. This is already done for you.

- Do the first 4 labs
  - “Blink PocketBeagle on-board USRx LED”
  - “Read a button”
  - “Read an analog sensor”
  - “Fade an LED”

- I will interrupt with hints and discussion at intervals
Walk me through the getting started process
Single cable development

- Power, network, develop
- You can add a network and power many other ways
Write image to microSD with Etcher
Insert microSD and boot
Connect to the USB network
Open the IDE
OK, how is this working?
USB gadgets

• Linux name for device/slave drivers
  - ie., when not host

• USB devices have “classes”
  - Mass storage
  - Camera
  - Audio
  - Printer
  - “HID” or human-interface device like mouse and keyboard
  - Communications
USB gadgets

• Default image USB gadgets
  - Virtual mass storage
    • Serves you up README.htm
  - Virtual serial
    • Provides access to console after kernel boot
  - Virtual network
    • Enables access to ssh and web servers
Cloud9 IDE

- Open source project
- Written in JavaScript
  - Node.js
- Hosted on PocketBeagle
  - No cloud server involved
  - No special tools on your computer, just your web browser
Yes, but what happens at startup?
Device Tree

- See kernel documentation for bindings
  - devicetree/bindings/eeprom/eeprom.txt

- Local copies enable you to extend on the fly
  - /opt/source/dtb-4.14-ti
  - /opt/source/bb.org-overlays

- Overlays loaded in u-boot, but also possible via kernel configfs
Here are some more gory details for your reference
TI AM335x: bootrom


SYSBOOT[15:14] = 01 = 24Mhz
SYSBOOT[4:0] = 11000

1. SPI0
2. MMC0 - going to use today
3. USB0 - (node-beagle-boot)
4. UART0
26.1.8.5.3 Booting Procedure

The high level flowchart of the eMMC / eSD and MMC/SD booting procedure is depicted in Figure 26-22.

Figure 26-22. MMC/SD Booting

- **MMC/SD Booting**
  - Initialize the MMC / SD driver
  - Detected card or embedded memory
    - Detected
      - Configure the card address (RSA)
    - Not detected
      - MLO
        - Booting file found? (No)
          - Failed
        - Booting file not found?
          - Get the booting file
        - MLO detected?
          - Raw mode detected?
            - Yes
              - Get raw data
            - No
              - Failed

Page: 5053

TI AM335x: bootrom: raw mode:


1. 0x0 <- (FAT Boot Sector, let's leave it blank...)
2. 0x20000 (128KB) <- We are going to use this location
3. 0x40000 (256KB) <- (2nd “backup” location)
4. 0x60000 (384KB) <- (3rd “backup” location)

Only 128KB in size… (hint, only 128KB of SRAM)
Das U-Boot (the Universal Boot Loader) U-Boot

Original Author: Wolfgang Denk, now maintained by Tom Rini

- https://www.denx.de/wiki/U-Boot
- http://git.denx.de/?p=u-boot.git;a=summary
U-Boot: AM335x

Outputs two files for TI am335x targets:

- MLO = SPL (or Secondary Program Loader)
- u-boot.img (or u-boot-dtb.img) (U-Boot)
U-Boot: SPL

1. Initializes main memory (DDRx for am335x)
2. Loads full (U-Boot) into DDR memory

Or:

1. Initializes main memory (DDRx for am335x)
2. Loads Linux Kernel into DDR memory (aka: Falcon mode, faster boot mode/etc)
U-Boot:

- Network
- USB
- MMC
- File System (fat/extX)
- Shell

Sometimes you don’t need a full OS, have U-Boot init and then have U-Boot load/run your application.
U-Boot:

CPU : AM335X-GP rev 2.1
I2C: ready
DRAM: 512 MiB
Some drivers were not found
Reset Source: Power-on reset has occurred.
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
Model: SeeedStudio BeagleBone Green:
U-Boot: microSD

Insert USB-microSD adapter, and type “lsblk”

```bash
voodoo@hestia:~/Supercon-2017-PocketBeagle$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda     8:0    0 465.8G  0 disk
  └─sda1   8:1    0 465.8G  0 part /
sde     8:64   1   7.4G  0 disk
  └─sde1   8:65   1   7.4G  0 part
```

That’s our 8GB USB Flash Drive

Open: system.sh change: MMC=/dev/sde
U-Boot: Format microSD

```
sudo dd if=/dev/zero of=${MMC} bs=1M count=10

sudo sfdisk ${MMC} <<-__EOF__
4M,,L,*
__EOF__

sudo mkfs.ext4 -L rootfs ${MMC}1

voodoo@hestia:~/Supercon-2017-PocketBeagle$ ./scripts/format_drive.sh
```
U-Boot: (refresh for your memory)


1. 0x0
2. 0x20000 (128KB) <- We are going to use this location
3. 0x40000 (256KB)
4. 0x60000 (384KB)

sudo dd if=./deploy/MLO of=${MMC} count=1 seek=1 bs=128k
sudo dd if=./deploy/u-boot.img of=${MMC} count=2 seek=1 bs=384k
Base Rootfs: Debian 9.x (Stretch)

Maintainer: Robert Nelson (with lots of help from all the Debian Developers and 1000’s of other users)

- [https://www.debian.org/](https://www.debian.org/)
- [https://github.com/beagleboard/image-builder](https://github.com/beagleboard/image-builder)
See more labs in the Handouts

- Do the last 3 labs
  - “Using Node-RED to read and write files”
  - “Explore the Linux command line”
  - “Toggle LED based on a button press using a PRU”
- I will interrupt with hints and discussion at intervals
In Linux, everything becomes a file

- Much to learn
  - I'm used to microcontrollers: just give me the datasheet with register definitions and set me free!

- Training on boot & device drivers useful
  - Often geared more at system bring-up
  - What about the everyday user?
  - Where is that abstraction benefit?

- Let's just walk a working system!
What is the baseline?

http://refsspecs.linuxfoundation.org/lsb.shtml

• Every Linux system may be customized
  - This is the nature of open source
  - Stuff still needs to work together

• The Linux Standard Base
  - Umbrella for various Linux Foundation groups
  - A specification and a testkit
  - Documents typical libraries, functions and files expected to be found by the developer
lsb_release

debian@beaglebone:~$ sudo apt install -y lsb
debian@beaglebone:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 9.5 (stretch)
Release: 9.5
Codename: stretch
Filesystem Hierarchy Standard

http://www.pathname.com/fhs/

- /tmp: temporary files
- /var: data that is changes at run-time
- /proc: "information from processes" (virtual)
- /sys: "system filesystem" (virtual)
- /dev: device files
- /media: mount point for removable media
- /lost+found: data without directory entry
- /mnt: mount point for temporary mounted file systems
- /opt: add-on application software packages
debian@beaglebone:~$ ls /proc
1  1692  22  3354  878  990  fb  misc  sysv ipc
10  17  23  3362  89  apm  filesystems  modules  thread-self
11  18  2370  34  9  asound  fs  mounts  timer_list
1110  1857  2375  4  90  buddyinfo  interrupts  mtd  tty
1112  19  2377  6  91  bus  iomem  net  uptime
1119  1951  2379  69  913  cgroups  iports  pagetypeinfo  version
1150  1964  24  7  918  cmdline  irq  partitions  vmalloc info
1151  2  25  70  92  config.gz  kallsyms  pvr  vmstat
1152  20  26  71  93  consoles  keys  sched_debug  zoneinfo
12  21  27  72  945  cpu  key-users  schedstat
1215  2107  28  73  951  cpuinfo  kmsg  self
1247  2120  29  74  959  crypto  kpagecgroup  slab info
13  2149  30  8  973  devices  kpagecount  softirqs
1440  2152  31  800  977  device-tree  kpageflags  stat
15  2153  32  820  980  diskstats  loadavg  swaps
159  2155  33  821  983  driver  locks  sys
16  2173  3353  858  984  execdomains  meminfo  sysrq-trigger
debian@beaglebone:~$ cat /proc/cpuinfo
processor : 0
model name : ARMv7 Processor rev 2 (v7l)
BogoMIPS : 995.32
Features : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x3
CPU part : 0xc08
CPU revision : 2

Hardware : Generic AM33XX (Flattened Device Tree)
Revision : 0000
Serial : 1741GPB42934
The file interface abstraction

• What can I do with files?
  - open, read, write, close, delete
  - What is an "ioctl"?
  - What is "mmap"?

• What is a virtual file system?
  - Looks like a file, but executes code in the kernel
  - Not really storing anything to media
  - A bit like a "ram disk"
Kernel.org documentation

http://www.kernel.org/doc/

- Documentation extracted from the Linux kernel and mirrored on the web where Google can find it:
  - Documentation - Text files in the kernel source tarball's Documentation subdirectory
  - htmldocs - Kernel Documentation maintained in docbook format (output of "make htmldocs")
  - Menuconfig - help text for each kernel configuration option (from kconfig source)
  - README various README files scattered around Linux kernel source
  - RFC - List of IETF RFCs referred to by kernel source files. Links to both the text of the RFC and the source files that refer to it
  - Output of kernel's "make help"
- Standards documents applicable to the Linux kernel
- Other web pages containing kernel documentation
- Translations to other languages
- Documentation on memory management
- Miscellaneous
Kernel Application Binary Interface

http://www.kernel.org/doc/Documentation/ABI/

- Low-level kernel interface from "userland"
- Status of interface
  - Stable
    - Encouraged to use freely
    - Guaranteed for at least two years
  - Testing
    - Mostly complete, but might change
    - Let developers know how you are using
    - Where you’ll find most of the good stuff
  - Obsolete
    - Scheduled for removal
  - Removed
Kernel Application Binary Interface

http://www.kernel.org/doc/Documentation/ABI/

- Types of interfaces
  - Syscalls
    - Trap interface with IDs
    - May be possible to have a direct entry
  - SYSFS
    - Virtual file system
    - See also DEBUGFS and CONFIGFS
Syscalls


- open/read/write/lseek/close/unlink
- ioctl
- mknod
- fork/select/poll/...
- mkdir/...
- mount/umount
- mmap
What is SYSFS?

- Virtual file system that exposes drivers to userspace
- `mount l grep sysfs`
  - sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
- `/sys/devices` - driver hierarchy
- `/sys/bus` - symbolic links to bus owners
- `/sys/class` - common interfaces
- `/sys/block` - block interface
- How about some examples?
```
ls /sys/module
```

**/sys/module/MODULENAME**
- .../parameters: options you can provide
- .../refcnt: number of times in use
/sys/class/leds

https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-class-led

- /sys/class/leds/LED
  - ./brightness: 0-max_brightness, >0 = on
  - ./max_brightness: default is 255
  - ./trigger: triggers available from kernel
  - ./inverted: invert on/off state
/sys/class/gpio

http://www.kernel.org/doc/Documentation/ABI/testing/sysfs-gpio

- Must be explicitly exported to userspace and not claimed by kernel code
- /sys/class/gpio
  - .../export: asks the kernel to export a GPIO to userspace
  - .../unexport: to return a GPIO to the kernel
  - .../gpioN: for each exported GPIO #N
    - .../value: always readable, writes fail for input GPIOs
    - .../direction: r/w as: in, out (low); write: high, low
    - .../edge: r/w as: none, falling, rising, both
  - .../gpiochipN: for each gpiochip; #N is its first GPIO
    - .../base: (r/o) same as N
    - .../label: (r/o) descriptive, not necessarily unique
    - .../ngpio: (r/o) number of GPIOs; numbered N to N + (ngpio - 1)
On-chip peripherals (OCP)

debian@beaglebone:~$ ls /sys/devices/platform/ocp
40300000.ocmcram  480c8000.mailbox  53100000.sham  ocp:P1_32_pinmux ocp:P2_20_pinmux
44e070000.gpio  480ca000.spinlock  53500000.aes  ocp:P1_33_pinmux ocp:P2_22_pinmux
44e090000.serial  4819c000.i2c  56000000.sgx  ocp:P1_34_pinmux ocp:P2_24_pinmux
44e0b0000.i2c  481a0000.mailbox  driver_override  ocp:P1_35_pinmux ocp:P2_25_pinmux
44e0d0000.tscadc  481a8000.serial  modalias  ocp:P1_36_pinmux ocp:P2_27_pinmux
44e350000.wdt  481ac000.gpio  ocp:cape-universal  ocp:P2_01_pinmux ocp:P2_28_pinmux
44e3e0000 rtc  481ae000.gpio  ocp:l4_wkup@44c00000 ocp:P2_02_pinmux ocp:P2_29_pinmux
47400000.usb  481cc000.can  ocp:P1_02_pinmux ocp:P2_03_pinmux ocp:P2_30_pinmux
480220000.serial  481d0000.can  ocp:P1_04_pinmux ocp:P2_04_pinmux ocp:P2_31_pinmux
480240000.serial  48300000.epwmss  ocp:P1_06_pinmux ocp:P2_05_pinmux ocp:P2_32_pinmux
4802a0000.i2c  48302000.epwmss  ocp:P1_08_pinmux ocp:P2_06_pinmux ocp:P2_33_pinmux
480300000.spi  48304000.epwmss  ocp:P1_10_pinmux ocp:P2_07_pinmux ocp:P2_34_pinmux
480420000.timer  48310000.rng  ocp:P1_12_pinmux ocp:P2_08_pinmux ocp:P2_35_pinmux
480440000.timer  49000000.edma  ocp:P1_20_pinmux ocp:P2_09_pinmux of_node
480460000.timer  49800000.tptc  ocp:P1_26_pinmux ocp:P2_10_pinmux power
480480000.timer  49900000.tptc  ocp:P1_28_pinmux ocp:P2_11_pinmux subsystem
4804a0000.timer  49a000000.tptc  ocp:P1_29_pinmux ocp:P2_17_pinmux uevent
4804c0000.gpio  4a326004.pruss-soc-bus  ocp:P1_30_pinmux ocp:P2_18_pinmux
480600000.mmc  4c000000.emif  ocp:P1_31_pinmux ocp:P2_19_pinmux
Reading events

- TBD
config-pin


debian@beaglebone:~$ config-pin -i p1.36
Pin name: P1_36
Function if no cape loaded: pwm
Function if cape loaded: default gpio gpio_pu gpio_pd gpio_input spi_sclk pwm pruout pruin
Function information: ehrpwm0a default gpio3_14 gpio3_14 gpio3_14 gpio3_14 spi1_sclk ehrpwm0a pru0_out0 pru0_in0
Kernel GPIO id: 110
PRU GPIO id: 142
debian@beaglebone:~$ config-pin -q p1.36
P1_36 Mode: default Direction: in Value: 0
debian@beaglebone:~$ config-pin p1.36 pruout
debian@beaglebone:~$ config-pin -q p1.36
P1_36 Mode: pruout
show-pins.pl

- perl /opt/scripts/device/bone/show-pins.pl -v
Enabling PRU

- 2 possible drivers: remoteproc or uio
- Enabled via device tree at boot
  - Different systems might have different defaults
mikroBus Click usage

- See bbb.io/pbmb
- Supported with device-tree overlays loaded in u-boot
Some work in progress

- Add proxy for various services (in Buster IoT images today)
- Integrate common web-based WiFi provisioning
  - SeeedStudio BeagleBone Green Wireless ships with ‘wifidog’ → we will unify approach
- Cross-platform distro installer app
  - See USB NETCONSOLE presentation
- Support for Grove modules and mikroBus clicks
  - Focus on device-tree overlays and kernel patches
- Integration alignment with complete domain solutions
  - Intelligent Agent Replicape/Revolve, Bela Mini, BeagleLogic, PocketPilot, etc.
- Improved and integrated PRU examples
- Move to distro friendly approaches for customizations
Contributions and issues

- Cape/add-on support
  - https://github.com/beagleboard/bb.org-overlays

- Image deltas
  - https://github.com/beagleboard/image-builder

- In-system examples
  - https://github.com/beagleboard/bone101
Thank you!