AN13104

RT1170 时钟和低功耗特性

Rev. 0 — January 4, 2021 Application Note

by: NXP Semiconductors

目录

1 介绍

i.MX RT1170 是一款突破性的跨界处理器，它把最高运行频率提高到了 1 GHz，而且还在结合了高性能运算和多媒体功能的同时，增强了可用性及实时功能。双核 i.MX RT1170 分别 以 1 GHz，400 MHz 的频率在 Arm® Cortex®-M7 和 Arm Cortex-M4 上运行，同时还兼具良好的安全性。这款处理器可以在不同的温度下使用，以及在不同的领域，尤其在工业及汽车行业。

本文档不但讨论了 RT1170 的时钟和低功耗特性，而且还介绍了一些 debug 和应用技巧来应对低功耗情况下的开发。

2 RT1170 的电源域

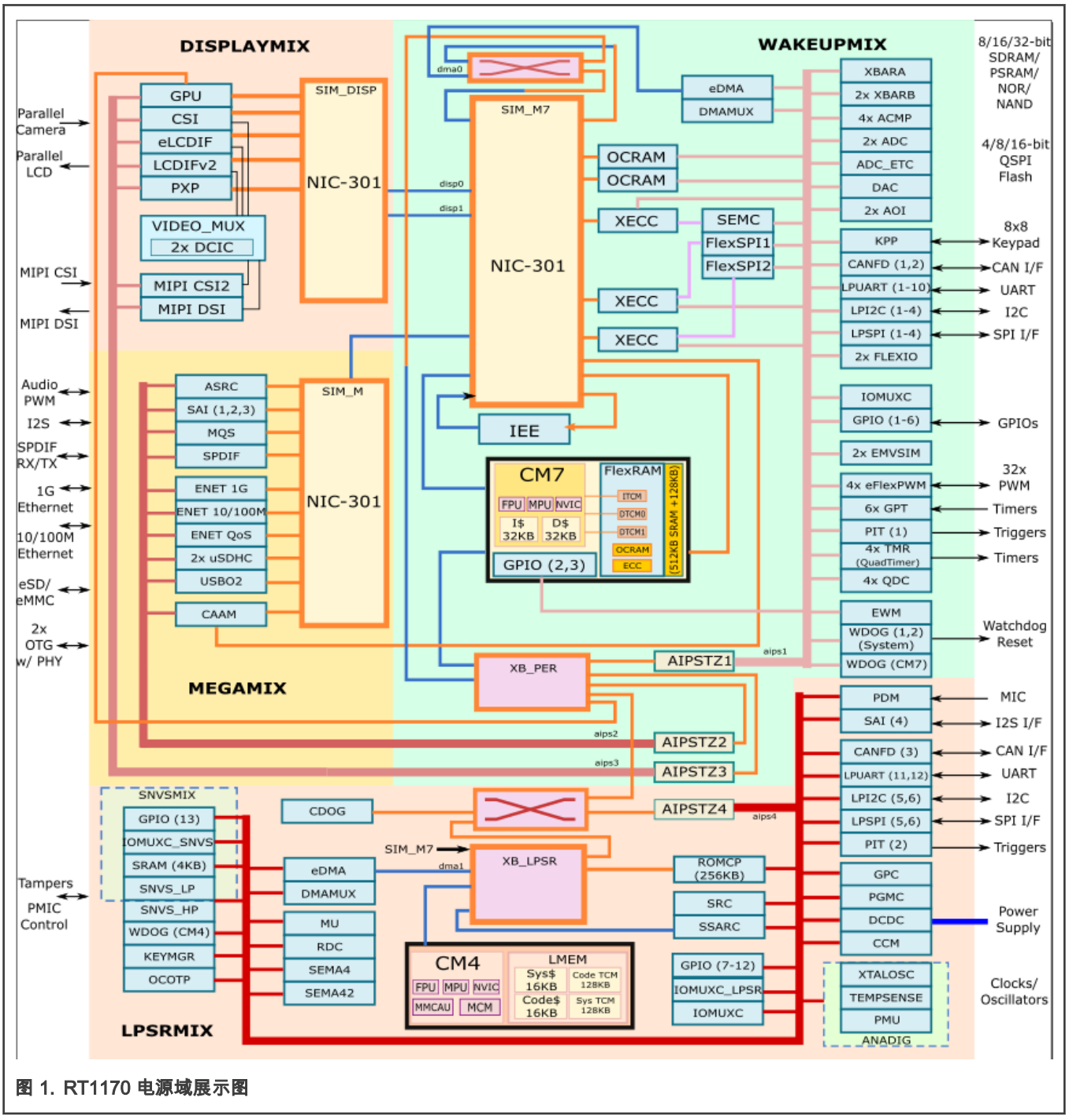


图 1. RT1170 电源域展示图

RT1170 的外部设备根据特性和功能分配到不同的电源域。SoC 域由 DCDC 供电，它包括 WAKEUP MIX, MEGA MIX 以及 DISP MIX。如果 DCDC 关闭，这些 MIX 和其外设都会断电关闭。同时，WAKEUP MIX 还区别于其他的两种 MIX，那就是 WAKEUP MIX 是由 DCDC 直接供电，二者之间没有电源开关（Power Gate），但是其他的两个都有专门的电源开关。LPSR 域包含了一些外设和 CM4 平台（CM4 Platform），而且 CM4 有自己的功能域。表 1 给出了不同电源域的详细描述。

表 1. 不同电源域的外设列表

域	描述
CM7	Cortex-M7 核以及其平台
CM4	Cortex-M4 核以及其平台
WAKEUP	EWM, WDOG1, WDOG2, WDOG3, ACMP1, ACMP2, ACMP3, ACMP4, ADC1, ADC2, DAC, KPP, LPUART1, LPUART2, LPUART3, LPUART4, LPUART5, LPUART6, LPUART7, LPUART8, LPUART9, LPUART10, CANFD1, CANFD2, PIT1, GPT1, GPT2, GPT3, GPT4, GPT5, GPT6, Quad Timer1, Quad Timer2, Quad Timer3, Quad Timer4, FlexPWM1, FlexPWM2, FlexPWM3, FlexPWM4, GPIO1, GPIO2, GPIO3, GPIO4, GPIO5, GPIO6, LPI2C1, LPI2C2, LPI2C3, LPI2C4, LPSPI1, LPSPI2, LPSPI3, LPSPI4, FlexIO1, FlexIO2, EMVSIM1, EMVSIM2, FlexSPI1, FlexSPI2, SEMC, eDMA, DMAMUX, ADC_ETC, XBAR1, XBAR2, XBAR3, IOMUXC, MTR, QNC1, QNC2, QNC3, QNC4, AOI1, AOI2, OCRAM1, OCRAM2, XECC, IEE
LPSR	CANFD3, LPI2C5, LPI2C6, LPUART11, LPUART12, GPIO7, GPIO8, GPIO9, GPIO10, GPIO11, GPIO12, MIC, PIT2, WDOG3, SAI4, SEMA4, SEMA42_1, SEMA42_2, GPC, SRC, MU, IOMUXC_LPSR, DMAMUX, eDMA_LPSR, SSARC
MEGA	ENET1G, ENET, ENET QoS, uSDHC1, uSDHC2, USB_OTG1, USB_OTG2, SPDIF, SAI1, SAI2, SAI3, MQS, ASRC, CAAM
DISPLAY	GC355, PXP, LCDIF, LCDIF v2, CSI, MIPI DSI, MIPI CSI

3 RT1170 中的电源状态

RT1170 中有三种电源状态，分别是 CPU 模式，设点模式（Setpoint，SP），待机模式（Standby，STBY）,以下给出了几个例子:

- Run Mode, SP1, No STBY
- Wait Mode, SP5, STBY
- Stop Mode, SP10, No STBY
- Suspend Mode, SP1, STBY

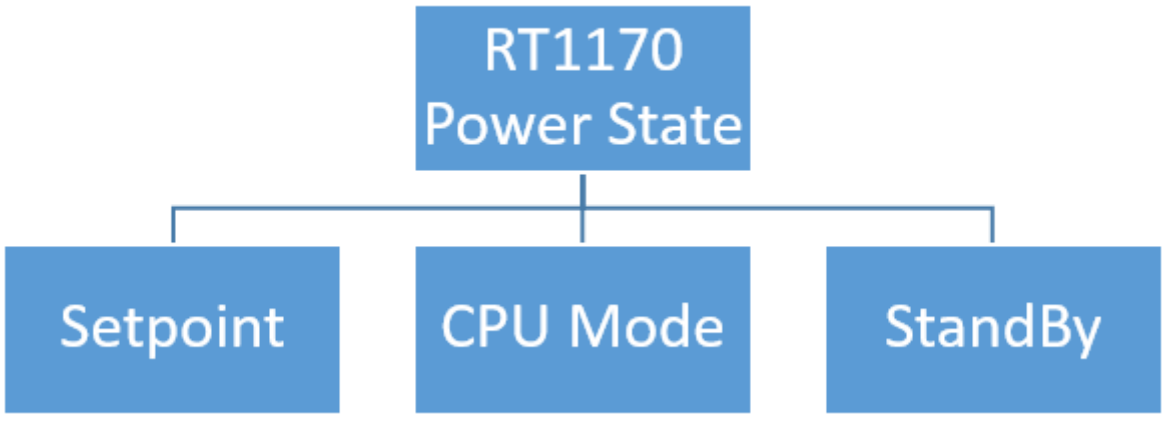


图 2. RT1170 电源模式

3.1 CPU 模式

每一个 CPU 平台都有它自己的电源模式。RT1170 的 CPU 模式有四种：Run, Wait, Stop 和 Suspend。表 2 给出了这四种模式的不同：

表 2. CPU 四种模式的区别

Power mode	CPU	CPU clock	CPU power	M7 cache	M4 cache	M7 TCM	M4 TCM	On-platform peripheral[1]	Exit time latency
RUN	Run	ON	ON	ON	ON	ON	ON	ON	—
WAIT	WFI/WFE	OFF	ON	Clock OFF	ON	ON[2]	ON[2]	ON	Extremely short
STOP	WFI/WFE	OFF	ON[2]	Clock OFF	ON	ON[2]	ON[2]	ON	Short
SUSPEND	WFI/WFE	OFF	OFF[2]	Power OFF	ON	OFF[2]	OFF[2]	OFF	Long

1. 与 CPU 电源相关的平台上的外设。这表示了比如在 M7 平台上有一个 fast GPIO, 通常情况下 GPIO 是可以作为整个系统的唤醒源，但是如果 CPU 进入了断电模式（比如在 Suspend 模式下），fast GPIO 就不可以作为唤醒源了。
2. 可以通过 PGMC 设置。

从 CPU 方面来说，最大的区别就是 CPU 的状态。除了 Run 模式，CPU 会进入 WFI 或者 WFE 模式然后等待唤醒信号，同时，CPU 的时钟和电源都会进入对应的的模式，这些都会影响唤醒时间。在 Suspend 模式下，CPU 会关闭电源，然后导致需要花比 Wait 和 Stop 模式更多的时间去唤醒芯片。

3.2 设点模式（SP）

设点模式（SP）是 RT1170 的一项新功能，它通常可以用来设置时钟还有供电。在 RT1170 中一共有 16 个 SP，这就意味着一共有 16 组关于时钟和电源的设定。

SP 可以通过硬件使能或者关闭时钟源。例如，ARM PLL 需要在 SP1 开启但是在 SP9 关闭，当 SP 从 1 切换到 9 的时候，ARM PLL 就会关闭。反之，如果从 9 切换到 1，ARM PLL 就会开启。这些过程都由硬件来实现不需要软件介入。除了 RC16M (OSCPLL0)，其余的 28 个时钟源都可以通过 SP 来控制。RC16M 常规下一直开启不能关闭，只能在 STBY 模式下关闭。

表 3. SP 不同 SP 下的时钟源设置

Clock source	SP0	SP1	SP7	SP9
ARM PLL	ON	ON	OFF	OFF
PLL1G	OFF	ON	OFF	OFF
PLL528	ON	ON	OFF	OFF
RC16M	ON	ON	ON	ON

比如，在 SP1 的时候，M7 时钟运行的频率是 1 GHz, 它的时钟源是 PLL1G, DIV 是 0。当 SP 切换 SP0 的时候，时钟源就变成了 ARMPLL, 运行的频率在 700 MHz。如果开发人员想要 SP 在 350 MHz 下运行，DIV 可以设置成 1，这些操作都由 SP 完成。

表 4. 不同 SP 下的 Clock_Root

	SP0		SP1		SP2		SP3	
Clock root	Source	Divider	Source	Divider	Source	Divider	Source	Divider
CM7	ARM_PLL	1	SYS_PLL1	1	SYS_PLL1	1	SYS_PLL1	1
CM4	PLL3	2	RC400	2	RC400	2	RC400	4

Table continues on the next page...

表 4. 不同 SP 下的 Clock_Root（续）

	SP0		SP1		SP2		SP3	
Clock root	Source	Divider	Source	Divider	Source	Divider	Source	Divider
BUS	RC400	2	PLL3	2	PLL3	2	PLL3	2
BUS_LPSR	PLL3	4	PLL3	3	PLL3	4	RC400	4

除了时钟外，SP 还可以控制电源的设置。比如可以在 SP1 中 CPU 在 1GHz 的频率运行时使能 FBB。SP 还可以控制 DCDC，LDO 以及电源域的开关。还有其他的一些内存电源，像 TCM 和 LMEM，都可以通过 SP 来控制。

SP 还用来控制 CCM, GPC, SRC, SSRAC, PGMC, DCDC 和 PMU 这些模块。SP 不仅可以用来控制外设的状态，还可以设定具体的数值。比如 DCDC，SP 可以调节每个 SP 上 DCDC 的电压。

3.3 待机模式(STBY)

STBY 模式是 CPU 的第三种低功耗模式。STBY 模式通过以下操作达到更省电的目的：

- 根据 STBY 模式下的 SP 设置进行动作。
- 模拟器件部分进入 STBY。
- 停止内部总线。
- 除了唤醒源其他的系统都停止。

当芯片进入 STBY 模式时，DCDC_IN, LPSR_IN 和 SNVS_IN 的总电流只有几毫安。除了 STBY 模式外，其他模式的电流值都会远大于这个值。

在进入 STBY 模式之前，两个 CPU 都要进入低功耗模式，它们可以是 Wait, Stop 或者 Suspend 模式。如果任意一个 CPU 没有进入低功耗模式，那么芯片就不会进入 STBY 模式。STBY 的要求只是两个 CPU 都进入低功耗状态而不关心具体进入的是哪个低功耗模式。

4 调试与应用技巧

4.1 时钟输出（Clock Output）

有时候用户需要检查一个时钟或 PLL 是否正常工作，包括开启和关闭状态，在 Run mode 或者低功耗模式下的频率。 时钟输出管脚可以解决这个问题，在 RT1170 中有两个输出管脚，CLKO1 和 CLKO2。这些管脚可以将时钟信号在管脚上输出后，使用示波器测量这些信息。所以，强烈推荐用这两个输出管脚来确认时钟源的工作状态。

- `CLKO1:GPIO_EMC_B1_40`

  `CLKO1` 可以输出以下时钟:

  - kCLOCK_OscRc48MDiv2
  - kCLOCK_Osc24Mout
  - kCLOCK_OscRc400M
  - kCLOCK_OscRc16M
  - kCLOCK_SysPll2Pfd2
  - kCLOCK_SysPll2Out
  - kCLOCK_SysPll3Pfd
  - kCLOCK_SysPll1Div5

- `CLKO2:GPIO_EMC_B1_41`

  `CLKO2` 可以输出以下时钟:

— kCLOCK_OscRc48MDiv2

— kCLOCK_Osc24Mout

— kCLOCK_OscRc400M

— kCLOCK_OscRc16M

— kCLOCK_SysPll2Pfd3

— kCLOCK_OscRc48M

— kCLOCK_SysPll3Pfd1

— kCLOCK_AudioPllOut

因为管脚的最大输出频率有限，建议将时钟进行分频后输出。以 CLKO2 输出 Audio PLL 为例：

1. 初始化管脚:

```
IOMUXC_SetPinMux(IOMUXC_GPIO_EMC_B1_41_CCM_CLKO2,0U);
```

2. 设置一个时钟源和分频系数，Audio PLL 通常工作在较高的频率，因此需要选择一个合适的分频系数。

```
rootCfg.mux = 7;  //Select kCLOCK_AudioPllOut
rootCfg.div = 20; // Divison factor is 20
CLOCK_SetRootClock(kCLOCK_Root_Cko2, &rootCfg);
```

3. 最后，如果 Audio PLL 已经使能开启，输出的波形可以通过示波器监测到。

这里有一个限制：当第一个时钟源处于关闭状态，并且尝试切换到输出其他时钟源时就会产生失败，比如:

把所有的时钟源从 RC48M_DIV2 切换到 OSC24M。这一步是因为大部分时钟的默认时钟源就是 RC48M_DIV2，然后关闭 RC48M:

```
ANADIG_OSC->OSC_48M_CTRL &= ~ANADIG_OSC_OSC_48M_CTRL_TEN_MASK;
```

如果首先配置 **rootCfg.mux = 0** 作为时钟源，接着将配置改为 **rootCfg.mux = 1**。会发现即使 OSC24M 时钟是正常工作状态，示波器也检测不到波形。

4.2 时钟观测器(CLK OBS)

RT1170 有一个时钟观测器，它就像示波器一样可以检测时钟的频率。这个功能在开发与低功耗相关的应用时非常有用。

```
void PrintSystemClocks(void);
```

这个 API 可以打印 M7_Clock_Root 的频率：

uint32_t CLOCK_GetFreqFromObs(uint32_t obsSigIndex, uint32_t obsIndex)

以 Audio PLL 为例：

```
uint32_t CLOCK_GetFreqFromObs(uint32_t obsSigIndex, uint32_t obsIndex)
```

Audio PLL 的频率可以通过这个 API 获得。

这个 OBS 不仅可以使用软件获取频率，还可以将时钟信号输出到管脚上。这对于低功耗模式下的 debug 非常有用。

4.3 CPU 的电源模式和 SP 状态

在开发低功耗模应用时，CPU 模式和 SP 的值通常会根据不同的要求而改变。有一些寄存器可以检测并记录这些变化。在 GPC 中有一个 CPU 模式的寄存器，这是一个只读寄存器。位[1:0]可以获得现在的 CPU 状态，位[3:2]可以获得之前的 CPU 状态。

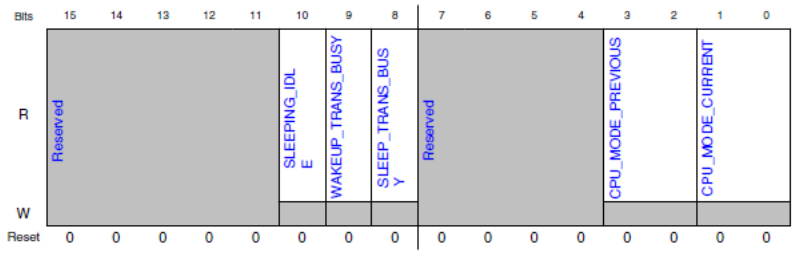


图 3. CPU 模式状态寄存器

SP 也有这样的寄存器，在 debug 的时候有一种情况可能会引起问题。那就是要想得到预期的目标值，SP 传输需要 M7 和 M4 都传输完成才可以，下面是一个例子:

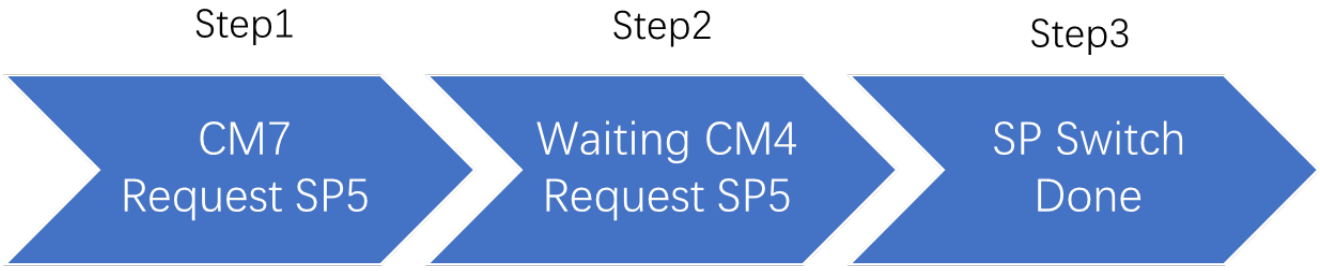


图 4. SP 转换步骤

图 4 显示了从 SP1 转换成 SP5 的一个简单过程。当芯片在 SP1 时，CM7 向 SP5 发出一个请求，然后立马检查现在和之前的寄存器，现在寄存器的值仍然是 SP1，因为 SP 的转换需要 M7 和 M4 都完成。直到第三步，数值才转换完成。如果 CM4 先向 SP 发出请求，所得到的结果跟上述是一样的。

4.4 芯片是否进入 STBY 模式

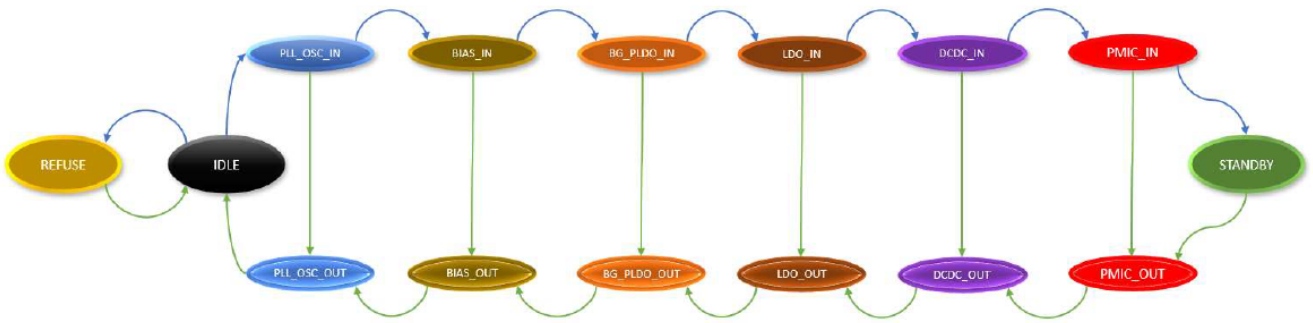


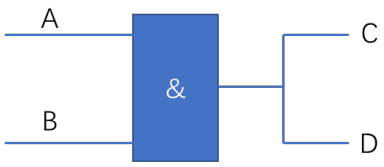
图 5. Standby 流程图

根据 图 5 所示，STBY 进程的最后一步是 PMIC_IN，这表示当 PMIC_IN 进入了 STBY，整个进程就完成了。PMIC_STBY_REQ 管脚可以指示 PMIC 已经进入了 STBY。当系统进入了 STBY 模式的时候，PPC 会给 PMIC_STBY_REQ 发出一个信号，接着这个引脚会输出一个高电平信号。因此，这个管脚可以用于检测芯片是否进入了 STBY 模式。需要注意的是，如果有电阻或者设备连接到了这个管脚，将会产生额外的功耗作用在 SNVS_IN 上。在 EVK 板上，因为管脚会连接到一些设备，所以 SNVS 域会有更多的电量消耗。

还有一个办法来检测系统是否进入了 STBY 模式。RC16M 常规是打开的时钟源，仅在系统进入 STBY 模式之后才能关闭。因此可以使用 OBS 或者 Clock Output 功能观测这个时钟源，如果 RC16M 关闭了，就说明系统进入了 STBY 模式。

4.5 握手（Handshake）

握手的目的是为了确保在 CPU 开始执行触发低功耗 WFI 指令之前，所有的外设都应该完成传输并且没有新的传输开始，以确保安全的进入低功耗模式。如果一个外设没有被分配到一个域里，那么需要 2 个寄存器来完成握手。Stop_req 需要从 IOMUXC_GPR 和 IOMUXC_LPSR_GPR 中发送出来，这时停止接收信号才能被确认。



A	B	C	D
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

A：CAN1_STOP_REQ from IOMUXC_LPSR_GPR35

B：CAN1_STOP_REQ from IOMUXC_GPR70

C：CAN1_STOP_ACK from IOMUXC_LPSR_GPR40

D：CAN1_STOP_ACK from IOMUXC_GPR75

NOTE: Assuming the peripheral completed transmissions

图 6. 握手

以 CAN1 为例，CAN1_STOP_REQ 需要从 IOMUXC_LPSR_GPR35 bit[9]和 IOMUXC_GPR70 bit[9]发送：

```
IOMUXC_GPR->GPR70 |= 1<<9;
IOMUXC_LPSR_GPR->GPR35 |= 1<<9;
```

然后检查传输是否完成，检查 IOMUXC_LPSR_GPR40 bit[3] 和 IOMUXC_GPR75 bit[3]：

```
while((IOMUXC_GPR-->GPR75 & 0x8) == 0);      //检查 bit3
while((IOMUXC_LPSR_GPR-->GPR40 & 0x8) == 0); //检查 bit3
```

如果一个外设被指令到一个域，那么只有一个 stop_req 需要被发送。以 CAN1 为例，它可以分配到 CM4 域，CAN1_STOP_REQ 只需要从 IOMUXC_LPSR_GPR35 bit[9]中被发送：

```
IOMUXC_LPSR_GPR->GPR35 |= 1<<9;
```

然后，通过检查 IOMUXC_LPSR_GPR40 bit[3]来检查传输是否完成。

```
while((IOMUXC_LPSR_GPR->GPR40 & 0x8) == 0); //Check bit3
```

在进入低功耗模式之前，请确保所有的外设都完成了传输。

除了这些，对于握手的序列需要注意的是，DMA/ENET（Bus Master）需要首先完成握手，然后才是 Flash/内存（Bus Slave）。

4.6 关闭 OCOTP

关闭 OCOTP 可以减少电量的消耗。在大多数情况下，OCOTP 可以在系统启动后关闭，但是当系统应用了与安全相关的功能时，请谨慎关闭 OCOTP，关闭电源需要格外注意：

```
OCOTP->HW_OCOTP_PDN |= OCOTP_HW_OCOTP_PDN_PDN0_MASK;
```

4.7 SNVS_XX Pin Leakage

RT1170 的 SNVS_xx 管脚可以用作于篡改管脚。默认下，SNVS_xx 管脚是输入的，而且内部开关电阻是关闭的。所以如果 SNVS_xx 管脚是悬空的，那么被 SNVS_xx 驱动的内部逻辑也是悬空的，这样一来就会产生 leakage。所以使能内部上拉电阻可以避免这个问题：

```
IOMUXC_SNVS_GPR->GPR37 |= IOMUXC_SNVS_GPR_GPR37_SNVS_TAMPER_PUE_MASK;
```

4.8 使能 DCDC DCM 模式

DCDC 模块支持连续（CCM）和断续（DCM）工作模式。在低功耗的情况下，用 DCM 模式会提高 DCDC 的效率：

具体配置：

- `REG1[RLOAD_REG_EN_LPSR] = 1'b0`
- `REG0[PWD_ZCD] = 1'b0`
- `REG3[DISABLE_IDLE_SKIP] = 1'b0`
- `REG3[DISABLE_PULSE_SKIP] = 1'b0`
- `REG0[PWD_CMP_OFFSET] = 1'b0`
- `REG2[LOOPCTRL_EN_RCSCALE] = 3'b101`
- `REG1[3] = 1’b1`

可以把 DCDC 的低功耗模式设置为比运行模式低一个 step，比如，如果运行模式的电压是 1.0 V 和 1.8 V，就需要设置低功耗模式为 0.975 V 和 1.775 V。还要把 ANA_STYBY_TRG_SP[3:0] 和 DIG_STBY_TRG_SP[3:0] 设置成比运行模式低一点（例子同上）。

4.9 唤醒源的配置

在低功耗模式指令 WFI 之前，唤醒源需要被配置。唤醒源可以在低功耗模式下唤醒芯片。请注意握手步骤时，所有的外设传输需要被完成。假设握手完成，外设已经完成初始化，用 GPIO 和 GPT 作为唤醒源举例。

4.9.1 GPIO 作为唤醒源

```
/* Enable GPIO pin interrupt */
GPIO_EnableInterrupts(APP_WAKEUP_BUTTON_GPIO, 1U << APP_WAKEUP_BUTTON_GPIO_PIN);
/* Enable the Interrupt */
EnableIRQ(APP_WAKEUP_BUTTON_IRQ);
/* Mask all interrupt first */
GPC_DisableAllWakeupSource(GPC_CPU_MODE_CTRL);
/* Enable GPC interrupt */
GPC_EnableWakeupSource(APP_WAKEUP_BUTTON_IRQ);
```

图 7. 配置 GPIO 唤醒源

1. 使能管脚中断。
2. 使能 GPIO IRQ。
3. 关闭所有在 GPC 中的唤醒源。
4. 使能在 GPC 中的 GPIO 作为唤醒源。

4.9.2 GPT 计时器作为唤醒源

```
/* Enable GPT Output Compare1 interrupt */
GPT_EnableInterrupts(EXAMPLE_GPT, kGPT_OutputCompare1InterruptEnable);

/* Enable at the Interrupt */
EnableIRQ(GPT_IRQ_ID);
GPC_DisableAllWakeupSource(GPC_CPU_MODE_CTRL);
/* Enable GPC interrupt */
GPC_EnableWakeupSource(GPT_IRQ_ID);
```

图 8. 配置 GPT 唤醒源

1. 使能 GPT 输出比较中断。
2. 使能 GPT IRQ。
3. 关闭所有在 GPC 的唤醒源。
4. 使能 GPC 中的 GPT 作为唤醒源。

4.10 芯片不能进入低功耗模式

导致 CPU 不能进入低功耗模式的大部分情况是由于以下原因：

- 中断挂起（a pending interrupt）

  中断挂起会阻止 GPC 进入低功耗模式。以下方法可帮助检查是否存在中断挂起。

  — GPC 寄存器中：CM_IRQ_WAKEUP_STAT_x

  x 表示 CM_IRQ_WAKEUP_STAT 寄存器的系数。在 GPC 中总共有 8 个寄存器. 也就是说在双核系统中, 16 个 CM_IRQ_WAKEUP_STAT 寄存器需要检查，在单核系统中，8 个寄存器需要被检查。

  — GPC 寄存器中：CM_NON_IRQ_WAKEUP_STAT

  当一个 debugger 连接到一个芯片，它可能会产生一个中断挂起去阻止芯片进入低功耗模式。

  通过这些寄存器可以检查系统是否进入了低功耗模式。如果中断挂起出现了，

  1. 找到是哪个操作或者外设产生了这个中断.
  2. 关闭和清零这个中断状态。

- 握手

  如果用轮询的方法来检查握手，系统有可能因为未完成传输而不能进入低功耗模式。

4.11 进入 SNVS 模式

SNVS 模式将通过关闭内部 DCDC 的方法，关闭除了 SNVS 域外所有的电源域。软件和硬件都可以进入 SNVS 模式。

- 软件模式：

```
SNVS->LPCR |= SNVS_LPCR_TOP_MASK;
```

- 硬件模式：

  按住 ONOFF 按钮大约 5 秒。芯片会进入 SNVS 模式。

4.12 从 SNVS 模式中唤醒芯片

任何可以产生中断的外设都可以作为唤醒源将芯片从 SNVS 模式中唤醒。比如，唤醒管脚（Wake Up Pin），当有一个输入信号被接收到时，它可以作为一个 GPIO 然后产生一个中断。对于 RT1170，一些处于 SNVS 域的 tamper pin 也可以作为 GPIO，这就意味着管脚也可以唤醒芯片。除了 GPIO，RTC 也可以在 SNVS 模式中作为唤醒源。

ONOFF 按钮是 SNVS 模式中的另一个唤醒源。按住 ONOFF 可以唤醒芯片。

4.13 在低功耗模式下外设的状态

通常一个外设需要在两个基本条件下工作，时钟和供电。这个两个条件可以判断一个外设是否能在低耗能模式下工作。下面是一个简单检查方法。

4.13.1 时钟

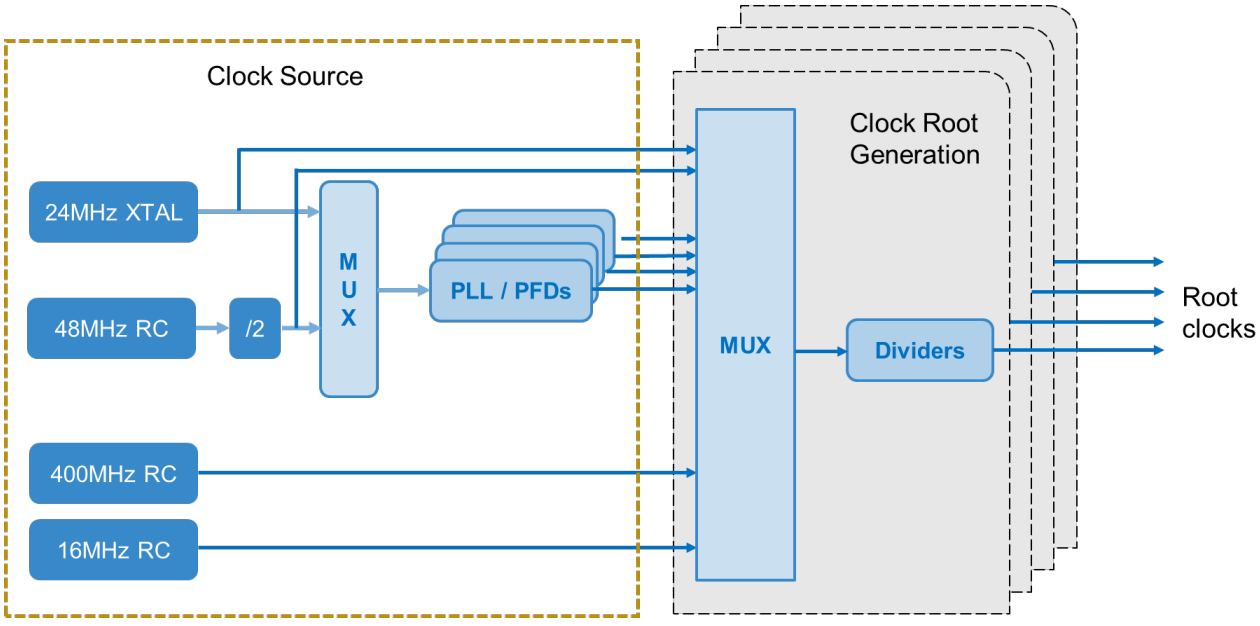


图 9. Clock Source and Clock Root

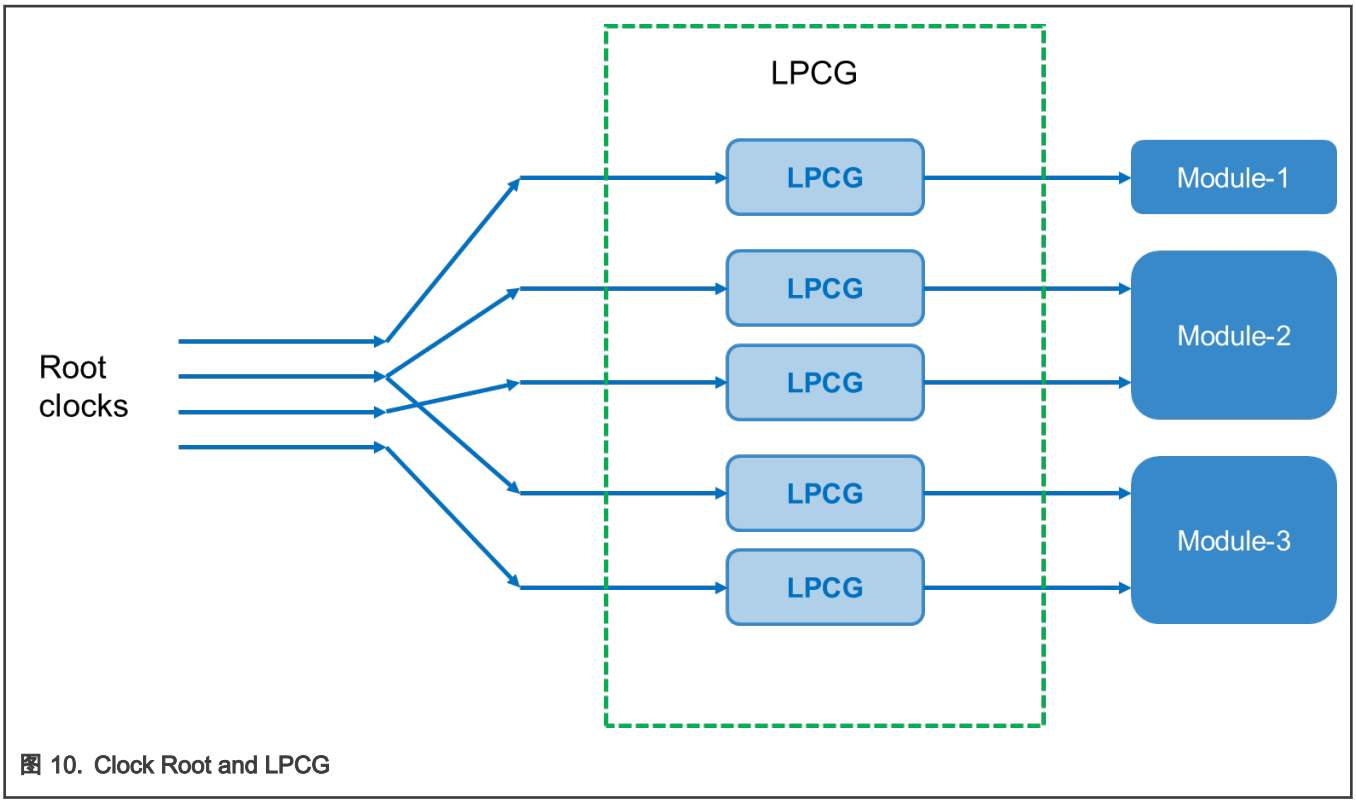


图 10. Clock Root and LPCG

如 图 9 和 图 10 所示，整个时钟包含三部分，Clock Source，Clock Root 和 LPCG。 为了确保外设的时钟可用，这三个部分需要正常工作。

1. 时钟源（Clock Source）：检查时钟源是否可用。

   在 RT1170 中有若干个时钟源，这些时钟源通常被 SP 控制。因此，第一步需要检查被外设所用的时钟在当前的 SP 下是否被使能。如果需要在 STBY 模式中，通常所有的时钟源都要被关闭以获得最少的电量消耗。在 STBY 模式下，能够产生异步中断的外设同样可以作为一个唤醒源。比如 LPUART, LPIIC 或 LPSPI，对于这种情况，RC16M 和 RC48M 或其他低频时钟源可以被使用。除此之外，一些外设可以使用 32K 作为时钟源，像 GPT, RTC 和 WDOG，32K 是一个在任何 SP，STBY 和电源模式下都在工作的时钟，这表示着这些外设可以在任何模式下工作。

2. 时钟根（Clock Root）：使能时钟根，选择一个时钟源和设定一个合适频率的分频系数。

   Clock Root 用来为外设选择一个时钟源。一些时钟根被 SP 控制，比如 M7,M4,BUS, BUS_LPSR。但是大多数的时钟根都被 SW 控制。

3. LPCG：检查目标 CPU 模式和 LPCG 的设置。

   LPCG 是一个在时钟根和外设之间的 gate。对于外设作为唤醒源的情况，LPCG 总是被 CPU 电源模式控制，以下是配置的情况：

   - 在任何模式下均不可用。
   - 在 Run 模式可用。
   - 在 Run, Wait 模式下可用。
   - 在 Run, Wait, Stop 模式下可用。
   - 在 Run, Wait, Stop 和 Suspend 模式下可用。

4.13.2 外设的供电

在检查过外设时钟之后，还需要检查供电。RT1170 中的外设属于不同的域，所以如果一个电源域或者 MIX 被关闭，该域内的外设也会被断电。

比如，在 RT1170 中有 3 个 CANFD，CANFD 1 和 2 属于 WAKEUP MIX 但是 CANFD 3 属于 LPSR MIX。假设时钟系统可以正常工作，第二步就是检查哪个 CANFD 作为唤醒源。如果 CANFD1 或者 2 作为唤醒源，那么 WAKEUP MIX 就要在低功耗模式下打开。电源 MIX 的开关可以被软件或者硬件 SP 控制。通常推荐使用 SP 因为它会使开关更简单安全，这说明只需要检查 SP 中 WAKEUP MIX 是否打开。对于 CANFD3，使用以上同样的办法。

4.14 唤醒之后的外设状态

唤醒之后的外设状态取决于以下几点。

- 时钟

  时钟源通常被 SP 控制，请检查芯片是否进入 STBY 模式中的时钟源。

- 电源

  如果一个外设所在的 power mix 断电了，那么这个外设的设置就会丢失，在下次用之前需要重新初始化。

  比如在 EVK 上，GPIO_AD_04 连接到一个 LED 可以用来指示状态。

  — 这个引脚作为 GPIO 使用时是属于 WAKEUP MIX。

  — 如果芯片转换到 WAKEUP MIX 关闭的 SP 时，这个 GPIO 也会关闭设置就会丢失。当芯片被重新唤醒时，GPIO 需要再次初始化才能使用。

  — 如果 WAKEUP MIX 一直没有断电，那么 GPIO 就不用再次初始化了。

- SSARC

  状态保持和恢复控制器（SSARC）在关闭电源之前保存了寄存器的功能模块以及当打开电源时，从内存中恢复寄存器。

  如果电源 MIX 在低功耗和唤醒过程中关闭，SSARC 可以帮助保存寄存器的设定然后当电源打开后可以恢复设定。以 GPIO 为例，GPIO_AD_04 连接到 LED 上，当 SSARC 使用时，即使在进入低功耗状态后 WAKEUPMIX 被关闭，GPIO 不需要重新初始化就能被唤醒。

4.15 复位后跳转到指定地址运行

当芯片从 Suepend 模式被唤醒时，需要复位。当芯片被唤醒后，它可以跳转到指定的地址运行。CM7 和 CM4 都支持这个功能。以 CM7 为例:

```
/*
    0x9e00 is vector table base address
    0x9f00 is the SP after reset
*/
*(uint32_t *)(0x9e00) = 0x9f00;
/*
    the value in 0x9e04 is the PC after reset
    test address is the PC after reset
*/
*(uint32_t *)(0x9e04) = ((uint32_t)test);
// set start address after waking up from SUSPEND mode
IOMUXC_LPSR_GPR->GPR26 &= ~IOMUXC_LPSR_GPR_GPR26_CM7_INIT_VTOR_MASK;
IOMUXC_LPSR_GPR->GPR26 |= IOMUXC_LPSR_GPR_GPR26_CM7_INIT_VTOR((0x9e00) >> 7);
//This API is able to reset CM7 to verify the jump functon
SRC_AssertSliceSoftwareReset(SRC, kSRC_M7CoreSlice);
```

图 11. 配置跳转地址

```
//Put the test() to TCM
AT_QUICKACCESS_SECTION_CODE(void test(void));
void test(void)
{
    while(1)
    {
        GPIO_PortToggle(GPIO9,1<<29);    //Toggle a GPIO LED
        Delay();
    }
}
```

图 12. 测试 API

实现这个函数需要三个条件:

- 跳转地址需要与 0x80 字节对齐。这是由 ARM 核决定的。
- 跳转地址的内存或外设需要通电或者在保持模式，这里的内存通常指 RAM。当内存关闭的时候，数据就会丢失。对于外设，是指 FlexSPI, 如果 Flex SPI 关闭了，寄存器的配置就会丢失。当芯片被唤醒后，它会去获取 FlexSPI 的指令，但是由于 FlexSPI 还没有初始化，芯片就会产生一个死锁复位（Lockup reset）然后从 image entry address 开始运行。利用 SSARC 可以解决关于 FlexSPI 的问题。
- 图 11 和 图 12 中所展示的地址应该是没有使用过的内存地址。

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: January 4, 2021

Document identifier: AN13104

arm