AN13104

Debug and Application for i:MX RT1170 Clock and Low Power Feature

Rev. 0 — January 4, 2021 Application Note

by: NXP Semiconductors

1 Introduction

The i.MX RT1170 crossover processor sets speed records at 1 GHz. This ground-breaking family combines superior computing power and multiple media capabilities with more usable as well as real-time functionality. The dual-core i.MX RT1170 runs on the Arm® Cortex®-M7 core at 1 GHz and Arm Cortex-M4 at 400 MHz. It provides best-in-class security. The i.MX RT1170 MCU supports a wide range of temperature and is qualified for consumer, industrial and automotive markets.

This application note introduces not only clock and low-power features in RT1170, but also some debug and application skills when developing a low-power use case.

Contents

2 RT1170 power domains

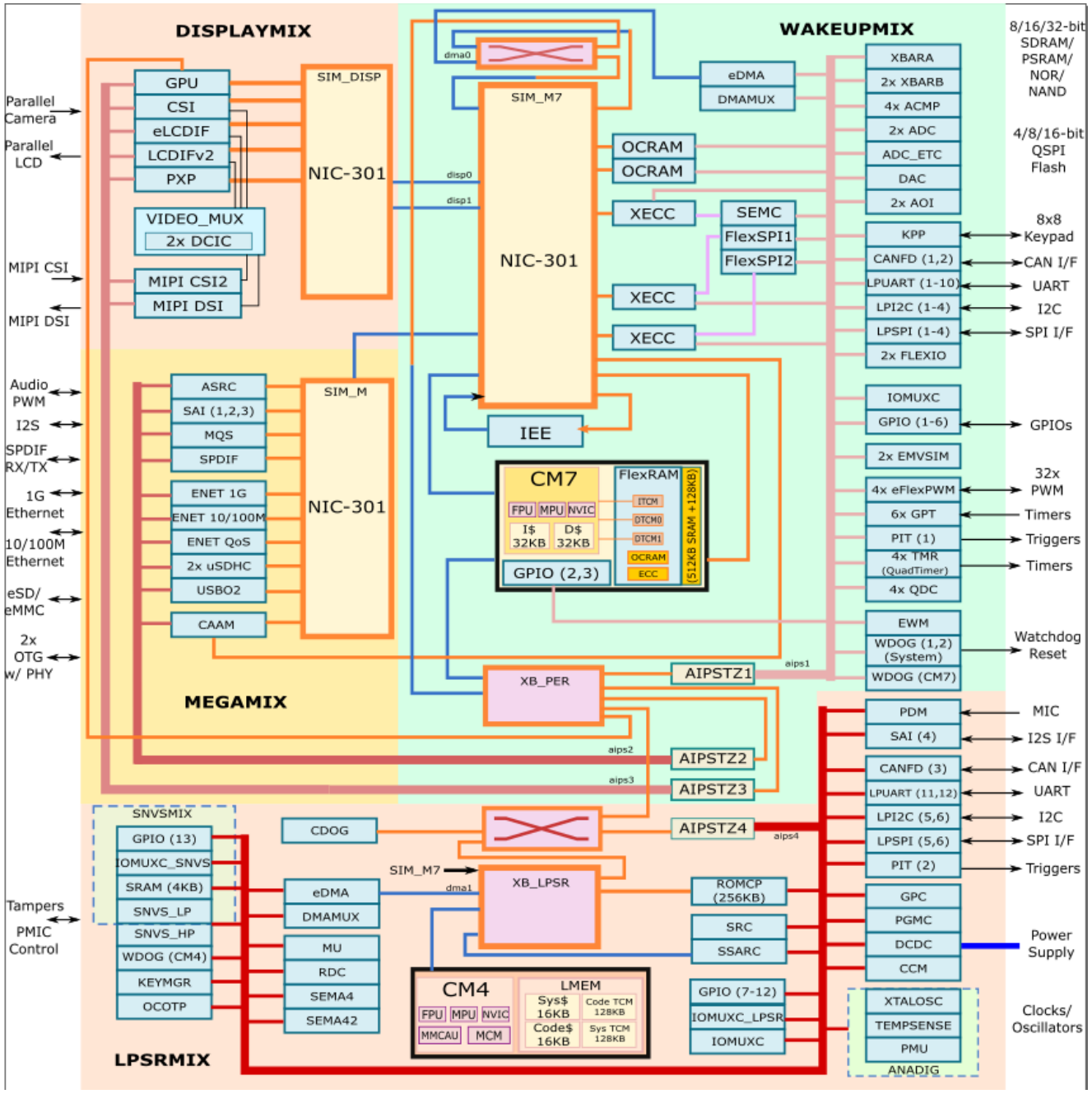


Figure 1. RT1170 power domain diagram

The peripherals of RT1170 are allocated to each power domain according to the function and attribute.

SoC domain includes WAKEUP MIX, MEGA MIX and DISP MIX. SoC domain is powered by DCDC. If the DCDC is powered down, these MIX and its peripherals are also powered down. However, WAKEUP MIX is powered down directly by DCDC without a power gate while MEGA MIX and DISP MIX have their own power gate. LPSR domain includes some peripherals and CM4 platform, but CM4 platform has its own function domain. Table 1 lists peripherals in each domain.

Table 1. Peripherals in each domain

Domain	Description
CM7	Cortex-M7 core and its platform
CM4	Cortex-M4 core and its platform
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 Power state of RT1170

The power state of RT1170 include CPU mode, Setpoint (SP) mode, and StandBy (STBY) mode. Here are some examples:

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

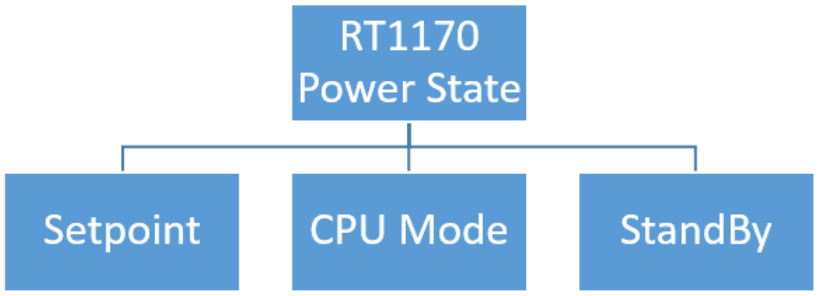


Figure 2. Power state

3.1 CPU mode

Each CPU platform has its own power mode. RT1170 contains four modes: RUN, WAIT, STOP and SUSPEND. Table 2 shows the differences of CPU modes.

Table 2. Differences of CPU modes

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. The On-platform peripheral state is related with the CPU power state. For example, there is a fast GPIO in M7 platform and usually GPIO can be a wakeup source for the whole system. But if the CM7 enters the suspend mode or other CPU power-down mode, the fast GPIO cannot be a wakeup source, because the fast GPIO is powered down.
2. The settings can configured by PGMC.

From the CPU side, the main difference is the CPU state. CPU enters the WFI or WFE state and wait for the wakeup signals. Meanwhile, the clock and power of CPU enters their own state. The exit latency or wakeup time is impacted by the clock and power state. The CPU is powered down in the SUSPEND mode, which means it takes a longer wakeup latency than WAIT and STOP mode.

3.2 Setpoint

Setpoint (SP) is a new concept in RT1170. It is a group settings for clock and power supply. There are 16 SP in RT1170 that means there are 16 group settings for clock and power supply.

SP can help to enable or disable a clock source by hardware. For example, ARM PLL needs to power on at SP1 but power down at SP9. When trigger a SP transition from 1 to 9, the ARM PLL will be powered down. If the SP transition is from 9 to 1, ARM PLL is powered on. These power-on and power-off steps can be performed on the hardware. Except for RC16M (OSCPLL0), the other 28 clock sources can be controlled by SP. RC16M is an Always-ON clock source and it can only be powered down in the STBY mode.

Table 3. Different clock source state under different SP

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

For example, at SP1, the M7 clock root run at 1 GHz, the clock source is **PLL1G**, and DIV is **0**. Once the SP transits to SP0, the source of `clock_root` is switched to Arm PLL which runs at 700 MHz. If the developer wants this SP to run at 350 Mhz, the DIV can also set as **1**. These operations are performed by SP.

Table 4. Different clock root settings under different SP

	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...

Table 4. Different clock root settings under different SP (continued)

	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

Besides the clock, SP can help to control power supply. Such as Enable FBB at SP1 which the CPU run at 1 GHz, disable FBB at other SP. Enable DCDC or Disable DCDC in some SP. SP can control the internal LDO enable, disable or bypass. Power domain MIX (WAKEUP, MEGA and DISP MIX) power ON or OFF and some memory power level settings can be set by SP, such as TCM and LMEM.

SP is used to control the state of the CCM, GPC, SRC, SSRAC, PGMC, DCDC and PMU modules. There are up to 16 setpoints which can be used to configure clock and other power related peripherals. SP controls the enable or disable state of a peripherals and configures a detailed value to a peripherals. Such as DCDC, SP can enable or disable in each setpoint and adjust the DCDC voltage in each setpoint. In other words, SP is a group of settings used to control some peripherals under different setpoint level.

3.3 Stand-by (STBY)

Standby mode is the third kind of low-power mode besides CPU mode and SP. It is related to state of all CPU platforms and has a much shorter transition time than SP. STBY further reduces power consumption through the following operations.

- Load the STBY SP settings to each module.
- Analog modules enter STBY mode.
- Internal BUS stop working.
- Almost all of system are stopped and only the wakeup source alive.

When the chip enters the STBY mode, the currents of `DCDC_IN`, `LPSR_IN`, and `SNVS_IN` are about a few mA in total. This is an important current value because in other power mode or non STBY mode, the current is much bigger.

Before entering the STBY mode, both CPUs should be in the low-power mode, WAIT, STOP, or SUSPEND. If a core doesn't enter a low-power mode, the chip is not able to enter the STBY mode. The low-power mode for each core can be different, such as, CM7 WAIT STBY and CM4 SUSPEND STBY.

4 Debug and application skills

4.1 Clock output

Clock output pins help users to know whether a clock or a PLL work well or not, including enable or disable status, the frequency in the run or low-power mode. RT1170 contains two clock output pins, `CLKO1` and `CLKO2`. They can fan out the clock to a pin and an oscilloscope can be used to measure the information as needed. Two testing pads are required for this function.

- `CLKO1:GPIO_EMC_B1_40` can fan out the following clocks:
  - `kCLOCK_OscRc48MDiv2`
  - `kCLOCK_Osc24Mout`
  - `kCLOCK_OscRc400M`
  - `kCLOCK_OscRc16M`
  - `kCLOCK_SysPll2Pfd2`
  - `kCLOCK_SysPll2Out`
  - `kCLOCK_SysPll3Pfd`

  - kCLOCK_SysPll1Div5

- CLKO2:GPIO_EMC_B1_41 can fan out the following clocks:
  - kCLOCK_OscRc48MDiv2
  - kCLOCK_Osc24Mout
  - kCLOCK_OscRc400M
  - kCLOCK_OscRc16M
  - kCLOCK_SysPll2Pfd3
  - kCLOCK_OscRc48M
  - kCLOCK_SysPll3Pfd1
  - kCLOCK_AudioPllOut

The output capacity of the pin is limited, so it is recommended to use the internal frequency divider to output after frequency division for higher frequency clocks. Taking CLKO2 output Aduio PLL as an example:

1. Initiate the pin.

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

2. Configure the clock source and divider. Audio PLL may work at a high frequency. It means that a suitable frequency division factor needs to be configured.

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

3. A waveform can be measured by an oscilloscope if the audio PLL is enabled.

There is a limitation: when the first output clock source is turned and tries to go another clock source, it will fail. For example:

Switch all the clock roots using **RC48M_DIV2** as the clock source to **OSC24M**. The reason for this step is that most clock roots use **RC48M_DIV2** as a default clock source. Then power off RC48M.

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

If using **rootCfg.mux = 0** as the clock source, switch to **rootCfg.mux = 1**. Even if the OSC24M clock is available, the clock waveform cannot get it by an oscilloscope.

4.2 Clock observer

In RT1170, the clock observer, like an oscilloscope, is used to measure the clock frequency. It is a useful function on developing the power mode and other cases related with clock. There is an example in power mode switch:

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

This API can print the frequency of M7_Clock_Root, M4_Clock_Root, Bus_Clock_Root, LPSR_Clock_Root. An API can get more Clock_Root frequency:

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

Taking audio PLL as an example:

```
CLOCK_GetFreqFromObs(CCM_OBS_PLL_AUDIO_OUT);
```

Then the frequency of Audio PLL can be obtained by this API.

This observer can measure by the software and fan out the clock to a pad. It is useful when debugging in the lower power mode.

4.3 CPU power mode and Setpoint State

When developing a low-power case, the CPU mode and SP value vary with requirements. Some registers can monitor the current power mode, SP value, and the previous state. There is a CPU mode status register in GPC. It is a read-only register. Reading `bit[1:0]` can get the current power mode state and `bit[3:2]` can get the previous state.

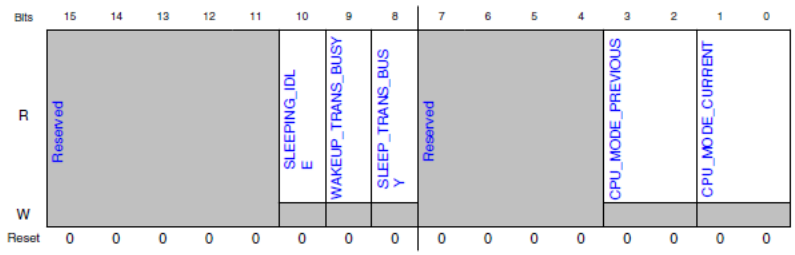


Figure 3. CPU mode state register

When debugging with SP current and previous register, one case may cause confusion: SP transition needs both M7 and M4 transition done before you can get the expect target value. There is an example:

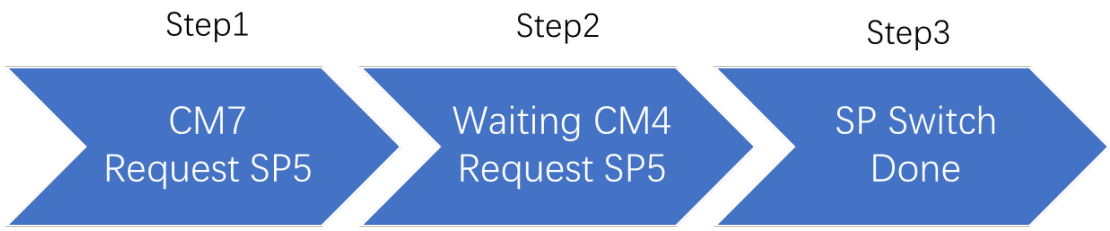


Figure 4. SP transition steps

Figure 4 shows a simple process about the SP transition from SP1 to SP5. If the chip is under SP1, CM7 sends SP5 request and then immediately checks the current and previous register. The current register value is still SP1 not SP5. The reason is that the SP transition needs both M7 and M4 done. It is similar with Step2 in Figure 4. If checking the value in Step3, a correct value is obtained. This will also happen when CM4 sends SP transition first.

4.4 Chip enters standby mode or not

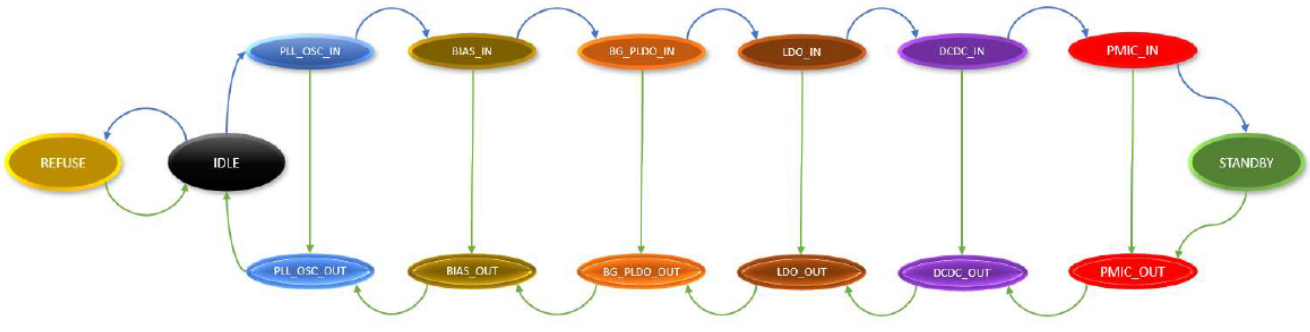


Figure 5. Standby flow

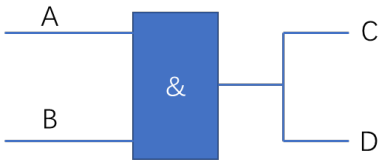
As shown in Figure 5, the last step in standby transition sequence is PMIC_IN, which means if the PMIC enters STBY, the whole sequence ends successfully. PMIC_STBY_REQ pin indicates that the PMIC enters STBY mode. When the system enters STBY mode, PPC sends a signal to PMIC_STBY_REQ and outputs a high-level signal. The PIN can be used to check whether the Chip is in the STBY mode or not. This method may consume more power if a resister or a device is connected to this PIN. In the EVK board, it connects some devices and consumes more power in SNVS domain.

There is another method to monitor the Standby. The RC16 oscillator is an always-on clock source. When entering the STBY mode, it is powered down, which means a CLKO pin can be used to indicate this state. If the RC16 is powered down, the system enters the STBY mode.

4.5 Handshake

The purpose of handshake is to ensure that before the CPU issues a low-power instruction WFI, all peripherals complete all transmissions and no longer receive new transmissions, to safely enter a low-power mode.

If a peripheral isn't assigned to a domain, the handshake needs to complete in two registers. The stop requset needs to send from IOMUXC_GPR and IOMUXC_LPSR_GPR. When both stop requests are sent and the peripheral completes all transmissions, the stop_ack signal is asserted.



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

Figure 6. Handshake

Taking CAN1 as an example, CAN1_STOP_REQ needs to be sent from IOMUXC_LPSR_GPR35 bit[9] and IOMUXC_GPR70 bit[9].

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

Then, check whether the transmission completes or not. Check IOMUXC_LPSR_GPR40 bit[3] and IOMUXC_GPR75 bit[3].

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

If a peripheral is assigned to a domain, only one stop request needs to be sent. Taking CAN1 as an example, CAN1 is assigned to CM4 domain and CAN1_STOP_REQ needs to be sent from IOMUXC_LPSR_GPR35 bit[9].

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

Then, check whether the transmission completes or not. Check IOMUXC_LPSR_GPR40 bit[3].

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

Before entering a low-power mode, make sure that all the peripherals have completed the transmission.

NOTE

DMA/ENET (bus masters) should complete the handshake first and then flash/memories (bus slaves).

4.6 Power down OCOTP

Powering down OCOTP can reduce the power consumption. For most use cases, OCOTP can be powered down after system boot, but for applications involving security, the power-down operation needs to be careful.

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

4.7 SNVS_XX pin leakage

`SNVS_xx` pins can be used as tamper pins. By default, `SNVS_xx` pins are input and internal pull resistors are not enabled. If `SNVS_xx` pins are floating, the internal logic driven by `SNVS_xx` pins are floating and then the leakage happens. So enabling the internal pull up resister to avoid the leakage.

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

4.8 Enable DCDC DCM mode

DCDC module supports Continuous Conduction Mode (CCM) and Discontinuous Conduction Mode (DCM) operations. In low-power applications, DCM helps to increase the efficiency of the DCDC module to achieve lower power consumption.

Detailed configurations include:

- `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`

The targets of DCDC low-power mode can be set to one step lower than the targets of run mode. For example, if the target of run mode is 1.0 V and 1.8 V, set target of low-power mode to 0.975 V and 1.775 V. In addition, set `ANA_STYBY_TRG_SP[3:0]` and `DIG_STBY_TRG_SP[3:0]` one step lower than the run mode targets.

4.9 Wakeup source configuration

Before entering a low-power mode and executing WFI instruction, configure wakeup sources first. Follow steps as below and an interrupt wakeup source can be used to wake up chip in the low-power mode. The handshake step is to guarantee that all the peripherals have completed the transmission. Assuming the handshake has been completed and the peripheral has been initialized, take GPIO and GPT as wakeup source as an example.

4.9.1 GPIO as a wakeup source

```
/* 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);
```

Figure 7. Configure GPIO as a wakeup source

1. Enable a PIN interrupt.
2. Enable GPIO IRQ.
3. Disable all of wakeup sources registers in GPC.
4. Enable GPIO as a wakeup source in GPC.

4.9.2 GPT timer as a wakeup source

```
/* 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);
```

Figure 8. Configure GPT as a wakeup source

1. Enable a GPT output compare interrupt.
2. Enable GPT IRQ.
3. Disable all of wakeup source registers in GPC.
4. Enable GPT as a wakeup source in GPC.

4.10 Chip cannot enter a low-power mode

CPU cannot enter a low power mode due to the the following reasons.

- **Pending interrupt**

  A pending interrupt prevents the GPC entering a low-power mode. There are a couple of methods to check whether there is a pending interrupt.

  — `CM_IRQ_WAKEUP_STAT_x` in GPC register

  `x` means the index of `CM_IRQ_WAKEUP_STAT` register. There are eight registers for a GPC. That means for Dual Core silicon, 16 `CM_IRQ_WAKEUP_STAT` registers needs to check and eight registers for Single Core silicon.

  — `CM_NON_IRQ_WAKEUP_STAT` in GPC register

When a debugger is connected to a silicon, it may generate a pending interrupt to prevent the chip entering a low-power mode.

These registers can check whether there is a pending interrupt to prevent the system entering a low-power mode. If there is a pending interrupt,

1. Find which operation or peripheral generated this interrupt.
2. Disable and clean the status bit of this interrupt.

- **Handshake**

  If the polling method is used to check the handshake result, the system may be blocked by an uncompleted transmission.

4.11 Enter SNVS mode

SNVS mode can power down all of domain except SNVS domain by powering down the external DCDC circuit. Either software or hardware can help to enter the SNVS mode.

- Software mode:

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

- Hardware mode:

  Press the **ON/OFF** button for more than five seconds, and the chip enters the SNVS mode.

4.12 Wake up from SNVS mode

To wake up the chip from SNVS mode, use the peripheral running and generating an interrupt as a wakeup source. For example, the **WAKE UP** pin can configure as a GPIO and generate an interrupt when there is a input signal, edge trigger or level trigger.

For RT1170, the tamper pin on some parts can be used as a GPIO, which means these pins can wake up the chip.

Besides GPIO, RTC is a wakeup source in the SNVS mode.

The **ON/OFF** button is another wakeup source to wake up chip from SNVS mode. Pressing **ON/OFF** can wake up the chip.

4.13 Peripheral state in a low-power mode

Usually a peripheral needs two basic conditions to work normally, clock and power supply. These two condition are also used to judge whether a peripheral can work in a low-power mode. The below describes simple check steps.

4.13.1 Clock

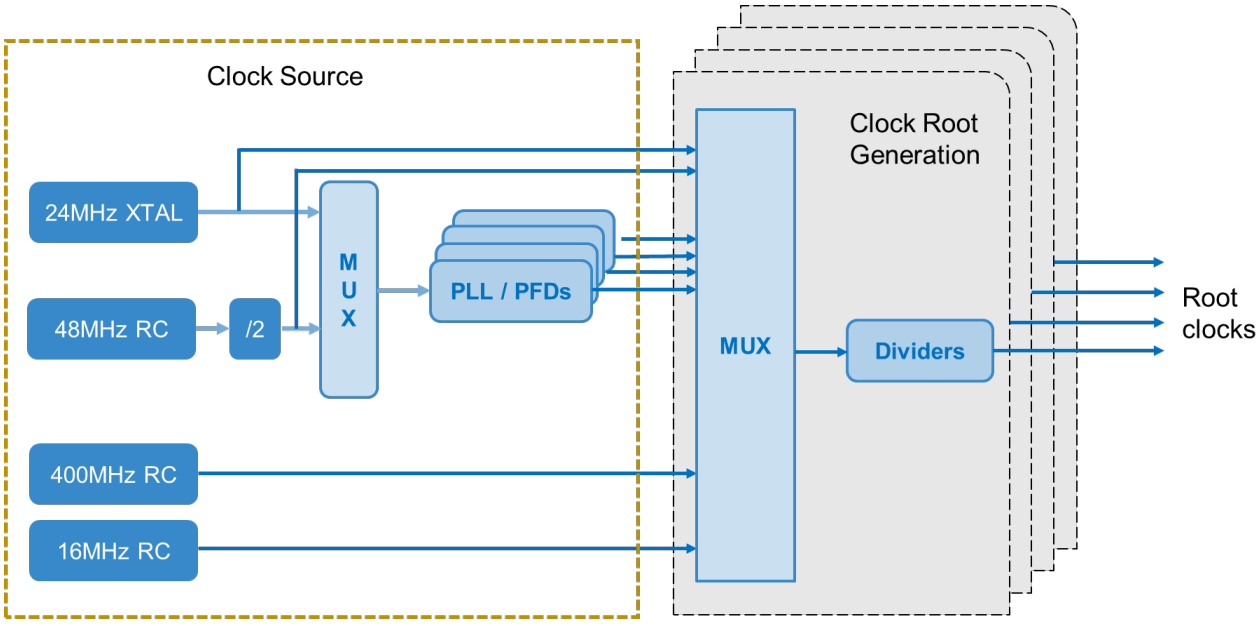


Figure 9. Clock source and clock root

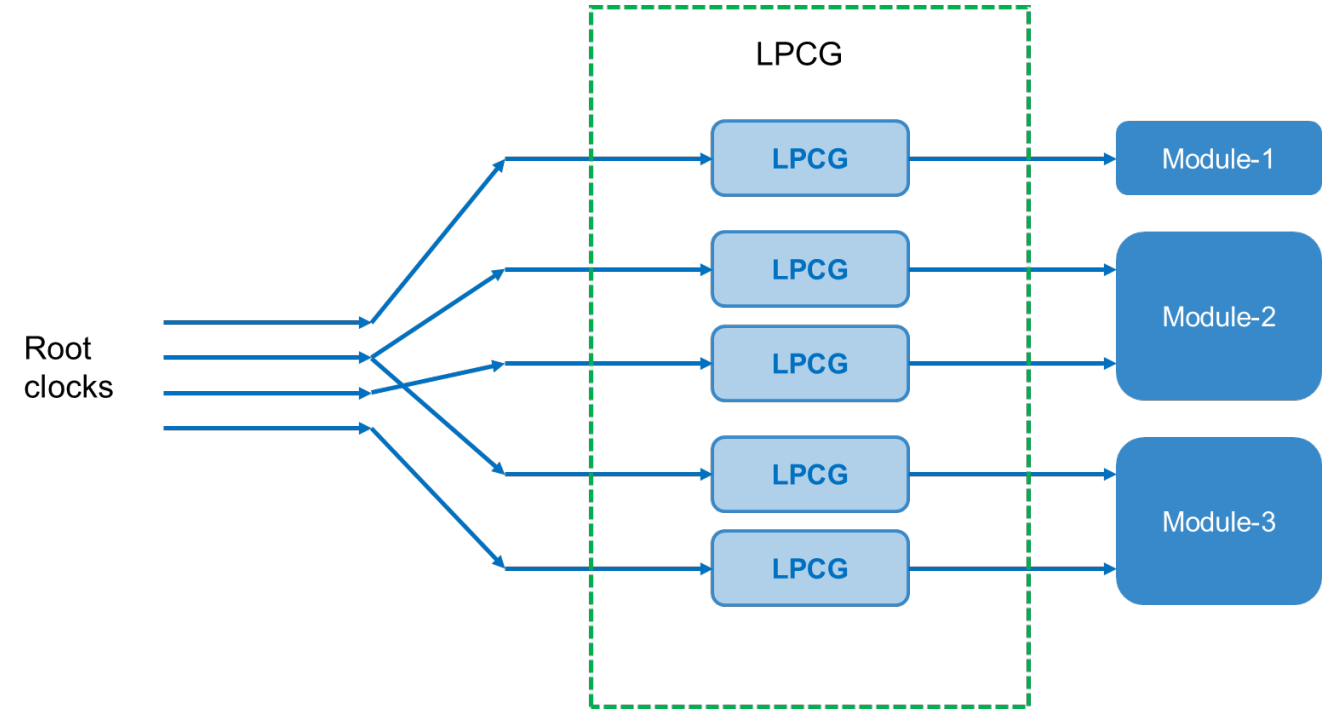


Figure 10. Clock root and LPCG

As shown in Figure 9 and Figure 10, clock includes clock source, clock root and LPCG. To make sure the clock for a peripheral is available, these three parts need to be enabled and work at a suitable frequency.

Steps to check whether a peripheral can work in a low-power mode:

1. **Clock source**: Check whether the clock source is used for the peripheral enable under an SP.

There are several clock sources in RT1170. In a NOT-STBY mode, these clock sources are controlled by SP. If RT1170 needs to work in the STBY mode, power down all clock sources to get a minimum power consumption data. However, if a peripheral is the wakeup source by an asynchronous interrupt such as LPUART, LPIIC or LPSPI, keep RC16M and RC48M or other low-frequency clock sources alive. Besides this, some peripherals such as GPT, RTC and WDOG can use 32 K as a clock source. 32 K is a always online clock source in any SP, power mode and even STBY mode. These peripheral can work under any SP, power mode and even STBY mode.

2. **Clock root**: Enable the clock root, select a clock source for clock root, and set an appropriate frequency division factor.

   The clock root is used to select a clock source for a peripheral. For some clock roots, it can be controlled by SP, such as M7, M4, BUS, BUS_LPSR. But most clock roots are controlled by SW.

3. **LPCG**: Check the target CPU mode and LPCG settings.

   LPCG is a clock gate between clock root and a peripheral. For a peripheral which will be used as a wakeup source, LPCG is always controlled by CPU power mode. The settings are as below:

   - The clock source is not needed in any mode and can be turned off.
   - The clock source is needed in RUN mode but not needed in WAIT or STOP mode.
   - The clock source is needed in RUN and WAIT mode but not needed in STOP mode.
   - The clock source is needed in RUN, WAIT, and STOP mode.
   - The clock source is always ON in any mode include SUSPEND.

4.13.2 Peripheral power supply

The peripheral in RT1170 belong to different domain. If a power domain or a power MIX is powered down, the peripheral under this MIX will also power down.

For example, there are three CANFD in RT1170, **CANFD1** and **CANFD2** belong to WAKEUP MIX and **CANFD3** belongs to LPSR MIX. Assuming that the clock system is working properly, check which CANFD is used as a wakeup source. If **CANFD1** and **CANFD2** are used, power on WAKEUP MIX in the lower power mode. MIX ON or OFF can be controlled through either hardware SP or software. SP is always recommended because it makes the power mode switch simpler and safer. To make sure that the WAKEUP MIX is powered on in the low power mode, just check the SP table. For **CANFD3**, follow the same steps.

4.14 Peripherals state after wakeup

The state of a peripheral after the chip wakeup depends on the following items.

- **Clock**

  The clock source is controlled by SP. Check the clock source as described in Chip enters standby mode or not.

- **Power**

  If the power mix which the peripheral belongs to is powered down, the peripheral's settings are lost and there is no need to re-initialize it.

  For example, `GPIO_AD_04` is connected to an LED to indicate the state.

  — When this pin works as GPIO, it belongs to WAKEUP MIX.

  — If the chip switches to SP15 or other SP with WAKEUP MIX OFF, this GPIO is powered down. After the chip is waken up, this GPIO cannot be used directly until re-initialized.

  — If the chip switches to an SP with WAKEUP MIX ON, the GPIO doesn't need to be re-initialized.

- **SSARC**

  The State Save and Restore Controller (SSARC) saves the registers of functional modules in memory before power-down and restores the registers from memory after the module is powered on.

  If the power mix is powered down in the low power-mode and wakeup process, SSARC can help to save the register settings before a MIX power-down and restore the settings after the MIX power-on. Taking GPIO as an example, `GPIO_AD_04` is

connected to an LED. If the SSARC is used, even when the WAKEUP MIX is powered down in the low-power mode, once the chip wakes up, the GPIO can be used directly without initialization.

4.15 Jump to a specified address after reset

A reset is required if the chip wakes up from the suspend mode. After the chip wakes up, the silicon is able to jump to a specified address. Both CM7 and CM4 support this function. Taking CM7 as an example:

```
/*
    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);
```

Figure 11. Configure the jump address

```
//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();
    }
}
```

Figure 12. Test API

To achieve this function, the following conditions are required:

- The jump address needs to be aligned with 0x80.This requirement is determined by Arm core.
- The memory (RAM) or peripheral (FlexSPI) of the jump address need to be powered on or in the retention mode. Once RAM is power down, the data is lost. If the FlexSPI is powered down, the registers settings are lost. Once the chip wakes up, it fetches the instruction from the FlexSPI. But the FlexSPI is in the uninitialized state, the chip generates a lockup reset and re-boot from the image entry address. For FlexSPI, SSARC helps to solve this problem.
- The address listed in Figure 11 and Figure 12 is unused.

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