

How to Create A HALCoGen-Based Project For CCS

ABSTRACT

HALCoGen is a GUI-based initialization, configuration and driver code generator for TI's Hercules™ ARM® Safety MCUs. HALCoGen's editing feature provides a strong motivation to be able to regenerate source code and build/debug the source through Code Composer Studio™ (CCS) integrated development environment (IDE) in the same working folder.

Contents

1	Motivation	1
2	Creating a Code Composer Studio Project.....	3
3	Creating a HALCoGen Project	4
4	Generate the Code – Check the Code Composer Studio Project.....	6

List of Figures

1	Create a New Code Composer Studio Project	3
2	HALCoGen Startup With No Project	4
3	Creating a HALCoGen Project Sharing the Code Composer Studio Project Folder.....	5
4	Code Composer Studio View of Shared Project Folder - After HALCoGen Generation Step.....	6
5	Adding HALCoGen Include Folder to #include Search Path.....	7

1 Motivation

HALCoGen is able to maintain edits that you make to the generated source code, as long as these edits appear between comments *USER CODE BEGIN* and *USER CODE END*.

For example, by default HALCoGen generates the empty main() function (in file 'sys_main.c') as shown in [Example 1](#).

Example 1. The Function main() as Initially Generated by HALCoGen

```

/* USER CODE BEGIN (2) */
/* USER CODE END */

void main(void)
{
/* USER CODE BEGIN (3) */
/* USER CODE END */
}

```

Any code that you add to main() between the comments *USER CODE BEGIN (x)* and *USER CODE END* will be maintained even if you make another pass through HALCoGen and regenerate code after having changed various settings.

You can edit sys_main.c through the Code Composer Studio IDE, and add functionality.

Example 2. User Additions to main()

```

/* USER CODE BEGIN (2) */
#include <stdio.h>           /* Correct */
#include <gio.h>             /* Correct */
/* USER CODE END */
#include <het.h>             /* Incorrect */

void main(void)
{
    hetInit();              /* Incorrect */
/* USER CODE BEGIN (3) */
    gioInit();              /* Correct */
/* USER CODE END */
}
    
```

Suppose you build the code, load it into a Hercules TMS570 or RM4 device, start a debugging session, and then decide that you need to change one of the PLL settings in HALCoGen.

You can go back to the HALCoGen GUI, make changes to the PLL settings, and regenerate the code. Any code additions that you made in the areas marked for user additions will be preserved through the regeneration process. As [Example 3](#) illustrates, as long as you place the code in the correct location (between *USER CODE BEGIN* and *USER CODE END* comments); then the code changes are preserved through a cycle of regeneration.

Example 3. User Modified Code After Regeneration By HALCoGen

```

/* USER CODE BEGIN (2) */
#include <stdio.h>           /* Correct */
#include <gio.h>             /* Correct */
/* USER CODE END */

void main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();              /* Correct */
/* USER CODE END */
}
    
```

[Example 3](#) should provide motivation as to why one would want to have HALCoGen share a project directory with a Code Composer Studio project. Any updates made through HALCoGen would immediately appear in the Code Composer Studio project whenever the code is regenerated. And any edits made to the same files in the Code Composer Studio IDE will be retained through a cycle of HALCoGen regeneration as long as the code is added in areas specified for user code.

2 Creating a Code Composer Studio Project

This section explains how to configure both HALCoGen and Code Composer Studio to share the same project folder.

1. Create a project in Code Composer Studio.
2. Select File → New → CCS Project, from the Code Composer Studio menu bar, after starting the Code Composer Studio IDE. A window will open asking you where to create the Code Composer Studio project. In this case, create the project in the existing workspace, and name the project `halcogen_example` as shown in [Figure 1](#).
3. Create the project using the template of the appropriate RM, TMS570 device. In this case, settings for TMS570L20216SZWT are illustrated in [Figure 1](#).
 - (a) Select the Target
 - (b) Select the Emulator (XDS510USB Emulator)
 - (c) Create the project in the existing workspace (use default location), and name the project "halcogen_example"
 - (d) Select the Compiler version
 - (e) Leave Advanced settings to default. (Output type - Executable, Output format - eabi)
 - (f) Under Project Templates select "Empty Project"

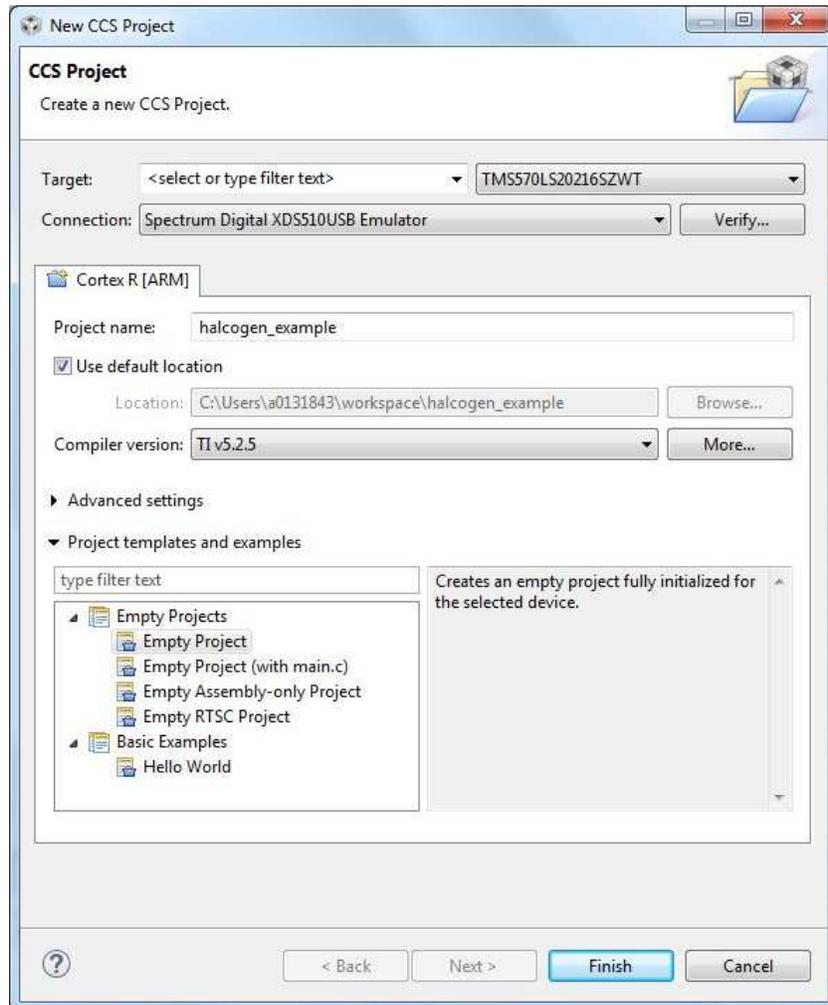


Figure 1. Create a New Code Composer Studio Project

4. Select Finish and a Code Composer Studio project will be created in the folder `.../workspace/halcogen_example`.

5. Create a HALCoGen project that will generate source files directly into this folder.

3 Creating a HALCoGen Project

Once the Code Composer Studio project has been created, it is time to create a HALCoGen project that generates code directly into the Code Composer Studio Project directory.

1. Start the HALCoGen program, which will bring up a startup screen as illustrated in [Figure 2](#).

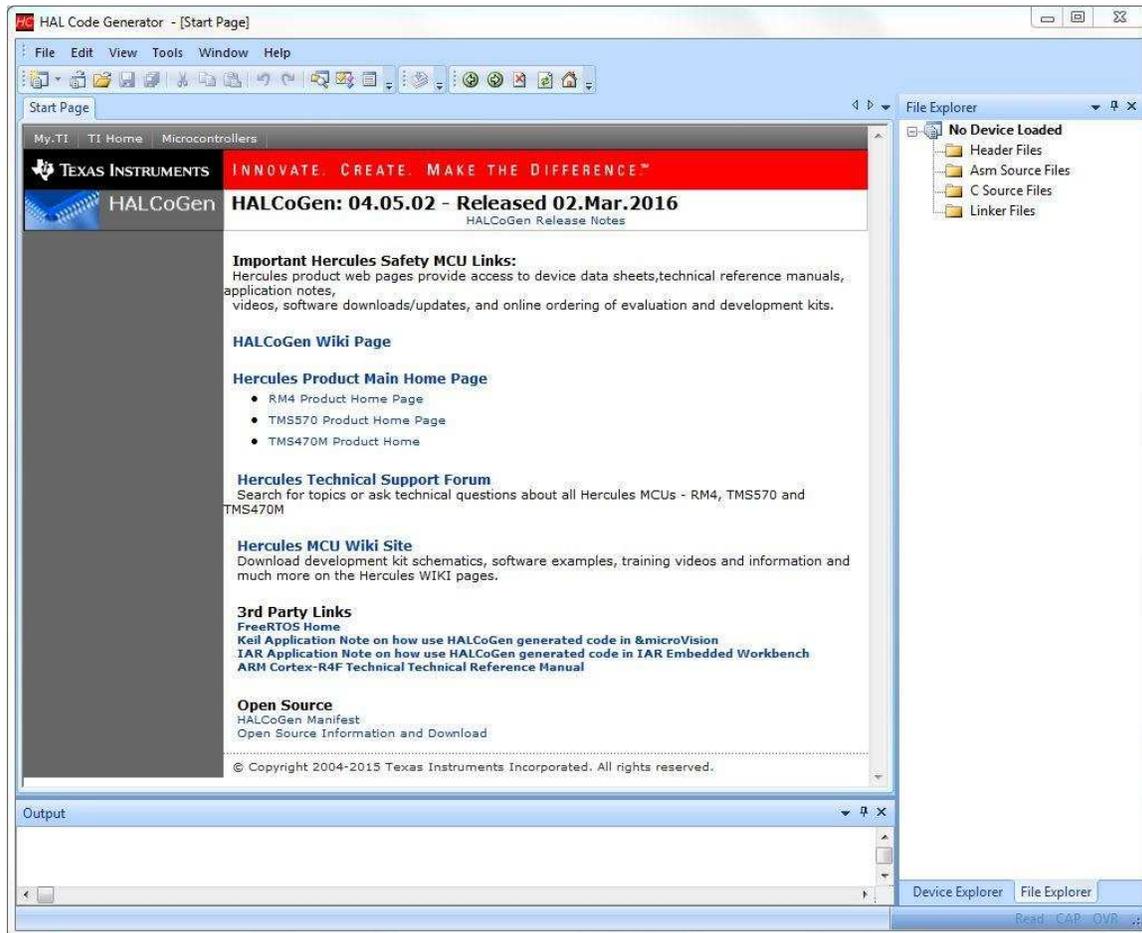


Figure 2. HALCoGen Startup With No Project

2. Select File → New → Project from the HALCoGen startup screen. A window appears asking you to select a device type, a working folder, and to select a name for the project.

[Figure 3](#) shows how you would create the new HALCoGen project so that code is generated directly into the Code Composer Studio project directory. In this case, the Code Composer Studio project directory was `.../workspace/halcodegen_example`.

The corresponding HALCoGen project should be created:

- With the same TMS570 device as was selected when creating the Code Composer Studio Project
- With any 'name' – although in this case the HALCoGen project name was chosen to match the Code Composer Studio Project name `halcodegen_example`.
- With location matching the top level Code Composer Studio Project Folder: `.../workspace/halcodegen_example`.
- With *Create directory for project* unchecked

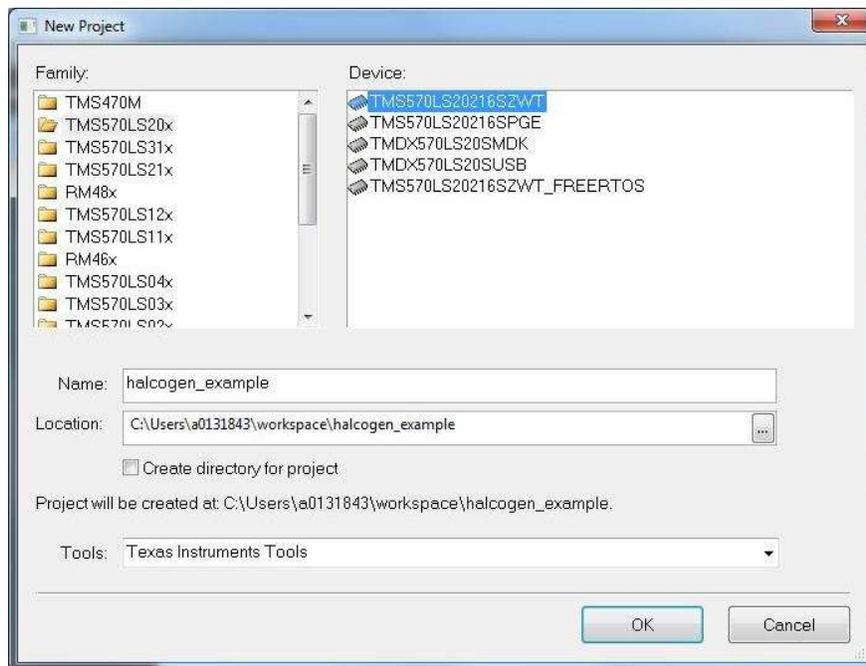


Figure 3. Creating a HALCoGen Project Sharing the Code Composer Studio Project Folder

That's it! Now you have a pair of HALCoGen and Code Composer Studio projects that are set up to allow rapid iteration between the environments.

4 Generate the Code – Check the Code Composer Studio Project

At this point, you should be able to generate code from HALCoGen (you can try this immediately after creating the project, without changing any of the default settings.)

If everything is setup correctly, when you look at the corresponding Code Composer Studio project, you will see that all of the source files (Source and Include) generated by HALCoGen have automatically been added to the Code Composer Studio project, as shown in the left hand pane of the Code Composer Studio project view (Figure 4)

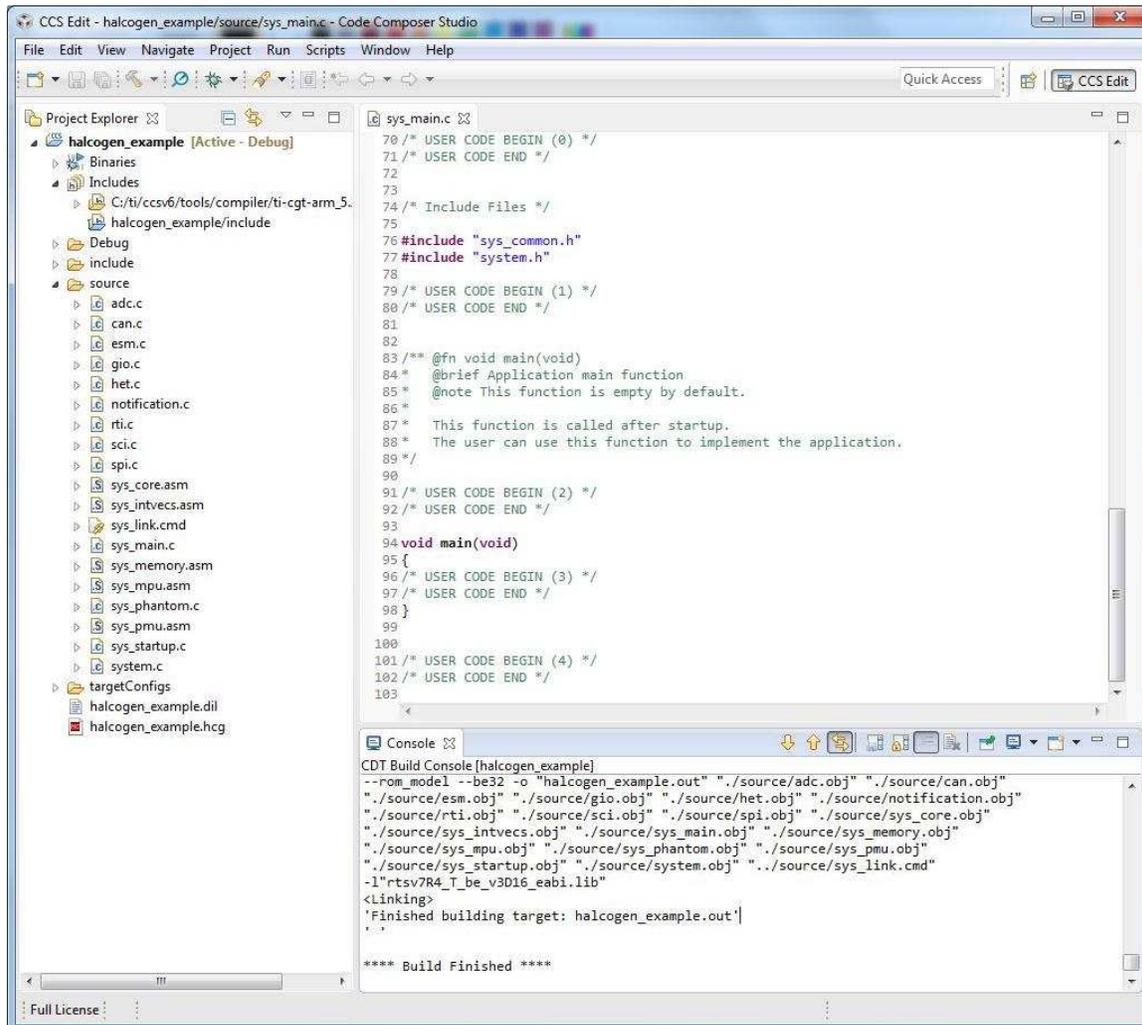


Figure 4. Code Composer Studio View of Shared Project Folder - After HALCoGen Generation Step

Add the HALCoGen generated Include folder to the #include search path under the compiler Include Options as illustrated in the [Figure 5](#).

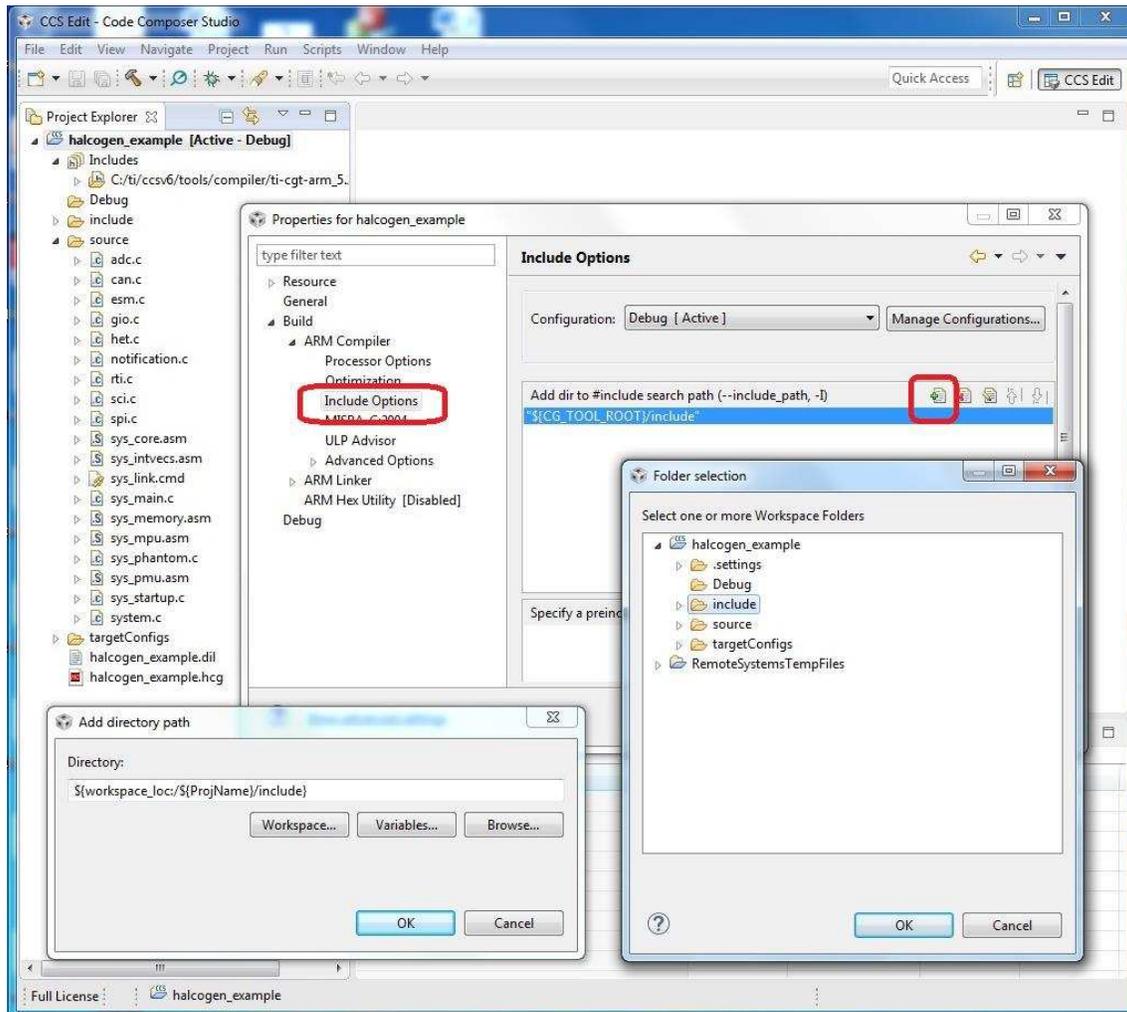


Figure 5. Adding HALCoGen Include Folder to #include Search Path

Now you will be able to make changes to the same project through either the HALCoGen environment or through the Code Composer Studio editor. And you will be able to regenerate code without losing your edits in the Code Composer Studio environment, as long as they are placed in the designated user code areas.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from A Revision (September 2011) to B Revision	Page
• Information was updated in Section 2 .	3
• Information was updated in Section 3 .	4
• Information was updated in Section 4 .	6

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

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com