Application Note

DA14680/681 Recovery from System Level ESD Events

AN-B-056

Abstract

This document describes a method to recover the DA14680/681 from a possible system level ESD event. Such an ESD event might happen in applications using the external 32.768kHz crystal. The document presents a simple and cheap external Watchdog Circuit to Reset the DA14680/681 and recover from the event. Applications using the internal RCX oscillator and no external 32.768 KHz crystal do not require this circuitry.

DA14680/681 Recovery from System Level ESD Events

Contents

Figures

Tables

1 Terms and Definitions

BLE	Bluetooth Low Energy ®
CD	Contact Discharge
ESD	Electro Static Discharge
SoC	System on Chip
WD	WatchDog

2 References

[1] DA14681, Datasheet, Dialog Semiconductor.

[2] AN-B-061, Application Hardware Design Guidelines, Application Note, Dialog Semiconductor.

[3] JEDEC, System Level ESD Part 1, JEP161, JEDEC, 2011.

[4] JEDEC, System Level ESD Part 2, JEP162, JEDEC, 2013.

[5] IEC, Electromagnetic Compatibility Part 4-2, IEC 61000-4-2, IEC, 2009.

3 Introduction

During non-destructive ESD testing of the final application (sytem level ESD testing) by e.g. shooting at the charger contact pins of a wearable application when the DA14680/681 SoC is in sleep mode, there is a chance the DA14680/681 does not wake up anymore by itself. When the system (application) is not recovering by itself, it is considered as a fail from a system level ESD point of view. The system level ESD test is considered as a pass when the DA14680/681 would reboot, start to advertise and can re-connect. A temporary loss of function is allowed as long as it recovers its normal operation without operator intervention. [3] [4] [5].

When asleep, typically the DA14680/681 will keep operating at ESD contact discharge levels upto +/- 4KV. At higher ESD levels, depending on the ESD protection, there is a chance the system will not wake up. This document explains a HW solution to overcome such a limitation. Since system level ESD testing is critical to a product design, we recommend customers using the DA14680/681 to implement the suggested solution in their system if an external 32.768 KHz crystal is used.
Some general system level ESD protection guidelines for applications are given in [2].

When subjected to ESD testing with the DA14680/681 in active mode does not reveal problems, the chip continues to operate up to the required ESD levels: typically +/- 6KV contact discharge.

Applications using the internal RCX oscillator and no external 32.768 KHz crystal are expected to pass the system level ESD testing as long as the general recommendations are applied for ESD testing as indicated in [2].

4 Not Waking Up and Introducing Methods to Recover

As mentioned, the reason the DA14680/681 SoC does not wake up anymore is because the 32.768 KHz crystal oscillator (XTAL32K) stops functioning. It stops because the XTAL32K pins change function when subjected to high level ESD pulses and typically will change to GPIO function.
The root cause lies in the fact the XTAL32K pins P2_0 and P2_1 are multiplexed with digital GPIO functions. The assigned XTAL32K functions of these pins are latched during sleep which can get lost when the SoC is subjected to high level ESD discharges and the latches are changing state.

The effect on the system (application) is that the DA14680/681 does not wake up anymore from sleep, and stops advertising or loses connection.

To recover from the stuck situation, a HW reset on the RST pin of the DA14680/681 is required. This hard reset signal can come from an external microcontroller when present, or from an external watchdog circuit. A simple and cheap watchdog circuit is described in the next section.

4.1 Description of the External Watchdog Circuit

The suggested external watchdog circuit is shown in Figure 1 below. This watchdog circuit is driven by a LED port of the DA14680/681 and has been qualified by Dialog Semiconductor for a battery supply range of 3V upto 4.2V, and a temperature range of -30°C to +80°C.

The circuit compromises an R-C timing network, a small low power Schmitt trigger buffer (non-inverting) suitable for 5V operation, an R-C filter and a diode.

During active mode of the DA14680/681, the voltage on C2 is kept low (~0V) by a low frequency PWM signal from the DA14680/681 port. The output of the Schmitt trigger chip (NL17SZ17) is also kept low, and the reset line which is connected the RST pin of the DA14680/681 is also low. The DA14680/681 can continue to operate.

In case the DA14680/681 is not waking up from sleep anymore, the PWM signal from the LED1 port will not be present anymore (the state of the LED port will be open drain, high impedance) and the C2 will be slowly charged via R2, until the input threshold voltage of the Schmitt trigger is reached. The Schmitt trigger output will switch from low to high, the C4 and R5 network will create a pulse at the RESET/RST line which will reset the DA14680/681 SoC. The SoC reboots, and again will produce the PWM signal from the LED port which will quickly discharge the capacitor C2. The whole system will be in normal operating mode again.

The RESET pulse width typically is 26 ms, and varies just a little over the battery supply voltage range. After the pulse has been generated, the RESET line will be low, independent on the state of the Schmitt trigger buffer output and so it is ensured that a permanent reset state (RST = High) will not occur. The series diode has been added to protect the RST pin of the DA14680/681 for negative voltages when the Schmitt trigger output switches from high to low.

To avoid a reset condition, capacitor C2 must be regularly discharged. This is accomplished by software, producing a low frequency PWM signal at a LED port. As long the system is active (no sleep mode), LED1 is activated and capacitor C2 is discharged during the low period of the PWM signal. The PWM signal's low frequency is selected such that the low period of the LED PWM signal is at least as long as a connection or an advertising interval: 28Hz.

The time needed for the C2 voltage to reach the Schmitt trigger input threshold voltage - which roughly is half the value of the supply voltage - is about 4 seconds. So, the Bluetooth® advertising/connection interval time needs to be smaller than the 4 seconds to guarantee normal operation of the DA14680/681 SoC. Else the R-C timing network has to be adapted for a bigger R-C time constant.

The LED port is used for the recommended circuitry because these ports do not have a latch included like any other GPIO that can be affected by a system level ESD hit.

Dialog Semiconductor continues testing with other brands and types of Schmitt trigger ICs and diodes. When good alternatives are found, these will be added to this application note.

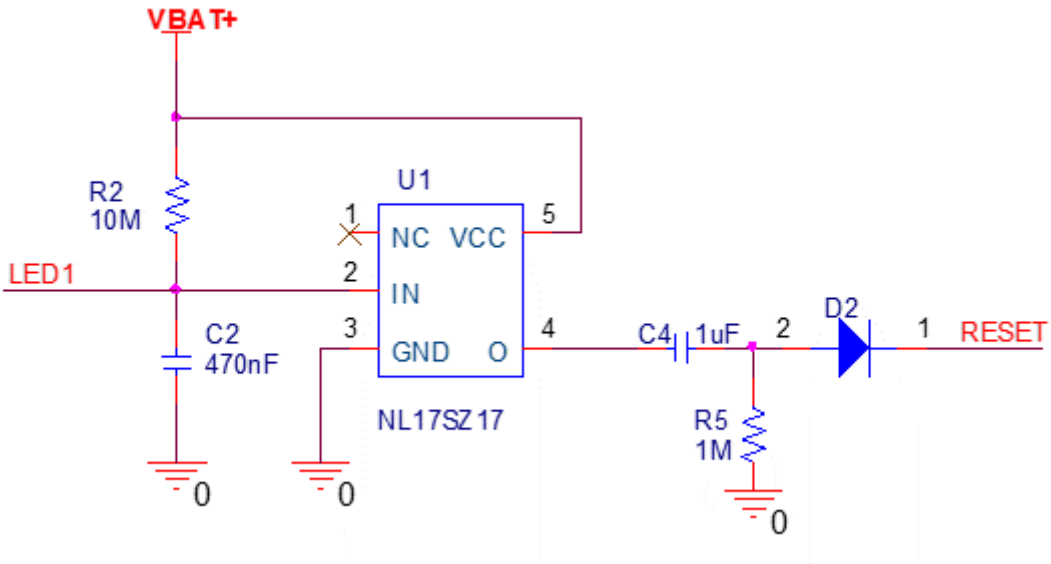

Figure 1: Watchdog Circuit Driven by PWM from a LED Port

Table 1: List of Used Components Belonging to the Watchdog Circuit in Figure 1.

R2	C2	U1	C4	R5	D2
10 MOhm	470 nF	OnSemi NL17SZ17	1 µF	1 MOhm	Diode 1N4148W

The current consumption of the whole watchdog circuit in normal operating state is less than 500nA. It was observed that the circuit's current consumption increases when the Vin voltage of the Schmitt trigger is higher than 0.5V. Based on this, it is advised to keep Vin below 0.5V in normal operating condition. For the indicated R2-C2 values, C2 will be charged upto 0.5V after 0.7 second. This leads to a maximum connection or advertising interval time of 0.7 second.

If a larger Bluetooth® advertising/connection interval time is required, the R2-C2 time constant has to be increased, preferably by increasing the capacitance of C2.
For instance, for a wanted interval time of 1 second and Vin to stay below 0.5V, the value of C2 should be 680nF. For a 1.5 second interval time, C should be 1µF to avoid Vin is rising above 0.5V.

4.2 Alternative External Watchdog Circuit

In case a LED port cannot be used, because these are used for driving LEDs, alternatively a GPIO can be used to drive the external watchdog circuit. But since the state of a GPIO cannot be guaranteed when the chip got stuck, additional components are required. This circuitry is shown in Figure 2.

In order to guarantee that C2 will charge when the DA14680/681 has stopped to function, independent on the GPIO state which can be low or high, the C1-R1 network, D1 and Q1 are added.

The PWM signal from the GPIO, P1_1 in this example, is AC coupled to Q1's gate. As long the PWM signal at the GPIO output is present, the gate of Q1 will be driven, and Mosfet Q1 will discharge C2. When the XTAL32K oscillator stops working, and the chip is stuck, the PWM signal from P1_1 is not generated anymore, Q1 will not be driven anymore and thus will not discharge C2. C2 starts to charge via R2 until the Schmitt trigger input threshold level is reached and the DA14680/681 is reset. The DA14680/681 reboots, and the PWM signal at P1_1 is generated again, resulting in a quickly discharged C2.

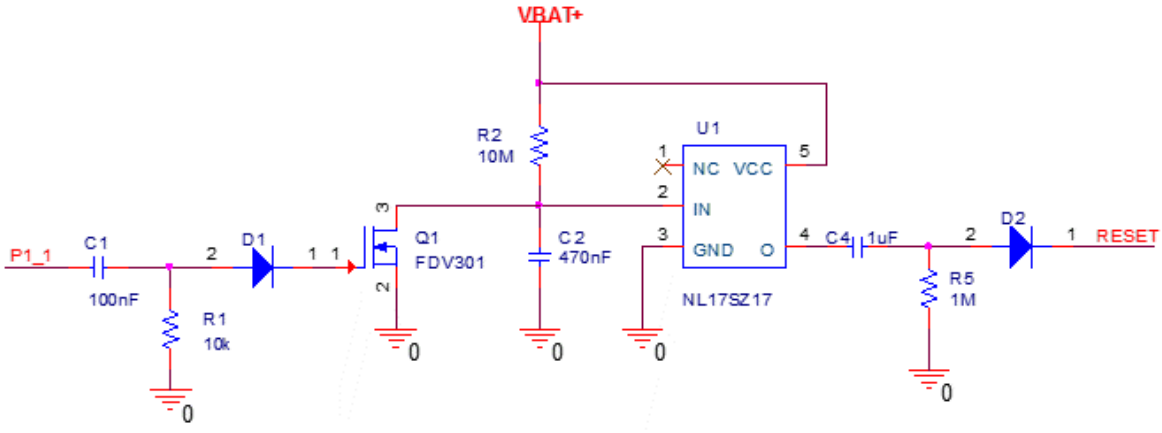

Figure 2: Watchdog Circuit Driven by a GPIO

Table 2: List of Components for the Watchdog in Figure 2

C1	R1	D1	Q1	R2	C2	U1	C4	R5	D2
100 nF	10 KOhm	Diode 1N4148W	FDV301	10 MOhm	470 nF	OnSemi NL17SZ17	1 µF	1 MOhm	Diode 1N4148W

4.3 Alternative solution: using the internal RCX oscillator

Alternatively, using the internal RCX oscillator instead of the XTAL32K oscillator eliminates all problems with the XTAL32K pins and stopping of the XTAL32K oscillator when executing ESD testing or when subjected to electro static discharges when touched by the user. The internal RCX oscillator has been qualified to be used as Bluetooth® low power clock.

There is a disadvantage when using the RCX oscillator as low power clock and for a real time clock (RTC) function. Although the RCX is calibrated at every wake-up against the 16 MHz crystal oscillator, it will not be as accurate as the 32.768 KHz crystal. The expected drift in absolute time is about 10 seconds per 24 hours when using the RTC function based on the internal RCX oscillator.

5 Required Software

5.1 Software modifications for External Watchdog circuit using LED1

Below, the software that was added to the SDK PXP_REPORTER software using the Breath Timer for generating the PWM signal from the LED1 port driving the external watchdog circuit is listed.

It was added to Pxp_Reporter -> Main.c -> Static void periph_init(void).

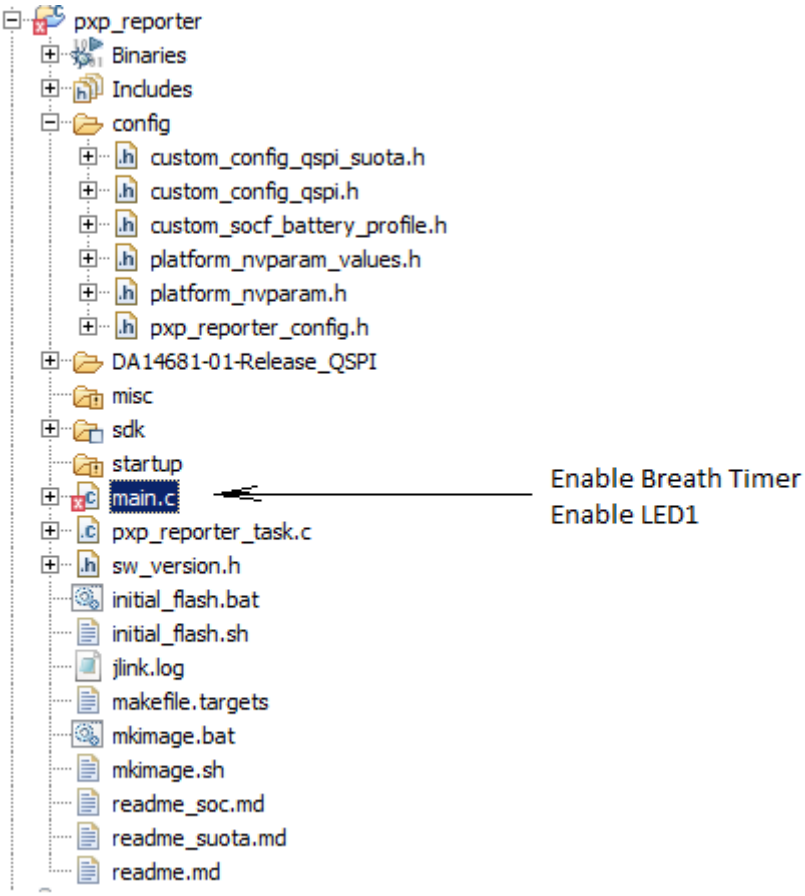

Figure 3: Modifications required for enabling LED1

Software modification in **main.c** is listed below:

```
/*****************************************************************************
*                                                                            *
* @file main.c                                                               *
*                                                                            *
* @brief Proximity Reporter                                                  *
*                                                                            *
*****************************************************************************

#include "hw_breath.h"     //Added for External WatchDog Circuit using LED1

static void periph_init(void)
{

breath_config cfg = {
     .dc_min = 127,
     .dc_max = 128,
     .dc_step = 255,
     .freq_div = 4,
     .polarity = HW_BREATH_PWM_POL_POS,};

     hw_breath_init(&cfg);

REG_SETF(GPREG, LED_CONTROL_REG, LED1_SRC_SEL, HW_LED_SRC2_BREATH); //set source
REG_SETF(GPREG, LED_CONTROL_REG, LED1_EN, 1); //Enable led1

     hw_breath_enable();
```

5.2 Software modifications for External Watchdog circuit using GPIO, P1_1

Below, the software that was added to the SDK PXP_REPORTER software using the Timer0 for generating the PWM signal from the GPIO, P1_1 port driving the external watchdog circuit is listed:

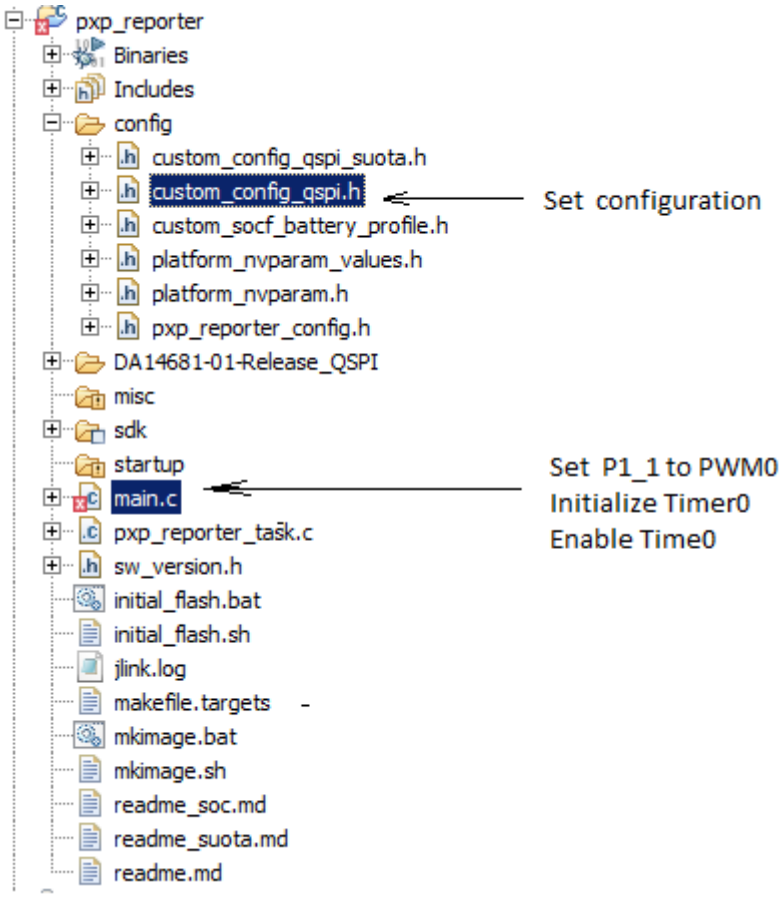

Figure 4: Modifications required for enabling GPIO P1_1

Software modification on **custom_config_qspi.h** below:

```
/**
****************************************************************************************
*                                                                                      *
* @file custom_config_qspi.h                                                           *
*                                                                                      *
****************************************************************************************
*/


#define dg_configUSE_HW_TIMER0   (1)     //External WatchDog Circuit using  GPIO P1_1
```


Software modification on **main.c** below:

```
*****************************************************************************
*                                                                           *
* @file main.c                                                              *
*                                                                           *
* @brief Proximity Reporter                                                 *
*                                                                           *
*****************************************************************************



#include "hw_timer0.h"   //Add for External WatchDog Circuit with GPIO P1_1


static void periph_init(void)

{

// Alternative External WatchDog Circuit using  GPIO P1_1


// Make sure to set configuration flag on "custom_config_qspi.h" as below:
// #define dg_configUSE_HW_TIMER0   (1)


REG_SET_BIT(CRG_PER, USBPAD_REG, USBPAD_EN);
                  //force on the power to the USB pads -not on by default


hw_gpio_set_pin_function(HW_GPIO_PORT_1, HW_GPIO_PIN_1,
HW_GPIO_MODE_OUTPUT_PUSH_PULL, HW_GPIO_FUNC_PWM0);

// Initialize Timer0 /

const timer0_config pwm_config = { HW_TIMER0_CLK_SRC_SLOW,
HW_TIMER0_FAST_CLK_DIV_1, HW_TIMER0_MODE_PWM, 4, 1, 1};

hw_timer0_init(&pwm_config);
hw_timer0_enable();
```

Revision History

Revision	Date	Description
1.1	22-Feb-2022	Updated logo, disclaimer, copyright.
1.0	24-May-2017	Initial version.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.