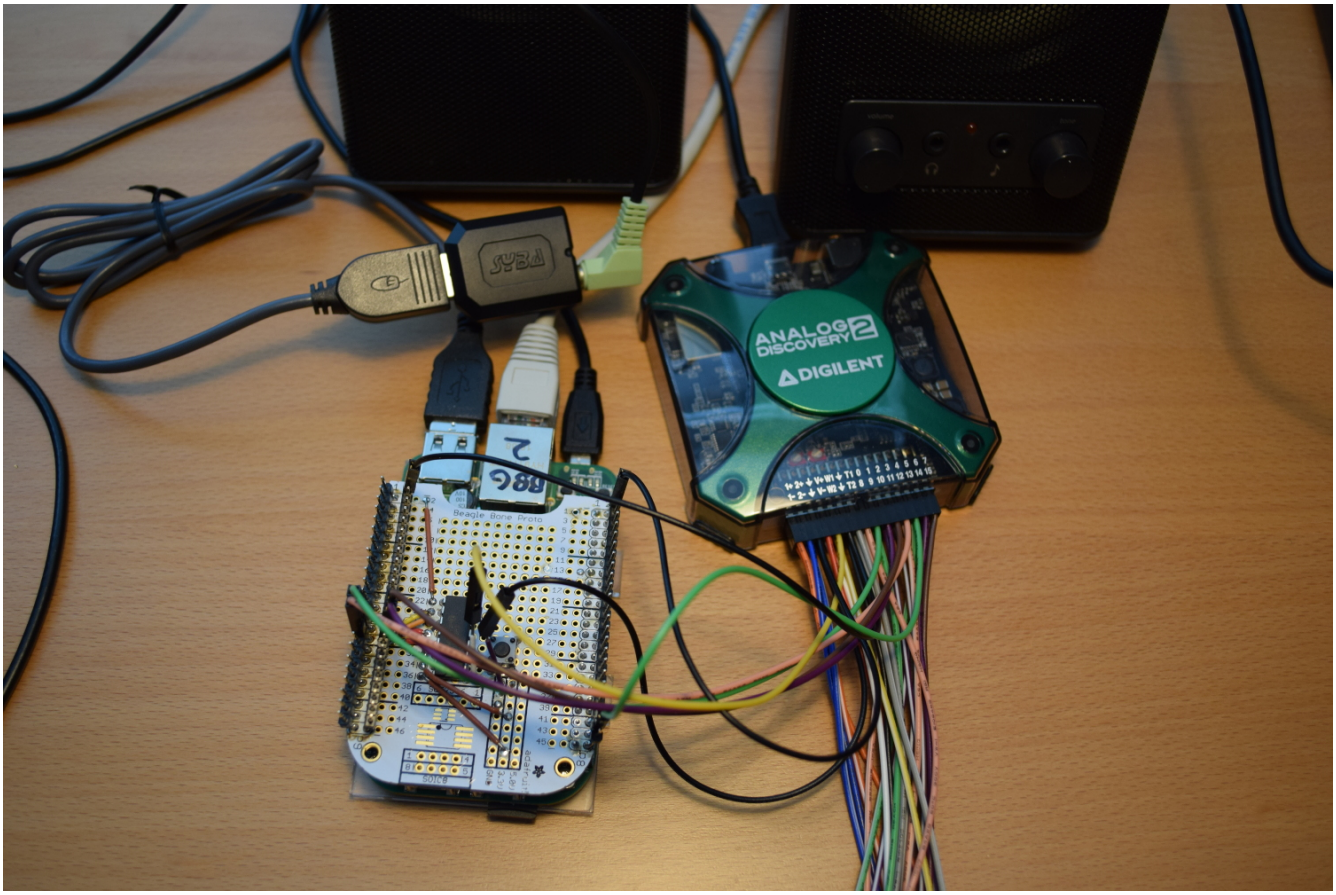

Using the Beaglebone Black Programmable Real-Time Unit with the RemoteProc and Remote Messaging Framework to Capture and Play Data from an ADC

Gregory Raven

October 15, 2016

Contents

List of Tables

List of Figures

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 project repository is located here:

https://github.com/Greg-R/pruadc1

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 much additional supporting documentation. This project utilizes these frameworks and is entirely 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 Resources chapter.

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. The Analog Discovery 2 was an indispensable tool during the development process.

1.1 Project Goals

This project does not solve a specific practical problem. It is a laboratory experiment.

The primary goal is to learn to program the PRUs, and to control and communicate with them from the operating system's "user space". Here is a list of the project goals and sub-goals:

- Write C code for the PRUs which implement a SPI bus.
- Build a "proto-cape" with an ADC.
- Use the "Universal IO" to configure the PRU interface to the GPIO pins.
- Write a user-space C program which communicates with the PRUs via character devices.

- Use the Analog Discovery 2 as a logic analyzer to debug the PRU SPI C code.
- Stream the digitized audio data from the ADC to the "Advanced Linux Sound Architecture" in real time. Play the audio to a speaker.
- Document and publish the project to Github.

1.2 Limitations

The BeagleBone's Sitara "System On Chip" is quite powerful, and this power is probably masking several inefficiencies in the project's design. All of the testing and debugging was done with a bare-bones Debian distribution with no other significant processes running.

The speaker audio does not have noticeable distortion or glitches when listening with the speaker. The audio was not examined with a spectrum or distortion analyzer. It may not be the best quality audio.

The audio sample rate was limited to 8 kHz, which is the lowest rate accepted by ALSA. A follow-up investigation will see if the sample rate can be increased. It is not known what aspect of the system will begin to break down as sample rate is increased.

The sample rate had to be "tuned" to prevent "buffer underruns" reported by the ALSA system. An approach to make the real-time data stream robust is not known by the author and needs further investigation.

Chapter 2

System Diagram

The system diagram as shown includes all of the facilities used during development.

The “Linux Box” is a desktop PC architecture machine running Ubuntu 14.04/16.04. Communication to the BBG was done via “Secure Shell” (SSH). The Linux Box also served as host for the Analog Discovery 2 and GUI via the HDMI display. The Analog Discovery 2 also provided analog audio to the ADC input.

The “Beagle Bone Green” (BBG) is the TI Sitara-based platform board manufactured by Seeed Studio.

The “ADC Cape” is an Adafruit breadboard with the MCP3008 and headers soldered to it. A few wires are required to complete the connections to the ADC to the header pins, DC bias and ground.

The “USB Codec” plugs into the USB connector on the BBG. Due to interference with the adjacent ethernet connector, a short USB extension cable is recommended.

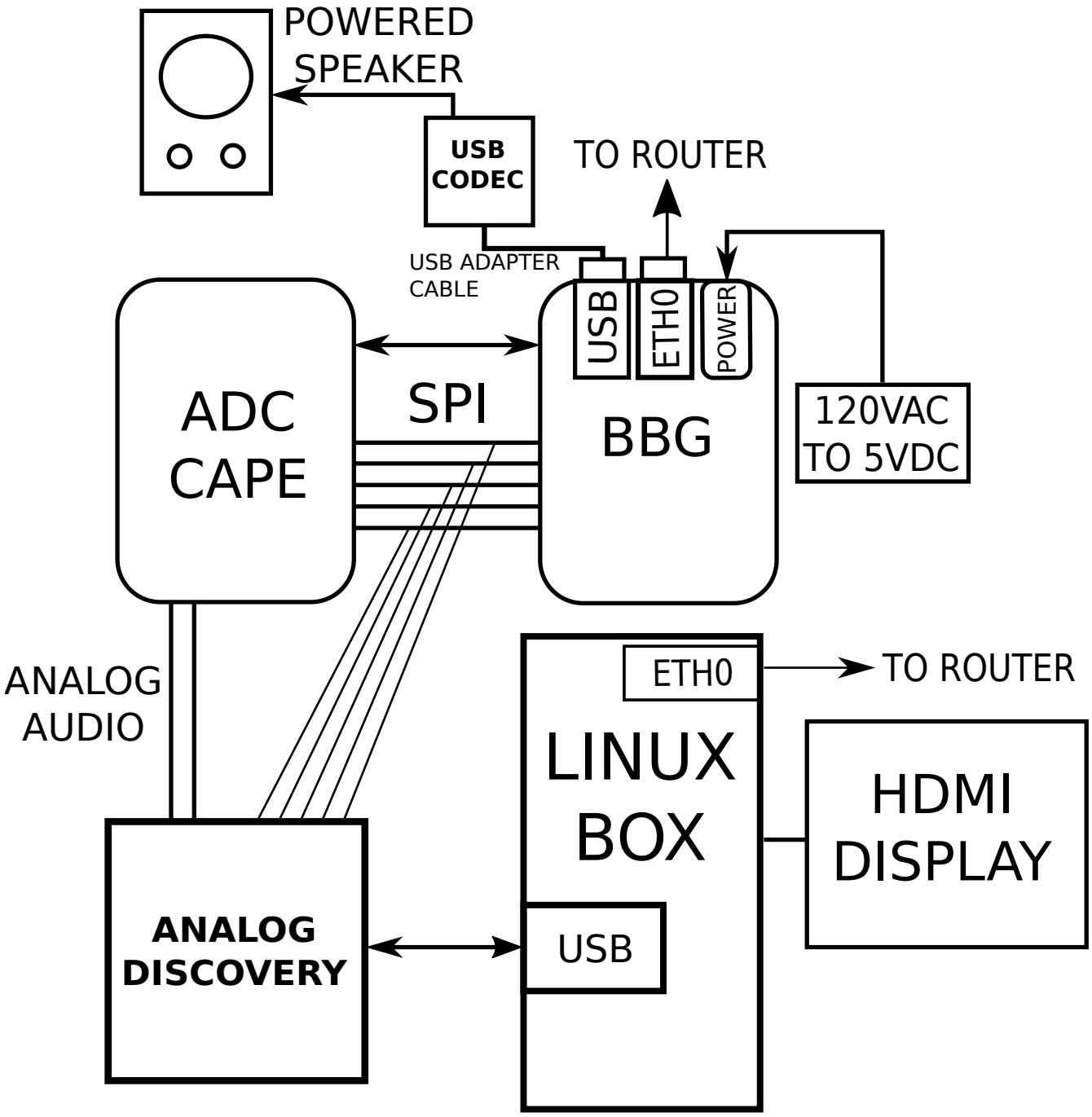

Figure 2.1: PRU-ADC System Diagram

Chapter 3

Prototype Cape Detail

The proto-cape schematic is simple, and is composed of a single component, the MCP3008 ADC.

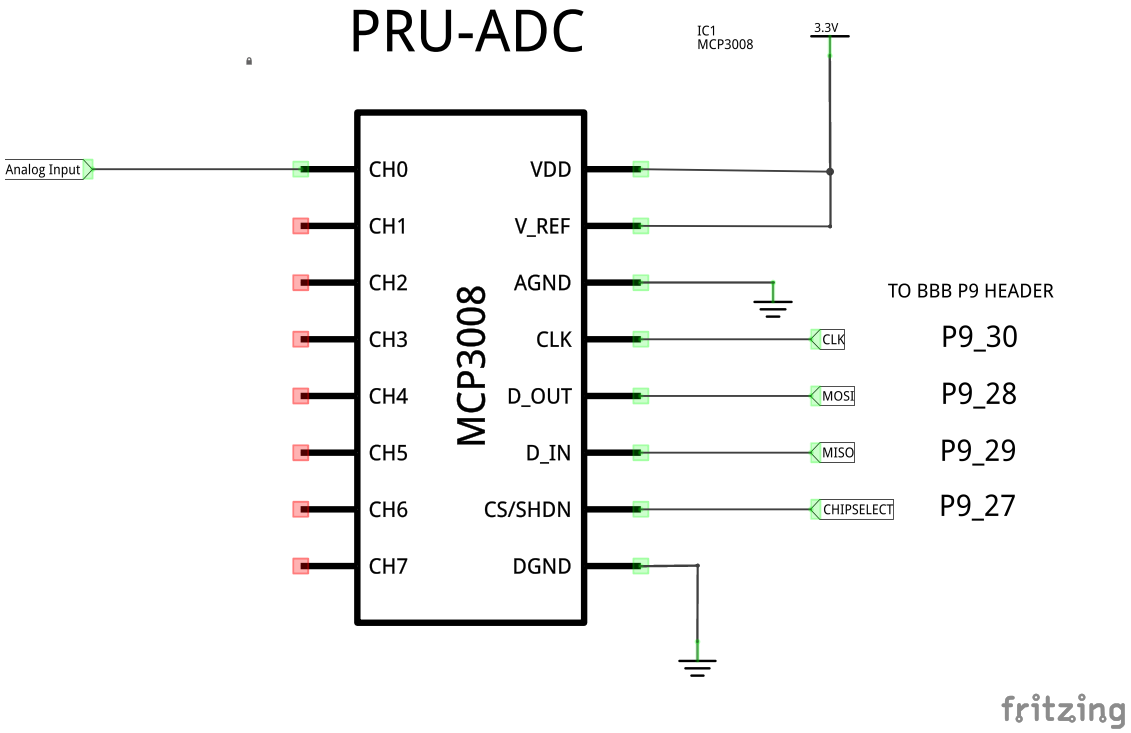

Figure 3.1: PRU-ADC Cape Breadboard

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:
In addition to the header pins required to plug into the BBB, extra rows of headers were soldered to the top of the breadboard. This allows easy connection to the Discovery Analog 2 as seen in the following photograph.

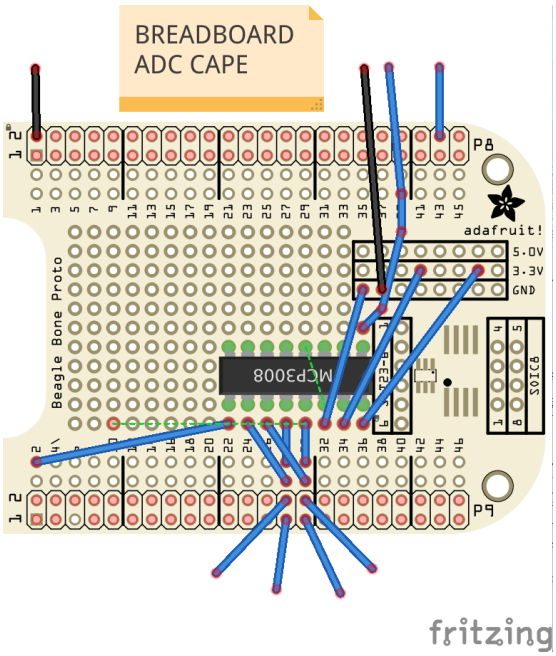

Figure 3.2: PRU-ADC Cape Breadboard

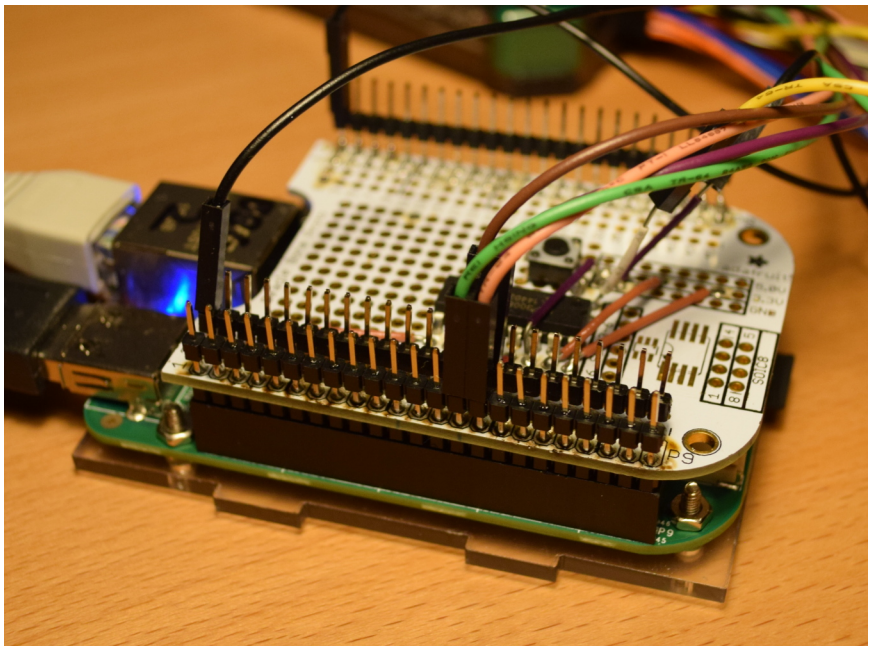

Figure 3.3: PRU-ADC Cape Breadboard

Chapter 4

PRU Firmware

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

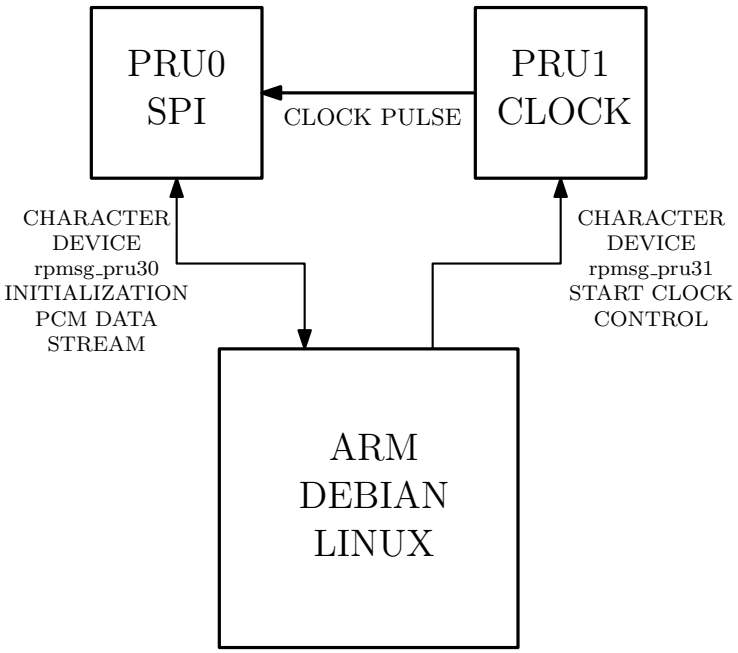

Figure 6.1: PRU<->ARM Character Devices

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.

Chapter 8

User Space Program: Fork and Named Pipe

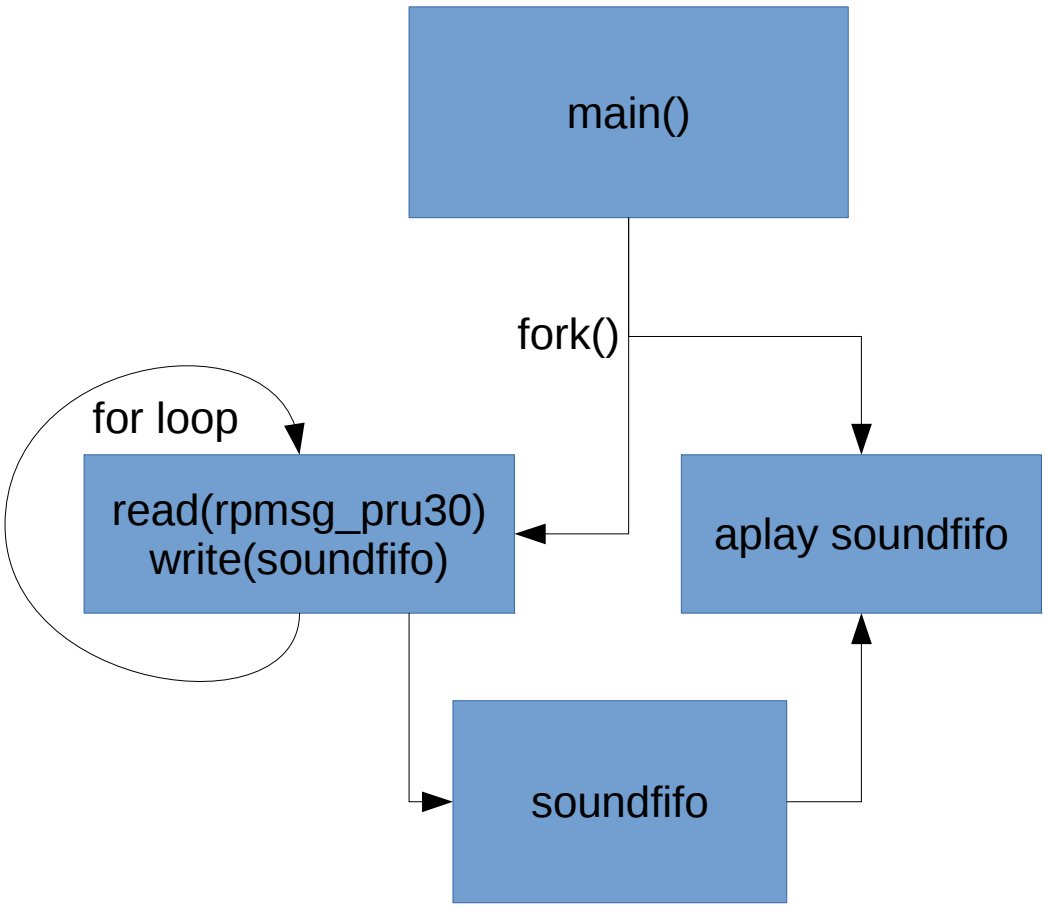

Figure 8.1: Data Flow in User Space Program

Chapter 9

Shell Scripts

Chapter 10

Setting up the PRU Compiler on the Beaglebone Green

The following describes the simplest possible set-up. Everything was done via the command line, and the vim editor was used extensively to develop the C code and shell scripts.

SSH was used to remotely access the BBG from a 64 bit desktop computer running Ubuntu 14.04.

For reference, here is the link to the TI PRU support package:

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

The above package can be cloned to the BBG. There is a good set of examples and labs included. The labs are documented here:

http://processors.wiki.ti.com/index.php/PRU_Training:_Hands-on_Labs

Note that the files appropriate for the BBG are in the folders with name am335x.

The Makefiles in the labs and examples were designed to work with a particular set-up which can be easily implemented on the BBG.

How much is currently stock on the official images?

The following is a list of recommended steps to prepare a BBG for compiling PRU C files.

1. Flash IOT image to micro-sd. 2. Insert micro-sd into BBG slot, press boot and power buttons and release. 3. ssh root@192.168.1.7 4. uname -r to verify kernel -> 4.4.9-ti-r25 YES, it is the correct kernel. 5. apt-get update 6. cd / and then find . -name cgt-pru, and the path is /usr/share/ti/cgt-pru. This is the location of the PRU library and includes. However, the clpru compiler binary is not there: which clpru /usr/bin/clpru So the compiler binary is in a different location. This is a problem for the labs make files. cd /usr/share/ti/cgt-pru mkdir bin cd bin ln -s /usr/bin/clpru clpru So now the make files will find the compiler executable in the correct location via the link. 7. cd /home/debian git clone git://git.ti.com/pru-software-support-package/pru-software-support-package.git This will clone a copy of the latest pru support package. 8. cd into lab_5 in the package: cd lab_5/solution/PRU_Halt make This will fail, it is looking for environment variable $PRU_CGT export PRU_CGT=/usr/share/ti/cgt-pru Now try make again. It should succeed. 9. cd gen cp PRU_Halt.out am335x-pru0-fw cp am335x-pru0-fw /lib/firmware 10. Now cd into the PRU_RPMsg_Echo_Interrupt1 directory in the same lab_5. Edit main.c as follows: //#define CHAN_NAME "rpmsg-client-sample" #define CHAN_NAME "rpmsg-pru" 11. Now almost the same as #9, this time for pru1: cd gen cp PRU_RPMsg_Echo_Interrupt1.out am335x-pru1-fw cp am335x-pru1-fw /lib/firmware 12. Reboot 13. cd /dev look for rpmsg_pru31 device file. It will be there!

Chapter 11

Using the Analog Discovery 2

This table shows the Discovery 2 to PRU-ADC cape connections.

Table 11.1: Analog Discovery 2 Wiring

Logic Wire Number	Color	SPI	BBG Header
1	Green	Chip Select	P9.27
2	Purple	Clock	P9.30
3	Brown	MISO	P9.28
4	Pink	MOSI	P9.29
5	Green	PRU1 Clock	MOVE!

The repository includes a set-up file for the Analog Discovery 2 in the “discovery2” directory. The Logic Analyzer will appear as in the image below.

The Analog Discovery also includes an audio waveform generator, and it will appear in the GUI as shown in this image:

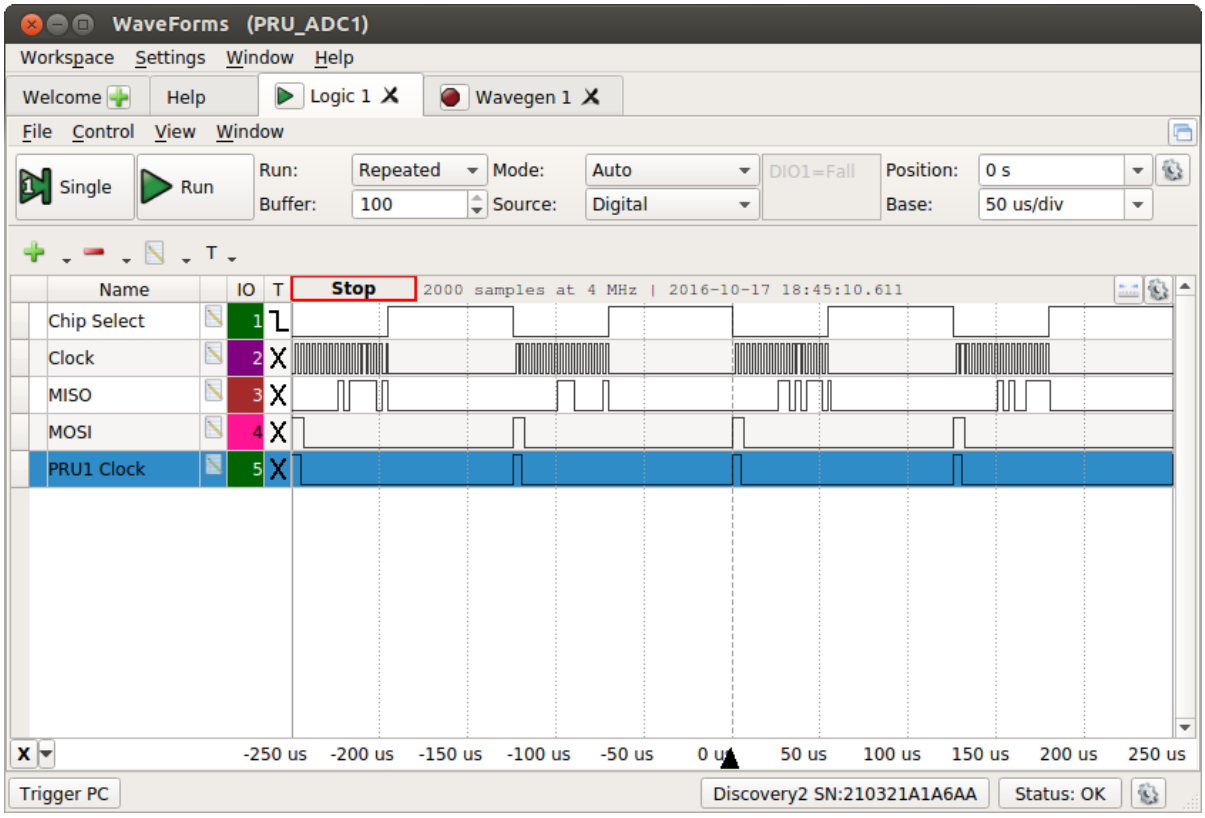

Figure 11.1: PRU-ADC System Diagram

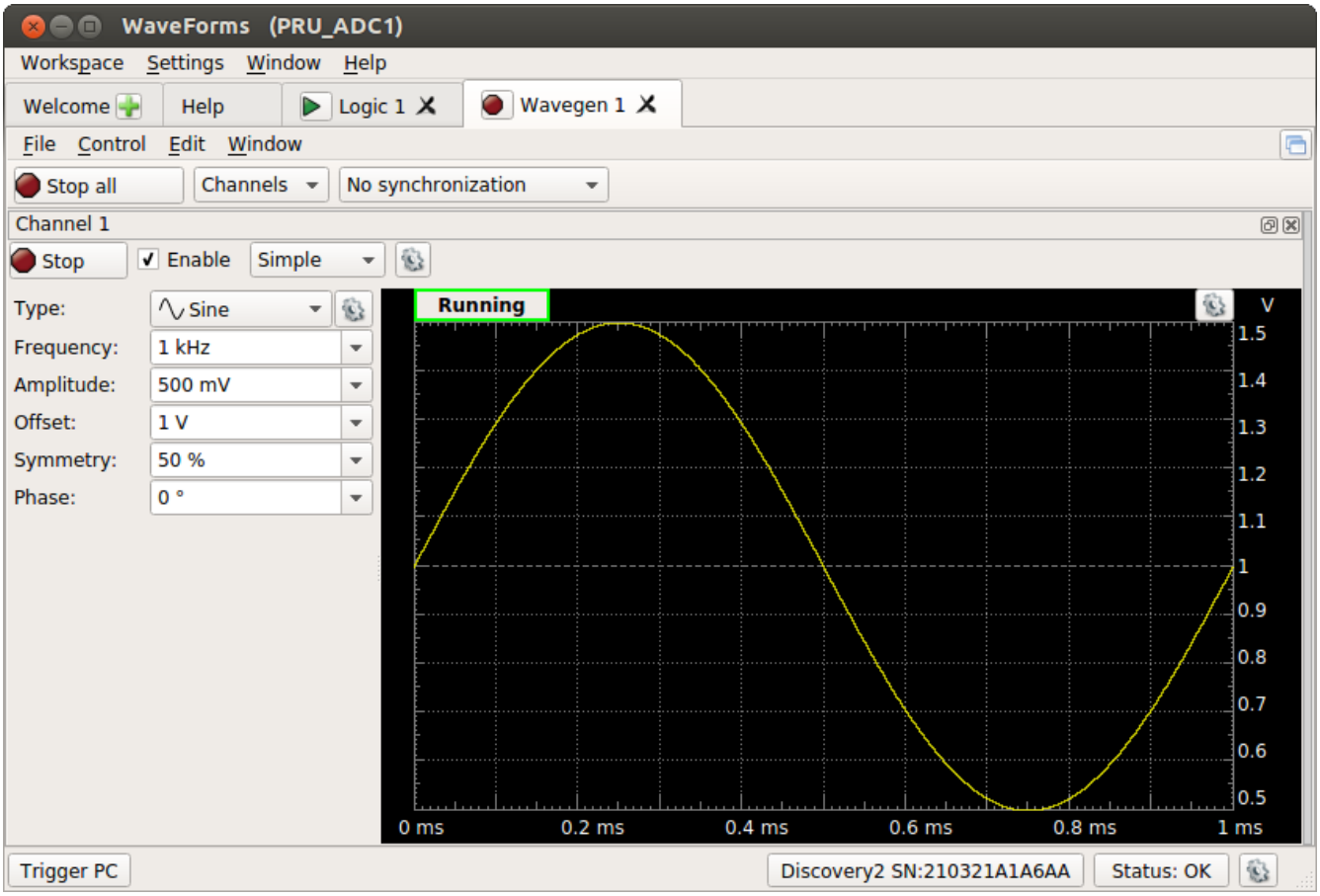

Figure 11.2: PRU-ADC System Diagram

Chapter 12

Resources

Github repository for this project:
https://github.com/Greg-R/pruadc1
http://software-dl.ti.com/codegen/non-esd/downloads/download.htm#PRU
Beagle Bone Green:
https://www.seeedstudio.com/SeeedStudio-BeagleBone-Green-p-2504.html
The Remoteproc Framework and Remote Messaging:
http://processors.wiki.ti.com/index.php/PRU-ICSS_Remoteproc_and_RPMsg
The Analog Discovery 2 by Digilent:
http://store.digilentinc.com/analog-discovery-2-100msps-usb-oscilloscope-logic-analyzer
The USB Codec:
https://www.amazon.com/gp/product/B001MSS6CS/ref=oh_aui_search_detailpage?ie=UTF8&psc=1
Adafruit Beaglebone breadboard cape:
https://www.adafruit.com/products/572