A low-cost SoC power measurement platform

Evaluating software impact on power consumtion is not easy. Using a simple multimeter on the power line is not accurate enough because there are big variation in input current. A higher sampling frequency is needed. Also you have to correlate the power data with the various phases of a software (separate initialisation from the computation part for example).

Our idea is to build a low-cost power measurement tool around a cheap microcontroller, in this case from the pic18 family. We use its ADC to get current and voltage sample of the power line at a high enough rate. We also use its GPIOs as input, connected to GPIOs of the SoC being evaluated, to track and report various software state on the SoC.

We choose to use a pic18fj53. Its has a 12 bits ADC which can do up to 80ksp/s, runs at 48Mhz and has a USB interface. Our plan was to use the USB interface to transfer the data to the host.Unfortunably during developement we found that the USB interface, when enabled, would dramatically decrease the ADC's accuracy. So we choose to use the serial port on the PIC side and added an external USB/serial bridge. The electronic gets power from the USB connection, the external power supply being used only for the SoC being monitored.

The requirement are:

- inputs between 0 and 2A, 0 and 20V
- 4 GPIOs to connect with the SoC being evaluated

We came up with the schematic at ?https://www-soc.lip6.fr/trac/soc-conso/export/1/hardware/pic_conso.sch.pdf. Hardware and software sources are available in the source section of the wiki, or via svn at ?https://www-soc.lip6.fr/svn/soc-conso/.

A 1.235V precision diode is used as a voltage reference for the ADC. The input voltage is feed to ADC input 0 (AN0) via a voltage diviser. To cover the input range with enough accuracy 3 different values can be selected with a jumper.

3 Shut resistors are used for current measurements, selectable with a jumper. An amplifier will multiply the shut's voltage before feeding it to AN1.

RB0-RB3 are used to get 4 GPIO status from the SoC.

Each sample reported to the host is composed of 3 values: GPIO state (integer between 0 and 15), voltage (AN0 value between 0 and 4095) and current (AN1 value between 0 and 95), in the format: *gvvvcccX*, with *g*, *v* and *c* being hexadecimal digits and *X* being the letter X used as separator. With 8 bytes per sample, at 921600 bauds we could report in theory up to 11520 samples/s. To be safe we choose to run at 5000 samples/s (which requires 10000 A/D conversions/s).

calibration and performance

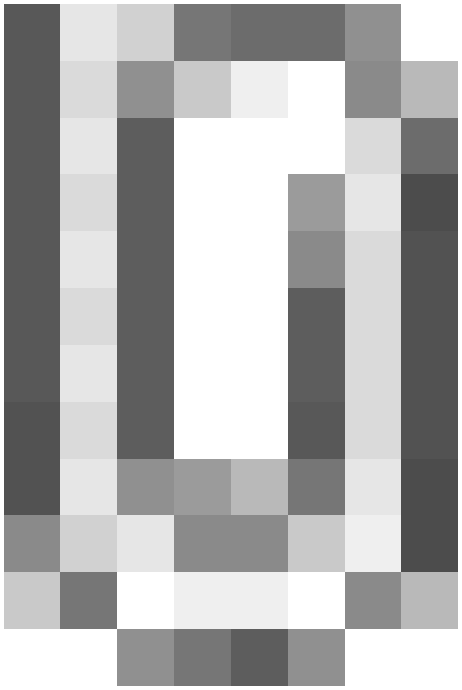
For the calibration process, I used a good linear power supply (both current and voltage adjustable) and a 4000 points digital multimeter. A switching power supply was not good enough, ripple noise impacted values reported by our circuit.

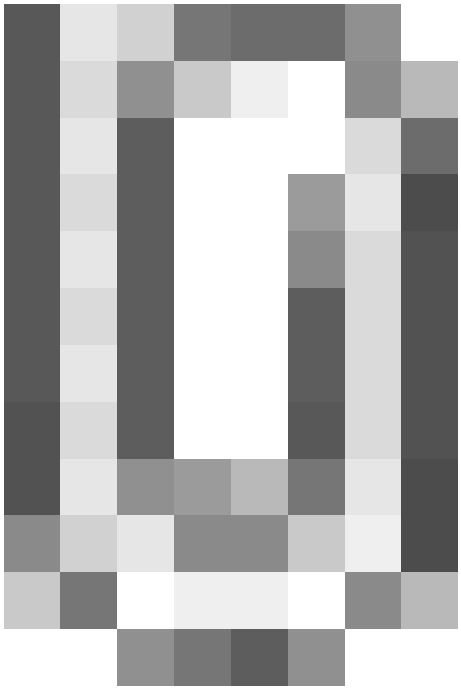
For voltage input, we selectable 3 divider values: 4.92, 11 and 19.33, which gives ranges of 6.07V, 12.35V and 23.88V respectively.

For current input, we have 3 selectable shuts: 0.05, 0.1 and 0.2 ohms. The shunt's volatge is then amplified by 10 by the AD623 amplifier. This gives ranges of 2.45A, 1.23A and 0.61A respectively.

For each tension or current range, we take 8 samples between 0 and a value close to the max of the range. For each sample we get a few tousand values from pic's output, while we note the real value using the digital multimeter.

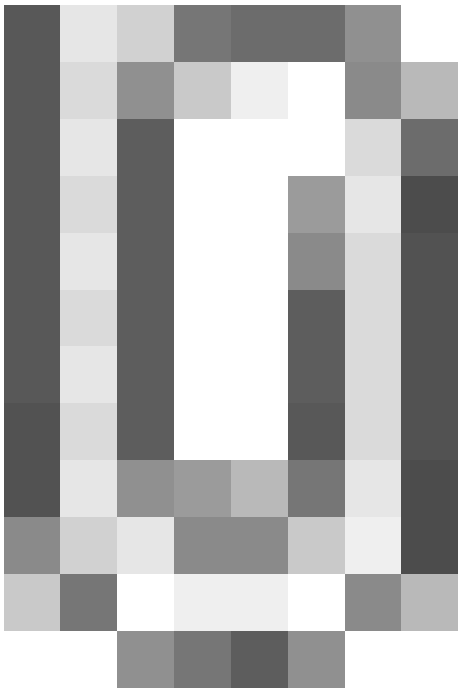
From this process, we found that that the AD623 output can't go down to 0, it stays positive by anout 5.5mV. This means we can't measure currents smaller than 2.75mA on the 0.61A scale, and 11mA on the 2.45A scale. We also find that, assuming an ideal power supply without noise or ripple, we have a +/-7 unit error on the ADC's output for current values. This is less than 0.18% of the full range (or +/- 4.4mA on the 2.45A range input). Below are plots for measures on the 0.61A scale.

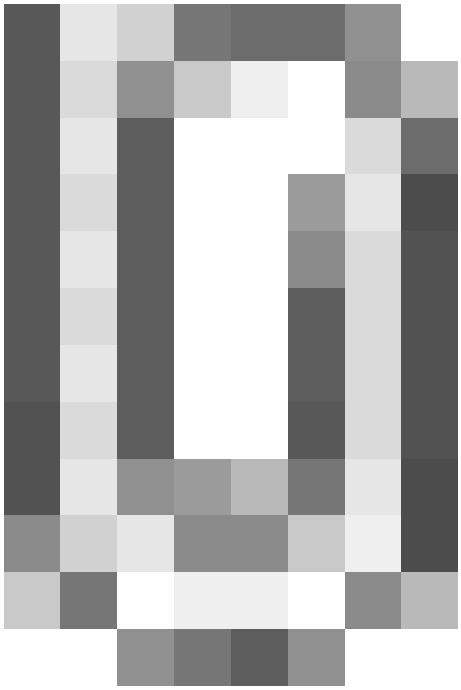




From the 536.7mA measure we have a 8% error for the ADC's output (we expect 3559, we get 3851). We can use this to correct our convertion ratio: adc(i)=3851/536.7 * i (with i in mA). With this conversion function, and excluding the value at 0mA, the error with the measured ADC output is at most 4, which is within the noise.

We repeated the process for the tension inputs. Here we have a +/-5 unit error on the ADC's output for voltage inputs. This is less than 0.13% of the full range (or +/- 29mV on the 23.88V range). Below are plots for measures on the 6V scale.





For 5.996V, we expect 4045 at the ADC output and we did read 4041 0r 4042. This is less than 0.1%. As for current we use the 5.996V result to fix our conversion ration: adv(v)=V*4041.5/5.996. With this function, the error with the measured ADC output is at most 5.

version 2

The version 2 of the hardware got several improvements:

- as the USB part of the pic18 is not used, we switched to a pic18f27j13
- There are now 8 GPIOs available
- a 4th current range (up to 5A) has been added
- The USB/serial part is electrically decoupled from the target board, to avoid possible ground loop affecting current values. The USB/Serial bridge is powered from USB, while the pic18 is powered from the load power supply.
- The pic18 can power on or off the load
- the pic18 controls two solid state relays, which can drive e.g. power or reset switches for the target board.
- the pic18 has one optocoupled input, which can monitor e.g. a power led from the target board.
- a second serial port is available from the pic18, which can be used to connect to a TTL-level serial console on the target board, avoiding an extra serial/USB converter.

The software also got major reworks. To use the new features, a bidirectionnal connection is now needed. We also used this opportunity to store calibration value in the pic18's flash, so we don't need to match a board with its calibration values on the host's side. The USB/serial port of the board runs at 921600 bauds. The board accepts the following commands from the host:

- **B<n>** connect to the target's serial port at speed *n*. If *n* is ommited, use the previously set speed. exit with *#*.
- **C** get calibration data.

- **C<string>** set calibration data string. It is just a string stored in flash and echo'ed back on request; it's up to the host software to interpret it. In order to keep the firmware simple, the string has to be exactly 83 char longs, and can hold only digits, . or spaces.
- **E** causes the firmware to exit, resetting the pic18
- **Mn** with *n* being 1 or 0: start or stop measures.
- **Pn** with *n* being 1 or 0: power on or off the load
- **Rrn** with *r* being 1 or 2 and *n* being 1 or 0: turn on/off relay *r*.
- **S** get various input states

Each command is replied with **OK**, unless and error occured.

Host software getting values from the board now has to start measures with **M1** first; it should also send **M0** before exiting.