

## Stellaris® LM3S9B92 RevC Errata

This document contains known errata at the time of publication for the Stellaris® LM3S9B92 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.

Date	Revision	Description
July 2010	3.1	<ul style="list-style-type: none"> <li>Added issue "The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled" on page 13.</li> <li>Clarified workaround and corrected silicon revisions affected on "Slow V<sub>DD</sub> ramp may occasionally cause device failures" on page 14.</li> </ul>
June 2010	3.0	<ul style="list-style-type: none"> <li>Added statement to "Reset patch is required" on page 2 that the vector table must be located at address 0x1000.</li> </ul>
June 2010	2.9	<ul style="list-style-type: none"> <li>Clarified "Reset patch is required" on page 2.</li> <li>Added issue "Mass erase must not be used if Flash protection bits are used" on page 6.</li> <li>Added issue "Page erase or program must not be performed on a protected Flash page" on page 6.</li> <li>Added issue "Slow V<sub>DD</sub> ramp may occasionally cause device failures" on page 14.</li> </ul>
May 2010	2.8	<ul style="list-style-type: none"> <li>Clarified "Reset patch is required" on page 2.</li> <li>Added issue "Boot Loader in ROM is unusable" on page 5.</li> </ul>
April 2010	2.7	<ul style="list-style-type: none"> <li>Added information to "Software Considerations" appendix.</li> <li>Additional minor clarifications and corrections.</li> </ul>
April 2010	2.6	<ul style="list-style-type: none"> <li>Started tracking revision history.</li> </ul>

Erratum Number	Erratum Title	Revision(s) Affected
1.1	Reset patch is required	C1
1.2	Deep-Sleep mode should not be used	C1, C3
2.1	Boot Loader in ROM is unusable	C1
3.1	Mass erase must not be used if Flash protection bits are used	C1, C3
3.2	Page erase or program must not be performed on a protected Flash page	C1, C3
4.1	The $\mu$ DMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules	C1, C3
5.1	Schmitt input feature does not function correctly	C1

Erratum Number	Erratum Title	Revision(s) Affected
6.1	The General-Purpose Timer match register does not function correctly in 32-bit mode	C1, C3
6.2	A spurious DMA request is generated when the timer rolls over in Input-Edge Time mode	C1, C3
6.3	A spurious DMA request is generated when the timer rolls over the 16-bit boundary	C1, C3
6.4	The value of the prescaler register is not readable in Edge-Count mode	C1, C3
6.5	ADC trigger and Wait-on-Trigger may assert when the timer is disabled	C1, C3
6.6	Wait-on-Trigger does not assert unless the TnOTE bit is set	C1, C3
6.7	Do not enable match and timeout interrupts in 16-bit PWM mode	C1, C3
6.8	Do not use $\mu$ DMA with 16-bit PWM mode	C1, C3
6.9	Writing the GPTMTnV register does not change the timer value when counting up	C1, C3
6.10	The prescaler does not work correctly when counting up in periodic or one-shot mode	C1, C3
6.11	Snapshot must be enabled in both Timer A and B when in 32-bit snapshot mode	C1, C3
7.1	Writes to Watchdog Timer 1 module WDTLOAD register sometimes fail	C1, C3
8.1	ADC hardware averaging produces erroneous results in differential mode	C1, C3
9.1	Phantom interrupts occur in Smart Card mode	C1
9.2	The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled	C1, C3
10.1	Encoding error in the Ethernet MAC LED Encoding (MACLED) register	C1, C3
11.1	Startup time after power-on reset exceeds specification	C1, C3
11.2	Slow $V_{DD}$ ramp may occasionally cause device failures	C1

# 1 Internal Memory

## 1.1 Reset patch is required

### Description:

An error in the design of a production test mode can cause software to crash in certain rare instances. A reset patch pre-programmed into Flash memory provides an effective workaround.

In addition, there is an approximately 1 in 100,000 chance that executing any type of reset other than a power-on reset may cause the core to hang. These reset types include:

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

**Workaround:**

The required software change is pre-programmed into the 4-KB block of Flash memory at address 0x0000.0000 and is protected from being erased or programmed. In addition, the second 2-KB block at address 0x0000.0800 is not readable. The patch also requires an interrupt handler, which must be hooked to the Flash interrupt in the main vector table.

The patch places the following requirements on applications:

- Interrupts must be enabled for a minimum of 5 continuous seconds every 2 hours because the patch runs in the context of the Flash interrupt handler.
- Any interrupt that is higher priority than the Flash interrupt may not program or erase Flash memory.
- Applications must be aware that the master Cortex-M3 interrupt in the NVIC is enabled as part of the workaround initialization, which is likely to occur before the application would normally enable interrupts. Care must be taken to ensure that this condition is taken into account when initializing other peripherals and interrupts.
- On rare occasions, a Flash interrupt may take up to 2 ms to be processed. Worst case, this may happen twice per every 12 hours of running time.
- The ROM boot loader does not function. If you need a boot loader, put the StellarisWare boot loader in Flash memory, ensuring that the linker file has a start address of 0x0000.1000
- The option to force the ROM boot loader to execute at reset with an external pin does not function. Changing the `PORT` and `PIN` fields of the **Boot Configuration (BOOTCFG)** register has no effect.

To protect the pre-programmed code in Flash memory, the following restrictions apply to the use of the device:

- The mass erase function using the `MERASE` bit in the **Flash Memory Control (FMC)** register does not erase the part.
- A toggle-mass erase must not be executed, meaning that the debug port unlock sequence in LMFlash Programmer must not be executed.
- The vector table must be located at address 0x1000 and application code must be configured to run from address 0x1000.
- Because the first 4-KB block of Flash memory is protected, the **Flash Memory Protection Read Enable 0 (FMPRE0)** and **Flash Memory Protection Program Enable 0 (FMPPE0)** registers cannot be used by application software.

Refrain from implementing the internal watchdog, brown-out, and main oscillator failure resets and don't use an external reset. If the device hangs, execute a power-on reset.

A software reset can still be executed by calling `WorkaroundSysCtlReset()` in the place of `SysCtlReset()` and `ROM_SysCtlReset()`.

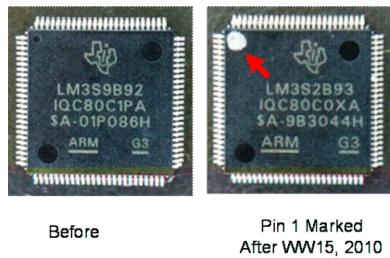
See Appendix A on page 15 for further information on how to implement code for these devices and use debugging tools.

Note that two versions of the patch code have been released. Devices that have the patch code pre-programmed in Flash memory have a silver metallic dot located on the pin1 indicator. Devices that have the patch code plus a UART-only flash loader pre-programmed in Flash memory (see

“Boot Loader in ROM is unusable” on page 5) have a date code of 05 or later. See Part Marking Examples below. In addition, a 32-bit date code is programmed at address 0x0000.07F8 and a 32-bit patch revision number programmed at address 0x0000.07FC.

### Part Marking Examples

The silver metallic dot on the part indicating that the patch code was pre-programmed in Flash memory looks like the following:



Parts with a date code of 05 (May 2010) or later contain the patch plus a UART-only flash loader (see “Boot Loader in ROM is unusable” on page 5). To determine the date code of your part, look at the third line in the part markings, at the fourth and fifth characters following the dash (highlighted in red below). The first number after the dash indicates the last decimal digit of the year. The next character indicates the month, in hexadecimal. So, in the below example, the 9B indicates a date code of November 2009.



The table below shows some example date codes:

Date Code	Date
9B	November 2009
9C	December 2009
01	January 2010
02	February 2010
03	March 2010
04	April 2010
05	May 2010
06	June 2010
07	July 2010
08	August 2010
09	September 2010
0A	October 2010
0B	November 2010
0C	December 2010

### Silicon Revision Affected:

C1

**Fixed:**

Fixed in Rev C3, although a side effect of the fix is that all of the reset sources listed under the "Description" head above will take approximately 2 ms to complete on C3 devices.

## 1.2 Deep-Sleep mode should not be used

**Description:**

Deep-sleep mode should not be used.

**Workaround:**

Use Sleep or Hibernation mode.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 2 ROM

### 2.1 Boot Loader in ROM is unusable

**Description:**

The reset patch that is pre-programmed into Flash memory (see "Reset patch is required" on page 2) makes the Boot Loader in ROM unusable.

**Workaround:**

On parts with a date code of 05 or later (see "Part Marking Examples" on page 4), a UART-only flash loader is pre-programmed into Flash memory at 0x1000 (similar to the serial flash loader that is provided on Stellaris parts that do not have a ROM). This flash loader uses the same protocol as boot loader in ROM but only operates via UART0. Unlike the boot loader, it is a one-time-use loader; if true boot loader functionality is required, the flash loader should be used to load a boot loader. LM Flash Programmer can be used to communicate with the flash loader by selecting the serial interface on the main tab.

For more details on the flash loader protocol, see the UART section in the Boot Loader appendix in your data sheet.

**Silicon Revision Affected:**

C1

**Fixed:**

Fixed in Rev C3.

## 3 Flash

### 3.1 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 Read Enable n (FMPREN)** or **Flash Memory Protection Program Enable n (FMPPE n)** bits. For Rev C1 devices, this means that mass erase must not be used because bits in the **FMPRE 0** and **FMPPE 0** registers are cleared to protect the reset patch that is stored in the first block of Flash memory. For Rev C3 devices, mass erase can be used as long as none of the **FMPREN** or **FMPPE n** 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:

C1, C3

#### Fixed:

Not yet fixed.

### 3.2 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 **FMPREN** or **FMPPE n** that corresponds to that page of Flash. For C1 devices, this means that erase and program of locations 0x0 through 0xFFFF must not be used because bits in the **FMPRE 0** and **FMPPE 0** registers are cleared to protect the reset patch that is stored in the first block of Flash memory. For Rev C3 devices, erase and program can be used as long as neither of the corresponding **FMPREN** or **FMPPE n** bits are cleared.

#### Workaround:

Only erase and program memory that is not protected by the corresponding **FMPPE n** or **FMPREN** bits.

#### Silicon Revision Affected:

C1, C3

#### Fixed:

Not yet fixed.

## 4 $\mu$ DMA

### 4.1 The $\mu$ DMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules

**Description:**

The  $\mu$ DMA 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 5 GPIO

### 5.1 Schmitt input feature does not function correctly

**Description:**

The Schmitt input on digital inputs may generate spurious transitions when connected to low slew-rate signal sources. If the input signal has a slew rate of less than  $1\text{V}/\mu\text{s}$ , a negative edge can generate several additional transitions into the microcontroller even though the input signal is still within the hysteresis band. Positive edges are not affected.

The additional transitions can cause anomalous operation in any peripherals or GPIOs that use digital inputs. Most at risk are peripherals that use pull-up resistors (I<sup>2</sup>C, GPIOs) or that typically involve slower signals (sensor inputs). This behavior can affect the noise immunity of digital inputs. As a result, arbitration may be lost during communication when the I<sup>2</sup>C module is the master.

**Workaround:**

Ensure that all signals connected to digital inputs have a slew rate of at least  $1\text{V}/\mu\text{s}$ . In some applications, reducing the resistance value of pull-up and pull-down resistors may be necessary. Note that R-C filters, such as low pass, on digital input signals should only be used if the slew-rate is still above  $1\text{V}/\mu\text{s}$ , or if additional transitions on the falling edge can be tolerated. Adding an external Schmitt-trigger circuit is a requirement for circuits where slow transitions are unavoidable and system noise levels are high.

**Silicon Revision Affected:**

C1

**Fixed:**

Fixed in Rev C3.

## 6 General-Purpose Timer

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

**Description:**

The **GPT MTimer A Match (GPT MTAMATCHR)** 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:**

C1, C3

**Fixed:**

Not yet fixed.

### 6.2 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:**

C1, C3

**Fixed:**

Not yet fixed.

### 6.3 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:**

C1, C3



**Fixed:**

Not yet fixed.

## 6.4 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 **GPT MTimer n (GPT MTnR)** register as a 32-bit value, the bits [23:16] always contain the initial value of the **GPT MTimer n Prescale (GPT MTnPR)** register, that is, the "load" value of the 8-bit extension.

**Workaround:**

None.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

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

**Description:**

If the value in the **GPT MTimer n Match (GPT MTn MATCHR)** register is equal to the value of the timer counter and the **TnOTE** bit in the **GPT MControl (GPT MCTL)** register is set, enabling the ADC trigger, the trigger fires even when the timer is disabled (the **TnEN** bit in the **GPT MCTL** register is clear). Similarly, if the value in the **GPT MTn MATCHR** register is equal to the value of the timer counter and the **TnWOT** bit in the **GPT MTimer n Mode (GPT MTn MR)** 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.6 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 **GPT MCTL** register.

**Workaround:**

If the **TnWOT** bit in the **GPT MTimer n Mode (GPT MTn MR)** register is set, enabling the Wait-on-Trigger mode, the **TnOTE** bit must also be set in the **GPT MCTL** 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.7 Do not enable match and timeout interrupts in 16-bit PWMmode

**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 **GPT MTn MR** and **GPT MIMR** registers.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.8 Do not use $\mu$ DMA with 16-bit PWMmode

**Description:**

16-bit PWM mode generates match and timeout  $\mu$ DMA triggers in the same manner as periodic mode.

**Workaround:**

Do not use  $\mu$ DMA to transfer data when the timer is in 16-bit PWM mode.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.9 Writing the GPT MTn V register does not change the timer value when counting up

**Description:**

When counting up, writes to the **GPT MTimer n Value (GPT MTn V)** register do not change the timer value.

**Workaround:**

None.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.10 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 6.11 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 **GPT MTimer A (GPT MTAR)** register.

**Workaround:**

If both the **TASNAPS** and **TBSNAPS** bits are set in the **GPT MTimer A Mode (GPT MTAMR)** register, the entire 32-bit snapshot value is stored in the **GPT MTAR** register.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 7 Watchdog Timer 1

### 7.1 Writes to Watchdog Timer 1 module WDTLOAD register sometimes fail

**Description:**

Due to the independent clock domain of the Watchdog Timer 1 module, writes to the **Watchdog Load (WDTLOAD)** register may sometimes fail, even though the **WRC** bit in the **WDTCTL1** register is set after the write occurs.

**Workaround:**

After performing a write to the **WDTLOAD** register, read the contents back and verify that they are correct. If they are incorrect, perform the write operation again.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 8 ADC

### 8.1 ADC hardware averaging produces erroneous results in differential mode

**Description:**

The implementation of the ADC averaging circuit does not work correctly when the ADC is sampling in differential mode and the difference between the voltages is approximately 0.0V.

**Workaround:**

Do not use hardware averaging in differential mode. Instead, use the FIFO to store results and average them in software.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

## 9 UART

### 9.1 Phantom interrupts occur in Smart Card mode

**Description:**

In Smart Card mode, after receiving a valid TX interrupt, phantom parity error interrupts occur, even though all **UARTRIS** and **UARTMIS** bits are clear.

**Workaround:**

Make sure to always clear the parity error interrupt in the interrupt handler, even when the `PERIS` and `PEMIS` bits are clear.

**Silicon Revision Affected:**

C1

**Fixed:**

Fixed in Rev C3.

## 9.2 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 10 Ethernet Controller

### 10.1 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:**

C1, C3

**Fixed:**

Not yet fixed.

## 11 Electrical Characteristics

### 11.1 Startup time after power-on reset exceeds specification

**Description:**

When performing a power-on reset, the startup time is approximately 1.66 ms longer than specified in the data sheet.

**Workaround:**

None.

**Silicon Revision Affected:**

C1, C3

**Fixed:**

Not yet fixed.

### 11.2 Slow $V_{DD}$ ramp may occasionally cause device failures

**Description:**

A slow  $V_{DD}$  power-on ramp may cause the LDO to start improperly. When this occurs, the microcontroller may not come out of reset properly, causing it to be in an unknown state. In most cases, a subsequent POR addresses the issue. In rare cases, flash corruption may occur. If flash corruption occurs, a POR will not address the issue.

Asserting the external  $\overline{RST}$  signal during a slow  $V_{DD}$  ramp does not resolve this issue.

**Workaround:**

The  $V_{DD}$  power supply must have a ramp-up time of 1 ms or less (5% to 90%). Because the use of the brown-out reset is not possible due to "Reset patch is required" on page 2, ensure that the power supply is stable. If the  $V_{DD}$  voltage drops, the voltage should collapse to 5% of VDD. An external Supply Voltage Supervisor (SVS) controlling a power switch may be used to meet these requirements.

**Silicon Revision Affected:**

C1

**Fixed:**

Fixed on C3.

# Appendix A Software considerations for devices with the software patch in Flash memory

## A.1. Flash workaround interrupt handler

The “Reset patch is required” on page 2 erratum requires an interrupt handler which must be hooked to the Flash interrupt in the main vector table. This handler, `WorkaroundIntHandler()`, is located at address 0x880. A user's Flash Interrupt Handler should contain code similar to the following:

```
//*****
//
// The FLASH workaround interrupt handler
//
//*****
void
FlashIntHandler(void)
{
    //
    // Call the PATCH Code Interrupt handler
    //
    ((void (*)(void))0x881)();
}
```

... Or ...

```
#define WorkaroundIntHandler ((void (*)(void))0x881)
//*****
//
// The FLASH workaround interrupt handler
//
//*****
void
FlashIntHandler(void)
{
    //
    // Call the PATCH Code Interrupt handler
    //
    WorkaroundIntHandler();
}
```

Alternatively, the vector table could simply be hard-coded as follows, to avoid one layer of function calls.

...

```
IntDefaultHandler,    // Analog Comparator 2
IntDefaultHandler,    // System Control (PLL, OSC, BO)
0x00000881,           // FLASH Control
IntDefaultHandler,    // GPIO Port F
IntDefaultHandler,    // GPIO Port G
```

...

## A.2. Software reset workaround

The “Reset patch is required” on page 2 erratum requires that SysCtlReset() and ROM\_SysCtlReset() not be used. Instead, the following code should be used:

```
#define WorkaroundSysCtlReset ((void (*)(void))0x801)
//
// Issue a software reset
//
WorkaroundSysCtlReset();
```

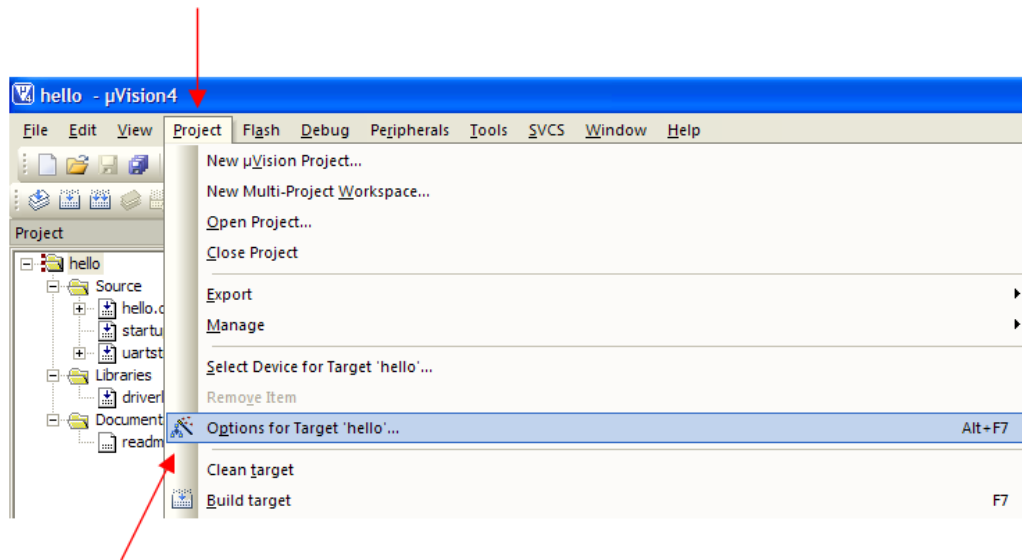
## A.3. Debugging considerations

When using debugging tools with the software patch, certain considerations must be observed as detailed in the following sections.

### A.3.1. Keil Real View® MDK-ARM Microcontroller Development Kit

If the debugger is configured to simply reset/connect, it attempts to disassemble code in a protected region of Flash memory. The simple solution is to configure the debugger to “load/run to main”. With this configuration, the patch code gets run and the Flash interrupt gets enabled. The PC runs to the main() function when the debugger starts. This option can be found in: Project □ Options for target '<project name>' as shown in Figure A-1 on page 16.

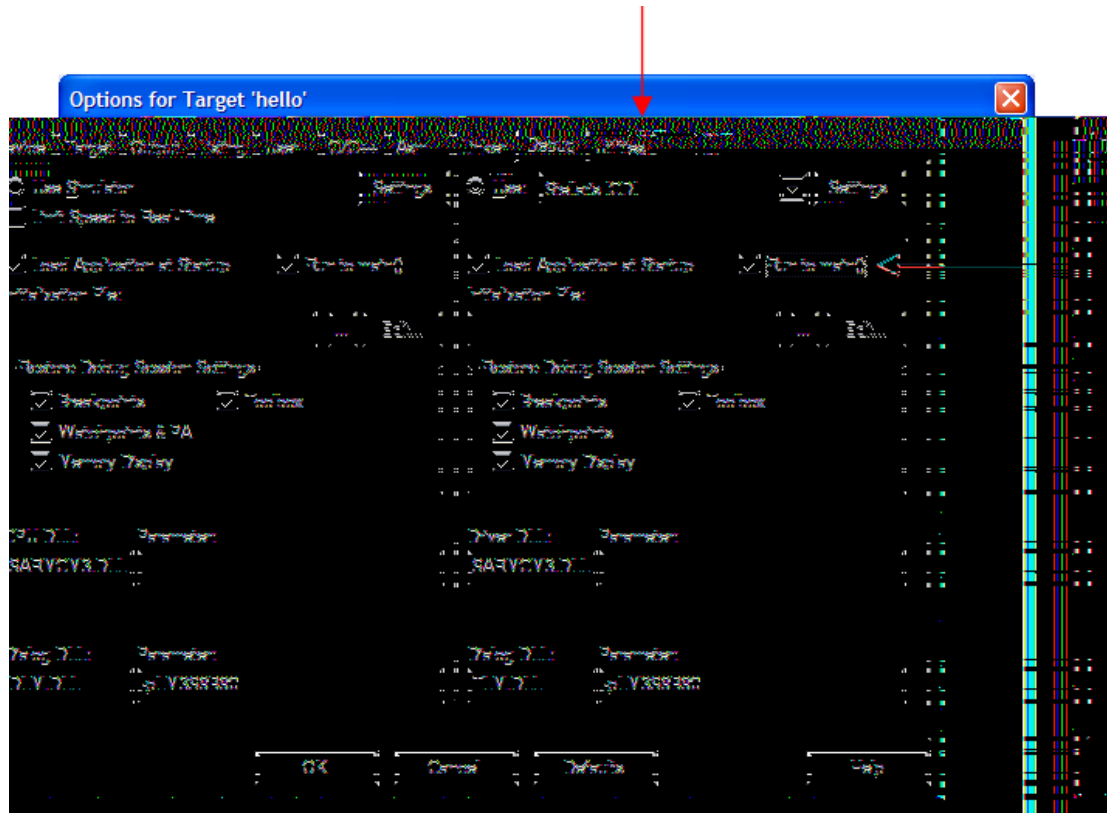
**Figure A-1. Navigating to the “Run to Main()” Option**



When the Options window opens, select the Debug tab. Under Debug select “Run to main()” as shown in Figure A-2 on page 17.



Figure A-2. Selecting the "Run to Main()" Option



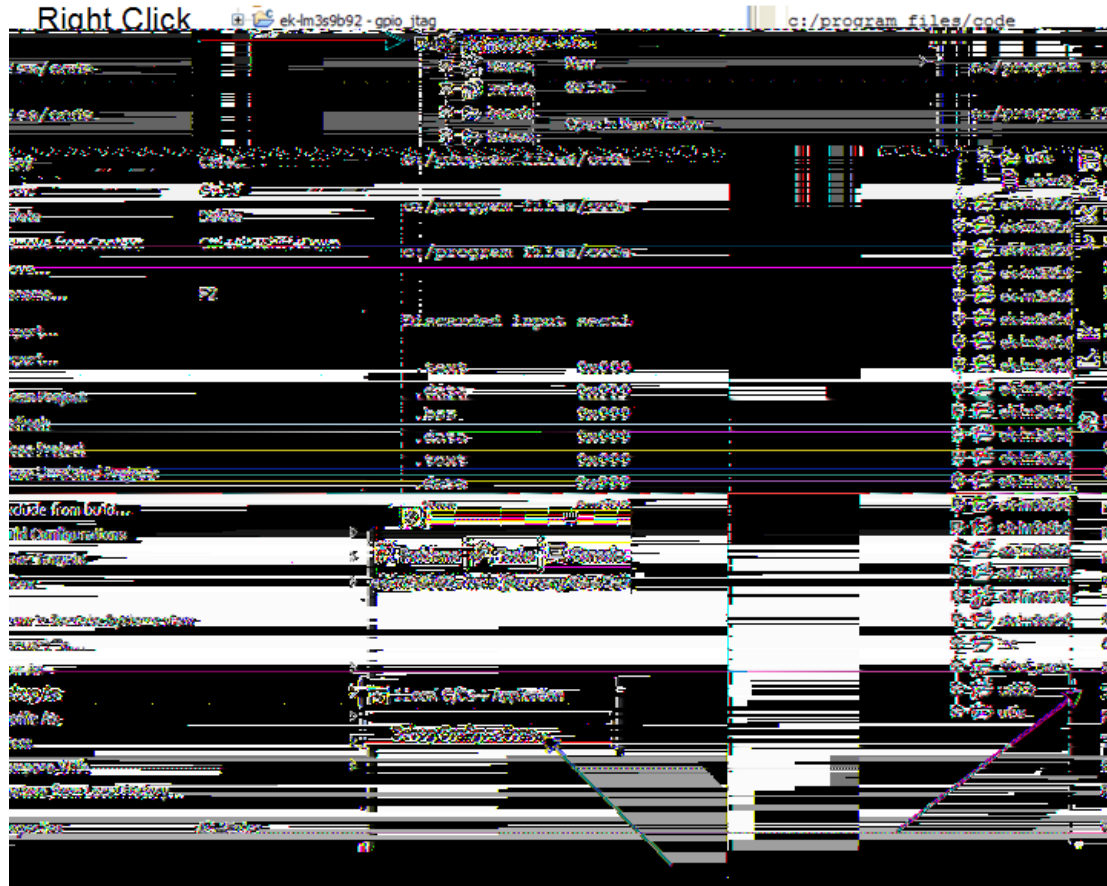
### A.3.2. Code Red Tools Suite

Code Red Tools Suite works seamlessly. Upon startup of the debugger (in default settings), the patch code gets run and the Flash interrupt gets enabled. The code starts executing at the main() function.

### A.3.3. CodeSourcery Sourcery G++

CodeSourcery does not run the patch code by default; it starts at the main() function. To get the patch code to run, manually set the stack pointer (SP) and program counter (PC). These register can be set by accessing the "Debugger Configurations..." tab. To do this, right click on the project and select Debug As ☐ Debug Configurations as shown in Figure A-3 on page 18.

Figure A-3. Navigating to the Debugger Configurations Window

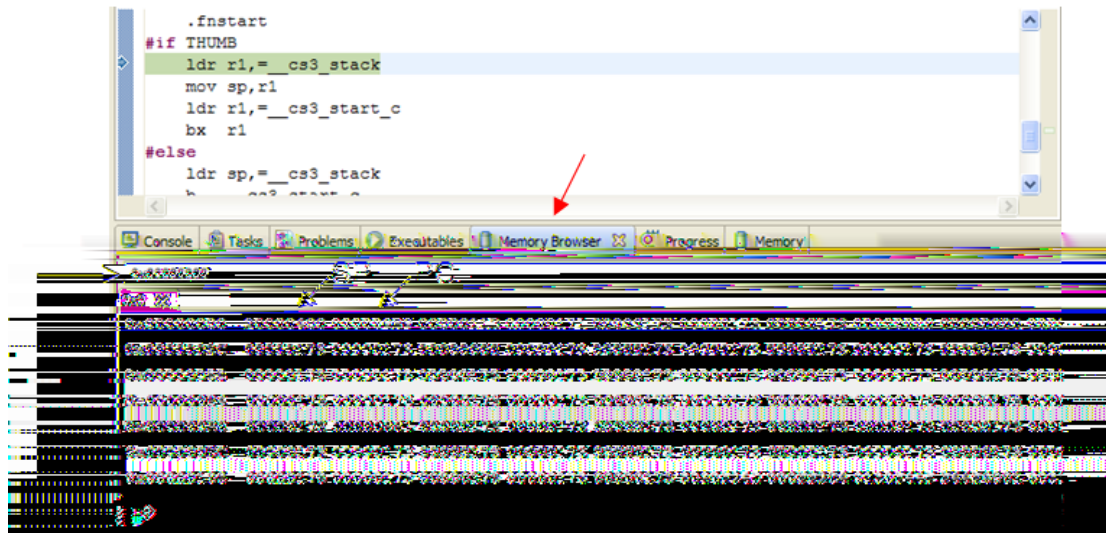


I had a hard time finding configuration options for the flash programmer and the debugger. Fortunately, the flash programmer does page erases by default, which allows us to program the part without any issues. The problem with the CodeSourcery debugger is that it does not run through the patch code by default, it just starts at your `main()` function. To get the patch code to run, you have to manually set the stack pointer (SP) and program counter (PC). The first thing you need to do is access the “Debugger Configurations...” tab. To do this, right click on your project and select **Debug As** ☐ **Debugger Configurations**.

Once the “Debugger Configurations” window is up, select the “Debugger” tab and check “Stop on startup at:” and enter “main” in the text box; also check “Stop on first assembler instruction” as shown in Figure A-4 on page 19.

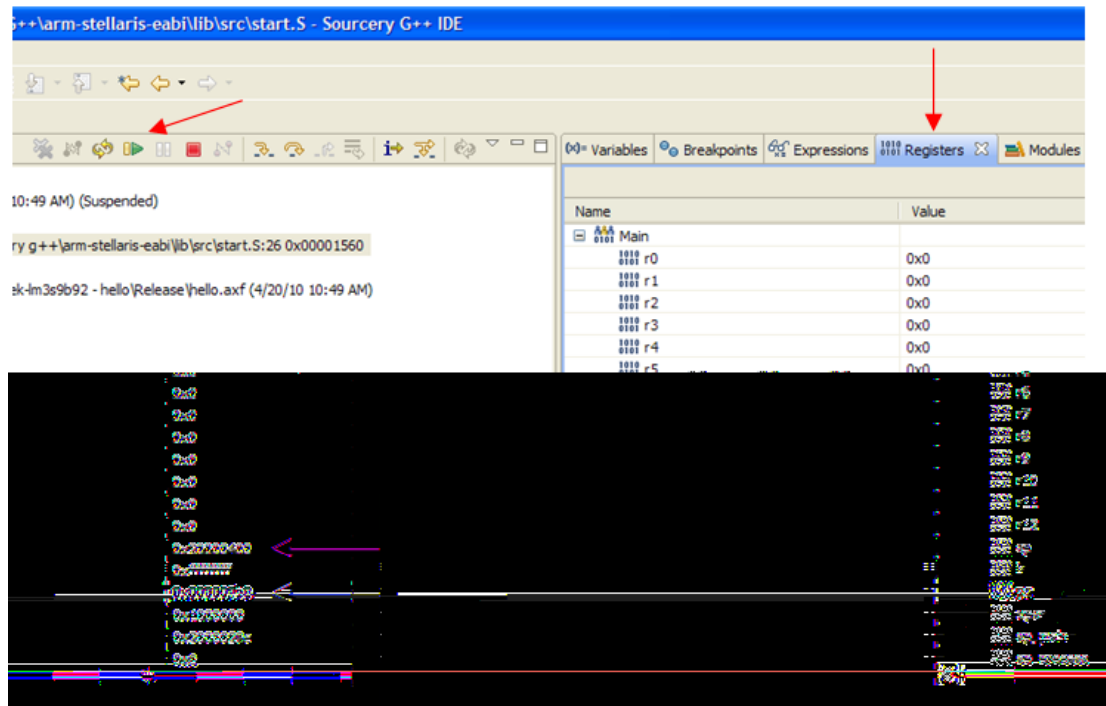
---

Figure A-5. Finding the Proper SP and PC to Use




Go to the “Registers” tab and enter these values for the SP and PC manually as shown in Figure A-6 on page 20.

Figure A-6. Entering the SP and PC Manually

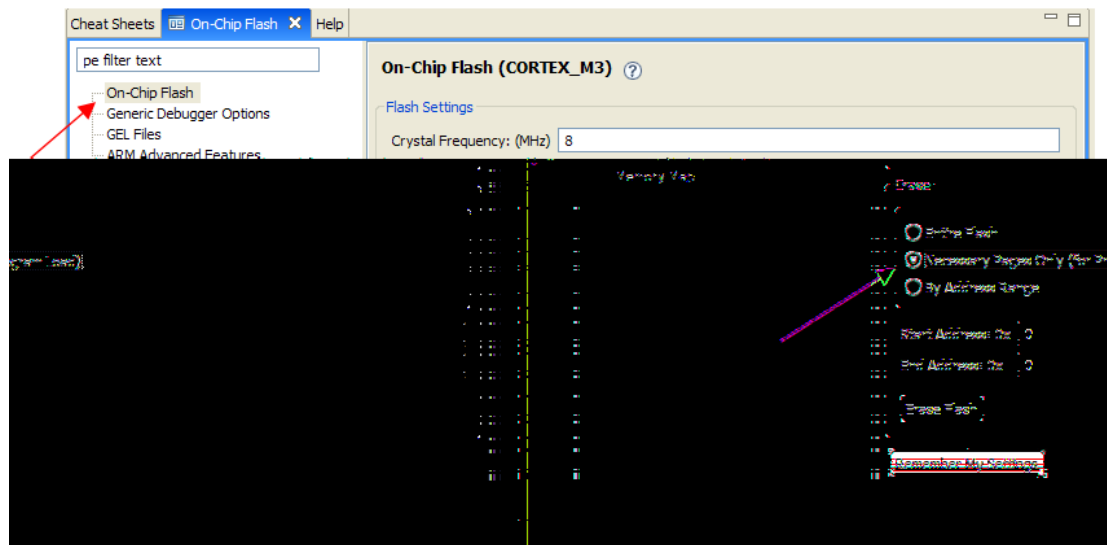


Once the new PC and SP have been entered, click the run button. The debugger will now successfully run through the patch code and stop at the main() function.

### A.3.4. Code Composer Studio

CCS does not work by default - it hangs when the debugger loads because the default flash erase option is mass erase (which won't work because the patch code is write protected). The flash erase option must be changed to page erase. To do this, start a debug session (even though it won't work correctly) to get access to the Flash settings. Once in the debug session, go to Tools  On-Chip Flash then under Erase select "Necessary Pages Only (for Program Load)" as shown in Figure A-7 on page 21.

**Figure A-7. Changing the Flash Erase Option**






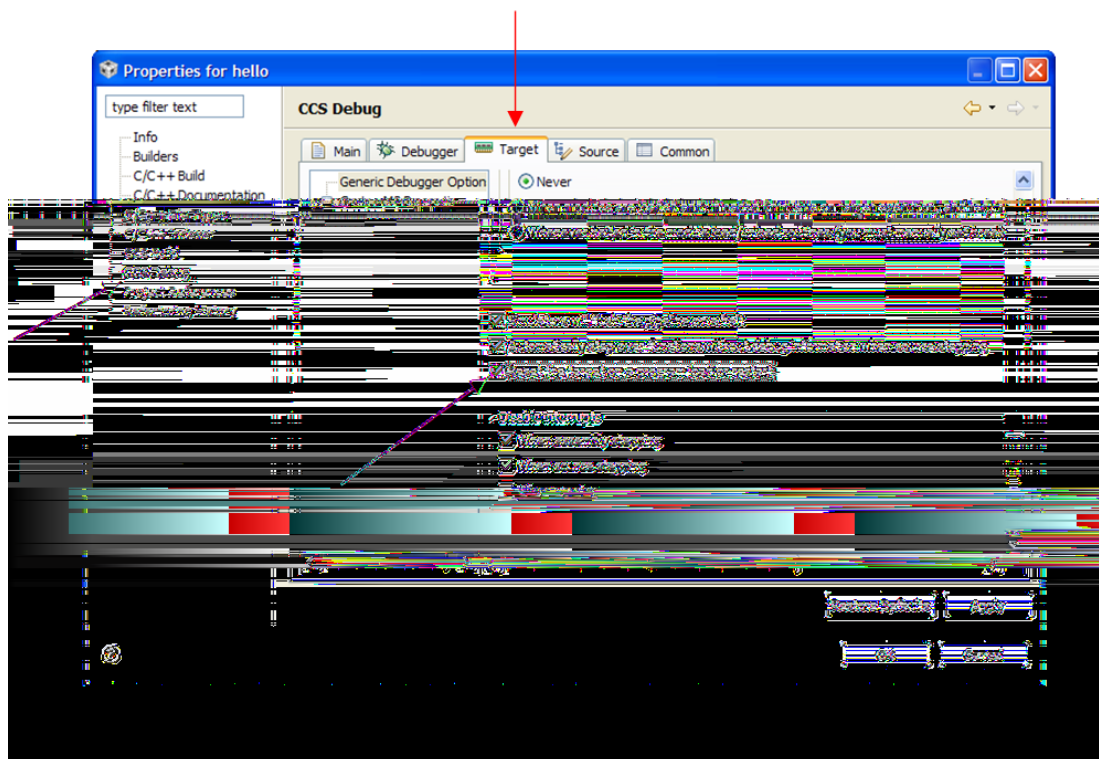
To get the patch code to run, a "Core Reset" must be executed in the IDE. Go to Target  Reset  Core Reset to reset the core, which results in the patch code being run. To configure the IDE to perform a core reset automatically, go to Target  Properties and on the left hand side of the window select CCS Debug. Once you select CCS Debug click on Target on the right hand side of the window. Scroll down until you see an option that says "Reset the target on program load or restart" and select it as shown in Figure A-8 on page 22.

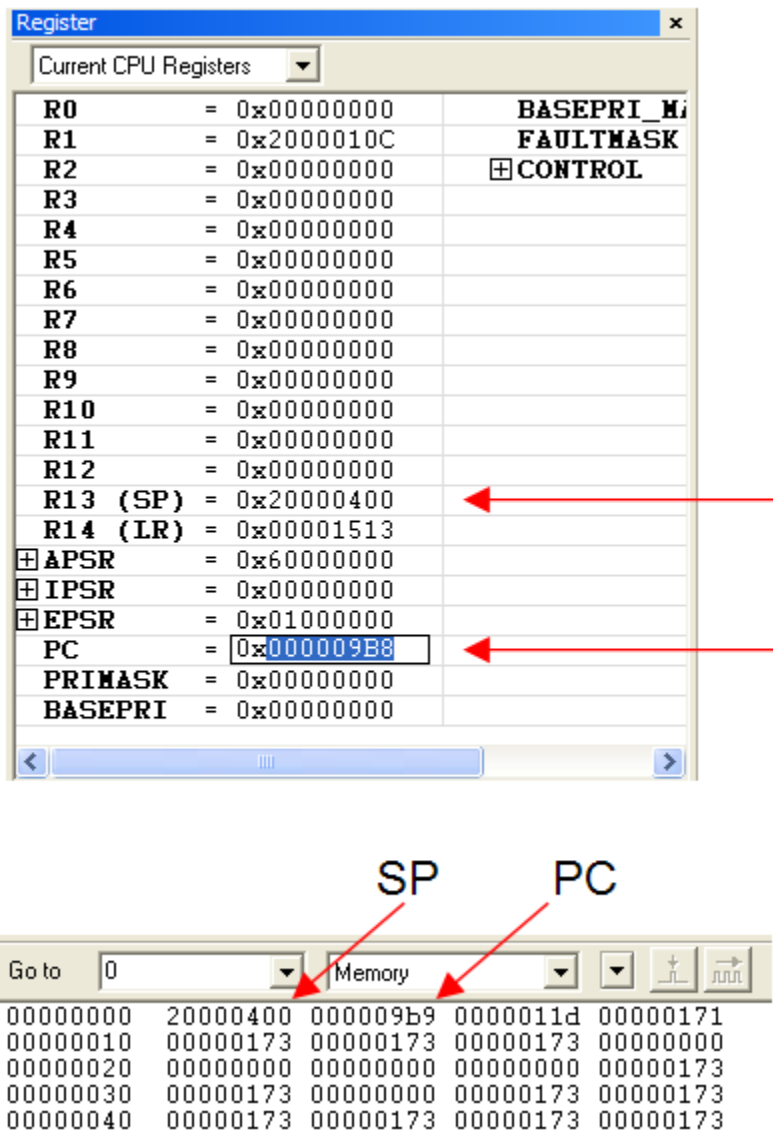
Figure A-8. Configuring the IDE to Reset on Program Load



### A.3.5. IAR Embedded Workbench™ for ARM and Cortex-M3

IAR has problems executing code at address 0x0000.0000 if it is not the main() application (such as bootloader examples). As a result, the PC and SP must be manually loaded. The values to load can be found at address 0x0000.0000 and 0x0000.0004 in the memory window and loaded into the PC (0x000009B8) and SP (0x20000400) as shown in Figure A-9 on page 23.

Figure A-9. Manually Configuring the PC and SP Registers



Set a breakpoint at the main() function and run the program. This process causes the Flash memory interrupt to be enabled and the debugger to operate as expected. Note: Every time you startup the debugger or do a reset in the IDE, you need to perform this sequence.

Copyright © 2008-2010 Texas Instruments Inc. All rights reserved. Stellaris and StellarisWare are registered trademarks of Texas Instruments. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as the property of others.

Texas Instruments  
108 Wild Basin, Suite 350  
Austin, TX 78746  
<http://www.ti.com/stellaris>





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