Design Guide
SPRAC74A–February 2017–Revised March 2017

AM335x Low Power Design Guide

This document discusses techniques to develop a low power, low cost system based on the AM335x series processor. The goal is to explain how system design choices affect power consumption and how power consumption can vary significantly depending on how the developer's application runs in that system under specific conditions.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The AM335x processor family features an ARM Cortex-A8 and was created with power-sensitive and cost-sensitive applications in mind. These applications include industrial communications, IoT gateways, renewable energy gateways, substation automation and streaming audio players. These diverse set of applications require a processor family that provides a wide range of performance levels (ranging from 300 MHz to 1 GHz), a comprehensive integration of connectivity peripherals and key enhancements in industrial spaces, which require 3D graphics and multiple display options.

At the end of this document, the developer will know about many of the low power features of the AM335x, and how to optimize power consumption for their specific application. First, the hardware considerations, when designing a schematic to take advantage of all the low power features of AM335x, are discussed such as power management IC selection. Then, low power features in Linux are highlighted to illustrate how software can be used to optimize runtime power even in existing designs. Finally, several low power optimized use cases are documented with corresponding power consumption figures to demonstrate the potential power savings.

2 Hardware Low Power Design Guidelines

When designing a system for low power, there are several special considerations to be made to ensure an efficient final product. Concepts discussed in this section include designing a power distribution network, I/O planning and configuration, thermal considerations, memory technology selection, and design for measurability. This section focuses on design considerations specifically for low power design of the AM335x processor. There are many other design decisions to make when creating a system, and for a more complete checklist of design decisions, see the schematic checklist in the AM335x Schematic Checklist and the AM335x Hardware Design Guide wiki pages.

2.1 Power Tree

When designing a system, begin with outlining the power requirements for the system by creating a power tree. A power tree maps out where power comes from in the system, from an initial AC wall socket, battery, and so forth, to how it is translated to voltages used within the system, and finally to the actual consumers of power in a system such as a processor or peripherals such as displays and sensors. A good power tree clearly outlines all the voltage rails and currents required and indicate the number of voltage translations necessary to generate these voltages. Having a power tree helps the system designer define the power supply requirements of the device.

Here are some points and places for optimization that must be addressed by a complete power tree design.

- How many voltage translations does the power tree have? Are all strictly necessary? Minimize number of voltage level translations to reduce conversion losses.
- How efficient are voltage regulators? Choose high efficiency regulators where it counts (high current rails supplying processor, DDR, peripherals).
- What peripheral voltage(s) are used? Use 1.8 V I/O peripherals to minimize power consumption.
- What memory technology is being used? Is the optimum voltage being used? Sometimes there is a trade off to be made, for example, some PMICs cannot support a dedicated 1.35 V output for DDR3L, and many PMICs will already supply a 1.8 V rail since it is a common I/O voltage. The system designer must weigh lower BOM cost, vs higher power consumption.

In order to determine the power requirements of the AM335x, see the *Maximum Current Ratings at AM335x Power Terminals* table in the *AM335x Sitara™ Processors Data Manual*. This table details the maximum current draw for each of the AM335x power domains.

2.1.1 Power Supply Selection

Although it is possible to design a discrete power supply solution, it is highly recommended to use a Power Management IC (PMIC) as a power supply for the AM335x. PMICs offer several features that are useful for low power design such as dynamic voltage rail adjustments, power sequencing, and easy to use control interfaces.

When selecting a PMIC, refer back to the power tree to make sure the outputs of the PMIC can satisfy the voltage and power requirements of the identified voltage supply rails. In many cases, rails can be sourced from the same PMIC output as long as they are of the same voltage and the total current draw is less than the maximum output current of the PMIC. In some cases where other devices on a system consume large amounts of power, it makes sense to use the PMIC primarily as a power supply for the processor, and use an alternative power solution to power the rest of the device.

In addition to checking to make sure that the PMIC or power supply chosen satisfies the power requirements, special consideration must be made for specific use cases. For example, DDR3L requires a 1.35 V supply rail, and if Real-Time Clock (RTC) mode is to be supported, the PMIC must allow for a configuration where it is on during suspend. Finally, a balance between cost and power efficiency must be made since some PMIC features may come at increased cost.

2.1.2 Recommended PMICs

Table 1. Recommended PMICs

Feature	TPS650250	TPS65217	TPS65910	TPS65218
Power (DCDC/LDO)	3 DCDC/ 3 LDO	3 DCDC/ 2 + 2 LDOs	3 DCDC/ 9 LDO	4 + 2 DCDC / 1 LDO
Voltage Range	2.5 - 6.0 V	2.75 - 5.5 V (20 V Tolerant)	2.7 - 5.5 V	2.75 - 5.5 V
Dynamic Voltage Scaling OPP Supported	No 300/ 600 MHz	Yes 300/ 600/ 720/ 800/ 1000 MHz	Yes 300/ 600/ 720/ 800/ 1000 MHz	Yes 300/ 600/ 720/ 800/ 1000 MHz
Supervisor	No	No	No	Yes (± 4%, ± 5%)
Power Sequencing	No Requires external circuit	Yes	Yes	Yes
Memory Support	LPDDR1, DDR2, DDR3	LPDDR1, DDR2,DDR3, DDR3L	LPDDR1, DDR2, DDR3	LPDDR1, DDR2, DDR3, DDR3L
Battery Charger	No	Linear w/ Power-Path	No	No
RTC	No (requires external LDO)	No	Yes	No uP controls power to RTC
Other features	Power fail comparator	• WLED Driver • 2 LDO can be configured as load switches • I2C Interface	• 5 V Boost (100 mA) • 2x I2C Interface	• Three load switches • Power fail comparator • Supports warm reset • I2C Interface
Temperature Range	-40, 85°C/ -40, 125°C (Q1)	-40, 105°C	-40, 85°C	-40, 105°C

2.2 I/O Considerations

I/O configuration is very important when designing a system for low power. Properly configured I/O will minimize static leakage current from misconfigured pull-up/ pull-down networks, improper termination, as well as maximizing functionality by enabling built-in power saving features on the AM335x.

In addition, for low power use cases, consider using 1.8 V I/O where possible; reducing the switching voltage significantly reduces system power consumption.

2.2.1 Pin configuration

I/O planning can be done using the pinmux tool that is both available as a constantly updated cloud tool, or as a downloadable program for Windows® and Linux®. This tool generates relevant configuration files for software to configure I/Os on the AM335x as intended. It is important to note that the hardware configuration in the design and configuration done in software match in order to prevent leakage current. Software configurations are discussed in the Linux power management section.

2.2.1.1 Input Buffer Leakage

Input buffers are implemented as two CMOS transistors between V_{CC} and ground. When an input is driven either high or low, only one of these transistors will be switched on, and there is no leakage current. However, if the input is floating, or in a high impedance state, sometimes the voltage at the input can be between V_{IL} and V_{IH} (the points where an input is considered either low or high, respectively); in this state, both transistors are partially switched on creating a resistive path to ground.

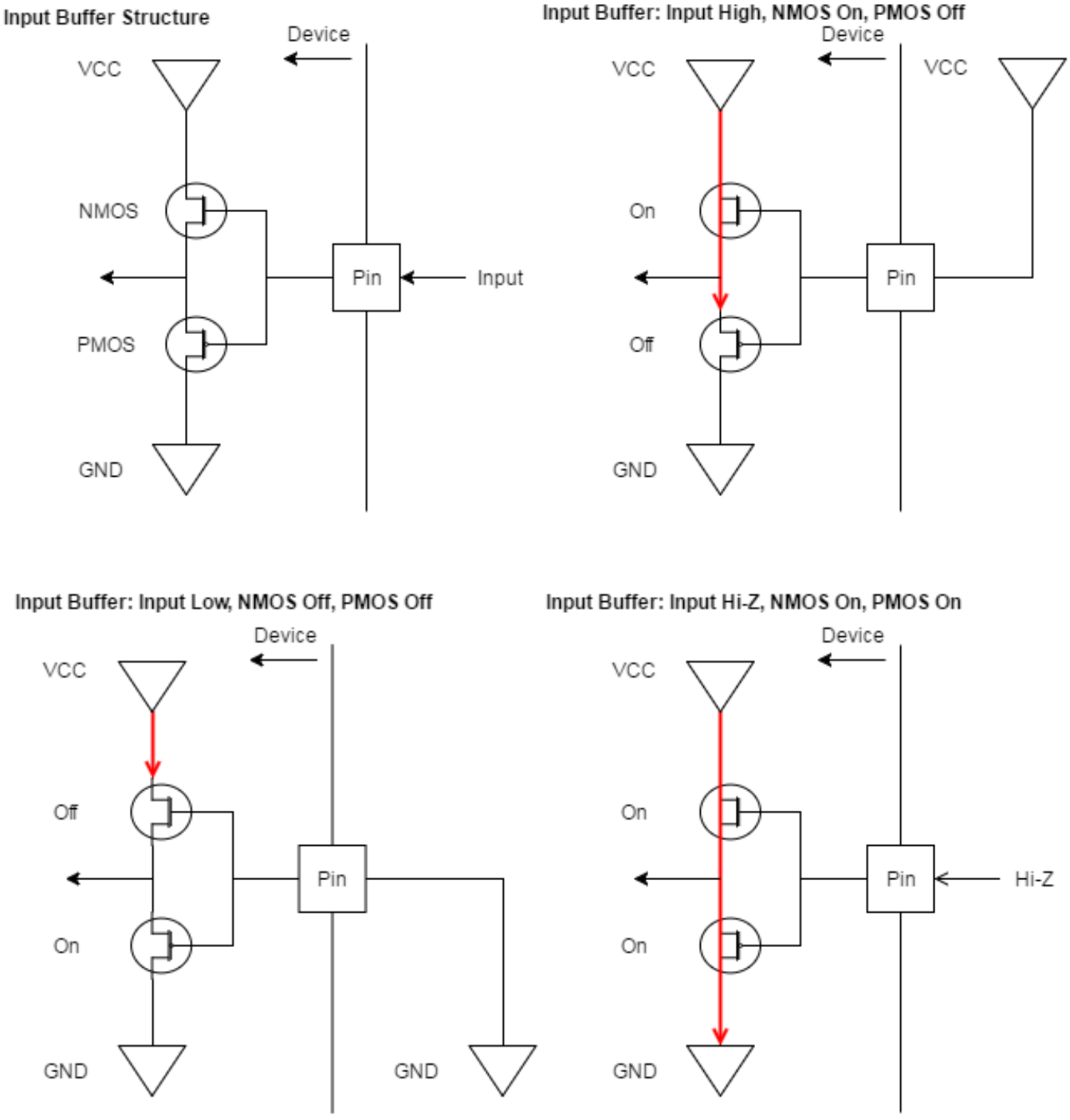

Figure 1. Input Buffer

2.2.1.2 GPIO Leakage Scenarios

I/O buffers are implemented with two buffers: an output buffer with an enable signal and an input buffer. The enable line is controlled by internal logic and the specific implementation of the enable signal is device specific. When the output buffer is enabled, it drives the input buffer, but when the pin is configured as an input, the pin must be driven or else the input buffer will be floating and can draw leakage current.

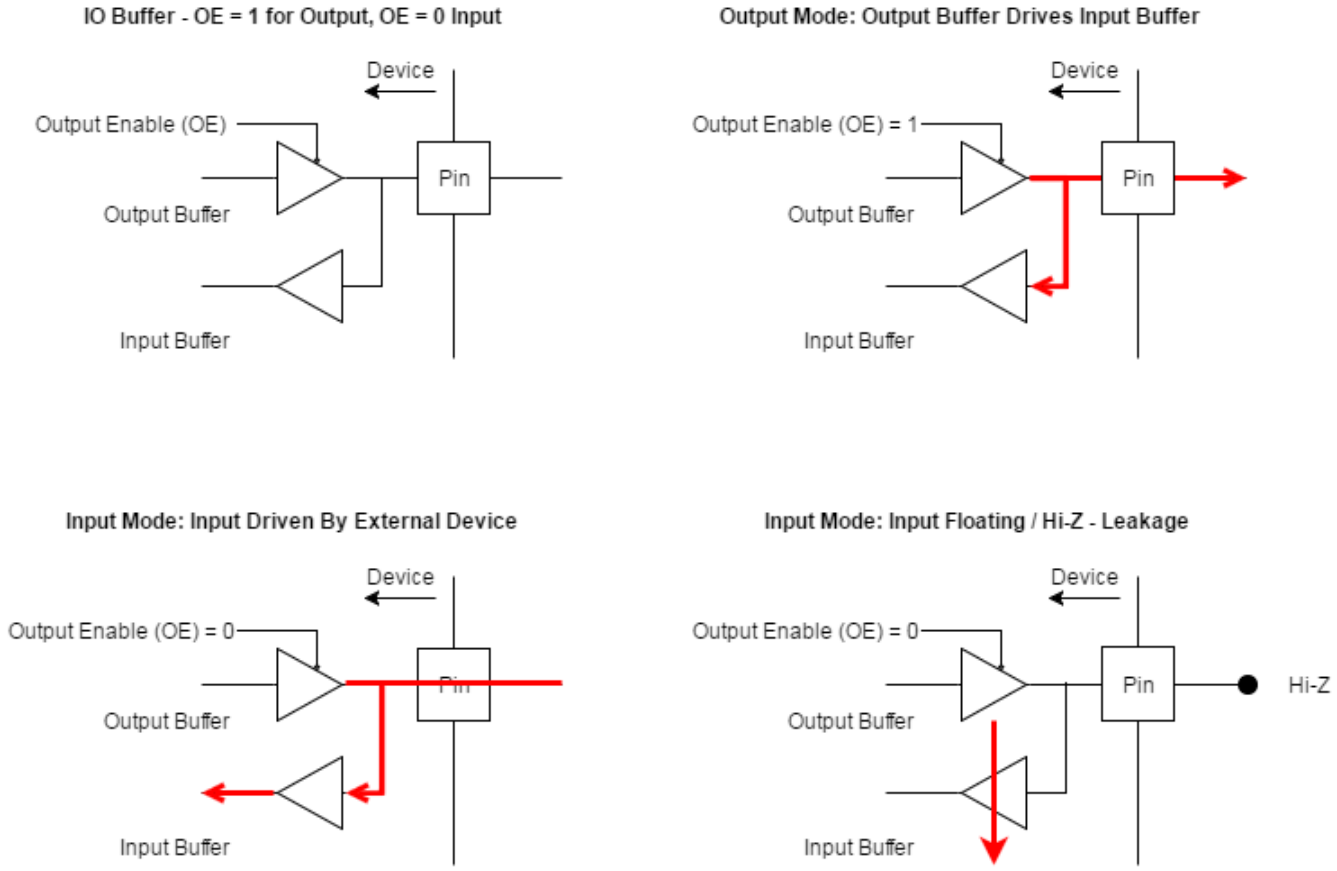

Figure 2. GPIO Leakage Scenarios

2.2.1.3 Internal Pull Up / Down

The internal pull-up/pull-down resistors are implemented with weak transistors. Since the effective resistance of these "resistors" vary according to their gate voltage, the internal pull up/down resistors will not consume additional power if the internal and external pulls match. To prevent excessive static power consumption, be sure that there are no pin conflicts between the board schematic and the internal configuration of the AM335x. Inputs that can be in a high impedance (Hi-Z) state should have an appropriate pull up/down resistor to accommodate for the times when the input is not driven.

Of the cases in the diagram below, only Case 5 will register as device static power consumption since the processor is sourcing current. However, both Case 5 and Case 6 have static leakage current, Case 6 just consumes power from an external power rail.

Ensure all pullups connected to AM335x are pulled up to the correct I/O voltage to avoid any leakage between the I/O rails of AM335x. Also ensure that the system generates inputs either below V_{IL} or above V_{IH}. For example, a 1.8 V peripheral connected to a 3.3 V input will be in between V_{IL} and V_{IH} when the 1.8 V signal is high. This means that not only will the peripheral not work properly, it causes the processor to consume extra power.

A mid-supply will have a big effect on power consumption. Each terminal has an associated voltage used to power its I/O cell, the corresponding voltage rail for each pin can be found in the *AM335x Sitara™ Processors Data Manual*, in the Ball Characteristics table under the "ZCE Power/ ZCZ Power" column.

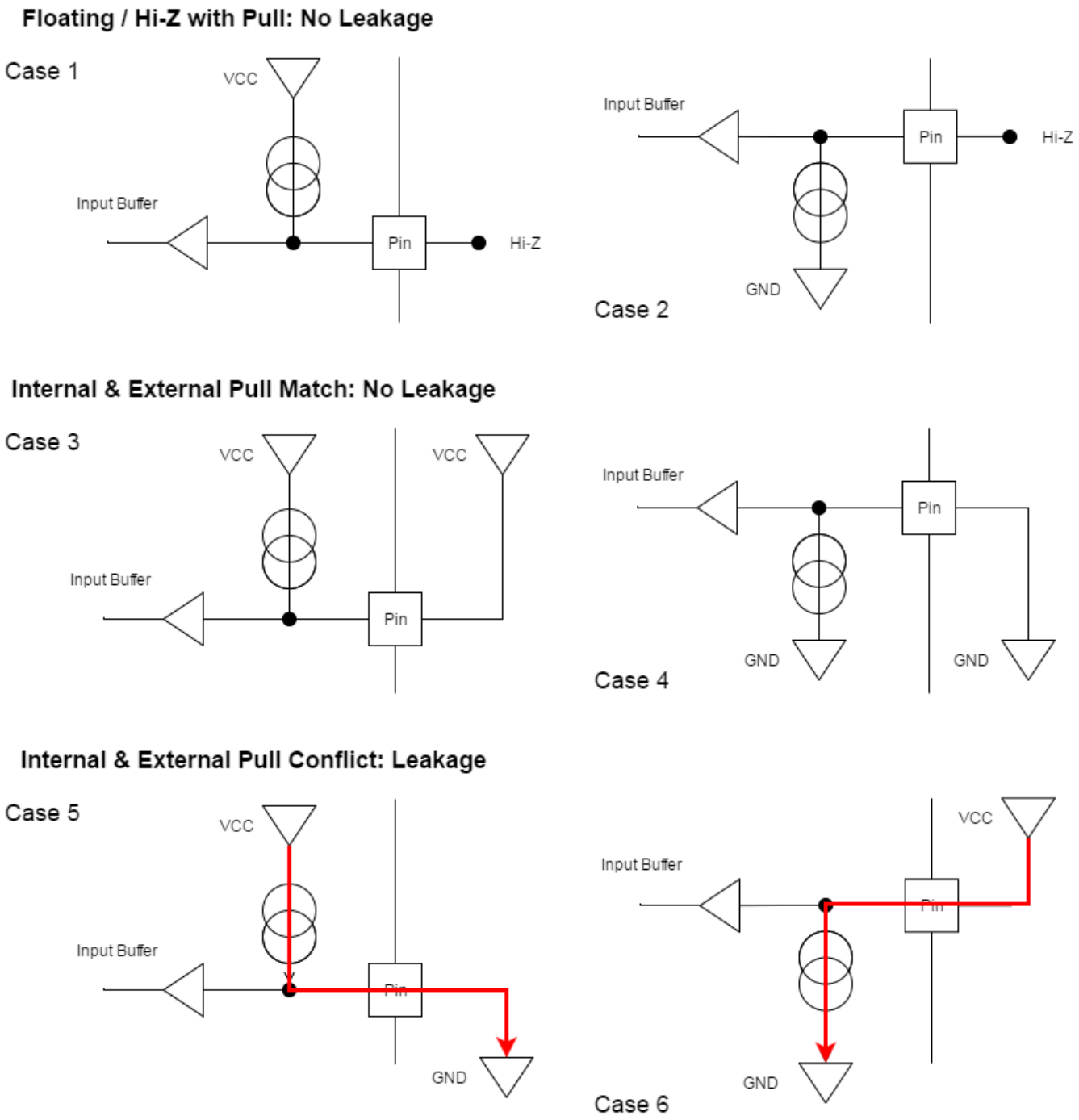

Figure 3. Internal Pull Up/Down

2.2.2 Unused Pins

For the most part, unused interface pins can be left as no-connect. However, since unused pins are often initialized at reset as inputs (RXACTIVE), it is up to the software to mark the unused pins as RXACTIVE=0, which disables the inputs and prevents unwanted input switching. During system power up, before Linux software has a chance to configure unused pins, there is a chance that these inputs can float and cause some current draw if they float to mid-supply levels. However, this is only a concern if low boot power is a priority and can be fixed with external pull up/down resistors, if absolutely necessary.

2.2.2.1 Analog-to-Digital Converter (ADC)

If the ADC is unused, connect all the TSC_ADC signals to the same ground as the V_{SS} terminals:

- VREFP
- VREFN
- AIN[7:0]
- VDDA_ADC
- VSSA_ADC

2.2.2.2 USB

If either USB0 or USB1 is unused, connect the respective VDDA1P8V_USB terminal to any 1.8-V power supply and respective VDDA3P3V_USB terminal to any 3.3-V power supply. If the system does not have a 3.3-V power supply, the VDDA3P3V_USB terminal may be connected to ground. The respective VBUS, ID, DP, and DM terminals may be connected to ground or left floating. The CE pin should be left floating.

Since the USB PHYs are on by default, software needs to turn these interfaces off to maximize the power savings.

2.2.2.3 RTC

If the RTC internal oscillator is not used, such as when an external LVCMOS clock is used, or if RTC is not used, leave the unused RTC_XTAL pins open circuit, and connect VDD_RTC to VSS if using the ZCZ package.

2.2.3 Wakeup and Deep Sleep Connections

AM335x has several peripherals in the Wakeup Power Domain. Since these peripherals remain powered on during deep sleep and suspend, they can be used to generate wakeup interrupts when the processor is in a suspend or sleep mode. In addition, these peripherals can be used to control on board peripherals during the suspend/resume sequence.

Modules in the PD_WKUP domain include:

- GPIO0
- DMTIMER0_dmc
- DMTIMER1
- Inter-Integrated Circuit (I2C0)
- TSC
- WDT1

For a full list, see the *Device Modules and Power Management Attributes List* section in the *AM335x and AMIC110 Sitara™ Processors Technical Reference Manual*.

GPIO0 is the only GPIO bank that can generate a wakeup interrupt, connect wakeup sources only to GPIO0[0:31]. In addition, if DDR3 is used with a VTT regulator, route the enable signal to an output pin in GPIO0 so that the processor can shut it off when in DS0.

For low power designs, consider using a crystal oscillator instead of a LVCMOS square wave clock. The AM335x has internal circuitry that can entirely turn off the crystal during deep sleep (DS0). Similar functionality with an external LVCMOS clock would require additional external circuitry.

The AM335x has a Cortex M3 that uses I2C0 to adjust voltages during DS0, so to take advantage of this feature, make sure the PMIC is connected on the I2C0 bus.

2.3 Core Leakage

Core leakage is a result of the physical properties of transistors. On some level it is unavoidable since it is impossible to remove thermal noise from a system. However it can become an issue when junction temperature on the device increases. Therefore, if operating conditions for your application include high temperatures, or if significant load on the processor is expected, special attention to the cooling of the device should be taken.

These efforts could include using thicker copper layers on the ground plane to provide better thermal sinking, the inclusion of a heatsink, or even incorporating an active cooling solution. Higher cooling capacity will enable the processor to run at a lower temperature which would improve core leakage. For more detailed information about how to manage thermal considerations, see the *Thermal Design Guide for DSP and ARM Application Processors*.

2.4 DDR System Design

The DDR interface can be a major consumer of power on both the System-on-Chip (SoC) and at the system level. For new system designs, it is recommended to use a single DDR3L memory configured in a point-to-point topology without VTT termination. Using only a single DRAM module reduces the component count that in turn reduces power consumption. In addition, eliminating VTT is a significant power savings on its own. Since a single DDR3L chip now can have densities of 1 GB, there is no capacity tradeoff for using only one DRAM chip. Actual memory capacity should still be tailored to the needs of the specific application.

If other topologies are needed for a specific application, see the device-specific technical reference manual (TRM) for other supported topologies, as well as more information on supporting DDR2, and LPDDR1.

For existing designs with VTT, if possible, ensure that VTT is turned off when DDR enters self-refresh as this generates a significant power savings.

2.5 Design for measurability

During the development and prototyping stage it is useful to integrate some features to make measuring power consumption easier. The AM335x has the following power rails:

- VDD_MPU
- VDD_CORE
- VDDS
- VDDSHV1-6
- VDDS_DDR

All of these rails can all be measured independently. There are two basic ways to monitor power consumption on the AM335x: first is through test points and shunt resistors, and second is by integrating in INA226 analog-to-digital converters to monitor power.

2.5.1 Shunt Resistors

The simplest way to implement power monitoring capabilities is to include low resistance, high precision shunt resistors in the power supply rails. By measuring the voltage drop across the resistor and the supply rail voltage, it is simple to calculate the power consumed by the measured rail. Using a shunt to compute the current through the power rail instead of measuring current directly, the impact of unknown voltage drops is negated due to probe resistances. Typical values for these resistors can be found in the *AM335x Evaluation Module (EVM) Hardware User's Guide*, it can also be advantageous to consider the sensitivity of the multimeter used to ensure that the expected voltage range is measurable with the desired precision.

It may be convenient to use jumpers or test points placed close to the shunt resistors to monitor the voltage across the resistor. Otherwise, measurement can be made directly at the terminals of the shunt resistor.

2.5.2 On Board Power Monitoring Devices

The INA226 is a current shunt and power monitoring device with an I2C or PMBus interface. Use of the INA226 allows for easily automated power monitoring via the I2C interface. If there are many rails that require continuous power measurement, the INA226 removes the requirement for several multimeters or sets of probes running all over the board. Using a digitally controlled measurement device can also make it easier to automate measurements, since the system can be controlled using a script that can collect power data while cycling the device through different use cases.

One drawback of using an integrated power monitoring solution is that depending on the setup, it may require additional hardware beyond the power monitoring device. For example, on the AM335x GP EVM, the integrated INA226 devices sit on an external I2C bus. This allows power consumption to be monitored when the SOC is in suspend or sleep mode, but requires an additional device to read data from the monitors. It is relatively trivial to get a USB to I2C adapter, or another development board with I2C to gather this telemetry, but it is another consideration to be made. One potential compromise would be to connect the INA device to an AM335x I2C bus and create a multiple master bus.

As a reference, AM335x GP EVM uses two INA devices to monitor power consumption on the VDD_CORE and VDD_MPU voltage supply rails on an external I2C bus.

2.5.3 Note on Remote Sensing

Remote sensing, also sometimes known as four (4) terminal measurement, is when separate sets of terminals are used to measure current (I) and voltage(V).

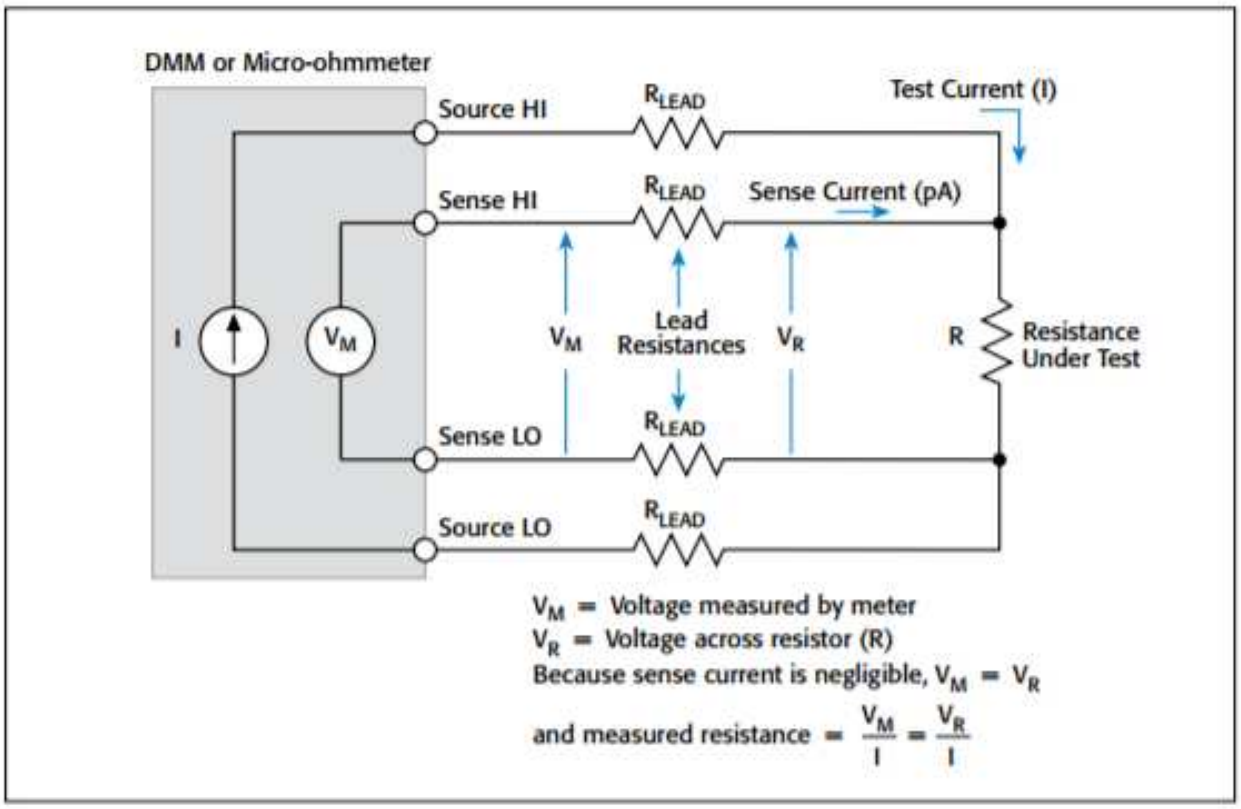

Figure 4. Four-Wire Resistance Measurement Configuration

As opposed to traditional local, or two terminal sensing, in the four terminal example, there is no voltage drop due to lead resistance since the current through the voltage sense terminals is negligible.

For situations where current or voltage is sourced from the multimeter as opposed to an on-board power supply, four wire sensing is required to ensure accuracy of the reading, especially for the low currents and resistances encountered in the power measurement shunts. Even when current or voltage is not sourced from the multimeter, remote sensing is going to be more accurate than two terminal measurement. However, if only current or only voltage is being measured, not both, it is generally acceptable to use local, or two terminal sensing.

3 Linux Power Optimization Features

The previous section outlined the hardware design practices and techniques that can lower power in an AM335x system. However, system power consumption is highly dependent on how software configures the underlying hardware. Once the board level optimizations such as DDR and PMIC selection are done, the rest of the power savings possible is determined by how well the software controls the processor and the on-die peripherals. For example, scaling the processor speed can have large power implications, not only does the switching rate increase, but the voltage supply must increase to support the higher frequency. This means that power consumption is not linear with clock speed, and also implies that if lower clock speeds are used, the power savings can be significant. In this section, optimization is possible using the TI Linux Processor SDK v3.0.1 that is discussed, however, many of these concepts are transferable to another Linux distribution, or even to other operating systems.

In this section configuration to the Kernel, U-boot, and device tree that must be done prior to booting the device is discussed. The next section illustrates how some runtime configuration choices such as choosing an operating performance point can have significant impacts on power consumption. The last segment discusses how to diagnose power consumption issues on a Linux based platform.

3.1 Pre-Deployment Optimizations

This section explains how to enable and disable desired peripherals in the Linux Device Tree, as well as how to customize and enable features such as dynamic voltage and frequency scaling (DVFS), and the CPU Idle driver. All these features require modifications to either the Linux kernel, U-boot, or device tree, and must be performed before the system is powered on. Runtime tweaks are useful for quick experimentation and discussed in the next section.

3.1.1 Hardware and Software Alignment

For optimal results, the hardware and software must be configured to the same specification to take advantage of the power saving features designed into the hardware as well as to avoid leakage current.

3.1.1.1 Pinmux Tool

The pinmux tool generates relevant sets of files, header files for AM335x starterware, and pin control device tree entries. These files contain the necessary data to configure the instantiated pins in the design. It is still up to the user to define behavior for unused pins.

For inputs ensure that an external pull does not conflict with an internal pull, as it would provide a current path if they are opposite.

3.1.1.2 Clock Tree Tool

The clock tree tool is a standalone tool that allows for configuration of the AM335x clock tree. The tool allows users to visualize the clock tree, interactively modify the clock tree elements and view the effect on Power, Reset, and Control Management (PRCM) registers. Additionally, they can use the tool to modify PRCM registers and view their effects on the clock tree, and track the impact of user defined changes in the device clock tree. The clock tree tool also allows for import and export of the current clock tree settings as a text file.

In terms of power optimization, clock tree tool can help determine the register writes required to attain a certain clocking frequency. Certain peripherals have timing constraints that prevent them from having their clock frequency reduced, but other peripherals can be configured to run at lower clock speeds, which can help save power. Determining the necessary Phase Locked Loop (PLL) parameters is also needed when creating custom Operating Performance Points (OPPs) in the Device Tree.

3.1.2 Configuring U-Boot

U-Boot needs to be re-compiled to support RTC-Only mode. However, this is not currently supported out of the box with Processor SDK. The parameters used to boot Linux can be changed without recompiling the U-Boot binary.

3.1.3 Linux Kernel Modifications

Linux has provisions to support dynamic CPU frequency scaling, dynamic CPU idling, as well as some device-specific features such as the Cortex®-M3 (CM3) power management coprocessor. Dynamic frequency scaling is especially important because it enables the full performance of the processor, when necessary, but allows for a return to a lower frequency and voltage combination setting for power savings. In addition, the Linux kernel will dynamically clock gate the Microprocessor Unit (MPU) and put it in various idle states based on the current and predicted future workload.

Processor SDK does have many of these features enabled by default, but this section documents the relevant configuration settings and specific source files involved in the power saving features of the Linux kernel in case these features need to be enabled.

3.1.3.1 Supporting CPU Frequency Scaling

As mentioned previously, higher operating frequencies also require higher MPU supply voltages. This can cause significant increases in power consumption, thus, it is important to allow the CPU to operate at the lowest possible frequency. Linux offers several different policies called governors to control the scaling behavior of the CPU, this behavior is known as Dynamic Voltage and Frequency Scaling (DVFS). How to control this behavior, as well as definitions for the various governors is documented in the DVFS section. DVFS is by default enabled in Processor SDK.

Source location:

`drivers/cpufreq/cpufreq-voltdm.c`

Menuconfig:

When configuring the Linux kernel using menuconfig, the CPU frequency scaling feature is found under:

`CPU Power Management ---> CPU Frequency Scaling --->`

CPU Freq Options

```
[*] CPU Frequency scaling
   <*>   CPU frequency translation statistics
   [*]     CPU frequency translation statistics details
         Default CPUFreq governor (userspace)  --->
   <*>   'performance' governor
   <*>   'powersave' governor
   -*-   'userspace' governor for userspace frequency scaling
   <*>   'ondemand' cpufreq policy governor
   <*>   'conservative' cpufreq governor
         *** CPU frequency scaling drivers ***
   <M>   Generic DT based cpufreq driver
   <M> Generic DT based cpufreq driver using clk notifiers
  ...
```

3.1.3.2 CM3 Firmware

The AM335x has a small ARM Cortex M3 co-processor that is used to enable low power state transitions. This processor is used to configure pins to their sleep state, and is used to save and restore context to all peripherals on the AM335x. To enable this functionality, the CM3 must be loaded with an appropriate firmware from the kernel at run-time. The name of the binary file containing this firmware is `am335x-pm-firmware.elf` for the AM335x SoC. The git repository containing the source and pre-compiled binaries of this file can be found in the TI git repository.

There are two options for loading the CM3 firmware: through the file system or by including it in the Kernel itself. If using the Processor SDK, the firmware is included in /lib/firmware and the root filesystem should handle loading it automatically. Placing any version of `am335x-pm-firmware.elf` at this location causes it to load automatically during boot. However, due to changes in the upstream kernel it is now required that CONFIG_FW_LOADER_USER_HELPER_FALLBACK be enabled if the CONFIG_WKUP_M3_IPC is being built-in to the kernel so that the firmware can be loaded once userspace and the root filesystem becomes avaiable. The kernel configuration should be as follows, and enables not only the runtime loading of CM3 firmware, but also suspend and resume:

CM3 Firmware Runtime Load Configuration

```
# Firmware Loading from rootfs
CONFIG_FW_LOADER_USER_HELPER=y
CONFIG_FW_LOADER_USER_HELPER_FALLBACK=y

# AMx3 Power Config Options
CONFIG_MAILBOX=y
CONFIG_OMAP2PLUS_MBOX=y
CONFIG_WKUP_M3_RPROC=y
CONFIG_SOC_TI=y
CONFIG_WKUP_M3_IPC=y
CONFIG_TI_EMIF_SRAM=y
CONFIG_AMX3_PM=y

CONFIG_RTC_DRV_OMAP=y
```

Another option is to bake the CM3 firmware directly into the kernel. Using make menuconfig, select `Device Drivers ---> Generic Driver Options`.

In this menu, find the following section and add the appropriate firmware name and directory:

```
...
-*- Userspace firmware loading support
[*] Include in-kernel firmware blobs in the kernel binary
(am335x-pm-firmware.elf) External firmware blobs to build into the kernel binary
(firmware) Firmware blobs root directory
```

During the boot process, if the CM3 firmware was properly loaded, you will see the following message:

CM3 Firmware Load Successful

```
wkup_m3_ipc 44e11324.wkup_m3_ipc: CM3 Firmware Version = 0x191
```

wkup_m3_rproc Driver

This driver is responsible for loading and booting the CM3 firmware on the wkup_m3 inside the SoC using the remoteproc framework.

Source Location:

The source code for this driver can be found in:

```
drivers/remoteproc/wkup_m3_rproc.c
```

wkup_m3_ipc Driver

This driver exposes an API to be used by the PM code to provide board and SoC specific data from the kernel to the CM3 firmware, request certain power state transitions, and query the status of any previous power state transitions performed by the CM3 firmware.

Source Location:

The source code for this driver can be found in:

```
drivers/soc/ti/wkup_m3_ipc.c
```

3.1.3.3 Deep Sleep Voltage Scaling

It is possible to scale the voltages on both the MPU and CORE supply rails down to 0.95 V when in DeepSleep once powerdomains are shut off. The I2C sequences need to scale voltage varies from board to board and are dependent on which PMIC is in use, so board-specific binaries are used that are passed to the CM3 firmware to define the sequences needed during the sleep and wake paths. The CM3 firmware is then able to write these sequences out at the proper location in the Deep Sleep path on I2C0. It is not necessary to regenerate this blob if using a power tree based off of an existing TI EVM; however, if the system has a different PMIC or power management system, it may be necessary to reconfigure this blob for your specific device. Binary blobs are available for the following AM335x designs:

- am335x-evm-scale-data.bin
  - AM335x GP EVM
  - AM335x Starterkit
- am335x-bone-scale-data.bin
  - AM335x Beaglebone
  - AM335x Beaglebone Black

The name of the binary blob used can be configured in the Device Tree and is detailed in the next section.

If a different combination of processor and PMIC are used in a system, it would be necessary to create your own binary blob in order to enable this functionality.

Scale Data Format

Each binary file contains a small header with a magic number and offsets to the sleep and wake sections. The sleep and wake sections themselves consist of two bytes to specify the I2C bus speed for the operation and blocks of bytes that specify the message. The header is 4 bytes long and is shown here:

Size (bytes)	Field
2	Magic Number (0x0C57)
1	Offset to sleep data
1	Offset to wake data

Offsets for sleep and wake data are in bytes starting at the first byte after the header, with the first byte being counted as 0.

Message structure for each section is as follows:

Size (bytes)	Field
2	Bus speed in KHz (little endian)
1	Message size, counting from first byte after I2C Bus address
1	I2C Bus address
1	First byte of message (typically I2C register address)
1	Second byte of message (typically value to write to register)
1	Nth byte of message

Example Binary Blob and Explanation

```
### Explanation Of Values ###

0c57        # Magic number
00           # Offset from first byte after header to sleep section
06          # Offset from first byte after header to wake section

0034        # Sleep sequence section, starts with two bytes to describe i2c bus in khz (100)
02 2d 25 1f # Length of message, evm i2c bus addr, then message (i2c reg 0x25, write value 0x1f)

0034        # Wake sequence section, starts with two bytes to describe i2c bus in khz (100)
02 2d 25 2b # Length of message, evm i2c bus addr, then message (i2c reg 0x25, write value 0x2b)
```

Multiple messages can be written, one after another, if more than one address needs to be written.

3.1.4 Device Tree Configuration

Device tree is a tree structure with nodes and properties; it is used by Linux as a way of describing the hardware the Linux kernel is running on without having to modify kernel source code. Nodes can be used to describe entire platforms such as the AM335x evm, processors, external peripherals, and internal modules. Each node has a set of key, value pair properties that can be defined by the system designer, and is interpreted by Linux. Therefore, device tree can be used to describe almost every aspect of the hardware in the SOC, and as long as Linux has a defined behavior for the device specified, device tree can be used to control the configuration of hardware on a system. Unused peripherals can be disabled in the device tree. Certain interfaces may require a different pin configuration when in system sleep or suspend, these different pin configurations can be controlled through the device tree pinmux. Finally, OPPs can be defined to enable more clocking speeds or combinations than exist in the default device tree.

3.1.4.1 Disabling Peripherals in Device Tree

When designing for low power, it is always a good idea to use only the peripherals necessary to support your specific application. However, simply ignoring the other peripherals on the device is not a good idea since Processor SDK leaves many peripherals turned on by default in order to showcase the full functionality of the device. Even when idle, most peripherals consume 2 mW or more simply from being clocked, thus, it is important to tell Linux to turn off power and clock gate unused peripherals.

In general, all that is needed to turn off a module on the AM335x is to mark its status in device tree as "disabled". As long as the default reset mode of the module is to be in idle, then the module is never powered on during the Linux startup sequence. Devices such as USB that do not default to a sleep state require further steps to turn the module off. For the default power and clocking settings for each module, see to the device-specific TRM.

Device tree also allows for an include functionality where additional nodes can be described in Device Tree Source Include files. Be sure to check the am33xx.dtsi for other peripherals that are not described in the `am335x-evm.dts` file.

LCD Panel Device Tree Fragment

```
panel {
   compatible = "ti,tilcdc,panel";
   status = "okay";
   pinctrl-names = "default";
   pinctrl-0 = <&lcd_pins_s0>;
   panel-info {
      ...
   };
      display-timings {
      800x480p62 {
         ...
      };
   };
};
```

This peripheral could be turned off by changing line 3 to "disabled" giving a fragment that looks like the following.

LCD Panel Disabled Device Tree Fragment

```
panel {
   compatible = "ti,tilcdc,panel";
   status = "disabled";
   pinctrl-names = "default";
   pinctrl-0 = <&lcd_pins_s0>;
   panel-info {
      ...
   };
   display-timings {
      ...
   };
};
```

Or, the module can be disabled with a subsequent reference using an ampersand and label:

LCD Panel Device Tree Reference

```
&panel {
     status = "disabled";
};
```

3.1.4.2 Defining Pinmux States in Device Tree

In order to optimize power on the I/O supply rails during low-power modes (for example, standby, suspend), each pin can be given a "sleep" configuration in addition to its run-time configuration. Sleep pin states should be defined to align with external pulls to avoid contention. Pinctrl states are defined in the board device tree for each peripheral and are used to configure the PAD_CONF registers found in the control module of the device, which allow for selection of the MUXMODE of the pin and the operation of the internal pull resistor. Typically, a device defines its pinctrl state for normal operation:

MDIO Device Tree Default Pin Configuration

```
davinci_mdio_default: davinci_mdio_default {
    pinctrl-single,pins = <
        /* MDIO */
        0x148 (PIN_INPUT_PULLUP | SLEWCTRL_FAST | MUX_MODE0)    /* mdio_data.mdio_data */
        0x14c (PIN_OUTPUT_PULLUP | MUX_MODE0)                   /* mdio_clk.mdio_clk */
    >;
};
```

In order to define a sleep state for the same device, another pinctrl state can be defined:

MDIO Device Tree Sleep Pinmux

```
davinci_mdio_sleep: davinci_mdio_sleep {
    pinctrl-single,pins = <
        /* MDIO reset value */
        0x148 (PIN_INPUT_PULLDOWN | MUX_MODE7)
        0x14c (PIN_INPUT_PULLDOWN | MUX_MODE7)
    >;
};
```

Then, the two states for the driver are defined:

MDIO Device Tree Driver Fragment

```
&davinci_mdio {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&davinci_mdio_default>;
    pinctrl-1 = <&davinci_mdio_sleep>;
    status = "okay";
};
```

Depending on the driver, it may be necessary to tell the driver which pin state to select on suspend and resume from suspend. This is accomplished by putting a call to `pinctrl_pm_select_sleep_state` at the end of the system suspend handler routine. In addition, it may be necessary during the beginning of the resume from suspend handler to call `pinctrl_pm_select_default_state` in order to restore the proper pin configuration for runtime. Undefined sleep pin states will not cause the driver to crash on suspend or resume; however, if a sleep state is defined there MUST be a default pin configuration, or else the pm driver will select the only valid configuration which is the sleep state.

The required pinctrl states differ from board to board; configuration of each pin is dependent on the specific use of the pin and what it is connected to. Generally, the most desirable configuration is to have an internal pull-down and GPIO mode set that gives minimal leakage. However, in a case where there are external pull-ups connected to the line (like for I2C lines) it makes more sense to disable the pull on the pin. The pins are supplied by several different rails, which are described in the device-specific data manual. By measuring current draw on each of these rails during suspend it may be possible to fine tune the pin configuration for maximum power savings. The AM335x EVM has pinctrl sleep states defined for its peripherals and serves as a good example. In addition, there is the optimizing deep sleep and suspend power workbook that can help identify power consumers during suspend and standby.

Even pins that are not in use and not connected to anything can still leak some power so it is important to consider these pins as well when implementing the pad configuration. This can be accomplished by defining a pinctrl state for unused pins and then assigning it directly to the pinctrl node itself in the board device tree. This allows the state to be configured during boot even though there is no specific driver for these pins:

Device Tree Unused Pins Fragment

```
&am33xx_pinmux {
         pinctrl-names = "default";
         pinctrl-0 = <&unused_wireless>;
         ...
         unused_pins: unused_pins {
                 pinctrl-single,pins = <
                        0x80    (PIN_INPUT_PULLDOWN | MUX_MODE7) /* gpmc_csn1.mmc1_clk */
                        ...
```

3.1.4.3 Controlling VTT

On the AM335x, GPIO bank GPIO0 remains on even during system suspend. This can be used to control power to the VTT regulator if it is used on the system. To do so, the M3 coprocessor needs to be configured as follows:

CM3 VTT Toggle Device Tree Fragment

```
&wkup_m3 {
        ti,needs-vtt-toggle;
        ti,vtt-gpio-pin = <7>;
};
```

3.1.4.4 Deep Sleep Voltage Scaling

In order to enable deep sleep voltage scaling, the CM3 module needs to be told which binary blob to execute during a suspend/ resume cycle. This is defined in the wkup_m3_ipc node.

```
/* From arch/arm/boot/dts/am335x-evm.dts */
&wkup_m3_ipc {
        ti,scale-data-fw = "am335x-evm-scale-data.bin";
};
```

3.1.4.5 Adding MPU OPP Table Entries

Starting in Processor SDK 3.01, OPP configuration information is contained in the device tree. For AM335x, this information is defined in the file `am33xx.dtsi`. The first entry of the OPP table from `am33xx.dtsi` is quoted below:

```
cpu0_opp_table: opp_table0 {
   compatible = "operating-points-v2";
   opp50@300000000 {
                            # OPP name and frequency
      opp-hz = /bits/ 64 <300000000>;
                            # Target frequency in Hz
         opp-microvolt = <950000 931000 969000>;
                            # PMIC voltage specification. < target ...
                              minimum maximum >
         opp-supported-hw = <0x06 0x0010>;
                            # Specify which silicon revisions and device ...
                              efuse values support this OPP
         opp-suspend;
                            # flag for lowest OPP setting -> ...
                              target OPP before processor
                              hits suspend
};
...
```

It is possible to modify the OPP table to include your own OPPs, however, improper frequency and voltage settings can cause system instability, or potentially damage the system. Be sure to extensively test frequency and voltage combinations to ensure that they are suitable for the use case and CPU load, and there may be some OPP combinations that are just not stable. In order to have the best chances of success implementing a new OPP, use the following recommendations to guide the process:

- Different OPPs can have the same target voltage, but cannot have the same frequency
- Lowering frequency while maintaining voltage should be safe
- When adding a new target voltage recommended min/max voltages are ±4% of the target voltage
- For safe maximum and minimum voltage levels for the MPU, see the device-specific data sheet.
- To enable the OPP for all platforms and silicon revisions, use the cell field <0xFF 0xFFFF> for the opp-supported-hw property.
  - This cell field matches whichever bits actually happen to be set on the device.
  - If this is used, be sure to check that there are no frequency conflicts in the OPP table.
  - For specific revision bits, check the DEVICE_ID register values in the TRM

To test new entries in the OPP table:

- Check the cpufreq interface in sysfs. If your OPP frequency is displayed, then the device tree table was successfully updated.
- Export the CPU governor to userspace and select your new OPP as a target frequency. At this point, if the board is designed for measurability, the target voltage should be observed on the VDD_MPU voltage rail. In addtion, the BogoMIPS value in CPU info should be approximately your clock speed in MHz.

- As a last test, a CPU benchmark like Dhrystone should scale linearly and can be used as a yardstick by referencing results from known good OPPs.

```
# First check the cpufreq interface in sysfs
root@am335x-evm:~# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
300000 <your opp> 600000 720000 800000 1000000
# Set governor to userspace
root@am335x-evm:~# echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
# Set target OPP to your OPP
root@am335x-evm:~# echo <your opp> > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
# Check BogoMIPS
root@am335x-evm:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev2 (v7l)
BogoMIPS        : 717.22    # 717.22 BogoMIPS indicates CPU at 720 MHz
...
```

3.2 Userspace Power Management

The TI Linux Processor SDK kernel comes configured with several features that can be accessed in userspace to control the power and performance characteristics of the system. First, the scaling governors and the Dynamic Voltage and Frequency Scaling (DVFS) system. Second, the CPUIdle driver that targets C-states based on the current governor. Third, this section discusses how to power or clock gate unused peripherals while the system is running.

3.2.1 Dynamic Voltage and Frequency Scaling

Dynamic Voltage and Frequency Scaling (DVFS) allows for the clocking up and down of MPU frequency in response to changing loads on the MPU. Frequency and voltage combinations are known as Operating Performance Points (OPPs) that are defined in the device tree. Transition policies are defined using scaling "governors," which select an OPP based on CPU load and usage patterns. There are several pre-defined governors that can be selected during kernel configuration:

- **ondemand:** This governor samples the load of the cpu and scales it up aggressively in order to provide the proper amount of processing power.
- **conservative:** This governor is similar to ondemand but uses a less aggressive method of increasing the the OPP of the MPU.
- **performance:** This governor statically sets the OPP of the MPU to the highest possible frequency.
- **powersave:** This governor statically sets the OPP of the MPU to the lowest possible frequency.
- **userspace:** This governor allows the user to set the desired OPP using any value found within scaling_available_frequencies by echoing it into scaling_setspeed.

Available governors vary by configuration, and can be checked using sysfs:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
conservative userspace powersave ondemand performance
```

The current governor can be viewed by using:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
ondemand
```

To select a different governor:

```
$ echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

To view the current OPP:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
300000
```

To view available OPPs:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
300000 600000 720000 800000 1000000
```

To manually set the OPP, first you must be in userspace governor. Then write the desired MPU frequency:

```
$ echo 600000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

Linux also offers a way to bound the frequency range of the scaling governors. Use scaling_min_freq and scaling_max_freq to set the minimum and maximum frequencies to which the CPU will scale. The following command would cap MPU speed at 600 MHz.

```
$ echo 600000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

3.2.2 CPU Idle Interface

During periods where the MPU is inactive, the CPUIdle driver will place the MPU in a low power state, as determined by the CPUIdle governor. The governor is called whenever the idle loop is called, which happens when the Linux thread scheduler has no more tasks to run.

For information regarding CPU Idle Linux kernel driver of the most recent release, and an archive of previous releases, see the Linux_Core_Power_Management_User's_Guide wiki page.

The availability of CPUIdle governor is determined by your kernel configuration. There are no necessary actions to enable CPUIdle, so long as it is enabled in the kernel, it operates transparently in the background.

CPUIdle offers a sysfs entry that provides some statistics on the usage of each of the idle states:

```
root@am335x-evm:~# ls -l /sys/devices/system/cpu/cpu0/cpuidle/state0
-r--r--r--    1 root     root          4096 Dec 20 16:49 desc
-rw-r--r--    1 root     root          4096 Dec 20 16:49 disable
-r--r--r--    1 root     root          4096 Dec 20 16:49 latency
-r--r--r--    1 root     root          4096 Dec 20 16:49 name
-r--r--r--    1 root     root          4096 Dec 20 16:49 power
-r--r--r--    1 root     root          4096 Dec 20 16:49 residency
-r--r--r--    1 root     root          4096 Dec 20 16:49 time
-r--r--r--    1 root     root          4096 Dec 20 16:49 usage
root@am335x-evm:~# ls -l /sys/devices/system/cpu/cpu0/cpuidle/state1
-r--r--r--    1 root     root          4096 Dec 20 16:49 desc
-rw-r--r--    1 root     root          4096 Dec 20 16:49 disable
-r--r--r--    1 root     root          4096 Dec 20 16:49 latency
-r--r--r--    1 root     root          4096 Dec 20 16:49 name
-r--r--r--    1 root     root          4096 Dec 20 16:49 power
-r--r--r--    1 root     root          4096 Dec 20 16:49 residency
-r--r--r--    1 root     root          4096 Dec 20 16:49 time
-r--r--r--    1 root     root          4096 Dec 20 16:49 usage
```

3.2.3 System Suspend

Using the CM3 module on the AM335x, it is possible for the system to go into DeepSleep0 (DS0) or a suspend state where many peripherals are powered down. In DS0, all peripherals except for those in the PM_WKUP domain will be powered off. For a list of the peripherals in this domain, see Section 3.3.3.1.

When in DS0, the following are valid wakeup sources for the AM335x GP EVM:

- UART0
- GPIO0
- Touchscreen

To enter DS0:

```
$ echo mem > /sys/power/state
```

To enter suspend:

```
$ echo standby > /sys/power/state
```

The PM driver will notify the user if there was an issue entering either of these states.

3.2.4 Power and Clock Gating Peripherals

Certain peripherals can be controlled in userspace via simple command line instructions. This should really only be used if it is absolutely critical that certain peripherals can be turned on and off while the system is running, or for a quick and dirty experiment to determine power savings from shutting down a particular module. If a module should be disabled permanently, it should be done in device tree and kernel configurations, not only is this the simplest and cleanest way to disable peripherals, but it actually will help optimize kernel boot times as well since the Linux kernel won't load drivers for peripherals that are disabled.

3.2.4.1 PRU-ICSS

The PRU can be temporarily turned off using a sequence of terminal commands and memory writes. However, if the device is still enabled in the device tree, going to suspend with the PRU off will cause a system crash since the resume driver will attempt to communicate with the PRU and fail.

PRU Poweroff Sequence

```
rmmod rpmsg_pru
rmmod virtio_rpmsg_bus
devmem2 0x44E000E8 w 0x0 # disable module clocks
devmem2 0x44E00140 w 0x1 # request SW_SLEEP transition
devmem2 0x44E00C00 w 0x2 # assert SW_RST
devmem2 0x44E00C00 w 0x0 # de-assert SW_RST
```

3.2.4.2 Ethernet

Ethernet can be controlled entirely through ifconfig:

```
$ ifdown eth0
```

3.3 Diagnosing Device Power Consumption

Power optimization is often a compromise, often functionality or performance is traded for the sake of power budget. It is important to verify that these sacrifices actually result in power savings. In order to reliably document the power optimization process, a three phase method is recommended. First is to establish a baseline power consumption and develop an expected power reduction figure. Second, make the desired changes and document the power consumption. Third, if the power reduction was as desired, confirm the hardware reflects the desired changes as well, otherwise, examine the hardware configuration to determine other places for potential power savings, or to diagnose which optimization did not work as intended.

3.3.1 Expected Power Consumption Estimation

When debugging a system's power consumption, it is important to know what your expected power consumption is for each use case. This way it can be determined if the system is operating normally, or if there is something actually wrong with the device.

During board bringup, it is recommended that a baseline power consumption is established for various use cases, current power consumption figures from the Linux Kernel Performance wiki page is a good reference for un-optimized out of box numbers, and section 4 of this document is useful as an optimized example for selected use cases. In addition, the Power Estimation Tool enables a more tailored benchmark for the peripheral and interface loadout on the actual system.

3.3.2 Measuring Actual Power Consumption

It is assumed that the system has some provision for measuring the SOC power rails directly. Although system power is also important, some of the SoC level optimizations can get lost in the noise of the system as a whole and it is highly recommended that system power is not the only metric used when optimizing SoC power consumption.

3.3.2.1 Shunt Resistors

Measuring voltage across shunt resistors is relatively straightforward, it is important that the multimeter used has enough resolution and range to capture meaningful voltage readings, typically expect tenths of milivolts to single digit milivolt range.

Since typically using shunts involve some aspect of manual measurement, be sure that the power measured actually corresponds to the desired state or behavior. Checking this can be simple as console output, or using a GPIO or LED toggle to indicate different modes.

3.3.2.2 INA Devices

Since INAs are I2C devices, it is possible to access the devices directly over the I2C bus. There are also several software tools created with this functionality. A simple bash script for reading the INA226, and the open source powertool will be discussed. Depending on the hardware setup, an external I2C master may or may not be required. For power consumption measurement when SOC is inactive, an external device is required.

3.3.2.2.1 Powertool

Powertool is an open source tool for reading INA226 devices on an I2C bus. The source code for Powertool can be found on github. Powertool requires the FTDI USB to I2C adapter from Spectrum Digital, or one can use powertool directly from a device connected to the I2C bus with the relevant INA devices. On the AM335x EVM, the INA devices are on an external I2C bus, and therefore require an external device, such as a BeagleBone Black or the FTDI USB I2C adapter to sample power data.

Powertool uses configuration files to define measured rails, several Texas Instruments EVMs are supported out of the box, including the AM335x GP EVM. For specific syntax and build requirements, see the powertool documentation.

3.3.2.2.2 Manual I2C Data Collection

An example shell script is included in the Appendix A to demonstrate that, if desired, INA devices can be read directly from the I2C bus.

3.3.3 Confirm and Validate Hardware Changes

As changes are made to the default configuration, verification should be done to ensure that the changes had the desired effect without any unintended consequences. This section highlights two tools used to read the power and clock state of the underlying hardware. The included Power Reset Clock Management register dump script comes with set of tools for interpreting the output. Omapconf is another utility that provides more detailed information and can be used to verify the state of modules on the AM335x SOC.

3.3.3.1 PRCM Register Dump

The Power Reset Clock Management (PRCM) module contains several status registers for each of the various power and clocking domains. Table 2 summarizes the power domains on the AM335x. For a detailed breakdown of the clocking structure of the AM335x, see the clock tree tool. And, for further information on both power and clocking, see the *PRCM* section of the device-specific TRM. For the most part, these registers are accessible through userspace, and can be key to providing insight on which peripherals are actually active. An example set of tools to help the user collect and visualize the PRCM data is available at this wiki page: AM335x_PRCM_Tools.

Table 2. Power Domains on the AM335x

Power Supply	Power Domain	Modules
VDD_CORE	PD_WKUP	CM3, PRCM, Control Module, GPIO0, DMTIMER0_dmc, DMTIMER1, UART0, I2C0, TSC, WDT1, SmartReflex_c2, DDR_PHY, WKUP_DFTSS, Debugs, VDD of CORE_PLL, PER_PLL, Display PLL, and DDR_PLL, Emulation, VDD of I/O, RC Oscillator
	PD_PER	L3, L4_PER, L4_Fast, EMIF4, EDMA, GPMC, OCMC Controller, L3/ L4_PER/ L4_Fast Peripherals, PRU-ICSS, LCD Controller, Ethernet Switch, USB Controller, GPMC, MMC 0-2, DMTIMER 2-7, UART 1-5, SPI 0-1, I2C 1-2, DCAN 0-1, McASP 0-1, ePWM 0-2, eCAP 0-2, eQEP 0-1, GPIO 1-3, ELM, Mailbox 0, Spinlock, OCP_WP, USB2 PHY CORE (digital section), USB2PHYCM (digital section)
	PD_GFX	SGX530 (GPU)
VDD_MPU	PD_MPU	CPU, L1 L2 of MPU
	PD_WKUP	Interrupt controller of MPU, MPU PLL (digital section)
VDD_RTC	PD_RTC	RTC, VDD for 32,768 Hz Crystal Osc, VDD I/O for alarm pin

A suite of tools have been created to make the process of reading and decoding the meaning of these registers easier. These tools are accessible on bitbucket. The sitara power tools contain three key components:

- **sitara_pwrst.sh**: Takes a list of register names and addresses and uses devmem2 to read the register values at the specified addresses. Writes output to *.rd1 files.
- **AMXXXX_reglist.txt**: A text file with a named register and address pair on every line with name and address separated by whitespace, for example, PRM_PER_PM_PER_PWRSTST 0x44E00C08
- **amXXXX_prcm_dump.xlsxm**: Excel workbook takes copy and paste text from *.rd1 files and visualizes power and clock domains.

3.3.3.2 OMAPCONF

omapconf is a built-in utility that allows for the reading of various clock configuration and power state registers. omapconf can also be used to access I2C devices and read and write device registers on an individual basis. Here some useful functions of omapconf:

- **omapconf export ctt** [filename] - this writes out a file compatible with the clock tree tool for further visualization of the clock tree. If no filename is specified, this command will print output to stdout.
- **omapconf show dpll** - this command prints a table showing the state of each PLL.
- **omapconf show opp** - this command prints a table showing the current OPP
- **omapconf show pwst** - this command outputs the power state of each power domain.

It may be necessary to tell omapconf which processor it is running on, to do this use the option `--force <cpu>`

Relevant cpu values are: `am3352, am3354, am3356, am3357, am3358, am3359`

For other omapconf commands, see the output of `omapconf --help`.

3.3.4 Linux Debug Features

Linux provides some interfaces to diagnose system issues, although none of them are specifically oriented towards power, they can provide some useful insight on what Linux is trying to do with the underlying hardware.

3.3.4.1 Checking Kernel Boot Arguments

If kernel boot arguments are modified to enable certain features, it might be necessary to confirm the exact boot command line argument used. This information is available in `/proc/cmdline`:

```
$ cat /proc/cmdline
console=tty0 console=ttyO0,115200n8 root=/dev/mmcblk0p1 rootfstype=ext4 rootwait coherent_pool=1M
quiet cape_universal=enable
```

3.3.4.2 Serial Console Output During Suspend/Resume

Sometimes it is helpful when debugging a suspend or resume issue to have console output throughout the whole process. To enable this, add no_console_suspend to the kernel boot arguments. This is most easily done by modifying the bootargs variable in U-Boot.

Note that if this is used then the serial port cannot be used as a wakeup source for the processor.

3.3.4.3 Sysfs Debug Interface

The kernel exposes debug information using the sysfs system. This information is available in `/sys/kernel/debug`. One useful interface here is the clock summary:

```
$ cat /sys/kernel/debug/clk/clk_summary
```

This displays every clock in the system, and whether or not it has any consumers.

3.3.4.4 Low Power Debugging

During development, debugging suspend/ resume issues can be challenging due to the nature of low power modes, and limited debugging ability. The wiki link below is geared toward Linux developers using the AM335x family of embedded processors. Many of the topics contained in the page, however, will still apply to users of other operating systems. The goal of this page is to provide some insight into common issues they might bump into, as well as tools to help identify issues. For more information, see Debugging_AM335x_Suspend-Resume_Issues.

4 Example Low Power Use Cases

This section highlights low power use cases using the techniques presented in Section 2 and Section 3.

Use cases that require low power while only needing a subset of SoC peripherals are prime candidates for targeted system power optimization. Modifications to the processor's internal bus speeds (L3, DDR, and so forth) and disabling unused peripherals are some of the basic knobs available to the end user.

The AM335x EVM and Processor SDK are presented as a baseline starting point. The EVM and Processor SDK are designed to showcase full functionality and performance, thus, it is not optimized for any specific application.

To demonstrate the low power capabilities of the AM335x, two optimized device trees for the AM335x GP EVM are used to disable many of the unnecessary peripherals. Although these examples will not cover all of the optimizations possible, the optimizations used are easily reproducible and can serve as a good starting point for an engineer creating a low power system. The process of selecting these device trees is documented and some runtime adjustments that may be needed, so that the designer can follow along and try out these optimizations for themselves.

Using the device tree optimizations, a hypothetical minimal use case is defined where a large portion of peripherals are disabled and the power consumption is recorded. From here, specific sets of peripherals are enabled to demonstrate increasing levels of functionality, while documenting the increasing power demands of the system. By enabling devices on an "as-needed" basis, it is ensured that no extra peripherals are there to consume power without adding any functionality to the system. In an actual design, the developer must go through and make their own decisions on what a minimal system may look like. For a true low power system, such an approach is recommended.

From the baseline system the following scenarios are covered:

- Baseline OS Idle
- Networked OS Idle
- Heavy CPU Load (Dhrystone)
- Heavy Ethernet Traffic (IPerf)
- Multimedia Playback

For un-optimized power figures from similar benchmarks, check the Linux Performance Wiki: Processor SDK Linux Kernel Performance Guide.

4.1 Device Setup Summary

To ensure consistency, the exact devices, software and OPP tables are documented here.

Demonstration Hardware

- AM335x GP EVM
- Modified Beaglebone Black (see below)
- FTDI USB to UART adapter
- Linux Host with TI Linux Procesor SDK v3.0.1 (Kernel 4.4.19) installed
- Keithley 2400 digital multimeter

Hardware Summary

The standard AM335x GP EVM was used for these tests. Additionally, the BeagleBone Black was used to compare power consumption of different DDR topologies.

- Measurements were taken on the bench at room temperature
- BeagleBone Black DDR I/O and memory supply (VDCDC1) was severed from the PMIC at R6, and was sourced and measured externally with a calibrated Keithley 2400-series source meter, using 4-wire sensing.

Software Summary

The TI Linux Processor SDK version 3.0.1 was used with several minor tweaks to both Linux and U-Boot to achieve lower power. The same SD card was used on both AM335x GP EVM and BeagleBone Black.

- U-Boot version 2016.05
  - DDR frequency set to 303MHz for BeagleBone Black to match AM335x GP EVM
- Linux Kernel version 4.4.19
  - Minimal device tree used (Ethernet, UART0, I2C0 enabled, other peripherals marked disabled)

OPP Summary

The AM335x has several pre-defined Operating Performance Points (OPPs). Each OPP specifies a frequency and voltage pairing for the MPU and core power domains. Table 3 summarizes the OPPs used in this document and the voltage and frequency specifications of each.

Table 3. AM335x MPU Supported Operating Performance Points

OPP	Frequency (MHz)	Voltage (V)
	MPU	MPU
OPP50	300	0.95
OPP100	600	1.1
OPP120	720	1.2
Turbo	800	1.26
Nitro	1000	1.325

Table 4. AM335x Core Supported Operating Performance Points

VDD_CORE OPP	VDD_CORE (V)		Frequency (MHz)			
		mDDR	DDR2	DDR3/ DDR3L	L3	L4
OPP50	0.95	90	125	-	100	50
OPP100	1.10	200	266	400	200	100

This OPP data is provided as a reference only for Silicon Revision "A" or higher. For full details on recommended OPPs and limitations, see Section 5.4 in the *AM335x Sitara™ Processors Data Manual*.

Due to errata Advisory 1.0.24 - *Boot: System Boot is Not reliable if Reset is Asserted While Operating in OPP50*, supporting Core OPP50 requires that the voltages remain at OPP100 levels during system reset. In addition, according to errata Advisory 1.0.15 - *ARM Cortex-A8: OPP50 on MPU Domain Not Supported running Core OPP50*, running Core OPP50 requires OPP100 voltage levels to ensure stability, this applies to the Sillicon revision 1.0 hardware only. More information about these errata can be found in the *AM335x Sitara™ Processors Silicon Revisions 2.1, 2.0, 1.0 Silicon Errata*.

All test cases are run at Core OPP100.

To specify the MPU OPP, control over the CPU frequency must be given to userspace, and then the desired frequency can be selected:

Userspace MPU Frequency Selection

```
$ echo userspace > /sys/device/system/cpu/cpu0/cpufreq/scaling_governor
$ echo $FREQ_IN_KHZ > /sys/device/system/cpu/cpu0/cpufreq/scaling_setspeed
```

To view available frequencies:

Show Available MPU Frequencies

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

AM335x GP EVM vs BeagleBone Black DDR Topology

The AM335x GP EVM has been designed to showcase the broad feature set of the AM335x processor, and has not been optimized for low power consumption. The BeagleBone Black external memory interface design is better for low-power applications. Table 5 highlights the differences between the two platforms.

Table 5. AM335x GP EVM vs. BeagleBone Black DDR Design

Feature	AM335x GP EVM	BeagleBone Black	Benefit
Density	2 x 512 MB	1 x 512 MB	Reducing memory chips and memory density saves power
Frequency	303 MHz	303 MHz	Lower frequency reduces power consumption at expense of bandwidth
Technology	DDR3	DDR3L	
Topology	Flyby with VTT	Terminationless point-to-point	Eliminating VTT significantly reduces power consumption
Voltage	1.5	1.35	Lower operating voltage reduces power consumption

4.2 Optimizations Used for Low Power Use Cases

In order to attain these low power states, two device tree configurations were used. The first is called the am335x-evm-powersave device tree and the second is a modification of this device tree that allows for a multimedia use case, named am335x-evm-powersave-multimedia. The multimedia use case turns on all the modules required for an LCD display and audio output. Additionally, ethernet was disabled except for the network test cases. Table 6 summarizes the configurations used in each test case.

Table 6. Device Tree and Peripherals Use

Test Case	Device Tree	eth0	SGX
OS Idle	am335x-evm-powersave	No	No
OS Idle with Ethernet	am335x-evm-powersave	Yes	No
IPerf	am335x-evm-powersave	Yes	No
Dhrystone	am335x-evm-powersave	No	No
Multimedia	am335x-evm-powersave-multimedia	No	No

Table 7. Peripherals Disabled by Device Tree

Device Tree	Devices Marked "Disabled"
am335x-evm-powersave	backlight, panel (lcd), lcdc, usb, usb_ctrl_mod, usb0_phy, usb1_phy, usb0, usb1, elm, epwmss0, gpmc, mcasp1, sham, aes, sgx, pruss
am335x-evm-powersave-multimedia	usb, usb_ctrl_mod, usb0_phy, usb1_phy, usb0, usb1, elm, epwmss0, gpmc, sham, aes, sgx, pruss

4.2.1 Device Tree Binary

4.2.1.1 Compiling the Device Tree Binary (DTB)

Using the included device tree source, it is simple to compile the device tree binary in Processor SDK.

Place the device tree source (DTS) file in the same directory as the other DTS files found here:

```
<sdk path>/board-support/linux-4.4.19*/arch/arm/boot/dts
```

Update your path as follows to include the arm gcc cross compilers:

```
export PATH=<sdk path>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin:$PAT
```

In your sdk root directory, invoke make with the following command:

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x-evm-powersave.dtb
```

Where the target is the name of the DTB to be generated from a DTS of the corresponding name.

Once this is complete copy the compiled DTB, which will be in the DTS source folder, to the boot directory of the EVM.

4.2.1.2 Loading the DTB

U-Boot loads the device tree for the Linux kernel. The following instructions are for the default out-of-box U-Boot environment that boots off of an SD card through the MMC0 interface. If your configuration differs from this, further modifications may be necessary. The key is to ensure that U-Boot loads the powersave device tree instead of another device tree, which may be present in the boot media.

First compile the device tree sources included (see Appendix A), and place the generated DTBs in the boot directory of the root file system partition on the SD card.

When powering on the evm, enter the U-Boot command line by pressing the space bar.

Use the following command sequence to select the desired device tree:

```
=> env default -f -a                        # load default boot environment
=> printenv bootcmd                         # show the default boot command
boodcmd=run findfdt;...
=> setenv bootcmd '....'                    # change the default boot command to not run findfdt
=> setenv fdtfile am335x-evm-powersave.dtb  # manually set the fdtfile desired
=> saveenv                                  # (Optional) save this as the default boot
environment
=> boot                                     # Boot as usual
```

4.2.2 Ethernet

For test cases that do not require ethernet, use ifconfig to bring down the ethernet interface. Ethernet is enabled by default to support network boot. The ethernet is not disabled so that a runtime optimization can be demonstrated.

```
root@am335-evm:~# ifconfig eth0 down
```

4.2.3 Graphics (SGX)

Although SGX is disabled in device tree, its hardware default configuration is to be powered on. To realize the lowest power state, it must be turned off using devmem. Note that if this is done when SGX is enabled in device tree, it causes a kernel panic.

```
root@am335-evm:~# devmem2 0x44E01100 w 0x0 # Poweroff state on next reset
root@am335-evm:~# devmem2 0x44E01104 w 0x1 # Assert reset
root@am335-evm:~# devmem2 0x44E01104 w 0x0 # De-assert reset
```

4.3 Optimized Idle States

This is the amount of power consumed by the SoC at runtime, running the Linux Kernel, but no active tasks. Since the MPU is clock gated in this state, power consumption does not increase significantly with the scaling of CPU frequency.

4.3.1 OS Idle

Device Tree Name	Ethernet	SGX
am335x-evm-powersave.dtb	No	No

This is the lowest power consumption state, with the powersave device tree binary, and no Ethernet or SGX enabled.

Table 8. Optimized OS Idle Power Consumption

Power Rail	Voltage (V) (1)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
vdd_core_power	1.1	148.71	148.73	148.75	148.74	148.73
vdd_mpu_power	DVFS (0.95 - 1.325)	1.64	3.89	4.46	5.98	7.92
1.8 V I/O	1.8	32.22	33.69	34.66	35.41	37.05
3.3 V I/O	3.3	20.85	20.84	20.85	20.86	20.84
SoC Power without DDR3		203.42	207.16	208.71	210.99	214.55
vdds_ddr (2)	1.5	139.52	141.96	138.33	138.16	140.73
vddsddrmem	1.5	94.40	94.19	93.72	92.37	92.21
DDR3 Total Power (includes VTT)		233.92	236.15	232.05	230.54	232.94
Total Power with VTT		437.34	443.31	440.76	441.53	447.49

(1) Voltages reported are nominal and will vary in actual measurements.

(2) AM335x GP EVM PMIC does not support 1.35V output for PMIC rail - power optimized designs should select proper PMIC.

Using DDR3L data sourced from the BeagleBone Black, the power consumption of a VTT-less DDR3L system can be approximated. VDCDC1 includes both the DDR3L memory power as well as the EMIF interface on the AM335x. SoC power from AM335x GP EVM data summarized in Table 8.

Table 9. Power Consumed by DDR on the BeagleBone Black

Power Rail	Voltage (V) (1)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
DCDC1 (DDR+I/O)	1.35 (2)	40.31	40.21	40.11	40.37	40.19
SoC Power		203.42	207.16	208.71	210.99	214.55
Total Power (SoC + DDR3L) without VTT		243.73	247.37	248.82	251.36	254.74

(1) All voltages listed are nominal. Rail voltages will vary based on operating conditions and should always be measured when collecting power data.

(2) Sourced externally by Keithley 2400 source.

4.3.2 Networked OS Idle

Device Tree Name	Ethernet	SGX
am335x-gp-evm-powersave.dtb	Yes	No

Table 10 is an example showing that the ethernet is enabled and setup to ping on 1 second intervals to ensure ethernet is active. However, the system is still mostly idle; therefore, this configuration also demonstrates little power increase as clock speed goes up.

Table 10. Optimized Networked OS Idle Power Consumption

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
vdd_core_power	1.1	157.94	157.95	157.98	157.98	157.96
vdd_mpu_power	DVFS (0.95 - 1.325)	3.70	4.12	5.15	10.28	9.71
1.8V I/O	1.8	36.40	37.90	38.83	39.61	41.22
3.3V I/O	3.3	69.12	69.12	69.14	69.52	69.13
SoC Power without DDR3		267.17	269.11	271.10	271.40	278.03
vdds_ddr	1.5	138.30	135.35	137.45	140.39	124.36
vddsddrmem	1.5	94.47	94.97	94.41	92.18	92.42
DDR3 Total Power (Includes VTT)		231.16	230.91	226.72	234.87	238.00
Total Power (SoC + DDR3) with VTT		498.33	500.02	497.82	512.27	516.02

Using DDR3L data sourced from the BeagleBone Black, the power consumption of a VTT-less DDR3L system can be approximated. VDCDC1 includes both the DDR3L memory power as well as the EMIF interface on the AM335x. SoC power from AM335x GP EVM data summarized in Table 10.

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
DCDC1 (DDR+I/O)	1.35	39.98	39.86	39.89	39.90	40.04
SoC Power		267.17	269.11	271.10	271.10	278.03
Total Power (SoC + DDR3L) without VTT		307.15	308.97	310.99	311.30	318.07

4.4 Optimized Systems Under Load

Now that you know what these systems are like at idle, what happens when they are required to perform computation or communicate with the outside world? The Dhrystone and IPerf cases use the same setup as the OS Idle, and Networked OS Idle cases, respectively. The only difference here is that in Dhrystone, a load was put on the MPU and memory, and in IPerf, MPU, I/O, and memory power domains are all involved.

4.4.1 Dhrystone

Device Tree Name	Ethernet	SGX
am335x-evm-powersave.dtb	No	No

Dhrystone version run is the standard compiled version that comes with Processor SDK.

Table 11. Optimized Dhrystone Power Consumption

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
vdd_core_power	1.1	153.77	158.40	159.86	160.81	164.42
vdd_mpu_power	DVFS (0.95 - 1.325)	106.79	282.56	407.28	502.32	680.71
1.8V I/O	1.8	34.74	38.70	40.68	41.73	45.16
3.3V I/O[1]	3.3	20.83	20.78	20.75	20.75	20.72
SoC Power without DDR3		316.13	500.44	628.57	725.61	911.01
vdds_ddr	1.5	134.43	137.10	140.77	150.13	148.35
vddsddrmem	1.5	117.22	131.58	130.30	153.40	159.81
DDR3 Total Power (Includes VTT)		251.65	268.68	271.07	303.53	308.15
Total Power (SoC + DDR3) with VTT		567.78	769.12	899.64	1029.17	1219.17

Using DDR3L data sourced from the BeagleBone Black, the power consumption of a VTT-less DDR3L system can be approximated. VDCDC1 includes both the DDR3L memory power as well as the EMIF interface on the AM335x. SoC power from AM335x GP EVM data summarized in Table 11.

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
DCDC1 (DDR+I/O)	1.35	56.27	69.80	75.15	79.04	106.56
SoC Power		316.13	500.44	628.57	725.61	911.01
Total Power (SoC + DDR3L) without VTT		372.40	570.24	703.72	804.65	1017.57

4.4.2 IPerf

Device Tree Name	Ethernet	SGX
am335x-gp-evm-powersave.dtb	Yes	No

AM335x EVM is configured as client UDP, PC as UDP server transmitting 100 Mbps. AM335x is able to handling 100 Mbps at MPU OPP50.

Table 12. Optimized Network Load Power Consumption

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
vdd_core_power	1.1	173.22	174.35	175.01	174.73	175.04
vdd_mpu_power	DVFS (0.95 - 1.325)	81.75	201.77	269.82	321.49	419.29
1.8V I/O	1.8	35.87	38.33	39.29	39.89	41.88
3.3V I/O	3.3	55.76	78.62	78.86	78.62	78.82
SoC Power without DDR3		372.84	493.08	562.99	614.72	715.02
vdds_mem	1.5	136.62	143.73	145.76	147.08	146.33
vddsddrmem	1.5	94.47	94.97	94.41	92.18	92.42
DDR3 Total Power (Includes VTT)		231.08	240.17	240.17	239.26	238.75
Total Power (SoC + DDR3) with VTT		603.92	730.78	803.17	853.98	953.77

Using DDR3L data sourced from the BeagleBone Black, the power consumption of a VTT-less DDR3L system can be approximated. VDCDC1 includes both the DDR3L memory power as well as the EMIF interface on the AM335x. SoC power from AM335x GP EVM data summarized in Table 12.

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) Power (mW)	MPU OPP100 (600 MHz) Power (mW)	MPU OPP120 (720 MHz) Power (mW)	MPU OPP Turbo (800 MHz) Power (mW)	MPU OPP Nitro (1000 MHz) Power (mW)
DCDC1 (DDR+I/O)	1.35	52.29	53.12	53.15	52.84	52.20
SoC Power		372.84	492.08	562.99	614.72	715.02
Total Power (SoC + DDR3L) without VTT		425.13	545.20	616.14	667.56	767.22

4.5 Multimedia

Device Tree Name	Ethernet	SGX
am335x-gp-evm-powersave.dtb	No	No

Multimedia uses the standard `runMpeg4AacDecode.sh` script that is included with all Processor SDK distributions. It simply configures the AM335x GP EVM to play a short clip from a movie trailer and output audio onto a line out jack. LCD brightness and audio levels were left at the default settings.

In order to support multimedia display the following peripherals were enabled:

- LCD panel
- PWM backlight
- LCD controller
- Touch screen controller (adc)
- McASP

Table 13. Optimized Multimedia Playback Power Consumption

Power Rail	Voltage (V)	MPU OPP50 (300 MHz) [1]	MPU OPP100 (600 MHz)	MPU OPP120 (720 MHz)	MPU OPP Turbo (800 MHz)	MPU OPP Nitro (1000 MHz)
		Power (mW)	Power (mW)	Power (mW)	Power (mW)	Power (mW)
vdd_core_power	1.1	181.66	185.76	187.66	188.39	190.63
vdd_mpu_power	DVFS (0.95 - 1.325)	85.51	223.17	317.80	386.93	512.13
1.8V I/O	1.8	35.90	38.65	40.01	40.69	42.81
3.3V I/O[1]	3.3	100.98	103.69	105.55	104.08	102.99
SoC Power without DDR3		404.05	551.27	651.02	720.08	848.56
vdds_ddr	1.5	150.91	151.77	153.58	154.22	156.28
vddsddrmem	1.5	117.21	133.29	139.00	139.67	146.72
DDR3 Total Power (Includes VTT)		268.12	285.06	292.58	293.89	303.01
Total Power (SoC + DDR3) with VTT		672.17	836.33	943.06	1013.97	1151.57

(1) MPEG4 + AAC decode at this OPP experienced significant numbers of dropped frames

4.6 Low Power Modes

Standby and Suspend modes both put the processor into a low-power state with DDR is in self-refresh.

Table 14. Optimized Low Power Mode Power Consumption

Power Rail	Standby Power (mW)	Suspend Power (mW)
vdd_core_power	13.9	1.07
vdd_mpu_power	0.26	0.25
1.8V I/O	5.61	3.74
3.3V I/O	1.36	1.33
SoC Power without DDR3	21.13	6.39
vdds_ddr	0.4	0.3
vddsddrmem	27.49	27.52
DDR3 + I/O + VTT	27.53	27.55
Total Power (SoC + DDR3) with VTT	48.66	33.94

Using DDR3L data sourced from the BeagleBone Black, the power consumption of a VTT-less DDR3L system can be approximated. VDCDC1 includes both the DDR3L memory power as well as the EMIF interface on the AM335x. SoC power from AM335x GP EVM data summarized in Table 14.

Power Rail	Voltage (V)	Standby Power (mW)	Suspend Power (mW)
DCDC1 (DDR+I/O)	1.35	11.40	11.42
SoC Power		21.13	6.39
Total Power (SoC + DDR3L) without VTT		32.53	17.81

5 References

- *AM335x Power Estimation Tool* wiki page
- Debugging AM335x Suspend-Resume Issues wiki page
- *Optimizing AM335x IO Power in DeepSleep0* wiki page
- *Linux Core Power Management User's Guide* wiki page
- PMIC driver information wiki (for example, *Linux Driver for TPS65910*)
- *AM335x PRCM Tools* wiki page
- *AM335x Schematic Checklist* wiki page
- *AM335x Hardware Design Guide* wiki page
- *AM335x Sitara™ Processors Data Manual*
- *AM335x and AMIC110 Sitara™ Processors Technical Reference Manual*
- *AM335x Sitara™ Processors Silicon Revisions 2.1, 2.0, 1.0 Silicon Errata*
- *Thermal Design Guide for DSP and ARM Application Processors*

Additional Information

A.1 Example Source Code

This section documents the longer code snippets discussed in the application note. First are the diffs for the device tree sources used to create the low power optimized use cases. Then the an example shell script to read INA226 devices over the I2C bus is included. These source files are provided as is, and have not been extensively tested beyond the benchmarks documented above, or a cursory functionality test. Further testing is highly recommended before using any of this code.

Diffs were generated by comparing the modified dts files to the standard am335x-evm.dts.

am335x-evm-powersave diff

```
--- am335x-evm.dts  2016-10-03 16:11:08.000000000 -0500
+++ am335x-evm-powersave.dts    2017-01-11 16:51:13.576594450 -0600
@@ -95,6 +95,7 @@
    };

    backlight {
+       status = "disabled";
        compatible = "pwm-backlight";
        pwms = <&ecap0 0 50000 0>;
        brightness-levels = <0 51 53 56 62 75 101 152 255>;
@@ -103,7 +104,7 @@

    panel {
        compatible = "ti,tilcdc,panel";
-       status = "okay";
+       status = "disabled";
        pinctrl-names = "default";
        pinctrl-0 = <&lcd_pins_s0>;
        panel-info {
@@ -795,3 +796,73 @@
 &sgx {
    status = "okay";
 };
+
+/*** Begin modifications to turn peripherals OFF ***/
+
+&usb {
+   status = "disabled";
+};
+
+&usb_ctrl_mod {
+   status = "disabled";
+};
+
+&usb0_phy {
+   status = "disabled";
+};
+
+&usb1_phy {
+   status = "disabled";
+};
+
+&usb0 {
```

```
+   status = "disabled";
+};
+
+&usb1 {
+   status = "disabled";
+};
+
+
+&lcdc {
+   status = "disabled";
+};
+
+&elm {
+   status = "disabled";
+};
+
+
+&epwmss0 {
+   status = "disabled";
+};
+
+&gpmc {
+   status = "disabled";
+};
+
+
+&mcasp1 {
+   status = "disabled";
+};
+
+&sham {
+   status = "disabled";
+};
+
+&aes {
+   status = "disabled";
+};
+
+
+&sgx {
+   status = "disabled";
+};
+
+&pruss {
+   status = "disabled";
+};
+
+&tsadc {
+   status = "disabled";
+};
```

am335x-evm-powersave-multimedia diff

```
--- am335x-evm.dts  2016-10-03 16:11:08.000000000 -0500
+++ am335x-evm-powersave-multimedia.dts 2017-01-11 16:50:17.020593389 -0600
@@ -795,3 +795,53 @@
 &sgx {
    status = "okay";
 };
+
+/*** Begin modifications to turn peripherals OFF ***/
+
+&usb {
+   status = "disabled";
+};
+
+&usb_ctrl_mod {
+   status = "disabled";
+};
+
+&usb0_phy {
+   status = "disabled";
+};
+
+&usb1_phy {
+   status = "disabled";
+};
+
+&usb0 {
+   status = "disabled";
+};
+
+&usb1 {
+   status = "disabled";
+};
+
+&elm {
+   status = "disabled";
+};
+
+&gpmc {
+   status = "disabled";
+};
+
+&sham {
+   status = "disabled";
+};
+
+&aes {
+   status = "disabled";
+};
+
+&pruss {
+   status = "disabled";
+};
+
+&tsadc {
+   status = "disabled";
+};
+
+&sgx {
+   status = "disabled";
+};
```

DIY INA I2C Measurement

```
#!/bin/bash
#INA226 Registers
CONFIG_REG=0x0
SHUNTV_REG=0x1
BUSV_REG=0x2
#EVM Specific
declare -a INA_ADDRS=(0x41);
declare -a SUPPLIES=('VDD_MPU ');
declare -a RES=(0.001);
#Take shunt voltage measurements #Re-initialize INA226
i2cset -y 1 ${INA_ADDRS[0]} $CONFIG_REG 0x0080 w
i2cset -y 1 ${INA_ADDRS[0]} $CONFIG_REG 0xFF4F w
shuntv=$(i2cget -y 1 ${INA_ADDRS[0]} $SHUNTV_REG w)
let "temp = shuntv >> 8"
let "temp2 = shuntv << 8 | $temp"
let "shuntv = $temp2 & 0xffff"
let "neg_test = $shuntv & 0x8000"
if [ "$neg_test" -gt 0 ]; then #handle negative
 shuntv=0
fi
shuntv=$(echo "$shuntv*0.0025" | bc)
#Take bus voltage measurements #Re-initialize INA226
i2cset -y 1 ${INA_ADDRS[0]} $CONFIG_REG 0x0080 w
i2cset -y 1 ${INA_ADDRS[0]} $CONFIG_REG 0xFF4F w
busv=$(i2cget -y 1 ${INA_ADDRS[0]} $BUSV_REG w)
let "temp = $busv >> 8"
let "temp2 = $busv << 8 | $temp"
let "busv = $temp2 & 0xffff"
busv=$(echo "$busv*0.00125" | bc)
#Calculate power
current=$(echo "${shuntv}/${RES[0]}" | bc -l)
power=$(echo "${current}*${busv}" | bc -l)
#Output to console
printf "Supply\t\tRes (ohm)\tShunt (mV)\tBus (V)\t\tCurrent (mA)\tPower (mW)\n"
printf "%s\t%.3f\t\t%f\t%f\t%f\t%f\n" "${SUPPLIES[0]}" "${RES[0]}" "${shuntv}" "${busv}"
"${current}" "${power}"
```

Revision History

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

Changes from Original (February 2017) to A Revision	Page
• Update was made in Section 3.2.2.	19
• Added new Section 3.3.4.4.	23
• Update was made in Section 4.	23
• Update was made to Section 5.	33

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