

# A2D Guide

by Brian Fraser

Last update: Jan 28, 2021

## This document guides the user through

1. Reading the value of an A2D input on the BeagleBone via the command line terminal.
2. Using a C program to access the A2D.

## Target kernels v4.9; no cape manager

## Table of Contents

1. A2D Basics.....	2
2. Enabling the A2D in Linux.....	2
3. C Code.....	4

## Formatting

1. Commands for the host Linux's console are show as:  
`(host)$ echo "Hello PC world!"`
2. Commands for the target (BeagleBone) Linux's console are shown as:  
`(bbg)$ echo "Hello embedded world!"`
3. Almost all commands are case sensitive.

## Revision History

- Sept 17, 2019: Changed to support not using the cape manager.
- Jan 28: Changed how prompts are shown

## 1. A2D Basics

Internally in a computer, values are stored as digital: either on (1) or off (0). In reality, signals are analog, which means they are a voltage level and not just on or off. The potentiometer, for examples, is a knob the user can turn and generate a voltage between some min and max levels. The Analog to Digital converter (A2D or ADC) is used convert an analog signal (such as 1.2V) into a digital number in the computer (such as 2731).

The hardware has a limited range of voltages it can tolerate without being damaged. On the BeagleBone, this range is 0 to 1.8V. Be very careful not to exceed 1.8V on the input, even though there are 3.3V and 5V voltages also on the BeagleBone. On the Zen cape, the potentiometer (POT) has been wired to give a value between 0 and 1.8V.

## 2. Enabling the A2D in Linux

All A2D pins are controlled through Linux, so we must know which pin we want to access.

1. Determine which A2D input channel is being used. On the BeagleBone, the P9 expansion headers allow easy access to the 7 analog inputs.

# 7 analog inputs (1.8V)

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Source: <http://beagleboard.org/support/bone101>

- For the Zen cape, the potentiometer is wired to AIN0 (analog input 0)

2. Change to the `sysfs` directory for the A2D readings. The `sys` file system gives access to many Linux devices. The folder is `/sys/bus/iio/devices/iio:device0` however the ':' must be escaped for the Linux command line (but not in your C program!):<sup>1</sup>

```
(bbg)$ cd /sys/bus/iio/devices/iio\:device0
```

3. To read voltage 0:

```
(bbg)$ cat in_voltage0_raw
```

- Change the 0 to the desired channel number.
- The value will be between 0 and 4095 (4K – 1)

You can compute the voltage with the formula:

$$\text{voltage} = (\text{value} / \text{max}) * \text{reference\_voltage}$$

So, if you just read 3103 it relates to the real world voltage of:

$$\begin{aligned} \text{voltage} &= (3103 / 4095) * 1.8 \\ &= 1.36\text{V} \end{aligned}$$

- *Tip: If doing this sort of math in C, make sure you use the correct data types to avoid a rather unhelpful integer division.*

4. Troubleshooting:

- If you are changing the position of the potentiometer but the value being read from `in_voltage0_raw` does not change, ensure the hardware is correctly connected. For example, if you are using the Zen cape, ensure it is fully seated on the BBG (virtually none of the P8/P9 pins should be visible between the BBG and Zen).
- If the file `/sys/bus/iio/devices/iio:device0/in_voltage0_raw` does not exist, then double check the following in `/boot/uEnv.txt`:
  - Enabled the UBoot overlays:  
`enable_uboot_overlays=1`
  - Commented out (with a # in front) the A2D disable command:  
`#disable_uboot_overlay_adc=1`

<sup>1</sup> On older kernels, such as early 4.x, the A2D cape must be loaded using the cape manager, with a command such as:  
`echo BB-ADC > /sys/devices/platform/bone_capemgr/slots`  
On the 4.9 kernel, this functionality is done through the UBoot bootloader, with the A2D cape loaded by default. See `/boot/uEnv.txt` for where to configure the device overlays in UBoot.

### 3. C Code

Assuming the cape is already loaded, the following program will display the current A2D reading and the voltage it relates to until the user hits control-c:

```
// Demo application to read analog input voltage 0 on the BeagleBone
// Assumes ADC cape already loaded by uBoot:

#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>

#define A2D_FILE_VOLTAGE0  "/sys/bus/iio/devices/iio:device0/in_voltage0_raw"
#define A2D_VOLTAGE_REF_V  1.8
#define A2D_MAX_READING    4095

int getVoltage0Reading()
{
    // Open file
    FILE *f = fopen(A2D_FILE_VOLTAGE0, "r");
    if (!f) {
        printf("ERROR: Unable to open voltage input file. Cape loaded?\n");
        printf("        Check /boot/uEnv.txt for correct options.\n");
        exit(-1);
    }

    // Get reading
    int a2dReading = 0;
    int itemsRead = fscanf(f, "%d", &a2dReading);
    if (itemsRead <= 0) {
        printf("ERROR: Unable to read values from voltage input file.\n");
        exit(-1);
    }

    // Close file
    fclose(f);

    return a2dReading;
}

int main()
{
    while (true) {
        int reading = getVoltage0Reading();
        double voltage = ((double)reading / A2D_MAX_READING) * A2D_VOLTAGE_REF_V;
        printf("Value %5d ==> %5.2fV\n", reading, voltage);
    }
    return 0;
}
```

Compile with:

```
arm-linux-gnueabihf-gcc -std=c99 -D _POSIX_C_SOURCE=200809L potDriver.c -o potDriver
```