XMC7000 external power supply design guide

About this document

Scope and purpose

This application note explains the procedure to design an external power supply for XMC7000 family MCUs.

Intended audience

This application note is intended for hardware designers.

Associated part family

XMC7000 family of XMC™ industrial microcontrollers.

Table of contents

Table of contents

1 Introduction

This document describes the procedure to design external power supply for XMC7000 MCU family, XMC7100, and XCM7200 series.

These MCUs have two built-in regulators (active regulator and DeepSleep regulator). In addition, MCUs of XMC7100/XCM7200 series have a high current regulator controller (REGHC). REGHC is used to control the external power system. This document explains the design recommendations and restrictions regarding the external power system using the REGHC controller. The document also describes the handover procedure between internal regulators and external power with various use cases.

See Table 1 for external power structures supported by XMC7000.

For more details on the device features and relevant settings, see the XMC7000 architecture reference manual and the dedicated device datasheet.

2 XMC7000 power supply system

The following are the features of the XMC7000 power supply system:

- V_{DDD} power supply voltage range of 2.7 V to 5.5 V
- Core supply[1] at V_{CCD}
- Multiple on-chip regulators:
  - Active regulator to power the MCU in Active or Sleep mode in case of low current consumption
  - DeepSleep regulator to power peripherals in DeepSleep mode. A PMIC may be configured to supply power in DeepSleep power mode, in which case the DeepSleep regulator can be disabled.
- The REGHC, which supports higher load current in Active and Sleep power modes using a pass transistor or by controlling a PMIC

2.1 Block diagram

The XMC7000 device has two built-in regulators (Active regulator and DeepSleep regulator). In addition, the MCU has a REGHC controller for external power supply. The REGHC controller controls V_{CCD} through four terminals: DRV_VOUT and EXT_PS_CTL [0:2]. Figure 1 shows the power supply system for the XMC7000 device.

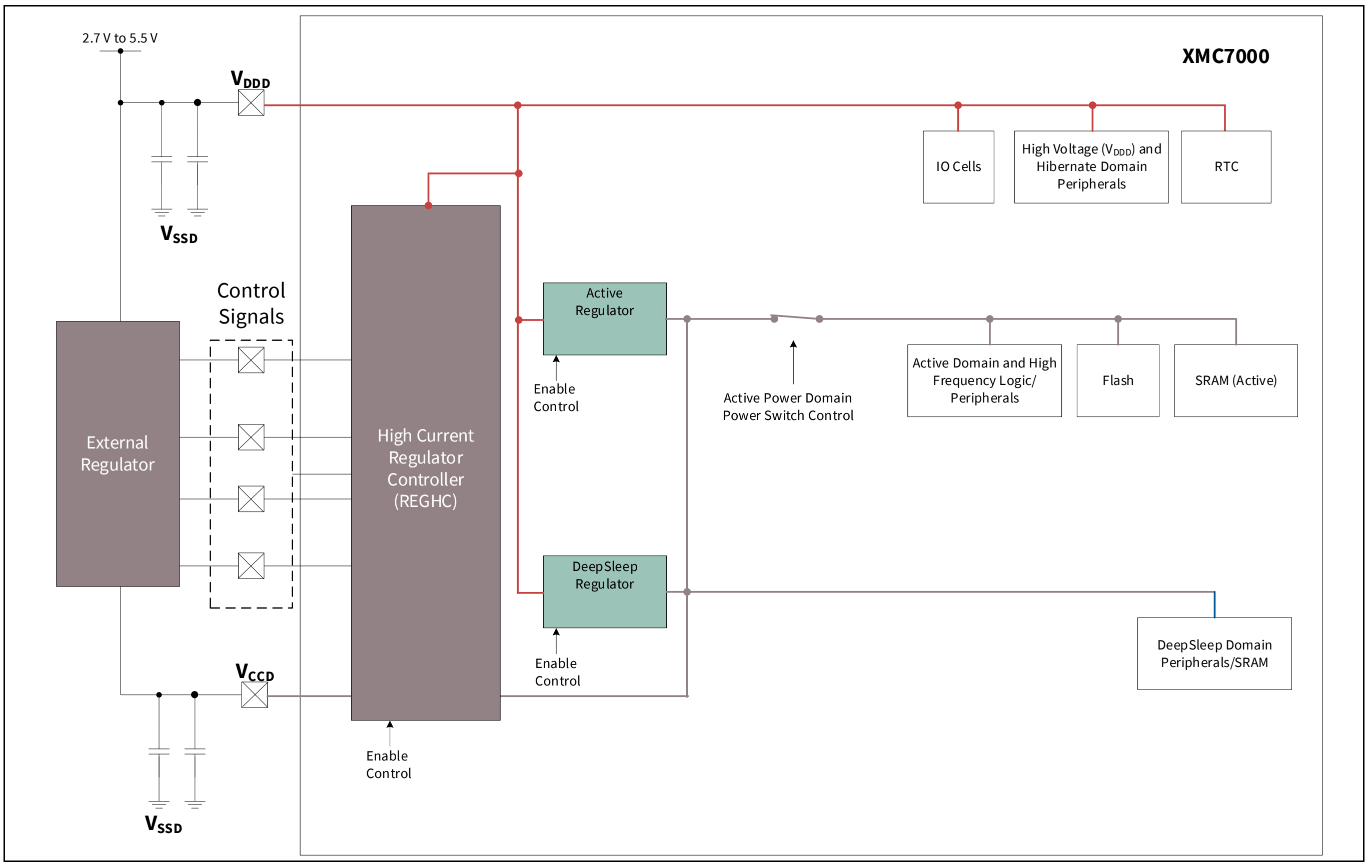

2.7 V to 5.5 V
VDDD
VSSD
XMC7000
IO Cells
High Voltage (VDDD) and Hibernate Domain Peripherals
RTC
Control Signals
External Regulator
High Current Regulator Controller (REGHC)
Active Regulator
Enable Control
Active Power Domain Power Switch Control
Active Domain and High Frequency Logic/ Peripherals
Flash
SRAM (Active)
DeepSleep Regulator
Enable Control
DeepSleep Domain Peripherals/SRAM
VCCD
Enable Control
VSSD

Figure 1 XMC7000 power supply system

[1] See the datasheet for specified core supply voltage.

2.1.1 Internal regulators and REGHC controller

V_{CCD} is connected to the core power supply, but when the load current of the Active regulator exceeds 300 mA, V_{CCD} must be supplied from outside via the REGHC controller.

Note: In Hibernate mode, all regulators are OFF and V_{CCD} is not driven.

The Hibernate circuit operates from V_{DDD} directly. For details, see the "Device power modes" chapter in the architecture reference manual. The device starts with the Active regulator before the handover to the external power supply.

Active regulator:

This is a linear low-dropout (LDO) regulator to power the MCU in Active and Sleep modes. This regulator generates the core voltage (V_{CCD}) from V_{DDD} during Active and Sleep modes. It supports load currents up to 300 mA and is operational during device boot with power ON and high voltage (HV) resets (XRES, POR, BOD, OVD, OCD, HIBERNATE wakeup) or handover to an internal regulator from an external power supply. The Active regulator may support the operation of the MCU at low frequencies if the load current is lower than 300 mA. This regulator can be enabled or disabled by a register setting when the PMIC is enabled. When the Active regulator is enabled, you can use the overcurrent detection (OCD) feature in the Active regulator.

DeepSleep regulator:

This is a linear LDO regulator to power the core voltage (V_{CCD}) during DeepSleep mode. When compared to the Active regulator, this regulator has lower drive capability (up to 18 mA) and lower current consumption. A power source in DeepSleep power mode can be selected in the DeepSleep regulator or the PMIC. The PMIC can be enabled in DeepSleep power mode.

REGHC controller:

The MCU supports higher load currents using the REGHC controller and external components. The REGHC controller controls the handover between an external power supply and the internal regulators. The configuration and control pins of the REGHC controller are initialized only by HV resets (XRES, POR, BOD, OVD, OCD, HIBERNATE wakeup) and not by any low voltage (LV) reset.

Note: When standard GPIO pins are used for external power control, they may be initialized by an LV reset and need to be reconfigured. For more information, see the architecture reference manual.

2.2 REGHC in XMC7000 series

As mentioned above, for controlling the external power supply, XMC7000 series has the REGHC.

2.2.1 External power supply configuration

Table 1 shows the external power supply configuration.

Table 1 External power supply configuration

Item	XMC7000 series
Controller	REGHC
Supported External Power Structure	Pass transistor PMIC direct connection PMIC with load switch
Control pins	See Table 3

2.2.2 REGHC registers

Table 2 shows the REGHC registers for XMC7000. See the registers reference manual for register details.

Table 2 Difference in registers

XMC7000 series	
Register name	**Bit field name**
PWR_CTL2	DPSLP_REG_DIS
PWR_REGHC_CTL	REGHC_CONFIGURED
	REGHC_PMIC_STATUS_WAIT
	REGHC_PMIC_STATUS_ POLARITY
	REGHC_PMIC_STATUS_INEN
	REGHC_PMIC_CTL_POLARITY
	REGHC_PMIC_CTL_OUTEN
	REGHC_PMIC_RADJ
	REGHC_PMIC_USE_RADJ
	REGHC_PMIC_USE_LINREG
	REGHC_VADJ
	REGHC_PMIC_DRV_VOUT
	REGHC_MODE
PWR_REGHC_STATUS	REGHC_SEQ_BUSY
	REGHC_PMIC_STATUS_OK
	REGHC_ENABLED
PWR_REGHC_CTL2	REGHC_EN
PWR_REGHC_CTL4	REGHC_PMIC_DPSLP
	REGHC_PMIC_VADJ_DIS

2.3 Handover between external power supply and internal regulator

This section provides an overview of the handover between external power supply and internal regulator. For more details, see Pass transistor and PMIC (switching regulator).

The REGHC controller is initially disabled and must be enabled by the API. The Active regulator supports the core current until the REGHC controller is configured, enabled, operational, and ready. The core current must not exceed the limit of the Active regulator until the handover is completed. See System Call API for API details.

The REGHC has two configurations; one for a pass transistor and another for a PMIC to drive higher load currents.

Table 3 lists the REGHC pins for each configuration.

Table 3 REGHC pins

Pin Name	Pass transistor		PMIC	
	Direction	Description	Direction	Description
DRV_VOUT	OUT	Base of the pass transistor	Unused	–
EXT_PS_CTL0	IN	Positive terminal of the current sense resistor	IN	Reset out (RO) or Power Good (PG) signal input from the PMIC
EXT_PS_CTL1	IN	Negative terminal of the current sense resistor	OUT	Enable output for the PMIC
EXT_PS_CTL2	Unused	–	OUT (optional)	Reset threshold adjustment for some PMIC

The handover between the external power supply and internal regulator takes place when:

- User software switches between the internal regulator and the external power supply with dedicated APIs and register
- The REGHC controller is disabled by hardware to transition to OFF and XRES states and Hibernate. When reset is released, the MCU starts to operate with the Active regulator. Software enables the REGHC controller after the device reboots. For OFF and XRES states, see the “Device power modes” chapter in the architecture reference manual.

2.3.1 Internal regulator configuration using PMIC

For the REGHC controller to configure the PMIC, enable or disable the Active and DeepSleep regulators.

When the PMIC is enabled with the Active regulator enabled in parallel, you can use the overcurrent detection (OCD) feature of the Active regulator while supplying power from the PMIC. OCD is an integrated part of the Active regulator. Hence, the OCD feature is enabled only when the Active regulator is enabled. In power save modes such as DeepSleep mode, the Active regulator is OFF; therefore, OCD is also OFF. The MCU can detect V_{CCD} brownout, such as the PMIC dropping out of regulation range or being unable to provide a fast load current increase. This is effective when the PMIC does not have the OCD feature. This feature is disabled when the Active regulator is disabled.

The DeepSleep regulator supplies the core supply only in DeepSleep power mode. You can use the register setting to decide whether the PMIC or the DeepSleep regulator supplies the MCU in DeepSleep power mode.

Therefore, you can select the configurations, listed in Table 4, depending on the system.

Table 4 Internal regulator configuration for PMIC case

Use case	Configuration	Active regulator (including integrated OCD)	DeepSleep regulator (including integrated OCD)
Use PMIC without own OCD feature. PMIC is disabled in DeepSleep power mode.	OCD is enabled when supplying power from the PMIC. DeepSleep regulator supplies power in DeepSleep power mode.	Enabled	Enabled
Use PMIC with own OCD feature. PMIC is disabled in DeepSleep power mode.	OCD is disabled when supplying power from the PMIC. DeepSleep regulator supplies power in DeepSleep power mode	Disabled	Enabled
Use a PMIC with own OCD feature. The PMIC is enabled in DeepSleep power mode.	OCD is disabled when supplying power from the PMIC. The PMIC supplies power in DeepSleep power mode.	Disabled	Enabled/Disabled

Operation examples are explained in PMIC direct connection.

Note: *The use of OCD for overcurrent monitoring of the external power supply is not recommended, as the monitoring is not a direct shunt implementation of the V_{CCD} power rail. It is an integral part of the Active regulator.*

Note: *When you set the DeepSleep regulator to "Disabled" (PWR_CTL2.DPSLP_REG_DIS = "1"), you cannot switch back from the PMIC to the internal regulator. If your system is required to switch back to the internal regulator again after handover to the PMIC, the DeepSleep regulator should be enabled. PWR_CTL2.DPSLP_REG_DIS can be set by the* `SwitchOverRegulators` *API with blocking call or application software.*

2.3.2 System Call API

As mentioned above, when a handover is performed between the external power supply and internal regulators, a specific System Call API and register are used.

The XMC7000 device has a supervisory ROM that contains the Boot ROM code and SROM APIs. SROM APIs are designed to be used with Arm® Cortex®-M0+ (CM0+) for specific operations, such as flash programming and lifecycle change. These APIs are also used for the handover between the external power supply and internal regulators. The APIs control the REGHC controller appropriately by setting the register and handling according to the API parameters and execute the handover. For details on the APIs, see the "Non-volatile memory programming" chapter in the architecture reference manual.

Note: It is recommended using system calls from CM0+ for performing handover, instead of writing directly to registers.

The following APIs are used for the handover:

ConfigureRegulator API:

This API is called initially to configure the only desired regulator before the handover between the external power supply and internal regulator by the `SwitchOverRegulators` API. This API is required if PWR_REGHC_CTL.REGHC_CONFIGURED = 0. Table 5 lists the API parameters.

Table 5 Parameters of ConfigureRegulator API

Register/ SRAM_SCRATCH	Name	Bits	Description
IPC_STRUCT.DATA0	SRAM_SCRATCH_ADDR1	[31:0]	Address of SRAM_SCRATCH1 where the API parameters are stored. Must be a 32-bit-aligned address.
IPC_STRUCT.DATA1	SRAM_SCRATCH_ADDR2	[31:0]	Address of SRAM_SCRATCH2 where the API parameters are stored. Must be a 32-bit-aligned address.
SRAM_SCRATCH1	Opcode	[31:24]	Opcode for the `ConfigureRegulator` API 0x15
	RadjValue	[15:13]	This field is used to configure the reset voltage adjustment for the PMIC. For details, see PWR_REGHC_CTL.REGHC_PMIC_RADJ in the registers reference manual.
	VADJ trim value	[12:8]	This field is used to provide the adjust value to obtain the desired voltage output from the REGHC. When operating mode (Bits[1]) is external transistor, this field set to 16 (0x10). For details, see PWR_REGHC_CTL.REGHC_VADJ in the registers reference manual.
	Vadj	[7]	This bit configures to REGHC_PMIC_VADJ_DIS. This bit should be set to "1" in the PMIC configuration because DRV_VOUT pin is not used in XMC7000 MCU. For details, see PWR_REGHC_CTL4. REGHC_PMIC_VADJ_DIS in the registers reference manual.
	UseRadj	[6]	This bit configures the use of reset voltage adjustment for the PMIC. 0: No use RADJ. 1: Use RADJ For details, see PWR_REGHC_CTL. REGHC_PMIC_USE_RADJ in the registers reference manual.
	UseLinReg	[5]	This bit keeps the Active regulator enabled in external PMIC mode. This bit field is valid when operating mode is '1' (External PMIC mode). 0: Internal Active regulator is disabled after PMIC enabled. 1: Internal Active regulator kept enabled after PMIC enabled. For details, see PWR_REGHC_CTL.REGHC_PMIC_USE_LINREG in the registers reference manual. When the `SwitchOverRegulators` API is called with a blocking call, PWR_CTL2.DPSLP_REG_DIS is set by the API according to this bit configuration.
	DeepSleep	[4]	This bit configures PMIC behavior during DeepSleep. This field is valid when operating mode is '1' (External PMIC mode). 0: PMIC is disabled in DeepSleep power mode.

Register/ SRAM_SCRATCH	Name	Bits	Description
			1: PMIC is enabled in DeepSleep power mode. (DeepSleep regulator is disabled in DeepSleep power mode) For details, see PWR_REGHC_CTL4.REGHC_PMIC_DPSLP in registers reference manual. When the `SwitchOverRegulators` API is called with a blocking call, PWR_CTL2.DPSLP_REG_DIS is set by the API according to this bit configuration.
	Reset Polarity	[3]	This sets the PMIC error condition (PMIC Reset out (RO) asserted level or PMIC PG signal de-asserted level). It is used to trigger a reset action based on the RO or PG signal input. If reset occurs by this factor, it is indicated in the RES_CAUSE.RESET_PMIC. This field is valid when operating mode is set to 1 (External PMIC mode). 0: Trigger reset when RO or PG signal input is "0". (RO or PG signal indicates power good status with "1") 1: Trigger reset when RO or PG signal input is "1" (RO or PG signal indicates power good status with "0") For more details, see PWR_REGHC_CTL.REGHC_PMIC_STATUS_POLARITY in the registers reference manual. Also, this API sets the PWR_REGHC_CTL.REGHC_PMIC_STATUS_INEN bit.
	Enable Polarity	[2]	The polarity is used to enable the PMIC. REGHC uses REGHC_PMIC_CTL_POLARITY to enable the PMIC, and it uses the complement to disable the PMIC. This field is valid when operating mode set to 1. (External PMIC mode) 0: PMIC is enabled by "0". 1: PMIC is enabled by "1". For more details, see PWR_REGHC_CTL.REGHC_PMIC_CTL_POLARITY in the registers reference manual. Also, this API sets the PWR_REGHC_CTL.REGHC_PMIC_CTL_OUTEN bit.
	Operating Mode	[1]	This bit defines the operating mode (pass transistor or PMIC). 0: External transistor 1: External PMIC For more details, see PWR_REGHC_CTL.REGHC_MODE in registers reference manual. This bit must match the Operating mode of the `SwitchOverRegulators` API. For devices without REGHC, the operating mode is ignored.
	Others	–	Not used
SRAM_SCRATCH2	WaitCount	[8:0]	Wait count in steps of 4 µs after the PMIC status is OK. This is used by the hardware sequencer to allow

Register/ SRAM_SCRATCH	Name	Bits	Description
			additional settling time before disabling the internal regulator. For more details, see PWR_REGHC_CTL. REGHC_PMIC_STATUS_WAIT in registers reference manual. Note that `ConfigureRegulator` API supports Wait Count field [8:0], it does not support Wait Count bit [9]. Therefore, if you want the Wait Count to be greater than 0x1FF, user software must set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT[29:20] register after the `ConfigureRegulator` API is successful.

When the WaitCount needs to be greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register directly using the application software. Figure 2 shows the flow when REGHC_PMIC_STATUS_WAIT is set.

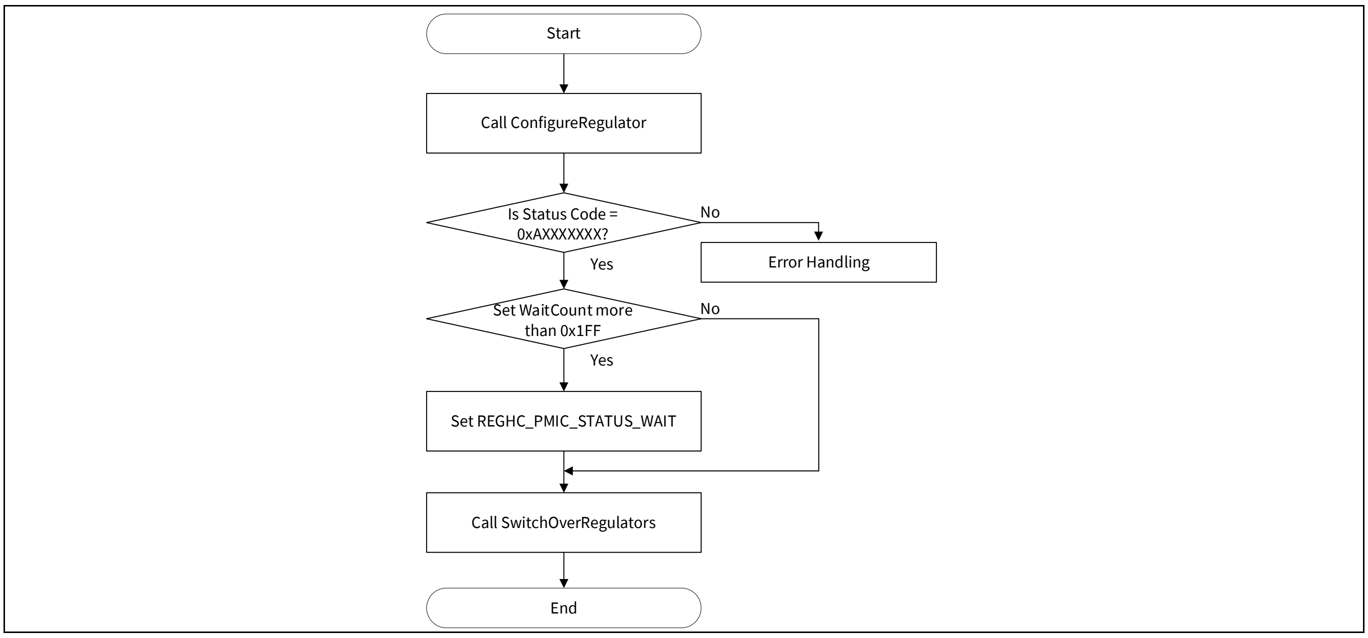

Start
Call ConfigureRegulator
Is Status Code = 0xAXXXXXXX?
No
Error Handling
Yes
Set WaitCount more than 0x1FF
No
Yes
Set REGHC_PMIC_STATUS_WAIT
Call SwitchOverRegulators
End

Figure 2 REGHC_PMIC_STATUS_WAIT setting flow

This register must be set after the completion of the `ConfigureRegulator` API and before calling the `SwitchOverRegulators` API.

Note: When the `ConfigureRegulator` API is called, IPC_STRUCT.DATA1 where SRAM_SCRATCH_ADDR2 is stored should be set a valid user RAM area, even if the application software sets the WaitCount. If you set an invalid address, a bus error may occur.

SwitchOverRegulators API:

This API is used to select between the external power supply and internal regulator for the core voltage supply V_{CCD} in Active power mode.

When an external supply is selected, this API needs to be called only once; the REGHC controller controls the switching between the internal DeepSleep regulator and the external regulator automatically. This API needs to be called before the activation of the application core(s), because the current consumption might be higher than the internal provisioning of the Active regulator. This API must be called after the `ConfigureRegulator` API.

Table 6 shows this API parameters.

Table 6 Parameters of the SwitchOverRegulators API

Register/ SRAM_SCRATCH	Name	Bits	Description
IPC_STRUCT.DATA	SRAM_SCRATCH_ADDR	[31:0]	Address of the SRAM where the API parameters are stored. Must be a 32-bit-aligned address.
SRAM_SCRATCH	Opcode	[31:24]	Opcode for the `SwitchOverRegulators` API 0x11
	Blocking	[23:16]	This field defines whether this API will be blocking CM0+ core or non-blocking. 0: Non-blocking call 1: Blocking call Blocking calls will come out of the syscall only after the complete handover has happened. Non-blocking calls will return the API status even before waiting for the actual handover to happen. That is, if non-blocking calls are used, user software needs to wait for switching to complete. See the SwitchOverRegulators API non-blocking call handling for details. When this API is called in the blocking call, internal regulators will be changed to the proper state for handover to external power supply. Note that if the API is called in non-blocking call, the stage will not change. In addition, set PWR_CTL2.DPSLP_REG_DIS to "1" when this API is called in the blocking call and PMIC is enabled in DeepSleep power mode (UseLinReg = "0" and DeepSleep = "1" of `ConfigureRegulator` API).
	Select regulator	[15:8]	This field is used to define the handover to an external power supply from the Active regulator or to Active regulator from the external power supply. 0: Switch over to the external power supply 1: Switch over to the Active regulator
	Operating Mode	[1]	This bit defines the operating mode (pass transistor or PMIC). 0: External transistor 1: External PMIC For more details, see PWR_REGHC_CTL.REGHC_MODE in the registers reference manual. This bit must match the operating mode of the `ConfigureRegulator` API. For devices without REGHC, the operating mode is ignored.
	Others	–	Not used

LoadRegulatorTrims API:

This API is used to adapt the output voltage to transition to DeepSleep power mode and for the internal regulators during the handover. Table 7 and Table 8 list the parameters of this API. This API must be called every time a load transition requires switching between external and internal regulators, except when using the `SwitchOverRegulators` API with a blocking call.

Table 7 Parameters of the LoadRegulatorTrims API

Register/SRAM_SCRATCH	Name	Bits	Description
IPC_STRUCT.DATA	SRAM_SCRATCH_ADDR	[31:0]	Address of the SRAM where the API parameters are stored. Must be a 32-bit-aligned address.
SRAM_SCRATCH	Opcode	[31:24]	Opcode for the `LoadRegulatorTrims` API 0x16
	Use case	[3:2]	This field defines the case in which the `LoadRegulatorTrims` API is being called. 0: Force trim setting. The syscall will ignore regulator config, as it will set requested trims 1: DeepSleep Entry use case 2: DeepSleep Exit use case 3: Reset Recovery use case See Table 8 for use case description.
	Operating Mode	[1]	This bit specifies output voltage for the internal regulators. 0: Internal regulators 1: External power supply This field is applicable only when the use case is “0”.
	Others	–	Not used

Table 8 Use case of the LoadRegulatorTrims API

Use case	Description
Force trim setting	This use case requires that you bypass the regulator configuration and set the required output voltage. See Table 9 for output states.
DeepSleep entry	In this use case, you need to change the DeepSleep regulator output voltage before the transition to DeepSleep power mode. However, this use case is not valid when the DeepSleep regulator is configured as OFF in DeepSleep power mode.
DeepSleep Exit	This use case describes the system wakeup from DeepSleep power mode. DeepSleep entry must be executed to enter DeepSleep power mode.
Reset Recovery	If a reset interrupt occurs (LV reset) during a handover, it will not affect the power rail transition, but the application is reset. In this situation, you need to use this use case after reset release to verify that the output of the internal regulator complies with the requirements.

Note: The system can transition to Hibernate power mode directly without any prior action.

The `SwitchOverRegulators` and `LoadRegulatorTrims` APIs change the output state of the internal regulators, as listed Table 9, according to the handover transition case. The output voltage of the internal regulators is applied according to the output status.

Table 9 Output state of internal regulators after core power supply transition

Handover transition case	Internal regulator output state	Description
Internal regulators to external power supply	External state	Core is supplied by the PMIC. The PMIC and internal regulators are ON.
	OFF	Core is supplied by the PMIC. The PMIC is ON and internal regulators are OFF.
External power supply to internal regulators	Internal state	Core is supplied by internal regulators. The PMIC is OFF and internal regulator are ON.

A status code is returned to the SRAM_SCRATCH address after the API handling is complete. Table 10 lists the status codes.

Table 10 Status codes

Status code	Description	Handling example
0xAXXXXXXX	API completed successfully. X: don't care	–
0xF00000E0	Regulator is configured for "Manual" mode	Call the `ConfigureRegulator` API.
0xF00000E1	Regulator is currently in transition. - Wait for completion.	Wait for API completion.
0xF00000E2	Regulator is already enabled.	Additional action is not required.
0xF00000E3	Regulator is not configured with `ConfigureRegulator`.	Call the `ConfigureRegulator` API.
0xF00000E4	Returned by the `SwitchOverRegulators` API if syscall is called with a different OpMode parameter other than the `ConfigureRegulator` API.	Call the API with the correct combination.

If these unintended errors reoccur, perform appropriate error handling, such as resetting the entire chip, according to the system design.

2.3.3 Handover sequence overview

This section describes the handover sequence example using APIs and the timing behavior of the REGHC operation.

Figure 3 shows an overview of the handover sequence between the internal regulator and an external power supply. This sequence is used by the `LoadRegulatorTrims`, `ConfigureRegulator`, and `SwitchOverRegulators` APIs with blocking call.

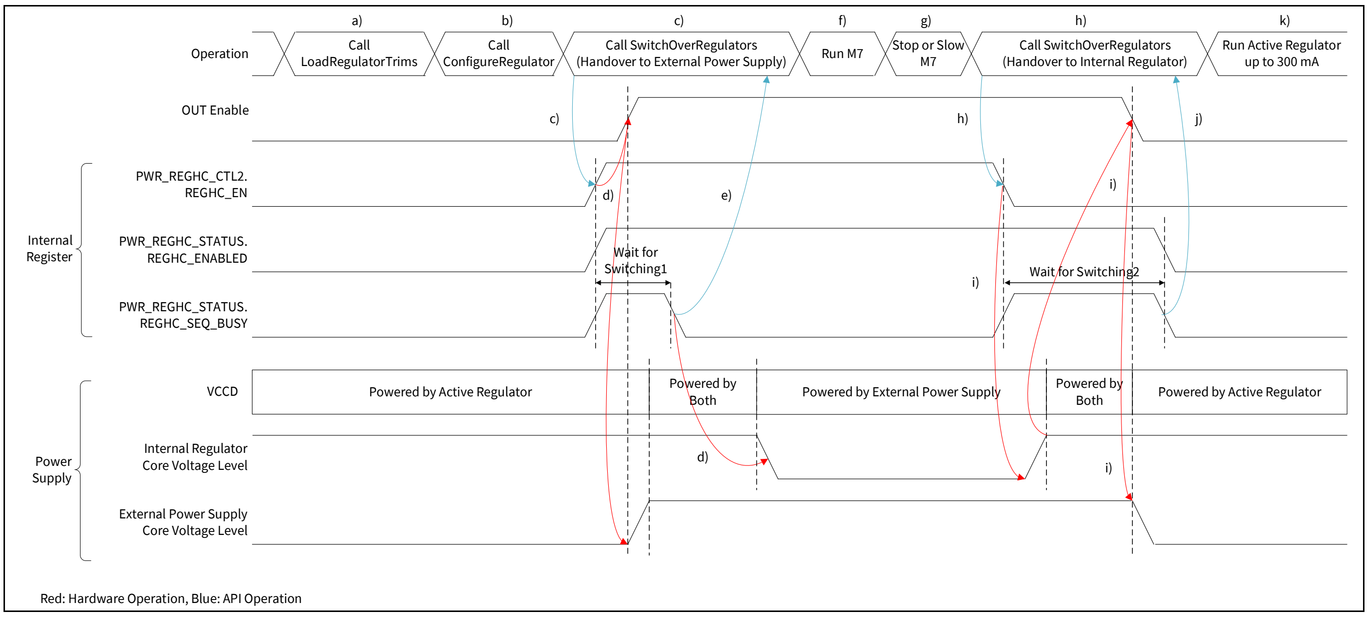

a)
Call LoadRegulatorTrims
b)
Call ConfigureRegulator
c)
Call SwitchOverRegulators (Handover to External Power Supply)
f)
Run M7
g)
Stop or Slow M7
h)
Call SwitchOverRegulators (Handover to Internal Regulator)
k)
Run Active Regulator up to 300 mA
Operation
OUT Enable
Internal Register
PWR_REGHC_CTL2. REGHC_EN
PWR_REGHC_STATUS. REGHC_ENABLED
PWR_REGHC_STATUS. REGHC_SEQ_BUSY
Wait for Switching1
Wait for Switching2
Power Supply
VCCD
Powered by Active Regulator
Powered by Both
Powered by External Power Supply
Powered by Both
Powered by Active Regulator
Internal Regulator Core Voltage Level
External Power Supply Core Voltage Level
d)
e)
i)
j)
Red: Hardware Operation, Blue: API Operation

Figure 3 Handover sequence between internal regulator and external power supply

During startup, the MCU starts with the internal regulator:

Handover from the internal regulator to an external power supply:

a) SW: Call the `LoadRegulatorTrims` API with the reset recovery use case if PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1". See Reset recovery for details.

b) SW: Call the `ConfigureRegulator` API for the REGHC with the pass transistor or PMIC configuration. Select the operating mode according to the external power supply.

Note: *For devices without a REGHC, the operating mode is ignored.*

c) SW: Call the `SwitchOverRegulators` API. Perform handover. Select the same operating mode that was selected in the `ConfigureRegulator` API. Then, set PWR_REGHC_CTL2.REGHC_EN to '1'.

d) HW: The external power supply is enabled, and internal regulators are disabled or set to external state according to the `ConfigureRegulator` API.

e) SW: The `SwitchOverRegulators` API returns the status code. If called with as a blocking call, the API waits for the handover to complete.

"Wait for switching1" corresponds to the wait time for an external power supply to be fully ready. In the pass transistor configuration, wait time completion occurs within 15 µs. In PMIC configuration, wait time completion depends on the power good status signal and the value of PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT.

Software can program REGHC_PMIC_STATUS_WAIT. So, the longer the counter, the longer the hardware sequencer keeps the internal Active regulator enabled. This function can wait for a while before launching the PMIC. For more details, see the registers reference manual.

f) Now, the MCU can run high load cases with the external power supply after successful completion of the `SwitchOverRegulators` API.

Note: *The operating clock frequency should be increased step-by-step by the application software to avoid V_{CCD} voltage undershoot.*

Handover from external power supply to internal regulator:

a) The MCU must run with the clock settings that correspond to the current consumption limits, which are within the specification of the internal regulators before starting the handover.
b) SW: Call the `SwitchOverRegulators` API. Select the same operating mode that was selected in the `ConfigureRegulator` API. Then, set PWR_REGHC_CTL2.REGHC_EN to '0'.
c) HW: Internal regulators are enabled by setting PWR_REGHC_CTL2.REGHC_EN to '0'. When internal regulator starts suppling power, the OUT Enable is deasserted and the external power supply is disabled.
d) SW: The `SwitchOverRegulators` API returns the status code. If called as a blocking call, the API waits for the handover to complete.

"Wait for switching2" corresponds to the waiting time until the internal regulator is ready. In the pass transistor configuration, wait time completion occurs within 10 µs. In PMIC configuration, wait time completion depends on the power good status.

The MCU runs with the internal regulator after the `SwitchOverRegulators` API indicates a successful completion.

Note: *If the transition does not complete within the time specified by "Wait for switching1" and "Wait for switching2" or an unintended API error reoccurs, it is recommended to reset the entire chip, including the power system, by using the WDT or requesting an external system controller trigger XRES_L.*

2.3.4 SwitchOverRegulators API non-blocking call handling

The `SwitchOverRegulators` API has two calling modes: blocking call and non-blocking call. When calling the `SwitchOverRegulators` API in blocking call mode, the API waits for the handover to complete and changes the internal regulator to an appropriate state. In addition, the API sets PWR_CTL2.DPSLP_REG_DIS to "1" when PMIC is enabled in DeepSleep power mode (UseLinReg = "0" and DeepSleep = "1" of the `ConfigureRegulator` API).

However, when the call to the `SwitchOverRegulators` API is applied in non-blocking call mode, the API returns the states even before handover operation starts. In addition, the output state of internal regulators will not be changed and PWR_CTL2.DPSLP_REG_DIS is not configured.

This section describes how to handle the `SwitchOverRegulators` API in non-blocking mode. Table 11 lists the differences between blocking and non-blocking calls.

Table 11 Differences between blocking and non-blocking calls

Operation	Blocking call	Non-blocking call
Wait for handover completion	Handled by the `SwitchOverRegulators` API	Handled by the application software
Internal regulator output state	Handled by the `SwitchOverRegulators` API	Need to run the `LoadRegulatorTrims` API
DeepSleep regulator configuration	Handled by the `SwitchOverRegulators` API When the `ConfigureRegulator` API sets UseLinReg = "0" and DeepSleep = "1" , the `SwitchOverRegulators` API sets PWR_CTL2.DPSLP_REG_DIS to "1"	Handled by the application software. When the `ConfigureRegulator` API sets UseLinReg to "0" and DeepSleep to "1", the `SwitchOverRegulators` API does not set PWR_CTL2.DPSLP_REG_DIS to "1".

2.3.4.1 Behavior of the REGHC_SEQ_BUSY flag

This section describes how to check handover completion by user software. The status of the automatic transition between the internal regulator and PMIC for the core voltage is indicated by the REGHC_SEQ_BUSY status flag.

Transition: Internal regulators to external power supply

When the PG signal is still deasserted after enabling the PMIC, the flag is set to "1". Then, when the PG signal is asserted, the flag is set to "0". The flag indicates that the transition sequence completed properly.

Note: *If the PG signal is asserted during soft-start, the internal regulator may be turned OFF before the PMIC becomes fully operational; it may cause the V_{CCD} voltage to undershoot. You can configure a delay time to mitigate this problem. The flag is not set to “0” by the programmable delay even after the PG signal is asserted by the PMIC. This delay can be controlled by PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT.*

Figure 4 shows the REGHC_SEQ_BUSY operation when enable to the PMIC is active HIGH and power good status from the PMIC is active HIGH.

Transition: External power supply to internal regulators

When the PG signal is still asserted after disabling the PMIC, the flag is set to "1". Then, when the PG signal is deasserted, the flag is set to "0". The flag indicates that the transition sequence completed properly (transition to internal regulators completed).

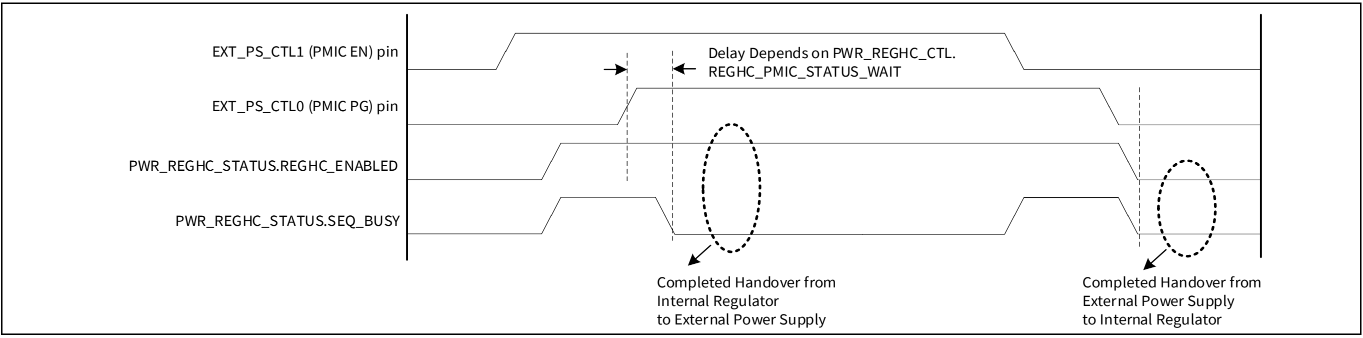


Figure 4 REGHC_SEQ_BUSY operation

- Completion states of handover from internal regulators to the external power supply

  PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0, PWR_REGHC_STATUS.REGHC_ENABLED = 1.

- Completion states of handover from the external power supply to internal regulators

  PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0, PWR_REGHC_STATUS.REGHC_ENABLED = 0.

2.3.4.2 DeepSleep regulator configuration

This section describes the differences in the configuration of internal regulations between blocking and non-blocking calls of the `SwitchOverRegulators` API and how to use each.

The MCU can set the DeepSleep regulator to “Disabled” (set PWR_CTL2.DPSLP_REG_DIS to "1") by disabling the Active regulator and enabling the PMIC in DeepSleep power mode.

The `SwitchOverRegulators` API will disable the DeepSleep regulator when:

- The `ConfigureRegulator` API sets UseLinReg = "0" and DeepSleep = "1"
- The `SwitchOverRegulators` API is called as a blocking call

When the `ConfigureRegulator` API is not configured to set UseLinReg to "0" and DeepSleep to "1" or when the `SwitchOverRegulators` API is called as a non-blocking call, the DeepSleep regulator is not set to "Disabled".

As mentioned above, when the DeepSleep regulator is set to “Disabled” (set PWR_CTL2.DPSLP_REG_DIS to "1"), you cannot run handover from the PMIC to internal regulators using the `SwitchOverRegulators` API. If your system requires to run the handover from the PMIC to internal regulators by the `SwitchOverRegulators` API, you need to call the `SwitchOverRegulators` API as a non-blocking call when running the handover to the PMIC.

2.3.4.3 Example of handover handling in non-blocking call

Figure 5 shows an example on how to run the `SwitchOverRegulators` API as a non-blocking call for handover from internal regulators to the PMIC.

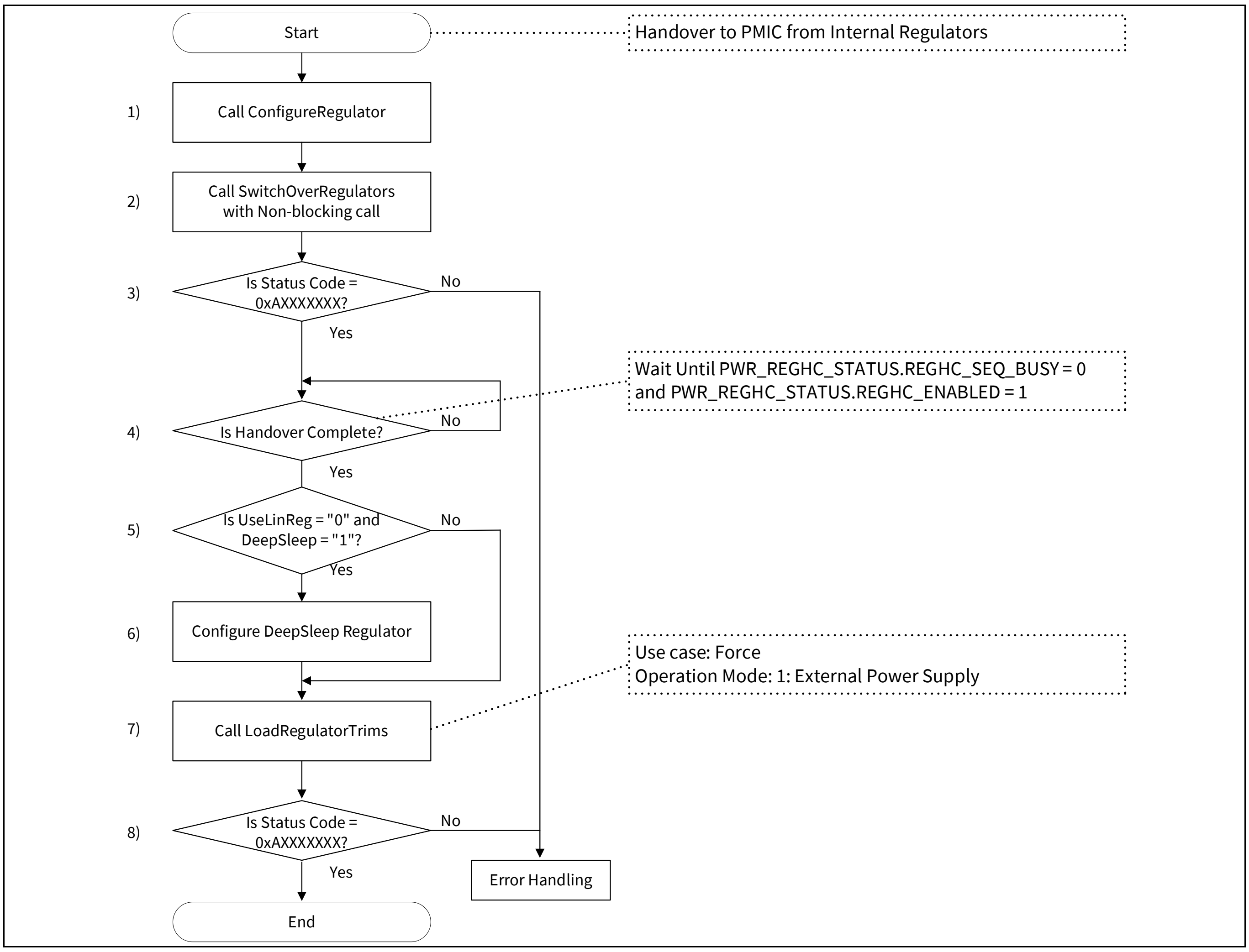

Start
Handover to PMIC from Internal Regulators
1)
Call ConfigureRegulator
2)
Call SwitchOverRegulators with Non-blocking call
3)
Is Status Code = 0xAXXXXXXX?
No
Yes
Wait Until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
4)
Is Handover Complete?
No
Yes
5)
Is UseLinReg = "0" and DeepSleep = "1"?
No
Yes
6)
Configure DeepSleep Regulator
Use case: Force
Operation Mode: 1: External Power Supply
7)
Call LoadRegulatorTrims
8)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
End

Figure 5 Example of handover from the internal regulator to the PMIC in non-blocking call

1. Call the `ConfigureRegulator` API for configuring internal regulators.
2. Call the `SwitchOverRegulators` API as a non-blocking call.
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system. However, the handover is not complete.
4. Wait for handover completion until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
5. Check if the PMIC is enabled in DeepSleep power mode.
6. Set PWR_CTL2.DPSLP_REG_DIS to "1". The DeepSleep regulator is disabled. However, if PWR_CTL2.DPSLP_REG_DIS is set to "1", you cannot run handover from the PMIC to the internal regulators by the `SwitchOverRegulators` API again. Therefore, when your system wants to switch back to internal regulators using the API, do not set PWR_CTL2.DPSLP_REG_DIS to "1".
7. Call the `LoadRegulatorTrims` API with force trim setting use case and external power supply operating mode. The output of internal regulators is changed to external state.

   When the `SwitchOverRegulators` API is called in non-blocking mode, the internal regulators output state may not change to the proper state. In this case, the `LoadRegulatorTrims` API needs to be called to change the output state of internal regulators.

8. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system. However, the handover is not complete.

2.3.5 LoadRegulatorTrims handling for each use case

The `LoadRegulatorTrims` API is used to change the internal regulator output state and has four types of use case settings. See the LoadRegulatorTrims for details.

Force trim setting

This use case will set the internal regulator output state, specified in the operating mode, ignoring the current regulator configuration. In many cases, the internal output state is properly controlled by the `SwitchOverRegulators` API as a blocking call and by the `LoadRegulatorTrims` API for other use cases.

DeepSleep entry

This use case will set the internal regulator output state to internal state before the transition to DeepSleep power mode. Therefore, you need to execute the `LoadRegulatorTrims` API with the DeepSleep entry use case before transitioning to DeepSleep power mode. However, this use case is not required when the DeepSleep regulator is configured as OFF in DeepSleep power mode.

DeepSleep exit

This use case will set the internal regulator output state to external state after the transition from DeepSleep power mode to Active power mode. You need to execute the `LoadRegulatorTrims` API with the DeepSleep exit use case after the system wakes up from DeepSleep power mode and the handover to the PMIC is complete. This use case is required only if the `LoadRegulatorTrims` API with DeepSleep Entry use case was executed to the transition to DeepSleep power mode.

Reset recovery

This use case will set the appropriate internal regulator output state depending on the current REGHC controller status after reset. The REGHC controller configuration is maintained after the LV reset, but the internal regulator output state is initialized. You need to use this use case after reset release and apply the internal regulator output state according to the current situation. Note that some configuration such as GPIO settings are also initialized.

Figure 6 shows an example flow for the `LoadRegulatorTrims` API with the reset recovery use case after reset using the PMIC.

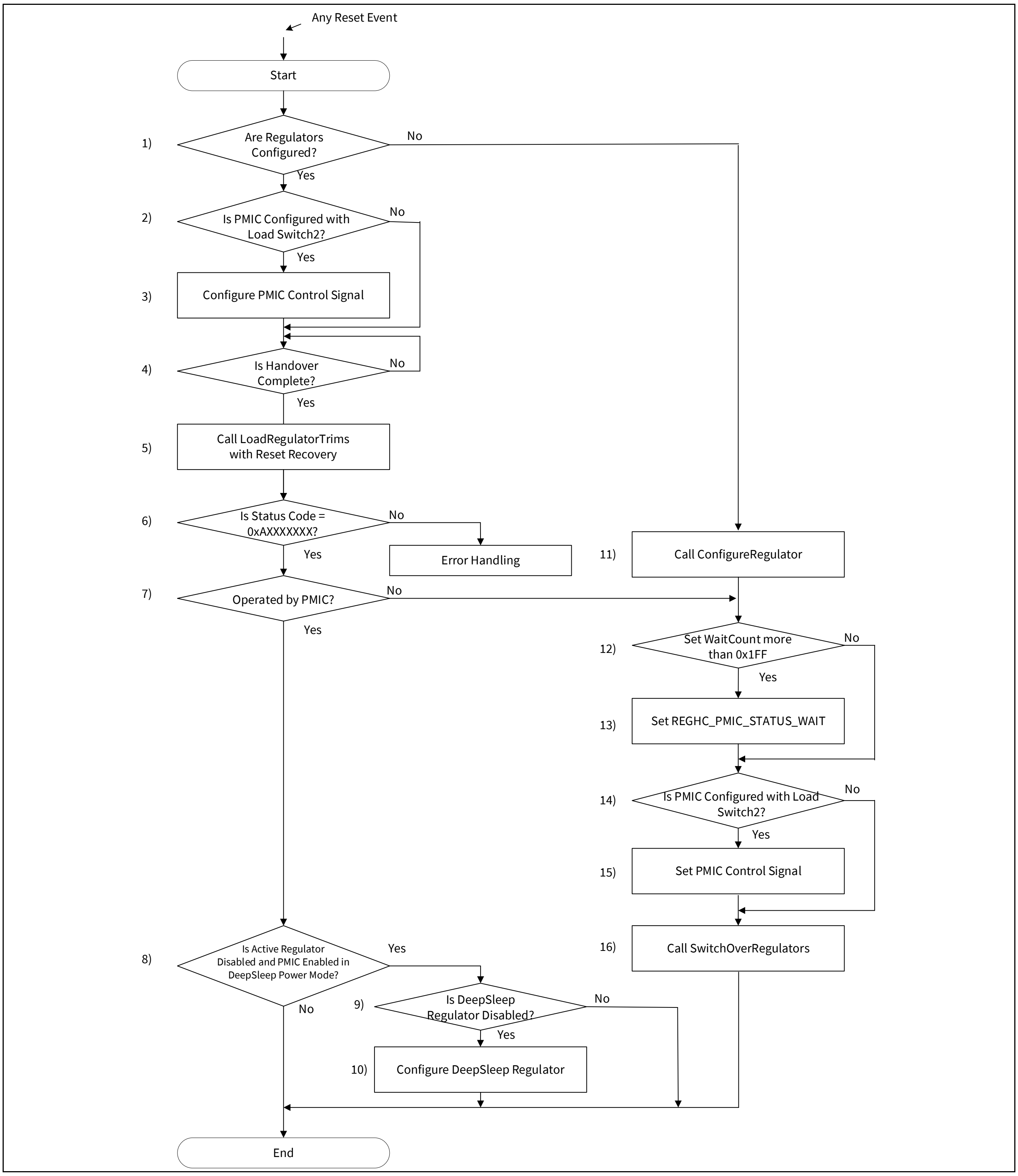

Any Reset Event
Start
1)
Are Regulators Configured?
No
Yes
2)
Is PMIC Configured with Load Switch2?
No
Yes
3)
Configure PMIC Control Signal
4)
Is Handover Complete?
No
Yes
5)
Call LoadRegulatorTrims with Reset Recovery
6)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
7)
Operated by PMIC?
No
Yes
11)
Call ConfigureRegulator
12)
Set WaitCount more than 0x1FF
No
Yes
13)
Set REGHC_PMIC_STATUS_WAIT
14)
Is PMIC Configured with Load Switch2?
No
Yes
15)
Set PMIC Control Signal
16)
Call SwitchOverRegulators
8)
Is Active Regulator Disabled and PMIC Enabled in DeepSleep Power Mode?
Yes
No
9)
Is DeepSleep Regulator Disabled?
No
Yes
10)
Configure DeepSleep Regulator
End

Figure 6 **Example of the LoadRegulatorTrims API with reset recovery**

1. Check if the REGHC controller configuration is already completed.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1", the configuration is already complete and you do not need to call the `ConfigureRegulator` API. When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete.

   If the REGHC configuration is not completed, go to (11).

2. Check if you have the PMIC with Load Switch2 configuration. If not, go to (4).
3. Configure the GPIO for PMIC enable control. In this case, the PMIC enable state before LV reset is reconfigured. The following is an example of GPIO reconfiguration:

   a) Write ˜ (PWR_REGHC_CTL2.REGHC_EN ^ PWR_REGHC_CTL.REGHC_PMIC_CTL_POLARITY) to the GPIO data register.

   b) Configure to GPIO output.

   You need to configure to GPIO output after write GPIO data register. Different procedures cause unintentional output to the PMIC.

   For a different configuration, this operation is not required.

4. Wait for the hardware to complete the handover.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

5. Call the `LoadRegulatorTrims` API with the reset recovery use case.

   The internal regulator output state is set appropriately. If PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the internal regulator output state is set to external state; if it set to "0", the internal regulator output state is set to internal state.

6. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
7. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (12).
8. Check if the Active regulator is disabled and the PMIC is configured to enable in DeepSleep power mode (PWR_REGHC_CTL.REGHC_PMIC_USE_LINREG ="0" and PWR_REGHC_CTL4.REGHC_PMIC_DPSLP = "1"). If not, go to End.
9. Check if the DeepSleep regulator is configured to "Disabled". If not, go to End.
10. Set PWR_CTL2.DPSLP_REG_DIS to "1".
11. Call the `ConfigureRegulator` API for regulator configuration.
12. Check if the WaitCount needs to be configured to a value greater than 0x1FF. If not, go to (14).
13. Set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register using the application software. See the `ConfigureRegulator` API in System Call API for more details.
14. Check if you have the PMIC with Load Switch2 configuration. If not, go to (16).
15. Set the GPIO for PMIC enable control.
16. Call the `SwitchOverRegulators` API for handover to the PMIC.

2.4 Configuration of external power supply

The REGHC has two external power supply configurations: pass transistor and PMIC.

Table 12 lists the features of external power supply configuration.

Table 12 Feature of external power supply configuration

Type	BOM cost	Conversion efficiency
Pass transistor	Low	Low
PMIC	High	High

If a high-efficiency power supply is required, it is better to select a PMIC. However, when using a PMIC, take the precautions discussed in PMIC (switching regulator).

See Table 1 for external power structures supported by XMC7000 series.

3 Pass transistor

3.1 Hardware configuration

Figure 7 shows an example configuration using a pass transistor. It consists of a pass transistor, a smoothing capacitor at V_{CCD} and V_{DDD}, and a sense resistor. The MCU senses voltages at EXT_PS_CTL0 and EXT_PS_CTL1 inputs and controls the DRV_VOUT output.

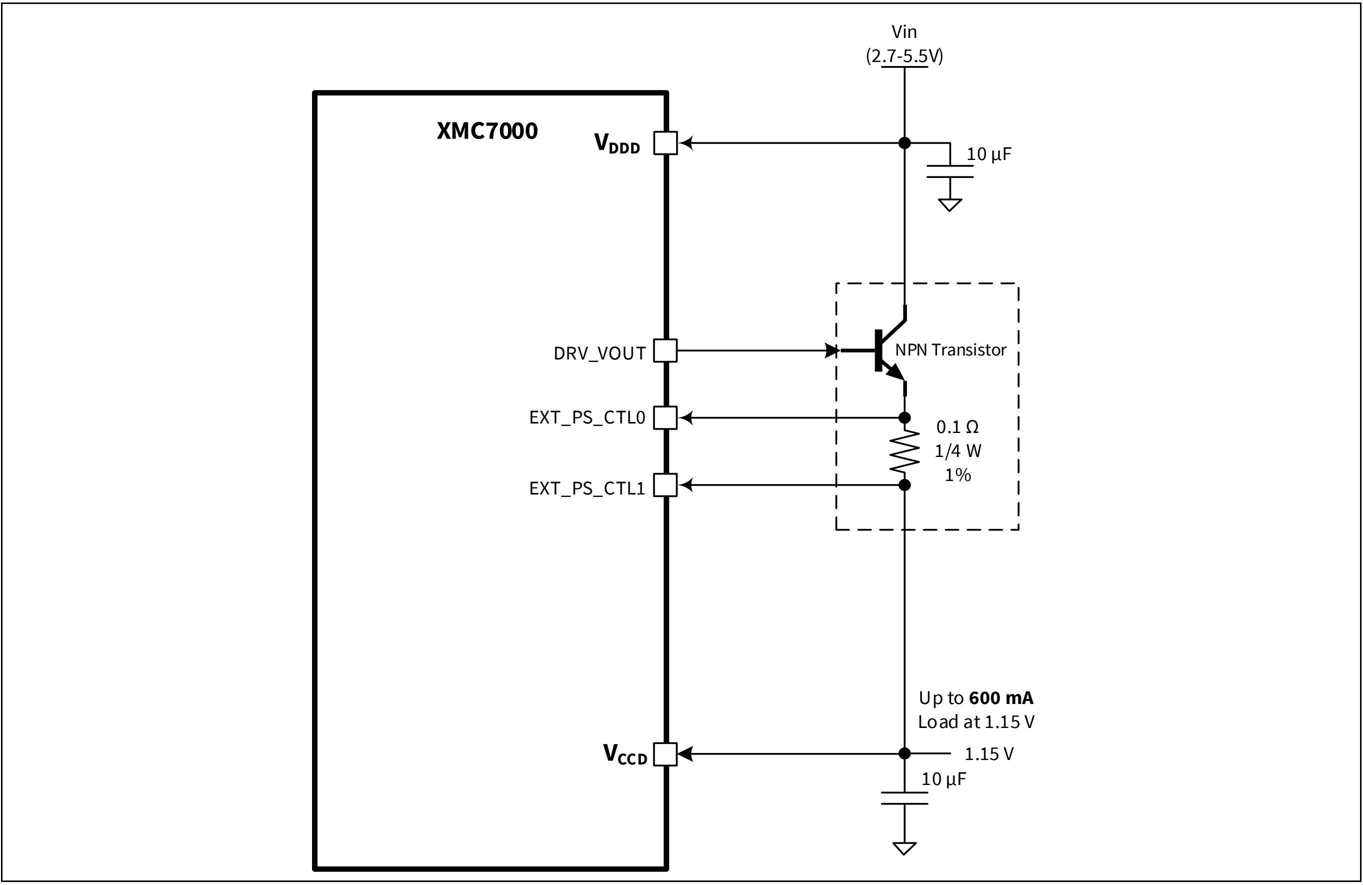


Figure 7 Pass transistor hardware configuration

3.2 Handover timing chart

Figure 8 shows the handover sequence example. This example includes the regulator configuration, handover from internal regulators to the pass transistor, transition to DeepSleep power mode, wakeup from DeepSleep power mode, and handover from the pass transistor to internal regulators.

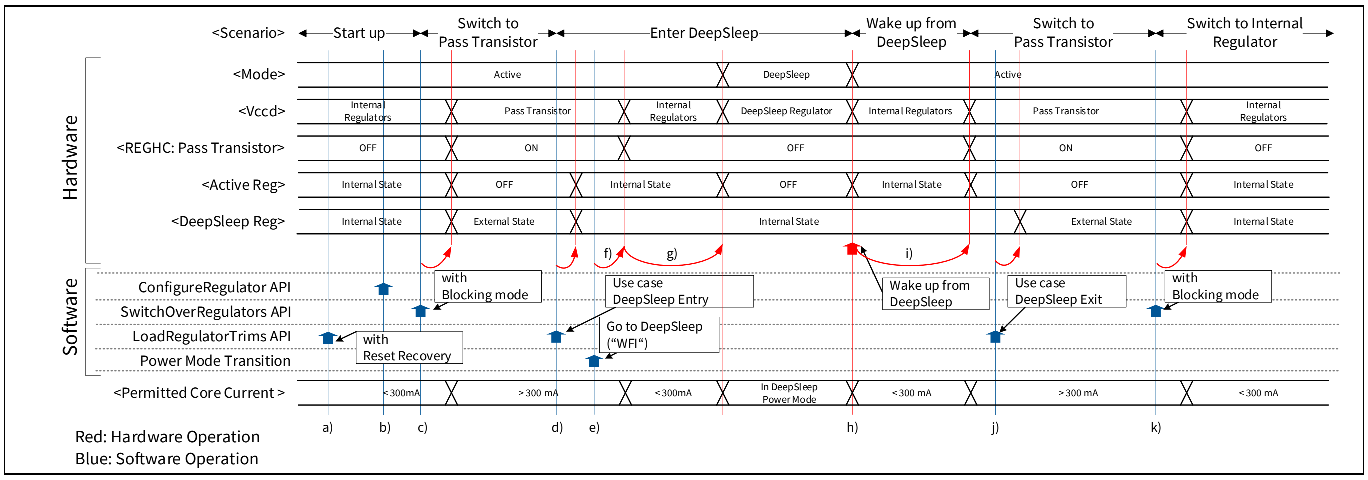


Figure 8 Handover sequence example

a) Call the `LoadRegulatorTrims` API with the reset recovery use case if PWR_REGHC_CTL.REGHC_CONFIGURED is set to “1”. See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC with the pass transistor configuration, after startup.

c) Call the `SwitchOverRegulators` API as a blocking call for handover from internal regulators to the pass transistor. The pass transistor is enabled and Active regulator is disabled. The DeepSleep regulator changes to external state.

d) Call the `LoadRegulatorTrims` API with the DeepSleep entry use case for changing the regulator internal state.

The Active and DeepSleep regulators change to internal state.

e) Reduce the current consumption so that it is within the limit of the Active regulator, which is 300 mA. Then, you can execute the transition to DeepSleep power mode using the “WFI” instruction.

f) When the software executes the transition to DeepSleep power mode, the hardware switches from the pass transistor to internal regulators.

g) The MCU transitions to DeepSleep power mode after completing the switch to internal regulators.

h) A power mode transition from DeepSleep to Active happens after the wakeup event, and then the Active regulator starts with internal state.

i) The pass transistor is enabled and the Active regulator is turned OFF after the completion of Active power mode transition.

j) Call the `LoadRegulatorTrims` API with the DeepSleep exit use case for changing internal regulators to external state.

The DeepSleep Regulators change to external state.

k) Call the `SwitchOverRegulators` API for the handover from the pass transistor to internal regulators.

The pass transistor is disabled. The Active and DeepSleep regulators change to internal state.

3.3 Software flow

3.3.1 Handover from internal regulator to pass transistor

Figure 9 shows the software flow of the handover from the internal regulator to the pass transistor.

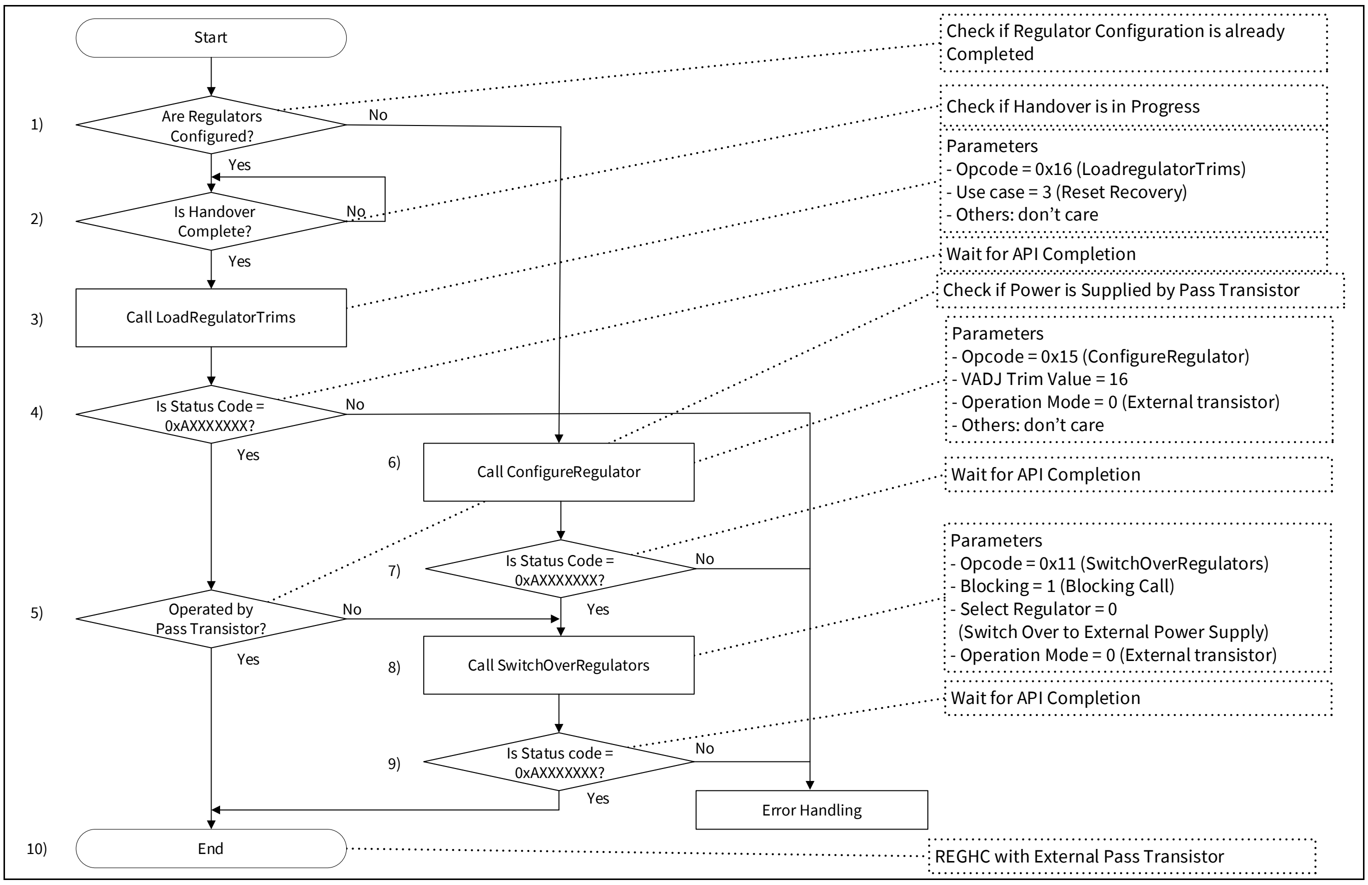

Start
1)
Are Regulators Configured?
No
Yes
2)
Is Handover Complete?
No
Yes
3)
Call LoadRegulatorTrims
4)
Is Status Code = 0xAXXXXXXX?
No
Yes
6)
Call ConfigureRegulator
7)
Is Status Code = 0xAXXXXXXX?
No
Yes
5)
Operated by Pass Transistor?
No
Yes
8)
Call SwitchOverRegulators
9)
Is Status code = 0xAXXXXXXX?
No
Yes
Error Handling
10)
End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by Pass Transistor
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- VADJ Trim Value = 16
- Operation Mode = 0 (External transistor)
- Others: don't care
Wait for API Completion
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 0 (Switch Over to External Power Supply)
- Operation Mode = 0 (External transistor)
Wait for API Completion
REGHC with External Pass Transistor

Figure 9 Flow for handover from internal regulator to pass transistor

1. Check if the regulator configuration is already completed. Once the REGHC is set up, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (6).

2. Wait for the handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset Recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with the pass transistor. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the pass transistor is supplying power. Go to (10). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (8).
6. Call the `ConfigureRegulator` API with the following parameters:

- Opcode = 0x15 (Call the `ConfigureRegulator` API)
- VADJ trim value = 16 to output voltage at 1.15 V with adjustment. See the registers reference manual for more details.
- Operating Mode = 0 (External transistor)
- Others = Don't care

Note: *IPC_STRUCT.DATA1 where the SRAM_SCRATCH_ADDR2 is stored should be set a valid user RAM area. If you set an invalid address, a bus error may occur.*

7. Wait for the completion of the `ConfigureRegulator` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.

To enable the REGHC with the pass transistor, perform these steps:

8. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over external power supply)
   - Operating Mode = 0 (External transistor)
9. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform the appropriate error handling according to your system.
10. The device is now operating REGHC with pass transistor.

Once REGHC is setup, it can be enabled without reconfiguring.

3.3.2 DeepSleep entry and exit

Figure 10 shows the transition to DeepSleep power mode.

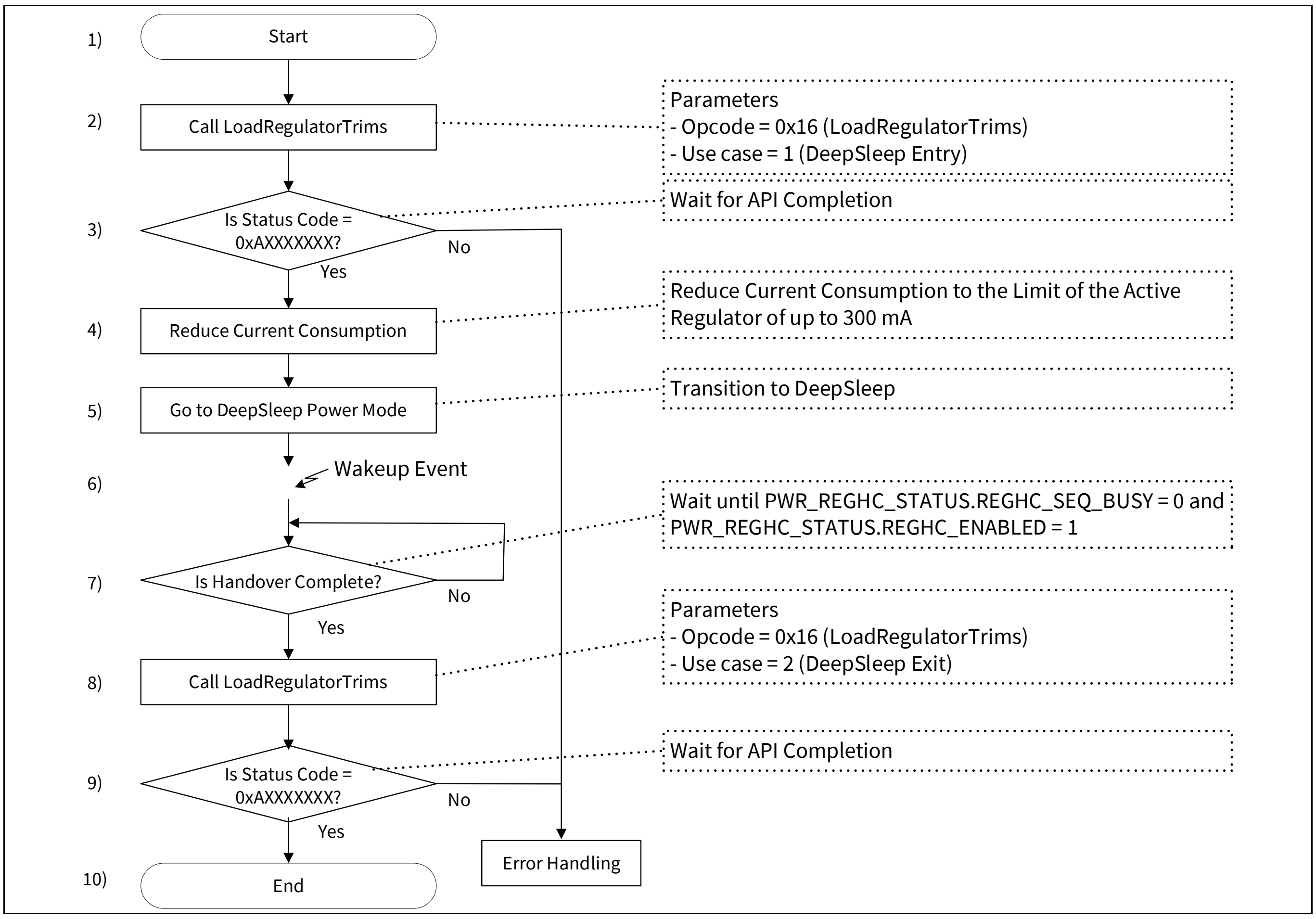

1)
Start
2)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 1 (DeepSleep Entry)
3)
Is Status Code = 0xAXXXXXXX?
Wait for API Completion
No
Yes
4)
Reduce Current Consumption
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
5)
Go to DeepSleep Power Mode
Transition to DeepSleep
6)
Wakeup Event
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
7)
Is Handover Complete?
No
Yes
8)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 2 (DeepSleep Exit)
9)
Is Status Code = 0xAXXXXXXX?
Wait for API Completion
No
Yes
10)
End
Error Handling

Figure 10 DeepSleep entry and exit flow (Pass transistor)

1. The MCU is powered by the pass transistor.
2. Call the `LoadRegulatorTrims` API with the following parameters before the transition to DeepSleep power mode:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 1 (DeepSleep Entry)
3. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. Reduce the current consumption to within the limit of the Active regulator, which is 300 mA.
5. Start the transition to DeepSleep power mode using the “WFI” instruction. The MCU performs the handover to internal regulator mode, and the pass transistor is disabled. Then, the Active regulator is disabled; the DeepSleep regulator supplies power during DeepSleep power mode.

   The MCU is powered by the DeepSleep regulator.
6. Due to the wakeup event, the MCU transitions into Active power mode. Then, the pass transistor is enabled by the hardware.
7. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1

8. Call the `LoadRegulatorTrims` API with the following parameters. Active and DeepSleep regulators are set to external state.
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 2 (DeepSleep Exit)
9. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
10. The MCU is powered by the pass transistor.

3.3.3 Handover from the pass transistor to internal regulators

Figure 11 shows the flow of the handover from pass transistor mode, controlled by the REGHC, to the internal regulator.

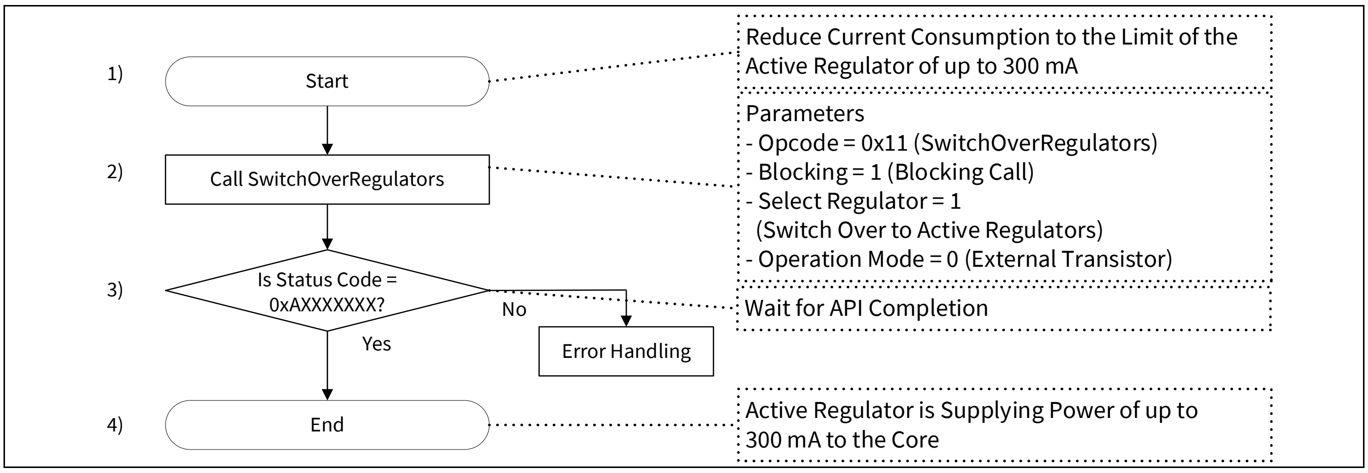

1)
2)
3)
4)
Start
Call SwitchOverRegulators
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
End
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 1 (Switch Over to Active Regulators)
- Operation Mode = 0 (External Transistor)
Wait for API Completion
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 11 Transition flow from the pass transistor to internal regulator

To transition from REGHC with the pass transistor to internal regulator:

1. Reduce the current consumption to within the limit of the Active regulator, which is 300 mA.
2. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 1 (switch over to the Active regulator)
   - Operating Mode = 0 (External transistor)
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. The device is now operating with the Active regulator.

3.4 Component selection

A pass transistor has the following components:

Smoothing capacitor of V_{CCD}:

For capacitance, see the “Recommended operating conditions for the smoothing capacitor” in the device datasheet. Select a ceramic capacitor that does not have the influence of DC bias at 1.15 V with a temperature characteristic of X7R, +/- 10% tolerance.

Pass transistor

Table 13 lists the recommended smoothing capacitor.

Table 13 Recommended smoothing capacitor of V_{CCD}

Specification	Part number	Quantity	Vendor
10 uF, 6.3 V, X7R, +/- 10 percent	CGA4J1X7R0J106K	1	TDK

Current sense resistor: 0.1 ohm 1/4W +/-1 percent.

External transistor type: NPN

Table 14 lists the requirements for an external transistor.

Table 14 External transistor requirement specification (Type: NPN)

Parameter	Symbol	Min value	Unit	Conditions
Static forward current transfer ratio	H_{FE}	100	–	$I_C = 1$ A, $V_{CE} = 1$ V
Collector-emitter voltage	V_{CEO}	10	V	–
Transition frequency	f_T	100	MHz	–
Collector current	I_C	1	A	–
Collector power dissipation	P_{CD}	2	W	–

Additionally, do not exceed the rated temperature; it is necessary to estimate the junction temperature.

The loss at the pass transistor under the operating conditions is calculated by Equation 1:

$$P_{Loss} = (V_{DDD\_max} - V_{CCD\_min}) \times I_{VCCD\_max}$$

Equation 1

Where:

- P_{Loss}: Loss of the pass transistor (W)
- V_{DDD_max}: Maximum V_{DDD} voltage (V)
- V_{CCD_min}: Minimum V_{CCD} voltage (V)
- I_{VCCD_max}: Maximum V_{CCD} load current (A)

The maximum power dissipation of the pass transistor must be greater than or equal to P_{Loss}.

The junction temperature of the pass transistor is calculated by Equation 2:

$$T_J = P_{Loss} \times \theta_{JA} + T_A$$

Equation 2

Where:

- T_J: Pass transistor junction temperature (°C), which must be below the rated temperature
- T_A: Ambient temperature (°C)

- P_{Loss}: Loss of the pass transistor (W)
- θ_{JA}: Thermal resistance from the junction to the ambient temperature (°C/W)

Table 15 shows examples of maximum θ_{JA}:

Table 15 Examples of maximum θ_{JA}

$T_{A\ max}$ (°C)	V_{DDD}	$P_{LOSS\ max}$ (W)	θ_{JA} (°C/W)
+125	5 V power rail	2.64	< 9.5
+105			< 17.0
+85			< 24.6
+125	3.3 V power rail	1.50	< 16.7
+105			< 30.0
+85			< 43.3

Note: V_{DDD} maximum voltage = 5.5 V/3.6 V, V_{CCD} minimum voltage = 1.10 V, V_{CCD} maximum load current = 0.6 A.

The loss by the pass transistor can be reduced by supplying 3.3 V at V_{DDD}.

3.5 Layout design guidelines

Here are a few PCB layout design guidelines:

- Place the V_{CCD} smoothing capacitor close to the MCU, and place it next to the current sense resistor, pass transistor, and V_{DDD} smoothing capacitor.
- DRV_VOUT is sensitive to noise. Do not route the DRV_VOUT trace close to the source of noise.
- Signal routing between the EXT_PS_CTL0 and EXT_PS_CTL1 terminals and the current sense resistor terminal should be separated from the power wiring and should be wired parallel and close to each other. Because these signals are sensitive to noise, keep away from the source of noise and make them as short as possible.
- The pass transistor generates more heat depending on the operating conditions. It is necessary to lay out the board so that the heat is dissipated from the exposed pad (EP) to the board. The land area is placed to cover the EP on the mounting surface of the pass transistor and the bottom of the PCB. The via holes in the footprint of the EP are electrically connected and thermally coupled to the land area where these are placed to cover the EP on each layer of the PCB including the power supply layer. Follow the layout guidelines of the NPN transistor from the supplier for other precautions.

4 PMIC (switching regulator)

This section describes how to power V_{CCD} with a PMIC.

Figure 12 shows an overview of connections to a PMIC.

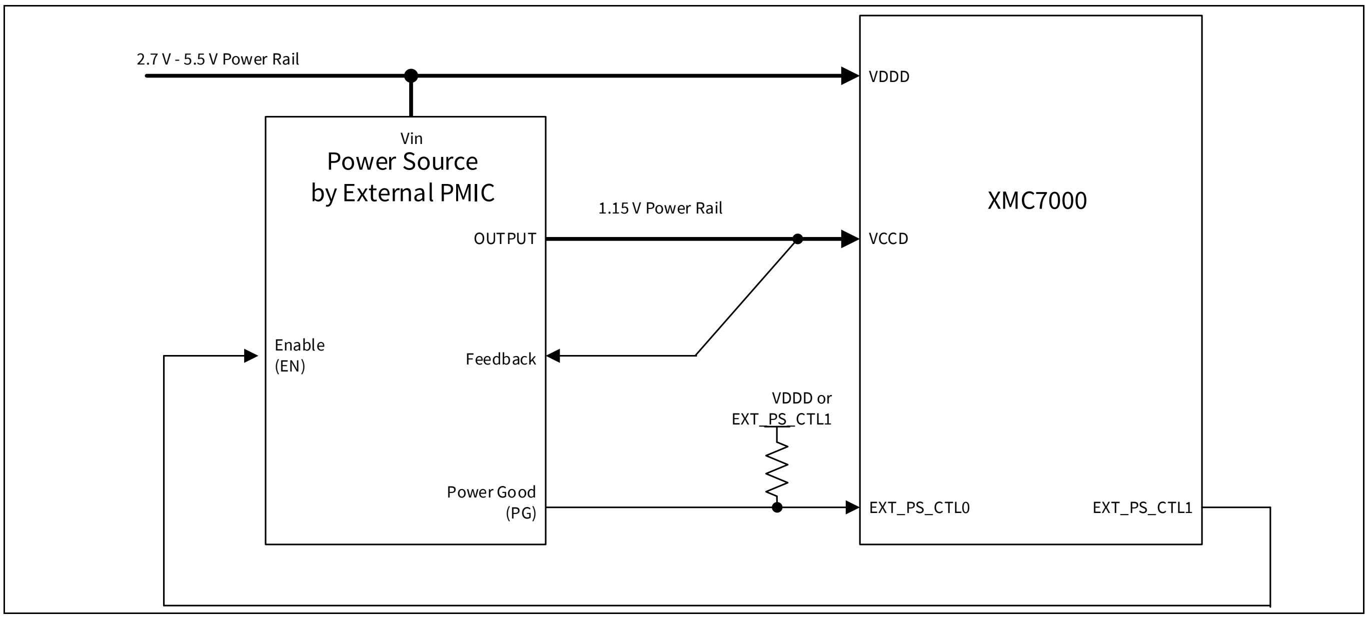


Figure 12 Connections between MCU and PMIC

- OUTPUT of the PMIC is connected to V_{CCD} of the MCU.
- Feedback of the PMIC is connected to V_{CCD} of the MCU.
- EN of the PMIC is connected to EXT_PS_CTL1 of the MCU. A pull-down/up resistor is required if EN voltage is unspecified when EN is opened.
- PG of the PMIC is connected to EXT_PS_CTL0 of the MCU. A pull-up resistor is required if PG is open-drain.

Figure 13 shows basic power handover sequence of the V_{CCD} power rail with the connection.

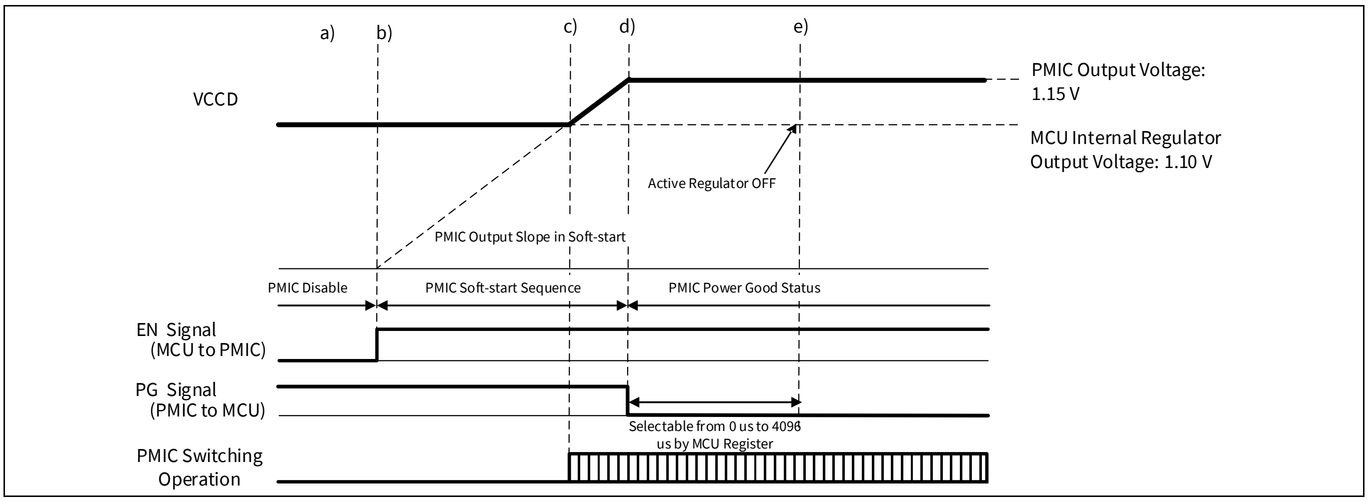


Figure 13 Basic V_{CCD} power handover sequence

a) The Active regulator as the internal regulator supplies power to the V_{CCD} power rail.

b) The MCU enables the PMIC. However, the PMIC does not start the switching operation because the target voltage of the PMIC output in the soft-start sequence is lower than the V_{CCD} voltage output by the Active regulator.

c) The PMIC target voltage of the soft-start sequence crosses the V_{CCD} voltage output by the Active regulator. At that time, the PMIC starts the switching operation and supplies power to the V_{CCD} power rail.

d) The VCCD voltage rises to the PMIC target voltage for the steady state and the PMIC soft-start sequence completes; then, the PMIC drops down the PG signal.

e) The MCU turns OFF the Active regulator when the MCU detects that the PG signal is in power good status.

4.1 Required PMIC specification

The MCU controls two power sources: Active regulator on the MCU and PMIC to supply V_{CCD} power. So, consider the following points while selecting the PMIC:

- The PMIC must provide correct and stable voltage to the MCU.
- If the PMIC has a discharge function, the PMIC sinks current during DeepSleep mode and increases the current consumption.
- Power handover may cause voltage drop and overcurrent due to the PMIC sinking the current from the MCU when the VCCD power source switches from the Active regulator to the PMIC.
- The MCU controls the PMIC enable with the PMIC EN terminal.
- The MCU can detect the PMIC voltage by using the power good function.

Feature of the selected PMIC determines whether a load switch for isolation between PMIC and V_{CCD} terminals is required. Table 16 shows PMIC feature requirement for power structures, PMIC direct connection and PMIC with load switch.

Table 16 PMIC requirement

Feature	**Mandatory feature for power structure**		**Comment**
	PMIC direct connection	PMIC with load switch	
Output voltage precision/Current ability	Yes	Yes	For voltage and current ability, see the “Recommended operating conditions” for V_{CCD} in the device datasheet.
No discharge function	Yes	N/A	The PMIC must not operate the discharge function when the PMIC is disabled to prevent unintended current consumption.
Pre-Bias soft-start	Yes	N/A	The PMIC must not sink from the Active regulator of the MCU.
Enable	Yes	Yes	Required to control the PMIC output
Power good	Yes	Yes	Required to indicate the PMIC output status

No discharge function

The discharge function discharges the output capacitance by connecting a resistor (for example, 100-Ω range) between V_{CCD} and ground, when the PMIC is disabled. So, when such a PMIC is disabled in DeepSleep mode, V_{CCD} would be discharged. Some PMICs discharge when under voltage lock out (UVLO) also. The PMIC must not have the discharge function because the discharge current might cause unintended current consumption.

Pre-bias startup and Forced 0 V soft-start

Soft-start is a function that gradually increases the output voltage to prevent the rush current when switching starts in a fixed, defined ramp-up time, independent of the applied voltage level at the starting point of ramp-up (pre-bias startup). There are two types of soft-start operations. Pre-bias soft-start does not sink current from the MCU and does not start the switching operation until the output voltage of the PMIC reaches the voltage level of the Active regulator. The forced 0-V soft-start forcibly starts the output from 0 V by sinking the current from the MCU through the V_{CCD} terminal. See the PMIC datasheet or contact the PMIC vendor to find out which soft-start is employed. Figure 14 shows the timing chart of each soft-start.

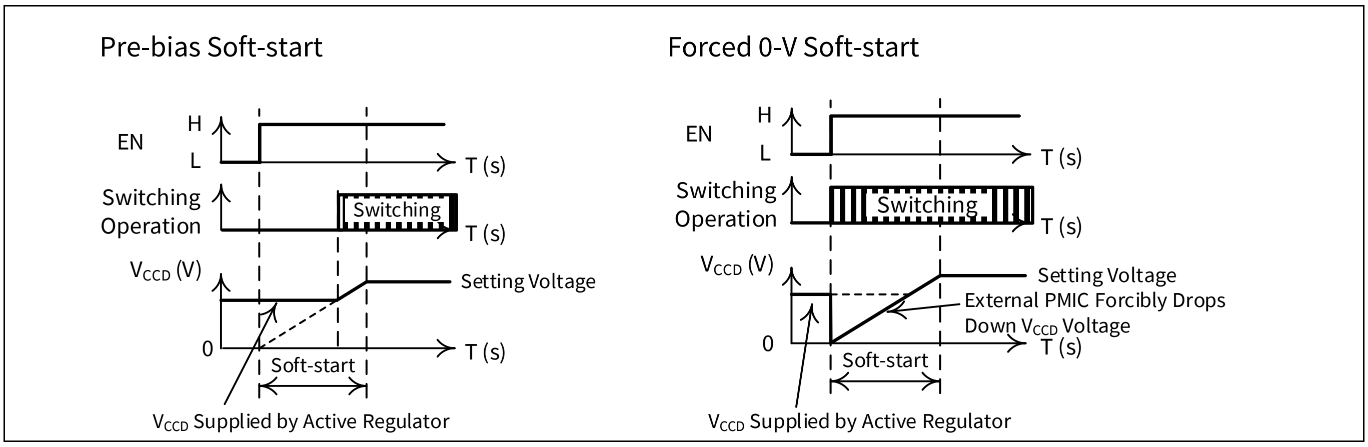


Figure 14 **Soft-start timing chart**

Note: *Certain PMICs activate overvoltage detection during soft-start. In this case, the PMIC detects overvoltage during soft-start; the PMIC does not start switching. When it reaches its tolerant range of overvoltage detection, the PMIC starts switching. A lower limit of this tolerant range results in the PMIC actively pulling down V_{CCD} that is supplied by the Active regulator. Such PMIC must be treated in the same manner as forced 0-V soft-start.*

Note: *It is strongly recommended to check the implementation of the pre-bias startup at the starting point. A minor drop might happen, but the voltage must be kept within the operating range of the internal Active regulator.*

Power good

PG should not indicate power good state during the time the PMIC is disabled. Furthermore, it is better that the function does not indicate power good status during the soft-start function or before reaching 100% of the target voltage, because the PMIC cannot supply power correctly at that time. Otherwise, the Active regulator may be turned OFF before the PMIC becomes fully operational. If PG indicates power good state during soft-start before reaching 100%, software with the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register can prevent early turn OFF of the Active regulator. For more details of PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register, see the “Software flow” of each power structure.

4.2 Recommended PMIC topology

Table 17 lists the recommended PMIC topology for each use case.

Table 17 PMIC requirement

Use case	Topology	Advantage
PMIC is disabled in DeepSleep mode Case 1, Case 2	Forced PWM mode	Small switching ripple voltage and small voltage fluctuation by the load transient response. Fixed switching frequency.
PMIC enabled in DeepSleep mode Case 3, Case 4	Auto PFM/PWM mode	Power saving for V_{CCD} supplied by the external PMIC in DeepSleep mode

Forced PWM mode vs. auto PWM/PFM mode

Forced PWM mode always operates with only PWM operation.

Auto PWM/PFM mode is an operation where the PMIC automatically switches between PWM operation and PFM operation depending on the load condition.

This mode works with the PWM operation when there is a heavy load current condition, and with the PFM operation when there is a light load current condition.

The PWM operation works with a constant frequency. It regulates the output voltage by changing the on-duty of the switching FET. The load transient response is better than PFM operation and its output ripple voltage is smaller than the PFM operation. Furthermore, EMI noise analysis is easier than PFM operation because of the fixed switching frequency.

PFM regulates the output voltage by changing the switching frequency according to the load current. The advantage of PFM operation is to improve the conversion efficiency in light load current conditions.

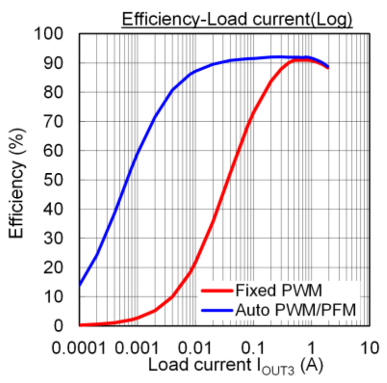


Figure 15 Conversion efficiency example

Note: PMIC does not sink current in PFM operation mode, because only the high-side switch is active. Therefore, the soft-start function behaves similar to pre-bias soft-start if the PMIC can operate the PFM mode during soft-start also. However, make sure that Auto PWM/PFM mode of the PMIC operates with PFM mode during soft-start.

Note: A PMIC in PFM mode has lower operation accuracy than in PWM mode.

4.3 PMIC direct connection

This section describes the PMIC connection, handover timing, and handover software flow example based on the following use cases depending on the internal regulator configuration for PMIC direct connection:

- Case 1: Use of OCD in Active mode with the PMIC disabled in DeepSleep power mode
- Case 2: No use of OCD in Active mode with the PMIC disabled in DeepSleep power mode
- Case 3: No use of OCD in Active mode with the PMIC enabled in DeepSleep power mode (using SwitchOverRegulators with blocking call)
- Case 4: No use of OCD in Active mode with the PMIC enabled in DeepSleep power mode (using SwitchOverRegulators with non-blocking call)

For details, see Internal regulator configuration using PMIC.

Note: *The use of OCD for overcurrent monitoring of the external power supply is not recommended, because the monitoring is not a direct shunt implementation of the VCCD power rail. It is an integral part of the Active regulator.*

Note: *The configuration of OCD usage refers only to the MCU operation with an external PMIC supply in Active mode. After POR, for example, the OCD feature is enabled, because it is part of the internal regulator.*

Note: *In Case 3, it is not possible to switch back to the internal regulator after handover to the PMIC. If you want to return to the internal regulator, you need to reset the entire chip, including the power system, by using the WDT or requesting an external system controller trigger XRES_L.*

4.3.1 Hardware configuration

Figure 16 shows the circuit diagram for PMIC direct connection.

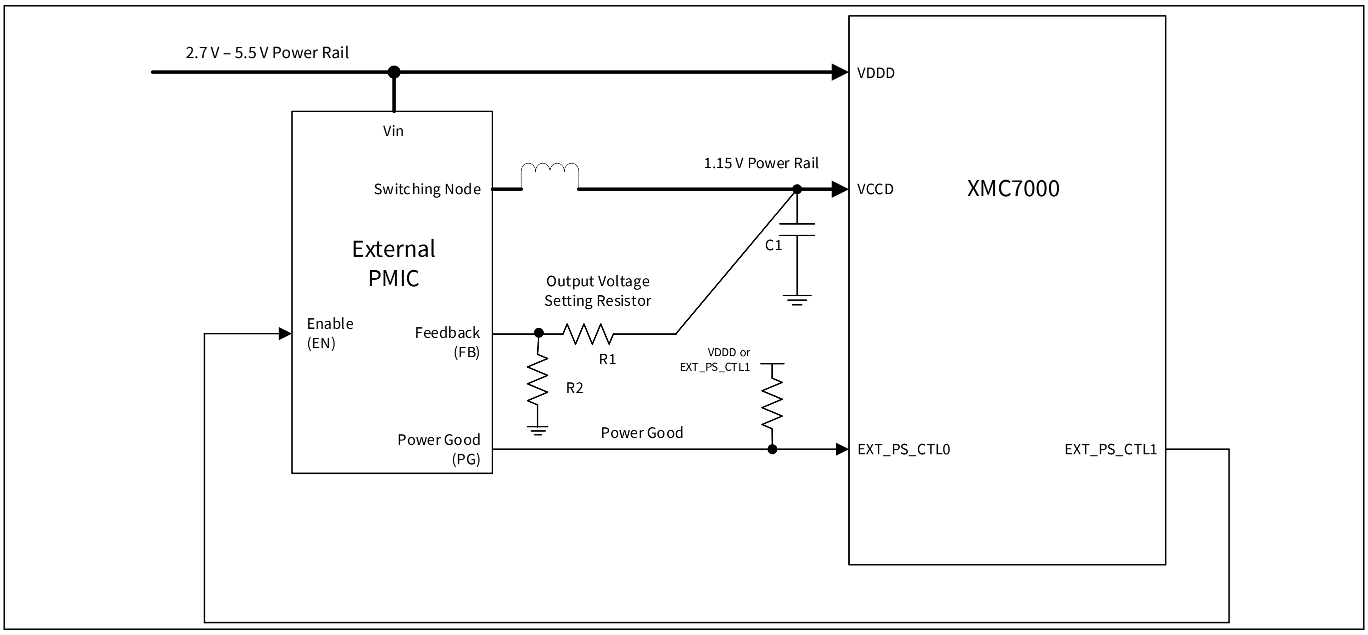


Figure 16 Circuit diagram for PMIC direct connection

- PMIC EN polarity is HIGH for enable. PMIC PG polarity is HIGH for power good

  RESET_PMIC is generated when PG of PMIC goes LOW. This is HV reset. When HV reset is applied, the MCU, including the REGHC controller and PMIC, is initialized.
- FB terminal of the PMIC is connected to $V_{CCD.}$
- Output voltage setting resistors are needed according to the selected PMIC.
- The EN terminal of the PMIC is connected to the EXT_PS_CTL1 terminal of MCU. A pull-down resistor on the PCB or in the PMIC is required to disable the PMIC during XRES and Hibernate.
- If the EN terminal does not have the internal pull-down resistor, an external pull-down resistor must be placed to keep the PMIC disabled during POR.
- The V_{CCD} capacitor (C1) depends on the MCU requirement (see the device datasheet for the capacitance).
- Output voltage setting resistors (R1 and R2) depend on the selected PMIC.

This section describes the software flow when the PMIC enable polarity is “1” and PG status polarity is “1”.

Enable polarity in the `ConfigureRegulator` API specifies the polarity of the enable signal to the PMIC and Reset polarity in the `ConfigureRegulator` API specifies the polarity of the PG state from the PMIC. Note that the Reset polarity bitfield corresponds the polarity of the deassertion level of the PG signal.

4.3.2 Case 1

This case explains the use of OCD in Active mode; the PMIC is disabled in DeepSleep power mode.

Note: The use of OCD for overcurrent monitoring of the external power supply is not recommended, because the monitoring is not a direct shunt implementation of the VCCD power rail. It is an integral part of the Active regulator.

4.3.2.1 Handover timing chart

In Case 1, the Active regulator is enabled (OCD is used), and DeepSleep regulator is enabled (DeepSleep regulator is enabled during DeepSleep power mode). Figure 17 shows an example of the handover sequence. This example includes the regulator configuration, handover from internal regulators to the PMIC, transition to DeepSleep power mode, wakeup from DeepSleep power mode, and handover from the PMIC to internal regulators.

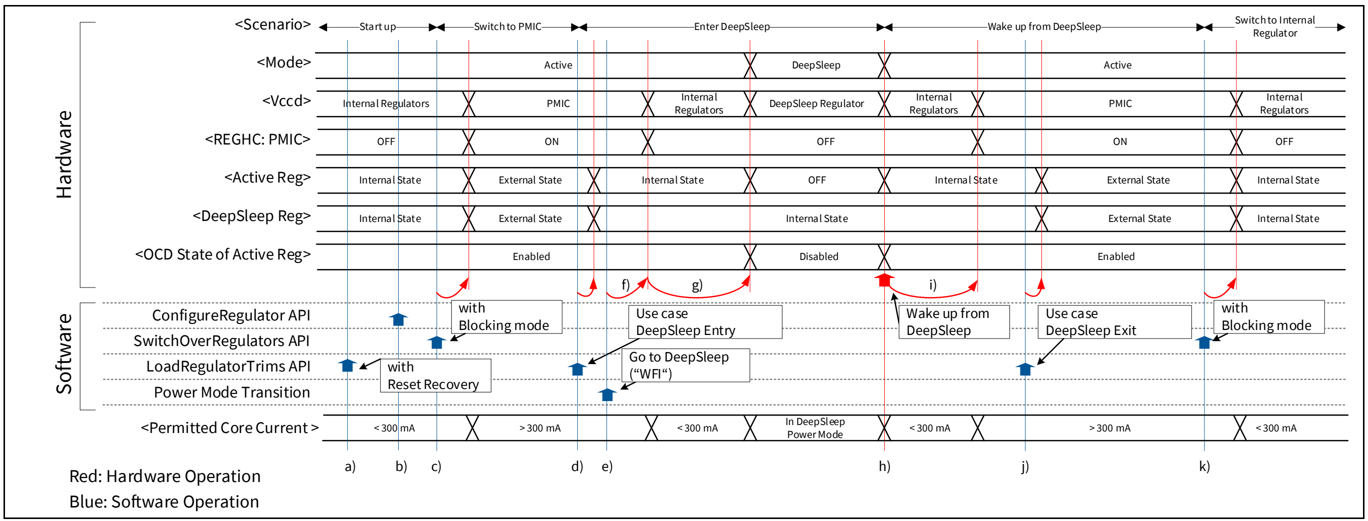


Figure 17 Handover sequence example (Case 1)

a) Call the `LoadRegulatorTrims` API with reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1". See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC controller with PMIC configuration after startup.

c) Call the `SwitchOverRegulators` API for the handover from internal regulators to the PMIC.

The PMIC is enabled. The Active regulator is still enabled due to the OCD feature, but changes to external state. The DeepSleep regulator changes to external state.

d) Call the `LoadRegulatorTrims` API with DeepSleep entry use case for changing the internal state of the regulator.

The Active and DeepSleep regulators change to internal state.

e) Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA. Then, you can execute the transition to DeepSleep power mode using the "WFI" instruction.

f) When the software executes the transition to DeepSleep power mode, the hardware switches from the PMIC to internal regulators.

g) The MCU transitions to DeepSleep power mode after the switching to internal regulators is complete. The Active regulator turns OFF.

h) A power mode transition from DeepSleep to Active happens after the wakeup event, and then the Active regulator starts with the internal state.

i) The PMIC is enabled after Active power mode transition is complete.

j) Call the `LoadRegulatorTrims` API with DeepSleep Exit use case for changing the internal regulators to external state.

The Active and DeepSleep regulators change to external state.

k) Call the `SwitchOverRegulators` API for the handover from the PMIC to internal regulators.

The PMIC is disabled. The Active and DeepSleep regulators change to internal state.

4.3.2.2 Software flow

Handover from the internal regulator to the PMIC

Figure 18 shows the handover flow from the internal regulator to the PMIC.

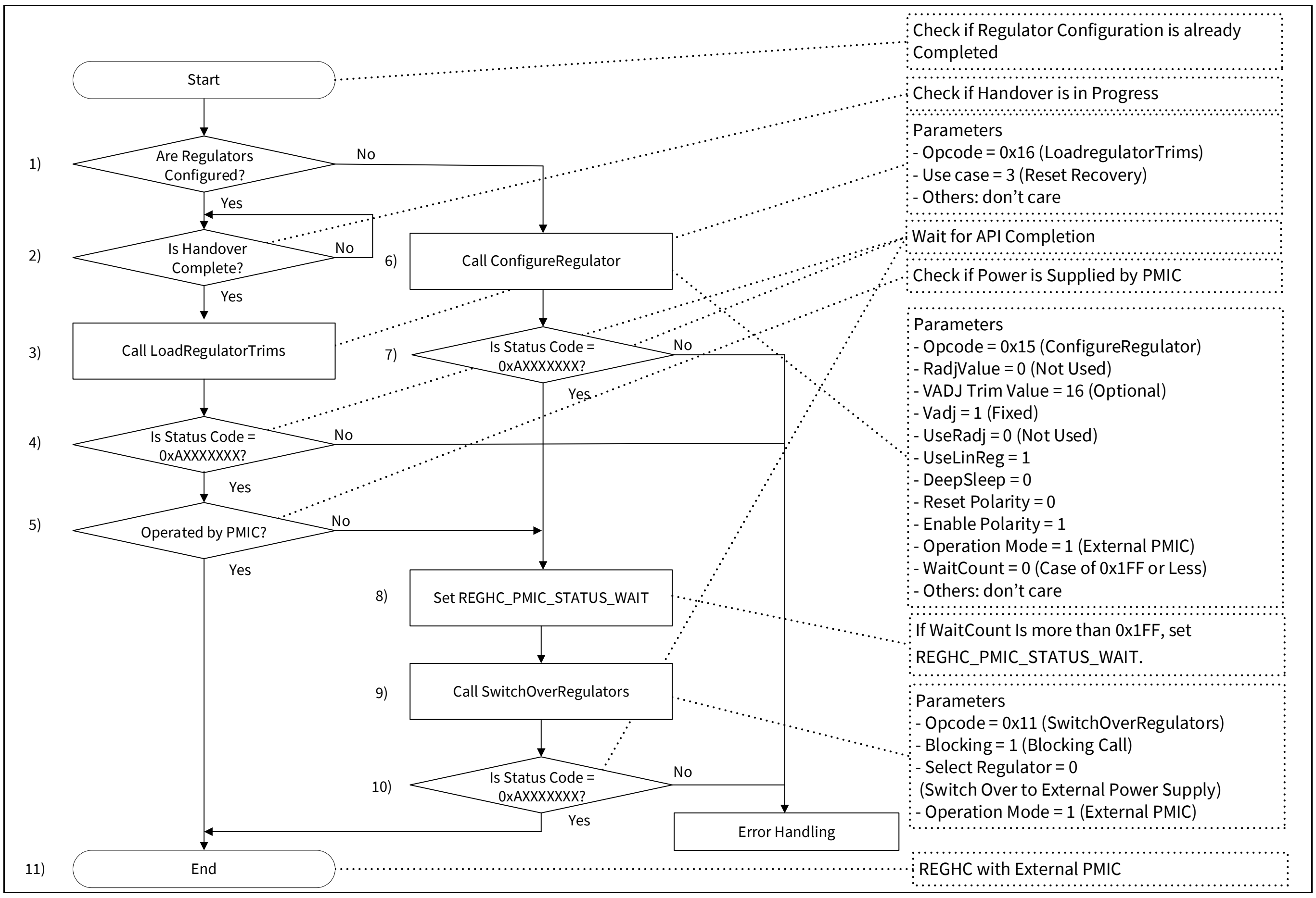

Start
1)
Are Regulators Configured?
No
Yes
2)
Is Handover Complete?
No
Yes
3)
Call LoadRegulatorTrims
4)
Is Status Code = 0xAXXXXXXX?
No
Yes
5)
Operated by PMIC?
No
Yes
6)
Call ConfigureRegulator
7)
Is Status Code = 0xAXXXXXXX?
No
Yes
8)
Set REGHC_PMIC_STATUS_WAIT
9)
Call SwitchOverRegulators
10)
Is Status Code = 0XAXXXXXXX?
No
Yes
Error Handling
11)
End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ Trim Value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 1
- DeepSleep = 0
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 0
(Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
REGHC with External PMIC

Figure 18 Handover flow from the internal regulator to the PMIC (Case 1)

1. Check if the regulator configuration is already completed. Once the REGHC controller is setup, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (6).

2. Wait for the handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset Recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with a PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. Go to (11). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (8).

6. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 1 (Internal Active regulator is kept as enabled after the PMIC is enabled)
   - DeepSleep = 0 (PMIC is disabled in DeepSleep power mode)
   - Reset Polarity = 0 (PMIC PG signal indicates power bad status with "0")
   - Enable Polarity = 1 (PMIC is enabled by "1")
   - Operating Mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)
7. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
8. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in the System Call API for more details.

To enable the REGHC controller with PMIC, perform these steps:

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over to external power supply)
   - Operating Mode = 1 (External PMIC) This parameter must match the Operating Mode in the `ConfigureRegulator` API
2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. The device is now operating with the PMIC. Active and DeepSleep regulators operate in external state.

DeepSleep entry and exit

Figure 19 shows transition flow to DeepSleep power mode.

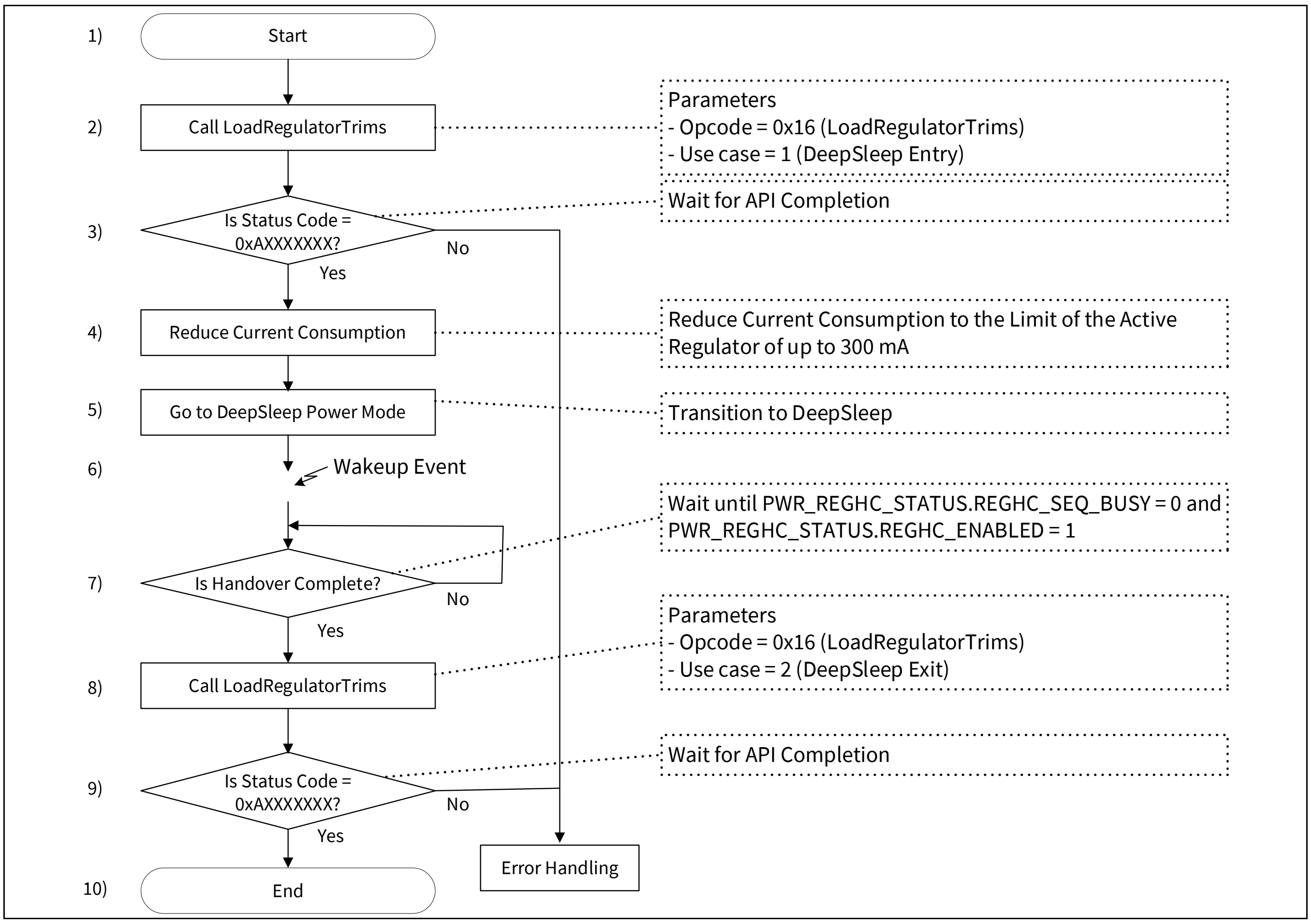

1)
Start
2)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 1 (DeepSleep Entry)
Wait for API Completion
3)
Is Status Code = 0xAXXXXXXX?
No
Yes
4)
Reduce Current Consumption
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
5)
Go to DeepSleep Power Mode
Transition to DeepSleep
6)
Wakeup Event
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
7)
Is Handover Complete?
No
Yes
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 2 (DeepSleep Exit)
8)
Call LoadRegulatorTrims
Wait for API Completion
9)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
10)
End

Figure 19 **DeepSleep entry and exit flow (Case 1)**

1. The MCU is powered by the PMIC.
2. Call the `LoadRegulatorTrims` API with the following parameters before transitioning to DeepSleep power mode:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 1 (DeepSleep Entry)
3. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. Reduce the current consumption to within the limit of Active Regulator, which is 300 mA.
5. Start the transition to DeepSleep power mode using the "WFI" instruction. Perform handover to internal regulator; and the PMIC is disabled. Then, the Active regulator is disabled, and DeepSleep regulator supplies power during DeepSleep power mode.

   The MCU is powered by the DeepSleep regulator.
6. Due to the wakeup event, the MCU transitions to Active power mode. Then, the hardware enables the PMIC.
7. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1.
8. Call the `LoadRegulatorTrims` API with the following parameters. Active and DeepSleep regulators are set to external state.
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 2 (DeepSleep exit)

9. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
10. The MCU is powered by the PMIC.

Handover from the PMIC to the internal regulator

Figure 20 shows the handover flow from the PMIC to the internal regulator.

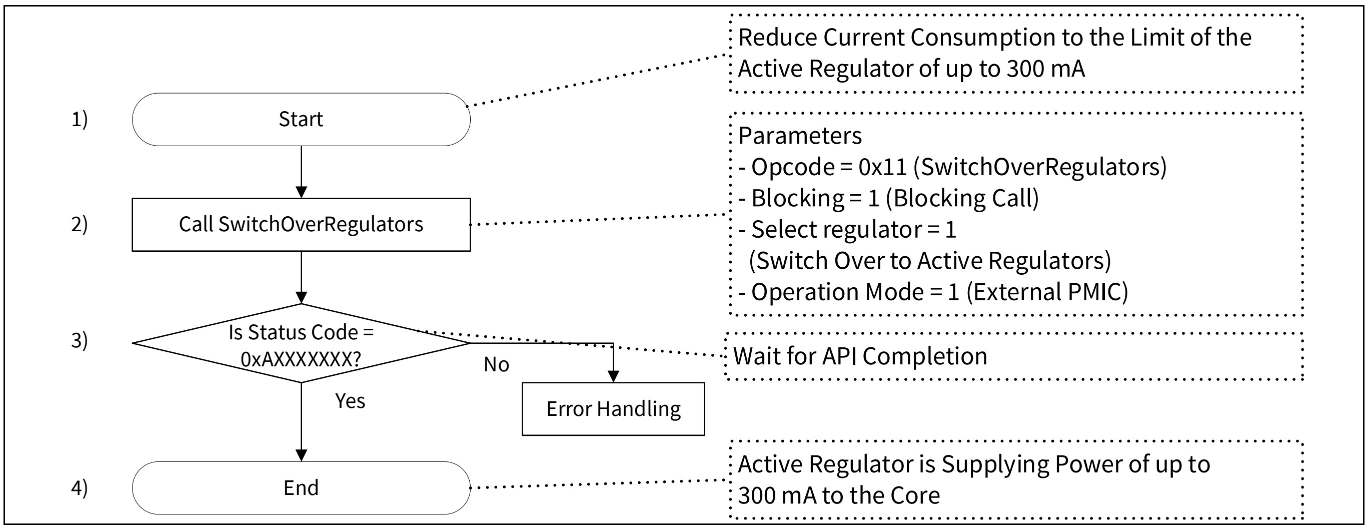

1)
Start
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
2)
Call SwitchOverRegulators
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select regulator = 1
(Switch Over to Active Regulators)
- Operation Mode = 1 (External PMIC)
3)
Is Status Code = 0xAXXXXXXX?
No
Wait for API Completion
Yes
Error Handling
4)
End
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 20 **Handover flow from the PMIC to the internal regulator (Case 1)**

To transition from the PMIC to the internal regulator:

1. Reduce the current consumption of the application to the maximum possible current of Active regulator, which is 300 mA (by configuring the appropriate application such as clock, cores, peripheral).
2. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 1 (Switch over to the Active regulator)
   - Operating mode = 1 (External PMIC)
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. The device is now operating with the internal regulator.

4.3.3 Case 2

In this case, OCD is not used in Active mode, the PMIC is disabled in DeepSleep power mode.

4.3.3.1 Handover timing chart

In Case 2, the Active regulator is disabled (OCD is not used) while the PMIC supplies the MCU, and the DeepSleep regulator is enabled in DeepSleep mode (that is PMIC is disabled). Figure 21 shows the handover sequence example. This example includes regulators configuration, handover from internal regulators to the PMIC, transition to DeepSleep power mode, and wake up from DeepSleep power mode.

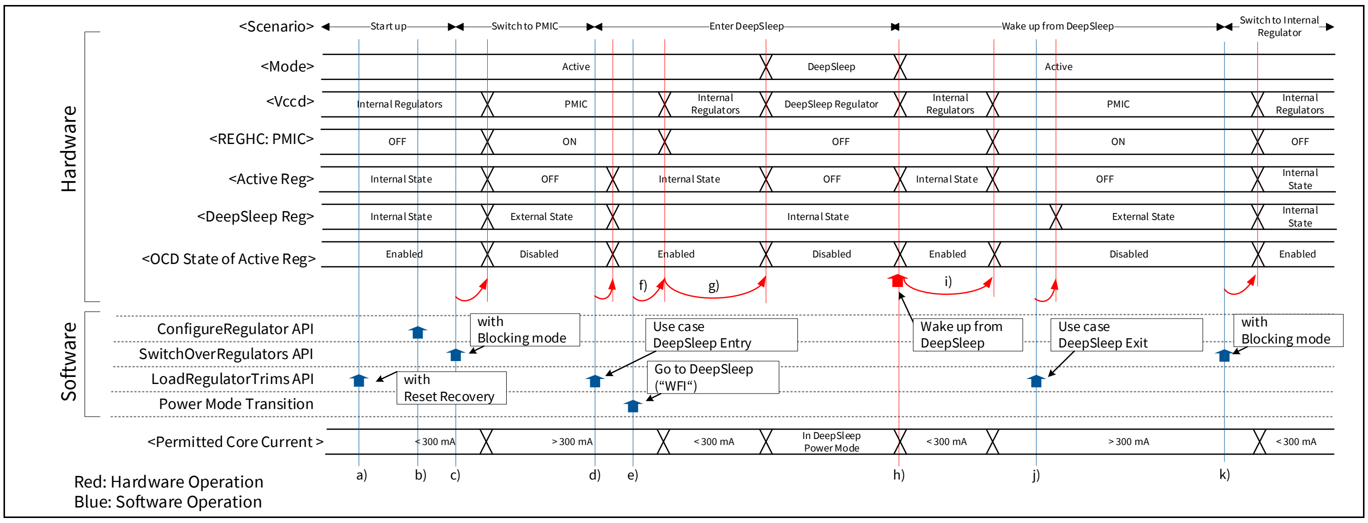


Figure 21 Handover sequence example (Case 2)

a) Call the `LoadRegulatorTrims` API with reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to “1”. See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC controller with the PMIC configuration after starting up.

c) Call the `SwitchOverRegulators` API for the handover from internal regulators to the PMIC.

The PMIC is enabled. The Active regulator is disabled because the OCD feature is not requested. The DeepSleep regulator changes to external state.

d) Call the `LoadRegulatorTrims` API with the DeepSleep entry use case for changing the internal state of the regulator.

The Active and DeepSleep regulators change to internal state before transitioning to DeepSleep power mode.

e) Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA. Then, execute the transition to DeepSleep power mode using the “WFI” instruction.

f) When the software executes the transition to DeepSleep power mode, the hardware switches from the PMIC to internal regulators.

g) The MCU transitions to DeepSleep power mode after the switching to internal regulators is complete. The Active regulator is disabled after transition to DeepSleep power mode.

h) A power mode transition from DeepSleep to Active happens after the wakeup event, and then the Active regulator starts with internal state.

i) After the Active power mode transition is complete, the PMIC is enabled and Active regulator is disabled.

j) Call the `LoadRegulatorTrims` API with DeepSleep exit use case for changing the external state of the regulator. The DeepSleep regulator changes to external state.

k) Call the `SwitchOverRegulators` API for the handover from the PMIC to internal regulators.

The PMIC is disabled. The Active and DeepSleep regulators change to internal state.

4.3.3.2 Software flow

Handover from the internal regulator to the PMIC

Figure 22 shows the handover flow from the internal regulator to the PMIC.

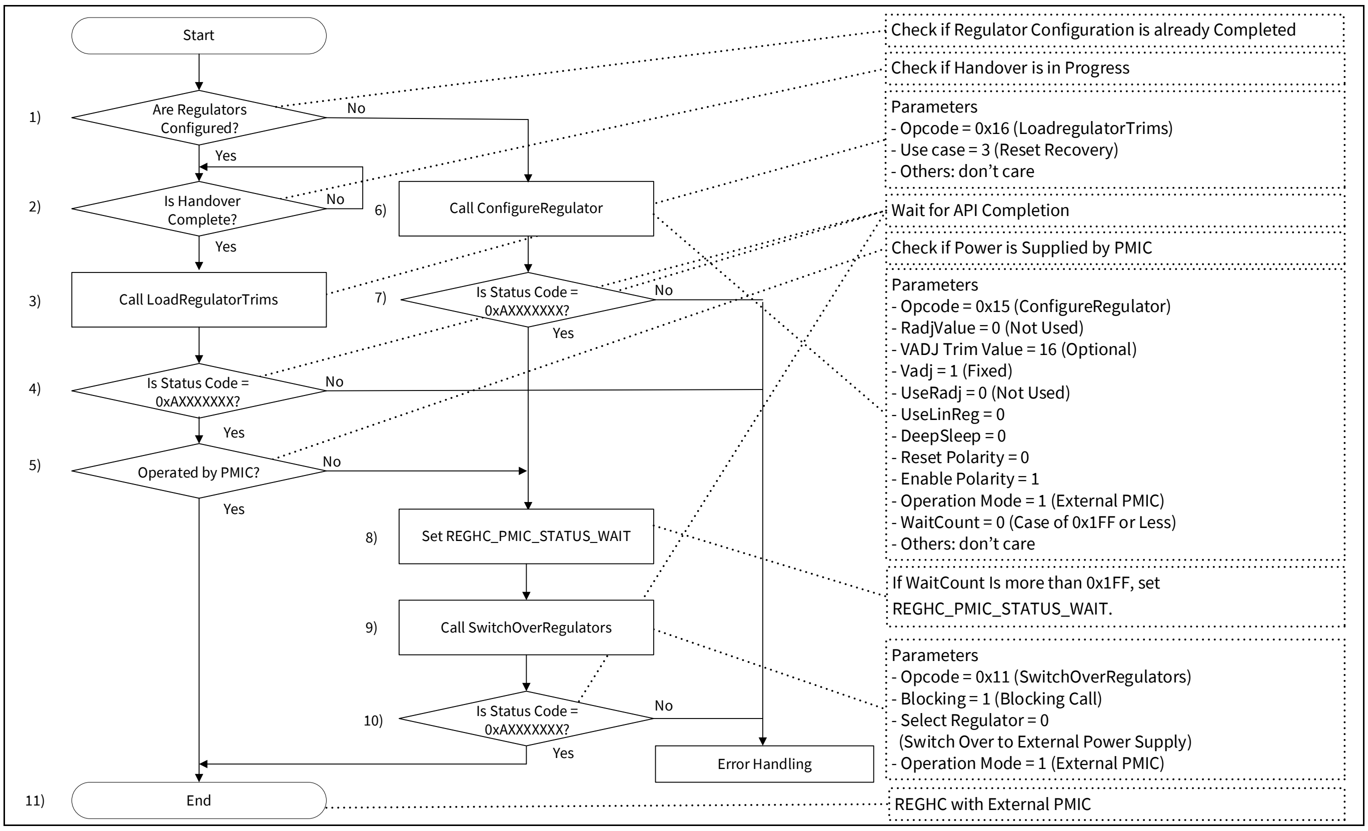

Start
1) Are Regulators Configured?
No
Yes
2) Is Handover Complete?
No
Yes
3) Call LoadRegulatorTrims
4) Is Status Code = 0xAXXXXXXX?
No
Yes
5) Operated by PMIC?
No
Yes
6) Call ConfigureRegulator
7) Is Status Code = 0xAXXXXXXX?
No
Yes
8) Set REGHC_PMIC_STATUS_WAIT
9) Call SwitchOverRegulators
10) Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
11) End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ Trim Value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 0
- DeepSleep = 0
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 0
(Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
REGHC with External PMIC

Figure 22 Handover flow from the internal regulator to the PMIC (Case 2)

1. Check if the regulator configuration is already completed. Once the REGHC controller is set up, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (6).

2. Wait for handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. Go to (11). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (8).

6. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj[1] = 0 (Not used)
   - UseLinReg = 0 (Internal Active regulator is disabled after the PMIC is enabled)
   - DeepSleep = 0 (PMIC is disabled in DeepSleep power mode)
   - Reset polarity = 0 (PMIC PG signal indicates power bad status with “0”)
   - Enable polarity = 1 (PMIC is enabled by “1”)
   - Operating mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)
7. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
8. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.

To enable the REGHC controller with PMIC, perform these steps:

9. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over to the external power supply)
   - Operating Mode = 1 (External PMIC) This parameter must match the operating mode in the `ConfigureRegulator` API
10. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
11. The device is now operating with the PMIC. Active and DeepSleep regulators operate in external state.

DeepSleep entry and exit

Figure 23 shows the transition to DeepSleep power mode.

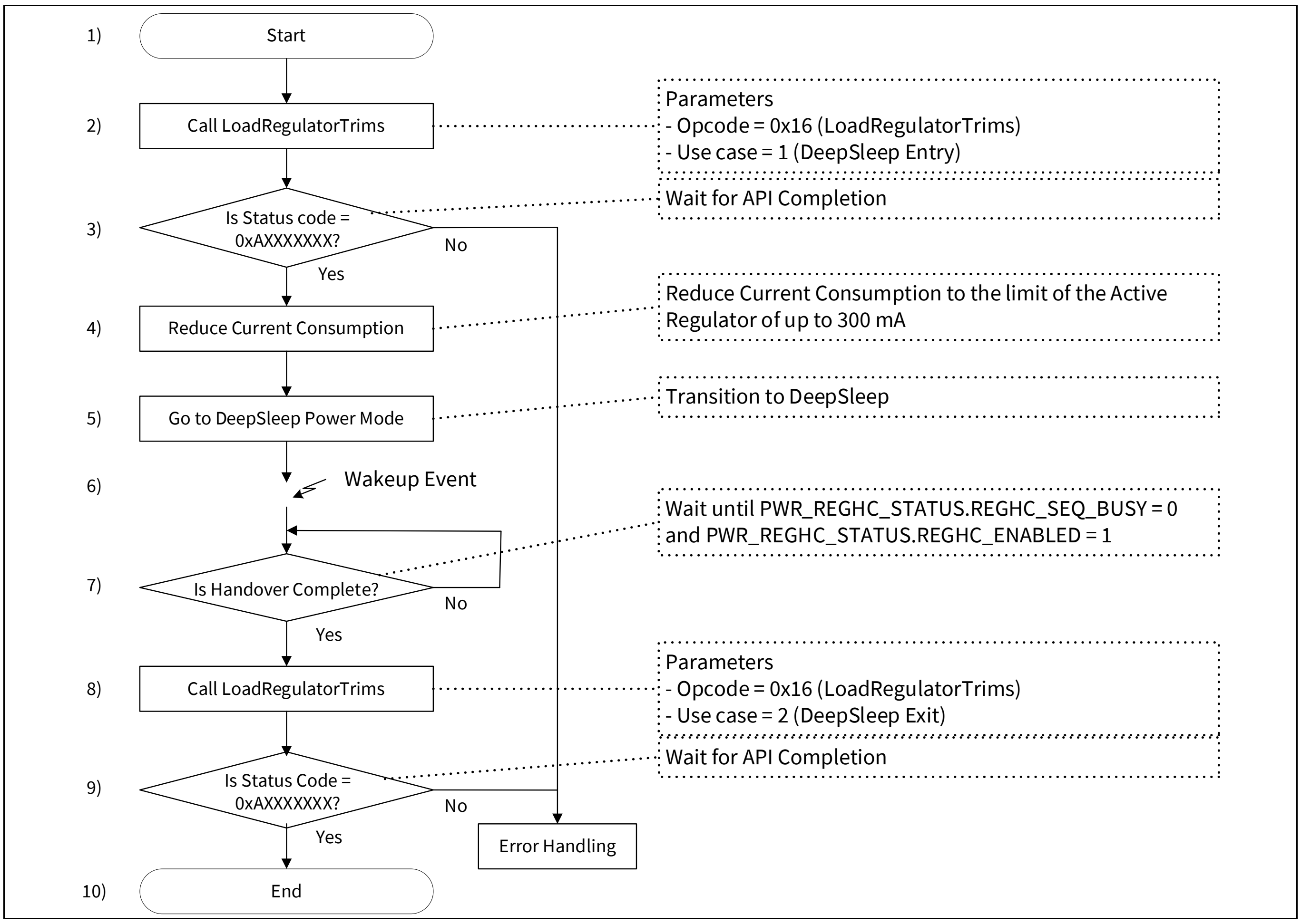

1)
Start
2)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 1 (DeepSleep Entry)
Wait for API Completion
3)
Is Status code = 0xAXXXXXXX?
No
Yes
4)
Reduce Current Consumption
Reduce Current Consumption to the limit of the Active Regulator of up to 300 mA
5)
Go to DeepSleep Power Mode
Transition to DeepSleep
6)
Wakeup Event
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
7)
Is Handover Complete?
No
Yes
8)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 2 (DeepSleep Exit)
Wait for API Completion
9)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
10)
End

Figure 23 DeepSleep entry and exit flow (Case 2)

1. The MCU is powered by the PMIC.
2. Call the `LoadRegulatorTrims` API with the following parameters before transitioning to DeepSleep power mode:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 1 (DeepSleep entry)
3. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA.
5. Start the transition to DeepSleep power mode using the "WFI" instruction. The MCU performs handover to the internal regulator, and the PMIC is disabled. Then, the Active regulator is disabled, and DeepSleep regulator supplies power during DeepSleep power mode.

   The MCU is powered by DeepSleep regulator.
6. Due to the wakeup event, the MCU transitions to Active power mode. Then, the hardware enables the PMIC.
7. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
8. Call the `LoadRegulatorTrims` API with the following parameters. Active and DeepSleep regulators are set to external state:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 2 (DeepSleep Exit)

9. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
10. The MCU is powered by the PMIC.

Handover from the PMIC to the internal regulator

Figure 24 shows the handover flow from the PMIC to the internal regulator.

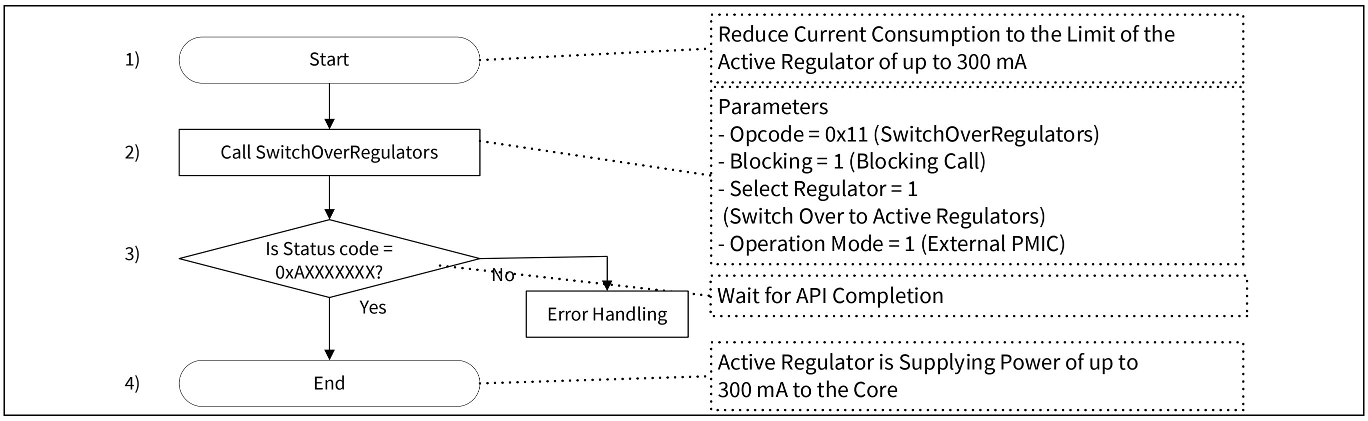

1)
Start
2)
Call SwitchOverRegulators
3)
Is Status code = 0xAXXXXXXX?
No
Yes
Error Handling
4)
End
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 1 (Switch Over to Active Regulators)
- Operation Mode = 1 (External PMIC)
Wait for API Completion
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 24 Handover flow from the PMIC to the internal regulator (Case 2)

To transition from the PMIC to the internal regulator:

1. Reduce the current consumption of the application to the maximum possible current of the Active regulator, which is 300 mA (by configuring the appropriate application such as clock, cores, peripheral).
2. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 1 (Switch over to Active regulator)
   - Operating Mode = 1 (External PMIC)
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. The device is now operating with the internal regulator.

4.3.4 Case 3

In this case, OCD is not used in Active mode, the PMIC is enabled in DeepSleep power mode. (Using SwitchOverRegulators as a blocking call)

4.3.4.1 Handover timing chart

In Case 3, the Active regulator is disabled (OCD is not used), and the DeepSleep regulator is disabled (the PMIC supplies power during DeepSleep power mode). Figure 25 shows an example of the handover sequence. This example includes the regulator configuration, handover from internal regulators to the PMIC, transition to DeepSleep power mode, and wakeup from DeepSleep power mode. In this configuration, the API cannot be used to switch back from the PMIC to the internal regulator. When operating with the internal regulator, you need to reset the entire chip (HV reset).

If your system requires to run the handover from the PMIC to internal regulators using the `SwitchOverRegulators` API, you need to call the `SwitchOverRegulators` API as a non-blocking call when running the handover to the PMIC. See Case 4.

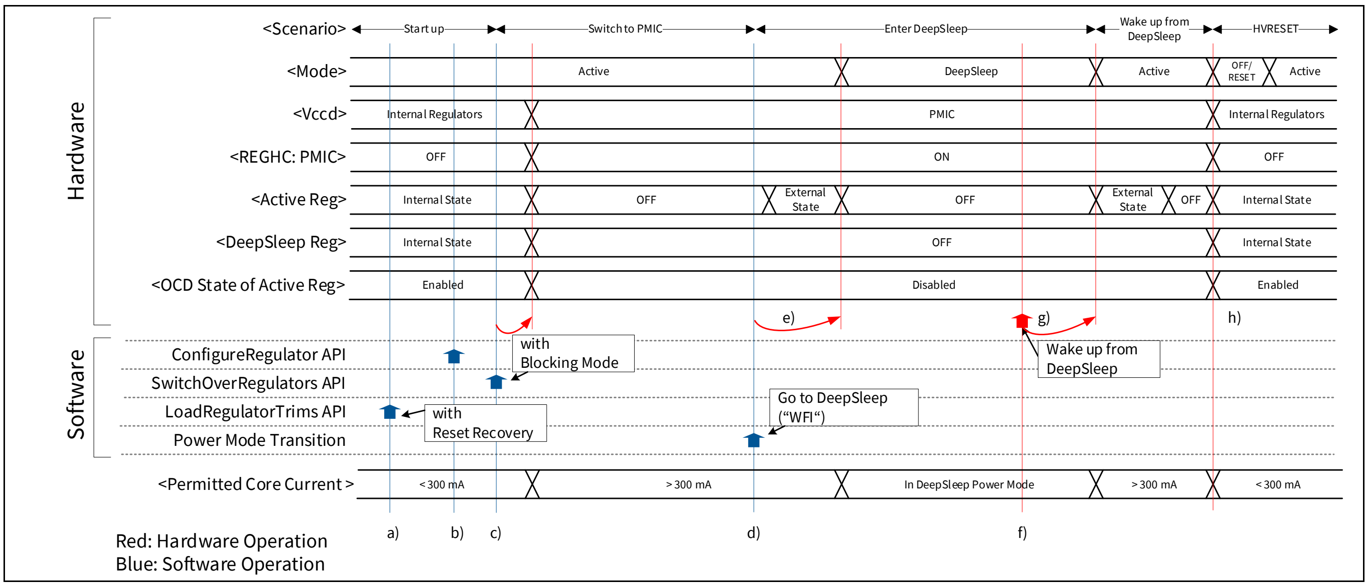


Figure 25 **Handover sequence example (Case 3)**

a) Call the `LoadRegulatorTrims` API with the reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1". See the reset recovery for details.
b) Call the `ConfigureRegulator` API for the REGHC controller with the PMIC configuration after starting up.
c) Call the `SwitchOverRegulators` API as a blocking call for the handover from the internal regulators to the PMIC.

Note: *When you call the `SwitchOverRegulators` API as a blocking call, the API sets the PWR_CTL2.DPSLP_REG_DIS to "1"; you cannot switch back from the PMIC to internal regulators.*

The PMIC is enabled. Active and DeepSleep regulators are disabled, because OCD is not used and the DeepSleep regulator is disabled in DeepSleep power mode.

d) Execute the transition to DeepSleep power mode using the "WFI" instruction. In this case, the system can transition to DeepSleep power mode directly without reducing the current consumption. The PMIC will continue to supply power.
e) The MCU transitions to DeepSleep power mode.

In this case, Active and DeepSleep regulators are disabled after calling the `SwitchOverRegulators` API. Power is always supplied from the PMIC. Therefore, the `LoadRegulatorTrims` API is unnecessary for the transition to DeepSleep power mode or Active power mode.

f) A power mode transition from DeepSleep to Active happens after the wakeup event. Active and DeepSleep regulators are disabled.
g) The MCU transitions to Active power mode; the PMIC will continue to supply power.
h) When an HV reset occurs, the power system is initialized. Therefore, the MCU starts up with the Active regulator after the reset is released.

4.3.4.2 Software flow

Handover from the internal regulator to the PMIC

Figure 26 shows the handover flow from the internal regulator to the PMIC.

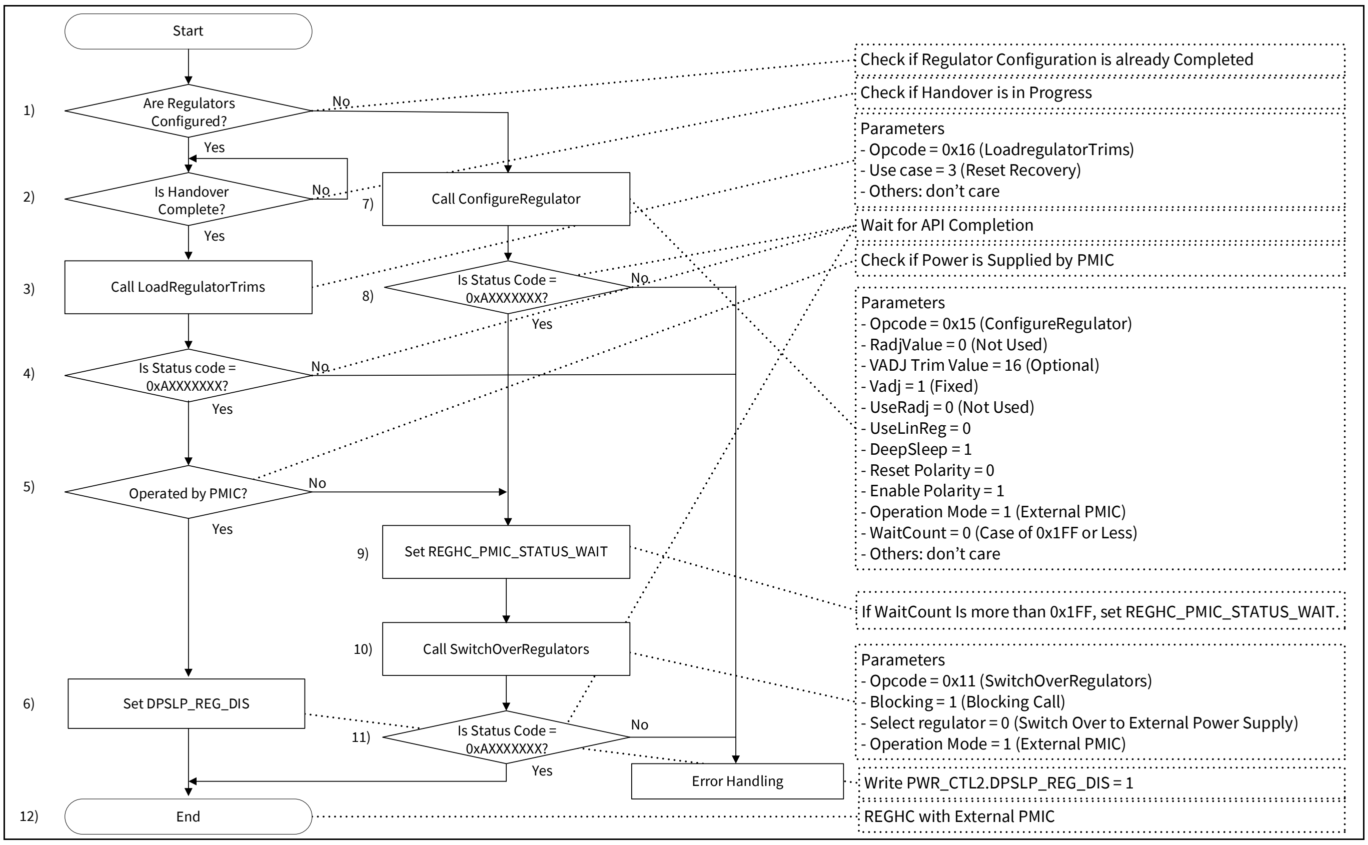

Start
1)
Are Regulators Configured?
No
Yes
2)
Is Handover Complete?
No
Yes
7)
Call ConfigureRegulator
3)
Call LoadRegulatorTrims
8)
Is Status Code = 0xAXXXXXXX?
No
Yes
4)
Is Status code = 0xAXXXXXXX?
No
Yes
5)
Operated by PMIC?
No
Yes
9)
Set REGHC_PMIC_STATUS_WAIT
10)
Call SwitchOverRegulators
6)
Set DPSLP_REG_DIS
11)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
12)
End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ Trim Value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 0
- DeepSleep = 1
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select regulator = 0 (Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
Write PWR_CTL2.DPSLP_REG_DIS = 1
REGHC with External PMIC

Figure 26 Handover flow from the internal regulator to the PMIC (Case 3)

1. Check if the regulator configuration is already completed. Once the REGHC controller is set up, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to “0”, the configuration is not complete. Go to (7).

2. Wait for the handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = “0” and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to “1”, the PMIC is supplying power. Go to (6). When PWR_REGHC_STATUS.REGHC_ENABLED is set to “0”, the internal regulator is supplying power. Go to (9).
6. Set PWR_CTL2.DPSLP_REG_DIS to “1”

In Case 3, the PMIC is enabled in DeepSleep power mode. Therefore, if the handover to the PMIC has already been completed, the user software needs to set PWR_CTL2.DPSLP_REG_DIS to "1". If not, PWR_CTL2.DPSLP_REG_DIS is set by the `SwitchOverRegulators` API in blocking mode.

7. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 0 (Internal Active regulator is disabled after the PMIC is enabled)
   - DeepSleep = 1 (PMIC is enabled in DeepSleep power mode)
   - Reset polarity = 0 (PMIC PG signal indicates power bad status with “0”)
   - Enable polarity = 1 (PMIC is enabled by “1”)
   - Operating mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)
8. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
9. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.

To enable the REGHC controller with the PMIC, perform these steps:

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over to external power supply)
   - Operating Mode = 1 (External PMIC). This parameter must match the Operating mode in the `ConfigureRegulator` API.

Note: *In this case, the `SwitchOverRegulators` API sets PWR_CTL2.DPSLP_REG_DIS to “1”; you cannot switch back from the PMIC to internal regulators. If your system requires to run the handover from the PMIC to internal regulators by the `SwitchOverRegulators` API, you need to call `SwitchOverRegulators` as a non-blocking call when running the handover to the PMIC. See SwitchOverRegulators API non-blocking call handling for details.*

2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. The device is now operating with the PMIC. Active and DeepSleep regulators are disabled.

DeepSleep entry and exit

Figure 27 shows the transition to DeepSleep power mode. In this case, you can transition to DeepSleep power mode without any additional action.

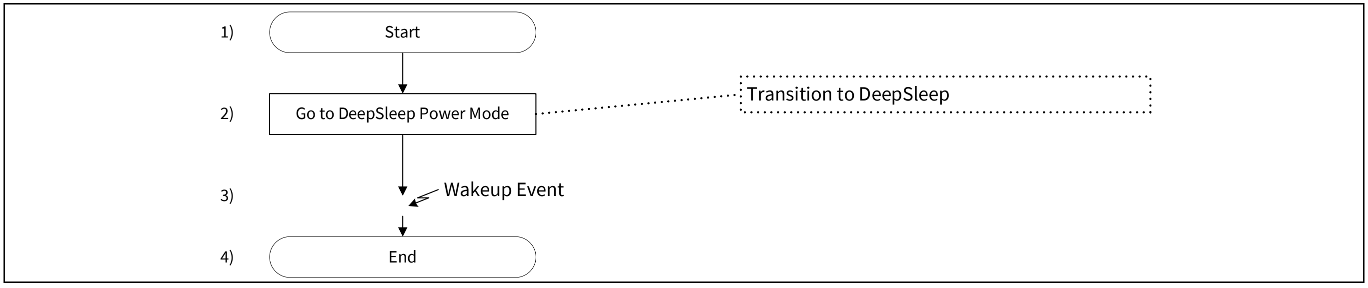


Figure 27 DeepSleep entry and exit flow (Case 3)

1. The MCU is powered by the PMIC.
2. Start the transition to DeepSleep power mode using the “WFI” instruction. The Active regulator remains disabled, and the PMIC continues to supply power.

The MCU is powered by the PMIC.

3. Due to the wakeup event, the MCU transitions to Active power mode. Active and DeepSleep regulators remain disabled.
4. The MCU is powered by the PMIC.

4.3.5 Case 4

In this case, OCD is not used in Active mode; the PMIC is enabled in DeepSleep power mode (using `SwitchOverRegulators` as a non-blocking call).

4.3.5.1 Handover timing chart

In Case 4, the Active regulator is disabled (OCD is not used), and the DeepSleep regulator is enabled; the PMIC supplies power during DeepSleep power mode. Figure 28 shows an example of the handover sequence. This example includes the regulator configuration, handover from internal regulators to the PMIC, and handover from the PMIC to internal regulators.

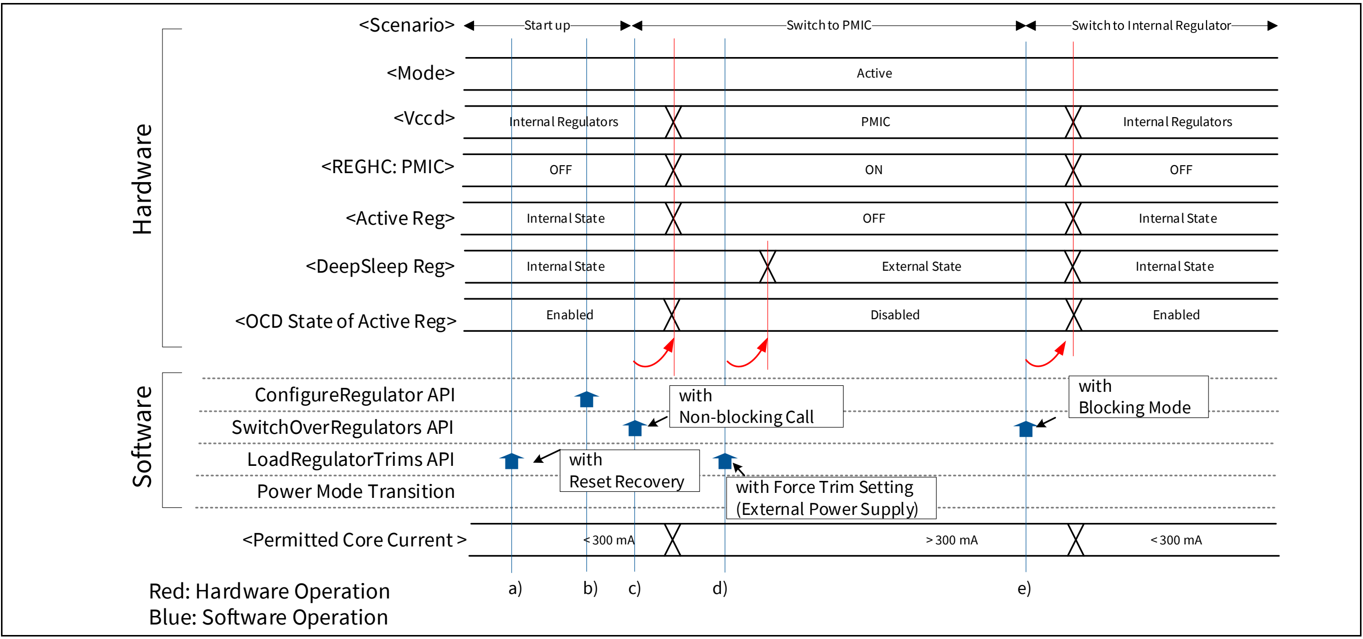


Figure 28 Handover sequence example (Case 4)

a) Call the `LoadRegulatorTrims` API with the reset recovery use case if PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1". See the reset recovery for details.
b) Call the `ConfigureRegulator` API for the REGHC controller with the PMIC configuration after startup.
c) Call the `SwitchOverRegulators` API as a non-blocking call for the handover from internal regulators to the PMIC.

Note: When calling the `SwitchOverRegulators` API as a non-blocking call, the API does not set PWR_CTL2.DPSLP_REG_DIS to "1". Therefore, you can switch back to internal regulators from the PMIC using the `SwitchOverRegulators` API. However, if the application software sets PWR_CTL2.DPSLP_REG_DIS to "1", you cannot switch back to internal regulators from the PMIC. See SwitchOverRegulators API non-blocking call handling for details.

The PMIC is enabled. The Active regulator is disabled because the OCD feature is not requested.

d) Call the `LoadRegulatorTrims` API with the Force trim setting use case for changing the external state of the regulator. The DeepSleep regulator changes to external state.
e) Call the `SwitchOverRegulators` API for the handover from the PMIC to internal regulators. The PMIC is disabled. Active and DeepSleep regulators change to internal state.

4.3.5.2 Software flow

Handover from the internal regulator to the PMIC.

Figure 29 shows the handover flow from the internal regulator to the PMIC.

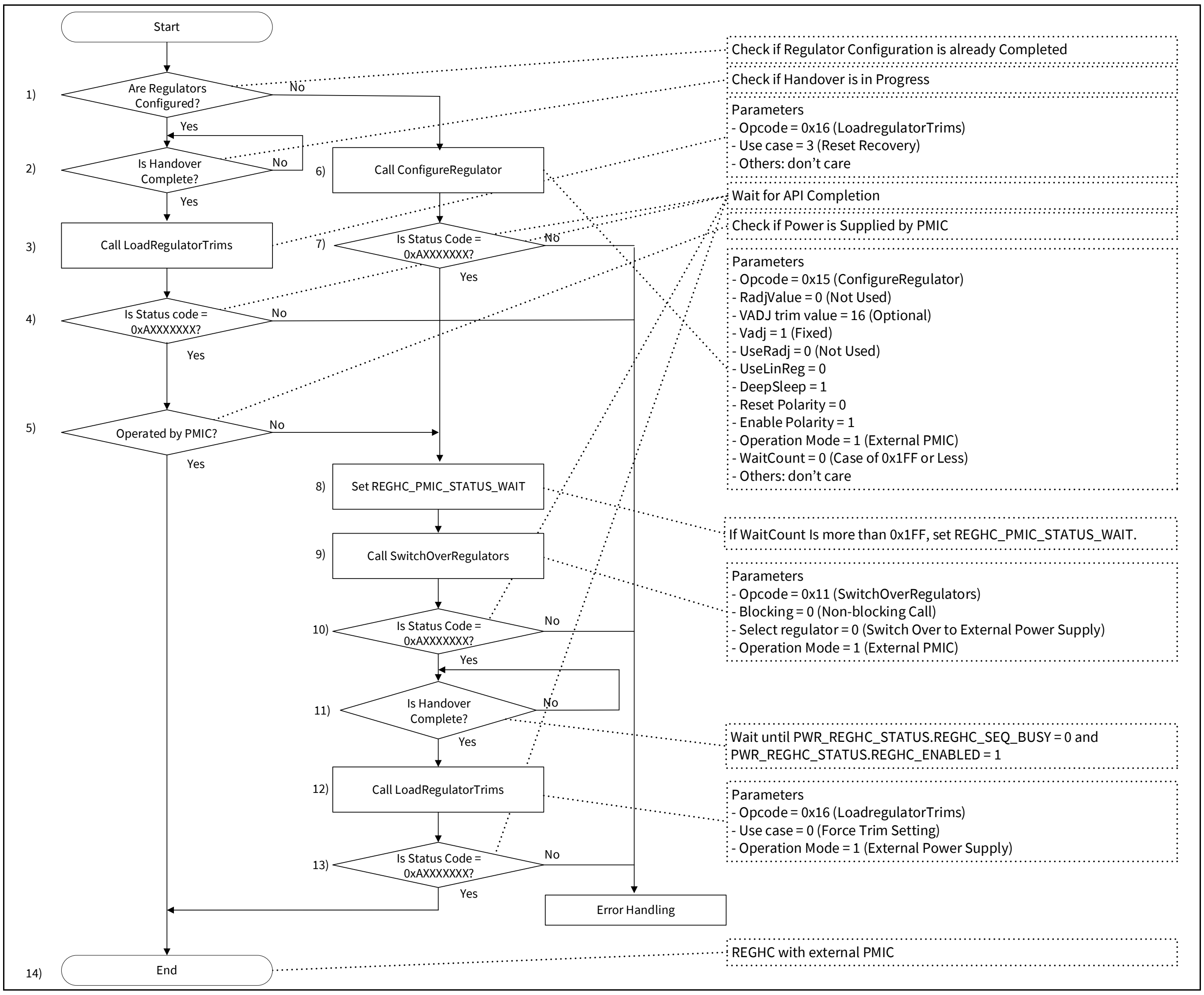

Start
1) Are Regulators Configured?
No
Yes
2) Is Handover Complete?
No
Yes
3) Call LoadRegulatorTrims
4) Is Status code = 0xAXXXXXXX?
No
Yes
5) Operated by PMIC?
No
Yes
6) Call ConfigureRegulator
7) Is Status Code = 0xAXXXXXXX?
No
Yes
8) Set REGHC_PMIC_STATUS_WAIT
9) Call SwitchOverRegulators
10) Is Status Code = 0xAXXXXXXX?
No
Yes
11) Is Handover Complete?
No
Yes
12) Call LoadRegulatorTrims
13) Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
14) End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ trim value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 0
- DeepSleep = 1
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 0 (Non-blocking Call)
- Select regulator = 0 (Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 0 (Force Trim Setting)
- Operation Mode = 1 (External Power Supply)
REGHC with external PMIC

Figure 29 Handover flow from the internal regulator to the PMIC (Case 4)

1. Check if the regulator configuration is already completed. Once the REGHC controller is set up, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (6).

2. Wait for the handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. Go to (14). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (8).

6. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 0 (Internal Active regulator is disabled after the PMIC is enabled)
   - DeepSleep = 1 (PMIC is enabled in DeepSleep power mode)
   - Reset polarity = 0 (PMIC PG signal indicates power bad status with “0”)
   - Enable polarity = 1 (PMIC is enabled by “1”)
   - Operating mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)
7. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
8. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.

To enable the REGHC controller with the PMIC, perform these steps:

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 0 (Non-blocking call)
   - Select regulator = 0 (Switch over to the external power supply)
   - Operating Mode = 1 (External PMIC) This parameter must match the Operating mode in the `ConfigureRegulator` API.

Note: *The `ConfigureRegulator` API is configured UseLinReg = 0 and DeepSleep =1, but the `SwitchOverRegulators` runs as a non-blocking call. Therefore, the `SwitchOverRegulators` API does not set PWR_CTL2.DPSLP_REG_DIS to "1".*

2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1.
4. Call the `LoadRegulatorTrims` API with force trim setting:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 0 (Force trim setting)
   - Operating mode = 1 (External power supply)

This process changes the internal regulator output state to external state.

5. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
6. The device is now operating with the PMIC. The Active regulator is in OFF state; the DeepSleep regulator operates in external state.

Handover from the PMIC to the internal regulator

Figure 30 shows the handover flow from the PMIC to the internal regulator.

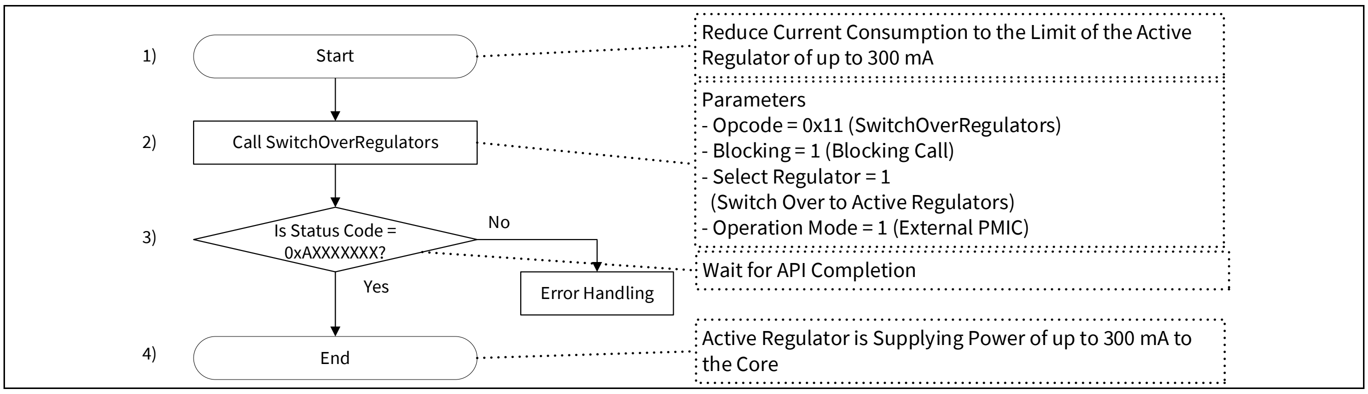

1)
Start
2)
Call SwitchOverRegulators
3)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
4)
End
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 1 (Switch Over to Active Regulators)
- Operation Mode = 1 (External PMIC)
Wait for API Completion
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 30 Handover flow from the PMIC to the internal regulator (Case 4)

To transition from the PMIC to the internal regulator:

1. Reduce the current consumption of the application to the maximum possible current of the Active regulator, which is 300 mA (by configuring the appropriate application such as clock, cores, peripheral).
2. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 1 (Switch over to the Active regulator)
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. The device is now operating with the internal regulator.

4.4 PMIC with load switch

This section considers the deployment of a load switch for the core supply rail, when decoupling between the PMIC output and MCU V_{CCD} is required. See Required PMIC specification for the conditions.

In general, variants with a load switch are not recommended. The reasons are the circuit complexity to ensure the low voltage drop by the drain-source resistance of the load switch under all operating conditions, and the risk of sink current by the PMIC when disabling the PMIC.

This section describes the PMIC connection, handover timing, and handover software flow example based on the following use cases depending on the internal regulator configuration for the PMIC with a load switch.

- Case 1: The Load switch gate is controlled by the power good signal of the PMIC. OCD is used in Active mode, and PMIC is disabled in DeepSleep power mode.
- Case 2: The Load switch gate is controlled by the REGHC (EXT_PS_CTL1) pin. OCD is not used in Active mode, and PMIC is enabled in DeepSleep power mode (using SwitchOverRegulators as a blocking call)
- Case 3: Load switch gate is controlled by the REGHC (EXT_PS_CTL1) pin. OCD in used in Active mode, and PMIC is enabled in DeepSleep power mode (using SwitchOverRegulators as a non-blocking call)

4.4.1 Case 1

This section describes the solution using a load switch. A load switch is required for isolating the PMIC from V_{CCD} depending on the choice of PMIC topology. In this solution, the PMIC output must be sufficiently stable when PMIC indicates a PG status.

In this case, OCD is used in Active mode; the PMIC is disabled in DeepSleep power mode. The load switch gate is controlled by the power good signal of the PMIC.

4.4.1.1 Hardware configuration

Figure 31 shows the circuit diagram for the PMIC with load switch 1. In this configuration, the load switch gate is connected to the PG signal, which means that the load switch is controlled directly by the PG signal.

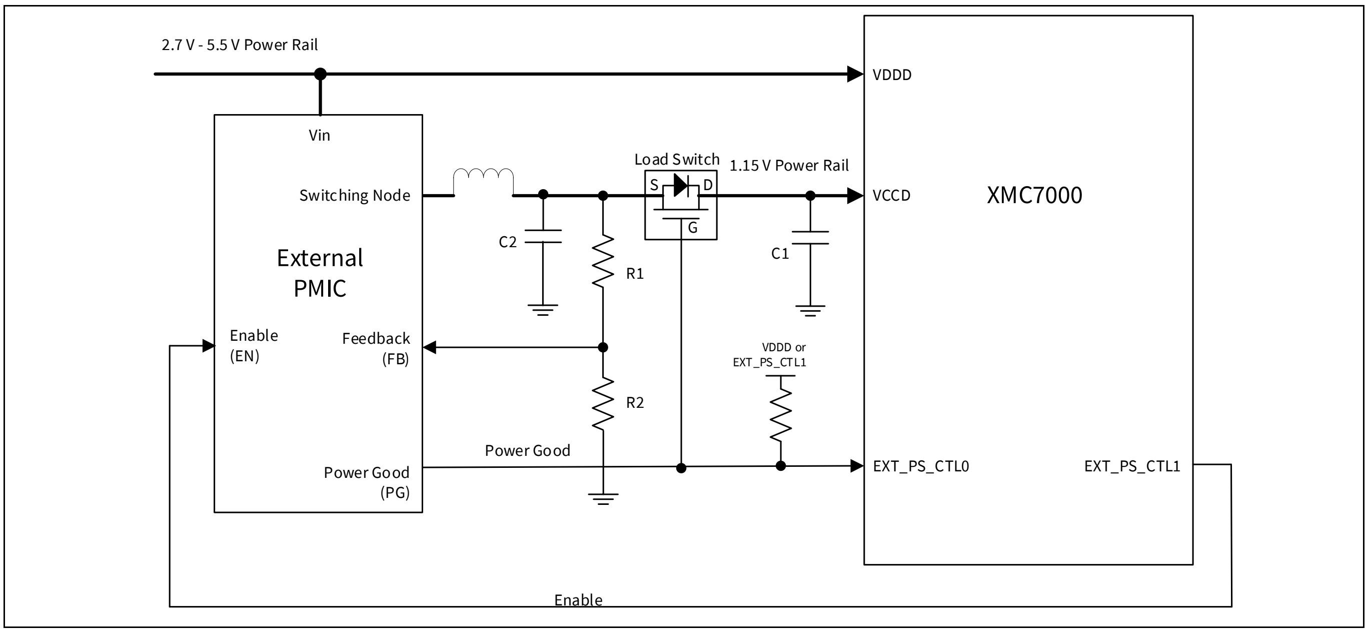


Figure 31 Circuit diagram for PMIC with load switch 1

- The PMIC EN polarity is HIGH for enable. The PMIC PG polarity is HIGH for PG.

  RESET_PMIC is generated when PG of the PMIC goes LOW, which causes a HV reset in the MCU.
- The PMIC feedback signal is connected to the load switch on the PMIC side.
- The Load switch is connected between the PMIC output and V_{CCD} of the MCU.
- The Gate signal of load switch and PG pin of the PMIC are connected to the EXT_PS_CTL0 pin of the MCU. PG pulls up to the power rail (V_{DDD}), however:

  When the Vin voltage for the PMIC starts to ramp up, the PG terminal voltage may rise before the PMIC pulls down PG because of incomplete Vin voltage and the High-Z status of the PMIC PG terminal. As a result, the load switch may turn ON at an unintended timing. One of the ways to prevent the unintended turning ON is to connect the pull-up resistor of EXT_PS_CTL0 (PMIC Power Good) to EXT_PS_CTL1 (PMIC EN) instead of V_{DDD}.
- The EN pin of the PMIC is connected to the EXT_PS_CTL1 pin of the MCU. A pull-down resistor on the PCB or in the PMIC is required to disable the PMIC during XRES and Hibernate.

- If the EN pin does not have the internal pull-down resistor, an external pull-down resistor must be placed to keep the PMIC disabled during POR.
- V_{CCD} capacitor (C1) depends on the MCU requirement (see the device datasheet for the capacitance).
- Output voltage setting resistors (R1, R2) and output capacitor (C2) depend on the selected PMIC.

The subsequent software flow assumes that the register setting of the PMIC enable polarity is “1” and the PG status polarity is “1”.

The Enable polarity in the `ConfigureRegulator` API specifies the polarity of the enable signal to the PMIC, and the Reset polarity in the `ConfigureRegulator` API specifies the polarity of the power good state from the PMIC. Note that the Reset polarity is set to the PG deasserted polarity. These settings depend on the PMIC specifications.

In this use case, the internal regulator configuration uses OCD; the PMIC is disabled in DeepSleep power mode.

4.4.1.2 Handover timing chart

Figure 32 shows an example of the handover sequence with load switch 1. This example includes the regulator configuration, handover from internal regulators to the PMIC, transition to DeepSleep power mode, wakeup from Deep Sleep power mode, and handover from the PMIC to internal regulators. This section assumes the register setting of the PMIC enable polarity is “1” and PG status polarity is “1”.

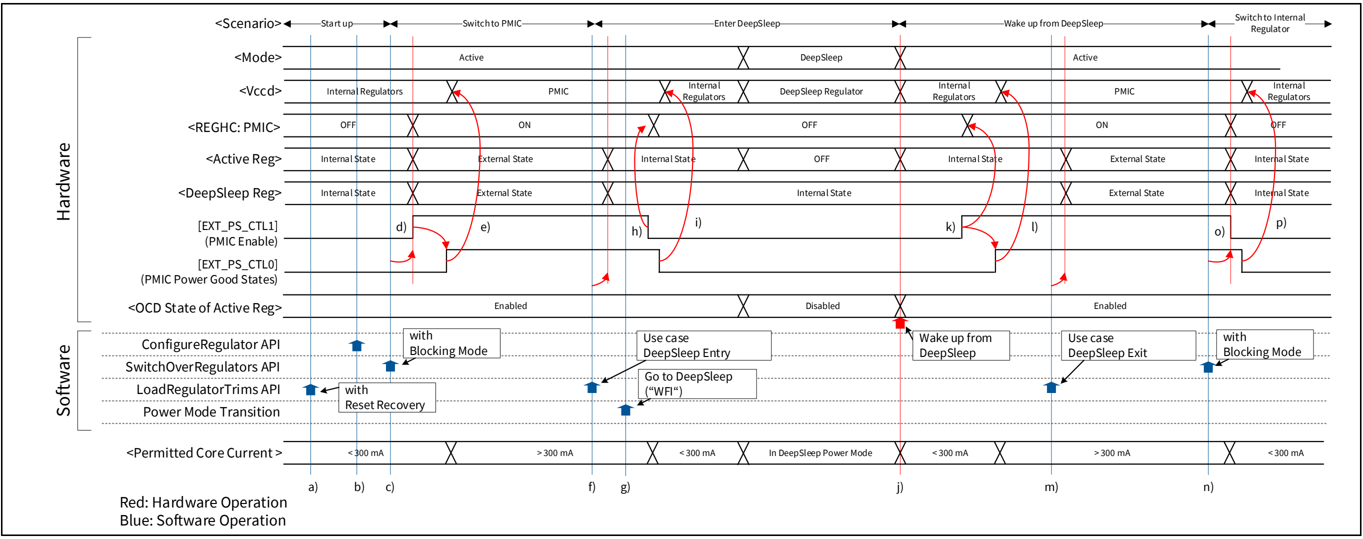


Figure 32 Handover sequence example of the PMIC with load switch 1

a) Call the `LoadRegulatorTrims` API with the reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to “1”. See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC with the PMIC configuration after startup.

c) Call the `SwitchOverRegulators` API for handover to the PMIC from internal regulators.

The Active regulator is enabled to use OCD, and changed to external state. The DeepSleep regulator changes to external state. EXT_PS_CTL1 outputs the PMIC enable signal by the `SwitchOverRegulators` API, and the PMIC gets enabled.

d) EXT_PS_CTL0 is in power good state after the PMIC is ready.

e) The load switch turns ON when EXT_PS_CTL0 is in power good state. The PMIC supplies power to the MCU.

f) Call the `LoadRegulatorTrims` API with the DeepSleep entry use case for changing the state of the internal regulators. The Active and DeepSleep regulators change to internal state.

g) Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA. Then, you can execute the transition to DeepSleep power mode using the "WFI" instruction.

h) Transition to DeepSleep triggers the handover to internal regulators. EXT_PS_CTL1 outputs the PMIC disable signal, and EXT_PS_CTL0 is not in power good state.

i) The load switch is OFF when EXT_PS_CTL0 is not in power good state. Internal regulators supply power to the MCU.

j) A power mode transition from DeepSleep to Active happens after the wakeup event, and then the Active regulator starts with the internal state.

k) EXT_PS_CTL1 outputs the PMIC enable signal, and the PMIC is enabled. EXT_PS_CTL0 outputs the PG state signal after the PMIC is ready.

l) The load switch is ON when EXT_PS_CTL0 is in power good state. The PMIC supplies power to the MCU.

m) Call the `LoadRegulatorTrims` API with the DeepSleep Exit use case for changing the external state of the regulator.

n) Call the `SwitchOverRegulators` API for the handover from the PMIC to internal regulators.

Internal regulators change to internal state.

o) EXT_PS_CTL1 outputs the PMIC disable signal, and EXT_PS_CTL0 is not power good state.

p) The load switch is OFF when EXT_PS_CTL0 is not in power good state. Internal regulators supply power to the MCU.

4.4.1.3 Software flow

Handover from the internal regulator to the PMIC

Figure 33 shows the handover from the internal regulator to the PMIC.

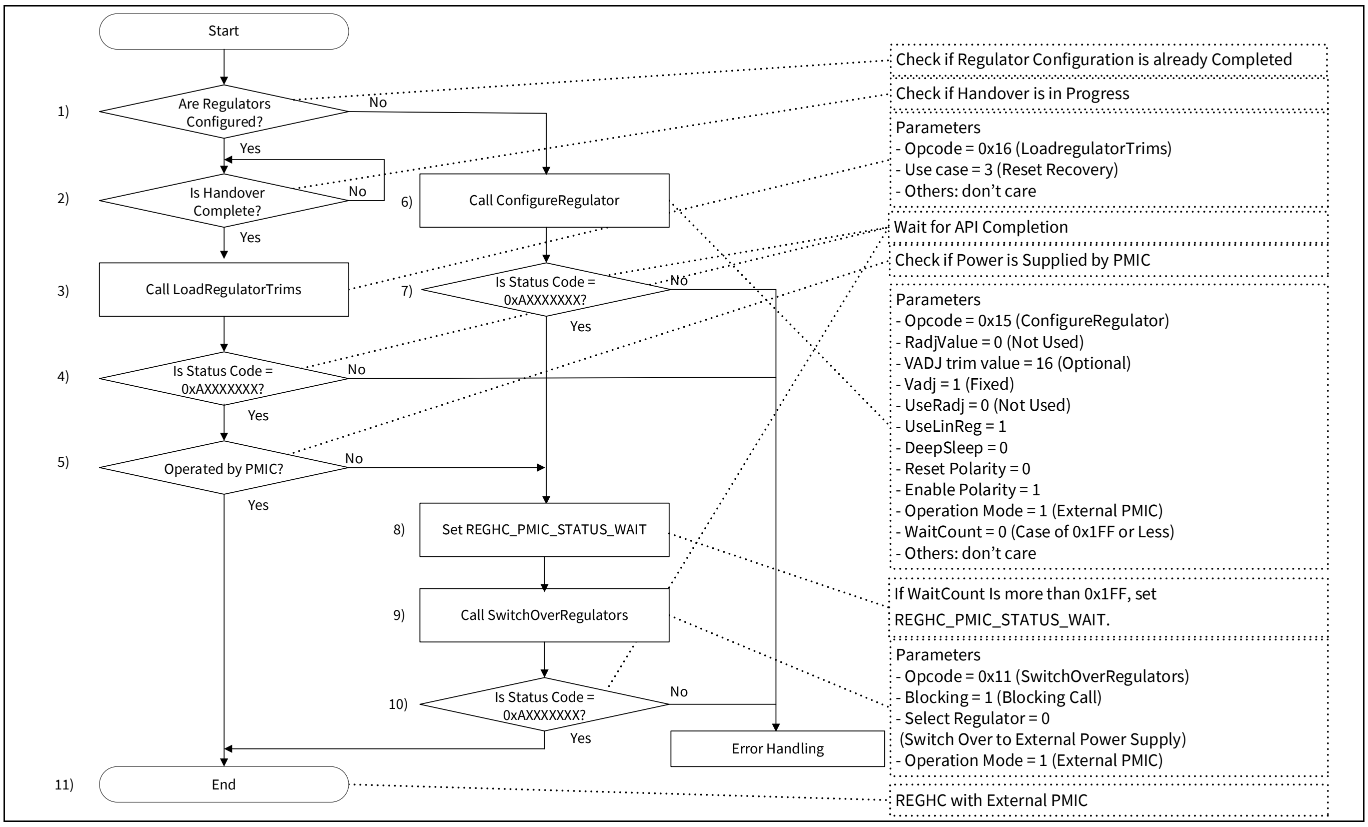

Start
1) Are Regulators Configured?
Yes
No
2) Is Handover Complete?
Yes
No
3) Call LoadRegulatorTrims
4) Is Status Code = 0xAXXXXXXX?
Yes
No
5) Operated by PMIC?
Yes
No
6) Call ConfigureRegulator
7) Is Status Code = 0xAXXXXXXX?
Yes
No
8) Set REGHC_PMIC_STATUS_WAIT
9) Call SwitchOverRegulators
10) Is Status Code = 0xAXXXXXXX?
Yes
No
Error Handling
11) End
Check if Regulator Configuration is already Completed
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ trim value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 1
- DeepSleep = 0
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 0 (Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
REGHC with External PMIC

Figure 33 **Handover flow from the internal regulator to the PMIC with load switch 1**

1. Check if the regulator configuration is already completed. Once the REGHC is set up, it can be enabled without configuring again.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (6).

2. Wait for the handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

3. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)
4. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
5. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. Go to (11). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (8).
6. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 1 (Internal Active regulator is kept as enabled after the PMIC is enabled)

PMIC (switching regulator)

- DeepSleep = 0 (PMIC is disabled in DeepSleep power mode)
- Reset Polarity = 0 (PMIC PG signal indicates power bad status with "0")
- Enable Polarity = 1 (PMIC is enabled by "1")
- Operating Mode = 1 (External PMIC)
- WaitCount = 0 (0x1FF or less)

7. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
8. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.

To enable the REGHC with the PMIC, perform these steps:

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over to external power supply)
   - Operating Mode = 1 (External PMIC) This parameter must match the operating mode in the `ConfigureRegulator` API.
2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. The device is now operating with the PMIC. Active and DeepSleep regulators operate in external state.

DeepSleep entry and exit

Figure 34 shows the transition to DeepSleep power mode in the PMIC with load switch 1.

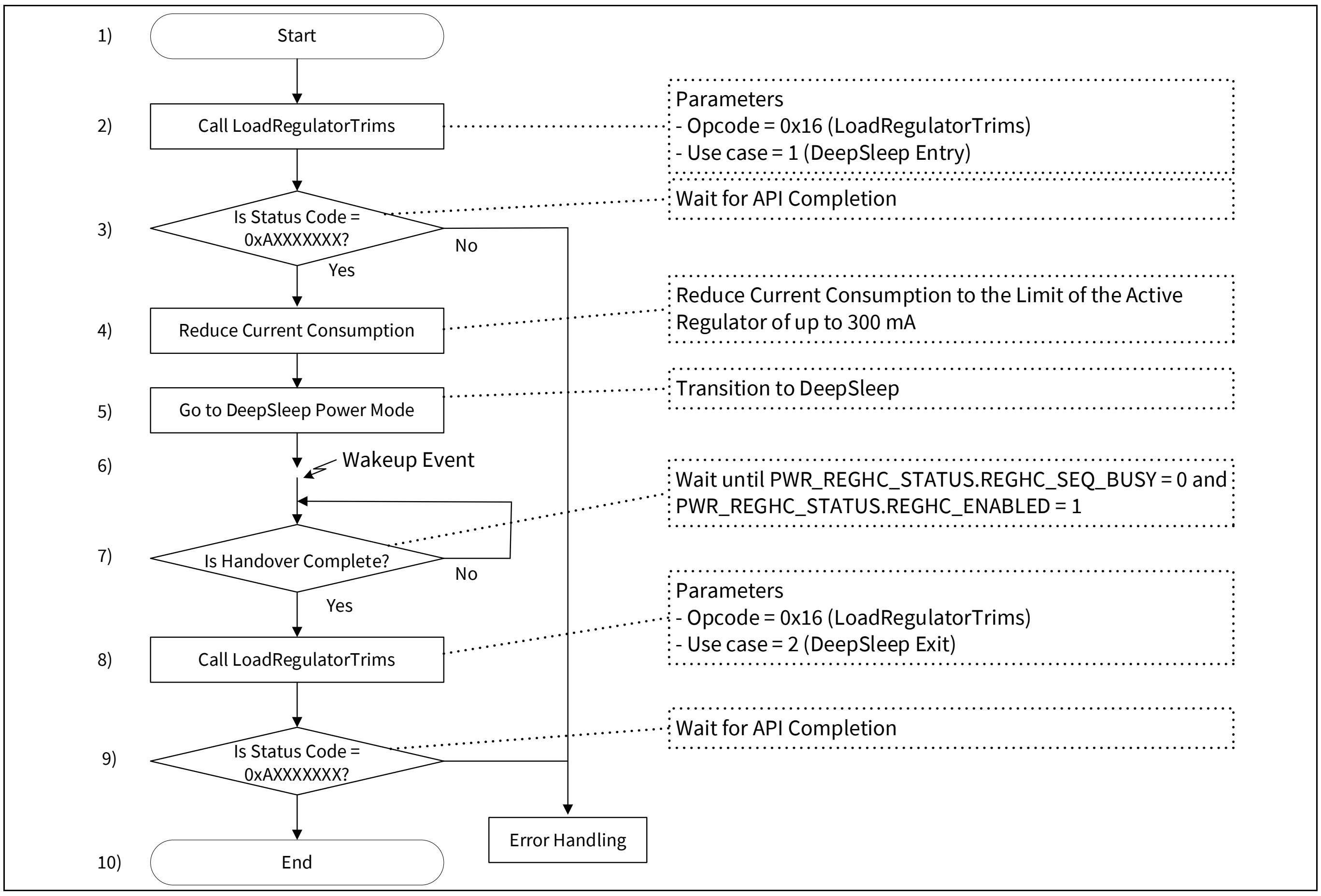

1)
Start
2)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 1 (DeepSleep Entry)
3)
Is Status Code = 0xAXXXXXXX?
No
Yes
Wait for API Completion
4)
Reduce Current Consumption
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
5)
Go to DeepSleep Power Mode
Transition to DeepSleep
6)
Wakeup Event
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
7)
Is Handover Complete?
No
Yes
8)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadRegulatorTrims)
- Use case = 2 (DeepSleep Exit)
9)
Is Status Code = 0xAXXXXXXX?
Wait for API Completion
Error Handling
10)
End

Figure 34 **DeepSleep entry and exit flow in the PMIC with load switch 1**

1. The MCU is powered by the PMIC.
2. Call the `LoadRegulatorTrims` API with the following parameters before transitioning to DeepSleep power mode:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 1 (DeepSleep Entry)
3. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA.
5. Start the transition to DeepSleep power mode using the "WFI" instruction. Perform handover to the internal regulator; the PMIC is disabled. Then, the Active regulator is disabled; the DeepSleep regulator supplies power during DeepSleep power mode.

The MCU is powered by the DeepSleep regulator.

6. Due to wakeup event, the MCU transitions to Active power mode. Then, the hardware enables the PMIC.
7. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
8. Call the `LoadRegulatorTrims` API with the following parameters. Active and DeepSleep regulators are set to external state:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 2 (DeepSleep exit)

9. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
10. The MCU is powered by the PMIC.

Handover to the internal regulator from the PMIC

Figure 35 shows handover from the PMIC to the internal regulator.

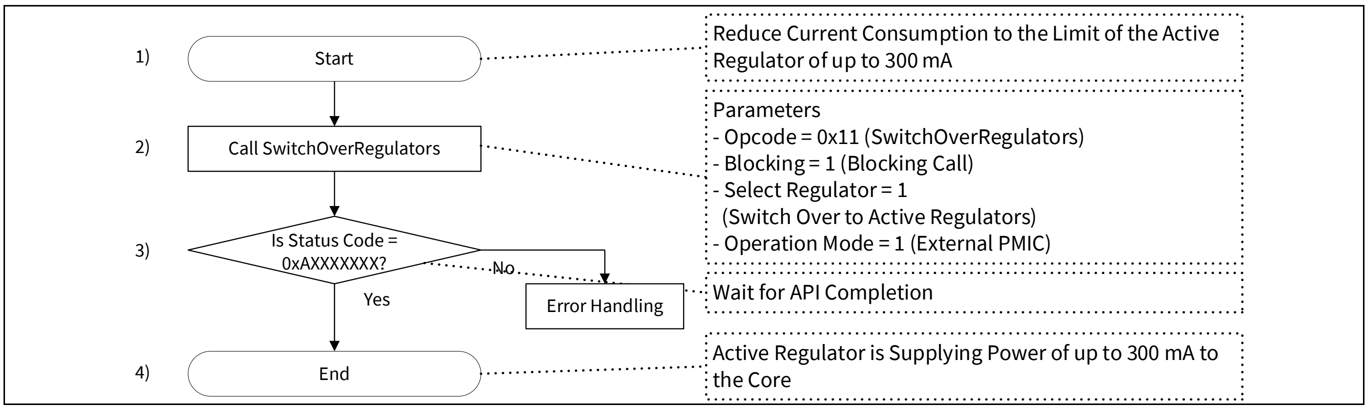

1)
Start
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
2)
Call SwitchOverRegulators
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 1
(Switch Over to Active Regulators)
- Operation Mode = 1 (External PMIC)
3)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
Wait for API Completion
4)
End
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 35 Handover flow from the PMIC with load switch 1 to the internal regulator

To transition from the REGHC with the PMIC to the internal regulator:

1. Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA.
2. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 1 (Switch over to Active regulator)
   - Operating Mode = 1 (External PMIC)
3. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. The device is now operating with the internal regulator.

4.4.2 Case 2

In this case, OCD is not used in Active mode, the PMIC is enabled in DeepSleep power mode. (Using SwitchOverRegulators as a blocking call) The load switch gate is controlled by the REGHC pin.

The load switch is required for isolating the PMIC from V_{CCD} depending on the choice of the PMIC topology. See Required PMIC specification for more details. The REGHC with the PMIC configuration does not initialize by LV resets (Software, FAULT, WDT, MCWDT, CSV) but by HV resets. (XRES, POR, BOD, OVD, OCD, HIBERNATE wakeup) See the “Reset system” chapter in the architecture reference manual for reset sources.

4.4.2.1 Hardware configuration

Figure 36 shows the circuit diagram for the PMIC with load switch 2.

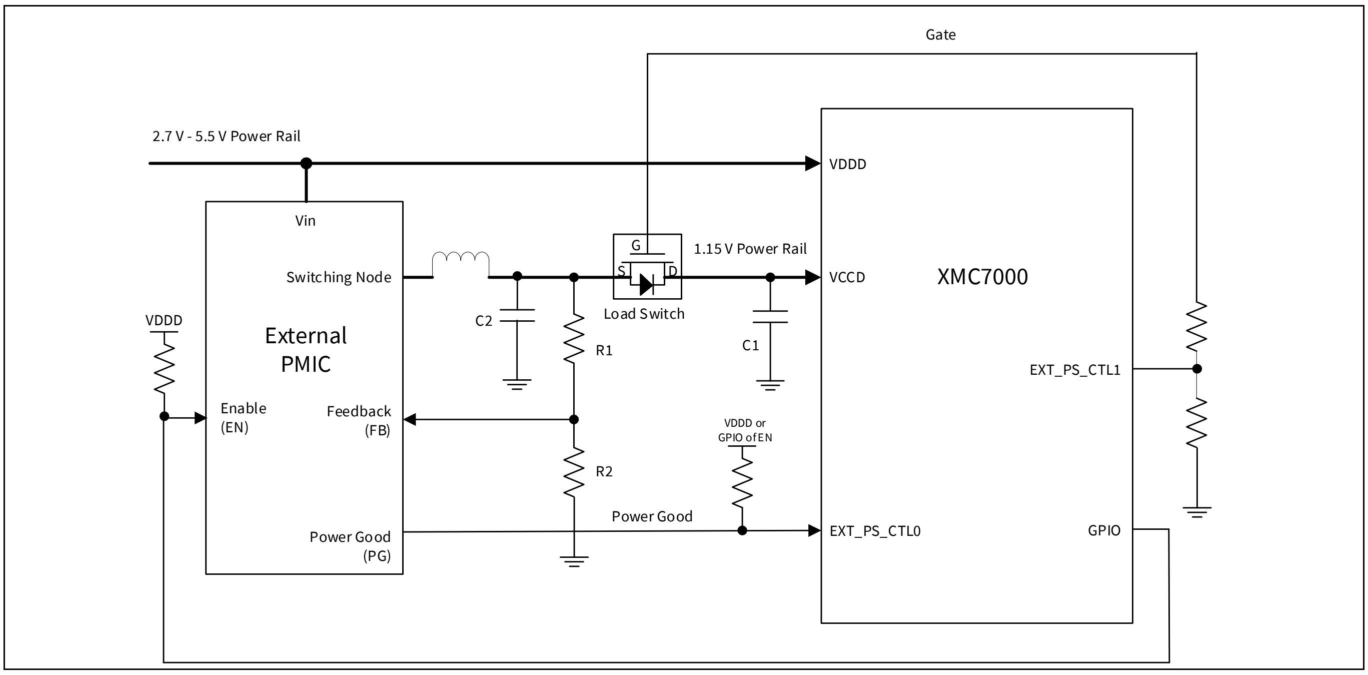


Figure 36 **Circuit diagram for the PMIC with load switch 2**

- PMIC EN polarity is HIGH for enable. The PMIC PG polarity is HIGH for PG.

  RESET_PMIC is generated when PG of the PMIC goes LOW. This is HV reset. HV reset is applied and the microcomputer including the REGHC with the PMIC is initialized.
- The PMIC feedback signal is connected to the load switch on the PMIC side.
- The load switch is connected between the PMIC output and V_{CCD} of the MCU.
- The gate signal of the load switch is connected to the EXT_PS_CTL1 pin of the MCU.
- The EN pin of the PMIC is connected to the GPIO of the MCU. To maintain correct operation during DeepSleep mode, use the GPIO in the power domain that keeps the power even during DeepSleep mode.
- A pull-up resistor is connected to the GPIO (EN) signal to enable the PMIC during LV reset
- A resistor to restrict the current of the load switch gate charge to less than 1 mA is put between EXT_PS_CTL1 and the gate of the load switch.
- V_{CCD} capacitor (C1) depends on the MCU requirement (see the device datasheet for the capacitance).
- Output voltage setting resistors (R1, R2) and output capacitor (C2) depend on the selected PMIC.

Note: *In this use case, the current consumption increases during DeepSleep. If this is acceptable, you can use this use case.*

The following software flow assumes that the register setting of the PMIC enable polarity is “1” and PG status polarity is “1”.

Enable polarity in the `ConfigureRegulator` API specifies the polarity of the enable signal to the PMIC and Reset polarity in the `ConfigureRegulator` API specifies the polarity of the PG state from the PMIC. Note that Reset polarity is set to power good deasserted polarity. These settings depend on the PMIC specifications.

In this use case, the internal regulator configuration does not use OCD, and the PMIC is enabled in DeepSleep power mode.

4.4.2.2 Handover timing chart for the PMIC

Figure 37 shows an example of the handover sequence with load switch 2. This example includes regulators configuration, handover from internal regulators to the PMIC, transition to DeepSleep power mode, and wakeup from DeepSleep power mode. In this configuration, the API cannot be used to switch back from the PMIC to the internal regulator. To switch back to the internal regulator, you need to reset the entire chip (HV reset). If your system requires to run the handover from the PMIC to internal regulators by the `SwitchOverRegulators` API, see Case 3 for details.

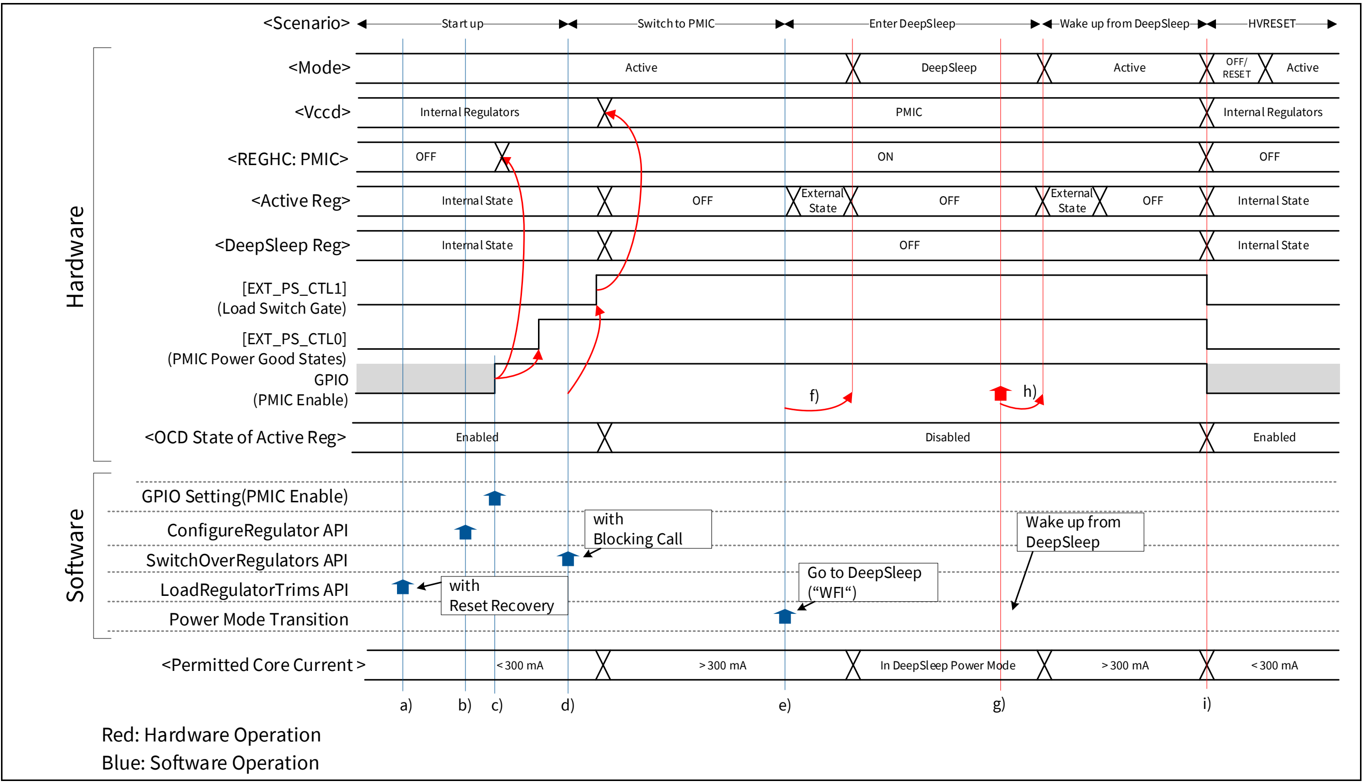


Figure 37 Handover sequence example of the PMIC with load switch 2 (Case 2)

The initial state of the GPIO (PMIC enable) depends on the pin termination.

a) Call the `LoadRegulatorTrims` API with the reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to "1". See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC with the PMIC configuration after starting up.

c) User software sets the GPIO to "1" to enable the PMIC. Then, the PMIC is enabled, and EXT_PS_CTL0 is in power good state after the PMIC is ready. Note that the PMIC is enabled here, but no handover has been performed.

d) Call the `SwitchOverRegulators` API as a blocking call for the handover from internal regulators to the PMIC.

EXT_PS_CTL1 turns ON the load switch using the `SwitchOverRegulators` API, then the MCU supplies power from the PMIC. The Active regulator and DeepSleep regulator are disabled.

If your system requires to run the handover from the PMIC to internal regulators using the `SwitchOverRegulators` API, you need to call the `SwitchOverRegulators` API as a non-blocking call when running the handover to the PMIC.

e) You can execute the transition to DeepSleep power mode using the "WFI" instruction. In this case, the system can transition to DeepSleep power mode directly without reducing the current consumption.

When transitioning to DeepSleep power mode, the PMIC is still enabled, and Active and DeepSleep regulators are disabled.

f) The MCU transitions to DeepSleep power mode.

g) A power mode transition from DeepSleep to Active power mode happens after the wakeup event.

h) Active power mode transition is completed; the PMIC will continue to supply power. Active and DeepSleep regulators are disabled.

i) When an HV reset occurs, the power system is initialized. Therefore, the MCU starts up with the Active regulator after reset is released.

4.4.2.3 Software flow

Transitioning from the internal regulator to the PMIC

Figure 38 shows the transition from the internal regulator to the PMIC.

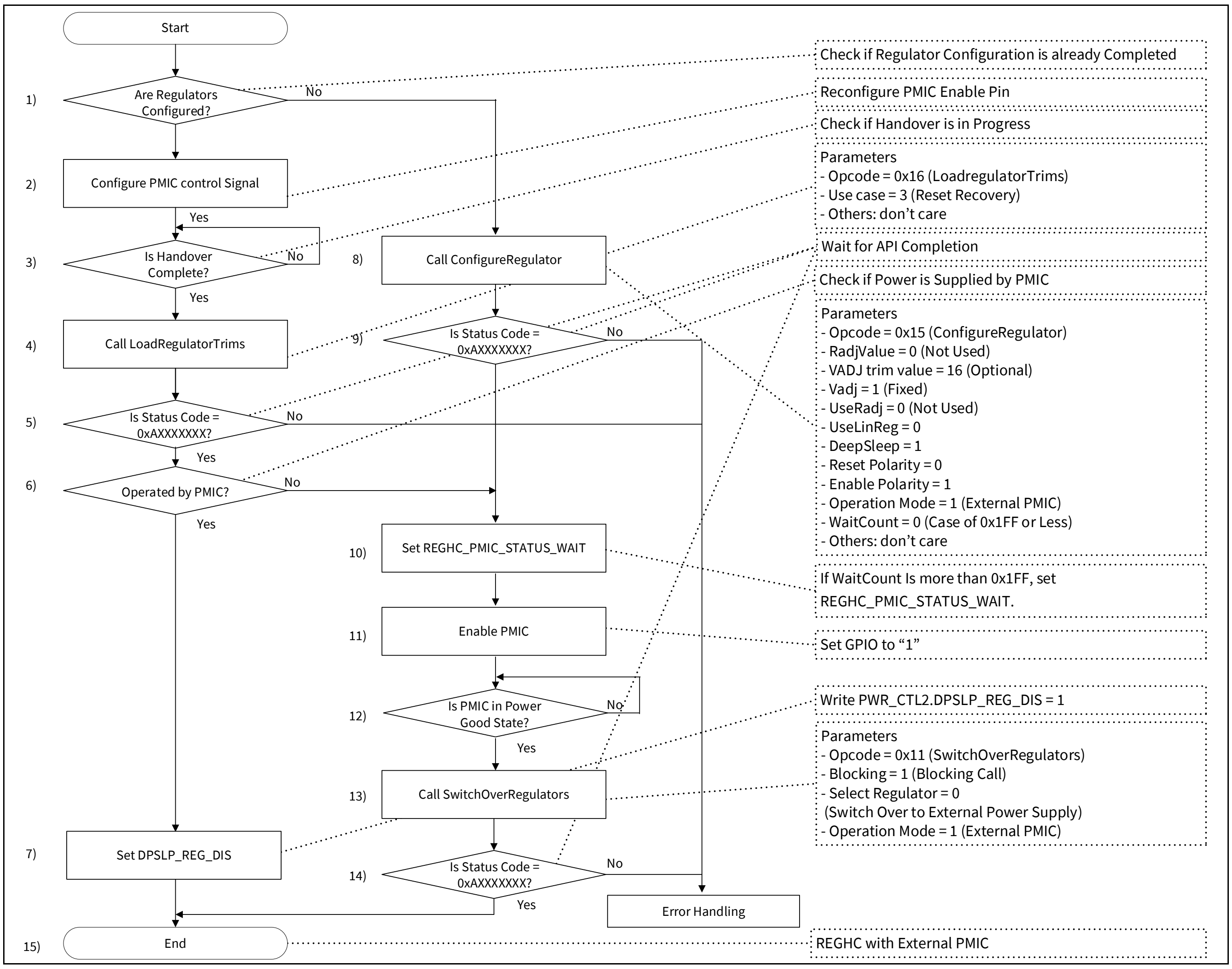

Start
1) Are Regulators Configured?
No
2) Configure PMIC control Signal
Yes
3) Is Handover Complete?
No
Yes
4) Call LoadRegulatorTrims
5) Is Status Code = 0xAXXXXXXX?
No
Yes
6) Operated by PMIC?
No
Yes
7) Set DPSLP_REG_DIS
8) Call ConfigureRegulator
9) Is Status Code = 0xAXXXXXXX?
No
10) Set REGHC_PMIC_STATUS_WAIT
11) Enable PMIC
12) Is PMIC in Power Good State?
No
Yes
13) Call SwitchOverRegulators
14) Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
15) End
Check if Regulator Configuration is already Completed
Reconfigure PMIC Enable Pin
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don't care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ trim value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 0
- DeepSleep = 1
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don't care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Set GPIO to "1"
Write PWR_CTL2.DPSLP_REG_DIS = 1
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 1 (Blocking Call)
- Select Regulator = 0
(Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
REGHC with External PMIC

Figure 38 Handover flow from the internal regulator to the PMIC with load switch 2 (Case 2)

1. Check if the regulator configuration is already completed. Once the REGHC is set up, it can be enabled without configuring again.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to "0", the configuration is not complete. Go to (8).

2. Configure the GPIO for PMIC enable control. In this case, the PMIC enable state before an LV reset is reconfigured. The following is an example of the GPIO reconfiguration:

   a) Write ˜ (PWR_REGHC_CTL2.REGHC_EN ^ PWR_REGHC_CTL.REGHC_PMIC_CTL_POLARITY) to the GPIO data register.

   b) Configure to GPIO output.

3. Wait for handover completion.

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

4. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)

5. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
6. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to “1”, the PMIC is supplying power. Go to (7). When PWR_REGHC_STATUS.REGHC_ENABLED is set to “0”, the internal regulator is supplying power. Go to (12).
7. Set PWR_CTL2.DPSLP_REG_DIS to “1”.

   In Case 2, the PMIC is enabled in DeepSleep power mode. Therefore, if the handover to the PMIC has already been completed, the user software needs to set PWR_CTL2.DPSLP_REG_DIS to "1". If not, PWR_CTL2.DPSLP_REG_DIS is set by the `SwitchOverRegulators` API in blocking mode.

8. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 0 (Internal Active regulator is disabled after the PMIC is enabled)
   - DeepSleep = 1 (PMIC is enabled in DeepSleep power mode)
   - Reset polarity = 0 (PMIC PG signal indicates power bad status with “0”)
   - Enable polarity = 1 (PMIC is enabled by “1”)
   - Operating Mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)

Note: *In this configuration, a resistor to restrict the current is placed between EXT_PS_CTL1 (PMIC enable signal output) and the load switch. The load switch may not turn ON immediately due to this resistor and capacitance of the load switch. Therefore, the handover completion need to wait for the load switch to completely turn ON. You can use a hardware timer to wait for the load switch to turn ON. This wait time can be configured using WaitCount in the `ConfigureRegulator` API. See Load switch for calculating the required wait time.*

9. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
10. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.
11. Set GPIO to “1” for enabling the PMIC. Then, the PMIC is enabled.
12. Wait until PWR_REGHC_STATUS.REGHC_PMIC_STATUS_OK = 1. It means waiting for the PMIC to enter the power good state. If the PMIC indicates PG state before the PMIC output stabilizes sufficiently, software can wait for the PMIC output to stabilize.

To enable the REGHC with the PMIC, perform these steps:

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 1 (Blocking call)
   - Select regulator = 0 (Switch over to external power supply)

- Operating Mode = 1 (External PMIC) This parameter must match the operating mode in the `ConfigureRegulator` API.

Note: *If your system requires to run the handover from the PMIC to internal regulators using the `SwitchOverRegulators` API, you need to call the `SwitchOverRegulators` API as a non-blocking call when running the handover to the PMIC.*

2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. The device is now operating with the PMIC. Active and DeepSleep regulators are disabled.

DeepSleep entry and exit

Figure 39 shows the transition to DeepSleep power mode in the PMIC with load switch 2. In this case, you can transition to DeepSleep power mode without any additional action.

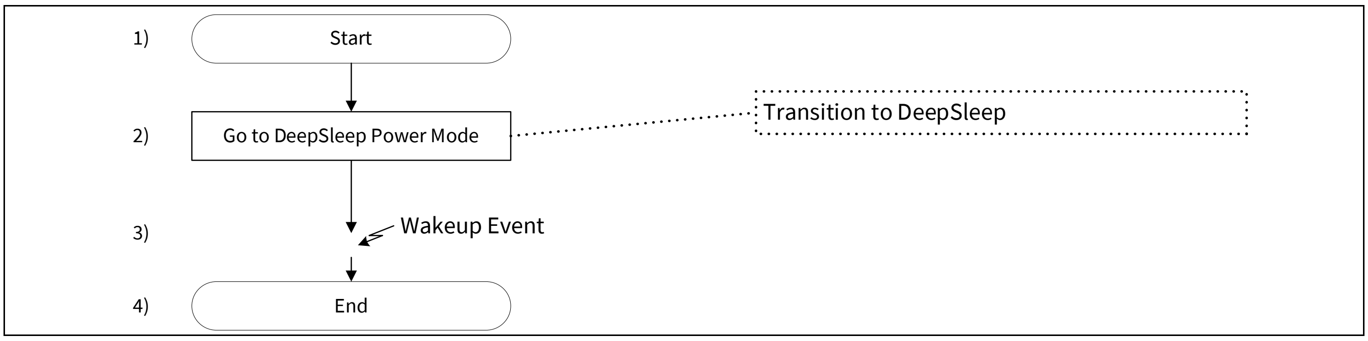


Figure 39 DeepSleep entry and exit flow in the PMIC with load switch 2 (Case 2)

1. The MCU is powered by the PMIC.
2. Start the transition to DeepSleep power mode using the "WFI" instruction. The Active regulator is disabled, and the PMIC continues to supply power.
3. Due to the wakeup event, the MCU transitions to Active power mode. The Active regulator changes to external state.
4. The MCU is powered by the PMIC.

4.4.3 Case 3

In this case, OCD is not used in Active mode; the PMIC is enabled in DeepSleep power mode (using SwitchOverRegulators as a non-blocking call). The load switch gate is controlled by the REGHC pin.

The circuit diagram of this case is the same as Figure 36.

Figure 40 shows an example of the handover sequence with load switch 2. This example includes regulators configuration, handover from internal regulators to the PMIC, and handover from the PMIC to internal regulators.

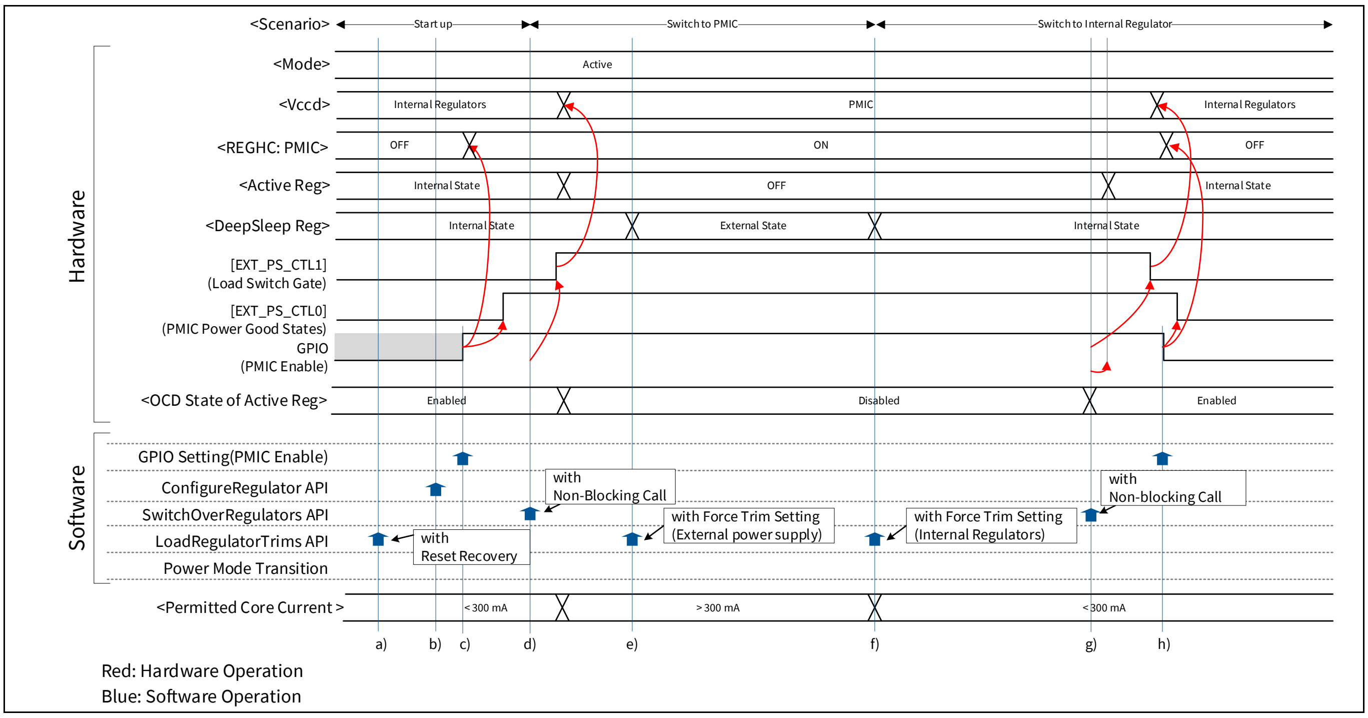


Figure 40 Handover sequence example of the PMIC with load switch 2 (Case 3)

The initial state of the GPIO (PMIC enable) depends on the pin termination.

a) Call the `LoadRegulatorTrims` API with reset recovery use case, if PWR_REGHC_CTL.REGHC_CONFIGURED is set to “1”. See the reset recovery for details.

b) Call the `ConfigureRegulator` API for the REGHC with the PMIC configuration after starting up.

c) User software sets GPIO to “1” to enable the PMIC. Then, the PMIC is enabled, and EXT_PS_CTL0 is in power good state after the PMIC is ready. Note that the PMIC is enabled here, but no handover has been performed.

d) Call the `SwitchOverRegulators` API as a non-blocking call for the handover from internal regulators to the PMIC.

Note: *When calling the `SwitchOverRegulators` API as a non-blocking call, the API does not set PWR_CTL2.DPSLP_REG_DIS to “1”. Therefore, you can switch back to internal regulators from the PMIC using the `SwitchOverRegulators` API. However, if the application software sets the PWR_CTL2.DPSLP_REG_DIS to “1”, you cannot switch back to internal regulators from the PMIC.*

The PMIC is enabled. The Active regulator is disabled, because the OCD feature is not requested.

e) Call the `LoadRegulatorTrims` API with the Force trim setting use case and external power supply operating mode for changing the internal regulator state. The DeepSleep regulator changes to external state.

f) Call the `LoadRegulatorTrims` API with the Force trim setting use case and internal regulators operating mode for changing the internal regulator state. The DeepSleep regulator changes to internal state.

g) Call the `SwitchOverRegulators` API as a non-blocking call for the handover from the PMIC to internal regulators. Then, EXT_PS_CTL1 turns OFF the load switch. User software must wait for the handover completion.

h) User software sets GPIO to “0” to disable the PMIC. Then, the PMIC is disabled and EXT_PS_CTL0 is in power good deasserted state. The PMIC is disabled.

4.4.3.1 Software flow

Transitioning from the internal regulator to the PMIC

Figure 41 shows the transition from the internal regulator to the PMIC.

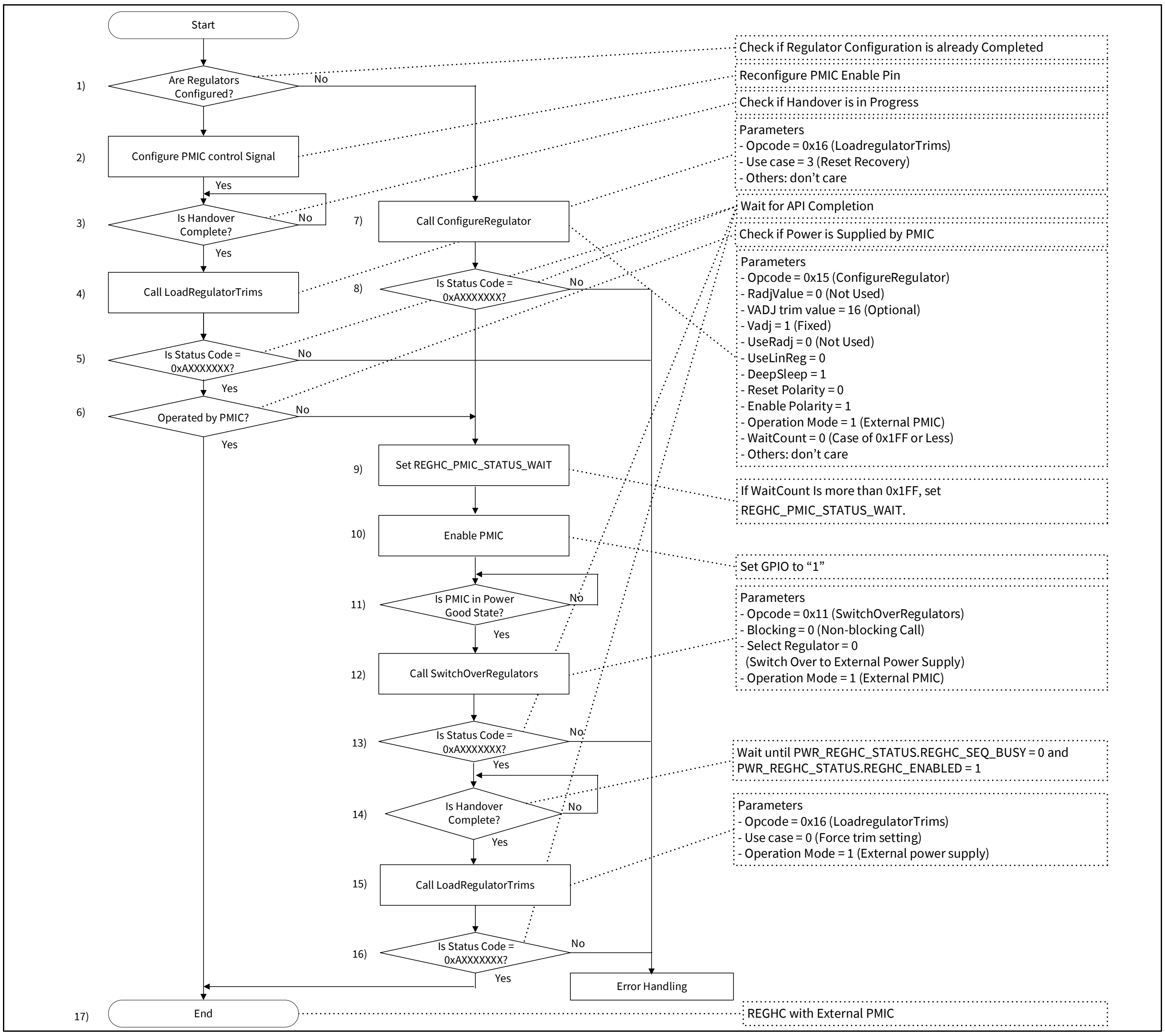

Start
1)
Are Regulators Configured?
No
2)
Configure PMIC control Signal
Yes
3)
Is Handover Complete?
No
Yes
4)
Call LoadRegulatorTrims
5)
Is Status Code = 0xAXXXXXXX?
No
Yes
6)
Operated by PMIC?
No
Yes
7)
Call ConfigureRegulator
8)
Is Status Code = 0xAXXXXXXX?
No
9)
Set REGHC_PMIC_STATUS_WAIT
10)
Enable PMIC
11)
Is PMIC in Power Good State?
No
Yes
12)
Call SwitchOverRegulators
13)
Is Status Code = 0xAXXXXXXX?
No
Yes
14)
Is Handover Complete?
No
Yes
15)
Call LoadRegulatorTrims
16)
Is Status Code = 0xAXXXXXXX?
No
Yes
Error Handling
17)
End
Check if Regulator Configuration is already Completed
Reconfigure PMIC Enable Pin
Check if Handover is in Progress
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 3 (Reset Recovery)
- Others: don’t care
Wait for API Completion
Check if Power is Supplied by PMIC
Parameters
- Opcode = 0x15 (ConfigureRegulator)
- RadjValue = 0 (Not Used)
- VADJ trim value = 16 (Optional)
- Vadj = 1 (Fixed)
- UseRadj = 0 (Not Used)
- UseLinReg = 0
- DeepSleep = 1
- Reset Polarity = 0
- Enable Polarity = 1
- Operation Mode = 1 (External PMIC)
- WaitCount = 0 (Case of 0x1FF or Less)
- Others: don’t care
If WaitCount Is more than 0x1FF, set REGHC_PMIC_STATUS_WAIT.
Set GPIO to “1”
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 0 (Non-blocking Call)
- Select Regulator = 0
(Switch Over to External Power Supply)
- Operation Mode = 1 (External PMIC)
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 0 (Force trim setting)
- Operation Mode = 1 (External power supply)
REGHC with External PMIC

Figure 41 Handover flow from the internal regulator to then PMIC with load switch 2 (Case 3)

1. Check if the regulator configuration is already completed. Once the REGHC is set up, it can be enabled without reconfiguring.

   When PWR_REGHC_CTL.REGHC_CONFIGURED is set to “0”, the configuration is not complete. Go to (7).

2. Configure the GPIO for PMIC enable control. In this case, the PMIC enable state before an LV reset is reconfigured. The following is an example of the GPIO reconfiguration:

a) Write (PWR_REGHC_CTL2.REGHC_EN ^ PWR_REGHC_CTL.REGHC_PMIC_CTL_POLARITY) to the GPIO data register.

b) Configure to GPIO output.

3. Wait for the handover completion:

   Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = "0" and PWR_REGHC_STATUS.REGHC_ENABLED = PWR_REGHC_CTL2.REGHC_EN.

4. Call the `LoadRegulatorTrims` API with reset recovery:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 3 (Reset recovery)
5. Wait for the completion of the `LoadRegulatorTrims` API. If the status code is other than 0xAXXXXXXX, perform appropriate error handling according to your system.
6. Check if the MCU operates with the PMIC. When PWR_REGHC_STATUS.REGHC_ENABLED is set to "1", the PMIC is supplying power. Go to (17). When PWR_REGHC_STATUS.REGHC_ENABLED is set to "0", the internal regulator is supplying power. Go to (9).
7. Call the `ConfigureRegulator` API with the following parameters:
   - Opcode = 0x15 (Call the `ConfigureRegulator` API)
   - RadjValue = 0 (Not used)
   - VADJ trim value = 16
   - Vadj = 1 (Fixed)
   - UseRadj = 0 (Not used)
   - UseLinReg = 0 (Internal Active regulator is disabled after the PMIC is enabled)
   - DeepSleep =1 (PMIC is enabled in DeepSleep power mode)
   - Reset Polarity = 0 (PMIC PG signal indicates power bad status with "0")
   - Enable Polarity = 1 (PMIC is enabled by "1")
   - Operating Mode = 1 (External PMIC)
   - WaitCount = 0 (0x1FF or less)

Note: *In this configuration, a resistor to restrict the current is placed between EXT_PS_CTL1 (PMIC enable signal output) and the load switch. The load switch may not turn ON immediately due to this resistor and capacitance of the load switch. Therefore, the handover completion need to wait for the load switch to completely turn ON. You can use a hardware timer to wait for the load switch to turn ON. This wait time can be configured using WaitCount in the `ConfigureRegulator` API. See Load switch for calculating the required wait time.*

8. Wait for the `ConfigureRegulator` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
9. When you need to set the WaitCount to a value greater than 0x1FF, set the PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT register. If not, this process is unnecessary because it can be set by the `ConfigurRegulator` API. See the `ConfigureRegulator` API in System Call API for more details.
10. Set GPIO to "1" for enabling the PMIC. Then, the PMIC is enabled.
11. Wait until PWR_REGHC_STATUS.REGHC_PMIC_STATUS_OK = 1. It means waiting for the PMIC to enter the power good state. If the PMIC indicates the PG state before the PMIC output stabilizes sufficiently, software can wait for the PMIC output to stabilize.

To enable the REGHC with the PMIC, perform these steps:

PMIC (switching regulator)

1. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 0 (Non-blocking call)
   - Select regulator = 0 (Switch over to external power supply)
   - Operating Mode = 1 (External PMIC) This parameter must match the operating mode in the `ConfigureRegulator` API.

Note: The `SwitchOverRegulators` API as a non-blocking call does not set PWR_CTL2.DPSLP_REG_DIS to "1". See SwitchOverRegulators API non-blocking call handling for details.

2. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
3. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 1.
4. Call the `LoadRegulatorTrims` API with the force trim setting:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 0 (Force trim setting)
   - Operating Mode = 1 (External power supply)

This process changes the internal regulator output state to external state.

5. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
6. The device is now operating with the PMIC. The Active regulator is in OFF state, and DeepSleep regulator operates in external state.

Handover from the PMIC to the internal regulator

Figure 42 shows the handover flow from the PMIC to the internal regulator.

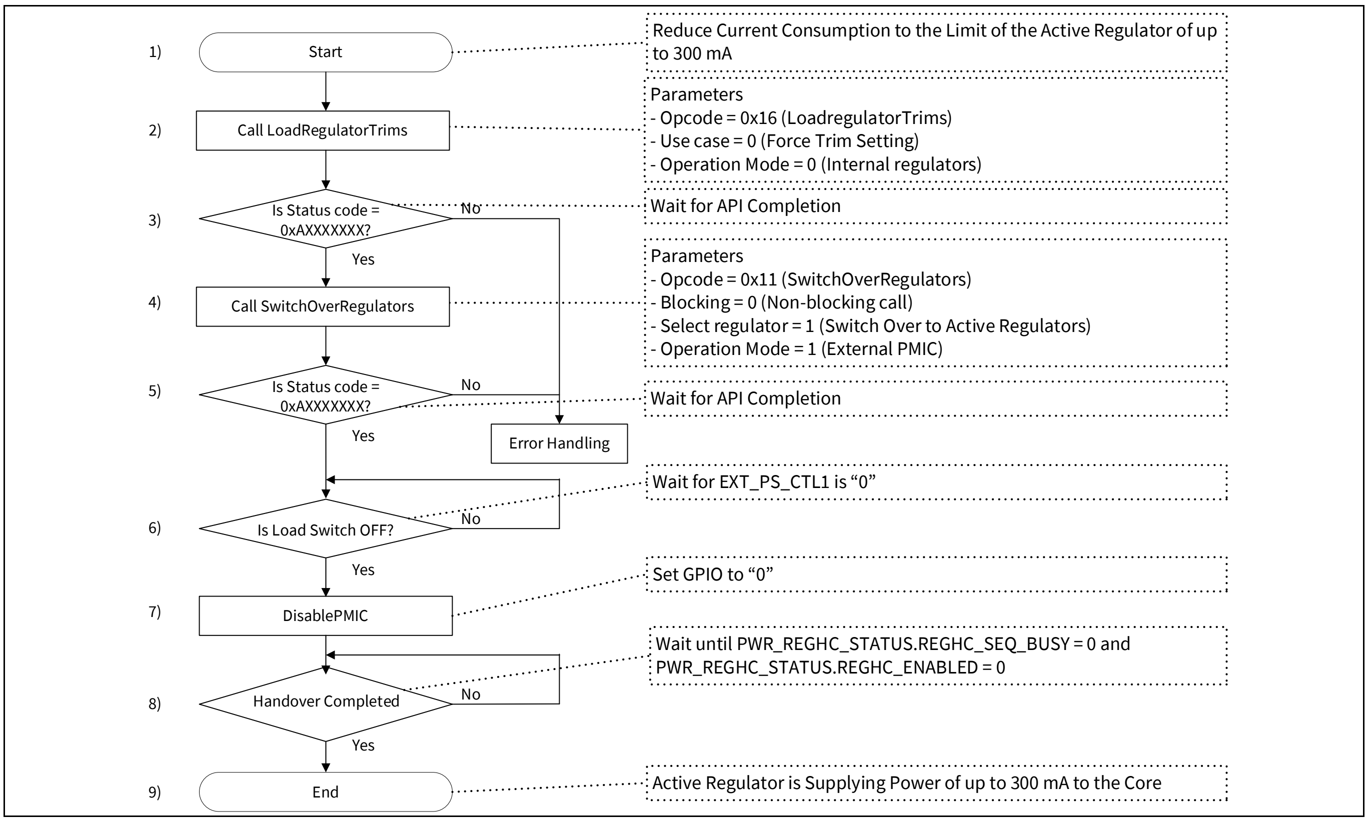

1)
Start
Reduce Current Consumption to the Limit of the Active Regulator of up to 300 mA
2)
Call LoadRegulatorTrims
Parameters
- Opcode = 0x16 (LoadregulatorTrims)
- Use case = 0 (Force Trim Setting)
- Operation Mode = 0 (Internal regulators)
3)
Is Status code = 0xAXXXXXXX?
No
Yes
Wait for API Completion
4)
Call SwitchOverRegulators
Parameters
- Opcode = 0x11 (SwitchOverRegulators)
- Blocking = 0 (Non-blocking call)
- Select regulator = 1 (Switch Over to Active Regulators)
- Operation Mode = 1 (External PMIC)
5)
Is Status code = 0xAXXXXXXX?
No
Yes
Wait for API Completion
Error Handling
Wait for EXT_PS_CTL1 is “0”
6)
Is Load Switch OFF?
No
Yes
Set GPIO to “0”
7)
DisablePMIC
Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 0
8)
Handover Completed
No
Yes
9)
End
Active Regulator is Supplying Power of up to 300 mA to the Core

Figure 42 **Handover flow from the PMIC to the internal regulator with load switch 2 (Case 3)**

To transition from the PMIC to the internal regulator:

1. Reduce the current consumption to be within the limit of the Active regulator, which is 300 mA.
2. Call the `LoadRegulatorTrims` API with the force trim setting:
   - Opcode = 0x16 (Call the `LoadRegulatorTrims` API)
   - Use case = 0 (Force trim setting)
   - Operating Mode = 1 (internal regulators)

This process changes the internal regulator output state to internal state.

3. Wait for the `LoadRegulatorTrims` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
4. Call the `SwitchOverRegulators` API with the following parameters:
   - Opcode = 0x11 (Call the `SwitchOverRegulators` API)
   - Blocking = 0 (Non-blocking call)
   - Select regulator = 1 (Switch over to the Active regulator)
   - Operating Mode = 1 (External PMIC)
5. Wait for the `SwitchOverRegulators` API to complete. If the status code is other than 0xAXXXXXXX, you need to perform appropriate error handling according to your system.
6. Wait for software timer to end, which means wait for the load switch to be completely turned OFF.

Note: *In this configuration, a resistor to restrict the current is placed between EXT_PS_CTL1 (PMIC enable signal output) and the load switch. The load switch may not turn OFF immediately after PMIC disable output due to this resistor and the capacitance of using the load switch. Therefore, the* *handover completion need to wait for the load switch to completely turn OFF. Implement the timer using software, and wait for the load switch to turn OFF. See Load switch for calculating the required waiting time.*

7. Set GPIO to "0" for disabling the PMIC.
8. Wait until PWR_REGHC_STATUS.REGHC_SEQ_BUSY = 0 and PWR_REGHC_STATUS.REGHC_ENABLED = 0.
9. The device is now operating with the internal regulator.

4.5 Component selection

This section describes the selection of components.

4.5.1 Peripheral components of the PMIC

In the PMIC direct connection, the output capacitance of the PMIC is shared with the V_{CCD} capacitance. For more details on selecting the output capacitor, see Component selection.

Also, the output voltage setting resistor should consider the bias current by itself. Note that this bias current also flows during DeepSleep power mode.

For other PMIC components, see the PMIC datasheet.

4.5.2 Load switch

Table 18 lists the load switch requirement.

Table 18 Load switch requirement specification

Parameter	Symbol	Value	Unit
Source-drain voltage	V_{DSO}	$\geq V_{DDD}$	V
Source-gate voltage	V_{GSO}	$\geq V_{DDD}$	V
Drain current	I_D	≥ 0.6	A

R_{ON} of the load switch may cause a V_{CCD} voltage drop. V_{CCD} drop voltage by R_{ON} is calculated using Equation 3.

$$V_{CCD\_drop} = R_{ON\_max} \times I_{VCCD\_max}$$

Equation 3

Where:

- V_{CCD_drop}: V_{CCD} drop voltage (V)
- R_{ON_max}: Load switch maximum on resistance (Ω)
- I_{VCCD_max}: Maximum V_{CCD} load current (A)

Select a load switch that fits the V_{CCD} voltage requirement of the MCU including this voltage drop. Low R_{ON} is recommended in consideration of voltage drop when current flows.

The source-gate voltage of the load switch during the ON state is the source-gate minimum voltage of the load switch in Equation 4. The load switch must turn ON with this voltage.

$$V_{GS\_min} = V_{G\_min} - V_{CCD\_max}$$

Equation 4

Where:

- V_{GS_min}: Source-gate minimum voltage of the load switch (V)
- V_{G_min}: Ground-gate minimum voltage of the load switch (V)
  - For PMIC with load switch case 1, V_{G_min} means power good pull-up minimum voltage.
  - For PMIC with load switch case 2, V_{G_min} means minimum voltage of EXT_PS_CTL1=H.
- V_{CCD_max}: Maximum V_{CCD} voltage (V)

In case of the PMIC with load switch 2, EXT_PS_CTL1 might supply overcurrent momentarily to drive the load switch gate. In this case, a current restriction resistor is needed to satisfy the maximum current of EXT_PS_CTL1. (For more details, see device datasheet).

Required resistance of the load switch gate is calculated using Equation 5.

$$R_{Load\ switch\ gate} \geq V_{DDD\_max}/I_{EXT\_PS\_CTL1\_peak}$$

Equation 5

Where:

- $R_{Load\_switch\_gate}$: EXT_PS_CTL1 current restriction resistor (Ω)
- $I_{EXT\_PS\_CTL1\_peak}$: EXT_PS_CTL1 peak current to drive the load switch gate (A)
- V_{DDD_max}: Maximum V_{DDD} voltage (V)

For example, if V_{DDD} maximum voltage is 5.5 V, a resistor more than 5.5 kΩ is required to reduce the current to 1 mA or less.

Furthermore, by this current restriction resistor and Ciss of the load switch, the voltage waveform of the load switch gate becomes a discharge curve. As a result, ON/OFF status transient of load switch is delayed. This time constant is calculated using Equation 6.

$$T_{EXT\_PS\_CTL1} = R_{Load\ switch\ gate} \times C_{iss\_load\ switch}$$

Equation 6

Where:

- $T_{EXT\_PS\_CTL1}$: Time constant of load switch gate and current restriction resistors
- $C_{iss\_load\_switch}$: Input capacitance of load switch (F)
- $R_{Load\_switch\_gate}$: EXT_PS_CTL1 current restriction resistor (Ω)

The load switch gate delay time that rises to 95% of EXT_PS_CTL1 high voltage, is three times the value of τ. The load switch gate delay time that falls to 5% of EXT_PS_CTL1 high voltage, is also the same.

4.6 Layout design guidelines

Here are a few PCB layout design guidelines:

- The output of the PMIC and load switch should be located as close as possible to the V_{CCD} terminal of the MCU.

For other precautions, follow the PMIC layout guidelines from the PMIC supplier.

5 Appendix A. Hardware sequence

The following shows an example sequence of XMC7000 series.

Handover sequence between the pass transistor and Active regulator

Figure 43 shows the handover sequence of the hardware between the pass transistor and Active regulator.

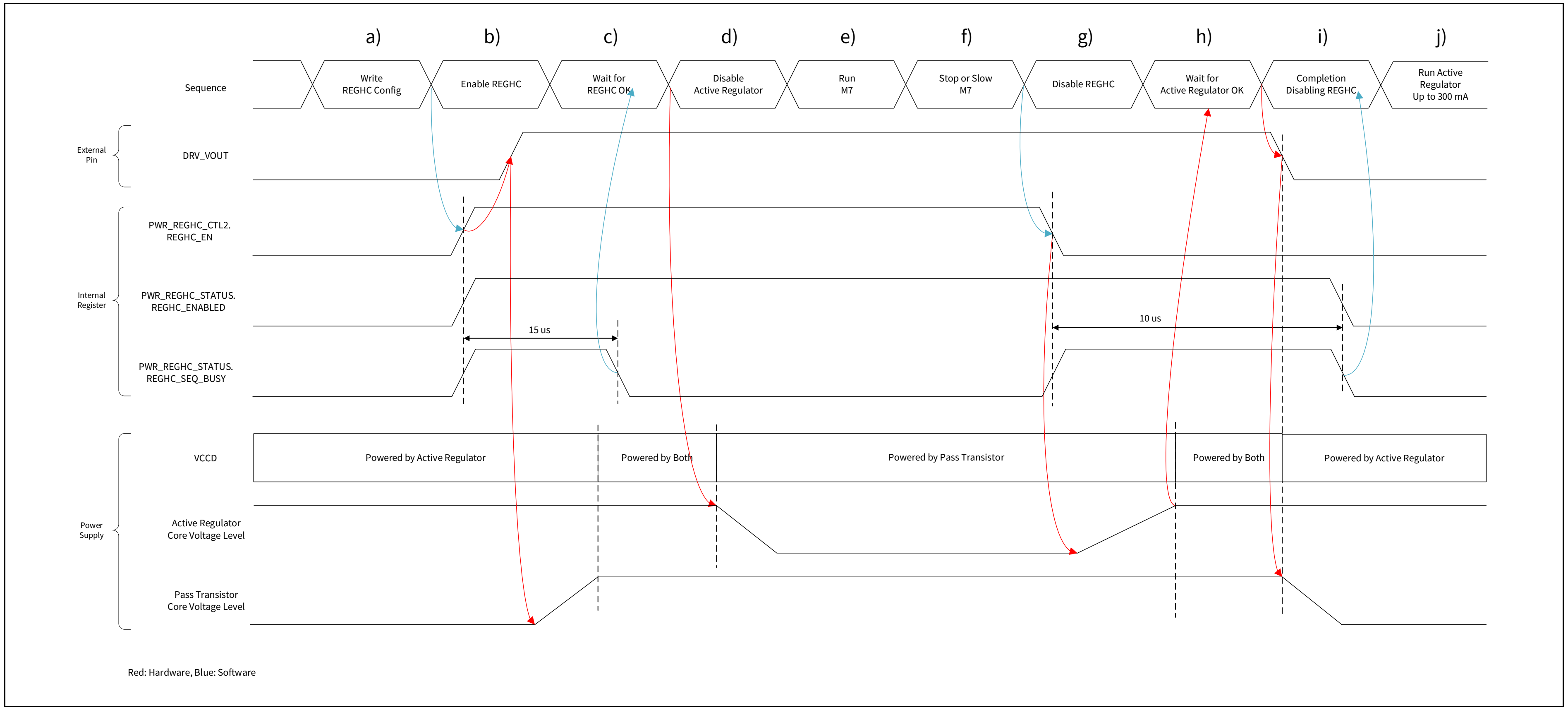

a)
b)
c)
d)
e)
f)
g)
h)
i)
j)
Sequence
Write REGHC Config
Enable REGHC
Wait for REGHC OK
Disable Active Regulator
Run M7
Stop or Slow M7
Disable REGHC
Wait for Active Regulator OK
Completion Disabling REGHC
Run Active Regulator Up to 300 mA
External Pin
DRV_VOUT
Internal Register
PWR_REGHC_CTL2. REGHC_EN
PWR_REGHC_STATUS. REGHC_ENABLED
PWR_REGHC_STATUS. REGHC_SEQ_BUSY
15 us
10 us
Power Supply
VCCD
Powered by Active Regulator
Powered by Both
Powered by Pass Transistor
Powered by Both
Powered by Active Regulator
Active Regulator Core Voltage Level
Pass Transistor Core Voltage Level
Red: Hardware, Blue: Software

Figure 43 Handover sequence between the pass transistor and Active regulator

The MCU starts with the Active regulator during startup:

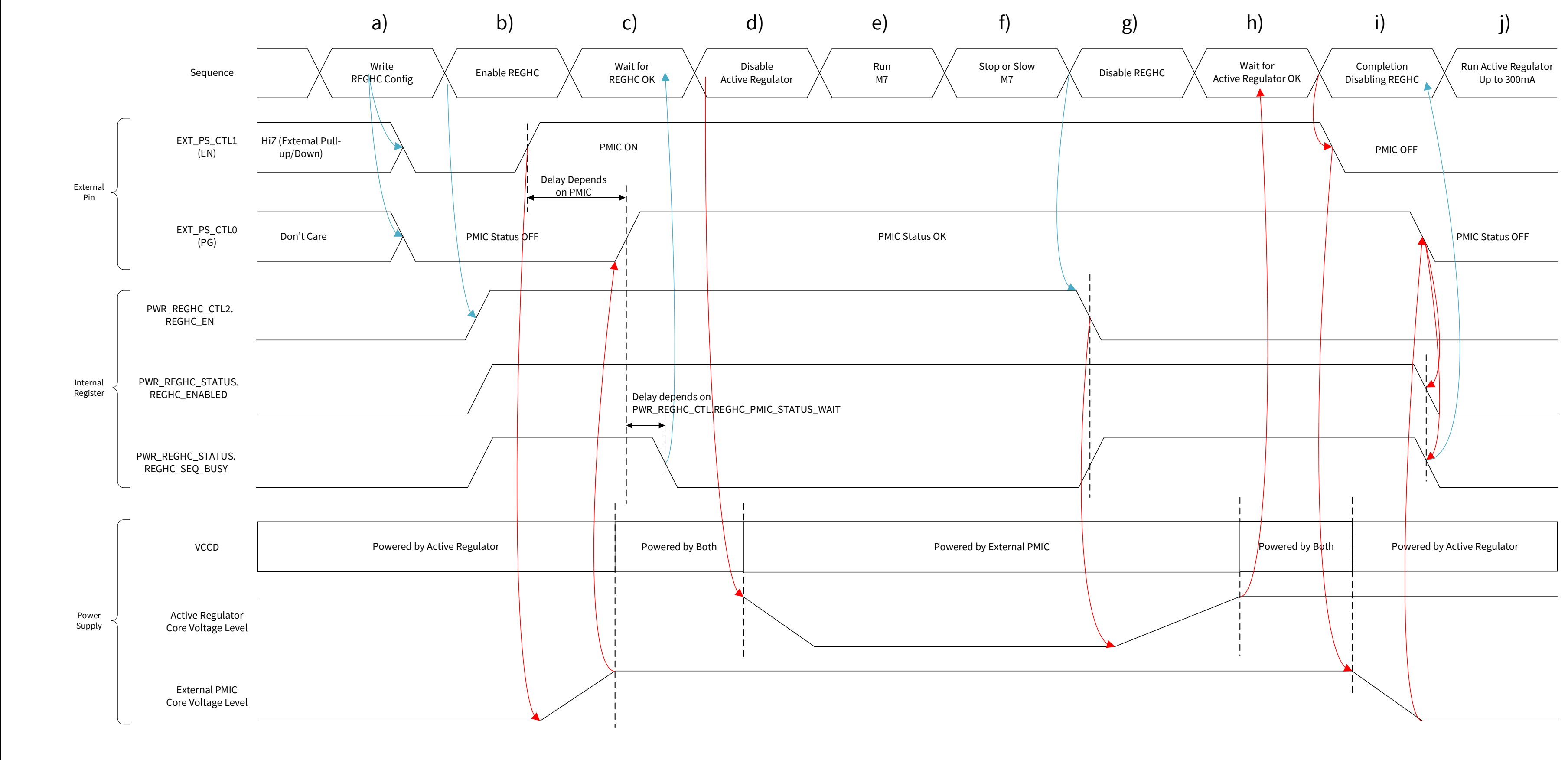
a) SW: Configures the REGHC with the pass transistor in external pass transistor mode.
b) SW: Enables the REGHC with the pass transistor. Then, the REGHC starts to operate the pass transistor by driving the DRV_VOUT pin.
c) SW: Waits for the REGHC OK flag, until the external core voltage level is reached.
d) HW: Disables the Active regulator.
e) MCU runs with the REGHC (pass transistor).

Note: The application software should increase the operating clock frequency step-by-step to avoid V_{CCD} voltage undershoot.

f) Prior to the transition to the Active regulator, the current consumption of the application must be within the limit of the Active regulator.
g) SW: Disables the REGHC with the pass transistor.
h) HW: Waits for the Active regulator to be fully active.
i) HW: Completes the REGHC disabling procedure.
j) MCU runs with the Active regulator.

Handover sequence between the PMIC and Active regulator

Figure 44 shows the handover sequence of the hardware between the PMIC and Active regulator.



Condition: PMIC EN polarity is HIGH for enable. PMIC PG polarity is HIGH for power good.
Red: Hardware, Blue: Software

Figure 44 Handover sequence between the PMIC and Active regulator

The MCU starts with the Active regulator during startup:

a) SW: Configures the REGHC with the PMIC.

b) SW: Enables the REGHC with the PMIC. Then, the external PMIC starts to operate.

c) SW: Waits for the REGHC OK flag. The REGHC OK flag is set when EXT_PS_CTL0 is in power good state after the PMIC is ready.

d) If EXT_PS_CTL0 is in power good state during soft-start before 100% of the external target voltage is reached, the internal Active regulator may be turned OFF and V_{CCD} voltage undershoot might occur after an immediate increase in the current consumption of the application (for example, PLL). To mitigate this problem, you can use REGHC_PMIC_STATUS_WAIT to cover the settling time of the PMIC. This is used by the hardware sequencer to set an additional settling time before disabling the internal Active regulator. HW disables the Active regulator. V_{CCD} is powered by the external PMIC.

e) The MCU can run with the external PMIC.

Note: *The application software must increase the operating clock frequency step-by-step to avoid V_{CCD} voltage undershoot.*

f) The MCU needs to run below the limit of the Active regulator prior to transitioning to the Active regulator.

g) SW: Starts the procedure of handover from the PMIC to internal regulator. Disables the REGHC with the PMIC.

h) HW: Waits for the Active regulator to be fully active.

i) HW: Completes the disabling of the external PMIC.

j) The MCU runs with the Active regulator.

Abbreviations

Abbreviation	Description
API	Application programming interface
BOD	Brown-out detection
BOM	Bill of materials
EP	Exposed pad
GPIO	General-purpose I/O
HVD	High-voltage detection
HV reset	High-voltage reset
LDO regulator	Low-dropout regulator
LV reset	Low-voltage reset
MCU	Microcontroller unit
OCD	Overcurrent detection
OVD	Overvoltage detection
PCB	Printed circuit board
PG	Power Good output signal from the PMIC
PFM	Pulse frequency modulation
PMIC	Power management integrated circuit
PWM	Pulse width modulation
POR	Power-on-reset
REGHC	High current regulator controller
SROM	Supervisory ROM
WFI	Wait for interrupt CPU instruction

References

Contact Technical Support to obtain XMC7000 family reference documents and PDL.

[1] Device datasheet

- XMC7100 series 32-bit Arm® Cortex®-M7 microcontroller datasheet
- XMC7200 series 32-bit Arm® Cortex®-M7 microcontroller datasheet

[2] Reference manuals

- XMC7000 MCU family architecture reference manual
- XMC7100 register reference manual
- XMC7200 register reference manual

[3] Application notes

- AN234802 - Secure system configuration in XMC7000 family
- AN234334 - Getting started with XMC7000 MCU on ModusToolbox™
- AN234254 - Multi-core handling guide in XMC7000
- AN234196 - Using the CRYPTO module in XMC7000 family

Revision history

Document version	Date of release	Description of changes
**	2022-01-31	Initial release
*A	2024-07-18	Updated references and links

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-07-18

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2024 Infineon Technologies AG.
All Rights Reserved.

Do you have a question about this document?

Email: **erratum@infineon.com**

Document reference

002-34415 Rev. *A

Important notice
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Please note that this product is not qualified according to the AEC Q100 or AEC Q101 documents of the Automotive Electronics Council.

Warnings
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.