Application Note

Restricted to customers under NDA

GAP9 VOLTAGE CONTROL USING INTERNAL POWER MANAGEMENT UNIT

Release 1.0

November 14th, 2023

DISCLAIMER

This information is subject to change without notice.
Information on this document is provided “as is” without any warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, suitability for a particular purpose, or non-infringement. The information provided in this document is intended for informational purposes only. Information may be changed or updated without notice.

Table of contents

1. INTRODUCTION

This document describes VDD_LOGIC voltage setting using GAP9 Power Management Unit (PMU), including some limitations uncovered on the chip. It provides recommendations and workarounds to circumvent, or at least limit, the impact of these limitations. Please be aware that the proposed solutions may not be the best option every time and should be adapted to the precise context of the use of GAP9.

GAP9 PMU is in charge of executing power-related sequences at reception of a triggering command or hardware event (please refer to GAP9 datasheet for more information on GAP9 power architecture and predefined PMU sequences). The PMU runs on an internally generated clock, that is supposed to last while the PMU is active, then to stop when it is no more needed, in order to save power. Therefore, at boot time, or when a command or hardware event is received, the PMU first start its internal clock, in order to process the required command or sequence. Depending on the executed command or sequence of commands, the PMU might wait for an acknowledgment signal from GAP9 IPs it configures (e.g., a power switch or a voltage regulator), stating that the command has been well received and taken into account.

The identified slowdowns presented in this document are related to this command/acknowledgement mechanism between the PMU and the RAR voltage regulator. When it receives a command from the PMU, the RAR configuration interface applies the new configuration and acknowledges the command. However, it then waits for a delay counter to be completed before going back to its default idle state – this delay being made to avoid timing violations in case two successive configurations are sent to the RAR. In certain cases, the PMU ends its processing and stops its clock before the RAR configuration interface is back in an idle state, because the delay counter is still running.

2. CONTROLLING VOLTAGE ON GAP9

2.1. USE CASES

Use cases for configuring a different voltage in GAP9 are usually related to changes in the frequency: it is desirable to always set the voltage to the minimal required value to achieve the target frequency, in order to save power. A change of voltage can then correspond to a reduction of chip activity (by reducing the frequency and then the voltage), or to a processing speed-up (by increasing voltage and then the frequency).

It is worth noticing that the order between the frequency and the voltage configuration steps is important: for a given voltage, we cannot go above a given frequency. Therefore the new (increased) voltage must be guaranteed before increasing the frequency, whereas one must ensure that the frequency has effectively been reduced before possibly decreasing the voltage. GAP9 datasheet provides the maximum frequency for the various GAP9 clocks, depending on the current voltage.

GAP9 SDK offers the following API to control GAP9 voltage setting :

- `int32_t pi_pmu_voltage_set(pi_pmu_voltage_domain_e domain, uint32_t voltage);`
- `uint32_t pi_pmu_voltage_get(pi_pmu_voltage_domain_e domain);`

2.2. CHIP INTERNAL BEHAVIOR WHEN SETTING THE VOLTAGE

There are two different categories of voltage changes to be taken into consideration here: increasing the voltage, or decreasing it (or setting it again at its current value).

When increasing the voltage on GAP9, the RAR configuration interface waits for a “regulation OK” signal from the regulator before acknowledging the command. Thus, the regulated voltage value is guaranteed to be consistent with the setting when the PMU receives the acknowledgment. This waiting time allows the delay counter mentioned above to continue to run.

When decreasing the voltage on GAP9, however, the regulator output is already above the configured value. Therefore, the PMU receives the acknowledgment signal as soon as the new configuration is applied. The new voltage is therefore properly set, but the configuration interface is not yet back to its default idle state. If the PMU has no more command to process, it then stops the clock, halting the delay counter. Before accepting any new command, the RAR configuration interface will first have to complete the delay count, which requires the PMU clock to be on. This possibly creates slowdowns in some cases described below in this document.

2.3. SLOWDOWN MEASUREMENTS

As explained above, a slowdown occurs for a sequence called after a voltage decrease, for which the PMU has stopped its clock before the end of the delay count. The observed slowdown matches the remaining delay of the counter, which then completes before processing the next voltage setting command.

Here are the three main voltage-related commands that may be called after decreasing the voltage, and that may be impacted:

> SETTING THE VOLTAGE VALUE

By default, GAP9 is starting at 0.8V. To reduce consumption, we can go down to 0.65V. It is not rare to go back to 0.8V (or an intermediate voltage) later, to speed-up some processing.

Setting back the voltage to 0.8V when it has been set to a lower value takes 334µs, to be compared to 95µs to set it from a given value to a lower one.

> WAKING-UP FROM LIGHT SLEEP

When GAP9 goes into light sleep, its voltage goes down to 0.65V automatically – triggering the slowdown issue – and is set back to its previous voltage at wake-up.

In the case where the previous voltage was 0.8V, we observe a start-up sequence time of 360µs.

Voltage (V)	0.65	0.7	0.75	0.8
Wake-up time (µs)	70	360	360	360

Table : GAP9 wake-up time depending on selected voltage when entering light sleep

> WAKING-UP FROM RETENTIVE SLEEP

When GAP9 goes into light sleep, the regulator switches to LDO mode (instead of DC-DC) at 0.65V – which also triggers the slowdown issue. It is set back in DC-DC mode at wake-up, at its previous voltage.

In the case where the previous voltage was 0.8V, we observe a start-up sequence time of 270µs.

Voltage (V)	0.65	0.7	0.75	0.8
Wake-up time (µs)	70	360	360	270

Table : GAP9 wake-up time depending on selected voltage when entering retentive sleep

3. PROPOSED MITIGATION MEASURES

3.1. MANUAL VOLTAGE DECREASE

As seen above, the slowdowns originate from a voltage decrease that occurred beforehand, for which the delay count is not completed. The first workaround is to have the PMU clock running enough cycles to fix this. This can be done through dummy accesses to PMU internal registers, or to always-on registers of the SoC controller peripheral. Note that if the PMU or always-on registers are accessed by the application along its execution, this automatically contributes to completing the delay count.

The following function can be called after a call to the *pi_pmu_voltage_set* function to ensure that the delay count is completed :

C Code:

```
pi_pmu_keep_on(120); // Ensure that no extra delay will occur at the next regulator config
```

3.2. SLEEP USECASES

> LIGHT SLEEP

To reduce GAP9 light sleep wake-up time, it is possible to manually decrease the voltage down to 0.65V before going to sleep. This way, the PMU detects that the voltage is already set to the expected value, and does not reconfigure the regulator. This way, the slowdown does not occur, as the regulator configuration interface is not used. The voltage can later be increased to a higher value.

We observe a wake-up time of 70µs using this workaround.

C Code:

```
pi_pmu_voltage_set(PI_PMU_DOMAIN_CHIP, 650); // Set voltage to 0.65V
pi_pmu_domain_stage_change(PI_PMU_DOMAIN_CHIP,
                           PI_PMU_DOMAIN_STATE_LIGHT_SLEEP, 0); // Go to light sleep
/* Light sleep is occuring */
pi_pmu_voltage_set(PI_PMU_DOMAIN_CHIP, 800); // Go back to 0.8V after wake-up
```

Notes:

This workaround is useful if the wake-up time must be as slow as possible. For instance if you are waking-up from UART and you want to receive bytes as fast as possible after wake-up.

> RETENTIVE SLEEP

No workaround has been identified in this case, as switching back the regulator in DC-DC mode requires the PMU to anyway access its configuration interface.

DOCUMENT HISTORY

Drafts A-B *– October, 2023 (Paul Paillet, Jérôme Martin)*
Initial drafts

Rel.1.0 *– November 14, 2023 (Paul Paillet, Jérôme Martin)*
Initial release