STM32MP1 系列使用低功耗模式

引言

STM32MP1 系列器件基于配备单核或双核 MPU 子系统且结合 Arm® Cortex®-M4 CPU 的 Arm® Cortex®-A7。

STM32MP1 系列器件可配置多种低功耗模式，以便在必要时降低功耗。

本应用笔记介绍 STM32MP1 系列器件的各种低功耗模式，以及如何配置和退出这些模式。本文档还提供了在系统级别使用低功耗模式以及使用外部 STPMIC1x 功率调节器组件时需要考虑的一些指导。

有关 STM32MP1 系列器件的更多信息，请参阅可在 www.st.com 上获取的以下文件和可交付成果。

- STM32MP1 系列参考手册（详细信息见表 1. STM32MP1 系列的配置）
- STM32MP1 系列数据手册
- STPMIC1x 数据手册
- *STM32MP1 系列硬件开发入门*（AN5031）
- *STM32MP1 系列和 STPMIC1x 硬件与软件集成*（AN5089）
- STM32CubeMP1 封装
- STM32MP1 系列嵌入式软件

1 概述

本应用笔记适用于 STM32MP1 系列的所有器件。下表描述了 STM32MP1 系列所有产品的主要特性。

根据器件产品编号，该系统包括 Cortex-M4 以及单核或双核 Cortex-A7。在本文件中，Cortex-A7 称为 MPU，Cortex-M4 称为 MCU。

全功能系统（参见下表）包括：

- 一个 MPU 子系统：双 Cortex-A7，配 L2 缓存
- 一个 MCU 子系统：Cortex-M4，相关外设根据 CPU 活动进行时钟计时

本文件假定是全功能器件（例如 STM32MP157）。

表 1. STM32MP1 系列的配置

线路	参考手册	Cortex-A7 配置	Cortex-M4	GPU	DSI	FDCAN
STM32MP151	RM0441	单核	有	无	无	无
STM32MP153	RM0442	双核	有	无	无	有
STM32MP157	RM0436	双核	有	有	有	有

在本文档中，MCU 子系统有时称为“MCU”，MPU 子系统被称为“MPU”，以便于文档阅读。

本文档适用于基于 Arm®的器件。

提示 *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

2 词汇表

表 2. 词汇表

术语	意义
AHB	高级高性能总线
ATF	Arm®可信固件
AVD	模拟电压检测器
BKPSRAM	备份 SRAM
BOR	欠压复位
BUCK	降压开关电源转换器
CSS	时钟安全系统
DDR	双数据速率（SRAM）
DDRCTRL	DDR 控制器
DDRPHYC	DDR 物理接口控制
DSI	显示器串行接口
ETH	以太网控制器
FDCAN	具有灵活数据速率的控制器局域网络。 还支持时间触发的 CAN（TT）
GPIO	通用输入输出
GPU	图形处理单元
HAL	硬件抽象层
HDP	硬件调试端口
IRQ	中断请求
IWDG	独立看门狗
LDO	低压降调节器
LpDDR	低功耗 DDR
LSE	低速外部石英振荡器
LSI	低速内部振荡器
MCU	微控制器
MLAHB	多层 AHB / 基于 AHB 的互连
MPU	微处理器
PLL	锁相环
PVD	可编程电压检测器
PWR	电源控制块
QSPI	四路数据通道串行外设接口
RCC	复位和时钟控制
RETRAM	保留存储器
RTC	实时时钟
SDMMC	安全数字和多媒体卡接口。 支持 SD、MMC、eMMC 和 SDIO 协议
SMP	对称式多处理
SPL	二级程序加载器

术语	意义
SRAM	静态随机存取存储器
STPMIC1x	电源管理集成电路通过信号和串行接口提供具有很大可控性的各种平台电源的外部电路。
SYSRAM	系统 SRAM
SW	软件
TEMP	温度传感器
TEMPH-L	温度传感器高-低监控
USART：	通用同步异步收发器
USB OTG	通用串行总线（USB） on-the-go（OTG）。 一种标准 USB 接口，能够成为主机或设备
VBATH-L	VBAT 高-低监控
WFE	等待事件
WFI	等待中断

3 电源管理理念

本节描述 STM32MP1 系列高级电源管理理念。

有关详细信息，请参阅相应的参考手册（参见表 1. STM32MP1 系列的配置）。

3.1 STM32MP1 系列系统架构

STM32MP1 系列基于一个主要的双核 MPU 子系统和一个 MCU 子系统。多核架构要求在系统级别和各个 MPU 和 MCU 子系统级别都有特定的电源模式。

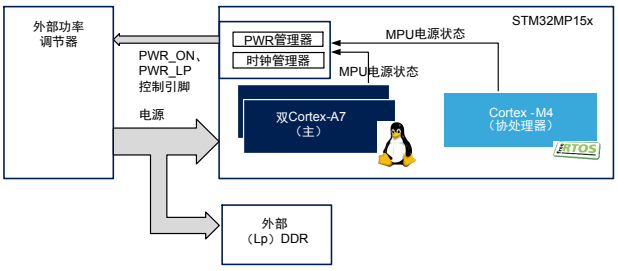
STM32MP1 系列的内部数字逻辑由外部提供，这不同于大多数 STM32 MCU，后者有一个内部 LDO（低压降调节器）用于此目的。该关键差异导致 STM32MP1 系列不同的电源管理，需要使用外部信号和/或外部调节器编程来控制所需的电压供应。

电源管理特性分布在 STM32MP1 系列器件的 RCC（复位和时钟控制）和 PWR（电源）块之间。

- RCC 块确保时钟树处理（PLL、mux、分频器、时钟门控）和复位（外设的本地复位、Cortex-A7 复位、Cortex-M4 复位、平台复位）
- RCC 块允许根据 MCU 和 MPU 子系统各自的电源状态选择电源模式
- PWR 块负责低功耗模式的进入/退出。它基于电源模式驱动到外部调节器的控制引脚（PWR_ON、PWR_LP）。

下图显示了 STM32MP1 系列的高级系统架构。

图 1. STM32MP1 系列高级系统架构



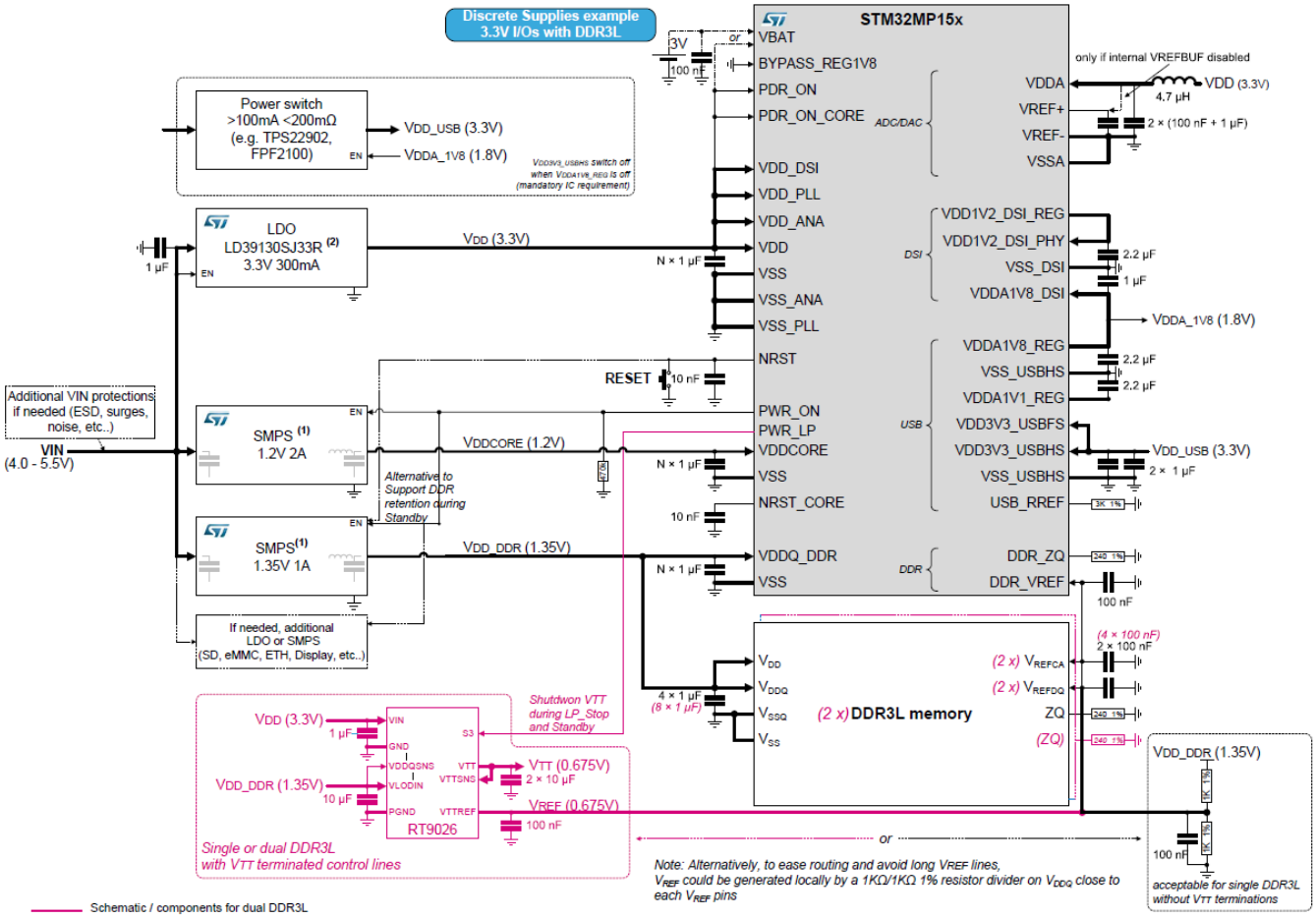
3.2 系统电源（V_{DD} 和 V_{DDCORE}）

STM32MP1 系列器件需要多种电源，详见相应的数据手册和参考手册（参见表 1. STM32MP1 系列的配置）。在这些电源中，V_{DD} 和 V_{DDCORE} 在低功耗模式配置中起关键作用。

- V_{DD} 为 IO 和系统模拟（如复位、电源管理振荡器和 PLL）提供电源输入。
- V_{DDCORE} 数字核心域电源，取决于 V_{DD} 供应。V_{DD} 应该在 V_{DDCORE} 供电。

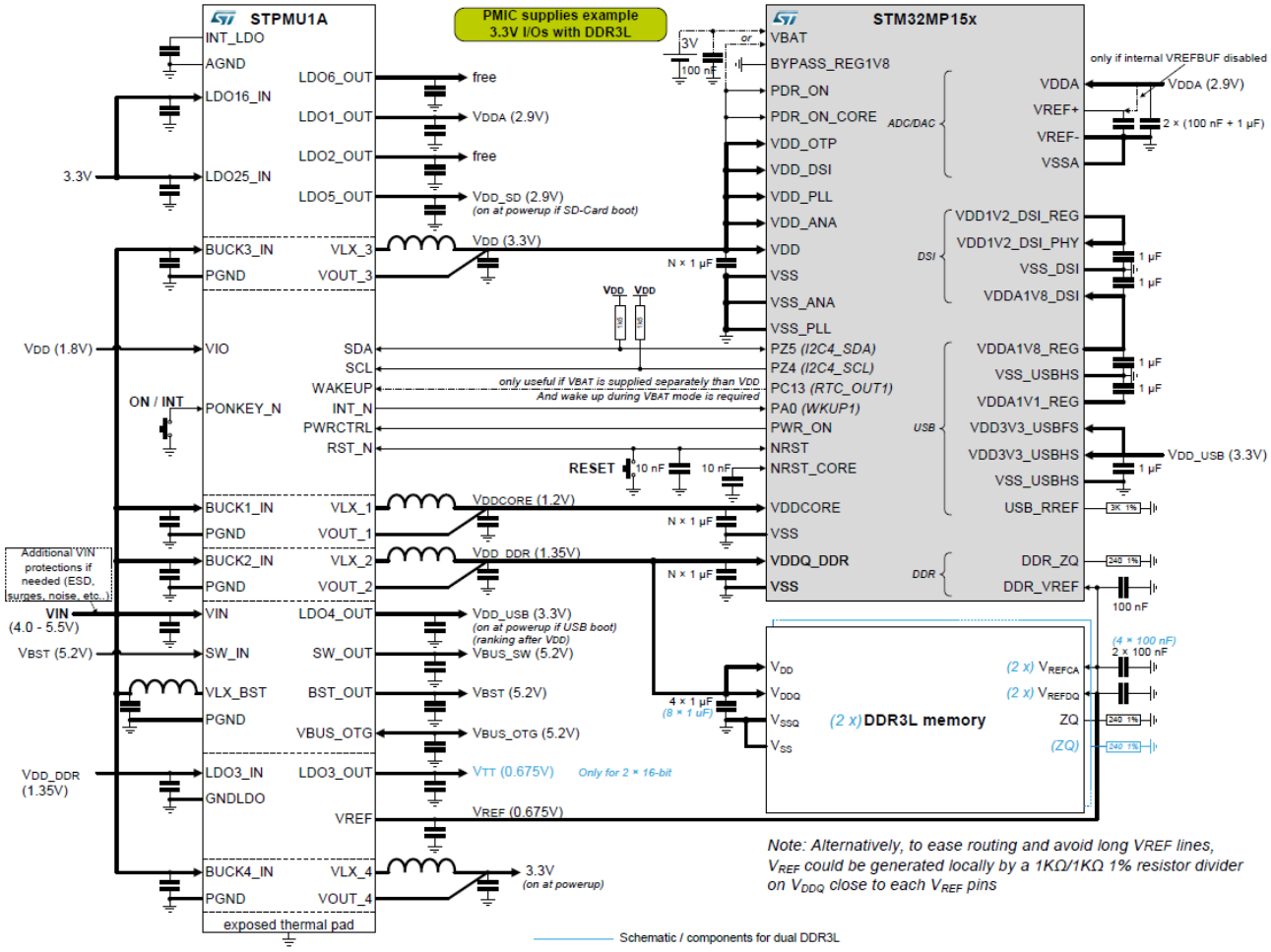
STM32MP1 系列器件的各种电源引脚可以由一些外部分立电源或使用 STPMIC1x 供电，详见应用笔记 *STM32MP1 系列硬件开发入门*（AN5031）中的参考设计部分。

图 2. 采用 DDR3L 的 3.3V I/O 分立电源示例



提示　　支持几种类型的 DDR（LpDDR2、LpDDR3、DDR3），这将影响低功耗模式管理。请参见图 7. 可用的电源状态转换。

图 3. 采用 DDR3L 的 3.3 V I/O 的 STPMIC1x 示例



STPMIC1x 提供配电功能。它由 I²C 和来自 STM32MP1 系列器件的控制信号驱动。控制由 Cortex-A7 MPU 子系统核心完成。使用 STPMIC1x 供电相对于使用分立调节器供电的主要优点是拥有一种体积更小、价格更便宜的集成式解决方案。详细信息请参阅应用笔记 *STM32MP15x 和 STPMIC1x 硬件与软件集成*（AN5089）。

4 工作模式

本节介绍 STM32MP1 系列器件的各种电源模式及其激活方法。

4.1 工作模式描述

工作模式允许控制分配给不同系统部分的时钟和控制系统的电源供应。系统工作模式由 MPU 子系统和 MCU 子系统驱动。

电源管理根据系统工作模式控制 V_{DDCORE} 供应。

下表给出了系统以及 MPU 子系统和 MCU 子系统可用的工作模式。系统级的电源模式取决于各个子系统级（MPU 和 MCU）的电源模式。

表 3. 功耗模式

-	电源模式	说明	注释
系统	运行	V_{DDCORE}（电源打开，时钟打开）	-
	停止	V_{DDCORE}（电源打开，时钟关闭）	-
	LP-Stop	V_{DDCORE}（电源打开，时钟关闭）	(1)
	LPLV-Stop	V_{DDCORE}（降低功率电平和供应负荷，时钟关闭）	(1)(2)
	待机	V_{DDCORE}（电源关闭，时钟关闭）	-
	VBAT	V_{SW} 由电池供电，V_{DDCORE} 关闭，RTC 仍然是由 LSE 晶振驱动的活跃时钟	(3)
	掉电	所有电源关闭	-
MPU / MCU	CRun	V_{DDCORE}（电源打开，时钟打开）	-
	CSleep	V_{DDCORE}（电源打开，CPU 时钟关闭，外部时钟打开/关闭）	(4)
	CStop	VDDCORE（电源打开，CPU 子系统时钟关闭）	-
MPU	CStandby	V_{DDCORE}（电源打开或关闭，时钟关闭）	(5)

1. *输出控制引脚 PWR_ON 和 PWR_LP 在 LP-Stop 和 LPLVStop 两种模式下没有区别（参见第 4.3 节 外部控制信号 PWR_ON、PWR_LP 引脚）。虽然可以使用 STPMIC1x 激活 LP-Stop 和 LPLV-Stop（先对其内部寄存器进行编程，然后进入目标低功耗模式（参考第 4.3.1 节 使用 STPMIC1x 功率调节器）），但如果使用其他外部功率调节器，这两种模式可能不可用。*
2. *与停止模式的主要区别是：使用 STPMIC1x 外部功率调节器时，PWR_ON 输出信号切换到 0，可以采取一些行动（如切断 DDR 终端电阻电源）。*
3. *为了在 V_{DD} 关闭后保留 VSW 域（RTC、备份寄存器、备份 RAM 和 retention RAM）的内容，可以将 VBAT 引脚连接到由电池或其它电源供电的可选备用电压。*
4. *MCU/MPU 子系统分配的外部时钟根据 RCC PERxLPEN 运行。当 MPU 进入带 WFE（等待事件）功能的 CSleep 时，MPU 子系统分配的外部时钟的运行方式与在 MPU CRun 模式下的运行方式相同，而不考虑 RCC PERxLPEN。*
5. *在 CStandby 模式下，只有系统模式为待机（MCU 处于 CStop，PDDS=1）时，MPU 才关闭电源。请参见表 5. 电源系统模式 VS 子系统模式总结。*

使用哪种低功耗模式取决于节电目标，必须根据所选的低功耗模式下需要/可用的唤醒中断和预期的唤醒持续时间进行选择。

用户应始终确保所选的低功耗模式与可用的唤醒源一致。例如，LPLV-Stop 或待机模式的唤醒源能力非常有限。

此外，如果电源电压降低或关闭 - 原因是 V_{DDCORE} 降低、DDR 自刷新而引起 DDR 电阻端接电源关闭、或整个 DDR 关闭（可能需要重新从闪存加载固件）- 则唤醒持续时间更长。下表给出了系统的低功耗模式唤醒能力。

表 4. 系统的低功耗模式唤醒能力

系统功耗模式	唤醒源
Stop/LP-Stop	DBG、PVD、AVD、USBH、OTG、CEC、ETH、MDIOS、USARTx、I2Cx、SPIx TEMP、LPTIMx、GPIOs
LPLV-Stop	PVD、AVD、TEMP、GPIOs
待机	五个 GPIO 引脚
所有模式	BOR、VBATH/VBATL、TEMPH/TEMPL、LSE CSS、RTC/自动唤醒、tamper 引脚、IWDGx

4.2 低功耗模式控制

4.2.1 低功耗模式控制寄存器

下面介绍的控制寄存器位与低功耗模式有关（关于低功耗模式的描述，请参见第 4 节 工作模式）。

PWR MPU 控制寄存器（PWR_MPUCR）中的 PDDS 位

0：当 MPU 进入 CStop 时，保持停止模式。
1：当 MPU 进入 CStop 时，允许待机模式。
当 CSTBYDIS = 0 时，允许 MPU 进入 CStandby 模式。

PWR MCU 控制寄存器（PWR_MCUCR）中的 PDDS 位

0：当 MCU 进入 CStop 时，保持停止模式。
1：当 MCU 进入 CStop 时，允许待机模式。

PWR MPU 控制寄存器（PWR_MPUCR）中的 CSTBYDIS 位

0：启用 MPU CStandby 模式。
1：禁用 MPU CStandby 模式。

MPU 和 MCU 都可以访问 PWR_MPUCR 和 PWR_MCUCR。但是，PWR_MPUCR 可以设置为安全模式（Linux 用例），只允许 MPU 写访问。

MCU 可以使用 HAL_PWR_EnterStandbyMode()函数写入 PWR_MCUCR 寄存器位。

- HAL 函数 HAL_PWR_EnterStandbyMode()将 MCU PDDS 位设为‘1’，通过 WFI（等待中断）让 MCU 子系统进入 CStop 模式。仅当 MPU 子系统模式进入 CStop 模式（PDDS=1 & CSTBYDIS=1）或 CStandby 时，才进入待机模式（参见 4.4 节：低功耗模式进入顺序）。

下表分别详细说明了系统与 MPU/MCU 子系统之间可能的功耗模式组合以及系统的低功耗模式总结。

表 5. 电源系统模式 VS 子系统模式总结

-		MPU[1]				
		CRun	CSleep	CStop（PDDS = 0）	CStop（PDDS = 1 CSTBYDIS=1）	CStandby[2]
MCU	CRun	运行	运行	运行	运行	运行[3]
	CSleep	运行	运行	运行	运行	运行[3]
	CStop（PDDS = 0）	运行	运行	停止 LP-Stop LPLV-Stop	停止 LP-Stop LPLV-Stop	停止[3] LP-Stop LPLV-Stop
	CStop（PDDS = 1）	运行	运行	停止 LP-Stop LPLV-Stop	待机[4]	待机[4]

1. *请参考第 4.6.7 节 从 MPU CStop 和 CStandby 模式退出获取关于 CStop 和 CStandby 的详细信息。*
2. *STM32MPU OpenSTLinux Distributions 中不支持 MPU 的 CStandby 模式。当 MPU 处于 CStop 模式（PDDS=1，CSTBYDIS=1）时，系统待机。*
3. *当系统处于运行、停止、LP-Stop 或 LPLV-Stop 模式时，从 MPU CStandby 状态退出，通过本地 MPU 复位使程序执行重新启动。这种"重启"可能会影响 MCU 运行，因为当从外部 Flash 重新加载 MPU 初始化代码时，BootROM 可能会重新初始化一些外部寄存器。*
4. *当系统处于待机状态时，从 MPU CStandby 或 CStop (PDDS=1, CSTBYDIS=1)退出，程序执行重新启动的方式与上电复位（加载选项字节，获取复位向量，或其他）后相同。*

表 6. 系统低功耗模式总结

系统	MPU	DDR[1]	MCU	系统振荡器[2]	hclk4	PWR_LP	PWR_ON	
							LPCFG = 0	LPCFG = 1
运行	CRun 或 CSleep	主动 / 自动刷新	CRun 或 CSleep	开启	开启	1	1	1
	CStop 或 CStandby	自刷新						
	CRun 或 CSleep	主动 / 自动刷新	CStop					
停止（LPDS = 0）[3]	CStop 或 CStandby	自刷新	CStop	开/关[4]	关闭	1	1	1
LP-Stop LPLV-Stop[3]（LPDS = 1）						0[5]		0[5]
待机	CStandby 或（CStop 和 MPU PDDS = 1 且 MPU CSTBYDIS = 1）	关闭/自刷新	CStop 和 MCU PDDS = 1	关闭		0[6]	0[6]	0[6]

1. *DDR 运行状态仅供参考。例如，当 MPU 处于 CRun（就像从 BootROM 执行时一样）时，DDR 可能关闭。*
2. *系统振荡器是在 RCC 中配置为"打开"的 HSI、HSE 或 CSI 振荡器。*
3. *至少 1 个 PDDS 位（MCU 或 MPU）选择"停止（Stop）"（PDDS = 0）。*

4. *当使用系统振荡器 HSI、HSE 或 CSI 时，状态由各自 xxxKERON（RCC 振荡器时钟使能设置寄存器（RCC_OCENSETR）和 RCC 时钟振荡器使能清除寄存器（RCC_OCENCLRR）中的 HSIKERON、HSEKERON 或 CSIKERON 位）控制，否则系统振荡器关闭。*
5. *PWR 控制寄存器 3（PWR_CR3）中 POPL 位的设置对 LP-Stop 和 LPLV-Stop 模式的 PWR_ON 和 PWR_LPPWR_LP 脉冲低电平时间没有影响。*
6. *PWR 控制寄存器 3（PWR_CR3）中的 POPL 位可以定义保证的最短 PWR_ON 和 PWR_LP 脉冲低电平时间。*

4.3 外部控制信号 PWR_ON、PWR_LP 引脚

两个输出引脚 PWR_ON 和 PWR_LP 与 VDDCORE 供电有关。它们被外部元件/调节器用来识别应该施加在 V_{DDCORE} 上的电压水平。参见第 4 节 工作模式获取关于 PWR_ON 和 PWR_LP 配置的详细信息。

- PWR_ON：V_{DDCORE} 供电请求

  它是由硬件根据 STM32MP1 系列器件的状态和 PWR 控制寄存器 1（PWR_CR1）中的 LPCFG（PWR_ON 引脚配置）位的值自动生成。
- PWR_LP：V_{DDCORE} 低功耗模式控制（低电平有效）

  它是由硬件根据 STM32MP1 系列器件的状态和 PWR 控制寄存器 1（PWR_CR1）中的 LPDS （低电压深度睡眠 LPLV-Stop 模式选择）位的值自动生成。

下表显示了 PWR_ON 和 PWR_LP 的输出引脚值，取决于不同的电源模式和 LPDS、LPCFG、LVDS 位配置。

- LPDS 是 PWR 控制寄存器 1（PWR_CR1）位 0
- LPCFG 是 PWR 控制寄存器 1（PWR_CR1）位 1
- LVDS 是 PWR 控制寄存器 1（PWR_CR1）位 2

该表还显示了引脚 PWR_ON 和 PWR_LP 的使用差异，具体取决于 STM32MP1 系列器件是与 STPMU1x 一起使用还是与其他外部电源组件一起使用。

电源模式在相应参考手册的 PWR 部分有详细说明（参见表 1. STM32MP1 系列的配置）。

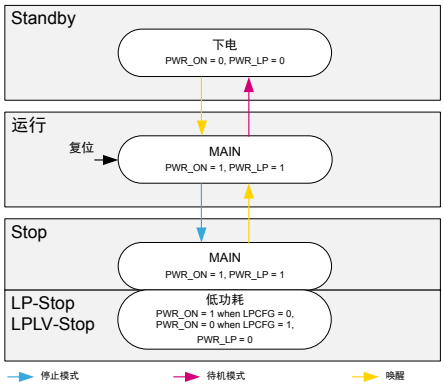
表 7. PRW_ON 和 PWR_LP 电平取决于电源模式、LPDS、LVDS 和 LPCFG 位

STM32MP15x 状态	LPDS/LVDS 位	LPCFG 位	PWR_ON	PWR_LP	V_{DDCORE}
启动（直至 V_{DD} 达到 POR 门限电平）	X/X	X	0	0	关闭
运行模式	X/X	X	1	1	开启
停止模式	0/X	X	1	1	开启
LP-Stop 模式	1/0	0	1	0	开启
		1(1)	0(2)	0(2)	
LPLV-Stop 模式	1/1	0(3)	1	0	开启(4)
		1(1)	0(2)	0(2)	
待机模式	0/0	X	0(2)	0(2)	关闭
VBAT 模式（V_{DD} 断电）	X/X	X	高阻态	高阻态	关闭

1. *当 STPMIC1x PWRCTRL 引脚连接到 PWR_ON 时使用的配置。*
2. *PWR_ON 的 LP-Stop、LPLV-Stop 和待机模式没有区别，当 LPCFG 位=1 时，PWR_LP 输出值为'00'。*
3. *根据所使用的外部功率调节器，此配置可能不可用（无法降低电源电平）。*
4. *V_{DDCORE} 可降低供电电平和供电负荷。*

下图说明了 STM32MP1 系列器件的调节器控制的 PWR_ON 和 PWR_LP 输出引脚与系统状态之间的关系。

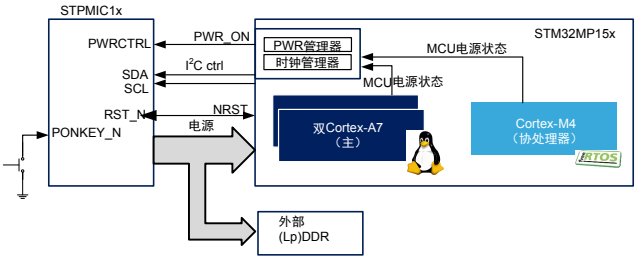
图 4. 电源状态和外部调节器控制



4.3.1 使用 STPMIC1x 功率调节器

下图显示了使用 STPMIC1x 功率调节器时在系统级别上的主要区别。

图 5. 使用 STPMIC1x 时的 STM32MP1 系列高层系统架构



使用 STPMIC1x 功率调节器时，只需 STM32MP15x 的 PWR_ON 输出控制引脚（通过 PWRCTRL 引脚）即可控制 STPMIC1x 电源模式。在这种情况下，用户需要将 PWR_CR1 寄存器的 LPCFG 位设置为“1”。

表 7. PRW_ON 和 PWR_LP 电平取决于电源模式、LPDS、LVDS 和 LPCFG 位 在灰色单元中高亮显示 PWR_ON 的 LP-Stop、LPLV-Stop 和待机模式没有区别，当 LPCFG 位=1 时，PWR_LP 输出值为‘00’。

STPMIC1x 区分 LP-Stop、LPLV-Stop 和待机低功耗模式，并通过 STPMIC1x 内部寄存器应用正确的 VDDCORE 值。这些寄存器编程是通过 I2C 接口完成的，而不是通过 PWR_ON 和 PWR_LP 引脚值。

使用 STM32MPU OpenSTLinux Distributions 时，此编程由第一级引导程序（TF-A 或 OPTEE）处理。因此，在进入 LP-Stop、LPLV-Stop 或待机模式之前，应用程序应该将 STPMIC1x 内部寄存器配置为所需的系统供电电平（如 V_{DDCORE}、V_{DD_DDR} 等）。

进入 LP-Stop 模式后，尽管 V_{DDCORE} 电平没有降低（仅在 LPLV-Stop 模式下有效），仍有可能对 STPMIC1x 进行编程，以便在不需要时关闭其他电源（例如 DDR 终端电阻电源）。

STPMIC1x 包括两个 8 位寄存器，面向每个 BUCK（1，2，3，4）和 LDO（1，2，3，4，5，6）：

- BUCKx/LDOx 控制寄存器（HP 模式），存储运行模式期间的预期输出电压值。
- BUCKx/LDOx 控制寄存器（LP 模式），存储低功耗模式期间的预期输出电压值。

这些寄存器中的每一个均有如下结构：

7	6	5	4	3	2	1	0
输出值<5:0>						hplp	ena

Bits 7:2: 输出值：6 位上的输出值对应一个指定的电压值（参考 STPMIC1x 数据手册）。

Bit 1：hplp：强制高/低负载能力（允许在 STPMIC1x 输出上增加/减少负载）。

0：高负载能力

1：低负载能力

提示 *hplp 位仅适用于 BUCKx 寄存器，不适用于 LDOx。*

Bits 0：ena：使能位。

0：禁止 BUCK/LDO

1：使能 BUCK/LDO

下表分别显示了 STPMIC1x 在运行模式下的编程方式，以及 STPMIC1x 在进入 LP-Stop、LPLV-Stop 和待机模式前的编程方式。这两个表都列出了所有的电源（不仅是 V_{DDCORE} 电源），而且使用的设置与上面的示例一致。

图 3. 采用 DDR3L 的 3.3 V I/O 的 STPMIC1x 示例.

表 8. 运行模式下的 STPMIC1x（HP 模式）编程

电源名称	控制寄存器（HP 模式）/@	运行
V_{DDCORE}	BUCK1 / 0x20	0x61（1.2 V）
V_{DD_DDR}	BUCK2 / 0x21	0x79（1.35 V）
V_{DD}	BUCK3 / 0x22	0xD9（3.3 V）
3V3	BUCK4 / 0x23	0xD9（3.3V）
V_{REF_DDR}	VREFDDR / 0x24	0x1
V_{DDA}	LDO1 / 0x2A	0x51（2.9 V）
V_{DD_USB}	LDO4 / 0x28	0x1（3.3 V）
V_{DD_SD}	LDO5 / 0x29	0x51（2.9 V）
V_{TT}	LDO3 / 0x27	0x7D（0.675 V）(1)

1. *LDO3 输出值等于 BUCK2 输出 /2 = 0.675 V（适用于 DDR3 用例）。*

表 9. 进入 LP-Stop、LPLV-Stop、以及待机模式前的 STPMIC1x（LP 模式）编程

电源名称	控制寄存器（LP 模式）/@	LP-Stop	LPLV-Stop	待机，DDR SR 开启	待机，DDR SR 关闭
V_{DDCORE}	BUCK1 / 0x30	0x61（1.2 V）	0x33（0.9 V）	0x30（关闭）	
V_{DD_DDR}	BUCK2 / 0x31	0x79（1.35 V）			0x7A（关闭）
V_{DD}	BUCK3 / 0x32	0xD9（3.3 V）			
3V3	BUCK4 / 0x33	0xD9（3.3 V）		0xD8（关闭）	
V_{REF_DDR}	VREFDDR / 0x34	0x1			0x0（关闭）
V_{DDA}	LDO1 / 0x3A	0x51（2.9 V）		0x50（关闭）	
V_{DD_USB}	LDO4 / 0x38	0x1（3.3 V）		0x0（关闭）	
V_{DD_SD}	LDO5 / 0x39	0x51（2.9 V）		0x50（关闭）	
V_{TT}	LDO3 / 0x37	0x7C（关闭）			

提示 *将 bit0 设为'0'会禁用相应的 BUCK/LDO，其值设为"关闭"。*

例如：当 bit0='1'时，V_{DDA} = 2.9 V (0x51)；当 bit0='0'时，其值为"关闭"。

应用笔记*"STM32MP15x 和 STPMIC1x 硬件与软件集成"*（AN5089）详细说明如何使用 STPMIC1x。

当用户将 STPMIC1x PONKEY_N 引脚设置为"0"时，STM32MP1 系列的 NRST 引脚可以通过 STPMIC1x RST_N 引脚激活，从而使 STPMIC1x 和 STM32MP15x 器件都被复位。

4.3.2 使用其他外部电源

当使用 STPMIC1x 以外的外部电源时，如果应用程序需要低于停止模式的低功耗模式（如 LP-Stop、LPLV-Stop 和待机模式），可以同时使用 PWR_ON 和 PWR_LP 输出控制引脚来控制外部电源。

当 STM32MP1 系列器件进入 LP-Stop 或 LPLV-Stop 模式时，PWR_ON 应设为‘1’；当它们处于待机状态时，PWR_ON 应设为‘0’。要激活这种用法，用户必须将 LPCFG 位设为‘0’。

4.4 低功耗模式进入顺序

STM32MP1 系列在子系统级别（MPU，MCU）和系统级别有专用的电源模式（参见第 4 节 工作模式）。本节详细介绍如何在子系统级别和系统级别进入这些电源模式。

MPU 子系统

MPU 子系统的低功耗模式（CSleep、CStop 和 CStandby）由 MPU 在执行 WFI（等待中断）或 WFE（等待事件）指令时进入。只有在进入 CSleep 时，WFE 才可用。为了让 MPU 进入子系统的低功耗模式，RCC 和 PWR 寄存器必须已被正确编程（参照第 5.1 节 外设分配）。使用 STM32MPU OpenSTLinux Distributions 时，这些机制由第一级引导程序处理（参照第 4.5 节 Linux 系统下的电源管理）。

MCU 子系统

通过执行 WFI（等待中断）或 WFE（等待事件）指令，或者 Cortex®-M4 系统控制寄存器中的 SLEEPONEXIT 位设为“开启”（Return from ISR）时，即可使得 MCU 进入 MCU 子系统的低功耗模式（CSleep 和 CStop）。

在 MCU 上使用 STM32CubeMP1 包时，为低功耗模式定义了几个函数。下表描述了 STM32CubeMP1 包中根据其专用的低功耗模式定义的每个函数。

表 10. 在 STM32CubeMP1 包中的 MCU 上使用的低功耗模式函数

低功耗模式	功能	参数	说明
CSleep	HAL_PWR_EnterSLEEPMode	PWR_MAINREGULATOR_ON PWR_LOWPOWERREGULATOR_ON	STM32MP1 系列器件不使用调节器参数，为了与低功耗系列（如 STM32L 系列）兼容，将其保留为参数。
		PWR_SLEEPENTRY_WFI PWR_SLEEPENTRY_WFE	指定是否使用 WFI 或 WFE 指令进入睡眠模式
	HAL_PWR_EnableSleepOnExit	无	设置 SCR 寄存器的 SLEEPONEXIT 位。 设置该位后，当中断处理结束时，处理器重新进入睡眠模式
	HAL_PWR_DisableSleepOnExit	无	清除 SCR 寄存器的 SLEEPONEXIT 位
CStop， PDDS=0	HAL_PWR_EnterStopMode	PWR_MAINREGULATOR_ON PWR_LOWPOWERREGULATOR_ON	STM32MP1 系列器件不使用调节器参数，为了与低功耗系列（如 STM32L 系列）兼容，将其保留为参数。
		PWR_STOPENTRY_WFI PWR_STOPENTRY_WFE	指定是否使用 WFI 或 WFE 指令进入停止模式
CStop， PDDS=1	HAL_PWR_EnterStandbyMode	无	该功能使 MCU 转到 CStop（WFI），使 MCU PDDS 位 = 1；当 MPU 进入 CStandby 或 in (CStop 和 MPU PDDS = 1, MPU CSTBYDIS = 1)时，系统进入待机状态。

系统

当所有 EXTI 唤醒源被清除，MPU 和 MCU 都处于 CStop 或 CStandby 模式时，系统可以进入 Stop、LP-Stop、LPLV-Stop 或待机模式。

只有当 MPU 处于 CStandby 或 CStop 状态（MPU PDDS=1, MPU CSTBYDIS =1）且 MCU 处于 CStop 状态（MCU PDDS=1）时，系统才进入待机状态（电源关机、时钟关闭）。

进入待机模式必须由 MCU 协处理器固件仔细管理：只有在能够管理待机状态时，才允许通过 MCU“CStop，PDDS=1”进入，所以在此期间至少可以丢失 MCU SRAM 内容。

通过使用 HOLD_BOOT 特性可以关闭分配给 MCU 的时钟和外设（相当于 CStop 模式）。这个特性由 MPU 通过位于 RCC 全局控制寄存器（RCC_MP_GCR）中的 BOOT_MCU 位控制。它允许系统被置于停止、LPLV-Stop 或待机模式。该特性在 MCU 复位之后被激活（当 MPU 正在设置 RCC 全局重置控制设置寄存器（RCC_MP_GRSTCSETR）的 MCURST 位时）。

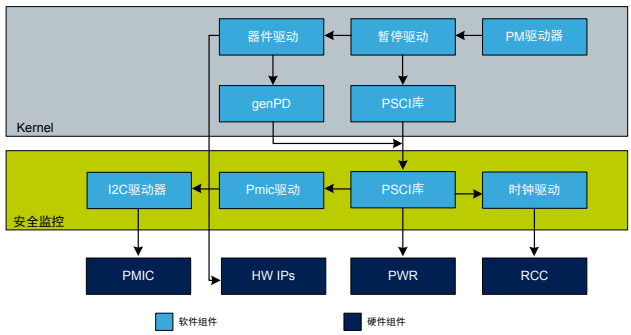
也可以只在 MPU 上运行代码（MCU 处于 CStop 等效模式），只要让 MPU 不在 MCU 端加载任何代码即可。因此，在启动系统时，MCU 默认处于非活动状态（除了启动模式 BOOT[2:0] = 0b100）。

提示 *进入 LPLV-Stop 时，如果 VDDQ_DDR 没有关断，为避免过度消耗 VDDQ_DDR，DDR 内存必须置于“SelfRefresh”，而 DDR PHY 必须设为保留模式（设置 DDRRETEN 位：启用 PWR 控制寄存器 3（PWR_CR3）的 DDR 保留模式）。*

4.5 Linux 系统下的电源管理

Linux 电源管理是通过下图所示的软件框架完成的。

图 6. Linux 电源管理软件框架



Linux 通用时钟框架（CCF）用于处理时钟树，并通过标准化的 API 将其公开给器件驱动：

- clk_prepare_enable，clk_disable_unprepare
- clk_get rate
- clk_set_parent，clk_get_parent
- CCF 依赖于 ST 时钟驱动

Linux 暂停框架进入低功耗模式。它依赖于：

- genPD（通用电源域）框架，允许定义电源域并允许在那些电源域上映射 IP
- PSCI 1.0 标准，执行低水平电源管理过程

其他信息可以在*电源概述*维基文章中找到。

安全监控调节器框架用于为每个电源模式配置外部功率调节器电压，它依赖于一个 ST STPMIC1x（PMIC）驱动和一个 I2C 驱动。

4.5.1 Linux 电源命令映射到 STM32MP1 系列电源模式

使用 Linux 操作系统时，电源模式命令是预定义的，本节解释如何将它们映射到 STM32MP1 系列硬件低功耗模式。本节还讲解使用 MCU 协处理器实现低功耗模式控制。

在 Linux 操作系统中，用户可以通过以下直接输入命令让系统进入低功耗模式：

```
'shutdown -h 0'
```

或

```
echo 'mem' > /sys/power/state
```

STM32MP1 系列不支持‘disk’、‘freeze’和‘standby’命令。

进入低功耗模式后，Cortex®-A7 处于 WFI（等待中断）状态。但是，只有提前将 MCU 子系统配置为预期的低功耗模式，系统才能达到想要的低功耗模式。

例如，只有提前将 MCU 设置为允许待机的 CStop 模式（例如使用 HAL_PWR_EnterStandbyMode()函数），Linux 系统的‘mem’命令才将系统置于待机模式。

所有低功耗模式下可用的唤醒源列表并不相同，因此通过 genPD 框架实现了一种软件机制，以确保低功耗模式和已激活的唤醒源是一致的。

策略是允许 MPU 根据当前已激活的唤醒源进入可用的最深低功耗模式。

示例 1

用户将 UART4 作为唤醒源激活。UART 由 V_{DDCORE} 供电，所以，所有修改 V_{DDCORE} 的低功耗模式自动被禁止。

然后，调用 `echo mem > /sys/power/state`，使 MPU 进入 CStop 模式，允许停止或 LP-Stop 模式。

示例 2

用户选择 RTC 作为唤醒源。无论低功耗模式状态如何，RTC 始终是通电的。

然后，调用 `echo mem > /sys/power/state`，使 MPU 进入 CStop 模式，允许待机模式。

这种机制确保不会发生阻塞情况，比如将 UART4 作为唤醒源激活并请求 SoC 备用。

系统的电源模式是由 MPU 和 MCU 电源状态共同决定的。

下表列出了可能的最深电源模式（根据唤醒源组），还列出了 Linux 的标准低功耗模式和 STM32MP1 系列系统低功耗模式之间的等效性，包括外部(Lp) DDR 电源状态（开、关、自刷新(SR)），因为它是在 STM32MPU OpenSTLinux Distributions BSP 提供的环境中实现的。

使用 32 位 DDR3 接口的系统最有可能在地址和命令信号上使用外部电阻端接。这些系统应采用电阻端接电源（VTT）供电，在低功耗模式下，VTT 对整个系统的功耗起重要作用。

因此，通过使用 STPMIC1x，当 PWR_ON 信号被下拉（LP-Stop、LPLV-Stop、待机）时，可以关闭该电源；而且最好使用 LP-Stop 而不是 Stop 模式，因为 LP-Stop 模式在这种情况下不会切换 PWR_ON。

表 11. 每个唤醒源组的最深电源模式，以及 Linux 和 STM32MP1 系列之间的等效性

唤醒源	Linux 命令	STM32MP1 系列系统最深电源模式	系统 DDR	Linux 内核状态	功耗	唤醒时间	评论/应用指南
组 1：USB、CEC、ETH、USART、I²C、SPI、LPTIM	"mem"	Stop 或 LP-Stop	SR（VTT 关闭）	“Suspend-to-ram”	中	中	LP-Stop：驱动外部 PWR_LP/PWR_ON 允许为外部调节器设计定制策略。典型应用是在 DDR3 32 位设计中关闭 DDR3 端接电源（VTT）
组 2：PVD、AVD、IWDG、GPIO	“mem”	LPLV-Stop	SR（VTT 关闭）	“Suspend-to-ram”	低	中	LPLV-Stop：功率保存特性确保能够节电。适用于带有侵入性功率限制且容忍唤醒源限制的应用（参照表 4. 系统的低功耗模式唤醒能力）
组 3：BOR、Vbat mon、Temp mon、LSE CSS、RTC、TAMP、唤醒引脚	“mem”	待机	SR	“Suspend-to-ram”	低	中	待机节省更多的电能，以唤醒时间为代价
	“关断”	待机或关闭/VBAT	关闭	关断	极低	高	唤醒顺序和应用影响指导待机和关机之闭的选择

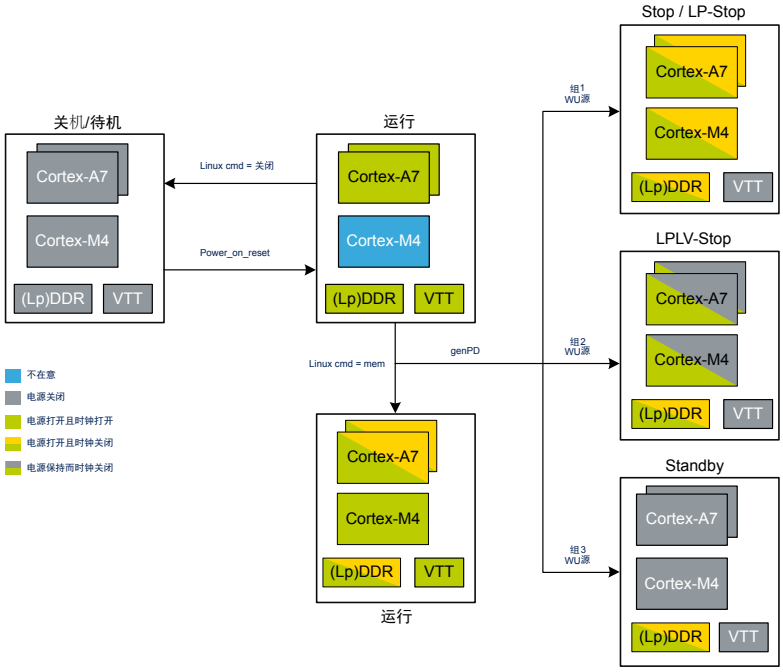
提示

该表假设 Cortex®-M4 在所有预期的停止、LP-Stop、LPLV-Stop 系统电源模式下均处于 CStop 模式，在待机系统电源模式下则处于 CStop（PDDS=1）模式。

下图显示了系统中可用的电源状态转换。图中考虑了上表中使用的每个唤醒源组的相同定义。组 3 唤醒源对于组 2 和组 1 唤醒源也有用。组 2 也兼容组 1。

如果在不同的组中激活了多个唤醒源，则根据上表中所述的层次结构选择低功耗模式。选择 Stop 或 LP-Stop 是通过安全监控设备树中的设置完成的。

图 7. 可用的电源状态转换



4.5.2 Linux 操作系统中的 MCU 电源模式控制

在 Linux 操作系统中，有两种方法通过 MCU 控制电源模式：要么通过进程间通信（IPC）使用“rpmsg”，要么使用“独立集群”。

通过进程间通信（IPC）使用“rpmsg”

MPU 可以看作主处理器，而 MCU 可以看作协处理器。两个处理器之间的通信使用 rpmsg 完成。一个 Linux 内核以 SMP 模式运行在双核 MPU 上，一个 free-rtos 运行在 MCU 上。

一些 IP 在处理器之间共享，并设置了保护以确保寄存器的一致性。当一个核心需要访问另一个核心负责的 IP 时，它通过 rpmsg 发送请求以完成操作。

对于进入/退出待机状态，在 MPU“CStop 且 PDDS=1 CSTBYDIS=1”状态之前，Linux 会通知 MCU：IPC 链路被挂起。当使用 MCU_BEN 从系统待机状态中醒来时，MCU 固件不通过 IPC 与 Linux 通信。它只能在 MPU 需要唤醒时发送一个事件（SEV 指令）到 MPU。然后，Linux 再次与 MCU 建立 IPC 链路。

独立集群

MCU 可以在其软件代码中管理其低功耗模式状态（不需要 MPU 的 rpmsg）。

4.6 低功耗模式退出顺序

根据进入低功耗模式的方式（WFI 或 WFE），可以通过中断或事件触发从低功耗模式退出（待机模式除外）。在 MPU 上退出待机状态只能由应用复位（如 NRST、IWDG）、WKUPx 引脚事件或 RTC 事件触发。

在退出低功耗模式时，可以只唤醒 MPU、只唤醒 MCU，或同时唤醒两者。

- 从低功耗模式（待机模式除外）退出时，仅唤醒 MPU、仅唤醒 MCU 或两者均唤醒取决于 IRQ 检测是如何编程的（在 MPU、MCU 或两者上都可见）。
- 从待机模式退出时，通过在 RCC 引导控制寄存器（RCC_MP_BOOTCR）中设置 MPU_BEN 位和 MCU_BEN 位完成。从待机状态唤醒过程中，BootROM 使用此配置。BootROM 然后唤醒 MPU、MCU，或两者均被唤醒。

提示 *在使用 STM32MPU OpenSTLinux Distributions 时，系统总是只引导一个子系统：我们假设 MCU_BEN 和 MPU_BEN 是排他的，以避免两个子系统并行启动时可能出现的竞争情况。*

- 因此，MCU 应该能够说明一些针对 MPU 子系统的唤醒源。例如，如果唤醒是由于 RTC 警报（Linux 所预期的），那么 MCU 应该唤醒 MPU。

当 STM32MP15x 器件从低功耗模式退出时，重要的是确保所有外部调节器输出电压设置为适当的水平，且在应用运行模式之前保持稳定。

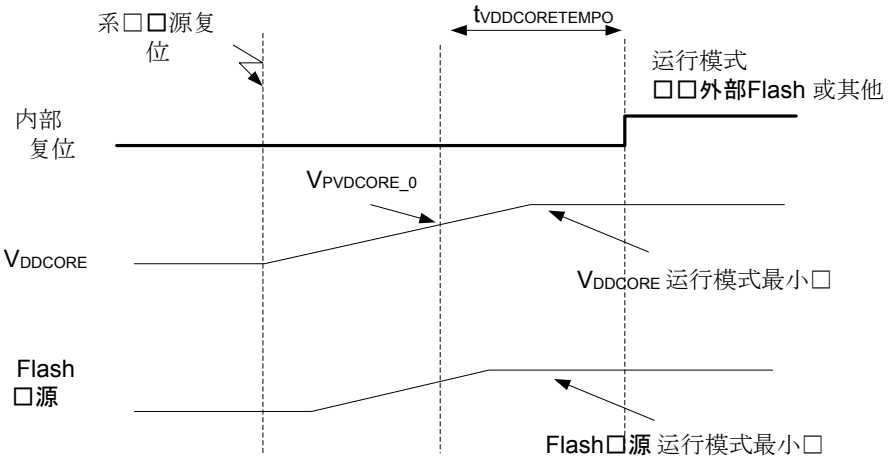
4.6.1 退出系统电源复位

在系统电源复位时，STM32MP1 系列器件监控 V_{DD}，直到其达到指定的 VBOR 阈值。此外，STM32MP15x 器件还监控 V_{DDCORE}，直至其达到电压检测器最小阈值（$V_{PVDCORE\_0}$）；之后的期望情形是：当内部 STM32MP15x 器件复位信号失效后，在 $t_{VDDCORETEMPO}$ 延迟之前正确地建立 V_{DDCORE}。

如要查找 $t_{VDDCORETEMPO}$ 的值，请参照 STM32MP15x 数据手册中的表格“嵌入式复位和电源控制块特性”（$t_{VDDCORETEMPO}$ 最小值是 200 us）。期望的情形是：在 STM32MP15x 器件进入运行模式并从 Flash 读取数据以进行启动之前，外部 Flash 电源准备就绪。

- 当使用 STPMIC1x 时，这是自动处理的，因为在所有供电达到预期值之前，STPMIC1x 不会释放 STM32MP15x 器件 NRST 引脚。
- 当使用分立调节器组件时，设计应确保正确处理这些约束。

图 8. 从系统复位唤醒顺序



4.6.2 退出停止模式

退出停止模式时，电压没有改变，所以没有问题。然而，要注意的是，当 PLL1 和 PLL2 设置因为硬件自动“时钟恢复”特性而被恢复时，其他 PLL 应根据需要重新配置。MCU 根据 HSI 时钟重启（HSI 时钟频率设置与进入停止模式前的设置保持相似）。这同样适用于其他“停止”模式（LP-Stop、LPLV-Stop）。

4.6.3 退出 LP-Stop 模式

从 LP-Stop 模式退出时，V_{DDCORE} 没有改变，然而，系统级别的其他电压可能已经关闭或降低，可能需要设置回各自的运行模式值。

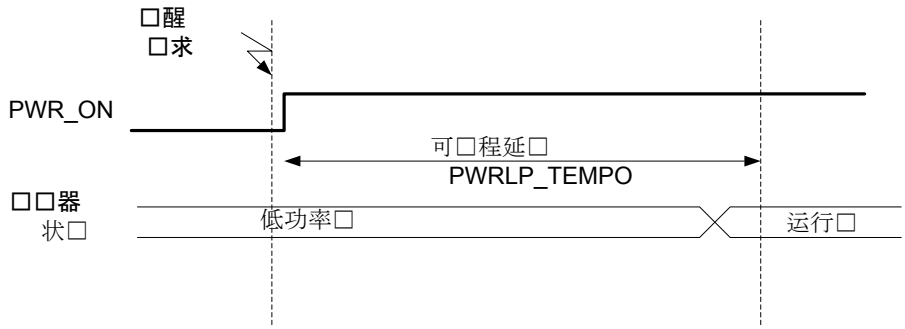
例如，DDR3/DDR3L（x32 位总线宽度）端接电阻电源 VTT 可以在 LP-Stop 模式下关闭。STM32MP15x 器件等待 RCC PWR_LP 延迟控制寄存器（RCC_PWRLPDLYCR）中的可编程延迟（PWRLP_TEMPO），以允许外部调节器运行模式值恢复。

- 当使用 STPMIC1x 时，STM32MP15x 器件的 PWR_ON 引脚会通知退出 LP-Stop 模式 - 该引脚会切换回 1，从而请求改变 STPMIC1x 输出电源的供电水平。STPMIC1x 确保 PWRLP_TEMPO 可以设置为 1000 us 左右（典型条件）的最小值。
  - STM32MP1 最大输出电压上升时间由下面的公式给出：

    对于 BUCK 输出：Max {100 us; V2-V1/3.6 10^{-3} us}（典型）

    其中的 V2: 最小运行模式电压 V1：低功耗模式下的电压。
  - 对于 LDO 输出：升压时间由 LDO 输出上的电流限制和总电容量定义。如果 LDO3 上的总输出电容为 10 µF，典型的升压时间为 20 µs。
  - 如果 VDD_USB 电压为 3.3 V（来自 BUCK 输出），最大升压时间应为 3.3/3.6 10^{-3} = 917 us （典型）。
  - 在最坏情况下，精确值必须从 STPMIC1x 数据手册计算，但如果唤醒时间不重要，建议使用一些合理的余量（例如 5 ms）。
- 当使用分立外部调节器组件时，硬件应该将所需的电源设置为各自的运行模式值（在 PWRLP_TEMPO 延迟范围内）。

图 9. 从 LP-Stop 模式的唤醒顺序



4.6.4 退出 LPLV-Stop 模式

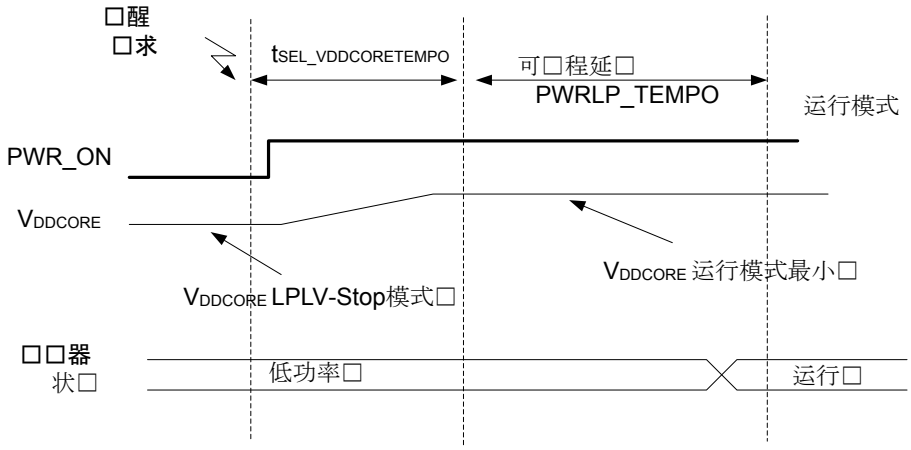
退出 LPLV-Stop 模式时，除了改变电源（就像退出 LP-Stop 模式时一样）之外，在 LPLV-Stop 期间被降低的 V_{DDCORE} 电压需要重新设为其运行模式值。

STM32MP15x 器件等待 $t_{SEL\_VDDCORETEMPO}$ 延迟，在此期间，V_{DDCORE} 应达到其最小运行模式值。在这个 $t_{SEL\_VDDCORETEMPO}$ 延迟之后，STM32MP15x 器件等待一个可编程延迟（PWRLP_TEMPO）（由 RCC PWR_LP 延迟控制寄存器（RCC_PWRLPDLYCR）确定），允许其他外部调节器运行模式值恢复（关于上述的 LP-Stop 模式退出顺序）。

如要查找 $t_{SEL\_VDDCORETEMPO}$ 的值，请参阅相应 STM32MP1 系列数据手册中的表格“嵌入式复位和电源控制块特性”（$t_{SEL\_VDDCORETEMPO}$ 最小值是 234 us）。

- 当使用 STPMIC1x 时，STM32MP1 的 PWR_ON 引脚会通知退出 LP-Stop 模式 - 该引脚会切换回 1，从而请求改变 STPMIC1x 输出电源的供电水平。STPMIC1x 确保：V_{DDCORE} 输出电源在 $t_{SEL\_VDDCORETEMPO}$ 延迟时间内达到其最小运行模式值，而且 PWRLP_TEMPO 可以设为第 4.6.3 节 退出 LP-Stop 模式中提供的相同公式定义的最小值。
  - 如果在 LPLV-Stop 期间，V_{DDCORE} 电源（由 BUCK 输出供电）降低到 0.9 V，然后 V1= 0.9 V 且 V2= 1.2 V，则最大升压时间为 0.3 / 3.6 10^{-3} = 83 us（典型值）。最坏情况下的值应该从 STPMIC1x 数据手册计算，而且应该小于 $t_{SEL\_VDDCORETEMPO}$（234 us）。
  - 同样的公式也适用于其他需要升压时间（在最坏情况下）小于 PWRLP_TEMPO 的电源。
  - 如果唤醒时间不重要，建议使用合理的余量（例如 5 ms）。
- 当使用分立外部调节器组件时，硬件应该能够在 $t_{SEL\_VDDCORETEMPO}$ 延迟时间内将 V_{DDCORE} 供电设为其最小运行模式值，并且应该在 PWRLP_TEMPO 延迟时间内将所需的其他电源设为各自的运行模式值。

图 10. 从 LPLV-Stop 唤醒顺序



4.6.5 退出待机模式

退出待机模式时，除了改变电源（就像在 LP-Stop 模式下）之外，还需要打开 V_{DDCORE} 电压（在待机模式下已关闭）并将其设置为运行模式值。

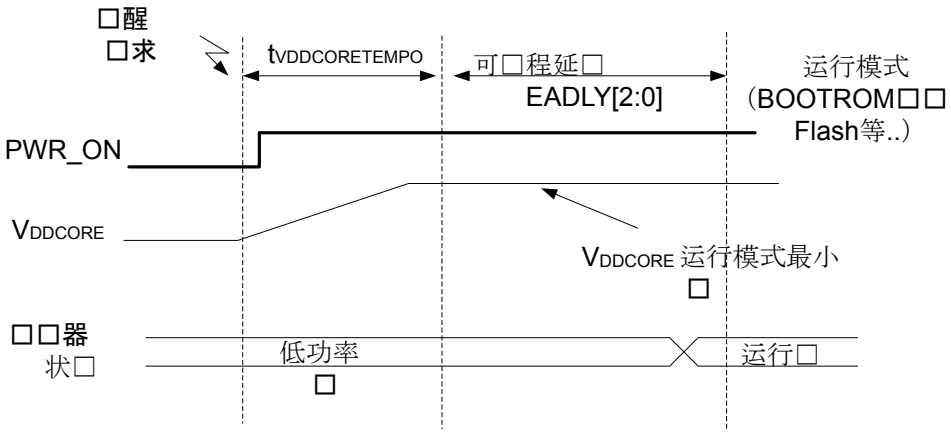
在待机期间，V_{DD} 供电保持活跃；因此在从待机状态唤醒时，引脚 PWR_ON 被设置回 1 (PWR 控制器中的部分数字逻辑由 VDD 供电，因此可以驱动 PWR_ON 引脚)。

STM32MP15x 器件等待 $t_{VDDCORETEMPO}$ 延迟，在此期间，V_{DDCORE} 应达到其最小运行模式值。然后，在 BOOTROM 访问任意外部存储器之前，STM32MP15x 器件等待可编程延迟（EADLY[2:0]）（在 RCC 复位持续时间内，由 LSI 控制寄存器（RCC_RDLSICR）确定），允许所有需要的外部功率调节器运行模式值恢复（外部 Flash 电源或其他）。

如要查找 $t_{VDDCORETEMPO}$ 的值，请参阅相应 STM32MP1 系列数据手册中的表格“嵌入式复位和电源控制块特性”（$t_{VDDCORETEMPO}$ 最小值是 200 us）。

- 当使用 STPMIC1x 时，STM32MP15x 的 PWR_ON 引脚会通知退出待机模式 - 该引脚会切换回 1，从而请求改变 STPMIC1x 输出电源的供电水平。STPMIC1x 器件确保：V_{DDCORE} 输出电源在 $t_{VDDCORETEMPO}$ 延迟时间内达到其最小运行模式值，EADLY[2:0]延迟可以设为第 4.6.3 节 退出 LP-Stop 模式中提供的相同公式定义的最小值。建议为 EADLY[2:0]使用合理的余量 5-10 ms。
- 当使用分立外部调节器组件时，硬件应该能够在 $t_{VDDCORETEMPO}$ 延迟时间内将 V_{DDCORE} 供电设为其最小运行模式值，并且应该在编程确定的 EADLY[2:0]延迟时间内将所需电源设为各自的运行模式值。

图 11. 从待机模式的唤醒顺序



许多系统配置在从待机模式唤醒后被复位：

- 时钟树返回到 HSI 源，因此保留固件（如果存在）应该始终使用 HSI 作为它需要使用的外设的内核时钟。
- 建议保留固件停留在 HSI 上，以避免与 Linux（如果使用）发生时钟树冲突。

4.6.6 从 VBAT 模式退出

当 STM32MP15x 器件处于 VBAT 模式时，只有 VSW 供电是由外部 VBAT 电源维持的。

可用的唤醒源（TAMP、RTC）可激活 PC13 引脚，以将唤醒请求通知外部调节器。然后，外部调节器重新启动供电次序，就像一次系统电源复位。

4.6.7 从 MPU CStop 和 CStandby 模式退出

MPU Cstop（PDDS=1，CSTBYDIS=1）和 CStandby 的区别在于退出模式不同（参见表 5. 电源系统模式 VS 子系统模式总结）。

当系统处于运行、停止、LP-Stop 或 LPLV-Stop 状态时，从 MPU CStandby 退出，程序通过本地 MPU 复位执行重启。这种“重启”会影响 MCU 运行，因为当从外部 Flash 重新加载 MPU 初始化代码时，BootROM 可能会重新初始化一些外设寄存器。请参见表 5. 电源系统模式 VS 子系统模式总结。

如果外部闪存处于 BootROM 不期望看到的模式（例如，如果 SD-Card 设为 UHS-I 模式，而 BootROM 应处于标准模式)，则无法正确执行这种“重启”。由于这些原因，通常建议使用 MPU CStop 模式而不是 MPU CStandby 模式。注意！ST Linux distribution 中没有使用 MPU Cstandby 模式。

如果当系统处于待机状态时从 MPU CStandby 或 CStop（PDDS=1，CSTBYDIS=1）退出，程序执行重新启动的方式与上电复位（加载选项字节，或获取复位向量）后相同。请参见表 5. 电源系统模式 VS 子系统模式总结。在这种退出模式下，MCU 应从复位重新启动，而从 MPU 重新启动没有影响。

提示

在 ST Linux distribution 没有使用 MPU CStandby，因为其唤醒是不确定的（退出 CStandby 是在不重启的情况下完成的，而退出待机意味着一次重启）。

关于硬件约束的详细信息，请参阅应用笔记 *STM32MP1 系列硬件开发入门*（AN5031）。

如果当系统处于停止、LP-Stop 或 LPLV-Stop 模式时从 MPU CStop 退出，程序通过执行中断处理程序重新启动（如果使用 WFI（等待中断）指令或‘Return from ISR’进入低功耗模式），或者在 WFE（等待事件）之后从指令重新启动（如果使用 WFE 指令进入低功耗模式）。如果 MPU 在进入系统待机状态之前被重新启动，该模式是确保独立 MCU 行为的推荐模式。

5 STM32MP1 系列外设的低功耗配置

STM32MP1 系列实现了许多外设。这些外设的配置在降低功耗方面起着重要的作用。本节描述如何配置相关外设。

5.1 外设分配

运行时有三个可用的上下文：

- Cortex-A7 安全
- Cortex-A7 非安全
- Cortex-M4

每个外设可以“分配”给一个或多个（共享）此类上下文。这是由扩展的信任区保护控制器（ETZPC）完成的。

每个外设在复位时（A7、A7 安全、M4 或共享）根据参考手册寄存器内容分配，或者在使用 STM32MPU OpenSTLinux Distributions 起动时分配（参见图 13. 使用 OpenSTLinux distribution 时的 STM32MP1 系列外设分配）。软件稍后可以在运行时更新该默认配置。

当一个上下文希望使用分配给它的外设时，它必须确保已给该外设提供时钟。所以在使用外设之前，MPU 或 MCU 必须启用它，它还可以定义该外设是否在 CSleep 模式下保持活动。该时钟启用由 RCC 完成，被称为从外设到处理器的“分配”。

- MPU 可以为自己和 MCU 分配或取消分配外设。
- MCU 可以为自身分配或取消分配外设，但不能对 MPU 进行此操作。

通过在 MPU 端的 RCC_MP_xxxxENSETR 寄存器上和 MCU 端的 RCC_MC_xxxxENSETR 寄存器上设置专用 PERxEN 位以启用外设。

通过在 MPU 端的 RCC_MP_xxxxENCLRR 寄存器上和 MCU 端的 RCC_MC_xxxxENCLRR 寄存器上设置专用 PERxEN 位以禁用外设。

在 MPU 端的 RCC_MP_xxxxLPENSETRR 寄存器上和 MCU 端的 RCC_MC_xxxxLPENSETR 寄存器上设置专用 PERxLPEN 位，以便在 CSleep 期间激活外设。

在 MPU 端的 RCC_MP_xxxxLPENCLRR 寄存器上和 MCU 端的 RCC_MC_xxxxLPENCLRR 寄存器上设置专用 PERxLPEN 位，以便在 CSleep 期间禁用外设。

使用 STM32Cube_FW_MP1 时，用户可以通过 STM32MP15xx_hal_rcc.h 中定义的专用函数在 CSleep 模式下分配外设和控制外设时钟：

```
__HAL_RCC_xxxx_CLK_ENABLE();
__HAL_RCC_xxxx_CLK_SLEEP_ENABLE();
```

在 MPU 端，IP 驱动通过分配启用各自的时钟。详细信息请参阅相应参考手册（参见表 1. STM32MP1 系列的配置）中的“外设分配”和“通用时钟概念概述”章节。

对于大多数外设，一旦“分配”给处理器，应用程序就可能控制其内核和总线接口时钟的激活/禁用。

RCC 使用外设分配根据 CPU 模式自动控制时钟门控。

根据工作期间供电电压的不同，某些外设可能只提供有限的功能和性能。详细信息请参阅相应数据手册中“通用工作条件”一节。

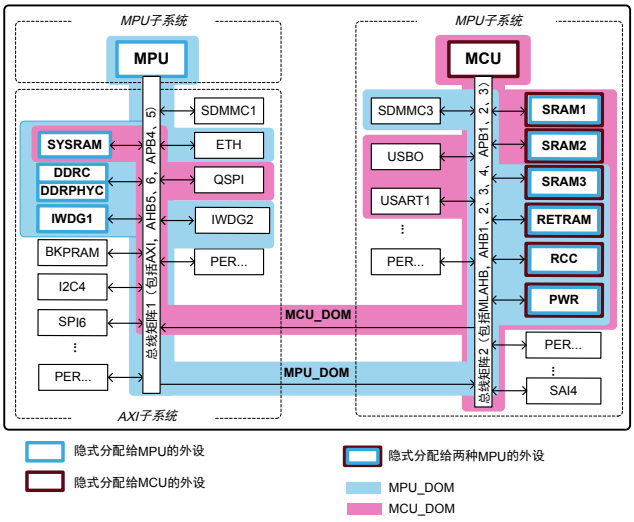
下图给出了一个外设分配示例，需要考虑以下因素：

- MPU 启用 ETH、SDMMC3、SRAM3 和 RETRAM。注意！SYSRAM、DDRC、DDRPHYC 和 IWDG1 被隐式分配给 MPU。MPU、总线矩阵 1、总线矩阵 2 组成的组，以及通过 RCC_MP_xxxx 寄存器分配的外设构成 MPU 外设域（MPU_DOM）。
- MCU 启用 SYSRAM、QSPI、USBO 和 USART1。MPU、总线矩阵 1、总线矩阵 2 组成的组，以及通过 RCC_MC_xxxx 寄存器分配的外设构成 MCU 外设域（MCU_DOM）。

提示 *PWR 和 RCC 是公共资源，隐式分配给 MPU 和 MCU。SRAM1、SRAM2、SRAM3 和 SRAM4 也隐式分配给 MPU 和 MCU，因为如果其中一个处理器处于 CRun 状态，则 MLAHB 桥启用。RETRAM 也隐式分配给 MPU 和 MCU，以便在系统退出待机状态时允许访问。*

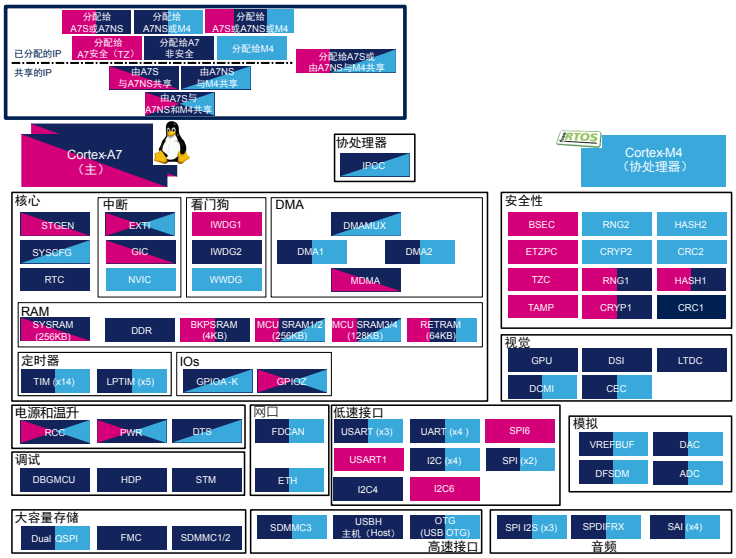
提示 *BKPSRAM 具有专用的使能位，可对总线接口时钟进行门控。在使用使能位之前，需要启用 BKPSRAM。*

图 12. 外设分配示例



如果使用 ST Linux distribution，下图显示了外设列表以及它们被分配给 Cortex-A7 安全、Cortex-A7 非安全、Cortex-M4 或共享上下文的可能性。

图 13. 使用 OpenSTLinux distribution 时的 STM32MP1 系列外设分配



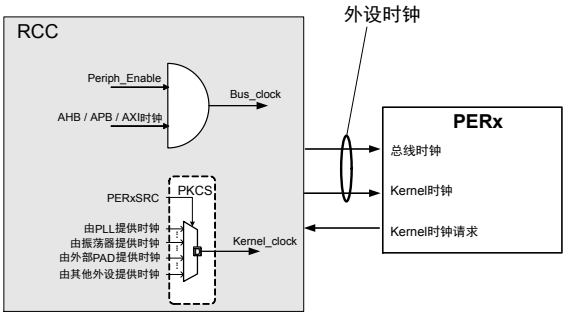
5.2 外设时钟分布

外设时钟是指 RCC 为外设提供的时钟。具体包括两类时钟：

- 总线接口时钟。
- 内核时钟。

下图描述了 STM32MP1 系列器件上的外设时钟分布。

图 14. 外设时钟分布



内核时钟允许外设为外设功能使用不同的时钟频率（与外设总线时钟相比）。

使用 STM32CubeMP1 包时，可以使用 stm32mp1_xx_hal_rcc_ex.h 文件中定义的专用函数来选择外设内核时钟：

```
__HAL_RCC_PERx_CONFIG (__PERxCLKSource__);
```

在 MPU 端，Linux 时钟驱动提供一个可以选择内核时钟的“clk_set_parent”服务。

有关内核时钟分布和外设时钟门控的详细信息，请参阅相应参考手册中的“外设时钟门控控制”一节（参见表 1. STM32MP1 系列的配置）。

5.3 Stop、LP-Stop、LPLV-Stop 外设分配

在 Stop 和 LP-Stop 模式下，使用 LSI 或 LSE 时钟的外设和拥有一个内核时钟请求（从能够在停止模式下运行的 RCC 中选择时钟源）的外设仍然能够运行。

为了让 I2C 或 U(S)ART 在 CStop 或系统 Stop、LP-Stop 模式下工作，用户必须选择一个振荡器作为内核时钟：hse_ker_ck、hsi_ker_ck 或 csi_ker_ck，取决于 RCC PERxEN。所选振荡器在外设生成内核时钟请求时提供给外设。

在 Stop 和 LP-Stop 模式下，如果 xxxKERON=1，振荡器保持激活状态，允许在收到从 Stop 和 LP-Stop 模式中唤醒的请求或外设内核时钟请求时立即提供时钟。

参照表格：功能取决于相应参考手册中的系统工作模式（参见表 1. STM32MP1 系列的配置）。

在 LPLV-Stop 模式下，预期外部电源会降低其电压水平。因此，只有少数外设仍在运行（BOR、PVD、AVD、VBATH-L 监测、TEMPH-L 监测、LSI、LSE、LSE CSS、RTC、IWDG），另外还有一些外设能够唤醒系统（TEMP、GPIO）。参照表格：功能取决于相应参考手册中的系统工作模式（参见表 1. STM32MP1 系列的配置）。

6 调试技巧

本节描述如何在低功耗模式下使用调试特性。

6.1 在低功耗模式下启用调试：低功耗模式仿真

进入低功耗模式后，可关闭处理器时钟、子系统时钟、或核心电源。此操作阻止调试访问相关的无时钟域。STM32MP1 系列支持低功耗模式仿真，在低功耗模式下保持调试能力。这是由 DBGMCU 配置寄存器（DBGMCU_CR）中的 DBGSLEEP、DBGSTOP 和 DBGSTBY 位控制的。

提示 *DBGMCU 寄存器集适用于 MPU 和 MCU 子系统。为了与传统 STM32 MCU 产品保持一致，保留了 DBGMCU 名称。*

全面信息可以在相应 STM32MP1 系列参考手册的以下部分中找到（参见表 1. STM32MP1 系列的配置）：

- 低功耗仿真模式在 RCC 章节中。
- 微控制器调试单元（DBGMCU）在“调试支持”章节中。

提示 *使用低功耗仿真模式对节电有重大影响（各种时钟和电源保持开启状态）。这种模式应仅用于调试目的。当使用低功耗模式仿真时，功耗测量无关紧要。*

表 12. HAL 调试功能

功能	说明
HAL_DBGMCU_EnableDBGSleepMode() HAL_DBGMCU_DisableDBGSleepMode()	在 CSleep 模式下启用/禁用调试模块
HAL_DBGMCU_EnableDBGStopMode() HAL_DBGMCU_DisableDBGStopMode()	在停止模式下启用/禁用调试模块
HAL_DBGMCU_EnableDBGStandbyMode() HAL_DBGMCU_DisableDBGStandbyMode()	在待机模式下启用/禁用调试模块

6.2 使用 HDP 调试低功耗模式

硬件调试端口（HDP）允许观察内部信号，这些信号可用于调试低功耗模式的进入和退出。这些信号可以输出到 STM32MP15x 器件的一些引脚上，很容易（通过一个示波器或逻辑分析仪）监测。参见相应 STM32MP1 系列参考手册（参见表 1. STM32MP1 系列的配置）中的*硬件调试端口*一节，获取有关信号及其混合配置的完整列表。

为了降低功耗，应该在最终应用中禁用 HDP 和相关的 GPIO。

下表描述了用于电源模式调试的最重要信号。

表 13. 监测 CSleep 模式

信号	说明
CA7 STANDBYWFI0 (HDP7, HDP_MUX=0) CA7 STANDBYWFI1 (HDP6, HDP_MUX=0) CA7 STANDBYWFE0 (HDP7, HDP_MUX=1) CA7 STANDBYWFE1 (HDP6, HDP_MUX=1)	CA7 STANDBYWFI0 和 CA7 STANDBYWFI1（面向 WFI）或 CA7 STANDBYWFE0 和 CA7 STANDBYWFE1 （面向 WFE）信号可用于监测 CA7 CSleep 模式。 这些信号表明一个核心是否处于 WFI 或 WFE 状态。
CM4 SLEEPING (HDP4, HDP_MUX=1)	CM4 SLEEPING 信号可用于监测 CM4 CSleep 模式（WFI 或 WFE）。

表 14. 监测 CStop 模式和停止模式的进入

信号	说明
RCC pwrds_mcu (HDP2, HDP_MUX=6) RCC pwrds_mpu (HDP1, HDP_MUX=6)	RCC pwrds_mcu（或 RCC pwrds_mpu）信号可用于监测 CM4 CStop 模式（或 CA7 Cstop 模式）。
RCC pwrds_sys (HDP3, HDP_MUX=6)	RCC pwrds_sys 信号可用于监测 STM32MP15x 器件停止模式。

表 15. 监测从停止模式退出

信号	说明
pwrwake_sys (HDP0, HDP_MUX=0)	Pwrwake_sys 信号可用于监测 EXTI 已经从外设接收到唤醒中断。
pwrwake_mpu (HDP2, HDP_MUX=0) pwrwake_mcu (HDP1, HDP_MUX=0)	Pwrwake_mpu（或 pwrwake_mcu）信号可用于监测 CA7 子系统（或 CM4 子系统）已收到请求返回 CRun 模式。

裸跑系统HDP 初始化示例代码

```
void HPDConfig(void) {
GPIO_InitTypeDef GPIO_InitStruct;
/*
* HDP AF2 (AF0)
* 0 PI12 (PA4)
* 1 PI13 (PC6)
* 2 PJ5 (PE13)
* 3 PJ6 (PE15)
* 4 PK1 (PC7)
* 5 PK2 (PD3)
* 6 PK5 (PB8)
* 7 PK6 (PB9)
*/

/* 为 GPIOI GPIOJ 和 GPIOK 提供时钟 */
__HAL_RCC_GPIOI_CLK_ENABLE();
__HAL_RCC_GPIOJ_CLK_ENABLE();
__HAL_RCC_GPIOK_CLK_ENABLE();
/*** 将 HDP 输出路由到 GPIO ***/
/* AF2,PI12 -> HDP0 */
/* AF2,PI13 -> HDP1 */
GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Alternate = GPIO_AF2_HDP;
HAL_GPIO_Init(GPIOI, &GPIO_InitStruct);

/* AF2,PJ5 -> HDP2 */
/* AF2,PJ6 -> HDP3 */
GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6;
HAL_GPIO_Init(GPIOJ, &GPIO_InitStruct);

/* AF2,PK1 -> HDP4 */
/* AF2,PK2 -> HDP5 */
/* AF2,PK5 -> HDP6 */
/* AF2,PK6 -> HDP7 */
GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_5|GPIO_PIN_6;
HAL_GPIO_Init(GPIOK, &GPIO_InitStruct);

/* 为 HDP 提供时钟 */
__HAL_RCC_HDP_CLK_ENABLE();

/*** 为 HDPx 输出选择信号 ***/
/* 选择 CM4 SLEEPDEEP 信号作为 HDP0 输出 */
MODIFY_REG(HDP->HDP_MUX, HDP_MUX_MUX0, (HDP_MUX_MUX0_1 ));
/* 选择 RCC pwrds_mpu 信号作为 HDP1 输出 */
MODIFY_REG(HDP->HDP_MUX, HDP_MUX_MUX1, (HDP_MUX_MUX1_6 ));
/* 选择 RCC pwrds_mcu 信号作为 HDP2 输出 */
MODIFY_REG(HDP->HDP_MUX, HDP_MUX_MUX2, (HDP_MUX_MUX2_6 ));
/* 选择 RCC pwrds_sys 信号作为 HDP3 输出 */
MODIFY_REG(HDP->HDP_MUX, HDP_MUX_MUX3, (HDP_MUX_MUX3_6 ));
/* 选择 CM4 SLEEPING 信号作为 HDP4 输出 */
MODIFY_REG(HDP->HDP_MUX, HDP_MUX_MUX4, (HDP_MUX_MUX4_1 ));
/* 使能 HDP */
SET_BIT(HDP->HDP_CTRL, HDP_CTRL_EN);
}
```

6.3 Linux 调试提示

查看时钟概要，使用：cat /sys/debug/kernel/clk/clk_summary。

- 该命令用频率和状态分层方式显示时钟树。

使用‘devregs’命令监测电源状态标志。

- PWR_MPUCR：包含 MPU 的停止标志。
- PWR_MCUCR：包含 MCU 的停止标志。
- RCC_MP_RSTSR：包含复位原因，包括待机和 Cstandby。

如果系统没有唤醒，检查是否已启用至少一个唤醒源的，唤醒外设是否已由 HSI 或 HSE 时钟提供时钟，是否已设置唤醒引脚极性和上拉/下拉。

如果系统没有进入低功耗模式：

- 检查在调用该模式时，唤醒事件是否尚未挂起。
- cat /proc/interrupts 可以提供此类事件的迹象。
- 检查 EXTI 设置。

如果总体功耗过高：

- 检查时钟是否在不使用外设时被释放。可用运用 clk_summary 命令进行该检查。

版本历史

表 16. 文档版本历史

日期	版本	变更
2019 年 1 月 30 日	1	初始版本。
2019 年 2 月 11 日	2	更新了： • 第 4.5.1 节 Linux 电源命令映射到 STM32MP1 系列电源模式 • 表 11. 每个唤醒源组的最深电源模式，以及 Linux 和 STM32MP1 系列之间的等效性
2019 年 2 月 18 日	3	更新了 表 6. 系统低功耗模式总结

目录

表一览

图一览