NXP Semiconductors
Application Notes

Document Number:ANxxxx
Rev. 0, 06/2022

Power Management for S32K3XX

by: NXP Semiconductors

1. Introduction

The power consumption of devices and the implications around designing for low power are common topics currently. The S32K3xx 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.

The application note discusses how to use the power management system, provides use case examples and shows current measurement results for these cases. Tips are given for using each of the power modes available on S32K3xx family.

Contents

2. Overview of power modes

The typical power modes in other embedded systems are Run, Wait and Stop. The ARM® Cortex™ M7 power modes are Run, Sleep and Deep Sleep. The system level power modes and MCU level power modes and their relationship with typical and ARM Cortex M7s modes are depicted in below figure.

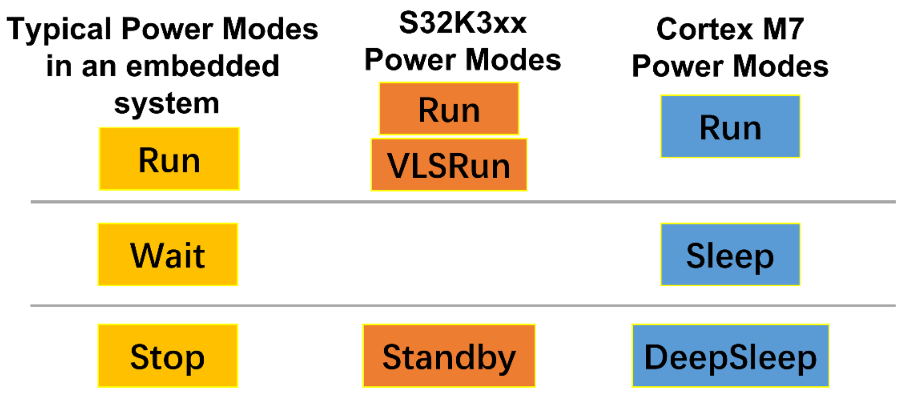


Figure 1. Power modes comparison

2.1. ARM Cortex-M7 power modes implementation

The ARM Cortex-M7 core has three primary modes of operation: Run, Sleep, and Deep Sleep. The Wait For Interrupt (WFI) instruction invokes sleep and deep sleep modes for the S32K3xx chips.

Figure 2 shows the Cortex -M7 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 S32K3xx 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-M7 Devices Generic User Guide.

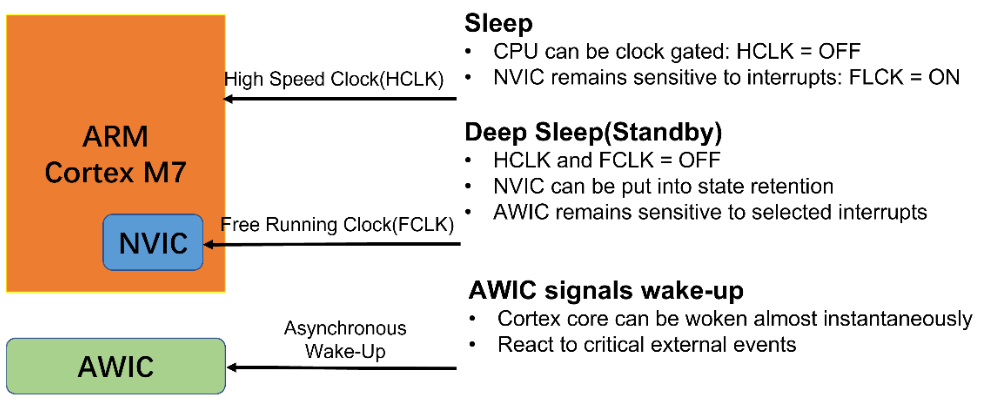

ARM Cortex M7
NVIC
AWIC
High Speed Clock(HCLK)
Free Running Clock(FCLK)
Asynchronous Wake-Up
Sleep
CPU can be clock gated: HCLK = OFF
NVIC remains sensitive to interrupts: FLCK = ON
Deep Sleep(Standby)
HCLK and FCLK = OFF
NVIC can be put into state retention
AWIC remains sensitive to selected interrupts
AWIC signals wake-up
Cortex core can be woken almost instantaneously
React to critical external events

Figure 2. ARM Cortex M7 architecture

3. Power Modes description

S32K3xx 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 Standby mode refer to chapter in the Modules in Standby modes in the Application note or S32K3xx' reference manual. The following table compares the available power modes.

Table 1. S32K3xx power modes

S32K3xx mode	**Description**	**Core mode**	**Normal recovery method**
Run(FPM)	Default mode out of reset. Main operation mode having full-chip performance and a higher current consumption.	Run	-
Very Low Speed Run(VLSR)	On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. • FIRC provides a low power 3 MHz source for the core, the bus and the peripheral clocks. • Reduced-frequency core and flash memory access mode (750 KHz)	Run	-
Standby(LPM)(via WFI instruction)	Low-performance mode of the chip in which the Run domain is turned off. Most of the cores and peripherals turn off in this mode.	Deep Sleep	Interrupt (or Reset)

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. Standby domain registers content are held in all modes of operation.

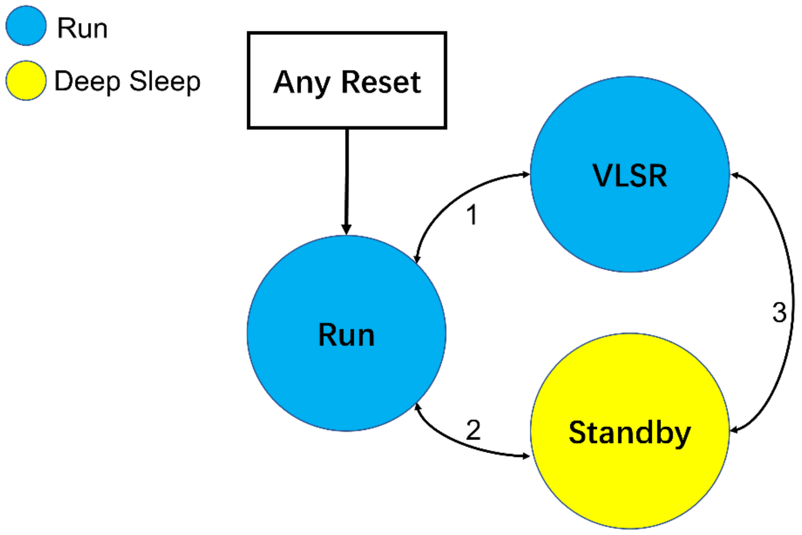


Figure 3. Power mode state transition diagram

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	VLSR	Clock switch from PLL to FIRC (3MHz)[1]
	VLSR	Run	Clock switch from FIRC (3MHz)[1] to PLL
2	Run	Standby	Standby mode entry sequence[2]
	Standby	Run	Clock switch from FIRC (24MHz) to PLL after generating wake-up or reset event
3	VLSR	Standby	Standby mode entry sequence[2]
	Standby	VLSR	Clock switch to FIRC (3MHz)[1] after generating wake-up or reset event

1. CONFIG_REG_GPR[FIRC_DIV_SEL] is 2 and FIRC’s divider is 16.
2. The detail information of standby mode entry sequence refer to S32K3xx reference manual.

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 REQxx filed of PRTNx_COFBx_CLKEN register int the MC_ME module. These fields are cleared after any reset, which disable the clocks of corresponding peripherals. Before initializing a peripheral, the corresponding field in the PRTNx_COFBx_CLKEN peripheral control register need to be set to enable the peripheral clock. Be sure to disable the peripheral before turning off its clock.

4.1. System Clock generation Module (MC_CGM)

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

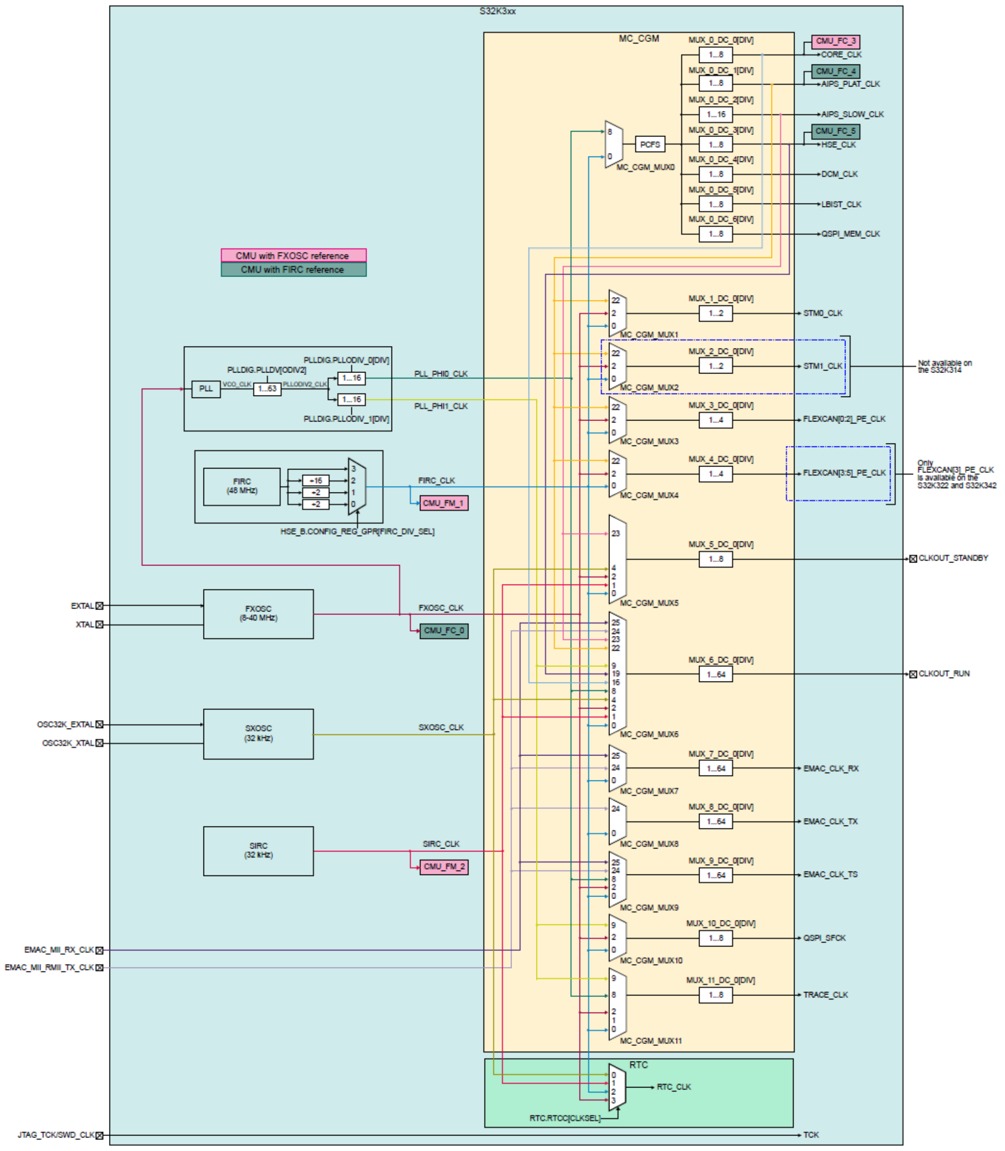


Figure 4. Clock system diagram

NOTE

S32K312 and S32K311 clock system diagram are slightly different. Refer to S32K3xx's reference manual for detailed information.

The system clock generation circuitry provides several clock dividers and selectors allowing different modules to be clocked at a specific frequency. Five main clock sources can be seen on MC_CGM module: Fast Internal Reference Clock (FIRC), Slow Internal Reference Clock (SIRC), Fast external crystal Oscillator (FXOSC), Slow external crystal Oscillator (SXOSC) and Phase-locked loop (PLL). For Run modes, (RUN, VLSR) different sources can be used to provide clock signal to core. The below figure shows all possible sources for core clock that can be used in different power modes.

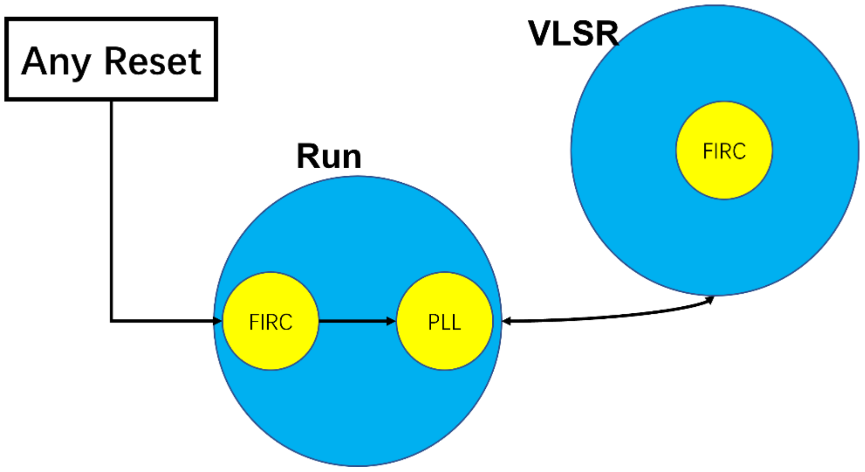


Figure 5. MC_CGM Core clock valid mode transition diagram

For standby mode, as core clock is gated off, no source is used.

When entering standby mode, the system clock must be switched from system PLL to FIRC and system PLL must be disabled by software in RUN mode before making any mode transition. But FIRC and SIRC are optional in standby mode, which can be configured on SIRC and FIRC module.

Before switching clock sources, be sure to meet requirements listed in the section "***Clocking details***" in S32k3xx Reference Manual.

5. Power Management controller (PMC)

The power management system generates, monitors, and controls power supplies of the Run and the standby domains. The below picture shows all possible sources for core clock that can be used in different power modes.

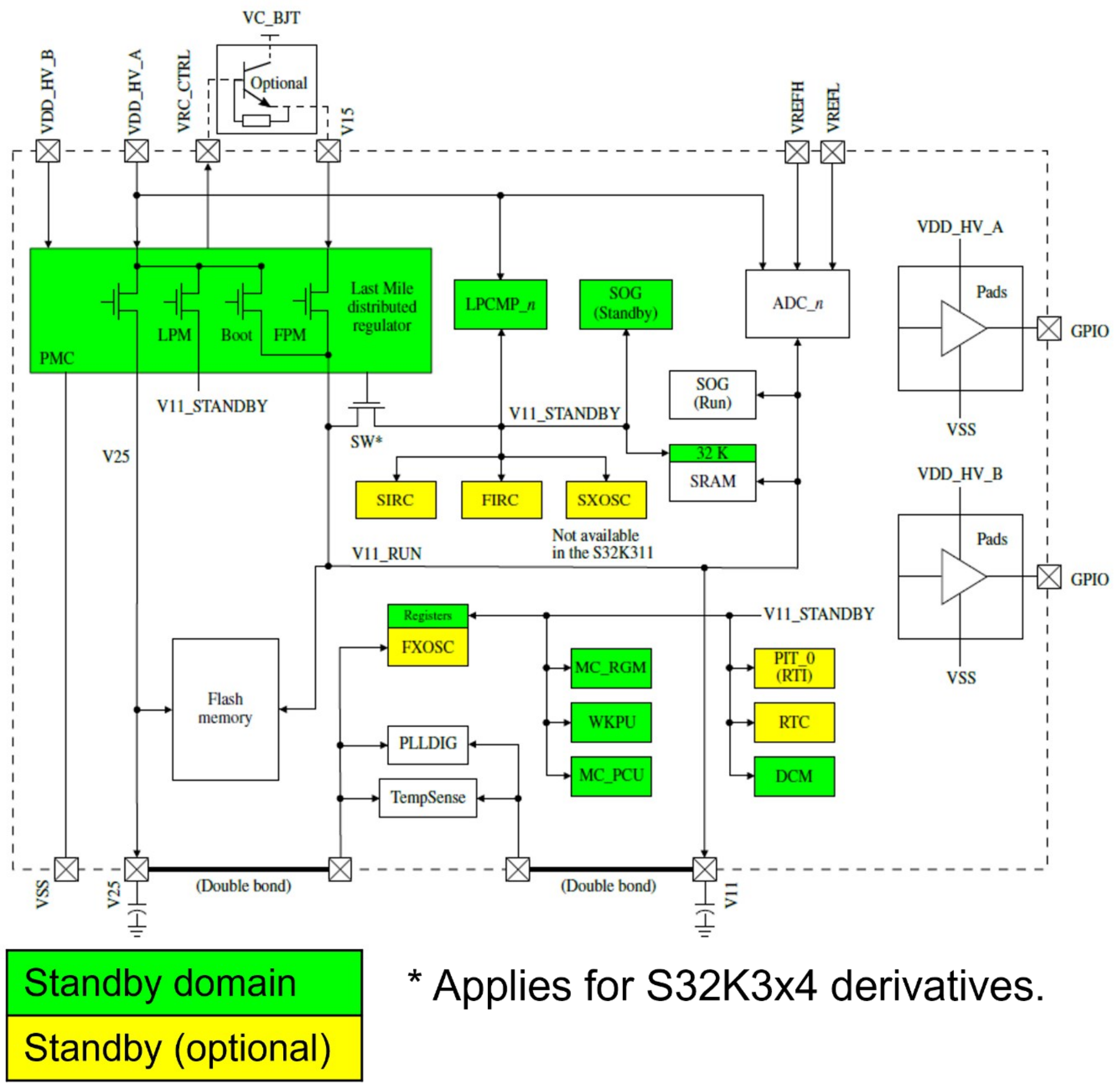


Figure 6. S32K3xx Power management system

S32K3xx power management features:

- Linear regulator to use an optional external ballast (BJT) for 1.5V generation.
- PMC uses 1.5V to generate 1.1V supply for the core logic (Boot, FPM).
- Low-power regulator (LPM) supplying core logic during Standby mode supplies by VDD_HV_A.
- Linear regulator generates a 2.5V supply from VDD_HV_A for Flash, PLL, TempSense & FXOSC, which is optionally enable/disable 2.5V supply for NVM and clocking IP during Standby mode.
- Separates ADC reference supplies (VREFH and VREFL).
- Power-up and I/O rails, supplied by VDD_HV_A or VDD_HV_B.
- Voltage monitors ensuring transitions to a safe state (POR) when a supply is out of range.

Voltage supply details refer to S32K3xx's reference manual.

NOTE

VDD_HV_A must be kept always ON during Run and Standby modes, otherwise a LVR (low-voltage reset) event will be generated. If LVRBLPEN is disabled before entering Standby mode, VDD_HV_B can be OFF during Standby mode and will not generate a low-voltage reset event.

6. Power Mode Entry/Exit

When entering, or exiting low-power modes, the system must follow a correct sequence to manage the transitions safely.

The MC_ME initiates entry into low power mode. The MC_PCU controls entry into and exit from low power mode. The PMC regulates and monitors power supplies of the Run and Standby domains. FIRC, WKUP, and other chip peripherals control Standby mode wake-up and operations in Run and Standby modes.

6.1. RUN mode entry

RUN mode is the main operation mode that has full-chip performance and a higher current consumption as compared to Standby mode. All peripheral modules and Flash are powered. All modules can be clock-gated to reduce power. In RUN mode, the MCU is running on 48MHz from FIRC by default and can switch to max 160MHz from PLL for full performance. The on-chip voltage regulator remains in a running regulation state and uses V15 (1.5V) as input to generate V11 (1.1V) for the cores.

To enter RUN mode (With 160MHz PLL as clock source):

1. Enable last mile regulator bit
2. Configure PLL for RUN mode

6.2. RUN mode exit

Transition from RUN to VLSR mode can be made by either a Reset event or switch system clock from PLL to FIRC. FIRC must be divided by 16 to generate a 3MHz clock for the MCU. As in RUN mode core clock can be set to maximum value (160MHz) and at VLSR mode core clock is down to 3MHz.

6.3. VLSR mode entry

In VLSR mode, the on-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. All modules and Flash are powered. All modules can be clock-gated to reduce power. Default to operate from 48MHz FIRC. FIRC must be divided by 16 to provides a 3 MHz clock source for the core, the bus, the peripheral clocks and flash memory access mode (750 KHz). Refer to the section "Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz)" of S32K3xx reference manual for the detailed clock configuration.

6.4. VLSR mode exit

Transition from VLSR to RUN mode can be made by either a Reset event or System clock switching from FIRC to PLL. Core clock can be set to maximum value (160MHz) in RUN mode.

6.5. Standby mode entry sequence

The Standby mode entry sequence includes three phases of operation:

1. Standby mode entry configuration phase or software Standby mode entry sequence
2. Standby mode entry handshake phase or hardware Standby mode entry sequence
3. Standby mode entry or PMC Standby mode entry

The detailed information can be found in chapter "40.4 Standby mode entry sequence" and chapter "44.6 Standby entry" of S32K3xx reference manual.

6.6. Standby mode exit sequence

This chip supports exiting Standby mode from a wake-up event, a functional reset event, or a destructive reset event. The sources which cause chip exiting Standby mode are:

- MC_RGM functional reset event
- MC_RGM destructive reset event
- WKUP wake-up events, WKUP[0]-WKUP[63]. See the WKUP chapter of S32K3xx reference manual for more information.

After Standby mode exit, the following events occur:

1. A wake-up event arrives
2. FIRC gets powered up (if disabled in Standby mode)
3. PMC starts the transition process to FPM (for example, enables the last-mile regulator and provides V11_RUN supply to the chip)
4. MC_PCU removes the isolation between Run and Standby domains
5. Run domain reset deasserts (it's asserted on Standby entry) and the chip undergoes a functional reset exit sequence for this domain
6. The chip enters Run mode of operation

6.6.1. Normal and Fast Standby mode exit

The chip supports normal and fast exit from Standby mode. The register DCMRWF5 (address 0x402AC610) is used to select normal or fast exit from Standby mode. This register gets clear on POR. Application is supposed to program this register before entering the standby mode. The below flow is normal and fast exiting from Standby mode.

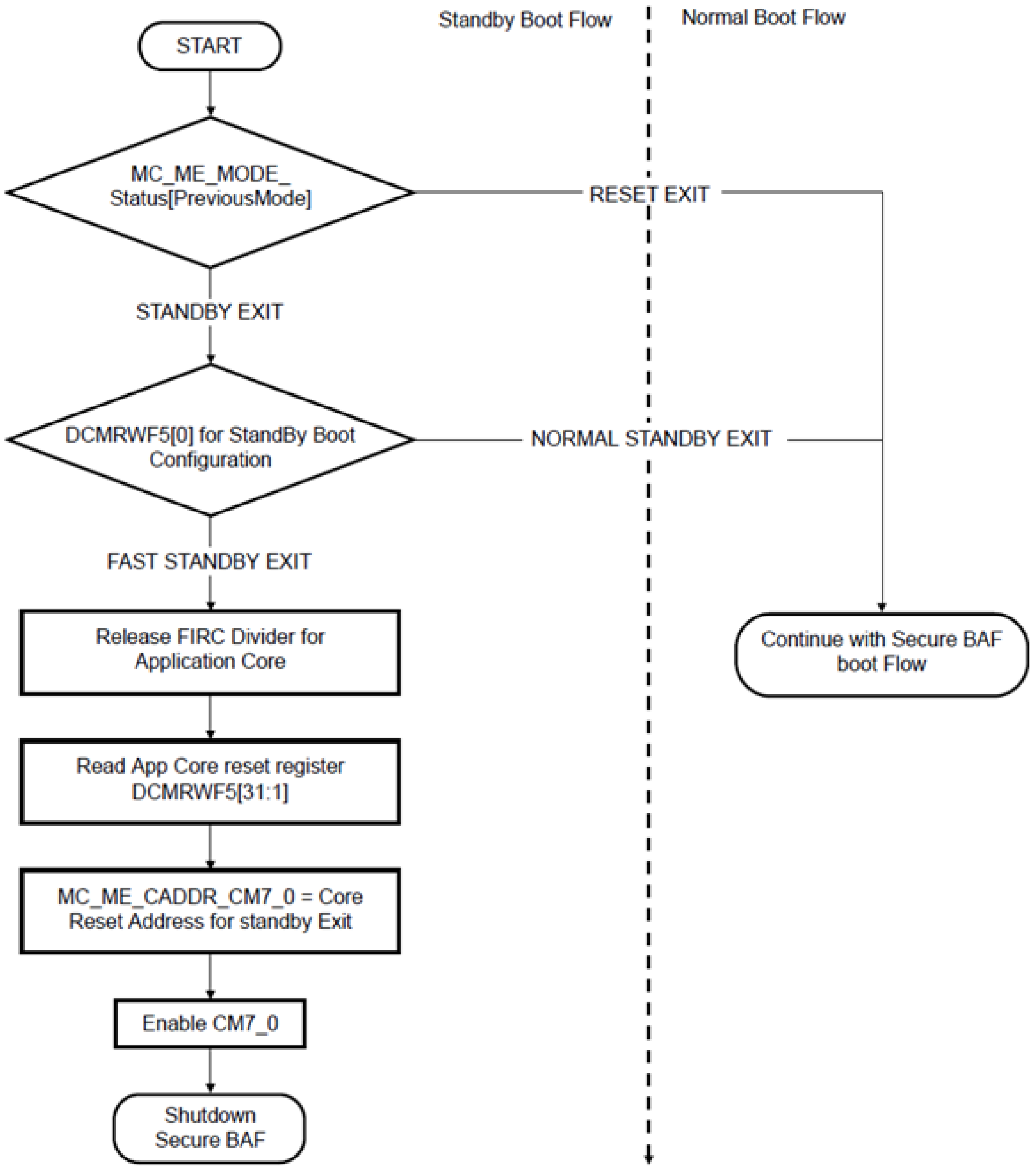

Standby Boot Flow
Normal Boot Flow
START
MC_ME_MODE_ Status[PreviousMode]
RESET EXIT
STANDBY EXIT
DCMRWF5[0] for StandBy Boot Configuration
NORMAL STANDBY EXIT
FAST STANDBY EXIT
Release FIRC Divider for Application Core
Continue with Secure BAF boot Flow
Read App Core reset register DCMRWF5[31:1]
MC_ME_CADDR_CM7_0 = Core Reset Address for standby Exit
Enable CM7_0
Shutdown Secure BAF

Figure 7. Normal and Fast Standby exit flow

The time of fast exit from Standby mode to Run mode is smaller than the time of normal Standby exit. This chip supports an optional feature that bypasses SIRC trimming, FIRC trimming, PMC trimming and DCM scanning phases in case of Standby mode exit sequence by writing 1 to DCM's DCMRWF2[6], DCMRWF2[5], DCMRWF2[4], and DCMRWF2[3] fields respectively before Standby mode entry for bypass operation. This results in a considerable reduction in Standby mode exit duration.

7. Pad keeping in Standby mode

The device is capable to retain the output value of some selected pads when device is power gated in Standby mode. These pads reside in always alive power domain.

To enable this feature, set DCM’s DCMRWF1[STANDBY_IO_CONFIG] to ‘0’ before Standby mode entry. The PAD’s output values are retained until software writes a ‘1’ to this bit on exit from Standby mode. However, if the related SIUL2 registers are correctly configured before writing 1 to this bit on exit from Standby mode, the SIUL2 registers take effective and start to control the behavior of these pads after writing 1 to this bit. If this bit is set to 1 before entering Standby mode, the isolation removal hardware removes pad keeping on Standby mode exit. So the latched PAD values are automatically cleared on Standby exit and the PADs go to Hi-z. The below flow shows pad keeping.

If you write 0 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field before entering standby mode.

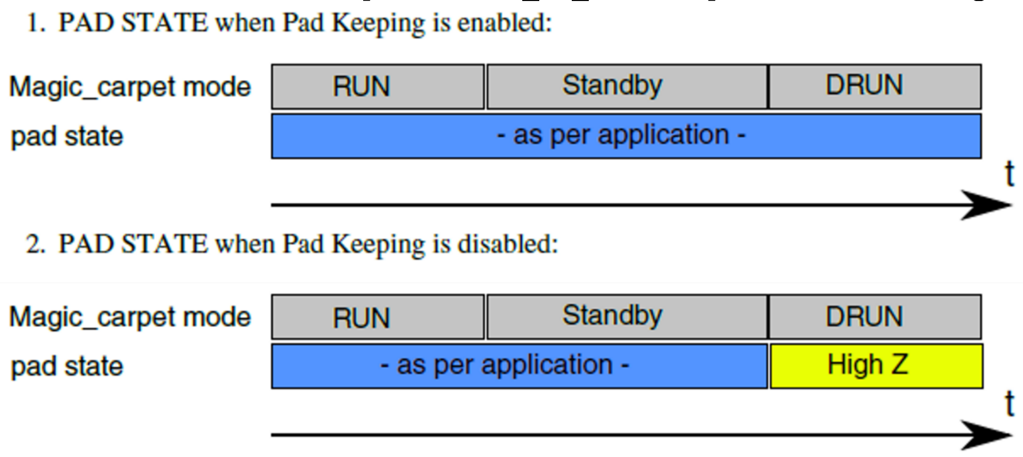


Figure 8. Pad keeping flow

NOTE

If you write 0 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field before entering Standby mode, you must write 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field after exiting Standby modes, otherwise all the configurations of SIUL2 will not work and cannot connect debugger.

8. Modules in Standby modes

The below table shows modules which are available in Standby mode.

Table 3. Modules available in Standby mode

Category	IP
Modes & Power Management	PMC
	MC_PCU
	WKUP
Reset	MC_RGM
Memories	32K Standby RAM

Clocking	FIRC, SIRC, SXOSC, FXOSC, CLKOUT_STDBY
Timer	SWT0
	PIT0 (RTI)
	RTC (API, Timeout)
Analog	3 * LPCMP (24 analog inputs)

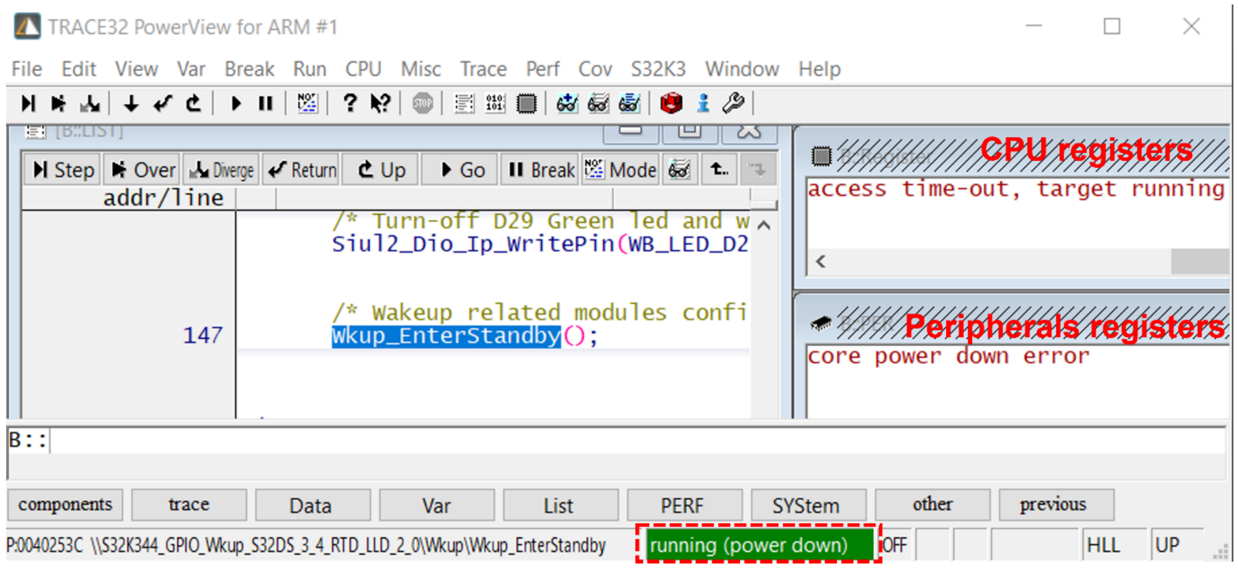
9. Low-power debug handshake protocol

If the debugger handshake is enabled, the debugger must perform some sequence to enter or exit Standby mode. The detail information of the sequence refer to the "Low-power debug handshake protocol" section of S32K3xx reference manual.

The below steps are how to re-connect Lauterbach debugger to S32K344 after standby wakeup.

- Add the command "SYStem.Option PWRDWNRecover ON" to Lauterbach download script
- Refer to the "Low-power debug handshake protocol" section of S32K3xx reference manual to enable the debugger handshake

The below is the result after entering standby mode. The core is power down after entering standby, but Lauterbach's state is running (power down). CPU and peripherals registers cannot be read during standby mode.


TRACE32 PowerView for ARM #1
File Edit View Var Break Run CPU Misc Trace Perf Cov S32K3 Window Help
Step Over Diverge Return Up Go Break Mode
addr/line
CPU registers
Peripherals registers
components trace Data Var List PERF SYStem other previous
OFF HLL UP

```
        /* Turn-off D29 Green led and w
        Siul2_Dio_Ip_WritePin(WB_LED_D2

        /* Wakeup related modules confi
147     Wkup_EnterStandby();
```

```
access time-out, target running
```

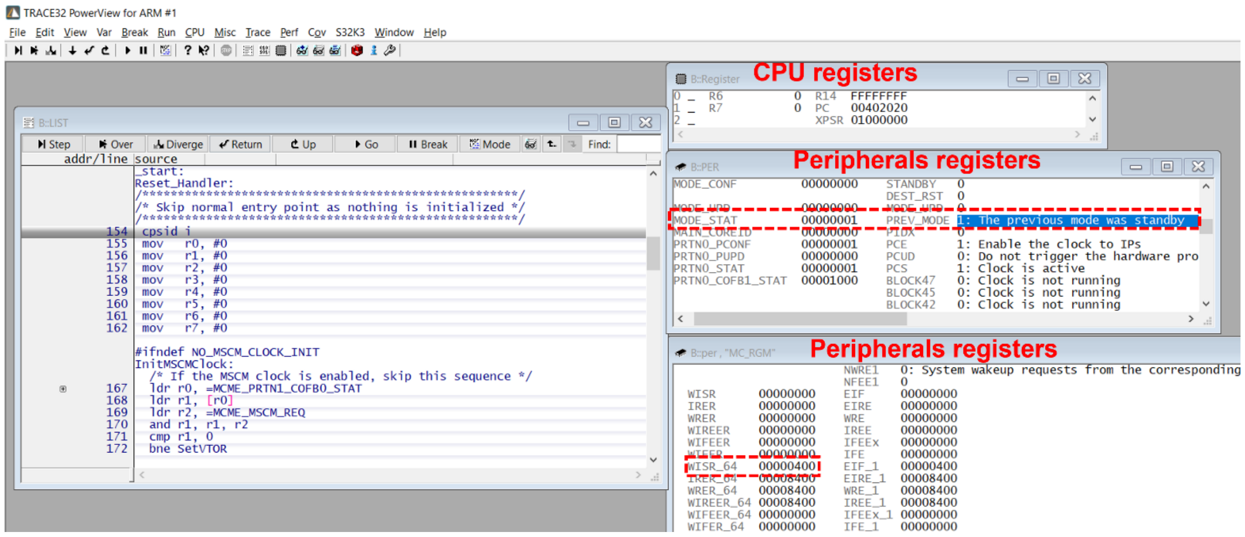
```
core power down error
```

```
B::
```

```
P:0040253C \\S32K344_GPIO_Wkup_S32DS_3_4_RTD_LLD_2_0\Wkup\Wkup_EnterStandby    running (power down)
```

Figure 9. Debug status during standby mode

The below is the result after normally exiting from standby mode. PC pointer will automatically stop "Reset_Handler". From the values of MODE_STAT.PREV_MODE and WISR_64, it can be obtained that MCU is indeed woken up.


TRACE32 PowerView for ARM #1
File Edit View Var Break Run CPU Misc Trace Perf Cov S32K3 Window Help
B::LIST
Step Over Diverge Return Up Go Break Mode Find:
CPU registers
B::Register
Peripherals registers
B::PER
Peripherals registers
B::per , "MC_RGM"

```
addr/line source
_start:
Reset_Handler:
/*************************************************************/
/* Skip normal entry point as nothing is initialized */
/*************************************************************/
 cpsid i
 mov   r0, #0
 mov   r1, #0
 mov   r2, #0
 mov   r3, #0
 mov   r4, #0
 mov   r5, #0
 mov   r6, #0
 mov   r7, #0

#ifndef NO_MSCM_CLOCK_INIT
InitMSCMClock:
  /* If the MSCM clock is enabled, skip this sequence */
  ldr r0, =MCME_PRTN1_COFB0_STAT
  ldr r1, [r0]
  ldr r2, =MCME_MSCM_REQ
  and r1, r1, r2
  cmp r1, 0
  bne SetVTOR
```

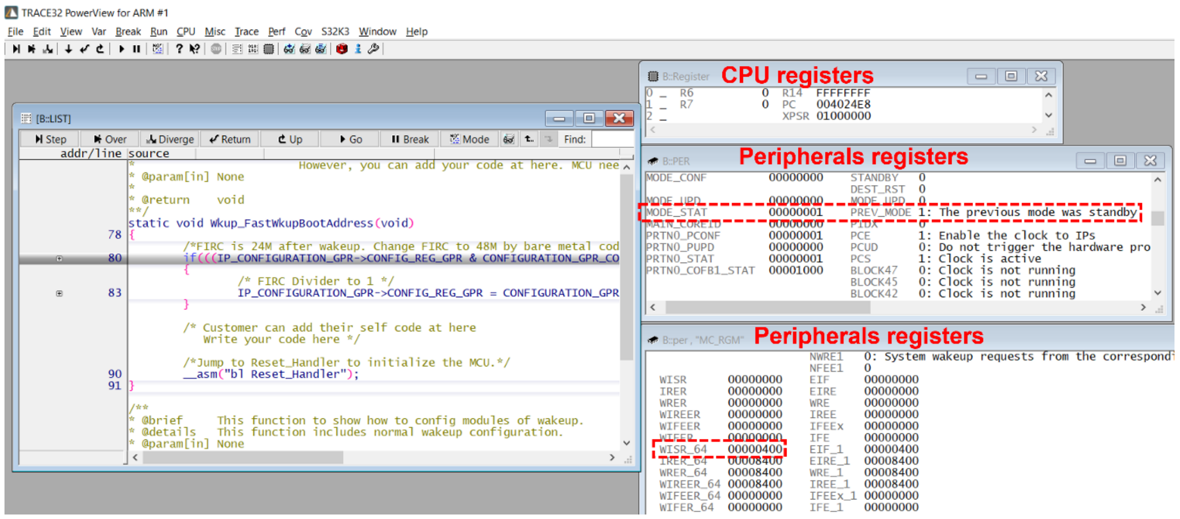
```
0 _   R6          0  R14   FFFFFFFF
1 _   R7          0  PC    00402020
2 _                  XPSR  01000000
```

```
MODE_CONF          00000000    STANDBY    0
                               DEST_RST   0
MODE_UPD           00000000    MODE_UPD   0
MODE_STAT          00000001    PREV_MODE  1: The previous mode was standby
MAIN_COREID        00000000    PIDX       0
PRTN0_PCONF        00000001    PCE        1: Enable the clock to IPs
PRTN0_PUPD         00000000    PCUD       0: Do not trigger the hardware pro
PRTN0_STAT         00000001    PCS        1: Clock is active
PRTN0_COFB1_STAT   00001000    BLOCK47    0: Clock is not running
                               BLOCK45    0: Clock is not running
                               BLOCK42    0: Clock is not running
```

```
                       NWRE1     0: System wakeup requests from the corresponding
                       NFEE1     0
 WISR        00000000  EIF       00000000
 IRER        00000000  EIRE      00000000
 WRER        00000000  WRE       00000000
 WIREER      00000000  IREE      00000000
 WIFEER      00000000  IFEEx     00000000
 WIFER       00000000  IFE       00000000
 WISR_64     00000400  EIF_1     00000400
 IRER_64     00008400  EIRE_1    00008400
 WRER_64     00008400  WRE_1     00008400
 WIREER_64   00008400  IREE_1    00008400
 WIFEER_64   00000000  IFEEx_1   00000000
 WIFER_64    00000000  IFE_1     00000000
```

Figure 10. Low Power Debug of normal wake-up

The below is the result after fast exiting from standby mode. PC pointer will automatically stop "FastWkupBootAddress". From the values of MODE_STAT.PREV_MODE and WISR_64, it can be obtained that MCU is indeed woken up.


TRACE32 PowerView for ARM #1
File Edit View Var Break Run CPU Misc Trace Perf Cov S32K3 Window Help
[B::LIST]
Step Over Diverge Return Up Go Break Mode Find:
CPU registers
B::Register
Peripherals registers
B::PER
Peripherals registers
B::per , "MC_RGM"

```
addr/line source
*                        However, you can add your code at here. MCU nee
* @param[in] None
*
* @return    void
**/
static void Wkup_FastWkupBootAddress(void)
{
        /*FIRC is 24M after wakeup. Change FIRC to 48M by bare metal cod
        if(((IP_CONFIGURATION_GPR->CONFIG_REG_GPR & CONFIGURATION_GPR_CO
        {
                /* FIRC Divider to 1 */
                IP_CONFIGURATION_GPR->CONFIG_REG_GPR = CONFIGURATION_GPR
        }

        /* Customer can add their self code at here
           Write your code here */

        /*Jump to Reset_Handler to initialize the MCU.*/
        __asm("bl Reset_Handler");
}

/**
* @brief     This function to show how to config modules of wakeup.
* @details   This function includes normal wakeup configuration.
* @param[in] None
```

```
0 _   R6          0  R14   FFFFFFFF
1 _   R7          0  PC    004024E8
2 _                  XPSR  01000000
```

```
MODE_CONF          00000000    STANDBY    0
                               DEST_RST   0
MODE_UPD           00000000    MODE_UPD   0
MODE_STAT          00000001    PREV_MODE  1: The previous mode was standby
MAIN_COREID        00000000    PIDX       0
PRTN0_PCONF        00000001    PCE        1: Enable the clock to IPs
PRTN0_PUPD         00000000    PCUD       0: Do not trigger the hardware pro
PRTN0_STAT         00000001    PCS        1: Clock is active
PRTN0_COFB1_STAT   00001000    BLOCK47    0: Clock is not running
                               BLOCK45    0: Clock is not running
                               BLOCK42    0: Clock is not running
```

```
                       NWRE1     0: System wakeup requests from the correspondi
                       NFEE1     0
 WISR        00000000  EIF       00000000
 IRER        00000000  EIRE      00000000
 WRER        00000000  WRE       00000000
 WIREER      00000000  IREE      00000000
 WIFEER      00000000  IFEEx     00000000
 WIFER       00000000  IFE       00000000
 WISR_64     00000400  EIF_1     00000400
 IRER_64     00008400  EIRE_1    00008400
 WRER_64     00008400  WRE_1     00008400
 WIREER_64   00008400  IREE_1    00008400
 WIFEER_64   00000000  IFEEx_1   00000000
 WIFER_64    00000000  IFE_1     00000000
```

Figure 11. Low Power Debug of fast wake-up

10. Hardware considerations

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

10.1. Hardware considerations

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

- Digital GPIO pin should be floating
- DAC output pin should be floating

- ADC pins should be grounded

NOTE

For unused digital and analog pins, the pin function should be set to DISABLED by setting the corresponding. The DISABLED function is default state for all pins not initialized. For pins with ADC functionality software should not trigger ADC channel conversion on the channel which is multiplexed with the unused pin. For pins with CMP functionality software should not enable CMP channel connected to the unused pin.

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

10.2.1. External

The following suggestions address the most common issues encountered when trying to duplicate the standby currents specs in Data Sheet.

- ***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 Standby mode back to Run. The DMM has changed the range to a micro-amp or nano-amp range while the MCU is in the Standby 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 VDD_ MCU and ground. A 5 K potentiometer across a 3.3 V supply pulls 720 μA. This is huge when considering that the MCU consumes around dozens μA in lowest power modes.
- ***Impedance matching 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 VDD_MCU will be affected by any input higher than VDD_MCU. The higher input pin will supply power to the MCU through the input pin, resulting in negative IDD reading in low-power modes.
- ***Reduce pin loading of the MCU.*** When one GPIO is set as output high in Standby mode and the external load consumes a lot of current from this output pin, this may cause much high current measurement result. This is the most evident when you output high frequency signals to an

output pin, like clock and address/data pins of external memory interfaces.

10.2.2. Internal

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

- ***The clock monitor is not disabled which may cause resets***. Disable all clock monitors.
- ***The CLKOUT signal is being output to a pin***. Any pin that is constantly changing state will draw power.
- ***The clock gate for a module that must acknowledge the MC_ME request is turned off prior to low-power mode entry***.
- ***According to requirements, optional enable FIRC, SIRC, SXOSC, FXOSC.*** The current will increase after enable any these clock.
- ***The frequency of wake-up events is too high***, which means that the MCU spends more time in Run or VLSR 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 needed to accomplish the work***. Throttle the clock with the MC_CGM dividers or reduce the clock. Obviously, the higher MCU frequency brings higher IDD in RUN or VLSR mode. Reducing the clock frequency can lower the current in Run or VLSR mode. However, there is some trade-off. The total power consumed in the application is mainly determined by both the current in RUN mode and the time duration that MCU executes in RUN mode. A high system frequency in RUN mode may have the work done quickly and brings a low accumulated power consumption compared to a low system frequency with long running time.
- ***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.

10.3. Current Consumption measurements

As current consumption depends on various factors such as which modules are enabled, what frequency is feeding Core and other peripherals, temperature conditions, and so on. S32K3xx Datasheet lists the test cases for various power modes. Please refer to the “Table14 STANDBY mode supply currents” in Datasheet for more details about test conditions of the current consumption measurements.

11. Wakeup use cases

S32k3xx Low Power Management includes 17 wakeup use cases demo projects. Every project has “example description” in main.c file which describes the main functions of the project.

12. Quiescent current

Current numbers of Standby mode refer to the “supply currents” section of S32K3xx datasheet.

13. Normal and Fast wakeup time

13.1. Normal wakeup time

The below picture is the time of normal wakeup. SIRC, PMC, FIRC trimming and DCM scanning are enabled. The capacitor fast charging of PMC is enabled. This results in a considerable reduction in Standby mode exit duration.

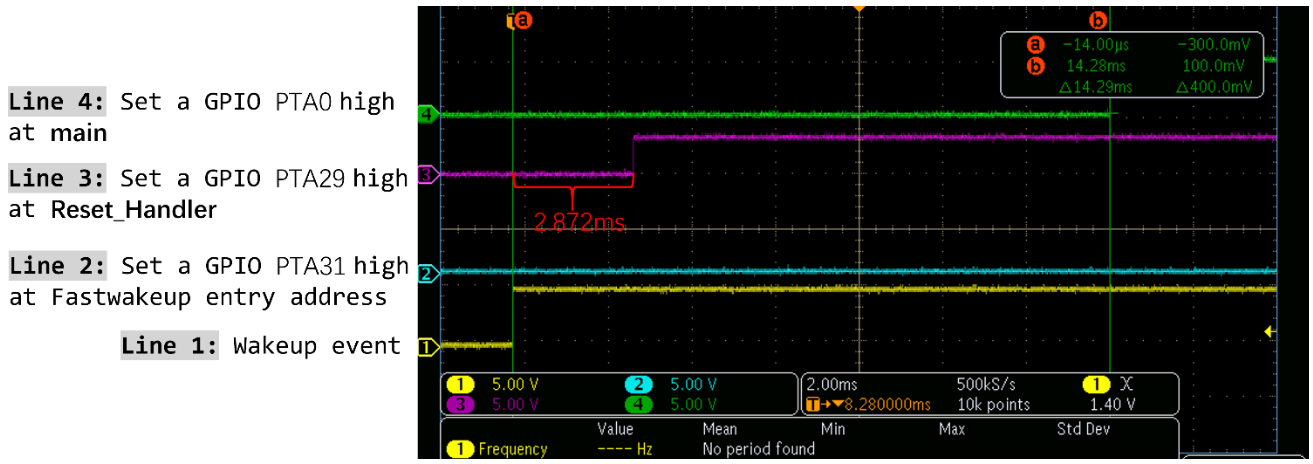


Figure 12. Normal wakeup time

13.2. Fast wakeup time

The below picture is the time of fast wakeup. SIRC, PMC, FIRC trimming and DCM scanning are enabled. The capacitor fast charging of PMC is enabled. This results in a considerable reduction in Standby mode exit duration.

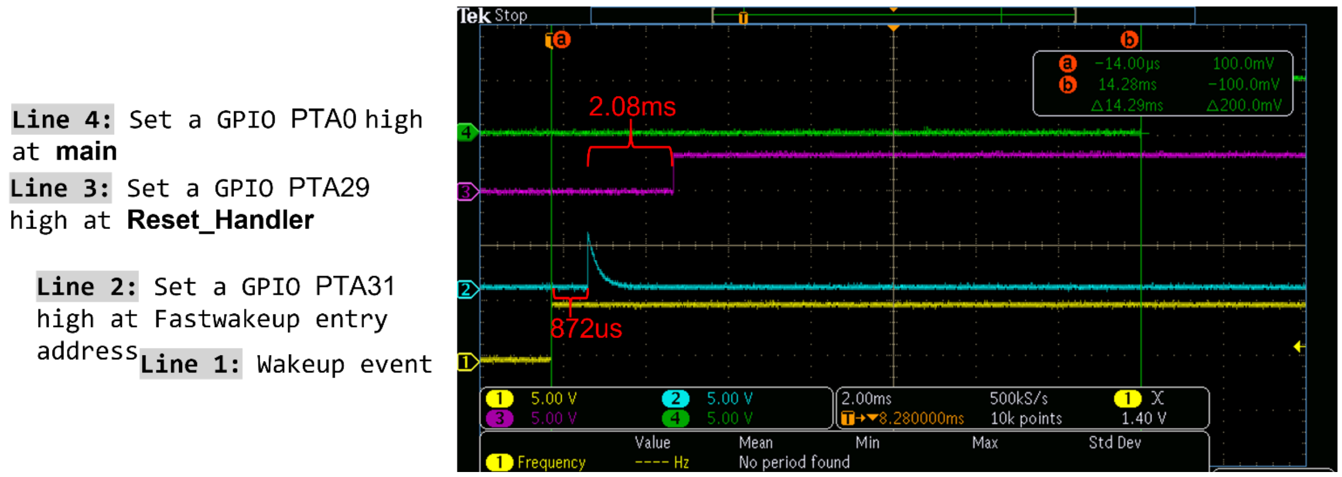


Figure 13. Fast wakeup time

14. Revision History

Version Number	Revision Date	Description of changes
REV0	June 2022	Initial version

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number:ANxxxx
Rev. 0
06/2022