AN12680

K32L2A 中时钟管理与分配

版本 0 — 2020 年 1 月

应用笔记

目录

1 介绍.....1
2 时钟架构.....1
3 SCG（系统时钟生成器）.....3
3.1 SCG 架构.....3
3.2 SCG 和 MCG/MCG_Lite 之间的差异.....4
4 PCC（外部时钟控制）.....5
5 SCG 时钟模式转换.....7
5.1 SCG 有效时钟模式.....7
5.2 SCG 时钟模式转换示例.....9
5.3 HSRUN 与 VLPR 模式下 SCG 配置.....10
5.4 STOP 模式中时钟配置.....11

1 介绍

本应用笔记介绍了 K32L2A 的时钟架构和时钟分配。尤其是 K32L2A 中两个与以往不同的新时钟模块：

- SCG（系统时钟生成器）
- PCC（外部时钟控制）

SCG 提供的参考时钟范围比 MCG/MCG_Lite 更精确，对于不同的应用也更加灵活。借助 SCG，内核时钟和外设时钟可以来自不同的时钟源。这使外设时钟甚至高于内核时钟/总线时钟。

PCC 提供外设时钟控制和配置寄存器，例如时钟多路复用器和时钟分频器。与旧的时钟门和 SIM 模块中的配置不同，PCC 模块更易于选择外设时钟源，对软件设计者更加友好。

2 时钟架构

时钟架构的核心是系统时钟生成（SCG）模块。SCG 控制四个时钟源（SIRC，FIRC，SOSC 和 SPLL），并将它们分配给各内核，总线，内存模块和外设。外设通常有两个时钟，一个是外设接口时钟，内核/ DMA 用它来与外设的寄存器连接，第二个是外设功能时钟，用于外设的主要功能实现（例如，它为串行通信外设提供波特率，或者作为定时器外设计数器的输入时钟）。外设接口时钟通常直接从 SCG 到外设。通过外设时钟控制（PCC）模块选择外设功能时钟。 SCG 通过 PCC 模块提供带有可选分频器的附加外设功能时钟。

注意

1. 所有外设功能时钟均为异步时钟。
2. 对于具有 MCG /_Lite 的其他设备，外设接口时钟也称为总线时钟。

图 1 展示了 K32L2A 的各种时钟源和时钟树。

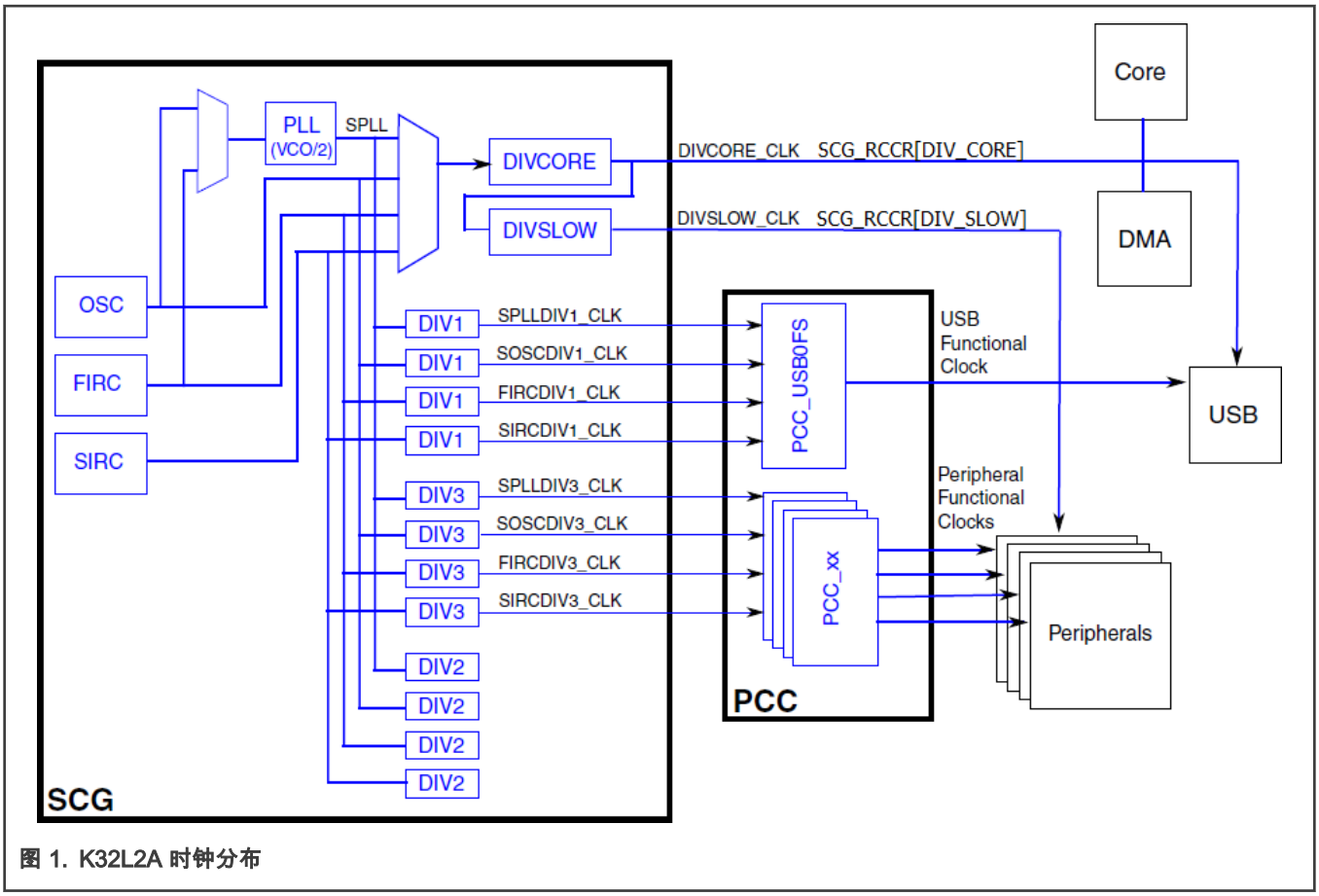

图 1. K32L2A 时钟分布

通常来说，DIVCORE_CLK 是旧的内核/平台时钟，DIVSLOW_CLK 是旧的总线时钟。DIVCORE_CLK 和 DIVSLOW_CLK 都可作为外设接口时钟。外设功能时钟来自 DIV1-DIV3_CLK。CLK_SRC 是 SCG 时钟源，可由 PCC 模块选择。

注意

1. K32L2A 闪存时钟来自 DIV_SLOW 时钟。
2. 除 USB 模块外所有外设使用<CLK_SRC>DIV1 作为功能时钟，而 USB 模块使用<CLK_SRC>DIV3 作为功能时钟。

表 1 详细描述了 K32L2A 中的时钟定义。

表 1. 详细的时钟定义

时钟名称	运行模式时钟频率	VLPR 模式时钟频率	HSRUN 模式时钟频率	时钟源	时钟可以在以下情形禁用
SOSCDIV1_CLK, SOSCDIV3_CLK	高达 48 MHz	高达 8 MHz	高达 48 MHz	SCG	没有被任何外设和/或 DIVCORE_CLK 使用或反馈至 PLL
SPLLDIV1_CLK, SPLLDIV3_CLK	高达 72 MHz	PLL 关闭	高达 96 MHz	SCG	没有被任何外设和/或 DIVCORE_CLK 使用

下页继续...

表 1. 详细的时钟定义（续上页）

时钟名称	运行模式时钟频率	VLPR 模式时钟频率	HSRUN 模式时钟频率	时钟源	时钟可以在以下情形禁用
FIRCDIV1_CLK, FIRCDIV3_CLK	高达 60 MHz	FIRC 被禁用	高达 60 MHz	SCG	没有被任何外设和/或 DIVCORE_CLK 使用或反馈至 PLL
SIRCDIV1_CLK, SIRCDIV3_CLK	8 MHz	8 MHz	8 MHz	SCG	没有被任何外设使用

3 SCG（系统时钟生成器）

3.1 SCG 架构

系统时钟生成器（SCG）模块提供 MCU 的系统时钟。其包含：

- SOSC：外部振荡器输出（晶体或外部施加的时钟输入）
- SIRC：低速（8 MHz）内部 RC 振荡器输出
- FIRC：高速（48 MHz）内部 RC 振荡器输出
- SPLL：PLL 输出，它由 SOSC 参考时钟或 FIRC 产生

SCG 可以选择四个输出时钟之一作为 MCU 的系统时钟源。它还具有可对 DIVCORE / DIVSLOW 和外部功能时钟的时钟输出进行分频。

图 2 展示了 SCG 模块框图。

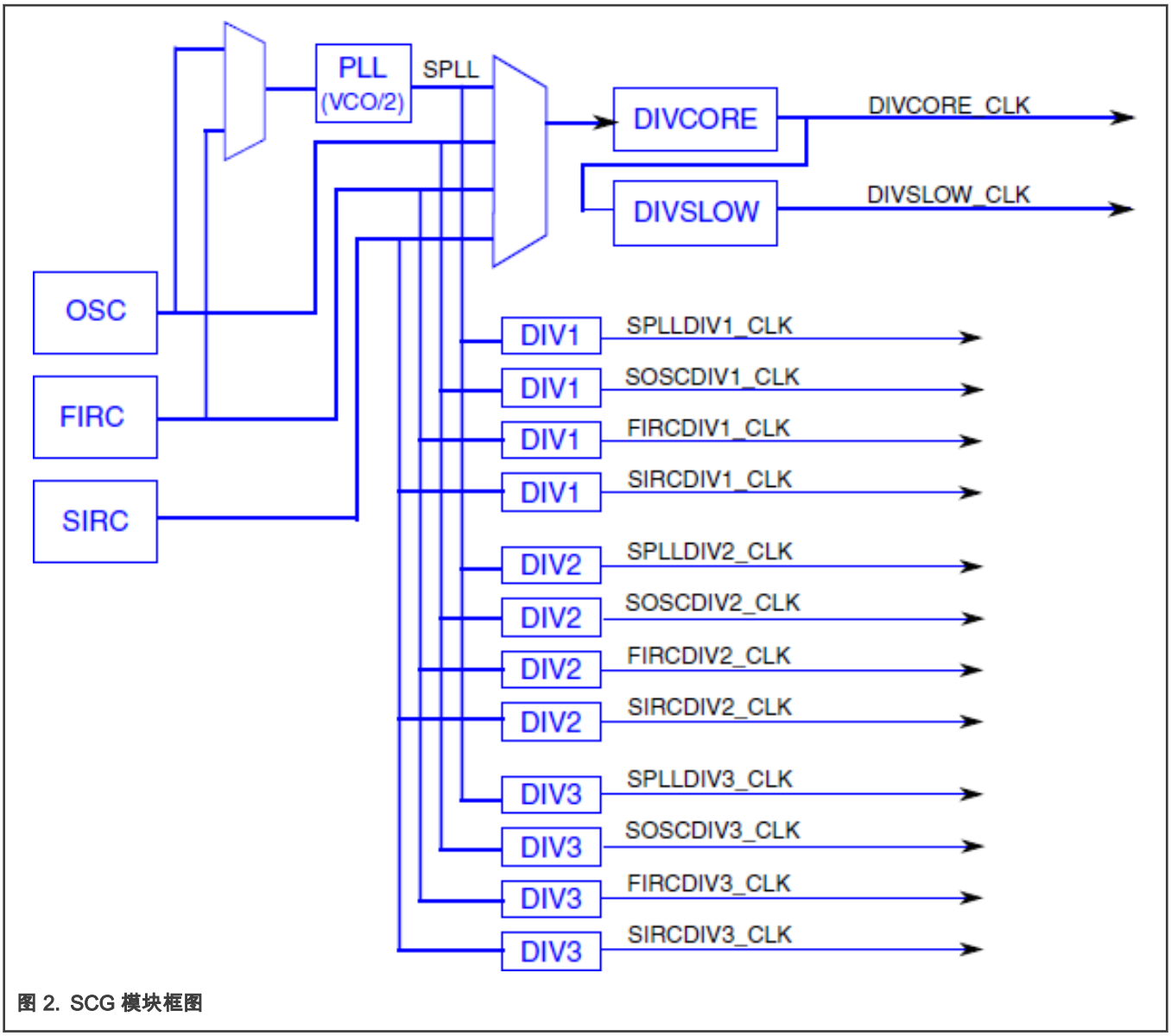

图 2. SCG 模块框图

3.2 SCG 和 MCG/MCG_Lite 之间的差异

表 2 展示了 MCG/Lite 和 SCG 模式之间的不同特点。

表 2. SCG vs. MCG/MCG_Lite 模式之间的不同特点

特点	SCG	MCG/Lite	优势
内部参考时钟源（IRC）频率，精度及可调整性	SIRC=IRC8M:2/8 MHz, 3 % 温度漂移 FIRC=IRC48M:48 – 60 MHz, 1 % 温度漂移，可编程范围 IRC 用户均可调整： SIRC 频率可以微调 +/-10%; FIRC - trim accuracy: ~0.7 % or	IRC8M:2/8MHz, 3 % 温度漂移： IRC48M: 固定频率，1% ~1.5 % 温度漂移； IRC 用户均不可以调整	SCG 提供比 MCG / _Lite 更高精度的参考时钟范围，同时对于不同的应用更加灵活

下页继续...

表 2. SCG vs. MCG/MCG_Lite 模式之间的不同特点（续上页）

特点	SCG	MCG/Lite	优势
	0.04 %		
独立时钟源 分频器可被禁用	四个时钟源都有独立的 PCC 标准化分频器，以为不同的外设生成独立的外设功能时钟。 分频器输出可以被禁用，可用于门控一组外部功能时钟。	IRC48M 没有分频器；只有 IRC8M 具有两个分频器，它们共用外设且输出无法禁止	标准化的分频器使 SCG 易于编码； 独立的时钟源和分频器可以轻易地动态更改外设比特率/频率/占空比；
每种操作模式单独配置的寄存器 (RUN/ VLPR/HSRUN)	Yes. SCG_RCCR/VCCR/HCCR	No. 只有一组 MCG_Cn	使用 SCG，在模式切换（RUN / VLPR / HSRUN）时进行自动时钟配置，模式切换无需额外的代码工作，编写程序十分简单。 而使用 MCG / Lite，需要花费大量代码才能进行时钟模式切换，编写程序比较复杂。
外部功能时钟与外部接口时钟/总线时钟分开	是的，允许外设比 CPU 平台更慢或更快速地运行； 允许不同组的外设由不同的特定功能时钟提供计时，例如，一组低功耗外设，其时钟频率低于高分辨率定时器/高速串行通信外设的频率。	是的，但是不允许外设比 CPU 更快运行。 分组外部功能时钟需要访问每个外设寄存器和/或 SIM 控 制寄存器	使用 SCG 和 PCC 组合对编程十分友好。通过外设时钟分组轻松降低功耗。

4 PCC（外部时钟控制）

外设时钟控制模块（PCC）提供外部时钟控制和配置寄存器。 该配置包括时钟多路复用器选择和时钟分频器配置。 与 SIM 模块中的旧时钟门控方法不同，PCC 中的每个外设都具有一个相同的时钟选项，这使得软件或 RTOS 可以轻松地实时管理所有外设的使用，尤其是在多核平台上。 此外，PCC 还可以提供检查外围设备存在和使用中状态的功能，可用于检测外设

图 3 展示了 PCC 模块功能图。

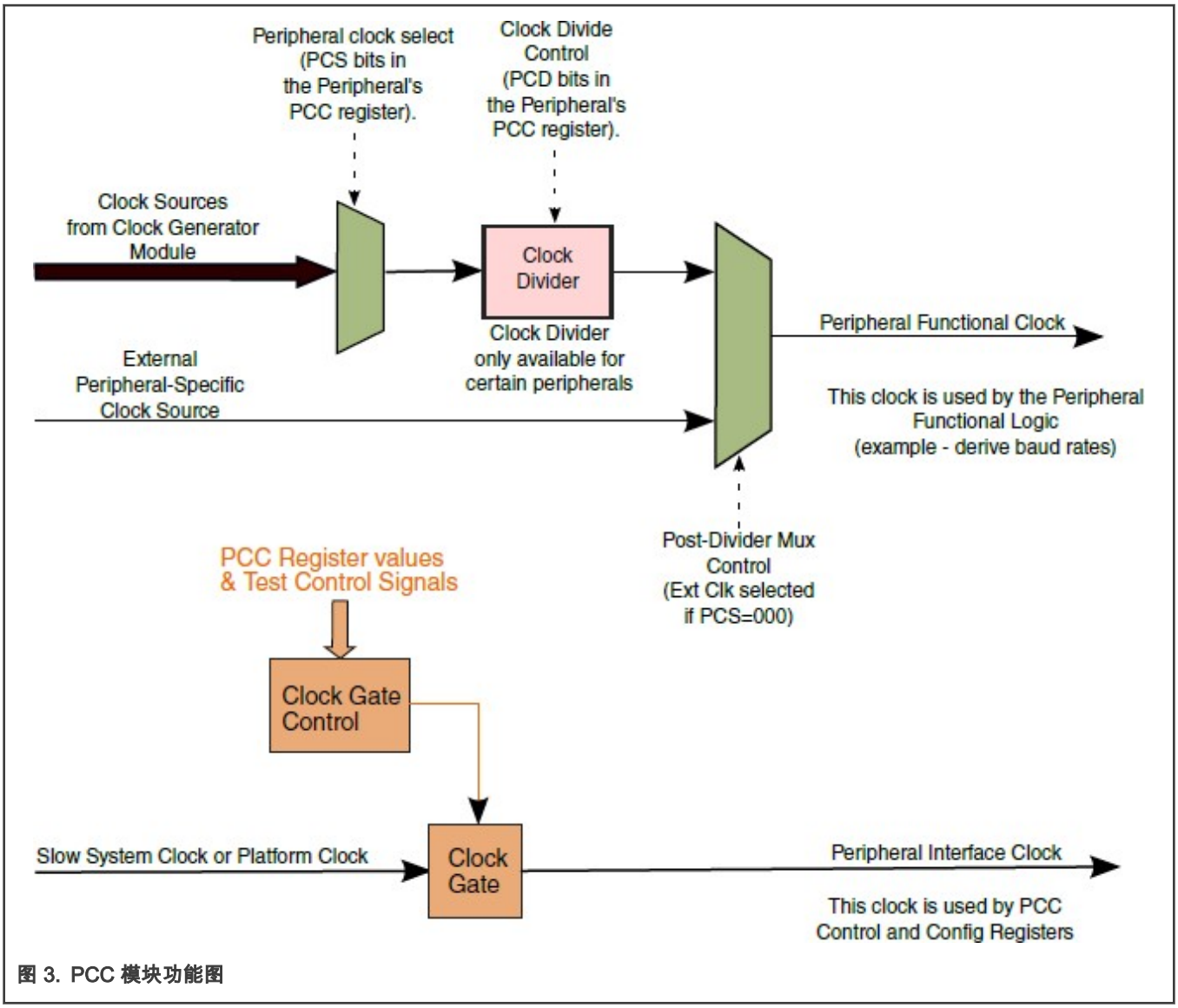

图 3. PCC 模块功能图

要启用和配置模块的接口和功能时钟，您必须执行以下步骤：

1. 检查外设存在情况 (可选)，通过检查相应 PCC 中的 PR 位。
2. 检查外设位在使用中 (可选)，如果正在使用外围设备，则软件需要采取正确的措施以停止外设模块的活动。
3. 清除 CGC 字段以关闭外围功能时钟。
4. 配置 PCS 字段以选择正确的外部功能时钟。
5. 将 CGC 位置 1 以启用外设时钟。

```
/* make sure the LPUART transmit and receive activity are stopped */
/* ... */
     PCC_LPUART0 &= ~PCC_CLKCFG_CGC_MASK;     /* gate off LPUART0 */
     PCC_LPUART0 = PCC_CLKCFG_PCS(3);    /* select functional clock to be FIRC */
     PCC_LPUART0 |= PCC_CLKCFG_CGC_MASK;      /* enable clock */
```

注意

1. 并非所有 PCC 外设都有 PCS 字段。
2. 仅当 CGC 位为 0（禁用时钟）时，才能写入 PCS 字段。 同样地，如果设置了 INUSE 标志，则该字段将被锁定。
3. 外设接口时钟可以是 DIVSLOW_CLK 或 DIVCORE_CLK。 有关详细信息请参考用户手册中每个外设的 PCCn 寄存器。

5 SCG 时钟模式转换

5.1 SCG 有效时钟模式

SCG 时钟模式切换比旧版 MCG 模块容易得多。图 4 显示了 SCG 支持的有效时钟模式转换。

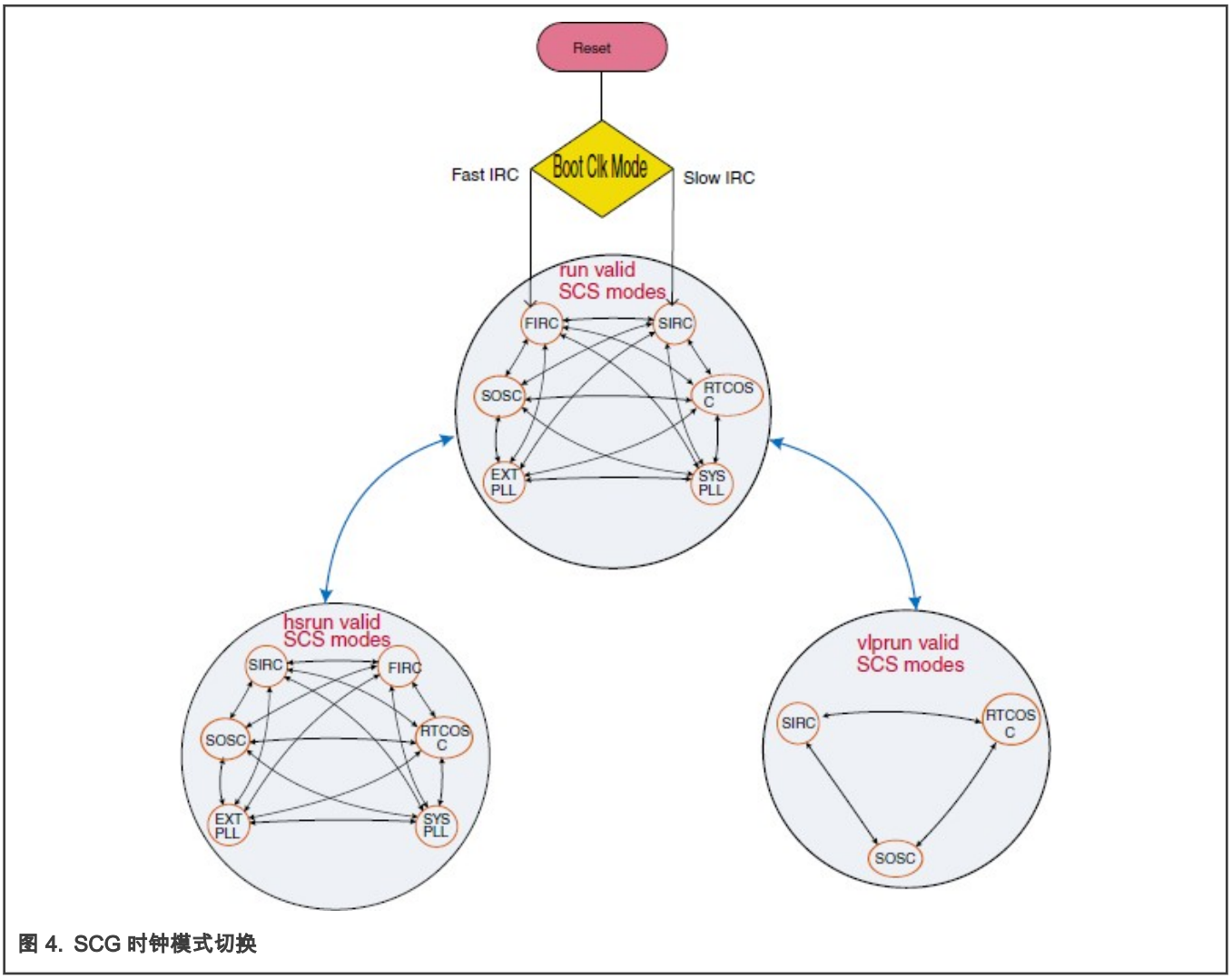

图 4. SCG 时钟模式切换

表 3 定义了 图 4 中每一个 SCG 时钟模式。

表 3. SCG 操作模式

模式	描述
低速内部参考时钟（SIRC）	当下列所有条件满足时将进入低速内部参考时钟（SIRC）： RUN MODE: 0010 被写入 RCCR[SCS]。 VLRUN MODE: 0010 被写入 VCCR[SCS] 以及 1 被写入 SIRCCSR[SIRCLPEN]。 HSRUN MODE: 0010 被写入 HCCR[SCS]。SIRCEN = 1 SIRCVLD = 1 在 SIRC 模式，SCSCLKOUT 及系统时钟来自低速内部参考时钟。 正如在 SIRCCFG[RANGE] 寄存器定义中描述的那样，两个频率范围可用于 SIRC 时钟。当 SIRC 时钟启用时对 SIRC 范围的设置将会被忽略。
高速内部参考时钟（FIRC）	当下列所有条件满足时将进入高速内部参考时钟（FIRC）： RUN MODE: 0011 被写入 RCCR[SCS]。 VLRUN MODE: 无效模式。将 SCG 编程为 FIRC 模式将被忽略。 HSRUN MODE: 0011 被写入 HCCR[SCS]。 FIRCEN = 1 FIRCVLD = 1 在 FIRC 模式, SCSCLKOUT 及系统时钟来自高速内部参考时钟。 正如在 FIRC[RANGE] 寄存器定义中描述的那样，两个频率范围可用于 FIRC 时钟。当 FIRC 时钟启用时对 FIRC 范围的设置将会被忽略。
系统震荡时钟（SOSC）	当下列所有条件满足时将进入系统震荡时钟（SOSC）模式： RUN 模式: 0001 被写入 RCCR[SCS]。 VLRUN 模式: 0001 被写入 VCCR[SCS]。 HSRUN 模式: 0001 被写入 HCCR[SCS]。 SOSCEN = 1 SOSCVLD = 1 在 SOSC 模式， SCSCLKOUT 及系统时钟来自外部系统震荡时钟（SOSC）。Sys PLL (SPLL) Sys PLL （SPLL）模式
Sys PLL (SPLL)	当下列所有条件满足时将进入 Sys PLL (SPLL) 模式： RUN MODE: 0110 被写入 RCCR[SCS]. VLRUNMODE: 无效模式.将 SCG 编程为 SPLL 模式将被忽略. HSRUNMODE: 0110 被写入 HCCR[SCS]. SPLLEN = 1 SPLLVLD = 1 在 SPLL 模式， SCSCLKOUT 和系统时钟来自 PLL 输出，由系统振荡器时钟(SOSC)或内部高速参考时钟 (FIRC).

下页继续...

表 3. SCG 操作模式（续上页）

模式	描述
	选定的 PLL 时钟频率锁定为乘积因数，其由相应的 VDIV 指定，乘以选定的 PLL 参考频率.PLL 的可编程参考分频器必须配置为产生有效的 PLL 参考时钟。 SPLL 输入时钟应在 8Mhz-16 Mhz 的范围内.

5.2 SCG 时钟模式转换示例

示例 1：转换至 SIRC 模式（系统电源模式 = RUN, SIRC 输出 = 8 M）：

```
#define SCG_SIRC 2
uint32_t tmp;

SCG->SIRCCSR |= SCG_SIRCCSR_SIRCEN_MASK; /* enable SIRC */
while(0 ==(SCG->SIRCCSR & SCG_SIRCCSR_SIRCVLD_MASK)); /* wait until ready and stable */
tmp = SCG->RCCR;
tmp &= ~SCG_RCCR_SCS_MASK;
tmp |= SCG_RCCR_SCS( SCG_SIRC );
SCG->RCCR = tmp;
While ( SCG_SIRC != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /*wait for system mode
switch */
```

注意

RCCR 只能使用 32 位来写入。

示例 2：转换至 FIRC 模式（系统电源模式 = RUN, FIRC 输出 = 48 M）：

```
#define SCG_FIRC 3
uint32_t tmp;
SCG->FIRCCSR |= SCG_FIRCCSR_FIRCEN_MASK; /* enable RIRC */
while(0 == (SCG->FIRCCSR & SCG_FIRCCSR_FIRCVLD_MASK)); /* wait until ready and stable */
tmp = SCG->RCCR;
tmp &= ~SCG_RCCR_SCS_MASK;
tmp |= SCG_RCCR_SCS(SCG_FIRC);
SCG->RCCR = tmp;
While (SCG_FIRC != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /* wait for system mode
switch */
```

示例 3：转换至 SOSC 模式（系统电源模式 = RUN, 外部参考时钟 = 8 M ,SOSC 输出 = 8 Mhz）：

```
#define SCG_SOSC 1
uint32_t tmp;
SCG->SOSCCFG &= ~SCG_SOSCCFG_RANGE_MASK;
SCG->SOSCCFG |= SCG_SOSCCFG_RANGE(2) | SCG_SOSCCFG_EREFS_MASK;
SCG->SOSCCSR |= SCG_SOSCCSR_SOSCEN_MASK; /* enable SOSC */
while(0 == (SCG->SOSCCSR & SCG_SOSCCSR_SOSCVLD_MASK)); /* wait until ready and stable */
tmp = SCG->RCCR;
tmp &= ~SCG_RCCR_SCS_MASK;
tmp |= SCG_RCCR_SCS(SCG_SOSC);
SCG->RCCR = tmp;
While (SCG_SOSC != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /* wait for system mode
switch */
```

示例 4：转换至 SPLL 模式（系统电源模式 = RUN, SPLL 源是 FIRC, SPLL 输出 = 72 M）：

```
#define SCG_SPLL 6
/* make sure FIRC is functional */
/* ... */
/* SPLL source = FIRC, mult=2, prediv=6, SPLL output = 48/(6)*18/2 = 72M */
SCG->SPLLCFG = SCG_SPLLCFG_SOURCE(1) | SCG_SPLLCFG_MULT(2) |SCG_SPLLCFG_PREDIV(5);
SCG->SPLLCSR |= SCG_SPLLCSR_SPLLEN; /* enable SPLL */
while(0 == (SCG->SPLLCSR & SCG_SPLLCSR_SPLLVLD_MASK)); /* wait for ready */
tmp = SCG->RCCR;
tmp &= ~SCG_RCCR_SCS_MASK;
tmp |= SCG_RCCR_SCS(SCG_SPLL);
SCG->RCCR = tmp;
While (SCG_SPLL != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /* wait for system mode
switch */
```

注意

1. SPLL 的输入范围是 8 – 32 Mhz。
2. SPLL 在内部默认具有分频器 (/2) 。 因此 SPLL 最终输出 = ((input clock / preDiv) * mult) / 2。

示例 5：转换至 SPLL 模式（系统电源模式 = RUN, SPLL 源是 SOSC, SOSC = 8 Mhz, SPLL 输出 = 72 M）：

```
#define SCG_SPLL 6
#define SCG_SOSC 1
/* make sure SOSC is enabled and functional */
/* ... */
/* SPLL source = SOSC, mult=2, prediv=1, SPLL output = 8/(1)*18/2 = 72M */
>SCG->SPLLCFG = SCG_SPLLCFG_SOURCE(0) | SCG_SPLLCFG_MULT(2) |SCG_SPLLCFG_PREDIV(0);
SCG->SPLLCSR |= SCG_SPLLCSR_SPLLEN; /* enable SPLL */
while(0 == (SCG->SPLLCSR & SCG_SPLLCSR_SPLLVLD_MASK)); /* wait for ready */
tmp = SCG->RCCR;
tmp &= ~SCG_RCCR_SCS_MASK;
tmp |= SCG_RCCR_SCS(SCG_SPLL);
SCG->RCCR = tmp;
While (SCG_SPLL != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /* wait for system mode
switch */
```

5.3 HSRUN 与 VLPR 模式下 SCG 配置

在 RUN 模式下为 HSRUN 及 VLPR 配置 SCG 很简单。然而用户需要注意时钟限制，正如在中解释的那样。

表 4. 时钟限制

时钟模式	可用的 SCG 来源	时钟限制
HSRUN	SOSC，SPLL SIRC，FIRC	DIVCORE 96 Mhz max DIVSLOW 24 Mhz max
VLPR	SOSC，SIRC	DIVCORE 8 Mhz max DIVSLOW 1 Mhz max

SCG 有单独用于 RUN/HSRUN/VLPR 模式的时钟控制寄存器，使用户在模式之间变换更加容易。按以下步骤进入 HSRUN/VLPR：

1. 像 RUN 模式那样为 SCG 配置时钟源。
2. 跟据你想进入的模式配置 SCG RCCR/VCCR/HCCR。
3. 设置系统模式控制器以进入特定的 RUN 模式。

4. 读取 SMC->PMSTAT 寄存器以检查当下的系统模式。

示例：转换为 SPLL 模式（系统电源模式 = HSRUN, SPLL 的源是 SOSC, SOSC = 8 Mhz, SPLL 输出 = 96 M, 同时 LPUART 功能时钟为 SOSC）：

```
#define SCG_SPLL 6
#define SCG_SOSC 1
/* make sure SOSC is enabled and functional */
/* ... */
/* SPLL source = SOSC, mult=8, prediv=1, SPLL output = 8/(1)*24/2 = 96M */
SCG->SPLLCFG = SCG_SPLLCFG_SOURCE(0) | SCG_SPLLCFG_MULT(8) |SCG_SPLLCFG_PREDIV(0);
SCG->SPLLCSR |= SCG_SPLLCSR_SPLLEN; /* enable SPLL */
while(0 ==(SCG->SPLLCSR & SCG_SPLLCSR_SPLLVLD_MASK)); /* wait for ready */
tmp = SCG->HCCR;
tmp &= ~SCG_HCCR_SCS_MASK;
tmp |= SCG_HCCR_SCS( SCG_SPLL );
SCG->HCCR = tmp;
While ( SCG_SPLL != ((SCG_CSR_SCS_MASK & SCG_CSR) >> SCG_CSR_SCS_SHIFT)); /* wait for system mode
switch*/
/* enter HSRUN */
SMC->PMPROT |= SMC_PMPROT_AHSRUN_MASK;
SMC->PMCTRL |= SMC_PMCTRL_RUNM(3);
while((SMC->PMSTAT & 0x80) == 0);
/* set LPUART source to be SOSC */
PCC_LPUART0 &= ~PCC_CLKCFG_CGC_MASK;
PCC_LPUART0 = PCC_CLKCFG_PCS(SCG_SOSC);
PCC_LPUART0 |= PCC_CLKCFG_CGC_MASK;
```

5.4 STOP 模式中时钟配置

许多时钟源在 SCG 模式下 可以在各种 STOP 模式下运行，并且每一个时钟源在各自的控制寄存器中都有禁止使能位。表 5展示了 STOP 模式下时钟行为的详细信息。

表 5. STOP 模式下的时钟行为

模块	STOP	VLPS	LLSx	VLLSx
SCG	SOSC SIRC FIRC SPLL 可以被使能	SOSC SIRC 可以被使能	SOSC 可以被使能	SOSC 可以被使能

表 6 描述了在 STOP 模式下如何使能每一个时钟源。

表 6. STOP 模式如何使能时钟

SCG 时钟源	STOP 模式下如何使能时钟
SIRC	当下列所有条件都满足时，SIRCCLK 在 Normal Stop 和 VLPS 模式下可用： • SIRCCSR[SIRCEN] = 1 • SIRCCSR[SIRCSTEN] = 1 • SIRCCSR[SIRCLPEN] = 1 in VLPS
FIRC	当下列所有条件都满足时，FIRCCLK 仅在 Normal Stop 下可用： • FIRCCSR[FIRCSTEN] = 1

下页继续...

表 6. STOP 模式如何使能时钟（续上页）

SCG 时钟源	STOP 模式下如何使能时钟
SOSC	当下列所有条件都满足时，SOSCLK 在以下低功耗模式（Normal Stop，VLPS，LLS）可用。在 VLLS stop 模式下，SOSCLK 被禁用： • SOSCCSR[SOSCEN] = 1 • SOSCCSR[SOSCSTEN] = 1 • SOSCCSR[SOSCLPEN] = 1（只在 Low-Power Stop 模式（VLPS 和 LLS）下使用）
SPLL	当下列所有条件都满足时，SPLLCLK 在 Normal Stop 下可用： • SPLLCSR[SPLLEN] = 1 • SPLLCSR[SPLLSTEN] = 1

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — 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: 2020 年 1 月
Document identifier: AN12680

arm