Application Report
SLUA807–January 2017

Understanding CPU Lock-Up Mechanisms in UCD31xx Devices

Koushik Vaithyanathan

ABSTRACT

The UCD31xx family of devices from Texas Instruments is a highly integrated digital controller for isolated power applications[1]. Typical applications are in power supplies, power-factor correction, isolated DC-DC modules, and so on. These devices include a 32-bit ARM7TDMI-S™ RISC CPU core, multiple memory banks, three independent PID-based hardware feedback loops, communications ports (PMBus, I^2C, SPI, UART, and JTAG), data converters, general purpose-I/O, and other peripherals.

Users compile and download application-specific firmware in the program flash banks and develop solutions for a wide variety of power applications including phase shifted full bridge, resonant LLC converters, and so on. Firmware development involves considerable engineering effort, taking into account the device-specific hardware features and memory map. Embedded code development without detailed understanding of hardware specifics is susceptible to errors and results in unexpected behavior. One such unexpected behavior is CPU lockup, discussed in detail in this document.

Contents

List of Figures

1 CPU Lockup

Lockup is broadly defined as the symptom of a function or task using the CPU and not releasing it for a period of time. The lockup behavior is more often caused by an application use case and occurs during firmware code development, engineering evaluation, or at production programming. When the CPU is locked up, it does not perform any power supply operations and also does not respond to any read/write commands on its communication ports. A CPU lockup can occur when the device is either in ROM or in flash mode. The following are the most common mechanisms through which a CPU lockup can occur.

- Device in ROM mode
  - Invalid trim flash checksum
  - Invalid program flash checksum
- Device in flash mode
  - Incorrect image
  - No response on communication ports
    - Executing a long thread
    - Waiting on a hardware event
  - Stuck in an infinite loop
    - Stuck in an interrupt service routine (ISR)
    - Incorrect handler routine
  - Stuck in a reset loop
    - Invalid address
    - Illegal access
    - Timer watchdog
  - CPU stalled

2 Device in ROM Mode

On device power up or reset, all UCD31xx devices execute the TI boot ROM code. The ROM code copies the trim values from the trim flash to the analog control registers, validates program flash checksum, and so on. The initial start-up execution of the user firmware code in program flash does not happen if there is an invalid trim flash checksum or invalid program flash checksum.

2.1 Invalid Trim Flash Checksum

The trim flash checksum is programmed in each device at Texas Instruments ATE before the device is shipped, and the user should not write to or read from this trim flash. The UCD31xx devices depend on a valid trim flash checksum to execute the code in program flash.

2.2 Invalid Program Flash Checksum

On every device power up or release from any hardware or software reset, the TI boot ROM calculates the program flash checksum and verifies if it has the correct checksum value.

The most common reason for an invalid program flash checksum is an incorrect program download to the wrong starting address for the program flash. The starting address for the program flash differs across the UCD31xx devices and ensures that the correct content is programmed in the expected location. In addition, the TI boot ROM code also checks for a valid 0xEA opcode in the starting location of the program flash before execution.

The TI Fusion Digital Power Studio GUI can be used to verify the program flash checksum before executing the user firmware. The GUI supports device-specific PMBus commands to calculate the checksum for different code sizes and then verify if the calculated value matches the programmed value in the checksum location.

If the checksum is incorrect, the device stays in ROM mode. This would effectively prevent any code from being executed. Typically, the PMBus slave address in the firmware (example: 0x58) is different from the default PMBus slave address of the device in ROM mode (0xB). This would result in the device NACKing any PMBus commands to the programmed slave address in the firmware (0x58).

The GUI supports scanning for the device in ROM mode and in flash mode. The GUI scans for the ROM revision to identify the device when it is in ROM mode and scans for the manufacturer device ID when the device is in flash mode.

3 Device in Flash Mode

Whenever the boot ROM verifies the program flash has the correct checksum value, the UCD31xx device transitions to program flash code execution. CPU lockup during flash mode is the condition when there is no device communication or evidence of any program flash execution. The firmware does not meet the expected user behavior such as no response to any read/write commands on the communication pins, no power supply regulation, and so on. The reasons can be as simple as using an incorrect set of header and linker files or as complex as executing a long thread without servicing requests on communication ports. Typical cases in which there is no evidence of any program flash execution are discussed in the following paragraphs.

3.1 Incorrect Image

A valid checksum in the program flash does not necessarily mean there will be any evidence of proper device communication. For example, if the contents of the program flash are all zeroes in the UCD3138 device, the program flash checksum matches an expected value. This action results in the device executing the firmware in program flash, and also results in a permanently locked up device.

3.2 No Response on Communication Ports

In telecom and server power applications, multiple controllers perform different functions, such as AC-DC conversion, DC-DC regulation, and so on. Communication ports (for example, PMBus or UART) are responsible for sequencing start-up and also share critical data between the controllers. An unexpected response on the communication ports can be due to the CPU either executing a long thread or waiting on a hardware event. Whenever there is no response on the ports, a simple debug procedure is to reset the device using the nRESET pin and retry the command.

3.2.1 Executing a Long Thread

It is very common to have functions and task calls in a firmware routine that requires a long time to completely execute. For example, a firmware-based CPCC function involves significant computation and a large number of CPU clock cycles. If the firmware does not service any PMBus communication requests during the execution of the CPCC thread, clock stretching occurs on the SCL pin, resulting in a PMBus clock low time-out.

3.2.2 Waiting on a Hardware Event

In embedded firmware, it is a common practice to wait on an external hardware event to exit a loop. Examples of a hardware event follow:

- Waiting for a rising edge on an ADC_EXT_TRIG pin
- Expecting the voltage on the analog comparator inputs to cross a minimum threshold

If there is no underlying time-out feature in the loop, this external trigger method makes the device prone to CPU lockup.

3.3 Stuck in an Infinite Loop

During firmware execution, it is possible for the device to be stuck in an infinite loop and never exit it. Firmware loops occur when the CPU is stuck in an ISR or incomplete handler routines discounting received data.

3.3.1 Stuck in an ISR

In the UCD31xx devices, whenever an interrupt is triggered, the code in the ISR is executed. It is a common practice to have the trigger mechanism disabled at the end of the ISR. If the trigger mechanism is not cleared at the end of the ISR, the CPU constantly services the ISR over and over again. This would effectively result in a firmware infinite loop in the ISR.

3.3.2 Incorrect Handler Routine

Firmware handler routines for communication protocols can either be based on interrupts or polling. In the polling methodology, it is a frequent practice to use loops in the background. Such loops consist of multiple nested if-else conditional statements triggered by status flags. The flags are cleared only when the received buffer values match one of the supported commands.

Another common programming mistake can occur if the handler routine does not account for any received buffer value, where the status flag is not cleared. This mistake results in an infinite loop and the device does not respond to the commands. When the device is stuck in an infinite loop, a simple debug procedure is to reset the device using the nRESET pin and check if the infinite loop can be exited.

3.4 Stuck in a Reset Loop

If the CPU reads or writes to an invalid address, or tries an illegal access, or does not service the timer watchdog periodically, the following sequence results in a *reset loop* and the device does not perform the expected programming behavior.

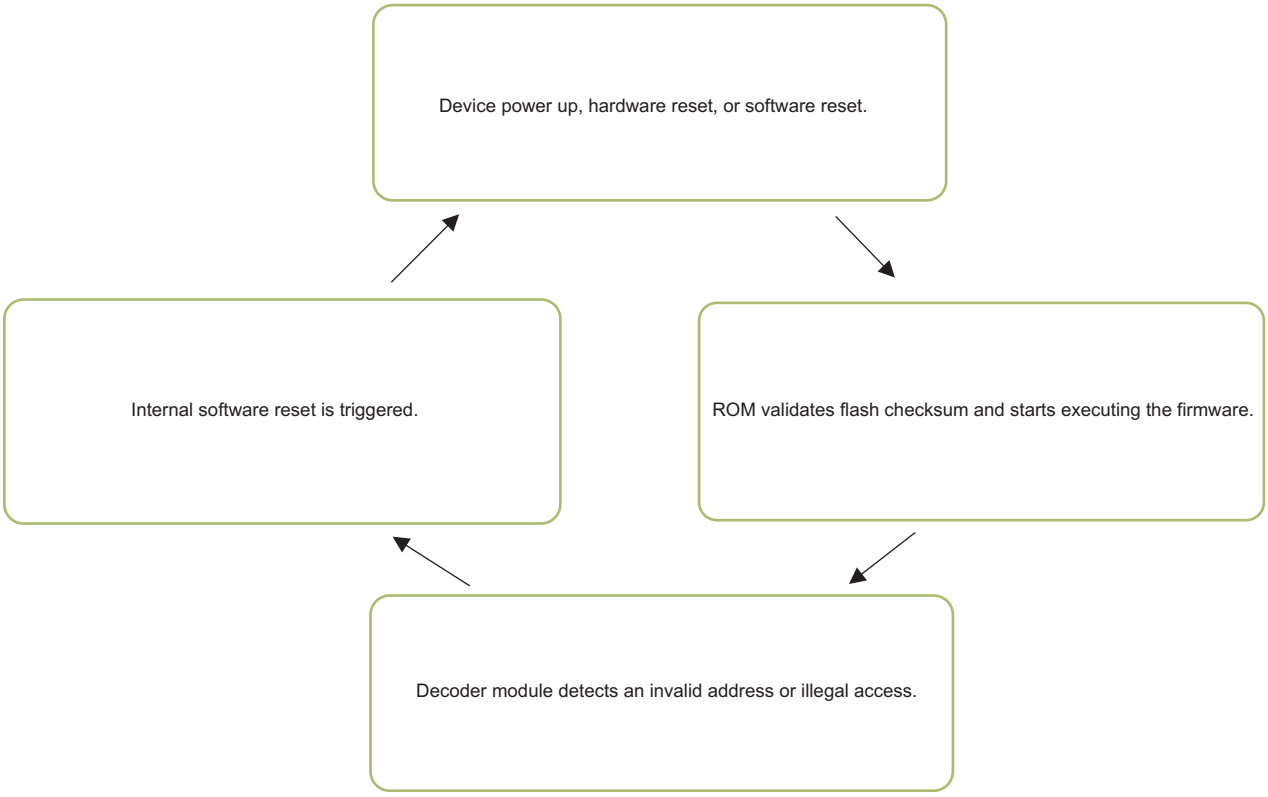

Figure 1. Reset Loop Mechanism

The current consumed by the device is different for ROM and flash modes. When the device is stuck in a reset loop, it alternates between the ROM and flash modes. The simplest debug procedure to determine if a device is stuck in a reset loop is to probe the current consumption on an oscilloscope.

3.4.1 Invalid Address

Each UCD31xx device has a predefined memory map with hardcoded address locations for flash, RAM, and peripheral registers. Each UCD31xx device has an address decoder block which constantly checks if the address being accessed for a read or write operation is within the defined address space. Any access outside this defined address space triggers an illegal address exception and causes an internal device reset.

Additional caution must be taken when using pointer-based reads and writes to locations outside the memory mapped variables. TI highly recommends following the guidelines defined in the programmer's manual. The ILLADR status bit in the SYSESR register can be used to detect an invalid address access.

3.4.2 Illegal Access

Registers in the decoder module, central interrupt module, and the system module must be accessed only in privilege mode, not in user mode. For example, to issue a software reset, any writes to SysRegs.SYSECR.bit.RESET must be done in privilege mode. When the device is in user mode any write to the SysRegs.SYSECR register causes an illegal access reset. The ILLACC status bit in the SYSESR register can be used to detect an invalid address access.

3.4.3 Timer Watchdog

The timer watchdog feature in the UCD31xx devices is used to ensure that the device is not stuck in any infinite firmware loop. The watchdog can be programmed to reset the device periodically if it is not serviced. If any CPU thread occupies the CPU for long periods of time and does not service the watchdog, the device is automatically reset. TI strongly recommends using this firmware protection technique.

3.5 CPU Stalled

The UCD31xx devices have an internal high frequency oscillator (HFO). If this free-running oscillator is shut down, then the CPU is stalled. The program counter will be stuck at its last value until the HFO runs again. A simple way to detect an HFO shut down is to measure the current consumed by the UCD31xx device. If overall current consumption is below 20 mA, there is a possibility that the HFO has been shut down by unintended firmware access.

3.6 Supplementary Root Cause Analysis

If the root cause still cannot be determined after ruling out the previous possibilities, gather the following details before submitting for failure analysis through the TI Quality Tracking System.

- Short description of application use case
- Detailed schematics and layout information on all the pins of the UCD31xx device
- Firmware image (x0 file) and the necessary build files
- Relevant master communication commands to observe device response
- Data from the previously discussed debug procedures

3.7 Summary

This document discussed in detail the potential causes of CPU lockup mechanisms. In addition, the existing prevention techniques and the detection methods are also listed. Whenever a CPU lockup mechanism is observed on UCD31xx devices, TI recommends executing a step-by-step debug for walking through the previously mentioned possibilities.

3.8 References

1. Texas Instruments, UCD3138A Highly-Integrated Digital Controller for Isolated Power, Data Sheet

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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