Clock Fail Detection for Stellaris® LM3SX00 MCUs

ABSTRACT

The Stellaris® on-chip clock failure detection circuitry allows a microcontroller to respond appropriately if the main crystal oscillator stops functioning. This capability can be extended to situations in which the crystal oscillator fails due to an open or short circuit.

Contents

1 Introduction

The clock fail detection logic on the Stellaris® LM3SX00 microcontrollers detects clocking failures of the main crystal oscillator circuit (MOSC) on the PCB and automatically switches to the internal crystal oscillator circuit (IOSC) as the clock source. In addition to situations in which the crystal oscillator stops operating correctly, the crystal oscillator could fail due to an open or short circuit. The clock fail detection logic is designed to detect failures of the first type. With a minor modification to the crystal clock circuit, the detection logic can also detect failures of the second type. This application note discusses the hardware modification required for the second type of failure as well as the code required to properly use the clock fail detection feature. Table 1 shows the devices for which this application note is applicable.

Table 1. Applicable Stellaris® Devices

Series	Devices
100 Series	LM3S101, LM3S102
300 Series	LM3S300, LM3S301, LM3S308, LM3S310, LM3S315, LM3S316, LM3S317, LM3S328
600 Series	LM3S600, LM3S601, LM3S608, LM3S610, LM3S611, LM3S612, LM3S613, LM3S615, LM3S617, LM3S618, LM3S628
800 Series	LM3S800, LM3S801, LM3S808, LM3S811, LM3S812, LM3S815, LM3S817, LM3S818, LM3S828

2 Hardware Modification

A 5-MΩ resistor must be added to the Stellaris® microcontroller's crystal oscillator (MOSC) circuit on the PCB. The resistor should be connected between the OSC0 pin and ground (see Figure 1). The resistor leads should be as short as possible to minimize any interference with the oscillator circuit operation.

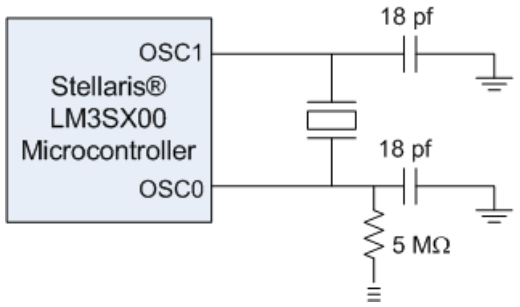

Figure 1. Adding a 5-MΩ Resistor to the Crystal Oscillator Circuit on the PCB

3 Software Support

There are two small sections of code required to properly use the clock fail detection feature. Including this code is straightforward using StellarisWare® DriverLib APIs. The following code assumes that the MCU clock source is configured to use the MOSC clock input source. At startup, in the main section of code, interrupts are first cleared and then enabled. Then the clock verification hardware is enabled. In the ISR, the code first checks for the MOSC failure. If a failure is detected, the system clock source is formally switched to the IOSC clock source. Then the interrupt MOSC_FAIL bit is cleared, and the clock verification function is disabled. At this point, the system software can take whatever action is appropriate to handle the failed MOSC crystal oscillator circuit. The following are example code segments:

```
//*****************************************************************************
// // Include the header files required for the MOSC fail setup/handling code.
// //*****************************************************************************
#include "inc/hw_ints.h"
#include "inc/hw_sysctl.h"
#include "inc/hw_types.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
//*****************************************************************************
//
// The following code enables the MOSC fail detection. It should be called
// during the initialization code for the application.
//
//*****************************************************************************
void
EnableMOSCFailDetect(void)
{
     //
     // Enable MOSC verification.
     //
     SysCtlMOSCVerificationSet(true);

     //
     // Clear any previously generated MOSC failure interrupt.
     //
     SysCtlIntClear(SYSCTL_INT_MOSC_FAIL);

     //
     // Enable the MOSC failure interrupt.
     //
```

```
        SysCtlIntEnable(SYSCTL_INT_MOSC_FAIL);
        IntEnable(INT_SYSCTL);

        //
        // Enable the interrupt controller.
        //
        IntMasterEnable();
    }
    //*****************************************************************************
    //
    // The interrupt handler for the SysCtl interrupt. This handles the MOSC fail
    // detect interrupt that is generated when a failure on MOSC is detected.
    //
    // In order for the MOSC fail detect feature to work properly, this function
    // (or a function with the equivalent functionality) must be installed as the
    // SysCtl interrupt handler in the vector table. The method for achieving this
    // varies from toolchain to toolchain.
    //
    //*****************************************************************************
    void
    SysCtlInterruptHandler(void)
    {
        //
        // See if a MOSC fail detection is the cause of this interrupt.
        //
        if((SysCtlIntStatus(true) & SYSCTL_INT_MOSC_FAIL)==SYSCTL_INT_MOSC_FAIL)
        {
            //
            // Disable and clear the MOSC fail detect interrupt.
            //
            SysCtlIntDisable(SYSCTL_INT_MOSC_FAIL);
            SysCtlClkVerificationClear();
            SysCtlIntClear(SYSCTL_INT_MOSC_FAIL);

            //
            // Switch to using IOSC, disable the main oscillator,and disable MOSC
            // fail detection. The switch to IOSC has already occurred (in order
            // to allow the processor to continue execution), but this formalizes
            // the change.
            //
            HWREG(SYSCTL_RCC) = SYSCTL_RCC_OSCSRC_INT | SYSCTL_RCC_MOSCDIS;

            //
            // Perform whatever application-specific actions that are necessary to
            // respond to the MOSC fail detection.
            //
        }
    }
```

4 Conclusion

The clock fail detection logic provides a mechanism for a system to deal with a crystal oscillator failure. By adding an additional resistor, this logic can also address a failure due to an open or short circuit in the crystal oscillator circuit. This application note shows the necessary circuit modification to add this increased

functionality. In addition, the code required to properly use the clock fail detection feature is included using StellarisWare® Driver Library APIs. System designers may choose to address a clock failure in varying ways, so this application note does not discuss methods for recovering from a crystal oscillator failure.

5 References

The following are available for download at www.ti.com/stellaris:

- Stellaris microcontroller data sheet, Publication Number DS-LM3S8nn (where 8nn is the part number for that specific Stellaris family device)
- *Stellaris® Peripheral Driver Library User's Guide*, Document order number SW-DRL-UG
- Stellaris Peripheral Driver Library, Order number SW-DRL

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

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