Power overview

A quality version of this page, approved on *1 December 2020*, was based off this revision.

Contents

1 Framework purpose

The purpose of this article is to explain how to handle the STM32MP15x low-power modes:

- Low-power modes available on the device
- Linux software overview
- How to enter and exit the low-power modes on Arm®Cortex®-A7 core
- How to enter a platform low-power mode

2 Low-power modes available on the device

Refer to STM32MP15 reference manuals for the full description of low-power modes.
The AN5109 low-power application note also gives much more information on these modes, including:

- the detailed description of the operating modes,
- the low-power mode entry and exit sequences,
- the low-power mode control registers.

The modes are handled by the RCC and the PWR peripherals.

The table below summarizes the device hardware states corresponding to each low-power mode.

The term "subsystem" either refers to Arm®Cortex®-A7 (also called MPU) or to Arm®Cortex®-M4 (also called MCU). A mode prefixed by 'C' corresponds to a subsystem mode.

A platform mode is the combination of MPU and MCU modes.

Level	Mode	Vddcore state	Clocks state
Subsystem	MPU CRun	on	on
	MPU CStop	on	Subsystem off
	MPU CStandby	on	Subsystem off
	MCU CRun	on	on
	MCU CStop	on	Subsystem off

MPU mode	MCU mode	Platform mode	Vddcore state	Clocks state
CRun	CRun	Run	On	On
CStop	CRun	Run	On	On
CStandby	CRun	Run	On	On
CRun	CStop	Run	On	On
CStop	CStop	Stop/LPLV-Stop/Standby	On/Retention/Off	Off/Off/Off
CStandby	CStop	Stop/LPLV-Stop/Standby	On/Retention/Off	Off/Off/Off

2.1 Wakeup sources

The above modes are exited due to a wakeup event.

Again, the AN5109 low-power application note details, among other things, the wakeup sources, the software mechanism that ensures the consistency between the low-power mode and the activated wakeup source, and the low-power mode exit sequence.

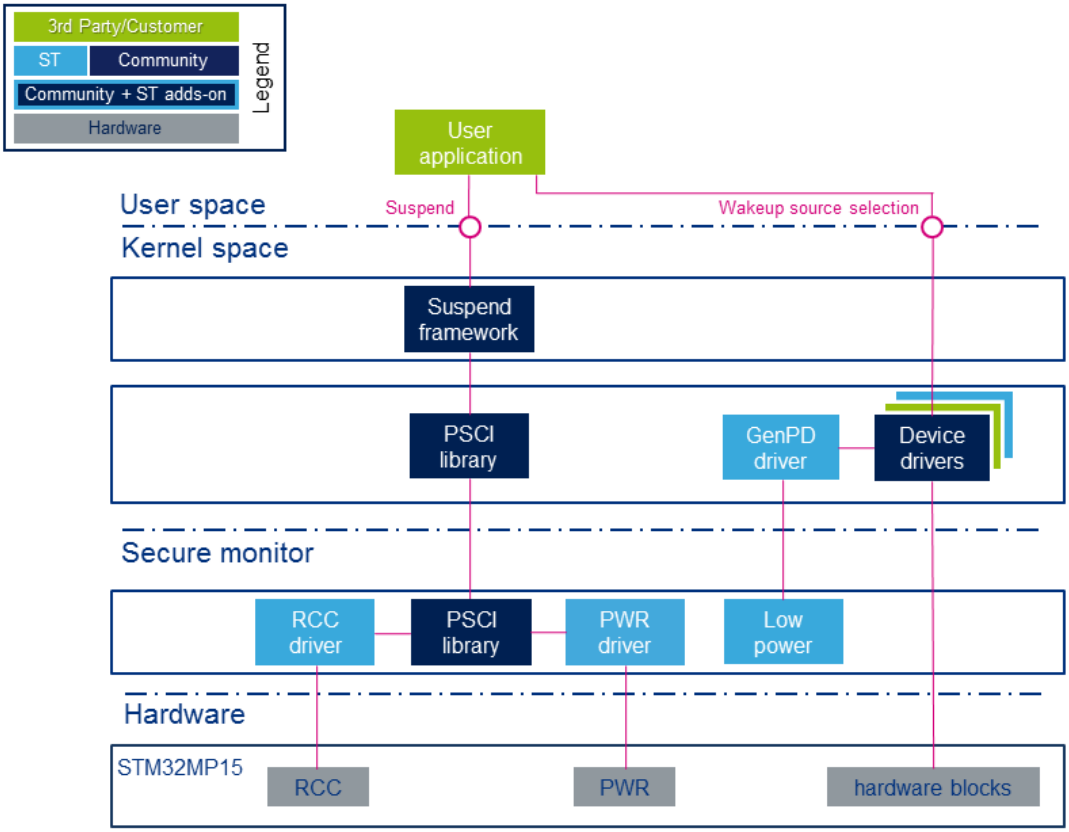
The following table gives the list of wakeup sources available in each mode.

Mode	Available wakeup sources
CStop /CStandby /Stop	BOR, PVD, AVD, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, USB, CEC, ETH, USART, I²C, SPI, LPTIM, IWDG, GPIO, Wakeup pins (from PWR)
LPLV-Stop	BOR, PVD, AVD, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, IWDG, GPIO, Wakeup pins (from PWR)
Standby	BOR, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, IWDG, Wakeup pins (from PWR)

3 Software overview

The Linux® suspend framework is used to trigger a low-power mode entry/exit sequence.

Refer to Documentation/power for more details.



The user application issues a suspend request to the kernel. This request is handled by the suspend Framework, which notifies all the device drivers to prepare for low-power entry. It then calls the PSCI service.

In addition to this centralized suspend process, most of the drivers implement the runtime pm feature. It is used to dynamically disable the resources of the peripherals (clocks and power when applicable) in case of inactivity (see Documentation/power /runtime_pm.rst).

3.1 Component description

Kernel components:

- **Suspend framework**: this framework schedules the overall sequence by stopping all the ongoing tasks
- **GenPD driver**: this driver is used for low-power mode selection according to the activated wakeup sources.
- **PSCI library**: this is a set of standardized functions to request a low-power service to the secure monitor
- **RCC driver**: this driver handles the circuit non-secure clocks

Secure monitor components:

- **PWR driver**: this driver is responsible for configuring the low-power mode
- **PSCI library**: this is a set of standardized functions handling the low-power services
- **Low power driver**: the role of this driver is to choose the low-power mode according to the programmed wakeup source(s)
- **RCC driver**: this driver handles the circuit secure clocks

3.2 API description

The suspend process is triggered from the user space through standard commands.

The system sleep control file is the *state* file, located under: /sys/power/

Only the 'mem' command is supported:

- The whole system activity is stopped and a low-power mode is entered. The software selects the deepest mode according to the activated wakeup source(s).

```
Example: Board $> echo mem > /sys/power/state
```

Further details can be found in Documentation/power/interface.rst

STMicroelectronics deliveries propose a default mapping of the low-power modes for each type of board.

Note that this default mapping can be changed thanks to the device tree. Refer to paragraph 3.3.2.

3.3 Software configuration

3.3.1 Menuconfig (Linux® kernel)

The suspend to RAM feature is activated by default in STMicroelectronics deliveries.

It can be deactivated through the kernel menuconfig using Power management options/Suspend to RAM and standby: Menuconfig or how to configure kernel .

3.3.2 Device tree (secure monitor)

The default system low-power mode mapping can be modified through the secure monitor device tree.

Below an example:

```
&pwr {
    system_suspend_supported_modes = <
        STM32_PM_CSLEEP_RUN
        STM32_PM_CSTOP_ALLOW_STOP
```

```
            STM32_PM_CSTOP_ALLOW_LP_STOP
            STM32_PM_CSTOP_ALLOW_LPLV_STOP
            STM32_PM_CSTOP_ALLOW_STANDBY_DDR_SR
        >;
        system_off_soc_mode = <STM32_PM_CSTOP_ALLOW_STANDBY_DDR_OFF>;
    };
```

For detailed information on the device tree concept, refer to Device tree.

3.3.3 Example of wakeup source activation

The activation of a wakeup source is done in the corresponding driver.

For example, activating UART4 as wakeup source is done thanks to the following commands:

```
Board $> echo enabled > /sys/devices/platform/soc/40010000.serial/tty/ttySTM0/power/wakeup
Board $> echo enabled > /sys/devices/platform/soc/40010000.serial/power/wakeup
```

It is possible to check the state of each wakeup source (activated or not) by displaying the 'wakeup' attribute.

Note that the software implements a consistency check between the selected wakeup source and the appropriate low-power mode.

4 How to enter and exit low-power modes

4.1 Platform low-power

Select the platform allowed modes depending on the required wakeup source.

Activate the wakeup source(s) (peripheral dependent).

Call the low-power mode on both sides (MPU and MCU).

4.2 MPU side

Activate the wakeup source(s) (peripheral dependent)

Call the low-power mode by issuing the following command:

echo mem > /sys/power/state

```
  Note that in Weston configuration the low-power mode is entered upon a 'systemctl
suspend' command.
```

4.3 MCU side

Please refer to Coprocessor power management for Arm®Cortex®-M4 commands.

4.4 Example: entering/exiting MPU CStop mode

Enable at least one wakeup source from table 2.1 in CStop category, for example USART:

```
Board $> echo enabled > /sys/devices/platform/soc/40010000.serial/tty/ttySTM0/power/wakeup
Board $> echo enabled > /sys/devices/platform/soc/40010000.serial/power/wakeup
```

Call the low-power entry:

```
Board $> echo mem > /sys/power/state
```

or for the Weston configuration:

```
Board $> cat /etc/systemd/sleep.conf
[Sleep]
SuspendMode=
HibernateMode=
HybridSleepMode=
SuspendState=mem
HibernateState=mem
HybridSleepState=mem
```

```
 Board $> systemctl suspend
```

The MPU is now in CStop mode, and can be woken up by sending a character to the console.

5 How to trace and debug

The suspend/resume process execution is logged in the MPU console. It gives useful information on the platform state (sleeping or active).

```
root@stm32mp1:~# echo mem > /sys/power/state
[ 1072.267571] PM: suspend entry (deep)
[ 1072.269687] PM: Syncing filesystems ... done.
[ 1072.279114] Freezing user space processes ... (elapsed 0.008 seconds) done.
[ 1072.292835] OOM killer disabled.
[ 1072.296046] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[ 1072.303431] Suspending console(s) (use no_console_suspend to debug)
[ 1072.332520] dwc2 49000000.usb-otg: suspending usb gadget configfs-gadget
[ 1072.332537] dwc2 49000000.usb-otg: dwc2_hsotg_ep_disable: called for ep0
[ 1072.332546] dwc2 49000000.usb-otg: dwc2_hsotg_ep_disable: called for ep0
[ 1072.468536] Disabling non-boot CPUs ...
[ 1072.507876] CPU1 killed.
[ 1072.509635] Enabling non-boot CPUs ...
[ 1072.510508] CPU1 is up
[ 1072.527553] dwmac4: Master AXI performs any burst length
[ 1072.527583] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support found
[ 1072.527621] stm32-dwmac 5800a000.ethernet eth0: ERROR failed to create debugfs
directory
[ 1072.527631] stm32-dwmac 5800a000.ethernet eth0: stmmac_hw_setup: failed debugFS
registration
[ 1072.588234] dwc2 49000000.usb-otg: resuming usb gadget configfs-gadget
[ 1072.738469] OOM killer enabled.
[ 1072.741575] Restarting tasks ... done.
[ 1072.752596] PM: suspend exit
```

It is also possible to monitor the hardware signals related to the system low-power modes thanks to the HDP internal peripheral. Please refer to HDP Linux driver for its configuration.

6 To go further

Refer to STM32MP15 reference manuals for a detailed description of low-power modes and peripheral wakeup sources.

The AN5109 low power application note gives additional information on the hardware settings used for low-power management.

Microprocessor Unit

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

Brownout reset

Programmable Voltage Detector

Analog Voltage Detector

Low Speed External oscillator (STM32 clock source)

Cascading Style Sheets (web standard)

Real Time Clock

Tamper

Consumer Electronics Control (HDMI standard)

Ethernet

Universal Synchronous/Asynchronous Receiver/Transmitter

Serial Peripheral Interface

low-power timer (STM32 specific)

Independent Watchdog

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Power State Coordination Interface

Reset and Clock Control

Application programming interface

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Low Power (MIPI® Alliance DSI standard)

Doubledata rate (memory domain)

Out Of Memory

Configuration File System (See https://en.wikipedia.org/wiki/Configfs for more details)

Debug File System (See https://en.wikipedia.org/wiki/Debugfs for more details)