Application Report
SPNA233A–May 2016–Revised June 2016

Hercules PLL Advisory SSWF021#45 Workaround

Bob Crosby

ABSTRACT

This application report provides software source code and additional information on how to implement a workaround that helps to minimize the impact of Hercules Phase Locked Loop (PLL) Advisory SSWF021#45.

Project collateral and source code mentioned in this document can be downloaded from the following URL: http://www.ti.com/lit/zip/spna233.

Contents

List of Figures

List of Tables

1 Background

Texas Instruments has found that on rare occasions some Hercules Safety Microcontrollers have an issue with PLL startup. While Texas Instruments does test for and screen these devices, at the time of publication of this report, our screens are not 100% effective. Parts that are affected have advisory SSWF021#45 listed in the errata document for that device, if the errata document was published in May of 2016 or later. The software workaround described in this report, while not 100% effective, significantly helps to reduce the occurrence of failures.

2 Implementation

The header file “errata_SSWF021_45.h” contains the function prototypes and should be included in the user’s source file that calls the workaround function. The header file “errata_SSWF021_45_defs.h” defines values used in the “errata_SSWF021.c” file. It makes the source file independent of HALCoGen.

NOTE: The workaround functions do not set the PLL to the customer's desired frequency, nor do they leave the PLL enabled. These steps should be performed after successful completion of the workaround routine. The PLL settings in the workaround routine were chosen to minimize the lock time and to be valid over the range of 5 MHz to 20 MHz crystal frequency. Changing these settings may affect the proper execution of the workaround routine.

2.1 Which Function to Use

There are three functions provided in the source code: one for PLL1, a second for PLL2, and the third for locking PLL1 and PLL2 at the same time

2.1.1 _errata_SSWF021_45_pll1()

This function only attempts to lock PLL1. This function is to be used on devices that have only one PLL, or in applications that do not use PLL2.

2.1.2 _errata_SSWF021_45_pll2()

This function only attempts to lock PLL2, and is provided for completeness. Usually if PLL2 is used, both PLLs are used and the workaround function that locks both PLLs simultaneously is preferred because it reduces the overall execution time.

2.1.3 _errata_SSWF021_45_both_plls()

This function attempts to lock both PLLs simultaneously. This is the most efficient routine to use if both PLLs will be used in the application.

2.2 Where to Place the Function Call

The workaround function should be called only after a power-on reset. For example, in HALCoGen, the call to the workaround routine should be placed in the USER CODE section inside the check for the POWERON_RESET as shown in the example of Figure 1.

*sys_startup.c

```
136     /* check for power-on reset condition */
137     /*SAFETYMCUSW 139 S MR:13.7 <APPROVED> "Hardware status bit read check" */
138     if ((SYS_EXCEPTION & POWERON_RESET) != 0U)
139     {
140 /* USER CODE BEGIN (12) */
141         if(_errata_SSWF021_45_pll1(5) != 0 )
142         {
143             routine_to_put_into_safe_fail_state();
144         }
145 /* USER CODE END */
146
147         /* clear all reset status flags */
148         SYS_EXCEPTION = 0xFFFFU;
149
150 /* USER CODE BEGIN (13) */
151 /* USER CODE END */
```

Figure 1. Example Call to Workaround Routine

2.3 Parameters and Return Value

2.3.1 Input Parameter

There is only one input parameter, an unsigned integer "count". This parameter determines the maximum number of PLL lock attempts to try before exiting with an error. If a count of zero is given, the routine continues to attempt a proper PLL lock until successful. The value chosen for count determines the maximum execution time of the workaround function.

2.3.2 Return Value

The workaround functions return an unsigned integer that indicates the pass or fail status of the function. The possible return values are:

- 0 = Success (the PLL or both PLLs have successfully locked and then been disabled)
- 1 = PLL1 failed to successfully lock in "count" tries
- 2 = PLL2 failed to successfully lock in "count" tries
- 3 = Neither PLL1 nor PLL2 successfully locked in "count" tries
- 4 = The workaround function was not able to disable at least one of the PLLs. The most likely reason is that a PLL is already being used as a clock source. This can be caused by the workaround function being called from the wrong place in the code.

2.4 Execution Time

The execution time is a function of the crystal frequency and the number of iterations required for the PLL to lock. The maximum execution time is then a function of the maximum iterations allowed, which is the only parameter passed to the function. The *_errata_SSWF021_45_both_plls()* function minimizes execution time by locking both PLLs simultaneously. Example execution times for the single PLL *_errata_SSWF021_45_pll1()* and *_errata_SSWF021_45_pll2()* functions are shown in Table 1. Example execution times for the dual PLL *_errata_SSWF021_45_both_plls()* function is shown in Table 2.

Table 1. Single PLL Maximum Execution Time

Maximum Tries	5 MHz Crystal	8 MHz Crystal	16 MHz Crystal	20 MHz Crystal
1	505 µs	316 µs	158 µs	126 µs
2	1010 µs	632 µs	316 µs	253 µs
3	1515 µs	948 µs	474 µs	379 µs
4	2020 µs	1264 µs	632 µs	505 µs
5	2525 µs	1580 µs	790 µs	632 µs

Table 2. Dual PLL Maximum Execution Time

Maximum Tries	5 MHz Crystal	8 MHz Crystal	16 MHz Crystal	20 MHz Crystal
1	596 µs	372 µs	186 µs	149 µs
2	1191 µs	744 µs	372 µs	298 µs
3	1786 µs	1116 µs	558 µs	447 µs
4	2381 µs	1488 µs	744 µs	596 µs
5	2976 µs	1860 µs	930 µs	744 µs

3 Detailed Description

The purpose of the workaround routines is to get the PLL to successfully lock one time after power-on reset. The workaround routine does not initialize the PLL to the desired frequency nor does it leave the PLL enabled. After the PLL has locked once, it successfully locks each time using the desired settings until the next loss of power.

The workaround routine polls for either the clock source to be valid (PLL lock) or for a PLL slip, as indicated in the ESM register. In the case of the function that locks both PLLs simultaneously, it checks for a lock or slip in each PLL. Then, if the PLL indicates that it has locked, the frequency of the PLL is checked using the Dual Clock Compare (DCC) module. Since the DCC measures the ratio of the PLL frequency to the oscillator frequency, the routine can be used with any valid oscillator frequency without modification.

The following function is the workaround for both PLLs.

```
uint32 _errata_SSWF021_45_both_plls( uint32 count)
{
    uint32 failCode,retries,clkCntlSav;

    /* save CLKCNTL, */
    clkCntlSav = systemREG1->CLKCNTL;
    /* First set VCLK2 = HCLK */
    systemREG1->CLKCNTL = clkCntlSav & 0x000F0100U;
    /* Now set VCLK = HCLK and enable peripherals */
    systemREG1->CLKCNTL = SYS_CLKCNTRL_PENA;
    for(retries = 0U; (retries < count) || (count == 0U); retries++)
    {
        failCode = 0U;
        /* Disable PLL1 and PLL2 */
        if(failCode = disable_plls(SYS_CLKSRC_PLL1 | SYS_CLKSRC_PLL2) != 0U)
        {
            break;
        }
        /* Clear Global Status Register */
        systemREG1->GBLSTAT = 0x00000301U;
        /* Clear the ESM PLL slip flags */
        esmREG->SR1[0] = ESM_SR1_PLL1SLIP;
        esmREG->SR4[0] = ESM_SR4_PLL2SLIP;
        /* set both PLLs to OSCIN/1*27/(2*1) */
        systemREG1->PLLCTL1 = 0x20001A00U;
        systemREG1->PLLCTL2 = 0x3FC0723DU;
        systemREG2->PLLCTL3 = 0x20001A00U;
        systemREG1->CSDISCLR = SYS_CLKSRC_PLL1 | SYS_CLKSRC_PLL2;
        /* Check for (PLL1 valid or PLL1 slip) and (PLL2 valid or PLL2 slip) */
        while  ((((systemREG1->CSVSTAT & SYS_CLKSRC_PLL1) == 0U) &&
                 ((esmREG->SR1[0U] & ESM_SR1_PLL1SLIP) == 0U)) ||
                (((systemREG1->CSVSTAT & SYS_CLKSRC_PLL2) == 0U) &&
                 ((esmREG->SR4[0U] & ESM_SR4_PLL2SLIP) == 0U)))
        {
          /* Wait */
        }
        /* If PLL1 valid, check the frequency */
        if((esmREG->SR1[0] & ESM_SR1_PLL1SLIP) != 0U)
        {
          failCode |= 1U;
        }
        else
        {
            failCode |= check_frequency(dcc1CNT1_CLKSRC_PLL1);
        }
        /* If PLL2 valid, check the frequency */
        if((esmREG->SR4[0] & ESM_SR4_PLL2SLIP) != 0U)
        {
          failCode |= 2U;
        }
        else
        {
            failCode |= (check_frequency(dcc1CNT1_CLKSRC_PLL2) << 1U);
        }
        if (failCode == 0U)
        {
            break;
        }
    }
    disable_plls(SYS_CLKSRC_PLL1 | SYS_CLKSRC_PLL2);
```

```
    /* restore CLKCNTL, VCLKR and PENA first */
    systemREG1->CLKCNTL = (clkCntlSav & 0x000F0100U);
    /* restore CLKCNTL, VCLK2R */
    systemREG1->CLKCNTL = clkCntlSav;
    return failCode;
}
```

The *check_frequency* function is used to measure the PLL frequency before the PLL is used as a clock source.

```
static uint32 check_frequency(uint32 cnt1_clksrc)
{
    /* Setup DCC1 */
     /* DCC1 Global Control register configuration */
    dccREG1->GCTRL      = (uint32)0x5U |       /** Disable  DCC1 */
                          (uint32)((uint32)0x5U << 4U) | /** No Error Interrupt */
                          (uint32)((uint32)0xAU << 8U) | /** Single Shot mode */
                          (uint32)((uint32)0x5U << 12U); /** No Done Interrupt */
    /* Clear ERR and DONE bits */
    dccREG1->STAT = 3U;
    /** DCC1 Clock0 Counter Seed value configuration */
    dccREG1->CNT0SEED   = 68U;
    /** DCC1 Clock0 Valid Counter Seed value configuration */
    dccREG1->VALID0SEED = 4U;
    /** DCC1 Clock1 Counter Seed value configuration */
    dccREG1->CNT1SEED   = 972U;
    /** DCC1 Clock1 Source 1 Select */
    dccREG1->CNT1CLKSRC = (uint32)((uint32)10U << 12U) | /** DCC Enable/Disable Key */
                          (uint32) cnt1_clksrc;  /** DCC1 Clock Source 1 */

    dccREG1->CNT0CLKSRC = (uint32)DCC1_CNT0_OSCIN;  /** DCC1 Clock Source 0 */

    /** DCC1 Global Control register configuration */
    dccREG1->GCTRL      = (uint32)0xAU |       /** Enable  DCC1 */
                          (uint32)((uint32)0x5U << 4U) | /** No Error Interrupt */
                          (uint32)((uint32)0xAU << 8U) | /** Single Shot mode */
                          (uint32)((uint32)0x5U << 12U); /** No Done Interrupt */
    while(dccREG1->STAT == 0U)
    {
        /* Wait */
    }
    return (dccREG1->STAT & 0x01U);
}
```

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from May 20, 2016 to June 3, 2016 (from Initial Revision (May 2016) to A Revision)	Page
• Added a note to emphasize that the workaround functions do not initialize the customer settings of the PLLs.	2
• Corrected method of saving and restoring CLKCNTL register	5

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265