hwspelunk-elc2020

Spelunking for Hardware Data

Matt Porter <mporter@konsulko.com>

e-ale

The E-ALE (Embedded Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the field of Embedded Linux.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you've learned.

> This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at **https://e-ale.org/**

Contents

Chapter 1

Preliminaries

e-ale

1.1 Introductions

About Me

- CTO at Konsulko Group
- Using Linux since 1992
- Professional embedded Linux engineer since 1998
- Previously maintained kernel support for embedded PPC platforms, RapidIO subsystem, and Broadcom Mobile SoCs
- Various contributions around the kernel

About Konsulko Group

- Konsulko Group is a services company founded by embedded Linux veterans
- Community and commercial embedded, Linux, and Open Source Software development
- Linux Foundation training partners
- See `https://www.konsulko.com` for more information

1.2 Getting Started

Slides

- Download the slides for local reference
- https://cm.e-ale.org/2020/ELC2020/hwspelunk/hwspelunk-elc2020-SLIDES.pdf

What Are We Going To Do?

Using real world examples of a paddle-style joystick on the BaconBits cape and an accelerometer-based joystick on the TechLab cape we will learn the following:

- How to read schematics
- How to follow schematic entities to datasheets
- How to convert datasheet info to Device Tree or driver data
- Practice our new skills with some lab exercises!

Chapter 2

Spelunking Baconbits

2.1 BaconBits Hardware

Component Placement

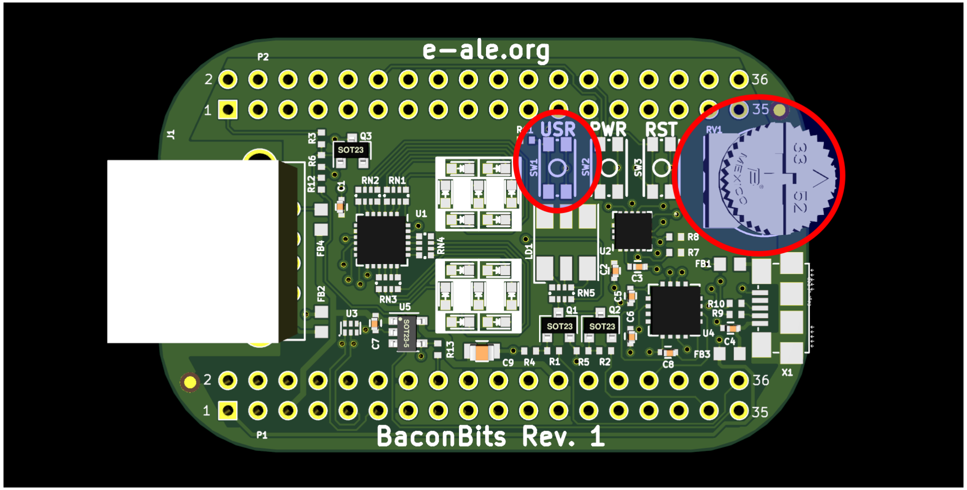

Figure 2.1: **BaconBits Component Identification**

- **RV1** is the thumbwheel device

BaconBits Schematic Overview

- https://github.com/e-ale/BaconBitsCapeHW/blob/master/baconbits.pdf

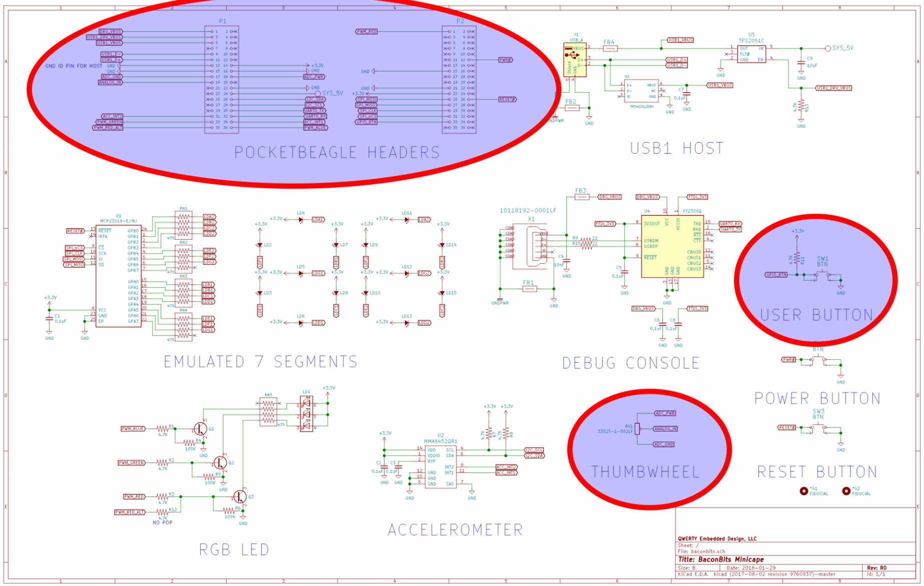

Figure 2.2: **BaconBits Schematic**

BaconBits Thumbwheel

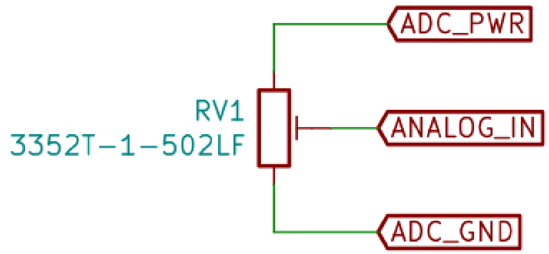

Figure 2.3: **BaconBits Thumbwheel**

- Signals:
  - **ADC_GND**
  - **ADC_PWR**
  - **ANALOG_IN**

BaconBits P1 Connector

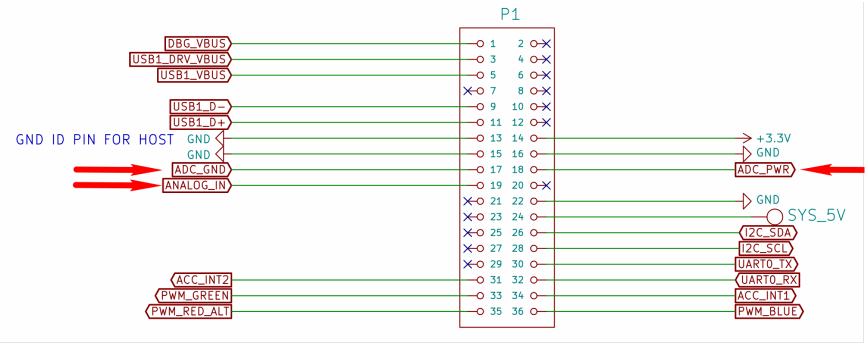

Figure 2.4: **BaconBits P1 Connector**

- Pins:
  - **ADC_GND : P1-17**
  - **ADC_PWR : P1-18**
  - **ANALOG_IN : P1-19**

2.2 PocketBeagle Hardware

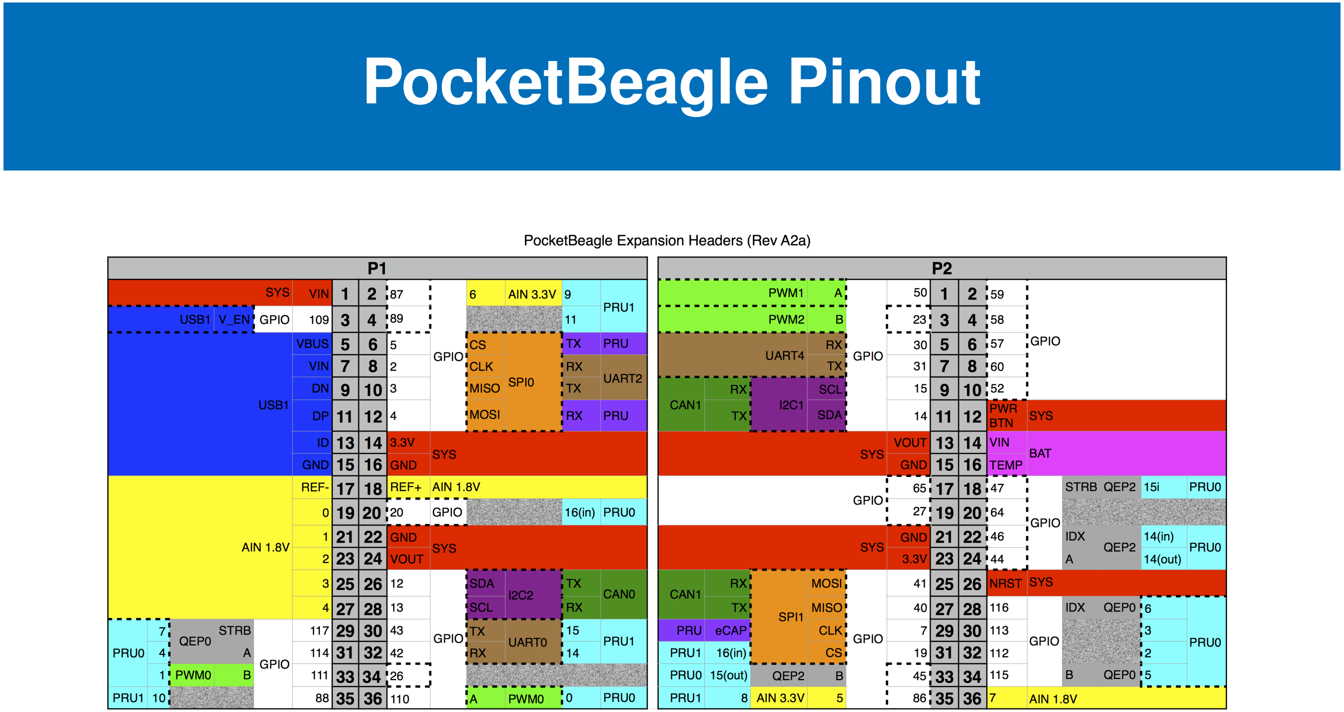

Figure 2.5: **PocketBeagle Expansion Header**

2.3 Summary

Hardware Investigation Results

- Thumbwheel:
  - Connected to analog input 0 (**AIN0**)
- User Button:
  - Connected to **GPMC_AD13** which can be muxed as **GPIO1_13**
  - Active low

2.4 Device Tree

What is Needed?

- Mux the **GPMC_AD13** pin as **GPIO1_13**
- Create a paddle device with a compatible string
- Link to the GPIO pinmux node
- Link to ADC channel 0 for the thumbwheel
- Link to **GPIO1_13** for the button

AM335x GPIO1_13 Pin Mux Register

- Note that **GPIO1_13** is at offset **0x834**
- https://www.ti.com/lit/ug/spruh73p/spruh73p.pdf

800h	conf_gpmc_ad0	See the device datasheet for information on default pin mux configurations. Note that the device ROM may change the default pin mux for certain pins based on the SYSBOOT mode settings.	Section 9.3.1.50
804h	conf_gpmc_ad1		Section 9.3.1.50
808h	conf_gpmc_ad2		Section 9.3.1.50
80Ch	conf_gpmc_ad3		Section 9.3.1.50
810h	conf_gpmc_ad4		Section 9.3.1.50
814h	conf_gpmc_ad5		Section 9.3.1.50
818h	conf_gpmc_ad6		Section 9.3.1.50
81Ch	conf_gpmc_ad7		Section 9.3.1.50
820h	conf_gpmc_ad8		Section 9.3.1.50
824h	conf_gpmc_ad9		Section 9.3.1.50
828h	conf_gpmc_ad10		Section 9.3.1.50
82Ch	conf_gpmc_ad11		Section 9.3.1.50
830h	conf_gpmc_ad12		Section 9.3.1.50
834h	conf_gpmc_ad13		Section 9.3.1.50
838h	conf_gpmc_ad14		Section 9.3.1.50
83Ch	conf_gpmc_ad15		Section 9.3.1.50
840h	conf_gpmc_a0		Section 9.3.1.50

Figure 2.6: **AM335x Pin Mux Registers**

IIO provider binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

```
==IIO providers==

Required properties:
#io-channel-cells: Number of cells in an IIO specifier; Typically 0 for nodes
                   with a single IIO output and 1 for nodes with multiple
                   IIO outputs.

Example for a simple configuration with no trigger:

        adc: voltage-sensor@35 {
                compatible = "maxim,max1139";
                reg = <0x35>;
                #io-channel-cells = <1>;
        };
        .
        .
        .
```

IIO consumer binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

```
==IIO consumers==

Required properties:
io-channels:    List of phandle and IIO specifier pairs, one pair
                for each IIO input to the device. Note: if the
                IIO provider specifies '0' for #io-channel-cells,
                then only the phandle portion of the pair will appear.

Optional properties:
io-channel-names:
                List of IIO input name strings sorted in the same
                order as the io-channels property. Consumers drivers
                will use io-channel-names to match IIO input names
                with IIO specifiers.

For example:

        device {
                io-channels = <&adc 1>, <&ref 0>;
                io-channel-names = "vcc", "vdd";
        };
```

TI TSC ADC binding

Documentation/devicetree/bindings/input/touchscreen/ti-tsc-adc.txt:

```
                .
                .
                .
- child "adc"
        compatible: Should be
                    "ti,am3359-adc" for AM335x/AM437x SoCs
                    "ti,am654-adc", "ti,am3359-adc" for AM654 SoCs
        ti,adc-channels: List of analog inputs available for ADC.
                         AIN0 = 0, AIN1 = 1 and so on till AIN7 = 7.

                .
                .
                .
```

Pinctrl client binding

Documentation/devicetree/bindings/pinctrl/pinctrl-bindings.txt:

```
Required properties:
pinctrl-0:      List of phandles, each pointing at a pin configuration
                node. These referenced pin configuration nodes must be child
                nodes of the pin controller that they configure.
.
.
.
Optional properties:
pinctrl-1:      List of phandles, each pointing at a pin configuration
                node within a pin controller.
.
.
.
For example:

        /* For a client device requiring named states */
        device {
                pinctrl-names = "active", "idle";
                pinctrl-0 = <&state_0_node_a>;
                pinctrl-1 = <&state_1_node_a &state_1_node_b>;
        };
```


GPIO consumer binding

Documentation/devicetree/bindings/gpio/gpio.txt:

```
.
.
.
GPIO properties should be named "[<name>-]gpios", with <name> being the purpose
of this GPIO for the device.
.
.
.
Example of a node using GPIOs:

        node {
                enable-gpios = <&qe_pio_e 18 GPIO_ACTIVE_HIGH>;
        };

GPIO_ACTIVE_HIGH is 0, so in this example gpio-specifier is "18 0" and encodes
GPIO pin number, and GPIO flags as accepted by the "qe_pio_e" gpio-controller.
.
.
.
```


Implementation

DT changes shown against mainline kernel am335x_pocketbeagle.dts

- User button GPIO pinmux configuration:

```
gpio1_13_pin: pinmux-gpio1-13-pin {
        pinctrl-single,pins = <
                AM33XX_IOPAD(0x0834, PIN_INPUT | MUX_MODE7)
        >;
};
```

- Paddle device node:

```
paddle {
        compatible = "e-ale,baconbits-paddle";
        pinctrl-0 = <&gpio1_13_pin>;
        io-channels = <&am335x_adc 0>;
        io-channel-names = "thumbwheel";
        button-gpios = <&gpio1 13 GPIO_ACTIVE_LOW>;
};
```

Chapter 3

Spelunking TechLab

3.1 TechLab Hardware

TechLab Cape

- The TechLab Cape has an accelerometer and a couple buttons that can be used as the basis for a joystick device.

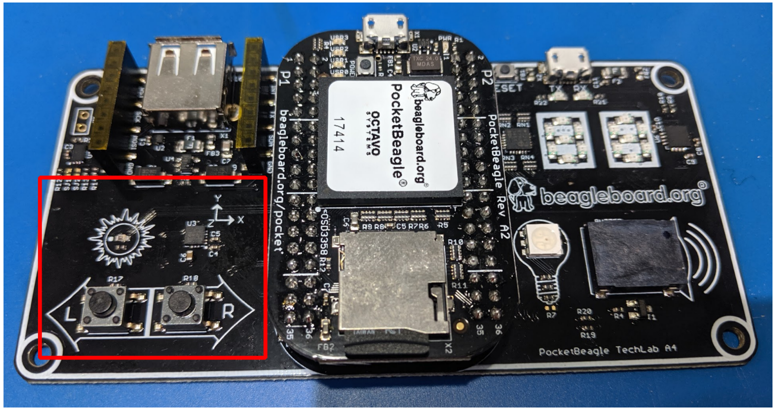

Figure 3.1: **TechLab Cape**

TechLab Schematic Overview

- https://github.com/beagleboard/capes/raw/master/pocketbeagle/TechLab/TechLab_Cape_sch.pdf

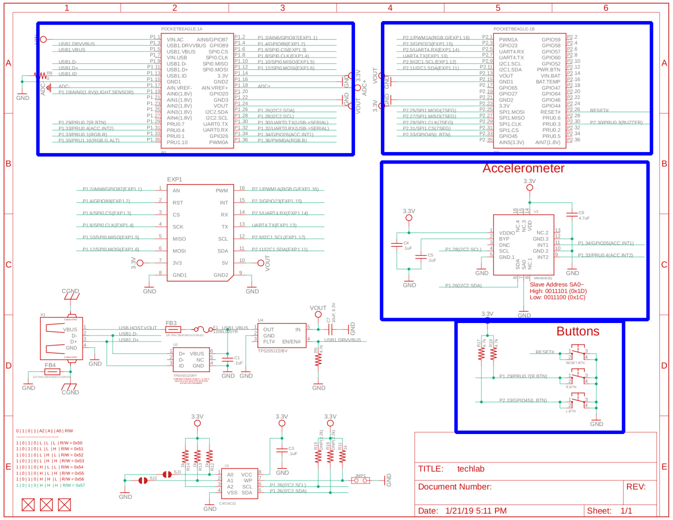

Figure 3.2: **TechLab Schematic**

TechLab Accelerometer

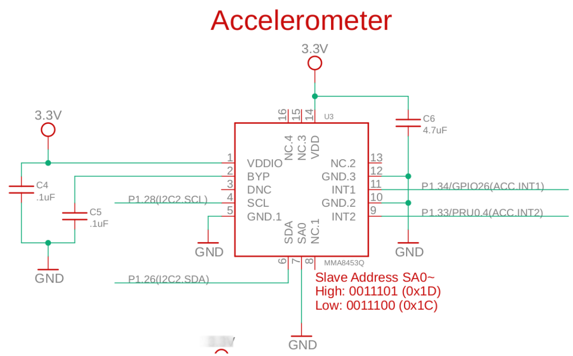

Figure 3.3: **TechLab Accelerometer**

- Signals:
  - **P1.28(I2C2.SCL)**
  - **P1.26(I2C2.SDA)**
- I2C address **0x1C**

TechLab Buttons

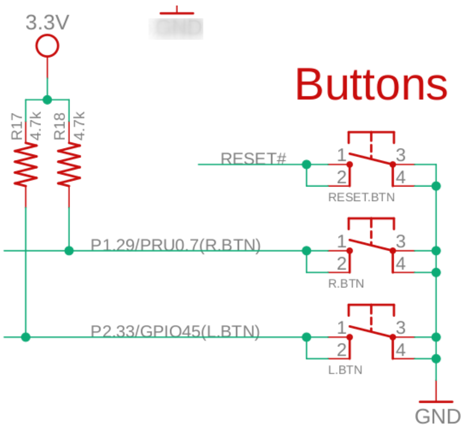

Figure 3.4: **TechLab Buttons**

- Signals:
  - **P1.29/PRU0.7(R.BTN)**
  - **P2.33/GPIO45(L.BTN)**

3.2 PocketBeagle Hardware

PocketBeagle Pinout

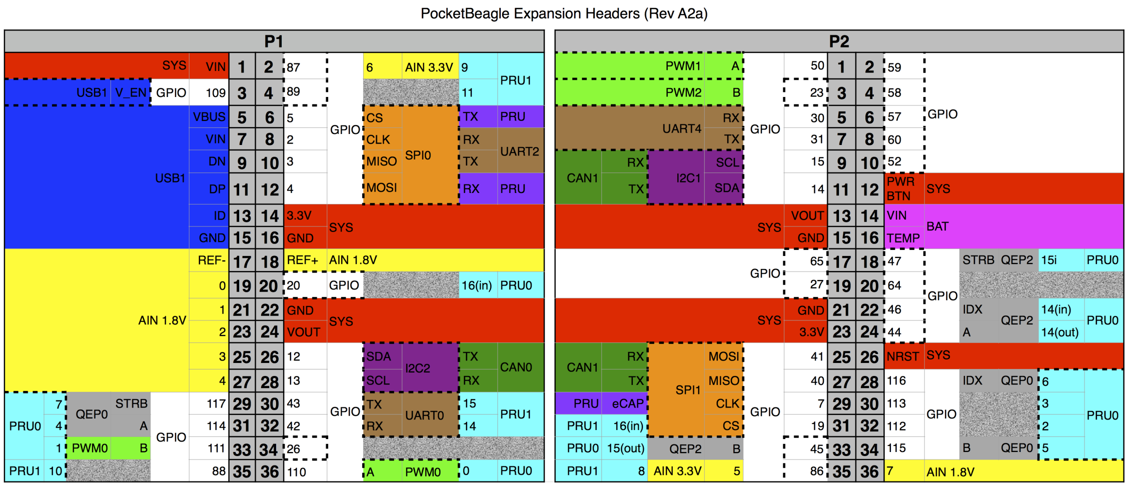

Figure 3.5: **PocketBeagle Expansion Header**

3.3 MMA8453

MMA8453 Resolution

https://www.nxp.com/docs/en/data-sheet/MMA8453Q.pdf

5.2 8-bit or 10-bit data

The measured acceleration data is stored in the OUT_X_MSB, OUT_X_LSB, OUT_Y_MSB, OUT_Y_LSB, OUT_Z_MSB, and OUT_Z_LSB registers as 2's complement 10-bit numbers. The most significant 8-bits of each axis are stored in OUT_X (Y, Z)_MSB, so applications needing only 8-bit results can use these three registers and ignore OUT_X,Y, Z_LSB. To do this, the F_READ bit in CTRL_REG1 must be set. When the F_READ bit is cleared, the fast read mode is disabled.

When the full-scale is set to 2 *g*, the measurement range is –2 *g* to +1.9961 *g*, and each count corresponds to 1 *g*/256 (3.9 mg) at 10-bits resolution. When the full-scale is set to 8 *g*, the measurement range is 8 *g* to +7.9844 *g*, and each count corresponds to 1 *g*/64 (15.6 mg) at 10-bits resolution. The resolution is reduced by a factor of 4 if only the 8-bit results are used. For more information on the data manipulation between data formats and modes, refer to NXP application note AN4076. There is a device driver available that can be used with the Sensor Toolbox demo board (LFSTBEB8451, 2, 3Q).

Figure 3.6: **MMA8453 Resolution**

- 10-bit samples (verified by inspection of mma8452.c kernel driver)
- In full resolution 2g mode, this means a range of -255 to 256

3.4 Summary

Hardware Investigation Results

- Accelerometer:
  - I2C SCL (**P1.28(I2C2.SCL)**)
  - I2C SDA (**P1.26(I2C2.SDA)**)
- Buttons:
  - Left (GPIO pull-up) (**P2.33/GPIO45(L.BTN)**)
  - Right (GPIO pull-up) (**P1.29/PRU0.7(R.BTN)**)

3.5 Device Tree

IIO provider binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

```
==IIO providers==

Required properties:
#io-channel-cells: Number of cells in an IIO specifier; Typically 0 for nodes
                   with a single IIO output and 1 for nodes with multiple
                   IIO outputs.

Example for a simple configuration with no trigger:

        adc: voltage-sensor@35 {
                compatible = "maxim,max1139";
                reg = <0x35>;
                #io-channel-cells = <1>;
        };
        .
        .
        .
```

IIO consumer binding

Documentation/devicetree/bindings/iio/iio-bindings.txt:

```
==IIO consumers==

Required properties:
io-channels:    List of phandle and IIO specifier pairs, one pair
                for each IIO input to the device. Note: if the
                IIO provider specifies '0' for #io-channel-cells,
                then only the phandle portion of the pair will appear.

Optional properties:
io-channel-names:
                List of IIO input name strings sorted in the same
                order as the io-channels property. Consumers drivers
                will use io-channel-names to match IIO input names
                with IIO specifiers.

For example:

        device {
                io-channels = <&adc 1>, <&ref 0>;
                io-channel-names = "vcc", "vdd";
        };
```

MMA8453 binding

Documentation/devicetree/bindings/iio/accel/mma8452.txt:

```
Freescale MMA8451Q, MMA8452Q, MMA8453Q, MMA8652FC, MMA8653FC or FXLS8471Q
triaxial accelerometer

Required properties:

  - compatible: should contain one of
    * "fsl,mma8451"
    * "fsl,mma8452"
    * "fsl,mma8453"
    * "fsl,mma8652"
    * "fsl,mma8653"
    * "fsl,fxls8471"

  - reg: the I2C address of the chip

Optional properties:

  - interrupts: interrupt mapping for GPIO IRQ

  - interrupt-names: should contain "INT1" and/or "INT2", the accelerometer's
                     interrupt line in use.

Example:

        mma8453fc@1d {
                compatible = "fsl,mma8453";
                reg = <0x1d>;
                interrupt-parent = <&gpio1>;
                interrupts = <5 0>;
                interrupt-names = "INT2";
```

Pinctrl client binding

Documentation/devicetree/bindings/pinctrl/pinctrl-bindings.txt:

```
Required properties:
pinctrl-0:      List of phandles, each pointing at a pin configuration
                node. These referenced pin configuration nodes must be child
                nodes of the pin controller that they configure.
.
.
.
Optional properties:
pinctrl-1:      List of phandles, each pointing at a pin configuration
                node within a pin controller.
.
.
.
For example:

        /* For a client device requiring named states */
        device {
                pinctrl-names = "active", "idle";
                pinctrl-0 = <&state_0_node_a>;
                pinctrl-1 = <&state_1_node_a &state_1_node_b>;
        };
```


GPIO consumer binding

Documentation/devicetree/bindings/gpio/gpio.txt:

```
.
.
.
GPIO properties should be named "[<name>-]gpios", with <name> being the purpose
of this GPIO for the device.
.
.
.
Example of a node using GPIOs:

        node {
                enable-gpios = <&qe_pio_e 18 GPIO_ACTIVE_HIGH>;
        };

GPIO_ACTIVE_HIGH is 0, so in this example gpio-specifier is "18 0" and encodes
GPIO pin number, and GPIO flags as accepted by the "qe_pio_e" gpio-controller.
.
.
.
```


pocketbeagle.dts P1.29 pinctrl node

```
/* P1_29 (ZCZ ball A14) pruin */
P1_29_pinmux {
        compatible = "bone-pinmux-helper";
        status = "okay";
        pinctrl-names = "default", "gpio", "gpio_pu", "gpio_pd",
                        "gpio_input", "qep", "pruout", "pruin";
        pinctrl-0 = <&P1_29_default_pin>;
        pinctrl-1 = <&P1_29_gpio_pin>;
        pinctrl-2 = <&P1_29_gpio_pu_pin>;
        pinctrl-3 = <&P1_29_gpio_pd_pin>;
        pinctrl-4 = <&P1_29_gpio_input_pin>;
        pinctrl-5 = <&P1_29_qep_pin>;
        pinctrl-6 = <&P1_29_pruout_pin>;
        pinctrl-7 = <&P1_29_pruin_pin>;
};
```


pocketbeagle.dts P2.33 pinctrl node

```
/* P2_33 (ZCZ ball R12) */
        P2_33_pinmux {
                compatible = "bone-pinmux-helper";
                status = "okay";
                pinctrl-names = "default", "gpio", "gpio_pu", "gpio_pd",
                                "gpio_input", "qep", "pruout";
                pinctrl-0 = <&P2_33_default_pin>;
                pinctrl-1 = <&P2_33_gpio_pin>;
                pinctrl-2 = <&P2_33_gpio_pu_pin>;
                pinctrl-3 = <&P2_33_gpio_pd_pin>;
                pinctrl-4 = <&P2_33_gpio_input_pin>;
                pinctrl-5 = <&P2_33_qep_pin>;
                pinctrl-6 = <&P2_33_pruout_pin>;
        };
```

P1.29 GPIO resource

http://www.ti.com/lit/ds/symlink/am3352.pdf

ZCZ BALL NUMBER [1]	PIN NAME [2]	SIGNAL NAME [3]	MODE [4]
A14	MCASP0_AHCLKX	mcasp0_ahclkx	0
		eQEP0_strobe	1
		mcasp0_axr3	2
		mcasp1_axr1	3
		EMU4	4
		pr1_pru0_pru_r30_7	5
		pr1_pru0_pru_r31_7	6
		gpio3_21	7

Figure 3.7: **P1.29 ZCZ A14 (gpio3_21)**

P2.33 GPIO resource

http://www.ti.com/lit/ds/symlink/am3352.pdf

R12	GPMC_AD13	gpmc_ad13	0
		lcd_data18	1
		mmc1_dat5	2
		mmc2_dat1	3
		eQEP2B_in	4
		pr1_mii0_txd1	5
		pr1_pru0_pru_r30_15	6
		gpio1_13	7

Figure 3.8: **P2.33 ZCZ R12 (gpio1_13)**

Describing the Joystick Hardware

- Specify the MMA8453 accelerometer
- Specify the Left and Right buttons used by joystick
- Specifiy the MMA8453 channels used by joystick
- Specify pinmux settings needed for buttons
- Specify GPIO resources needed for buttons

Implementation

Shown as a modification of the beagleboard.org Linux kernel am335x_pocketbeagle.dts

```
&i2c2 {
        #address-cells = <1>;
        #size-cells = <0>;
        mma8453: mma8453@1c {
                #io-channel-cells = <1>;
                status = "okay";
                compatible = "fsl,mma8453";
                reg = <0x1c>;
        };

};
```


Implementation (continued)

```
        .
        .
        .
P2_33_pinmux { status = "disabled"; };  /* Left - gpio3_21 */
P1_29_pinmux { status = "disabled"; };  /* Right - gpio1_13 */
cape-universal { status = "disabled"; };

joystick {
        compatible = "bborg,techjoy";
        pinctrl-0 = <&P2_33_gpio_input_pin>,
                    <&P1_29_gpio_input_pin>;
        io-channels = <&mma8453 0>, <&mma8453 1>;
        io-channel-names = "accel_x", "accel_y";
        button-gpios = <&gpio3 21 GPIO_ACTIVE_LOW>,
                       <&gpio1 13 GPIO_ACTIVE_LOW>;
        };
};
        .
        .
        .
```


Chapter 4

Labs

4.1 Preparation

Documentation Required

- https://github.com/beagleboard/capes/raw/master/pocketbeagle/TechLab/TechLab_Cape_sch.pdf
- https://github.com/beagleboard/pocketbeagle/raw/master/PocketBeagle_sch.pdf
- https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf
- http://www.ti.com/lit/ds/symlink/am3352.pdf
- Figure 3.5: Pocketbeagle Expansion Header
- https://github.com/beagleboard/linux/blob/4.14/arch/arm/boot/dts
- https://www.kernel.org/doc/Documentation/devicetree/bindings/

4.2 Lab 1

TechLab Light Sensor

Do the following:

- Find the Light Sensor on the TechLab
- Document hardware signal(s) it uses
- Document the DT property and value that expresses the ADC channel used by this sensor

4.3 Lab 2

TechLab Buzzer

Do the following:

- Find the Buzzer on the TechLab
- Document the hardware signal(s) it uses
- Document the DT property and value that expresses the GPIO used by the buzzer

4.4 Lab 3

TechLab Multi-colored LED

Do the following:

- Find the Multi-colored LED on the TechLab
- Document the hardware signal(s) it uses for RGB.
- Document the DT property and value that expresses the PWMs used by the LED

Hint: see Documentation/devicetree/bindings/pwm/pwm.txt (or .yaml) and pwm-tiehrpwm.txt for PWM binding properties. Also consult the AM335x TRM for hardware details on the Enhanced PWM peripheral.