Extra PMIC features on Raspberry Pi 4 and Compute Module 4

Raspberry Pi Ltd

2023-02-10: githash: c65fe9c-clean

Colophon

build-date: 2023-02-10
build-version: githash: c65fe9c-clean

Legal Disclaimer Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME ("RESOURCES") ARE PROVIDED BY RASPBERRY PI LTD ("RPL") "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL's Standard Terms. RPL's provision of the RESOURCES does not expand or otherwise modify RPL's Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Document version history

Release	Date	Description
1.0	16 Dec 2022	• Initial release

Scope of document

This document applies to the following Raspberry Pi products:

Pi Zero			Pi 1				Pi 2		Pi 3			Pi 4	Pi 400	CM1	CM3	CM4	Pico
Zero	W	H	A	B	A+	B+	A	B	B	A+	B+	All	All	All	All	All	All
												*	*			*	

Introduction

Raspberry Pi 4 and Raspberry Pi Compute Module 4 devices use a Power Management Integrated Circuit (PMIC) to supply the various voltages required by the various components on the PCB. They also sequence power ups to ensure the devices are started in the correct order.

Over the duration of production of these models, a number of different PMIC devices have been used on both Raspberry Pi 4 and CM4. All the PMICs have provided extra functionality over and above that of voltage supply:

- Two ADC channels that can be used on CM4.
    - On later revisions of Raspberry Pi 4 and Raspberry Pi 400, the ADCs are wired up to the USB-C power connector on CC1 and CC2.
- An on-chip sensor that can be used to monitor the PMIC's temperature, available on Raspberry Pi 4 and CM4.

This document describes how to access these features in software.

WARNING

There is no guarantee that this functionality will be maintained in future versions of the PMIC, so it should be used with caution.

You may wish to also refer to the following documents:

- Raspberry Pi CM4 datasheet: https://datasheets.raspberrypi.com/cm4/cm4-datasheet.pdf
- Raspberry Pi 4 reduced schematics: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

This white paper assumes that the Raspberry Pi is running Raspberry Pi OS, and is fully up to date with the latest firmware and kernels.

Using the features

Originally these features were only available by directly reading registers on the PMIC itself. However, the register addresses vary depending on the PMIC used (and therefore on the board revision), so Raspberry Pi Ltd has provided a revision-agnostic way of getting this information.

This involves using the command line tool `vcgencmd`, which is a program that allows user space applications to access information stored in or accessed from the Raspberry Pi Ltd device's firmware.

The available `vcgencmd` commands are as follows:

Command	Description
`vcgencmd measure_volt usb_pd`	Measures the voltage on the pin marked usb_pd (See CM4 IO schematic). CM4 only.
`vcgencmd measure_volt ain1`	Measures the voltage on the pin marked ain1 (See CM 4 IO schematic). CM4 only.
`vcgencmd measure_temp pmic`	Measures the temperature of the PMIC die. CM4 and Raspberry Pi 4.

All of these commands are run from the Linux command line.

Using the features from program code

It is possible to use these `vcgencmd` commands programmatically if you need the information inside an application. In both Python and C, an OS call can be used to run the command and return the result as a string.

Here is some example Python code that can be used to call the `vcgencmd` command:

```
import subprocess

# call vcgencmd and pass in a command
output = subprocess.check_output(['vcgencmd', 'measure_temp', 'pmic'])

# print the output of the command
print(output)
```

This code uses the Python `subprocess` module to call the `vcgencmd` command and pass in the `measure_temp` command targetting the `pmic`, which will measure the temperature of the PMIC die. The output of the command will be printed to the console.

Here is a similar example in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
  char *cmd = "vcgencmd measure_temp pmic";
  char buffer[128];
  FILE *pipe = popen(cmd, "r");
```

```
  if (!pipe)
  {
    perror("popen");
    return 1;
  }

  while (!feof(pipe))
  {
    if (fgets(buffer, 128, pipe) != NULL)
    {
      printf("%s", buffer);
    }
  }

  pclose(pipe);
  return 0;
}
```

The C code uses `popen` (rather than `system()`, which would also be an option), and is probably a little more verbose than it needs to be because it can handle multiple line results from the call, whereas `vcgencmd` returns only a single line of text.

NOTE

These code extracts are supplied only as examples, and you may need to modify them depending on your specific needs. For example, you may want to parse the output of the `vcgencmd` command to extract the temperature value for later use.

Raspberry Pi