Kneron Inc

Document Name: **Kneron KL520 Power Management**

Kneron KL520 Power Management

Kneron Inc

Engineering Design Document

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to define KL520 Power Management scheme and the API with data structures for managing KL520 power.

1.2 Scope

The scope of this document is about KL520 SCPU, NCPU and system power. API for device driver power management is described but not on specific device detail.

2 Reference

Kneron KL520 Design Specification, Rev. 0.5, Feb. 2019 (Internal)

3 Acronyms, Abbreviations, Definitions

KL520 – Kneron SOC 520
NPU – Kneron Neural Network Processing Unit
NCPU – NPU CPU
SCPU – System CPU
IPC – Inter Processor Communication
ISR – Interrupt Service Routine

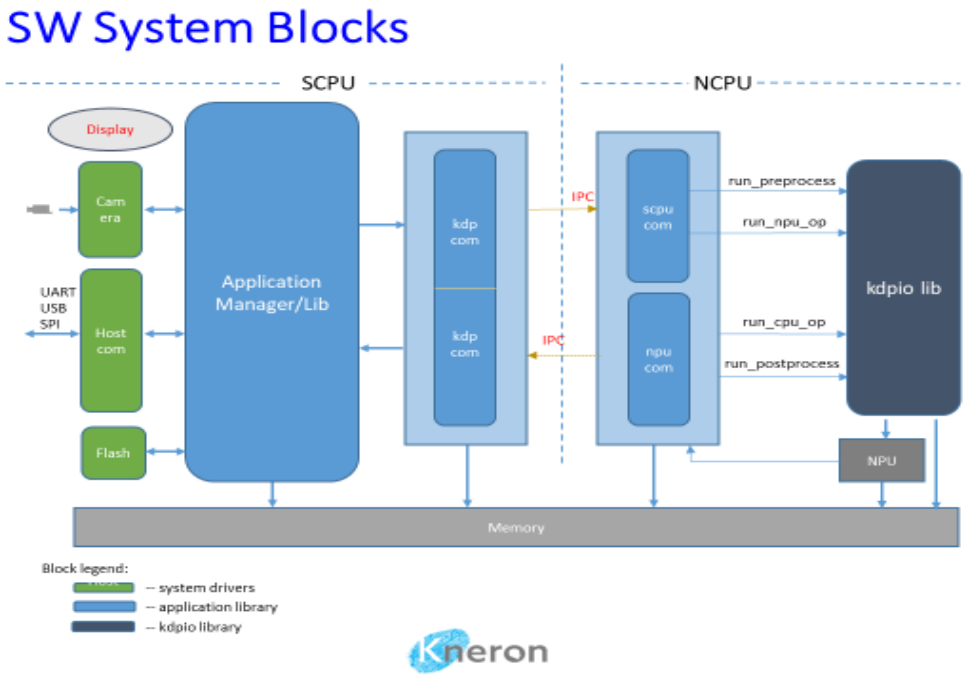
4 Architecture

4.1 System Architecture

KL520 is a system with one NPU, two CPUs, and peripherals. One CPU handles system requests such as host communication, camera video and display, and peripherals. Another CPU assists NPU to do works like input image preprocessing and postprocessing. Two CPUs use shared memory and interrupt for their communication (IPC).

Upon power-on, default power domain will be turned on and the initial bootloader code (IPL) in ROM starts to run on SCPU. Once the secondary bootloader (SPL) is loaded to system internal SRAM by IPL, SPL will load SCPU OS and NCPU OS in SRAM and pass over the execution SCPU OS.

Once NCPU OS is started (by SCPU OS), it will stay in idle thread and listen to commands from SCPU. SCPU will also stay in idle thread and listen to host commands in companion mode or listen to user commands in standalone mode.



4.2 Power Domains

On KL520 there are three switchable power domains and one always-on RTC domain:

- RTC power domain
  RTC module, Power Control Unit, basic PLL

- X_PSW_DEFAULT
  Default power domain:
  PLL, clock, SCPU, UART, and peripherals

- X_PSW_NPU
  NPU power domain:

NPU, NCPU, NiRAM, NdRAM, NMEM, USB, DDR Ctrl,

- X_PSW_DDRCK
  DDR power domain:
  DDR

4.2.1 System Power Modes

The design of power modes is to save as much power as possible while maintaining proper performance level. Based on the available power domains, these power modes are designed on KL520:

- Idle
  - Enter: CPU idle
    - Both CPUs stops at WFI
  - Exit: any interrupt wakeup
- Nap
  - Enter: CPU idle for a period of time (seconds)
    - Both CPUs stops at WFI
    - Clock gating on NCPU and NPU
  - Exit: any interrupt wakeup
- Deep Nap
  - Enter: CPU idle for a period of time (minutes)
    - Both CPUs stops at WFI
    - Clock gating on ncpu and npu
    - DDR self refresh
  - Exit: any interrupt wakeup
- Sleep (Retention)
  - Enter: command from host or user
    - NPU power domain off (X_PSW_NPU off)
    - DDR self refresh
  - Exit: PWR button or RTC alarm
- Deep Sleep (Deep Retention)
  - Enter: command from host or user
    - NPU power domain off (X_PSW_NPU off)
    - Default power domain off (X_PSW_DEFAULT off)
    - DDR self refresh
  - Exit: PWR button or RTC alarm
- Shutdown (RTC mode)
  - Enter: command from host or user
    - DDR power domain off (X_PSW_DDRCK off)
    - NPU power domain off (X_PSW_NPU off)
    - Default power domain off (X_PSW_DEFAULT off)
  - Exit: PWR button

In above power modes, the higher mode (lower one) has the more power saving.

And for power saving purposes, OS should be tickless so that it can enter idle mode as often as possible.

Except NCPU's own Idle mode, all other power modes are initiated and controlled by SCPU.

5 Power Management API

5.1 Power Mode Callback Registration API

5.1.1 Callback Prototype and PM Structure

```
/* Prototypes for callback functions */
typedef int (*pm_call)(enum pm_device_id dev_id);

/* PM structures */
struct pm_s {
   pm_call    nap;
   pm_call    wakeup_nap;
   pm_call    deep_nap;
   pm_call    wakeup_deep_nap;
   pm_call    sleep;
   pm_call    wakeup_sleep;
   pm_call    deep_sleep;
   pm_call    wakeup_deep_sleep;
};
```

'enum pm_device_id dev_id' is a list of unique IDs for all devices which need to handle PM calls for power saving and sleep/resume purposes.

5.1.2 Registration API

```
int power_manager_register(enum pm_device_id dev_id, struct pm_s *pm_p);

void power_manager_unregister(enum pm_device_id dev_id, struct pm_s *pm_p);
```

A driver with the need should register its callbacks (not necessarily all callbacks) for the needed power mode notification calls.

The driver who registered first will be called last in nap/sleep call chain, and will be called first in wakeup call chain. This is to support parent child relation order if any.

5.2 System Power Mode Handling API

5.2.1 Idle API

```
__NO_RETURN void power_manager_cpu_idle(void);
```

5.2.2 Init API

```
void power_manager_init(void);
```

5.2.3 CPU Error Notification API

```
void power_manager_error_notify(uint32_t code, void *object_id);
```

5.2.4 SW Reset API

```
void power_manager_reset(void);
```

5.2.5 Sleep API

```
void power_manager_sleep(void);
```

5.2.6 Deep Sleep API

void power_manager_deep_sleep(void);

5.2.7 Shutdown API

void power_manager_shutdown(void);

6 Special Cases

6.1 Companion mode

When KL520 is used as companion chip, the communication channel between KL520 and the Host could be powered off depending on which power domain the communication channel is supplied from. Therefore, special attention is needed to avoid certain power modes if the communication is expected to be alive on that power mode.

6.1.1 Host-KL520 communication: UART

UART power on KL520 is supplied from default power domain. So its power will be shut off when entering Deep Sleep mode, but alive in other lighter power modes.

6.1.2 Host-KL520 communication: USB

USB power on KL520 is supplied from NPU power domain. So its power will be shut off when entering Sleep mode, but alive in other lighter power modes.