Stellaris® LM3S6G11 RevA2 Errata

This document contains known errata at the time of publication for the Stellaris LM3S6G11 microcontroller. The table below summarizes the errata and lists the affected revisions. See the data sheet for more details.

See also the ARM® Cortex™-M3 errata, ARM publication number PR326-PRDC-009450 v2.0.

Table 1. Revision History

Date	Revision	Description
October 2012	2.2	■ Clarified issue “Debug interface is reset by any type of reset” on page 4. ■ Added issue “JTAG state machine may advance after certain resets” on page 6. ■ Added issue “Non-word-aligned write to SRAM can cause incorrect value to be loaded” on page 8. ■ Added issue “Internal reset supervisors may not prevent incorrect device operation during power transitions” on page 10. ■ Added issue “Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module” on page 22. ■ Added issue “Watchdog Timer 1 module asserts reset signal even if not programmed to reset” on page 22. ■ Added issue “WDTLOAD yields an incorrect value when read back” on page 23. ■ Added issue ???. ■ Added issue ???. ■ Added issue ???. ■ Added issue ???. ■ Added issue “When UART LIN or SIR mode is enabled, µDMA burst transfer does not occur” on page 24. ■ Added issue “UART transfers fail at certain system clock frequency and baud rate combinations” on page 24. ■ Added issue “Freescale SPI Mode at low SSIClk frequencies can yield data corruption” on page 25. ■ Added issue ???.
June 2012	2.1	■ Clarified how to read the date code on Stellaris devices.
March 2011	2.0	■ Removed issue "The Reset Cause register always reports POR regardless of reset type" as it does not affect this device. ■ Added issue “The ROM_FlashProgram() function may not correctly program the Flash memory above 50 MHz” on page 15.
September 2011	1.9	■ Started tracking revision history. ■ Added issue “Boundary scan is not functional” on page 3. ■ Added issue "The Reset Cause register always reports POR regardless of reset type". ■ Added issue “LIN mode Sync Break does not have the correct length” on page 24.

Table 2. List of Errata

Erratum Number	Erratum Title	Module Affected	Revision(s) Affected
1.1	Boundary scan is not functional	JTAG	A2
2.1	Debug interface is reset by any type of reset	System Control	A2
2.2	JTAG state machine may advance after certain resets	System Control	A2
2.3	Non-word-aligned write to SRAM can cause incorrect value to be loaded	System Control	A2
2.4	Internal reset supervisors may not prevent incorrect device operation during power transitions	System Control	A2
3.1	VDD3ON mode may not be used	Hibernation Module	A2
3.2	The WRC bit in the Hibernation Control register is R/W	Hibernation Module	A2
3.3	Writes to Hibernation module registers may change the value of the RTC	Hibernation Module	A2
3.4	Hibernation Module 4.194304-MHz oscillator supports a limited range of crystal load capacitance values	Hibernation Module	A2
4.1	The ROM_FlashProgram() function may not correctly program the Flash memory above 50 MHz	ROM	A2
5.1	Deep-Sleep mode must not be used	Flash Memory	A2
5.2	Mass erase must not be used if Flash protection bits are used	Flash Memory	A2
5.3	Page erase or program must not be performed on a protected Flash page	Flash Memory	A2
5.4	Flash memory endurance cycle specification is 100 cycles	Flash Memory	A2
6.1	The µDMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules	µDMA	A2
6.2	The µDMA does not generate a completion interrupt when transferring to and from GPTM 2A and 2B	µDMA	A2
7.1	PB1 has permanent internal pull-up resistance	GPIO	A2
8.1	The General-Purpose Timer match register does not function correctly in 32-bit mode	General-Purpose Timers	A2
8.2	A spurious DMA request is generated when the timer rolls over in Input-Edge Time mode	General-Purpose Timers	A2
8.3	A spurious DMA request is generated when the timer rolls over the 16-bit boundary	General-Purpose Timers	A2
8.4	The value of the prescaler register is not readable in Edge-Count mode	General-Purpose Timers	A2
8.5	ADC trigger and Wait-on-Trigger may assert when the timer is disabled	General-Purpose Timers	A2
8.6	Wait-on-Trigger does not assert unless the TnOTE bit is set	General-Purpose Timers	A2
8.7	Do not enable match and timeout interrupts in 16-bit PWM mode	General-Purpose Timers	A2
8.8	Do not use µDMA with 16-bit PWM mode	General-Purpose Timers	A2

Erratum Number	Erratum Title	Module Affected	Revision(s) Affected
8.9	Writing the GPTMTnV register does not change the timer value when counting up	General-Purpose Timers	A2
8.10	The prescaler does not work correctly when counting up in periodic or one-shot mode	General-Purpose Timers	A2
8.11	Snapshot must be enabled in both Timer A and B when in 32-bit snapshot mode	General-Purpose Timers	A2
9.1	Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module	Watchdog Timers	A2
9.2	Watchdog Timer 1 module asserts reset signal even if not programmed to reset	Watchdog Timers	A2
9.3	WDTLOAD yields an incorrect value when read back	Watchdog Timers	A2
10.1	The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled	UART	A2
10.2	LIN mode Sync Break does not have the correct length	UART	A2
10.3	When UART LIN or SIR mode is enabled, µDMA burst transfer does not occur	UART	A2
10.4	UART transfers fail at certain system clock frequency and baud rate combinations	UART	A2
11.1	Freescale SPI Mode at low SSIClk frequencies can yield data corruption	SSI	A2
12.1	Encoding error in the Ethernet MAC LED Encoding (MACLED) register	Ethernet Controller	A2

1 JTAG

1.1 Boundary scan is not functional

Description:

The boundary scan is not functional on this device.

Workaround:

None.

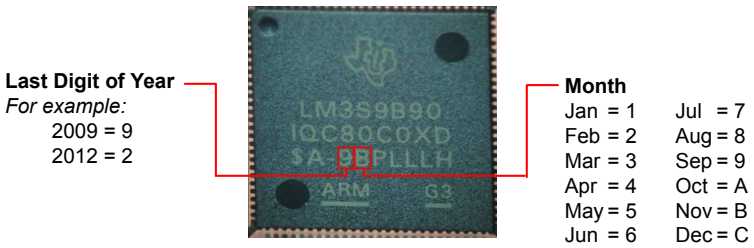
Silicon Revision Affected:

A2

Fixed:

Fixed on devices with date codes of 1A (October, 2011) or later.

Note: To determine the date code of your part, look at the first two characters following the dash on the third line of the part markings (highlighted in red in the following figure). The first number after the dash indicates the last decimal digit of the year. The second character indicates the month. Therefore, the following example shows a date code of 9B which indicates November 2009.



2 System Control

2.1 Debug interface is reset by any type of reset

Description:

The Serial Wire JTAG Debug Port (SWJ-DP) is reset by any reset condition. Therefore, any access to a debugger is lost, including breakpoints, watchpoints, vector catch, and trace. These reset types include:

- Watchdog reset
- Brown-out reset
- Software reset
- Reset pin assertion
- Main oscillator fail

Normal operation of the device is not affected by the reset of the SWJ-DP, however, users should bear this functionality in mind during development and debugging of applications. If a debugger does a SYSRESREQ, or if the debugger is being used in a session and a system reset occurs due to one of the reset sources above, then the debugger loses its state, including breakpoints, watchpoints, vector catch, and trace. Most debuggers attempt a recovery, usually after reporting the error to the user. If the debugger is able to recover control, the state of the application at that time reflects that the code has been running from reset and has not stopped on any breakpoints. If the application has breakpoint instructions physically in the code, such as for system calls that run through the debugger, then the code will have entered the fault handlers.

Workaround:

Because some ARM debuggers expect to maintain connectivity when a system reset is requested, the SYSRESREQ bit in the **Application Interrupt and Reset Control (APINT)** register should not be used when using these debuggers; instead the VECTRESET bit, which only resets the core, should be used so that debug connectivity is uninterrupted. VECTRESET does not reset on-chip peripherals, which must be reset with specific reset operations.

When debugging code that requires a software reset, the SYSRESREQ software reset mechanism in the NVIC (which is used by the Stellaris Peripheral Driver Library SysctlReset() and ROM_SysCtlReset() APIs) should not be used; instead, use the sequence of register writes with a VECTRESET in the NVIC as shown in the code below.

In addition, the ROM is mapped into address 0x0 during reset. The ROM code determines if boot loading is needed, and if not, transfers control to the normal application in Flash memory. As a result, the ROM is visible to the debugger on the reset entry. Debugging can be affected during Flash memory verification because the debugger compares the expected image with the ROM

contents and not the Flash memory as intended. The disassembly shown to the user is also affected. To avoid these issues, debuggers must switch off the ROM mapping. However, if the debugger in use does not switch off the ROM, the user can either step through the first assembly instructions until the ROM gets remapped or write a 1 to the BA bit in the **ROM Control (ROMCTL)** register at location 0x400F.E0F0 using the debugger GUI, debugger command line, or debugger startup script.

Use of any reset source listed above other than software reset causes the debugger to lose connectivity.

```
//
// Disable processor interrupts.
//
IntMasterDisable();

//
// Disable the PLL and the system clock divider (this is a NOP if they are
// already disabled).
//
HWREG(SYSCTL_RCC) = ((HWREG(SYSCTL_RCC) & ~(SYSCTL_RCC_USESYSDIV)) |
                     SYSCTL_RCC_BYPASS);
HWREG(SYSCTL_RCC2) |= SYSCTL_RCC2_BYPASS2;

//
// Now, write RCC and RCC2 to their reset values.
//
HWREG(SYSCTL_RCC) = 0x078e3ad0 | (HWREG(SYSCTL_RCC) & SYSCTL_RCC_MOSCDIS);
HWREG(SYSCTL_RCC2) = 0x07806810;
HWREG(SYSCTL_RCC) = 0x078e3ad1;

//
// Reset the deep sleep clock configuration register.
//
HWREG(SYSCTL_DSLPCLKCFG) = 0x07800000;

//
// Reset the clock gating registers.
//
HWREG(SYSCTL_RCGC0) = 0x00000040;
HWREG(SYSCTL_RCGC1) = 0;
HWREG(SYSCTL_RCGC2) = 0;
HWREG(SYSCTL_SCGC0) = 0x00000040;
HWREG(SYSCTL_SCGC1) = 0;
HWREG(SYSCTL_SCGC2) = 0;
HWREG(SYSCTL_DCGC0) = 0x00000040;
HWREG(SYSCTL_DCGC1) = 0;
HWREG(SYSCTL_DCGC2) = 0;

//
// Reset the remaining SysCtl registers.
//
HWREG(SYSCTL_PBORCTL) = 0;
HWREG(SYSCTL_IMC) = 0;
HWREG(SYSCTL_GPIOHBCTL) = 0;
HWREG(SYSCTL_MOSCCTL) = 0;
HWREG(SYSCTL_PIOSCCAL) = 0;
```

```
HWREG(SYSCTL_I2SMCLKCFG) = 0;

//
// Reset the peripherals.
//
HWREG(SYSCTL_SRCR0) = 0xffffffff;
HWREG(SYSCTL_SRCR1) = 0xffffffff;
HWREG(SYSCTL_SRCR2) = 0xffffffff;
HWREG(SYSCTL_SRCR0) = 0;
HWREG(SYSCTL_SRCR1) = 0;
HWREG(SYSCTL_SRCR2) = 0;

//
// Clear any pending SysCtl interrupts.
//
HWREG(SYSCTL_MISC) = 0xffffffff;

//
// Wait for any pending flash operations to complete.
//
while((HWREG(FLASH_FMC) & 0xffff) != 0)
{
}
while((HWREG(FLASH_FMC2) & 0xffff) != 0)
{
}

//
// Reset the flash controller registers.
//
HWREG(FLASH_FMA) = 0;
HWREG(FLASH_FCIM) = 0;
HWREG(FLASH_FCMISC) = 0xffffffff;
HWREG(FLASH_FWBVAL) = 0;

//
// Issue the core reset.
//
HWREG(NVIC_APINT) = NVIC_APINT_VECTKEY | NVIC_APINT_VECT_RESET;
```

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

2.2 JTAG state machine may advance after certain resets

Description:

Due to an issue with the reset logic, the JTAG state machine may advance to a random state if one of the following resets occurs:

- Hardware Reset via the $\overline{\text{RST}}$ pin

- Brown-Out Reset
- Software System Request Reset (using SYSRESREQ)
- Watchdog Reset
- MOSC Failure Reset

On most devices, these state transitions do not cause any noticeable problems. Some devices, however, eventually execute random JTAG instructions after multiple resets of the type listed above. Since some JTAG instructions can interfere with device operation, steps must be taken to avoid this behavior in a production environment.

In a development environment where a JTAG debugger is being used, this issue is likely to go unnoticed. Many JTAG debuggers pull TCK Low after establishing a connection with the device, which prevents the random state transitions from happening. Also, many JTAG debuggers reset the microcontroller using VECTRESET instead of SYSRESREQ, which also avoids the problem.

Workaround:

There are two workarounds, each with trade-offs:

- Connect TCK to GND through a 10-K resistor in the final board design. This avoids the problem completely but it causes the chip to draw a small amount of additional current, since the default pin configuration of TCK includes a weak internal pull-up resistor.
- Implement a software routine to explicitly reset the JTAG state machine after every system reset. This works for most cases, but it does not protect the ROM boot loader from erroneous JTAG instruction execution.

```
void
ResetJTAGState(void)
{
    volatile unsigned char ucToggleCount, ucDelayCount;

    //
    // Enable GPIO port C
    //
    HWREG(SYSCTL_RCGC2) = SYSCTL_RCGC2_GPIOC;

    //
    // Dummy read to make sure GPIO port C has time to enable before we
    // proceed.
    //
    ucToggleCount = HWREG(SYSCTL_RCGC2);

    //
    // Unlock the GPIOs on port C
    //
    HWREG(GPIO_PORTC_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY_DD;
    HWREG(GPIO_PORTC_BASE + GPIO_O_CR) = 0x01;

    //
    // "Toggle" TCK at least 5 times while TMS remains high. We're relying on a
    // digitally disabled pin to feed a "zero" into the JTAG module, and we're
    // also relying on an external pull-up to feed us our "one". To be extra
```

```
        // conservative, let's try 10 toggles.
        //
        for(ucToggleCount = 0; ucToggleCount < 10; ucToggleCount++)
        {
            //
            // Turn off the digital enable for PC0
            //
            HWREG(GPIO_PORTC_BASE + GPIO_O_DEN) &= ~(0x01);

            //
            // Delay a little to make sure the signal propagates through the JTAG
            // state machine (make sure these delays do not get optimized out by
            // the compiler).
            //
            for(ucDelayCount=0; ucDelayCount<100; ucDelayCount++)
            {
            }

            //
            // Turn on the digital enable for PC0
            //
            HWREG(GPIO_PORTC_BASE + GPIO_O_DEN) |= 0x01;

            //
            // Delay a little
            //
            for(ucDelayCount=0; ucDelayCount<100; ucDelayCount++)
            {
            }
        }
    }
```

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

2.3 Non-word-aligned write to SRAM can cause incorrect value to be loaded

Description:

If a word-aligned value is loaded from an SRAM location into a core register, then altered by storing a byte or halfword at an unaligned offset, the altered word-aligned value is not correctly indicated when loaded into a core register. The loaded value from the SRAM location into a core register reflects the original value, not the modified value.

The following assembly sequence causes the altered value loaded into a core register to not load the correct value, even though the correct value is visible in the SRAM memory location.

```
//
// Load a word-aligned value from an SRAM location into a
```

```
// core register (such as R0)
//
LDR        R0, [SP, #+0];

//
// Store byte or halfword from the core register to
// the SRAM location at a non-word-aligned offset
//
STRB     R0, [SP, #+1];
      OR
STRB     R0, [SP, #+2];
      OR
STRB     R0, [SP, #+3];
      OR
STRH     R0, [SP, #+1];

//
// Load the same word-aligned value of the same SRAM location
// into a core register (such as R0)
//
LDR       R0, [SP, #+0];
```

This assembly sequence causes erroneous values only if these three instructions are executed in this order. However, the three instructions do not have to be consecutive, which means that other instructions can be placed in between the first and the second instructions, or the second and the third instructions, and the false value still occurs. Other instructions include, but are not limited to, branches in Flash, accesses to non-SRAM locations such as peripherals, and writes to other SRAM locations.

Pointers, structures, and unions are common C code methods that can be found in user code that may generate this assembly sequence and, therefore, result in incorrect values for variables. If using interrupts, it is possible to continue the assembly sequence in the interrupt handler, which could also return incorrect data.

For more information about this erratum as well as C code examples that may generate this assembly sequence, refer to the document, *Non-Word-Aligned Write to SRAM Additional Information* (SPMA047).

Workaround:

The type of compiler and optimization settings used in your application affects whether the problematic assembly code is generated from your user code. Each compiler behaves a little differently with respect to this erratum. The behavior for each compiler is not guaranteed due to the large number of compiler and tool version combinations.

At the assembly level, loading a volatile 32-bit-aligned word value from a different address in SRAM after storing and before loading in the assembly instruction sequence yields a correct value. A dummy SRAM load of a volatile 32-bit-aligned word from a different SRAM memory location should be inserted after the second assembly instruction (storing a byte or halfword from the core register to the desired SRAM location at a non-word-aligned offset) and before the third assembly instruction (loading the same word-aligned value of the desired SRAM location into a core register). This also means that a dummy SRAM load of a volatile 32-bit-aligned word from a different SRAM memory location should also be placed at the beginning of any interrupt routine, in case the third assembly instruction is executed before leaving the handler.

For more information about this erratum as well as C code examples that may generate this assembly sequence, refer to the document, *Non-Word-Aligned Write to SRAM Additional Information* (SPMA047).

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

2.4 Internal reset supervisors may not prevent incorrect device operation during power transitions

Description:

This microcontroller incorporates internal Power-On Reset (POR) and Brown-Out Reset (BOR) supervisors to ensure that code only executes when power to the device is within specification. However, gaps in the voltage and timing thresholds of the internal supervisors result in a risk of incorrect operation during VDD power transitions.

Unexpected operation may occur that can include brief execution of random sections of user code including ROM functions and random instructions, as well as incorrect power-up initialization. The uncontrolled brief execution of random instructions may result in the undesired erasing or writing of non-volatile memories and GPIO state changes. There is also the possibility that the device may be left in a state where it does not operate correctly until a clean power cycle has been completed.

The Power-On Reset gap occurs because the supervisor can release internal state machine operation as soon as 6.0 ms after the VDD supply reaches 1.9 V. If VDD is still below the minimum operating voltage of 3.0 V after 6.0 ms, the power-up state machine may not function correctly, resulting in the effects described above. The $\overline{\text{RST}}$ pin of the device has no effect on the initialization state machine, therefore, a complete power-cycle is required to restore the initialization state machine.

The Brown-Out Reset threshold (V_{BTH}) gap occurs because the brown-out supervisor has a threshold as low as 2.85 V, which is less than the minimum operating voltage on VDD, and also because it can take several microseconds to respond. BOR gaps can be encountered after power up, during steady state operation power-on, if the VDD rail has glitches, and also during power-down.

Workaround:

After initial power-up, any processor operation with VDD below 3.0 V may result in unexpected code execution resulting in the effects described above. The processor must be halted or the $\overline{\text{RST}}$ signal must be driven Low prior to VDD dropping below 3.0 V and stay in that state until VDD is above 3.0 V.

If VDD falls below 2.1 V, it must continue to fall until it reaches 1.5 V. VDD must stay below 1.5 V for at least 36 µs to ensure that a POR is triggered correctly. Additionally, the VDD power-up time between 1.9 V and 3.0 V must be at most 6.0 ms. If VDD falls below 3.0 V but stays above 2.1 V, it is not necessary for the voltage to continue falling below 2.1 V. VDD can come back up to 3.0 V without any additional timing requirements.

The system designer must ensure they meet the requirements listed below for power-up, steady state, and power-down:

1. The VDD power-up, steady state, and power-down waveform meets the timing requirements shown in Figure 1 on page 11.

2. The power-up transition of VDD between 1.9 V and 3.0 V must not have any points where it decreases in voltage (must be monotonic).

3. The power-down transition of VDD between 3.0 V and 1.5 V must not have any points where it increases in voltage (must be monotonic).

4. Once steady-state operation between 3.0 V and 3.6 V is achieved, $\overline{\text{RST}}$ must go Low or the CPU execution must be halted prior to VDD falling below 3.0 V.

5. The **Brown-Out Reset Control (PBORCTL)** register must be set so that a brown-out event causes a reset.

Depending on the system environment requirement, items 3, 4, and 5 in the above list may be met by using a voltage supervisor, such as the TLV803M, to monitor a higher voltage rail from which the VDD supply is regulated. Figure 2 on page 12 shows this implementation with a voltage supervisor monitoring the 5-V rail and a voltage trip point of 4.38 V. A voltage supervisor with a lower voltage trip point can be used to monitor the VDD (3.3-V) rail, however this supervisor must assert reset before VDD reaches 3.0 V. Regardless of the implemented voltage supervisor circuit, the system designer must ensure that there is enough time to assert $\overline{\text{RST}}$ Low prior to VDD falling below 3.0 V. Figure 3 on page 12 shows the resulting waveform of the circuit shown in Figure 2 on page 12.

Figure 1. VDD Waveform Signature Limits

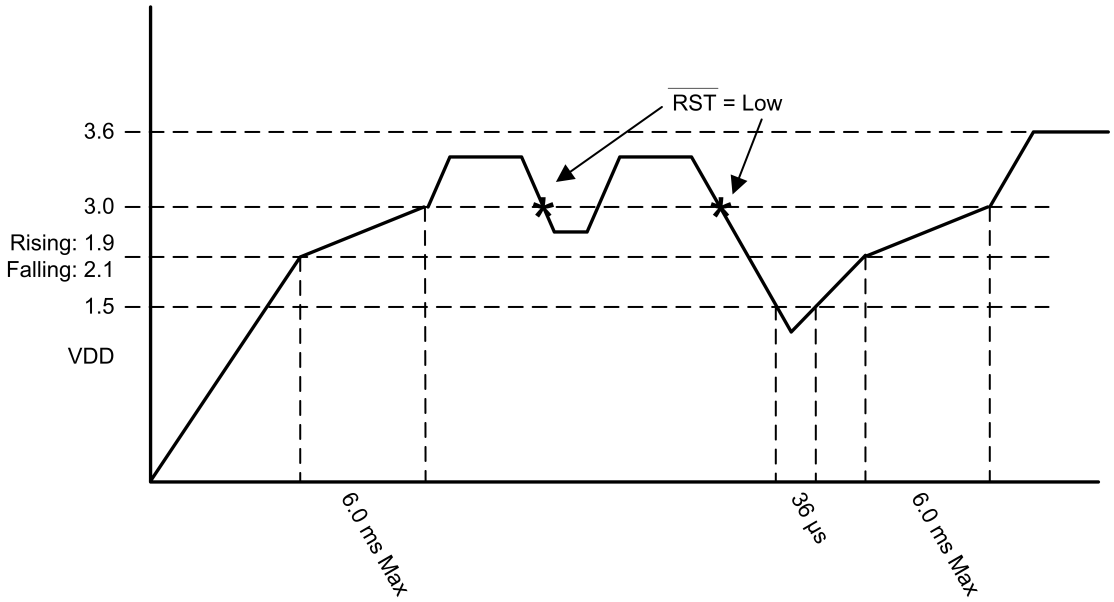


Figure 2. Using a Voltage Supervisor to Monitor the Voltage Rail

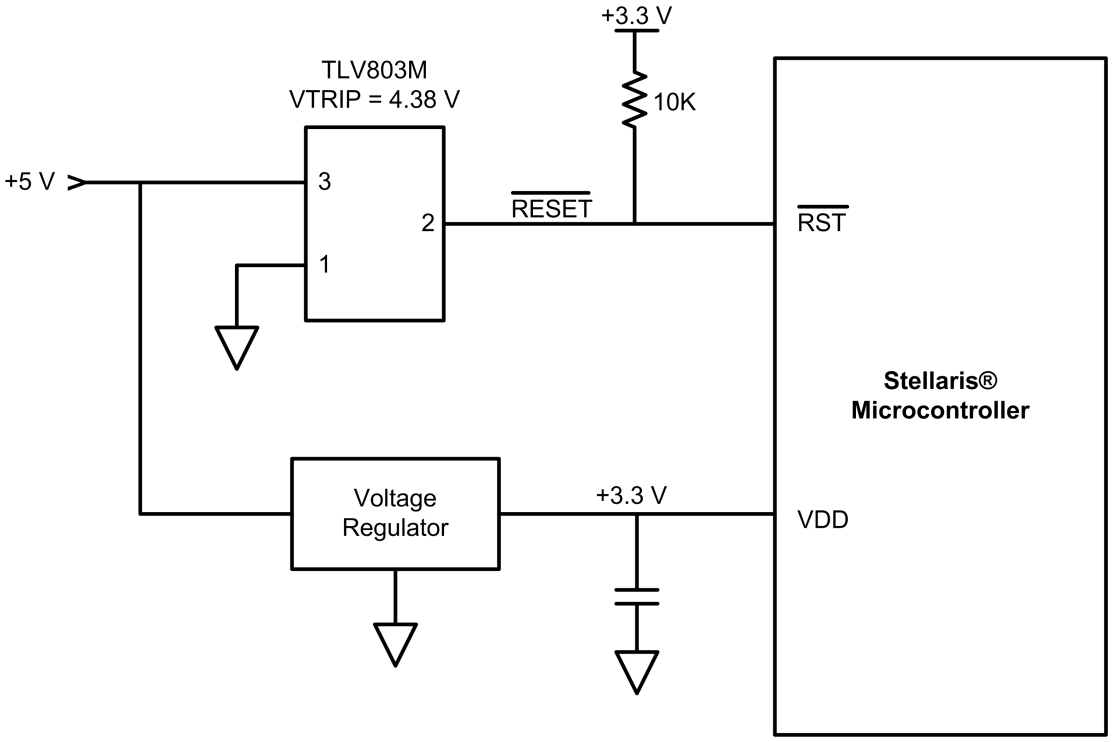
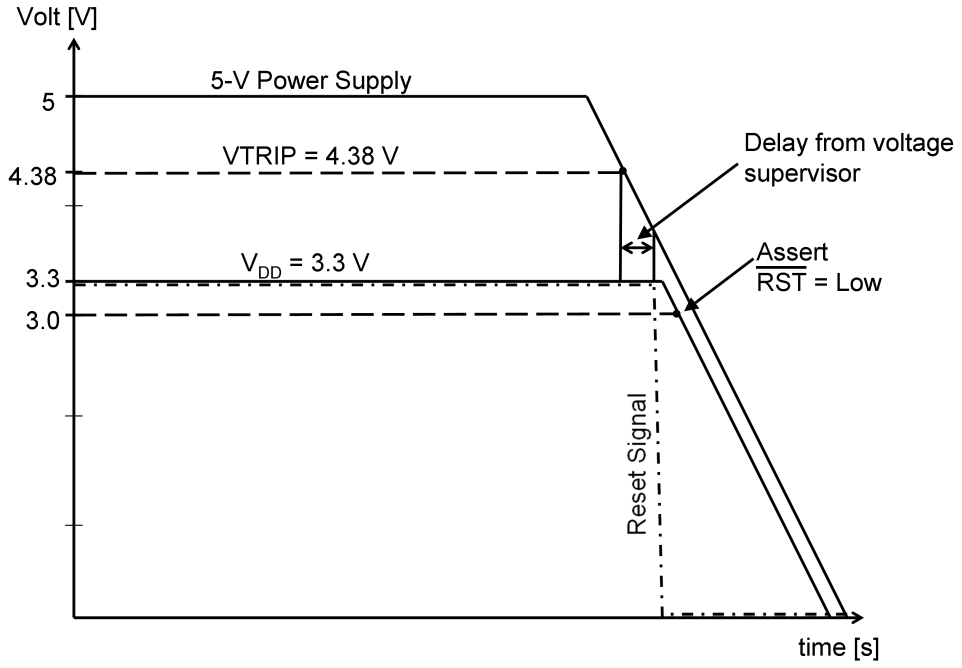


Figure 3. Resulting Waveform Using the Voltage Supervisor Circuit



Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

3 Hibernation Module

3.1 VDD3ON mode may not be used

Description:

The VDD3ON mode may not be used.

Workaround:

None. Do not use the VDD3ON mode to enter hibernation.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

3.2 The WRC bit in the Hibernation Control register is R/W

Description:

The WRC bit in the **Hibernation Control (HIBCTL)** register can be written. This bit should be a read-only bit.

Workaround:

Wait until the WRC bit is set before writing to the **HIBCTL** register. Always use a read-modify-write sequence when writing to the register to avoid changing the state of the WRC bit. Changing the value of the WRC bit can cause improper operation.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

3.3 Writes to Hibernation module registers may change the value of the RTC

Description:

If the Hibernation module's RTC counter is active, any write to certain Hibernation module registers that occurs while the RTC counter is changing from the current value to the next can cause corruption of the RTC counter stored in the **HIBRTCC** register. Registers affected are: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA**.

Workaround:

The user application must guarantee that writes to the affected Hibernation module registers cannot occur on the RTC counter boundary. Any initial configuration of the affected Hibernation module registers must be done before enabling the RTC counter.

There are two ways to update affected Hibernation Module registers after initial configuration:

1. Use the Hibernation RTC match interrupt to perform writes to the affected Hibernation module registers. Assuming the interrupt is guaranteed to be serviced within 1 second, this technique provides a mechanism for the application to know that the RTC update event has occurred and that it is safe to write data to the affected Hibernation module registers. This method is useful for applications that don't require many writes to Hibernation module registers.

2. Set up a secondary time-keeping resource to indicate when it is safe to perform writes to the affected Hibernation module registers. For example, use a general purpose timer in combination with the Hibernation RTC match interrupt. In this scenario, the RTC match interrupt is used to both update the match register value and enable the general purpose timer in one-shot mode. The timer must be configured to have a maximum time-out period of less than 1 second. In this configuration, a global variable is used to indicate that it is safe to perform writes to the affected Hibernation module registers. When the one-shot timer times out, the timer interrupt updates the global variable to indicate that writes are no longer safe. This procedure is repeated on every RTC match interrupt.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

3.4 Hibernation Module 4.194304-MHz oscillator supports a limited range of crystal load capacitance values

Description:

For some 4.194304-MHz crystals, the manufacturer-recommended crystal value may be outside of the capabilities of the Hibernate module oscillator. If the crystal manufacturer's recommended load capacitance is used, the hibernate oscillator may fail to start.

For a parallel-resonant oscillator circuit, the total load capacitance C_L (as specified by the manufacturer) is calculated as follows:

$C_L = (C_1 * C_2) / (C_1 + C_2) + C_S$

Due to the workaround, C_1 and C_2 are limited to 20 pF. Using 3 pF for stray capacitance (C_S), the formula above shows that a crystal with C_L of 13 pF is the highest value supported due to this errata. Refer to the crystal datasheet to determine which crystals have an acceptable load capacitance (CI) range.

Workaround:

Use load capacitors of 20 pF or less (18 pF is typical). Note that for some crystals, this value may pull the oscillator frequency slightly away from the crystal manufacturer's specified accuracy. Your crystal manufacturer can provide this information.

Alternatively, use an external 32.768-kHz oscillator as the source for the Hibernation module clock.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

4 ROM

4.1 The ROM_FlashProgram() function may not correctly program the Flash memory above 50 MHz

Description:

The ROM_FlashProgram() function may not correctly program the Flash memory when the system clock is above 50 MHz. As a result, the ROM boot loader may not function at system clock speeds above 50 MHz.

Workaround:

When using the ROM_FlashProgram() function, ensure that the system clock frequency is no higher than 50 MHz or load the StellarisWare version of FlashProgram() into Flash memory and use that version of the function. When invoking the ROM boot loader, do not use system clock frequencies above 50 MHz.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

5 Flash Memory

5.1 Deep-Sleep mode must not be used

Description:

Deep-sleep mode must not be used.

Workaround:

Use Sleep or Hibernation mode.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

5.2 Mass erase must not be used if Flash protection bits are used

Description:

The mass erase function using the MERASE bit in the **Flash Memory Control (FMC)** register must not be used in systems that clear any of the **Flash Memory Protection Program Enable n (FMPPEn)** bits. For Rev A1 and A2 devices, mass erase can be used as long as none of the **FMPPEn** bits are cleared.

Workaround:

Erase Flash memory with the page erase function using the ERASE bit in the **FMC** register instead of the mass erase function.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

5.3 Page erase or program must not be performed on a protected Flash page

Description:

The erase function using the ERASE bit in the **Flash Memory Control (FMC)** register and the program function using the WRITE bit in the **FMC** register or the WRBUF bit in the **FMC2** register must not be used in systems that clear the bit in **FMPPEn** that corresponds to that page of Flash. For Rev A1 and A2 devices, erase and program can be used as long as neither of the corresponding **FMPPEn** bits are cleared.

Workaround:

Only erase and program memory that is not protected by the corresponding **FMPPEn** bits.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

5.4 Flash memory endurance cycle specification is 100 cycles

Description:

The Flash memory endurance cycle specification (maximum program/erase cycles) is 100 cycles. Failure to adhere to the maximum number of program/erase cycles could result in corruption of the Flash memory contents and/or permanent damage to the device.

Workaround:

None. Because the failure mechanism is a function of the third-party Flash memory technology used in this device, there is no workaround. This third-party Flash memory technology is used only in the affected 130-nm Stellaris products and will not be used in any future devices. All other Stellaris products use Flash memory technology that exceeds industry quality and endurance cycle standards.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

6 µDMA

6.1 The µDMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules

Description:

The µDMA controller fails to generate DMA requests from Timer A in the General-Purpose Timer modules when in the Event Count and Event Time modes.

Workaround:

Use Timer B.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

6.2 The µDMA does not generate a completion interrupt when transferring to and from GPTM 2A and 2B

Description:

The µDMA module does not generate a completion interrupt on the Timer 2 interrupt vector when transferring data to and from Timers 2A and 2B. The µDMA can successfully transfer data to and from Timers 2A and 2B; however, there is no interrupt to indicate that the transfer is complete.

Workaround:

If a completion interrupt is required, use an alternate GPTM.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

7 GPIO

7.1 PB1 has permanent internal pull-up resistance

Description:

Regardless of its configuration, PB1 has a maximum internal pull-up resistance of 800 ohms that turns on when the voltage on the pin is approximately 1.2 V. Due to this internal resistance, up to 3 mA of current may be sourced during the transition from 1.2 V to 3.3 V.

Workaround:

When this pin is configured as an input, the external circuit must drive with an impedance less than or equal to 300 Ω to provide enough drive strength to over-drive the internal pull-up and achieve the necessary V_{IL} voltage level. Ensure that the driver can sink the temporary current. In addition, do not use PB1 in open-drain mode.

if this pin is configured as an output, be aware that if the output was driven high and a non-POR reset occurs, the output may be driven high after reset unless it has a 300-Ω resistor on it. Once the pin is configured as an output, the pin drives the programmed level.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8 General-Purpose Timers

8.1 The General-Purpose Timer match register does not function correctly in 32-bit mode

Description:

The **GPTM Timer A Match (GPTMTAMATCHR)** register triggers a match interrupt and a DMA request, if enabled, when the lower 16 bits match, regardless of the value of the upper 16 bits.

Workaround:

None.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.2 A spurious DMA request is generated when the timer rolls over in Input-Edge Time mode

Description:

When the timer is in Input-Edge Time mode and rolls over after the terminal count, a spurious DMA request is generated.

Workaround:

Either ignore the spurious interrupt, or capture the edge time into a buffer via DMA, then the spurious interrupt can be detected by noting that the captured value is the same as the previous capture value.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.3 A spurious DMA request is generated when the timer rolls over the 16-bit boundary

Description:

When the timer is in 32-bit periodic or one-shot mode and is enabled to generate periodic DMA requests, a spurious DMA request is generated when the timer rolls past 0x0000FFFF.

Workaround:

Only use DMA with a 16-bit periodic timer.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.4 The value of the prescaler register is not readable in Edge-Count mode

Description:

In Edge-Count mode, the prescaler is used as an 8-bit high order extension to the 16-bit counter. When reading the **GPTM Timer n (GPTMTnR)** register as a 32-bit value, the bits [23:16] always contain the initial value of the **GPTM Timer n Prescale (GPTMTnPR)** register, that is, the "load" value of the 8-bit extension.

Workaround:

None.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.5 ADC trigger and Wait-on-Trigger may assert when the timer is disabled

Description:

If the value in the **GPTM Timer n Match (GPTMTnMATCHR)** register is equal to the value of the timer counter and the TnOTE bit in the **GPTM Control (GPTMCTL)** register is set, enabling the ADC trigger, the trigger fires even when the timer is disabled (the TnEN bit in the **GPTMCTL** register is clear). Similarly, if the value in the **GPTMTnMATCHR** register is equal to the value of the timer counter and the TnWOT bit in the **GPTM Timer n Mode (GPTMTnMR)** register is set, enabling the Wait-on-Trigger mode, the trigger fires even when the timer is disabled.

Workaround:

Enable the timer before setting the TnOTE bit. Also, for the Wait-on-Trigger mode, ensure that the timers are configured in the order in which they will be triggered.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.6 Wait-on-Trigger does not assert unless the TnOTE bit is set

Description:

Wait-on-Trigger does not assert unless the TnOTE bit is set in the **GPTMCTL** register.

Workaround:

If the TnWOT bit in the **GPTM Timer n Mode (GPTMTnMR)** register is set, enabling the Wait-on-Trigger mode, the TnOTE bit must also be set in the **GPTMCTL** register in order for the Wait-on-Trigger to fire. Note that when the TnOTE bit is set, the ADC trigger is also enabled.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.7 Do not enable match and timeout interrupts in 16-bit PWM mode

Description:

16-bit PWM mode generates match and timeout interrupts in the same manner as periodic mode.

Workaround:

Ensure that any unwanted interrupts are masked in the **GPTMTnMR** and **GPTMIMR** registers.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.8 Do not use µDMA with 16-bit PWM mode

Description:

16-bit PWM mode generates match and timeout µDMA triggers in the same manner as periodic mode.

Workaround:

Do not use µDMA to transfer data when the timer is in 16-bit PWM mode.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.9 Writing the GPTMTnV register does not change the timer value when counting up

Description:

When counting up, writes to the **GPTM Timer n Value (GPTMTnV)** register do not change the timer value.

Workaround:

None.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.10 The prescaler does not work correctly when counting up in periodic or one-shot mode

Description:

When counting up, the prescaler does not work correctly in 16-bit periodic or snap-shot mode.

Workaround:

Do not use the prescaler when counting up in 16-bit periodic or snap-shot mode.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

8.11 Snapshot must be enabled in both Timer A and B when in 32-bit snapshot mode

Description:

When a periodic snapshot occurs in 32-bit periodic mode, only the lower 16-bit are stored into the **GPTM Timer A (GPTMTAR)** register.

Workaround:

If both the TASNAPS and TBSNAPS bits are set in the **GPTM Timer A Mode (GPTMTAMR)** register, the entire 32-bit snapshot value is stored in the **GPTMTAR** register.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

9 Watchdog Timers

9.1 Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module

Description:

Periodically reloading the count value into the **Watchdog Timer Load (WDTLOAD)** register of the Watchdog Timer 1 module will not restart the count, as specified in the data sheet.

Workaround:

Disable the Watchdog Timer 1 module before reprogramming the counter. Alternatively, clear the watchdog interrupt status periodically outside of the interrupt handler by writing any value to the **Watchdog Interrupt Clear (WDTICR)** register.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

9.2 Watchdog Timer 1 module asserts reset signal even if not programmed to reset

Description:

Even if the reset signal is not enabled (the RESEN bit of the **Watchdog Control (WDTCTL)** register is clear), the Watchdog Timer 1 module will assert a reset signal to the system when the time-out value is reached for a second time.

Workaround:

Clear the Watchdog Timer 1 interrupt once the time-out value is reached for the first time by writing any value to the **Watchdog Interrupt Clear (WDTICR)** register.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

9.3 WDTLOAD yields an incorrect value when read back

Description:

If the Watchdog Timer 1 module is enabled and configured to run off the PIOSC, writes to the **Watchdog Load (WDTLOAD)** register yield an incorrect value when read back.

Workaround:

None.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

10 UART

10.1 The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled

Description:

The RTRIS (UART Receive Time-Out Raw Interrupt Status) bit in the **UART Raw Interrupt Status (UARTRIS)** register should be set when a receive time out occurs, regardless of the state of the RTIM enable bit in the **UART Interrupt Mask (UARTIM)** register. However, currently the RTIM bit must be set in order for the RTRIS bit to be set when a receive time out occurs.

Workaround:

For applications that require polled operation, the RTIM bit can be set while the UART interrupt is disabled in the NVIC using the IntDisable(n) function in the StellarisWare Peripheral Driver Library, where n is 21, 22, or 49 depending on whether UART0, UART1 or UART2 is used. With this configuration, software can poll the RTRIS bit, but the interrupt is not reported to the NVIC.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

10.2 LIN mode Sync Break does not have the correct length

Description:

When operating as a LIN master, the microcontroller provides a Sync Break of the length that is programmed in the BLEN field in the **UART LIN Control (UARTLCTL)** register. However, the actual Sync Break length is 1 less than what is programmed in the BLEN field as shown in Table 3 on page 24.

Table 3. SyncBreak Length

BLEN Encoding	Data Sheet Value	Actual Value
0x0	13T bits	12T bits
0x1	14T bits	13T bits
0x2	15T bits	14T bits
0x3	16T bits	15T bits

Workaround:

Adjust the BLEN encoding to correspond to the actual Sync Break required.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

10.3 When UART LIN or SIR mode is enabled, µDMA burst transfer does not occur

Description:

If the LIN or the IrDA Serial Infrared (SIR) mode is enabled in the UART peripheral and the uDMA UARTn RX or UARTn TX channel is configured to do a burst transfer, the burst data transfer does not occur.

Workaround:

Clear the SETn bit in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register to have the uDMA UART channel respond to single or burst requests to ensure that the data transfer occurs.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

10.4 UART transfers fail at certain system clock frequency and baud rate combinations

Description:

UART data transfers using the TXRIS and RXRIS interrupt bits and FIFOs fail for certain combinations of the system clock frequency and baud rate.

System Clock Freq [MHz]	32	24	16	10	8	5	4	2	1
Failing Baud Rate [bps]	<460800	<460800	<230400	<460800	<115200	<230400	<57600	<38400	<19200

Workaround:

Use a system clock frequency above 32MHz if using the UART with the raw interrupt status bits or use µDMA UART data transfers instead of the TXRIS and RXRIS bits. When using µDMA UART data transfers, there are no system clock frequency and baud rate conflicts.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

11 SSI

11.1 Freescale SPI Mode at low SSIClk frequencies can yield data corruption

Description:

Data transmitted by the SPI slave may be corrupted when using Freescale SPI Mode 0 at an SSIClk frequency between 0.5 MHz to 1.1 MHz and a system clock frequency of 33 MHz or lower.

Workaround:

Operate the Freescale SPI Mode 0 at an SSIClk frequency above 1.1 MHz and use a system clock frequency above 33 MHz or use a different mode.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

12 Ethernet Controller

12.1 Encoding error in the Ethernet MAC LED Encoding (MACLED) register

Description:

Configuring the LED0 or LED1 field of the **Ethernet MAC LED Encoding (MACLED)** register to 0x8 should cause the corresponding LED to report a combined link + activity status. However, it instead only reports activity status (i.e. exactly the same as encoding 0x1).

Workaround:

None.

Silicon Revision Affected:

A2

Fixed:

Not yet fixed.

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746
http://www.ti.com/stellaris
http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm

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 which meet ISO/TS16949 requirements, mainly for automotive use. Components which have not been so designated are neither designed nor intended for automotive use; and TI will not be responsible for any failure of such components to meet such requirements.

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