NXP Semiconductors
Application Notes

Document Number:AN5425
Rev. 1, 05/2018

Power Management for S32K1xx

by: NXP Semiconductors

Contents

1. Introduction

The power consumption of devices and the implications around designing for low power are common topics currently. The S32K1xx family includes internal power management features that can be used to control the microcontroller's power usage and assist reaching the targets of embedded designs.

This application note discusses how to use the power management system, provide use case examples and shows current measurement results for these cases. Tips are given for using each of the power modes available on S32K1xx family.

2. Overview of power modes

The typical power modes in legacy cores and other embedded systems are Run, Wait and Stop. The ARM® Cortex™ M4 and M0+ power modes are Run, Sleep and Deep Sleep. The extended power modes and their relationship with typical and ARM Cortex M4 and M0+'s modes are depicted in below figure

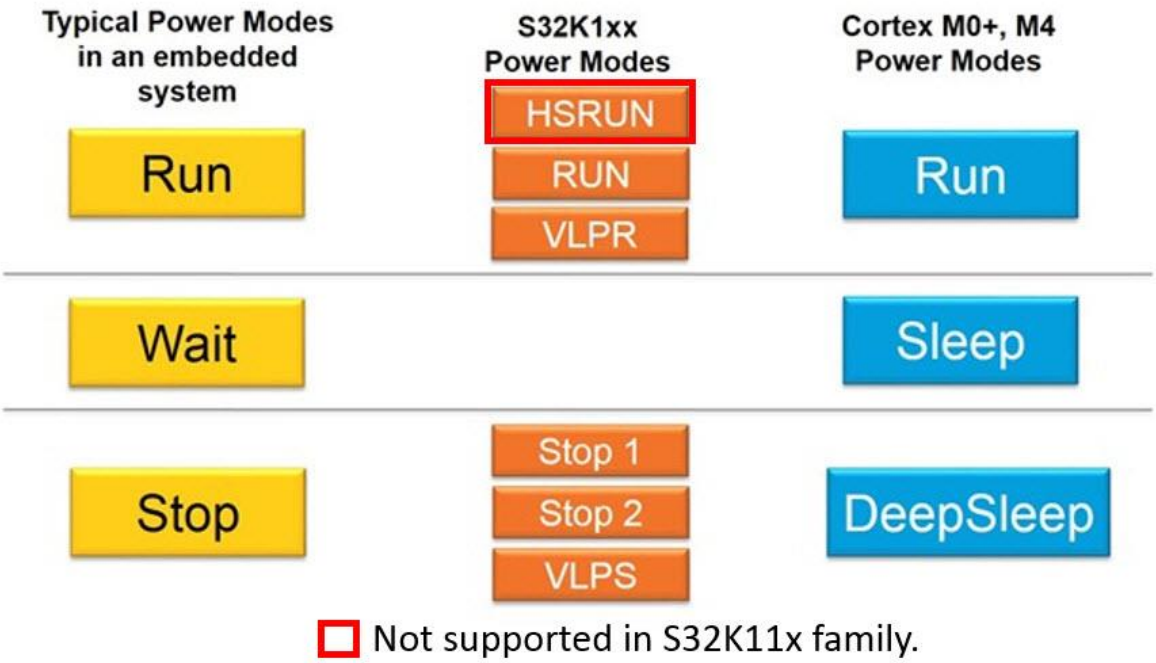

Figure 1. Power modes comparison

2.1. ARM Cortex-M4 and M0+ power modes implementation

The ARM Cortex-M4 and M0+ cores have three primary modes of operation: Run, Sleep, and Deep Sleep. The Wait For Interrupt (WFI) instruction invokes sleep and deep sleep modes for the S32K1xx chips.

Figure 2 shows the Cortex -M4 and M0+ architecture for low power implementation. The sleep and deep sleep states are architected in hardware to control the clock to the core and interrupt controller. When entering sleep, the NVIC logic remains active and either interrupts or a reset can wake the core from sleep. When entering deep sleep, an Asynchronous Wakeup Interrupt Controller (AWIC) is used to wake the MCU from a select group of sources. These sources are described in the S32K1xx family reference manual, please check the "Asynchronous Wake-Up Interrupt controller (AWIC) configuration " section within "Core Overview" chapter.

Using the Sleep-On-Exit feature, the ARM Cortex cores have one more way to enter low power modes. In the System Control Block is a register called the System Control Register (SCR) that contains several control bits related to sleep operation. The SLEEPDEEP bit selects whether Sleep or Deep Sleep mode is selected. Setting the SLEEPONEXIT bit to 1, enables the processor to immediately enter sleep modes when it completes the execution of all exception handlers. For more information please refer to ARM Cortex-M4 Devices Generic User Guide and ARM Cortex-M0+ Devices Generic User Guide.

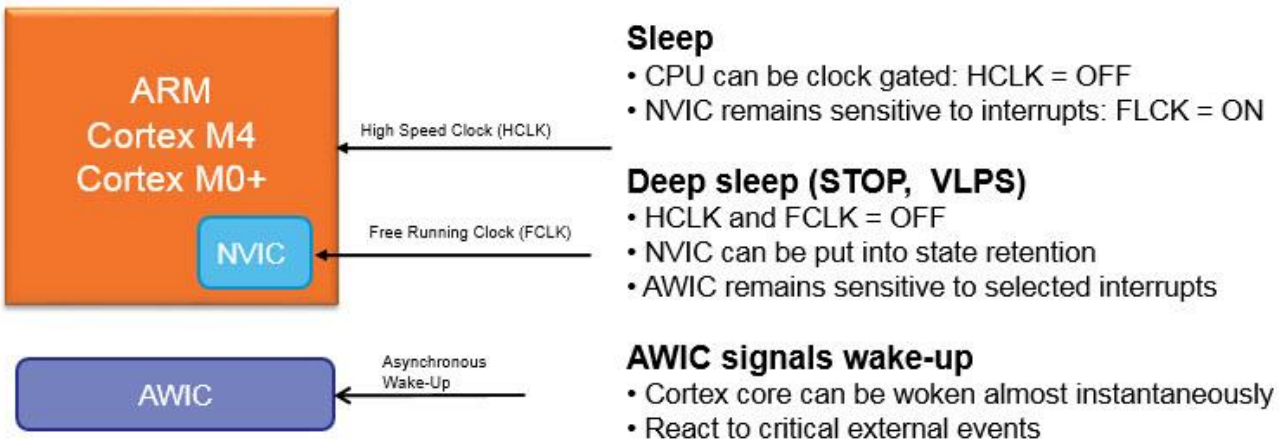

Figure 2. ARM Cortex M4 and M0+ architecture

3. Power modes description

S32K1xx provides multiple power options allowing users to optimize power consumption for the level of functionality needed.

Depending on the user application's requirements, a variety of power modes are available that provide state retention, partial power down or full power down on certain logic and/or memory. Input / Output (I/O) states are maintained during all power modes. For more information about module functionality in different power mode refer to chapter ***Modules in power modes*** in this Application note or S32K1xx' reference manual. The following table compares the available power modes.

Table 1. **S32K1xx power modes**

S32K1xx mode	Description	Core mode	Normal recovery method
Normal Run	Default mode out of reset, on-chip voltage regulator is on (internal supply is fully regulated).	Run	-
High Speed Run (HSRUN)[1]	Allows maximum performance of the chip. In this mode, the chip can operate at a higher frequency as compared to Normal Run mode but with restricted functionalities. Internal clocking requirements should be met.[2]	Run	-
Very Low Power Run (VLPR)	On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. • Reduced-frequency flash memory access mode (1 MHz) • LVD is off • SIRC provides a low power 4 MHz source for the core, the bus and the peripheral clocks.	Run	-

S32K1xx mode	Description	Core mode	Normal recovery method
Very Low Power Stop (VLPS) via WFI instruction	Places the chip in a static state with Low Voltage Detect (LVD) operation off. This is the lowest-power mode in which pin interrupts are functional. • Some peripheral clocks are stopped.[3] • LPTMR, RTC and CMP can be used. • NVIC is disabled. • AWIC is used to wake from interrupt. • Core is gated off. • All SRAM is operational (content is retained and I/O states are maintained).	Deep Sleep	Interrupt (or Reset)
Stop 1 (via WFI instruction)	Places the chip in static state. LVD protection is maintained. • NVIC is disabled. • AWIC is used to wake up from interrupt • Some peripheral clocks are stopped.[3] • The core clocks, system clocks and bus clock are all gated.	Deep Sleep	Interrupt (or Reset)
Stop 2 (via WFI instruction)	Places the chip in static state. LVD protection is maintained. • NVIC is disabled. • AWIC is used to wake up from interrupt • Some peripheral clocks are stopped.[3] • Only the core and system clocks are gated, but the bus clock remains active. The bus masters and bus slaves clocked by the system clock enter stop mode, but the bus slaves clocked by the bus clock remain in run mode. The clock generators in the SCG and the PMC's on-chip regulator also remain in Run mode.[4]	Deep Sleep	Interrupt (or Reset)

1. HSRUN is not available in S32K11x series of device.
2. Core and System clock must be 112 MHz or less, Bus clock must be programmed to 56 MHz or less and an integer divider of the core clock. Flash clock must be programmed to 28 MHz or less, the core clock to flash clock ratio is limited to a maximum value of 8.
3. See Modules in power modes chapter in this application note for more details.
4. The following can initiate an exit from STOP: Reset, an asynchronous interrupt from a bus master (valid also for STOP1) and A synchronous interrupt from a bus slave clocked by the bus clock (Valid only for STOP2).

3.1. Power mode transitions

The following figure shows the power mode transitions. Any reset always brings the chip back to Normal Run state. Depending on the needs of user application, a variety of stop modes are available that allow the state retention, partial power down or full power down of certain logic and/or memory. I/O states are held as well as registers retain their content in all modes of operation.

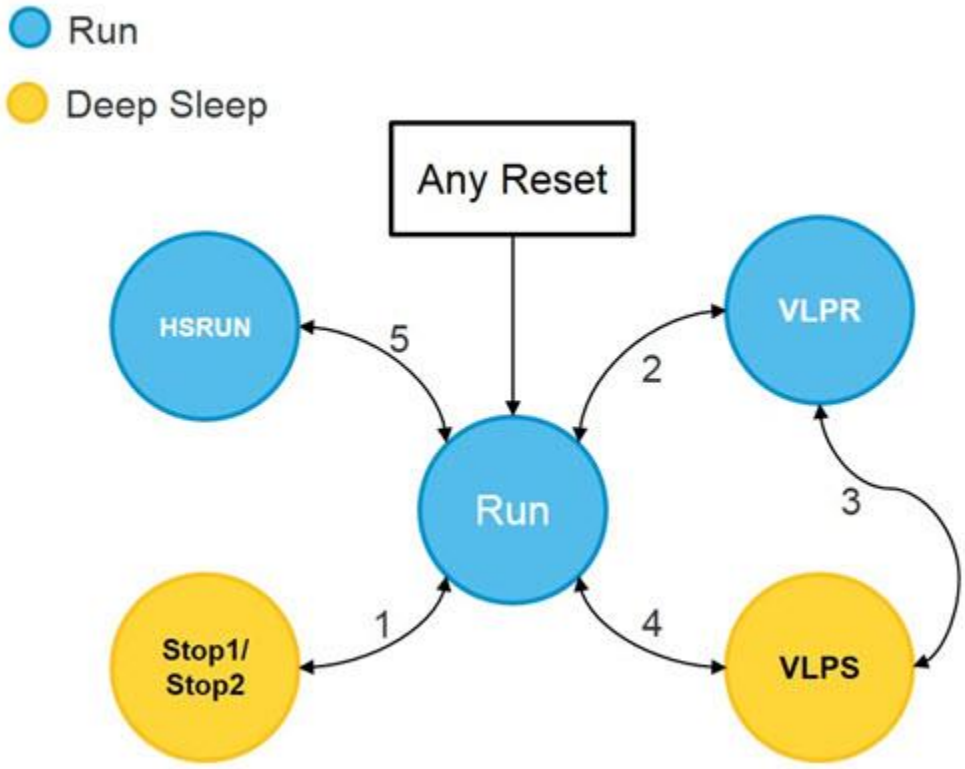

Figure 3. Power mode state transition diagram

Following table defines trigger for the various state transitions shown in figure above.

Table 2. **Power modes transition triggers**

Transition	From	To	Mode Transition Trigger Command/Condition
1	RUN	STOP	SMC_PMCTRL[RUNM] = 00, SMC_PMCTRL[STOPM] = 000[1] following a WFI instruction.[2]
	STOP	RUN	Interrupt[3] or Reset
2	RUN	VLPR	Set SMC_PMPROT[AVLP] = 1, SMC_PMCTRL[RUNM] = 0b10[4]
	VLPR	RUN	Set SMC_PMCTRL[RUNM] = 00 or Reset.
3	VLPR	VLPS	SMC_PMCTRL[STOPM] = 0b010, following a WFI instruction.[2]
	VLPS	VLPR	Interrupt.

Transition	From	To	Mode Transition Trigger Command/Condition
			Note: If VLPS was entered directly from RUN (Transition 4), hardware forces exit back to RUN and does not allow a transition to VLPR.
4	RUN	VLPS	SMC_PMPROT[AVLP] = 1, SMC_PMCTRL[STOPM] = 0b010, following a WFI instruction.[2]
	VLPS	RUN	Interrupt and VLPS mode was entered directly from RUN or reset.
5	RUN	HSRUN	Set SMC_PMPROT[AHSRUN] = 1, SMC_PMCTRL[RUNM] = 0b11.
	HSRUN	RUN	Set SMC_PMCTRL[RUNM] = 00 or Reset.

1. STOPO register must be configured to select the stop mode variant (STOP1 or STOP2).
2. Sleep-now (WFI instruction) or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.
3. Module capable of providing an asynchronous interrupt to the device.
4. Core, System and Bus clock must be 4 MHz (maximum) and flash clock must be set to 1MHz (maximum). Also, all asynchronous clock sources will be restricted to 4 MHz as configured SCG_SRICDIV.

4. Clock Operation in low-power modes

There are several clock sources available in the MCU. To conserve power, most module clocks can be turned off by configuring the CGC field of the peripheral control register in the PCC module. These fields are cleared after any reset, which disables the peripheral clock of the corresponding module. Before initializing a module, the corresponding field in the PCC peripheral control register need to be set to enable the module clock. Be sure to disable the module before turning off the clock.

NOTE

Before disabling a module, its interrupt and DMA request should be disabled.

4.1. System Clock Generator (SCG) clocks

The clocks for the MCU are generated by the System Clock Generator (SCG) module. Figure 4 shows block diagram of SCG module.

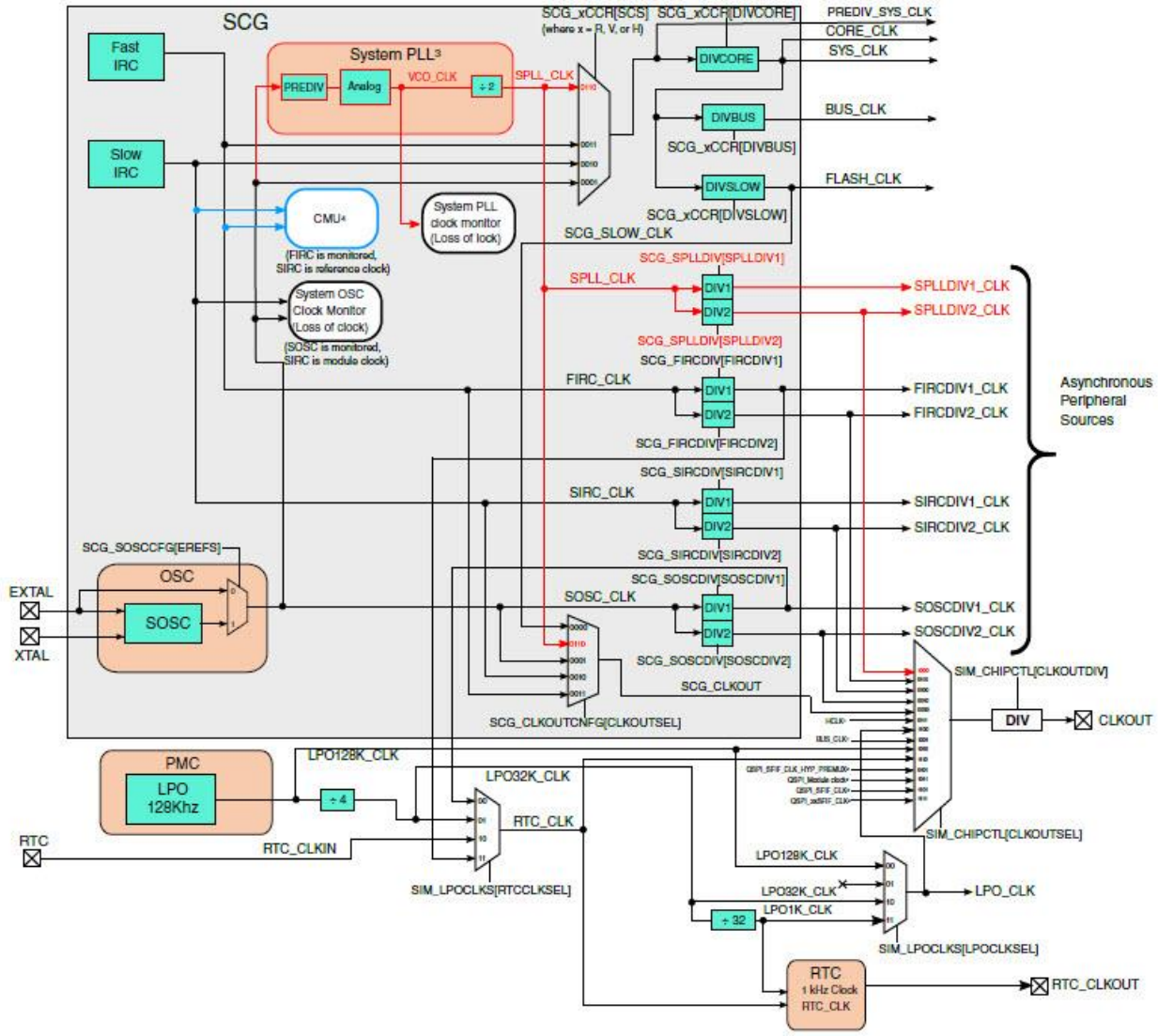

Figure 4. Clocking diagram

Muxes and inputs in red are not available in S32K11x. CMU module (in blue) in only available for S32K11x.

NOTE

System PLL is not available in S32K11x family.

The clock generation circuitry provides several clock dividers and selectors allowing different modules to be clocked at a specific frequency for that module. Four main clock sources (Besides Low Power Oscillator LPO module) can be seen on SCG module: Fast Internal Reference Clock (FIRC), Slow Internal Reference Clock (SIRC) System Oscillator (SOSC) and System PLL (SPLL). For Run modes, (HSRUN, normal RUN, VLPR) different sources can be used to provide clock signal to core. Next diagram shows all possible sources for core clock that can be used in different power modes.

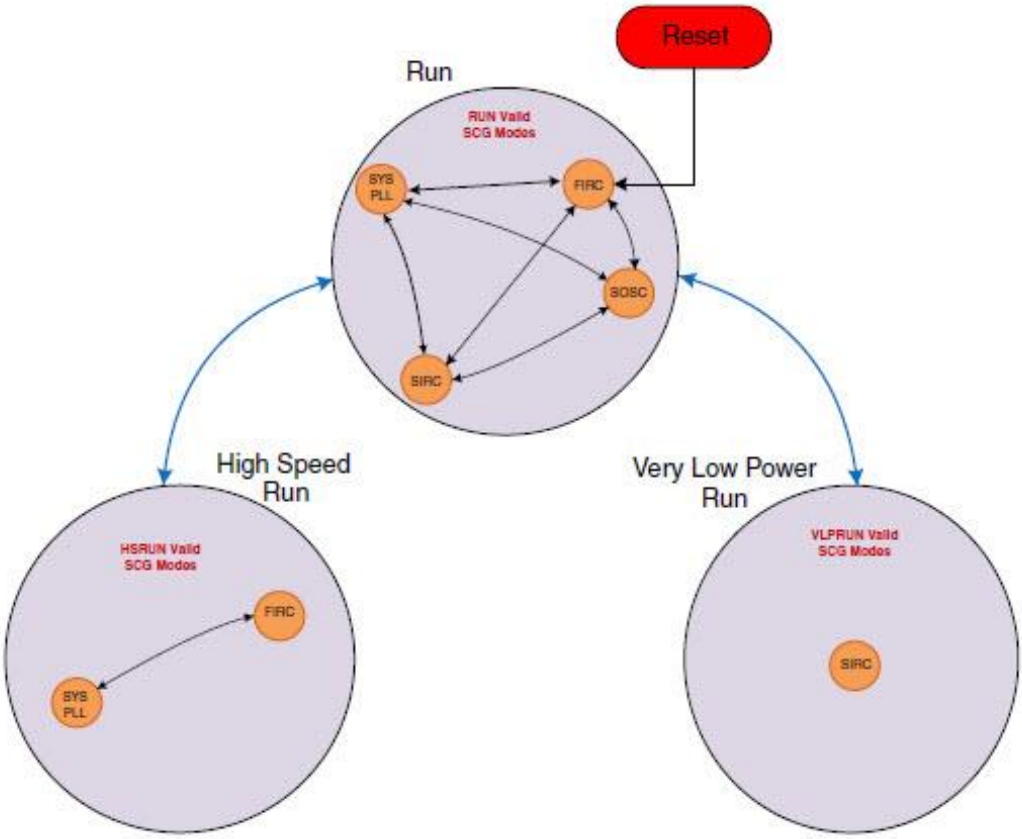

Figure 5. SCG Core clock valid mode transition diagram

For other power modes, such as STOP and VLPS, as core clock is gated off, no source is used.

When entering VLPR/VLPS mode, the System PLL and FIRC must be disabled by software in RUN mode before making any mode transition.

Before switching clock sources, be sure to meet requirements listed in section ***Internal clocking requirements*** and ***Module clocks*** in Reference Manual.

4.2. Power Management Controller (PMC)

An internally-generated low power clock with typical frequency of 128 kHz that can be used as clock source for modules that operate in low power modes can be configured on Power Management Controller (PMC) module. Figure 4 shows the low power oscillator (LPO) source and its different derivations.

5. Power Mode Entry/Exit

When entering, or exiting low-power modes, the system must confirm an orderly sequence to manage transitions safety.

The SMC manages the system's entry into and exit from all power modes.

5.1. HSRUN mode entry

In HSRUN mode, the on-chip voltage regulator remains in a run regulation state, but with a slightly elevated voltage output. In this state, the MCU can operate at a faster frequency compared to normal RUN mode though No Flash commands (FTFC) of any type, including CSEc commands, are available when the chip is in this mode.

While in this mode, the following restrictions must be adhered to:

- Modifications to clock gating control bits are prohibited.
- Flash programming/erasing is not allowed.

To enter HSRUN mode (With 112MHz SPLL as clock source):

1. Disable PLL. Configure FIRC as the RUN mode and HSRUN mode clock source by writing 0b0011 to SCG_RCCR[SCS] and SCG_HCCR[SCS].
2. Set PMPROT[AHSRUN] to allow HSRUN mode.
3. Write 0b11 to SMC_PMCTRL[RUNM] to enter HSRUN. (Now system will enter HSRUN mode with FIRC configured as system clock source).
4. Reconfigure SPLL for 112MHz and enable it.
5. Switch to PLL as the clock source by configuring SCG_HCCR[SCS] as 0b0110.

To enter HSRUN mode (With 80MHz as SPLL clock source):

1. Configure SPLL at 80MHz in RUN mode and use this clock source in HSRUN mode by writing 0b0110 to SCG_RCCR[SCS] and SCG_HCCR[SCS].
2. Configure PMPROT[AHSRUN] to allow HSRUN mode.
3. Write 0b11 to SMC_PMCTRL[RUNM] to enter HSRUN.

Before increasing clock frequencies, the PMSTAT register should be polled to determine when the system has completed entry into HSRUN mode. Next snippet code shows a basic RUN to HSRUN transition function.

NOTE

HSRUN mode is not supported in S32K11x family

```
void RUN_to_HSRUN (void)
{
    /* Disable SPLL and use FIRC as MCU's source clock */
    scg_disable_spll_enable_firc();
    /* Allow high speed run mode */
    SMC->PMPROT |= SMC_PMPROT_AHSRUN_MASK;
    /* Check if current mode is RUN mode */
    if(SMC->PMSTAT == 0x01)
    {
        /* Move to HSRUN Mode*/
        SMC->PMCTRL = SMC_PMCTRL_RUNM(0b11);
        /* Wait for Transition*/
        while(SMC->PMSTAT != 0x80);
        /* Configure SPLL for 112/80 MHz */
        scg_configure_spll();
    }
}
static void scg_disable_spll_enable_firc(void)
{
    /* SCG_RCCR register only accepts 32-bit writes */
    /* Use FIRC as clock for CPU and set valid dividers */
    SCG->RCCR = (uint32_t) (SCG_RCCR_SCS(0b11) |    /* FIRC */
                            /* Core clock is 48MHz / 1 */
                            SCG_RCCR_DIVCORE(0) |
                            /* Bus clock is Core clock / 1 */
                            SCG_RCCR_DIVBUS(0) |
                            /* Flash clock is Core clock / 2 */
                            SCG_RCCR_DIVSLOW(1));
    /* SCG_HCCR register only accepts 32-bit writes */
    /* Use FIRC as clock for CPU and set valid dividers */
    SCG->HCCR = (uint32_t) (SCG_HCCR_SCS(0b11) |    /* FIRC */
                            /* Core clock is 48MHz / 1 */
                            SCG_HCCR_DIVCORE(0) |
                            /* Bus clock is Core clock / 1 */
                            SCG_HCCR_DIVBUS(0) |
                            /* Flash clock is Core clock / 2 */
                            SCG_HCCR_DIVSLOW(1));
    /* Disable PLL and clock monitors */
    SCG->SPLLCSR &= ~(SCG_SPLLCSR_SPLLEN_MASK |
                      SCG_SPLLCSR_SPLLCM_MASK);
}
```

NOTE

Flash programming/erasing is not allowed. No FTFC commands of any type, including CSE commands (for CSEc parts), are available when the chip is in this mode.

NOTE

Modifications to clock gating control bits are prohibited

5.2. HSRUN mode exit

Transition from HSRUN to RUN mode can be made by either a Reset event or setting SMC_PMCTRL to 00. As in HSRUN mode core clock can be set to maximum value (112MHz) and at RUN mode core clock is up to 80MHz, it may be necessary to decrease the core frequency before going back into RUN mode. Next snippet code shows a basic HSRUN to RUN transition function.

```
void HSRUN_to_RUN (void)
{
    /* Adjust SCG settings to meet maximum frequencies value at Run mode */
    scg_configure_freq_for_RUN();
    /* Check if current mode is HSRUN mode */
    if(SMC->PMSTAT == 0x80)
    {
        /* Move to RUN Mode*/
        SMC->PMCTRL = SMC_PMCTRL_RUNM(0b00);
        /* Wait for Transition*/
        while(SMC->PMSTAT != 0x01);
    }
}
```

As HSRUN mode allows MCU to run at maximum clock speed, be sure to adjust frequencies in SCG module to meet clock requirements for RUN mode. These requirements can be consulted on section Internal clocking requirements from Reference Manual.

NOTE

HSRUN mode is not supported in S32K11x family

5.3. VLPR mode entry

In VLPR mode, the on-chip voltage regulator is put into a stop mode regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the PCC's registers.

Before entering this mode, the following conditions must be met:

- All clock monitors in the SCG must be disabled.
- Adjust the clock frequencies according their maximum allowed values: Core, System and Bus clock must be 4 MHz (maximum) and flash clock must be set to 1MHz (maximum). Also, all asynchronous clock sources will be restricted to 4 MHz. System PLL, System Oscillator and FIRC must be disabled by software prior mode entry.
- Mode protection must be set to allow VLP modes, that is, SMC_PMPROT[AVLP] to 1.
- SMC_PMCTRL[RUNM] must be set to 0b10 to enter VLPR.

Next snippet code shows a basic RUN to VLPR transition function.

```
void RUN_to_VLPR (void)
{
    /* Disable clock monitors on SCG module */
    disable_clock_monitors();
    /* Adjust SCG settings to meet maximum frequencies value
       Disable SPLL, System Oscillator and FIRC */
    scg_configure_freq_for_VLPR();
    /* Allow very low power run mode */
    SMC->PMPROT |= SMC_PMPROT_AVLP_MASK;
    /* Check if current mode is RUN mode */
    if(SMC->PMSTAT == 0x01)
    {
        /* Reduce MCU power consumption in Very Low Power modes*/
        PMC->REGSC |= PMC_REGSC_BIASEN_MASK;
        /* Move to VLPR Mode*/
        SMC->PMCTRL = SMC_PMCTRL_RUNM(0b10);
        /* Wait for Transition*/
        while(SMC->PMSTAT != 0x04);
    }
}
```

NOTE

Flash programming/erasing is not allowed. No FTFC commands of any type, including CSE commands (for CSEc parts), are available when the chip is in this mode.

NOTE

Do not increase the clock frequency while in VLPR mode, because the regulator is slow in responding and cannot manage fast load transitions. In addition, do not modify the clock source in the SCG module or any clock divider registers. Module clock enables in the PCC can be set, but not cleared.

NOTE

To reduce MCU power consumption in low power modes, PMC_REGSC[BIASEN] bit should be set. This bit enables source and well biasing for the core logic in low power modes. This bit must be set to 1 when using Very Low Power (VLP) modes.

5.4. VLPR mode exit

To reenter normal RUN mode, clear SMC_PMCTRL[RUNM] bits. If a higher execution frequency is desired, poll PMSTAT until it is set to RUN and then configure SCG module as desired. Also, a reset event causes the MCU to come back to RUN mode.

Next snippet code shows a basic VLPR to RUN transition function.

```
void VLPR_to_RUN (void)
{
    /* Check if current mode is VLPR mode */
    if(SMC->PMSTAT == 0x04)
    {
        /* Move to RUN Mode*/
        SMC->PMCTRL = SMC_PMCTRL_RUNM(0b00);
        /* Wait for Transition*/
        while(SMC->PMSTAT != 0x01);
    }
}
```

5.5. STOP and VLPS mode entry sequence

When entering stop/VLPS mode, clocks are shut off in an orderly sequence to safely place the chip in the targeted low-power state. All low-power entry sequences are initiated by the core executing a WFI instruction. When entering low power modes, the chip performs the sequence shown below:

1. The CPU clock is gated off immediately.
2. Requests are made to all non-CPU bus masters (DMA and ENET if available) to enter Stop mode.
3. After all masters have acknowledged they are ready to enter Stop mode, requests are made to all bus slaves to enter Stop mode.
4. After all slaves have acknowledged they are ready to enter stop mode, system and bus clocks are gated off depending on the target mode–VLPS/STOP1/STOP2. Note that, in STOP2 mode bus clocks will not be gated.
5. Additionally, for VLPS mode, clock generators are disabled in the SCG unless configured to be enabled.
6. The on-chip regulator in the PMC and internal power switches are configured to meet the power consumption goals for the targeted low-power mode. This step is valid only for VLPS and not for STOP as in STOP mode the PMC is in Run regulation.

NOTE

If SIRC is not enabled in VLPS modes, power consumption can be reduced by setting PMC_REGSC[CLKBIASDIS] bit. While using this bit, it must be ensured that respective clock modules are disabled in VLPS mode, else, severe malfunction of clock modules will happen.

5.5.1. Stop1/2 mode entry

STOP1/2 mode is entered via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System control register in ARM Core.

The SCG module can be configured to leave the reference clocks running. For STOP modes, all SCG clock are available (LPO, PLL, SIRC, FIRC and OSC). For more details, please check ***Modules in power modes***.

SMC_STOPCTRL[STOPO] bits selects whether MCU is sent to STOP1 (0b01) or STOP2 (0b10) mode.

Next snippet code shows a basic RUN to STOP1/STOP2 transition function.

```
void RUN_to_STOP (void)
{
    /* Enable SLEEPDEEP bit in the Core
     * (Allow deep sleep modes) */
    S32_SCB ->SCR|=FSL_SCB_SCR_SLEEPDEEP_MASK;
    /* Select Stop Mode */
    SMC->PMCTRL=SMC_PMCTRL_STOPM(0b00);
    /* Select which STOP mode (Stop1 or Stop2)
     * is desired (Stop1 - 0b01, Stop2 - 0b10) */
    SMC->STOPCTRL=SMC_STOPCTRL_STOPO(0b01);
    /* Check if current mode is RUN mode */
    if(SMC->PMSTAT == 0x01)
    {
        /* Go to deep sleep mode */
        asm("WFI");
    }
}
```

5.5.2. VLPS mode entry

There are two ways in which VLPS mode can be entered are listed below:

- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register while the MCU is in VLPR mode and PMCTRL[STOPM] = 0b010 or 0b000.
- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the Arm core while the MCU is in normal RUN mode and PMCTRL[STOPM] = 0b010. When VLPS is entered directly from RUN mode, exit to VLPR is disabled by hardware and the system will always exit back to RUN.

Transition to VLPS is almost the same as for Stop1/2 modes except allowing very low power modes in SMC_PMPROT register. Be sure to disable System PLL, System Oscillator and FIRC[1] before VLPS mode entry. Next snippet code shows a basic RUN to VLPS transition function.

[1] You can refer to section System clock switching in Reference Manual for more details.

```
void RUN_to_VLPS (void)
{
    /* Adjust SCG settings to meet maximum
frequencies value */
    scg_disable_pll_and_firc();
    /* Enable SLEEPDEEP bit in the Core
     * (Allow deep sleep modes) */
    S32_SCB ->SCR|=FSL_SCB_SCR_SLEEPDEEP_MASK;
    /* Allow very low power run mode */
    SMC->PMPROT |= SMC_PMPROT_AVLP_MASK;
    /* Select VLPS Mode */
    SMC->PMCTRL=SMC_PMCTRL_STOPM(0b10);
    PMC->REGSC |= PMC_REGSC_BIASEN_MASK;
    /* Check if current mode is RUN mode */
    if(SMC->PMSTAT == 0x01)
    {
        /* Go to deep sleep mode */
        asm("WFI");
    }
}
```

NOTE

To reduce MCU power consumption in low power modes, PMC_REGSC[BIASEN] bit should be set. This bit enables source and well biasing for the core logic in low power modes. This bit must be set to 1 when using Very Low Power (VLP) modes.

5.6. STOP and VLPS mode exit sequence

Exit from a low-power stop mode is initiated either by a reset or an interrupt event. The following sequence then executes to restore the system to a run mode (RUN or VLPR):

1. The on-chip regulator in the PMC, clock generators and internal power switches are restored. This step is valid only for VLPS and not for STOP as in STOP mode the PMC is in Run regulation.
2. System and bus clocks are enabled to all masters and slaves.
3. The CPU clock is enabled and the CPU begins servicing the reset or interrupt that initiated the exit from the low-power stop mode.

6. Modules in power modes

Section **Module operation in available low power modes** in Reference Manual lists all peripherals and their availability in different low power modes.

7. Hardware and software considerations

To reduce MCU power consumption, there are some hints that you can follow:

7.1. Hardware considerations

All unused pins in application (especially analog functionality) should follow some recommendations to eliminate possible current consumption increase:

- DAC output pin should be floating.
- ADC pins should be grounded.

7.2. Software considerations

To conserve power, most module clocks can be turned off by configuring the CGC field of the peripheral control register in the PCC module. These fields are cleared after any reset, which disables the peripheral clock of the corresponding module. Be sure to disable the module before turning off the clock.

Several peripherals support Peripheral Doze mode. In this mode, a register field can be used to disable the peripheral for the duration of a low-power mode.

7.3. Tips for making low-power measurements on the bench

7.3.1. External.

The suggestions below address the most common issues encountered when trying to duplicate the data sheet current specs.

- ***When using a digital multimeter (DMM), use "Manual Range Mode."*** Using a DMM with an auto-ranging function enabled may cause LVD and POR resets. This is most common when you are exiting from one of the low-power modes like LLS or VLPS back to Run. The DMM has changed the range to a micro-amp or nano-amp range while the MCU is in the low-power mode and the sudden inrush of current requires the DMM to change range. The range change does not happen fast enough and the MCU starves and pulls the VDD level below the LVD or POR limits.
- ***Disconnect the debugger and power cycle the MCU.*** With the JTAG debugger is attached, the MCU may have the debugger module in the MCU active, clocking and consuming power. The external debugger hardware may also load the I/O of the JTAG port when attached. Thus, your low-power measurements will be higher than expected.
- ***Isolate the MCU VDDs***. If you want to measure the current draw of the MCU, then remove the other IC and component networks that are sourced by the voltage supply sourcing the MCU. For example, some EVBs have a potentiometer connected between MCU_VDD and ground. A 5 K potentiometer across a 3.6 V supply pulls 720 µA. This is huge when considering that the MCU consumes around dozens µA in lowest power modes.
- ***Match impedance of inputs***. If the impedance of high speed signals (fast edge transitions) are not well matched, then the signals can "ring" and exceed the VDD supply of the device. This can result in the signal providing current to the device through the input protection diodes. This is

particularly true for high speed input clocks. This issue can result in negative IDD measurements while in the lowest power modes.

- ***Match voltage levels***. Although the MCU input pins are 5 V tolerant on some parts, when the MCU goes into the low-power modes, measurement of the current through MCU_VDD will be affected by any input higher than MCU_VDD. The higher input pin will back power the MCU through the input pin, resulting in negative IDD reading in low-power modes.
- ***Reduce pin loading of the MCU***. When the MCU sources current through the output pins, the power is being sourced through MCU_VDD. This is most evident when you output high frequency signals to an output pin as you might with the external memory interfaces such as clock and address/data pins.

7.3.2. Internal.

Below is a list of the most common issues that can prevent you from getting to the lowest data sheet current specs.

- ***Watchdog is not disabled***, causing resets. Disable or service the watchdog.
- ***The clock monitor is not disabled which may cause resets***. Disable all clock monitors.
- ***A crystal oscillator is enabled in low power mode***. The RTC oscillator, typically consumes <500 nA of current.
- ***The CLKOUT signal is being output to a pin***. Any pin that is constantly changing state will draw power.
- ***The requested low-power mode is not allowed with a corresponding bit in the PMPROT register***. For example, if AVLP is not set in the PMPROT and the WFI instruction is executed, you won't enter Stop mode.
- ***The clock gate for a module is not enabled before it is read or written***. This causes a reset before the MCU tries to enter the low-power mode.
- ***The clock gate for a module that must acknowledge the mode controller mode entry request is turned off prior to low-power mode entry***. This will result in a Stop Mode Acknowledge reset.
- ***Failure to un-comment out the call to the stop or sleep function after debugging is complete*** will keep you in the higher run current mode.
- ***The frequency of wake-up events is too high***, which means that the MCU spends more time in Run or VLPR mode than in a low-power mode. The transition time from low-power mode to Run mode is quick. If the MCU only spends 9 ms in run and 1 ms in a low-power mode, the average current of the system will be considerably higher than if the MCU was running only 1 ms every 1 second.
- ***The MCU is running at a much higher frequency than is needed to accomplish the work***. Throttle the clock with the SCG dividers or reduce the clock. Obviously, the higher the MCU frequency the higher the IDD of the MCU. Reduce the clock and lower the Run or Wait current. However, there is some trade-off. If the current of Run mode can be tolerated, then getting work

done as quickly as possible and going right back to low power mode is more advantageous than running a slower clock in Run mode.

- ***An input pin is floating without an internal or external pull device***. This can result in 50-80 uA of current per pin. This include the JTAG or SWD pins. Disable the JTAG pins on PORTA or properly terminate the inputs.

7.4. Current Consumption measurements

As current consumption depends on different factors such as which modules are enabled, what frequency is feeding Core and other peripherals, temperature conditions, among other, MCU's datasheet contains test cases for different power modes, so, for current consumption measurements, you can refer to MCU's datasheet.

8. Power modes use cases.

Next section shows some use-case examples along with their example codes built on S32DS version 2018.R1.

8.1. Power mode switching to achieve 100uA on system (VLPS + RUN modes)

One of the main concerns when designing system's architecture is to achieve the lowest power consumption, to achieve this, it is not only important to define the lowest power consumption mode but also the complementary mode where the main operations will be done, for instance, ADC readings, Communication transfers, I/O control signals, etc. Once the power modes have been defined, some other important factors such as frequency of operation, period between wakeups, switching timing between modes etc. are important.

Following use case implements a power mode transition between VLPS (lowest power consumption mode) and RUN mode to use features such as ADC readings and SPI transfers to achieve a system power consumption around 100uA.

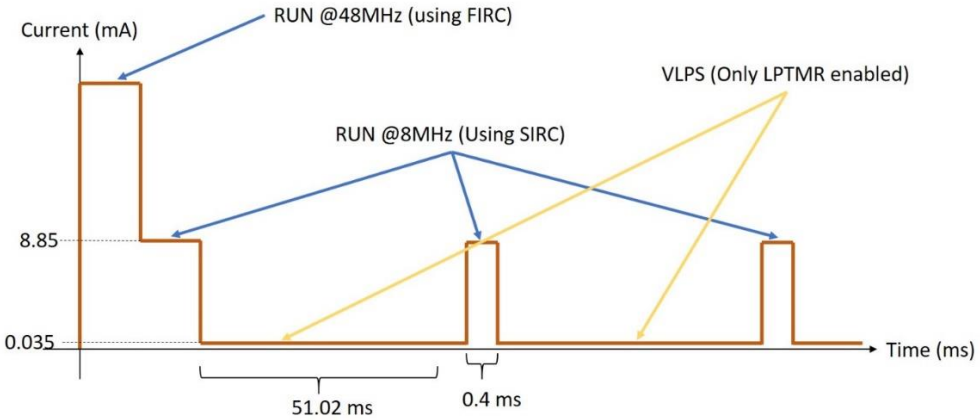

Figure 6. Power mode transitions diagram

The use case is implemented in S32K148EVB (S32K148_PowerManagement_VLPS_RUN) and consists of next operations.

- In RUN mode:
  - WDOG is disabled as well as all other clocks for unused modules (for instance: MPU clock, DMA, ERM, etc.)
  - Disable clock monitors for both SOSC and SPLL.
  - Change clock configuration. By default, FIRC is used and device is running at 48MHz so a change is needed to use SIRC as clock source (8MHz) and then, disable FIRC by software prior doing any power mode transition. Also, SIRC_DIV2 signal is configured as this will be used for SPI and ADC in RUN mode.
  - LPTMR is configured. This timer will be used to wake the system up from VLPS every 51.42 ms. LPTMR is using LPO clock as it remains active in VLPS.
  - SPI and ADC modules are initialized, there is no need to disable these modules when transitioning to VLPS as the clock source used won't be available in VLPS (SIRC is not enable in VLPS). Be aware to fulfill requirements for SIRC_DIV2 shown in ***Clock definitions*** section in device's reference manual as this signal is used to calculate operating frequency for these SPI and ADC modules.
  - To reduce power consumption, debugger pins are also disabled.
  - Before switching to VLPS, SPI transfer and ADC channels are read.
- In VLPS
  - As nothing except LPTMR is enabled in this mode, no operations are made in VLPS unless waiting for timeout period. Power consumption during this time reaches ~35uA.

Average current for whole system is calculated as follows:

$$I_{system} = \frac{I_{RUN} \cdot t_{RUN} + I_{VLPS} \cdot t_{VLPS}}{t_{total}}$$

Based on values from above figure:

$$I_{system} = \frac{8\,850\ \mu A \cdot 0.4ms + 35\mu A \cdot 51.02\ ms}{51.42\ ms} = 103.57uA$$

Following figure shows the measurement obtained from digital MM.

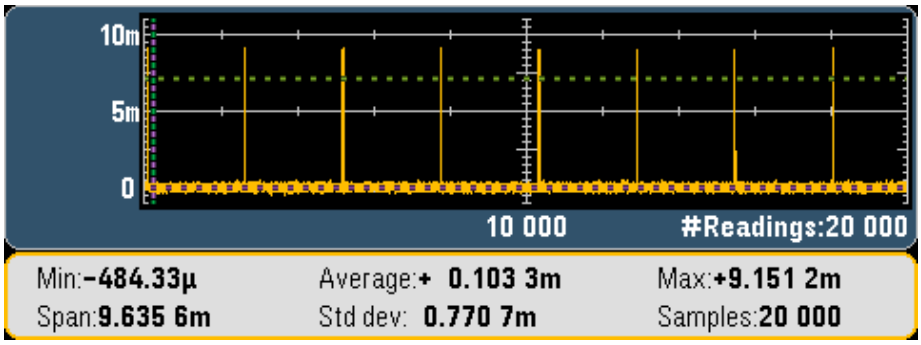

Figure 7. Power consumption when core is running at 8MHz (103.3 uA as average)

Beside this, user can vary operation frequency, module's settings, etc. to identify how the system current is impacted.

For this use case, we made a second test to set the core frequency to 4MHz (expecting the maximum current could be reduced although time spent in RUN would be longer) to correlate if system current can be reduced. Following image shows the results we had.

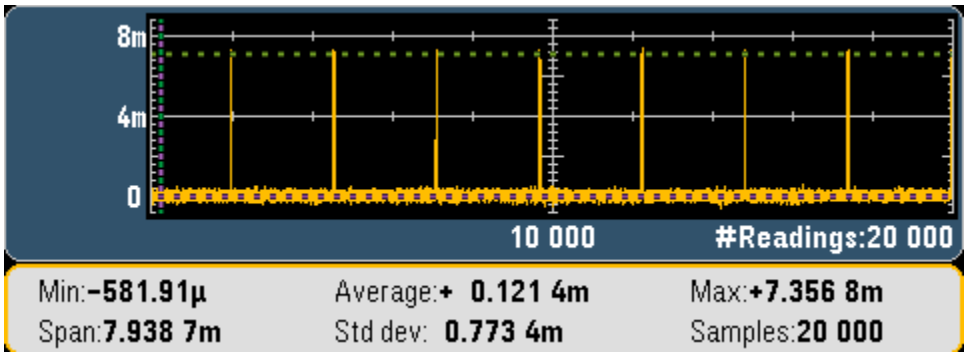

Figure 8. Power consumption when core is running at 4MHz (121 uA as average)

As seen in the figure above, average current for the system is increased using a lower frequency for the core, so the relationship between maximum current in RUN vs time spent in RUN plays a role to define the total current consumption for the system.

$$I_{system} = \frac{7.12\,mA \cdot 0.63\,ms + 35\mu A \cdot 50.79\,ms}{51.42\,ms} = 121.8\,uA$$

In summary, user needs to consider not only the lowest power mode when designing the whole system but also the operating frequency for the "awaken" mode, the configuration for used modules, the periodicity for the wake-up signal, etc. to achieve the desired current value.

8.2. Clock considerations when switching between different modes

There might be some cases when switching between power modes with different clocks configurations are needed, i.e. switching between HSRUN to use the maximum speed on the device and RUN mode to use some features not available in HSRUN. Another option for instance is, when low power is desired, switching between RUN and VLPR can also be used considering that not all the peripherals are enabled in VLPR so switching to RUN mode is needed.

As there are different multiplexes for Core, Bus and Flash dividers for each mode (SCG_RCCR for Run, SCG_HCCR for HSRUN and SCG_VCCR for VLPR) it should be quite easily to switch between different power modes and clocks sources, however, some clock considerations are needed to meet the requirements for each mode. 2 examples are shown below.

8.2.1. Switching between HSRUN and RUN (S32K14x devices only) to use CSEc/EEEPROM features when device is running at maximum speed.

As some features such as Programming/Erasing Flash and CSEc/EEEPROM operations are not available in HSRUN, user must switch between HSRUN and RUN mode to run device at maximum speed in

HSRUN and then execute some operations (CSEc/EEEPROM) in RUN when these features are required.

As the switching process between 112MHz in HSRUN mode using SPLL as the main clock source to 48MHz in RUN mode using FIRC as the main clock source can be easily done using the different SCG_xCCR registers settings shown in table below.

Register	System Clock Source (SCS)	Core Clock Divider (DIVCORE)	Bus Clock Divider (DIVBUS)	Flash Clock Divider (DIVSLOW)
SCG_HCCR	6 (SPLL)	0 (Divide by 1)	1 (Divide by 2)	3 (Divide by 4)
SCG_RCCR	3 (FIRC)	0 (Divide by 1)	0 (Divide by 1)	1 (Divide by 2)

For HSRUN mode, Core clock will be 112MHz, Bus clock 56MHz and Flash clock 28MHz.

For RUN mode, Core clock will be 48MHz, Bus clock 48MHz and Flash clock 24MHz.

Using this configuration, user can switch between HSRUN and RUN easily if Asynchronous peripheral clock sources are not used (SPLL_DIV1 and SPLL_DIV2) otherwise, some considerations must be taken.

According to ***Clock definitions*** section in Reference Manual, SPLL_DIV1 can be configured to 80MHz or less in RUN mode and to 112MHz or less in HSRUN. Also, SPLL_DIV2 can be configured to 40 MHz or less in RUN mode and to 56 MHz or less in HSRUN, besides this, whenever a peripheral clock source or its divider is changed, the corresponding module should be disabled. To avoid this, it is recommended to set a fix frequency in both signals that meets requirements for both Run and HSRUN to avoid the need of disabling the peripheral when the power mode switching takes place (For instance, a PWM signal driven by FTM which is using SPLL_DIV1 might not need to be paused while the power mode transition is happening). Following image shows the maximum frequencies for each source to facilitate the power transition between HSRUN and RUN without the need to change the dividers/clock source before the power mode transition.

Although SPLLDV1_CLK can be configured to 56 MHz (it does not violate the maximum allowed frequency), there is a restriction in FTM that requires FTM's functional clock (SPLLDVI1_CLK) not to exceed ¼ the functional clock (Clock feeding FTM logic, which is driven from SYS_CLK). So SPLLDIV1 is fixed to 14MHz (56 MHz / 4 = 14 MHz).

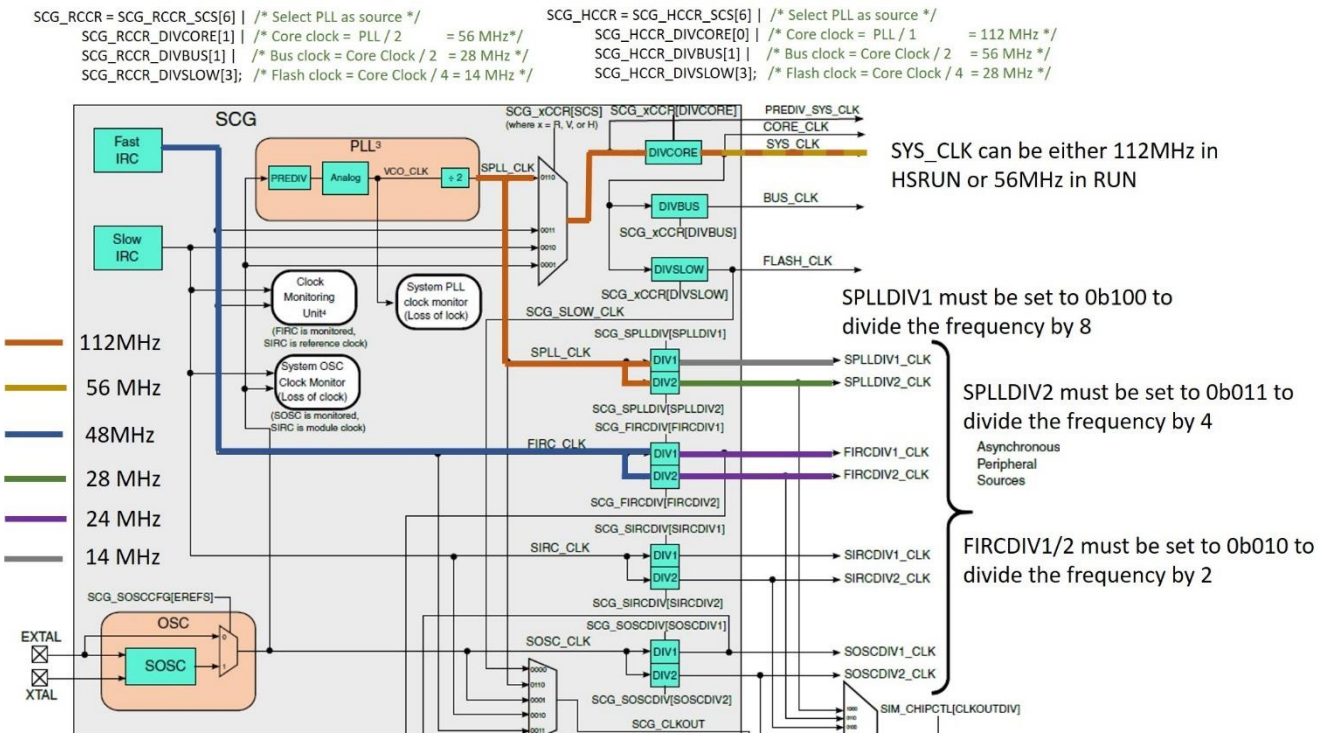

Figure 9. Clocking diagram for HSRUN / RUN switching scheme

For diagram above, all peripherals bonded to SPLL_DIV1 and SPLL_DIV2 do not need to be disabled/suspend for the switching process as the same value remains in these signals on every transition, on the other hand, peripherals linked to SYS_CLK and BUS_CLK might need to be disabled before the switching process and enable the peripherals again after the transition to desired power mode is completed.

NOTE

Although SCG_xCCR[DIVBUS] and SCG_xCCR[DIVSLOW] remain unchanged, DIVCORE is changed impacting the frequency value for Bus and Flash clocks, so peripherals using these clock sources need to be disabled prior the switching event.

Table 27-9 in reference manual lists all the peripherals and their available clock sources. Be sure to disable all those peripherals tied to SYS_CLK / BUS_CLK before any transition (Specifically all timer/communications peripherals which are more time-depending).

NOTE

Although FIRC_DIV1 and FIRC_DIV2 signals could be configured to 48MHz, most peripherals able to use these signals limit the allowed frequency for the peripheral as this governed by BUS_CLK / SYS_CLK. It is depicted in table ***Peripheral clock summary*** from reference manual.

Once all these considerations are taken, device can run on HSRUN for high speed performance and whenever is needed, switching to RUN to execute any CSEc/EEEPROM functionality. After these are completed, device can go back to HSRUN.

Example project was tested on S32K142EVB and it implements:

1. Clock configuration for SPLL (SPLL_CLK is 112MHz, SPLL_DIV1 is 14 MHz and SPLL_DIV2 is 28MHz), FIRC (FIRC_DIV1 and FIRC_DIV2 are 24MHz) and dividers for RUN and HSRUN are configured.
2. Pins are configured for GPIO signals used as reference and CLKOUT pin is configured as well to show SYS_CLK signal divided by 8 in PTB5.
3. Debug console is used to send message to terminal through LPUART1 module.
4. CSEc module is configured. RAM key is loaded for ECB operation.
5. FTM is configured to generate 4 PWM signals at 10 kHz, 50% duty cycle.
6. LPTMR is used to generate a periodic interrupt signal every 500ms. This interrupt will be used to switch to RUN mode to execute a CSEc operation.
7. PDB0 is configured to provide an interrupt every 1 ms when device is using 112MHz clock (HSRUN) and 2 ms when device is using 56MHz clock (RUN mode). Configuration for PDB is fixed (modulus value is set to 360 for both cases).
8. Transition to HSRUN mode is performed. MCU will remain in HSRUN until an interrupt from LPTMR is gotten.
9. On interrupt's arrival, transition to RUN mode is performed. Prior transition, modules using SYS_CLK and BUS_CLK are disabled and after transition is made, these modules are enabled again (FTM is not modified as it is using SPLL_DIV1 reference).
10. Once in RUN mode, two ECB operations are done. Once they are completed, transition to HSRUN is performed. Again, module using SYS_CLK and BUS_CLK are disabled before the transition is done and they are enabled again once the transition is completed (FTM is not modified as it is using SPLL_DIV1 reference).

Next image shows timing diagram for this use case.

- Channel 1 (Yellow signal) shows FTM0_CH0 output that generates a PWM signal of 10 kHz. PWM output remains unchanged even in the power mode transition.
- Channel 2 (Green signal) shows the power mode transition.
- Channel 3 (Blue signal) is connected to CLKOUT signal on PTB5. CLKOUT is mapped to HCLK (SYS_CLK) using a divider of 8, so output is 112MHz / 8 = 14MHz in HSRUN and 56MHz / 8 = 7 MHz in RUN mode.
- Channel 4 (Pink signal) is connected to PDB0 callback which is toggling a pin. We can see how signal changes its period on every power mode transition.

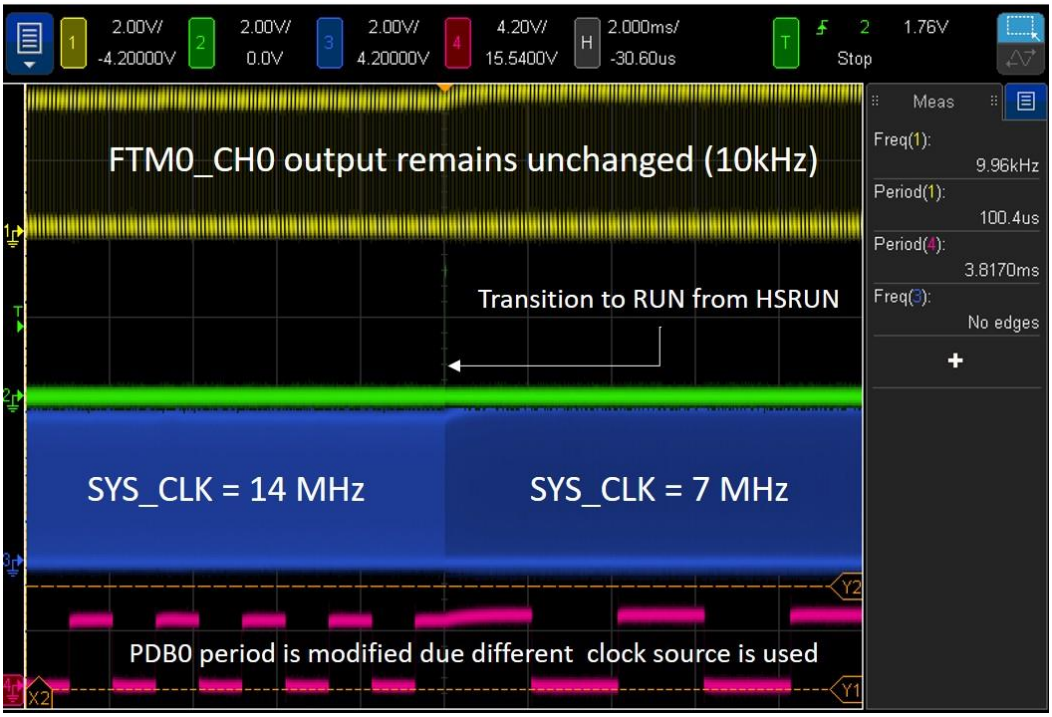

Figure 10. Clock signals on HSRUN to RUN transition

Once CSEc operations are completed, device will run again at maximum speed. Following figure shows clock signal after, during and before the transitions to RUN mode.

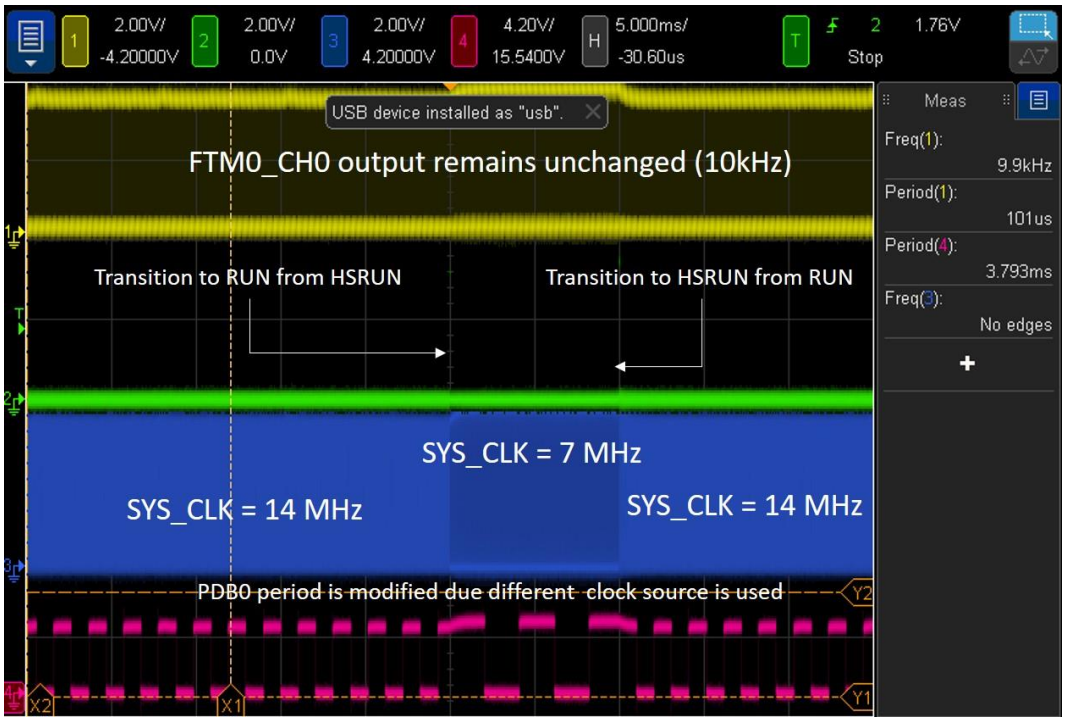

Figure 11. Clock signals on power modes transitions

8.3. Manage a transfer when MCU is sleeping: VLPS + DMA + LPUART (LIN) for S32K11x.

The most common approach for communication transfers when MCU is sleeping is to wake the system up for sending/receiving data, and then go back to sleep mode. Some features included in S32K devices allow the system to be able to send/receive data when device remains in lower power mode with MCU's clock gated off (VLPS). To do this DMA is required as it is fully functional in VLPS. Take in mind that any wake-up event (for instance, DMA's completion) during a power mode entry might result in possible system hang (Errata e11063).

Next use case shows the usage of LPUART + DMA to trigger a transfer when device remains in VLPS and only after all data have been transmitted, MCU can be awaken.

For this implementation, DMA is configured to move data from SRAM to LPUART_DATA register. The request is enabled in DMA_ERQ and DMA_EARS as the asynchronous operations is required. Once current TCD is completed, hardware request is disabled (DMA_TCD_CSR[DREQ] = 1) to stop sending more data to LPUART and start processing the DMA interrupt which wakes the system up.

Please be aware that once DMA request is enabled, transition to VLPS is triggered, this might set SMC_PMCTRL[VLPSA] as stated in Reference Manual though transition is done successfully.

Same approach can be used for reception or any other communication module available in VLPS such as I2C, SPI, etc.

Example code was created and tested in S32K116EVB. Following figure shows the data seen in LPUART_TX pin while MCU remains in VLPS/VLPR.

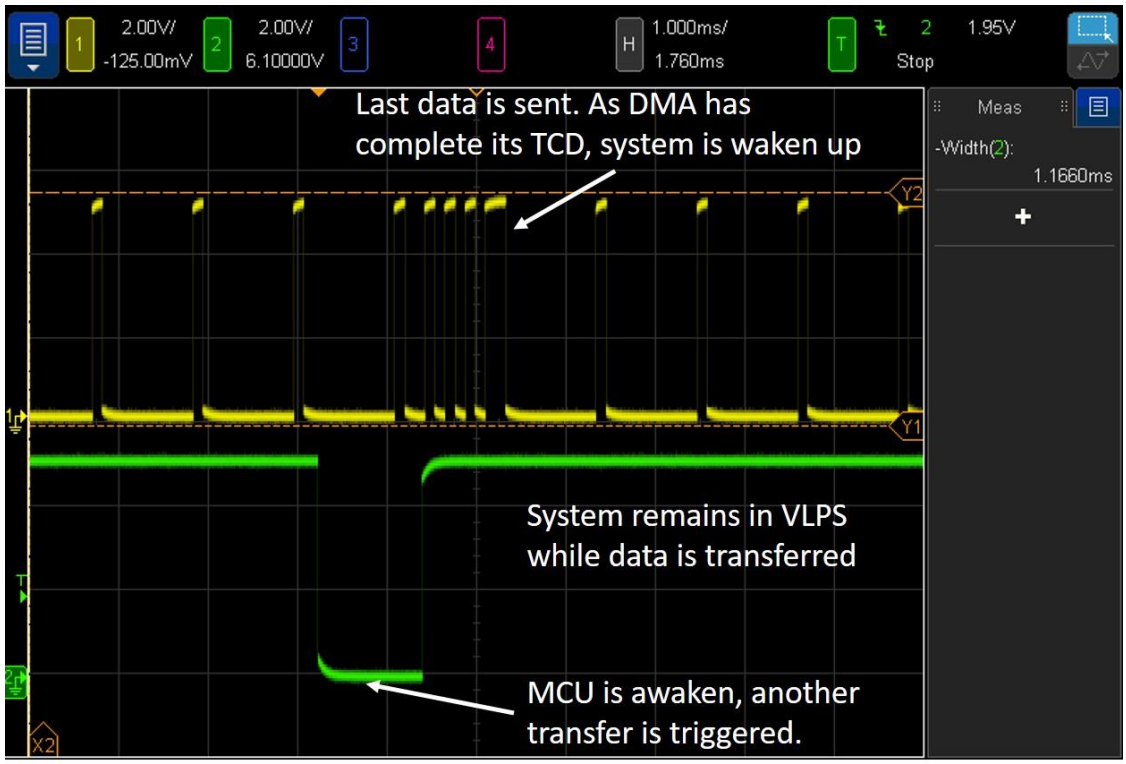

Figure 12. LPUART data while MCU remains in VLPS

Data can be transferred through LPUART interface as DMA can be active in VLPS. Once it is completed, MCU is awaken before last data is physically seen on TX pin. In this use case, LPUART is loaded with same data to be transferred and process is repeated endlessly. Following figure shows the power consumption for this use case. One important difference here is that, if more data is sent while MCU remains in VLPS, the current consumption is reduced.

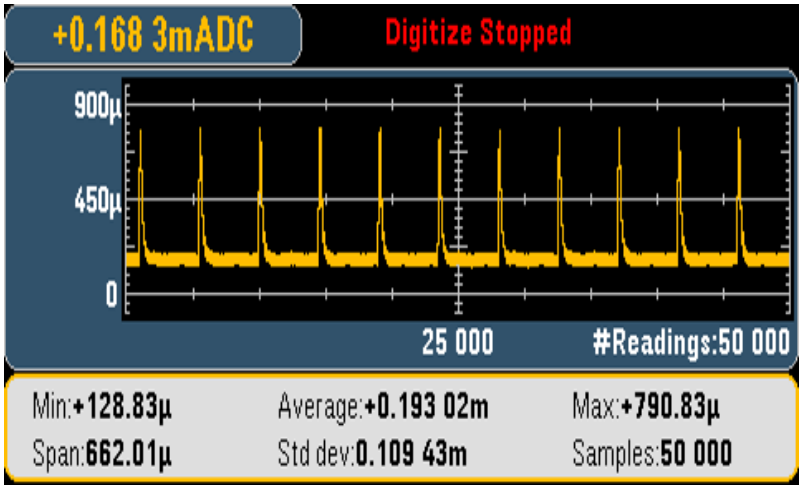

Figure 13. LPUART data while MCU remains in VLPS

9. Revision History

Version Number	Revision Date	Description of changes
REV 0	March 2017	Initial version
REV 1	Apr 2018	• Included support for S32K11x. • Updated the figures 4 and 5 to include S32K11x reference. • Added additional information for HSRUN and VLPS entry. • Removed tables in section Modules in power modes • Added Power modes use cases for use cases examples.

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number:AN5425
Rev. 1
05/2018