AN12680

Clock Management and Distribution in K32L2A

Rev. 0 — January 2020

Application Note

Contents

1 Introduction

This application note explains the clock architecture and clock distribution in K32L2A. Especially the two new clock-related modules different in K32L2A:

- SCG (System Clock Generator)
- PCC (Peripheral Clock Control)

SCG provides broad range of reference clock with more accuracy than MCG/ _Lite, and is more flexible to different applications. With the help of the SCG, Core clock and peripherals clock can be route from different clock source. It make the peripherals clock can be even higher than Core/Bus clock.

PCC provides peripheral clock control and configuration register, such as clock multiplexors and clock dividers. Unlike the old clock gate and configuration in SIM module, the PCC module is easier to select peripheral clock source and software-oriented design makes code more compatible.

2 Clock architecture

The heart of the clocking architecture is the System Clock Generator (SCG) module. The SCG controls four clock sources (SIRC, FIRC, SOSC, and SPLL) that can then be distributed to the main core platform, the memory modules, and the peripherals. The peripherals in general have two clocks, one being the peripheral interface clock which is used by the core/DMA to interface with the peripheral's registers, the second being the peripheral functional clock which is used for the main timing function of the peripheral (for instance it sources the baud rate for a serial communications peripheral, or is the input clock to the counter of a timer peripheral). The peripheral interface clock generally comes directly from the SCG to the peripheral. The second peripheral functional clock is selected via the Peripheral Clock Control (PCC) module. The SCG provides additional peripheral functional clocks with optional Dividers via the PCC module.

NOTE

1. All peripheral functional clocks are asynchronous clock.
2. Peripheral interface clock is also known as bus clock for other devices with MCG/_Lite.

Figure 1 shows the various clock sources and clock trees for K32L2A.

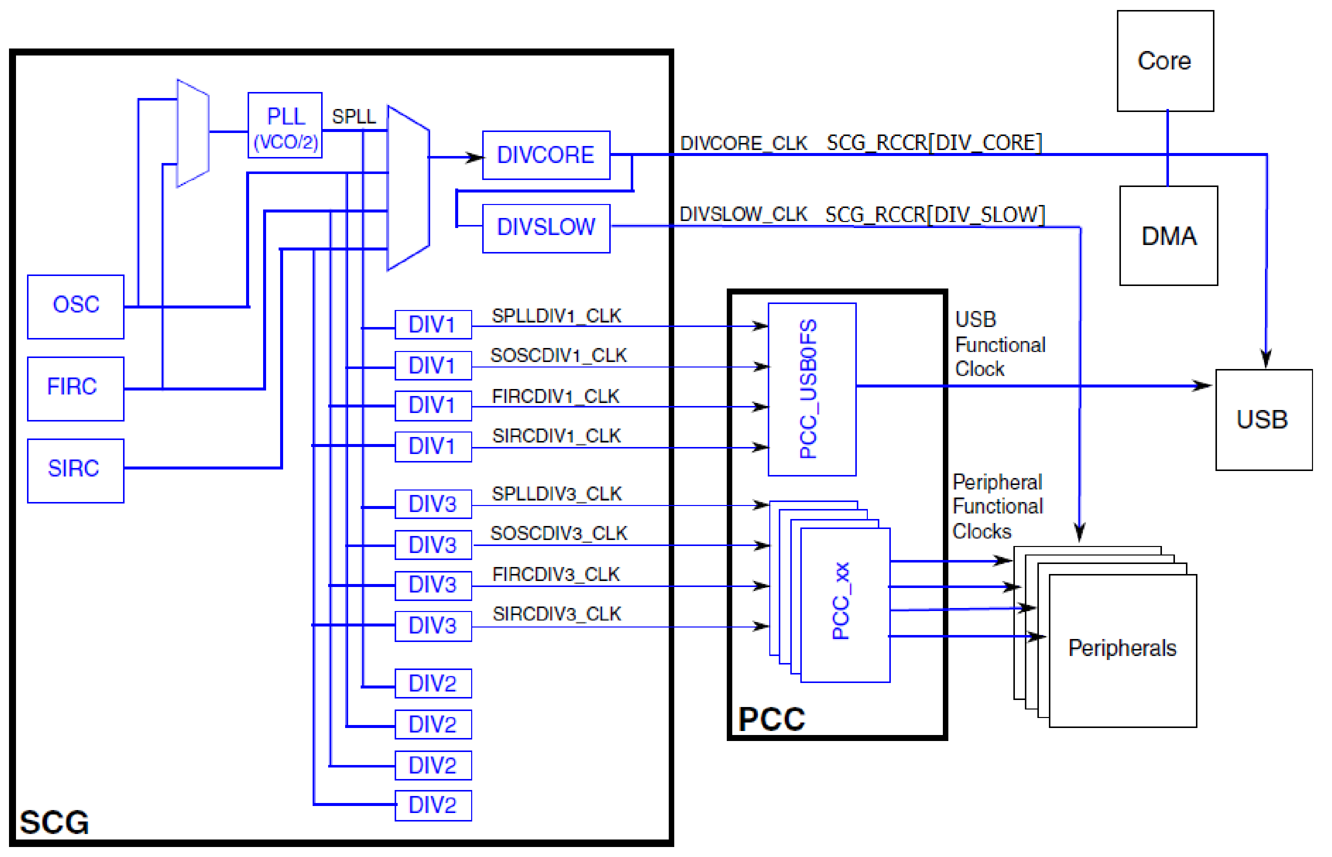

Figure 1. K32L2A High-Level Clock Distribution

In general, the DIVCORE_CLK is the old Core/Platform Clock, and DIVSLOW_CLK is the old Bus clock. Both the DIVCORE_CLK and DIVSLOW_CLK can act as peripheral interface clocks. The peripheral function clock is routed from DIV1 - DIV3_CLK. CLK_SRC is the SCG clock source and can be selected by the PCC module.

NOTE

1. K32L2A flash clock is source from DIV_SLOW clock.
2. All peripheral except USB module use <CLK_SRC>DIV1 clock as functional clock, the USB module use <CLK_SRC>DIV3 as functional clock.

Table 1 describes the details of clock definition in K32L2A:

Table 1. Detailed Clock Summary

Clock name	Run mode-clock frequency	VLPR mode clock frequency	HSRUN mode clock frequency	Clock Source	Clock can be disabled when
DIVCORE_CLK	Up to 72 MHz	Up to 8 MHz	Up to 96 MHz	SCG	When CPU is in any stop modes except for Partial stop modes.
DIVSLOW_CLK	Up to 24 MHz	DIVCORE_CLK Divide by 4 or more.	Up to 24 MHz	SCG	When CPU is in any stop modes except for partial stop modes.

Table continues on the next page...

Table 1. Detailed Clock Summary (continued)

SOSCDIV1_CLK, SOSCDIV3_CLK	Up to 48 MHz	MHz Up to 8 MHz	Up to 48 MHz	SCG	If not being used by any peripheral and/or DIVCORE_CLK or feeding the PLL
SPLLDIV1_CLK, SPLLDIV3_CLK	Up to 72 MHz	PLL disabled	Up To 96 MHz	SCG	If not being used by any peripheral and/or DIVCORE_CLK
FIRCDIV1_CLK, FIRCDIV3_CLK	Up to 60 MHz	FIRC is disabled	Up to 60 MHz	SCG	If not being used by any peripheral and/or DIVCORE_CLK or feeding the PLL
SIRCDIV1_CLK, SIRCDIV3_CLK	8 MHz	8 MHz	8 MHz	SCG	If not being used by any peripheral

3 SCG(System Clock Generator)

3.1 SCG architecture

The system clock generator (SCG) module provides the system clocks of the MCU. The SCG contains:

- SOSC – output of the external oscillator (a crystal or externally applied clock input).
- SIRC – output of the slow (8 MHz) internal RC oscillator
- FIRC – output of the fast (48 MHZ) internal RC oscillator
- SPLL – output of the PLL, which is sourced by either the SOSC reference clock or the FIRC.

The SCG can select either one of the four output clocks as the source for the MCU system clocks. It also has dividers that can divide the clock output for the DIVCORE/DIVSLOW and peripheral functional clock.

Figure 2 shows the SCG module block diagram.

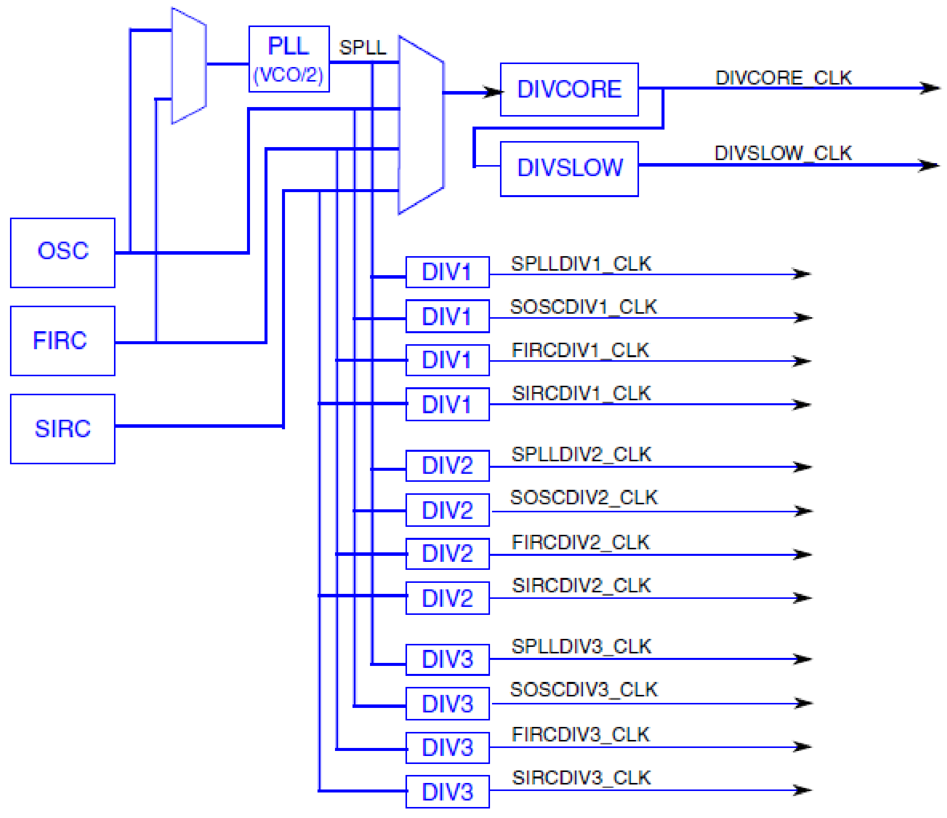

Figure 2. System Clock Generator (SCG) block diagram

3.2 Difference between SCG and MCG./MCG_Lite

Table 2 shows those different features between the old MCG/Lite and SCG mode.

Table 2. SCG vs. MCG/MCG_Lite

Feature	SCG	MCG/Lite	Advantages

Internal Reference clock (IRC) frequency, accuracy, and trimmable	SIRC=IRC8M: 2/8 MHz, 3 % temp drift
	FIRC=IRC48M: 48 – 60 MHz, 1 % temp drift, programmable range
	Both IRC user trimmable:
	SIRC - frequency can trim shift +/-10%;
	FIRC - trim accuracy: ~0.7 % or 0.04 %
		IRC8M: 2/8MHz, 3 %
		temp drift;
		IRC48M: fixed frequency, 1 % ~1.5 % temp drift;
		Both IRC are non-user trimmable
			SCG provides broad range of reference clock with more accuracy than MCG/_Lite, more flexible to different applications

Independent clock source dividers able to be disabled	All 4 clock sources have independent standardized dividers to PCC to	IRC48M has no divider; only IRC8M	Standardized divider makes SCG easy for coding;

Table continues on the next page...

Table 2. SCG vs. MCG/MCG_Lite (continued)

	generate independent peripheral functional clocks for different peripherals; and divider output can be disabled which can be used to gate a group of peripheral functional clocks.	has 2 dividers, each shared with multiple peripherals, and divider output cannot be disabled	**Independent clock source and divider make it easy to change peripheral bit rate/frequency/ duty cycle on the fly;**
Separate configuration register for each operating mode (RUN/ VLPR/HSRUN)	Yes. SCG_RCCR/VCCR/HCCR	No. Only one set of MCG_Cn	With SCG, automatic clock configuration when mode switching (RUN/VLPR/ HSRUN), no additional code effort for mode switch, user-friendly feature. With MCG/ Lite, need much code effort to do clock mode switch, not user-friendly.
Separate peripheral functional clock from peripheral interface clock/bus clock	Yes. allows peripherals to operate either slower or faster than CPU platform; Allows different groups of peripherals to be clocked by different specific functional clocks, e.g. a group of low-power peripherals to be clocked at lower frequency than high resolution timers/fast serial communication peripherals.	Yes, but not allow peripheral to operate faster than CPU. Grouping peripheral functional clock need access each peripheral register and/or SIM control register	User-friendly peripheral clock grouping with SCG plus PCC, simple for coding; Easy to reduce power consumption with peripheral clock grouping.

4 PCC(Peripheral Clock Control)

The Peripheral Clock Control module (PCC) provides peripheral clock control and configuration registers. The configuration including clock multiplexors select and clock dividers configuration. Unlike the old clock gate control method in SIM module, in PCC each peripheral has one identical clock option, which makes it easy for software or RTOS to manage all peripheral usage in real time, especially in multiple core platforms. In addition, the PCC can also provide the capability to check the peripheral exist and in-use status, which can be used for peripheral discovery

Figure 3 shows the PCC module functional diagram.

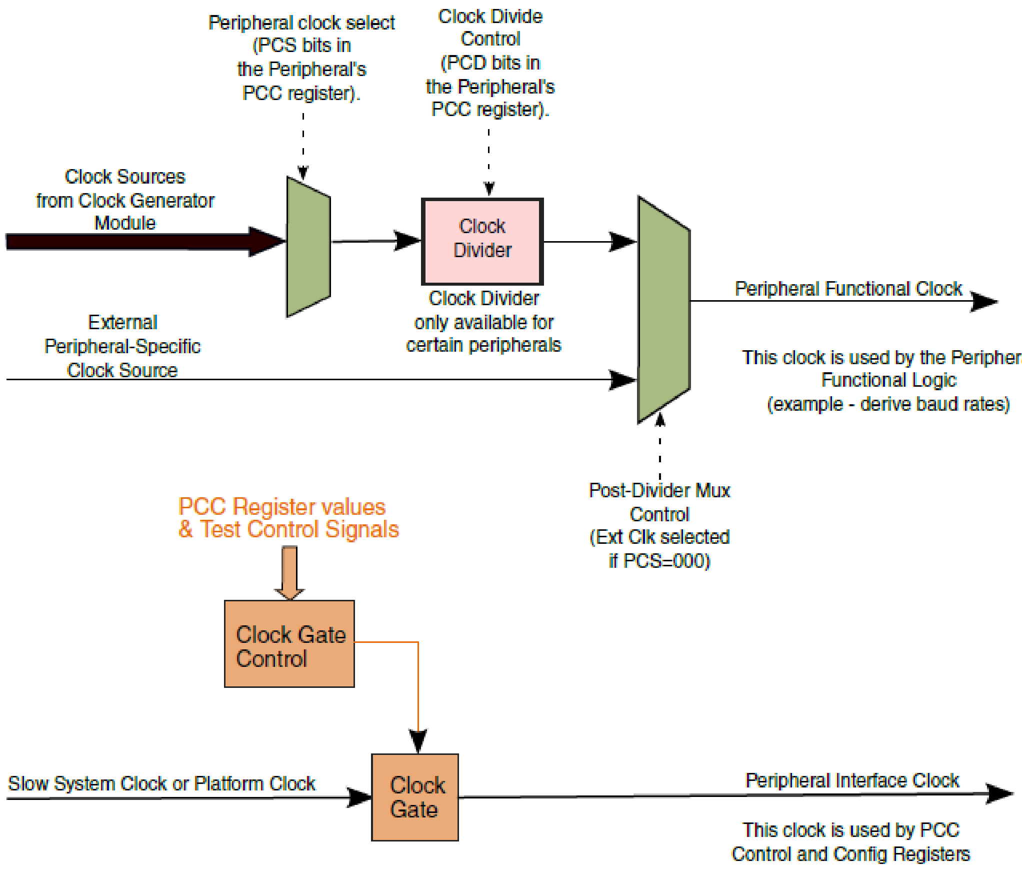
Peripheral clock select (PCS bits in the Peripheral's PCC register).
Clock Divide Control (PCD bits in the Peripheral's PCC register).
Clock Sources from Clock Generator Module
Clock Divider
Clock Divider only available for certain peripherals
External Peripheral-Specific Clock Source
Peripheral Functional Clock
This clock is used by the Peripheral Functional Logic (example - derive baud rates)
Post-Divider Mux Control (Ext Clk selected if PCS=000)
PCC Register values & Test Control Signals
Clock Gate Control
Slow System Clock or Platform Clock
Clock Gate
Peripheral Interface Clock
This clock is used by PCC Control and Config Registers

Figure 3. Peripheral Clock Control (PCC) block diagram

To enable and configure the module’s interface and functional clock, you must carry out the following steps:

1. Check the peripheral exists (optional), by checking the PR bit in corresponding PCC register.
2. Check the peripheral bit is in-use (optional), if peripheral is in-use, software need to do correct action to stop peripheral module activity.
3. Clear the CGC filed to gate off the peripheral functional clock.
4. Configurate the PCS field to select correct peripheral functional clock.
5. Set the CGC bit to 1 to enable the clock of peripheral.

```
/* 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 */
```

NOTE

1. Not all PCC peripheral has the PCS filed.
2. The PCS field can only be written when the CGC bit is 0 (clock disabled). Likewise, if the INUSE flag is set, this field is locked.
3. The peripheral interface clock can be either DIVSLOW_CLK or DIVCORE_CLK. See each peripherals PCCn register in reference manual for details.

5 SCG Clock Mode Transitions

5.1 SCG valid clock mode

The SCG clock mode switch is much easier than the old MCG module; Figure 4 shows the valid clock mode transitions supported by SCG.

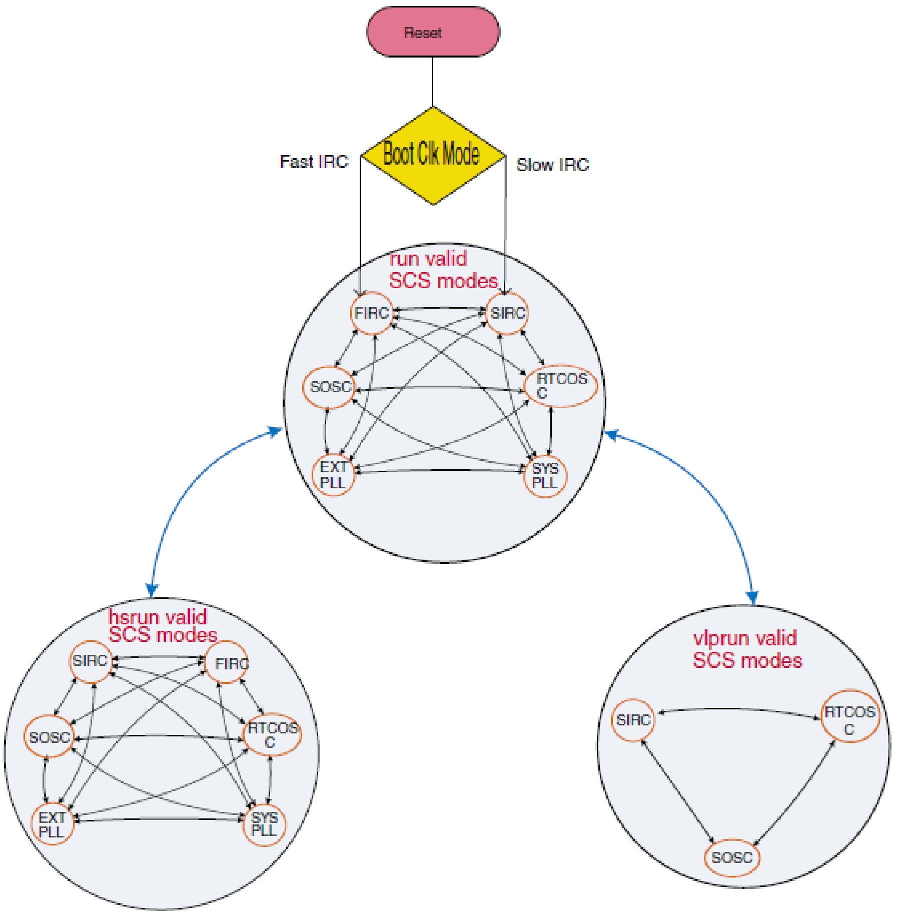

Figure 4. SCG clock mode transition

Table 3 defines each of the SCG clock modes shown in Figure 4.

Table 3. SCG mode of operation

Mode	Description

Table continues on the next page...

Table 3. SCG mode of operation (continued)

Slow Internal Reference Clock (SIRC)	Slow Internal Reference Clock (SIRC) mode is entered when all the following conditions occur: RUN MODE: 0010 is written to RCCR[SCS]. VLRUN MODE: 0010 is written to VCCR[SCS] and 1 is written to SIRCCSR[SIRCLPEN]. HSRUN MODE: 0010 is written to HCCR[SCS]. SIRCEN = 1 SIRCVLD = 1 In SIRC mode, SCSCLKOUT and system clocks are derived from the slow internal reference clock. Two frequency ranges are available for SIRC clock as described in the SIRCCFG[RANGE] register Definition. Changes to SIRC range settings are ignored when SIRC clock is enabled.
Fast Internal Reference Clock (FIRC)	Fast Internal Reference Clock (FIRC) mode is entered when all the following conditions occur: RUN MODE: 0011 is written to RCCR[SCS]. VLRUN MODE: Invalid mode. Programming SCG into FIRC mode is ignored. HSRUN MODE: 0011 is written to HCCR[SCS]. FIRCEN = 1 FIRCVLD = 1 In FIRC mode, SCSCLKOUT and system clocks are derived from the fast internal reference clock. Two frequency range settings are available for FIRC clock as described in the FIRC[RANGE] Register definition. Changes to FIRC range settings are ignored when FIRC clock is enabled.
System Oscillator Clock (SOSC)	System Oscillator Clock (SOSC) mode is entered when all the following conditions occur: RUN MODE: 0001 is written to RCCR[SCS]. VLRUN MODE: 0001 is written to VCCR[SCS]. HSRUN MODE: 0001 is written to HCCR[SCS]. SOSCEN = 1 SOSCVLD = 1 In SOSC mode, SCSCLKOUT and system clocks are derived from the external System Oscillator Clock (SOSC). Sys PLL (SPLL) Sys PLL (SPLL) mode
Sys PLL (SPLL)	Sys PLL (SPLL) mode is entered when all the following conditions occur:

Table 3. SCG mode of operation (continued)

	RUN MODE: 0110 is written to RCCR[SCS]. VLRUN MODE: Invalid mode. Programming SCG into SPLL mode is ignored. HSRUN MODE: 0110 is written to HCCR[SCS]. SPLLEN = 1 SPLLVLD = 1 In SPLL mode, the SCSCLKOUT and system clocks are derived from the output of PLL which is controlled by either the System Oscillator (SOSC) clock or the Fast internal reference clock (FIRC). The selected PLL clock frequency locks to a multiplication factor, as specified by its corresponding VDIV, times the selected PLL reference frequency. The PLL's programmable reference divider must be configured to produce a valid PLL reference clock. The SPLL input clock should be at range 8Mhz-16 Mhz.

5.2 SCG clock mode transitions examples

Examples 1: Switching to SIRC mode (system power mode = RUN, SIRC output = 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 */
```

NOTE

RCCR can only be written using a 32-bit write.

Examples 2: Switching to FIRC mode (system power mode = RUN, FIRC output = 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 */
```

Examples 3: Switching to SOSC mode (system power mode = RUN, external reference clock = 8 M ,SOSC output = 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 */
```

Examples 4: Switching to SPLL mode (system power mode = RUN, SPLL source is FIRC, SPLL output = 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 */
```

NOTE

1. The input range of SPLL is 8 – 32 Mhz
2. SPLL has an output divider (/2) inside by default. So the final SPLL output = ((input clock / preDiv) * mult) / 2

Examples 5: Switching to SPLL mode (system power mode = RUN, SPLL source is SOSC, SOSC = 8 Mhz, SPLL output = 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 SCG configuration in HSRUN and VLPR mode

Configuring SCG for HSRUN and VLPR is easy as in RUN mode. However the user needs to pay attention to the clock limitation, as explained in Table 4.

Table 4. Clock limitations

Clock Mode	Available SCG Source	Clock Limitation
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 has separate clock control register for RUN/HSRUN/VLPR mode; make it easier for user to switch between those modes.

To enter HSRUN/VLPR, need following steps:

1. Configure SCG clock source just like you did in normal RUN mode,
2. Configure the SCG RCCR/VCCR/HCCR according to the mode you want to enter.
3. Configure the System Mode Controller (SMC) to enter specific RUN mode.
4. Read SMC->PMSTAT register to check the current system mode.

Examples: Switching to SPLL mode (system power mode = HSRUN, SPLL source is SOSC, SOSC = 8 Mhz, SPLL output = 96 M, while LPUART functional clock is 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 Clock configuration in STOP mode

Many clock sources in SCG mode can be functional in various STOP modes, and each clock source have stop enable bit in their control register. Table 5 shows the detail for each clock behavior in STOP mode.

Table 5. How to enable clock in STOP mode

Modules	STOP	VLPS	LLSx	VLLSx
SCG	SOSC SIRC FIRC SPLL can be enabled.	SOSC SIRC can be enabled	SOSC can be enabled	SOSC can be enabled

Table 6 describes how to enable each clock source in STOP mode:

Table 6. How to enable clock in STOP mode

SCG clock source	How to enable clock in STOP mode
SIRC	SIRCCLK is available in Normal Stop and VLPS mode when all the following conditions become true: • SIRCCSR[SIRCEN] = 1 • SIRCCSR[SIRCSTEN] = 1 • SIRCCSR[SIRCLPEN] = 1 in VLPS

FIRC	FIRCCLK is available only in Normal Stop mode when all the following conditions become true:
	
	FIRCCSR[FIRCEN] = 1
	
	FIRCCSR[FIRCSTEN] = 1

SOSC	SOSCLK is available in following low-power stop modes (Normal Stop, VLPS, LLS) when all the Below conditions are true. In VLLS stop mode, SOSCLK is disabled.
	
	SOSCCSR[SOSCEN] = 1
	
	SOSCCSR[SOSCSTEN] = 1
	
	SOSCCSR[SOSCLPEN] = 1 (required only for Low-Power Stop modes (VLPS and LLS)

SPLL	SPLLCLK is available in Normal Stop mode when all the following conditions are true:
	
	SPLLCSR[SPLLEN] = 1
	
	SPLLCSR[SPLLSTEN] = 1