Using the Beaglebone Black Programmable Real-Time Unit with the RemoteProc and Remote Messaging Framework to Capture and Play Data from an ADC

Gregory Raven

October 2016

Contents

Chapter 1

Introduction

This is the documentation for a small embedded GNU/Linux project utilizing the RemoteProc and RPMsg framework in the Beaglebone Green (BBG) development board.

The inspiration for this project came from the superb book "Exploring Beaglebone" by Derek Molloy. Professor Molloy's project utilized assembly code and the UIO driver. The hardware used in this project is essentially a copy of the Molloy design.

Recent developments in the Texas Instruments PRU support include the RemoteProc and Remote Messaging frameworks, as well as an extensively documented C compiler and additional supporting documentation. This projects utilizes these frameworks and is entire dependent upon C code in both the PRU and GNU/Linux user space. The detailed examples provided by TI in the "PRU Support Package" were invaluable in developing this project:

https://git.ti.com/pru-software-support-package

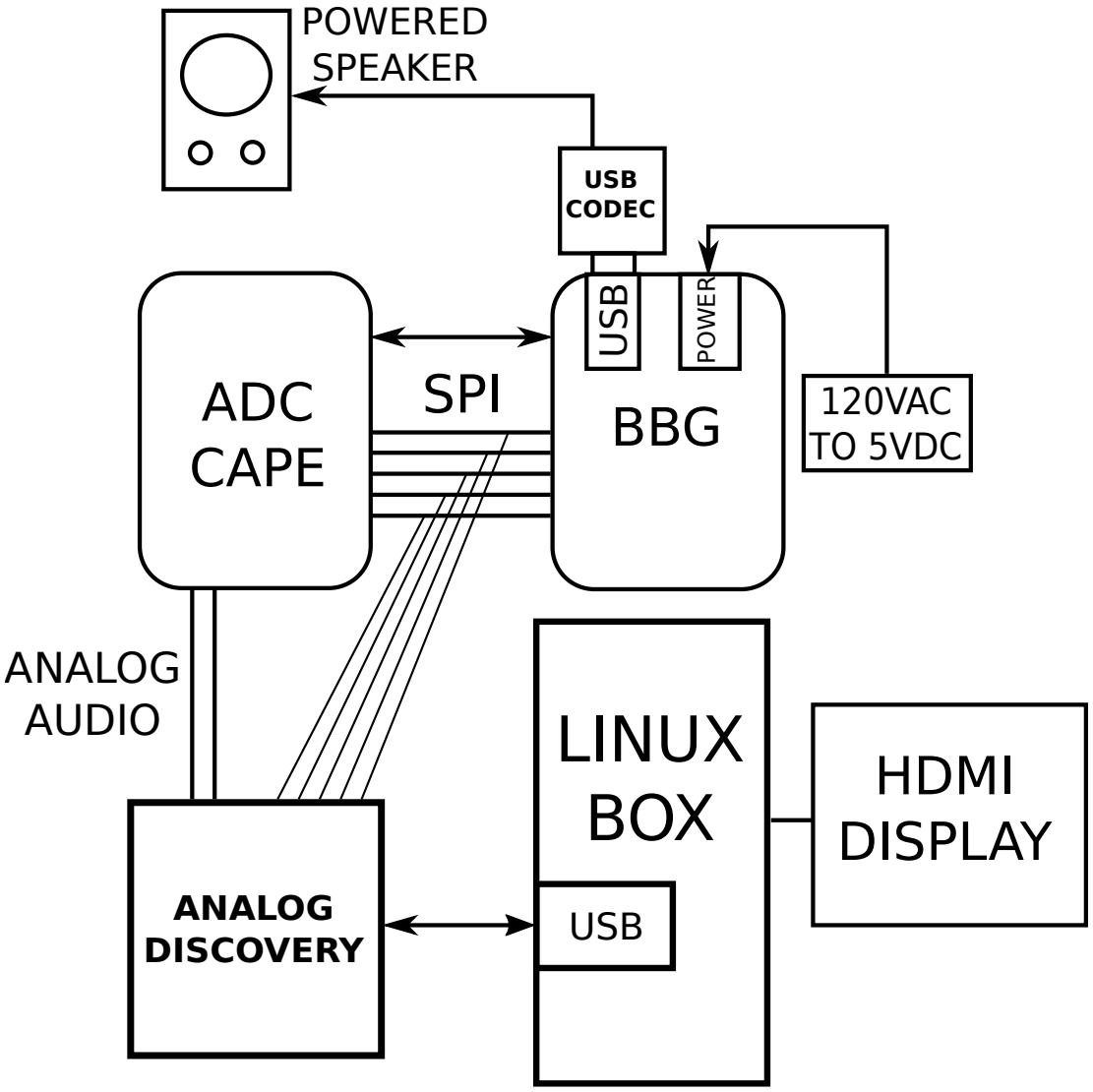
A listing of additional resources is found in the appendix.

An MCP3008 Analog-to-Digital Converter (ADC) IC is connected to the BBG GPIO pins which are in turn connected to the PRU using the "Universal IO" kernel driver which is deployed by default to the current distributions of Debian on the Beaglebones.

The Digilent Analog Discovery 2 was used to analyze and debug the SPI bus between the ADC and the BBG/PRU. This was an indispensable tool during the development process.

Chapter 2

System Diagram



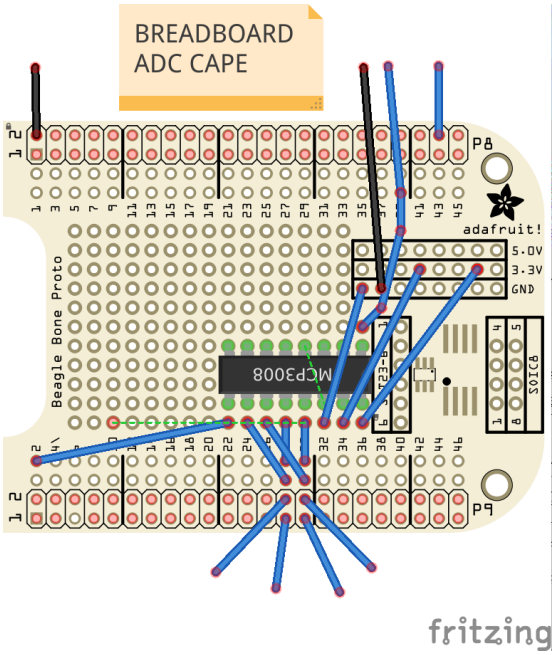
Chapter 3

Prototype Cape Detail

The “Cape” was built on an Adafruit proto-cape:

https://www.adafruit.com/products/572

Here is a breadboard diagram from Fritzing which shows how the proto-cape is wired:



Chapter 4

PRU Firmware

Implementing the SPI Bus in C
    Implementing the Timing Clock in C
    The User-Space Program

Chapter 5

Incorporating Advanced Linux Sound Architecture or “ALSA”

The system generates a stream of audio data samples at a rate of 8 kHz. This data could be stored to memory and later manipulated. Derek Molloy’s project used “GNU Plot” to graph data captured by the ADC-PRU system and stored to the Beaglebone’s memory (RAM).

One of the primary goals of this project was to investigate real-time data "streaming". So rather than a static capture, it was decided that the data stream would be somehow extracted from the system and “played” to an analog speaker.

The primary sound system in GNU/Linux is called the “Advanced Linux Sound Architecture”. This system is very mature and flexible, and also it has a very complex Applications Programming Interface (API).

Fortunately the ALSA system includes command line utilities which made meeting the project goals very simple!

5.0.1 Pulse Code Modulation

What kind of data is delivered by the PRU as it reads the ADC via SPI bus?

The ADC samples data with a “resolution” of 10 bits. What this means is that the analog input of the ADC is “quantized” into 2^{10} or 1024 slices.

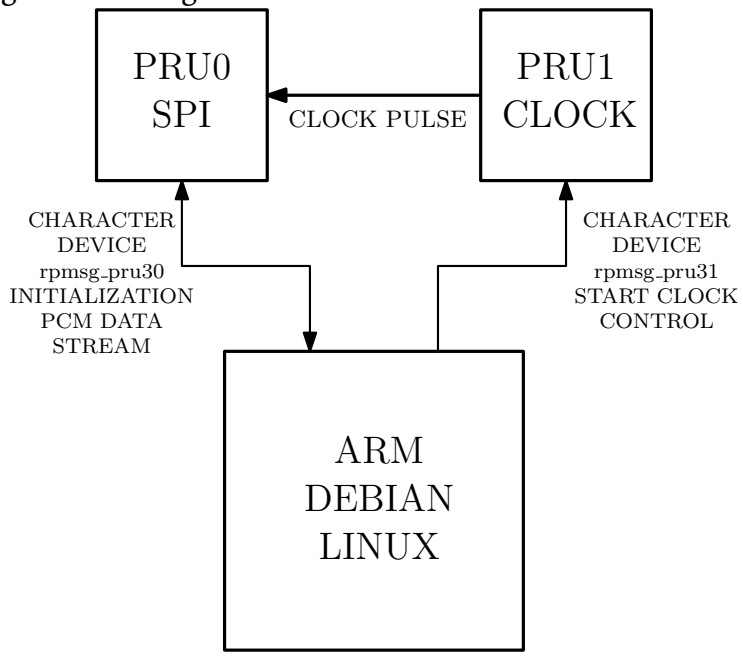
The ADC is capable of handling an input range from a little bit above 0 Volts to a little bit below 3.3 Volts. So a good approximation is a 3.0 Volts range. Splitting this into 1024 slices yields:

$$\frac{\text{3.0 Volts}}{1024} = \text{2.9 millivolts per step}$$

Chapter 6

RemoteProc Framework Messaging

Diagram showing Character Drivers Data Flow from PRU to ARM



Control of the PRU Clock Using Character Driver

Chapter 7

Universal IO and Connecting PRU to the Outside World

This project did not require a custom "Device Tree Overlay". Instead, the "Universal IO" driver was used along with a simple shell script.

The Universal IO project is located at this Github project:
https://github.com/cdsteinkuehler/beaglebone-universal-io
The configuration is as follows:

```
config-pin P8.44 pruout
config-pin P9.31 pruout
config-pin P9.27 pruout
config-pin P9.29 pruout
config-pin P9.28 pruin
config-pin P9.30 pruout
```

The above can also be put into a file, for example "pru-config":

```
P8.44 pruout
P9.31 pruout
P9.27 pruout
P9.29 pruout
P9.28 pruin
P9.30 pruout
```

The command to load the above would be:
config-pin -f pru-config
Note that another step required is create the $SLOTS environment variable and also to set on of the Universal-IO device trees as follows:

```
export SLOTS=/sys/devices/platform/bone_capemgr
echo univ-emmc > $SLOTS/slots
```

7.0.1 The PRU GPIO Spreadsheet

Use “git clone” to download this repository:

https://github.com/selsinork/beaglebone-black-pinmux

The spreadsheet file contained in this repository is pinmux.ods. The LibreOffice suite has a spreadsheet application which will read this file.

This spreadsheet is extremely useful when configuring the PRU or other functions to the Beaglebone pin multiplexer.