ADC-001 Instructions

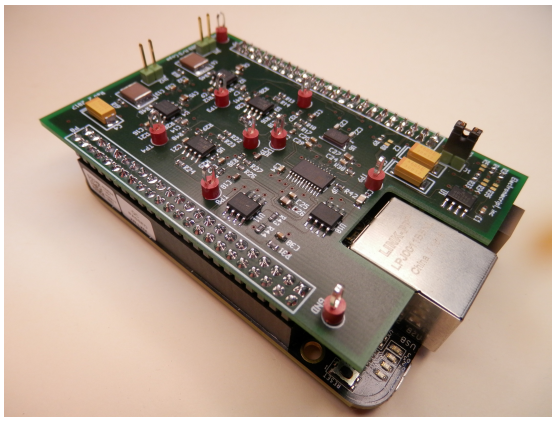
Last update: 6.22.2017, SDB.

The ADC-001 is an analog-to-digital converter designed as a cape for the Beaglebone Black (BBB). The converter is targeted as an experimenter's platform for audio signal processing. It is designed for use with electret microphones. Two ready-to-use microphones are included with the product.

The ADC-001 provides a hardware and software environment you can leverage for experimentation with audio signals. However, it is not a beginner kit – these instructions assume you already have some familiarity with the Beaglebone, both as a user and a programmer. It also assumes you are comfortable with electronics. If you are new to the Beaglebone, this book and website by Derek Molloy provide a good introduction to the platform: http://derekmolloy.ie/beaglebone/.

Hardware instructions

Basic documentation regarding the ADC-001 cape is kept at GitHub under https://github.com/brorson/ADC-001_hardware_information. Users should consult that GitHub page for the latest schematics and hardware information.



ADC-001 shown mounted on Beaglebone Black. In the foreground is the cape's cut-out around the BBB's Ethernet connector. In the background are the two microphone connectors, CONN1 and CONN2.

Install the ADC-001 on your BBB as shown in the photo on left. The cut-out on the cape fits around the Ethernet connector on the BBB. Once the cape is installed, it is recommend to power the BBB and cape using an external 5V supply through the power connector on the BBB. A suggested 5V power supply is available from Adafruit Industries at https://www.adafruit.com/product/276.

Two electret microphones are included with the ADC-001. The microphones are PUI Audio model POW-1644L-B-LW100-R (sensitivity=44dB, SNR=58dB, frequency range= 50Hz – 16KHz). The microphones are enclosed in an electromagnetic shield and are connected to the ADC-001 cape via a shielded cable. The cables are connected to connectors CONN3 and CONN4 on the board.

Software installation

The following repository holds beginner code which you can build and run on your Beaglebone Black to run the ADC-001: https://github.com/brorson/ADC-001_basic_code. Included in that repo are:

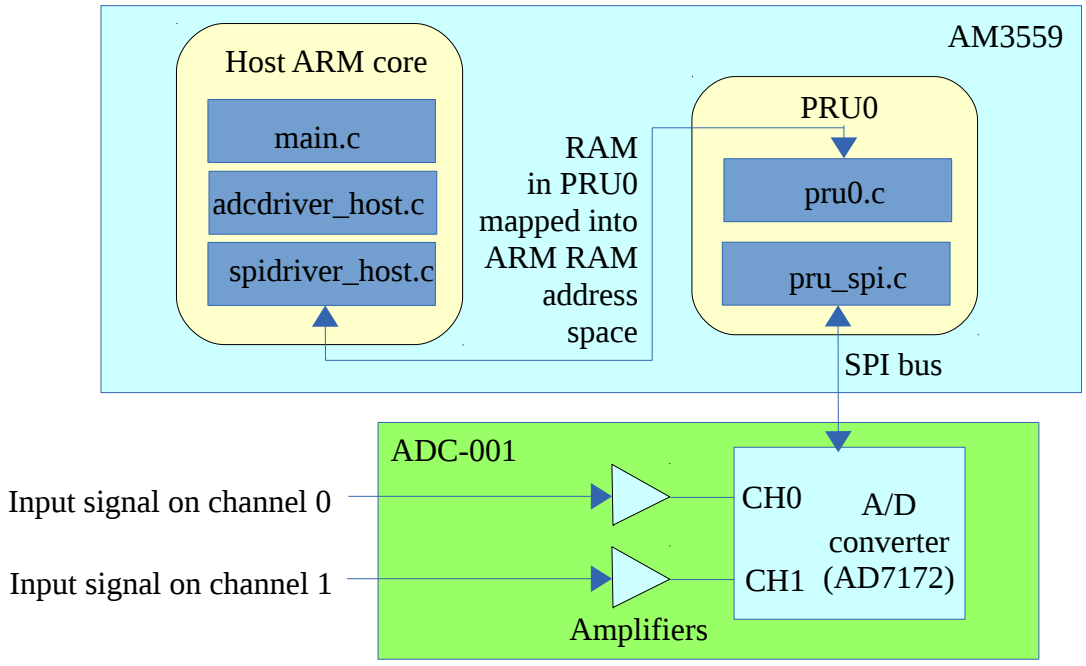
- Example main data acquisition program.
- ADC-001 driver programs.
- C programs running on the PRU which mediate data acquisition from the cape's A/D converter via SPI bus.
- Device tree overlay file for the ADC-001 cape.
- A directory holding include files used in the build process.

The code has been developed and built using Debian 7.9 2015-11-12 (Linux kernel 3.8.13-bone79) running on the BBB. Due to the rapid pace of development on the BBB platform, the location of various

system files can change from one release to the next. The code held in the repo makes some assumptions about the BBB's directory layout (e.g. device tree overlay files live in /sys/firmware). Therefore, it is strongly recommended that you reflash your BBB to use Debian 7.9 if you wish to follow the instructions here. Instructions explaining how to reflash a BBB and configure it for use with the ADC-001 are given in the appendix.

To get the beginner code running on your BBB, please do the following:

1. Install the the ADC-001 cape onto your BBB.
2. Get the basic code from https://github.com/brorson/ADC-001_basic_code.
3. Either clone the repo or download the code as a .zip file from the GitHub site, and install it into a new directory on your BBB.
4. Do a build by issuing the command “make” in your code directory on the BBB. This will create the following important files:
    - The main executable called, naturally enough, “main”. Then executed, this program runs through a series of tests, and exercises the AD7172 by using it to take data in a couple of different acquisition modes (e.g. single-point read, buffer read at different sample rates).
    - Binary images which will run the bit-banging SPI code on PRU0. These are called “text0.bin” and “data0.bin”.
    - The device tree overlay which tells the Linux kernel details about the GPIO configuration used for communication with the AD7172. The device tree overlay image is called SDB-PRU-ADC-00A0.dtbo. The Makefile will copy this overlay file to /sys/firmware.
4. Sometimes, the uio system does not come up correctly after system boot. This is a Linux bug. You can check the status of the uio system by issuing the command “ls /sys/class/uio/”. If the directory listing emitted is empty, then run the script “fix_uio.sh” (included in the files you downloaded). The script stops and starts the uio kernel module several times in order to get it running.
5. Run the main executable by issuing the command “./main” at the command prompt. If everything is working, the program will run through a series of tests, and print to the console its activities.



Software notes

A conceptual block diagram of an application using the A/D shield is shown above. The host ARM microprocessor (AM3359) is shown on top. The A/D shield, ADC-001 is shown beneath the ARM. The A/D itself is an Analog Devices AD7172 configured to support two input channels. Communication between the AM3359 and the A/D takes place via a SPI bus.

Communication between the main program running on the ARM and SPI program running on the PRU occurs via the UIO subsystem. Communication relies on the PRUSS subsystem developed by Texas Instruments which provides access to PRU memory to the ARM via direct read/write. More information about the overall purpose and goals of the UIO subsystem are presented at https://www.osadl.org/fileadmin/dam/interface/docbook/howtos/uio-howto.pdf. Specifics related to Texas Instrument's PRUSS system are available at http://processors.wiki.ti.com/index.php/PRU-ICSS.

Appendix: Reflashing your Beaglebone Black for use with the ADC-001

The code running the ADC-001 assumes you are using Debian 7.9 2015-11-12. It may fail on other Debian releases due to version skew. Use the following steps to reflash your Beaglebone for Debian 7.9 2015-11-12 prior to installing the ADC-001 software.

Set-up. The following instructions assume you have a PC (the “host”) connected to the internet. They also assume you can communicate from the host to your BBB via the serial debug port, as well as a TCP/IP connection. Before reflashing your BBB, make sure to remove all capes, including the ADC-001 from the BBB.

1. Use your host computer to download the desired image from https://beagleboard.org/latest-images. The image to get is “Debian 7.9 2015-11-12”.
2. On the host computer, use the command “`xz`” it to uncompress the image. This should leave an .img file in your working directory.
3. Now copy the image file to a microSD card. Note that the image contains both the boot and Debian partitions, so don't do any formatting of the microSD. Just copy the image onto the microSD card using the command:

   ```
   dd bs=4M if=bone-debian-7.9-lxde-4gb-armhf-2015-11-12-4gb.img of=/dev/mmcblk0
   ```
4. Hook up serial debug cable to debug header on BBB. Start “minicom”, “screen”, or your preferred terminal program to watch the boot sequence and interact with BBB.
5. Install the microSD into the BBB, then hold down S2 while inserting power to the BBB. Hold the button until LEDs flash, then release it. Let the BBB boot up. Watch the BBB debug spew in your terminal program. This can take many minutes. Once the login prompt appears log in as root, no password required.
6. Configure the BBB to reflash eMMC upon boot. To do so, open the file /boot/uEnv.txt in a text editor and navigate to the bottom line, which reads `#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh.` Delete the "#" at the beginning of this line, then save the file.
7. Reboot by power-cycling. Hold down S2, insert the +5V power, let the board boot, then wait for it to flash eMMC. Reflashing will take many minutes.
8. Once flashed, the unit will power itself down.
9. Remove microSD card, then apply power again and watch reboot. Verify your BBB now runs the

correct Debian release using the command "`uname -a`". You should get the response "`Linux beaglebone 3.8.13-bone79 #1 SMP Tue Oct 13 20:44:55 UTC 2015 armv7l GNU/Linux`".

10. Next change the IP address by editing /etc/network/interfaces. Set your preferred static IP address.
11. Also make sure eth0 is cycled after reboot. This is because eth0 is not always configured correctly upon boot. Create a file called /etc/network/cycle.sh. Put the following lines into it:

```
ifdown eth0
sleep 1
ifup eth0
```

12. Next give the script execute permissions using the command:

```
chmod a+x /etc/network/cycle.sh.
```

13. Now edit the file /etc/network/interfaces, and add this line after the stuff about interface eth0:

```
post-up /etc/network/cycle.sh
```

14. Next remove support for HDMI on output conns. This is required so the ADC-001 cap may be recognized. Edit the file /boot/uEnv.txt and comment in (i.e. remove the "#") the following line:

```
cape_disable=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN
```

15. Reboot.
16. Install the ADC-001 cape onto your BBB.
17. Power up the BBB and copy over the ADC-001 over project directory you downloaded from GitHub to your BBB. Make sure to copy over all subdirectories as well as the main directory.
18. Log into the BBB as root, enter the ADC-001 project directory, and issue the command "`make`". The build should run to a successful completion. The build process will also install the device tree overlay file into the correct place (/lib/firmware).
19. Make sure the .dtbo file ends up in /lib/firmware. Issue the command "`ls /lib/firmware`" and verify the file SDB-PRU-ADC-00A0.dtbo is present in that directory.
20. Now make sure uio system is up and running. Issue the following commands from the ADC-001 project directory:

```
sh fix_uio.sh
# Verify ios are in place:
ls /sys/class/uio/
```

    After "ls", you should find a set of GPIO pins reported in the directory /sys/class/uio/. If the directory is empty, something has gone wrong. In that case, try running fix_uio.sh again.
21. Finally, if everything has worked so far, run the program: "./main". It should run through a set of tests, make some measurements, then finally run a loop, filling a buffer full of measured points once every 1/2 second. Hook up the microphones and speak into them. You should see the values of the measured points vary corresponding to the intensity of the sound waves detected by the microphones.

Designed and Manufactured in the USA by Electroniscript, inc.
PO Box 406
Arlington, MA 02476