Application Report
SPRACA1–July 2017

Software Guidelines to EMIF/DDR3 Configuration on DRA7xx Devices

Ravi B

ABSTRACT

This application report provides information on EMIF/DDR configuration on dra7xx devices and guidelines to changes to be made in SPL/U-boot source code.

Contents

List of Figures

Trademarks

ARM is a registered trademark of ARM Lmited.
All other trademarks are the property of their respective owners.

1 Introduction

This scope of this document is to provide guidelines to configuration of the EMIF controller in SPL/U-boot for dra7xx custom platform for DDR memory part.

- The document focuses on guidelines that need to be done or are necessary changes in SPL/U-boot to configure the EMIF controller for DDR memory for the custom board.
- The document does not discuss how to obtain the timing parameters for the specific DDR part.

1.1 Software Requirements

This document is based on Processor SDK Linux Automotive 3.02, the u-boot version 2016.05 used as reference.

- Have a working Processor SDK Linux Automotive 3.02 installation.
- Are able to build U-boot and kernel

The Kernel and U-Boot commits corresponding to the 3.02 SDK are shown in Table 1.

Table 1. kernel/u-boot Release Commit Details

Repository	Commit id	Headline
Kernel	89944627d53a	Late Attach: Fix for accessing second level page table
U-Boot	850ffc07orba	defconfigs: dra7xx_hs_evm: Move OPTEE load address to avoid overlaps

The release downloads links and software developers guide can be found at: Category:Processor SDK Linux Automotive.

1.2 *Hardware Requirements*

The evaluation board is used for reference:

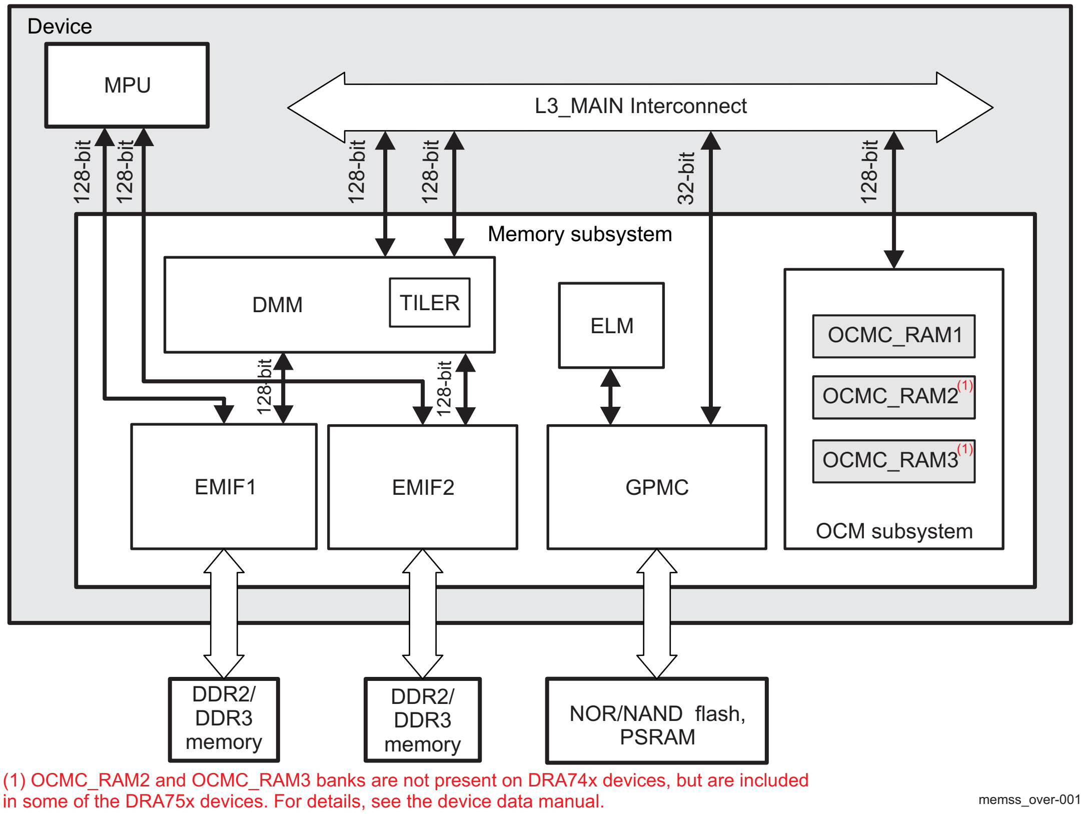
- J6 EVM Rev-H

1.3 *Build Instructions*

Building MLO and u-boot

2 Dra7xx Memory Subsystem

Figure 1 shows the memory subsystem of dra7xx SoC, which includes the DMM, EMIF1 and EMIF2 controller, OCMC RAMs and GPMC controller.



(1) OCMC_RAM2 and OCMC_RAM3 banks are not present on DRA74x devices, but are included in some of the DRA75x devices. For details, see the device data manual.

Figure 1. Memory Subsystem Architecture

The DRA7xx includes two EMIF controller EMIF0 and EMIF1, each EMIF controller provides the connectivity to DDR2/3 type of memories and manages data bus read/write access between external memories and the device subsystem that have access to the L3_MAIN interconnect and DMA capability.

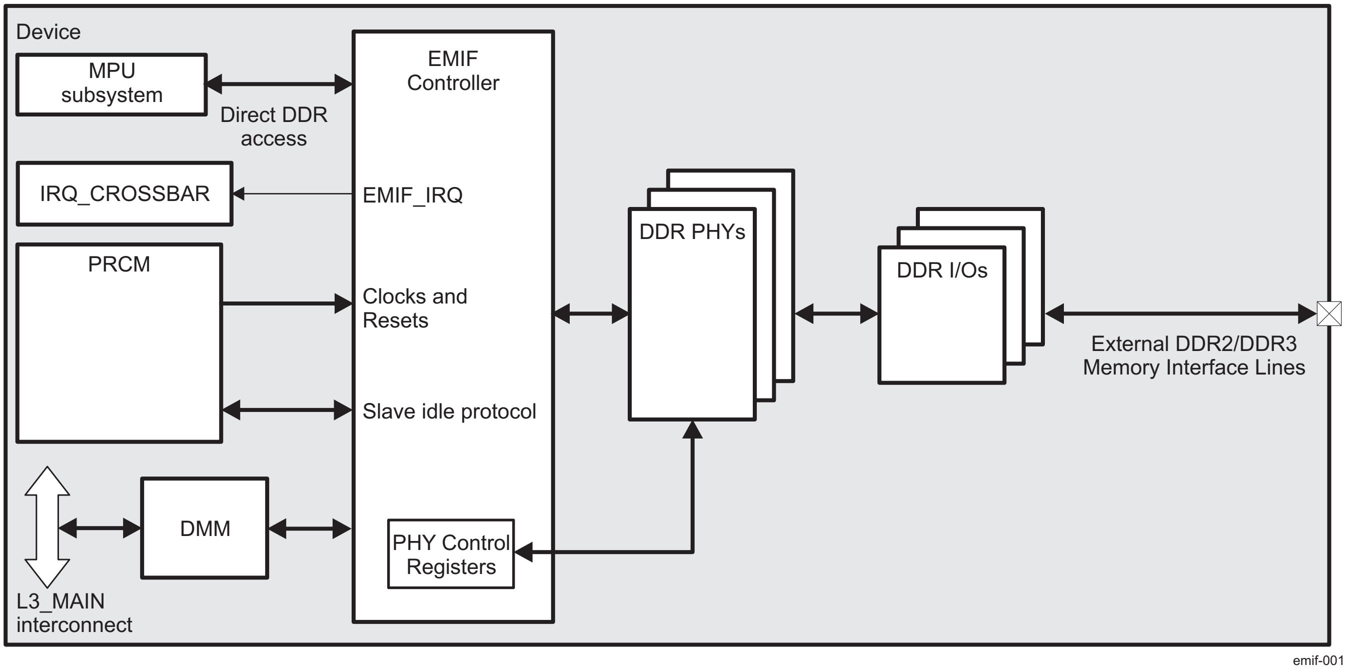


Figure 2. EMIF Controller

3 Software Guidelines to EMIF DDR Configuration

Section 3.1 through Section 3.2 details the step-by-step procedure and changes that need to be done in the SPL/U-boot code for EMIF/DDR configuration. The SPL/U-boot version 2016.05 is used as reference.

Note that on U-Boot version 2017.01 onwards, the location of startup code for omap/dra7xx platforms has been moved from arch/arm/cpu/armv7/ to arch/arm/mach-omp2/ directory.

The DRA75x reference board used here is populated with 4GB memory, where 2GB DDR interfaced on each EMIF1 and EMIF2 with interleaved memory with DDR base 0x8000-0000 and 16GB memory is reserved to trap the unmapped Tiler address. The LPAE support enables to use higher 2GB DDR memory in kernel. The higher 2GB address starts at 0x2_0000_0000.

3.1 *DDR3 Timing Parameter Generations Using EMIF Configuration Tools*

The first step is to generate the DDR3 timing parameters by using the EMIF configuration tools. For step-by-step procedures for the EMIF configuration tool, see https://cdds.ext.ti.com/ematrix/common/emxTree.jsp?objectId=28670.42872.1536.30960&fromContent=true.

- Provide the custom board details: Soc Part number, SYS_CLK1 frequency.
- Number of EMIF interfaces, DDR type, DDR frequency, DDR bus width per EMIF.
- Use TI recommended values for DDR and EMIF memory I/O settings (termination/output driver impedance).
- Provide the numerical values listed in your device-specific DDR data sheet for all timing parameters (like CAS, CWL latencies, tRP, tRCD, and so forth).

After entering all of the above fields, as specified in the EMIF tool, the tool generates the timing parameters in a separate sheet under “Register values (u-boot)” in ‘C’ structures" as shown in the following example code:

```
const struct dpll_params DRA75x_DDR3L_532MHz_TI_EVM_revG3_pll_params = {
.m = 266,
.n = 4,
.m2 = 2,
.m4_h11 = 8
};

const struct ctrl_ioregs DRA75x_DDR3L_532MHz_TI_EVM_revG3_ctrl_ioregs = {
.ctrl_ddr3ch = 0x80808080,
.ctrl_ddrch = 0x40404040,
.ctrl_ddrio_0 = 0x00094A40,
.ctrl_ddrio_1 = 0x04A52000,
.ctrl_emif_sdram_config_ext = 0x0000C123

};

const struct dmm_lisa_map_regs DRA75x_DDR3L_532MHz_TI_EVM_revG3_dmm_regs = {
.dmm_lisa_map_0 = 0x00000000,
.dmm_lisa_map_1 = 0x00000000,
.dmm_lisa_map_2 = 0x80640300,
.dmm_lisa_map_3 = 0xFF020100,
.is_ma_present = 0x1
};

const struct emif_regs DRA75x_DDR3L_532MHz_TI_EVM_revG3_emif_regs = {
.sdram_config_init = 0x61851AB2,
.sdram_config = 0x61851AB2,
.sdram_config2 = 0x00000000,
.ref_ctrl = 0x000040F1,
.ref_ctrl_final = 0x00001035,
.sdram_tim1 = 0xCCCF36B3,
.sdram_tim2 = 0x305A7FDA,
.sdram_tim3 = 0x407F8558,
.read_idle_ctrl = 0x00050000,
.zq_config = 0x5007190B,
.temp_alert_config = 0x00000000,
.emif_rd_wr_lvl_rmp_ctl = 0x80000000,
.emif_rd_wr_lvl_ctl = 0x00000000,
.emif_ddr_phy_ctlr_1_init = 0x0024400B,

.emif_ddr_phy_ctlr_1 = 0x0E24400B,
.emif_rd_wr_exec_thresh = 0x00000305

};

/*
* DLL Ratio Values are an estimate based on trace lengths. Either
* software leveling or hardware leveling should be performed to
* determine final DLL values.
*/
const unsigned int DRA75x_DDR3L_532MHz_TI_EVM_revG3_emif1_ext_phy_regs [] = {    ,
// EMIF1_EXT_PHY_CTRL_1
..    // EMIF1_EXT_PHY_CTRL_36
};

const unsigned int DRA75x_DDR3L_532MHz_TI_EVM_revG3_emif2_ext_phy_regs [] = {
,    // EMIF2_EXT_PHY_CTRL_1 to
// EMIF2_EXT_PHY_CTRL_36
};
```

3.2 Update the DDR3 Timing Parameters in u-boot C ode

Section 3.2.1 through Section 3.2.4 explains the changes that need to be done for APIs used by the SDRAM initialization routine in SPL/U-Boot. The SDRAM initialization sequence is explained in Section 4.1.

3.2.1 Update the EMIF1/2 DDR3 Timing Parameters

The EVM or custom board specific initialization APIs are implemented in board/ti/dra7xx/evm.c. Copy the respective DDR3 EMIF1 and EMIF2 timing parameter data from the EMIF configuration tool (see the xls sheets in EMIF configuration tool) to the EMIF register structures are shown as follows:

```
 };

    const struct emif_regs emif1_ddr3_532_mhz_1cs_2G = {
             .sdram_config_init              = 0x61851ab2,
             .sdram_config                   = 0x61851ab2,
             .sdram_config2                  = 0x08000000,
             .ref_ctrl                       = 0x000040F1,
             .ref_ctrl_final                 = 0x00001035,
             .sdram_tim1                     = 0xCCCF36B3,
             .sdram_tim2                     = 0x30BF7FDA,
             .sdram_tim3                     = 0x427F8BA8,
             .read_idle_ctrl                 = 0x00050000,
             .zq_config                      = 0x0007190B,
             .temp_alert_config              = 0x00000000,
             .emif_ddr_phy_ctlr_1_init       = 0x0024400B,
             .emif_ddr_phy_ctlr_1            = 0x0E24400B,
             .emif_ddr_ext_phy_ctrl_1        = 0x10040100,
             .emif_ddr_ext_phy_ctrl_2        = 0x00910091,
             .emif_ddr_ext_phy_ctrl_3        = 0x00950095,
             .emif_ddr_ext_phy_ctrl_4        = 0x009B009B,
             .emif_ddr_ext_phy_ctrl_5        = 0x009E009E,
             .emif_rd_wr_lvl_rmp_win         = 0x00000000,
             .emif_rd_wr_lvl_rmp_ctl         = 0x80000000,
             .emif_rd_wr_lvl_ctl             = 0x00000000,
             .emif_rd_wr_exec_thresh         = 0x00000305
     };

     const struct emif_regs emif2_ddr3_532_mhz_1cs_2G = {
             .sdram_config_init              = 0x61851B32,
             .sdram_config                   = 0x61851B32,
             .sdram_config2                  = 0x08000000,
             .ref_ctrl                       = 0x000040F1,
             .ref_ctrl_final                 = 0x00001035,
             .sdram_tim1                     = 0xCCCF36B3,
             .sdram_tim2                     = 0x308F7FDA,
             .sdram_tim3                     = 0x427F88A8,
             .read_idle_ctrl                 = 0x00050000,
             .zq_config                      = 0x0007190B,
             .temp_alert_config              = 0x00000000,
             .emif_ddr_phy_ctlr_1_init       = 0x0024400B,
             .emif_ddr_phy_ctlr_1            = 0x0E24400B,
             .emif_ddr_ext_phy_ctrl_1        = 0x10040100,
             .emif_ddr_ext_phy_ctrl_2        = 0x00910091,
             .emif_ddr_ext_phy_ctrl_3        = 0x00950095,
             .emif_ddr_ext_phy_ctrl_4        = 0x009B009B,
             .emif_ddr_ext_phy_ctrl_5        = 0x009E009E,
             .emif_rd_wr_lvl_rmp_win         = 0x00000000,
             .emif_rd_wr_lvl_rmp_ctl         = 0x80000000,
             .emif_rd_wr_lvl_ctl             = 0x00000000,
             .emif_rd_wr_exec_thresh         = 0x00000305
    };
```

Update the emif_get_reg_dump(u32 emif_nr, const struct emif_regs **regs) function in board/ti/dra7xx/evm.c, which returns emif1 or emif2 timing parameters based on input emif_nr based on SoC/board revision. This function is called by the *sdram_init()* function.

3.2.2 DDR3 Address Mapping Through DMM

The address mapping inside the DMM is configurable up to four sections. A DMM section is configured through DMM_LISA_MAP_i registers. Each section is based on system address, decoding range for the section, section size, physical address, single/pair of emif controller configured and memory interleave option. In this example, both EMIF1 and EMIF2 are used and 2GB DDR is interfaced to each EMIF controller.

The following structure shows the lisa_map configuration, defined in source file board/dra7xx/ti/evm.c. In this example, DRA75x EVM has a total of 4G memory: 2GB on each EMIF1 and EMIF2 interface and interleaved. The 2GB DDR base address starts from 0x8000-0000, the higher 2GB address (0x2-0000-0000) for user space that can be used with the ARM® LPAE feature support. Copy the lisa_map register values from the EMIF configuration tool as shown below:

```
const struct dmm_lisa_map_regs lisa_map_dra7_2GB = {
         .dmm_lisa_map_0 = 0x0,
         .dmm_lisa_map_1 = 0x0,
         .dmm_lisa_map_2 = 0x80740300,
         .dmm_lisa_map_3 = 0xFF020100,
         .is_ma_presetn = 0x1
```

The emif_get_dmm_regs(const struct dmm_lisa_map_regs **dmm_lisa_regs) defined in evm.c returns the lisa_map configuration is based on the SoC/board revision. This function is called during sdram initialization.

3.2.3 Configure DDR I/O Cells

Configure the software controls for the DDR3 IO cells associated with the DDR3 interface. For more information, see the control module section of the device-specific technical reference manual (TRM). Copy the IO cell configuration parameters from the EMIF configuration tool as shown in the structure below, which is defined in arch/arm/cpu/armv7/omap5/hw_init.c:

```
const struct ctrl_ioregs ioregs_dra7xx_es1 = {
         .ctrl_ddrch = 0x40404040,
         .ctrl_lpddr2ch = 0x40404040,
         .ctrl_ddr3ch = 0x80808080,
         .ctrl_ddrio_0 = 0x00094A40,
         .ctrl_ddrio_1 = 0x04A52000,
         .ctrl_ddrio_2 = 0x84210000,
         .ctrl_emif_sdram_config_ext = 0x0001C1A7,
         .ctrl_emif_sdram_config_ext_final = 0x0001C1A7,
         .ctrl_ddr_ctrl_ext_0 = 0xA2000000,
```

Update the get_ioregs(const struct ctrl_ioresg **regs) function in hw_init.c to return the respective DDR I/O cell values based on the Soc/board revision.

3.2.4 Update the DDR3 Software Leveling Values

This section is NOT required if hardware leveling is used. It is recommended to use DDR hardware leveling. This section can be skipped if software leveling is not used.

The EMIF configuration tool generates the DDR phy timings parameters for software leveling for EMIF1 and EMIF2; copy the values into the respective structure defined in source file arch/arm/cpu/armv7/omap5/sdram.c.

```
const u32 dra_ddr3_ext_phy_ctrl_const_base_es1_emif[] = 1{}
        0x10040100,
        0x00910091,
        ...
};
const u32 dra_ddr3_ext_phy_ctrl_const_base_es1_emif[2] = {
        0x10040100,
        0x00910091,
        ...
```

The emif_get_ext_phy_ctrl_const_regs(u32 emif_nr, const u32 **regs, u32 *size) function returns the ddr phy control values for EMIF1/2 based on silicon revision. This function is called during *sdram_init()*.

4 SDRAM Initialization in MLO/U-Boot

This section briefly discusses the MLO/u-boot initialization sequence followed by the SDRAM initialization.

On power-on-reset of dra7xx, the ROM resident code (based on the boot mode of the SYSBOOT pin setting) loads the initial bootloader (often referred as SPL or MLO from specific boot media (mmc/sd, emmc, QSPI, PC Host for USB-SPLDFU)) into the internal OCMC RAM and executes the SPL code from the internal RAM. The SPL further brings up the System-on-Chip (SoC) by enabling the required clocks and performing the DDR initialization based on the memory configuration of the custom board.

The startup code for SPL is at arch/arm/cpu/armv7/reset.S, which disables the cache, invalidating the L1 instruction/data cache, TLBs, disables MMU and calls low-level initialization (arch/arm/cpu/armv7/lowlevel_init.S) and the *invokes _main()* function at arch/arm/lib/crt0.S. This sets up the stack and invokes *board_init_f()* at arch/arm/cpu/armv7/omap-common/hwinitcommon. c, to perform early system initialization of pin muxes, enables the required clocks and sets up the memory (SDRAM).

Note that on U-Boot version 2017.01 onwards, the location of startup code for omap/dra7xx platforms has been moved from arch/arm/cpu/armv7/ to arch/arm/mach-omp2/ directory.

4.1 SDRAM Initialization

The *sdram_init()* function (defined in source omap-common/hwinit-common.c) performs DDR memory initialization. The *sdram_init()* is done once during execution of MLO (when the code is running from OCMC internal memory).

The *omap_sdram_size()* function computes the total memory size configured parsing through lisa_map register. The 16MB memory configured in section-4 of lisa-map is reserved to trap the unmapped Tiler entries, hence, the *omap_sdram_size()* returns the total DDR size minus 16MB, which is passed to kernel from u-boot through the device tree nodes. For more information on lisa-map, see the DMM component in the device-specific Technical Reference Manual (TRM).

The SDRAM initialization sequence (see arch/arm/cpu/armv7/omap-common/emif-common.c) is shown below:

1. Do the *sdrma_init()* only if code is running from OCMC internal memory.
   (a) //DMM and EMIF initialization
      (i) Reset the emif phy and perform ddr_phy_init.
      (ii) Get the lisa_map_i configuration details using *emif_get_dmm_register()*.
      (iii) Configure the dmm_lisa_map0/1/2/3 registers.
   (b) // DDR initialization
      - If emif1_enabled, then perform do_sdram_init(EMIF1_BASE).
      - If emif2_enabled, then perform do_sdram_init(EMIF2_BASE).

2. Get the configured SDRAM size (omap_sdram_size()) and compare it with the programmed size; if the size does not match then:
   (a) SDRAM: identified size not same as expected size identified: xx expected: yy".

4.2 Build u-boot and Verify

Build the u-boot source and copy the MLO, u-boot.img to boot partition mmc/sd card. Set the EVM to MMC/SD boot mode setting and reset the EVM. You should get the u-boot prompt.

To download the source and build procedures, see the Processor SDK Linux Automotive Software Developers Guide wiki.

4.3 Perform Memtester From Kernel

From kernel, you can verify DDR by running memtester from kernel. The memtester utility from kernel is used to check integrity and functionality of the DDR by performing various memory tests like address path test, data path test, stuck addres/data tests, and so forth.

Syntax : memtester [-p PHYSADDR] <Memory> [ITERATIONS]

```
root@dra7xx-evm# memtester 3500M
```

5 References

- *DRA72x (SR2.0, SR1.0) and DRA71x (SR2.0) SoC for Automotive Infotainment Technical Reference Manual*
- U-Boot 2016.05 source code reference
- EMIF Configuration Tool

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI”) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated