FAQ Application Note for TLE986xQX, TLE987xQX

Frequently Asked Questions and Application Hints

About this document

Scope and purpose

This Application Note is intended to provide helpful suggestions and hints how to set up and handle specific modules and functionalities which are not subject of the Users Manual or Data Sheet and might be interesting for end users. It is organized in a frequently asked question style and doesn't follow any specific order.

Note: The following information is given as a hint for the implementation of the device only and shall not be regarded as a description or warranty of a certain functionality, condition or quality of the device.

Intended audience

This template is intended for Customer and FAE to document frequently asked question and answers for the embedded Power IC, TLE986xQX and TLE987xQX device family.

Table of Contents

1 Introduction

This Application Note lists topics emerged from frequently asked questions of users or from changed user requirements. Each topic is organized in three sections:

- **Topic:**
  - Short description of the issue.
- **Description:**
  - More details about the topic
- **Implementation hint (optional):**
  - Instruction how to handle this topic

2 Collection of Questions and Topics

This chapter gives an overview of the collected Questions and Topics.

Table 1 Table of Questions

3 GPIO Port Map and Alternate Functions

What PIN is connected to which peripheral?

The TLE986xQX and TLE987xQX have 15 port pins organized in three parallel ports: Port 0 (P0), Port 1 (P1) and Port 2 (P2). Each port pin has a pair of internal pull-up and pull-down devices that can be individually enabled or disabled. Either pull-up or pull-down devices can be enabled at a time, for a single port pin. P0 and P1 are bidirectional and can be used as general purpose input/output (GPIO) or to perform alternate input/output functions for the on-chip peripherals. When configured as an output, the open drain mode can be selected. On Port 2 (P2) analog inputs are shared with general purpose input.

3.1 Description: GPIO Register description

Each port consists of 8-bit control and data registers. The registers are defined in Table 2.

Table 2 Port Register

Register Short Name	Register Long Name	Description
Px_DATA	Port x Data Register	x = {0,1,2}
Px_DIR	Port x Direction Register	x = {0,1,2}
Px_OD	Port x Open Drain Control Register	x = {0,1}
Px_PUDSEL	Port x Pull-Up/Pull-Down Select Register	x = {0,1,2}
Px_PUDEN	Port x Pull-Up/Pull-Down Enable Register	x = {0,1,2}
Px_ALTSEL0	Port x Alternate Select Register 0	x = {0,1}
Px_ALTSEL1	Port x Alternate Select Register 1	x = {0,1}

3.2 Implementation: Alternate Function configuration example

The ports P0 and P1 can be configured to four different output functions. The default configuration is the GPIO function. The three remaining functions are alternate output functions.

The alternate output function selection is splitted in two bitfields (e.g. **P1_ALTSEL0** and **P1_ALTSEL1**). **ALTSEL1** contains the most significant bit. **ALTSEL0** contains the least significant bit. The given example code shows how to configure these bitfields to connect UART2 module (TXD, RXD) with the GPIOs (P1.0, P1.1).

```
/* connect UART2 to GPIO                                          */
/* set P1.1 to UART2_TXD:                                         */
PORT->P1_DIR.bit.P1 = 1u;        /* PORT P1.1 output configuration */
PORT->P1_ALTSEL0.bit.P1 = 1u; /* UART2_TXD alternate function 3  */
PORT->P1_ALTSEL1.bit.P1 = 1u; /* UART2_TXD alternate function 3 */
/* Set P1.2 to UART2_RXD:                                         */
PORT->P1_DIR.bit.P2 = 0u;        /* PORT P1.2 input configuration  */
```

3.3 Implementation: Port Map of Alternate Functions

Graphical Portmap of Alternate Functions

Each pin is able to handle multiple purposes. Figure 1 shows the internal signals mapped to GPIOs. The arrow boxes contain the signal names and indicate the data flow direction.

GPIO Port Map and Alternate Functions

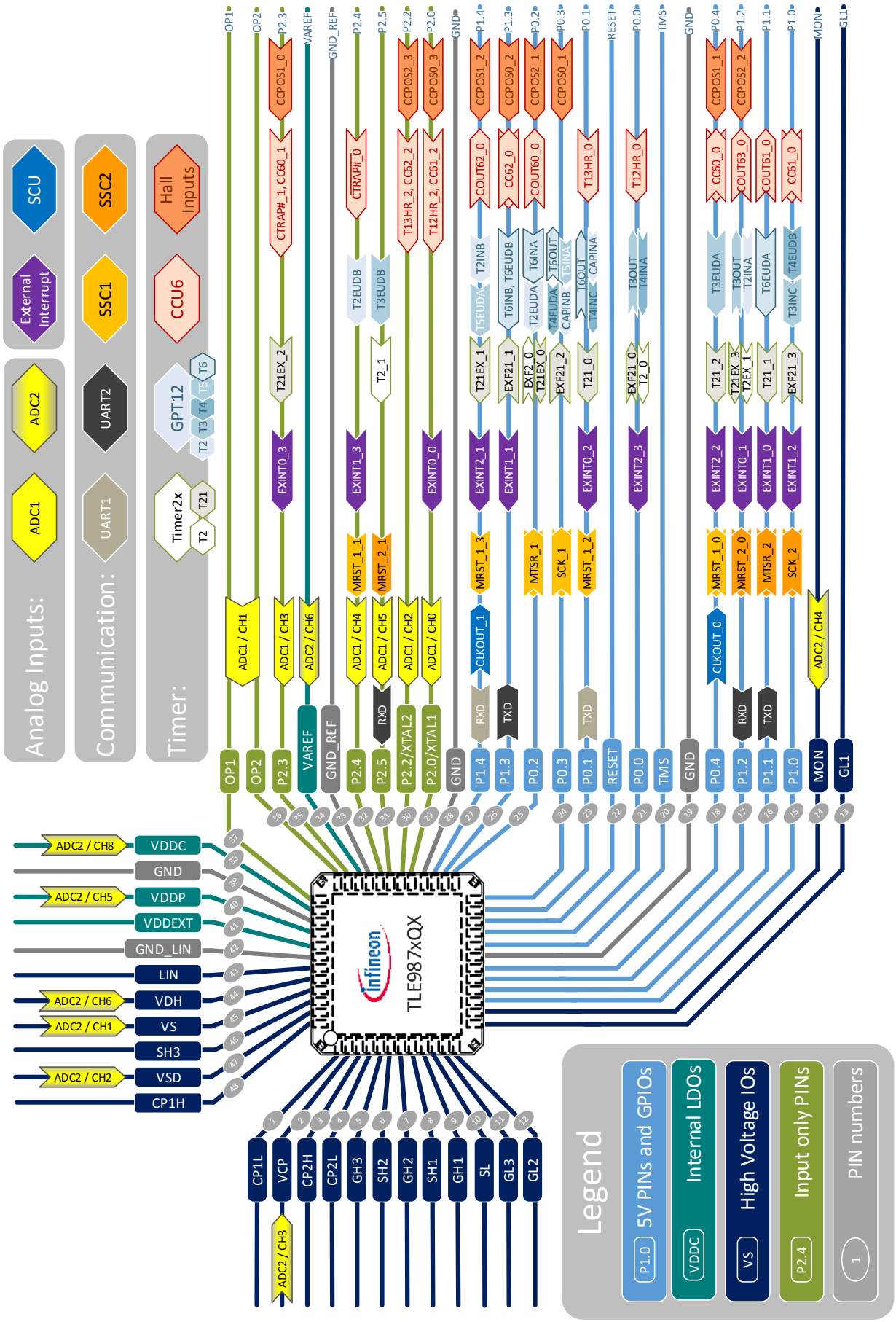

Figure 1 Port Map of Alternate Functions

4 SWD (Serial Wire Debug) Interface Circuitry

How to connect the Debug Interface?

The Serial Wire Debug interface is used to download code to the embedded Power IC or to debug the chip. This Topic explains how to implement the circuitry around the chip to achieve a successfull SWD connection.

4.1 Description: SWD (Serial Wire Debug) Interface

Serial Wire Debug (SWD) provides a debug port for severely pin limited packages, often the case for small package microcontrollers but also complex ASICs where limiting pin-count is critical and can be the controlling factor in device costs.

For SWD the TLE9879 uses the pins TMS (data) and P0.0 (clock). On the Evaluation boards, the signals are routed through a 5x2 pinheader (SWD connector). The following Implementation explains the connection between embedded Power IC and SWD Interface.

4.2 Implementation: SWD Interface connection to TLE986x/ TLE987x

The SWD Interface can be directly connected to the TLE987x and TLE986x family. The use of external pull up or pull down resistors is not needed, due to internal pull down resistors. Figure 2 shows the interconnections between TLE Device and and SWD Connector.

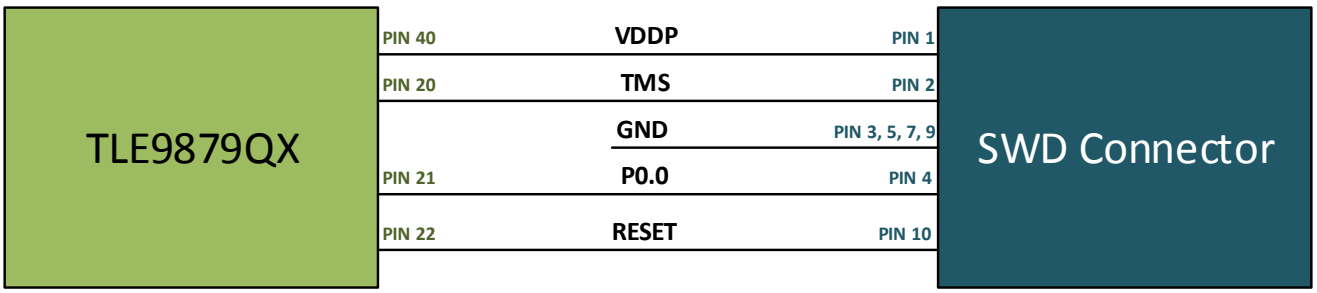

Figure 2 SWD Connection to the TLE987x and TLE986x Device

On TLE9879 and TLE9869 Evalkit SWD Interface PIN 9 is used to deactivate the onboard debugging circuit. For a typical implementation this PIN is used as GND. The Pinout is shown in Figure 3.

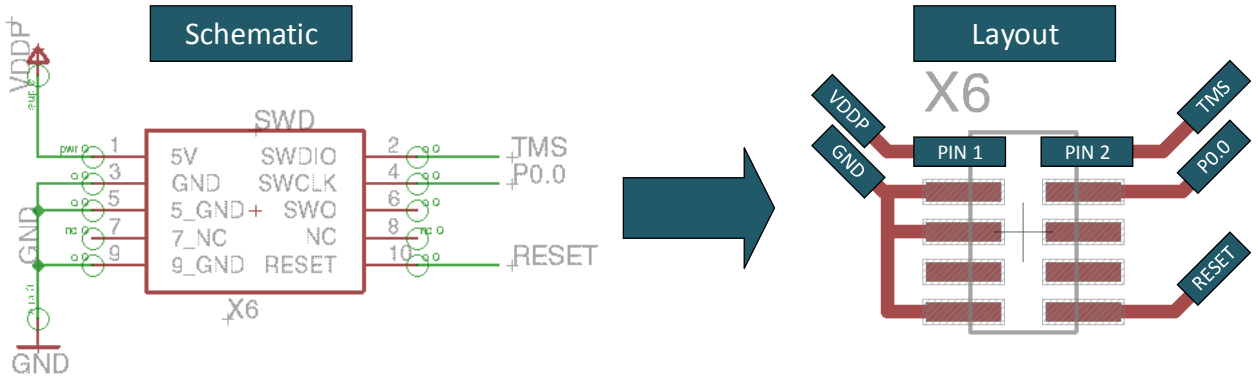

Figure 3 SWD Interface implementation for application

5 Bootup Configuration

Why does the chip not start up, after reset?

Using a new device can cause issues, due to the device is not executing user code. This chapter explains how to configure the chip to boot up and enter Users Code as expected.

5.1 Description: BSL Connection Window

After the reset PIN releases and some fundamental initializations of the device has been done, the BSL connection acception window starts. The duration of this timing window is defined by the None-Activity-Counter (**NAC**). In this time period, the device can be programmed via LIN. After the NAC expires, the chip will enter the user mode as shown in Figure 4.The following chapter provides some more inside view into the NAC value.

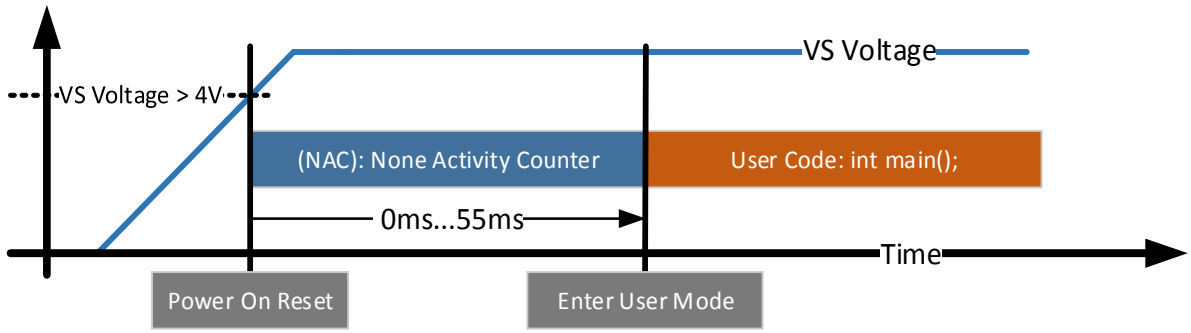

Figure 4 Startup Procedure

5.1.1 None-Activity-Counter - NAC

The NAC timer is started during startup before the BSL mode, inside the firmware, starts. Once the NAC timer has expired the boot-up process will be continued and the control of the device will be given to the user application. The NAC is a value which will be defined by the user and stored inside the code flash. Usually the NAC value is part of the user application which was downloaded before.

The NAC value is stored inside one byte. Only six bits of the NAC byte are defining the timeout value of the NAC. The NAC value inside the NVM is secured by storeing it as a 1s-complement value in the following byte . Only if the true NAC value and the complement NAC do match (means: not (true NAC) == complement NAC) the NAC value is valid and the NAC timer will be activated during start-up. In case of an invalid value, i.e. like for an erased flash where both bytes **containing 0xFF**, the **NAC timer never expires**, means the device will stay and wait in BSL mode. This behavior is especially usefull for fresh devices, where no user application has been downloaded to the device yet. Here the firmware will not branch into user mode, but instead it will stay in BSL mode and keeps waiting for any BSL communication to download any user application into the flash.

The NAC value and its complement is stored at the following addresses inside the user accessible flash:

Table 3 Address of the NAC values inside flash

Address	Usage
0x11000000 + (Total_Flash_Size - 0x1004)	true NAC value
0x11000000 + (Total_Flash_Size - 0x1003)	complement NAC value

The meaning of the NAC bits are listed in Table 4.

Table 4 NAC bit meaning

NAC Bit	Usage
5..0	NAC expire value n, (n - 1) * 5ms, 0ms..55ms 0x00, 0x0D..0x3F: NAC never expires, device stays in BSL Mode, BSL mode is active 0x01: BSL mode is skipped, BSL mode deactivated 0x02..0x0C: NAC timeouts between 5ms..55ms, BSL mode is active
7..6	0b01: BSL interface selection Fast-LIN 0b1x: BSL interface selection UART

A fresh device, or a completly ereased device always selects Fast-LIN as BSL interface and stays in BSL mode during start-up.

5.1.2 Node Address - NAD

The NAD is used for the BSL communication to select an individual node. The NAD is a 8 bit value, where only the values 0x01 to 0xFF are valid, 0x00 is an invalid value. The value 0xFF acts as a broadcast, this means all devices connected to the LIN line are addressed no matter which NAD value is programmed inside the device. The broadcast can be used to establish a BSL connection to devices where the programmed NAD value is unknown.

The NAD value is stored inside one byte inside the code flash area. The NAD value inside the NVM is secured by storeing it as a 1s-complement value in the byte following the NAD value. Only if the true NAD value and the complement NAD do match (means: not (true NAD) == complement NAD) the NAD value is valid otherwise the device will only react on the default NAD value, which is 0x7F.

The NAD value and its complement is stored at the following addresses inside the user accessible flash:

Table 5 Address of the NAD values inside flash

Address	Usage
0x11000000 + (Total_Flash_Size - 0x1002)	true NAD value
0x11000000 + (Total_Flash_Size - 0x1001)	complement NAD value

Table 6 NAD meaning

NAD Bit	Usage
0x00	invalid NAD value, device will not react on these value, default NAD = 0x7F is used
0x01..0xFE	valid NAD values
0xFF	broadcast NAD, all devices will react on this value

Note: A fresh device, or a completly ereased device always react on the default NAD (0x7F) only.

5.2 Implementation: Write NAC NAD values to the correct position in Flash

According to Chapter 5.1.1 and Chapter 5.1.2 the NAC and NAD Value have to be written to the correct position in the Code Flash. Figure 5 shows the correct position for every Device of the TLE987x and TLE986x embedded Power family.

The Implementation of the NAC-NAD value setting can be found in the Software Developement Kit based on Keil µVision5.

		TLE987xQX, TLE986xQX			
Sector No.	Sector Start Addr.	128K Device	64K Device	48K Device	36K Device
31	1101F000	Data Flash			
30	1101E000	NAC / NAD			
29	1101D000				
28	1101C000				
27	1101B000				
26	1101A000				
25	11019000				
24	11018000				
23	11017000				
22	11016000				
21	11015000				
20	11014000				
19	11013000				
18	11012000				
17	11011000				
16	11010000				
15	1100F000		Data Flash		
14	1100E000		NAC / NAD		
13	1100D000				
12	1100C000				
11	1100B000			Data Flash	
10	1100A000			NAC / NAD	
9	11009000				
8	11008000				Data Flash
7	11007000				NAC / NAD
6	11006000				
5	11005000				
4	11004000				
3	11003000				
2	11002000				
1	11001000				
0	11000000				

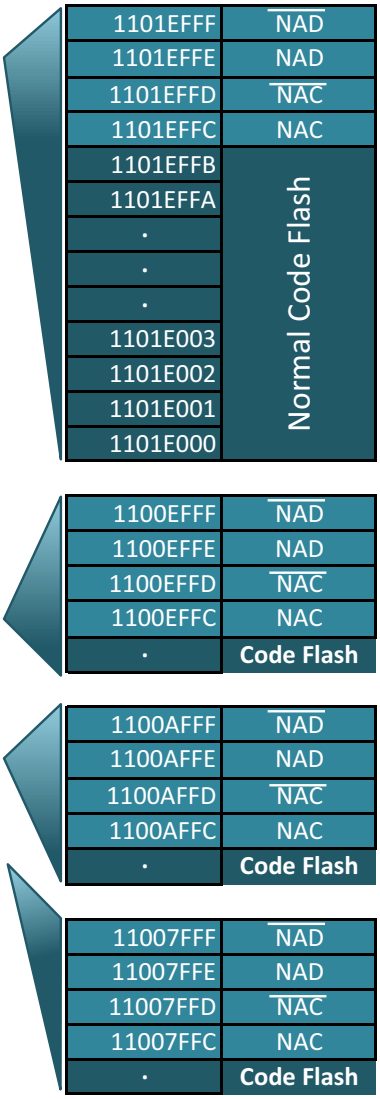

Figure 5 Memory Map

The following lines of code are part of the files “system_TLE987x.c/...6x.c”. The defines can be found in “TLE987x.h/...6x.h” and “tle_device.h”. The Code will also work “standalone”

```
#define ProgFlashSize    (0x8000U)  /*Flashsize for TLE9871/...61                */
#define ProgFlashSize    (0xB000U)  /*Flashsize for TLE9873QXW40                 */
#define ProgFlashSize    (0xF000U)  /*Flashsize for TLE9877/...67                */
#define ProgFlashSize    (0x1F000U) /*Flashsize for TLE9879/...69                */
```

Please, use only one of the definitions above, at a time. For example TLE9879(0x1F000U) for TLE9879 Evalkit.

```
#define NAD_NAC (0xFE01BA45u) /*Example: 4 bytes for NAC = 0x45u and NAD = 0x01u */
#define ProgFlashStart 0x11000000U)                    /*start address code flash*/
#define DataFlashStart(ProgFlashStart + ProgFlashSize) /*start address data flash*/
#define NACStart        (DataFlashStart - 4U)          /*start NAC value          */
/* Set NAC NAD values as attribute:                                               */
const uint32 p_NACNAD  __attribute__((at(NACStart),used)) = (uint32)NAD_NAC;
```

6 Watchdog Handling WDT1

Why does the chip do resets every second?

The WDT1 Watchdog will perform frequently resets, if it is not serviced within the open window.

6.1 Description: Window Watchdog Timer WDT1

The WDT1 provides a reliable and secure way to detect and recover from SW or HW failures. It has an independent clocking source and power supply. If the WDT1 is not serviced (refreshed) within the allowed window a system malfunction is assumed and an internal RESET is performed.
A reset occurs with each missed service, or servicing in the wrong window. If WDT1 servicing failed 5 times, the device enters SLEEP MODE. The window can be freely programmed. The WDT1 cannot be switched off in Active Mode (exception in Debug Mode).
Figure 6 shows the relation between closed and open window. The safe trigger area expects the clock accuracy.

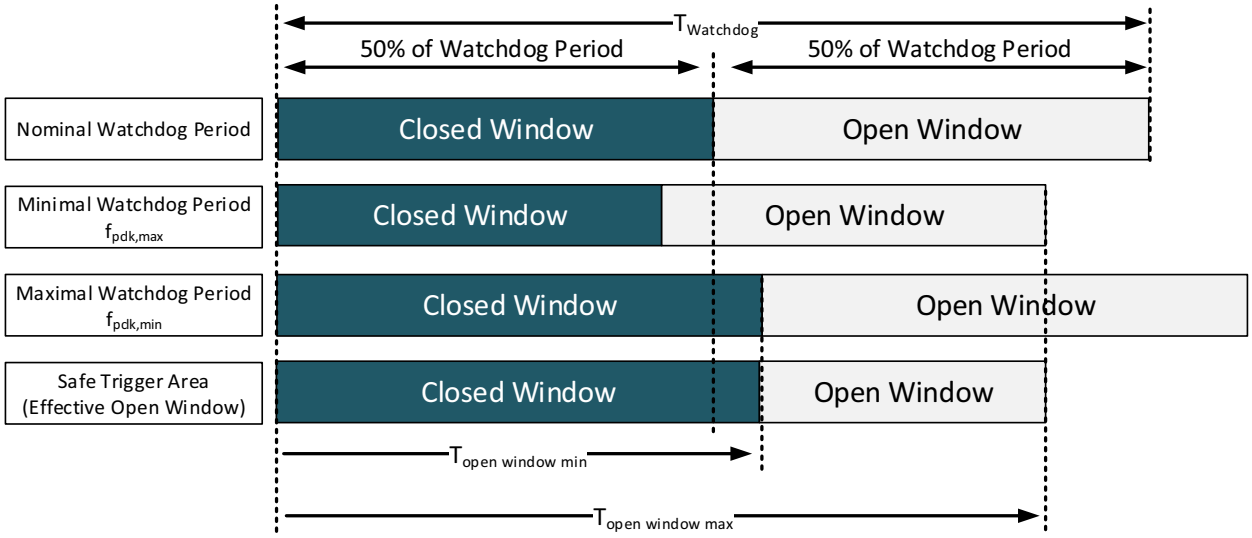

Figure 6 Watchdow Window Structure

6.2 Implementation: Watchdog Handling µVision 5

The following Code can be found in “wdt1.c”. The “WDT1_Init()” function is called in “SystemInit()”

```
extern uint32 WD_Counter; /*Part of "Wdt1.h"                              */
void WDT1_Init(void)      /*Part of "Wdt1.c"                              */
{
  uint32 ui;
  /*calc SysTick reload based on SystemFrequency                          */
  ui = (uint32)SCU_FSYS / SysTickFreq;
  CPU->SYSTICK_RL.reg = ui;  /* program SysTick timer                      */
  CPU->SYSTICK_CUR.reg = 0u; /* reset SysTick timer                        */
  CPU->SYSTICK_CS.bit.CLKSOURCE = 1u;/* CLKSRC=CPU clk                     */
  CPU->SYSTICK_CS.bit.TICKINT = 1u;  /* TICK Interrupt = enabled           */
  CPU->SYSTICK_CS.bit.ENABLE = 1u;   /* ENABLE SysTick Timer               */
  SCUPM->WDT1_TRIG.reg = (uint8) SCUPM_WDT1_TRIG;/* trigger inital WDT1 service */
  WD_Counter = 0u;           /* reset window counter                       */
  bSOWactive = false;        /* reset SOW active signal                    */
}
```

In the code examples, included in the pack file, the Watchdog is serviced with “WDT1_Service(void)”. The function checks if the window is already open. In this case, the Watchdog is triggered and a “1” is returned. If the Window Counter is less than 70% of the WDT1 period, the default value “0” is returned.

```
int WDT1_Service(void) /* This function is part of the file "wdt1.c"            */
{
  int result;
  result = 0;
  /* check if Window Counter is beyond 70% of WDT1 period                       */
  /* or if a SOW service has been done before                                   */
  if ((WD_Counter > SCUPM_WDT1_TRIGGER) || (bSOWactive == true))
  {
    SCUPM->WDT1_TRIG.reg = (uint8) SCUPM_WDT1_TRIG; /* service WDT1            */
    WD_Counter = 0u;   /* reset window counter                                 */
    bSOWactive = false;/* reset "short open window" active flag                */
    result = 1;
  }
  return (result);
}
```

6.3 Implementation: WDT1 Hints

In this Chapter some further Informations are given. They can be used to find the reason of a potential issue.

6.3.1 Potential WDT1 Traps:

- System seems not to run at all, but in Debugger it works
  - WDT1 gets serviced without the “Long Open” Window after reset
  - WDT1 would get serviced after the “Long Open” Window has expired
  - WDT1 is not getting serviced at all
- System runs to a certain point but then performs a reset
  - WDT1 gets serviced within the “Closed” Window part of the WatchDog period
  - WDT1 would get serviced after the WatchDog period has expired

6.3.2 Consequence:

- Flashing of new “fixed” user code might not be possible anymore

6.3.3 Root Cause

- The device enters SLEEP Mode after five WatchDog fails, debugger connection is not possible

6.3.4 Solution:

- The device needs to keep awake, using MON1 as a wake-up source, and RESET as wake-up trigger
- For this purpose connect the pin RESET with pin MON1
- Set VS below 8V (the MONx threshold is defined as VS/2, RESET drives 5V max.)
- The output of RESET is fed into the MON1 and is recognized as wake-up event
- By this the device stays alive and can be reflashed
- after successfull flash update, the connection between RESET and MON1 can be removed, VS can be risen again

7 Revision History

Revision	Date	Changes
1.0	2017-04-21	Released version

Edition 2017-04-21
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about any aspect of this document?
Email: erratum@infineon.com

Document reference
<Doc_Number>

IMPORTANT NOTICE

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

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

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

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