Application Report
SPNA190–September 2013

UART Bootloader for Hercules RM48x MCU

Quingjun Wang

ABSTRACT

This application report describes how to communicate with the Hercules™ UART bootloader. The UART bootloader is a small piece of code that can be programmed at the beginning of Flash to act as an application loader as well as an update mechanism for applications running on a Hercules Cortex™-R4 based RM48x microcontroller.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/spna190.

Contents

List of Figures

List of Tables

1 Introduction

An important requirement for most Flash memory-based systems is the ability to update firmware when installed in the end product. This ability is referred to as in-application programming (IAP). The UART bootloader provides a means of writing, reading, and erasing a predefined section of the program Flash memory that typically holds the user application code.

This document describes how to work with and customize the Hercules UART bootloader application. The bootloader is provided as source code which allows any part of the bootloader to be completely customized. The bootloader has been built and validated using Code Composer Studio™ v5 on the RM48x Hercules Development HDK.

Table 1 shows an overview of the organization of the source code provided with the bootloader.

Table 1. List of Source Code Files Used in SPI Bootloader

sys_startup.c	The start-up code used when TI's Code Composer Studio compiler is being used to build the bootloader.
bl_check.c	The code to check if a firmware update is required, or if a firmware update is being requested.
bl_check.h	Prototypes for the update check code.
bl_commands.h	The list of commands and return messages supported by the bootloader.
bl_config.h	Bootloader configuration file. This contains all of the possible configuration values.
bl_flash.c	The functions for erasing, programming the Flash, and functions for erase and program check
bl_flash.h	Prototypes for Flash operations
bl_link.cmd	The linker script used when the Code Composer Studio compiler is being used to build the bootloader.
bl_main.c	The main control loop of the bootloader.
bl_packet.c	The functions for handling the packet processing of commands and responses.
bl_packet.h	Prototypes for the packet handling functions.
bl_uart.h	Prototypes for the UART0 transfer functions.
bl_uart.c	The functions for transferring data via the COM0 port.
bl_vimram.c	VIM RAM table definition and initialization
hw_gio.c	Low-level GIO driver
hw_gio.h	Prototypes for low-level gio driver
hw_het.c	Low-level NHET driver
hw_het.h	Prototypes for low-level NHET driver
hw_interrupt_handler.c	Define the INT handlers
hw_pinmux.c	Function for define the pinmux
hw_pinmux.h	Prototypes for pinmux functions
hw_sci.c	Low-level SCI driver
hw_sci.h	Prototypes for low-level SCI driver
hw_system.c	Initialize system registers and PLL
bl_ymodem.c	Function for define the ymodem protocol
bl_ymodem.h	Prototype define the variables and functions
startup_eabi.c	Global variables initialization
sys_intvecs.asm	Interrupt vectors
sys_svc.asm	Software INT routines

2 Software Coding and Compilation

- The bootloader code is implemented in C, ARM® Cortex-R4F assembly coding is used only when absolutely necessary. The IDE is TI Code Composer Studio v5.4.
- The bootloader is compiled in the 32-BIT ARM mode.
- The bootloader is compiled and linked with the TI TMS470 code generation tools V 5.1.

3 On Reset

On reset, the MCU enters in supervisor mode and starts executing the bootloader. The interrupt vectors are setup as shown in Table 2.

Table 2. Vector Table in CAN Bootloader

Offset	Vector	Action
0x00	Reset Vector	Branch to entry point of bootloader (c_int00)
0x04	Undefined Instruction Interrupt	Branch to application vector table
0x08	Software Interrupt	Branch to application vector table
0x0C	Abort (Prefetch) Interrupt	Branch to application vector table
0x10	Abort (Data) Interrupt	Branch to application vector table
0x14	Reserved	Endless loop (branch to itself)
0x18	IRQ Interrupt	Branch to VIM
0x1C	FIQ Interrupt	Branch to VIM

4 During Bootloader Execution

During bootloader execution:

- MCU operates in supervisor mode
- MCU Clock is reconfigured and is maintained throughout the bootloader execution.
  - Clock Source: OSCIN = 16 MHz
  - System clock: HCLK = 80 MHz
  - Peripheral clock: VCLK = 40 MHz
- No interrupts are used
- Fix point is used throughout the bootloader execution
- F021 API V2.00.01 executes in RAM
- The SCI is configured as 115200, 8-N-1

5 MCU Initialization during Bootloader Execution

- Operating Mode: supervisor mode
- HCLK Frequency: OSCIN x 120 / 12
- VCLK Frequency: VCLK = HCLK / 2
- Peripheral Control: peripherals enabled
- SCI Setup: The SCI/LIN is setup for SCI communication. The setup is 115200 8-N-1.
- MCU do self-test, PBIST, parity check at startup

6 The Protocol Used in UART Bootloader

The bootloader is based on Ymodem protocol. The Ymodem protocol sends data in 1024-byte blocks. Error detection is applied to data blocks transmitted to the RM48x internal RAM. This is done through a comparison between the transmitted and received data. Blocks received unsuccessfully are acknowledged with a negative acknowledgment (NAK). For more details about the Ymodem protocol, see the existing literature.

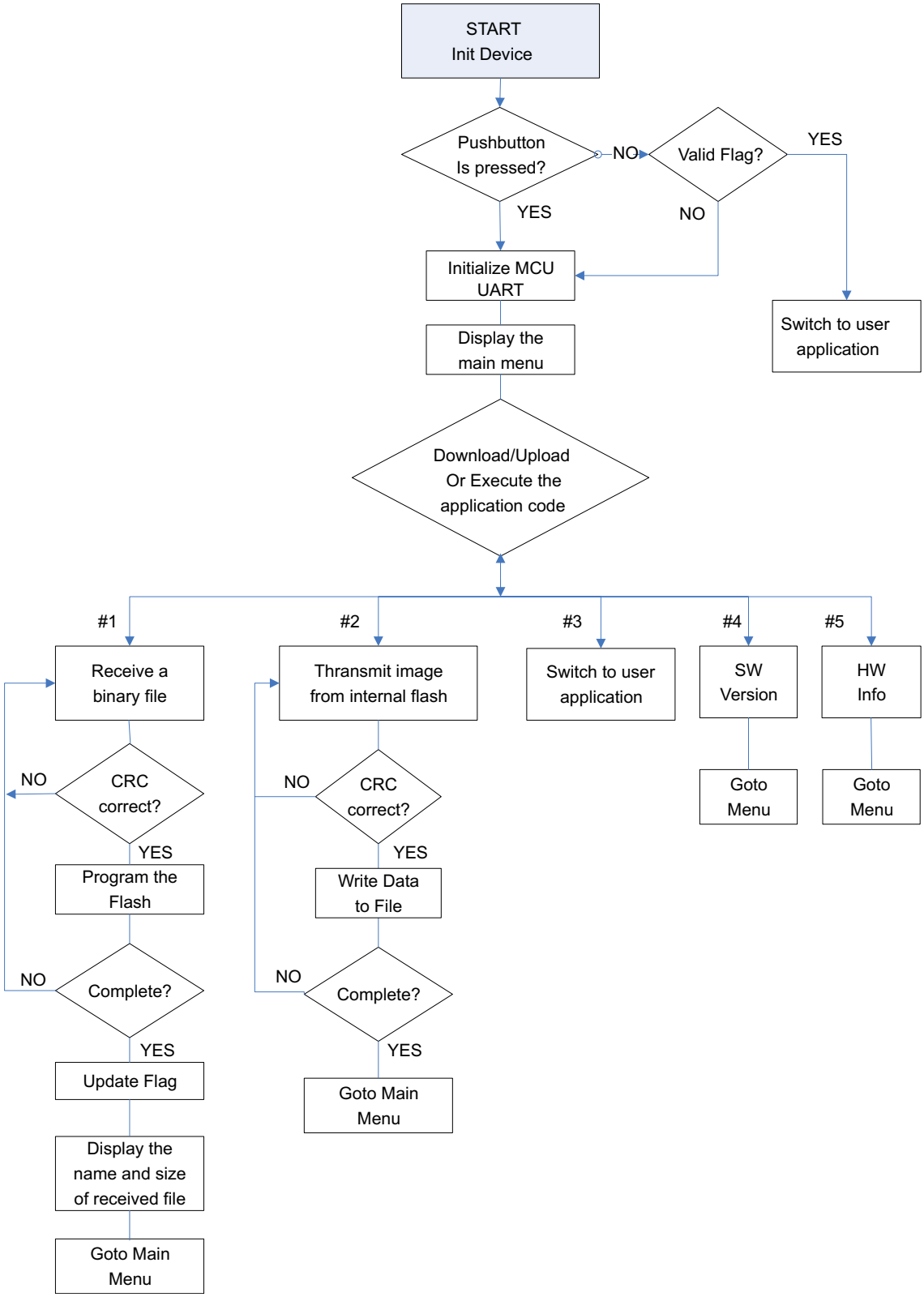

Figure 1. UART Bootloader Flowchart

7 UART Bootloader Operation

The UART bootloader is built with Code Composer Studio v5.x and loaded through the JTAG port into the lower part of the program memory at 0x0000.

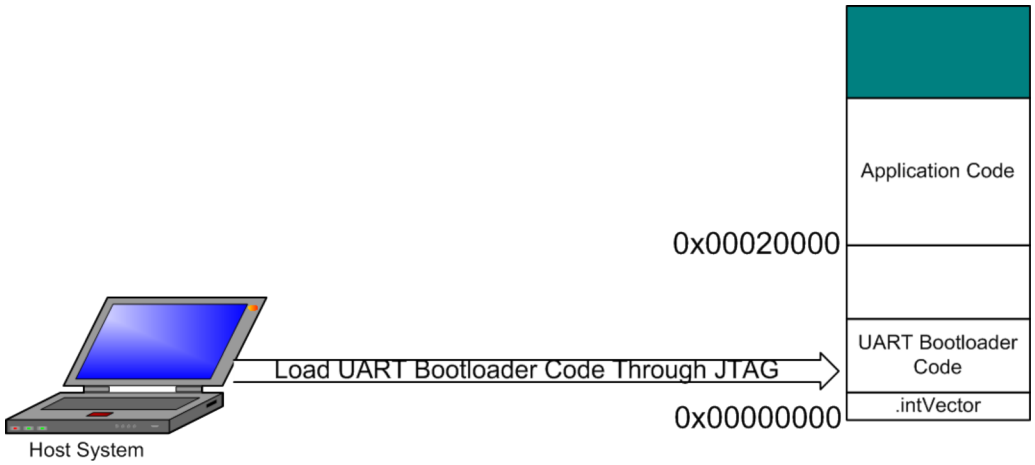

Figure 2. The UART Bootloader is Loaded Through the JTAG port

After HDK reset, the start-up code copies the Flash API of bootloader from Flash to SRAM, and execute the bootloader in Flash.

First, it will checks to see if the GPIO_A7 pin is pulled. If GPIO-A7 is pulled LOW, the application code is forced to be updated. The GPIO pin check can be enabled with ENABLE_UPDATE_CHECK in the *bl_config.h* header file. On RM48x HDK, the push button S1can toggle GPIO-A7.

Then, it will check the flag at 0x0007FF0. If the flag is a valid number (0x5A5A5A5A), the bootloader will jump to the application code at 0x00020000. If the flag is not the valid number, it will configure the UART, then start to update the user application code by calling *UpdaterUart()*. After all the application code is programmed successfully, the flag is also updated.

The updating results in Figure 3 displayed in the HyperTerminal window.

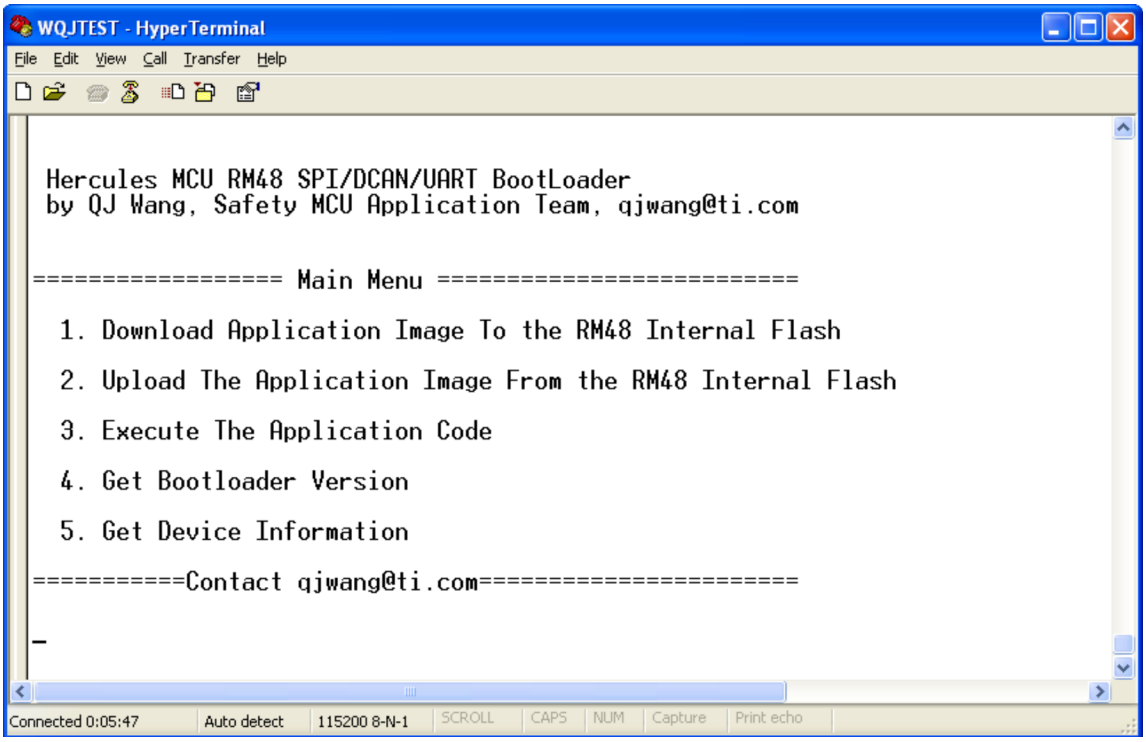

Figure 3. UART Bootloader Main Menu

1. Download the user application code to RM48x Flash.

   To download a binary file via HyperTerminal to the RM48x internal Flash, follow the procedure below:

   - Press “1” on the keyboard to choose the menu *Download image to internal Flash*.

     Then, in the Transfer menu, select *Send file…*
   - In the Filename field, type the name and the path of the binary file to be sent.
   - In the Protocol list, choose the Ymodem protocol
   - Click the *Send* button.

     Following these steps, the bootloader loads the binary file into the RM48x internal Flash. The bootloader will display the file name, and file size in the Hyperterminal window.

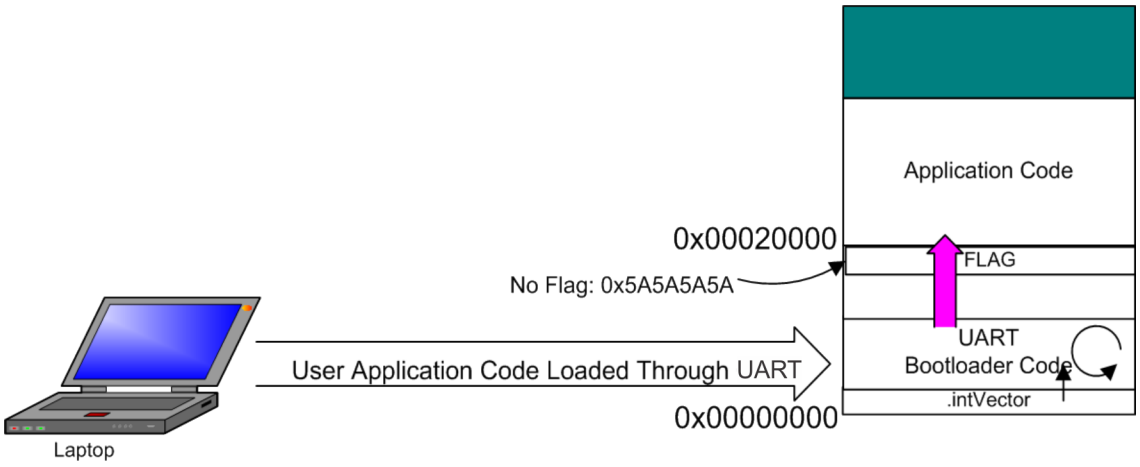

Figure 4. User Application Code is Loaded Through the UART Bootloader

2. Upload the application code from RM48x Flash

   To upload a copy of the internal Flash memory started from the user application address, do as follows:

   - Press 2 on the keyboard to select *Upload image* from the RM48x internal Flash
   - Select *Receive File* in the *Transfer* menu.
   - Choose the Directory of the binary file you want to create
   - From the *Protocol* list, select the Ymodem protocol
   - Click on the *Receive* button

   The file *UploadedApplicationImage.bin* is uploaded to the directory you defined in step 2.

3. Execute the application.

   Once the new application is downloaded successfully, press 3 on the keyboard to select the Execute.

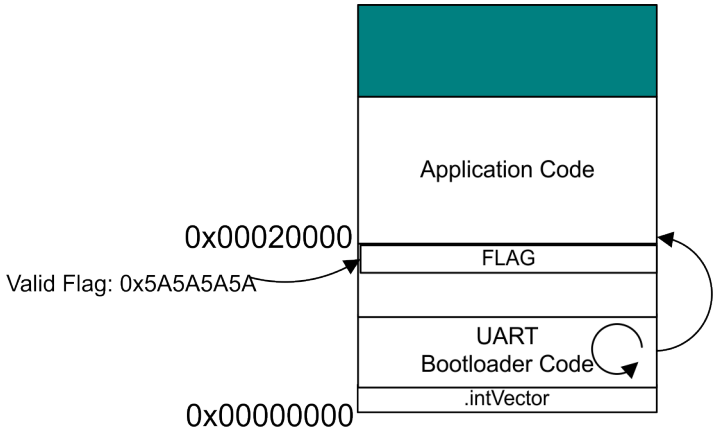

Figure 5. UART Bootloader Jumps to Application Code

4. Get Bootloader Software Version

   To get the software version, press 4 on the keyboard to retrieve the SW version from RM48x UAR bootloader.

5. Get the RM48x Device Information

   To get the device information, press 5 on the keyboard to retrieve the device information from RM48x UAR bootloader.

8 HyperTerminal Configuration

The bootloader requires a PC running HyperTerminal with the following settings:

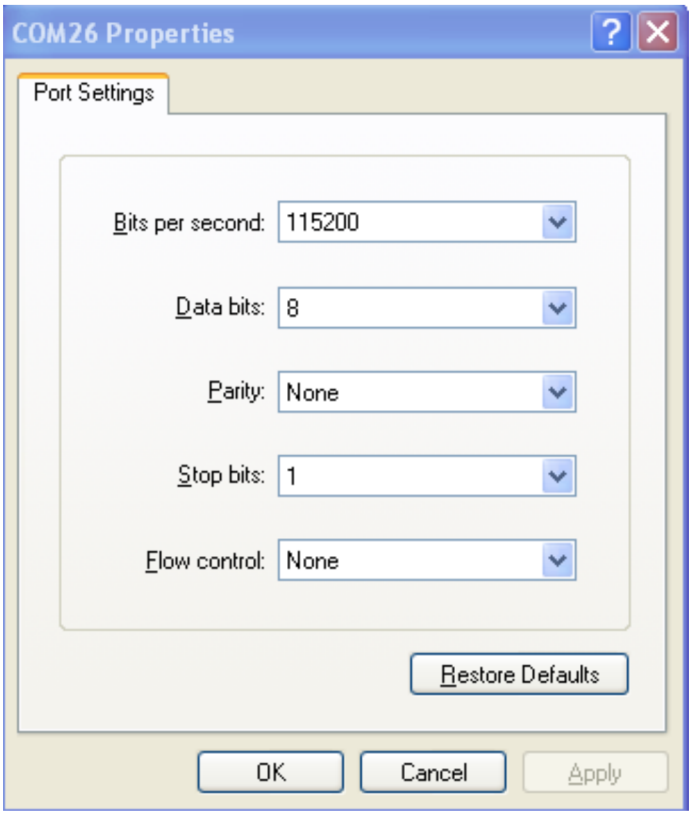

Figure 6. COM Port Properties

9 References

- *RM48Lx50 16/32-Bit RISC Flash Microcontroller Data Manual* (SPNS174)
- *F021 Flash API Version 2.00.01 Reference Guide* (SPNU501)

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

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

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

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