16/32-Bit

Architecture

XC2000/XE166 Family

16/32-Bit Single-Chip Microcontroller with
32-Bit Performance
Clock and Power Management

Programmer's Guide

V2.1 2009-07

Microcontrollers

Edition 2009-07

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, 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.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

16/32-Bit

Architecture

XC2000/XE166 Family

16/32-Bit Single-Chip Microcontroller with
32-Bit Performance
Clock and Power Management

Programmer's Guide

V2.1 2009-07

Microcontrollers

XC2000/XE166 Programmer's Guide

Revision History: V2.1 2009-07

Previous Versions: V2.0, 2009-05

Page	Subjects (major changes since last revision)
Page 5-22	Disabling of Flash module changed
Page 5-27	Setting of Wakeup Timer modified: Clear wakeup trigger added
Page 5-31	Setting of Wakeup Timer modified: Clear wakeup trigger added
Page 5-34	Enabling of Flash module changed
Page 5-37	VCOLCK ermergency not enabled
Page 5-40	Setting of Wakeup Timer modified: Clear wakeup trigger added
Page 5-45	Setting of Wakeup Timer modified: Clear wakeup trigger added

Trademark

C166™, TriCore™ and DAVE™ are trademarks of Infineon Technologies AG.

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

Table of Contents

1 Introduction

The microcontrollers of the XC2000 Family come with enhanced means for system clock control and power management. This document describes how to set-up the supply and clock system of the XC2000/XE166 in a practical, expedient, and robust way.

Scope

This manual describes guidelines for the following types of microcontrollers of the Infineon XC2000/XE166 Family: Base / Value / High Line.

These microcontrollers provide identical functionality to a large extent, but each device type has its own set of specific features.

Therefore, some sections of this manual do not refer to all of the XC2000/XE166 derivatives which are currently available or planned:

- The power-saving modes and transitions described in **Section 4.3** and **Section 5.3**ff are only applicable to the XC22xxy devices.
- The Ultra Low Power EVR is only avialble in the XC2xxy devices of the Value and Base Line.

For simplicity, these various device types are referred to by the collective term **XC2000/XE166** throughout this manual. The complete Pro Electron conforming designations are listed in the respective Data Sheets.

Using this Manual

The first more verbose part of the Programmer's Guide comprises all necessary information on the objectives and configuration of the Operating Modes, and it explains step by step how to conduct transitions between the modes in a robust manner. Several further chapters give additional background information regarding low level system programming and the XC2000/XE166 architecture.

The Operating Mode Transition sequences are compiled out of a pool of very basic Expert Level Functions, which are documented in the reference part of this document.
To ease the browsing through the ebook version of this document, the topics of the first part are closely hyperlinked. A **Glossary** at the end explains frequently used terms and abbreviations.

Reliability of the Content

All configuration settings and sequences have been tested for accurate functionality and robustness against unexpected events, such as voltage drop or clock failure.

Please note that the content of this document is subject to change due to technical improvements or other optimizations.

Note on Numbers and Values

All numbers given in this documentation are either typical values or, regarding time-out values, have been defined sufficiently high enough.

2 Power Budgeting

This chapter discusses the various types of currents in the chip and the difference between “current saving” and “power budgeting”. It has been put at this place, because the Operating Modes and Mode Transitions have been defined considering **power budgeting** as discussed here.

For a current sensitive **Application**, the overall current consumption inside the chip is the key factor.
In a stationary mode, i.e. where clock settings and supply settings are kept unchanged for a certain time frame, the current consumption of a microcontroller typically is composed of certain components, as shown in **Table 2-1**, and can be taken as stable.

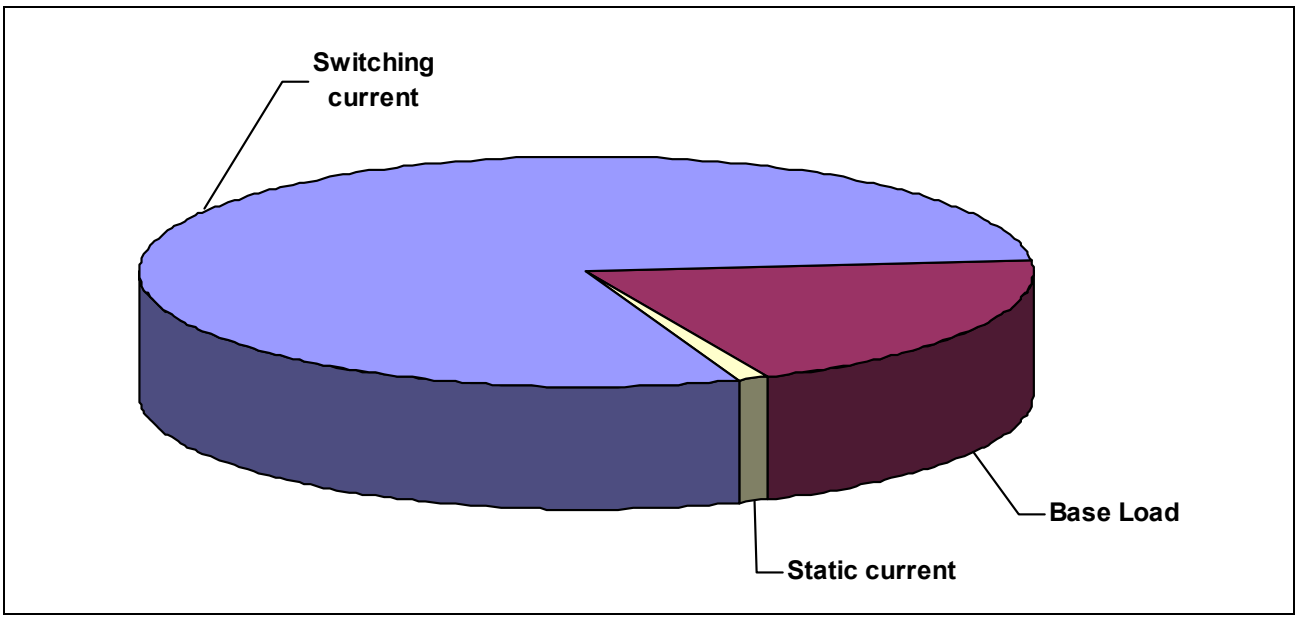

Figure 2-1 Typical composition of the overall current consumption in a high performance mode at room temperature

- **Switching Current**
  The switching current results from loading and unloading the clock lines. It directly and linearly depends on the system clock frequency and the area to which the clock is supplied. In a clocked Operating Mode with a reasonable CPU performance (= system clock speed), the switching current is the major part of the power consumption. Thus, to reduce the current consumption of the XC2000/XE166, reducing the clock frequency must always be the first measure.
- **Base Load Current**
  This part of the current consumption results out of bias and static currents inside analog components, such as oscillators and **EVVR**s. For the XC2000/XE166, the base load can sum up to a level of several mA. A way to reduce the Base Load portion of the current consumption is to switch off unused analog modules. But this only makes sense in case the system clock frequency has been reduced enough,

such that the switching current is in the range of the base load. For the XC2000/XE166, this frequency level is around 1 MHz.

- **Leakage Current**
  Leakage is the cross current which permanently drains from active powered silicon areas to VSS (ground). It depends linearly on the area of the supplied logic, and exponentially on the temperature. For the derivatives of the XC2000/XE166 Family, the leakage can be neglected at room temperature. But with temperatures above 80 °C, the leakage may raise into the mA range. For the XC2000/XE166, the only measure to get rid of the leakage is to switch off the VDDI power supply.

The XC2000/XE166 Datasheet provides a formula to calculate the share of the single contributors.

From current to power

Power budgeting means that, besides the level of the current itself, also the period of time the current flows is taken into account. Thus, for calculating a proper power budget, it is required to know:

- **How long does the microcontroller reside in a certain Operating Mode?**
  The application current profile, such as shown e.g. in **Figure 2-2**, is determined by the application needs. These can be a required minimum response time upon a wakeup event, or a defined minimum system clock frequency in the active phase, required to serve an external protocol.
- **How long does it take to transit from one mode to another?**
  This parameter is mainly determined by the microcontroller hardware and the implemented flows.

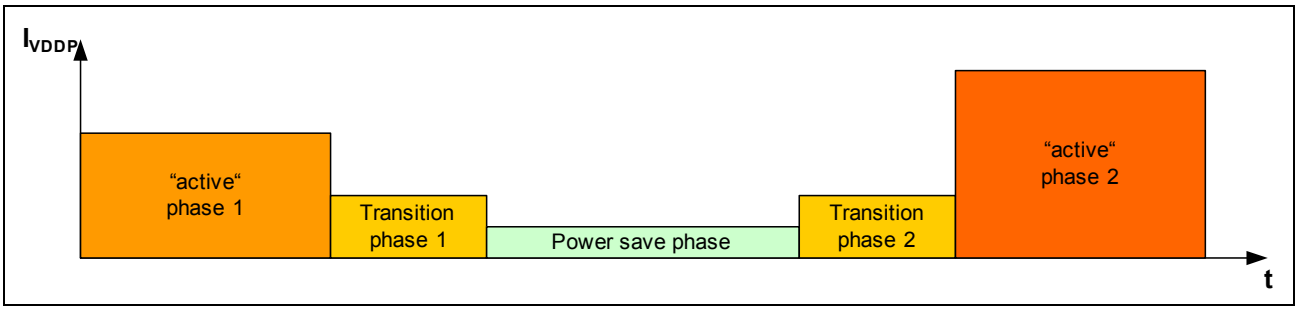

Figure 2-2 Sample power profile including phases with different levels of current consumption

To support a realistic power budgeting with the XC2000/XE166, the typical power consumption of an Operating Mode is given, as well as the time which is needed to transit between two states. The transition times listed in this document result out of an implementation of the Operating Mode Transitions **Sequence**s in C programming language. The transition times may differ depending on the starting and target clock frequency and the implementation language. This means, that the numbers given for transition times can only be used as a guiding value, and must be finally determined in the specific application.

3 General hints for programming essential system functions

This chapter gives some basic hints to support the setting and programming of the supply and the clock system of the XC2000 products.

The rules listed in the following have been taken into account for defining and testing the Mode Transitions documented in this Programmer's Guide:

- **Ramp-up and ramp-down times of analog blocks**
  Components with analog parts will not guarantee the specified behavior until they have properly finished their startup phase, and can not be considered as switched off, before a dedicated ramp-down time has elapsed.
  Software as well as hardware flows have to take this timing into account, either by implementing a waiting loop, or by design of the program flow (e.g. by working on other tasks within the required time frame).
  Table 11-1 contains a list of all timing constraints required to implement the sequences described in this document.
- **Sequence “configure - enable - use”**
  Components with **Analog** parts often show unwanted side effects when they are reconfigured in operation. Oscillators, for example, tend to generate wrong or corrupt clock signals, when their frequency settings are changed on-the-fly. Thus, the most robust way of using an analog component is to configure it completely, before using it.
  It is a good style, and it prevents possible failure, to always properly set up a feature or component first, before using it, instead of assuming reset values or to rely on previous settings. Following the sequence “configure - enable - use” unwanted side effects are avoided, such as triggering a reset while configuring an edge detection for the **ESR** pins.
- **Re-entrance of a sequence**
  When performing a configuration sequence which modifies clock or supply system settings, it is possible that the execution is suddenly interrupted, e.g. by an interrupt or by the **User Resets**. This may leave the system in an only partially configured state.
  As a consequence, it will be difficult to analyze where the interruption has taken place, and whether the state, especially of an **Analog Component**, is still the same. This is the reason why interrupts and User Resets have been disabled for most of the defined Operating Mode Transitions. See also **Chapter 7.4.1** on **Disrupting sequences**.
- **Check for availability of components before configuring them**
  This is primarily a matter of re-entrance, meaning that a piece of code is executed under different entry conditions. An example is the configuration the **SWD**, after it has been switched off by a power saving routine before.
  These conditions are often overseen, especially when the respective enable bit for

the component or feature is not contained in the same register which is used for configuration.

- **Avoid unconventional usage of components**
  Although every **Component** and feature of the XC2000/XE166 has been designed with respect to robustness, the focus of any design and verification of a **Unit** is always set on its dedicated function. Thus, using a component in an unconventional way may result in unwanted side effects, which may not even be detected in the test phase of the software.
  The basic rule to minimize the overall risk of a software flow is to use a feature or an analog part always for its primary and dedicated function.
- **Modular approach**
  Often, actions which impact the essential system functions, such as switching to another clock source, are done on-the-fly somewhere within a software flow, especially when it seems that only a single bit is needed to do it. This is very risky and error prone. Using a modular approach, where essential operations are conducted in dedicated flows, will minimize the risk of undetected mal-configuration, and does allow easier debugging.
- **A feature may cross over power domains**
  A functionality of the XC2000/XE166 may be distributed to different power domains. Thus, by changing the settings of a power domain, e.g. by switching off a domain, a feature that is currently used may suddenly become unavailable, or may be available only with restrictions. A simple example is, switching off the VDDI_1 domain while running on **PLL** clock: The PLL is located in the DMP_1 power domain. Switching the domain off will leave the chip without a clock.
  For a reference of the topology of the supply, see **Chapter 11.1**.
- **Provision of dedicated error handling scenario**
  Defining proper entry conditions and dedicated error handling scenarios for a sequential flow is a further building block on the way to a robust software. Upon an entry into an error handling routine, software should first thoroughly evaluate the state of the system, before changing essential system settings.
  This Guide gives hints, which flags are helpful to evaluate the current state of the system upon exiting from a defined sequence to an error routine.
  See also **Chapter 7** on **Exception Handling**.
- **Time-out implementation when polling for status signals**
  While polling for a status signal, e.g. the lock status from the VCO, the software will run into a deadlock, in case the expected signal state does not occur. A software concept generally shall have a respective **Exception Handling** strategy (see **Chapter 7**). Such a strategy typically stipulates a time-out exit for polling loops.
- **Design for external failure scenarios**
  Often a feature depends on the integrity of an external signal, such as, e.g. a proper clock signal. It must be expected that the integrity of an external signal may be distorted temporarily or permanently at anytime. For example, the PCB connection to a crystal connected to the XTAL pins may be improper.

Thus, the software must be able to react upon an external failure event at any time. See also **Chapter 7** on **Exception Handling**.

4 Operating Modes

This chapter details the variety of Operating Mode defined for the XC2000/XE166. Depending on the focus, an Operating Mode of a microcontroller can be defined in various ways. Because this document has the focus on the configuration of the clock and power system, an “Operating Mode” is defined by its selected system clock source, the range of the system clock frequency (f_{SYS}), and the present configuration of the supply system.

4.1 Overview

By changing clock and supply system configuration, the XC2000/XE166 is able to enter different Operating Modes. The power saving modes and cyclic waked up modes are only available for the XC22xxy devices.

4.1.1 Operating Modes

Prior to a detailed discussion of the individual Operating Modes, a transition flow is outlined in **Figure 4-1**. The modes are grouped into active modes, power-saving modes, and cyclic wakeup modes.

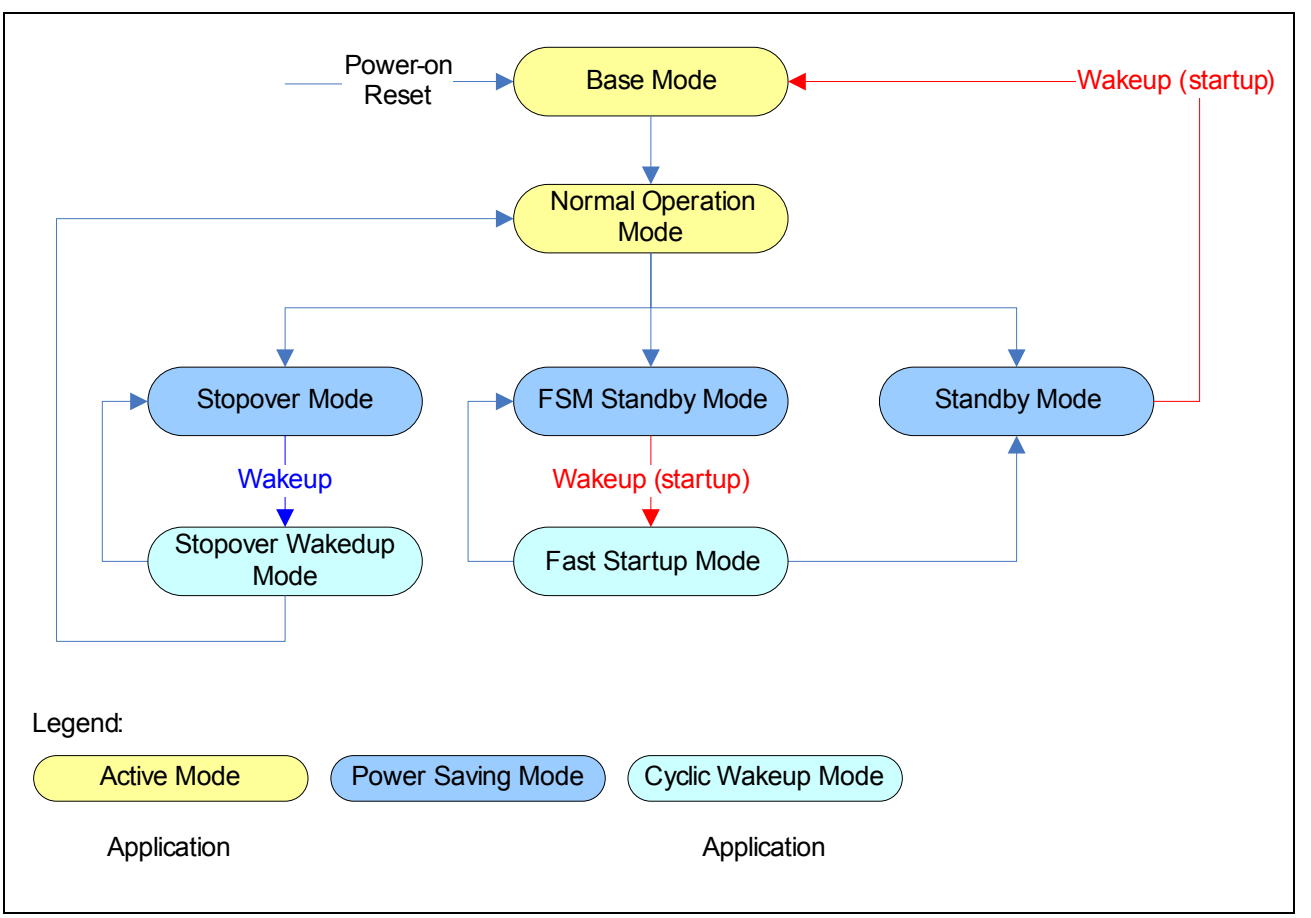

Figure 4-1 Operating Modes Overview

Base Mode

Coming from an initial power up of VDDP, or from one of the Power Resets, the application finds the system in Base Mode, running at 10 MHz from PLL locking on the internal clock. The power system is configured to run at high performance, i.e. maximum clock speed.

Base Mode is the starting point for the transition to Normal Operation Mode.

Entry: Upon Power-on Reset

Exit: Normal Operation Mode

Normal Operation Mode

Normal Operation Mode provides full system performance at a maximum accuracy of the system clock provided by the PLL. Following PLL input clock source options are possible: crystal/external clock at input XTAL, external clock at pin CLKIN1, or internal clock (f_{INT}). Normal Operation Mode is considered to be the standard mode for applications.

Normal Operation Mode is the starting point for the transition to a power-saving mode.

Entry: Base Mode

Exit: Power-saving mode (Stopover Mode, Standby Mode, or FSM Standby Mode)

Stopover Mode

Stopover Mode is preferably used for a more frequently clock based wakeup scenario.The contents of the RAM memories is preserved.

Upon a wakeup the **Stopover Wakedup Mode** is entered, where the CPU will be made available again. This allows a software based decision whether to enter Stopover Mode again, or whether to continue with Normal Operation Mode.

Entry: Normal Operation Mode

Exit: Normal Operation Mode via Stopover Wakedup Mode

Standby Mode

From low power Standby Mode, the way back to Base Mode leads through the startup software, which is performed after any reset. The full initialisation of the device takes comparably long but complex task that need larger code size can be executed from Flash.

Entry: Normal Operation Mode

Exit: Base Mode

FSM Standby Mode

This mode provides the same power-saving features but upon a wakeup in the low power FSM Standby Mode the **Fast Startup Mode** is entered by the startup software. In this mode the system is configured for medium performance while the Flash is deactivated. This allows a periodical check of the status of the system in order to take a decision whether a full wakeup to Base Modeis required, or whether FSM Standby Mode can be entered again.

Entry: Normal Operation Mode

Exit: Standby Mode via Fast Startup Mode

In the following the Operating Modes are described in more detail.

4.1.2 Wakeup Trigger Sources

A wakeup from a power-saving mode can be triggered by one or several events:

- Wakeup Timer (WUT)
- System Timer (STM)
- External Service Request (ESR): ESRx pins or associated serial input pins.

4.2 Active Modes

Active modes are used to run applications. The power consumption mainly depends on the actual system clock frequency and a certain base load current caused by Flash, PLL, ADC, EVRs, PVCs, and SWD.

4.2.1 Base Mode

Coming from an initial power up of VDDP, or from one of the Power-on Resets, the application finds the system in Base Mode, running at 10 MHz from PLL locking on the internal clock. The power system is configured to run at high performance, i.e. maximum clock speed.

Base Mode is dedicated to perform first intermediate high speed internal operations of the user application, such as data (re-)storage from/into SBRAM, code copy into PSRAM, number crunching, etc., after a temporary power-down or an initial power-up.

Clock and supply settings have been configured by the startup software flow or upon a return from Stopover Mode.

Traps, user resets, and interrupt handling are possible.

The Flash has been powered up, and the user code is executed from Flash.

In fact, the power system is configured for maximum system performance.

The PLL VCO is configured to lock on the internal clock. This mode allows to operate the system up to the maximum clock speed, with the drawback that the clock jitter is worse compared to using an external crystal.

4.2.2 Normal Operation Mode

Normal Operation Mode provides full system performance using a PLL for the system clock. It is considered to be the standard mode for applications with CAN or other serial protocol communication. Therefore, the system clock is provided by the PLL. The current system clock frequency f_{SYS} can be changed upwards and downwards using function **RampPll**.

One of the following PLL input clock source options can be selected for Normal Operation Mode:

- Crystal/external clock at input XTAL
- External clock at pin CLKIN1
- internal clock (f_{INT})

In any case, the VCO inside the PLL is used to generate a configurable system clock.

If a crystal is used, the function **EnableHighPrecOsc** may be called at a very early stage of the initialization to reduce the waiting time for stable oscillation in the transition from Base Mode to Normal Operation Mode.

If an external clock at pin CLKIN1 is used, the clock is expected to be already stable saving the time to start a crystal. Because an external clock signal usually caries some additional noise, the jitter of the system clock frequency in this mode may be higher.

In this mode, traps, user resets, and interrupt handling are possible.

The **PLL VCO Loss of Lock** feature (**Chapter 7.3.2**) is activated, i.e. in case the PLL encounters a loss of lock, the VCO is disconnected from its source and a trap is issued.

The power consumption is mainly determined by the dynamic switching current of the clock tree. Analog components will have no remarkable influence on the overall power consumption.

4.3 Power Saving Modes

The following power-saving modes are supported:

- **Stopover Mode** with following features:
  - Power domains DMP_M and DMP_1 are switched on
  - Flash is switched off
  - Contents of all RAM memories and all registers are preserved
  - Clock in domain DMP_1 is stopped
  - Option to have crystal permanently on
  - Wakeup from Stopover Mode results in a not fully set-up of the system, the Flash module and the High Precision Bandgap are still switched off, and the wakeup

clock is used as system clock.
Application code is executed from PSRAM.

- **Standby Mode** with following features:
  - Power domain DMP_1 switched off completely
  - Power domain DMP_M is powered by 1.5 V LPR or ULPEVR, if configured
  - Power supply control switched off
  - Flash is switched off
  - Contents of the SBRAM memory and the registers in domain DMP_M are preserved
  - Wakeup from Standby Mode results in a transition to Base Mode with a full initialisation of the system performed by the startup software.
- **FSM Standby Mode** (Standby Mode with Fast Startup Mode) with following features:
  - Power domain DMP_1 switched off completely
  - Power domain DMP_M is powered by 1.5 V LPR or ULPEVR, if configured
  - Power supply control switched off
  - Flash is switched off
  - Contents of the SBRAM memory and the registers in domain DMP_M are preserved
  - Wakeup from FSM Standby Mode results in a not fully set-up of the system, the Flash module and the High Precision Bandgap are switched off, and the internal clock is used as system clock.
    Application code is executed from PSRAM.

4.3.1 Stopover Mode

The transient Stopover Mode provides a moderate power saving capabiliy by switching off the system clock f_{SYS} in the DMP_1 domain, i.e. user code can not be executed anymore. Keeping the supply level has the advantage that contents of the RAM memories and registers are preserved and the CPU can be quickly brought back into operation upon a wakeup event. The supply has not to be set up again first.

This mode is preferably used for a more frequently clock based wakeup scenario. Furthermore, it is not neccessary to initialize system after wakeup. The user code is continued transparently. Furthermore, this mode is dedicated to applications where the system timer should continue in power-saving mode or immediately after wakeup from power-saving a clock with a higher frequency and higher accuracy is required. An option allows to switch the system clock to the clock provided by the crystal or the external clock at CLKIN1.

Switching off the clock implies that traps and interrupt handling is impossible, and must be disabled at this time. Although the reset controller remains clocked in DMP_M, it is not allowed to perform a user reset in this mode. Thus, user resets are de-activated.

It has to be considered that the main effect of power saving in this mode is contributed by the lack of a clock in the large power domain DMP_1. This mode will suffer from an increased leakage at high temperatures, since the supply level remains unchanged.

Stopover Wakedup Mode

Upon a wakeup from Stopover Mode the High Precision Bandgap and the Flash module are switched off; the High Precision Oscillator is optionally switched on. This mode offers medium performance while running the system clock on the wakeup clock f_{WU}.

Because the Flash is not available in this mode, the program code is executed from PSRAM. The PSRAM has been preserved in the preceding Stopover Mode.

This mode is dedicated to be used for a periodical wakeup from Stopover Mode in order to perform tasks with medium code size within a short time frame, i.e. upon a check for a port pin status to decide whether high performance actions will be required. Even CAN transfers are possible using a crystal or external clock as source for the system clock. The active phase is followed by a new entry into Stopover Mode.

In Stopover Mode the GSC is in Clock-off System Mode. In case peripherals shall be used, the GSC must enter Normal System Mode by using function **RequestSystemMode**. Before entering Stopover Mode the Clock-off System Mode must be set using the function **RequestSystemMode** again..

4.3.2 Standby Mode

Standby Mode offers the most power saving. The large power domain DMP_1 is switched off together with several features of the supply system, such as supply monitoring (SWD, PVCs). The current control logic (CC) is additionally switched off.

Because DMP_M can be kept clocked, the Wakeup timer can be used to leave this mode after a certain time.

By switching off the power domain DMP_1 the leakage of the core logic is reduced to a minimum.

Wakeup from this results in a transition to **Base Mode** with a full initialisation of the system performed by the startup software.

4.3.3 FSM Standby Mode

In **Standby Mode** a wakeup results in a transition to **Base Mode** with a full initialisation of the system. Since this takes comparably long, alternatively in FSM Standby Mode the Standby Mode is configured that the Fast Startup Mode will be entered upon a wakeup event. In FSM Standby Mode the same power-saving methods are used as in **Standby Mode**.l

Fast Startup Mode

After wakeup from FSM Standby Mode the High Precision Oscillator, the High Precision Bandgap and the Flash module are switched off. This mode offers medium performance while running the system clock on the internal clock f_{INT} = 5 MHz.

Because the Flash is not available in this mode, the program code is executed from PSRAM. The PSRAM has not been preserved in the preceding FSM Standby Mode, thereby the application specific program code is copied from the SBRAM by the startup software before. Hence, the size of the code in FSM does not only influence the execution time of the mode itself, but also the transition time to enter it.

The Fast Startup Mode is dedicated to be used for a periodical wakeup from FSM Standby Mode in order to perform small tasks within a short time frame. i.e. upon a check for a port pin status to decide whether high performance actions will be required. The active phase is followed by a new entry into FSM Standby Mode.

In Fast Startup Mode the GSC is in Clock-off Mode. In case peripherals shall be used, the GSC must enter Normal System Mode by using function **RequestSystemMode**. Before entering **Standby Mode** GSC Clock-off Mode must be set using the function **RequestSystemMode** again.

Fast Startup Mocde is always terminated by entering Standby Mode. To leave Fast Startup Mode in order to go back to Normal Operation Mode it is neccessary to go to **Standby Mode**. Standby Mode is temporarily entered setting back the entire power

domain DMP_1 to its default state. This has been done to end up always in the same system state when the startup software is doing the supply, clock system and Flash initialisation.

Fast Startup Mode and Watchdog

As described in **Chapter 6.2**, upon the user resets the power system settings will not be restored. This applies also for Fast Startup Mode wherefore user resets are not allowed in this mode due to the restricted supply settings. However, for FSM the WDT is configured to be up and running. Because the WDT will trigger a user reset when expired it is mandatory for FSM, to generate a PORST out of this user resert as described in a dedicated application note.

4.4 Operating Mode Reference

This section lists the attributes of the Operating Modes in detail, using tables. **Table 4-1** explains the content of these tables.

Table 4-1 Mode Property Table Explanation

Attribute	Value
Clock System	
Frequency [MHz]	The number given here reflects the maximum system clock frequency allowed for this Operating Mode. If the clock source can be configured, also lower frequencies are possible for the system clock f_{SYS}. However, running on a lower frequency will have an impact on the reaction time to interrupt requests and on the conduction time for Operating Mode Transitions
System Clock Source	This lists the clock source for the system clock frequency f_{SYS} on which the current mode is based on
PLL	Status of the digital part of the PLL. PLL off does not imply, that all analog parts of the PLL are disabled.
PLL Power Regulator	Status of the PLL internal power regulator
internal clock	Status of the internal clock
VCO	Status of the VCO inside the PLL
Wakeup clock	Status of the wakeup clock
High Precision Oscillator	Status of the High Precision Oscillator
VCOLCK emergency	States whether the emergency clock handling is activated to handle PLL loss of lock situations. See **Chapter 7.3.2** for details
System	
ADC	Status of the ADC
Peripherals	General status of the peripheral control
GSC Mode	Denotes, which mode the GSC currently broadcasts in this mode
Traps/user resets/ Interrupts	Gives information on whether operational disruptions are allowed in this Mode or not
Code Location	This gives information on from which location the program code is executed. This can be: Flash, PSRAM

Table 4-1 Mode Property Table Explanation (cont'd)

Attribute	Value
Wakeup by External Service Request	Defines whether a wakeup can be triggered by an External Service Request
Wakeup by Wakeup Timer	Same as above for Wakeup Timer
Wakeup by System Timer	Same as above for System Timer
Memories	
Flash	Status of the non-volatile Flash memory
SBRAM	Status of the Standby RAM memory
RAM	Status of the other RAM memories
Power System	
Supply Watchdog	Status of the Supply Watchdog
High Precision Bandgap	Status of the High Precision Bandgap
PVC_M	Status of the supply monitoring for DMP_M
DMP_M	Current supply setting for DMP_M supply domain
CC_M	Status of the current control unit of EVR_M
PVC_1	Status of the supply monitoring for DMP_1
DMP_1	Current supply setting for DMP_M supply domain

4.4.1 Active Modes

See **Table 4-1** (**Mode Property Table Explanation**) for details regarding the attributes or the values of the table below.

Table 4-2 Properties of Active Modes

Attribute	Base Mode	Normal Operation Mode
Clock System		
Frequency [MHz]	10	f_{SYS}, see datasheet
System Clock Source	PLL locked (intenal clock source)	PLL locked (crystal/ CLKIN1)
PLL	on	on
PLL Power Regulator	on	on
internal clock	on	on
VCO	on	on
Wakeup clock	default	default
High Precision Oscillator	off	on, if crystal; off, if CLKIN1
VCOLCK emergency	-	on
System		
Peripherals	default	default
GSC Mode	Normal	Normal
Traps/user resets/ Interrupts	enabled	enabled
Code Location	Flash	Flash
Wakeup by External Service Request	-	-
Wakeup by Wakeup Timer	-	-
Wakeup by System Timer	-	-
Memories		
Flash	on	on
SBRAM	on	on
PSRAM	on	on
other RAMs	on	on
Power System		

Table 4-2 Properties of Active Modes (cont'd)

Attribute	Base Mode	Normal Operation Mode
Supply Watchdog	on	on
High Precision Bandgap	on	on
PVC_M	on	on
DMP_M	1.5 V HP Bandgap	1.5 V HP Bandgap
CC_M	on	on
PVC_1	on	on
DMP_1	1.5 V HP Bandgap	1.5 V HP Bandgap

4.4.2 Power-saving Modes

See **Table 4-1** (**Mode Property Table Explanation**) for details regarding the attributes or the values of the table below.

Table 4-3 Properties of Power-saving Modes

Attribute	Stopover Mode	(FSM) Standby Mode
Clock System		
Frequency [MHz]	f_{WU}, see datasheet	f_{WU}, see datasheet
System Clock Source	Wakeup clock	Wakeup clock
PLL	off	off
PLL Power Regulator	off	off
internal clock	off	off
VCO	off	off
Wakeup clock	on	on
High Precision Oscillator	configurable	off
VCOLCK emergency	off	off
System		
Peripherals	off	off
GSC Mode	Clock-off	Clock-off
Traps/user resets/ Interrupts	disabled	disabled
Code Location	PSRAM	SBRAM[1)]
Wakeup by External Service Request	yes	yes
Wakeup by Wakeup Timer	yes	yes
Wakeup by System Timer	yes	yes
Memories		
Flash	off	off
SBRAM	on	on
PSRAM	on	off
other RAMs	on	off
Power System		
Supply Watchdog	on	configurable

Table 4-3 Properties of Power-saving Modes (cont'd)

Attribute	Stopover Mode	(FSM) Standby Mode
High Precision Bandgap	configurable	off
PVC_M	on	on
DMP_M	1.5 V LPR	ULPEVR[2], if selected; 1.5 V LPR, otherwise
CC_M	on	off
PVC_1	on	off
DMP_1	1.5 V LPR, clock off	0 V, clock off

1) Copy to PSRAM required for execution.

2) ULPEVR offers the most power saving.

4.4.3 Cyclic Wakeup Modes

See **Table 4-1** (**Mode Property Table Explanation**) for details regarding the attributes or the values of the table below.

Table 4-4 Properties of Wakedup Modes

Attribute	Stopover Wakedup Mode	Fast Startup Mode
Clock System		
Frequency [MHz]	f_{WU}, see datasheet	f_{INT}, see datasheet
System Clock Source	Wakeup clock	internal clock
PLL	off	on
PLL Power Regulator	off	on
internal clock	off	on
VCO	off	off
Wakeup clock	on	on
High Precision Oscillator	configurable	off
VCOLCK emergency	off	off
System		
Peripherals	application specific	application specific
GSC Mode	Clock-off[1)]	Clock-off[1)]
Traps/user resets/ Interrupts	disabled	disabled
Code Location	PSRAM	PSRAM
Wakeup by External Service Request	-	-
Wakeup by Wakeup Timer	-	-
Wakeup by System Timer	-	-
Memories		
Flash	off	off
SBRAM	on	on
PSRAM	on	copied from SBRAM
other RAMs	on	on, but not initialised
Power System		
Supply Watchdog	on	configurable

Table 4-4 Properties of Wakedup Modes (cont'd)

Attribute	Stopover Wakedup Mode	Fast Startup Mode
High Precision Bandgap	configurable	off
PVC_M	on	on
DMP_M	1.5 V LPR	1.5 V LPR
CC_M	on	off
PVC_1	on	on
DMP_1	1.5 V LPR	1.5 V LPR

1) Must be changed in order to use peripherals.

5 Operating Mode Transitions

A sequence of operations, which leads from one Operating Mode of the XC2000/XE166 to another, is named “Operating Mode Transition” in this document. An Operating Mode Transition may comprise a software flow as well as sequential hardware activities, such as performed by the **PSC**.

The sequences in this document have been designed with a focus on achieving a maximum of robustness against failures, a minimum of interaction of features for the sake of the reliability of the user application, minimum transition times and minimum PSRAM memory requirements.

When being implemented in the user application flow, any change in the sequences, e.g. towards run time optimization, may have a negative impact on the robustness of the transition sequence itself, or even on the stability of the target Operating Mode. Thus, any change should only be done in case the given flow exhibits significant restrictions, e.g. regarding system performance. If one is unsure whether a change might imply any risk, please contact the local **FAE**.

5.1 Overview

Prior to a detailed discussion of the Operating Mode transitions, a transition flow is outlined in **Figure 5-1**.

To increase robustness, the transitions and functions perform a lot of hardware status checks. If any problem is detected, an error code is returned. In case of PSRAM execution, a user function (**HANDLE_ERROR_PS** with configurable name) is called. The user can decide how to handle the error (e.g. perform a reset).

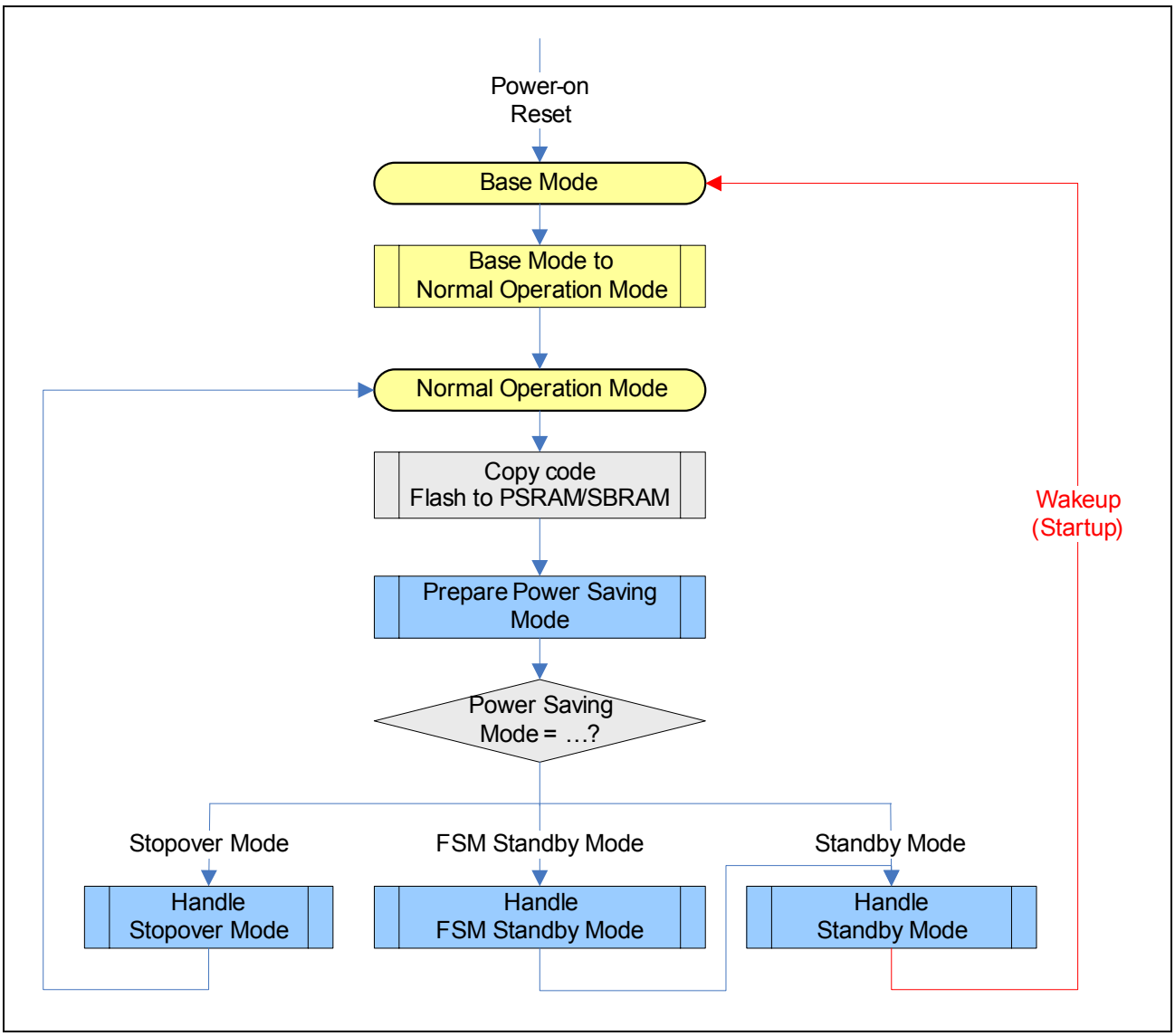

Figure 5-1 Operating Mode Transitions

5.2 Base Mode to Normal Operation Mode

After reset, the system will enter Normal Operation Mode via transition Base Mode to Normal Operation Mode. Both modes use the PLL as clock source for the system clock. One of the following PLL input clock source options can be selected for Normal Operation Mode:

- Crystal/external clock at pin XTAL
- External clock at pin CLKIN1
- Internal clock provided by the Trimmed Current Controlled Clock source

The frequency of the clock source is configured by the user.

The transitions to Normal Mode switches from a PLL configuration where the system frequency is provided by the PLL, locking on the internal clock, to a target clock mode, where the system clock is provided by an other setup of the PLL (PLL input clock source, N and P dividers and VCO band). With respect to the PLL input clock source used in Normal Operation Mode two different transtions from Base Mode to Normal Operation Mode are described in the following.

5.2.1 Base Mode to Normal Operation Mode with External Clock

Base Mode and Normal Operation Mode use the PLL as system clock source in different configurations: PLL clock source, VCO band and PLL divider settings. The PLL input clock is switched from the internal clock to an external clock: crystal or direct clock at CLKIN1. Because the target mode depends on an external clock source (crystal or direct clock), the transition configures and activates a clock exception handling.

In case of a crystal the high precision oscillator OSC_HP must be enabled.

The function **EnableHighPrecOsc** can be called at an early stage of initialization to reduce the waiting time for stable oscillation.

Flow

1. Prepare the external clock source used as PLL input clock:
   – In case of a crystal, the crystal oscillator is started by enabling the high precision oscillator. Stable oscillation of the high precision oscillator is checked as late as possible (for timing reasons).
   – If CLKIN1 is used then select CLKIN1 as clock source for the PLL. It is not required that the high precision oscillator is enabled.
2. To allow the VCO settings to be changed, the VCO is bypassed (PLL Prescaler mode). The internal clock is used as system clock.
3. Set PLL VCO dividers N, P, K2 for 10 MHz. VCO band 1 is selected (in Base Mode VCO band 0 is selected).
4. The input clock for the PLL has to be changed to the external clock source.
   To avoid glitches on the system clock the VCO is temporarily used as system clock.

The input of the VCO is disconnected. During this time the VCO frequency decreases to its base frequency while the maximum frequency is in an acceptable range.

5. In case of the usage of a crystal, wait unitl crystal oscillation is stable. At this point the stable oscillation of the crystal is required to continue. To achieve short transitions times the crystal should be started as early as possible before the transition to Normal Operation Mode starts.
6. The clock at the output of the K2-divider is now switched to the PLL output.
7. The input of the PLL can be switched without any impact on the system frequency, while the system is running on a relatively stable clock
8. The oscillator clock is connected to the VCO by releasing the VCO disconnect. The VCO will then lock after a while; meanwhile, the system clock frequency is smoothly increasing. To avoid system frequency overdrive when reaching the lock state, the K2-divider value has been configured to deliver 10 MHz in the lock state.
9. With the PLL being in a lock state, the clock system emergency cause (e.g. crystal break) handling for PLL lock operation is now activated.
10. The system frequency is configured to the target frequency by stepwise decreasing the K2-divider value.

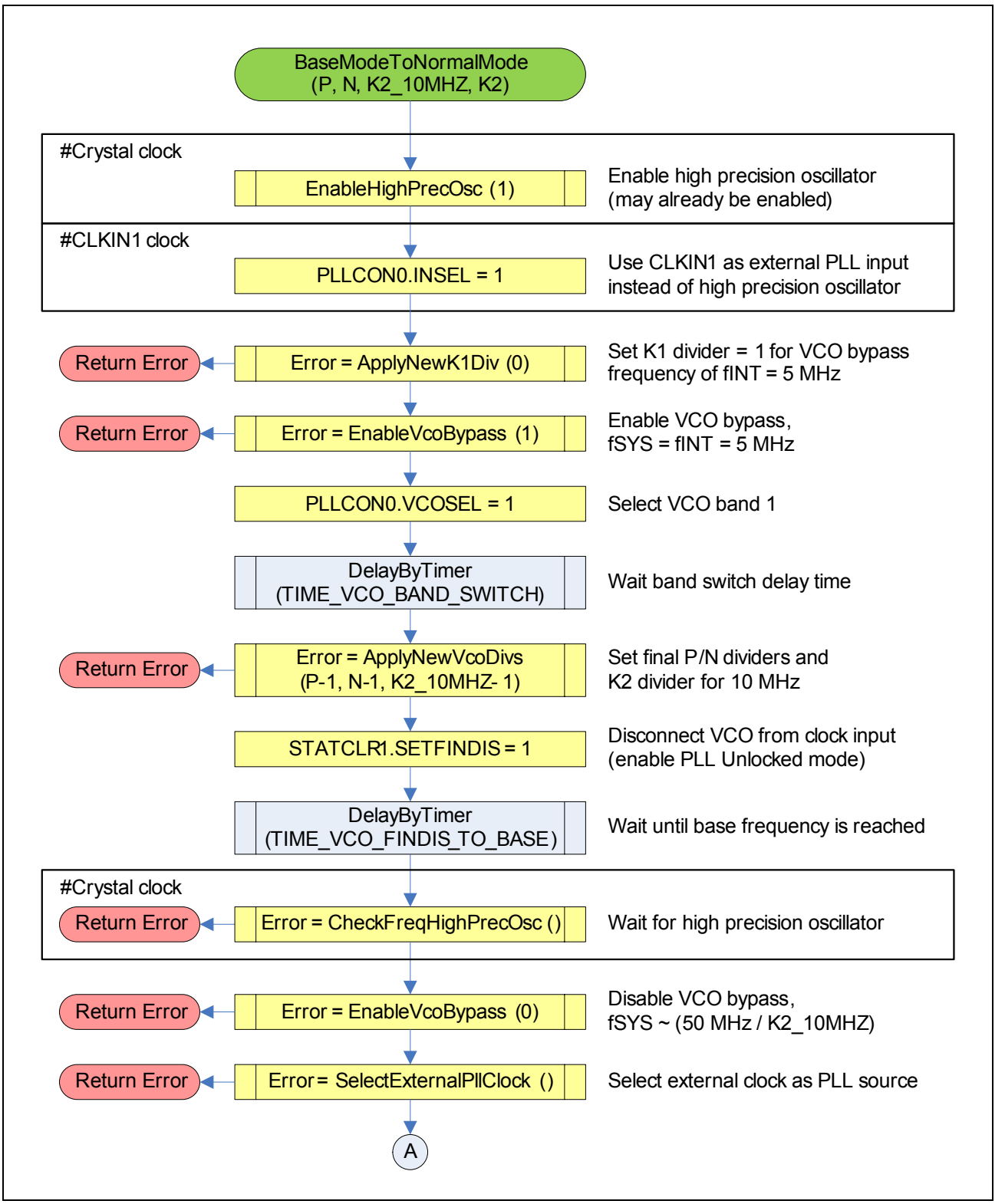

Figure 5-2 Flow-chart BaseModeToNormalMode (External Clock) (1)

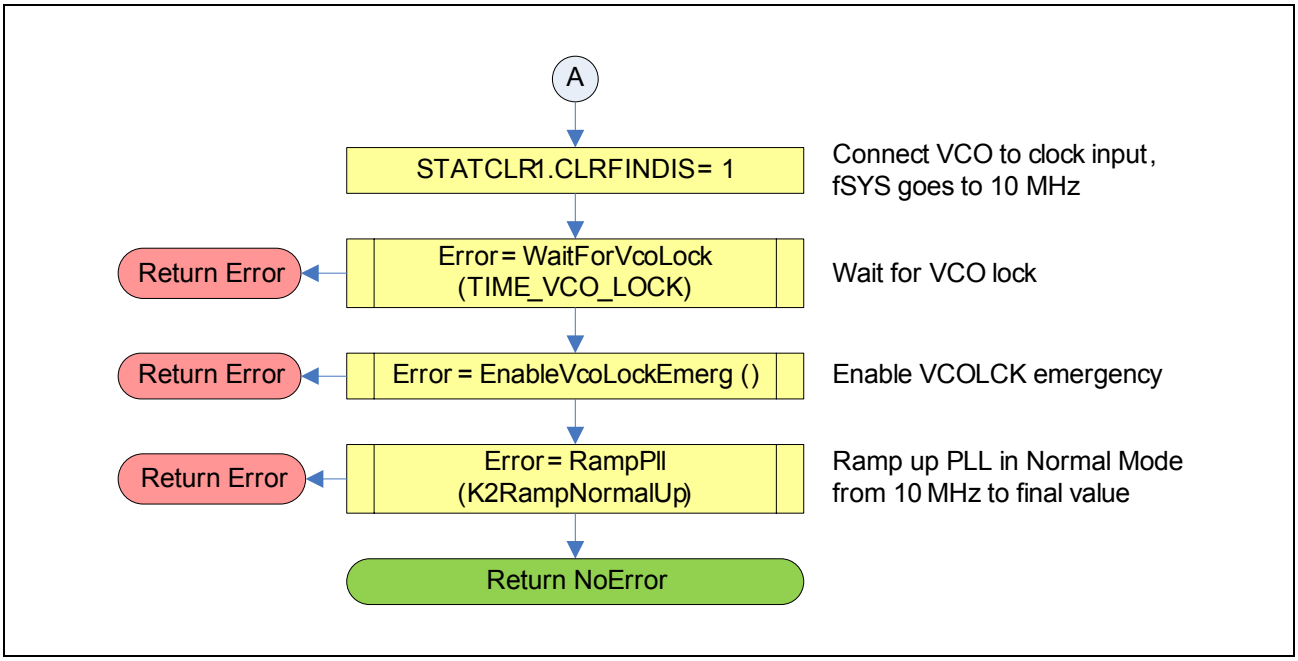

Figure 5-3 Flow-chart BaseModeToNormalMode (External Clock) (2)

5.2.2 Base Mode to Normal Operation Mode with Internal Clock

Base Mode and Normal Operation Mode use the PLL as system clock source in different configurations: VCO band and PLL divider settings. The internal clock is used as PLL input clock in both modes. Because the target mode does not depend on an external clock source (crystal or CLKIN1) a clock exception handling is not required.

Flow

1. To allow the VCO settings to be changed, the VCO is bypassed (PLL Prescaler mode). The internal clock is used as system clock source.
2. Set PLL VCO dividers N, P, K2 for 10 MHz. VCO range 1 will be used.
3. The clock at the output of the K2-divider is now switched to the PLL output
4. To avoid system frequency overdrive when reaching the lock state, the K2-divider value has been configured to deliver 10 MHz in the lock state
5. With the PLL being in a lock state, the clock system emergency handling for PLL lock operation is now activated.
6. The system frequency is configured to the target frequency by stepwise decreasing the K2-divider value

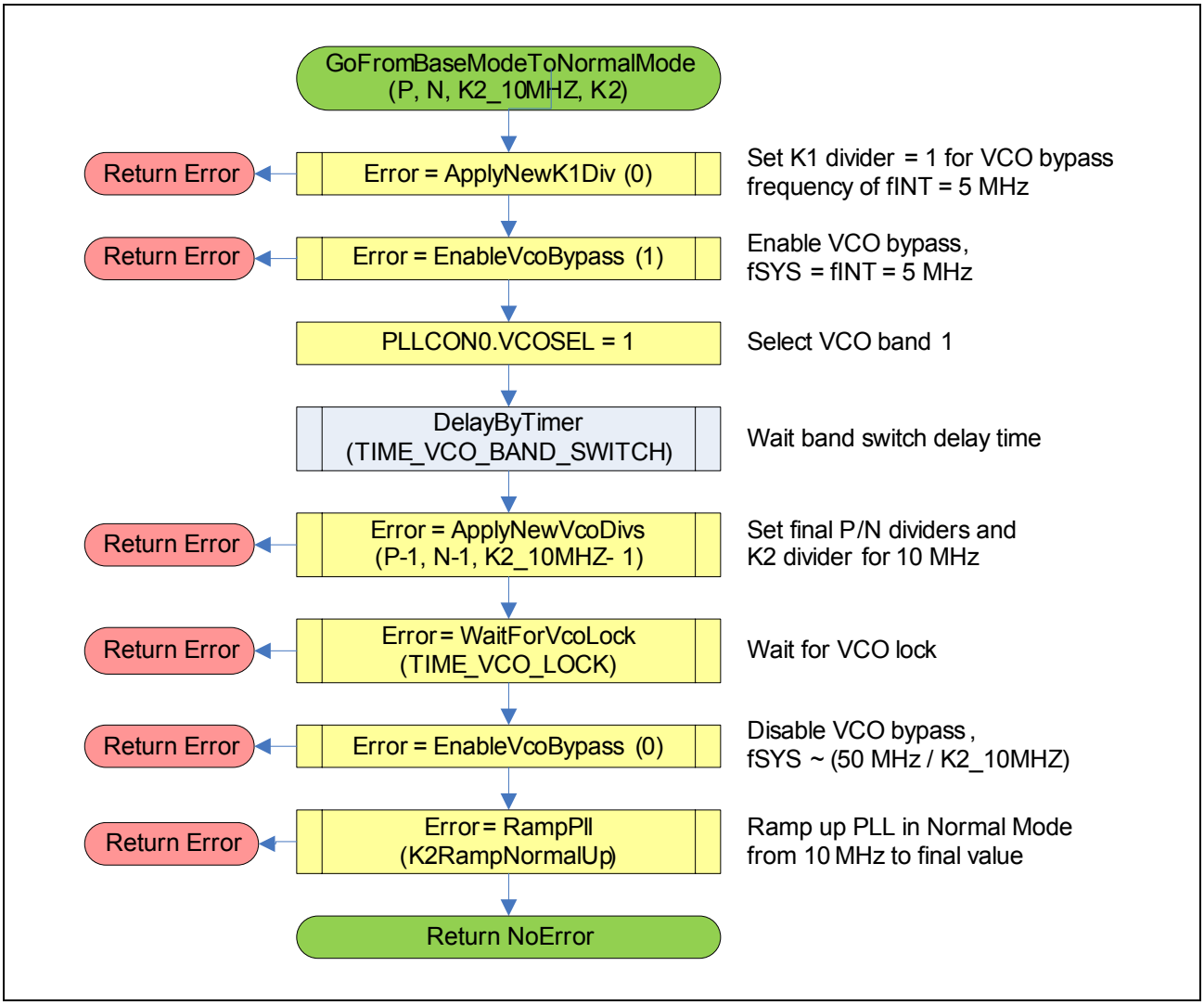

Figure 5-4 Flow-chart BaseModeToNormalMode (Internal Clock)

5.3 Enter and Exit a Power Saving Mode

The following power saving modes are supported:

- **Stopover Mode**
- **Standby Mode**
- **FSM Standby Mode**

The power saving modes are entered from Normal Operation Mode. The transitions to enter a power saving, to handle a wakeup and to exit a power saving mode have a similiar structure which will be outlined in the following. Coming from a general structure this eases the understanding of the differences in the transitions for the various power saving modes.

An overview of the transitions from **Normal Operation Mode** to one of the power saving modes is shown in **Figure 5-1 Operating Mode Transitions**.

5.3.1 Wakeup Configuration

A wakeup from a power saving mode can be triggered by one or several events:

- Wakeup Timer (WUT)
- System Timer (STM)
- External Service Request (ESR) pin(s)

The wakeup trigger must be enabled in SEQCON register.

Wakeup Timer

If the Wakeup Timer is to be used for wakeup from a power saving mode, it must be properly configured prior to entering the power saving mode:

- Wakeup Timer interval must be configured in register WUTREL
- Auto-stop on trigger for constant sleep time or no auto-stop for constant wakeup period

Note: The Wakeup Timer trigger for the GSC must not be used in conjunction with power saving modes.

System Timer

The System Timer is typically used for real-time clock purposes. Therefore, it is assumed that this timer is already used by the application. This means the relevant parameters (clock selection, clock divider and interval) are set, the timer is running and the DMPTMIT request flag is cleared.

External Service Requests

If External Service Requests inputs are to be used as trigger for wakeup from a power saving mode, the respective ESR logic must be properly configured prior to entering the power saving mode.

- The respective ESRCFG register must be configured regarding edge (rising/falling) and type of edge detection (synchronous with clock/asynchronous without clock).
- Reset via External Service Requests must be disabled.

5.3.2 Preconditions for Power Saving Mode

In Normal Operation Mode the application has to ensure that the preconditions mentioned in , e.g. code copied from Flash to PSRAM and SBRAM, are fulfilled before requesting a power saving mode.

Location of Code for Power Saving Mode

During the transition to a power saving mode, the code execution location must be switched from Flash to PSRAM, since the Flash will be switched off. Furthermore, after wakeup from Stopover Mode and FSM Standby Mode the code will be executed from PSRAM, too.

For power saving modes, some functions must be executed from PSRAM. These code sections must be located in special areas, e.g. via compiler directives, and they have to be copied from Flash to PSRAM via function **CopyWords**.

If interrupts or traps may occur during code execution from PSRAM, the corresponding vectors must be copied from Flash to PSRAM additionally, using function **CopyVectorToPsram**.

The code to be executed in Fast Startup Mode after wakeup from FSM Standby Mode is taken from Standby RAM (SBRAM). The SBRAM is located in power domain DMP_M and so it is powered during FSM Standby Mode. Because the SBRAM does not support code-execution - the code has to be copied into and started from the PSRAM by the startup software after wakeup. Again, these code sections must be located in special areas, e.g. via compiler directives (see compiler documentation), and they have to be copied from Flash to SBRAM using function **WriteToSbram**.

For further details see **Section 9.6.3 Memory Copy**.

System Setup

Before requesting a power saving mode used peripherals should be switched off by the application to save current and to ensure a correct deinitialisation of the peripherals.

Inputs/outputs should be brought to the state with the lowest power consumption.

Interrupts should be diabled as far as possible.

Register protection has to be disabled.

During entering or exiting a power saving mode, the internal watchdog is not served.

5.3.3 Prepare Power Saving Mode

Before a power saving mode can be entered, some preparation (configuration of the wakeup sources and wakeup triggers, setup of the clock and power system) is necessary. The steps to be done are equal for all power saving modes. The detailed flows are described in **Section 5.4 Prepare Power Saving Mode**.

After the preparation of the power saving mode the Flash is disabled and the code is executed from PSRAM. The internal clock f_{INT} is used as system clock.

5.3.4 Handle Power Saving Mode

An overview of the sequences to enter and exit the power saving modes is shown in **Figure 5-5**. The detailed flows for the different power saving modes are different and will be described for

- Stopover Mode in **Section 5.5**
- Standby Mode in **Section 5.6**
- FSM Standby Mode in **Section 5.7**

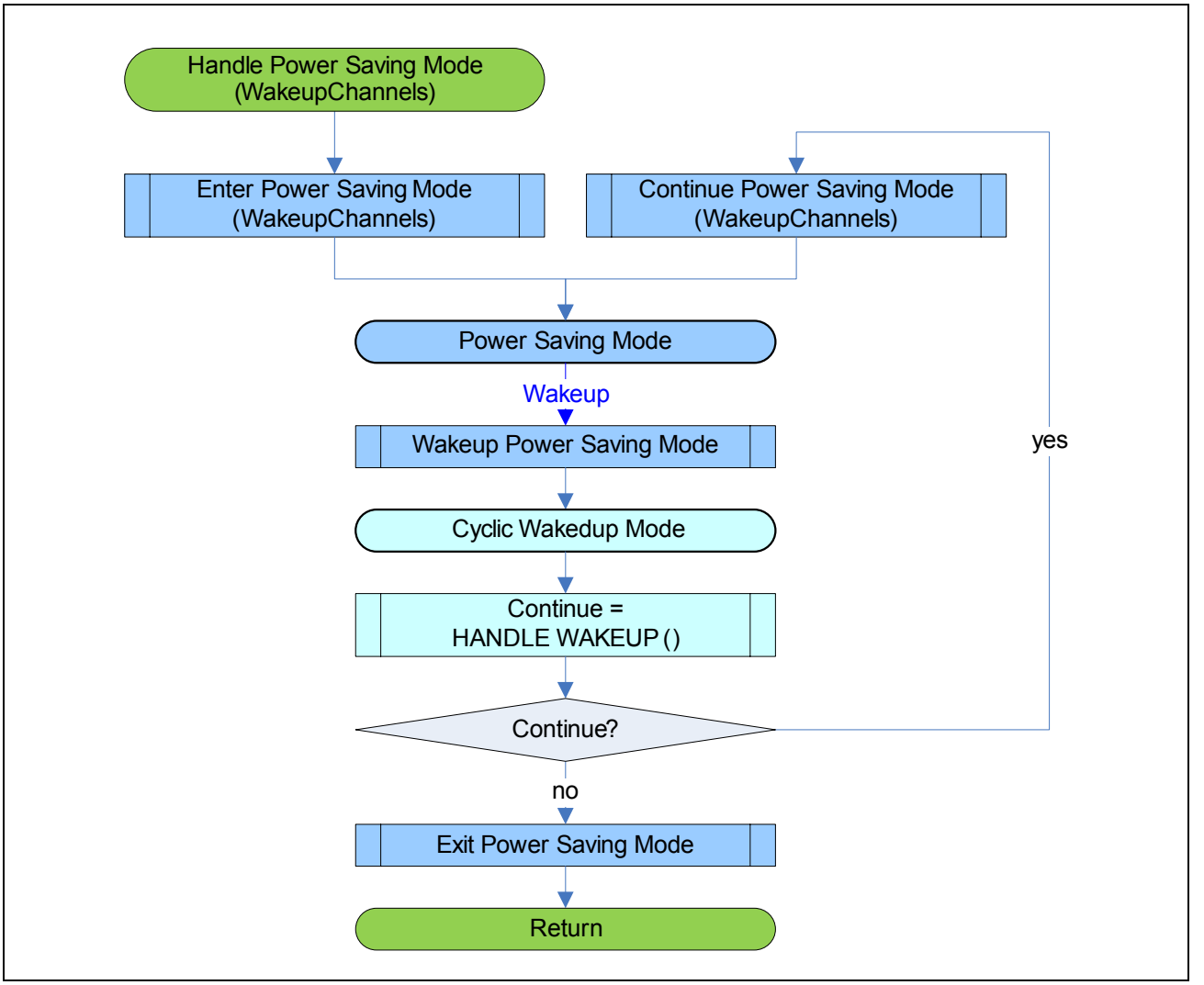

Figure 5-5 Flow-chart HandlePowerSavingMode

5.3.4.1 Enter Power Saving Mode

After the wakeup clock was selected as system clock and the power transistion sequences A and B are configured the power saving mode is reached via the hardware stepping mechanism of the PSC. The sequences are mode dependent. For Stopover Mode the clock in power domain DMP_1 is stopped, and for Standby Mode and FSM Standby Mode the power domain DMP_1 is powered down completely (Standby Mode and FSM Standby Mode) and the contents of the RAM memories (except SBRAM) will be lost.

5.3.4.2 Wakeup from Power Saving Mode

In case of a wakeup trigger the hardware stepping mechanism of the PSC executes the transition sequence B. The sequence was configured before entering the power saving

mode. For Stopover Mode the clock in power domain DMP_1 is enabled, and for Standby Mode and FSM Standby Mode the power domain DMP_1 is ramped-up.

5.3.4.3 Handle Wakeup from Power Saving Mode

After wakeup the XC2000/XE166 is not started completly, the Flash is still disabled and the code is executed from PSRAM. The wakeup clock (Stopover Mode) or the internal clock (Fast Startup Mode) is used as system clock.

An application specific function can perform short actions. The application function decides if the power saving mode shall be continued or not. The function can determine and clear the wakeup source or change the system clock with dedicated functions.

Determine Wakeup Source

After a wakeup the application can determine with function **GetWakeupSrc** the wakeup source if multiple wakeup sources are enabled. Then it is recommended to clear the wakeup source request calling function **ClearWakeupSrc**.

5.3.4.4 Continue Power Saving Mode

If the application specific function decided to continue the power saving mode the power saving mode is entered again. The flow to continue is similiar to the flow to enter the power saving mode.

5.3.4.5 Exit Power Saving Mode

If the application specific function decided to leave the power saving mode the power saving mode is exit. Normal Operation Mode is resumed for Stopover Mode or Standby Mode enter in case of Fast Startup Mode.

5.3.5 Handle Error

An error detected during execution in PSRAM can be handled with the user function **HANDLE_ERROR_PS**.

5.4 Prepare Power Saving Mode

Before a power saving mode can be entered the actions listed in **Preconditions for Power Saving Mode** have to be done. Further preparation (configuration of the wake-sources and wakeup triggers, setup of the clock and power system) is necessary.

The wakeup trigger sources are configured. Interrutps and functional resets are disabled to avoid any kind of disruption.

The system clock frequency is reduced to f_{INT} = 5 MHz. Up to this point, the CPU is still executing code from Flash. After the code location has been switched from Flash to PSRAM, the Flash is powered-down.

The power system is still configured for high performance. The next step is to disable the HP Bandgap and to use the LPR as reference.

The location of the executed code is changed from Flash to PSRAM. Target is to execute as many instructions as possible from Flash with highest possible system clock frequency. The preparation is done in two steps: The first step is executed from Flash, the second step is executed from PSRAM in which the Flash is disabled.

Transitions

Entry: Normal Operation Mode.

Exit: Stopover Mode, Standby Mode or FSM Standby Mode.

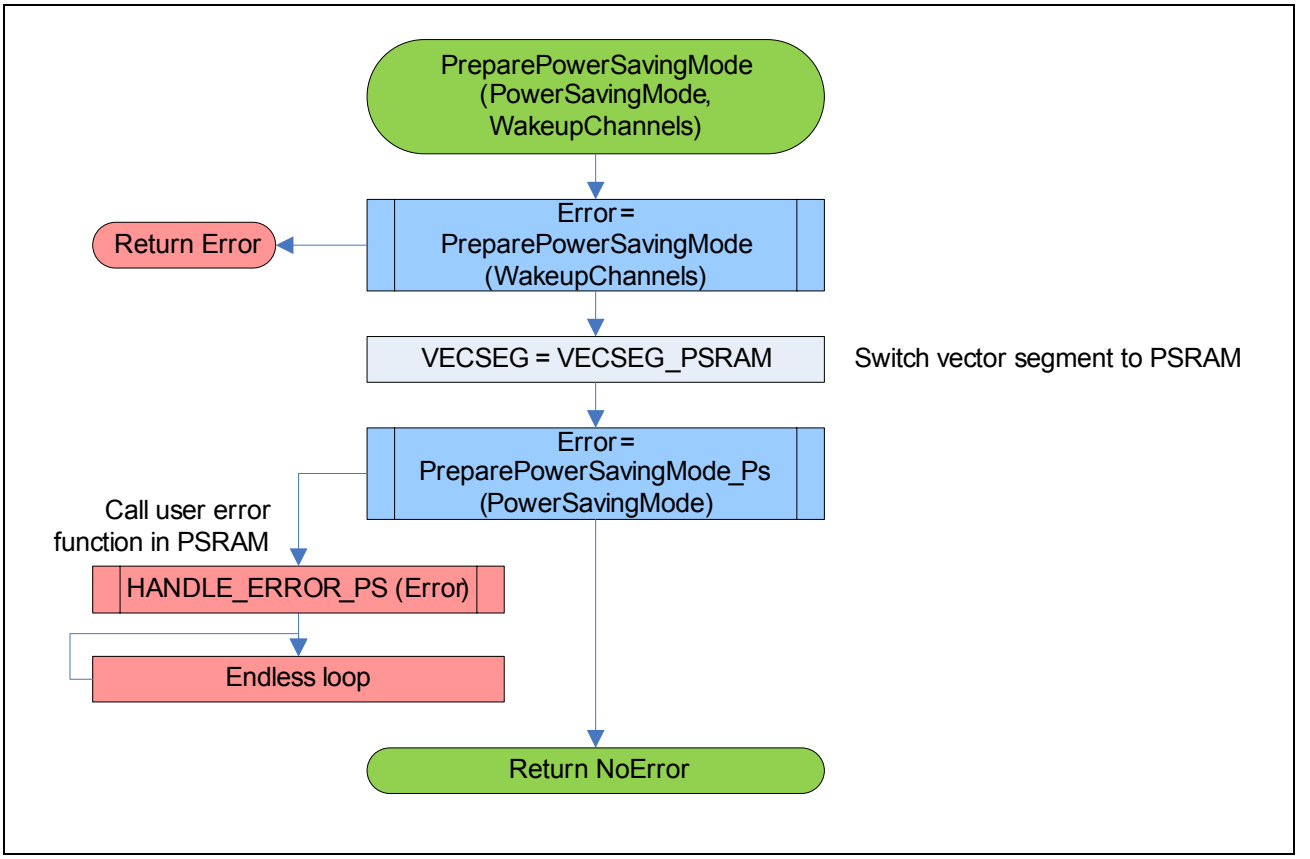

Figure 5-6 Flow-chart PreparePowerSavingMode

5.4.1 Wakeup Configuration

A wakeup from a power saving mode can be triggered by one or several events:

- Wakeup Timer (WUT)
- System Timer (STM)
- External Service Request (ESR) pin(s)

The wakeup trigger must be enabled in SEQCON register.

Wakeup Timer

If the Wakeup Timer is to be used for wakeup from a power saving mode, it must be properly configured prior to entering the power saving mode:

- Wakeup Timer interval must be configured in register WUTREL
- Auto-stop on trigger for constant sleep time or no auto-stop for constant wakeup period, only with WUTREL register

Note: The Wakeup Timer trigger for the GSC must not be used in conjunction with power saving modes.

System Timer

The System Timer is typically used for real-time clock purposes. Therefore, it is assumed that this timer is already used by the application. This means the relevant parameters (clock selection, clock divider and interval) are set, the timer is running and the DMPTMIT request flag is cleared.

External Service Requests

If External Service Requests inputs are to be used as trigger for wakeup from a power saving mode, the respective ESR logic must be properly configured prior to entering the power saving mode.

- The respective ESRCFG register must be configured regarding edge (rising/falling) and type of edge detection (synchronous with clock/asynchronous without clock).
- Reset via External Service Requests must be disabled.

The configuration is done in the following transition.

5.4.2 Prepare Power Saving Mode (Flash)

In the first step for the preparation of the power saving mode the code is executed from Flash.

Flow

1. The following steps are performed at system frequency of Normal Operation Mode.
2. Ultra Low Power EVR is enabled if the power saving mode Standby Mode or FSM Standby Mode is requested and if configured to be used.
3. The actions to be performed by the power transition is programmed to the sequence registers. This transition changes the supply settings from HP Bandgap based to LPR based, using the PSC power transition feature to save the base load current of the HP Bandgap.The sequence will be executed in transition **Prepare Power Saving Mode (PSRAM)**.
4. Possible wakeup triggers are enabled. This can be one or more of the ESR pins, the Wakeup Timer, or the STM.
   In case the wakeup transition leads to **Fast Startup Mode** only Wakeup Timer is allowed as wakeup trigger.
   Wakeup Timer auto-stop and divider parameters are set. Auto-stop on trigger selects between a constant sleep time or a constant wakeup period.
5. Clock-off Mode is requested to shut down the peripherals except Flash. It is done here to reduce the time for the shutdown and to save code space in the PSRAM. (GSC is configured to allow a Clock-off broadcast)
6. Because the following actions trigger hardware sequences, any kind of disruption such as user resets are disabled at this point.
   For Stopover Mode, IEN and reset control registers RSTCONx are saved.

7. The system frequency is reduced to 5 MHz (= f_{INT} of the internal clock). After the internal clock has been selected as PLL input clock the PLL is used in Prescaler Mode providing 5 MHz.
   In case a crystal or CLKIN1 is used as clock source the wakeup clock is used as intermediate clock to switch form the external to the internal clock.
8. The Wakeup clock is configured with its maximum frequency.
9. If a crystal is used as PLL input and if it is not necessary that the crystal in continously running the crystal can be disabled.
10. The PLL VCO is no longer required and is disabled.

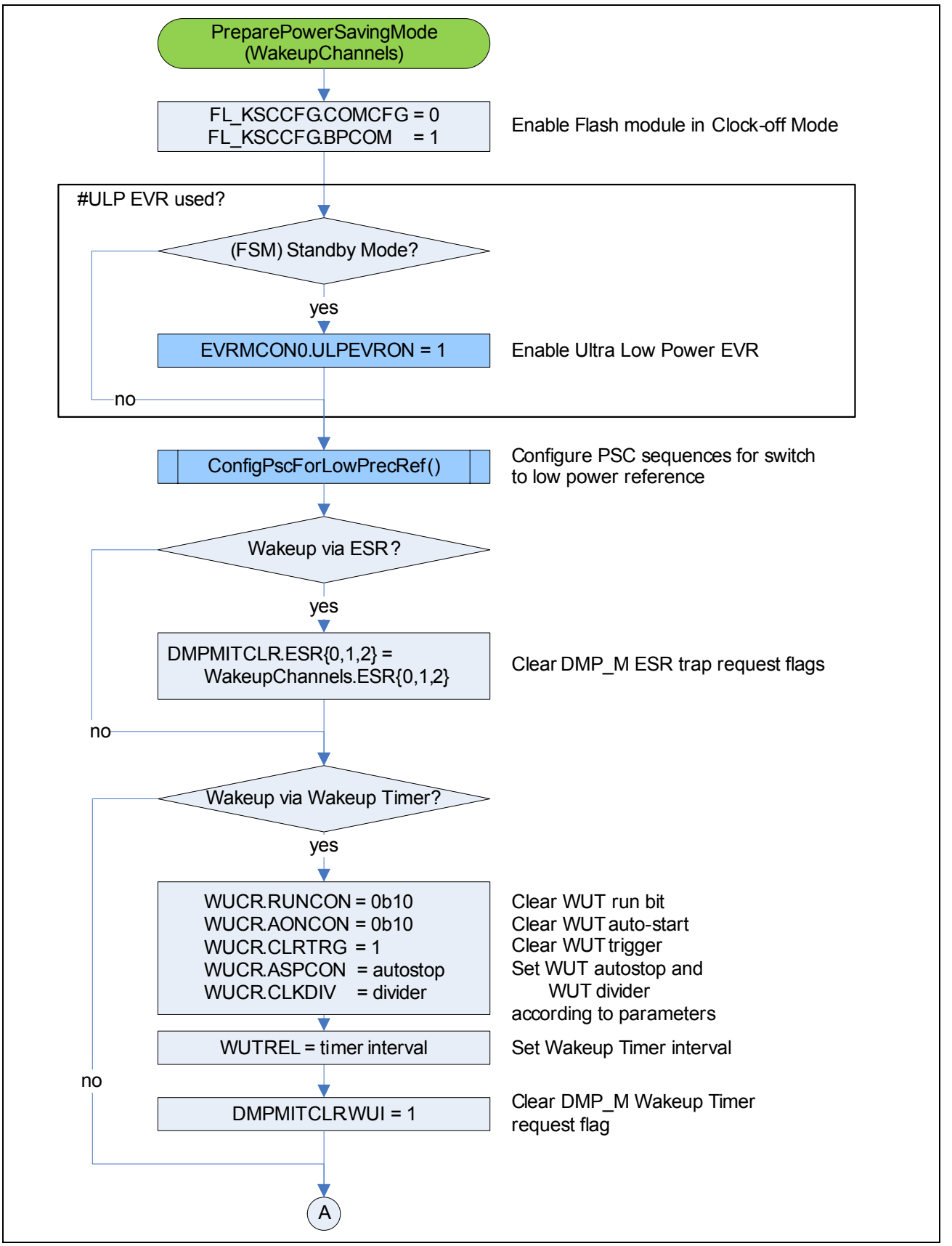

Figure 5-7 Flow-chart PreparePowerSavingMode (Flash) (1)

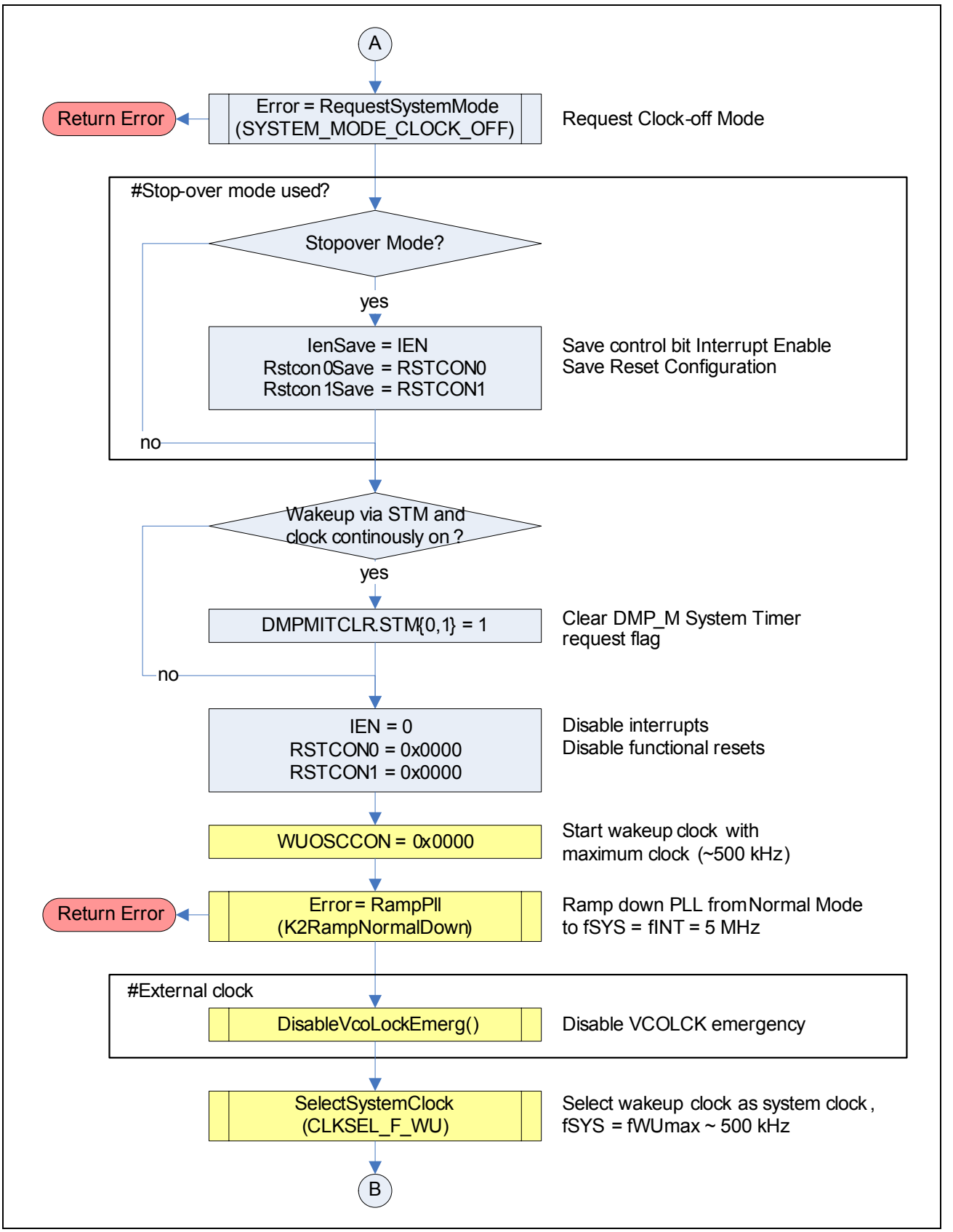

Figure 5-8 Flow-chart PreparePowerSavingMode (Flash) (2)

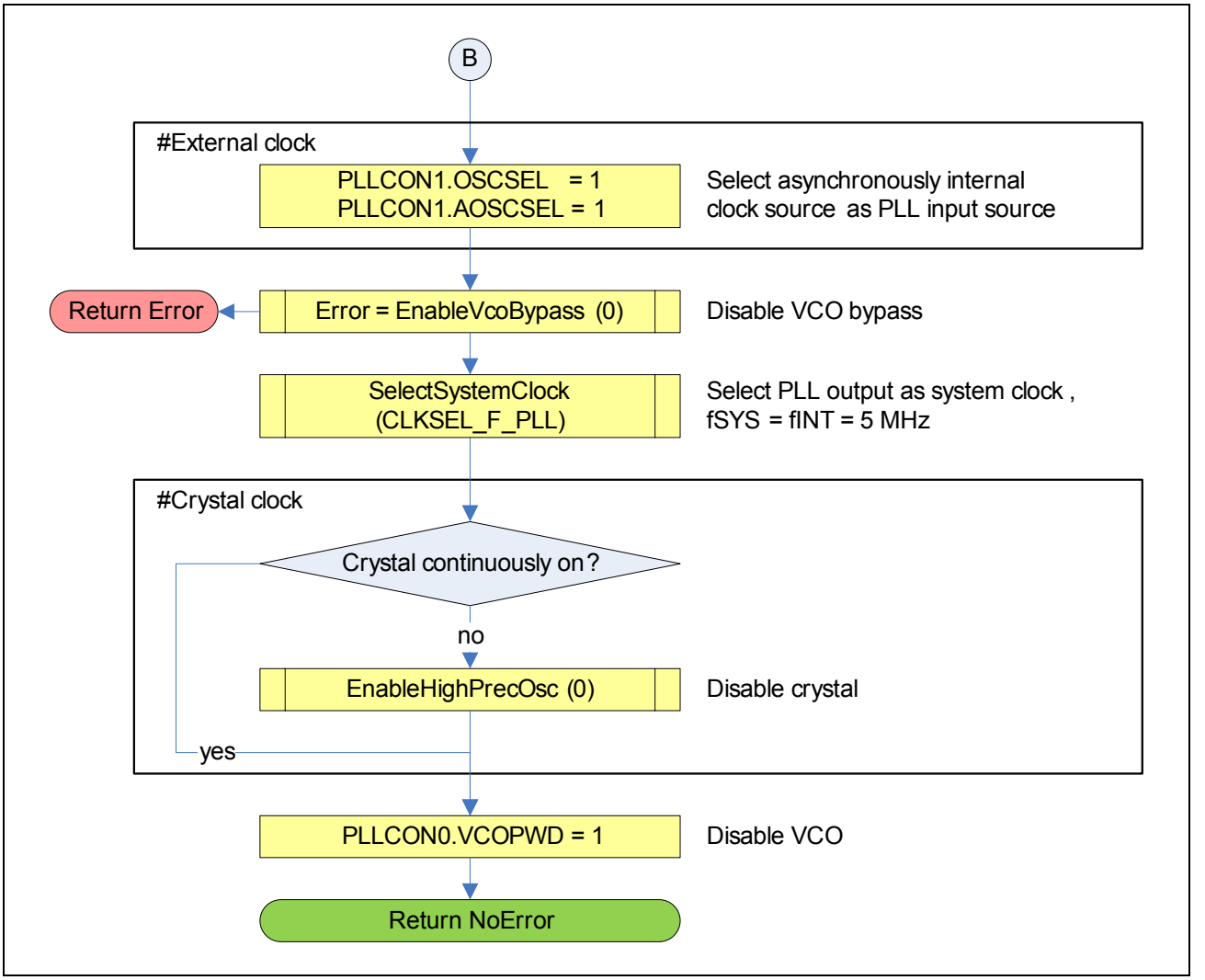

Figure 5-9 Flow-chart PreparePowerSavingMode (Flash) (3)

5.4.2.1 Prepare Power Saving Mode (PSRAM)

The transition continues with the preparation of the power saving mode.

Note: The code for the transition must be located in PSRAM.

Flow

1. Flash trap is disabled.
2. Flash module is disabled.
3. During waiting time for flash to power-down other action are performed for timing reasons
4. Wakeup clock with maximum frequency is used for the following PSC power transitions
5. The CPU triggers and supervises the power sequence A which was already configured in **Prepare Power Saving Mode (Flash)**. The sequence switches to LPR to release the HP Bandgap. To avoid glitches in the logic the system clock is switched off while the reference is switched from HP Bandgap to LPR.
   A dummy sequence B is executed with wakeup clock (configured for maximum clock frequency) during this mode transition, to indicate that a sequence B has been executed (PSCSTAT.LSTSEQ==1), and thereby to force the next sequence to be executed to be a sequence A. This is necessary to prevent a not initialized sequence B to be falsely triggered by a wakeup trigger, before a power saving mode has been entered using sequence A.
   The CPU remains executing code from PSRAM
6. Sequence B is performed with wakeup clock
7. After the sequence has been detected to be ready, and when running on the LPR, the HP Bandgap is switched off
8. Target wakeup clock frequency in power saving mode is set
9. Timer T13 in CCU60 module is stopped and then CCU60 module is disabled

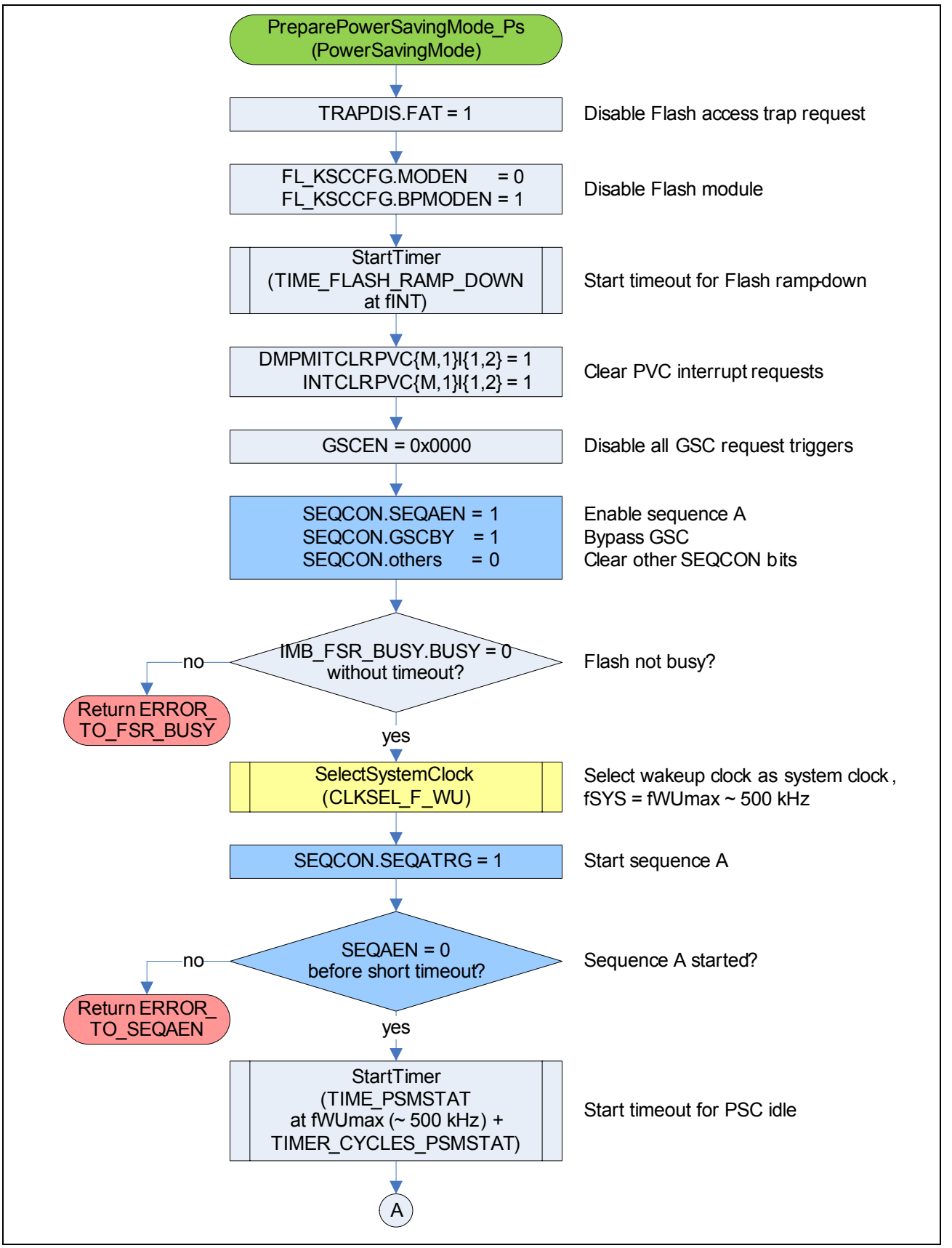

Figure 5-10 Flow-chart PreparePowerSavingMode_Ps (1)

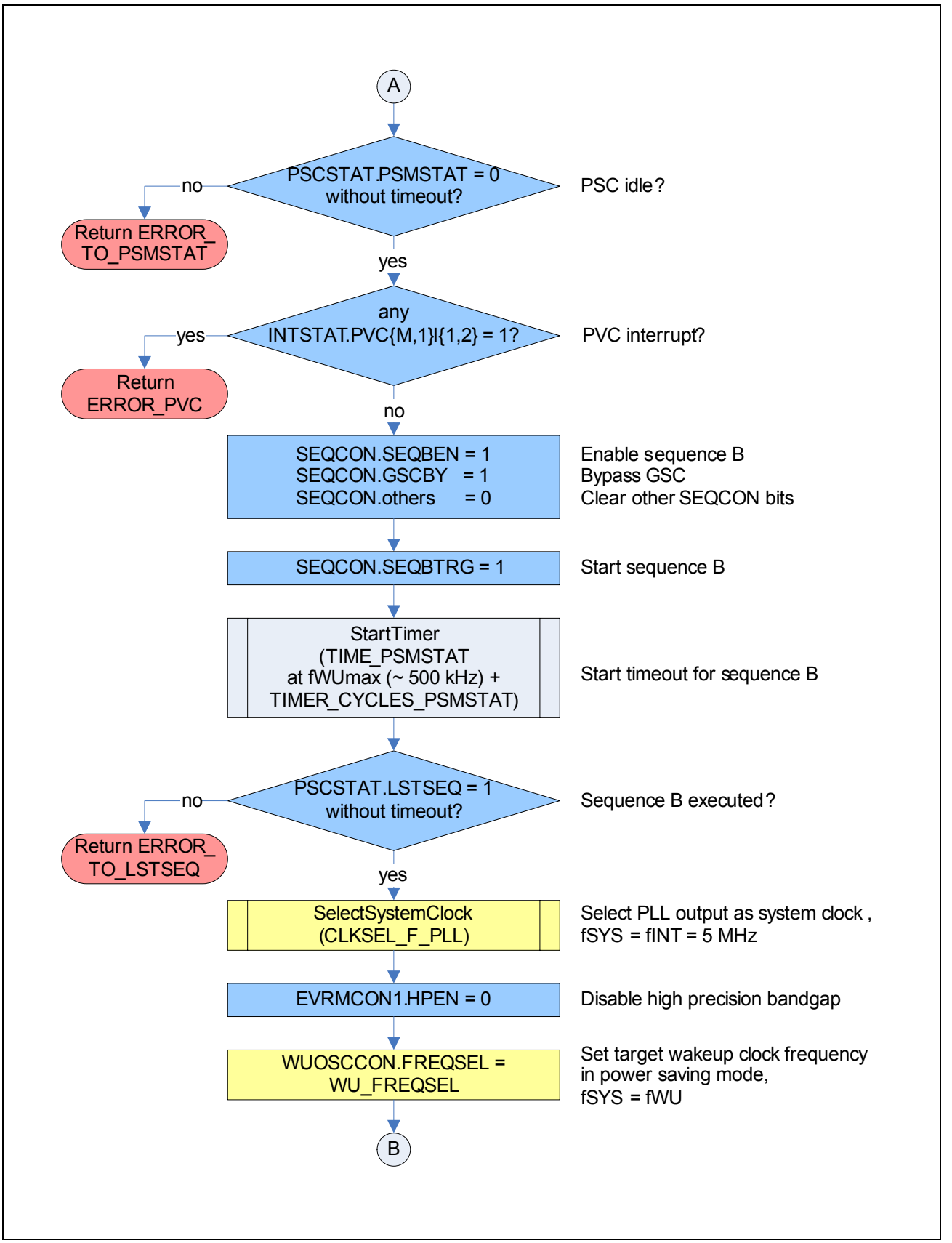

Figure 5-11 Flow-chart PreparePowerSavingMode_Ps (2)

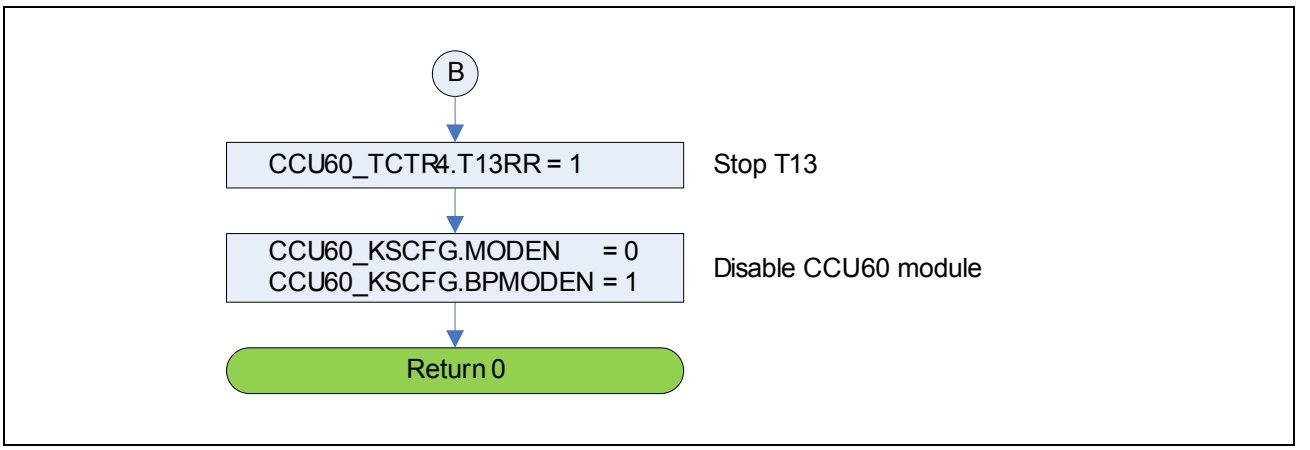

Figure 5-12 Flow-chart PreparePowerSavingMode_Ps (3)

5.5 Handle Stopover Mode

The transition performs the transition or repeated transition to enter Stopover Mode (i.e. clock stop for DMP_1) while code is exectued from PSRAM. After wakeup from Stopover Mode the code is executed from PSRAM. At this point Stopover Mode may be entered again or a transition can be started to return to Normal Operation Mode at the address the clock was stopped.

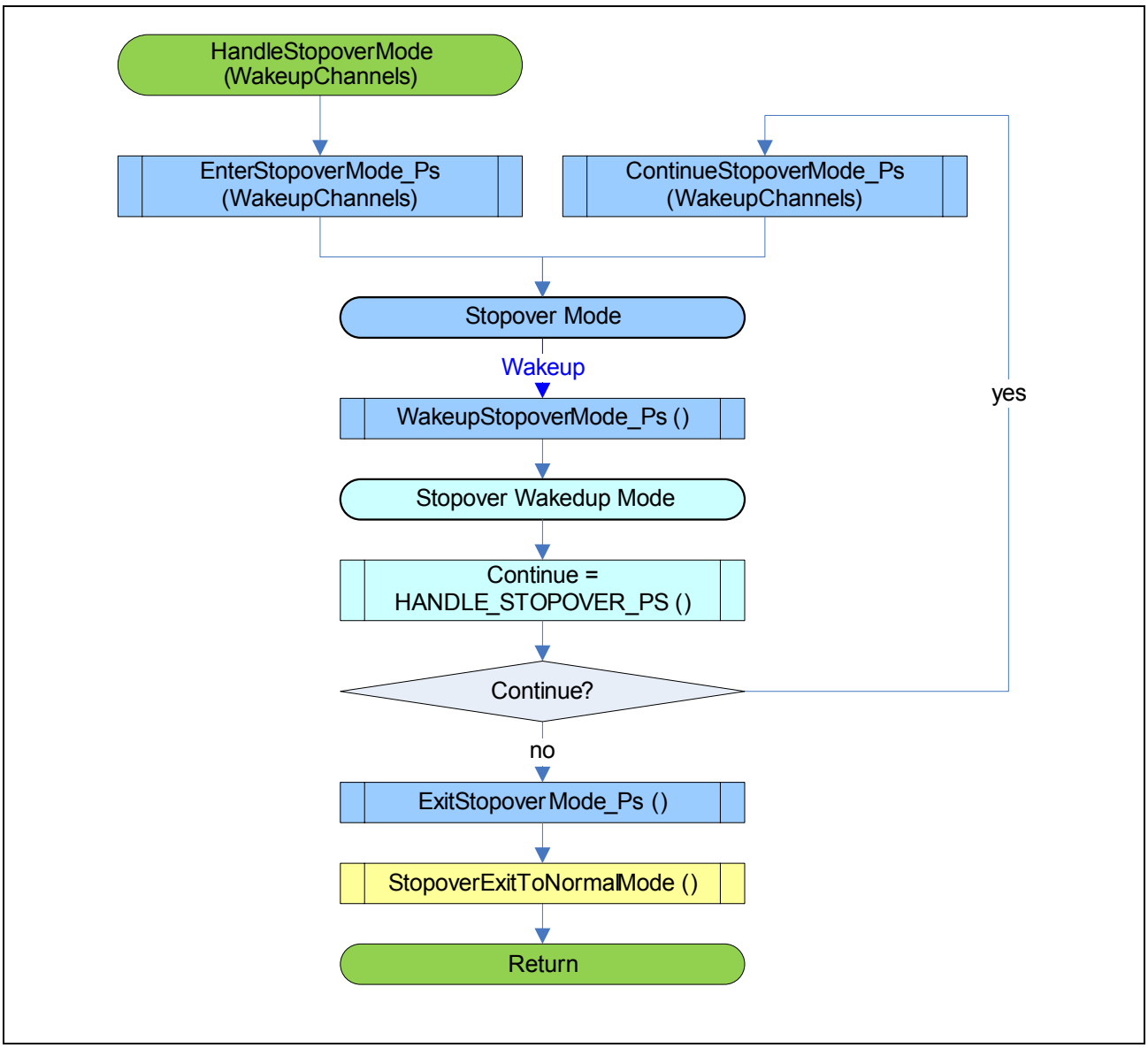

Figure 5-13 Flow-chart HandleStopoverMode

Transitions

Entry: Normal Operation Mode via transition **Prepare Power Saving Mode**.

Exit: Normal Operation Mode via Stopover Wakedup Mode.

5.5.1 Enter Stopover Mode

This transition is used if Stopover Mode is selected as power saving mode. An option is to let have crystal oscillator or external clock permanently on.

Stopover Mode (i.e. clock stop for DMP_1) is entered using the hardware stepping mechanism of the **PSC**, triggered by the CPU, but without applying further changes to the supply system.

Exit of this mode is done using the hardware stepping mechanism of the PSC. An advantage of clocking the wakeup domain DMP_M is that triggering the PSC can be done after a predefined time using the Wakeup Timer (**WUT**). This means, this mode is suitable for a periodical wakeup.

Note: The code for the transition must be located in PSRAM.

Flow

1. In order to stop the system clock in DMP_1, a new power mode sequence has to be executed. Thus, a new transition set is configured, overwriting the settings of the previous power transition A.
2. Wakeup clock is selected as system clock.
3. The PLL is shut down.
4. If the Wakeup Timer is selected as wakeup source then start Wakeup Timer (has no effect on running timer) and clear wakeup trigger. After enabling sequence B and wakeup trigger the wakeup trigger has to be cleared.
5. Sequence A is started to enter Stopover Mode with stopped system clock in power domain DMP_1.
6. While sequence A is running, the CPU is executing code until the system clock is stopped. Therefore, a delay using NOPs is inserted to avoid code execution.

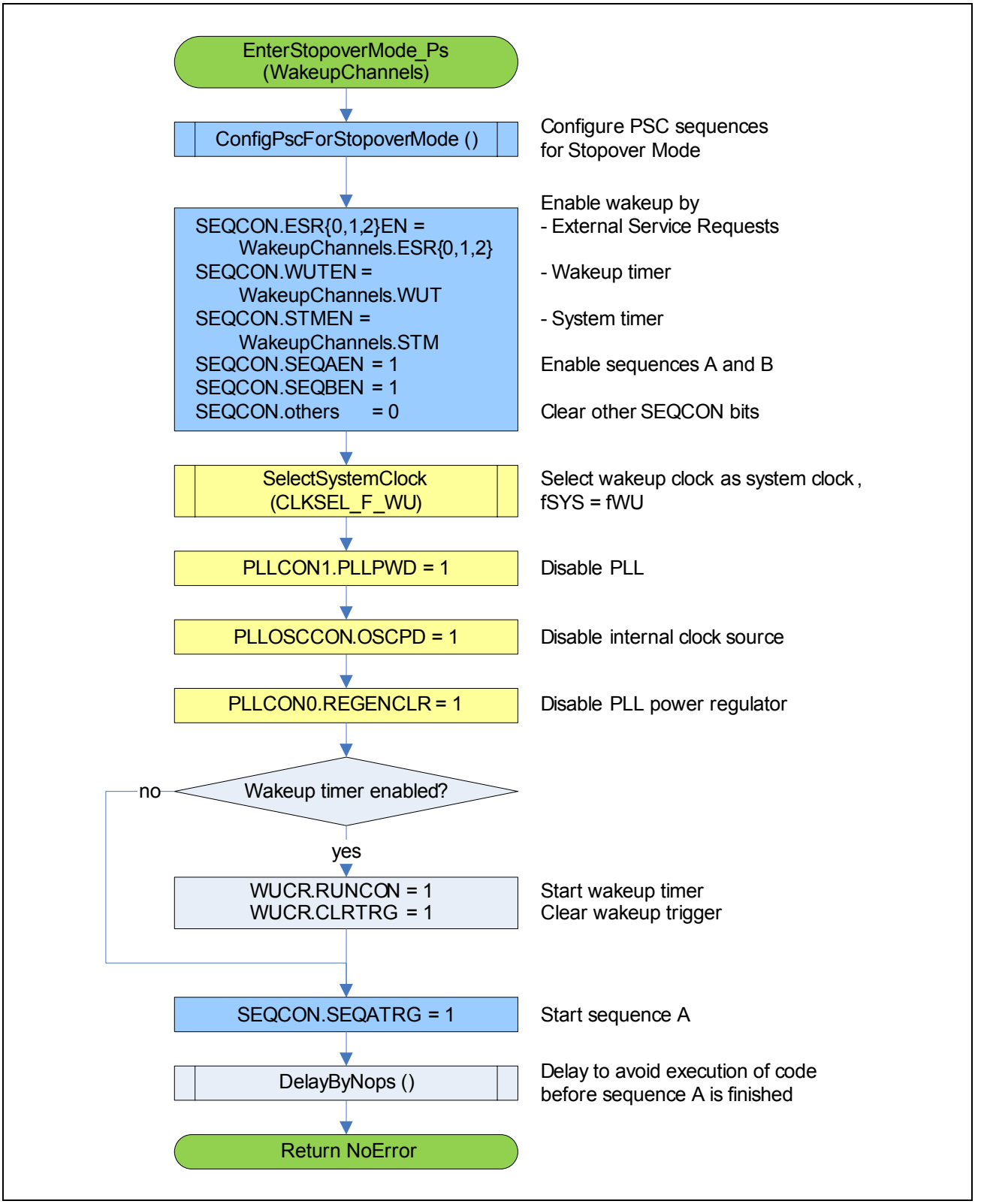

Figure 5-14 Flow-Chart EnterStopoverMode_Ps

5.5.2 Wakeup from Stopover Mode

Wakeup is done using the hardware stepping mechanism of the PSC. A wakeup can be triggered synchronously or asynchronously using the External Service Requests (e.g. ESRx pins). In case the DMP_M clock is running the Wakeup Timer can be used. The Wakeup Timer can be configured to run continously or to stop after the trigger was executed. Continous mode allows constant periods between wakeups (periodical wakeup). Note, that the device must be in Stopover Mode when the wakeup occurs.

Flow

1. Since the power transition A has stopped the system clock, the code execution will be resumed after the statement to enter Stopover Mode.
   It is checked whether the wakeup transition sequence B has been executed and terminated.
2. Wakeup triggers are disabled.

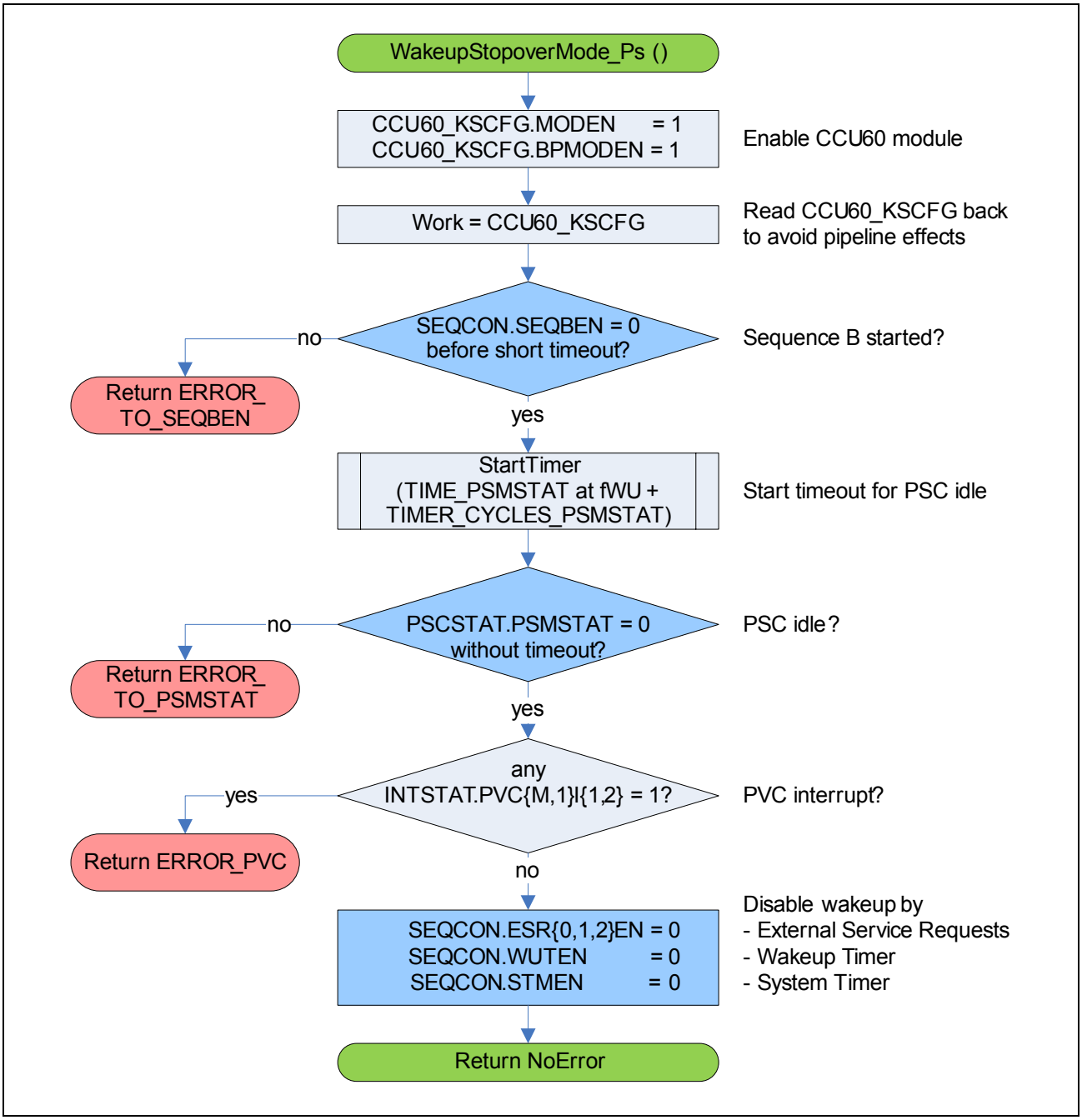

Figure 5-15 Flow-Chart WakeupStopoverMode_Ps

5.5.3 Handle Wakeup from Stopover Mode

After wakeup from Stopover Mode, the application specific user function **HANDLE_STOPOVER_PS** (with configurable name) is called. The function decides via the return value if the Stopover Mode shall be continued (re-entered) with transition **Continue Stopover Mode** or not. If not, Stopover Mode will be exit and Normal Operation Mode will be resumed with transition **Exit Stopover Mode**.

Wakeup Source

In case of multiple sources, the wakeup source can be determined via function **GetWakeupSrc**. The wakeup source request should be cleared using function **ClearWakeupSrc**.

System Clock

Initially the application function executes code at the configurable wakeup clock. If a higher and/or more stable clock is needed for short user actions, the frequency can be increased by calling function **UseFastClockInStopover_Ps** to use the configured external crystal/clock or the internal clock.

Note: The requested external crystal/clock must already be available; in case of crystal, the oscillator must be running continously.

Note: The frequency of the external crystal/clock should not be higher than 8 MHz.

Finally, function **UseWakeupClockInStopover_Ps** must be called to reduce the clock again. The configurable wakeup clock is selected as system clock.

Peripherals

If peripherals shall be activated during execution of the user function, function **RequestSystemMode** should be called with following parameters:

- SYSTEM_MODE_NORMAL to enable the peripherals
- SYSTEM_MODE_CLOCK_OFF to disable them again

Configuration of the KSCCFG registers may be done in Normal Operation Mode at high system frequency.

5.5.4 Continue Stopover Mode

If the application specific function **HANDLE_STOPOVER_PS** decided to continue Stopover Mode is entered again.

Flow

1. The transition to stop the system clock in DMP_1 is already configured.
   Wakeup clock is selected as system clock and the PLL is shut down.
2. Sequence A is started to enter Stopover Mode with stopped system clock in power domain DMP_1.
3. While sequence A is running, the CPU is executing code until the system clock is stopped. Therefore, a delay using NOPs is inserted to avoid code execution.

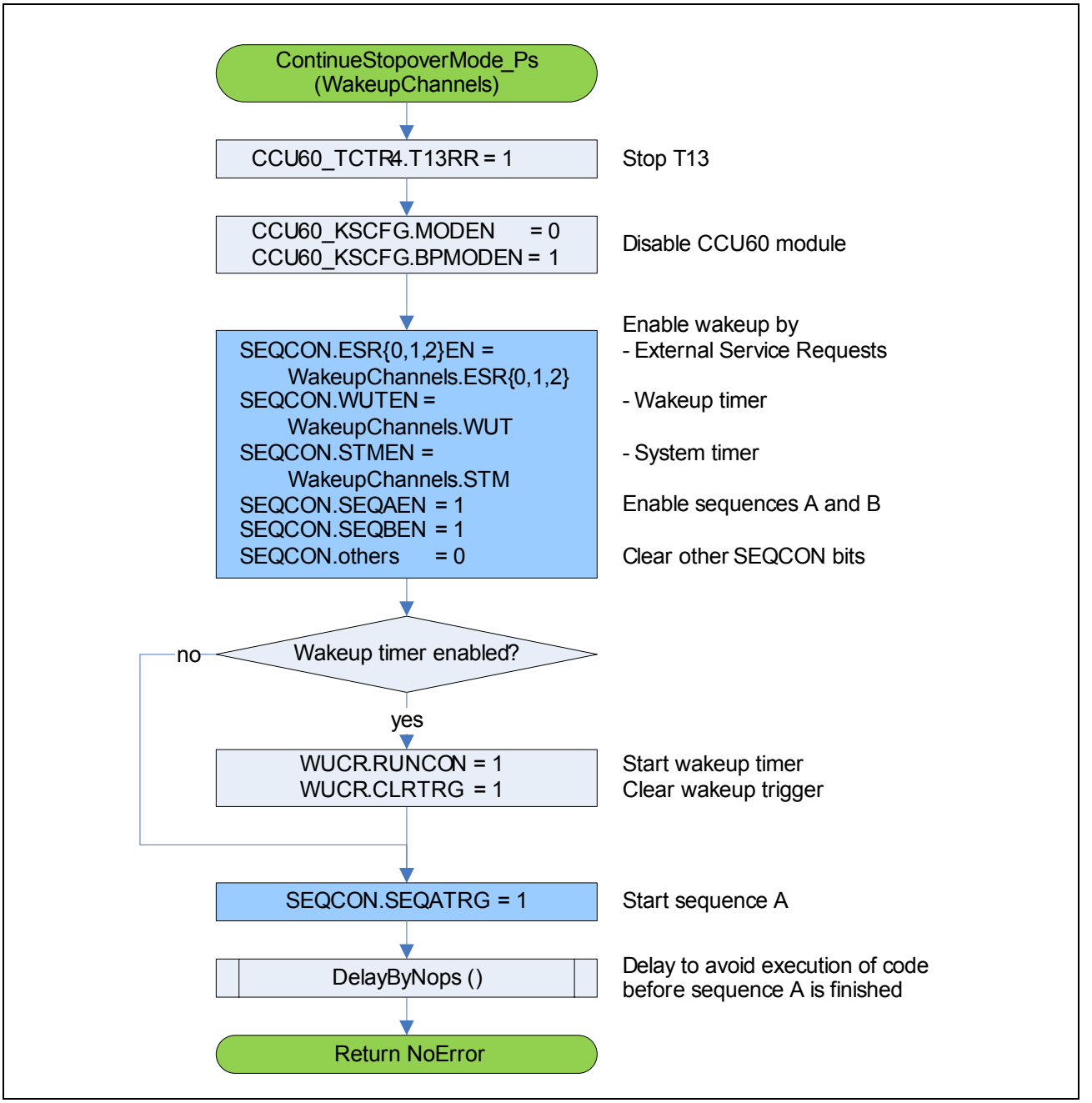

Figure 5-16 Flow-Chart ContinueStopoverMode_Ps

5.5.5 Exit Stopover Mode

The transition returns to Normal Operation Mode. The transition is executed if after a wakeup the application specific function **HANDLE_STOPOVER_PS** decided to exit Stopover Mode. During execution the location of the executed code is changed from PSRAM back to Flash.

5.5.5.1 Exit Stopover Mode (PSRAM)

Note: The code for the transition must be located in PSRAM.

Flow

1. In preparation of the power supply reconfiguration HP Bandgap is re-enabled.
2. Since Normal Operation Mode is the target mode, the PLL power regulator is enabled. While the analog parts are powered-up in sequential order considering the required delay the following steps are performed.
3. The transition sequence which conducts the switch from the LPR to the HP Bandgap is configured with function **ConfigPscForHighPrecRef**.
4. Power transition sequence A is enabled and started.
5. It is checked if the sequence A both started and then terminated.
6. The internal clock source is enabled. During the required power-up delay further configurations are done.
7. Flash is started early to allow Flash access as soon as possible.
   When the Flash is powered-up the interrupt vector table in the Flash is used and code execution from Flash can be started.

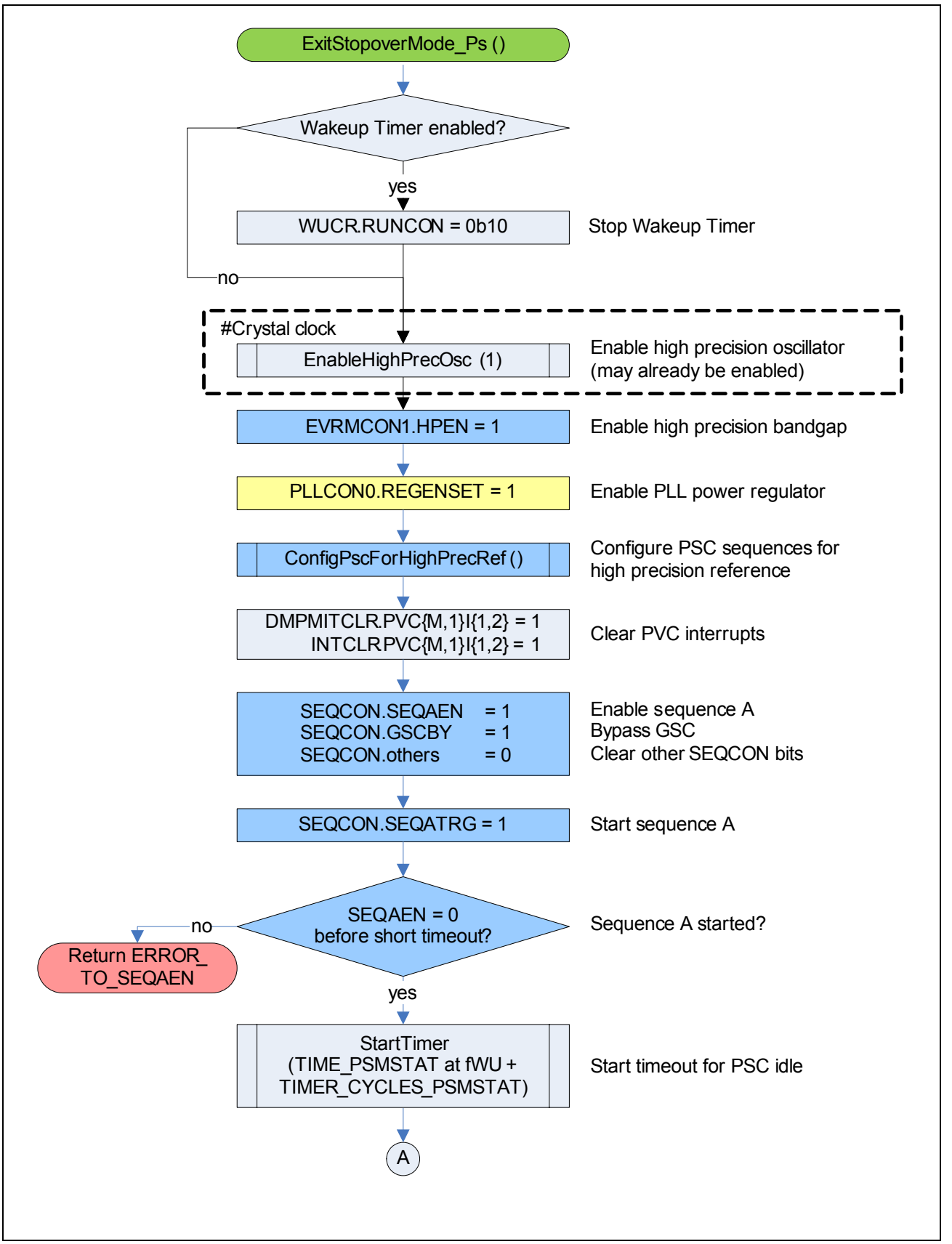

Figure 5-17 Flow-Chart ExitStopoverMode (1)

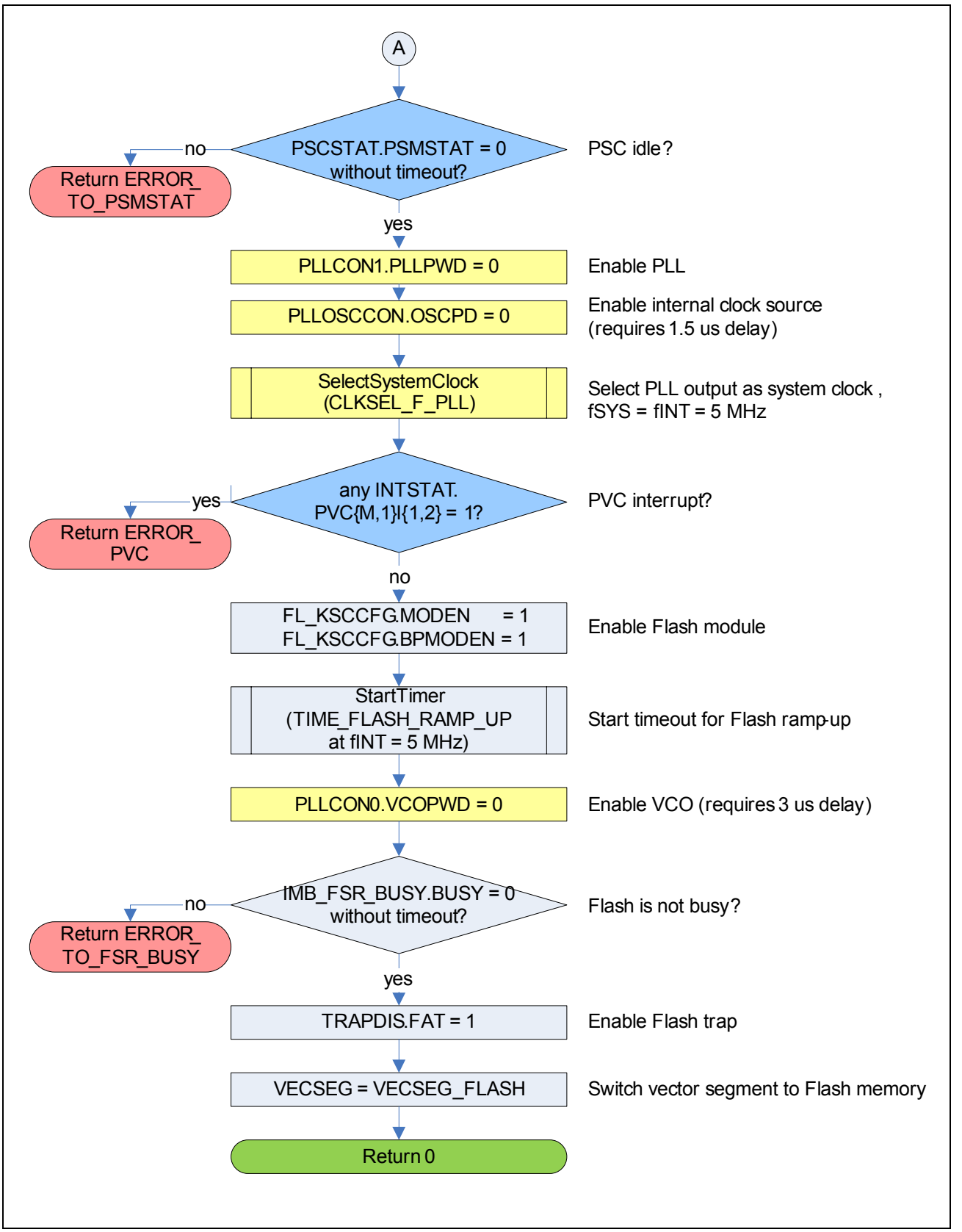

Figure 5-18 Flow-Chart ExitStopoverMode (2)

5.5.5.2 Exit Stopover Mode to Normal Operation Mode

After transition **Exit Stopover Mode (PSRAM)** the transition to Normal Operation Mode is continued from Flash. The flow is similiar to the one that is described in **Base Mode to Normal Operation Mode with External Clock** or **Base Mode to Normal Operation Mode with Internal Clock**.

Following PLL settings before entering Stopover Mode are assumed: PLL input clock source = internal clock, VCO is bypassed, K1 divider = 1, P and N dividers are set as required for Normal Operation Mode, K2 divider is set for f_{INT} ~ 5 MHz. Therefore, the PLL configuration registers defining these settings should not be changed in function **HANDLE_STOPOVER_PS**, otherwise the PLL settings have to be restored to the values before entering Stopover Mode.

Flow

1. As mentioned the PLL dividers are set as required for Normal Operation Mode. To prepare the switch to the VCO based clock the K2-divider is programmed such that its output frequency is within the range of that at the output of the K1 divider.
2. The following steps are similiar to the steps described in **Base Mode to Normal Operation Mode with External Clock** or **Base Mode to Normal Operation Mode with Internal Clock**.
3. Interrupt Enable and functional reset configuration are restored.
4. Request Normal System Mode. Note: The KSCCFG registers must be correctly configured.

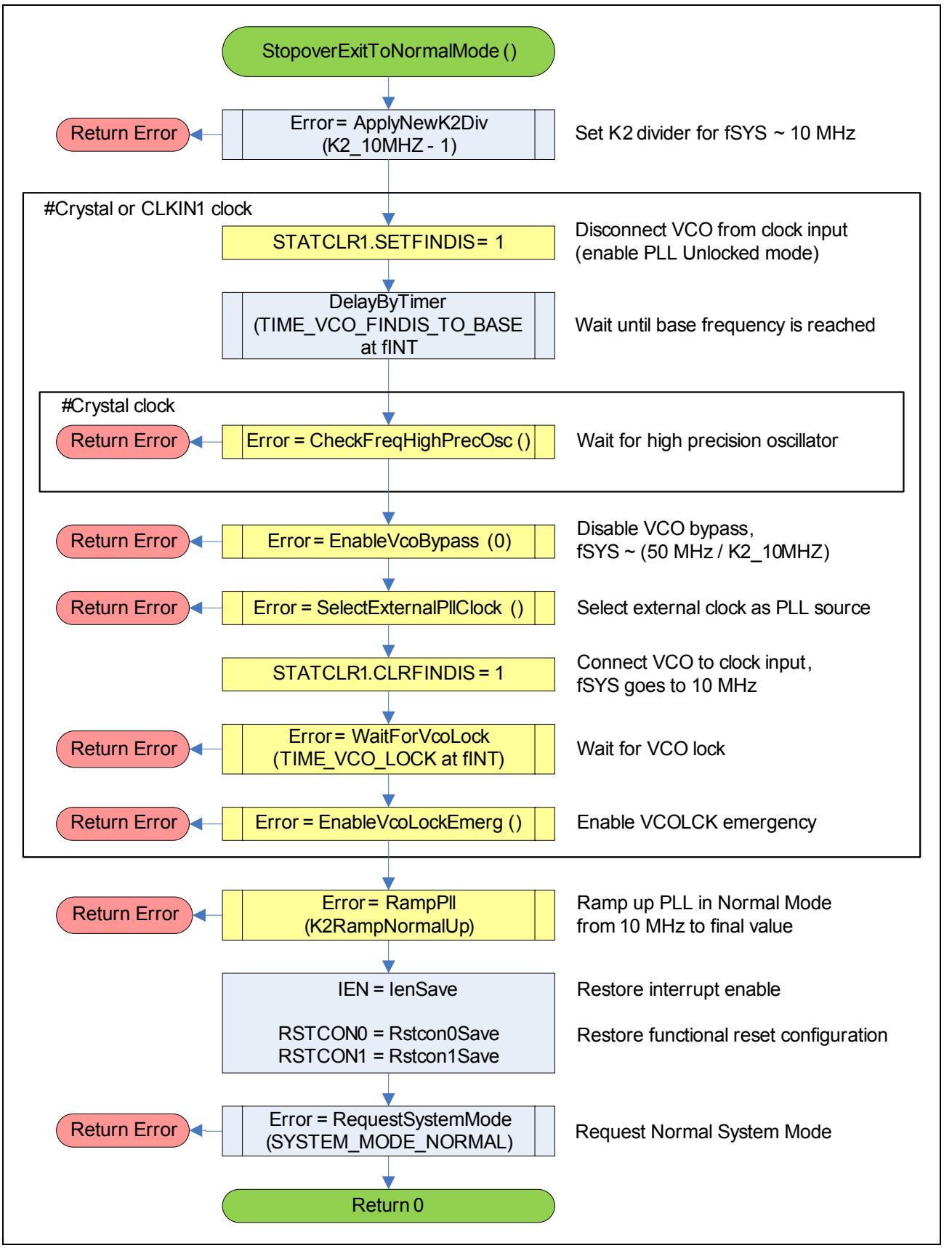

Figure 5-19 Flow-Chart StopoverExitToNormalMode (External Clock)

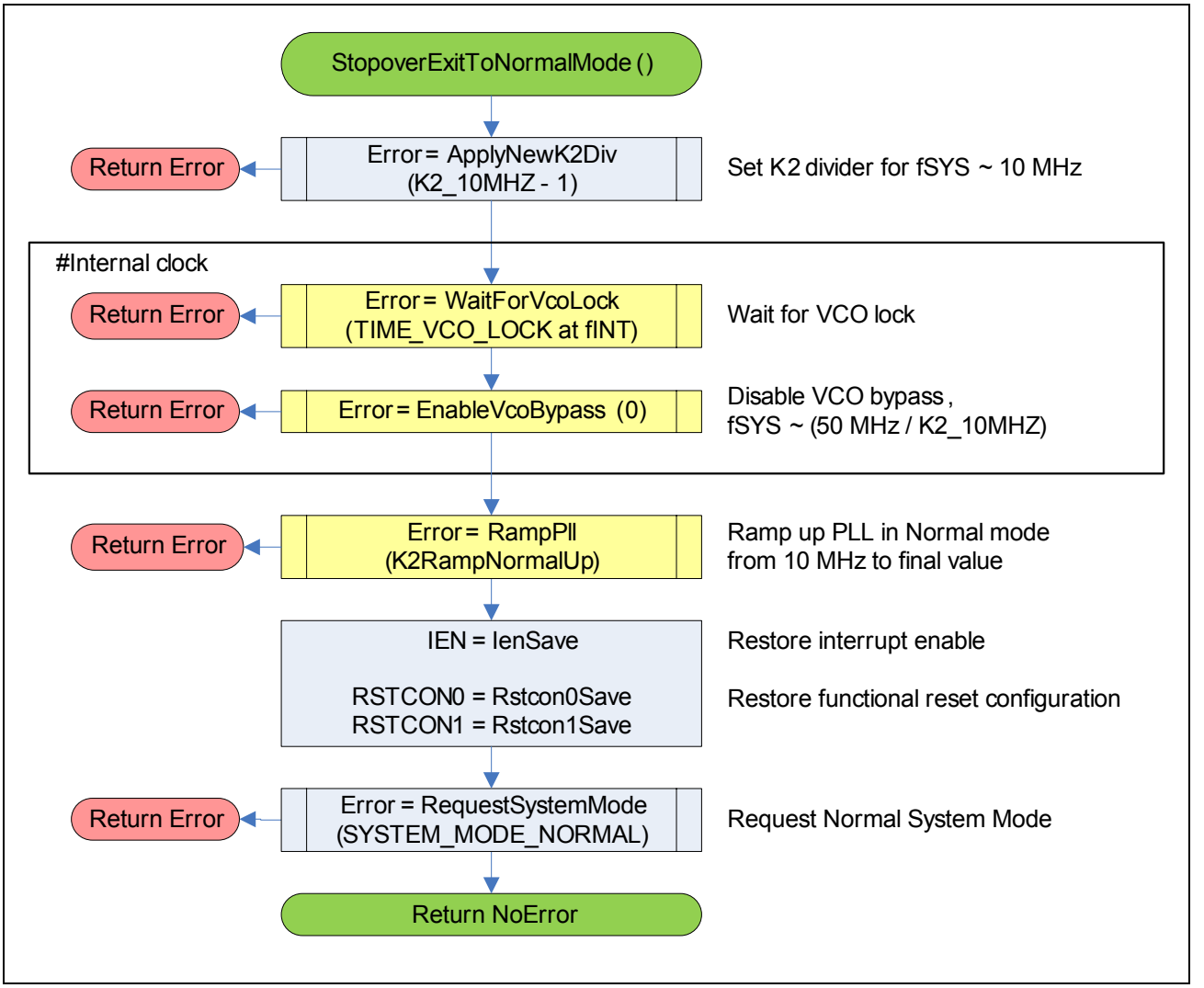

Figure 5-20 Flow-Chart StopoverExitToNormalMode (Internal Clock)

5.6 Handle Standby Mode

The Standby Mode is the most power saving mode by switching off the DMP_1 and some remaining contributors to the base load current in power domain DMP_M. The current consumption can further be reduced by disabling the supply watch-dog by configuration.

After wakeup all registers in the DMP_1 domain are reset. The startup software initializes registers in the DMP_M and DMP_1 domains.

5.6.1 Enter Standby Mode

The transition is executed if power saving mode Standby Mode is selected. The transition is similiar to the flow to enter FSM Standby Mode.

Note: The code for the transition must be located in PSRAM.

Flow

1. In order to disable DMP_1, a new power transition sequence has to be executed. The new transition set is configured with function **ConfigPscForStandbyMode**, overwriting the settings of the previous power transition A.
   During the power transition sequence the PVCs will be disabled.
   In addition, the wakeup, i.e. the ramp-up of the DMP_1 supply, via power transition sequence B is configured.
2. Upon a wakeup from Standby Mode there are two possible exit states, Base Mode and Fast Startup Mode. One of these exit states must be selected with bit field STMEM.SSFMOD prior to the entry into Standby Mode. .
3. In order to save additional current the SWD can be disabled (user configurable)
4. During the power transition sequence the PVCs will be disabled. To prepare for this the actions triggered by the PVCs are disabled.
5. In order to save additional current the CC for EVR_M is disabled, because this can not be done by the hardware sequence.
6. Wakeup clock with desired frequency during Standby Mode is selected as system clock and the PLL is shut down.
7. Sequence A is started to enter Standby Mode with powered down DMP_1.
8. While sequence A is running, the CPU is executing code until the power-domain DMP_1 is powered down. Therefore, a delay using NOPs is inserted to avoid further code execution.

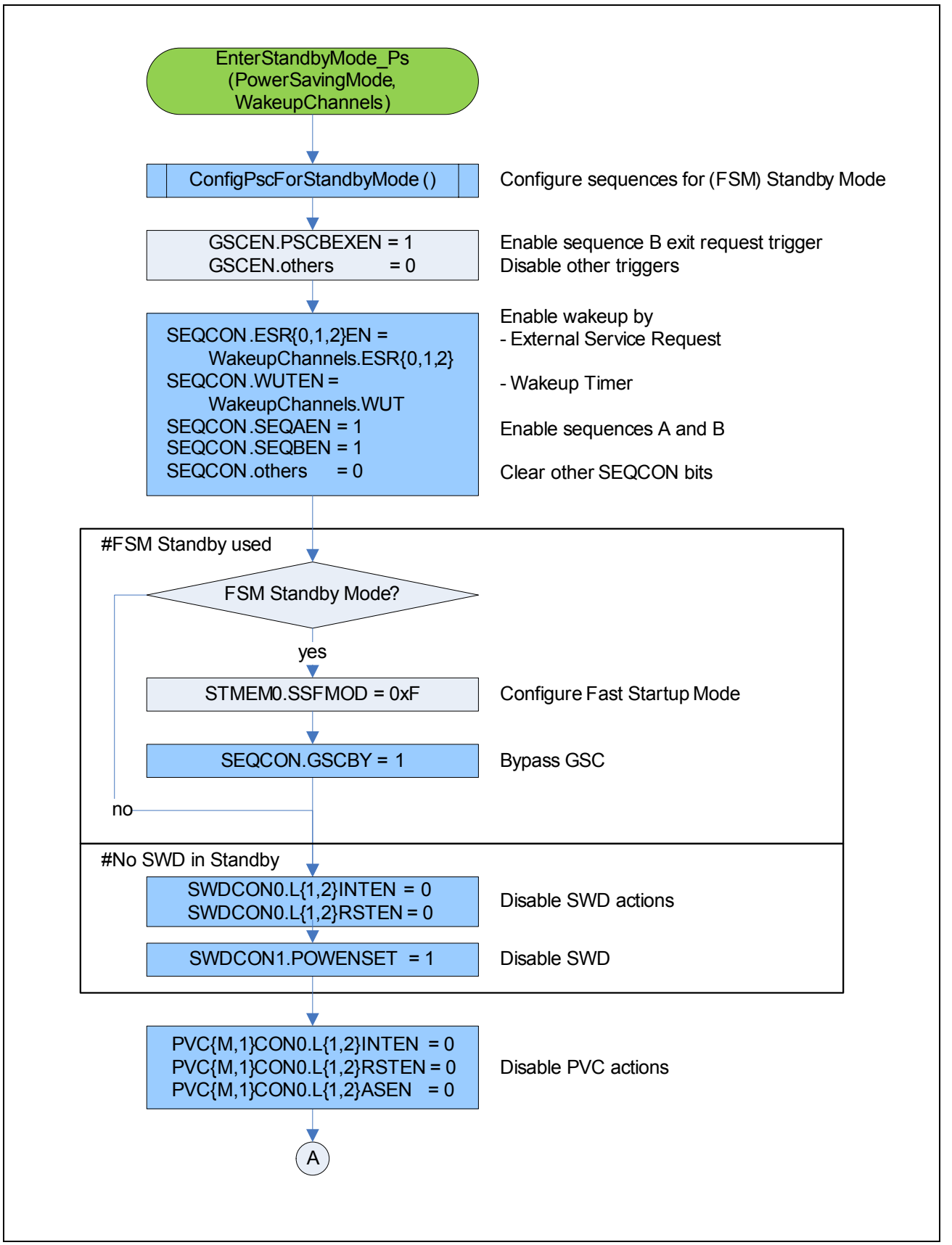

Figure 5-21 Flow-Chart Enter(FSM)StandbyMode_Ps (1)

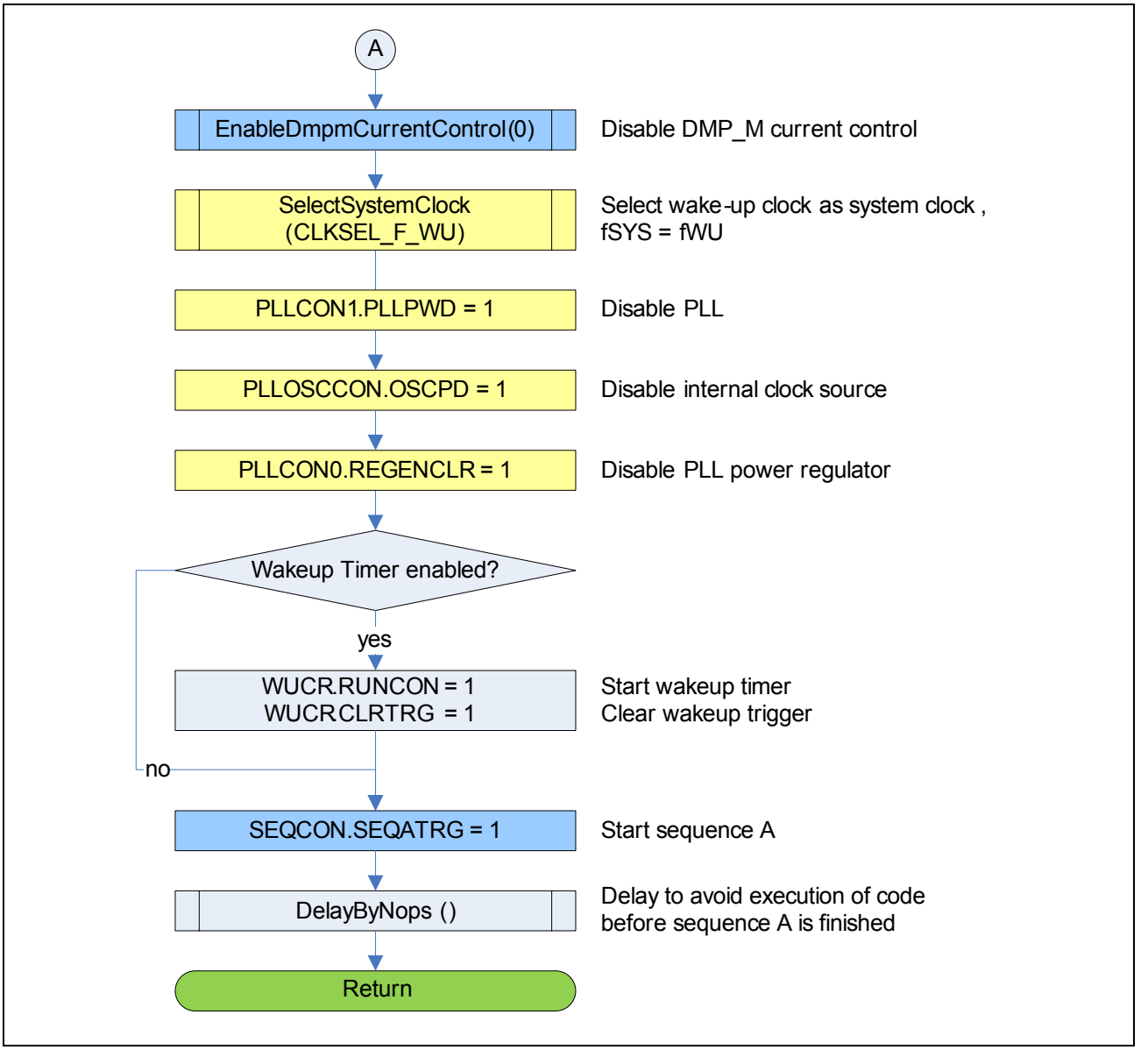

Figure 5-22 Flow-Chart Enter(FSM)StandbyMode_Ps (2)

5.6.2 Wakeup and Exit from Standby Mode

After wakeup from Standby Mode a DMP_1 reset is performed which results in a return to Base Mode via the startup software.

Wakeup Source

In case of multiple sources, the wakeup source can be determined via function **GetWakeupSrc**. The wakeup source request should be cleared using function **ClearWakeupSrc**.

5.7 Handle FSM Standby Mode

FSM Standby Mode has the same power saving feature as Standby Mode.

The transition performs the transition or repeated transition to enter FSM Standby Mode (i.e. power domain DMP_1 is switched off). After wakeup from FSM Standby Mode the startup software copies the code to be executed from Standby RAM (SBRAM) to PSRAM because the SBRAM does not support code execution. Then the code execution from PSRAM is started.

User function **HANDLE_STANDBY_FSM_PS_SB** (name is configurable) is called which decides about continuation of FSM Standby Mode, i.e. FSM Standby Mode is entered again, or a transition to Standby Mode is performed.

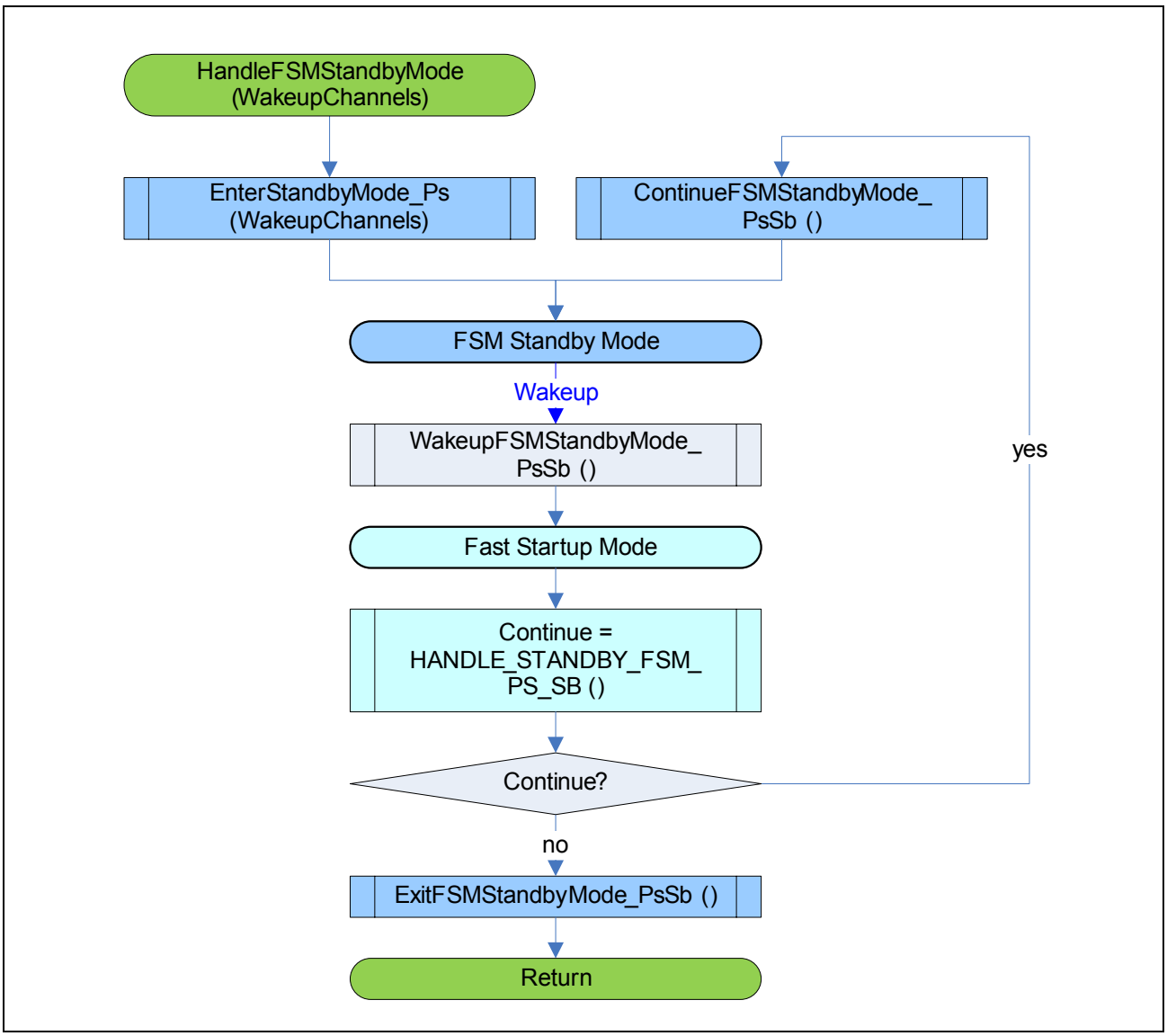

Figure 5-23 Flow-chart HandleFSMStandbyMode

5.7.1 Enter FSM Standby Mode

The transition is executed if power saving mode FSM Standby Mode is selected. The transition is similiar to the flow to enter Standby Mode.

Note: The code for the transition must be located in PSRAM.

Flow

1. In order to disable DMP_1, a new power transition sequence has to be executed. The new transition set is configured with function **ConfigPscForStandbyMode**, overwriting the settings of the previous power transition A.
   During the power transition sequence the PVCs will be disabled.
   In addition, the wakeup, i.e. the ramp-up of the DMP_1 supply, via power transition sequence B is configured.
2. Upon a wakeup from Standby Mode there are two possible exit states, Base Mode and Fast Startup Mode. One of these exit states must be selected with bit field STMEM.SSFMOD prior to the entry into Standby Mode. .
3. In order to save additional current the SWD can be disabled (user configurable)
4. During the power transition sequence the PVCs will be disabled. To prepare for this the actions triggered by the PVCs are disabled.
5. In order to save additional current the current control for EVR_M is disabled, because this can not be done by the hardware sequence.
6. Wakeup clock with desired frequency during Standby Mode is selected as system clock and the PLL is shut down.
7. Sequence A is started to enter Standby Mode with powered down DMP_1.
8. While sequence A is running, the CPU is executing code until the power-domain DMP_1 is powered down. Therefore, a delay using NOPs is inserted to avoid further code execution.

Flow-Chart

The flow-charts shown in **Figure 5-21** and **Figure 5-22** cover the transition to enter FSM Standby Mode, too.

5.7.2 Wakeup from FSM Standby Mode

After wakeup from FSM Standby Mode a DMP_1 reset is performed and the Fast Startup Mode is entered via startup software.

The code to be executed is taken from SBRAM - Standby RAM which is located in power domain DMP_M and powered during FSM Standby Mode. Because the SBRAM does not support code-execution - the code is copied into and started from the PSRAM.

Flow

1. Watchdog timer is disabled.

2. To avoid possible disruptions during the sequence all interrupts, and resets are disabled.

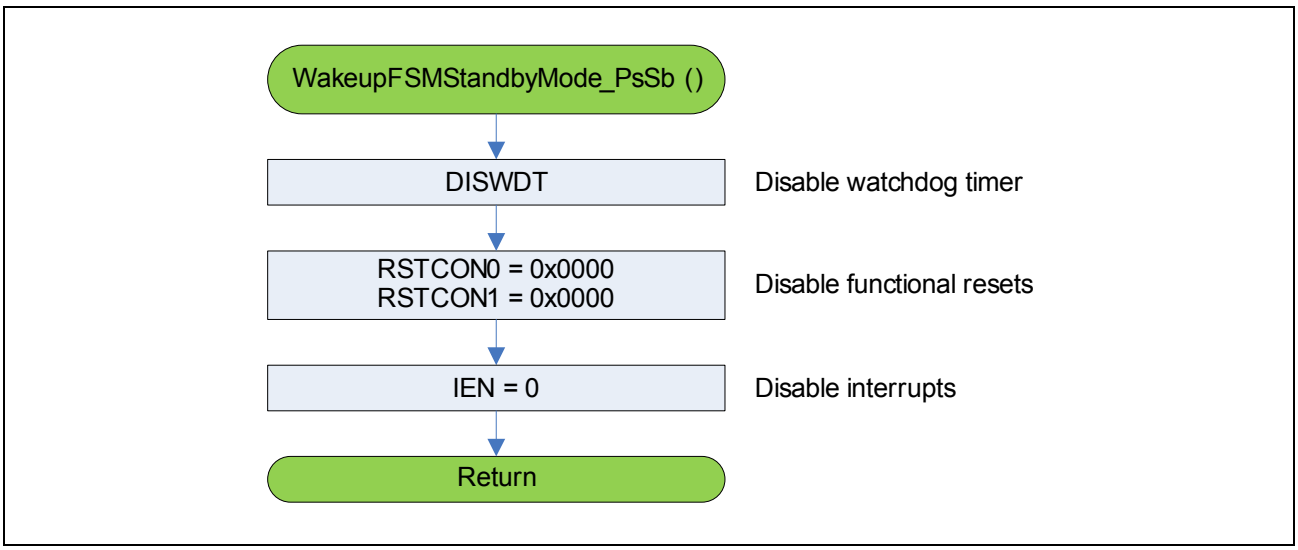

Figure 5-24 Flow-Chart WakeupFSMStandbyMode_PsSb

5.7.3 Handle Wakeup from FSM Standby Mode

After wakeup from FSM Standby Mode, the application specific user function **HANDLE_STANDBY_FSM_PS_SB** (with configurable name) is called. The function decides via the return value if the FSM Standby Mode shall be continued (re-entered) with transition **Continue FSM Standby Mode** or not. If not, Fast Startup Mode will be exit with the transition to Standby Mode which is immediately followed by a wakeup with DMP_1 reset.

Wakeup Source

In case of multiple sources, the wakeup source can be determined via function **GetWakeupSrc**. The wakeup source request should be cleared using function **ClearWakeupSrc**.

System Clock

When the function is started, the system clock is provided by the internal clock. After time-critical actions have been finished using internal clock as system clock, function **UseWakeupClockInStandbyFsm_Ps** must be called to reduce the clock to continue with FSM Standby Mode or to exit FSM Standby Mode/Fast Startup Mode. The configurable wakeup clock is then used as system clock.

Peripherals

If peripherals shall be activated during execution of the user function, function **RequestSystemMode** should be called with following parameters:

- SYSTEM_MODE_NORMAL to enable the peripherals
- SYSTEM_MODE_CLOCK_OFF to disable them again

The KSCCFG registers must be configured correctly.

5.7.4 Continue FSM Standby Mode

If the application specific function **HANDLE_STANDBY_FSM_PS_SB** decided to continue FSM Standby Mode is entered again.

It has to be ensured that the wakeup clock is selected as system clock and the PLL is shut down calling function **UseWakeupClockInStandbyFsm_Ps** before.

Flow

1. Because the next wakeup shall lead to the Fast Startup Mode again, the Wakeup Timer is started again immediately before the PSC is triggered to conduct a power transition sequence A.
2. Power transition sequence A is started.

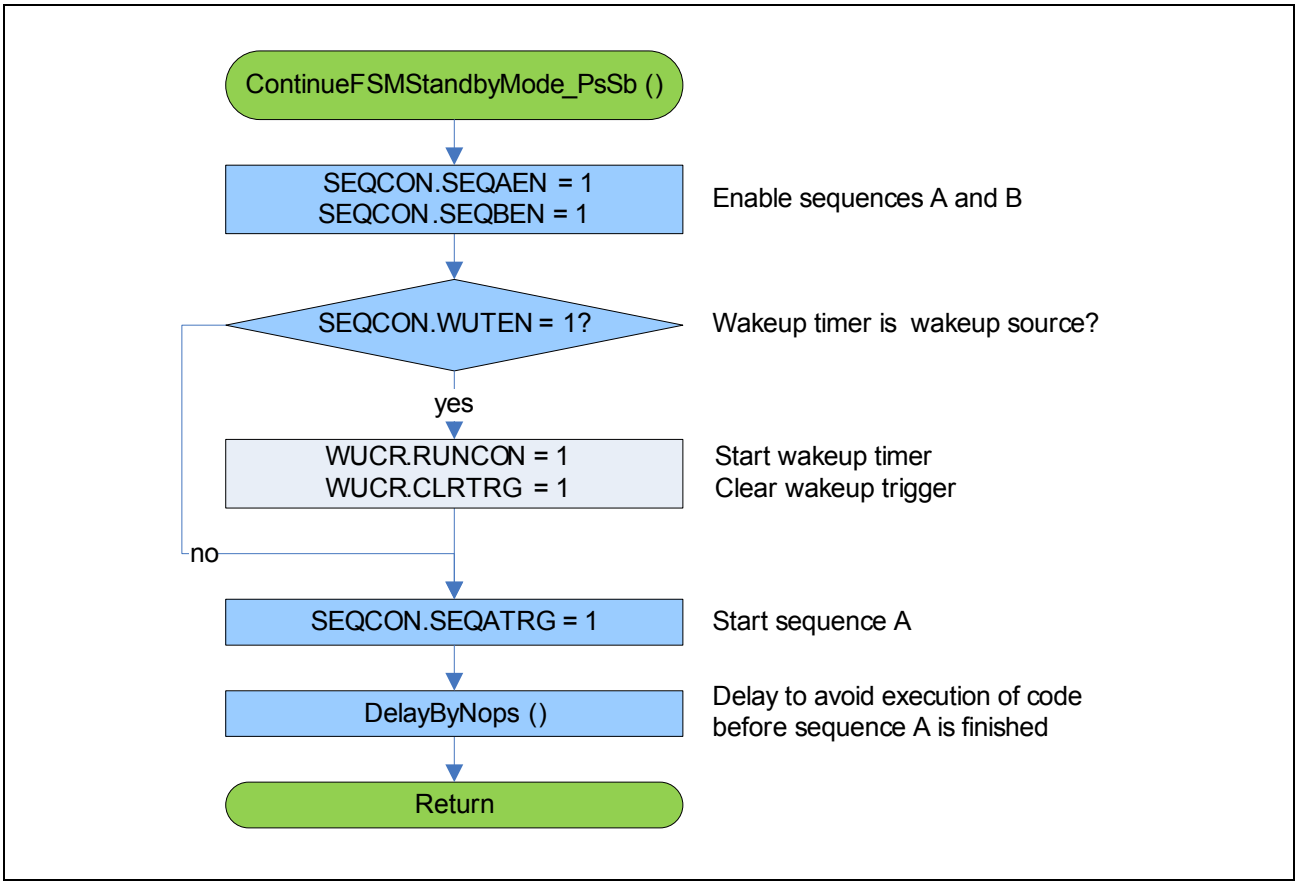

Figure 5-25 Flow-Chart ContinueFSMStandbyMode_PsSb

5.7.5 Exit FSM Standby Mode

This transition leads from the Fast Startup Mode back to Base Mode after the next wakeup.

It has to be ensured that the wakeup clock is selected as system clock and the PLL is shut down calling function **UseWakeupClockInStandbyFsm_Ps** before.

Note: Function must be located in PSRAM.

Flow

1. Because the next wakeup shall lead to Base Mode, the FSM request is cleared in STMEM.SSFMOD. To terminate Standby Mode immediately after power transition sequence A has finished, power transition sequence B is enabled and triggered by software. In doing so, the request to wakeup is already stored.
2. Power Transition Sequence A is started.

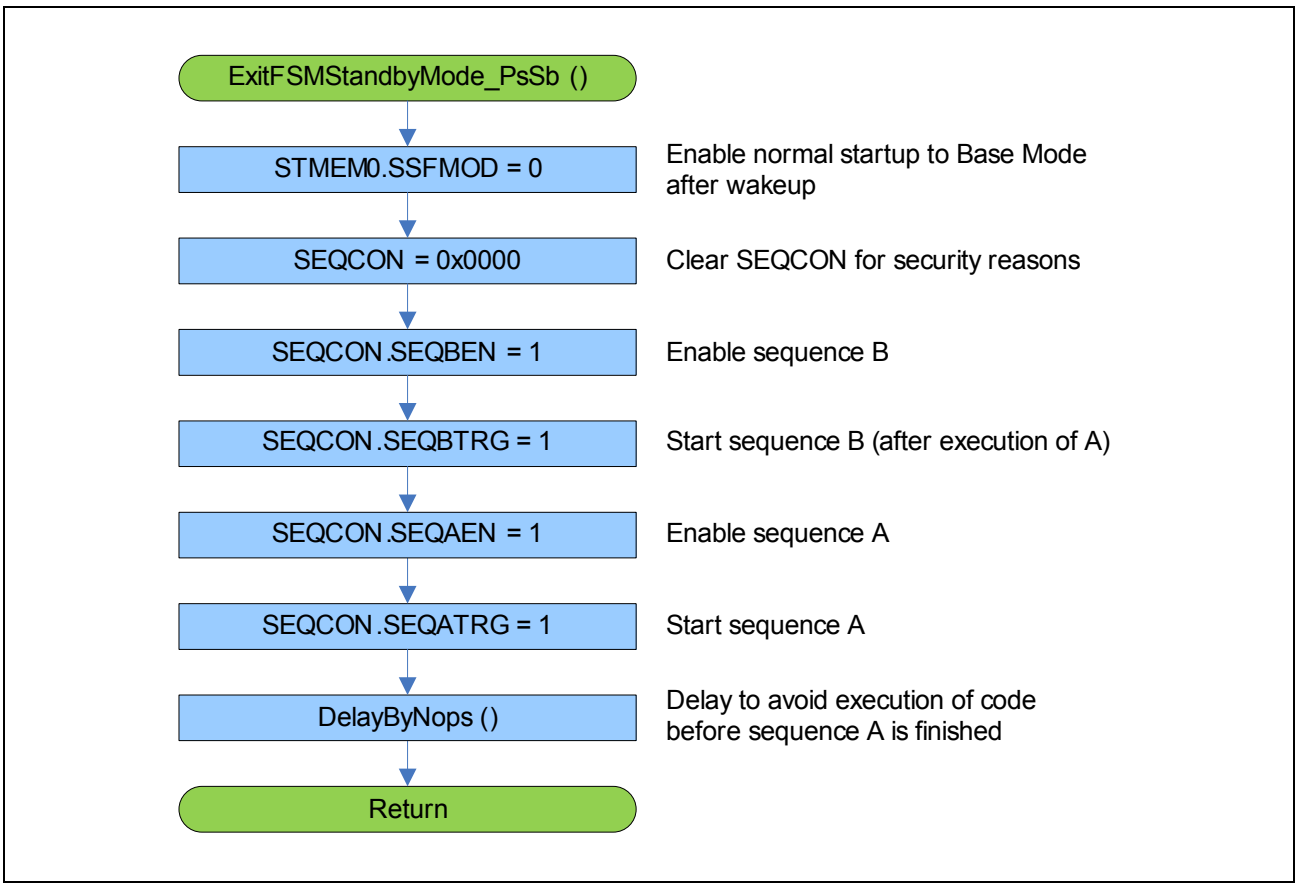

Figure 5-26 Flow-Chart ExitFSMStandbyMode_PsSb

6 Resets

This chapter provides an overview on all resets available for the XC2000/XE166 Family architecture, including a detailed description of the types of resets. Finally, practical hints are given when to use a specific reset, and what the software can do after a reset.

6.1 Some terms regarding resets

To be able to discuss the resets of the XC2000/XE166 in detail, it makes sense first to do some disambiguation on some terms regarding reset

Reset
The term “reset” used in this document comprises a high priority sequence of actions in which the settings of dedicated parts of a system are changed to a fixed configuration.

Reset Trigger
This is an external or internal event, mostly represented by a certain logical state of a signal, which is used to request the reset. If enabled and arbitrated, a trigger will activate a reset of a certain type (see **Figure 6-1**).
It is an important part of the XC2000/XE166 philosophy, that a reset is separated from its trigger event. This means that **a trigger per se does not necessarily make a reset**. Following this philosophy, almost every reset trigger in the XC2000/XE166 system has its dedicated enable bit.

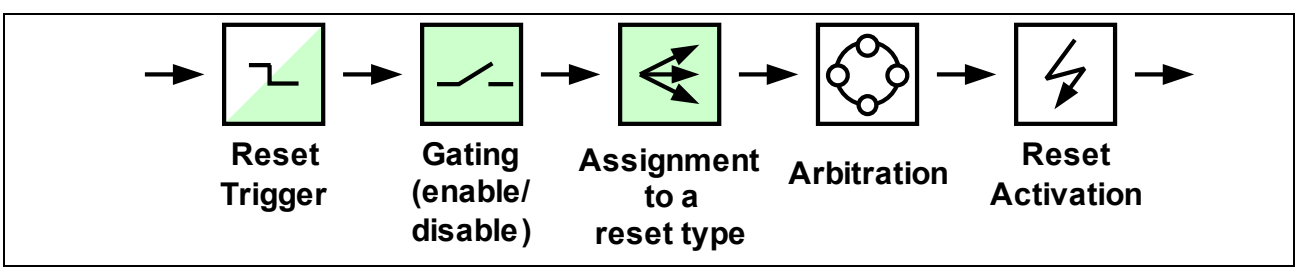

Figure 6-1 The way from a trigger to a reset. The shaded actions can be configured by software

For the **User Resets** gating (enabling of a trigger) and assignment to a type of reset is done for the using the RSTCONx **SFR**s. For the **Power Resets** enabling of the triggers is done with the respective bits in PVCxCON0 and SWDCON0.

Reset Signal
This is the actual signal, which - if activated - causes functional targets (see below) to be set to their default state.

Reset Class
The resets of the XC2000/XE166 are grouped into two classes

- Power Resets (**Asynchronous**)
- User Resets (**Synchronous**)

The Power Resets take care for data integrity in conjunction with power supply issues. Their reset signals are distributed asynchronously, i.e. they do not need a clock to become effective at their functional target.
The User Resets (also named **Functional Resets**) are offered for the convenience of the software **Application** and allow a re-configuration of the system, excluding clock and supply settings. The generation of the reset signals of the User Resets is handled by the Reset Control Logic (**RSC**). Because this unit works synchronously a User Reset needs a proper system clock f_sys to be conducted according to **Figure 6-2**.
Chapter 6.2 gives more detailed information on the difference between these two classes. of resets.

Reset Properties
Independent from its class, the type of a reset is defined by its properties:

Priority
Due to the fact that the XC2000/XE166 has several types of reset, it must be defined which type of a reset has priority over another. The priority is listed in **Table 6-1**, beginning with priority 1 as the ultimate priority level, and ending with priority 5 as the lowest level. Following this priority scheme, a reset always activates all resets of a lower priority level.

Objective
In order to find out why there are different types of resets in the XC2000/XE166 architecture, the key question is that of the purpose of a reset. **Table 6-1** lists the objectives of the different types of resets at a glance. In **Chapter 6.2**, all resets are discussed in detail.

Functional Target
This defines the object which is affected by the reset, i.e. which part of the system will get its default setting upon the reset. **Table 6-1** lists the functional targets of the XC2000/XE166's reset types.

Reset conduction
As already mentioned in its definition, there is more behind a reset than the act of re-setting some target. It also comprises additional actions. Thus, conducting a reset means the entire sequence from the occurrence of a possible trigger until the first instruction of the application software is executed again. **Figure 6-2** shows such a transaction flow, typical for an XC2000/XE166 reset.

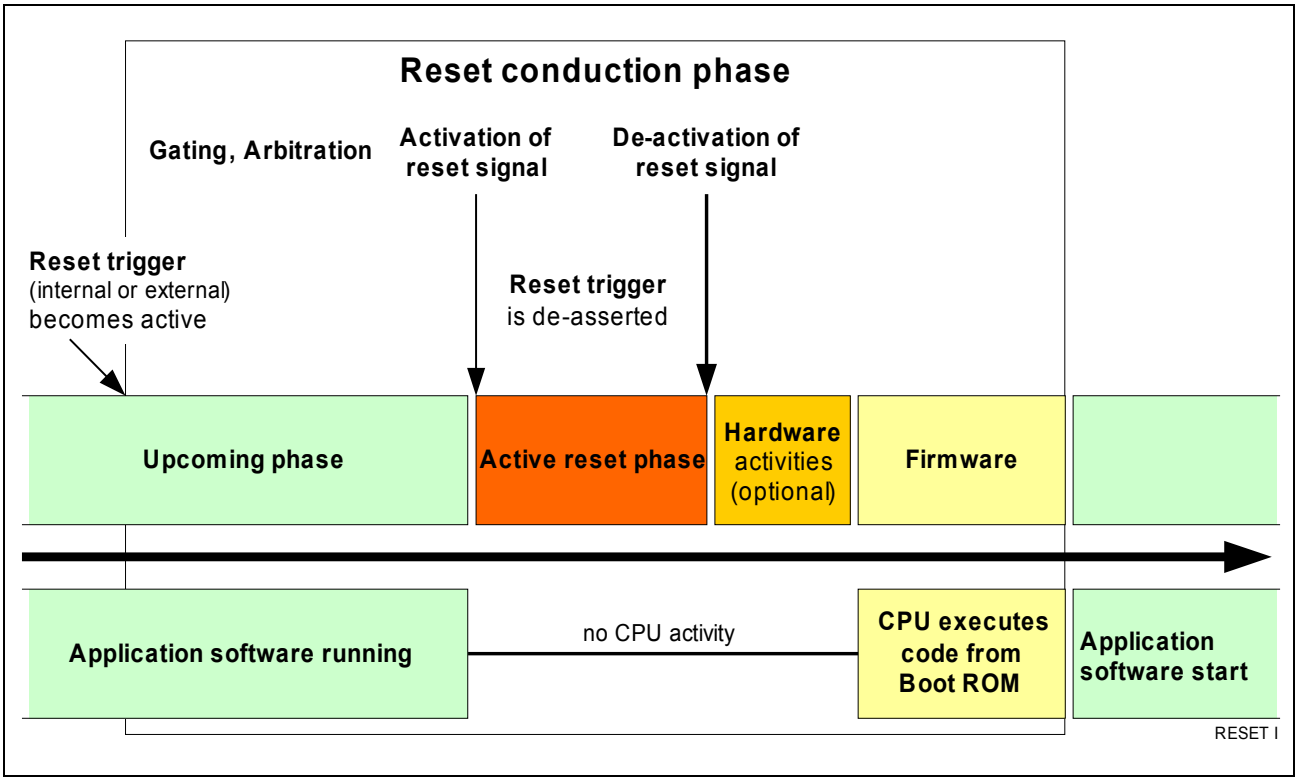

Figure 6-2 Typical reset conduction flow for the XC2000/XE166 Family

1. **Upcoming phase**
   This is the time frame in which a trigger occurs and the arbitration takes place. The trigger can be an external event, i.e. a changed signal level at a pin, or an internal event. Even the software can generate a reset trigger by setting the respective bit in register SWRSTCON or executing the SRST command. The software can disable a trigger from becoming arbitrated. This is possible for any reset, except the **Power-on Reset**.
2. **Active reset phase**
   After an enabled trigger has been arbitrated successfully, the assigned reset signal is activated. At this point, all storage elements within the target group of the reset are set to their default values.
   Note that, due to the fact that a reset might have more than one functional target, there can be more than one active reset phase in one reset conduction flow. See the description of the RSC for more details.
3. **Hardware activities**
   After the reset signal has been de-activated, some hardware activities may be performed such as, e.g., the ramp-up of the digital power domains DMP_M and DMP_1 (refer to **Figure 11-1 Topological overview of the XC2000/XE166 supply domains.**), which is done in conjunction with the **Power Resets**.
4. **Firmware**
   At the end of every reset conduction sequence, the CPU will execute some code from the startup ROM in order to perform set up actions specific to the type of the current

reset.
Upon any type of the **Power Resets**, the supply and the clock system will be set up to a default configuration of **Base Mode**, whereas after the **User Resets**, the Firmware flow does not touch the current clock and supply settings.

Depending on the type of reset, some activities may not be performed. See **Chapter 6.2** about **Types of Reset** for more details.

6.2 Types of Reset

This chapter explains all types of resets in detail. **Figure 6-3** shows the location of the reset signals within the supply domains of the XC2000/XE166.

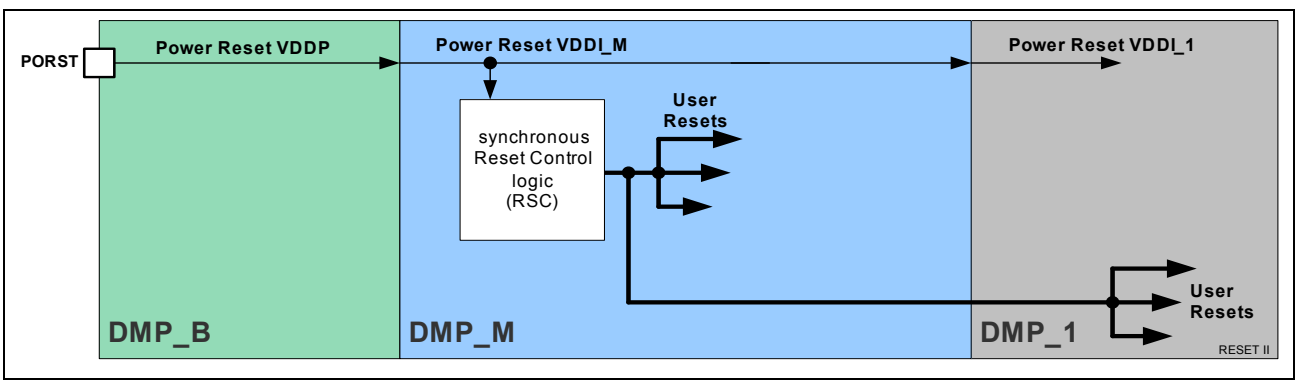

Figure 6-3 Distribution of the resets signals to the XC2000/XE166 supply domains

Each supply domain has its dedicated power reset signal. Additionally, the 1.5 V domains DMP_M and DMP_1 have the **User Resets** available, which are centrally generated by the Reset Control Logic (**RSC**). This makes clear that conducting a reset always affects targets in more than one power domain.

DMP_B
DMP_B hosts a non clocked combinational (asynchronous) logic, which is in charge to ramp up the DMP_M supply, and to monitor VDDPB and VDDI_M supply levels in the operating phases.

DMP_M
DMP_M works as a standby domain, and contains a unit to ramp up DMP_1, as well as for supervising VDDI_1 supply levels.

DMP_1
DMP_1 hosts the major part of the functional synchronous logic such as CPU, **Flash** and the on-chip peripherals.

Table 6-1 lists the resets in their hierarchical order.

Table 6-1 Objectives and functional targets of the XC2000/XE166's resets

Type	Class	Priority	Objective	Functional Target
Power-on Reset	Power Reset (asynchronous)	1	Initialize entire device upon power-on or regain data consistency as reaction on a severe problem with the VDDP supply or the system clock	Entire controller
Power Reset for VDDI_M and VDDI_1 supply domains	Power Reset (asynchronous)	2	Regain data consistency upon a power fail in one of the 1.5 V supply domains	Entire logic inside VDD_M and VDD_1 supply domain
Power Reset VDDI_1 supply domain	Power Reset (asynchronous)	3	Prevent data inconsistency upon entry into and exit from power saving modes, where VDDI_1 is switched off	Entire logic inside VDD_1 supply domain
Internal Application Reset	User Reset/ Functional Reset (synchronous)	4	Initialize the functional system including the port settings, and have the CPU resuming from a starting point without impact on supply and clock settings	Logic supplied by typ. 1.5 V, excluding supply and clock settings and pad control logic
Application Reset	User Reset/ Functional Reset (synchronous)	5	Initialize the functional system, and have the CPU resuming from a starting point without impact on supply and clock settings	Logic supplied by typ. 1.5 V, excluding supply and clock settings

6.2.1 Power-on Reset

Class
Power Reset

Objective
This Reset is the master reset for the device. The intention is that this reset is activated upon the initial power-up or a **Brown-out** of the VDDPB supply.
Additionally, the Power-on Reset is the first choice upon any kind of hardware related exceptions in the operating phase, such as a WDT event or upon the detection of any kind of clock related problems.
Due to these targets, it is required that every storage element within any domain will be set to a defined state by this reset immediately and at any time.
Because the controller might not be operable anymore at the time the reset shall be triggered, the Power-on Reset must be triggerable from outside.

Functional Target
Following its objective, the Power-on reset impacts the entire device. This means that all supply domains are set to their default state. In the active reset phase for the VDDP domain, the supplies for the VDDI domains are disabled. No information except inside the Flash is retained.

Conduction flow
This reset is triggered exclusively by applying a low level at the PORST pin. The PORST trigger can not be disabled and needs no arbitration. Thus, the active reset phase is entered immediately. The active reset phase inside the DMP_B domain remains until the low level at the PORST pin is released.
The following hardware flow now starts the DMP_M supply and waits until VDDP and VDDI_M supply levels are as expected. The reset of the flow is just like the conduction flow of the **Power Reset for VDDI_M and VDDI_1 supply domains**.

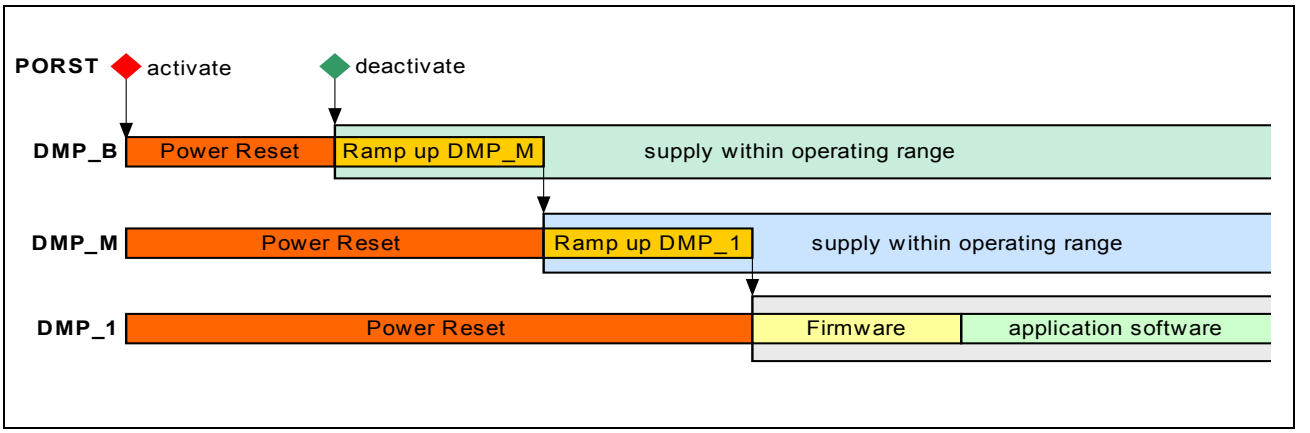

Figure 6-4 Conduction of a Power-on Reset. The reset trigger at pin PORST has to be held active until the VDDP supply is within the operating range

Limitations
Due to its universal impact, the Power-on Reset has no limitations. It can be triggered at

any time and will become effective immediately without physical damage to the chip. However, at the same time it wipes out any information upon previous activities except those stored in the Flash. Especially the content of the RAM is not reliable anymore.

6.2.2 Power Reset for VDDI_M and VDDI_1 supply domains

Class

Power Reset

Objective

This reset is to be activated in case of any trouble with the 1.5 V supplies, i.e. when the PVCs flag that the supply level is not as expected. Additionally, this reset shall be conducted in case the VDDP supply goes below the operating range limit, e.g. due to a supply voltage drop detected by the SWD.

Functional Target

This reset generally impacts both, DMP_M as well as the DMP_1 supply domain, even if the nonconformity of the power supply occurred in one of the domains. The reset will set the supply and clock settings to their default values. In the active reset phase, the EVR_1 is disabled, whereas the EVR_M is kept running.

Conduction flow

As for the Power-on Reset conduction flow, the reset active phase inside DMP_M is entered immediately upon an enabled trigger, e.g. after a power fail situation at one of the digital domains has been detected, or upon power-up.

When the active reset phase is terminated a hardware unit, it starts DMP_1 supply and waits until VDDI_1 supply level is as expected.

The further flow is the same as for the **Power Reset VDDI_1 supply domain**.

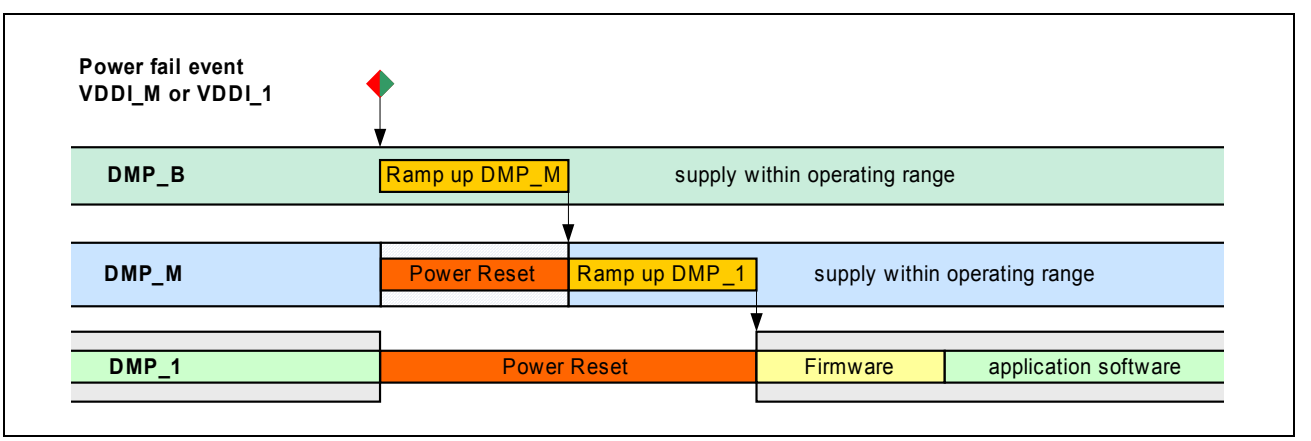

Figure 6-5 Conduction of a Power Reset for VDDI_M and VDDI_1. The supply of DMP_M may remain unaffected in case the failure appeared in DMP_1

Limitations

When SWD and PVCs are configured as described, there are no limitations regarding this type of reset. In contrast to the Power-on Reset, this reset does not set back the power-up logic hosted in the DMP_B domain. This allows to distinguish between these two types of reset. How this is done is described in **Chapter 6.3**.

6.2.3 Power Reset VDDI_1 supply domain

Class
Power Reset

Objective
This reset is dedicated to bring the logic of the DMP_1 domain to its default state, when this domain is going to be switched off or switched on again in conjunction with a power save mode.
This reset is triggered by the respective Operating Mode Transitions.

Functional Target
According to the objective, this reset is effective for the entire VDDI_1 power domain.

Conduction flow
The reset active phase is entered immediately and without arbitration upon a trigger from the DMP_M domain, and left when the DMP_1 supply levels are as expected and the reset trigger is released.
Now the CPU is ready to operate, and the following Firmware flow configures optimized settings for the supplies and the clock system, and thus performs the transition to **Base Mode**.

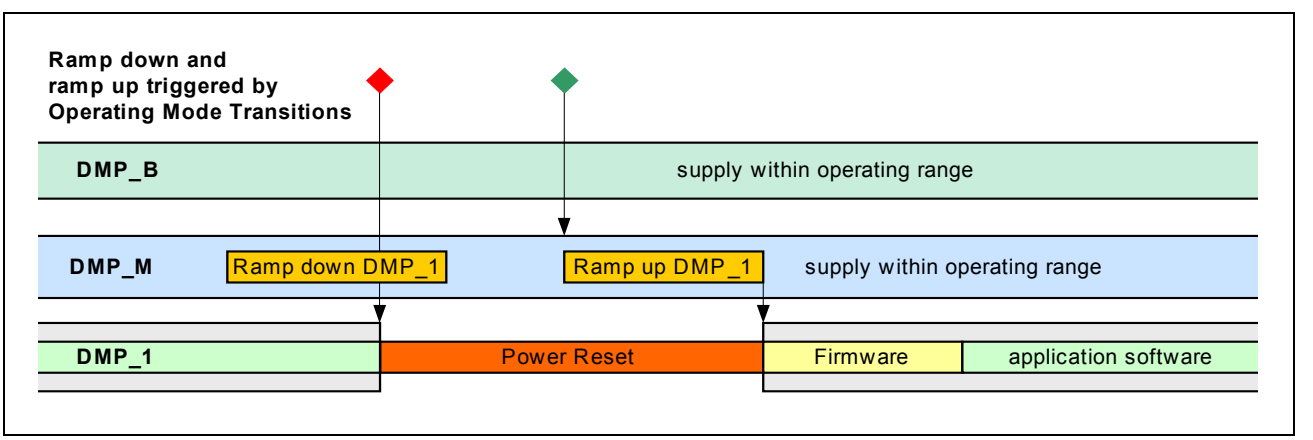

Figure 6-6 Conduction of a Power Reset VDDI_1. The supply of DMP_M remains unaffected

Limitations
Because this reset is only conducted in conjunction with a predefined flow, there are no further restrictions to it.

6.2.4 Internal Application Reset

Class
User Reset

Objective
This type of the reset shall re-initialize the entire functional logic including the CPU, but except the supply and the clock system. For convenience, it is triggerable by various external events as well as internally by software.

Functional Target
The active reset signal sets back the entire functional logic, i.e. CPU and all peripherals, including the pad control logic. Clock and power supply settings remain unchanged.

Conduction flow
When arbitrated successfully, the reset active phase is entered. It remains there until the reset elongation counter has been exceeded and the reset trigger is not active anymore. No further hardware activities are conducted afterwards, and the following Firmware flow does only perform minor configuration activities, leaving out supply and clock settings.

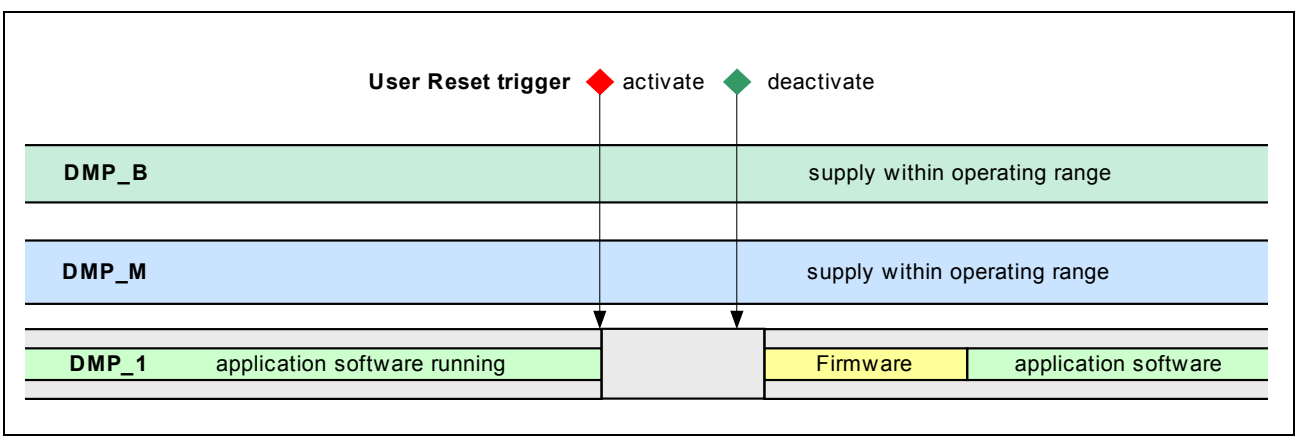

Figure 6-7 Conduction of the User Resets. The supply settings remain unaffected. The gap between de-activation of the reset trigger and the start of the Firmware results from the reset elongation feature of the RSC

Limitations
Because this reset is generated by a synchronous digital logic, it requires a proper clock to be processed.
Especially when triggered from outside, the external event must be long enough to be synchronized to the current system clock. It must be taken into account, that the system clock frequency may vary over the operating time.
Because it does not affect the clock settings, this reset can not be used to resolve any problems related to the clock system.

6.2.5 Application Reset

Class
User Reset

Objective
The purpose of this reset is the same as of the Internal Application Reset, except that it shall not impact the port settings in order to minimize the influence on the external communication interfaces.

Functional Target
The target is the same as for the internal Application Reset, except that the port control logic remains unaffected.

Conduction flow
See Internal Application Reset.

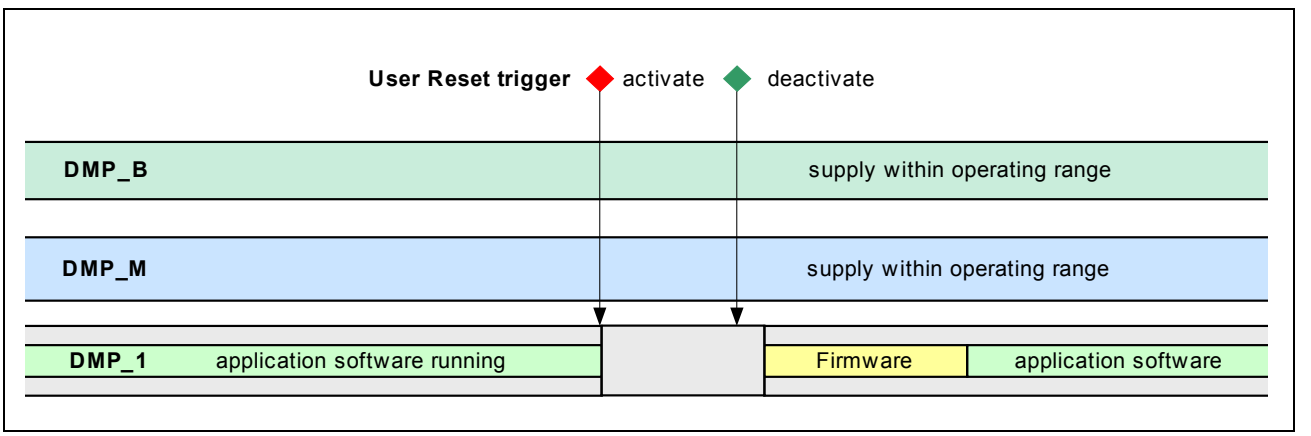

Figure 6-8 Conduction of the User Resets. The supply settings remain unaffected. The gap between de-activation of the reset trigger and the start of the Firmware results from the reset elongation feature of the RSC

Limitations
This reset has the same limitations as the Internal Application Reset.
Since it does not affect the current port settings, it should not be used when the external interfaces of the microcontroller shall be initialized.

6.3 Practical Application of a reset

The Operating Modes are defined by their settings of the clock and the supply system. Thus, it depends on the type of reset whether an Operating Mode will be affected by a reset or not.
As can be seen from **Table 6-2**, all Power Resets imitate an Operating Mode Transition to **Base Mode**, whereas the **User Resets** do not have an impact on the current Operating Mode.

Table 6-2 Impact of Resets to the Operating Mode, and possible Triggers

Reset Type	Effect on Operation Mode	Possible Triggers
Power-on Reset	Transition to Base Mode	PORST
Power Reset for VDDI_M and VDDI_1 supply domains	Transition to Base Mode	Power fail event in VDDI domain
Power Reset VDDI_1 supply domain	Transition to Base Mode	Power save Operating Mode Transition
Internal Application Reset	No impact on current Operating Mode	External pin or bit set or SRST command
Application Reset	No impact on current Operating Mode	External pin or bit set or SRST command

Which reset to use?

At the first glance, it seems that there are quite a number of resets to choose from. But when looking on the possible triggers for the resets in **Table 6-2** and the reset objectives in **Table 6-1**, it becomes evident that the Power Resets are dedicated to resolve or prevent certain kinds of failure modes or irregularities. Thus, only the User Resets are remaining to be triggered out of a functional application context.

Dealing with User Resets

It is important to realize that although the User Resets do not affect the current Operating Mode, they always will abruptly interrupt ongoing activities. If for example a Flash data programming sequence, which just has been started, will be stopped suddenly by a reset which has become active as a reaction to an external reset trigger, the content of the Flash will most probably become incorrect. The variety of the resulting possible failure modes can hardly be foreseen. Hence, triggering a User Reset by an external pin becomes a dubious pleasure when conducting sequential processes. This is the reason why User Resets are disallowed while performing the **Operating Mode Transitions**.

Triggering a reset by software, e.g. by executing the SRST command, allows first to finalize all running actions, such as external communication or Flash programming activities, before the reset will be triggered. Thus, this is the recommended way.

Dealing with Power Resets

Power Resets are actuated by the supply monitoring system automatically and immediately. They also interrupt current activities, but they set back everything inside the affected power domains to a default state including the clock and the supply system. Starting then from **Base Mode** the application software will be able to do some kind of analysis before making use of system resources.

In case the software discovers a severe problem, the recommended action is to activate a **Power-on Reset**. How to trigger a Power-on Reset by software is subject of a dedicated application note.

Regarding **Exception Handling**, see also **Chapter 7**.

6.3.1 How to evaluate the system state after a reset

After any reset, it makes sense first to do some analysis on what was the type of the latest reset, in order to take appropriate action. Therefore, the XC2000/XE166 offers some status information in SFRs RSTSTAT and SWDCON1. What to do after a reset is consolidated in **Table 6-3**.

Table 6-3 Identification of a reset and additional evaluation measures

Type of Reset	Identification and evaluation measure
Power-on Reset	**Identification:** SWDCON1.PON = 1_B RSTSTAT.STM = 11_B RSTSTAT.ST1 = 11_B **Action:** Clear PON bit to be able to identify a Power Reset for VDDI_M and VDDI_1 supply domain
Power Reset for VDDI_M and VDDI_1 supply domains	**Identification:** SWDCON1.PON = 0_B (unchanged after clearing) RSTSTAT.STM = 11_B RSTSTAT.ST1 = 11_B **Action:** No specific further action required
Power Reset VDDI_1 supply domain	**Identification:** SWDCON1.PON = 0_B (unchanged after clearing) RSTSTAT.STM = 00_B RSTSTAT.ST1 = 11_B STMEM0.SSFMOD = 0000_B **Action:** Check the ESR or WUT wake-up source in DMPMIT Check on erroneous return from FSM (identified by STMEM0.SSFMOD = 1111_B

Table 6-3 Identification of a reset and additional evaluation measures (cont'd)

Type of Reset	Identification and evaluation measure
Internal Application Reset	**Identification:** RSTSTAT.ST1 = 00_B RSTSTATx.YY = 10_B **Action:** Check loss of lock trap if running in **Normal Operation Mode**. Check WDT overflow has already occurred (in case double WDT feature is used)
Application Reset	**Identification:** RSTSTAT.ST1 = 00_B RSTSTATx.YY = 11_B **Action:** Same as for Internal Application Reset

Of course, there are additional application specific actions to be performed in order to set up the system properly. This overview puts a spot only on the reset related issues.

7 Exception Handling

This chapter gives an introduction into the field of handling failure modes.

A typical example of an exception, surfacing within a software **Flow**, is that while polling on a status bit within a loop, the expected level does not appear. Without any exception handling, the software will remain in this loop, causing unpredictable errors in the **Application**.
The preferred method to overcome this is to implement a time-out (see **Figure 7-1**).

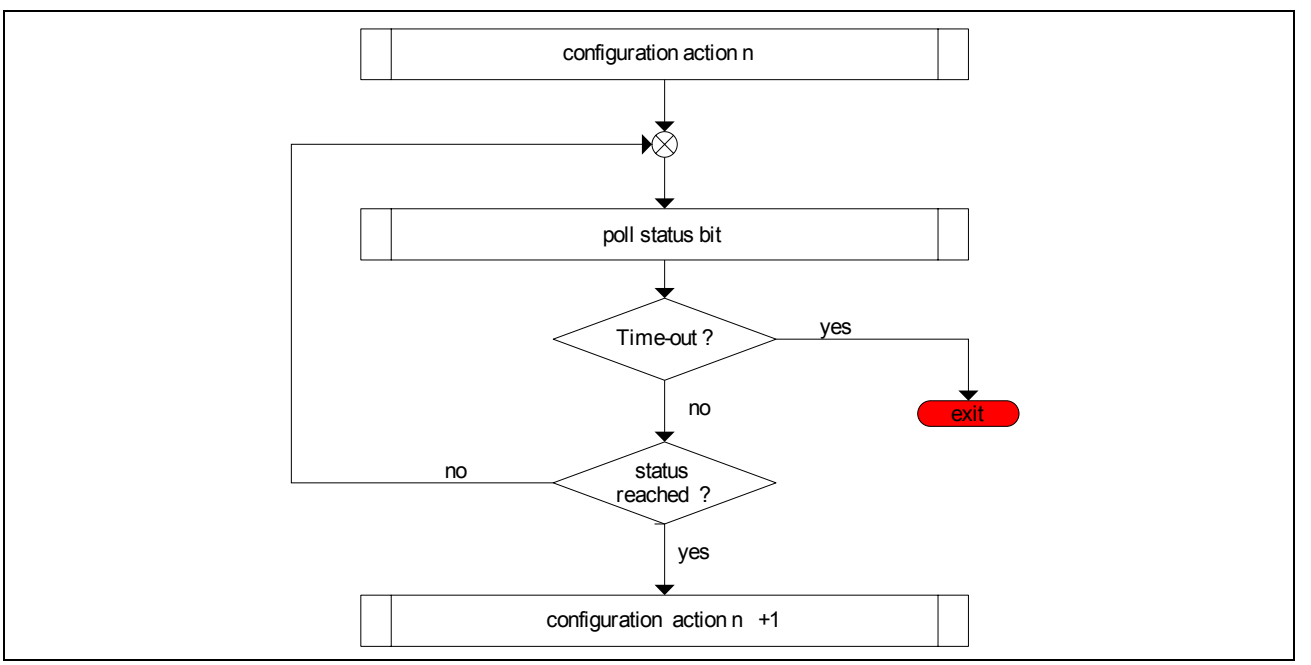

Figure 7-1 Typical exit point from a configuration flow upon a time-out

While this is a well known and simple measure, the question of what to do in case of a time-out really occurs is more tricky to answer. Unfortunately, there is no universal solution for this topic. However, the following sections may help on the way to an exception-safe software.
In the next part, some basic thoughts on error handling, together with an error handling checklist, will be discussed.
If one is very familiar with error handling, and is just interested in a reference of what is offered with the XC2000/XE166 Family architecture or in conjunction with the Operating Mode Transitions, the next two sections could be skipped. Please then continue with **Chapter 7.3** "**XC2000/XE166 Hardware error handling**".

7.1 What to do upon exceptions?

The most important question to raise in conjunction with exception handling is, what really are the needs of the application upon the different kinds of nonconformity in the program execution flow. The spectrum of possible requirements ranges from "**try again"**

via **“do a master reset**” up to sophisticated **analysis and logging** of the root cause of the error.

For support in compiling an error handling list, the next section contains a checklist with typical questions regarding requirements for exception handling.

7.2 Checklist for exception handling requirements

The following questions could be helpful setting up the proper error handling, or generally for checking the consistency of the software requirement list.

- Is there a general error mode which shall be entered upon exceptions?
  In case of “yes”:
  - What shall be the state of the external pins of the chip?
  - Is there a limit for power consumption of the chip?
  - How quickly must the error mode be entered?
  - Which event must be able to terminate the error mode?
  - Is it required that a previous entry into this mode can be detected after leaving the error mode?
- Are hardware actions inside or outside the chip required to resolve or to react on the exception?
- Is there a requirement for the detection/reaction time of a nonconformity?
- Must the error be detectable/visible outside?
- Must a debugging environment be able to follow the error flow?
- What about multiple occurrence of an exception?
- Is there an internal/external software watchdog available?

Independent from whatever list finally comes out, from the robustness point of view, the general design rule for error handling is: **“Keep it simple!**” Consider that upon an exception, one might already have only limited system resources and performance available. So it is worth to stress whether issuing a **Power-on Reset** could be sufficient for as much as possible exception cases. How to trigger a Power-on Reset by software is subject of a dedicated application note.
Regarding resets, see also **Chapter 6** (**Resets**).

7.3 XC2000/XE166 Hardware error handling

This chapter describes the features offered by the XC2000/XE166 architecture to handle hardware exceptions which can not be remedied by software.

7.3.1 OSCWDT clock emergency

One of the worst failure modes one can imagine for an application is that the clock inside the controller suddenly stops. This deadlock will bring any external communication to crash down, and there is no way to remedy this by software. This failure mode can only be resolved by hardware. Because the **WDT** also does not work,

theXC2000/XE166architecture offers the **OSCWDT** feature for detecting and handling a clock stop.

*Note: The **PLL VCO Loss of Lock** feature is used exclusively to detect an unlock of the PLL in **Normal Operation Mode**.*

Operating Modes vulnerable for an OSCWDT failure mode

This error can occur, when the system clock directly depends on an external clock source.

In these modes, a clock signal at the XTAL1 pin drives the system clock either directly or through a divider of the PLL.

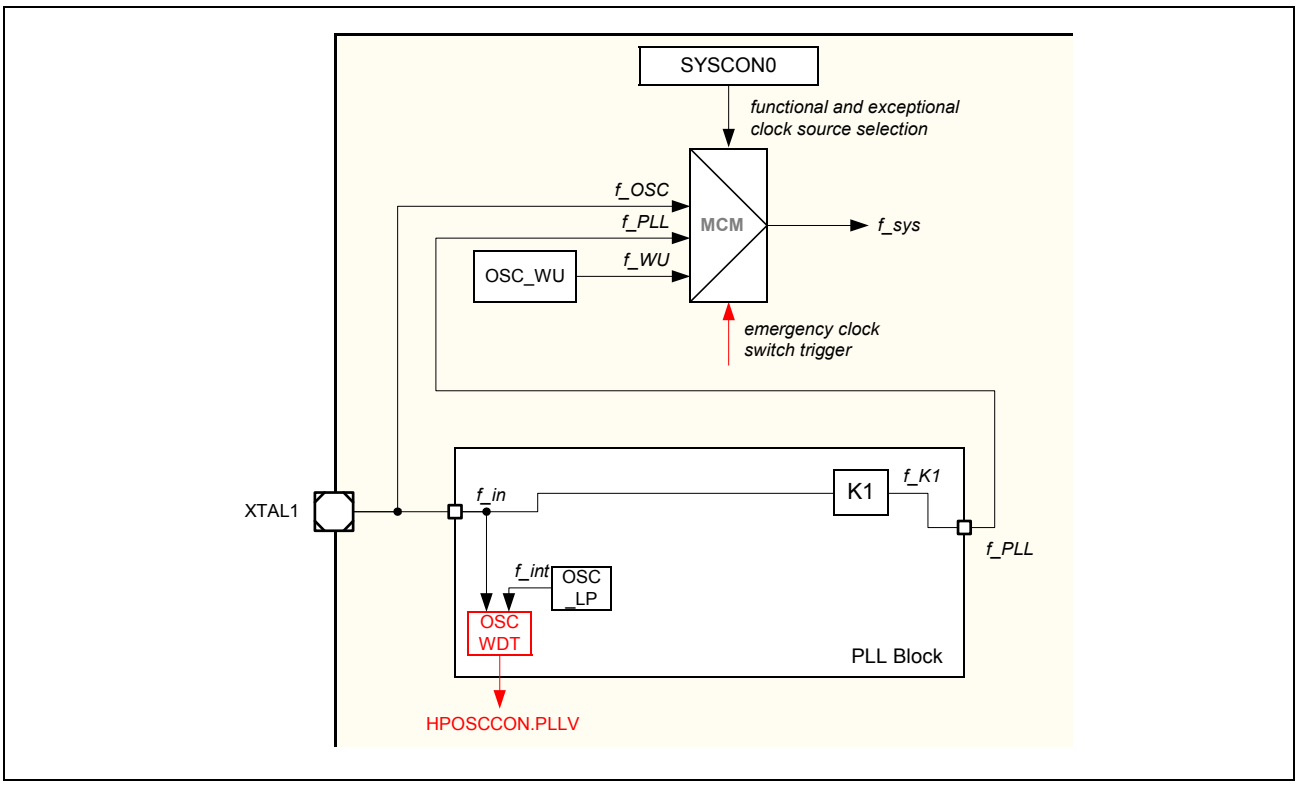

Figure 7-2 OSCWDT clock emergency handling

Functionality of the OSCWDT feature

Figure 7-2 illustrates the components involved in the feature. The OSCWDT permanently checks whether a clock is present at the XTAL1 input. To check the external clock, the OSCWDT unit uses f_int from **OSC_LP** as a reference. The result of the check is flagged and made visible in bit HPOSCCON.PLLV. In case a clock signal is detected at the XTAL1 pin, the flag will be set to “1”.

In case no clock edge of f_int occurred within the OWD latency (see Appendix **Table 11-1**), the flag will be set to “0”. The falling edge of the signal is used to set a trap flag and to trigger an emergency clock switch of the **MCM**. The MCM is configured to switch **OSC_WU** as a backup clock to the system clock tree. This resolves the failure

mode, and the CPU resumes code execution and can operate on the trap flag. Thereby, the application is notified on the exception and can take further actions. **Figure 7-3** shows a typical curve of the system frequency in case of an OSCWDT clock emergency event.

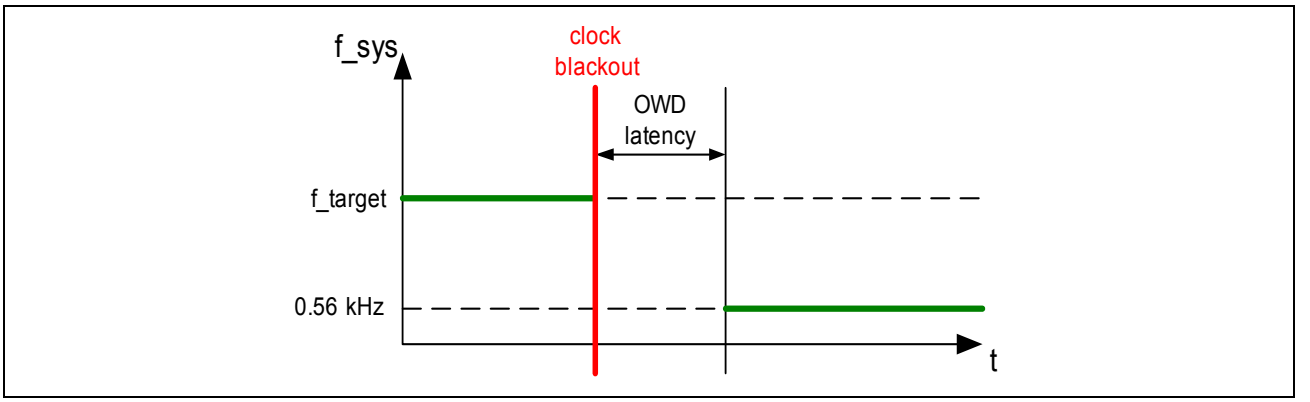

Figure 7-3 OSCWDT clock emergency timing

7.3.2 PLL VCO Loss of Lock

Another, but less harmful, clock system exception is when the **VCO** in the **PLL** faces a loss of lock. If everything is fine, the lock bit PLSTAT.VCOLOCK is set and indicates that the VCO is running on the defined frequency, and thus that the system clock is stable and at the target frequency. An accurate system frequency is important for most external communication protocols.

Without exception handling, an unlock of the PLL can appear permanently or temporarily. While a permanent unlock of the VCO is an indication that something is wrong with the clock at the XTAL1 pin, a temporary PLL unlock may be caused by noise or a disturbance at the XTAL1 pin or on the supply.

In both cases, the system clock is at least for a while not on the target frequency. The primary failure mode indication is the same as for an **OSCWDT clock emergency** case: Most types of external communication will crash down.

However, the big difference to an OSCWDT clock emergency failure mode is that despite the unlock condition, the system at any time has an operable clock, because the system clock is derived from the VCO clock.

In case the VCO unlock is caused due to in improper signal at the XTAL1 pin, the PLL VCO Loss-of-Lock feature is used for a quick disconnect of the VCO from the malfunctioning input, by activating the PLLSTAT.FINDIS bit

*Note: The PLL VCO Loss-of-Lock feature is used exclusively in **Normal Operation Mode**.*

*Note: The **OSCWDT clock emergency** is used exclusively to detect a missing clock at XTAL1.*

Operating Modes vulnerable for a PLL VCO Loss-of-Lock failure

This error can occur when the system clock is derived from the clock of the VCO, while the VCO is locked onto an external clock. The PLL VCO emergency feature can handle loss-of-lock failures for the following Operating Modes:

- **Normal Operation Mode**

In these modes, the system clock is derived from the VCO clock.

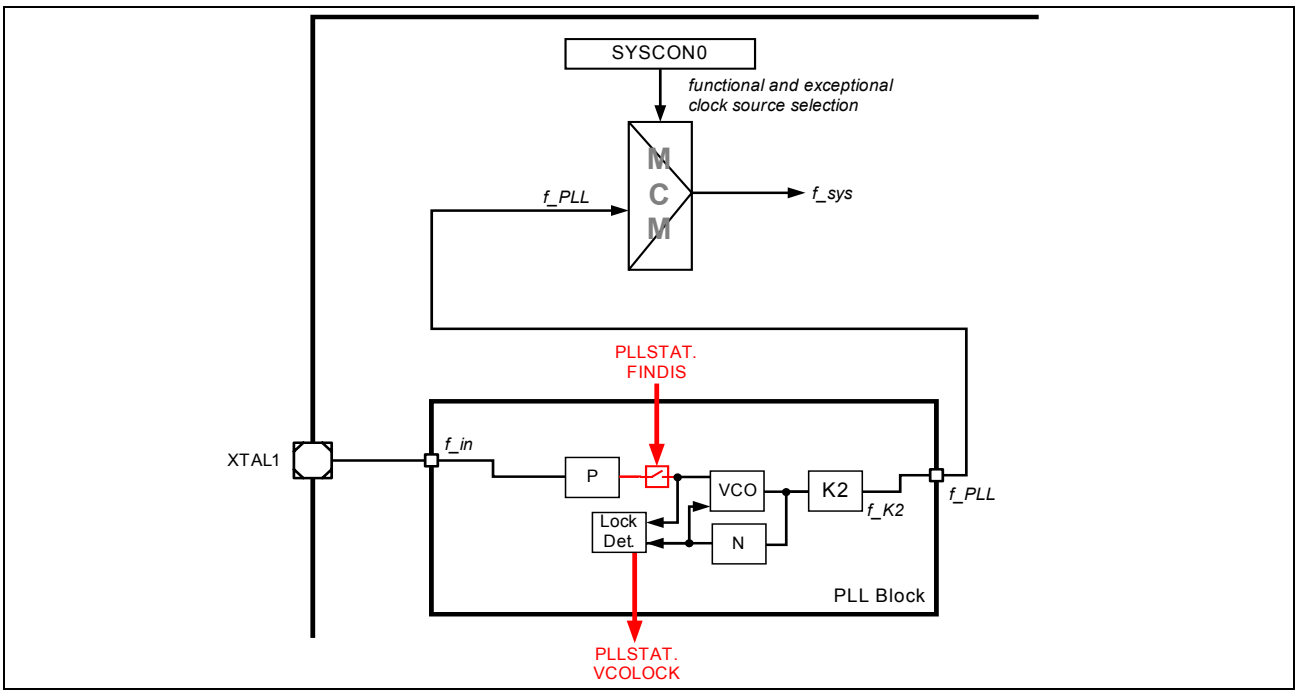

Figure 7-4 PLL VCO Loss of Lock handling

Functionality of the PLL VCO Loss-of-Lock feature

Figure 7-4 illustrates how the feature works. The PLL Lock Detection permanently checks the phase between the clock from the N-divider and the VCO reference clock input. If the phase difference is within the allowed range, the lock is indicated by setting bit PLLSTAT.VCOLOCK. In case the phase difference is too big, this is detected after a certain latency; bit VCOLOCK is cleared, a trap flag is set, and the input of the VCO is permanently disconnected from its input, which is indicated with bit PLLSTAT.FINDIS. The disconnection remains active until a “1” is written to bit STATCLR1.CLRFINDIS by software.

Figure 7-5 shows a typical curve of the system frequency, and the status of related bits upon a VCO Loss-of-Lock event. It is assumed that, due to the nonconformity of the system clock, an external communication has been corrupted anyway, thus, keeping the VCO input disconnected does not make the situation worse. In contrary, the permanent disconnection of the VCO prevents re-iterated Loss-of-Lock events, and keeps the system frequency free from possible disturbance at the XTAL1 pin.

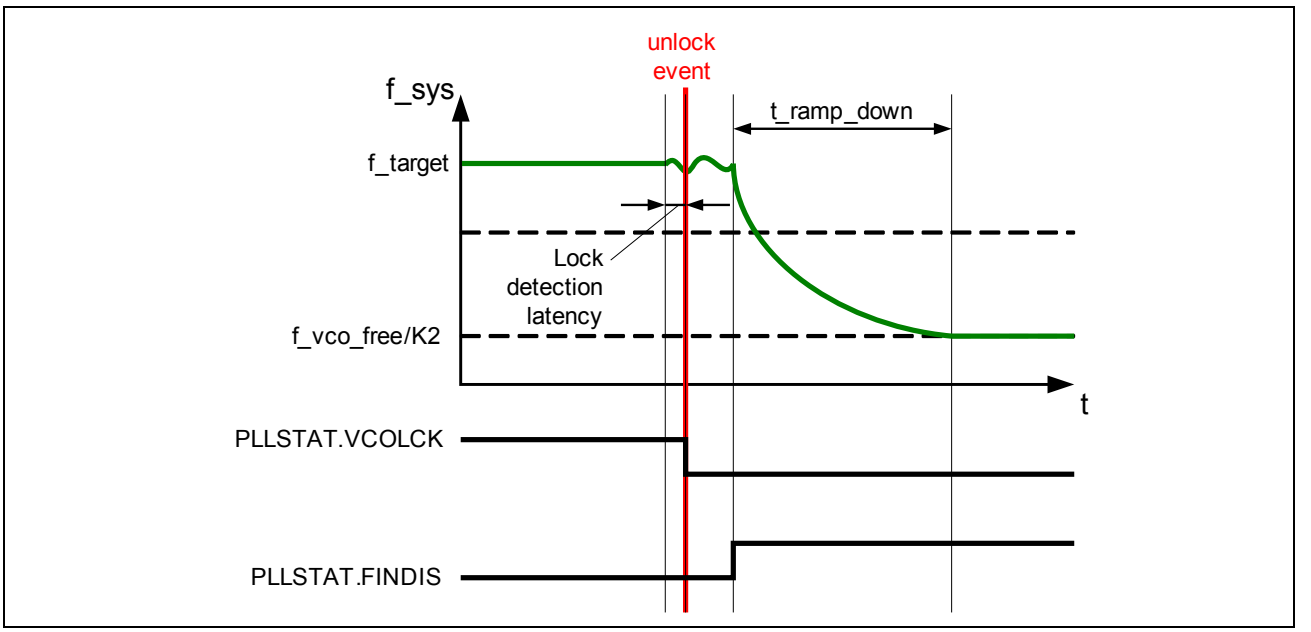

Figure 7-5 PLL VCO Loss of Lock timing

Configuration

The PLL VCO Loss-of-Lock feature is configured by function **EnableVcoLockEmerg** and **DisableVcoLockEmerg**, which are embedded in the respective Operating Mode Transitions.

7.4 Software error handling for the Operating Mode Transitions

This section is about handling software exceptions in conjunction with the Operating Mode Transitions.

7.4.1 Disrupting sequences

The Operating Mode Transitions, and also some basic **ELF** routines defined in this document, offer exit points after time-outs.

The Operating Mode Transitions in this document generally are not designed for **Reentrancy**. This means, that a sequence will only run properly under dedicated entry conditions. If one stops the flow in between in order to start it again from the beginning, the configuration will most likely fail.

This is the main reason why **User Resets** are not allowed while conducting the configuration **Flow**s. In contrast, **Power Resets** are allowed and armed for the Mode Transition flows, because they set back also the settings of the **Analog Component**s.

7.4.2 Exception handling for Operating Mode Transition software

Generally, the Operating Mode Transition sequences define no error handling. They only define exit points for further error handling. As already mentioned in **Chapter 7.2**, it is

recommend to trigger a **Power-on Reset** upon exceptions surfacing at runtime, as long as no other requirements are known.

However, to support in analysis of root causes of possible exceptions, **Table 7-1** lists all emergency exists defined for the Operating Mode Transitions, and gives proposals what can be investigated upon on a specific error at runtime or for analysis. If no runtime proposal is given, issuing a **Power-on Reset** is the recommended action.

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions

#	Context, Exception, Indication and Recommendation	
1	Context	Number of attempts to check f_osc_hp valid state has been exceeded
	Indication	-
	Exception	Crystal does not operate properly
	Recommendation	This indicates an improper or disturbed connection to the crystal. **For analysis:** • Check whether crystal is properly connected to XTAL pins • Check whether crystal is not broken
2	Context	Check for respective trap/interrupt request flag to be set after wakeup with selected trigger
	Indication	DMPMIT.ESR0T = 1_B, DMPMIT.ESR1T = 1_B, DMPMIT.ESR2T = 1_B, DMPMIT.WUI = 1_B
	Exception	Wakeup was from a different source than expected (not possible in case only one wakeup trigger is selected)
	Recommendation	This indicates noise or unexpected traffic on the inactive wakeup pin. **For analysis:** • Disable respective wakeup trigger

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
3	Context	Poll for system to enter Normal Mode
	Indication	GSCSTAT.CURRENT = 00_B
	Exception	Time-out occurs, because GSC is waiting for response from a slow or not responding peripheral
	Recommendation	This indicates noise or unexpected traffic on the inactive wakeup pin. **For analysis:** • Increase time-out
4	Context	Poll for system to enter Clock-off Mode
	Indication	GSCSTAT.CURRENT = 10_B
	Exception	Time-out occurs, because GSC is waiting for response from a slow or not responding peripheral
	Recommendation	**For analysis:** • Increase time-out **During runtime:** 1. Set back GSC to Normal Mode and terminate peripheral activities step by step 2. Try again GSC clock-off
5	Context	Poll for f_osc_hp to become stable, and therefore valid, for PLL input
	Indication	HPOSCCON.PLLV = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before, or the clock input faced a temporary disturbance
	Recommendation	**For analysis:** • Check whether crystal is properly connected • Change crystal • Change type of crystal **During runtime:** • Check oscillator settings in HPOSCCON • Ensure that required conditions are met • Try once again

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
6	Context	Poll for FLASH to be powered up
	Indication	IMB_FSR_BUSY.BUSY = 000_B
	Exception	• One or more Flash modules did not receive a wakeup trigger (1) • Hardware problem in Flash (2)
	Recommendation	**For analysis:** • Check FL_KSCCFG (1) **During runtime:** • Perform error handling routine in SRAM (2)
7	Context	Check status of power related interrupt request flags
	Indication	INTSTAT.PVCMI1 = 1_B, INTSTAT.PVCMI2 = 1_B, INTSTAT.PVC1I1 = 1_B, INTSTAT.PVC1I2 = 1_B
	Exception	Interrupt occurred during power transition, because power supply dropped below monitored limit
	Recommendation	**For analysis:** • Monitor VDDI supply during the transition **During runtime:** • Check critical data in RAM • Copy code back from FLASH to RAM
8	Context	Required for handshake to set a new K1DIV value
	Indication	PLLSTAT.K1RDY = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
9	Context	Required for handshake to set a new K1DIV value
	Indication	PLLSTAT.K1RDY = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
10	Context	Required for handshake to set a new K2DIV value
	Indication	PLLSTAT.K2RDY = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
11	Context	Required for handshake to set a new K2DIV value
	Indication	PLLSTAT.K2RDY = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
12	Context	Required for handshake to set a new NDIV value
	Indication	PLLSTAT.NRDY = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
13	Context	Required for handshake to set a new NDIV value
	Indication	PLLSTAT.NRDY = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
14	Context	Poll for OSC_HP to be selected as PLL input clock
	Indication	PLLSTAT.OSCSELST = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
15	Context	Poll for OSC_HP to be selected as PLL input clock
	Indication	PLLSTAT.OSCSELST = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
16	Context	Required for handshake to set a new PDIV value
	Indication	PLLSTAT.PRDY = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
17	Context	Required for handshake to set a new PDIV value
	Indication	PLLSTAT.PRDY = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
18	Context	Poll for VCO bypass status Prescaler Mode to be entered
	Indication	PLLSTAT.VCOBYST = 0_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
19	Context	Poll for VCO bypass status Unlocked/Normal Mode to be entered
	Indication	PLLSTAT.VCOBYST = 1_B
	Exception	Time-out occurs, because not all required conditions have been met before
	Recommendation	**For analysis:** • Ensure all required conditions of the ELF flow are properly met. Try again **During runtime:** • In case all conditions are met, these exceptions point to a hardware error. Further actions depend on the requirements of the application. The recommendation is to do a **Power-on Reset**
20	Context	Poll for VCO locked state
	Indication	PLLSTAT.VCOLOCK = 1_B
	Exception	• VCO can not lock using current settings (1) • Oscillator clock is not present (2)
	Recommendation	**For analysis:** • Check on correct PLL settings and oscillator settings (1, 2) **During runtime:** • This problem can be caused by a noisy environment, which prevents the VCO from locking. A first aid could be to reduce system activity, especially toggling of ports (1) • If the VCO permanently fails to lock, the crystal may not operate properly. Check oscillator status in HPOSCCON.PLLV, and the respective history bits HPOSCCON.OSC2LO1 and HPOSCCON.OSC2LO0 (2)

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
21	Context	Poll for last power transition sequence performed to be seq. B, to ensure the next sequence will be seq. A
	Indication	PSCSTAT.LSTSEQ = 1_B
	Exception	Sequence B is still running
	Recommendation	**For analysis:** Check PSCSTAT.PSMSTAT, whether a transition is still running: • If not, increase time-out value • If yes, provide external supply of 1.5 V and step through the transition using a debugger **During runtime:** • Check PSCSTAT.PSMSTAT, whether a transition is still running • This is an indication that something went wrong with the supplies at wakeup. The recommendation is to do a **Power-on Reset**
22	Context	Poll for seq. B to be started after trigger
	Indication	PSCSTAT.PSMSTAT = 000_B
	Exception	• Power transition state machine waits for a supply level condition which still has not occurred (1) • Extra waiting cycles have been inserted in the sequence flow (2)
	Recommendation	**For analysis:** • Check VDDI_1/VDD_M supplies (1) • Increase time-out (2) **During runtime:** • The recommendation is to do a **Power-on Reset**
23	Context	Poll for seq. A to be started after trigger
	Indication	SEQCON.SEQAEN = 0_B
	Exception	Other start trigger selected than software, and trigger not yet occurred or not enabled
	Recommendation	Check SEQCON. Other triggers are not recommended in this context

Table 7-1 List of error exits of Operating Mode Transitions and proposals for further actions (cont'd)

#	Context, Exception, Indication and Recommendation	
24	Context	Poll for seq. B to be started after trigger
	Indication	SEQCON.SEQBEN = 0_B
	Exception	Other start trigger selected than software, and trigger not yet occurred or not enabled
	Recommendation	Check SEQCON. Other triggers are not recommended in this context

8 Restrictions handling specific components

In the following, a list of really non-negotiable “must do” and “don't do” rules for dedicated **Component**s are given.

Besides the general hints for programming, there are also some “hard” restrictions resulting out of limitations or individual properties of digital or analog **Component**s. Violating these restrictions will lead to improper or instable functionality of the clock or the power supply system of the XC2000/XE166. The Operating Mode Transitions documented inside this Programmer's Guide – of course – take these restrictions into account, so the following items must only be known in case one needs to modify a transition flow. However, being familiar with these restrictions can be helpful in understanding the sequence applied for the mode transitions.

For the following items, the sequence of the rules is not related to any priority. In fact, the rules have been numbered to be able to reference to them.

8.1 Master clock multiplexer (MCM)

This lists requirements regarding switching between the system clock sources using the **MCM** by programming register SYSCON0.

- To perform a synchronous clock switch both, the current system clock as well as the new target clock must be available.
  In case a clock switch has been triggered, the system will be disconnected from the current clock source independent from whether there is a target clock available or not.The system will remain unlocked in this case.
- A new synchronous switch must not be triggered before a preceding one has been terminated. This is required to remedy pipeline effects, since a switch request must be properly synchronized to the new target clock. This may take a comparably long time, in case the new target clock is very much slower than the current clock.
- The rules for frequency stepping must be applied (see **Chapter 8.5.3** on **EVR**).
- For clock emergency conditions, where the current clock has completely disappeared and MCM performs an **OSCWDT** emergency clock switch (see **Chapter 8.4.2 Oscillator watchdog operation** and **Chapter 7.3.1 OSCWDT clock emergency**) there is no restriction regarding the frequency ratio between the target clock and the backup clock. But one should be aware that a sudden stop of the system clock may lead to a voltage overshoot mainly at the at the VDDI_1 supply. The resulting level of this overshoot depends on the actual system clock frequency before the fail condition has been occurred and it is limited by the external capacity.

8.2 Temporary clock off feature of the PSC stepper logic

STEP0 register provides clock enable bits for DMP_M and DMP_1. This logic controls the enable input of the master clock gates inside the power domains.

The clock-off bits only have an effect, when bitfield STEP0.TRGSEL is not equal to zero.

8.3 Clock sources

Generally, activating and de-activating of clock sources, which currently directly drive the system clock tree, holds the risk that clock spikes are generated and propagated on the clock path, resulting in an unpredictable system state.

When clock settings are modified in a way, which results in a change of frequency, the rules of frequency stepping (see **Chapter 8.5.3** on **EVR**) must be taken into account. This is especially valid for switching between different clock sources using the **Master clock multiplexer (MCM)** and other non scalable **Clock source multiplexers**.

8.3.1 Trimmed Current Controlled Wakeup Clock Source (OSC_WU)

The following rules apply to Trimmed Current Controlled Wakeup Clock.

- The value of bit field FREQSEL must not be changed when the wakeup clock is used as source for the system clock

8.3.2 High Precision Oscillator (OSC_HP)

The following rules relate to the OSC_HP:

- Any operation of OSC_HP requires to have the shaper activated. Therefore bit HPOSCCON.SHBY must be cleared.
  Operation with the shaper in bypass can lead to an improper clock signal at the output of OSC_HP. This may cause the PLL to become temporarily unlocked or it may even lead to an unpredictable system behavior in when the OSC_HP is the direct source of the system clock.
- Any modification of control inputs is critical regarding the clock output of the operating oscillator, especially when an external clock is connected to the XTAL1 input.
- Activating the oscillator with an external crystal connected can be done at any time, independent of whether the oscillator is started the first time or whether it has been de-activated lately.
- Before using the clock of OSC_HP in conjunction with an external crystal connected, a proper oscillation must be achieved. This means, that the amplitude of the oscillation has to be above the input hysteresis of the oscillator unit. This is assumed to be the case for standard crystals with a frequency of up to 16 MHz after about 1024 oscillator clock pulses. The software can implement a timing loop for that covering 1024 (2^{10}) oscillator clock cycles or it must be assured otherwise by the software flow. The state when 1024 oscillator clock cycles have occurred is called oscillator lock.

8.3.3 PLL

The **PLL** consists out of various sub **Component**s, some of them requiring a dedicated handling.

8.3.3.1 PLL Regulator

The **Analog** parts of the PLL (**VCO**, **OSC_LP**) are running on a 2.5 V supply. This supply is generated by a dedicated regulator integrated within the PLL unit. The next rules are dealing with controlling this regulator.

- For the analog 2.5 V supply, the regulator must be enabled separately before the analog blocks of the PLL are activated, i.e. Trimmed Current Controlled Clock and VCO must be kept off until the 2.5 V supply is stable.
- After activation, the PLL regulator will need its ramp-up time to properly ramp-up the analog PLL supply.
- VCO and Trimmed Current Controlled Clock may be activated together.
- When the regulator shall be disabled in conjunction with a power down of the PLL digital part, it must be taken into account that the digital part needs an active clock at the output of the PLL to ramp down. In case this clock is generated by one of the PLL oscillators, power down of PLL must be entered before the regulator is disabled.
- VCO and Trimmed Current Controlled Clock may be switched off together.

8.3.3.2 Trimmed Current Controlled Clock Source

The rules for the Trimmed Current Controlled Clock Source provides the internal clock fINT:

- To be able to use Trimmed Current Controlled Clock as a clock source for VCO or the system clock tree, bit PLLCON1.AOSCSEL must at least once be set to 1 to force the input clock selector of the PLL to Trimmed Current Controlled Clock.
- When running on K1 divider with an external clock source, switching to Trimmed Current Controlled Clock by setting PLLCON1.AOSCSEL will lead to clock spikes, which may result in an undefined behavior of the microcontroller.

8.3.3.3 VCO disable/enable

When enabling or disabling the VCO, the following applies:

- The moment the VCO is disabled, the output clock multiplexer of the PLL will be asynchronously forced to K1 clock signal. This may result in clock spikes on the PLL output f_PLL. These spikes can be propagated to the system clock path, in case the PLL is used as system clock source and the VCO is currently selected as clock source.
- Re-enabling of the VCO under this condition starts a synchronous switch of the clock multiplexer to the K2 divider output. If in this case no clock is available at K1 divider output, the switch to the VCO will not be accomplished.

8.3.3.4 VCO band

A change of the VCO band leads to an asynchronous frequency jump of the VCO. Clock spikes and improper clock pulses are to be expected. VCO band selection must only be done when the VCO is not involved in generating the current system clock.

8.3.3.5 VCO bypass with external clock selection

If VCO bypass is active together with a selection of the external clock and the K1 divider is set to 1_B, PLL works as a single wire bypass. Any improper clock signal from the external clock will directly be propagated onto the system clock path.
Operating the PLL in this way may add additional noise on the clock signal on its way through the PLL unit.

8.3.3.6 LOCK operation

The following paragraphs provide more information concerning the lock operation.

- The **VCO** has a non asymptotic frequency response curve when going to the lock state. This means that up to 10% frequency overshoot must be expected when entering the lock state.
  When operating the system at the maximum operation frequency, the output of the VCO must be pre-divided until the overshoot condition has been gone.
- VCO lock operation requires a locked oscillation of the crystal. See **Chapter 8.3.2** on **High Precision Oscillator (OSC_HP)** for details.
- The PLL lock detection has a certain inertia and its lock flag is synchronized to the system clock. Thus, after a modification of the PLL that impacts the lock state, one must be aware that the state of the lock flag will not be valid until the lock detection has recognized the new condition and the state has been synchronized onto the system clock path. This is especially true after a lock detection reset.
- For lock detection time-out, the maximum time-out value shall be used (see **Table 11-1**).
- The VCO will only lock properly when the input frequency is between 4 MHz and 16 MHz **at the VCO input (i.e. after the P-divider)**.
- See also shaper operation in conjunction with **High Precision Oscillator (OSC_HP)** in **Chapter 8.3.2**.

8.3.3.7 Divider handshake

The PLL provides several handshake interfaces for dividers and multiplexers. This section describes how a handshake is to be conducted upon a change of configuration.

The general conduction of the handshake is the same for all interfaces. However, a sample sequence is described here in conjunction with re-programming a divider.

Attention: The described handshake only works if the new setting (e.g. divider value) changes the current value upon the handshake, since PLL only evaluates a change of its actual configuration value but not the write trigger to an SFR.

The handshake must be performed with the following four steps:

1. Clear acknowledge bit together with setting the new divider value.
2. Poll on ready bit to be 0_B.
3. Set acknowledge bit.
4. Poll on ready bit to be 1_B.

This approach will even work in case the handshake has not been properly served before, and ready is already at 0_B from the beginning. In any case, a change of the divider value will set ready to 0_B.

Note: When polling a status bit working with a time-out action is strongly recommended.

8.3.3.8 K-Divider

These are the rules for handling the K-divider.

- The K-divider handshake must be performed according to the description in **Chapter 8.3.3.7**.
- Programming the K-Divider must not be done together with other **VCO** related settings of P, N, VCO band, or setting bit FINDIS. Together means executing subsequent commands before the handshake has finished.
- When running on VCO, e.g. in lock operation, and the lock frequency shall be reduced, the VCO may need a certain time before the new (lower) target frequency has been reached. This may lead to on overclocking situation, because the switch to the new K-divider value is performed faster than the VCO frequency is going down.
- Since the VCO delivers a clock with a worst case duty cycle of about 70/30, the recommended minimum value for K2 is 1_B (resulting in a divider factor of 2).
  An asymmetrical duty cycle of the system clock is disadvantageous regarding EMC.

8.3.3.9 P-Divider

These are the rules for handling the P-divider.

- The P-divider handshake must be performed according to the description in **Chapter 8.3.3.7**
- The minimum input frequency of the **VCO** is 4 MHz. Thus, in conjunction with an external clock source driving the VCO the value for the P-divider must follow the rule f_in / (P+1) > 4 MHz.
- For optimization towards low power consumption, the reference frequency f_ref = f_in / (P+1) should be configured as low as possible, i.e. ideally 4 MHz.
- For optimization towards low jitter in conjunction with an external crystal, the reference frequency f_ref = $f\_crystal$ / (P+1) should be configured as high as possible, i.e. ideally 16 MHz.

8.3.3.10 N-Divider

These are the rules for handling the N-divider.

- The N-divider handshake must be performed according to the description in Chapter 8.3.3.7
- Depending on the currently selected VCO band, the N-divider value may be chosen within the range defined in the User's Manual.
- For optimization towards low power consumption, the N-divider value should be chosen as low as possible.
- For optimization towards low jitter, the N-divider value should be chosen as high as possible.

8.3.3.11 Clock source multiplexers

A handshake according to the sequence described in Chapter 8.3.3.7 has to be performed when using the VCO clock multiplexer or the clock input multiplexer to assure that the switch really has been performed properly. Just like the MCM the clock source multiplexers inside the PLL require to have both clock enabled to perform a clock switch action.

8.3.3.12 PLL sleep

Disabling PLL is started by setting bit PLLCON1.PLLPWD to 1_B. This request is synchronized inside the PLL to the frequency at the output of the K2-divider.

- When using the PLL power down feature, it must be taken care of that the PLL output carries a clock. Otherwise, the power down request can not be synchronized.
- When the PLL regulator shall also be powered down, the requirements in Chapter 8.3.3.1 must be taken into account.
- To allow a wakeup when off, setting bit PLLCON1.PLLPWD to 0_B is asynchronously followed.
  This leads to problems in case a wakeup occurs just at the time the activating sleep edge has just been synchronized. The PLL might output spikes at the clock output.
  When the sleep mode is entered inside PLL all configuration inputs will be accepted asynchronously (sort of a preset). When other input parameters at the PLL interface changed again together with the wakeup, it must be taken into account that the new values may be active immediately. This is especially true for new K-divider values, where a handshake is not applicable.
  This means, that software should assure that the PLL has really entered its power down mode, before it is re-enabled again.

8.4 Clock emergency handling

The following chapters deal with handling clock emergency cases. Chapter 7 on Exception Handling explains the context in more detail.

8.4.1 General issues

Upon an **OSCWDT clock emergency** event, **MCM** will have switched to the backup defined by SYSCON0.EMCLKSEL. In case a sw **Flow** is going to resolve this state by clearing the flags in SYSCON0 writing to STATCLR0, it has to be taken care of that MCM will switch back to the normal clock selection defined by bitfield CLKSEL. Thereby SYSCON0.CLKSEL may need to be redefined before in order to follow the frequency stepping rule (see **Chapter 8.5.3** on **EVR**).

8.4.2 Oscillator watchdog operation

Supervision of oscillator frequency is done using **OSC_LP** as reference clock source. The detection flags frequencies below 300 kHz at the external clock input of the PLL.

- When using OSC_HP as external clock source for PLL, it has to be taken care of that the **OSCWDT** logic is not armed (especially traps shall not be enabled) before OSC_HP is in locked state.
- An emergency clock switch (which shall be conducted upon an **OSCWDT clock emergency** event) additionally requires bit HPOSCCON.OSC2L0 to be set.
  This bit carries the information that at least once, a proper clock has been detected at the oscillator output.
- Since the system reaction time upon an oscillator fail event is considerably long (minimum 11 µs) compared to the time constant of the EVR, increasing the system clock upon an **OSCWDT clock emergency** event has to be considered as a ramp-up of the system frequency f_sys from zero.
  Thus, according to the frequency stepping rules (see **Chapter 8.5.3** on **EVR**), the backup clock for the oscillator must be below 1 MHz. The suggestion is to use 500 kHz from **OSC_WU**.
- The **OSCWDT** logic must be deactivated before the oscillator is switched off to prevent traps being issued.

8.4.3 PLL Unlock

Supervision of **VCO** lock operation is done using a dedicated lock detection logic inside **PLL**.

- It has to be taken care of that the PLL lock detection logic is not armed (especially traps shall not be enabled) before **OSC_HP** is in locked state (see **Chapter 8.3.2**) and the VCO has locked (see **Chapter 8.3.3.6**).
- The PLL lock detection must be deactivated anytime a configuration is changed, which may impact the VCO frequency in PLL normal mode. Note that a VCO unlock can also be originated by modifying the settings of OSC_HP or any other driving clock sources.

8.5 Power system related resources

The following chapters discuss issues related to the power system resources.

8.5.1 SWD

The Supply Watchdog is controlled by registers SWDCON0 and SWDCON1.

- To (re-)configure comparator levels in the LEVxV bit fields and/or respective actions via the LxACON bit fields of register SWDCON0, any resulting activity must be disabled before. This is also true, when re-configuring the level of a currently unused second comparator.
- For a time of about 5 µs around reconfiguration the comparator level(s), all corresponding hardware events must be kept de-activated. This, in fact, will result in an observation gap for the VDDP supply. But since the probability of unwanted resets caused by reconfiguration is expected to be much higher than the probability of a (relatively fast) voltage drop at the time the configuration takes place, this is considered as to be acceptable.

8.5.2 BG_HP

The control interface for the **HP Bandgap** is register EVRMCON1. In conjunction with some power modes, it is required to disable the HP Bandgap in order to save its **Base Load Current**.

- When HP Bandgap is re-enabled, the software must take care of that it needs time to ramp-up (see **Table 11-1**) before the reference voltage is valid and usable. Especially the **Flash** must not be activated before HP Bandgap delivers a valid signal.
- In case the trimming value of BG_HP shall be changed, other mixed signal modules, such as **PVC** and **EVR**, must not be configured to run on HP Bandgap. Flash must reside in IDLE mode, neither read nor write access is allowed.
- Currently in the firmware, an adjustment value is read out of the configuration sector of the Flash and copied to the adjustment bitfield of the bandgap interface. This is no problem as long as the current value (i.e. the reset value after startup) is copied. Writing other values may be critical, when the Flash is not in standby mode. For the existing firmware, this has been evaluated and no problem has been identified. However, this requirement should be taken into account.

8.5.3 EVR

Handling of the EVR is discussed below.

- **Frequency stepping**
  Please also refer to condition **Smooth system clock frequency stepping must be applied** in **Chapter 9.8.2.2**.
  In order to achieve a smart load regulation of the **EVR**, the system frequency must

be changed only in consecutive steps.
Between the steps, a certain time must be given to allow the regulator to re-adjust.
Please check on “regulation loop” in **Table 11-1**.

- **Ramp-up**
  To implement a “soft ramp-up”, an EVR must always be enabled together with its **LPR**. This makes use of the ramp-up of the LPR to bring up the EVR smoothly. Thus, disabling EVR with the option to re-enable it requires that the LPR is disabled, too.
  – the target voltage level must be reachable upon a voltage ramp-up.
  – the target voltage may never be reached upon a voltage ramp down (e.g due to low leakage or external supply)
  – in case the Current Control logic (**CC**) shall be switched off, the respective current control level must be reduced to 00_B before. Respectively, a valid value has to be set again before the CC is re-enabled.
  Failure to take this into account will result in a loss of power in the respective VDDI domain the time the EVR is re-enabled again.
- **Ramp-down**
  The target voltage level may never be reached upon a voltage ramp down within the expected time frame e.g due to low leakage, a comparably big external capacity or an external supply connected to the VDDI pins.

8.5.4 PVC

In order to re-configure the comparator levels in the LEVxV bitfields, the software must first disable any activity via the LxACON bitfields before the levels are changed. This is also true when re-configuring the level of the currently not used second comparator, because both comparators use the same reference, and changing the settings influences the reference itself.

- For a period of about 2 µs around re-configuration of the comparator level(s), all corresponding hardware events are not reliable and must be de-activated.
- For a period of about 12 µs after re-enabling the PVC, its comparator output is unreliable.

This, in fact, will result in an observation gap. This is accepted so far, since the probability of unwanted resets caused by re-configuration is expected to be much higher than the probability of a (relatively fast) voltage drop at the time the configuration takes place.

8.5.5 LPR

LPR issues are handled in the following sections.

- When the low power reference is disabled in conjunction with a power transition, the **PVC** must also be switched off and re-enabled together with the **LPR** or later.
  Otherwise, if the PVC is kept running, it may flag an erroneous OK upon a re-activation of the power domain.

By re-enabling PVC together with the low power reference, the ramp-up time of the low power reference is covered by the ramp-up time of the PVC.

- When the reference is switched on-the-fly in order to switch between LPR based and **HP Bandgap** based supply setup, data path transitions must be avoided for about 20 µs.
  For DMP_1, where timing paths are more complex, the clock is expected to be switched off, e.g. by using bitfield STEP0.V1
  For DMP_M, which has less complex timing paths, the sequencing logic can be kept running on the system clock. This is expected to be less critical, and considered to be a theoretical risk.
  To assure a minimum supervision at the transition between the references, the supervision level for reset is expected to be set to 0.9 V.

8.5.6 PSC

Rules for PSC handling can be found below.

- Enabling and triggering of a **PSC** Power Transition Sequence (SEQCON.SEQxEN = 1_B / SEQCON.SEQxTRG = 1_B) must not be done at the same time using a single instruction. A sequence must always be triggered after it has been enabled.
- Power Transition value sets must not be programmed while a sequence is enabled, i.e. bits SEQCON.SEQxEN must be cleared.

8.5.6.1 Check whether a power transition has been terminated

When the software shall check whether a transition has been terminated, bits SEQCONxEN and bitfield PSCSTAT.PSMSTAT must be used. Depending on the transition to be checked, x stands for A or B, respectively. The following sequence will deliver a reliable information on whether a transition flow has been finished.

1. Read and check SEQCONxEN = 0_B with time-out.
   This bit will be 0 always 3 system clock cycles after the sequence has been started. The bit remains permanently cleared after the transition has been started.
2. Read and check PSCSTAT.PSMSTAT = 000_B with time-out.
   The time-out applied to this check depends on the current action to be performed and may depend on an analog delay within the stepping sequence (e.g. ramp-up/down of a domain).
   The maximum ramp-up time of the DMP_1 domain is considered to be less then 40 µs. But such a transition can not be polled by the CPU.

Note that the above order of the read accesses is essential to avoid logical glitches.

8.5.6.2 Dummy B-sequence

Executing a B-sequence requires that an A-sequence has been finished before. This interlock condition is used to prevent an “early wakeup condition”, where a B-sequence is triggered for a “wakeup” already before the A-sequence that will enter a power saving mode.
In case an A-sequence has been performed without a succeeding B-sequence, e.g. to perform a switch to **HP Bandgap**, the condition to execute a B-sequence is already valid. If now a new sequence flow shall be performed, which consists of entering a power saving mode with sequence A and leaving it with sequence B, the interlock is already released and the B-sequence might be started before the A-sequence, e.g. when triggered by an ESR pin.
To prevent this, the interlock must be re-installed by applying the following flow:

1. Check if PSCSTAT.LSTSEQ = 0_B, i.e. check whether the last sequence has been an A-sequence.
   If yes, perform the following flow.
2. Configure SEQBSTEPx.SEN = 0_B (x = 1..6) to disable all copy steps, and thus to prevent any changes to be applied to the power system.
3. Configure GSC to be bypassed with SEQCON.GSCBY = 1_B.
4. Disable all hardware triggers for a B-sequence with SEQCON[11:7] = 00000_B.
5. Enable sequence B with SEQCON.SEQBEN = 1_B.
6. Trigger the B-sequence with SEQCON.SEQBTRG = 1_B.
7. Check PSCSTAT until LSTSEQ = 1_B, i.e. the dummy B-sequence has been performed.

This will re-install the sequence interlock, i.e. now, sequence B will not be executed before an A-sequence has been terminated properly.
Step 1 shall be performed before any power saving transition flow which is going to use both sequences.

8.6 GSC

The following discusses GSC related issues.

- GSC must only be configured to run at an power transition sequence when a clock is available in DMP_1 at the end of the transition. Otherwise, the transition might hang-up, waiting for system components or peripherals, since GSC does not implement a time-out.
- Using the GSC may lead to comparably long periods during which the software has to wait until the GSC is ready. This is the case when the GSC issues a return to “Normal Mode”, and the CPU resumes code execution from SRAM. It has to be considered that some peripherals may still not be available again. The wakeup sequence has not been finished until GSC has terminated. This might be monitored using register GSCSTAT.

9 Function Reference

Functions are used in the different **Operating Mode Transitions**.

The list of functions (ELFs) has been split into one group, **Clock**, for functions dealing with the clock system, and another part, **Power System**, in which functions related to the power supply are described. Further sequences, not directly related to clock or power issues, are compiled in an additional group of **Services**. User functions to implement application specific short actions after wakeup from a power-saving mode are listed in **User Functions**.

9.1 Important hints for implementing the sequences

Before starting to enter code right now, please go through the following directions for implementation.

- The instructions inside a single instruction boxmay be distributed or combined as desired, whereas instructions in different boxes must not be transposed, but rather conducted sequentially.
  This is also true for the **Operating Mode Transitions** shown in **Chapter 5**.
- A function flow may call an additional function, but **Reentrancy** must be taken into account.
- When using an exit terminator, it shall usually point to an error handling routine. To allow a proper identification where an error has occurred within the software flow, defining a return value could be helpful. Please see more on **Exception Handling** in **Chapter 7**.
- The input parameters and return values defined may only be seen as an option and can be handled, e.g., by using constants.
- Enough time for system configuration should be spent. When configuring the clock or supply system, the flow has to wait for **Analog Component**s anyway. Thus, it makes very little sense trying to optimize configuration steps, e.g. by combining two activities, unless one has thoroughly checked out that it really speeds up the overall transition time.

9.2 Terminology of Functions

The description of a function is done using flow charts.

In order to distinguish between statements related to the clock and power system the statements have different background colors: for clock system statements yellow is used, for power system statements a blue background is used.

9.3 Timeout Parameters

For all time-out parameters, the following rules apply:

- While waiting a time-out period, the current system frequency must be considered

- During this time, a certain bit/bit field must be polled to reach an expected value
- As soon as the expected value has been reached, the time-out loop can be left. If this value is not reached until the time expires, a time-out has occurred and the function must be left
- All time-out parameters used in the functions are compiled in **Table 9-38 Time Information for Analog Modules** and **Table 9-39 Digital time-out Parameters**, as well as in **Table 11-1 Timings for software time-out calculation**

9.4 Clock

This chapter details the Clock functions.

9.4.1 High Precision Oscillator

This section describes the functions related to the high precision oscillator OSC_HP.

9.4.1.1 EnableHighPrecOsc

This function enables/disables the high precision oscillator OSC_HP.

Table 9-1 Specification of EnableHighPrecOsc

Function Name	EnableHighPrecOsc
Input Parameters	Enable: 0 = Disable, 1 = Enable
Return Value	None
Description	Enables or disables the high precision oscillator for external crystal / clock at crystal input. May be called by the user to reduce waiting time.
Notes	The user is responsible for disabling the register protection. System clock must not depend on high precision oscillator during execution. Direct drive of the system via XTAL1 is not covered.

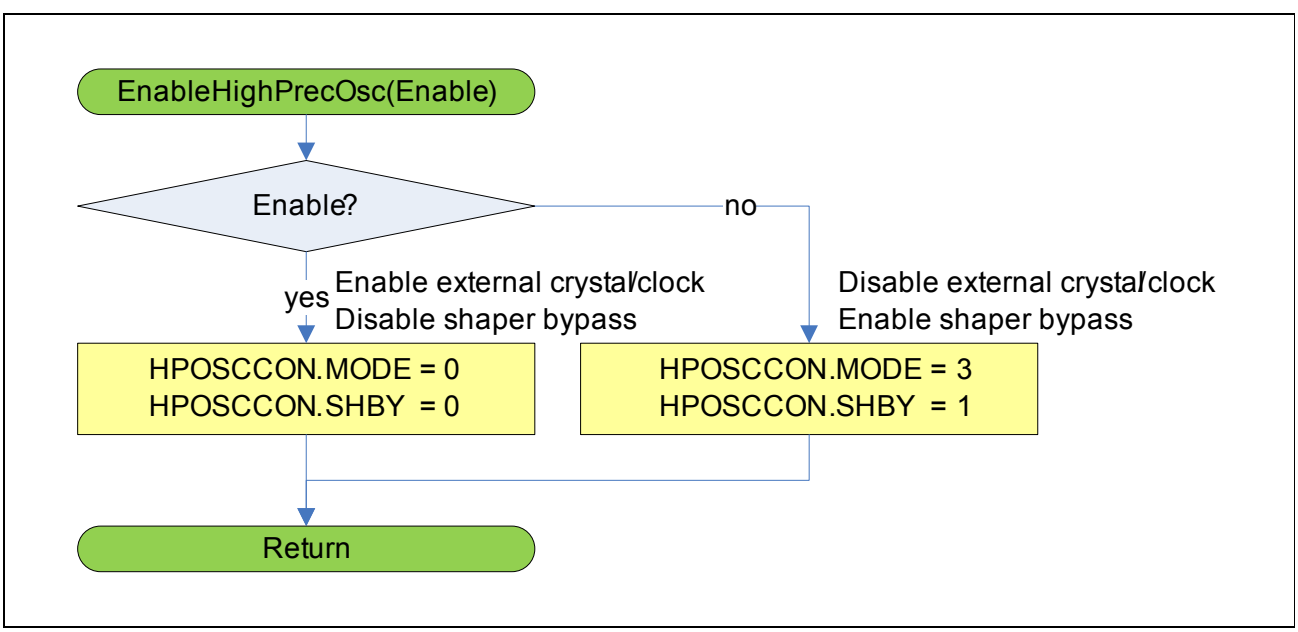

Figure 9-1 Flow-chart EnableHighPrecOsc

9.4.1.2 CheckFreqHighPrecOsc

This function checks the high precision oscillator (OSC_HP) for valid output frequency and is called after reset and after wakeup from Stopover Mode.

Table 9-2 Specification of CheckFreqHighPrecOsc

Function Name	CheckFreqHighPrecOsc
Input Parameters	None
Used Parameters	**ATTEMPTS_OSC_HP** **TIME_OSC_HP_1024** **TIME_OSC_HP_PLLV**
Return Values	Error code; 0 = NoError
Description	This function checks the high precision oscillator (OSC_HP) for a valid output clock.
Notes	The user is responsible for disabling the register protection. The loop for the attempts is implemented as a do-while loop. Timer 13 is used for the duration of OSC2L1 check instead of a software loop. For devices with PLLSTAT.OSCLOCK, the OSC2L1 check loop can be left on PLLSTAT.OSCLOCK = 1. Timeout based on Timer 13 is considered an error. This enhances security and/or reduces waiting time.

This function checks the high precision oscillator with the following steps:

- Wait until oscillator is usable (HPOSCCON.PLLV = 1) with timeout. In case of timeout, the function is left with an error.
- Wait until PLLSTAT.OSCLOCK = 1 (after 2^11 cycles) with timeout. In case of timeout or detection of an oscillator frequency error (HPOSCCON.OSC2L1 = 1), steps 1 and 2 are repeated. After a configurable number of attempts without success, the function is left with an error.

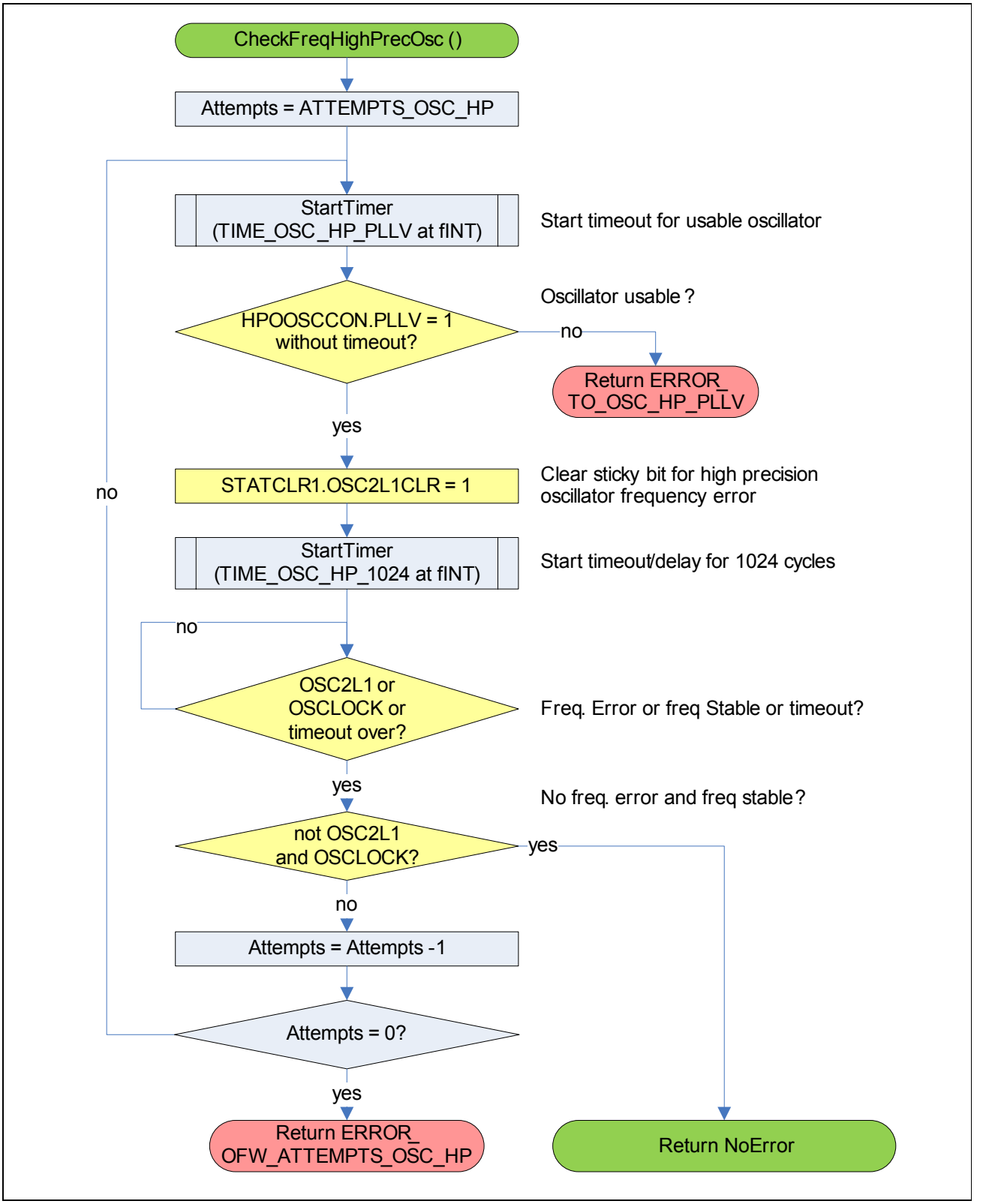

Figure 9-2 Flow-chart CheckFreqHighPrecOsc

9.4.2 PLL Digital Part

This section describes the functions related to the PLL Digital Part: PLL input clock and PLL dividers.

9.4.2.1 SelectExternalPllClock

This function selects the external clock source for the PLL. One of the following clock sources can be used:

- Output of the high precision oscillator
- Clock at CLKIN1

Table 9-3 Specification of SelectExternalPllClock

Function Name	SelectExternalPllClock
Input Parameters	None
Return Value	ErrorCode; 0 = NoError
Description	This function selects the external clock source for the PLL.
Notes	The user is responsible for disabling the register protection. The system clock may not depend on the PLL input during execution. The external clock must be available.

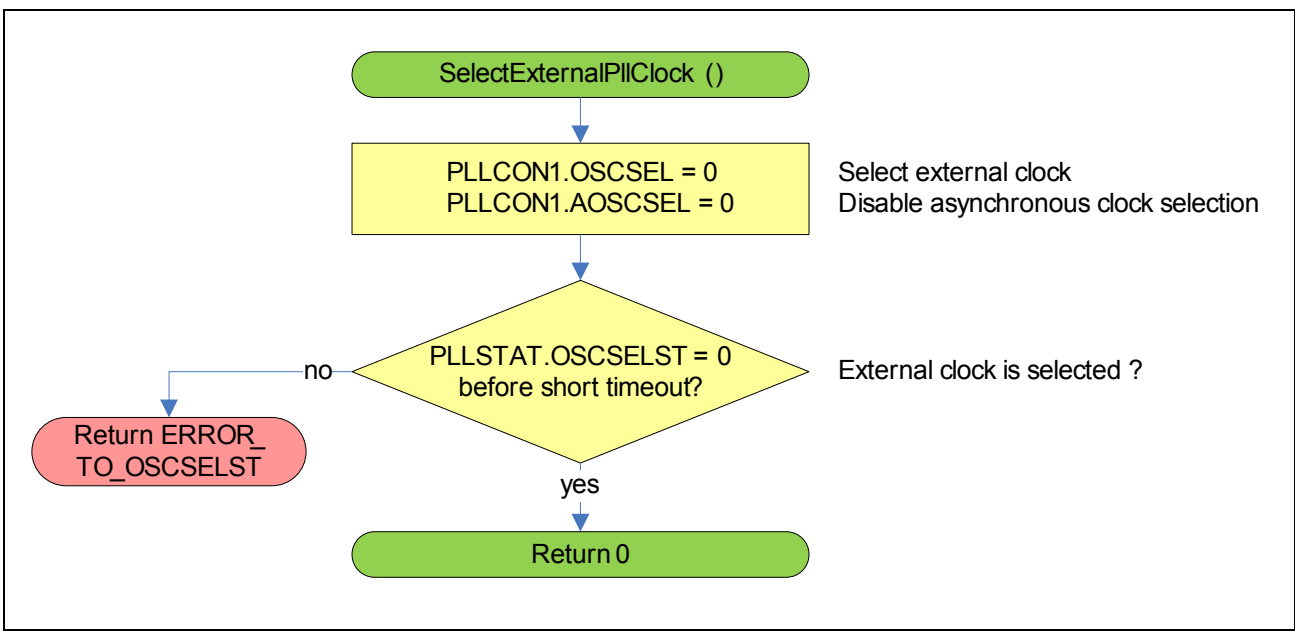

Figure 9-3 Flow-chart SelectExternalPllClock

9.4.2.2 ApplyNewK2Divider

This function is used to adjust the system frequency or to prepare a switch of the VCO bypass inside PLL.

Table 9-4 Specification of ApplyNewK2Divider

Function Name	ApplyNewK2Divider
Input Parameters	K2Div: K2 divider
Return Value	ErrorCode; 0 = NoError
Description	This function applies a new K2 divider value to the PLL. The effective divider is the parameter value + 1.
Notes	The user is responsible for disabling the register protection. If system clock depends on K2 during execution, the rules for smooth system clock frequency stepping must be applied.

A new divider value within PLL must be applied according to a handshake sequence, if the PLL digital part is on. In case the PLL digital part is off, any new divider value can be applied without handshake.

If the divider is already set to the specified value, the function will return immediately without error.

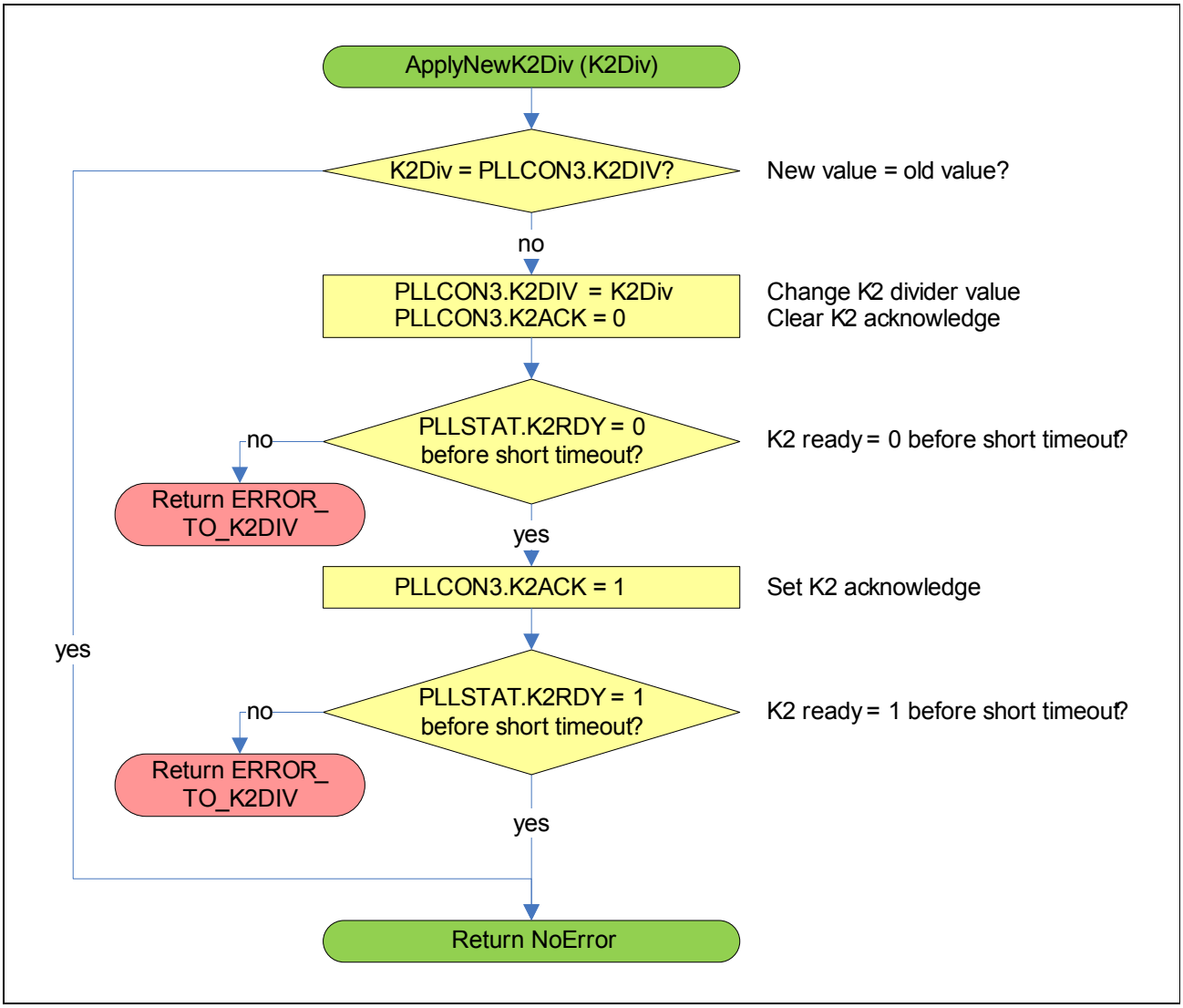

Figure 9-4 Flow-chart ApplyNewK2Div

9.4.2.3 ApplyNewK1Divider

This function is used to adjust the system frequency or to prepare a switch of the VCO bypass inside PLL.

Table 9-5 Specification of ApplyNewK1Divider

Function Name	ApplyNewK1Divider
Input Parameters	K1Div: K1 divider
Return Value	ErrorCode; 0 = NoError
Description	This function applies a new K1 divider value to the PLL. The effective divider is the parameter value + 1.
Notes	The user is responsible for disabling the register protection. If system clock depends on K1 during execution, the rules for smooth system clock frequency stepping must be applied.

If the divider is already set to the specified value, the function will return immediately without error.

The flow-chart is similiar to the flow-chart for **ApplyNewK2Divider**.

9.4.2.4 ApplyNewPDivider

This function is used to adjust the clock frequency for VCO.

Table 9-6 Specification of ApplyNewPDivider

Function Name	ApplyNewPDivider
Input Parameters	PDiv: P divider
Return Value	ErrorCode; 0 = NoError
Description	This function applies a new P divider value to the PLL. The effective divider is the parameter value + 1.
Notes	The user is responsible for disabling the register protection. System clock should not depend on VCO during execution.

If the divider is already set to the specified value, the function will return immediately without error.

The flow-chart is similiar to the flow-chart for **ApplyNewK2Divider**.

9.4.2.5 ApplyNewNDivider

This function is used to adjust the clock frequency for VCO.

Table 9-7 Specification of ApplyNewNDivider

Function Name	ApplyNewNDivider
Input Parameters	NDiv: N divider
Return Value	ErrorCode; 0 = NoError
Description	This function applies a new N divider value to the PLL. The effective divider is the parameter value + 1.
Notes	The user is responsible for disabling the register protection. System clock should not depend on VCO during execution.

If the divider is already set to the specified value, the function will return immediately without error.

The flow-chart is similiar to the flow-chart for **ApplyNewK2Divider**.

9.4.2.6 ApplyNewVcoDivs

This function is used to set all PLL VCO dividers: P, N, and K2.

Table 9-8 Specification of ApplyNewVcoDivs

Function Name	ApplyNewVcoDivs.
Input Parameters	PDiv: P divider value NDIV: N divider value K2Div: K2 divider value
Return Value	ErrorCode; 0 = NoError
Description	This private function applies new P, N and K2 dividers to the VCO.
Notes	The user is responsible for disabling the register protection. System clock should not depend on VCO during execution.

The function is a combination of ApplyNewPDiv, ApplyNewNDiv and ApplyNewK2Div.

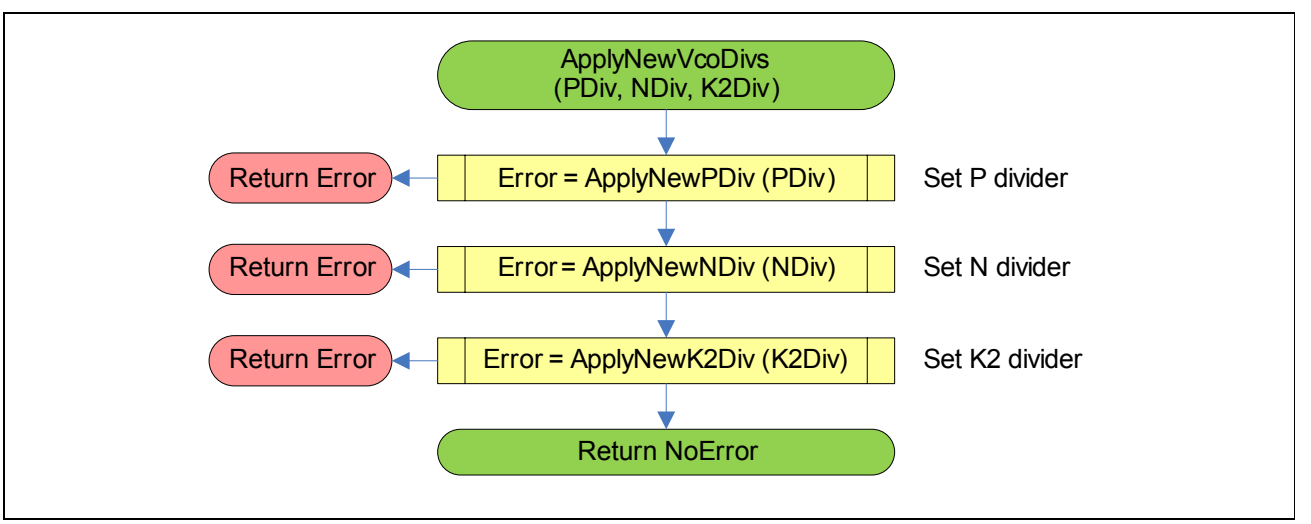

Figure 9-5 Flow-chart ApplyNewVcoDivs

9.4.2.7 RampPll

The function is used to change the system frequency smoothly when the frequency depends on the VCO output clock. For this purpose, the function applies a series of K2 steps and starts a delay after every step.

Table 9-9 Specification of RampPll

Function Name	RampPll
Input Parameters	K2 step/delay data
Return Value	ErrorCode; 0 = NoError
Description	This function applies a stepwise ramp-up or ramp-down of the output frequency using a set of K2 steps and delay values.
Notes	The user is responsible for disabling the register protection. For the K2 values, the rules for smooth system clock frequency stepping must be observed. The delay values must meet the requirements.

The K2 steps and the delay values are typically calculated offline by the compiler and are stored in a ROM array.

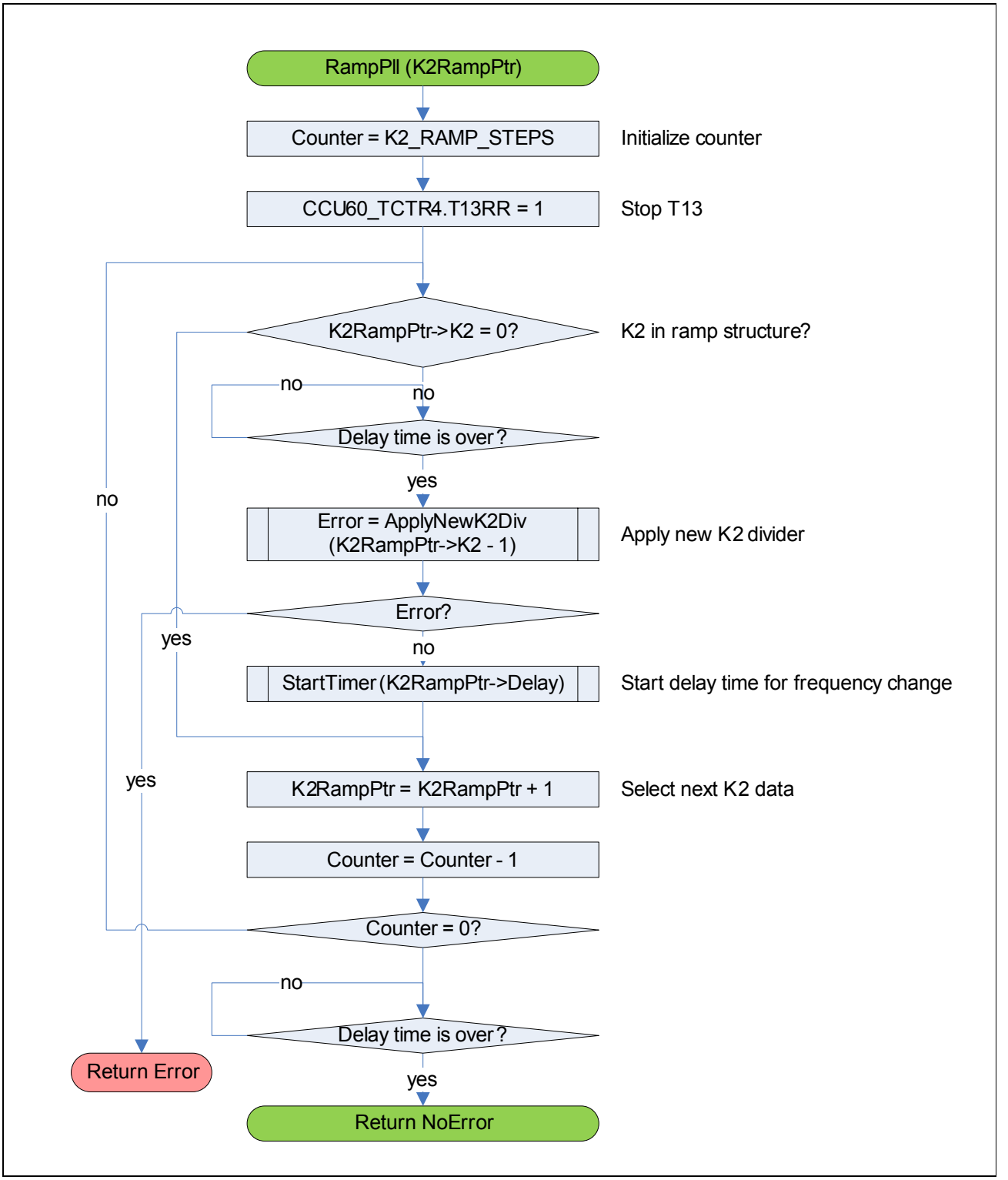

Figure 9-6 Flow-chart RampPll

9.4.3 PLL VCO

This section describes the functions related to the VCO.

9.4.3.1 WaitForVcoLock

The function restarts VCO lock detection and waits until lock is detected or a timeout occurs.

Table 9-10 Specification of WaitForVcoLock

Function Name	WaitForVcoLock
Input Parameters	Timeout: Timeout cycles for timer
Return Value	ErrorCode; 0 = NoError
Description	This function waits for VCO lock.
Notes	The user is responsible for disabling the register protection. Timeout cycles depend on the current system frequency.

The timeout cycles for the timer are usually calculated off-line by the compiler.

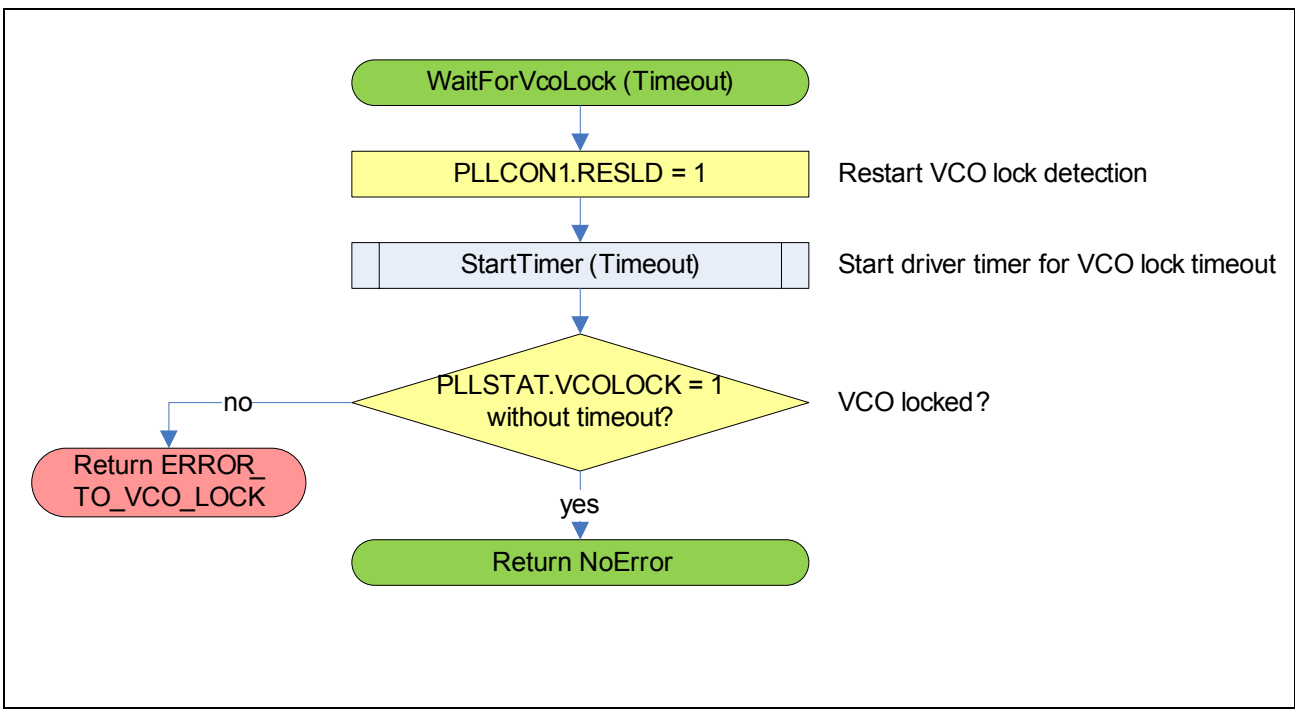

Figure 9-7 Flow-chart WaitForVcoLock

9.4.3.2 EnableVcoLockEmerg

This function enables emergency handling for the VCO loss of lock event. In an emergency case, bit SCU_PLLSTAT.FINDIS becomes active and disconnects VCO from its input. After a certain time, VCO will run on the base frequency of VCO band 1. A trap is generated.

Table 9-11 Specification of EnableVcoLockEmerg

Function Name	EnableVcoLockEmerg
Input Parameters	None
Return Value	ErrorCode; 0 = NoError
Description	This function enables the VCO loss-of-lock emergency handling.
Notes	The user is responsible for disabling the register protection. VCO must be locked.

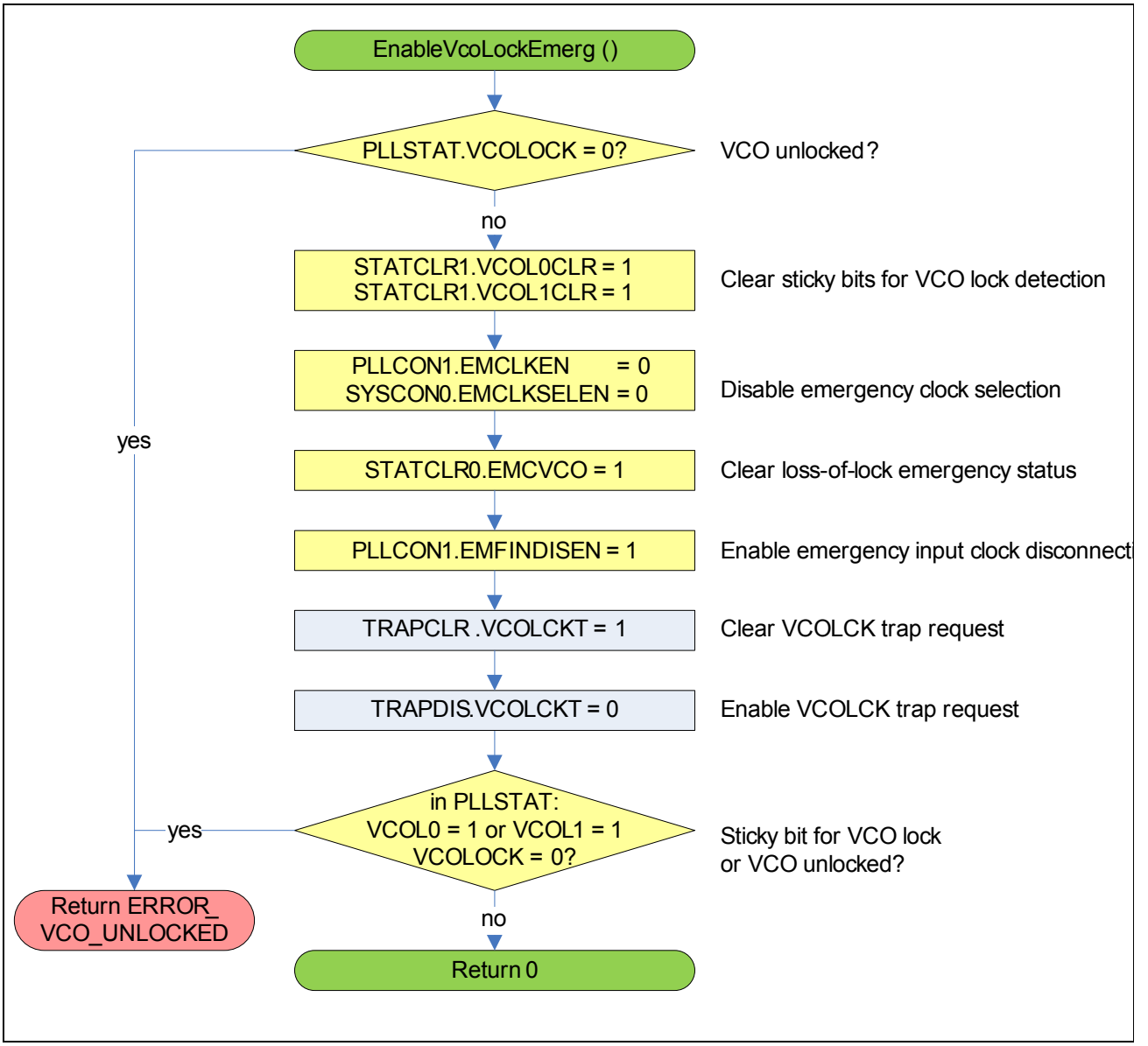

Figure 9-8 Flow-chart EnableVcoLockEmerg

9.4.3.3 DisableVcoLockEmerg

This function disables emergency handling for the VCO loss of lock event which was enabled by **EnableVcoLockEmerg**.

Table 9-12 Specification of DisableVcoLockEmerg

Function Name	DisableVcoLockEmerg
Input Parameters	None
Return Value	ErrorCode; 0 = NoError
Description	This function disables the VCO loss-of-lock emergency handling.
Notes	The user is responsible for disabling the register protection.

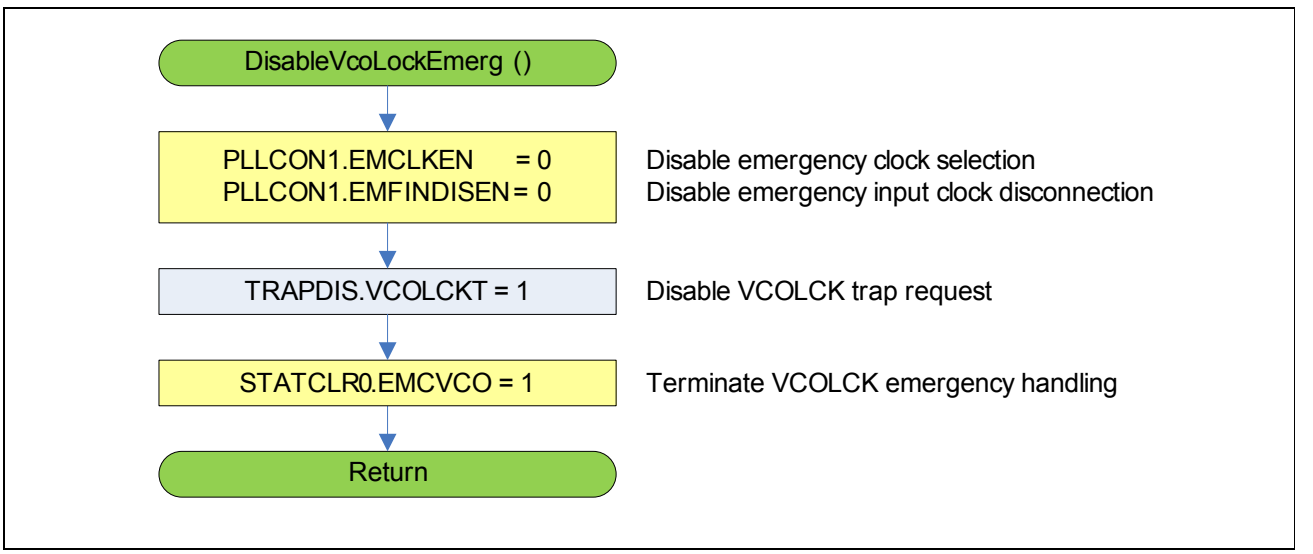

Figure 9-9 Flow-chart DisableVcoLockEmerg

9.4.3.4 EnableVcoBypass

This function enables or disables VCO bypass. If the VCO bypass is enabled, the PLL input clock, divided by K1, is selected as PLL output. If it is disabled, the frequency of the VCO in normal or free-running mode, divided by K2, is used as PLL output.

Table 9-13 Specification of EnableVcoBypass

Function Name	EnableVcoBypass
Input Parameters	Enable: 0 = Disable, 1 = Enable
Return Value	ErrorCode; 0 = NoError
Description	This function enables or disables VCO bypass.
Notes	The user is responsible for disabling the register protection. If PLL output is selected as system clock, the rules for smooth system clock frequency stepping must be applied.

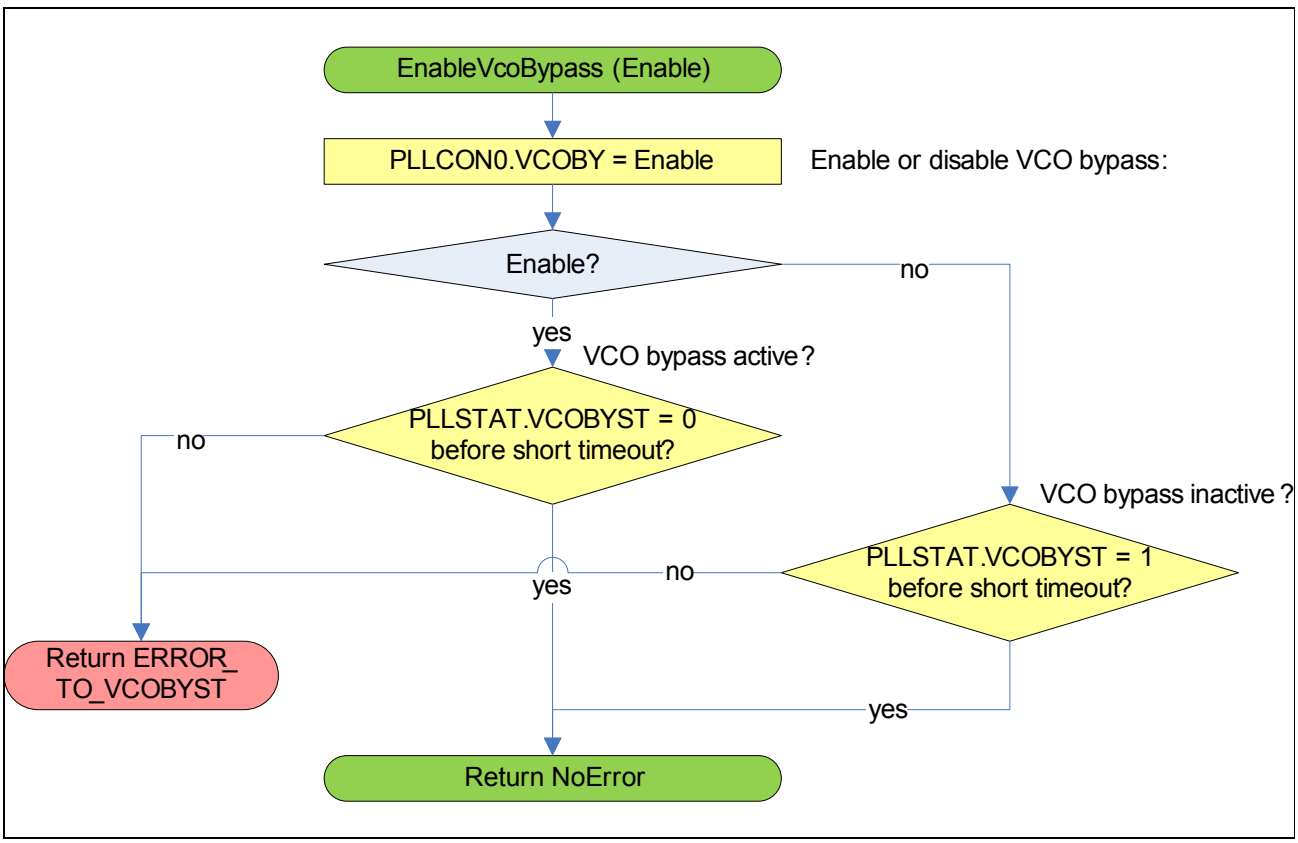

Figure 9-10 Flow-chart EnableVcoBypass

9.4.4 Clock System

This section describes the functions related to the clock system.

9.4.4.1 SelectSystemClock

The system clock is selected glitch-free via the Master Clock Multiplexer (MCM).

Table 9-14 Specification of SelectSystemClock

Function Name	SelectSystemClock
Input Parameters	ClkSel: System clock selection
Return Value	ErrorCode; 0 = NoError
Description	This function selects the required system clock.
Notes	The user is responsible for disabling the register protection. The selected clock must be available. The rules for smooth system clock frequency stepping must be observed.

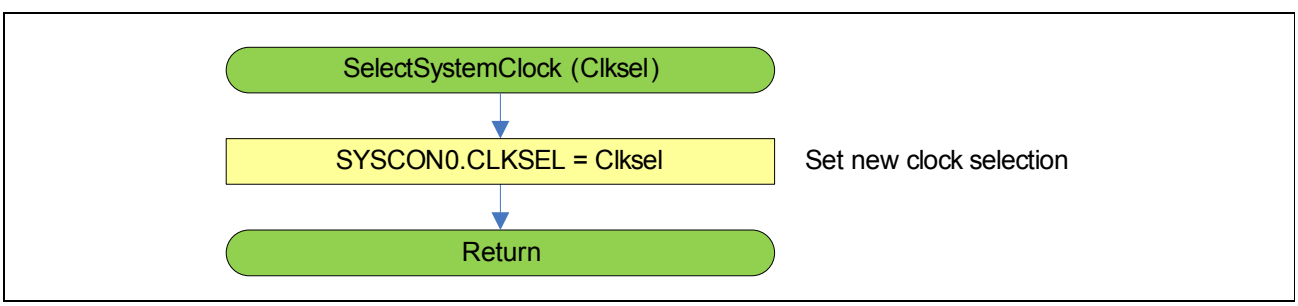

Figure 9-11 Flow-chart SelectSystemClock

9.4.5 Clock System after Wakeup from Stopover Mode

This section describes the functions related to the clock system after wakeup from Stopover Mode in the user function **HANDLE_STOPOVER_PS**.

9.4.5.1 UseFastClockInStopover_Ps

The function is called in the user function **HANDLE_STOPOVER_PS** if a higher and/or more stable clock is needed for short user actions.

Table 9-15 Specification of UseFastClockInStopover_Ps

Function Name	UseFastClockInStopover_Ps
Input Parameters	External: 0 : Internal clock source shall be used 1 : Configured external crystal/clock shall be used (crystal, external crystal clock or external clock at CLKIN1)
Return Value	None
Description	This function sets-up a a fast system clock after wakeup from Stopover Mode (**Stopover Wakedup Mode**).
Notes	The user is responsible for disabling the register protection. The requested external crystal/clock must already be available; in case of crystal, the oscillator must be running continously.

One of the following clocks can be selected:

- Configured external crystal/clock (crystal, external crystal clock or external clock at CLKIN1),
- Internal clock source f_{INT}

After the time-critical user actions function **UseWakeupClockInStopover_Ps** must be called to reduce the clock again.

Note: The following settings are assumed: PLL input clock = internal clock source, VCO is bypassed, K1 divider = 1.

Note: Function must be located in PSRAM. Code called by this function must be inlined or located in PSRAM. PSRAM code and constants must be copied to PSRAM via CopyWords.

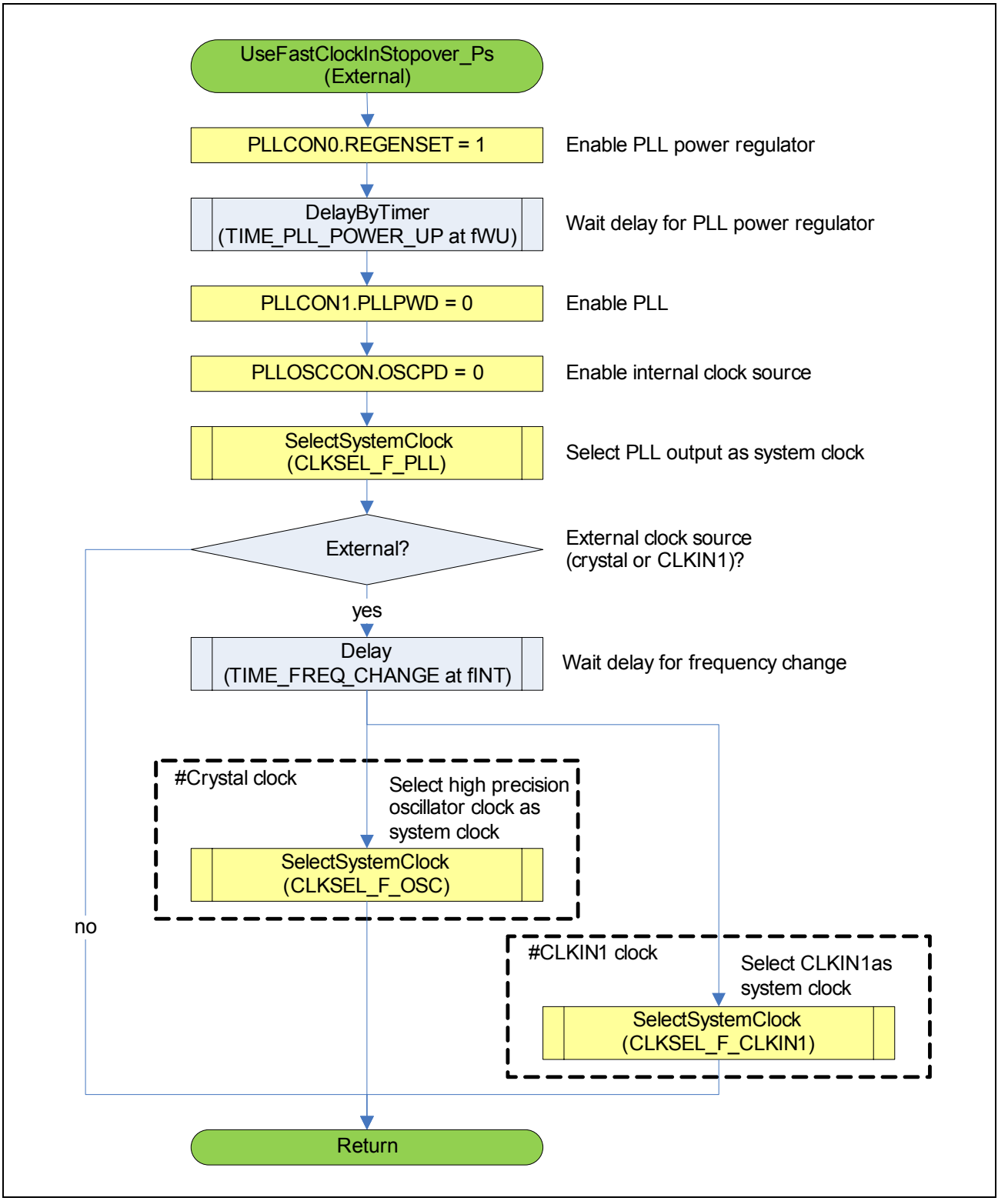

Figure 9-12 Flow-Chart UseFastClockInStopover_Ps

9.4.5.2 UseWakeupClockInStopover_Ps

The function selects the configurable wakeup clock as system clock. The function must be called in the user function **HANDLE_STOPOVER_PS** for handling the actions after a higher and/or more stable clock has been used.

Table 9-16 Specification of UseWakeClockInStopover_Ps

Function Name	UseWakeupClockInStopover_Ps
Input Parameters	None
Return Value	None
Description	This function selects wakeup clock after wakeup from Stopover Mode (**Stopover Wakedup Mode**) if a higher and/or more stable clock has been used.
Notes	The user is responsible for disabling the register protection.

*Note: The previous execution of **UseFastClockInStopover_Ps** and the following settings are assumed: PLL input clock = internal clock source, VCO is bypassed, K1 divider = 1.*

Note: It is assumed that the initial clock has been used for at least TIME_FREQ_CHANGE.

Note: Function must be located in PSRAM via compiler directives. Code called by this function must be inlined or located in PSRAM. PSRAM code and constants must be copied to PSRAM via function CopyWords.

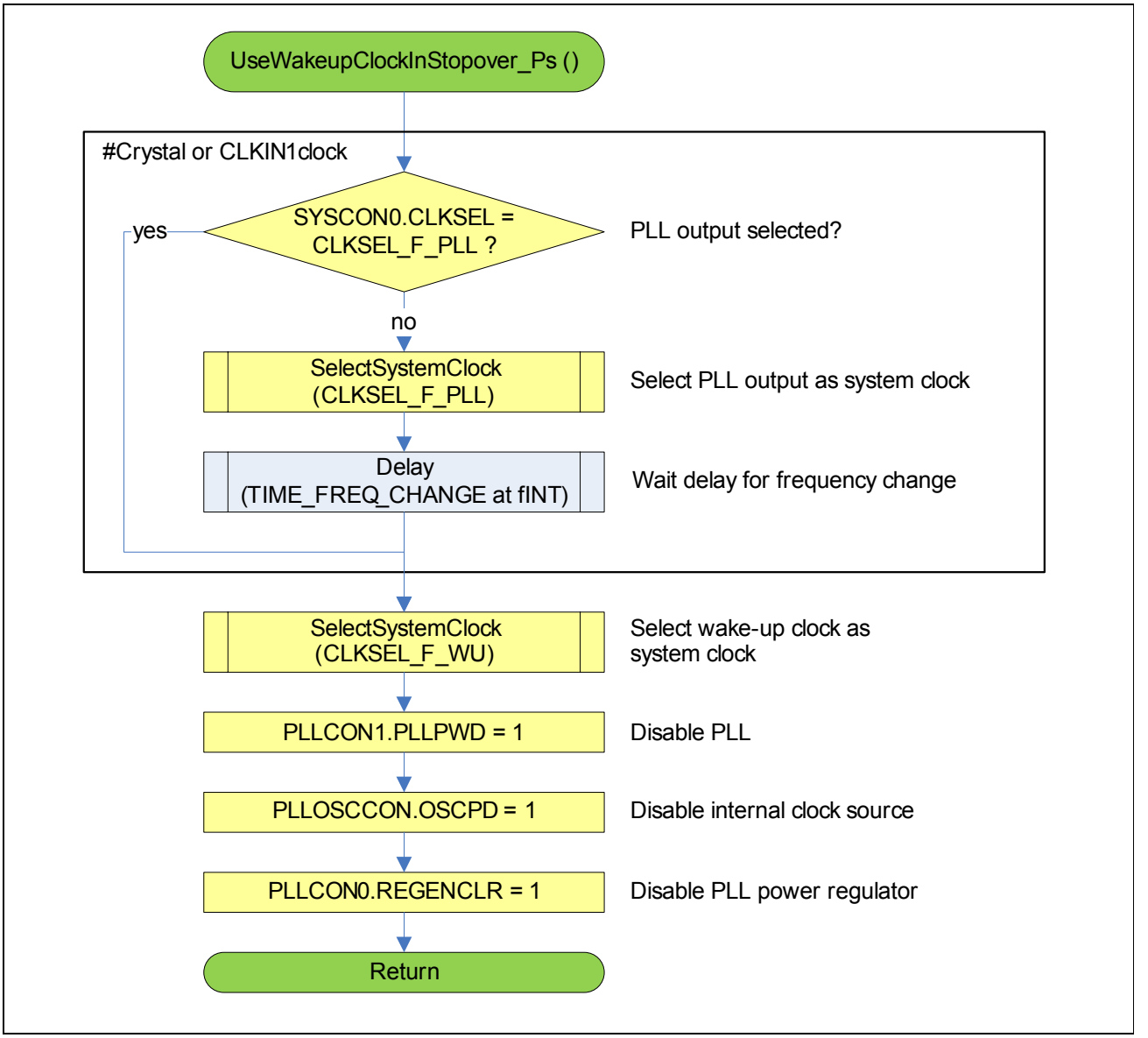

Figure 9-13 Flow-chart UseWakeupClockInStopover_Ps

9.4.6 Clock System after Wakeup from FSM Standby Mode

This section describes the function related to the clock system after wakeup from FSM Standby Mode (Fast Startup Mode) in the user function **HANDLE_STANDBY_FSM_PS_SB**.

9.4.6.1 UseWakeupClockInStandbyFsm_Ps

The function must be called in the user function **HANDLE_STANDBY_FSM_PS_SB** for handling **Fast Startup Mode**, after time-critical actions using internal clock as system clock have been finished. The configurable wakeup clock is then used as system clock.

Table 9-17 Specification of UseWakeupClockInStandbyFsm_Ps

Function Name	UseWakeupClockInStandbyFsm_Ps
Input Parameters	None
Return Value	None
Description	This function selects wakeup clock after wakeup from FSM Standby Mode (**Fast Startup Mode**). The PLL is shut down.
Notes	Wakeup clock must be running already. This function must be called by the user function **HANDLE_STANDBY_FSM_PS_SB**.

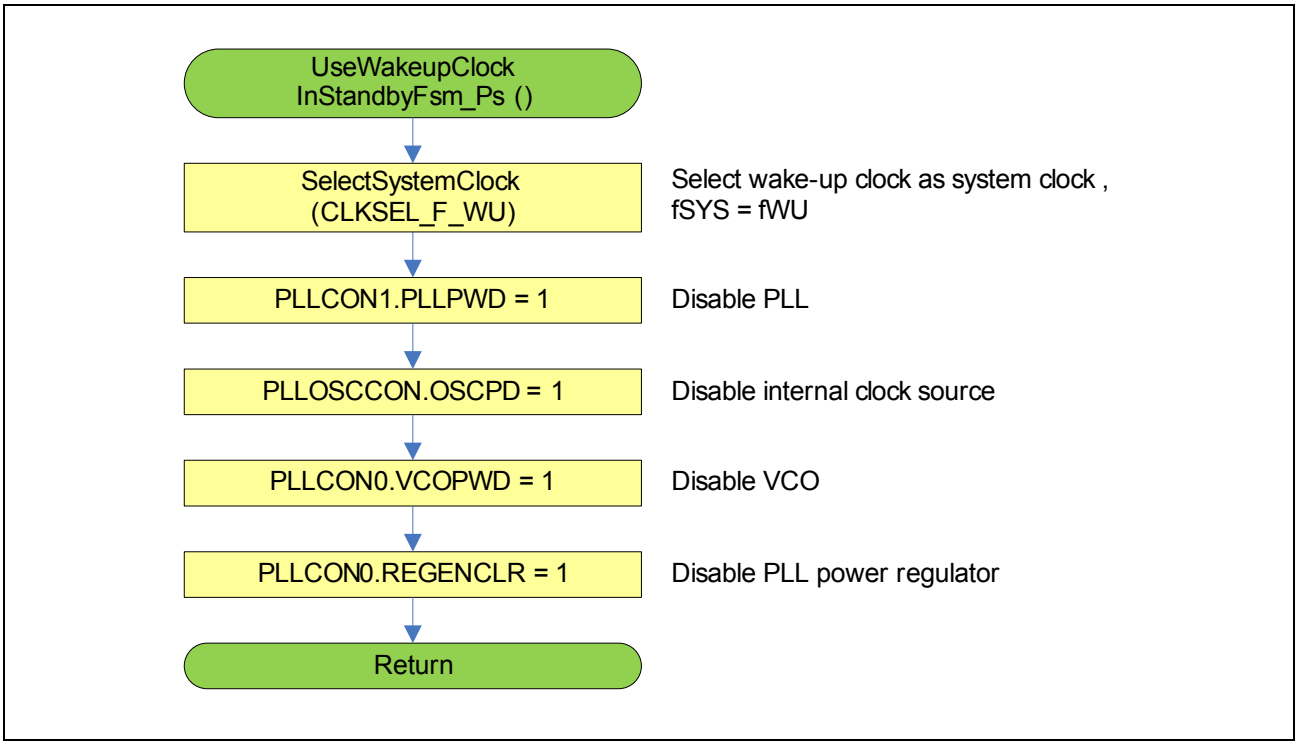

Figure 9-14 Flow-Chart UseWakeupClockInStandbyFsm_Ps

9.5 Power System

This chapter details the Power System functions.

9.5.1 Power Transition Sequences

Each supply component in the XC2000/XE166 architecture has one or more dedicated control SFRs. Writing to one of these registers by software directly influences the configuration of the related component. To be able to change the configuration of a supply component even while the CPU is unavailable, the content of some dedicated control SFRs can be modified additionally by a logic block named **PSC**. The PSC takes data from a set of shadow registers and copies it into the related control registers.

Conducting a Power Transition Sequence means that the PSC performs a data copy action in up to 6 consecutive steps to SFRs STEP0, PVC1CON0 and PVCMCON0. These SRFs control the main functions of the **EVVR**s for DMP_M and DMP_1.

The content of a set of shadow registers is named **Value Set**. Thus programming a specific Value Set into the shadow registers determines the type of a transition which will be conducted later by the PSC.

Two Sequences

Depending on the trigger which starts the data copy action, the PSC takes the data from one of two independent Value Sets A or B.

Sequence A transitions are typically used for transitions leading from a CPU operated mode into a state, where the CPU is unavailable as for example **Standby Mode**. Thus, Sequence A transitions are triggered by software. Shadow registers PVCMCONA[1-6], PVC1CONA[1-6] and SEQASTEP[1-6] are used.

Sequence B transitions are used to leave a power saving mode when the CPU is unavailable. Possible events to trigger this sequences are e.g. a pulse at an ESR pin or when the Wakeup Timer has been exceeded. Therefore, shadow registers PVCMCONB[1-6], PVC1CONB[1-6] and SEQBSTEP[1-6] are used.

Sequence interlock

Starting Power Transition Sequence **B** always requires a sequence **A** to be terminated properly before. This is flagged by bit PSCSTAT.LSTSEQ=0_B. In case a trigger for sequence **B** occurs already while sequence **A** is still performed or even before sequence **A** has been called, the request for sequence **B** will be held pending. This assures that the supply system is in a properly defined state when finally sequence **B** is started.
For Power Transition Sequences **A** there is no such interlock, since these transitions are always started by software.

Timing of a Power Transition Sequence

Typically the PSC needs two system clock cycles for each copy step to modify the content of registers STEP0, PVC1CON0 and PVCMCON0. This copy action is done in parallel.Power Transition Sequences have to be performed at a system clock frequency

of below 1 MHz. For the power transitions described in this document, wakeup clock is used resulting in a system clock frequency of either typically 140 kHz or typically 500 kHz. Thus the time between two steps can be considered to be about 4 us or longer.

After a copy action, proceeding with the next step may be additionally delayed due to the following reasons:

- When bit SYSDIV has been set to 1_B in register STEP0 by the current copy action, the next copy action will not be executed before 64 system clock cycles fSYS have been elapsed.
- When bitfield TRGSEL is unequal 0000_B, the next copy action will not be executed until the defined condition is true.

Figure 9-15 shows the decision flow. To continue with the next step both conditions must be true.

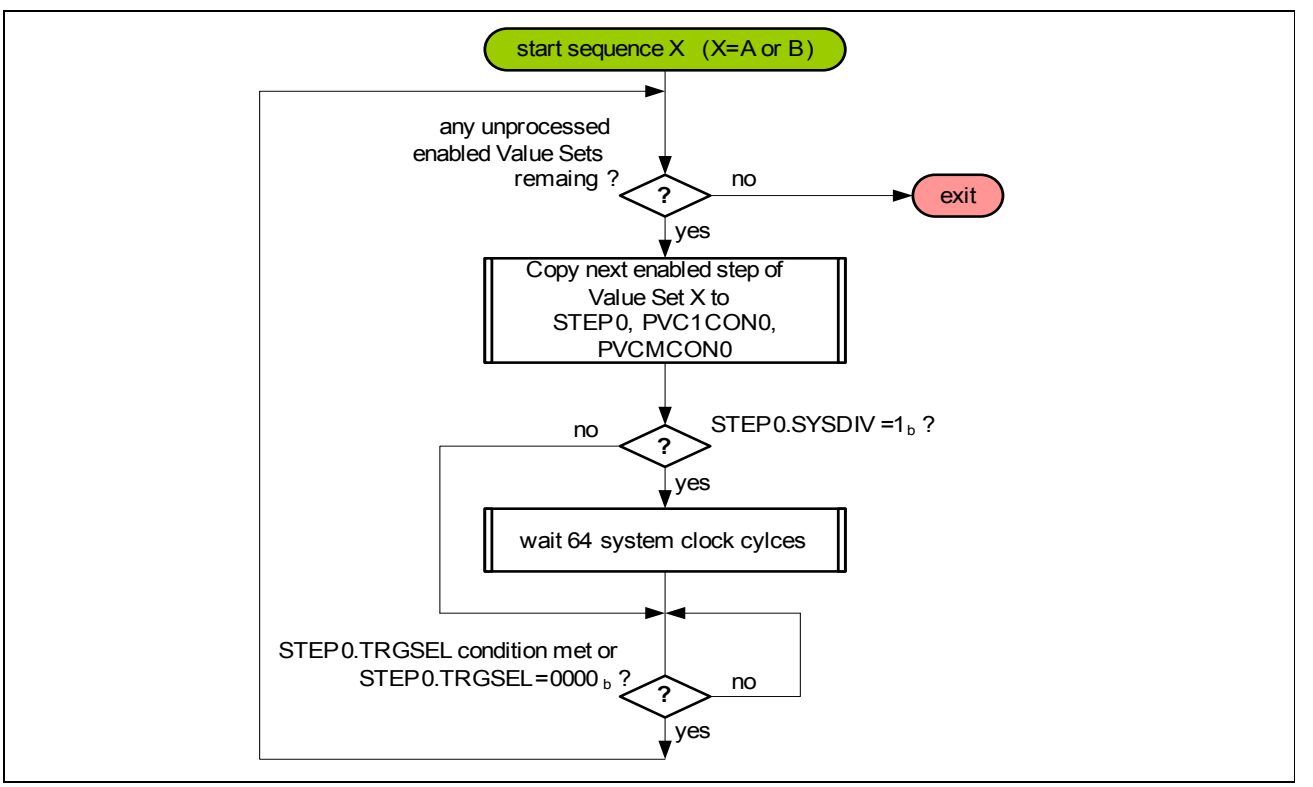

Figure 9-15 Sequence of PSC Hardware Copy Action

9.5.1.1 ConfigPscForLowPrecRef

This function configures sequence A for a power transition to switch from **BG_HP** based supply generation to **LPR** based supply generation. This is done by changing the supply reference to which the **EVVR**s are connected to.

Sequence A to Switch to Low Precision Reference

Use the value set in **Table 9-18** to configure sequence A for a power transition from 1.5 V HP Power Mode to 1.5 V LP Power Mode.

Table 9-18 Sequence A to Switch to Low Precision Reference

Step	PVCMCONAy	PVC1CONAy	SEQASTEPy
1	0504_H	0504_H	$80F8_H$
2	0504_H	0504_H	$80DB_H$
3	0504_H	0504_H	$80DB_H$
4	2544_H	2544_H	$80F3_H$
5 - 6	$XXXX_H$	$XXXX_H$	0000_H

Step 1

Disable any reset or interrupt triggered by the PVCs.

In parallel the clock in DMP_1 is stopped. This brings EVR_1 into a stable operating mode far-off its maximum operating conditions. However, the logic in DMP_1 is not used for the transition anyway. The clock in DMP_M can be kept running, because due to the slow system clock frequency and the small size of DMP_M, EVR_M is already in a stable Operating Mode.

Step 2

This step will perform the actual switch of the reference by reprogramming STEP0.V1/VM bit fields such that they select the new reference source (Full voltage with LPR).

The PSC is configured to wait for 1 system clock cycle before performing the next copy action.

Step 3

The PSC is configured to wait again for 1 system clock cycle before performing the next copy action.

This results in a total number of 4 wait cycles since 2 system cycles are required for each of the 3 copy steps. The transition is expected to be run at a maximum system frequency of typically 500 kHz, which means, that the waiting time is at minimum 16 µs.

Step 4

In the last operative step, the PVC triggers for nominal interrupt and reset levels are re-enabled together with the clock in DMP_1.

Step 5-6

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by the PSC.

Further Settings

The **GSC** is bypassed for this transition. This means that neither triggers for the GSC are issued upon a PSC run nor does the PSC wait for any acknowledge from GSC.

It is considered, that before changing the supply system any peripheral activity has been terminated.

Since GSC has no build in time-out logic the robust approach is, only to use the GSC when a time-out can be detected by software.

Debugging is generally enabled, but note that independent form the GSC mode the CPU will not operate as long as the system clock is stopped in DMP_1.

9.5.1.2 ConfigPscForHighPrecRef

This function configures sequence A for a power transition to switch from **LPR** based supply generation to high precison reference based supply generation. This is done by changing the supply reference the **EVVR**s are connected to.

Sequence A to Switch to High Precision Reference

Use the value set in **Table 9-19** to configure sequence A for a power transition from 1.5 V LP Power Mode to 1.5 V HP Power Mode.

Note: High Precision Bandgap ***BG_HP*** must be enabled before.

Table 9-19 Sequence A to Switch to High Precision Reference

Step	PVCMCONAy	PVC1CONAy	SEQASTEPy
1	0504_H	0504_H	$80FB_H$
2	0504_H	0504_H	$80C0_H$
3	0504_H	0504_H	$80C0_H$
4	2544_H	2544_H	$80F0_H$
5 - 6	$XXXX_H$	$XXXX_H$	0000_H

This transition has the same technical requirements than **ConfigPscForLowPrecRef**.

9.5.1.3 ConfigPscForStopoverMode

This function configures sequences A and B for power transitions related to **Stopover Mode**.

Sequence A to Enter Stopover Mode

Use the value set in **Table 9-20** to configure sequence A for a power transition to enter **Stopover Mode**.

This single step Value Set is used only to disable the clock in DMP_1 without changing any further settings of the clock or the supply system.

Table 9-20 Sequence A to Enter Stopover Mode

Step	PVCMCONAy	PVC1CONAy	SEQASTEPy
1 - 3	$XXXX_H$	$XXXX_H$	0000_H
4	2544_H	2544_H	$80FB_H$
5, 6	$XXXX_H$	$XXXX_H$	0000_H

Step 1-3

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by PSC. With this set-up the registers PVCxCONy must not re-programmed.

Step 4

By programming bit field STEP0.V1 to 111_B the system clock is stopped in DMP_1.

Note that this will not stop other clock sources located inside DMP_1, e.g. VCO or the clock of a connected debugger. This “clock stop” does only affect the system clock fSYS.

Step 5, 6

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by PSC.

Further Settings

Because CPU will not be available in the target mode, triggers for transition B are already enabled before the transition A is started.

Settings for GSCEN are the same as for further settings for **ConfigPscForLowPrecRef**.

Sequence B for Wakeup from Stopover Mode

Use this value set to configure sequence B for a power transition for wakeup from **Stopover Mode**.

This single step Value Set is used only to re-enable the clock in DMP_1 without changing any further settings of the clock or the supply system.

Table 9-21 Sequence B for Wakeup from Stopover Mode

Step	PVCMCONBy	PVC1CONBy	SEQBSTEPy
1	2544_H	2544_H	$80F3_H$
2 - 6	$XXXX_H$	$XXXX_H$	0000_HE

Step 1

By setting bit field STEP0.V1 to 110_B the system clock is re-enabled in DMP_1. This complies with the frequency stepping rules (see **Chapter 8.5.3 EVR**), because the system clock frequency is considered to be not above f_{WUmax} = 600 kHz at the time this step is performed.

Step 2-6

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by the PSC.

9.5.1.4 ConfigPscForStandbyMode

This function configures sequence A and B for a power transitions related to **Standby Mode** and **FSM Standby Mode**.

Sequence A to Enter (FSM) Standby Mode

Use the value set in **Table 9-22** to configure sequence A for a power transition to enter **Standby Mode**.

This value set defines the settings required to switch off the DMP_1 supply. This goes along with disabling the supply components PVCs in the step registers. The current control of domain DMP_M is disabled using function **EnableDmpmCurrentControl**.

Note that some components of the supply system such as **BG_HP** or **SWD** are not controlled by the PSC logic and thus must be de-activated before the PSC transition.

Table 9-22 Sequence A to Enter (FSM) Standby Mode

Step	PVCMCONAy	PVC1CONAy	SEQASTEPy
1	0504_H	0504_H	$80FB_H$
2	0504_H	0504_H	$E0E3_H$, if ULPEVR is used $E0E4_H$, otherwise
3 - 6	$XXXX_H$	$XXXX_H$	0000_H

Step 1

The reset and interrupt triggers for the PVCs are disabled. This prevents a reset being generated, when the PVCs are disabled and their comparator signals become invalid.

In preparation of disabling power domain DMP_1 the clock in DMP_1 is stopped. This is primarily not required for the ramp-down, because any logic in DMP_1 will get a reset anyway. But a disabled clock will be needed later when DMP_1 is being ramped-up again.

Step 2

The EVR_1 is switched off by setting bit field STEP0.V1 to 100_B.

The EVR_M is switched off by setting bit field STEP0.VM to 100_B if the Ultra Low Power EVR is used.

The PVCs are switched off, too.

Note that VDDI_1 will not actively be shorted to VSS when EVR_1 it is switched off. The resulting voltage slope of VDDI_1 mainly depends on the external VDDI capacitors and the actual leakage current.

Step 3-6

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by **PSC**.

Further Settings

For the transition to FSM Standby Mode the GSC is bypassed. This means that neither triggers for the GSC are issued upon a PSC run nor does the PSC wait for any acknowledge from GSC. It is considered, that before changing the supply system any peripheral activity has been terminated.

For transition from Standby Mode to Base Mode the GSC is triggered at the exit of sequence B.

Since GSC has no build in time-out logic the robust approach is, only to use the GSC when a time-out can be detected by software.

When DMP_1 is switched off, no debug access will be possible.

Triggers from possible wakeup sources are enabled in register SEQCON.

Sequence B for Wakeup from (FSM) Standby Mode

Use this value set to configure sequence B for a power transition after wakeup from **Standby Mode**.

The settings of this Value Set are used to perform a ramp-up of the DMP_1.

Table 9-23 Sequence B for Wakeup from (FSM) Standby Mode

Step	PVCMCONBy	PVC1CONBy	SEQBSTEPy
1	0504_H	0504_H	$C0E3_H$
2	0504_H	0504_H	$88DB_H$
3	0504_H	0504_H	$80EB_H$
4	2544_H	2544_H	$80F3_H$
5 - 6	$XXXX_H$	$XXXX_H$	0000_H

Step 1

If the Ultra Low Power EVR is used then it remains enabled and in addition the EVR_M is enabled again by setting bit field STEP0.VM to 011_B.

The PVC_M for DMP_M is (re-)activated. The monitoring level for PVC_M remains unchanged, but because the comparator signal of PVC_M is not valid at that point, possible triggers remain disabled.

Step 2

The EVR_1 and PVC_1 are enabled together.

The condition for the PSC to continue with the next step is VDDI_1 to reach the monitoring level of the second PVC_1 comparator PVC1_2. The default value of the PVC comparator outputs after being enabled is 0_B which literally means “level is not ok”. This is the reason why the condition to continue with the next step (bit field SEQBSTEP.TRGSEL) can be defined the same time the PVC is enabled.

Step 3

Power domain DMP_1 is powered with EVR_1.

Step 4

The reset and interrupt trigger of PVC_M are enabled together with the clock in DMP_1.

This complies with the frequency stepping rules, (see **Chapter 8.5.3 EVR**) because the system clock frequency is considered to be not above 500 kHz at the time this step is performed.

Step 5-6

These steps are disabled, i.e. the content of the respective configuration registers will not be copied by PSC.

9.5.2 Monitoring PSC State

To monitor the progress of the **PSC** while a power transition is active, the following sequence has been defined (x stands for A or B, depending on the used sequence):

1. Start sequence x by writing SEQCON.SEQxTRG = 1_B
2. Wait until SEQCON.SEQxEN = o_B with timeout
3. Wait until PSCSTAT.PSMSTAT = idle with timeout

9.5.3 GSC in Conjunction with Power Transitions

To improve overall robustness, GSC must never be used to disable peripherals in conjunction with a power transition, since there is always a risk of a dead lock due to a not responding peripheral. Therefore, if GSC is to be used before the power transition in order to shut down peripherals, it must be triggered by software to run. A time-out must supervise the GSC run to prevent a dead lock.

The exit of sequence B triggers the GSC after wakeup.

9.5.4 Power System

This section details the actual Power System functions.

9.5.4.1 EnableDmpmCurrentControl

This function enables or disables the current control in DMP_M. In order to reduce power consumption the current control in DMP_M for 1.5 V LP and 1.5 V HP setting can be disabled.

Table 9-24 Specification of EnableDmpmCurrentControl

Function Name	EnableDmpmCurrentControl
Input Parameters	Enable: 0 = Disable 1 = Enable
Return Value	None
Description	This function enables or disables the 1.5V DMP_M current control.
Notes	The user is responsible for disabling the register protection.

If current control is enabled, current control is set to maximum and then the control is switched on. If it is disabled, current control level is set to minimum and then the control is switched off.

At the moment, this function is only used to disable current control in Standby Mode.

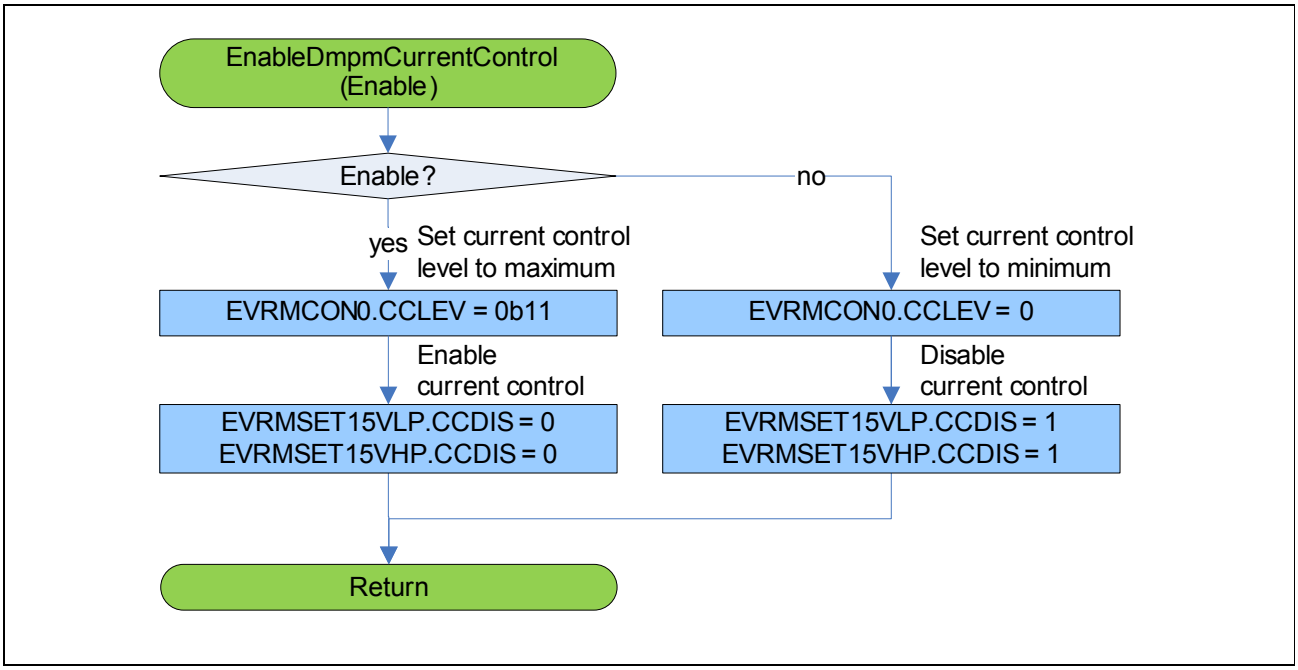

Figure 9-16 Flow-chart EnableDmpmCurrentControl

9.6 Services

This chapter details the Service functions.

9.6.1 System Modes

9.6.1.1 RequestSystemMode

This function can be called by the application e.g. if peripherals shall be activated during wakeup from FSM Standby Mode or Stopover Mode.

Table 9-25 Specification of RequestSystemMode

Function Name	RequestSystemMode
Input Parameters	SystemMode: SYSTEM_MODE_NORMAL: enable the peripherals SYSTEM_MODE_CLOCK_OFF: to disable the peripherals
Return Value	ErrorCode; 0 = NoError
Description	This function requests a new system mode.
Notes	The user is responsible for disabling the register protection.

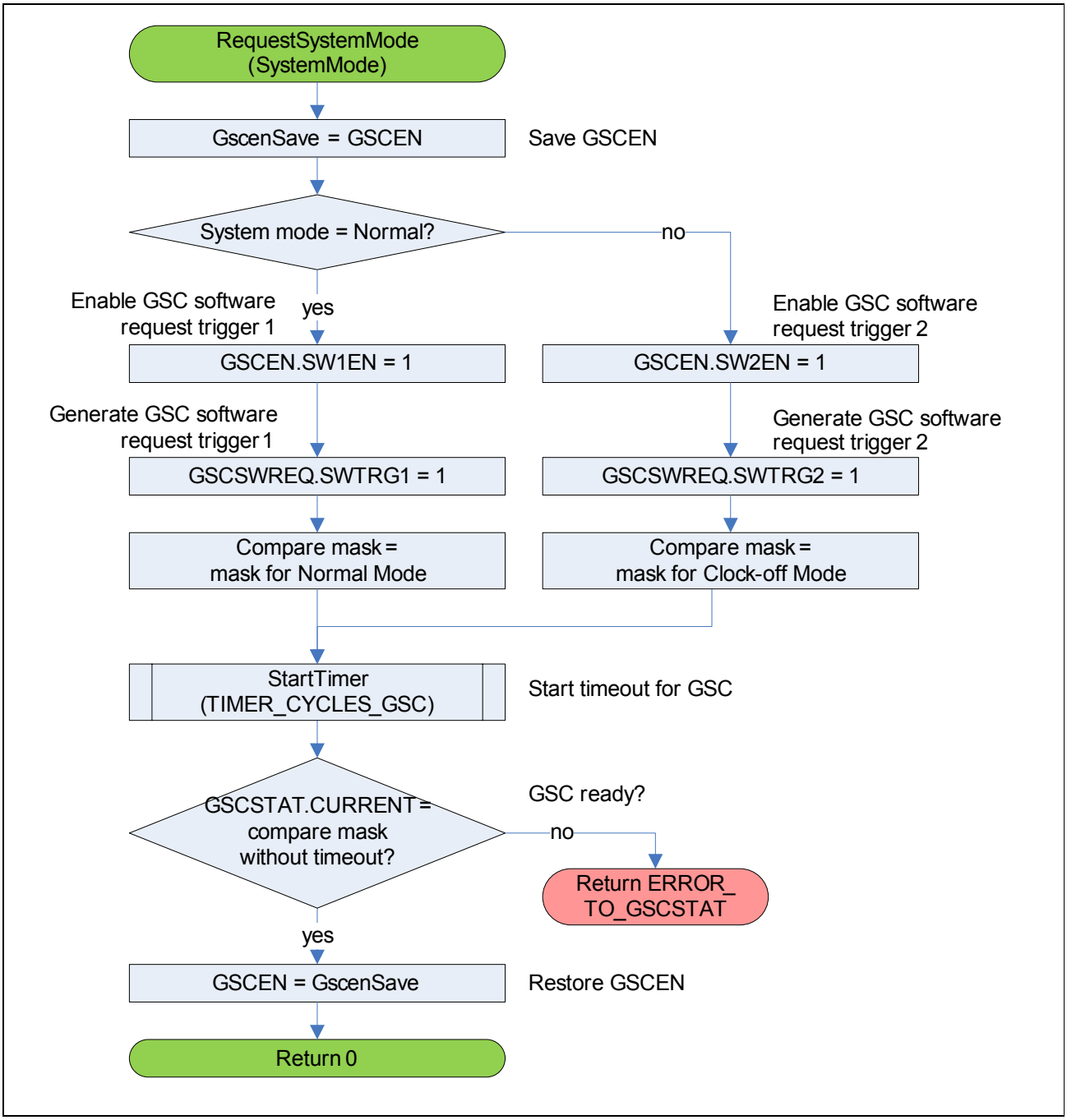

Figure 9-17 Flow-chart RequestSystemMode

9.6.2 Timer

Delays or timeouts are required in many cases. The following methods are used to meet these requirements:

Short Delays and Timeouts

For short delays, a series of NOPs is used.

For short timeouts, a polling algorithm which checks a condition several times is used. As soon as the condition is met, the program continues. If the condition is not met after the final check, the program exits with an error.

The algorithm has a well defined minimum number of instruction cycles before the final check is performed.

Longer Delays and Timeouts

Software loops might be used, but with some disadvantages:

- Dependency on compiler/optimization, or assembler programming necessary
- Dependency on used program memory and its alignment
- No useful actions possible during delay or timeout

Therefore, in the following the CCU6 timer T13 is used.

A timeout or delay given in time units must be converted to cycles, taking into account the actual system frequency (including tolerances).

This chapter details the Timer functions.

Function **InitTimer** configures CCU6 timer T13 as timer before functions with delays or timeouts are needed; previous CCU6 SFR contents may be saved. Function **RestoreTimer** can restore those registers after usage of the function.

9.6.2.1 InitTimer

This function is called by the application to initialize a timer which is required for delays and timeouts in the transitions.

Table 9-26 Specification of InitTimer

Function Name	InitTimer
Input Parameters	None
Return Value	None
Description	This function configures CCU6 timer T13 as driver timer; previous CCU6 register contents may be saved.
Notes	The user is responsible for disabling the register protection. This function must be called before other functions with timer usage. The function will enable CCU6 and will overwrite previous settings for T13.

The function will configure CCU6 timer T13 as one-shot driver timer. It counts from a variable start value upwards until the fixed period value 0xFFFF is reached. The resolution is 8 CCU6 clocks = 8 system clocks.

If T13 is not used by the application, it is sufficient to call this function once during initialization (before GoFromBaseModeToNormalMode is used). Otherwise the function must be called again whenever a operation mode transition is needed, e.g. for entering a power-saving mode.

Configuration

If configured, the function will save the previous CCU6 register contents; they may be restored via RestoreTimer.

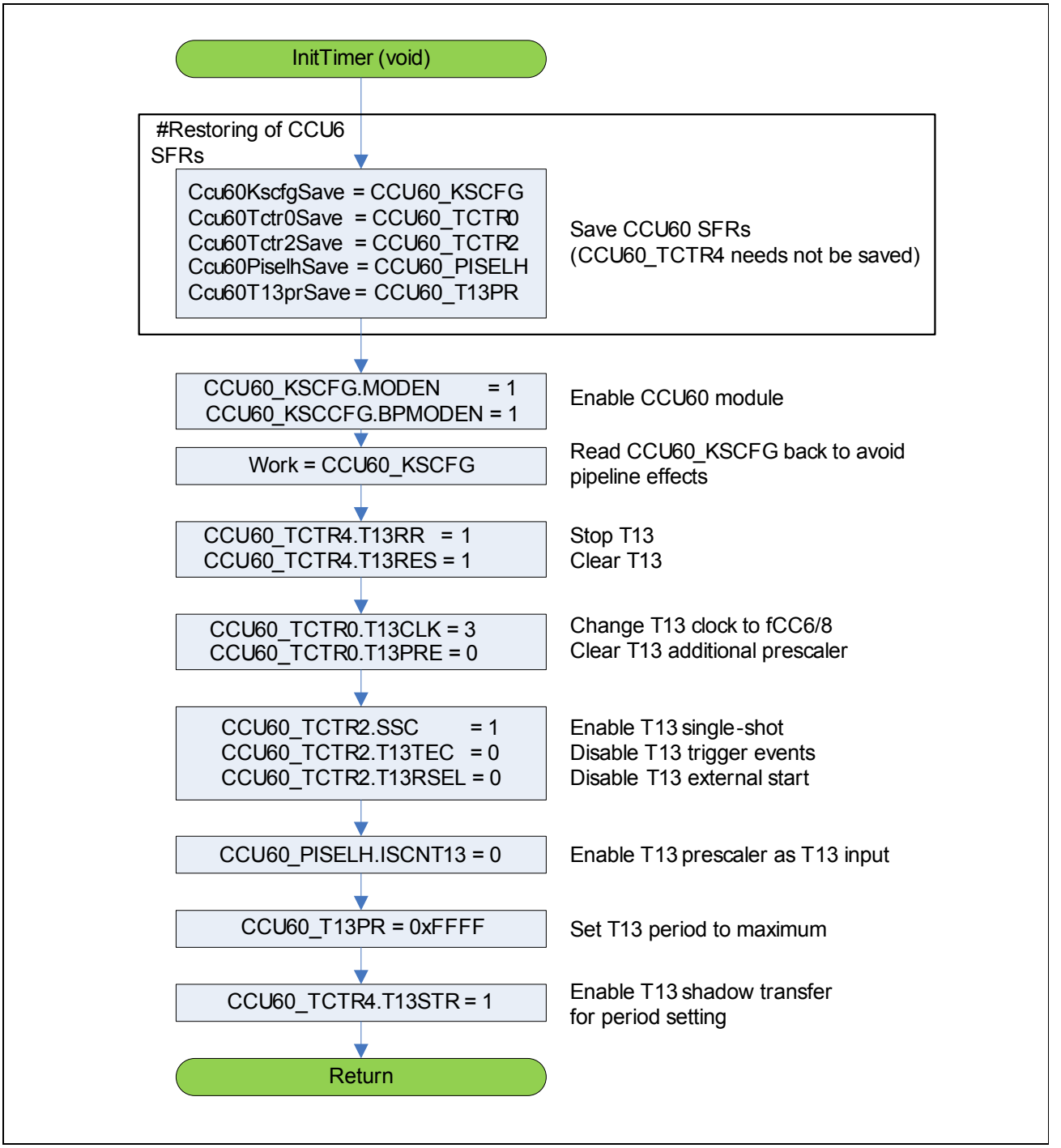

Figure 9-18 Flow-chart InitTimer

9.6.2.2 StartTimer

This function starts a timer for counting number of clocks.

Table 9-27 Specification of StartTimer

Function Name	StartTimer
Input Parameters	Cycles: Number of T13 clocks to count
Return Value	None
Description	This function starts timer T13 for counting a certain number of clocks.
Notes	It is assumed that the timer is already initialized by InitTimer.

The parameter Cycles is usually calculated off-line by the compiler.

Before setting and starting the timer, the function performs a correction of parameter Cycles because of HW and SW reasons.

The timer counts from a variable start value upwards until the fixed period value 0xFFFF is reached. The resolution is 8 CCU6 clocks = 8 system clocks.

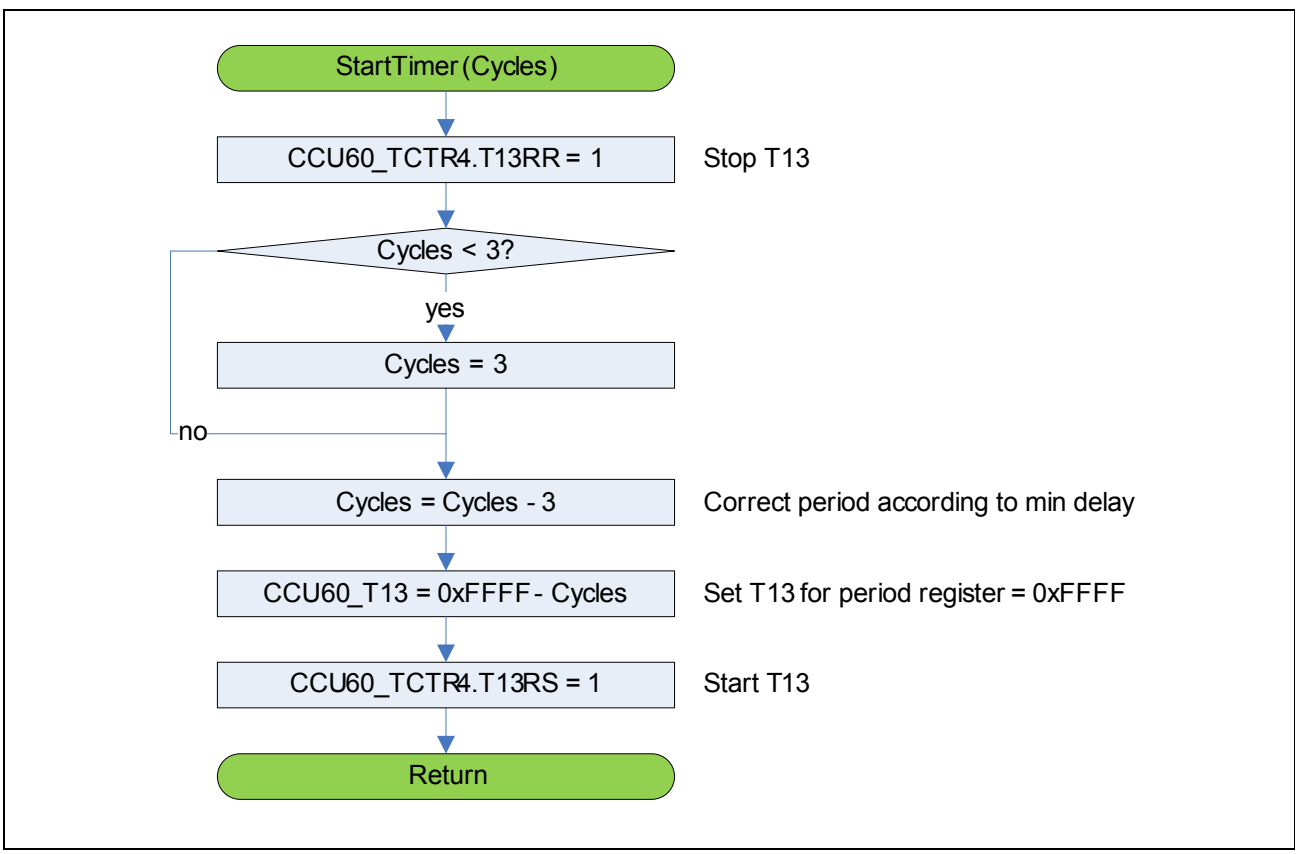

Figure 9-19 Flow-chart StartTimer

9.6.2.3 DelayByTimer

This function starts a timer and waits until the specified number of clocks is counted.

Table 9-28 Specification of DelayByTimer

Function Name	DelayByTimer
Input Parameters	Cycles: Number of clocks to count
Return Value	None
Description	This function starts timer and waits until the specified number of clocks is counted.
Notes	The user is responsible for disabling the register protection. It is assumed that the timer is already initialized by **InitTimer**.

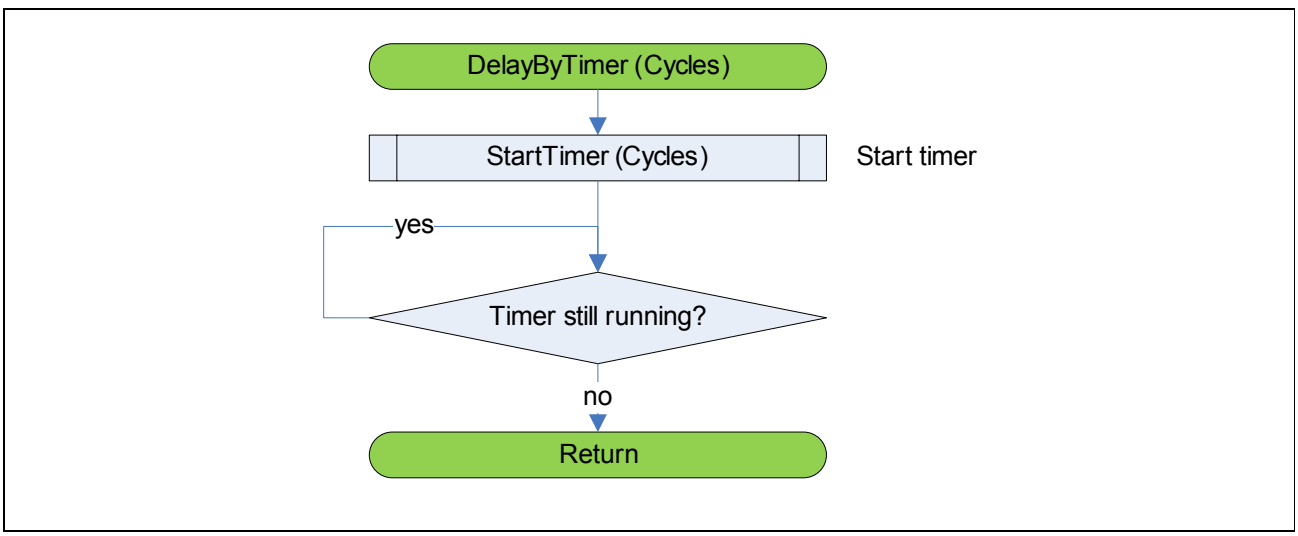

Figure 9-20 Flow-chart DelayByTimer

9.6.2.4 RestoreTimer

This function restores the saved timer register contents

Table 9-29 Specification of RestoreTimer

Function Name	RestoreTimer
Input Parameters	None
Return Value	None
Description	This function restores CCU6 register contents saved during the previous execution of InitTimer.
Notes	The user is responsible for disabling the register protection. The function will stop and clear T13.

The function may be called by the application to resume as far as possible the original CCU6/T13 function after usage by SCU Driver. The following CCU6 SFRs saved during the previous execution of InitTimer will be restored:

- CCU60_TCTR0
- CCU60_TCTR2
- CCU60_PISELH
- CCU60_TCT13PR
- CCU60_KSCFG

The original content of T13 cannot be restored.

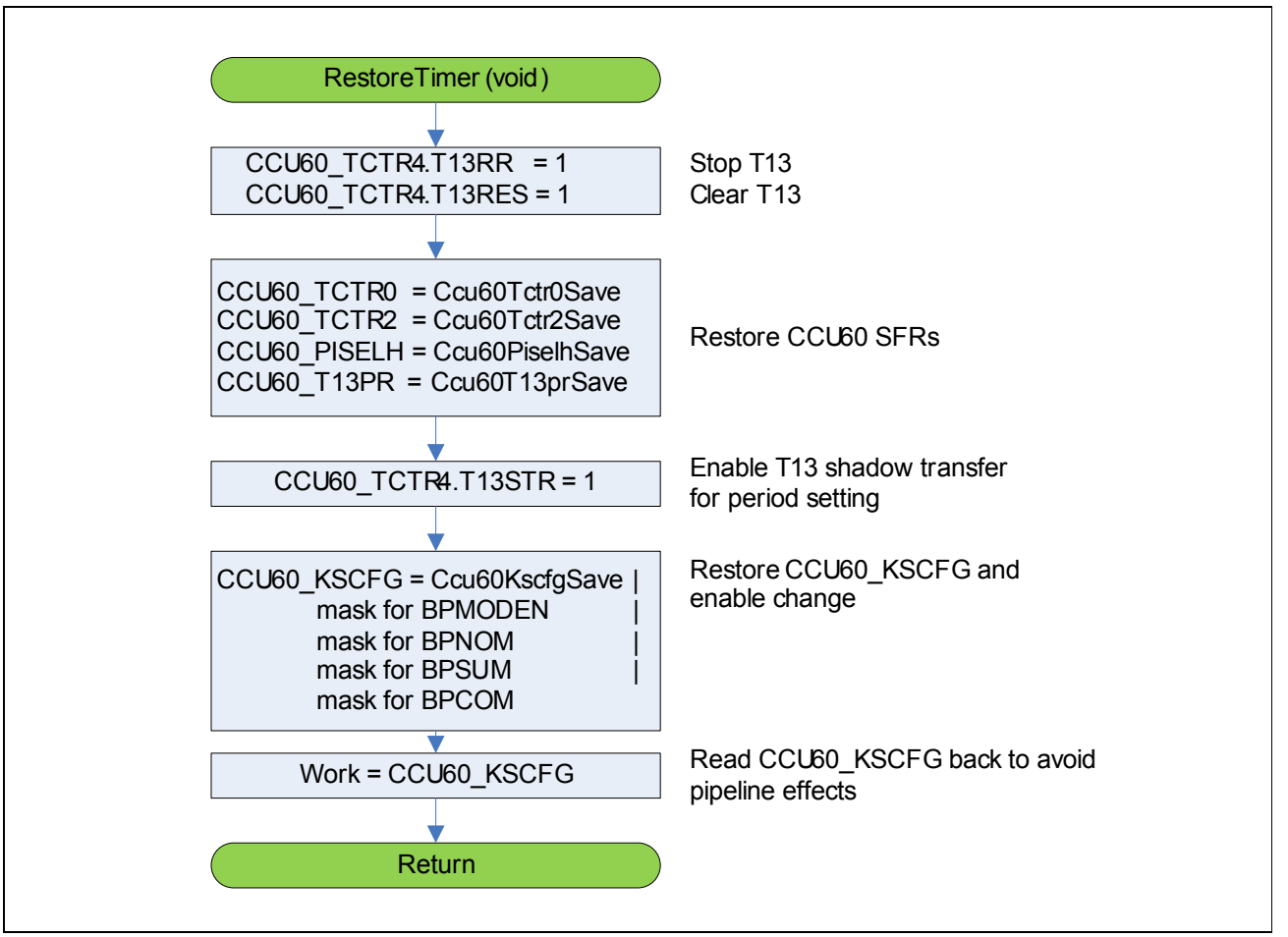

Figure 9-21 Flow-chart RestoreTimer

9.6.3 Memory Copy

This chapter details the memory copy functions to copy code and data from Flash to PSRAM and SBRAM.

9.6.3.1 CopyWords

This function copies words from source to destination.

Table 9-30 Specification of CopyWords

Function Name	CopyWords
Input Parameters	DestPtr:Destination pointer
I	SrcPtr:Source pointer
	Words:Number of words to be copied (> 0)
Return Value	None
Description	This global function copies words from source to destination.
Notes	There is no check for overlapping source and destination.

The function must be called by the application before a power-saving mode can be entered. For power-saving modes, parts of the function code and application code are executed from PSRAM. These code sections must be located in special areas via compiler directives, and they have to be copied from flash to PSRAM.

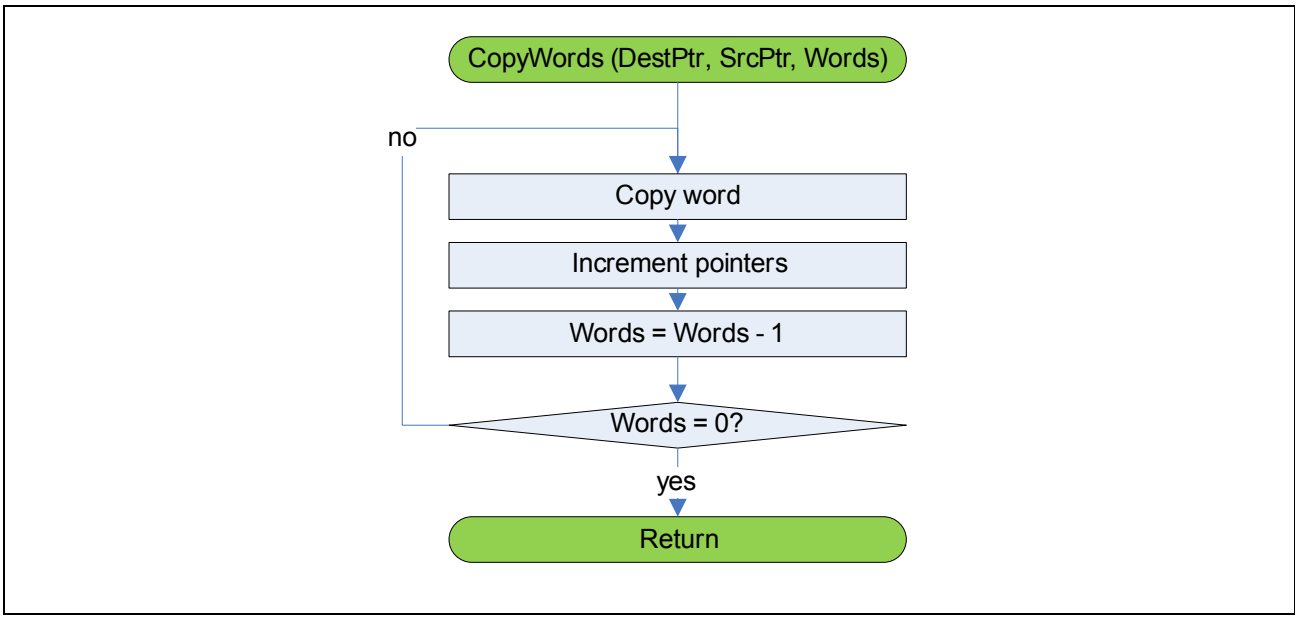

Figure 9-22 Flow-chart CopyWords

9.6.3.2 CopyVectorToPsram

This function copies an interrupt or trap vector from flash to PSRAM.

Table 9-31 Specification of CopyVectorToPsram

Function Name	CopyVectorToPsram
Input Parameters	Vector: Interrupt vector/trap number, 0...127
Return Value	None
Description	This function copies an interrupt or trap vector from flash to PSRAM.
Notes	The user is responsible for disabling the register protection.

The function is called by the application if interrupts or traps may occur during code execution from PSRAM. In this case, the corresponding vectors must be copied from flash to PSRAM.

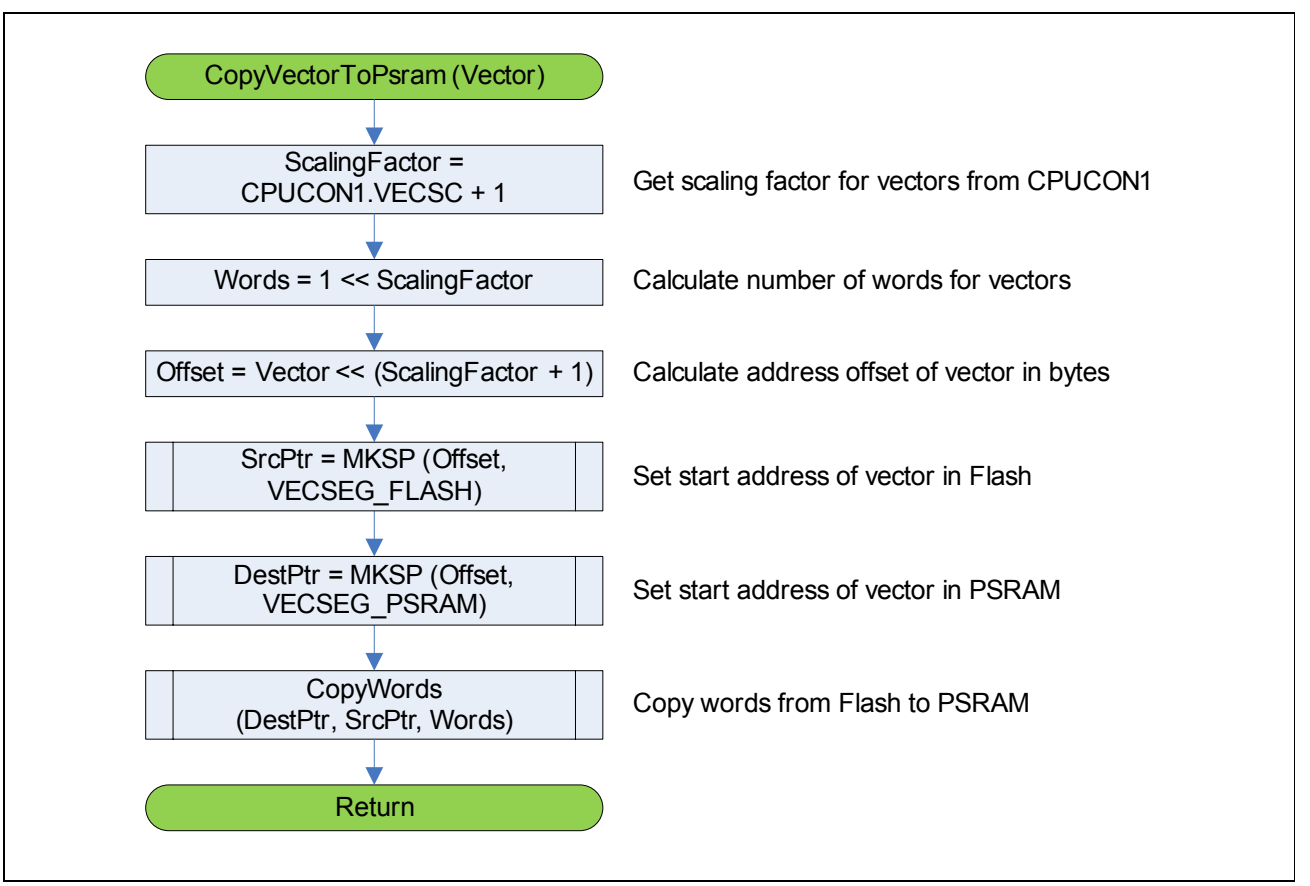

Figure 9-23 Flow-chart CopyVectorToPsram

9.6.3.3 WriteToSbram

This function writes words to SBRAM.

Table 9-32 Specification of WriteToSbram

Function Name	WriteToSBram
Input Parameters I	SrcPtr:Source pointer
	SbramOffset: Offset to SBRAM start (even)
	Words:Number of words to be copied (> 0)
Return Value	None
Description	This function writes words to SBRAM.
Notes	The user is responsible for disabling the register protection.

The function must be called by the application before FSM Standby Mode can be entered. For this mode, parts of function code and application code are stored in SBRAM and executed from PSRAM. These code sections must be located in special areas via compiler directives, and they have to be written from Flash to SBRAM via Scu_WriteToSbram. Additionally, an SBRAM header is written by this function.

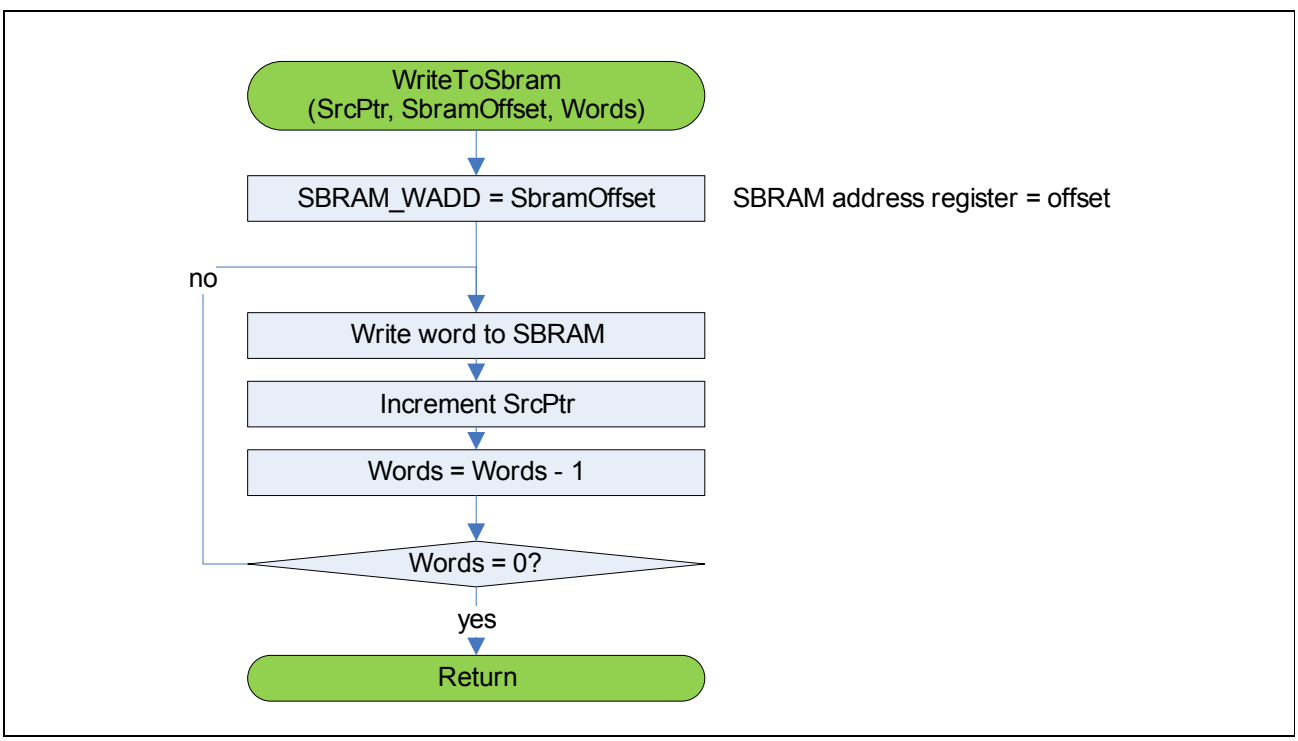

Figure 9-24 Flow-chart WriteToSbram

9.6.4 Wakeup Source

This section details functions to deal with wakeup source if multiple wakeup sources are enabled.

9.6.4.1 GetWakeupSrc

This function is called by the application after a wakeup. It determines the wakeup source if multiple wakeup sources are enabled.

*Note: Wakeup source should be cleared by **ClearWakeupSrc**.*

Table 9-33 Specification of GetWakeupSrc

Function Name	GetWakeupSrc
Input Parameters	None
Return Value	Combination of bits for wakeup sources
Description	This function returns the activated wakeup source request(s).
Notes	-

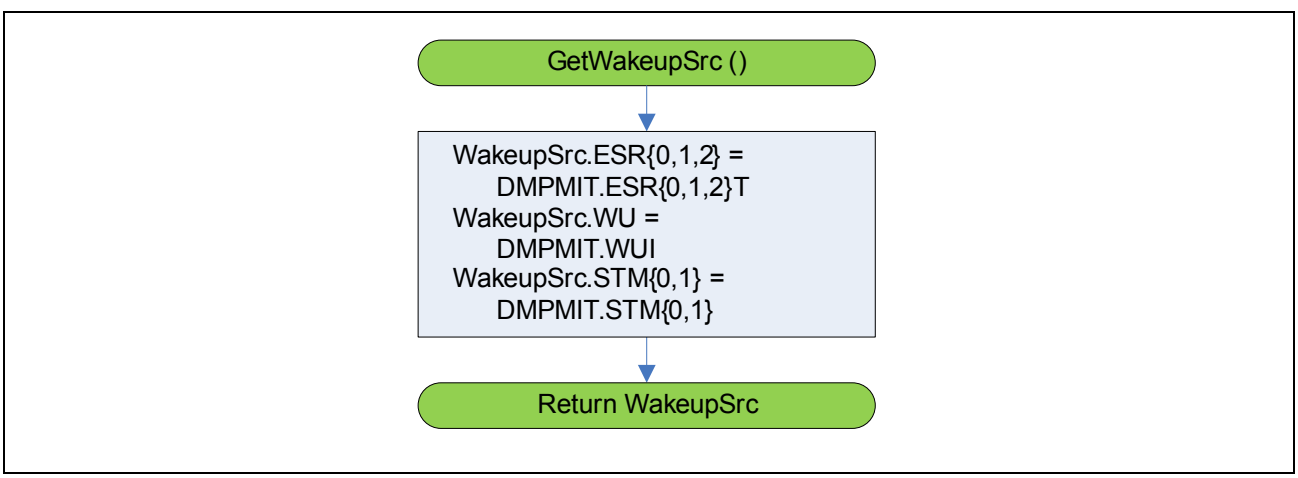

Figure 9-25 Flow-chart GetWakeupSrc

9.6.4.2 ClearWakeupSrc

The function is called by the application after a wakeup. It is recommended to clear the wakeup source request in this case.

*Note: If necessary, wakeup source should be read before by **GetWakeupSrc**.*

Table 9-34 Specification of ClearWakeupSrc

Function Name	ClearWakeupSrc
Input Parameters	WakeupSrc: Combination of bits for wakeup sources
Return Value	None
Description	This function clears the specified wakeup source request(s)
Notes	-

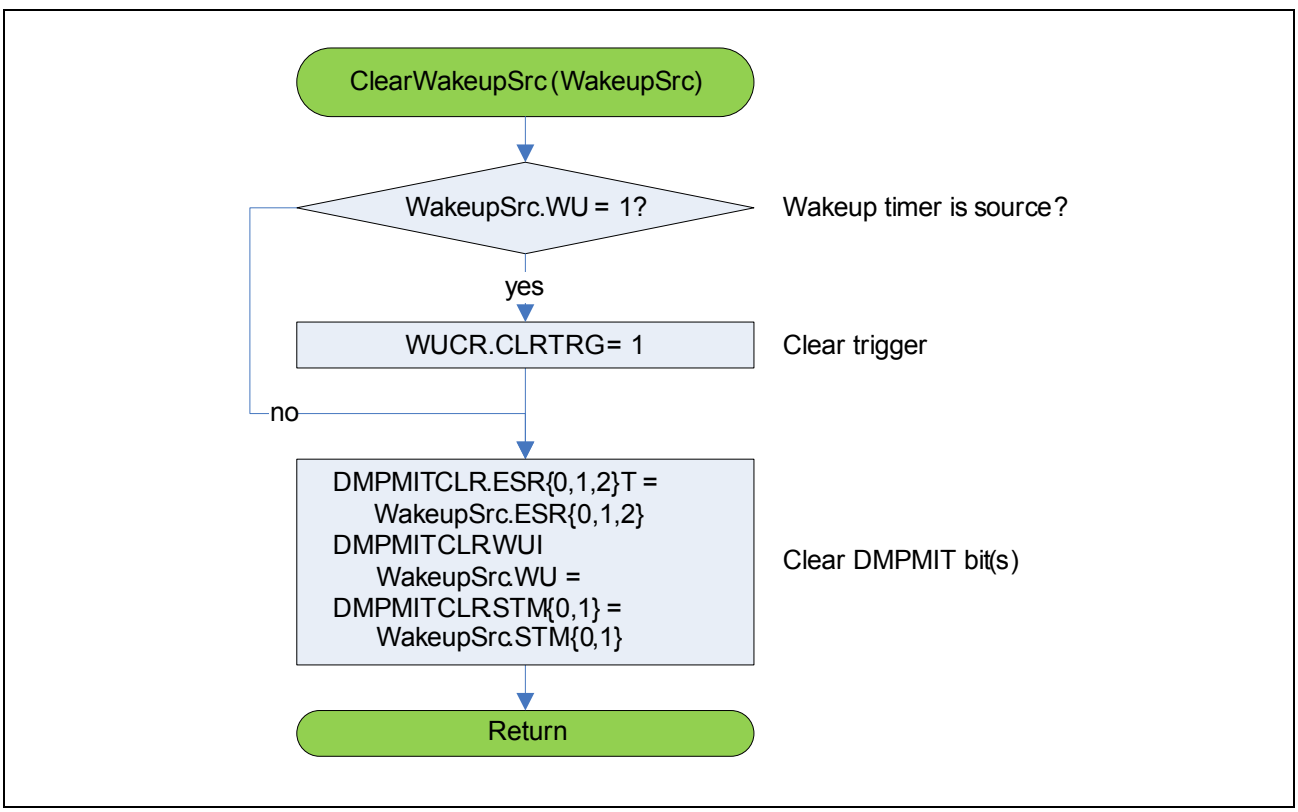

Figure 9-26 Flow-chart ClearWakeupSrc

9.7 User Functions

This chapter list the user functions allowing to adapt the operation mode transitions according the application requirements.

9.7.1 HANDLE_ERROR_PS

This function shall handle an error detected during the execution in PSRAM of a transition to or from a power-saving mode.

Table 9-35 Specification of HANDLE_ERROR_PS

Function Name	HANDLE_ERROR_PS
Input Parameters	Error: ErrorCode, 0 = NoError
Return Value	None
Description	This function shall handle an error detected during the execution in PSRAM of a transition. It can contain application specific code.
Notes	The user is responsible for disabling the register protection. The function should not return.

The function is only needed in the user program if a power-saving mode is used.

Function must be located in PSRAM via compiler directives. Code called by this function must be inlined or located in PSRAM. PSRAM code and constants must be copied to PSRAM via function CopyWords.

9.7.2 HANDLE_STOPOVER_PS

This function can contain application specific code and is called on a wakeup from Stopover Mode.

Table 9-36 Specification of

Function Name	HANDLE_STOPOVER_PS
Input Parameters	None
Return Value	0: Terminate 1: Continue Stopover Mode
Description	This function is called on a wakeup from Stopover Mode. The return value indicates if the Stopover Mode shall be continued or not. This function contains application specific code which defines the steps to be executed after wakeup and the return value.
Notes	The user is responsible for disabling the register protection. When the function is called, the system is running on f_{SYS} = f_{WU}.

The function is only needed in the user program if a power-saving mode is used.

When the function is started, the system clock is provided by the wakeup clock. If a higher and/or more stable clock is needed to execute the time-critical steps, function UseFastClockInStopover_Ps may be called to use the configured external crystal/clock or the internal clock f_{INT}. Finally, function UseWakeupClockInStopover_Ps must be called before Stopover Mode can be entered again.

In case of multiple sources, the wakeup source can be determined via function GetWakeupSrc. The wakeup source request should be cleared using function ClearWakeupSrc.

If peripherals shall be used, the following functions should be called:

- RequestSystemMode (SYSTEM_MODE_NORMAL) to enable the peripherals
- RequestSystemMode (SYSTEM_MODE_CLOCK_OFF) to disable them again.

The return value indicates if the Stopover Mode shall be continued or not. If not, function GoFromNormalModeToPowerSavingMode will resume Normal Operation Mode.

Function must be located in PSRAM via compiler directives. Code called by this function must be inlined or located in PSRAM. PSRAM code and constants must be copied to PSRAM via function CopyWords.

9.7.3 HANDLE_STANDBY_FSM_PS_SB

This function can contain application specific code and is called on a wakeup from Stopover Mode.

Table 9-37 Specification of

Function Name	HANDLE_STANDBY_FSM_PS_SB
Input Parameters	None
Return Value	0: Terminate 1: Continue FSM Standby Mode
Description	This function is called on a wakeup from FSM Standby Mode. The return value indicates if the FSM Standby Mode shall be continued or not. If not, a DMP_1 power reset will be generated. This function contains application specific code which defines the steps to be executed after wakeup and the return value.Description
Notes	When the function is called, the system is running on fSYS = f_{INT} = 5 MHz.

The function is only needed in the user program if a power-saving mode is used.

When the function is started, the system clock is provided by the internal clock source. After executing time-critical steps, function UseWakeupClockInStandbyFsm must be called which switches to wakeup clock.

In case of multiple sources, the wakeup source can be determined via function GetWakeupSrc. The wakeup source request should be cleared using function ClearWakeupSrc.

If peripherals shall be used, the following functions should be called:

- RequestSystemMode (SYSTEM_MODE_NORMAL) to enable the peripherals
- RequestSystemMode (SYSTEM_MODE_CLOCK_OFF) to disable them again.

Function must be located in PSRAM via compiler directives. Code called by this function must be inlined or located in PSRAM. PSRAM code and constants must be copied to PSRAM via function CopyWords.

9.8 Conditions

9.8.1 Timing Parameters

Some conditions for the analog components contain time restrictions. **Table 9-38** provides time information for all analog modules. All time values in this table are approximate maximum/worst case values.

Table 9-38 Time Information for Analog Modules

Name	Value	Explanation
TIME_WUCLK_RAMP_UP	0.3 µs	Delay time for wakeup clock to ramp-up
TIME_PLL_POWER_UP	50 µs	Delay time for PLL regulator to power up
TIME_INTCLK_RAMP_UP	1.5 µs	Delay time for internal clock to ramp-up
TIME_VCO_RAMP_UP	3.0 µs	Delay time for VCO to ramp-up
TIME_VCO_BAND_SWITCH	150 µs	Delay time for VCO band switch
TIME_VCO_LOCK	200 µs	Timeout for VCO to lock (PLLSTAT.VCOLOCK = 1)
TIME_VCO_FINDIS_TO_BASE	50 µs	Delay time for VCO to reach base frequency after STATCLR1.SETFINDIS is set
TIME_FREQ_CHANGE	5 µs	Delay time after frequency change
TIME_SWD_RAMP_UP	12.7 µs	Delay time for SWD to ramp-up
TIME_HPBG_RAMP_UP	3.9 µs	Delay time for BP bandgap to ramp-up
TIME_FLASH_RAMP_UP	500 µs	Timeout for flash not busy (FSR_BUSY.BUSY = 0) during ramp-up
TIME_FLASH_RAMP_DOWN	500 µs	Timeout for flash not busy (FSR_BUSY.BUSY = 0) during ramp-down
TIME_PSMSTAT	40 µs	Timeout for executed PSC sequence (PSCSTAT.PSMSTAT = 0 or PSCSTAT.LSTSEQ = 1)
TIMER_CYCLES_PSMSTAT	77 tSYS / 8	Additional timeout in timer cycles for executed PSC sequence (PSCSTAT.PSMSTAT = 0 or PSCSTAT.LSTSEQ = 1)

Table 9-39 Digital time-out Parameters

Name	Value	Explanation
ATTEMPTS_OSC_HP	10	Max. number of attempts to check OSC_HP frequency Depends on the crystal
TIME_OSC_HP_1024	1000 µs	Delay time/timeout in for 1024 cycles after high precision oscillator
TIME_OSC_HP_PLLV	5000 µs	Timeout for usable high precision oscillator (HPOSCCON.PLLV = 1) after oscillator start Depends on the crystal
TIMER_CYCLES_GSC	5000 tSYS/ 8	Timeout in timer cycles for GSC system mode change (GSCSTAT.CURRENT = compare value)

9.8.2 Other Conditions

This section describes some other conditions.

9.8.2.1 Register protection must be disabled

Description: Register protection must be disabled to allow write access to protected registers

How to check: SLS.SL == 00_b (Security Level Unprotected Mode, register protection globally disabled)

How to meet: Register protection can be disabled globally by configuring Unprotected Mode via appropriate command sequence using Security Level Control Register SLC. If register protection has been disabled globally, it should always be enabled again, after the write operation has been finished.

Instead of globally disabling register protection by selecting security level Unprotected Mode, Command 4 can be used for a single write access to protected registers (SFR, ESFR, XSFR) at security level Secured Mode. In this case, a single write access can be done atomically within four clock cycles.

Use the following sequence to write protected registers using Command 4:

```
ATOMIC    #3
EXTR      #2
MOV       SLC,#08EFFH
MOV       PROTECTED_REGISTER,NEW_VALUE
```

This approach is more time efficient, and successfully prevents interrupts/traps during register write access.

9.8.2.2 Smooth system clock frequency stepping must be applied

Description: The system clock frequency has to be changed gradually using small increments. This is required to avoid IR drop. This malicious supply condition occurs when a power regulator can not react fast enough on a sudden load variation, such as caused by a sudden change of the system clock frequency (see **Chapter 8.5.3** on restrictions regarding **EVR**). To avoid this, the system clock frequency has to be changed using the K2 divider values 2 - 3 - 5 - 16 and switching to the appropriate clock source.

For f_{SYS} = 80 MHz with f_{VCO} = 160 MHz following grid of frequencies is applicable:

- f_{VCO} / 2 = 80 MHz
- f_{VCO} / 3 = 53 MHz
- f_{VCO} / 5 = 32 MHz
- f_{VCO} / 16 = 10 MHz
- f_{INT} = 5 MHz
- f_{WU} = 140/500 kHz

For f_{SYS} = 100 MHz with f_{VCO} = 200 MHz following grid of frequencies is applicable:

- f_{VCO} / 2 = 100 MHz
- f_{VCO} / 3 = 66 MHz
- f_{VCO} / 5 = 40 MHz
- f_{VCO} / 16 = 12.5 MHz
- f_{INT} = 5 MHz
- f_{WU} = 140/500 kHz

*Note: The intermediate frequencies generated when using function **RampPll** in conjunction with PLL operation are considered to cover the grid above.*

Frequency matching must be applied in case of

- switching between different system clock sources using SYSCON0
- switching between different system clock sources inside PLL
- clock divider settings of the PLL are being modified

How to check: -

How to meet: PLL P- and N-divider settings must never be changed while VCO output is selected as system clock source. On the other hand, modifying PLL output frequency by means of the K-dividers while using PLL output as system clock source is always save.

Additionally, a smooth fall of the PLL output frequency back to f_{VCO_BASE}/(K2 + 1) is possible in conjunction with the FINDIS feature.

10 Glossary

A list of acronyms and terms with their explanation or definitions, as they are used in this document, can be found in this chapter.

ADC

On-chip A/D converter. This **Component** is mentioned here because it is one contributor to the **Base Load Current** (see **Chapter 2** on **Power Budgeting**).

Analog

The opposite of digital. In contrary to a digital **Feature**, which performs a cycle accurate operation, the timing behavior of most analog **Component**s depends on the current system environment, such as temperature and or its supply level. These conditions must be taken into account when enabling and (re-)configuring analog components.
Components with analog parts are **PLL**, **Flash**, the **EVVR**, and any kind of oscillators.

Application

The overall task an XC2000/XE166 device shall operate on together with the software. A typical automotive application is a window lift, or controlling heating ventilation air-conditioning (HVAC). Depending on the context, application means the user software **Flow** executed by the CPU or the chip embedded on a printed circuit board.

Application current profile

The chronological sorted list of how often and how long a microcontroller resides in its different Operating Modes. This is based on the assumption that in a typical application, the tasks on which a microcontroller operates re-occur frequently within a more or less defined time frame. When plotting the consumed current over time, this leads to an application specific current profile as shown in Fig. 2.

Asynchronous

Property of an operation or an entire **Component**, which performs its actions at an arbitrary time, not related to any clock. Asynchronous is the opposite of **Synchronous**. Asynchronous features do not need a clock for operation.

Base Load Current

This is a contributor to the overall current consumption of the chip. Please refer to the **Power Budgeting** chapter.

BG_HP

Abbreviation for (High Precision) **HP Bandgap**.

Brown-out

In this document, a brown-out means a slow ramp-down of the external supply voltage VDDPB. A brown-out can be detected by the **SWD**.

CAN

Stands for Controller Area Network, and is an on-chip peripheral for serial communication. The CAN protocol needs a comparably high clock speed to operate properly.

CC

Current Control. A part of the **EVR** that prevents overheating by reduction of the maximum current, which an EVR can deliver. The CC is a safeguard in case of an external short. As every analog component, it contributes to the **Base Load Current**.

Component

A building block or a part of the XC2000/XE166 that implements a feature.

ELF

Expert Level Function. A description of a basic sequential operation to allow the writer of software to handle a specific hardware feature. A typical ELF **Sequence** is how to change the value of a **PLL** divider.

ESR

External Service Request. This term stands for the ESR pins and their logic. The ESR pins are somehow special because their control logic is hosted in DMP_M, and thus can be used e.g. to trigger a wakeup at a time other pins are unavailable, because DMP_1 is disabled. Refer to **Figure 11-1** showing a **Topological overview of the XC2000/XE166 supply domains.**

EVR

Embedded Voltage Regulator. An analog component which generates the supply voltage of the nominal 1.5 V for the VDDI supply domains.

EVVR

Embedded Validated Voltage Regulator. An analog component combining an **EVR** and a **PVC**. Thus, an EVVR is not only able to generate a VDDI supply, it can also monitor it.

FAE

Field Application Engineer. An IFX specialist one should not hesitate to consult in case of questions when implementing software flows.

Feature

A functional capability of the XC2000/XE166, using one or more logical or analog components of the microcontroller, offered to be used in an application.

Firmware

A piece of software code located in the Bootrom of the XC2000/XE166 which is executed at the end of every reset. See also **Figure 6-2** in chapter **Resets**.

Flash

This means the set of on-chip non-volatile memory. The Flash is a major contributor to the Base Load Current (see **Chapter 2** on **Power Budgeting**).

Flow

An implementation of a **Sequence**. The implementation can be done in hardware or by software.

FSM

Fast Startup Mode. A Power Save Mode with CPU operation at very low speed which is used for periodical wakeup scenarios to check the **Application** environment on the necessity to resume high performance operation or to fall back in **Standby Mode** again. See **Fast Startup Mode** for more details.

GSC

The Global State Controller is in charge for the on-chip peripheral management. Using a broadcast protocol, the GSC defines the operation mode for the peripherals. In conjunction with power saving Mode Transitions, the GSC is used to broadcast Clock-off mode.

HARR

Halt after Reset Request. This **Feature** allows to stop the CPU just before executing the first instruction. Since this must be prepared before the **Application** starts, the **Firmware** checks on a pending HARR request is processed by after every reset.

HP Bandgap

High Precision Bandgap. A supply component which delivers a reference voltage for the **Flash** and **EVR**. Because of its high accuracy, this component has a non-negligible share on the Base Load Current of the system.
While the EVR can use the less current consuming **LPR**, operating the Flash always needs the HP Bandgap operating.

LPR

Low Power Reference A component of the EVR which delivers a reference voltage required for supply generation. Because the circuitry of the LPR is quite simple, it has a comparably low current consumption.

MCM

Master Clock Multiplexer. This **Component** allows a proper **Synchronous** switch between several clock sources for the system clock. Additionally, it can perform an emergency switch to a backup clock, in case the current system clock disappears.

Operating Mode

A state of the XC2000/XE166 is defined by its settings for clock and power supply as well as for analog components. The state is used within a certain time frame to serve a certain functionality which the application requires.

Operating Mode Transition

A sequence which is defined to get from one Operating Mode to another.

OSC_HP

High Precision Oscillator. The oscillator connects to the pins XTAL 1/2, the input of the PLL, and the MCM. This **Component** is also named Crystal Oscillator and has a non-negligible share of the **Base Load Current**.

OSC_LP

Trimmed Current Controlled Clock source providing an internal clock with a frequency of nominal 5 MHz (fINT).

OSC_WU

Stands for the Wakeup clock source. This free running clock generator is dedicated to clock the system at times of low performance activities, and upon any power-up activities. Its low **Base Load Current** combined with the low frequency lead to a low current consumption of the entire system in power save modes.

OSCWDT

Oscillator Watchdog, also named **OWD**. A component which monitors the clock at the output of **OSC_HP** and sets a flag in case a clock suddenly disappears. This is used together with the OSCWDT emergency feature.

OWD

Oscillator Watchdog. Synonym for **OSCWDT**.

PCB

Printed Circuit Board, where an XC2000/XE166 device and other electronic parts are mounted.

PLL

The abbreviation stands for Phase Locked Loop and is the name of the main clock component of the XC2000/XE166. **Chapter 11.2** in the **Appendix** shows a block diagram with the parts of the PLL.

PORST

Stands for Power On Reset. Pulling pin $\overline{\text{PORST}}$ to VSS will trigger a **Power-on Reset**. The same happens, whenever or as long VDDPB is below 2.0 V.

Power Resets

A group of resets provided to avoid data inconsistency in conjunction with supply problems. See **“Types of Reset” on Page 6-4** for details.

Power Transition

A hardware flow conducted by the PSC.

PSC

Power State Controller. This component of the XC2000/XE166 implements a programmable hardware flow to modify the power supply settings.

PSRAM

Program SRAM. A memory located in DMP_1. It is dedicated to store data or to execute program code. Its content is lost after the **Power Resets** and in conjunction with **Standby Mode** where DMP_1 is switched off.
See also **SBRAM**.

PVC

An **Analog** supervisory **Component** to monitor the supply level generated by an **EVR**. Each VDDI domain has its own PVC comprising two comparators each with a configurable monitoring voltage level. This means, the XC2000/XE166 has four comparators for its VDDI supplies: PVCM_1 and PVCM_2 for VDDI_1 as well as PVC1_1 and PVC_2 for VDDI_M.
A PVC (with its two comparators) and an **EVR** together are combined in an **EVVR**.

Reentrancy

The capability of a **Sequence** to be concurrently entered safely, i.e., the routine can be re-entered while it is already running. The Operating Mode Transitions defined in this document are not re-entrant, because the **Analog Component**s still have their previous settings when a sequence is called twice.

RSC

Reset State Controller. This component of the XC2000/XE166 arbitrates reset triggers and generates the reset signals of the **User Resets** with a configurable length of their active phase.

Sequence

A consecutive chain of actions which – when implemented – becomes a software or hardware **Flow**.

SBRAM

Standby SRAM. A backup memory located in the standby domain DMP_M. It is dedicated to store data in conjunction with **Standby Mode** or program code for **FSM** where the content of the **PSRAM** got lost.

SFR

Special Function Register.

SWD

Supply Watchdog. This is an independent circuitry monitoring whether the pad supply voltage VDDPB is within a configurable operating range. If not, it activates a status signal, which normally is used then to trigger a **Power Reset for VDDI_M and VDDI_1 supply domains**. A typical failure detected by the SWD is a **Brown-out**.

Synchronous

Property of an operation or an entire **Component** which performs its actions based on a clock, typically the system clock f_sys. Synchronous is the opposite of **Asynchronous**.

ULPEVR

Ultra Low Power Embedded Voltage Regulator. An analog component which generates the supply voltage of the nominal 1.5 V for the DMP_M power domain for data retention in order to fulfill low powe requirements.

Unit

A building block or a part of the XC2000/XE166 that implements a feature.

User Resets

A group of resets provided for the convenience of the **Application**. The reset signals of the User Resets are generated by the synchronous Reset State Controller (RSC). User Resets have a programmable length and need a proper system clock to operate.

VCO

Voltage Controlled Oscillator, a part of the PLL driven to a high frequency by a further external or internal clock source.

WDT

Watch Dog Timer. Running on the system frequency, the WDT awaits to be serviced in regular intervals by the application software. If the service fails to appear, it is assumed that the software flow does not work as expected, and several actions can be initiated.

WUT

This is a simple timer located in the standby domain DMP_M. It allows a wakeup from the Power Saving Modes after a defined time.

11 Appendix

This chapter provides some more information to which is referenced in various contexts.

11.1 Power Supply Topology

This overview allows identifying which features are bound to the supply domains.

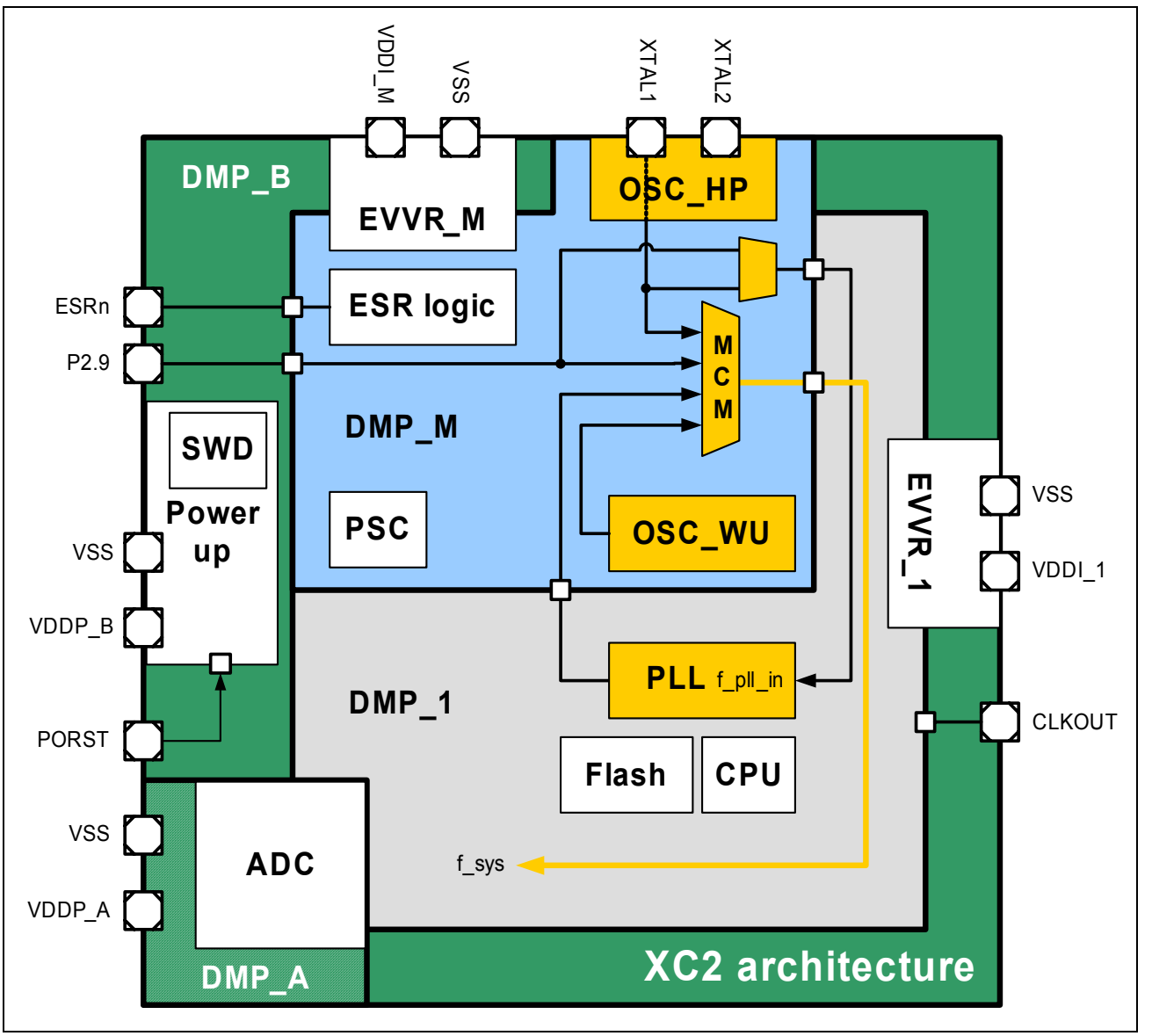

Figure 11-1 Topological overview of the XC2000/XE166 supply domains.

The control registers for the respective components are located in the same supply domain, except those for SWD, which are located in the DMP_M domain. As can be seen, the PLL can not be used when the DMP_1 is switched off.

11.2 PLL block diagram

This shows the elements of the PLL. The output f_{PLL} is used as system clock for some Operating Modes.

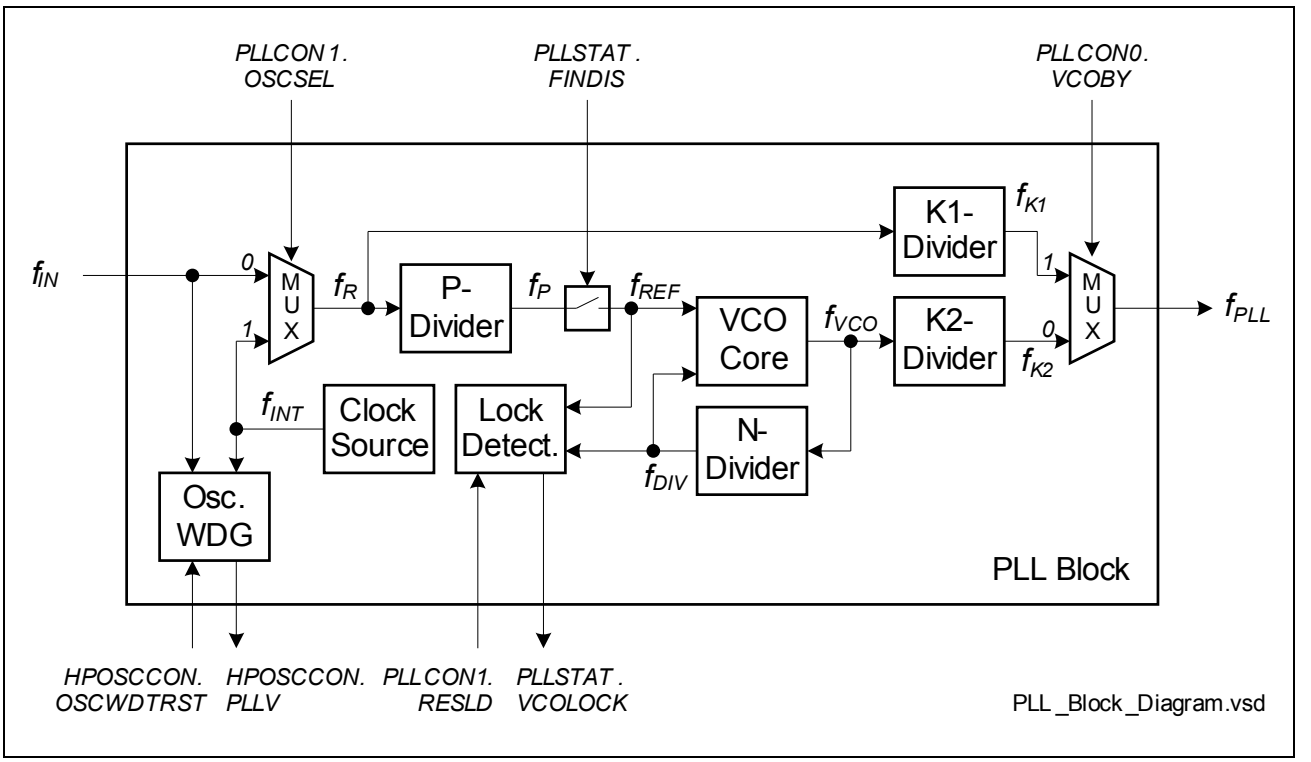

Figure 11-2 PLL Block Diagram

11.3 Time-out values

Here, the time-out values are listed, which have been successfully applied in the tests. An application may increase the time-outs freely or even use only one common time-out, covering all timings.

The timings below are partially used for the time-out variables listed in **Table 9-38**.

Table 11-1 Timings for software time-out calculation

Description	Polled bit	Type [1]	Time (µs)	Comment
Power-up time **OSC_LP**		DLY	1.5	Time until 1st clock edge occurs, considering analog supply of **PLL** is on and at specified level
Power-up time **PLL** power regulator		DLY	50	Time until supply is at the level at which VCO or OSC_LP can be operated, considering VDDP supply is available
Power-down time **PLL** power regulator		DLY	2 system clock cycles	The time the analog supply ramps down can not be determined, since it depends from operating conditions (leakage). PLLSTAT.REGSTAT is the synchronized information of a storage element in DMP_B domain, which controls the PLL regulator, thus, after 2 system clock cycles, the bit may be evaluated
Power-up time **PLL** (digital part)		DLY	immediately	A wakeup request for the PLL is not synchronized inside the PLL. PLLSTAT.PWDSTAT becomes active after 2 system clock cycles

Table 11-1 Timings for software time-out calculation (cont'd)

Description	Polled bit	Type [1]	Time (µs)	Comment
Power-down time **PLL** (digital part)		DLY	2 VCO clock cycles	PLL synchronizes a power down request with f_vco and enters sleep mode then. Thus, PLLSTAT.PWDSTAT becomes active after 2 VCO clock cycles + 2 system clock cycles
Power-up time **VCO**		DLY	3	Time until 1st clock edge occurs, considering analog supply of **PLL** is on and at specified level
Time for **VCO** to get down to base frequency after setting bit FINDIS		DLY	50	worst case from 160/112 MHz down to $f\_vcofree\_max$
Time for **VCO** to lock	PLLSTAT. VCOLOCK	TO	200	Worst case
Max. time for **VCO** to switch the VCO band		DLY	150	
Power-up time **HP Bandgap**		DLY	3.9	
Recovery time for **HP Bandgap** after reconfiguration		DLY	0.8	
Power-up time for **SWD** (from power save or off to full operation mode)		DLY	12.7	
Power-up time for **ADC**		DLY	10	
Power-down time for **ADC**		DLY	10.000 system clock cycles	Considering sample time and conversion time are configured at their maximum

Table 11-1 Timings for software time-out calculation (cont'd)

Description	Polled bit	Type [1]	Time (µs)	Comment
OWD latency		DLY	6.4	Gating time is 32 cycles of the reference clock. 5 MHz/0.2 µs from OSC_LP => 32 x 0.2 µs = 6.4 µs
Max. time **OSC_HP** needs to be disabled before it can be re-enabled	HPOSCCON. PLLV	DLY	0	Driving a crystal, **OSC_HP** can be re-enabled after an arbitrary time
Time required for system frequency to adjust to a new configured frequency, before another frequency can be configured. Used for frequency stepping		DLY	5	Time for the regulation loop of EVR to stabilize
Max. time for Flash enable	IMB_FSR_ BUSY.BUSY	TO	70	
Max. time for Flash disable	IMB_FSR_ BUSY.BUSY	TO	12	
Max. time for **OSC_HP** to produce a valid frequency after power enable	HPOSCCON. PLLV	TO	5000	This is the time proposed to be used for parameter HPOSCCON_PLLV_TIME_ OUT, typically 1024 OSC_HP clock cycles (worst case consideration is 4 MHz crystal = 250 ns/cycle).
Max. time to wait for asynchronous oscillator input selection to apply (PLLCON1. AOSCSEL)	PLLSTAT. OSCSELST	TO	1 system clock cycle	The SET bit does not reflect the feedback from **PLL**, but the status of the request signal to the PLL itself

Table 11-1 Timings for software time-out calculation (cont'd)

Description	Polled bit	Type [1]	Time (µs)	Comment
Max. time to wait for **Synchronous** oscillator input selection to apply	PLLSTAT. OSCSELST	TO	2 cycles of the slowest clock to be switched + 2 system clock cycles	
Max. time to wait for **VCO** bypass status selection to apply	PLLSTAT. VCOBYST	TO	2 cycles of the slowest clock to be switched + 2 system clock cycles	
Max. time for PLLSTAT.PRDY to be set/cleared after setting PLLCON1.PACK = 1	PLLSTAT. PRDY	TO	2 f_PLL_out cycles + 2 system clock cycles	
Max. time for PLLSTAT.PRDY to be cleared after clearing PLLCON1.PACK = 0, and modifying PLLCON1.PDIV	PLLSTAT. PRDY	TO	2 f_PLL_out cycles + 2 system clock cycles	
Max. time for PLLSTAT.NRDY to be set after setting PLLCON0.NACK = 1	PLLSTAT. NRDY	TO	2 f_PLL_out cycles + 2 system clock cycles	
Max. time for PLLSTAT.NRDY to be cleared after clearing PLLCON0.NACK = 0, and modifying PLLCON0.NDIV	PLLSTAT. NRDY	TO	2 f_PLL_out cycles + 2 system clock cycles	

Table 11-1 Timings for software time-out calculation (cont'd)

Description	Polled bit	Type [1]	Time (µs)	Comment
Max. time for PLLSTAT.K1RDY to be set after setting PLLCON2.K1ACK = 1	PLLSTAT. K1RDY	TO	$2 f\_PLL\_out$ cycles + 2 system clock cycles	
Max. time for PLLSTAT.K1RDY to be cleared after clearing PLLCON2.K1ACK = 0, and modifying PLLCON2.K1DIV	PLLSTAT. K1RDY	TO	$2 f\_PLL\_out$ cycles + 2 system clock cycles	
Max. time for PLLSTAT.K2RDY to be set after setting PLLCON3.K2ACK = 1	PLLSTAT. K2RDY	TO	$2 f\_PLL\_out$ cycles + 2 system clock cycles	
Max. time for PLLSTAT.K2RDY to be cleared after clearing PLLCON3.K2ACK = 0, and modifying PLLCON3.K2DIV	PLLSTAT. K2RDY	TO	$2 f\_PLL\_out$ cycles + 2 system clock cycles	
Max. time for PLLSTAT.FINDIS to be set after STATCLR1. SETFINDIS = 1	PLLSTAT. FINDIS	TO	1 system clock cycle	The SET bit does not reflect the feedback from **PLL**, but the status of the request signal to the PLL itself
Max. time for PLLSTAT.FINDIS to be cleared after STATCLR1. CLRFINDIS = 1	PLLSTAT. FINDIS	TO	1 system clock cycle	The SET bit does not reflect the feedback from **PLL**, but the status of the request signal to the PLL itself

Table 11-1 Timings for software time-out calculation (cont'd)

Description	Polled bit	Type [1]	Time (µs)	Comment
Max. time for **PSC** to enter idle state after requesting an A/B sequence (SEQCON. SEQxTRG = 1)	PSCSTAT. PSMSTAT	TO	see comment	Depends on whether the transition performed will wait on an analog state to be reached[2]
Max. time for power transition sequences, including **GSC** entry and exit sequences, to complete a run	PSCSTAT. xACT	TO	1000	Max. time for a power transition sequence to be run completely; to be applied for handshake
Max. time for SEQCON.SEQxEN to be cleared after starting sequence A/B, using software trigger (SEQCON. SEQxTRG)	SEQCON. SEQxEN	TO	2 system clock cycles	

[1] Type definition: DLY = Delay; TO = Time-Out.

[2] The overall maximum transition delay possible is 6 * 67 system clock cycles, plus analog delay. Maximum analog delay is expected to be about 40 µs to load DMP_1. Proposal: For the transitions as of today, consider one time a maximum digital delay (wait 64 cycles), whilst a 6-step transition waiting once for an entire ramp up of DMP_1. This results in an overall delay of 5 system clock cycles for the state machine plus 6 x 3 + 64 system clock cycles + 40 µs = 77 system clock cycles + 40 µs.

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG