*TI Designs*

# *EtherCAT® Master Reference Design for Sitara™ AM57x Gigabit Ethernet and PRU-ICSS with Time-Triggered Send*

## TI Designs

This TI Design details an EtherCAT® (EC) Master interface operating on the Sitara™ AM572x processor using the EC-Master stack from acontis. This EtherCAT Master solution can be used for EtherCAT-based PLC or motion control applications. EtherCAT Master is profiled on the Gigabit Ethernet ports and the PRU-ICSS Ethernet ports of the AM572x processor to give designers flexibility to use any of the two Gigabit Ethernet ports or four PRU-ICSS Ethernet ports on the device. The EtherCAT Master implementation can achieve less than 100-µs cycle times for both the gigabit and the PRU-ICSS Ethernet ports. Time-triggered send (TTS) can be enabled on the PRU-ICSS to reduce jitter, achieve shorter cycle times, and reduce latency in cases where distributed clocking is not used.

## Design Resources

| | |
|---|---|
| TIDEP0079 | Tools Folder |
| TMDXIDK5728 | Tools Folder |
| TMDSIDK437x | Tools Folder |
| AM572x | Product Folder |

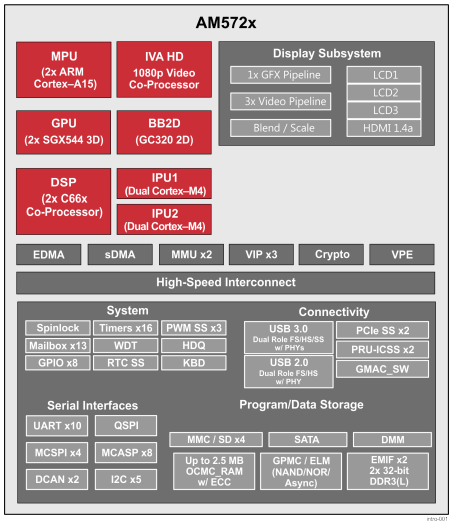
TI E2E™ Community — ASK Our E2E Experts

## Design Features

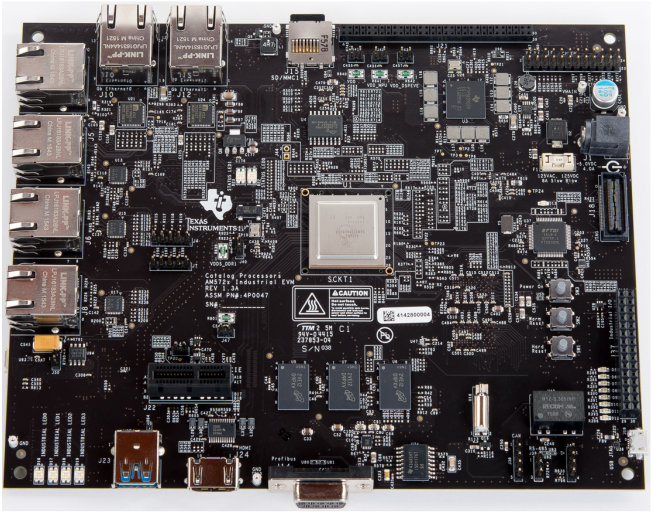
- EtherCAT Class A or Class B Master Stack According to ETG.1500 Specification
- High-Performance Ethernet Driver for Maximum EtherCAT Performance
- EtherCAT Feature: Pack Cable Redundancy Using Two GMAC Ports or Two ICSS_PRU Ports
- EtherCAT Feature: Pack Hot Connect to Support Flexible Configuration
- PRU-ICSS EMAC Feature: Time-Triggered Send
- Supported Operating System: TI-RTOS

## Featured Applications

- EtherCAT Programmable Logic Controller (PLC) System
- EtherCAT Motion Control Application
- EtherCAT Interface Boards
- EtherCAT Industrial Communication Gateways







## 1 Introduction

This TI Design document presents the TI Sitara AM572x implementing an EtherCAT Master using acontis EtherCAT Master stack (EC-Master). The acontis EtherCAT Master stack is a highly portable software stack, and when combined with a high-performance TI Sitara CPU it provides a sophisticated EtherCAT solution that users can use to implement EtherCAT communication-interface boards, EtherCAT-based PLC, or EtherCAT-based motion-control applications.

The EC-Master architectural design does not require additional tasks, making it easier to transport code to a different OS or to bare-metal systems. Because of this architecture and the high-speed Ethernet driver, it is possible to implement applications on AM572x platform with cycle times of less than 100 µs.

## 2 EtherCAT® Protocol

EtherCAT is an IEEE 802.3 Ethernet-based fieldbus system. EtherCAT is a new standard in communication speed. Because EtherCAT is inexpensive to implement, the system can use fieldbus technology in applications which previously omitted fieldbus. EtherCAT is an open technology that is standardized within the International Electrotechnical Commission (IEC). The technology is supported and powered by the EtherCAT Technology Group (an international community of users and vendors). The protocol is suitable for both hard and soft real-time requirements in automation technology. A primary advantage of EtherCAT is that it supports automation applications that require short data-update times with low communication jitter and reduced hardware costs.

In the EtherCAT protocol, the EtherCAT Master sends a telegram that passes through each node. Each EtherCAT Slave device reads the data that is addressed to it as soon as the data is detected. Then, the slave device inserts the data into the frame as the frame moves downstream. The frame is delayed by hardware-propagation delay times. The last node in a segment (or branch) detects an open port and sends the message back to the master using the full-duplex feature of Ethernet technology. The EtherCAT Master is the only node within a segment that actively sends an EtherCAT frame. All other nodes forward the frames downstream. This capability permits the network to achieve over 90% of the available network bandwidth, prevents unpredictable delays, and guarantees real-time system response.

The EtherCAT protocol is optimized for process data transfer and is transported within the Ethernet frame by using a special Ethertype identifier (0x88A4). EtherCAT communications consist of several EtherCAT telegrams. Each telegram serves a specific memory area of the logical process image, up to 4GB. The data sequence is independent of the physical order of the Ethernet terminals in the network.

In addition to data exchanges between the EtherCAT Master and Slave, EtherCAT is also suitable for communication between controllers (master to master). Freely addressable network variables for process data and a variety of services for parameterization, diagnosis, programming, and remote control cover a range of requirements. The data interfaces for master-to-slave and master-to-master communication are identical.

Two mechanisms are available for slave-to-slave communication. The first option is to use upstream devices to quickly communicate to downstream devices within the same cycle. The other option is using the freely configurable slave-to-slave communication that runs through the master device. The second option requires two bus cycles (although not necessarily two control cycles).

# 3 Block Diagram

## 3.1 TI Sitara™ AM572x Overview

The Sitara AM572x brings high processing performance through the maximum flexibility of a mixed processor solution that is fully integrated. The devices also combine programmable video processing with a broad and highly integrated peripheral set which is well suitable for industrial applications.

Programmability is provided by dual-core ARM® Cortex®-A15 RISC CPUs with ARM® NEON™ technology, and two TI C66x VLIW floating-point DSP cores. The ARM Cortex lets developers keep separate control functions from other algorithms that are programmed on the DSPs and coprocessors. The separated control functions reduce the complexity of the system software. The ARM Cortex-A15 CPU supports multiple operating frequencies at a range of up to 1.5GHz. The AM572x processor is configured with two dual-core Programmable Real-Time Unit and Industrial Communication Subsystems (PRU-ICSS). The PRU-ICSS can be used for communication protocols such as EtherCAT Master and Slave, PROFINET, Ethernet/IP, SERCOS, and so forth. This TI design shows two implementations of the acontis EC-Master: one operates by using Gigabit Ethernet Media Access Controller (GMAC) ports, the other operates by using Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) ports and Ethernet Media Access Controller (EMAC) ports. These options let users to select which type of EMAC to use when running an EtherCAT Master.

Figure 1 shows the functional block diagram.

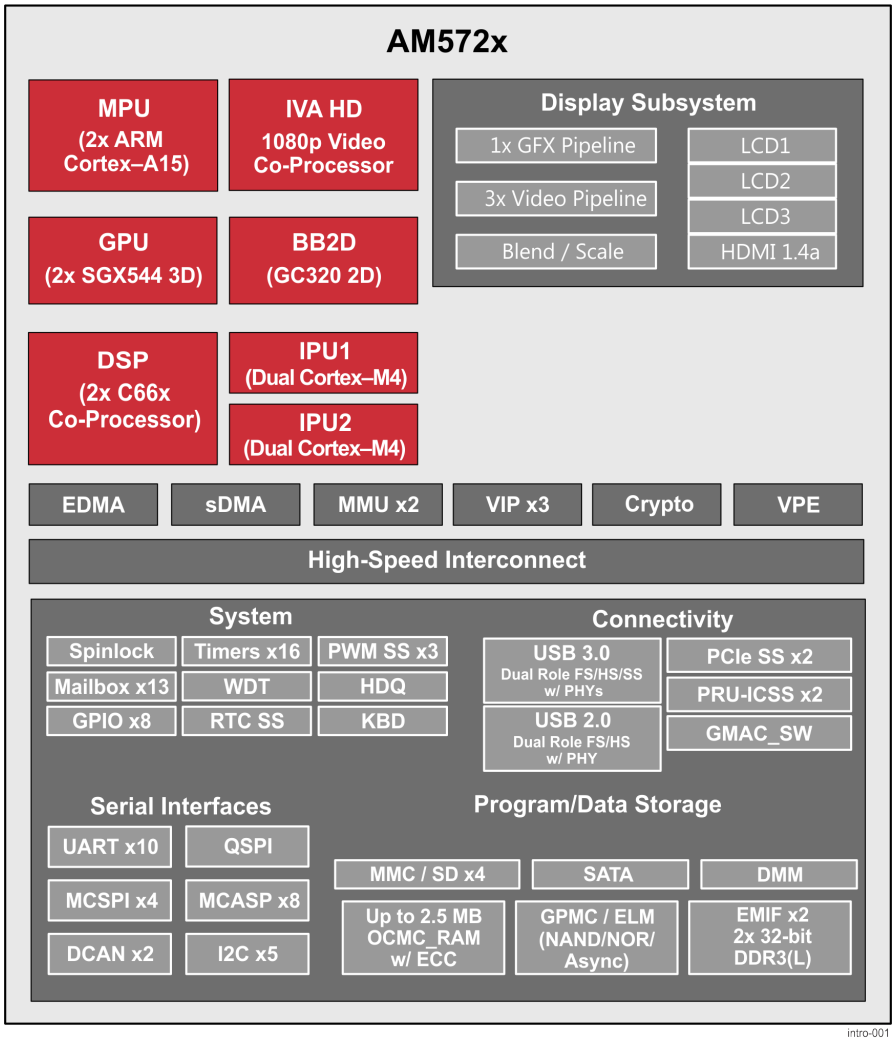




**Figure 1. AM572x Block Diagram**

# 4 acontis EtherCAT® Master Architecture

Figure 2 shows the module architecture for the acontis EC-Master. The EC-Master stack is divided into the following layers:

- EtherCAT Master Core
  - In the core module, cyclic (process data update) and acyclic (mailbox) EtherCAT commands are sent and received.
- Configuration Layer
  - The EtherCAT Master is configured using an XML file whose format is fixed in the EtherCAT specification ETG.2100. The EC-Master contains an OS-independent XML parser.
- Ethernet Link Layer
  - This layer exchanges Ethernet frames between the master and the slave devices.
- OS Layer
  - All OS-dependent system calls are encapsulated in a small OS layer.

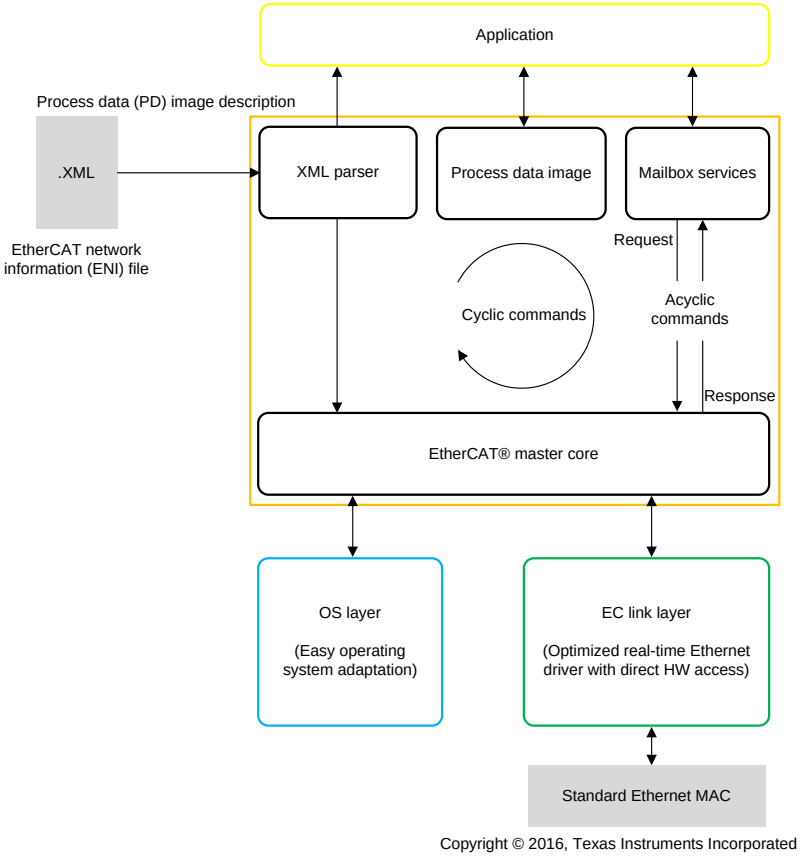



**Figure 2. acontis EC-Master Component Model Architecture**

# 5 Getting Started

## 5.1 Hardware

The following list shows the hardware required to support this design.

- TMDXIDK5728
- EtherCAT Slave device (TMDSIDK437x)

**NOTE:** This TI Design was tested using the TMDSIDK437x as the EtherCAT slave.

- PC with terminal connection (for example, Tera Term through USB)
- Windows® operating system PC with a minimum of 2-GB RAM

## 5.2 Software

The following list shows the software required to support this design.

- acontis EC-Master V2.9 for SYS/BIOS
- AM57x SYS/BIOS Processor SDK v3.1.0.6
- Code Composer Studio™ (CCS) 6.1.3 Compiler GNU v4.8.4 or higher
- SYS/BIOS 6.45.01.29
- XDC v3.32.00.06_core
- AM572x PDK v1.00.04
- Network development kit (NDK) 2.24.03.35
- Serial console terminal application (for example, Tera Term, minicom, and HyperTerminal)

  If using the AM437x IDK device as a slave, download Industrial SDK 2.1.0.1 prebuilt binaries

# 6 Preparing the Application

This TI Design was tested using the AM437x as an EtherCAT slave. To use the AM437x as a slave device, refer to Appendix A.

The TI Design prebuilt software packages include the ENI file for a topology where the AM437x is the only slave device that is connected. If the user switches to a different slave-bus topology or wants to recreate the ENI files for the AM437x, users can use the acontis EC-Engineer tool. Users can also use the acontis EC-Engineer tool to create an ENI file. To create an ENI file with the EC-Engineer Tool, refer to Appendix B.

1. Install AM57x Processor SDK RTOS v3.1.0.6 on the host PC
2. Unzip the file at EC_Master_Sysbios_SDK_Eval
3. Consider and select one of the following:
   (a) If the slave bus is the AM437x, copy the MasterENI.c file from the prebuilt package and paste inside <EC-master_installation_path>*Workspace**SYSBIOS**AM57xx*
   (b) If the slave bus is not the AM437x, convert the ENI file (eni.xml) to a C file with array
      (i) Obtain the eni.xml file by following the instructions in Appendix B
      (ii) Download and install Industrial SDK

(iii) Use the Industrial IDK converter tool found in </ISDK_installation_path>*sdk**tools**bin2header* to convert the eni.xml file

**NOTE:** The following are example parameters to use with bin2header.exe:

bin2header.exe eni.xml MasterENI.c MasterENI_xml_data

Add the file size information at the end of the new MasterENI.c file. See the following code example.

```
unsigned int MasterENI_xml_data_size = 16426;
```

The file size prints in the console.

4. Open CCS and import the EC-Master demonstration project
   (a) Navigate to *File→Import→CCS Projects*
   (b) Navigate to the installation directory: <ECmaster_installation_path>*Acontis_EC_master**EC_Master_Sysbios_SDK_Eval**Workspace**SYSBIOS**EcMasterDemo*
   (c) Click OK, and then click Finish
   (d) Check that SYSBIOS, XDC, and compiler versions are correct in the CCS Project Properties
   (e) Click the Clean Project option
   (f) Click the Build Project option

The EcMasterDemo.out output application is in the Debug Folder or Release Folder after building the project.

**NOTE:** Users can load and run EcMasterDemo.out from the prebuilt package.

To change the hardcoded parameters for the demonstration, use DEMO_PARAMETERS in ATEMDemoConfig.h. If the LINKLAYER_ICSS macro is defined in *Project Properties→Symbols*, the demo will run on ICSS_PRU2 ports. If the LINKLAYER_ICSS macro is not defined in *Project Properties→Symbols*, the demo will run on the Gigabit Ethernet port of the board (GMAC). See the following snippet of code.

```
#if defined SOC_AM572x
  #if defined LINKLAYER_ICSS
      #define DEMO_PARAMETERS    "-auxclk 2000 -v 2 -t 10000 -perf" \
        "-icss "        \
        "2 " /* Instance 1 or 2 Note: Use Instance 2 for AM572x IDK*/
        "1 " /* mode */        \
        "0"  /* Eth port 0 or 1 Note: AM572x IDK "0" = PRU2ETH0 (J6) and "1" = PRU2ETH1 (J8) */
#else
 #define DEMO_PARAMETERS        "-auxclk 2000 -v 2 -t 10000 -perf" \
        "-cpsw "        \
        "2 " /*port. Note: tested on port=2. AM572x IDK "1" - ETH0 (J10) and "2" = ETH1 (J12) */
        "1 " /* mode */                \
        "1 " /* priority */            \
        "m " /* master flag */        \
        "1 " /* PHY address */        \
        "1" /* PHY connection mode: RGMI */
#endif
```

The following list defines the characters in the previous code.

- -auxclk: the clock period in μs
- -t: specifies the time to run the demonstration application in ms
- -perf: enables job-performance measurement

# 7 Running the Application

Use the following instructions to run the application.

1. Power on the EtherCAT Slave device
2. Confirm that the Ethernet bus of the slave is connected to the correct port (if using the AM437x, connect the Ethernet cable to PRUETH0 [J6])
3. Confirm that the Ethernet bus is connected to the correct port in the master
   - If running the demo on the GMAC port, connect the Ethernet bus to ETH1 (J12)
   - If running the demo on ICSS_PRU, find the port that was configured on DEMO_PARAMS (the options are PRU2ETH0 [J6] or PRU2ETH2 [J8])
4. Connect a USB cable to the AM572x IDK port labeled USB JTAG
5. Set up a terminal connection to the Windows PC
6. Configure the host serial port as follows (see Figure 3): 115200 baud, no parity, 1 stop bit, and no flow control

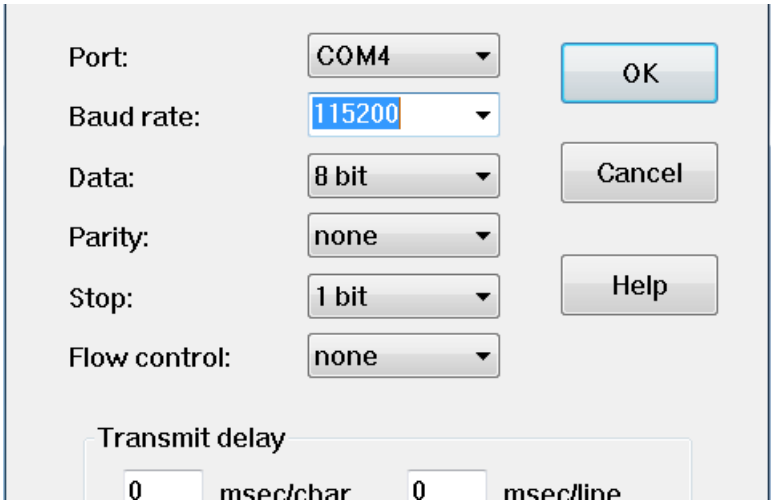



**Figure 3. Serial Port Settings**

7. Power on the (SW3) AM572x IDK board
8. Open CCS
9. Create a target configuration
   (a) Navigate to *View→Target Configuration*.
   (b) Right-click New Target Configuration
   (c) Create a file name (for example, AM572x_IDK.ccxml)
   (d) Set the connection to XDS100v2 USB Emulator for the onboard JTAG

   **NOTE:** The onboard JTAG is slow. If possible, TI recommends using an external JTAG such as a Spectrum Digital XDS560v2 STM USB.

   (e) Set the board or device to AM572x
   (f) Click Target Configuration and select Cortex-A15_0
   (g) Enter the initialization script as follows: ..\..\emulation\boards\am572x\gel\idk_am572x.gel
   (h) Click the Save button
   (i) Right-click AM572x_IDK.ccxml to launch the configuration
   (j) Right-click CortexA15_0 to connect to the target
   (k) Click *Scripts→Default*
   (l) Click OnTargetConnect_API

(m) Click on Load Program, and then click Browse Project

(n) Click EcMasterDemo

(o) Click on Debug or Release

(p) Click EcMasterDemo.out

(q) Click the Resume button

See Appendix C for an example of an output console.

## 8 EtherCAT® Master Benchmark

In this design, the EtherCAT Master has no internal tasks. As a result, the EtherCAT functions as a driver for the application. This implementation has benefits such as no synchronization issues between the application and the EtherCAT Master. The application controls the timing of events, enabling the cyclic portion to run within an interrupt service routine (ISR).

To benchmark the EtherCAT Master on the AM572x, the test includes seven commercially-available slave devices (EK110, 2x EL2004, 2x EL1004, EL4132, EK1110). The configuration used a frame load size of 579 bytes with 512 bytes of process data and a mailbox transfer to EL4132. The EtherCAT Master ran for 10 s, and TSC profiling data was collected for the following cycle jobs. Figure 4 shows the bus timing diagram.

- I: EtherCAT Master refresh inputs
- O: EtherCAT Master writer outputs
- A: EtherCAT Master administration functions
- M: EtherCAT Master acyclic datagrams and commands
- App: The application processes inputs and creates output values.

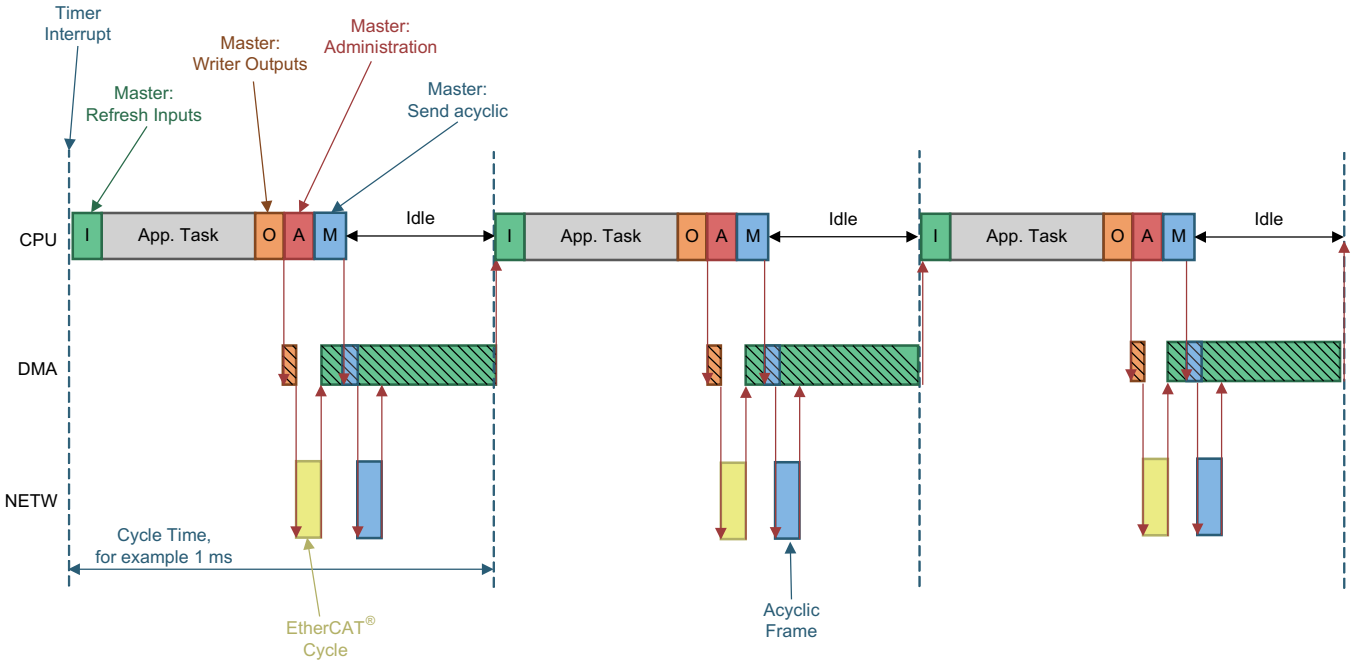





**Figure 4. Bus Timing Diagram**

# 9 Test Data

Table 1 and Table 2 show the measurements (average CPU load) in Figure 4.

**Table 1. AM572x EC-Master CPU Performance Operating on GMAC**

| Sitara AM572x<br>1000 MHz<br>NIC: GMAC<br>TI-RTOS (SYS/BIOS) | | |
|---|---|---|
| **NUMBER** | **EC-MASTER JOB** | **AVERAGE µs** |
| 1 | I: Process Inputs | 9 |
| 2 | O: Send Outputs | 5 |
| 3 | A: Administration | 4 |
| 4 | M: Send Acyclic Frame | 2 |
| **Total CPU Time** | | **20** |

**Table 2. AM572x EC-Master CPU Performance Operating on ICSS_PRU**

| Sitara AM572x<br>1000 MHz<br>NIC: ICSS_PRU<br>TI-RTOS (SYS/BIOS) | | |
|---|---|---|
| **NUMBER** | **EC-MASTER JOB** | **AVERAGE µs** |
| 1 | I: Process Inputs | 3 |
| 2 | O: Send Outputs | 15 |
| 3 | A: Administration | 5 |
| 4 | M: Send Acyclic Frame | 2 |
| **Total CPU Time** | | **25** |

# 10 EtherCAT® Master with Enabled Time-Triggered Send (TTS)

Time-triggered send is used to expand classical Ethernet to meet deterministic, time-critical, or safety-relevant conditions. TTS helps to reduce transmission jitter from microseconds to nanoseconds.

## *10.1 ICSS-PRU EMAC TTS*

When implementing ICSS EMAC firmware from TI, two ports are operated upon by two independent PRU cores. Each port operates as a separate MAC and has four independent data transmit queues (Q0 through Q3). Using the ICSS EMAC driver, the host (ARM Cortex-A15 in this case) inserts a packet in the queues. If the PRU firmware finds packets in a queue, the packets are inserted into TX FIFO and transmitted.

A TTS algorithm has been developed as part of the round robin approach used by the ICSS EMAC firmware; this feature is included in the ICSS EMAC firmware and runs as part of the transmit task.

With TTS enabled, all real-time (cyclic) packets queued by the host will be transmitted on precise, preset intervals. The acyclic packets will be sent based on time availability for the current TTS cycle. TTS can be dynamically enabled and disabled by the host and was implemented on top of PRU-ICSS EMAC with well-defined APIs that control all TTS related EMAC aspects by using EMAC IOCTL. See the API details and data structures section of ICSS_EMAC_LLD Developers guide.

## 10.2 TTS Implementation

TTS is designed to facilitate the transmission of packets at predefined cyclic instants and triggers. During TTS initialization, the application must provide the first cyclic trigger and the cycle period. The PRU firmware then sets cyclic triggers repeatedly and sends the packets cyclically (provided that they are queued before the trigger).

Figure 5 shows the use of TTS.

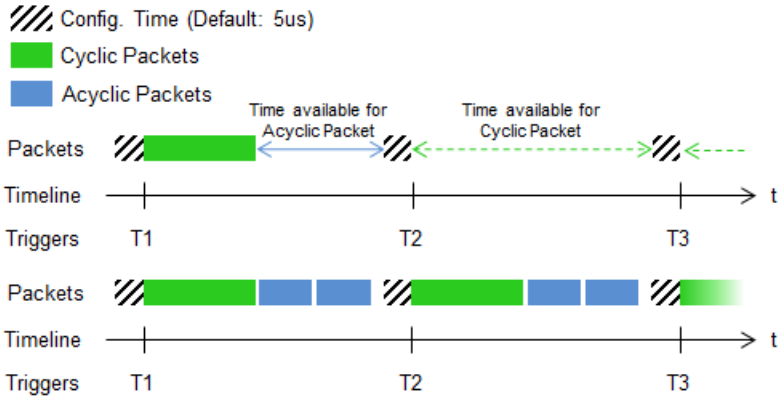



**Figure 5. TTS Timing Diagram**

For the host to queue the cyclic packet before the trigger time and avoid unnecessary jitter, the PRU firmware has two notification mechanisms: polling mode and interrupt mode.

In polling mode, the firmware sets a status bit when it is time to insert the cyclic frame; this bit is cleared when the time to insert the cyclic frame is over or when the firmware has found a cyclic frame in queue 0. This status bit can be queried by using the EMAC IOCTL.

In interrupt mode (in addition to setting the status bit), the firmware has the capability to give an interrupt to the host when it is time to insert a cyclic frame. See ICSS_EMAC_LLD Developers guide for more TTS cyclic frame notification details.

## 10.3 EtherCAT® Master with TTS

Adding TTS to the EC-Master helps to reduce transmission jitter from 10 µs to 40 ηs. The reduction in transmission jitter could be beneficial in deterministic EtherCAT system that require low latency and fast cycle times.

### 10.3.1 Implementation Overview

To add TTS functionality, the EC-Master must run on an ICSS-PRU interface. The EC-Master stack runs all periodic tasks in a do-while loop. TTS initialization is done inside of a do-while demo loop and a semaphore pending is inserted. The semaphore will be posted by the TTS insert cyclic packet interrupt callback function. The TTS insert cyclic packet interrupt is thrown by the firmware to notify the host when it is time to insert the cyclic packet.

TTS is enabled by using the interrupt notification mechanism inside of the EC-Master ICSS link layer. Additionally, the TTS initialization, TTS deinitialization, and ttsCycPortCallback functions are defined in the EtherCAT-Master ICSS link layer.

Figure 6 shows the EtherCAT Master and TTS timing diagram.

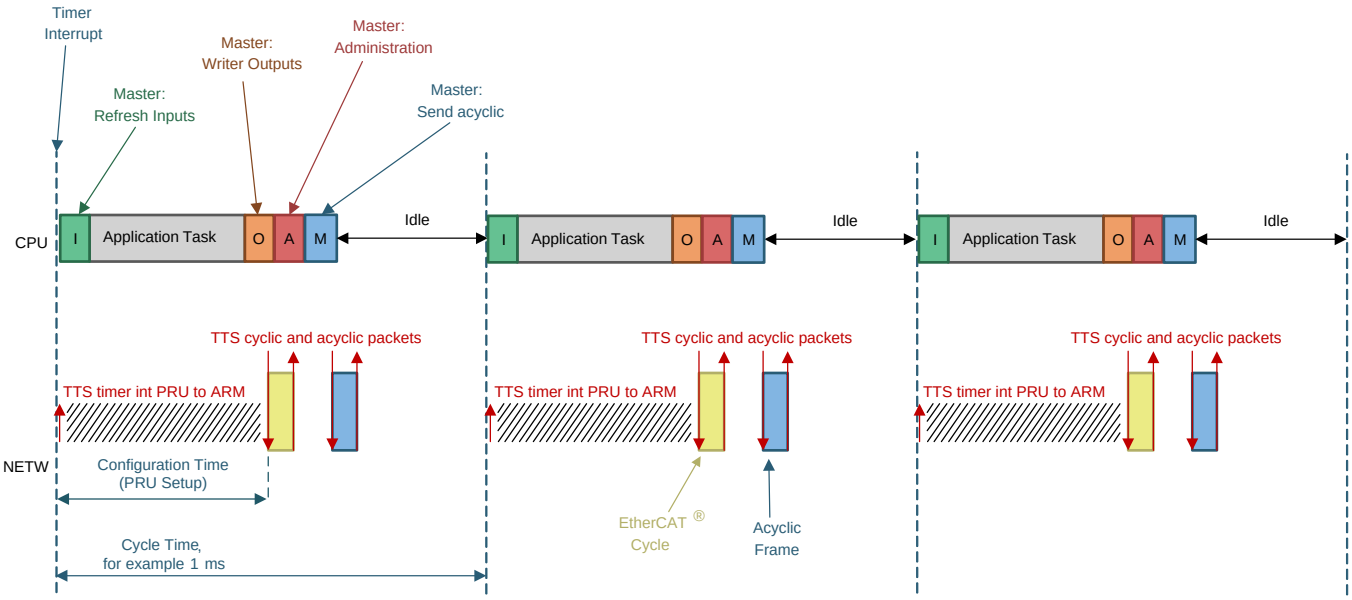





**Figure 6. EtherCAT® Master and TTS Timing Diagram**

### 10.3.2 Results

Wireshark was used to check the maximum jitter of the EC-Master TTS. to correctly check transmission jitter, Wireshark was configured to show time as *seconds since previous displayed packet*, and packets were filtered to only display Logical ReadWrite (LRW). Due to low jitter times, time resolution must be set to nanoseconds. For this test, a 2-ms EtherCAT cycle time was used.

Figure 7 shows variations up to 2 ms and ±40 ηs for TX packets.

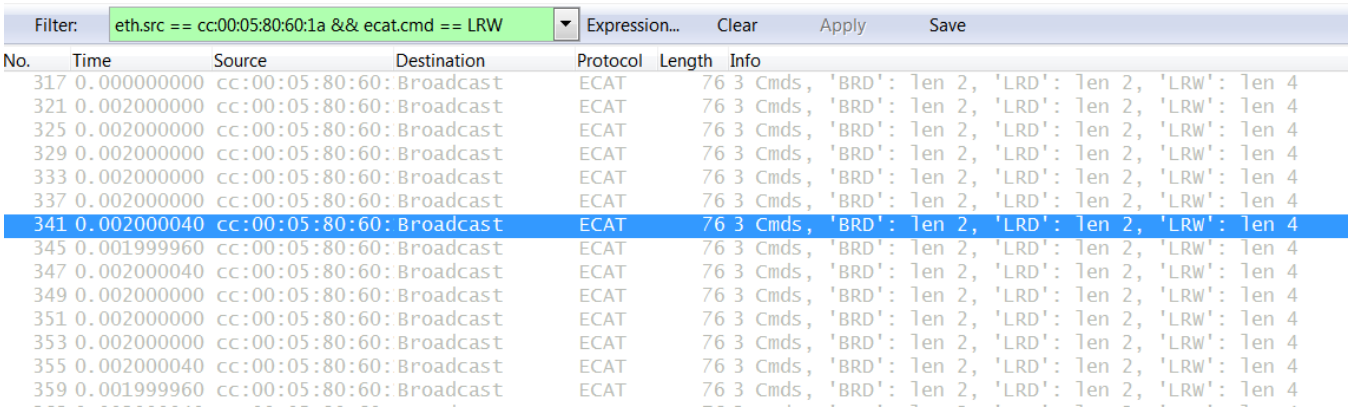



**Figure 7. Variation Times**

# 11 Design Files

## 11.1 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDEP0079.

## 11.2 Software Files

To download the software files, see the design files at the TIDEP0079.

# 12 References


1. *Technical Introduction and Overview*
2. *EtherCAT ® Master Stack for TI Sitara ™ CPU Family*, Design Guide
3. *EtherCAT ® on Sitara ™ Processors*
4. *Processor SDK RTOS Getting Started Guide*


# 13 About the Author

**PAULA CARRILLO** is a software engineer for the Embedded Processing group at TI. She obtained her MSEE from Florida Atlantic University and her Bachelor's Degree at Javeriana University, Colombia. Since joining TI in 2009, Paula has been working on different SoC and multicore DSP platforms developing applications for high-performance video codecs, synthetic aperture radar (SAR), and industrial communication protocols.

**ANJANDEEP SINGH SAHNI** is a Software Engineer for the Embedded Processing group at TI. He obtained his Bachelors in Electrical Engineering from PEC University of Technology, Chandigarh, India. Since joining TI in 2015, Anjandeep has been working with the Catalog Processor Industrial Software team, contributing to Processor SDK Industrial Software Development while working on different industrial SoCs, specifically on PRU-ICSS firmware development for multiple industrial protocols such as Ethernet MAC TTS, PROFINET, and TSN.

# 14 About the Author—acontis technologies

**STEFAN ZINTGRAF** graduated as a Software Engineer in 1987 and began working as software engineer in Sulzer and LP Elektronik. After several years working as team leader in LP Elektronik he co-founded acontis technologies GmbH in 2001, working as general manager. Stefan took over acontis' EtherCAT development activities in 2005 after being responsible for the acontis Windows® real-time products. He is also responsible for the fast growing EtherCAT market in Asia in worldwide sales activities.

## Revision History

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

| Changes from A Revision (July 2016) to B Revision | Page |
|---|---|
| • Changed Design Features | 1 |
| • Updated Processor SDK from v2.0.2.11 to v3.1.0.6 | 5 |
| • Updated AM572x PDK from v1.00.02 to v1.00.04 | 5 |
| • Added Section 10 | 9 |
| • Added Added second TI author | 12 |
| • Deleted irrelevant troubleshooting sections | 18 |

# Appendix A AM437x Running EtherCAT Slave

## A.1 Preparing SD Card

Refer to wiki.ti.com to prepare the SD card, or use the following instructions.

1. Ensure the HP USB disk storage format tool v2.0.6 is portable
2. Run the HP USB disk storage format tool v2.0.6 as portable executable. The executable detects the SD card that is plugged into the reader. If undetected, direct the executable to the new disk.
3. Choose FAT32 if the SD card is greater than 4GB. If not, use FAT
4. Click the Start button

**NOTE:** After formatting, the card can be populated by the files.

5. Install the ISDK prebuilt binaries (Industrial SDK 2.1.0.1)
6. Copy MLO from <Installation_path>*sysbios_ind_sdk_prebuilt_02_01_00_01\Bootloader\SD\am437x_release* to the SD card
7. Copy the EtherCAT Slave application from <Installation_path>*sysbios_ind_sdk_prebuilt_02_01_00_01\ethercat_slave\am437x_release*
8. Connect the Ethernet cable to the ICSS port (J6)
9. Use Wireshark to test the ecat (EtherCAT) packets

   Alternatively, use an EtherCAT Master, such as TwinCAT®, to test master-to-slave connectivity

For details using TwinCAT with TI IDK boards, see the user guide at wiki.ti.com.

## Appendix B acontis EC-Engineer Tool for Creating an ENI File

1. Register for a free evaluation version at www.acontis.com
2. Install the EC-Engineer tool on the PC
3. Connect the EtherCAT slave to the computer (if using the AM437x, connect the Ethernet cable to J6)
4. Open the EC-Engineer tool
5. Select Online Configuration
6. Select the EtherCAT Master unit (Class A) as the master unit
7. Click the OK button
8. Select 2000 as the Cycle Time (µs)
9. Select the desired network adapter as the slave that is connected to the local system.

**NOTE:** If using the AM437x as the slave, add TI's IDK.xml to the ESI manager as follows:

- Open the ESI manager
- Add the IDK.xml file
- Browse to <Installation_path>*sysbios_ind_sdk_2.1.0.1**sdk**examples**ethercat_slave**esi**TiEtherCATLib.xml*
- Open the file
- Navigate to the Network option
- Click Scan EtherCAT Network
- Click Export ENI after the slaves are found

Selecting Export ENI exports eni.xml.

## Appendix C Application Console Output

```
boardName: AM572IDK
board type is AM572IDK
SYS/BIOS EcMaster Sample application
Full command line: -auxclk 2000 -v 2 -t 10000 -perf -icss 2 1 0

Run demo now with cycle time 2000 usec Using AuxClock
==========================
Initialize EtherCAT Master
==========================
EC-Master V2.8.1.12 (Protected) for SYSBIOS Copyright acontis technologies GmbH @ 2016
Unlicensed version, stop sending ethernet frames after 60 minutes!
Bus scan successful - 1 slaves found
1 identical messages skipped
********************************************************************************

Slave ID.............: 0x00000000
Bus Index............: 0
Bus AutoInc Address.: 0x0000
Bus Station Address.: 0x03e9 (1001)
Bus Alias Address...: 0x0000 ( 0)
Vendor ID............: 0xE000059D = ----
Product Code........: 0x54490002 = Unknown
Revision.............: 0x00000001 Serial Number: 0
ESC Type.............: Texas Instruments (0x90) Revision: 2 Build: 971
Connection at Port A: yes (to 0x00010000)
Connection at Port D: no (to 0xFFFFFFFF)
Connection at Port B: no (to 0xFFFFFFFF)
Connection at Port C: no (to 0xFFFFFFFF)
Line Crossed........: no
Cfg Station Address.: 0x03e9 (1001)
PD IN Byte.Bit offset: 0.0 Size: 32 bits
PD OUT Byte.Bit offset: 0.0 Size: 32 bits
EtherCAT network adapter MAC: 18-78-06-80-38-9E

====================
Start EtherCAT Master
====================
Master state changed from <UNKNOWN> to <INIT>
Master state changed from <INIT> to <PREOP>
```

```
Master state changed from <PREOP> to <SAFEOP>
Master state changed from <SAFEOP> to <OP>

Job times during startup <INIT> to <OP>:
============================================================
PerfMsmt 'JOB_ProcessAllRxFrames' (avg/max) [usec]: 2.4/ 13.7
PerfMsmt 'JOB_SendAllCycFrames ' (avg/max) [usec]: 2.5/ 5.5
PerfMsmt 'JOB_MasterTimer ' (avg/max) [usec]: 4.6/ 31.3
PerfMsmt 'JOB_SendAcycFrames ' (avg/max) [usec]: 5.5/ 26.0
PerfMsmt 'Cycle Time ' (avg/max) [usec]: 1613.2/2008.4
PerfMsmt 'myAppWorkPd ' (avg/max) [usec]: 0.4/ 1.1

============================================================
PerfMsmt 'JOB_ProcessAllRxFrames' (avg/max) [usec]: 2.3/ 2.6
PerfMsmt 'JOB_SendAllCycFrames ' (avg/max) [usec]: 3.6/ 5.1
PerfMsmt 'JOB_MasterTimer ' (avg/max) [usec]: 3.8/ 5.1
PerfMsmt 'JOB_SendAcycFrames ' (avg/max) [usec]: 0.7/ 0.8
PerfMsmt 'Cycle Time ' (avg/max) [usec]: 1998.4/2009.2
PerfMsmt 'myAppWorkPd ' (avg/max) [usec]: 0.6/ 1.6
```

## Appendix D Troubleshooting

Use the following instructions for software troubleshooting.

1. Navigate to <ECmaster_installation_path>*\Acontis_EC_master\EC_Master_Sysbios_SDK_Eval\Doc*
2. Open *Acontis EC-Master Stack Class B Version 2.8 Document*
3. Reference the opened document for troubleshooting

## IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ('TI") reference designs are solely intended to assist designers ("Designer(s)") who are developing systems that incorporate TI products. TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.

TI's provision of reference designs and any other technical, applications or design advice, quality characterization, reliability data or other information or services 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 reference designs or other items.

TI reserves the right to make corrections, enhancements, improvements and other changes to its reference designs and other items.

Designer understands and agrees that Designer remains responsible for using its independent analysis, evaluation and judgment in designing Designer's systems and products, and has full and exclusive responsibility to assure the safety of its products and compliance of its products (and of all TI products used in or for such Designer's products) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to its applications, it has 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. Designer agrees that prior to using or distributing any systems that include TI products, Designer will thoroughly test such systems and the functionality of such TI products as used in such systems. Designer may not use any TI products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death (e.g., life support, pacemakers, defibrillators, heart pumps, neurostimulators, and implantables). Such equipment includes, without limitation, all medical devices identified by the U.S. Food and Drug Administration as Class III devices and equivalent classifications outside the U.S.

Designers are authorized to use, copy and modify any individual TI reference design only in connection with the development of end products that include the TI product(s) identified in that reference design. HOWEVER, 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 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 the reference design or other items described above 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 REFERENCE DESIGNS AND OTHER ITEMS DESCRIBED ABOVE ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, 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 DESIGNERS AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS AS DESCRIBED IN A TI REFERENCE DESIGN 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 THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TI's standard terms of sale for semiconductor products (http://www.ti.com/sc/docs/stdterms.htm) apply to the sale of packaged integrated circuit products. Additional terms may apply to the use or sale of other types of TI products and services.

Designer will fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

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