TMS320DM644x DMSoC
ARM Subsystem

Reference Guide

Literature Number: SPRUE14C
July 2010

Contents

List of Figures

List of Tables

Read This First

About This Manual

This document describes the ARM subsystem in the TMS320DM644x Digital Media System-on-Chip (DMSoC).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320DM644x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

The current documentation that describes the DM644x DMSoC, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: www.ti.com/c6000.

SPRUE15 — *TMS320DM644x DMSoC DSP Subsystem Reference Guide.* Describes the digital signal processor (DSP) subsystem in the TMS320DM644x Digital Media System-on-Chip (DMSoC).

SPRUE19 — *TMS320DM644x DMSoC Peripherals Overview Reference Guide.* Provides an overview and briefly describes the peripherals available on the TMS320DM644x Digital Media System-on-Chip (DMSoC).

SPRAA84 — *TMS320C64x to TMS320C64x+ CPU Migration Guide.* Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) to the TMS320C64x+ DSP. The objective of this document is to indicate differences between the two cores. Functionality in the devices that is identical is not included.

SPRU732 — *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide.* Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

SPRU871 — *TMS320C64x+ DSP Megamodule Reference Guide.* Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

SPRAAA6 — *EDMA v3.0 (EDMA3) Migration Guide for TMS320DM644x DMSoC.* Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) enhanced direct memory access (EDMA2) to the TMS320DM644x Digital Media System-on-Chip (DMSoC) EDMA3. This document summarizes the key differences between the EDMA3 and the EDMA2 and provides guidance for migrating from EDMA2 to EDMA3.

Introduction

Topic **Page**

1.1 Block Diagram

The TMS320DM644x Digital Media System-on-Chip (DMSoC) contains two primary CPU cores: 1) an ARM RISC CPU for general purpose processing and systems control and 2) a powerful DSP to efficiently handle image, video, and audio processing tasks. The DMSoC consists of the following primary components and sub-systems:

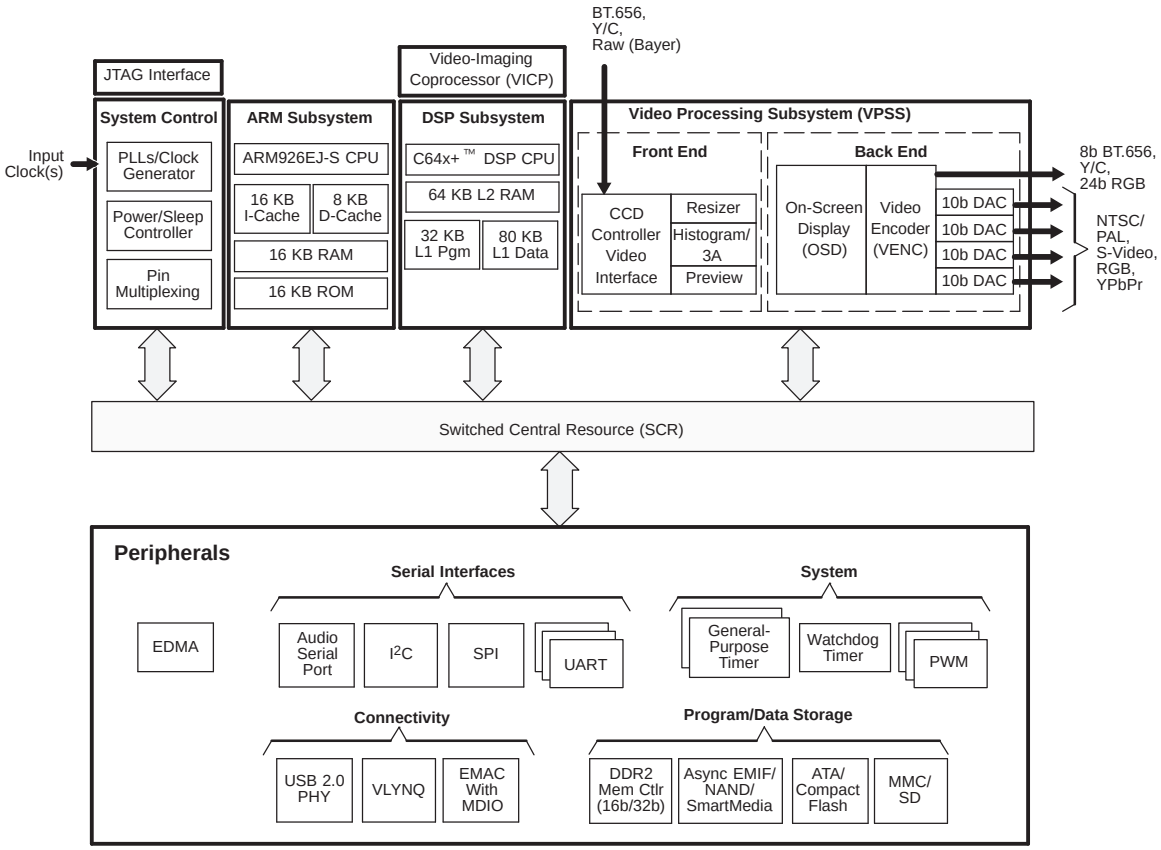
- ARM Subsystem (ARMSS), including the ARM926 RISC CPU core and associated memories
- DSP Subsystem (DSPSS), including the C64x+ DSP, the Video-Imaging Coprocessor (VICP), and associated memories
- Video Processing Subsystem (VPSS), including the Video Processing Front End (VPFE), Image Input and Image Processing Subsystem, and the Video Processing Back End (VPBE) Display Subsystem
- A set of I/O peripherals
- A powerful DMA Subsystem and DDR2 EMIF interface

An example block diagram (for the TMS320DM6446 DMSoC) is shown in Figure 1-1.

1.2 ARM Subsystem in TMS320DM644x DMSoC

The ARM926EJ 32-bit RISC processor in the ARMSS acts as the overall system controller. The ARM CPU performs general system control tasks, such as system initialization, configuration, power management, user interface, and user command implementation. Chapter 2 describes the ARMSS components and system control functions that the ARM core performs.

Figure 1-1. TMS320DM6446 DMSoC Block Diagram



ARM Subsystem Overview

Topic **Page**

2.1 Purpose of the ARM Subsystem

The ARM Subsystem contains the components required to provide the ARM926EJ-S (ARM) master control of the TMS320DM644x DMSoC system. In general, the ARM is responsible for configuration and control of the overall DM644x DMSoC system, including the DSP Subsystem, the VPSS Subsystem, and a majority of the peripherals and external memories.

In the DMSoC, the ARM is responsible for handling many system functions such as system-level initialization, configuration, user interface, user command execution, connectivity functions, interface and control of the DSP Subsystem, and overall system control. The ARM performs these functions because it has a larger program memory space and better context switching capability, and is thus more suitable for complex, multi-tasking, and general-purpose control tasks than the DSP.

2.2 Components of the ARM Subsystem

The ARM Subsystem (ARMSS) in the DM644x DMSoC consists of the following components:

- ARM926EJ-S RISC processor, including:
  - Co-Processor 15 (CP15)
  - MMU
  - 16 kB Instruction cache and 8 kB Data cache
  - Write Buffer
- ARM Internal Memories
  - 16 kB Internal RAM (32-bit wide access)
  - 8 kB Internal ROM (ARM boot loader for non-AEMIF boot options)
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB)
- System Control Peripherals
  - ARM Interrupt Controller
  - PLL Controller
  - Power and Sleep Controller
  - System Module

The ARM also manages/controls the following peripherals:

- DDR2 Port Controller
- Asynchronous EMIF (AEMIF) Controller, including the NAND flash interface
- Enhanced DMA (EDMA) System - Channel Controller (CC) and Transfer Controllers (TCs)
- UARTs
- Timers
- Pulse Width Modulator (PWM)
- Inter-IC Communication (I2C)
- Multimedia Card/Secure Digital (MMC/SD) Card Controller
- Audio Serial Port (ASP)
- Universal Serial Bus (USB) Controller
- ATA/Compact Flash (CF) Controller
- Serial Port Interface (SPI)
- Ethernet Media Access Controller (EMAC)
- Video Processing Front End (VPFE):
  - CCD Controller (CCDC)
  - Preview Engine
  - Resizer
  - H3A Engine (Hardware engine for computing Auto-focus, Auto white balance, and Auto exposure)
  - Histogram

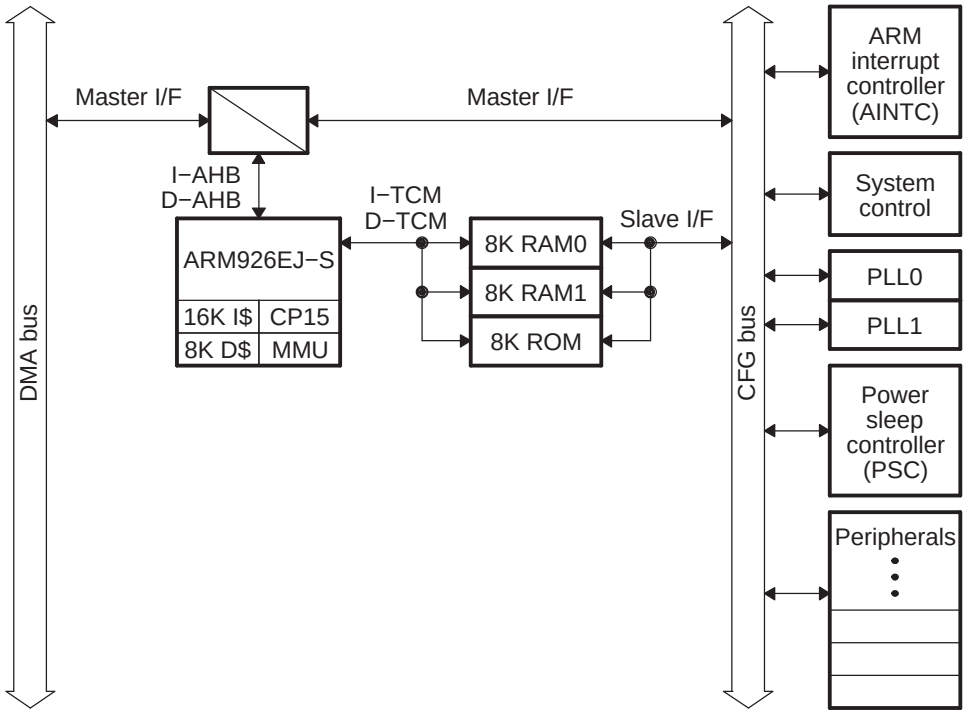
- Video Processing Back End (VPBE):
  - On-Screen Display (OSD)
  - Video Encoder Engine (VENC)

Figure 2-1 shows the functional block diagram of the DM644x DMSoC ARM Subsystem.

The DM644x DMSoC architecture uses two primary bus subsystems to transfer data within the system:

- The **DMA bus** (sometimes called data bus) is used for data transfer between subsystems and modules.
- The **CFG bus** (or configuration bus) is used to write to peripheral registers in various modules for configuration.

Figure 2-1. TMS320DM644x DMSoC ARM Subsystem Block Diagram



2.3 References

See the following DM644x DMSoC related documents for more information:

- For related documentation about the DM644x DMSoC other than the ARM core, see the *Related Documentation* section at the beginning of this document.
- For more detailed information about the ARM processor core, see ARM Ltd.'s web site (particularly, see the ARM926EJ-S Technical Reference Manual):
  - http://www.arm.com/documentation/ARMProcessor_Cores/index.html

ARM Core

Topic Page

3.1 Introduction

This chapter describes the ARM core and its associated memories. The ARM core consists of the following components:

- ARM926EJ-S - 32-bit RISC processor
- 16 kB Instruction cache
- 8 kB Data cache
- MMU
- CP15 to control MMU, cache, etc.
- Java accelerator
- ARM Internal Memory
  - 16 kB built-in RAM
  - 8 kB built-in ROM (boot ROM)
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB)
- Features:
  - The main write buffer has a 16-word data buffer and a 4-address buffer
  - Support for 32/16-bit instruction sets
  - Fixed little endian memory format
  - Enhanced DSP instructions

The ARM926EJ-S processor is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor targets multi-tasking applications where full memory management, high performance, low die size, and low power are all important.

The ARM926EJ-S processor supports the 32-bit ARM and the 16-bit THUMB instruction sets, enabling you to trade off between high performance and high code density. This includes features for efficient execution of Java byte codes and providing Java performance similar to Just in Time (JIT) Java interpreter without associated code overhead.

The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debugging. The ARM926EJ-S processor has a Harvard architecture and provides a complete high performance subsystem, including the following:

- An ARM926EJ-S integer core
- A Memory Management Unit (MMU)
- Separate instruction and data AMBA AHB bus interfaces
- Separate instruction and data TCM interfaces

The ARM926EJ-S processor implements ARM architecture version 5TEJ.

The ARM926EJ-S core includes new signal processing extensions to enhance 16-bit fixed-point performance using a single-cycle 32 × 16 multiply-accumulate (MAC) unit. The ARM Subsystem also has 16 kB of internal RAM and 8 kB of internal ROM, accessible via the I-TCM and D-TCM interfaces through an arbiter. The same arbiter provides a slave DMA interface to the rest of the DM644x DMSoC. Furthermore, the ARM has DMA and CFG bus master ports via the AHB interface.

3.2 Operating States/Modes

The ARM can operate in two states: ARM (32-bit) mode and Thumb (16-bit) mode. You can switch the ARM926EJ-S processor between ARM mode and Thumb mode using the BX instruction.

The ARM can operate in the following modes:

- User mode (USR): Non-privileged mode, usually for the execution of most application programs.
- Fast interrupt mode (FIQ): Fast interrupt processing
- Interrupt mode (IRQ): Normal interrupt processing
- Supervisor mode (SVC): Protected mode of execution for operating systems
- Abort mode (ABT): Mode of execution after a data abort or a pre-fetch abort
- System mode (SYS): Privileged mode of execution for operating systems
- Undefined mode (UND): Executing an undefined instruction causes the ARM to enter undefined mode.

You can only enter privileged modes (system or supervisor) from other privileged modes.

To enter supervisor mode from user mode, generate a software interrupt (SWI). An IRQ interrupt causes the processor to enter the IRQ mode. An FIQ interrupt causes the processor to enter the FIQ mode.

Different stacks must be set up for different modes. The stack pointer (SP) automatically changes to the SP of the mode that was entered.

3.3 Processor Status Registers

The processor status register (PSR) controls the enabling and disabling of interrupts and setting the mode of operation of the processor. The 8 least-significant bits PSR[7:0] are the control bits of the processor. PSR[27:8] are reserved bits and PSR[31:28] are status registers. The details of the control bits are:

- Bit 7 - I bit: Disable IRQ (I =1) or enable IRQ (I = 0)
- Bit 6 - F bit: Disable FIQ (F = 1) or enable FIQ (F = 0)
- Bit 5 - T bit: Controls whether the processor is in thumb mode (T = 1) or ARM mode (T = 0)
- Bits 4:0 Mode: Controls the mode of operation of the processor
  - PSR [4:0] = 10000 : User mode
  - PSR [4:0] = 10001 : FIQ mode
  - PSR [4:0] = 10010 : IRQ mode
  - PSR [4:0] = 10011 : Supervisor mode
  - PSR [4:0] = 10111 : Abort mode
  - PSR [4:0] = 11011 : Undefined mode
  - PSR [4:0] = 11111 : System mode

Status bits show the result of the most recent ALU operation. The details of status bits are:

- Bit 31 - N bit: Negative or less than
- Bit 30 - Z bit: Zero
- Bit 29 - C bit: Carry or borrow
- Bit 28 - V bit: Overflow or underflow

NOTE: See Chapter 2 of the Programmer's Model of the ARM926EJ-S TRM, downloadable from http://www.arm.com/arm/TRMs for more detailed information.

3.4 Exceptions and Exception Vectors

Exceptions arise when the normal flow of the program must be temporarily halted. The exceptions that occur in an ARM system are given below:

- Reset exception: processor reset
- FIQ interrupt: fast interrupt
- IRQ interrupt: normal interrupt
- Abort exception: abort indicates that the current memory access could not be completed. The abort could be a pre-fetch abort or a data abort.
- SWI interrupt: use software interrupt to enter supervisor mode.
- Undefined exception: occurs when the processor executes an undefined instruction

The exceptions in the order of highest priority to lowest priority are: reset, data abort, FIQ, IRQ, pre-fetch abort, undefined instruction, and SWI. SWI and undefined instruction have the same priority. Depending upon the status of VINTH signal or the register setting in CP15, the vector table can be located at address 0000 0000h (VINTH = 0) or at address FFFF 0000h (VINTH = 1).

NOTE: This is a feature of the standard ARM9 code. However, there is no memory in the DMSoC in this address region, so do not set this bit.

The default vector table is shown in Table 3-1

Table 3-1. Exception Vector Table for ARM

Vector Offset Address	Exception	Mode on entry	I Bit State on Entry	F Bit State on Entry
0h	Reset	Supervisor	Set	Set
4h	Undefined instruction	Undefined	Set	Unchanged
8h	Software interrupt	Supervisor	Set	Unchanged
Ch	Pre-fetch abort	Abort	Set	Unchanged
10h	Data abort	Abort	Set	Unchanged
14h	Reserved	-	-	-
18h	IRQ	IRQ	Set	Unchanged
1Ch	FIQ	FIQ	Set	Set

3.5 The 16-BIS/32-BIS Concept

The key idea behind 16-BIS is that of a super-reduced instruction set. Essentially, the ARM926EJ processor has two instruction sets:

- ARM mode or 32-BIS: the standard 32-bit instruction set
- Thumb mode or 16-BIS: a 16-bit instruction set

The 16-bit instruction length (16-BIS) allows the 16-BIS to approach twice the density of standard 32-BIS code while retaining most of the 32-BIS's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because 16-BIS code operates on the same 32-bit register set as 32-BIS code. 16-bit code can provide up to 65% of the code size of the 32-bit code and 160% of the performance of an equivalent 32-BIS processor connected to a 16-bit memory system.

3.5.1 16-BIS/32-BIS Advantages

16-bit instructions operate with the standard 32-bit register configuration, allowing excellent inter-operability between 32-BIS and 16-BIS states. Each 16-bit instruction has a corresponding 32-bit instruction with the same effect on the processor model. The major advantage of a 32-bit architecture over a 16-bit architecture is its ability to manipulate 32-bit integers with single instructions, and to address a large address space efficiently. When processing 32-bit data, a 16-bit architecture takes at least two instructions to perform the same task as a single 32-bit instruction. However, not all of the code in a program processes 32-bit data (for example, code that performs character string handling), and some instructions (like branches) do not process any data at all. If a 16-bit architecture only has 16-bit instructions, and a 32-bit architecture only has 32-bit instructions, then the 16-bit architecture has better code density overall, and has better than one half of the performance of the 32-bit architecture. Clearly, 32-bit performance comes at the cost of code density. The 16-bit instruction breaks this constraint by implementing a 16-bit instruction length on a 32-bit architecture, making the processing of 32-bit data efficient with compact instruction coding. This provides far better performance than a 16-bit architecture, with better code density than a 32-bit architecture. The 16-BIS also has a major advantage over other 32-bit architectures with 16-bit instructions. The advantage is the ability to switch back to full 32-bit code and execute at full speed. Thus, critical loops for applications such as fast interrupts and DSP algorithms can be coded using the full 32-BIS and linked with 16-BIS code. The overhead of switching from 16-bit code to 32-bit code is folded into sub-routine entry time. Various portions of a system can be optimized for speed or for code density by switching between 16-BIS and 32-BIS execution, as appropriate.

3.6 Co-Processor 15 (CP15)

The system control coprocessor (CP15) is used to configure and control instruction and data caches, Tightly-Coupled Memories (TCMs), Memory Management Units (MMUs), and many system functions. The CP15 registers are only accessible with MRC and MCR instructions by the ARM in a privileged mode like supervisor mode or system mode.

3.6.1 Addresses in an ARM926EJ-S System

Three different types of addresses exist in an ARM926EJ-S system. They are as follows:

Table 3-2. Different Address Types in ARM System

Domain	ARM9EJ-S	Caches and MMU	TCM and AMBA Bus
Address type	Virtual Address (VA)	Modified Virtual Address (MVA)	Physical Address (PA)

An example of the address manipulation that occurs when the ARM9EJ-S core requests an instruction is shown in Example 3-1

Example 3-1. Address Manipulation

The VA of the instruction is issued by the ARM9EJ-S core.

The VA is translated to the MVA. The Instruction Cache (Icache) and Memory Management Unit (MMU) detect the MVA.

If the protection check carried out by the MMU on the MVA does not abort and the MVA tag is in the Icache, the instruction data is returned to the ARM9EJ-S core.

If the protection check carried out by the MMU on the MVA does not abort, and the MVA tag is not in the cache, then the MMU translates the MVA to produce the PA.

NOTE: See Chapter 2 of the Programmers Model of the ARM926EJ-S TRM, downloadable from http://www.arm.com/arm/TRMs for more detailed information.

3.6.2 Memory Management Unit

The ARM926EJ-S MMU provides virtual memory features required by operating systems such as SymbianOS, WindowsCE, and Linux. A single set of two level page tables stored in main memory controls the address translation, permission checks, and memory region attributes for both data and instruction accesses. The MMU uses a single unified Translation Lookaside Buffer (TLB) to cache the information held in the page tables.

The MMU features are as follows:

- Standard ARM architecture v4 and v5 MMU mapping sizes, domains, and access protection scheme.
- Mapping sizes are 1 MB (sections), 64 kB (large pages), 4 kB (small pages) and 1 kB (tiny pages)
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions)
- Hardware page table walks
- Invalidate entire TLB, using CP15 register 8
- Invalidate TLB entry, selected by MVA, using CP15 register 8
- Lockdown of TLB entries, using CP15 register 10

NOTE: See Chapter 3 of the Memory Management Unit of the ARM926EJ-S TRM, downloadable from http://www.arm.com/arm/TRMs for more detailed information.

3.6.3 Caches and Write Buffer

The ARM926EJ-S processor includes:

- An Instruction cache (Icache)
- A Data cache (Dcache)
- A write buffer

The size of the data cache is 8 kB, instruction cache is 16 kB, and write buffer is 17 bytes.

The caches have the following features:

- Virtual index, virtual tag, addressed using the Modified Virtual Address (MVA)
- Four-way set associative, with a cache line length of eight words per line (32 bytes per line), and two dirty bits in the Dcache
- Dcache supports write-through and write-back (or copy back) cache operation, selected by memory region using the C and B bits in the MMU translation tables
- Perform critical-word first cache refilling
- Cache lockdown registers enable control over which cache ways are used for allocation on a line fill, providing a mechanism for both lockdown and controlling cache pollution.
- Dcache stores the Physical Address TAG (PA TAG) corresponding to each Dcache entry in the TAGRAM for use during the cache line write-backs, in addition to the Virtual Address TAG stored in the TAG RAM. This means that the MMU is not involved in Dcache write-back operations, removing the possibility of TLB misses related to the write-back address.
- Cache maintenance operations to provide efficient invalidation of the following:
  - The entire Dcache or Icache
  - Regions of the Dcache or Icache
  - The entire Dcache
  - Regions of virtual memory
- They also provide operations for efficient cleaning and invalidation of the following:
  - The entire Dcache
  - Regions of the Dcache
  - Regions of virtual memory

The write buffer is used for all writes to a non-cachable bufferable region, write-through region, and write misses to a write-back region. A separate buffer is incorporated in the Dcache for holding write-back for cache line evictions or cleaning of dirty cache lines.

The main write buffer has a 16-word data buffer and a four-address buffer.

The Dcache write-back has eight data word entries and a single address entry.

The MCR drain write buffer enables both write buffers to be drained under software control.

The MCR wait for interrupt causes both write buffers to be drained and the ARM926EJ-S processor to be put into a low power state until an interrupt occurs.

NOTE: See Chapter 4 of the Caches and Write Buffer of the ARM926EJ-S TRM, downloadable from http://www.arm.com/arm/TRMs for more detailed information.

3.7 Tightly Coupled Memory

The ARM926EJ has a tightly coupled memory interface enabling separate instruction and data TCM to be interfaced to the ARM. TCMs are meant for storing real-time and performance critical code.

The DM644x DMSoC supports both instruction TCM (I-TCM) and data TCM (D-TCM). The instruction TCM is located at 0000:0000h to 0000:5FFFh. The data TCM is located at 0000:8000h to 0000:DFFFh, as shown in Table 3-3.

Table 3-3. ITCM/DTCM Memory Map

I-TCM Address	D-TCM Address	Size (Bytes)	Description
0000:0000h - 0000:1FFFh	0000:8000h - 0000:9FFFh	8K	IRAM0
0000:2000h - 0000:3FFFh	0000:A000h - 0000:BFFFh	8K	IRAM1
0000:4000h - 0000:5FFFh	0000:C000h - 0000:DFFFh	8K	ROM
0000:6000h - 0000:7FFFh	0000:E000h - 0000:FFFFh	8K	Reserved

The status of the TCM memory regions can be read from the TCM status register, which is CP15 register 0. The instruction for reading the TCM status is given below:

```
MRC p15, #0, Rd, c0, c0, #2 ; read TCM status register
```

where Rd is any register where the status data is read into the register.

The format of the data in the TCM status register is:

31	17	16
Reserved		DTCM

15	1	0
Reserved		ITCM

Table 3-4. TCM Status Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	DTCM		Data TCM.
		0	Data TCM is not present.
		1	Data TCM is present.
15-1	Reserved	0	Reserved
0	ITCM		Instruction TCM.
		0	Instruction TCM is not present.
		1	Instruction TCM is present.

The format of the data in the TCM region setup register is:

31															16
ADDRESS															

15			12	11			6	5			2	1	0
ADDRESS				Reserved (000000)				SIZE				0	ENB

Table 3-5. TCM Region Setup Register Field Descriptions

Bit	Field	Value	Description
31-12	ADDRESS	00000h-FFFFFh	Base Address. The value programmed in this field is left-shifted by 12 to represent the physical base address of the memory block (ITCM or DTCM).
11-6	Reserved	0	Reserved
5-2	SIZE	0h-Fh	Memory block size. See Table 3-6.
1	0	0	This bits is always 0.
0	ENB		TCM enable.
		0	TCM is disabled.
		1	TCM is enabled.

Table 3-6. ITCM/DTCM Size Encoding

Binary Code	Size
0000	0 kB / absent
0001 and 0010	Reserved
0011	4 kB
0100	8 kB
0101	16 kB
0110	32 kB
0111	64 kB
1000	128 kB
1001	256 kB
1010	512 kB
1011	1 MB
11xx	Reserved

The instructions for reading and writing to the ITCM and DTCM are shown below:

```
MRC p15, #0, Rd, c9, c1, #0 ; read DTCM region register
MCR p15, #0, Rd, c9, c1, #0 ; write DTCM region register
MRC p15, #0, Rd, c9, c1, #1 ; read ITCM region register
MCR p15, #0, Rd, c9, c1, #1 ; write ITCM region register
```

Where Rd is any register where the data is read or written into the register.

On DM644x devices, the base address of the ITCM is 0000 0000h and the size is 16 kB. Hence, the address field of the ITCM register c9 should be programmed with 00000h. The memory block size field of the ITCM register c9 is fixed to the value of 5h. The memory block size field of the ITCM register c9 is read only and write has no effect.

On DM644x devices, the base address of the DTCM is 0000 8000h and the size is 32 kB. The DM644x DTCM includes 16 kB of RAM and 16 kB of ROM. The address field of the DTCM register c9 should be programmed with 00008h. The memory block size field of the DTCM register c9 is fixed to the value of 6h. The memory block size field of the DTCM register c9 is read only and write has no effect.

Example 3-2. TMS320DM644x ITCM Register c9 Programming

```
; Read ITCM
MRC p15, #00, R3, c9, c1, #1
NOP
NOP

; Enable ITCM
MOV R0, #0x1;
MCR p15, #00, R0, c9, c1, #1
NOP
NOP

; Read Back the ITCM value to check the ITCM Enable function
MRC p15, #00, R4, c9, c1, #1
NOP
NOP
```

Example 3-3. TMS320DM644x DTCM Register c9 Programming

```
DTCM_BASE_ADDR .word 0x8000
DTCM_MASK .word 0x0FFF

; Read DTCM
MRC p15, #00, R3, c9, c1, #0
NOP
NOP

; Create DTCM enable mask
LDR R0,DTCM_BASE_ADDR
LDR R1,DTCM_MASK
AND R1, R1, R3
NOP
ORR R0, R0, #0x1;
ORR R0, R0, R1

; Enable DTCM
MCR p15, #00, R0, c9, c1, #0
NOP
NOP

; Read Back the DTCM value to check the DTCM Enable function
MRC p15, #00, R5, c9, c1, #0
NOP
NOP
```

NOTE: See Chapter 5 of the Tightly-coupled Memory Interface of the ARM926EJ-S TRM, downloadable from http://www.arm.com/arm/TRMs for more detailed information.

System Memory

Topic Page

4.1 Memory Map

The TMS320DM644x DMSoC has multiple on-chip memories associated with its two processors and various subsystems. To help simplify software development, a unified memory map is used where possible to maintain a consistent view of device resources across all bus masters.

For detailed memory-map information, refer to the device-specific data manual.

4.1.1 ARM Internal Memories

The ARM has access to the following ARM internal memories:

- 16 kB ARM Internal RAM on TCM interface, logically separated into two 8 kB pages to allow simultaneous access on any given cycle, if there are separate accesses for code (I-TCM bus) and data (D-TCM) to the different memory regions.
- 8 kB ARM Internal ROM

4.1.2 External Memories

The ARM has access to the following external memories:

- DDR2 Synchronous DRAM
- Asynchronous EMIF / NOR / NAND Flash
- ATA / Compact Flash (CF)

These memory interfaces are described in Section 1.1. Additionally, the ARM has access to the various common media storage card interfaces.

For documentation related to these interfaces, see the *Related Documentation* section at the beginning of this document.

4.1.3 DSP Memories

The ARM has access to the following DSP memories:

- L2 RAM (Level 2 RAM)
- L1P RAM (Level 1 Program RAM)
- L1D RAM (Level 1 Data RAM)

4.1.4 Peripherals

The ARM has access to the following peripherals:

- EDMA Controller
- 3 UARTs (one with RTS and CTS flow control)
- I2C (Inter-IC Communication)
- Two timers that are configurable as two 64-bit or four 32-bit timers and one 64-bit watchdog timer
- PWM (Pulse-Width Modulator)
- USB (Universal Serial Bus Controller)
- ATA/CF Controller
- SPI serial interface up to 40 MHz with 2 chip selects
- GPIO (General-Purpose Input/Output)
- VPSS (Video Processing Subsystem)
- Asynchronous EMIF (AEMIF) Controller

The ARM Subsystem also has access to the following internal peripherals:

- System Module
- PLL Controllers
- Power and Sleep Controller
- ARM Interrupt Controller

4.2 Memory Interfaces Overview

This section describes the different memory interfaces of DM644x DMSoC. The DM644x DMSoC supports several memory and external device interfaces, including the following:

- DDR2 Synchronous DRAM
- Asynchronous EMIF / NOR / NAND Flash
- ATA / Compact Flash

4.2.1 DDR2 EMIF

The DDR2 EMIF port is a dedicated interface to DDR2 SDRAM. It supports JESD79D-2A standard compliant DDR2 SDRAM devices and can support either 16-bit or 32-bit interfaces.

DDR2 SDRAM plays a key role in a DM644x DMSoC-based system. Such a system is expected to require a significant amount of high-speed external memory for the following:

- Buffering input image data from sensors or video sources
- Intermediate buffering for processing/resizing of image data in the video processing front end (VPFE)
- Video processing back end (VPBE) display buffers
- Intermediate buffering for large raw Bayer data image files while performing still camera processing functions
- Buffering for intermediate data while performing video encode and decode functions
- Storage of executable firmware for both the ARM and DSP

4.2.2 External Memory Interface

The DM644x DMSoC external memory interface (EMIF) provides an 8-bit or 16-bit data bus, an address bus width of up to 24-bits, and 4 dedicated chip selects, along with memory control signals. These signals are statically multiplexed between two primary memory interface modules. The primary memory interface modules are:

- AEMIF module - providing asynchronous EMIF (AEMIF) and NAND interfaces
- ATA / CF module – providing ATA/IDE drive support and Compact Flash True-IDE Mode support

The upper EMIF address lines are multiplexed with the VLYNQ interface signals to allow use of that interface concurrently with interfaces that require a small number of address lines. Additionally, most of the EMIF address lines are configurable as GPIO signals if they are not required, as would be the case if only NAND or ATA/CF interfaces were used.

4.2.2.1 Asynchronous EMIF (AEMIF)

The Asynchronous EMIF (AEMIF) interface provides both the AEMIF and NAND interfaces. Four chip selects are provided. Each is individually configurable to provide either AEMIF or NAND support.

- The AEMIF Mode supports asynchronous devices (RAM, ROM, and NOR Flash)
- 128MB asynchronous address range over 4 chip selects (32MB each)
- Supports 8-bit or 16-bit data bus widths
- Programmable asynchronous cycle timings
- Supports extended waits
- Supports Select Strobe mode
- Supports TI DSP HPI interface
- Supports booting DM644x DMSoC ARM processor from CS0 (SRAM / NOR Flash)

4.2.2.2 NAND (NAND, SmartMedia, xD)

The asynchronous EMIF (AEMIF) interface provides both the AEMIF and NAND interfaces. Four chip selects are provided and each is individually configurable to provide either AEMIF or NAND support.

- The NAND Mode supports NAND Flash on up to 4 asynchronous chip selects
- Supports 8-bit and 16-bit data bus widths
- Programmable cycle timings
- Performs ECC calculation
- NAND Mode also supports SmartMedia/SSFDC (Solid State Floppy Disk Controller) and xD memory cards
- ARM ROM supports booting of the DM644x DMSoC ARM processor from NAND-Flash located at CS0

4.2.2.3 ATA/CF Controller

The ATA/CF controller provides the following capabilities:

- Supports PIO, multi-word DMA, and Ultra ATA 33/66/100/133
- Supports up to mode 4 timings on PIO mode
- Supports up to mode 2 timings on multi-word DMA
- Supports up to mode 6 timings on Ultra ATA
- Full scatter gather DMA capability
- Configurable as primary or secondary controller
- Programmable timing features enable timing parameters to be reprogrammed to support any ATA timing mode at any clock frequency
- Supports TrueIDE mode for Compact Flash

Additionally, the Host IDE Controller supports multi-word DMA and Ultra DMA data transfers between external IDE/ATAPI devices and a system memory bus interface. The timing and control registers in this core are compatible to the Intel register set in the PIIX family.

This core has a full scatter gather DMA capability, which is compatible with the Intel scatter gather DMA function on the PIIX chipset.

Device Clocking

Topic Page

5.1 Overview

The TMS320DM644x device requires two primary reference clocks. The primary reference clocks can be either crystal input or driven by external oscillators. A 27 MHz crystal is recommended for the system PLLs, which generate the clocks for the ARM, DSP, coprocessors, peripherals, DMA, and imaging peripherals. The recommended 27 MHz input enables you to use the video DACs to drive NTSC/PAL television signals at the proper frequencies. A 24 MHz crystal is also required if you are going to use the USB peripheral.

For detailed specifications on clock frequency and voltage requirements, see the device-specific data manual.

There are two clocking modes:

- PLL Bypass - power saving (boot configuration)
- PLL Active - PLL multiplies input clock up to the desired operating frequency

The clock of the major chip subsystems operate at fixed ratios of the primary system/DSP clock frequency within each mode, as shown in Table 5-1. Figure 5-1 shows the DM644x DMSoC clocking architecture.

Table 5-1. System Clock Modes and Fixed Ratios for Core Clock Domains

Subsystem	Core Clock Domain	Fixed Ratio vs. DSP Frequency
DSPSS	SYSCLK1	1:1
ARMSS VICP	SYSCLK2	1:2
EDMA VPSS	SYSCLK3	1:3
Peripherals	SYSCLK5	1:6

5.2 Clock Domains

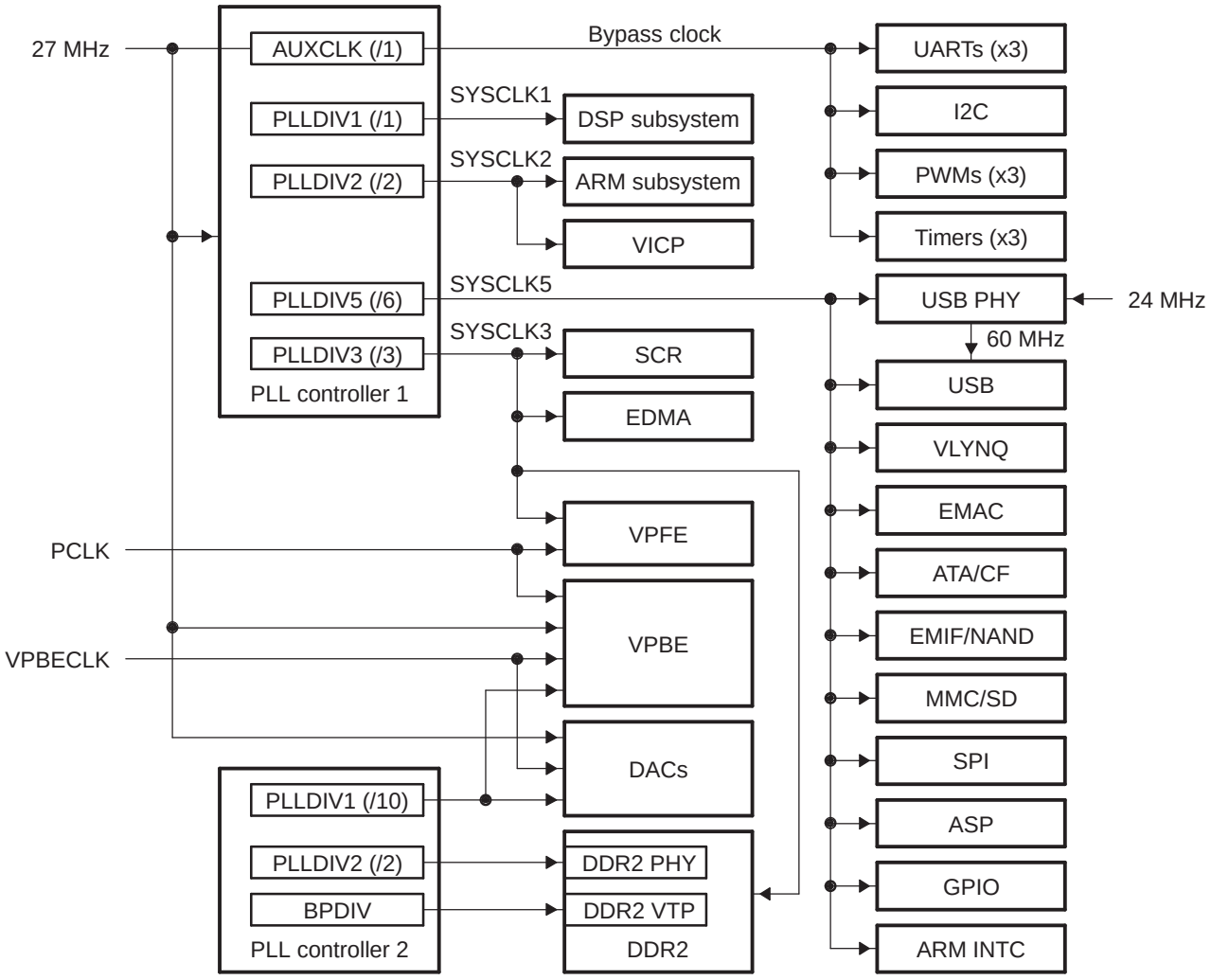
5.2.1 Core Domains

The core domains refer to the clock domains for all of the internal processing elements of the DM644x DMSoC, such as the DSP/VICP/EDMA, etc. All of the core clock domains are synchronous to each other, come from a single PLL (PLL1), have aligned clock edges, and have fixed divide by ratios, as shown in Figure 5-1 and Table 5-1.

The entire ARM subsystem is in the SYSCLK2 domain and runs at 1/2 the DSP frequency. The DSP subsystem is in the SYSCLK1 domain and receives the output of the PLLDIV1 block that is fixed at divide-by-1. The DSP has internal clock dividers that it uses to create DSP ÷ 2 and DSP ÷ 3 clock frequencies.

The video-imaging coprocessor (VICP) block is in the SYSCLK2 domain and runs at 1/2 the DSP frequency.

Figure 5-1. Overall Clocking Diagram



5.2.2 Frequency Flexibility

The core frequency domains are flexible, to a degree, within the following limitations:

- The PLLs can be driven by any input ranging from 20 to 30 MHz. However, a 27-MHz input is required if the Video Processing Back End (VPBE) Subsystem is needed to drive television displays with the integrated video DACs.
- The PLL1 multiplier setting can be changed within a range, as described below for a 27-MHz input. The PLL1 VCO frequency, as well as the final output clock after the PLLDIV must satisfy the limitations stated in the device-specific data manual. These limitations will vary based on the core voltage of the device.

Table 5-2 shows the possible PLL multiplier settings, along with the available PLL divider modes. The PLL divider modes are defined by the value programmed in the RATIO field of the PLL post-divider control register (POSTDIV). For Div1, Div2, and Div3 modes, the RATIO field would be programmed to 0, 1, and 2, respectively. The default configurations are listed in Table 5-2.

NOTE: In the instances where the pre-divider PLL output frequency is higher than the default, the PLL power consumption increases, offsetting the decrease in power consumption in the rest of the chip. For example, using the 337.5-MHz Div2 setting over the 351-MHz Div1 setting is not recommended for this reason.

Table 5-2. Example PLL1 Frequencies

		PLL1 Output		
PLL1 Multiplier	PLL1 VCO Frequency (MHz)	Div1	Div2	Div3
15	405.0	405.0	202.5	135.0
16	432.0	432.0	216.0	144.0
17	459.0	459.0	229.5	153.0
18	486.0	486.0	243.0	162.0
19	513.0	513.0	256.5	171.0
20	540.0	540.0	270.0	180.0
21	567.0	567.0	283.5	189.0
22	594.0	594.0	297.0	198.0
23	621.0	621.0	310.5	207.0(1)
24	648.0	648.0	324.0	216.0(1)
25	675.0	675.0	337.5	225.0(1)
26	702.0	702.0	351.0	234.0(1)
27	729.0	729.0	364.5	243.0(1)
28	756.0	756.0	378.0	252.0(1)
29	783.0	783.0	391.5	261.0(1)
30	810.0	810.0	405.0	270.0(1)

(1) For core voltage = 1.3V.

5.2.3 DDR2/EMIF Clock

The DDR2 interface has a dedicated clock driven from PLL2. This is a separate clock system from the PLL1 clocks provided to other components of the system. This dedicated clock allows the reduction of the core clock rates to save power while maintaining the required minimum clock rate (125 MHz) for DDR2. PLL2 must be configured to output a 2× clock to the DDR2 PHY interface.

The DM644x DMSoC video DACs are capable of driving high quality progressive television displays, if driven by a 54-MHz input clock sourced by PLL2 (see the *TMS320DM644x DMSoC Video Processing Back End (VPBE) User's Guide* (SPRUE37) for more detailed information). This will limit the possible PLL2 settings to a multiple of 54 MHz so that the VPBE clock can be derived with a simple integer clock divider.

The following frequency ranges in the device-specific data manual must be adhered to when configuring PLL2:

- Input clock frequency range (MXI/CLKIN)
- PLL2 VCO frequency range based on the core voltage of the device.

Table 5-3, Table 5-4, and Table 5-5 show the possible PLL2/DDR2 clock rates and the settings that are also a multiple of 54 MHz.

Table 5-3. Example PLL2 Frequencies (Core Voltage = 1.3V)

PLL2 Multiplier	PLL2 VCO Frequency (MHz)	Divider	PHY [2× clock] (MHz)	DDR2 Clock (MHz)	54 MHz Multiple
28	756.0	3	252.0	126.0	Yes
19	513.0	2	256.5	128.3	No
29	783.0	3	261.0	130.5	No
20	540.0	2	270.0	135.0	Yes
31	837.0	3	279.0	139.5	No
21	567.0	2	283.5	141.8	No
32	864.0	3	288.0	144.0	Yes
22	594.0	2	297.0	148.5	Yes
23	621.0	2	310.5	155.3	No
24	648.0	2	324.0	162.0	Yes
25	675.0	2	337.5	168.8	no
26	702.0	2	351.0	175.5	No
27	729.0	2	364.5	182.3	No
28	756.0	2	378.0	189.0	Yes

Table 5-4. Example PLL2 Frequencies (Core Voltage = 1.2V)

PLL2 Multiplier	PLL2 VCO Frequency (MHz)	Divider	PHY [2× clock] (MHz)	DDR2 Clock (MHz)	54 MHz Multiple
28	756.0	3	252.0	126.0	Yes
19	513.0	2	256.5	128.3	No
29	783.0	3	261.0	130.5	No
20	540.0	2	270.0	135.0	Yes
31	837.0	3	279.0	139.5	No
21	567.0	2	283.5	141.8	No
32	864.0	3	288.0	144.0	Yes
22	594.0	2	297.0	148.5	Yes
23	621.0	2	310.5	155.3	No
24	648.0	2	324.0	162.0	Yes

Table 5-5. Example PLL2 Frequencies (Core Voltage = 1.05V)

PLL2 Multiplier	PLL2 VCO Frequency (MHz)	Divider	PHY [2× clock] (MHz)	DDR2 Clock (MHz)	54 MHz Multiple
28	756.0	3	252.0	126.0	Yes
19	513.0	2	256.5	128.0	No
29	783.0	3	261.0	130.5	No
20	540.0	2	270.0	135.0	Yes

5.2.4 I/O Domains

The I/O domains refer to the frequencies of the peripherals that communicate through device pins. In many cases, there are frequency requirements for a peripheral pin interface that are set by an outside standard and must be met. It is not necessarily possible to obtain these frequencies from the on-chip clock generation circuitry, so the frequencies must be obtained from external sources and are asynchronous to the core frequency domain by definition.

Peripherals can be divided into 4 groups, depending upon their clock requirements. They are shown in Table 5-6.

Table 5-6. Peripherals

Peripheral Group	Peripheral Group Definition	Peripherals Contained within the Group	Frequency of Peripheral	Source of Peripheral
Fixed-Frequency Peripherals	As the name suggests, fixed-frequency peripherals have a fixed-frequency requirement. They are fed the fixed 27 MHz directly from the oscillator input.	UART	-	-
		I2C	-	-
		Timer/WDT	-	-
		PWM	-	-
Synchronous Peripherals	Synchronous peripherals have their frequencies derived from the core domain peripheral system clock frequency, which is PLL1÷6. The peripheral system clock frequency changes accordingly, if the PLL1 frequency changes. Most synchronous peripherals have internal dividers so they can generate their required clock frequencies.	NAND/SM/Async EMIF	-	-
		-	-	-
		MMC/SD	-	-
		SPI	-	-
		GPIO	-	-
		ATA/CF	-	-
Asynchronous Peripherals	Asynchronous peripherals are peripherals that require an asynchronous interface due to unique clocking requirements.	Video Processing Front End (VPFE)	10-108 MHz	External
		Video Processing Back End (VPBE)	6.25-75 MHz	External or Crystal or PLL2
		USB	24 MHz	Crystal
		-	<40 MHz	External
		DDR2 Memory Controller	126-189 MHz	PLL2
Synchronous/ Asynchronous Peripherals	Synchronous/ Asynchronous peripherals can be run with either internally generated synchronous clocks, or externally generated asynchronous clocks, selectable by MMR bits in the peripheral.	ASP	128 kHz-24.576 MHz	Up to 25.5 MHz (PLL1÷6 divided down)
		VLYNQ	Up to 99 MHz	Up to 99 MHz (PLL1÷6 divided down)
		I2C	Up to 400 kHz	Up to 400 kHz (27 MHz divided down)

5.2.5 Video Processing Back End

The Video Processing Back End (VPBE) is a sub-module of the VPSS (Video Processing Subsystem).

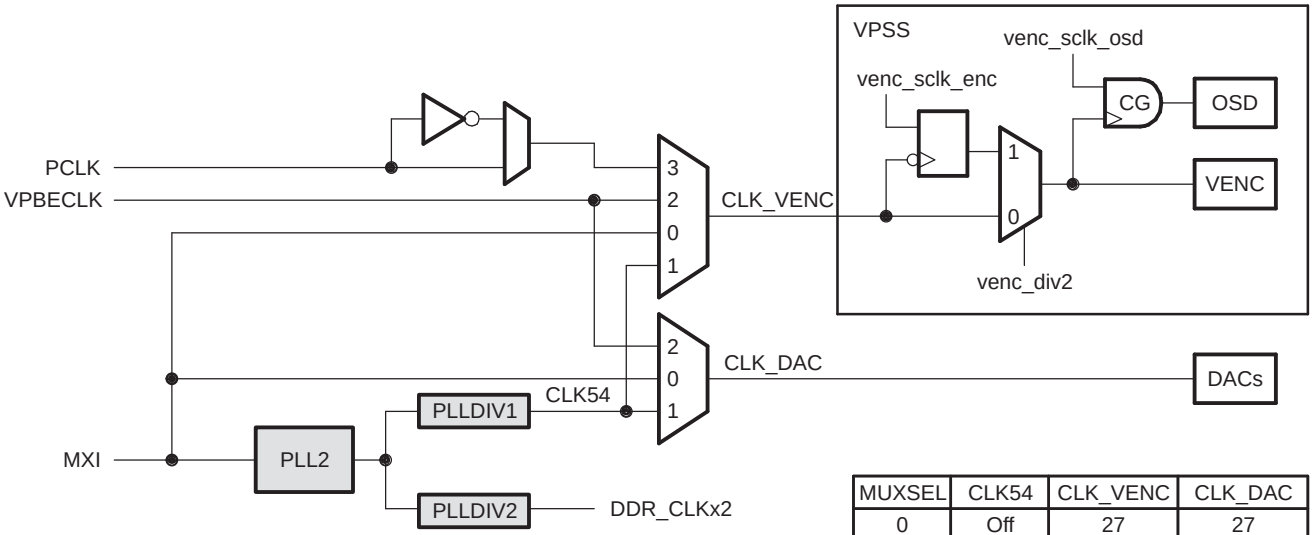
The VPBE must interface with a variety of LCDs, as well as the 4-channel DAC module. There are many different types of LCDs, which require many different specific frequencies. The range of frequencies that the pin interface needs to run is 6.25 MHz to 75 MHz.

There are two asynchronous clock domains in the VPBE - the external clock domain (6.25 MHz-75 MHz), and the internal (system) clock domain, which is at the PLL1 ÷ 3 clock rate.

The external clock domain can get its clock from 4 sources. The four sources are:

- The 27 MHz crystal input,
- The VPBECLK input pin,
- The VPFE pixel clock input (PCLK), or
- A divide down from PLL2.

Figure 5-2. VPBE/DAC Clocking



MUXSEL	CLK54	CLK_VENC	CLK_DAC
0	Off	27	27
1	54	54	54
2	Off	VPBECLK	VPBECLK
3	Off	PCLK	Off

The 4 DACs are hooked up to the VENC module that is inside the Video Processing Back End (VPBE). The data flow between the VPBE and DACs is synchronous. The various possible clocking modes are shown in Figure 5-2 and described in Table 5-7.

The DACs can have their clocks independently gated off when the DACs are not being used. This is handled in Chapter 8.

Table 5-7. Possible Clocking Modes

Clocking Mode	Description
MXI mode, MUXSEL = 0	Both the VENC and the DAC get their clock from the MXI 27 MHz crystal input.
PLL2 mode, MUXSEL = 1	The PLL2 (divided down) generates a 54 MHz clock. Both the DAC and the VENC receive the 54 MHz. The VENC can optionally divide it by 2 to create a 27 MHz clock. One limitation of this mode to be aware of is that the available DDR2 clock frequencies are restricted because the PLL multiplier must be an even number to be able to get 54 MHz from it.
VPBECLK mode, MUXSEL = 2	Both the DAC and the VENC receive the VPBECLK. The VENC has the option of dividing it by 2 for progressive scan support driving in 54 MHz on VPBECLK.
PCLK mode, MUXSEL = 3	The VENC receives the PCLK. The DAC receives no clock, and should be disabled. PCLK can be inverted for negative edge support, selectable by the MMR bit.

PLL Controller

Topic Page

6.1 PLL Module

The TMS320DM644x DMSoC has two PLL controllers that provide clocks to different parts of the system. PLL1 provides clocks (though various dividers) to most of the components of the DMSoC. PLL2 is dedicated to the DDR2 port and components fo the VPSS. The reference clock is the 27 MHz crystal input, as mentioned in Chapter 5.

The PLL controller provides the following:

- Glitch-Free Transitions (on changing clock settings)
- Domain Clocks Alignment
- Clock Gating
- PLL power down

The various clock outputs given by the controller are as follows:

- Domain Clocks: SYSCLK [1:n]
- Auxiliary Clock from reference clock source: AUXCLK
- Bypass Domain clock: SYSCLKBP

Various dividers that can be used are as follows:

- Post-PLL Divider: POSTDIV
- SYSCLK Divider: D1, …, Dn
- SYSCLKBP Divider: BPDIV

Various other controls supported are as follows:

- PLL Multiplier Control: PLLM
- Software programmable PLL Bypass: PLLEN

6.2 PLL1 Control

PLL1 supplies the primary DM644x DMSoC system clock. Software controls the PLL1 operation through the system PLL controller 1 (PLLC1) registers. Figure 6-1 shows the customization of PLL1 in the DM644x DMSoC.

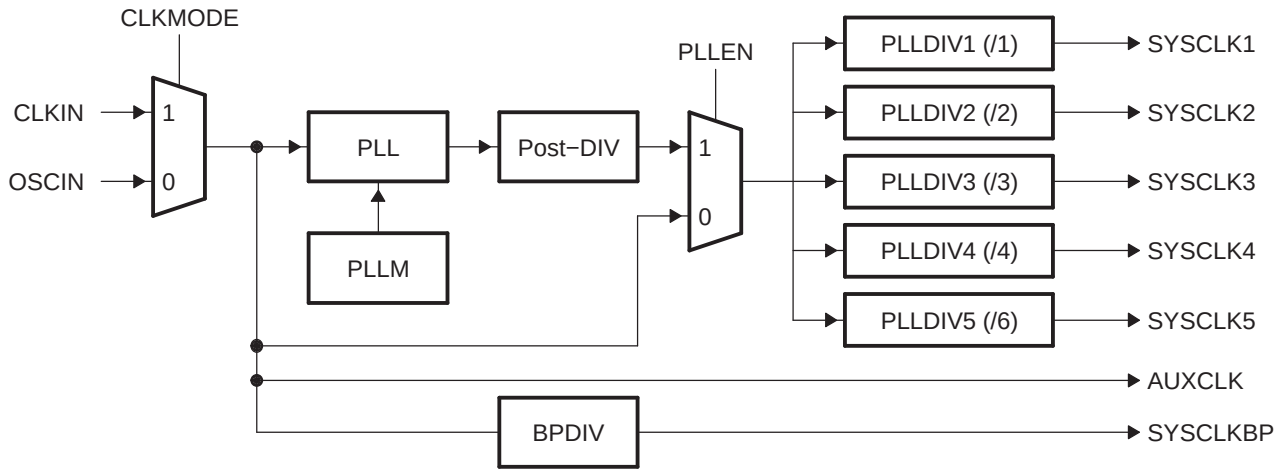
- The SYSCLK dividers are fixed (see Table 6-1).
- SYSCLKBP can be output to the CLK_OUT0 pin.
- AUXCLK is the clock provided to the fixed clock domains

The PLL1 multiplier is controlled by the PLLM bit in the PLL multiplier control register (PLLM) and is set to a default value of 0B10 000h at power-up, resulting in a PLL multiplier of 17×. The PLL1 output clock may be divided-down for slower device operation using the PLLC1 post-divider. This divider defaults to a ÷1 value, but may be modified by software (RATIO bit in POSTDIV) to achieve lower power device operation. These default settings yield a 459-MHz PLL output clock when using a 27-MHz clock source. The PLL1 multiplier may be modified by software (for example, set to 22× for 594-MHz operation).

At power-up, PLL1 is powered-down/disabled and must be powered-up by software through the PLLPWRDN bit in the PLL control register (PLLCTL). The system operates in bypass mode and the system clock is provided directly from the input reference clock (CLKIN or OSCIN). Once the PLL is powered-up and locked, software can switch the device to PLL mode operation. Set the PLLEN bit in PLLCTL to enable the PLL.

Registers used in PLLC1 are listed in Table 6-3.

Figure 6-1. PLL1 Structure in the TMS320DM644x DMSoC



6.2.1 Device Clock Generation

PLLC1 generates several clocks from the PLL1 output clock for use by the various processors and modules. These are summarized in Table 6-1. The output clock divider values SYSCLK1 to SYSCLK5 are fixed (locked by PLLC1). This maintains the clock ratios between the various device components no matter what reference clock (PLL or bypass) or PLL frequency is used.

Table 6-1. System PLLC1 Output Clocks

Output Clock	Used by	Divider	Notes
SYSCLK1	DSP Subsystem	/1	Fixed divider
SYSCLK2	VICP	/2	Fixed divider
SYSCLK3	SCR, EDMA, VPSS	/3	Fixed divider
SYSCLK5	Various peripherals	/6	Fixed divider
AUXCLK	Peripherals 27 MHz	n/a	Low jitter output clock
SYSCLKBP	CLKOUT0 Source		Programmable divider of the bypass clock

6.2.2 Steps for Changing PLL1/Core Domain Frequency

Refer to the appropriate subsection on how to program the PLL1/Core Domain clocks:

- If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), follow the full PLL initialization procedure in Section 6.2.2.1 to initialize the PLL.
- If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), follow the sequence in Section 6.2.2.2 to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and you only want to change the SYSCLK dividers, follow the sequence in Section 6.3.2.4.

Note that the PLL is powered down after the following device-level global resets:

- Power-on Reset ($\overline{\text{POR}}$)
- Warm Reset ($\overline{\text{RESET}}$)
- Max Reset

6.2.2.1 Initialization to PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), you must follow the procedure below to change PLL1 frequencies. The recommendation is to stop all peripheral operation before changing the PLL1 frequency, with the exception of the C64x+ DSP and DDR2. The C64x+ DSP must be operational to program the PLL controller. DDR2 operates off of the clock from PLLC2.

1. Select the clock mode by programming the CLKMODE bit in PLLCTL.
2. Before changing the PLL frequency, switch to PLL bypass mode:

   (a) Clear the PLLENSRC bit in PLLCTL to 0 to allow PLLCTL.PLLEN to take effect.
   (b) Clear the PLLEN bit in PLLCTL to 0 (select PLL bypass mode).
   (c) Wait for 4 MXI cycles to ensure PLLC switches to bypass mode properly.
3. Clear the PLLRST bit in PLLCTL to 0 (reset PLL)
4. Set the PLLDIS bit in PLLCTL to 1 (disable PLL output).
5. Clear the PLLPWRDN bit in PLLCTL to 0 to bring the PLL out of power-down mode.
6. Clear the PLLDIS bit in PLLCTL to 0 (enable the PLL) to allow PLL outputs to start toggling. Note that the PLLC is still at PLL bypass mode; therefore, the toggling PLL output does not get propagated to the rest of the device.
7. Wait for PLL stabilization time. See the device-specific data manual for PLL stabilization time.
8. Program the required multiplier value in PLLM.
9. Program the required divider value, if other than the default divider value, in POSTDIV.
10. Wait for PLL to reset properly. See the device-specific data manual for PLL reset time.
11. Set the PLLRST bit in PLLCTL to 1 to bring the PLL out of reset.

12. Wait for PLL to lock. See the device-specific data manual for PLL lock time.
13. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

6.2.2.2 Changing PLL Multiplier

If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0) and the PLL stabilization time is previously met (step 7 in Section 6.2.2.1), follow this procedure to change PLL1 multiplier. The recommendation is to stop all peripheral operation before changing the PLL multiplier, with the exception of the C64x+ DSP and DDR2. The C64x+ DSP must be operational to program the PLL controller. DDR2 operates off of the clock from PLLC2.

1. Before changing the PLL frequency, switch to PLL bypass mode:
   (a) Clear the PLLENSRC bit in PLLCTL to 0 to allow PLLCTL.PLLEN to take effect.
   (b) Clear the PLLEN bit in PLLCTL to 0 (select PLL bypass mode).
   (c) Wait for 4 MXI cycles to ensure PLLC switches to bypass mode properly.
2. Clear the PLLRST bit in PLLCTL to 0 (reset PLL).
3. Clear the PLLDIS bit in PLLCTL to 0 (enable the PLL) to allow PLL outputs to start toggling. Note that the PLLC is still at PLL bypass mode; therefore, the toggling PLL output does not get propagated to the rest of the device.
4. Program the required multiplier value in PLLM.
5. Wait for PLL to reset properly. See the device-specific data manual for PLL reset time.
6. Set the PLLRST bit in PLLCTL to 1 to bring the PLL out of reset.
7. Wait for PLL to lock. See the device-specific data manual for PLL lock time.
8. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

6.3 PLL2 Control

PLL2 provides the clock from which the DDR EMIF and optional VPBE clocks are derived. The DDR PLL controller (PLLC2) controls PLL2, which accepts the clock from the oscillator and also generates the various frequency clocks needed. Figure 6-2 shows the customization of PLL2 in the DM644x DMSoC.

- The post-divider should not be used.
- The SYSCLK dividers are programmable.
- AUXCLK is not used.

PLL2 supplies the DDR2 EMIF clock. Software controls PLL2 operation through the DDR2 PLL controller (PLLC2) registers. The PLLM bits in the PLL multiplier control register (PLLM) control the PLL2 multiplier. The PLL2 multiplier may be modified by software (for example, to tune the DDR2 interface for best performance).

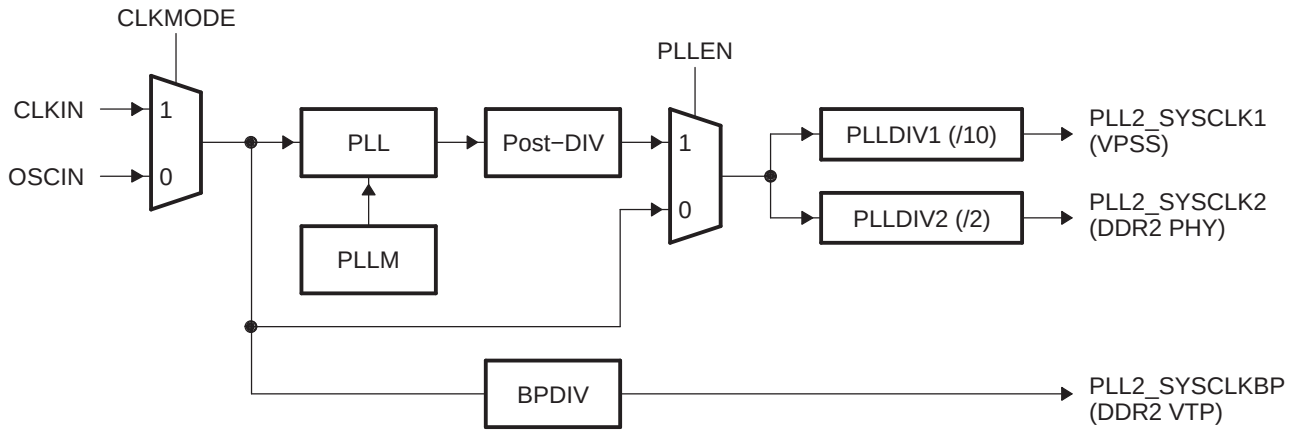
The PLL2 output clock must be divided-down to the DDR2 operating range.

At power-up, PLL2 is powered-down and must be powered-up by software through the PLLPWRDN bit in the PLL control register (PLLCTL). The PLLC2 is in bypass mode and the DDR clock is provided directly from the input reference clock. Once the PLL is powered-up and locked, software may switch the device to PLL mode operation by setting the PLLEN bit in PLLCTL.

Registers used in PLLC2 are listed in Table 6-4.

NOTE: PLLDIV1 defaults to /10 at reset can be modified after reset. PLLDIV2 defaults to /2 at reset can be modified after reset. PLLDIV3, PLLDIV4, and PLLDIV5 are not supported on PLL2.

Figure 6-2. PLL2 Structure in TMS320DM644x DMSoC



6.3.1 Device Clock Generation

PLLC2 generates two clocks from the PLL2 output clock for use by the DDR EMIF and VPSS modules. These are summarized in Table 6-2.

Table 6-2. DDR PLLC Output Clocks

Output Clock	Used by	Divider	Notes
SYSCLK1	VPSS	/10	Programmable divider
SYSCLK2	DDR Phy	/2	Programmable divider
SYSCLKBP	DDR VTP Controller	/2	Programmable divider

The SYSCLK2 output clock divider value defaults to /2, resulting in a 270 MHz DDR Phy clock (135 MHz DDR). It can be modified by software (RATIO bit in PLLDIV1) in combination with other PLL multipliers to achieve the desired DDR clock rate. The SYSCLK1 divider is programmable to allow a 54 MHz output to be generated from any even-multiple PLL output frequency.

6.3.2 Steps for Changing PLL2 Frequency

The PLLC2 is programmed similarly to the PLLC1. Refer to the appropriate subsection on how to program the PLL2 clocks:

- If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), follow the full PLL initialization procedure in Section 6.3.2.2 to initialize the PLL.
- If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), follow the sequence in Section 6.3.2.3 to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and you only want to change the SYSCLK dividers, follow the sequence in Section 6.3.2.4.

Note that the PLL is powered down after the following device-level global resets:

- Power-on Reset ($\overline{\text{POR}}$)
- Warm Reset ($\overline{\text{RESET}}$)
- Max Reset

In addition, note that the PLL2 frequency directly affects the DDR2 memory controller and the VPSS VPBE clock source (if PLLC2 SYSCLK2 is selected as the VPBE clock source). The DDR2 memory controller requires special sequences to be followed before and after you change the PLL2 frequency. You must follow the additional considerations for the DDR2 memory controller in Section 6.3.2.1 in order to not corrupt DDR2 operation.

6.3.2.1 DDR2 Considerations When Modifying PLL2 Frequency

Before changing PLL2 and/or PLLC2 frequency, you must take into account the DDR2 memory controller requirements. If the DDR2 memory controller is used in the system, follow the additional steps in this section to change PLL2 and/or PLLC2 frequency without corrupting DDR2 operation.

- If the DDR2 memory controller is in reset when you desire to change the PLL2 frequency, follow the steps in Section 6.3.2.1.1.
- If the DDR2 memory controller is already out of reset when you desire to change the PLL2 frequency, follow the steps in Section 6.3.2.1.2.

6.3.2.1.1 PLL2 Frequency Change Steps When DDR2 Memory Controller is In Reset

This section discusses the steps to change the PLL2 frequency when the DDR2 memory controller is in reset. Note that the DDR2 memory controller is in reset after these device-level global resets: power-on reset, warm reset, max reset.

1. Leave the DDR2 memory controller in reset.
2. Program the PLL2 clocks by following the steps in the appropriate section: Section 6.3.2.2, Section 6.3.2.3, or Section 6.3.2.4. (Discussion in Section 6.3.2 explains which is the appropriate subsection).
3. Initialize the DDR2 memory controller. The steps for DDR2 memory controller initialization are found in the *TMS320DM644x DMSoC DDR2 Memory Controller User's Guide* (SPRUE22).

6.3.2.1.2 PLL2 Frequency Change Steps When DDR2 Memory Controller is Out of Reset

This section discusses the steps to change the PLL2 frequency when the DDR2 memory controller is already out of reset.

1. Stop DDR2 memory controller accesses and purge any outstanding requests.
2. Put the DDR2 memory in self-refresh mode and stop the DDR2 memory controller clock. The DDR2 memory controller clock shut down sequence is in the *TMS320DM644x DMSoC DDR2 Memory Controller User's Guide* (SPRUE22).
3. Program the PLL2 clocks by following the steps in the appropriate section: Section 6.3.2.2, Section 6.3.2.3, or Section 6.3.2.4. (Discussion in Section 6.3.2 explains which is the appropriate subsection).
4. Re-enable the DDR2 memory controller clock. The DDR2 memory controller clock on sequence is in the *TMS320DM644x DMSoC DDR2 Memory Controller User's Guide* (SPRUE22).

6.3.2.2 Initialization to PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), you must follow the procedure below to change PLL2 frequencies.

1. Select the clock mode by programming the CLKMODE bit in PLLCTL.
2. Before changing the PLL frequency, switch to PLL bypass mode:

   (a) Clear the PLLENSRC bit in PLLCTL to 0 to allow PLLCTL.PLLEN to take effect.
   (b) Clear the PLLEN bit in PLLCTL to 0 (select PLL bypass mode).
   (c) Wait for 4 MXI cycles to ensure PLLC switches to bypass mode properly.
3. Clear the PLLRST bit in PLLCTL to 0 (reset PLL)
4. Set the PLLDIS bit in PLLCTL to 1 (disable PLL output).
5. Clear the PLLPWRDN bit in PLLCTL to 0 to bring the PLL out of power-down mode.
6. Clear the PLLDIS bit in PLLCTL to 0 (enable the PLL) to allow PLL outputs to start toggling. Note that the PLLC is still at PLL bypass mode; therefore, the toggling PLL output does not get propagated to the rest of the device.
7. Wait for PLL stabilization time. See the device-specific data manual for PLL stabilization time.
8. Program the required multiplier value in PLLM.

9. If necessary, program PLLDIV1 and PLLDIV2 registers to change the SYSCLK1 and SYSCLK2 divide values:
   (a) Program the RATIO field in PLLDIV1 and PLLDIV2 with the desired divide factors. For PLLC2, there is no specific frequency ratio requirements between SYSCLK1 and SYSCLK2.
   (b) Set the GOSET bit in PLLCMD to 1 to initiate a new divider transition.
   (c) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of phase alignment).
10. Wait for PLL to reset properly. See the device-specific data manual for PLL reset time.
11. Set the PLLRST bit in PLLCTL to 1 to bring the PLL out of reset.
12. Wait for PLL to lock. See the device-specific data manual for PLL lock time.
13. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

For information on initializing the DDR2 memory controller, see the *TMS320DM644x DMSoC DDR2 Memory Controller User's Guide* (SPRUE22).

6.3.2.3 Changing PLL Multiplier

If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0) and the PLL stabilization time is previously met (step 7 in Section 6.3.2.2), follow this procedure to change PLL2 multiplier.

1. Before changing the PLL frequency, switch to PLL bypass mode:
   (a) Clear the PLLENSRC bit in PLLCTL to 0 to allow PLLCTL.PLLEN to take effect.
   (b) Clear the PLLEN bit in PLLCTL to 0 (select PLL bypass mode).
   (c) Wait for 4 MXI cycles to ensure PLLC switches to bypass mode properly.
2. Clear the PLLRST bit in PLLCTL to 0 (reset PLL).
3. Clear the PLLDIS bit in PLLCTL to 0 (enable the PLL) to allow PLL outputs to start toggling. Note that the PLLC is still at PLL bypass mode; therefore, the toggling PLL output does not get propagated to the rest of the device.
4. Program the required multiplier value in PLLM.
5. If necessary, program PLLDIV1 and PLLDIV2 registers to change the SYSCLK1 and SYSCLK2 divide values:
   (a) Program the RATIO field in PLLDIV1 and PLLDIV2 with the desired divide factors. For PLLC2, there is no specific frequency ratio requirements between SYSCLK1 and SYSCLK2.
   (b) Set the GOSET bit in PLLCMD to 1 to initiate a new divider transition.
   (c) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of phase alignment).
6. Wait for PLL to reset properly. See the device-specific data manual for PLL reset time.
7. Set the PLLRST bit in PLLCTL to 1 to bring the PLL out of reset.
8. Wait for PLL to lock. See the device-specific data manual for PLL lock time.
9. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

6.3.2.4 Changing SYSCLK Dividers

This section discusses the software sequence to change the SYSCLK dividers. The SYSCLK divider change sequence is also referred to as GO operation, as it involves hitting the GO bit (GOSET bit in PLLCMD) to initiate the divider change.

1. Check for the GOSTAT bit in PLLSTAT to clear to 0 to indicate that no GO operation is currently in progress.
2. Program the RATIO field in PLLDIV1 and PLLDIV2 with the desired divide factors. For PLLC2, there is no specific frequency ratio requirements between SYSCLK1 and SYSCLK2.
3. Set the GOSET bit in PLLCMD to 1 to initiate a new divider transition.
4. Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).

6.4 PLL Controller Registers

Table 6-3 lists the memory-mapped registers for PLL controller 1 and Table 6-4 lists the memory-mapped registers for PLL controller 2.

Table 6-3. PLL Controller 1 Registers

Address	Acronym	Register Description	Section
1C4 0800h	PID	Peripheral ID Register	Section 6.4.1
1C4 08E4h	RSTYPE	Reset Type Status Register	Section 6.4.2
1C4 0900h	PLLCTL	PLL Control Register	Section 6.4.3
1C4 0910h	PLLM	PLL Multiplier Control Register	Section 6.4.4
1C4 0918h	PLLDIV1	PLL Controller Divider 1 Register	Section 6.4.5
1C4 091Ch	PLLDIV2	PLL Controller Divider 2 Register	Section 6.4.6
1C4 0920h	PLLDIV3	PLL Controller Divider 3 Register	Section 6.4.7
1C4 0928h	POSTDIV	PLL Post-Divider Control Register	Section 6.4.8
1C4 092Ch	BPDIV	Bypass Divider Register	Section 6.4.9
1C4 0938h	PLLCMD	PLL Controller Command Register	Section 6.4.10
1C4 093Ch	PLLSTAT	PLL Controller Status Register	Section 6.4.11
1C4 0940h	ALNCTL	PLL Controller Clock Align Control Register	Section 6.4.12
1C4 0944h	DCHANGE	PLLDIV Ratio Change Status Register	Section 6.4.13
1C4 0948h	CKEN	Clock Enable Control Register	Section 6.4.14
1C4 094Ch	CKSTAT	Clock Status Register	Section 6.4.15
1C4 0950h	SYSTAT	System Clock (SYSCLK) Status Register	Section 6.4.16
1C4 0960h	PLLDIV4	PLL Controller Divider 4 Register (Not Used)	—
1C4 0964h	PLLDIV5	PLL Controller Divider 5 Register	Section 6.4.17

Table 6-4. PLL Controller 2 Registers

Address	Acronym	Register Description	Section
1C4 0C00h	PID	Peripheral ID Register	Section 6.4.1
1C4 0D00h	PLLCTL	PLL Control Register	Section 6.4.3
1C4 0D10h	PLLM	PLL Multiplier Control Register	Section 6.4.4
1C4 0D18h	PLLDIV1	PLL Controller Divider 1 Register	Section 6.4.5
1C4 0D1Ch	PLLDIV2	PLL Controller Divider 2 Register	Section 6.4.6
1C4 0D28h	POSTDIV	PLL Post-Divider Control Register	Section 6.4.8
1C4 0D2Ch	BPDIV	Bypass Divider Register	Section 6.4.9
1C4 0D38h	PLLCMD	PLL Controller Command Register	Section 6.4.10
1C4 0D3Ch	PLLSTAT	PLL Controller Status Register	Section 6.4.11
1C4 0D40h	ALNCTL	PLL Controller Clock Align Control Register	Section 6.4.12
1C4 0D44h	DCHANGE	PLLDIV Ratio Change Status Register	Section 6.4.13
1C4 0D48h	CKEN	Clock Enable Control Register	Section 6.4.14
1C4 0D4Ch	CKSTAT	Clock Status Register	Section 6.4.15
1C4 0D50h	SYSTAT	System Clock (SYSCLK) Status Register	Section 6.4.16

6.4.1 Peripheral ID Register (PID)

The peripheral ID register (PID) is shown in Figure 6-3 and described in Table 6-5.

Figure 6-3. Peripheral ID Register (PID)

31	24	23	16
Reserved		TYPE	
R-0		R-1h	

15	8	7	0
CLASS		REV	
R-8h		R-2h	

LEGEND: R = Read only; -*n* = value after reset

Table 6-5. Peripheral ID Register (PID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	TYPE		Peripheral type
		1h	PLLC
15-8	CLASS		Peripheral class
		8h	Current class
7-0	REV		Peripheral revision
		2h	Current revision

6.4.2 Reset Type Status Register (RSTYPE)

The reset type status register (RSTYPE) is shown in Figure 6-4 and described in Table 6-6. Latches cause of the last reset. Although the reset value of all bits is 0 after coming out of reset, one bit is set to 1 to indicate the cause of the reset.

Figure 6-4. Reset Type Status Register (RSTYPE)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		SRST	MRST	XWRST	POR
R-0		R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-6. Reset Type Status Register (RSTYPE) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	SRST	0-1	System reset. If 1, the system reset was the last reset to occur that is of highest priority.
2	MRST	0-1	Maximum reset. If 1, the maximum reset was the reset to occur that is of highest priority.
1	XWRST	0-1	External warm reset. If 1, the external warm reset was the last reset to occur that is of highest priority.
0	POR	0-1	Power on reset. If 1, the power on reset was the last reset to occur that is of highest priority.

6.4.3 PLL Control Register (PLLCTL)

The PLL control register (PLLCTL) is shown in Figure 6-5 and described in Table 6-7.

Figure 6-5. PLL Control Register (PLLCTL)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		CLKMODE	Reserved		PLLENSRC	PLLDIS	PLLRST	Rsvd	PLLPWRDN	PLLEN
R-0		R/W-0	R-3h		R/W-1	R/W-1	R/W-0	R-0	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

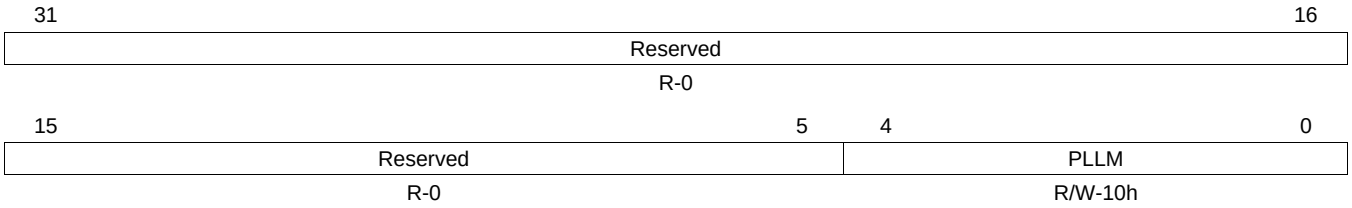
Table 6-7. PLL Control Register (PLLCTL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKMODE		Reference Clock Selection
		0	Internal oscillator
		1	CLKIN square wave
7-6	Reserved	1	Reserved
5	PLLENSRC	0	This bit must be cleared before PLLEN will have any effect.
4	PLLDIS		Asserts DISABLE to PLL if supported.
		0	PLL disable de-asserted
		1	PLL disable asserted
3	PLLRST		Asserts RESET to PLL if supported.
		0	PLL reset is asserted
		1	PLL reset is not asserted
2	Reserved	0	Reserved
1	PLLPWRDN		PLL power-down.
		0	PLL operation
		1	PLL power-down
0	PLLEN		PLL mode enable.
		0	Bypass mode
		1	PLL mode, not bypassed

6.4.4 PLL Multiplier Control Register (PLLM)

The PLL multiplier control register (PLLM) is shown in Figure 6-6 and described in Table 6-8.

Figure 6-6. PLL Multiplier Control Register (PLLM)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-8. PLL Multiplier Control Register (PLLM) Field Descriptions

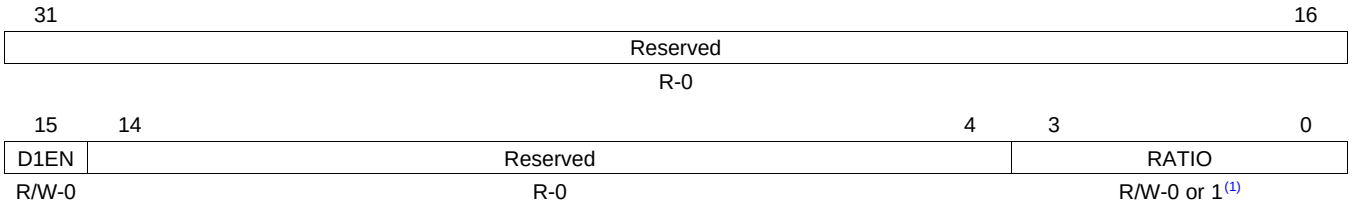
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PLLM	0-1Fh	PLL Multiplier Select. Multiplier Value = PLLM + 1. The valid range of multiplier values for a given MXI/CLKIN is defined by the minimum and maximum frequency limits on the PLL VCO frequency. See the device-specific data manual for PLL VCO frequency specification limits.

6.4.5 PLL Controller Divider 1 Register (PLLDIV1)

The PLL controller divider 1 register (PLLDIV1) is shown in Figure 6-7 and described in Table 6-9. Divider 1 controls the divider for SYSCLK1.

For PLL1, the RATIO bit is a fixed-field and cannot be changed.

Figure 6-7. PLL Controller Divider 1 Register (PLLDIV1)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

(1) For PLL1, RATIO defaults to 0 (PLL divide by 1); for PLL2, RATIO defaults to 1 (PLL2 divide by 2).

Table 6-9. PLL Controller Divider 1 Register (PLLDIV1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN		Divider Enable.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 0 (PLL divide by 1); for PLL2, RATIO defaults to 1 (PLL2 divide by 2).

6.4.6 PLL Controller Divider 2 Register (PLLDIV2)

The PLL controller divider 2 register (PLLDIV2) is shown in Figure 6-8 and described in Table 6-10. Divider 2 controls the divider for SYSCLK2.

For PLL1, the RATIO bit is a fixed-field and cannot be changed.

Figure 6-8. PLL Controller Divider 2 Register (PLLDIV2)

31	16
Reserved	
R-0	

15	14	4	3	0
D2EN	Reserved		RATIO	
R/W-1	R-0		R/W-1 or 9h[1]	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

(1) For PLL1, RATIO defaults to 1 (PLL divide by 2); for PLL2, RATIO defaults to 9h (PLL2 divide by 10).

Table 6-10. PLL Controller Divider 2 Register (PLLDIV2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN		Divider Enable.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 1 (PLL divide by 2); for PLL2, RATIO defaults to 9h (PLL2 divide by 10).

6.4.7 PLL Controller Divider 3 Register (PLLDIV3)

The PLL controller divider 3 register (PLLDIV3) is shown in Figure 6-9 and described in Table 6-11. Divider 3 controls the divider for SYSCLK3. PLLDIV3 is not used on PLL2.

For PLL1, the RATIO bit is a fixed-field and cannot be changed.

Figure 6-9. PLL Controller Divider 3 Register (PLLDIV3)

31	16
Reserved	
R-0	

15	14	4	3	0
D3EN	Reserved		RATIO	
R/W-1	R-0		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

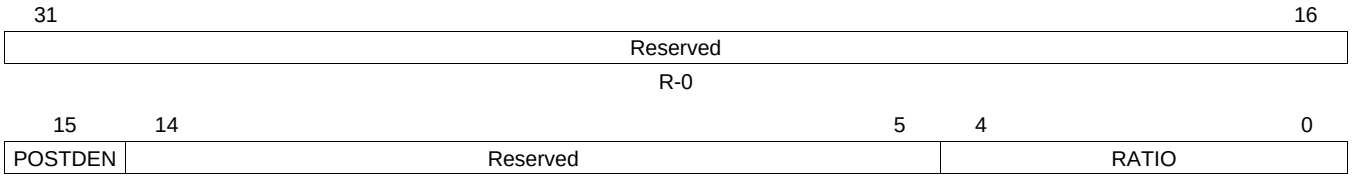
Table 6-11. PLL Controller Divider 3 Register (PLLDIV3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN		Divider Enable.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 2h (PLL divide by 3).

6.4.8 PLL Post-Divider Control Register (POSTDIV)

The PLL post-divider control register (POSTDIV) is shown in Figure 6-10 and described in Table 6-12. POSTDIV should not be used on PLL2.

Figure 6-10. PLL Post-Divider Control Register (POSTDIV)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

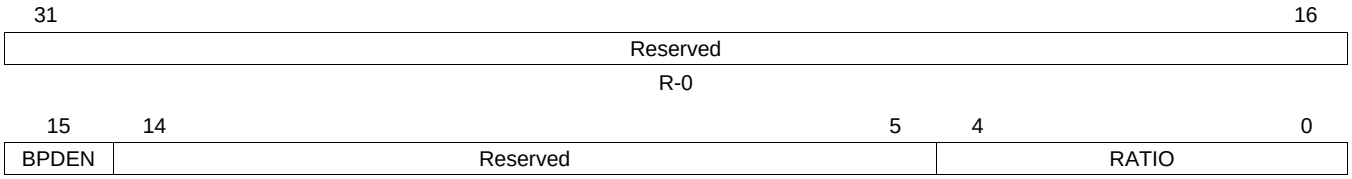
Table 6-12. PLL Post-Divider Control Register (POSTDIV) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	POSTDEN		Post_Divider enable.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. Valid Divider values: For PLL1, 1-3 (that is, RATIO = 0 to 2). See Table 5-2. For PLL2, keep set to the default value.

6.4.9 Bypass Divider Register (BPDIV)

The bypass divider register (BPDIV) is shown in Figure 6-11 and described in Table 6-13. Bypass divider controls the divider for SYSCLKBP.

Figure 6-11. Bypass Divider Register (BPDIV)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-13. Bypass Divider Register (BPDIV) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	BPDEN		Bypass Divider Enable.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1.

6.4.10 PLL Controller Command Register (PLLCMD)

The PLL controller command register (PLLCMD) is shown in Figure 6-12 and described in Table 6-14. Contains command bits for various operations. Writes of 1 initiate command; writes of 0 clear the bit, but have no effect.

Figure 6-12. PLL Controller Command Register (PLLCMD)

31	16
Reserved	
R-0	

15	1	0
Reserved		GOSET
R-0		R/W0C-0

LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear bit; -*n* = value after reset

Table 6-14. PLL Controller Command Register (PLLCMD) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSET		GO bit for SYSCLKx phase alignment.
		0	Clear bit (no effect)
		1	Phase alignment

6.4.11 PLL Controller Status Register (PLLSTAT)

The PLL controller status register (PLLSTAT) is shown in Figure 6-13 and described in Table 6-15.

Figure 6-13. PLL Controller Status Register (PLLSTAT)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		STABLE	Reserved	GOSTAT
R-0		R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-15. PLL Controller Status Register (PLLSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	STABLE		OSC counter done, oscillator assumed to be stable. By the time the device comes out of reset, this bit should become 1.
		0	No
		1	Yes
1	Reserved	0	Reserved
0	GOSTAT		Status of GO operation. If 1, indicates GO operation is in progress.
		0	GO operation is not in progress.
		1	GO operation is in progress.

6.4.12 PLL Controller Clock Align Control Register (ALNCTL)

The PLL controller clock align control register (ALNCTL) is shown in Figure 6-14 and described in Table 6-16. Indicates which SYSCLKs need to be aligned for proper device operation.

Figure 6-14. PLL Controller Clock Align Control Register (ALNCTL)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		ALN5	ALN4	ALN3	ALN2	ALN1
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-16. PLL Controller Clock Align Control Register (ALNCTL) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ALN5		SYSCLK5 needs to be aligned to others selected in this register.
		0	No
		1	Yes
3	ALN4		SYSCLK4 needs to be aligned to others selected in this register.
		0	No
		1	Yes
2	ALN3		SYSCLK3 needs to be aligned to others selected in this register.
		0	No
		1	Yes
1	ALN2		SYSCLK2 needs to be aligned to others selected in this register.
		0	No
		1	Yes
0	ALN1		SYSCLK1 needs to be aligned to others selected in this register.
		0	No
		1	Yes

6.4.13 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLDIV ratio change status register (DCHANGE) is shown in Figure 6-15 and described in Table 6-17. Indicates if SYSCLK divide ratio has been modified.

Figure 6-15. PLLDIV Ratio Change Status Register (DCHANGE)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		SYS5	SYS4	SYS3	SYS2	SYS1
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-17. PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	SYS5		SYSCLK5 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
3	SYS4		SYSCLK4 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
2	SYS3		SYSCLK3 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
1	SYS2		SYSCLK2 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
0	SYS1		SYSCLK1 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.

6.4.14 Clock Enable Control Register (CKEN)

The clock enable control register (CKEN) is shown in Figure 6-16 and described in Table 6-18. Clock enable control for miscellaneous output clocks.

Figure 6-16. Clock Enable Control Register (CKEN)

31	16
Reserved	
R-0	

15	1	0
Reserved		AUXEN
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-18. Clock Enable Control Register (CKEN) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	AUXEN		AUXCLK enable.
		0	Disable
		1	Enable

6.4.15 Clock Status Register (CKSTAT)

The clock status register (CKSTAT) is shown in Figure 6-17 and described in Table 6-19. Clock status for all clocks, except SYSCLK*n*.

Figure 6-17. Clock Status Register (CKSTAT)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		BPON	Reserved		AUXEN
R-0		R-1	R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-19. Clock Status Register (CKSTAT) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	BPON		SYSCLKBP on status.
		0	Off
		1	On
2-1	Reserved	0	Reserved
0	AUXEN		AUXCLK on status.
		0	Off
		1	On

6.4.16 System Clock Status Register (SYSTAT)

The system clock (SYSCLK) status register (SYSTAT) is shown in Figure 6-18 and described in Table 6-20. Indicates the SYSCLK on/off status. Actual default is determined by actual clock on/off status, which depends on the D*n*EN bit in PLLDIV*n* default.

Figure 6-18. System Clock Status Register (SYSTAT)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		SYS5ON	SYS4ON	SYS3ON	SYS2ON	SYS1ON
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-20. System Clock Status Register (SYSTAT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	SYS5ON		SYSCLK5 on status.
		0	Off
		1	On
3	SYS4ON		SYSCLK4 on status.
		0	Off
		1	On
2	SYS3ON		SYSCLK3 on status.
		0	Off
		1	On
1	SYS2ON		SYSCLK2 on status.
		0	Off
		1	On
0	SYS1ON		SYSCLK1 on status.
		0	Off
		1	On

6.4.17 PLL Controller Divider 5 Register (PLLDIV5)

The PLL controller divider 5 register (PLLDIV5) is shown in Figure 6-19 and described in Table 6-21. Divider 5 controls the divider for SYSCLK5. PLLDIV5 is not used on PLL2.

For PLL1, the RATIO bit is a fixed-field and cannot be changed.

Figure 6-19. PLL Controller Divider 5 Register (PLLDIV5)

31	16
Reserved	
R-0	

15	14	4	3	0
D5EN	Reserved		RATIO	
R/W-1	R-0		R/W-5h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-21. PLL Controller Divider 5 Register (PLLDIV5) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D5EN		Divider Enable.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 5h (PLL divide by 6).

Power and Sleep Controller

Topic **Page**

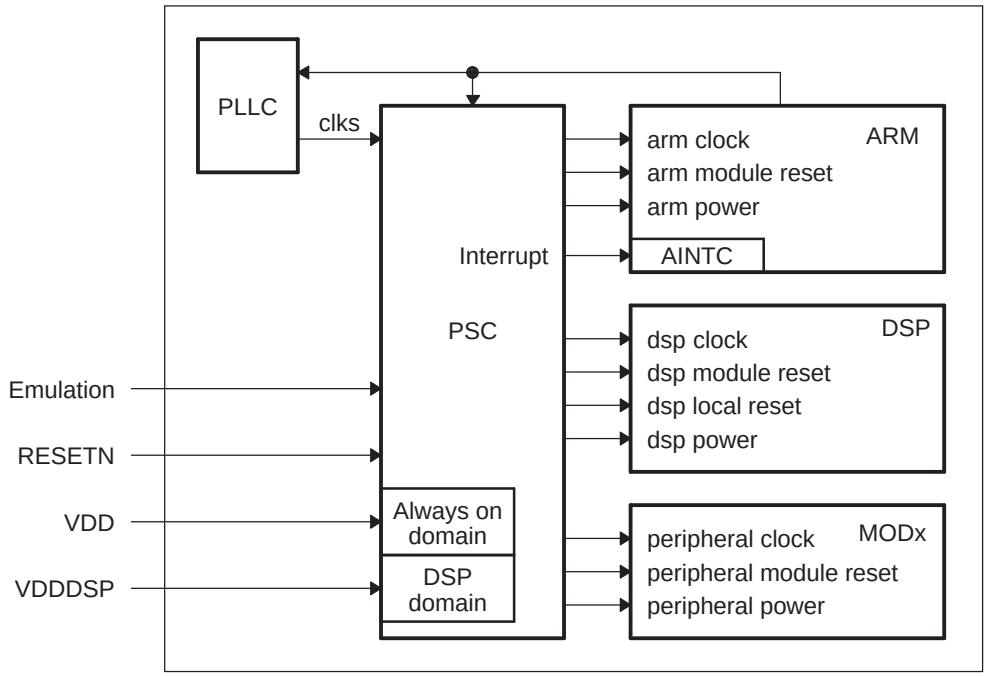
7.1 Introduction

In the TMS320DM644x DMSoC system, the Power and Sleep Controller (PSC) is responsible for managing transitions of system power on/off, clock on/off, and reset. A block diagram of the PSC is shown in Figure 7-1. Many of the operations of the PSC are transparent to software, such as power-on and hard reset operations. However, the PSC provides you with an interface to control several important power, clock, and reset operations. The power, clock, and reset operations are the focus of this chapter.

The PSC includes the following features:

- Manages chip power-on/off and resets
- Provides a software interface to:
  - Control DSP power domain
  - Control module clock ON/OFF
  - Control module resets
  - Control DSP local reset (CPU reset)
- Supports IcePick emulation features: power, clock, and reset

Figure 7-1. TMS320DM644x DMSoC Power and Sleep Controller (PSC)



7.2 TMS320DM644x DMSoC Power Domain and Module Topology

The DM644x DMSoC system includes two separate power domains and multiple separate modules, as shown in Figure 7-2 and summarized in Table 7-1. The AlwaysOn power domain is always on when the chip is on. The AlwaysOn domain is powered by the CV_{DD} pins of the DM644x DMSoC (see the device-specific data manual). The majority of the DM644x DMSoC modules lie within the AlwaysOn power domain. The DSP Subsystem lies in a separate domain that is not always on. This domain is referred to as the DSP domain. The DSP power domain is powered by the CV_{DDDSP} pins of the DM644x DMSoC (see the device-specific data manual).

NOTE: All DM644x devices do not support powering down the DSP power domain. See the device-specific data manual for more information on which DM644x devices support powering down the DSP power domain.

Figure 7-2. TMS320DM644x DMSoC Power Domain and Module Topology

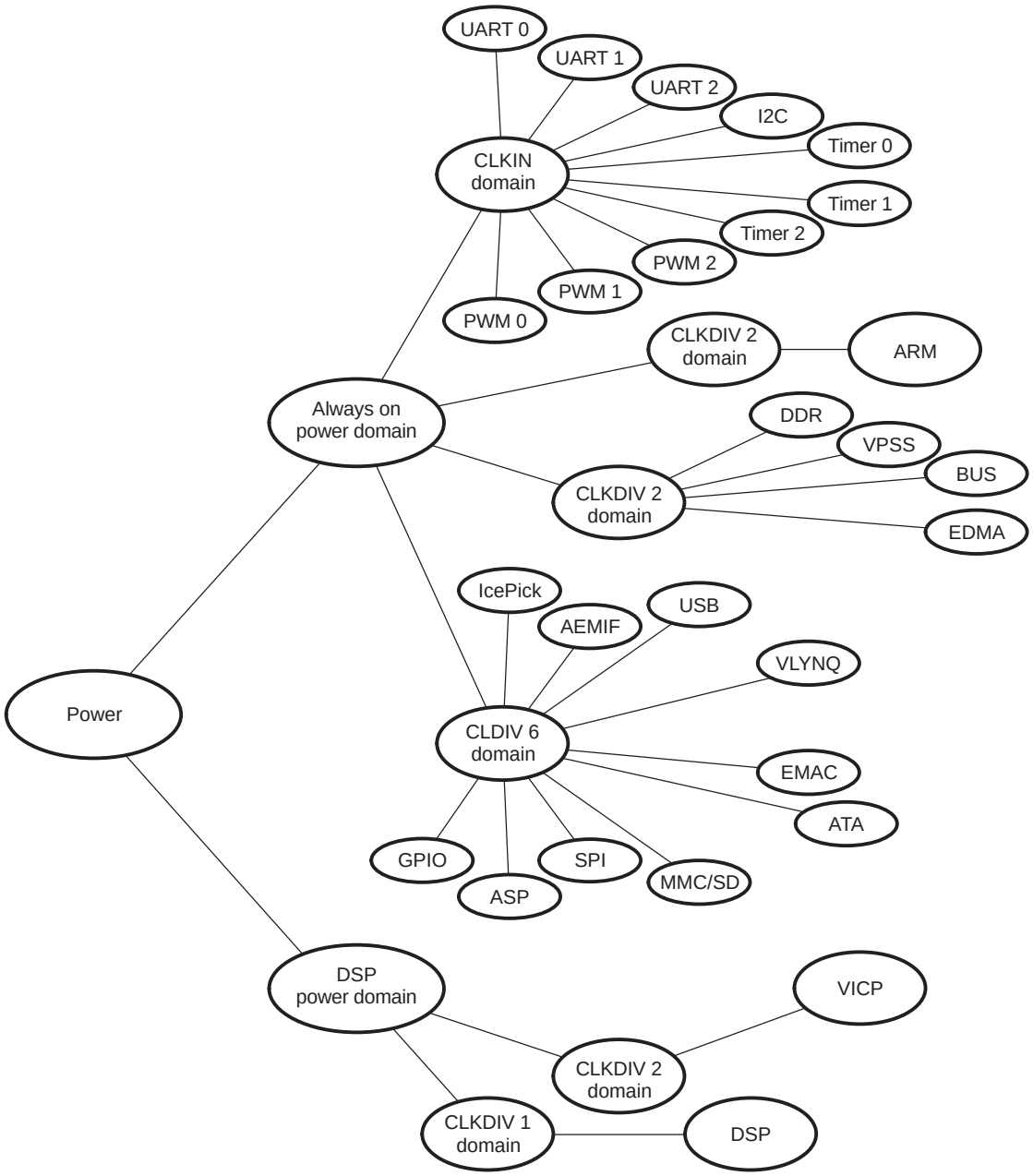


Table 7-1. Module Configuration

Module Number	Module Name	Power Domain	Default States		
			Power Domain State	Module State	Local Reset State
0	VPSS (master)	AlwaysOn	ON	Disable	-
1	VPSS (slave)	AlwaysOn	ON	Disable	-
2	CC	AlwaysOn	ON	BTSEL = 00: Enable (NAND/SPI) BTSEL = 01: Enable (NOR)	-
3	TC0	AlwaysOn	ON	BTSEL = 10: Reserved	-
4	TC1	AlwaysOn	ON	BTSEL = 11: Enable (UART)	-
5	EMAC	AlwaysOn	ON	Disable	-
6	EMAC	AlwaysOn	ON	Disable	-
9	USB	AlwaysOn	ON	Disable	-
10	ATA	AlwaysOn	ON	Disable	-
11	VLYNQ	AlwaysOn	ON	Disable	-
12	HPI	AlwaysOn	ON	Disable	-
13	DDR EMIF	AlwaysOn	ON	BTSEL = 11: Disable (UART)	-
14	AEMIF	AlwaysOn	ON	BTSEL = 00: Enable (NAND/SPI) BTSEL = 01: Enable (NOR) BTSEL = 10: Reserved BTSEL = 11: Enable (UART)	-
15	MMC/SD	AlwaysOn	ON	Disable	-
17	ASP	AlwaysOn	ON	Disable	-
18	I2C	AlwaysOn	ON	Disable	-
19	UART0	AlwaysOn	ON	BTSEL = 00: Disable (NAND/SPI) BTSEL = 01: Disable (NOR) BTSEL = 10: Reserved BTSEL = 11: Enable (UART)	-
20	UART1	AlwaysOn	ON	Disable	-
21	UART2	AlwaysOn	ON	Disable	-
22	SPI	AlwaysOn	ON	Disable	-
23	PWM0	AlwaysOn	ON	Disable	-
24	PWM1	AlwaysOn	ON	Disable	-
25	PWM2	AlwaysOn	ON	Disable	-
26	GPIO	AlwaysOn	ON	Disable	-
27	TIMER0	AlwaysOn	ON	Enable	-
28	TIMER1	AlwaysOn	ON	Disable	-
29	TIMER2	AlwaysOn	ON	Enable	-
30	System Module	AlwaysOn	ON	Enable	-
31	ARM	AlwaysOn	ON	Enable	-
32-38	Internal Bus	AlwaysOn	ON	Enable	-
39	DSP	DSP	OFF	COUT3_DSP_BT	COUT3_DSP_BT
40	VICP	DSP	OFF	Disable	-

7.3 Power Domain and Module States Defined

Table 7-2 shows the state of each module after chip power-on/hard reset. These states are defined in the following sections. The default state of the DSP Power Domain and the DSP module is determined by the DSP boot select pin (COUT3_DSP_BT). If the DSP is selected to self-boot at reset via this signal, the DSP domain powers-up by default.

Table 7-2. Module States

Module State	Module Reset	Module Clock	Module State Definition
Enable	De-asserted	On	A module in the enable state has its module reset de-asserted and it has its clock on. This is the normal run-time state for a given module.
Disable	De-asserted	Off	A module in the disable state has its module reset de-asserted and it has its clock off. This state is typically used for disabling a module clock to save power. The DM644x DMSoC is designed in full static CMOS, so when you stop a module clock, it retains the module's state. When the clock is restarted, the module resumes operating from the stopping point.
SyncReset	Asserted	On	A module in the SyncReset state has its module reset asserted and it has its clock on. After initial power-on, most modules are in the SyncRst state by default (see Table 7-1). Generally, software is not expected to initiate this state.
SwRstDisable	Asserted	Off	A module in the SwResetDisable state has its module reset asserted and it has its clock set to off. Generally, software is not expected to initiate this state.

7.3.1 Power Domain States

A power domain can only be in one of two states: ON or OFF, defined as follows:

- ON: power to the power domain is on.
- OFF: power to the power domain is off.

In the DM644x DMSoC, the AlwaysOn Power Domain is always in the ON state when the chip is powered-on. However, the DSP Power Domain can either be in the ON state or in the OFF state. (that is, the DSP Subsystem can be powered-down independently of the rest of the DMSoC to conserve power when the imaging and video functions are not needed.)

NOTE: All DM644x devices do not support powering down the DSP power domain. See the device-specific data manual for more information on which DM644x devices support powering down the DSP power domain.

7.3.2 Module States

A module can be in one of four states: Disable, Enable, SyncReset, or SwRstDisable. These four states correspond to combinations of module reset asserted or de-asserted and module clock on or off, as shown in Table 7-2.

NOTE: Module Reset is defined to completely reset a given module, so that all hardware returns to its default state. See Chapter 11 and Chapter 12 for more information on module reset.

For more information on the DM644x DMSoC power management, see Chapter 8.

7.3.3 Local Reset

In addition to module reset (described in Section 7.3.2), the DSP CPU can be reset using a special local reset. When DSP local reset is asserted, the DSP's internal memories (L1P, L1D, and L2) are still accessible. The local reset only resets the DSP CPU core, not the rest of the DSP subsystem, as the DSP module reset would. Local reset is useful when the DSP module is in the enable state or in the disable state; since module reset is asserted in the SyncReset and SwRstDisable states, and module reset takes precedence over local reset. The ARM uses local reset to reset the DSP to initiate the DSP boot process. See Section 10.5 for more detailed information on how to boot and control the DSP. See Chapter 11 and Chapter 12 for more information on local reset, as well as DSP boot.

The procedures for asserting and de-asserting DSP local reset are as follows:

- Clear the LRSTZ bit in MDCTL39 to 0 to assert the DSP local reset. After power-on reset or hard reset, boot configuration pin COUT3_DSP_BT determines the default state of the LRSTZ bit in MDCTL39. See Chapter 11 and Chapter 12 for more information on this boot configuration pin.
- Set the LRSTZ bit in MDCTL39 to 1 to de-assert DSP local reset. If the DSP is in the enable state, it immediately executes program instructions after reset is de-asserted.

7.4 Executing State Transitions

This section describes how to execute state transitions for power domains and modules.

NOTE: All DM644x devices do not support powering down the DSP power domain. See the device-specific data manual for more information on which DM644x devices support powering down the DSP power domain.

7.4.1 Power Domain State Transitions

This section describes the basic procedure for transitioning the state of a power domain, which is limited to the DSP power domain in the DM644x DMSoC. The procedure assumes that a device external to the DM644x chip and controlled by ARM software applies and removes power. The PSC handles all of the required internal operations, such as de-activating and activating isolation cells around the DSP on power transitions, so you must follow this process. The isolation cells must be de-activated for power-on and activated for power-off.

NOTE: As previously mentioned, there are two power domains in the DM644x DMSoC: the AlwaysOn power domain and the DSP power domain. The AlwaysOn power domain is always in the on state when the chip is powered-on; therefore, it is not possible to transition the AlwaysOn power domain to the off state. Conversely, the DSP power domain can be in the on and off states when the chip is powered-on. You must be aware of several system considerations to transition the DSP power domain. These system considerations and the procedures for transitioning DSP power domains are described in Chapter 13. Chapter 13 provides an overview of the procedure for power domain state transitions with respect to the PSC.

The procedure for power domain state transitions is as follows (*n* corresponds to the power domain):

1. Wait for the GOSTAT[*n*] bit in PTSTAT to clear to 0. You must wait for any previously initiated transitions to finish before initiating a new transition.
2. Set the NEXT bit in PDCTL*n* for an ON (1) or OFF (0) transition.

NOTE: When GO[*n*] is set to 1 in the next step, the NEXT bit in PDCTL*n* of this power domain and the NEXT bit in MDCTL*n* of all modules in this power domain are evaluated. Therefore, you may set the NEXT bit in MDCTL*n* for multiple modules before executing this step.

3. Set the GO[*n*] bit in PTCMD to 1 to initiate the state transition(s). For power on/off, the PSC de-actives/activates isolation cells that exist around the DSP.
4. Wait for the EPC bit in EPCPR to change to 1 to indicate that the PSC is ready for external power to be applied or removed. The PSC changes the EPC bit in EPCPR to 1, as an indication to software that it is pending confirmation that power was applied to or removed from the power pins.

NOTE: This step can be done by polling the EPC bit in EPCPR or by enabling the PSC's external power control pending interrupt. See Section 7.6 for detailed information on this interrupt.

5. Apply or remove power to or from the power pins of the power domain and toggle any associated system bits. The ARM coordinates with an external device to apply or remove power to or from the power pins, and the ARM coordinates with an external device to apply or remove power to or from the power pins, and the ARM toggles any necessary system bits in the DM644x DMSoC system. See Section 10.5 for specific information on ARM control of DSP power on/off.
6. Set the EPCGOOD bit in PDCTL*n* to 1, to indicate that power has been applied; or clear to 0, to indicate that power has been removed. The PSC proceeds with the transition after software writes the EPCGOOD bit in PDCTL*n*.
7. Wait for the GOSTAT[*n*] bit in PTSTAT to clear to 0. The domain is safely in the new state only after the GOSTAT[*n*] bit in PTSTAT is cleared to 0.

7.4.2 Module State Transitions

This section describes the procedure for transitioning the module state.

NOTE: The following procedure is directly applicable for all modules, except for the DSP in the DM644x DMSoC. To transition the DSP module state, you must be aware of several system considerations. These system considerations and the procedures for transitioning the DSP module state are described in detail in Chapter 13.

The procedure for module state transitions is as follows (*n* corresponds to the module):

- Wait for the GOSTAT[*n*] bit in PTSTAT to clear to 0. You must wait for any previously initiated transitions to finish before initiating a new transition.
- Set the NEXT bit in MDCTL*n* to SwRstDisable (0), SyncReset (1), Disable (2h), or Enable (3h).

NOTE: You may set transitions in multiple NEXT bits in MDCTL*n* in this step.

- Set the EMURSTIE bit in MDCTL*n* to 1, if the module you want to transition is any of the following:
  - VPSS (slave)
  - EMAC
  - USB
  - ATA
  - VLYNQ
  - DDR2 memory controller
  - ASYNC EMIF
  - MMC/SD
  - ASP
  - GPIO
  - VICP

This is a special step required for these particular modules. This step is not required for any module that is not listed.

NOTE: The EMURSTIE bit in MDCTL*n* is also used for PSC emulation features. The emulation features are described in Section 7.5.

- Set the GO[*n*] bit in PTCMD to 1 to initiate the transition(s).
- Wait for the GOSTAT[*n*] bit in PTSTAT to clear to 0. The module is safely in the new state only after the GOSTAT[*n*] bit in PTSTAT is cleared to 0.

7.5 IcePick Emulation Support in the PSC

The PSC supports IcePick commands that allow IcePick aware emulation tools to have some control over the state of power domains and modules.

In particular, the PSC supports the following IcePick emulation commands:

Table 7-3. IcePick Emulation Commands

Power On and Enable Features	Power On and Enable Descriptions	Reset Features	Reset Descriptions
Inhibit Sleep	Allows emulation to prevent software from transitioning the power domain out of the on state and to prevent software from transitioning the module out of the enable state	Assert Reset	Allows emulation to assert the module's local reset.
Force Power	Allows emulation to force the power domain into an on state	Wait Reset	Allows emulation to keep local reset asserted for an extended period of time after software initiates local reset de-assert.
Force Active	Allows emulation to force the power domain into an on state and force the module into the enable state.	Block Reset	Allows emulation to block software initiated local and module resets.

NOTE: When emulation tools assert the ForcePower or ForceActive states, state transition is dependent on the ARM applying power and notifying the PSC. If the ARM does not complete the process, then the state transition does not complete and the emulation tools may hang.

When emulation tools remove the above commands, the PSC immediately executes a state transition based on the current values in the NEXT bit in PDCTL*n* and the NEXT bit in MDCTL*n*, as set by software.

7.6 PSC Interrupts

The PSC has an interrupt that is tied to the ARM Interrupt Controller. This interrupt is named PSCINT in the ARM interrupt map. The PSC interrupt is generated when certain IcePick emulation events occur and during the DSP power domain on/off procedure.

7.6.1 Interrupt Events

The PSC interrupt is generated when any of the following events occur:

- Power Domain Emulation Event
- Module State Emulation Event
- Module Local Reset Emulation Event
- External Power Control Pending Event

These interrupt events are summarized in Table 7-4 and described in more detail in this section.

Table 7-4. PSC Interrupt Events

Interrupt Enable Bits		
Control Register	Enable Bit	Interrupt Condition
PDCTL*n*	EMUIHBIE	Interrupt occurs when the emulation alters the power domain state
MDCTL*n*	EMUIHBIE	Interrupt occurs when the emulation alters the module state
MDCTL*n*	EMURSTIE	Interrupt occurs when the emulation alters the module's local reset
EPCPR	EPC	Interrupt occurs during the power domain on/off sequence

7.6.1.1 Power Domain Emulation Events

A power domain emulation event occurs when emulation alters the state of a power domain. Status is reflected in the EMUIHB bit in PDSTAT*n*. In particular, a power domain emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the on state
- When force power is asserted by emulation and power domain is not already in the on state
- When force active is asserted by emulation and power domain is not already in the on state

NOTE: All DM644x devices do not support powering down the DSP power domain. See the device-specific data manual for more information on which DM644x devices support powering down the DSP power domain.

7.6.1.2 Module State Emulation Events

A module state emulation event occurs when emulation alters the state of a module. Status is reflected in the EMUIHB bit in the MDSTAT*n*. In particular, a module state emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the enable state
- When force active is asserted by emulation and module is not already in the enable state

7.6.1.3 Local Reset Emulation Events

A local reset emulation event occurs when emulation alters the local reset of a module. Status is reflected in the EMRST bit in MDSTAT*n*. In particular, a module local reset emulation event occurs under the following conditions:

- When assert reset is asserted by emulation although software de-asserted the local reset
- When wait reset is asserted by emulation
- When block reset is asserted by emulation and software attempts to change the state of local reset

7.6.1.4 External Power Control Pending Event

An external power control pending event occurs during the power domain power on or power off sequences. The PSC triggers this interrupt as an indication that it is ready for software to apply or remove power during a power on or power off transition sequence, respectively. Status for this interrupt is reflected in the EPC bit in EPCPR. See Section 7.4.1 for more information.

The external power control pending event occurs when the PSC is pending confirmation that power was applied to or removed from the power pins. See Section 7.4.1 for more information.

NOTE: All DM644x devices do not support powering down the DSP power domain. See the device-specific data manual for more information on which DM644x devices support powering down the DSP power domain.

7.6.2 Interrupt Registers

The PSC interrupt enable bits are: the EMUIHBIE bit in PDCTL*n*, the EMUIHBIE bit and the EMURSTIE bit in MDCTL*n*, and the EPC bit in EPCPR.

NOTE: To interrupt the ARM, the ARM's power and sleep controller interrupt (PSCINT) must also be enabled in the ARM interrupt controller. See Chapter 9 for more information on the ARM's power and sleep controller interrupt and the ARM interrupt controller.

The PSC interrupt status bits are the M0[*n*] bit in MERRPR0, the M[*n*] bit in MERRPR1, the P[*n*] bit in PERRPR, the EMUIHB bit in PDSTAT*n*, the EMUIHB bit and the EMURST bit in MDSTAT*n*, and the EPC bit in EPCPR. The status bits in MERRPR0, MERRPR1, and PERRPR are read by software to determine which power domain or which module has generated an emulation interrupt, and then software can read the corresponding status bits in PDSTAT*n* and MDSTAT*n* to determine which event caused the interrupt.

The PSC interrupt clear bits are the M[*n*] bit in MERRCR*n*, the P[*n*] bit in PERRCR, and the EPC bit in EPCCR.

The PSC interrupt evaluation bit is the ALLEV bit in INTEVAL. When set, this bit forces the PSC interrupt logic to re-evaluate event status. If any events are still active (if any status bits are set) when the ALLEV bit in INTEVAL is set to 1, the PSCINT is re-asserted to the ARM interrupt controller. Set the ALLEV bit in INTEVAL before exiting your PSCINT interrupt service routine to ensure that you do not miss any PSC interrupts while the ARM interrupts are globally disabled.

See Section 7.7 for complete descriptions of all PSC registers.

7.6.3 Interrupt Handling

Handle the PSC interrupts as described in the following procedure:

First, enable the interrupt.

1. Set the EMUIHBIE bit in PDCTL*n*, the EMUIHBIE bit and the EMURSTIE bit in MDCTL*n* to enable the interrupt events that you want.

NOTE: There is no enable bit for the external power control pending interrupt event, so effectively this event is always enabled. The PSC interrupt is sent to the ARM interrupt controller when at least one enabled event becomes active.

2. Enable the ARM's power and sleep controller interrupt (PSCINT) in the ARM interrupt controller. To interrupt the ARM, PSCINT must be enabled in the ARM interrupt controller. See Chapter 9 for more information.

The ARM enters the interrupt service routine (ISR) when it receives the interrupt.

1. Read the P[*n*] bit in PERRPR, the M[*n*] bit in MERRPR0, the M[*n*] bit in MERRPR1, and/or the EPC bit in EPCPR to determine the source of the interrupt(s).
2. For each active event that you want to service:
   - Read the event status bits in PDSTAT*n* and MDSTAT*n*, depending on the status bits read in the previous step to determine the event that caused the interrupt.
   - Service the interrupt as required by your application.
   - Write the M[*n*] bit in MERRCR*n*, the P[*n*] bit in PERRCR, and the EPC bit in EPCCR to clear corresponding status.
   - Set the ALLEV bit in INTEVAL. Setting this bit reasserts the PSCINT to the ARM's interrupt controller, if there are still any active interrupt events.

7.7 PSC Registers

Table 7-5 lists the memory-mapped registers for the PSC.

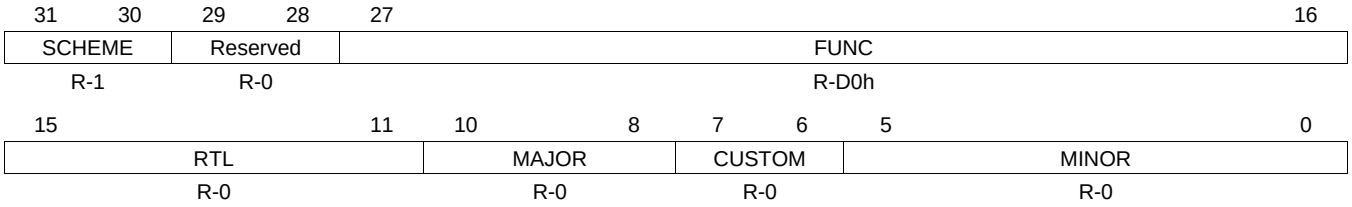
Table 7-5. Power and Sleep Controller (PSC) Registers

Address	Acronym	Register Description	Section
1C4 1000h	PID	Peripheral Revision and Class Information Register	Section 7.7.1
1C4 1018h	INTEVAL	Interrupt Evaluation Register	Section 7.7.2
1C4 1040h	MERRPR0	Module Error Pending Register 0 (modules 0-31)	Section 7.7.3
1C4 1044h	MERRPR1	Module Error Pending Register 1 (modules 32-63)	Section 7.7.4
1C4 1050h	MERRCR0	Module Error Clear Register 0 (modules 0-31)	Section 7.7.5
1C4 1054h	MERRCR1	Module Error Clear Register 1 (modules 32-63)	Section 7.7.6
1C4 1060h	PERRPR	Power Error Pending Register	Section 7.7.7
1C4 1068h	PERRCR	Power Error Clear Register	Section 7.7.8
1C4 1070h	EPCPR	External Power Error Pending Register	Section 7.7.9
1C4 1078h	EPCCR	External Power Control Clear Register	Section 7.7.10
1C4 1120h	PTCMD	Power Domain Transition Command Register	Section 7.7.11
1C4 1128h	PTSTAT	Power Domain Transition Status Register	Section 7.7.12
1C4 1200h	PDSTAT0	Power Domain (Always On) Status Register 0	Section 7.7.13
1C4 1204h	PDSTAT1	Power Domain (DSP) Status Register 1	Section 7.7.13
1C4 1300h	PDCTL0	Power Domain (Always On) Control Register 0	Section 7.7.14
1C4 1304h	PDCTL1	Power Domain (DSP) Control Register 1	Section 7.7.14
1C4 1800h	MDSTAT*n*	Module Status *n* Register (modules 0-40)	Section 7.7.15
1C4 1A00h	MDCTL*n*	Module Control *n* Register (modules 0-40)	Section 7.7.16

7.7.1 Peripheral Revision and Class Information Register (PID)

The peripheral revision and class information (PID) register is shown in Figure 7-3 and described in Table 7-6.

Figure 7-3. Peripheral Revision and Class Information Register (PID)



LEGEND: R = Read only; -*n* = value after reset

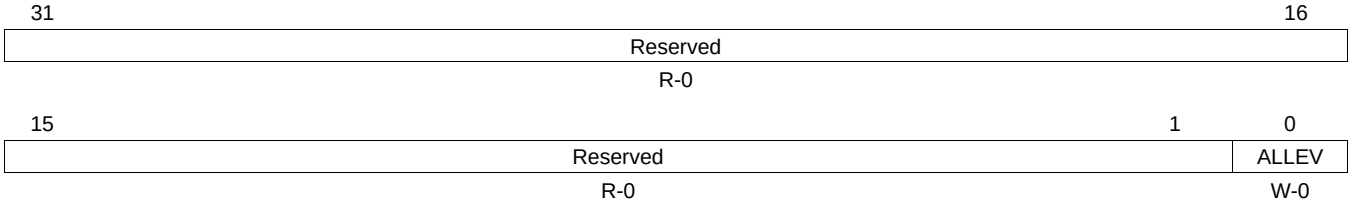
Table 7-6. Peripheral Revision and Class Information Register (PID) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	0-3h	Distinguishes between the old scheme and the current scheme. There is a spare bit to encode future schemes.
29-28	Reserved	0	Reserved
27-16	FUNC	0-FFFh	Indicates a software compatible module family.
15-11	RTL	0-1Fh	RTL Version.
10-8	MAJOR	0-7h	Major Revision.
7-6	CUSTOM	0-3h	Indicates a special version for a particular device.
5-0	MINOR	0-3Fh	Minor Revision.

7.7.2 Interrupt Evaluation Register (INTEVAL)

The interrupt evaluation register (INTEVAL) is shown in Figure 7-4 and described in Table 7-7.

Figure 7-4. Interrupt Evaluation Register (INTEVAL)



LEGEND: R = Read only; W= Write only; -*n* = value after reset

Table 7-7. Interrupt Evaluation Register (INTEVAL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ALLEV		Evaluate PSC interrupt.
		0	A write of 0 has no effect.
		1	A write of 1 re-evaluates the interrupt condition.

7.7.3 Module Error Pending Register 0 (MERRPR0)

The module error pending register 0 (modules 0-31) (MERRPR0) is shown in Figure 7-5 and described in Table 7-8.

Figure 7-5. Module Error Pending Register 0 (MERRPR0)

31	0
M[*n*]	
R- 0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-8. Module Error Pending Register 0 (MERRPR0) Field Descriptions

Bit	Field	Value	Description
31-0	M[*n*]		Module interrupt status bit for modules 0-31.
		0	Power domain interrupt *n* is not active.
		1	Power domain interrupt *n* is active.

7.7.4 Module Error Pending Register 1 (MERRPR1)

The module error pending register 1 (modules 32-40) (MERRPR1) is shown in Figure 7-6 and described in Table 7-9.

Figure 7-6. Module Error Pending Register 1 (MERRPR1)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		M[*n*]	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-9. Module Error Pending Register 1 (MERRPR1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8-0	M[*n*]		Module interrupt status bit for modules 32-40.
		0	Power domain interrupt *n* is not active.
		1	Power domain interrupt *n* is active.

7.7.5 Module Error Clear Register 0 (MERRCR0)

The module error clear register 0 (modules 0-31) (MERRCR0) is shown in Figure 7-7 and described in Table 7-10.

Figure 7-7. Module Error Clear Register 0 (MERRCR0)

31	0
M[*n*]	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 7-10. Module Error Clear Register 0 (MERRCR0) Field Descriptions

Bit	Field	Value	Description
31-0	M[*n*]		Clears the interrupt bit set in the corresponding module error pending register 0 (MERRPRO) bit field and the module status *n* register (MDSTAT*n*) interrupt bit fields. This is for modules 0-31.
		0	A write of 0 has no effect.
		1	Clears module interrupt *n*.

7.7.6 Module Error Clear Register 1 (MERRCR1)

The module error clear register 1 (modules 32-40) (MERRCR1) is shown in Figure 7-8 and described in Table 7-11.

Figure 7-8. Module Error Clear Register 1 (MERRCR1)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		M[*n*]	
R-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 7-11. Module Error Clear Register 1 (MERRCR1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8-0	M[*n*]		Clears the interrupt bit set in the corresponding module error pending register 1 (MERRPR1) bit field and the module status *n* register (MDSTAT*n*) interrupt bit fields. This is for modules 32-40.
		0	A write of 0 has no effect.
		1	Clears module interrupt *n*.

7.7.7 Power Error Pending Register (PERRPR)

The power error pending register (PERRPR) is shown in Figure 7-9 and described in Table 7-12.

Figure 7-9. Power Error Pending Register (PERRPR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		P	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-12. Power Error Pending Register (PERRPR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P		DSP Power domain interrupt status.
		0	DSP Power domain interrupt is not active.
		1	DSP Power domain interrupt is active.
0	P		Always On Power domain interrupt status.
		0	Always On Power domain interrupt is not active.
		1	Always On Power domain interrupt is active.

7.7.8 Power Error Clear Register (PERRCR)

The power error clear register (PERRCR) is shown in Figure 7-10 and described in Table 7-13.

Figure 7-10. Power Error Clear Register (PERRCR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		P	
R-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 7-13. Power Error Clear Register (PERRCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P		Clear DSP power domain interrupt.
		0	A write of 0 has no effect.
		1	Clears the DSP power domain interrupt.
0	P		Clear Always On power domain interrupt.
		0	A write of 0 has no effect.
		1	Clears the Always On power domain interrupt.

7.7.9 External Power Control Pending Register (EPCPR)

The external power control pending register (EPCPR) is shown in Figure 7-11 and described in Table 7-14.

Figure 7-11. External Power Control Pending Register (EPCPR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		EPC	Rsvd
R-0		R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 7-14. External Power Control Pending Register (EPCPR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	EPC		External power control pending bit. The PSC sets this bit, indicating it is ready for an external controller to apply power to the external power pins of the DSP power domain.
		0	The PSC is not requesting external power control.
		1	The PSC requests external power control.
0	Reserved	0	Reserved

7.7.10 External Power Control Clear Register (EPCCR)

The external power control clear register (EPCCR) is shown in Figure 7-12 and described in Table 7-15.

Figure 7-12. External Power Control Clear Register (EPCCR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		EPC	Rsvd
R-0		W-0	R-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 7-15. External Power Control Clear Register (EPCCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	EPC		External power control clear bit (DSP power domain) .
		0	A write of 0 has no effect.
		1	Clears the EPCPR interrupt.
0	Reserved	0	Reserved

7.7.11 Power Domain Transition Command Register (PTCMD)

The power domain transition command register (PTCMD) is shown in Figure 7-13 and described in Table 7-16.

Figure 7-13. Power Domain Transition Command Register (PTCMD)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		GO	
R-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 7-16. Power Domain Transition Command Register (PTCMD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GO		DSP Power domain GO transition command.
		0	A write of 0 has no effect.
		1	Writing 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including PDCTL.NEXT for this domain, and MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (PDSTAT.STATE, MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.
0	GO		Always On Power domain GO transition command.
		0	A write of 0 has no effect.
		1	Writing 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.

7.7.12 Power Domain Transition Status Register (PTSTAT)

The power domain transition status register (PTSTAT) is shown in Figure 7-14 and described in Table 7-17 .

Figure 7-14. Power Domain Transition Status Register (PTSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		GOSTAT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-17. Power Domain Transition Status Register (PTSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GOSTAT		DSP Power domain transition status.
		0	No transition in progress.
		1	DSP Power domain is transitioning (that is, either the power domain is transitioning or modules in this power domain are transitioning).
0	GOSTAT		Always On Power domain transition status.
		0	No transition in progress.
		1	Modules in Always On power domain are transitioning. Always On Power domain is transitioning.

7.7.13 Power Domain Status n Register (PDSTAT0-PDSTAT1)

The power domain status *n* register (PDSTAT*n*) is shown in Figure 7-15 and described in Table 7-18.

Figure 7-15. Power Domain Status *n* Register (PDSTAT*n*)

31	16
Reserved	
R-0	

15	12	11	10	9	8	7	5	4	0
Reserved		EMUIHB	Reserved	PORDONE	POR	Reserved		STATE	
R-0		R-0	R-0	R-0	R-0	R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

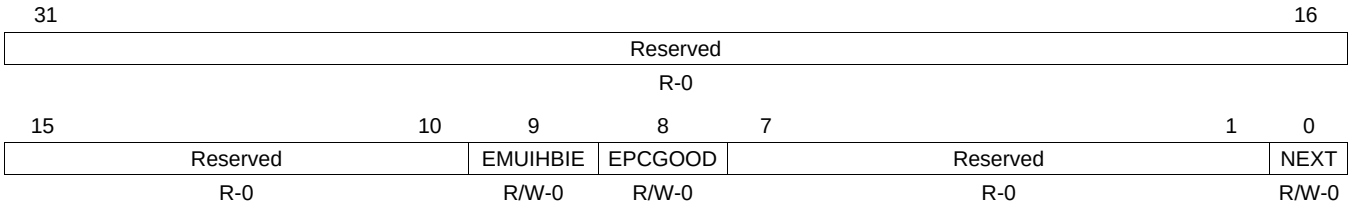
Table 7-18. Power Domain Status *n* Register (PDSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	EMUIHB		Emulation alters domain state.
		0	Interrupt is not active.
		1	Interrupt is active.
10	Reserved	0	Reserved
9	PORDONE		Power_On_Reset (POR) Done status
		0	Power domain POR is not done.
		1	Power domain POR is done.
8	POR		Power Domain Power_On_Reset (POR) status. This bit reflects the POR status for this power domain including all modules in the domain.
		0	Power domain POR is asserted.
		1	Power domain POR is de-asserted.
7-5	Reserved	0	Reserved
4-0	STATE		Power Domain Status
		0	Power domain is in the off state.
		1	Power domain is in the on state.

7.7.14 Power Domain Control n Register (PDCTL0-PDCTL1)

The power domain control *n* register (PDCTL*n*) is shown in Figure 7-16 and described in Table 7-19.

Figure 7-16. Power Domain Control *n* Register (PDCTL*n*)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-19. Power Domain Control *n* Register (PDCTL*n*) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	EMUIHBIE		Emulation alters power domain state interrupt enable.
		0	Disable interrupt.
		1	Enable interrupt.
8	EPCGOOD		External power control power good indication.
		0	External power control has turned off power to this domain.
		1	External power control has turned on power to this domain.
7-1	Reserved	0	Reserved
0	NEXT		Power domain next state.
		0	Power domain off.
		1	Power domain on.

7.7.15 Module Status n Register (MDSTAT0-MDSTAT40)

The module status *n* register (MDSTAT*n*) is shown in Figure 7-17 and described in Table 7-20.

Figure 7-17. Module Status *n* Register (MDSTAT*n*)

31	18	17	16
Reserved		EMUIHB	EMURST
R-0		R-0	R-0

15	13	12	11	10	9	8	7	6	5	0
Reserved		MCKOUT	MRSTDONE	MRST	LRSTDONE	LRST	Reserved		STATE	
R-0		R-0	R-0	R-0	R-0	R-0	R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-20. Module Status *n* Register (MDSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	EMUIHB		'Emulation Alters Module State' Interrupt active.
		0	Interrupt is not active.
		1	Interrupt is active.
16	EMURST		'Emulation Alters Module Reset' Interrupt active.
		0	Interrupt is not active.
		1	Interrupt is active.
15-13	Reserved	0	Reserved
12	MCKOUT		Module clock output status. Shows status of module clock.
		0	Module clock is off.
		1	Module clock is on.
11	MRSTDONE		Module reset done. Software is responsible for checking that mode reset is done before accessing the module.
		0	Module reset is not done.
		1	Module reset is done.
10	MRST		Module reset status. Reflects actual state of module reset.
		0	Module reset is asserted.
		1	Module reset is de-asserted.
9	LRSTDONE		Local reset done. Software is responsible for checking if local reset is done before accessing this module.
		0	Local reset is not done.
		1	Local reset is done.
8	LRST		Module local reset status (this bit applies to the DSP module only).
		0	Local reset is asserted.
		1	Local reset is de-asserted.
7-6	Reserved	0	Reserved
5-0	STATE	0-3Fh	Module state status: indicates current module status.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state
		4h-3Fh	Indicates transition

7.7.16 Module Control n Register (MDCTL0-MDCTL40)

The module control *n* register (MDCTL*n*) is shown in Figure 7-18 and described in Table 7-21.

Figure 7-18. Module Control *n* Register (MDCTL*n*)

31						16
Reserved						
R- 0						

15	11	10	9	8	7	3	2	0
Reserved		EMUIHBIE	EMURSTIE	LRST	Reserved		NEXT	
R- 0		R/W- 0	R/W- 0	R/W- 0	R- 0		R/W- 0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-21. Module Control *n* Register (MDCTL*n*) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	EMUIHBIE		Interrupt enable for emulation alters module state.
		0	Disable interrupt.
		1	Enable interrupt.
9	EMURSTIE		Interrupt enable for emulation alters reset.
		0	Disable interrupt.
		1	Enable interrupt.
8	LRST		Module local reset control (This bit applies to the DSP module only.)
		0	Assert local reset
		1	De-assert local reset
7-3	Reserved	0	Reserved
2-0	NEXT	0-3h	Module next state.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

Power Management

Topic Page

8.1 Overview

In many applications, there may be specific requirements to minimize power consumption for both power supply (or battery) and thermal considerations. There are two components to power consumption: active power and leakage power. Active power is the power consumed to perform work and scales roughly with clock frequency and the amount of computations being performed. Active power can be reduced by controlling the clocks in such a way as to either operate at a clock setting just high enough to complete the required operation in the required timeline or to run at a clock setting until the work is complete and then drastically cut the clocks (that is, to PLL Bypass mode) until additional work must be performed. Leakage power is due to static current leakage and occurs regardless of the clock rate. Leakage, or standby power, is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely from a device or subsystem.

The TMS320DM644x DMSoC has several means of managing the power consumption, as detailed in the following sections. There is extensive use of automatic clock gating in the design as well as software-controlled module clock gating to not only reduce the clock tree power, but to also reduce module power by basically freezing its state while not operating. Clock management enables you to slow the clocks down on the chip in order to reduce switching power. In particular, the DM644x DMSoC includes all of the power management features described in Table 8-1.

Table 8-1. Power Management Features

Power Management Features	Description
	Clock Management
PLL power-down	The PLLs can be powered-down when not in use to reduce switching power
Module clock ON/OFF	Module clocks can be turned on/off to reduce switching power
Module clock frequency scaling	Module clock frequency can be scaled to reduce switching power
	ARM and DSP Sleep Management
ARM Wait-for-Interrupt sleep mode	Disable ARM clock to reduce active power
DSP sleep modes	The DSP can be put into sleep mode to reduce switching power
	I/O Management
3.3 Volt I/O power-down	The 3.3 V I/Os can be powered-down to reduce I/O cell power
USB Phy power-down	The USB Phy can be powered-down to reduce USB I/O power
DAC power-down	The DAC's can be powered-down to reduce DAC power

8.2 PSC and PLLC Overview

The power and sleep controller (PSC) plays an important role in managing system power on/off, clock on/off, and reset. Similarly, the PLL controller (PLLC) plays an important role in device clock generation. The PSC and the PLLC are mentioned throughout this chapter. For detailed information on the PSC, see Chapter 7. For detailed information on the PLLC, see Chapter 5 and Chapter 6.

8.3 Clock Management

8.3.1 Module Clock ON/OFF

The module clock on/off feature allows software to disable clocks to module individually, in order to reduce the module's active power consumption to 0. The DM644x DMSoC is designed in full static CMOS; thus, when a module clock stops, the module's state is preserved. When the clock is restarted, the module resumes operating from the stopping point.

NOTE: Stopping clocks to a module only affects active power consumption, it does not affect leakage power consumption.

If a module's clock(s) is stopped while the configuration bus or the EDMA bus is accessing it, the access may not occur, and could potentially lock-up the bus. Ensure that all of the transactions to the module are finished prior to stopping the clocks.

The power and sleep controller (PSC) controls module clock gating. The procedure to turn module clocks on/off is described in Chapter 7. Furthermore, special consideration must be given to DSP clock on/off. The procedure to turn the DSP clock on/off is further described in Section 10.5.

8.3.2 Module Clock Frequency Scaling

Module clock frequency is scalable by programming the PLL's multiply and divide parameters. Reducing the clock frequency reduces the active switching power consumption linearly with frequency. It has no impact on leakage power consumption.

Chapter 5 and Chapter 6 describe the how to program the PLL frequency and the frequency constraints.

8.3.3 PLL Bypass and Power Down

You can bypass the PLLs in the DM644x DMSoC. Bypassing the PLLs sends the PLL reference clock to the post dividers of the PLLC instead of to the PLL VCO output. The PLL reference clock is typically at 27 MHz; therefore, you can use this mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity. Furthermore, you can power-down the PLL when bypassing it to save additional active power.

Chapter 5 and Chapter 6 describe PLL bypass and PLL power down.

8.4 ARM and DSP Sleep Mode Management

8.4.1 ARM Wait-For-Interrupt Sleep Mode

The ARM module cannot have its clock gated in the PSC module. However, the ARM includes a special sleep mode called “wait-for-interrupt”. When the wait-for-interrupt mode is enabled, the clock to the CPU core is shut off and the ARM9 is completely inactive and only resumes operation after receiving an interrupt. This mode does not affect leakage consumption.

You can enable the wait-for-interrupt mode via the CP15 register #7 using the following instruction:

- mcr p15, #0, rd, c7, c0, #4

The following sequence exemplifies how to enter wait-for-interrupt mode:

- Enable any interrupt (for example, an external interrupt).
- Enable wait-for-interrupt mode using the following CP15 instruction:
  - mcr p15, #0, rd, c7, c0, #4

The following sequence describes the procedure to wake up from the wait-for-interrupt mode:

- To wake up from the wait-for-interrupt mode, trigger any enabled interrupt (for example, an external interrupt).
- The ARM's PC jumps to the IRQ vector and you must handle the interrupt in an interrupt service routine (ISR).

Exit the ISR and continue normal program execution starting from the instruction immediately following the instruction that enabled wait-for-interrupt mode: mcr p15, #0, r3, c7, c0, #4.

NOTE: The ARM interrupt controller and the module sourcing the wakeup interrupt (for example, GPIO or watchdog timer) must not be disabled, or the device will never wake up.

For more information on this sleep mode, refer to the ARM926EJ-S Technical Reference Manual, which is available from ARM Ltd. at www.arm.com.

8.4.2 DSP Sleep Modes

The C64x+ megamodule of the DSP subsystem includes a power-down controller. The power-down controller can power-down all of the following components of the C64x+ megamodule and internal memories of the DSP subsystem:

- C64x+ CPU
- Program Memory Controller (PMC)
- Data Memory Controller (DMC)
- Unified Memory Controller (UMC)
- Extended Memory Controller (EMC)
- L1P Memory
- L1D Memory
- L2 Memory

Although the C64x+ megamodule documentation mentions both dynamic and static power-down, the DM644x DMSoC supports only static power-down.

- Static power-down: PDC initiates power-down of the entire C64x+ megamodule and all internal memories immediately upon command from software.

Static power-down affects all components of the C64x+ megamodule and all internal memories. Software can initiate static power-down via a register bit in the PDC register.

For more information on the DSP subsystem, see the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15).

8.5 I/O Management

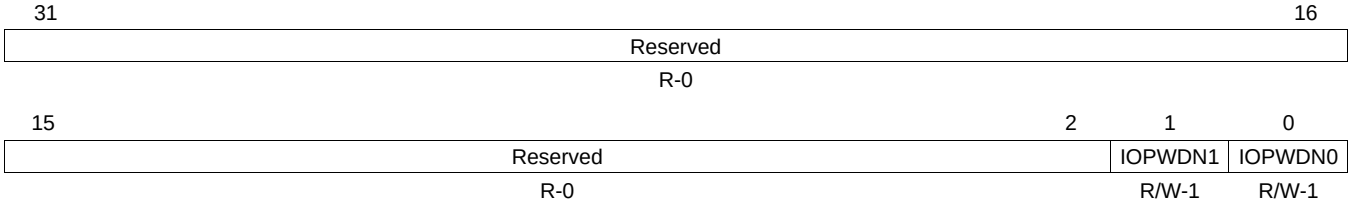
8.5.1 3.3 V I/O Power-Down

The 3.3 V I/O drivers are fabricated out of 1.8 V transistors with design techniques that require a DC bias current. These I/O cells have a power-down mode that turns off the DC current. There are two register bits in the VDD3P3V_PWDN register of the system control module to control this standby mode. One bit controls the I/Os for MMC/SD and another bit controls the I/Os for 3.3 V GPIO's. VDD3P3V_PWDN is described in Figure 8-1 and Table 8-2.

8.6 VDD3P3V_PWDN Register

The VDD3P3V_PWDN register is shown in Figure 8-1 and described in Table 8-2.

Figure 8-1. VDD3P3V_PWDN Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-2. VDD3P3V_PWDN Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	IOPWDN1		MMC/SD I/O powerdown controls SD_CLK, SD_CMD, SD_DATA[3:0] pins.
		0	I/O cells powered-up
		1	I/O cells powered-down
0	IOPWDN0		GPIOV33 I/O powerdown controls GPIOV33[16:0] pins
		0	I/O cells powered-up
		1	I/O cells powered-down

8.7 USB Phy Power Down

You can power-down the USB Phy peripheral when it is not in use. The USB Phy is powered-down via the PHYCLKGD bit in the USBPHY_CTL register of the system control module. USBPHY_CTL is described in Chapter 10.

8.8 Video DAC Power Down

The DM644x DMSoC video processing back end (VPBE) includes four video digital-to-analog converters (DACs) to drive analog television displays. The Video Encoder (VENC) module of the VPBE includes registers for enabling/disabling the DACs. You can use the VIE bit in VMOD to force the analog output of the 4 DACs to a low level, regardless of the video signal. Furthermore, you can use the DAPD[3:0] bits in DACTST to disable each DAC independently. See the *TMS320DM644x DMSoC Video Processing Back End (VPBE) User's Guide* (SPRUE37) for register descriptions and more detailed information on DAC power-down.

ARM Interrupt Controller

Topic **Page**

9.1 Introduction

The TMS320DM644x DMSoC ARM interrupt controller (AINTC) has the following features:

- Supports up to 64 interrupt channels (16 external channels)
- Interrupt mask for each channel
- Each interrupt channel is mappable to a Fast Interrupt Request (FIQ) or to an Interrupt Request (IRQ) type of interrupt.
- Hardware prioritization of simultaneous interrupts
- Configurable interrupt priority (2 levels of FIQ and 6 levels of IRQ)
- Configurable interrupt entry table (FIQ and IRQ priority table entry) to reduce interrupt processing time

The ARM core supports two interrupt types: FIQ and IRQ. See the ARM926EJ Technical Reference Manual for detailed information about the ARM's FIQ and IRQ interrupts. Each interrupt channel is mappable to an FIQ or to an IRQ type of interrupt, and each channel can be enabled or disabled. The AINTC supports user-configurable interrupt-priority and interrupt entry addresses. Entry addresses minimize the time spent jumping to interrupt service routines (ISRs). When an interrupt occurs, the corresponding highest priority ISR's address is stored in the AINTC's ENTRY register. The IRQ or FIQ interrupt routine can read the ENTRY register and jump to the corresponding ISR directly. Thus, the ARM does not require a software dispatcher to determine the asserted interrupt.

9.2 Interrupt Mapping

The AINTC takes up to 64 ARM device interrupts and maps them to either the IRQ or to the FIQ of the ARM. Each interrupt is also assigned one of 8 priority levels (2 for FIQ, 6 for IRQ). For interrupts with the same priority level, the priority is determined by the hardware interrupt number (the lowest number has the highest priority).

Table 9-1 shows the connection of device interrupts to the ARM.

9.3 AINTC Methodology

AINTC methodology is illustrated in Figure 9-1 and described below.

- When an interrupt occurs, the status is reflected in either the FIQn or the IRQn registers, depending upon the interrupt type selected.
- Interrupts are enabled or disabled (masked) by setting the EINTn register.

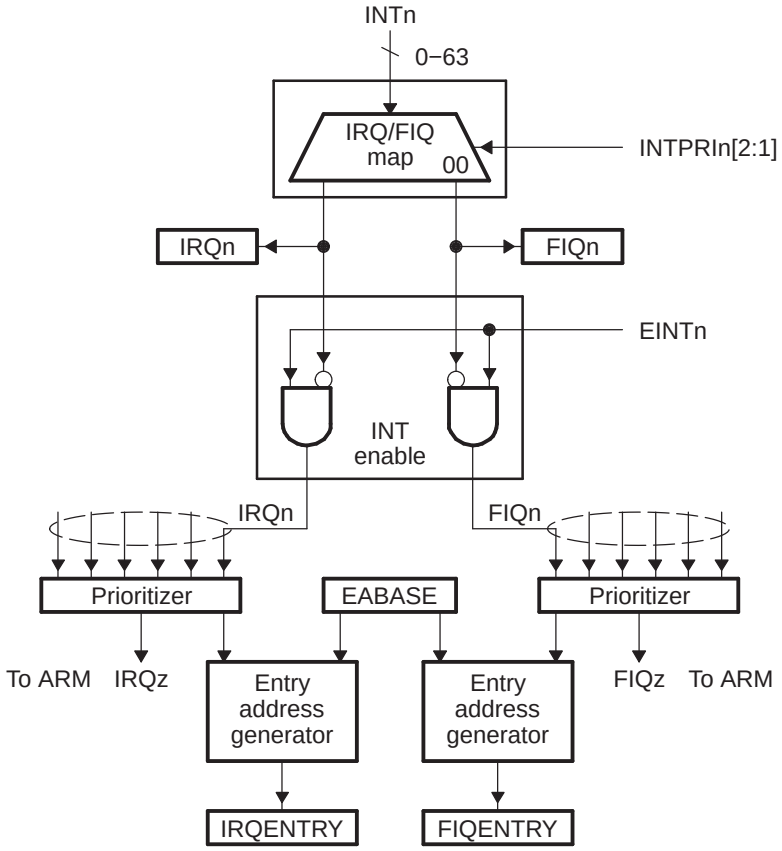
NOTE: Even if an interrupt is masked, the status interrupt is still reflected in the FIQn and the IRQn registers.

- When an interrupt from any interrupt channel occurs (for which interrupt is enabled), an IRQ or FIQ interrupt generates to the ARM926 core (depending on whether the interrupt channel is mapped to IRQ or FIQ interrupt). The ARM then branches to the IRQ or FIQ interrupt routine.
- The AINTC generates the entry address of the pending interrupt with the highest priority and stores the entry address in the FIQENTRY or the IRQENTRY register, depending on whether the interrupt is mapped to IRQ or FIQ interrupt. The IRQ or FIQ ISR can then read the entry address and its branch to the ISR of the interrupt.

Table 9-1. AINTC Interrupt Connections

Interrupt Number	Acronym	Source	Interrupt Number	Acronym	Source
0	VDINT0	VPSS - CCDC	32	TINT0	Timer 0 - TINT12
1	VDINT1	VPSS - CCDC	33	TINT1	Timer 0 - TINT34
2	VDINT2	VPSS - CCDC	34	TINT2	Timer 1 - TINT12
3	HISTINT	VPSS - Histogram	35	TINT3	Timer 1 - TINT34
4	H3AINT	VPSS - AE/AWB/AF	36	PWMINT0	PWM 0
5	PRVUINT	VPSS - Previewer	37	PWMINT1	PWM 1
6	RSZINT	VPSS - Resizer	38	PWMINT2	PWM 2
7	-	Reserved	39	IICINT	I2C
8	VENCINT	VPSS - VPBE	40	UARTINT0	UART0
9	ASQINT	VICP	41	UARTINT1	UART1
10	IMXINT	VICP	42	UARTINT2	UART2
11	VLCDINT	VICP	43	SPINT0	SPI
12	USBINT	USB	44	SPINT1	SPI
13	EMACINT	EMAC	45	-	Reserved
14	-	Reserved	46	DSP2ARM0	DSP Controller
15	-	Reserved	47	DSP2ARM1	DSP Controller
16	CCINT0	TPCC Region 0	48	GPIO0	GPIO
17	CCERRINT	TPCC Error	49	GPIO1	GPIO
18	TCERRINT0	TPTC0 Error	50	GPIO2	GPIO
19	TCERRINT	TPTC1 Error	51	GPIO3	GPIO
20	PSCINT	PSC - ALLINT	52	GPIO4	GPIO
21	-	Reserved	53	GPIO5	GPIO
22	IDEINT	ATA/CF	54	GPIO6	GPIO
23	-	Reserved	55	GPIO7	GPIO
24	MBXINT	ASP	56	GPIOBNK0	GPIO
25	MBRINT	ASP	57	GPIOBNK1	GPIO
26	MMCINT	MMC/SD	58	GPIOBNK2	GPIO
27	SDIOINT	MMC/SD	59	GPIOBNK3	GPIO
28	-	Reserved	60	GPIOBNK4	GPIO
29	DDRINT	DDR EMIF	61	COMMTX	ARMSS
30	AEMIFINT	Async EMIF	62	COMMRX	ARMSS
31	VLQINT	VLYNQ	63	EMUINT	E2ICE

Figure 9-1. AINTC Functional Diagram



9.3.1 Interrupt Mapping

Each event input is mapped to either the ARM IRQ or to the FIQ interrupt based on the priority level selected in the INTPRIn register. Events with a priority of 0 or 1 are designated as FIQs; events with priorities of 2-7 are designated as IRQs. The appropriate IRQ / FIQ registers capture interrupt events. Each event causes an IRQ or FIQ to generate only if the corresponding EINT bit enables it. The EINT bit enables or disables the event regardless of whether it is mapped to IRQ or to FIQ. The IRQ/FIQ register always captures each event, regardless of whether the interrupt is actually enabled.

9.3.2 Interrupt Prioritization

Event priority is determined using both a fixed and a programmable prioritization scheme. The AINTC has 8 different programmable interrupt priorities. Priority 0 and priority 1 are mapped to the FIQ interrupt with priority 0 being the highest priority. Priorities 2-7 are mapped to the IRQ interrupt (priority 2 is the highest, priority 7 is the lowest). Each interrupt is mapped to a priority level using the INTPRIn registers. When simultaneous events occur (multiple enabled events captured in IRQ or FIQ registers), the event with the highest priority is the one whose entry table address is generated when sending the interrupt signal to the ARM. When events of identical priority occur, the event with the lowest event number is treated as having the higher priority.

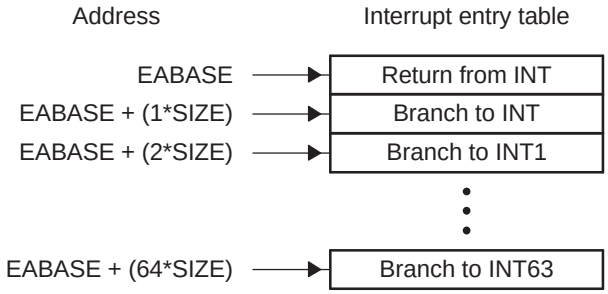
9.3.3 Vector Table Entry Address Generation

To help speed up the ISR, the AINTC provides two vectors into the ARM's interrupt entry table, which correspond to the highest priority effective IRQ and FIQ interrupts. This vector is generated by modifying a base address with a priority index. The priority index takes the size of each interrupt entry into account using the following formulas:

IRQENTRY = EABASE + ((highest priority IRQ EVT# + 1) × SIZE)
FIQENTRY = EABASE + ((highest priority FIQ EVT# + 1) × SIZE)

The EABASE base address is contained in a register. The SIZE value is a programmable register field, which selects 4, 8, 16, or 32 bytes for each interrupt table entry. The IRQENTRY or FIQENTRY register is read by the ARM, depending on which type of interrupt it is servicing. The ARM interrupt entry table format is shown in Figure 9-2.

Figure 9-2. Interrupt Entry Table



The highest priority effective IRQ or FIQ interrupt includes only those interrupts that are enabled by their corresponding EINT bit by default. However, the IERAW and FERAW register bits, if set, allow the highest priority event of any of those captured in the IRQ or FIQ register to be used in calculating IRQENTRY and FIQENTRY, respectively (regardless of the EINT state).

The IRQENTRY and FIQENTRY values are generated in real time as the interrupt events occur. Thus, their values may change from the time that the IRQ or FIQ is sent to the ARM to the time the ARM reads the register. They may also change immediately after a read by the ARM if a higher priority event occurs. If no IRQ mapped effective interrupt is pending, then the IRQENTRY value reflects the EABASE value. Similarly, if no FIQ mapped effective interrupt is pending, then the FIQENTRY value reflects the EABASE value.

1. For the FIQENTRY:
   - If FERAW is 0, FIQENTRY reflects the state of the highest priority pending enabled FIQ interrupt. If the active FIQ interrupt is cleared in FIQn, then FIQENTRY is immediately updated with the vector of the next highest priority pending enabled FIQ interrupt.
   - If FERAW is 1, FIQENTRY reflects the state of the highest priority pending FIQ interrupt (enabled or not). If the active FIQ interrupt is cleared in FIQn, then FIQENTRY is immediately updated with the vector of the next highest priority pending interrupt (enabled or not).
2. For the IRQENTRY:
   - If IERAW is 0, IRQENTRY reflects the state of the highest priority pending enabled IRQ interrupt. If the active IRQ interrupt is cleared in IRQn, then IRQENTRY is immediately updated with the vector of the next highest priority pending enabled IRQ interrupt.
   - If IERAW is 1, IRQENTRY reflects the state of the highest priority pending IRQ interrupt (enabled or not). If the active IRQ interrupt is cleared in IRQn, then IRQENTRY is immediately updated with the vector of the next highest priority pending IRQ interrupt (pending or not).

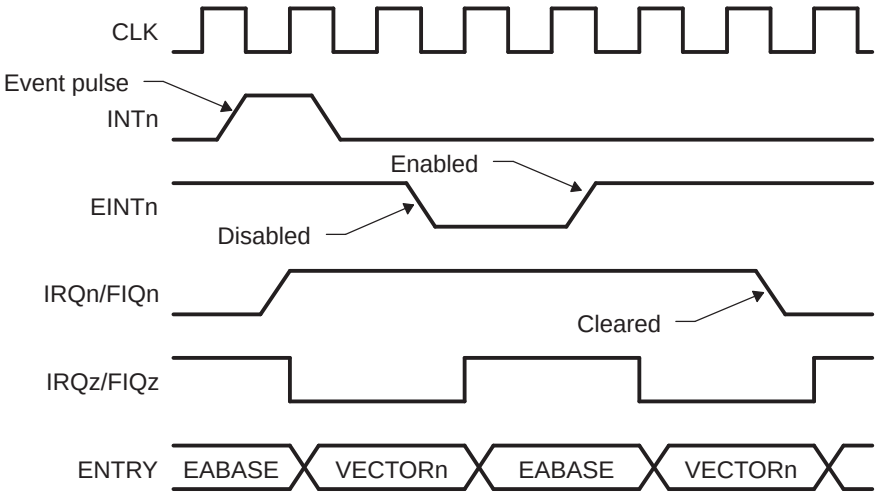
9.3.4 Clearing Interrupts

Events cause their matching bit in the FIQ or IRQ register (depending on the event priority) to be cleared to 0. An event is cleared by writing a 1 to the corresponding bit in the FIQ or IRQ register. Writing a 1 to the corresponding bit sets the bit back to a 1. Writing a 0 to an event bit does not affect its value.

9.3.5 Enabling and Disabling Interrupts

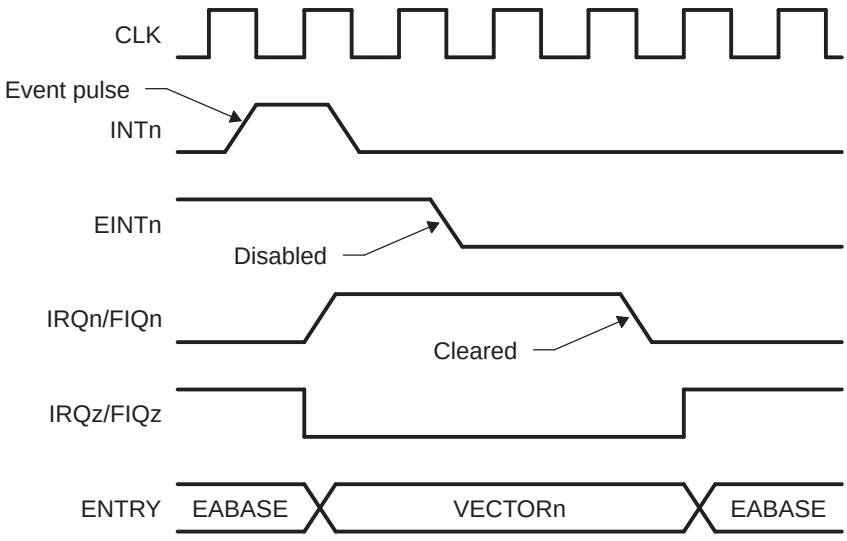
The AINTC has two methods for enabling and disabling interrupts: immediate or delayed, based on the setting of the IDMODE bit in the INTCTL register. When 0 (default), clearing an interrupt's EINT bit has an immediate effect. The prioritizer removes the disabled interrupt from consideration and adjusts the IRQ/FIQENTRY value correspondingly. If no other interrupts are pending, then the IRQz/FIQz output to the ARM may also go inactive. Enabling the interrupt if it is already pending takes immediate affect. This is shown in Figure 9-3.

Figure 9-3. Immediate Interrupt Disable/Enable



If IDMODE is 1, then the EINT effect is delayed. Essentially, the active interrupt status is latched until cleared by the ARM. If EINT is cleared, the prioritizer continues to use the interrupt and the IRQz/FIQz remains active. Once the ARM clears the pending interrupt, further interrupts are disabled. In the same way, setting EINT does not cause the previously pending interrupt event to become enabled until it has been cleared first. The disable operation is shown in Figure 9-4.

Figure 9-4. Delayed Interrupt Disable



9.4 AINTC Registers

Table 9-2 lists the memory-mapped registers for the AINTC.

Table 9-2. ARM Interrupt Controller (AINTC) Registers

Address	Acronym	Register Description	Section
1C4 8000h	FIQ0	Fast Interrupt Request Status Register 0	Section 9.4.1
1C4 8004h	FIQ1	Fast Interrupt Request Status Register 1	Section 9.4.2
1C4 8008h	IRQ0	Interrupt Request Status Register 0	Section 9.4.3
1C4 800Ch	IRQ1	Interrupt Request Status Register 1	Section 9.4.4
1C4 8010h	FIQENTRY	Fast Interrupt Request Entry Address Register	Section 9.4.5
1C4 8014h	IRQENTRY	Interrupt Request Entry Address Register	Section 9.4.6
1C4 8018h	EINT0	Interrupt Enable Register 0	Section 9.4.7
1C4 801Ch	EINT1	Interrupt Enable Register 1	Section 9.4.8
1C4 8020h	INTCTL	Interrupt Operation Control Register	Section 9.4.9
1C4 8024h	EABASE	Interrupt Entry Table Base Address Register	Section 9.4.10
1C4 8030h	INTPRI0	Interrupt 0-7 Priority Register 0	Section 9.4.11
1C4 8034h	INTPRI1	Interrupt 8-15 Priority Register 1	Section 9.4.12
1C4 8038h	INTPRI2	Interrupt 16-23 Priority Register 2	Section 9.4.13
1C4 803Ch	INTPRI3	Interrupt 24-31 Priority Register 3	Section 9.4.14
1C4 8040h	INTPRI4	Interrupt 32-39 Priority Register 4	Section 9.4.15
1C4 8044h	INTPRI5	Interrupt 40-47 Priority Register 5	Section 9.4.16
1C4 8048h	INTPRI6	Interrupt 48-55 Priority Register 6	Section 9.4.17
1C4 804Ch	INTPRI7	Interrupt 56-63 Priority Register 7	Section 9.4.18

9.4.1 Fast Interrupt Request Status Register 0 (FIQ0)

The fast interrupt request status register 0 (FIQ0) is shown in Figure 9-5 and described in Table 9-3. Interrupt status of INT[31:0] (if mapped to FIQ).

Figure 9-5. Fast Interrupt Request Status Register 0 (FIQ0)

31	16
FIQ[*n*]	
R/W-1	

15	0
FIQ[*n*]	
R/W-1	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 9-3. Fast Interrupt Request Status Register 0 (FIQ0) Field Descriptions

Bit	Field	Value	Description
31-0	FIQ[*n*]		Interrupt status of INT*n*, if mapped to fast interrupt request (FIQ31-0).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.

9.4.2 Fast Interrupt Request Status Register 1 (FIQ1)

The fast interrupt request status register 1 (FIQ1) is shown in Figure 9-6 and described in Table 9-4. Interrupt status of INT[63:32] (if mapped to FIQ).

Figure 9-6. Fast Interrupt Request Status Register 1 (FIQ1)

31	16
Reserved	
R-1	

15	0
FIQ[*n*]	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-4. Fast Interrupt Request Status Register 1 (FIQ1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	1	Reserved
15-0	FIQ[*n*]		Interrupt status of INT*n*, if mapped to fast interrupt request (FIQ47-32).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.

9.4.3 Interrupt Request Status Register 0 (IRQ0)

The interrupt request status register 0 (IRQ0) is shown in Figure 9-7 and described in Table 9-5. Interrupt status of INT[31:0] (if mapped to IRQ).

Figure 9-7. Interrupt Request Status Register 0 (IRQ0)

31	16
IRQ[*n*]	
R/W-1	

15	0
IRQ[*n*]	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-5. Interrupt Request Status Register 0 (IRQ0) Field Descriptions

Bit	Field	Value	Description
31-0	IRQ[*n*]		Interrupt status of INT*n*, if mapped to interrupt request (IRQ31-0).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.

9.4.4 Interrupt Request Status Register 1 (IRQ1)

The interrupt request status register 1 (IRQ1) is shown in Figure 9-8 and described in Table 9-6. Interrupt status of INT[63:32] (if mapped to IRQ).

Figure 9-8. Interrupt Request Status Register 1 (IRQ1)

31	16
IRQ[*n*]	
R/W-1	

15	0
IRQ[*n*]	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-6. Interrupt Request Status Register 1 (IRQ1) Field Descriptions

Bit	Field	Value	Description
31-0	IRQ[*n*]		Interrupt status of INT*n*, if mapped to interrupt request (IRQ63-32).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.

9.4.5 Fast Interrupt Request Entry Address Register (FIQENTRY)

The fast interrupt request entry address register (FIQENTRY) is shown in Figure 9-9 and described in Table 9-7. Entry address [28:0] for valid FIQ interrupt.

Figure 9-9. Fast Interrupt Request Entry Address Register (FIQENTRY)

31	29	28	16
Reserved		FIQENTRY	
R-0		R-0	

15	0
FIQENTRY	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 9-7. Fast Interrupt Request Entry Address Register (FIQENTRY) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-0	FIQENTRY	0-1FFF FFFFh	Interrupt entry table address of the current highest-priority fast interrupt request (FIQ).

9.4.6 Interrupt Request Entry Address Register (IRQENTRY)

The interrupt request entry address register (IRQENTRY) is shown in Figure 9-10 and described in Table 9-8. Entry address [28:0] for valid IRQ interrupt.

Figure 9-10. Interrupt Request Entry Address Register (IRQENTRY)

31	29	28	16
Reserved		IRQENTRY	
R-0		R-0	

15	0
IRQENTRY	
R-0	

LEGEND: R = Read only; -*n* = value after reset

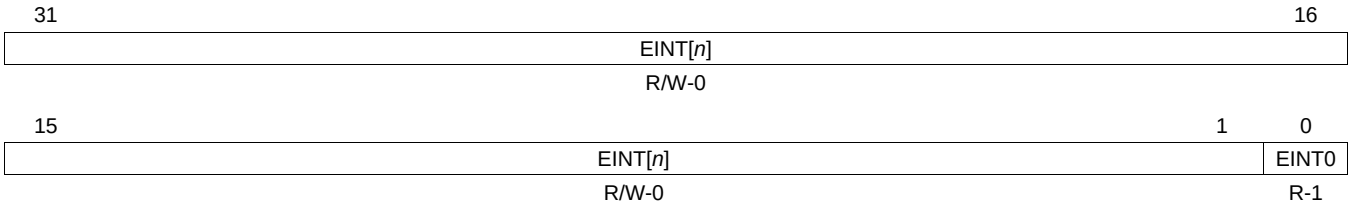
Table 9-8. Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-0	IRQENTRY	0-1FFF FFFFh	Interrupt entry table address of the current highest-priority interrupt request (IRQ).

9.4.7 Interrupt Enable Register 0 (EINT0)

The interrupt enable register 0 (EINT0) is shown in Figure 9-11 and described in Table 9-9.

Figure 9-11. Interrupt Enable Register 0 (EINT0)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

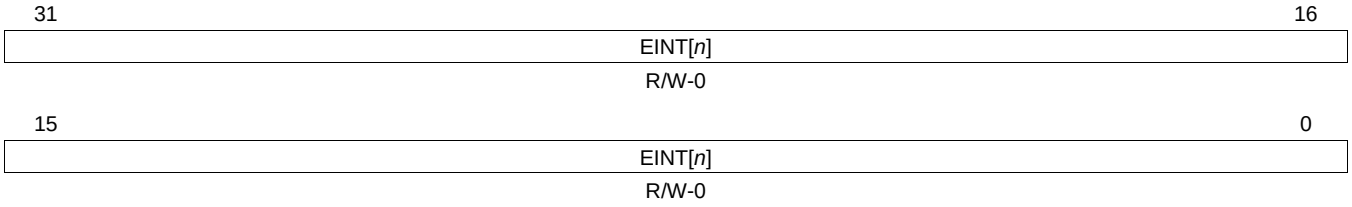
Table 9-9. Interrupt Enable Register 0 (EINT0) Field Descriptions

Bit	Field	Value	Description
31-1	EINT[*n*]		Interrupt enable for INT*n*. Bits 1 through 31 represent interrupts 1-31, respectively.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
0	EINT0	1	Interrupt 0 is nonmaskable and is always enabled.

9.4.8 Interrupt Enable Register 1 (EINT1)

The interrupt enable register 1 (EINT1) is shown in Figure 9-12 and described in Table 9-10.

Figure 9-12. Interrupt Enable Register 1 (EINT1)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

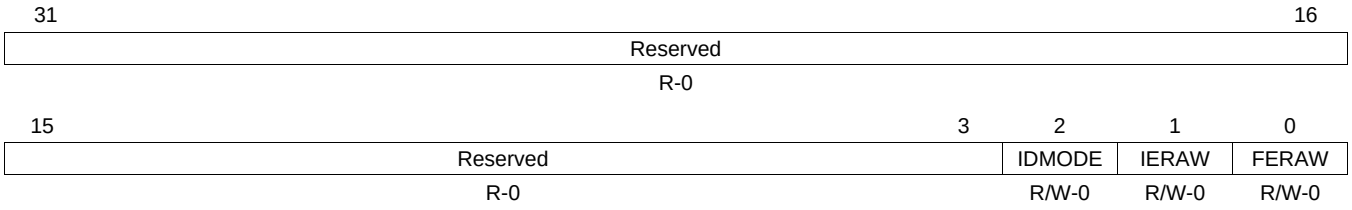
Table 9-10. Interrupt Enable Register 1 (EINT1) Field Descriptions

Bit	Field	Value	Description
31-0	EINT[*n*]		Interrupt enable for INT*n*. Bits 0 through 31 represent interrupts 32-63, respectively.
		0	Interrupt is disabled.
		1	Interrupt is enabled.

9.4.9 Interrupt Operation Control Register (INTCTL)

The interrupt operation control register (INTCTL) is shown in Figure 9-13 and described in Table 9-11.

Figure 9-13. Interrupt Operation Control Register (INTCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-11. Interrupt Operation Control Register (INTCTL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IDMODE		Interrupt disable mode.
		0	Disable immediately.
		1	Disable after acknowledgement.
1	IERAW		Masked interrupt reflected in the interrupt request entry address register (IRQENTRY).
		0	Disable reflect.
		1	Enable reflect.
0	FERAW		Masked interrupt reflect in the fast interrupt request entry address register (FIQENTRY).
		0	Disable reflect.
		1	Enable reflect.

9.4.10 Interrupt Entry Table Base Address Register (EABASE)

The interrupt entry table base address register (EABASE) is shown in Figure 9-14 and described in Table 9-12.

Figure 9-14. Interrupt Entry Table Base Address Register (EABASE)

31	29	28	16
Reserved		EABASE	
R-0		R/W-0	

15	3	2	1	0
EABASE		Reserved	SIZE	
R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-12. Interrupt Entry Table Base Address Register (EABASE) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-3	EABASE	0-3FF FFFFh	Interrupt entry table base address (8-byte aligned).
2	Reserved	0	Reserved
1-0	SIZE	0-3h	Size of each entry in the interrupt entry table.
		0	4 bytes
		1h	8 bytes
		2h	16 bytes
		3h	32 bytes

9.4.11 Interrupt Priority Register 0 (INTPRI0)

The interrupt priority register 0 (INTPRI0) is shown in Figure 9-15 and described in Table 9-13.

Figure 9-15. Interrupt Priority Register 0 (INTPRI0)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT7	Reserved	INT6	Reserved	INT5	Reserved	INT4
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT3	Reserved	INT2	Reserved	INT1	Reserved	INT0
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-13. Interrupt Priority Register 0 (INTPRI0) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.12 Interrupt Priority Register 1 (INTPRI1)

The interrupt priority register 1 (INTPRI1) is shown in Figure 9-16 and described in Table 9-14.

Figure 9-16. Interrupt Priority Register 1 (INTPRI1)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT15	Reserved	INT14	Reserved	INT13	Reserved	INT12
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT11	Reserved	INT10	Reserved	INT9	Reserved	INT8
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-14. Interrupt Priority Register 1 (INTPRI1) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.13 Interrupt Priority Register 2 (INTPRI2)

The interrupt priority register 2 (INTPRI2) is shown in Figure 9-17 and described in Table 9-15.

Figure 9-17. Interrupt Priority Register 2 (INTPRI2)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT23	Reserved	INT22	Reserved	INT21	Reserved	INT20
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT19	Reserved	INT18	Reserved	INT17	Reserved	INT16
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-15. Interrupt Priority Register 2 (INTPRI2) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.14 Interrupt Priority Register 3 (INTPRI3)

The interrupt priority register 3 (INTPRI3) is shown in Figure 9-18 and described in Table 9-16.

Figure 9-18. Interrupt Priority Register 3 (INTPRI3)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT31	Reserved	INT30	Reserved	INT29	Reserved	INT28
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT27	Reserved	INT26	Reserved	INT25	Reserved	INT24
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-16. Interrupt Priority Register 3 (INTPRI3) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.15 Interrupt Priority Register 4 (INTPRI4)

The interrupt priority register 4 (INTPRI4) is shown in Figure 9-19 and described in Table 9-17.

Figure 9-19. Interrupt Priority Register 4 (INTPRI4)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT39	Reserved	INT38	Reserved	INT37	Reserved	INT36
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT35	Reserved	INT34	Reserved	INT33	Reserved	INT32
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-17. Interrupt Priority Register 4 (INTPRI4) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.16 Interrupt Priority Register 5 (INTPRI5)

The interrupt priority register 5 (INTPRI5) is shown in Figure 9-20 and described in Table 9-18.

Figure 9-20. Interrupt Priority Register 5 (INTPRI5)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT47	Reserved	INT46	Reserved	INT45	Reserved	INT44
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT43	Reserved	INT42	Reserved	INT41	Reserved	INT40
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-18. Interrupt Priority Register 5 (INTPRI5) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.17 Interrupt Priority Register 6 (INTPRI6)

The interrupt priority register 6 (INTPRI6) is shown in Figure 9-21 and described in Table 9-19.

Figure 9-21. Interrupt Priority Register 6 (INTPRI6)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT55	Reserved	INT54	Reserved	INT53	Reserved	INT52
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT51	Reserved	INT50	Reserved	INT49	Reserved	INT48
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-19. Interrupt Priority Register 6 (INTPRI6) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

9.4.18 Interrupt Priority Register 7 (INTPRI7)

The interrupt priority register 7 (INTPRI7) is shown in Figure 9-22 and described in Table 9-20.

Figure 9-22. Interrupt Priority Register 7 (INTPRI7)

31	30 28	27	26 24	23	22 20	19	18 16
Reserved	INT63	Reserved	INT62	Reserved	INT61	Reserved	INT60
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

15	14 12	11	10 8	7	6 4	3	2 0
Reserved	INT59	Reserved	INT58	Reserved	INT57	Reserved	INT56
R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h	R-0	R/W-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-20. Interrupt Priority Register 7 (INTPRI7) Field Descriptions

Bit	Field	Value	Description
	Reserved	0	Reserved
	INT*n*	0-7h	Selects INT*n* priority level.

System Control Module

Topic Page

10.1 Overview of the System Control Module

The TMS320DM644x DMSoC system control module is a system-level module containing status and top-level control logic required by the device. The system control module consists of a set of status and control registers, accessible by the ARM (and DSP), supporting all of the following system features and operations:

- Device Identification
- Device Configuration
  - Pin multiplexing control
  - Device boot configuration status
- ARM-DSP Integration
  - ARM-DSP interrupt control and status
  - DSP boot address control and status
  - Chip power shorting switch control
- Power Management
  - V_{DD} 1.0 V/1.2 V adjustment status
  - V_{DD} 3.3 V I/O power-down control
- Special Peripheral Status and Control
  - USB PHY control
  - VPSS clock and DAC control
  - DDR I/O timing control and status
  - VLYNQ clock tuning control
- Bandwidth Management
  - Bus master DMA priority control
- Emulation Control
  - Set emulator suspend source
- Device Unique ID
  - 128-bit ID, unique to each device, suitable for digital rights management (DRM) implementation

This chapter describes the system control module.

10.2 Device Identification

The JTAG ID register (JTAGID) of the System Control Module contains a software readable version of the JTAG/Device ID. Software can use this register to determine the version of the device on which it is executing. The register format and description are shown in the device-specific data manual.

10.3 Device Configuration

The system control module contains registers for controlling pin multiplexing and registers that reflect the boot configuration status.

10.3.1 Pin Multiplexing Control

The DM644x DMSoC makes extensive use of pin multiplexing to accommodate the large number of peripheral functions in the smallest possible package. A combination of hardware configuration (at device reset) and program control controls pin multiplexing to accomplish this. Hardware does not attempt to ensure that the proper pin multiplexing is selected for the peripherals or that interface mode is being used.

Detailed information about the pin multiplexing and control is covered in the device-specific data manual.

10.4 Device Boot Configuration Status

The device boot configuration (the state of the BTSEL[1:0], AEAW[4:0], and EM_WIDTH signals are captured in the BOOTCFG register). See the device-specific data manual for details on this register.

10.5 ARM-DSP Integration

10.5.1 ARM-DSP Interrupt Control and Status

The system module includes a register for generating interrupts between the ARM and DSP. The INTGEN register format is shown in the device specific data manual. The ARM may generate an interrupt to the DSP by setting one of the four INTDSP[3:0] bits or the INTNMI bit. The interrupt set bit then self-clears and the corresponding DSP[3:0]STAT or NMISTAT bit automatically sets to indicate that the interrupt is generated. After servicing the interrupt, the DSP clears the status bit by writing 0. The ARM may poll the status bit to determine when the DSP has completed the interrupt service. The DSP may generate the interrupt to the ARM in the same manner using the INTARM[1:0] bits. See Chapter 13 for more detailed information.

10.5.2 DSP Boot Address Control and Status

The DSPBOOTADDR register contains the DSP reset. See the device-specific data manual for details on this register. The boot address defaults to 4220:0000h (AEMIF CS2 space) to allow DSP self-boot on power-up (selected by the DSP_BT pin), but may be changed by the ARM for ARM-controlled booting.

For detailed information on booting the DMSoC, see Chapter 12.

10.5.3 Chip Power Shorting Switch Control

The CHP_SHRTSW register controls the shorting switch between the device always-on and DSP power domains. This switch should be enabled after powering-up the DSP domain. Setting the DSPPWRON bit to 1 closes the switch. The default switch value is determined by the DSP_BT configuration input. If DSP self-boot is selected (DSP_BT = 1), then the DSP powers-up and DSPPWRON defaults to 1. For ARM boot operation (DSP_BT = 0), DSPPWRON defaults to 0 and must be set by the ARM after DSP domain power is turned on. See the device-specific data manual for details on this register.

10.6 Power Management

10.6.1 DV_{DD} 3.3 V I/O Power-Down Control

The VDD3P3V_PWDN register controls power to the 3.3 V I/O cells. The 3.3 V I/Os are separated into two groups for independent control. See Section 8.6 and the device-specific data manual for details on this register.

10.7 Special Peripheral Status and Control

Several of the DM644x DMSoC peripheral modules require additional system-level control logic. Those registers are discussed in detail in this section.

10.7.1 USB PHY Control

The USBPHY_CTL register controls various features of the USB PHY. See the device-specific data manual for details on this register.

10.7.2 VPSS Clock and DAC Control

Clocks for the video processing subsystem are controlled via the VPSS clock mux control register (VPSS_CLKCTL). See the device-specific data manual for details on this register.

10.8 Bandwidth Management

10.8.1 Bus Master DMA Priority Control

In order to determine allowed connections between masters and slaves, each master request source must have a unique master ID (mstid) associated with it. The master ID for each DM644x DMSoC master is shown in Table 10-1.

Table 10-1. TMS320DM644x DMSoC Master IDs

MSTID	Master
0	ARM Program
1	ARM Data
2	DSP Program / Data
3	DSP CFG
4-7	Reserved
8	VPSS
9	VICP
10	EDMA
11-15	Reserved
16	EDMA Channel 0 read
17	EDMA Channel 0 write
18	EDMA Channel 1 read
19	EDMA Channel 1 write
20-31	Reserved
32	EMAC
33	Reserved
34	USB
35	ATA
36	VLYNQ
37	HPI
38-63	Reserved

Prioritization within each switched central resource (SCR) is selected to be either fixed or dynamic. Dynamic prioritization is based on an incoming priority signal from each master. On the DM644x DMSoC, only the DSP, VPSS, and EDMA masters actually generate priority values. For all other masters, the value is programmed in the chip-level MSTRPRI registers. The default priority level for each DM644x DMSoC bus master is shown in Table 10-2. Application software is expected to modify these values to obtain the desired system performance.

Table 10-2. TMS320DM644x DMSoC Default Master Priorities

Master	Default Priority
VPSS	0(1)
EDMA Ch 0	0(2)
EDMA Ch 1	0(2)
ARM	1
ARM (CFG)	1
DSP	7(3)
DSP (CFG)	1
EMAC	4
USB	4
ATA	4
HPI	4
VLYNQ	4
VICP	5

(1) Default value in VPSS PCR register
(2) Default value in EDMA QUEPRI register
(3) Default value in DSP MDMAARBE.PRI field

10.8.2 Emulation Control

10.8.2.1 Set Emulator Suspend Source

The flexibility of the DM644x DMSoC architecture allows either the ARM or the DSP to control some various peripherals (setup registers, service interrupts, etc.). While this assignment is purely a matter of software convention, during an emulation halt, the device must know which peripherals are associated with the halting processor, so that only those modules receive the suspend signal. This allows peripherals associated with the other (unhalted) processor to continue normal operation. The SUSPSRC register indicates the emulation suspend source for those peripherals which support emulation suspend.

When the associated SUSPSRC bit is 0, the ARM emulator controls the peripheral's emulation suspend signal and when it is set to 1, the DSP emulator controls the peripheral's emulation suspend signal. See the device-specific data manual for details on this register.

10.9 System Control Register Descriptions

Table 10-3 lists the memory-mapped registers for the system control register. See the device-specific data manual for complete descriptions.

Table 10-3. System Control Registers

Address	Acronym	Register Description
1C4 0000h	PINMUX0	Pin multiplexing control register 0
1C4 0004h	PINMUX1	Pin multiplexing control register 1
1C4 0008h	DSPBOOTADDR	DSP boot address register
1C4 000Ch	SUSPSRC	Emulator suspend source register
1C4 0010h	INTGEN	ARM/DSP interrupt status and control register
1C4 0014h	BOOTCFG	Device boot configuration register
1C4 0028h	JTAGID	JTAG/Device ID register
1C4 0030h	HPI_CTL	HPI control register
1C4 0034h	USBPHY_CTL	USB PHY control register
1C4 0038h	CHP_SHRTSW	Chip shorting switch control register
1C4 003Ch	MSTPRI0	Bus master priority control register
1C4 0040h	MSTPRI1	Bus master priority control register
1C4 0044h	VPSS_CLKCTL	VPSS clock multiplexing control register
1C4 0048h	VDD3P3V_PWDN	VDD 3.3V I/O power-down control register

Reset

Topic Page

11.1 Reset Overview

There are six types of reset in the TMS320DM644x DMSoC. The types of reset differ by how they are initiated and/or by their effect on the chip. Each type is briefly described in Table 11-1 and further described in the following sections.

Table 11-1. Reset Types

Type	Initiator	Effect
POR (Power-On-Reset)	RESETN pin low and TRSTN low	Total reset of the chip (cold reset). Resets all modules including memory and emulation.
Warm Reset	RESETN pin low and TRSTN high (expected to be the ARM emulator).	Resets all modules including memory, except ARM emulation. DSP emulation is reset.
Maximum (max) Reset	ARM emulator or Watchdog Timer (WDT).	Same effect as warm reset.
System Reset	DSP emulator	A soft reset. A soft reset maintains memory contents, and does not affect or reset clocks or power states.
Module Reset	ARM software	Resets a specific module. Allows the ARM to independently reset any module. Module reset is intended as a debug tool, not necessarily as a tool to use in production.
DSP Local Reset	ARM software	Resets the DSP CPU. DSP internal memories (L1P, L1D, and L2) are not reset. Allows the ARM to reset and boot the DSP.

11.2 Reset Pins

Power-On-Reset (POR) and warm reset are initiated by the RESETN and TRSTN pins. The RESETN and TRSTN pins are briefly described in Table 11-2.

For more information, see the device-specific data manual.

Table 11-2. Reset Pins

Pin Name	Type [Input/Output]	Description
RESETN	Input	Active low global reset input pin
TRSTN	Input	JTAG test-port reset

11.3 Types of Reset

11.3.1 Power-On Reset (POR)

POR totally resets the chip, including all modules, memories, and emulation circuitry.

The following steps describe the POR sequence:

1. Apply power and clocks to the chip and drive TRSTN and RESETN low to initiate POR.
2. Drive RESETN high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 11.4.1 for more information.
4. Hardware resets all of the modules, including memory and emulation circuitry.
5. POR finishes, all modules are now in their default configurations, and hardware begins the boot process.

See the device-specific data manual for power sequencing and reset timing requirements.

11.3.2 Warm Reset

Warm reset is like POR, except the ARM emulation circuitry is not reset. Warm reset allows an ARM emulator to initiate chip reset using TRSTN and RESETN while remaining active during and after the reset sequence.

NOTE: The DSP emulator will not remain alive through a warm reset.

The following steps describe the warm reset sequence:

1. Emulator drives TRSTN high and RESETN low to initiate warm reset.
2. Emulator drives RESETN high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 11.4.1 for more information.
4. Hardware resets all of the modules including memories, but not ARM emulation circuitry.
5. Warm reset finishes, all modules except ARM emulation are in their default configurations, and hardware begins the boot process.

See the device-specific data manual for reset timing requirements.

11.3.3 Maximum Reset

Maximum (max) reset is like warm reset, except maximum reset is initiated by the Watchdog Timer (WDT) or by an emulation command. For debug, max reset allows an ARM emulator to initiate chip reset using an emulation command while remaining active during and after the reset sequence.

NOTE: The DSP emulator will not remain alive through a max reset.

The following steps describe the max reset sequence:

1. To initiate max reset, the WDT expires (indicating a runaway condition), or the ARM emulator initiates a max reset command via the IcePick emulation module.
2. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 11.4.1 for more information.
3. Hardware resets all modules including memories, but not ARM emulation circuitry.
4. Warm reset finishes, all modules except ARM emulation are in their default configurations, and hardware begins the boot process.

NOTE: Max reset may be blocked by an emulator command. This allows an emulator to block a WDT initiated max reset for debug purposes.

See the *TMS320DM644x DMSoC 64-Bit Timer User's Guide* (SPRUE26) for information on the watchdog timer. See Chapter 3 for information on emulation.

11.3.4 System Reset

The emulator initiates system reset via special DSP emulation or ICECrusher. It is considered a soft reset (that is, memory is not reset). None of the following modules are reset: DDR EMIF, PLL Controller (PLLC), Power and Sleep Controller (PSC), and emulation.

The following steps describe the system reset sequence:

1. The emulator initiates system reset.
2. The proper modules are reset.
3. The system reset finishes, the proper modules are reset, and the CPU is out of reset.

11.3.5 Module Reset

Module reset allows you to independently reset a module using the ARM software. You can use module reset to return a module to its default state (that is, its state as seen after POR, warm reset, and max reset). Module reset is intended as a debug tool; it is not necessarily intended as a tool for use in production.

The procedures for asserting and de-asserting module reset are fully described in Chapter 7. Furthermore, special considerations for DSP module reset are described in Section 10.5.

11.3.6 DSP Local Reset

You can use a special local reset to reset the DSP CPU. When DSP local reset is asserted, the DSP's internal memories (L1P, L1D, and L2) are still accessible. Unlike module reset, local reset only resets the DSP CPU. The ARM uses local reset to reset the DSP during the DSP boot process.

NOTE: Module reset supersedes local reset, so you can execute a module reset when local reset is asserted or de-asserted.

The procedures for asserting and de-asserting DSP local reset are fully described in the Section 10.5 and Chapter 7.

11.4 Default Device Configurations

After POR, warm reset, and max reset, the chip is in its default configuration. This section highlights the default configurations associated with PLLs, clocks, ARM boot mode, AEMIF, and DSP boot mode.

NOTE: Default configuration is the configuration before the boot process begins. The boot ROM updates the configuration. See Chapter 12 for more information on the boot process.

11.4.1 Device Configuration Pins

The device configuration pins are described in Table 11-3. The device configuration pins are latched at reset and allow you to configure all of the following options at reset:

- ARM Boot Mode
- Data bus width for Asynchronous EMIF (CS2 2 region)
- Address bus width for Asynchronous EMIF
- DSP Boot Mode

These pins are described further in the following sections.

NOTE: The device configuration pins are multiplexed with pins of the Video Processing Back End (VPBE). After the device configuration pins are lasted at reset, they automatically change to function as VPBE pins. Pin multiplexing is described in Chapter 10.

Table 11-3. Device Configuration

Device Configuration Input	Function	Sampled Pin	Default Setting (by internal pull-up/ pull-down)	Device Configuration Affected
BTSEL[1:0]	Selects ARM boot mode : 00 = Boot from ROM (NAND/SPI) 01 = Boot from AEMIF 10 = Reserved 11 = Boot from ROM (UART)	COUT[1:0]	00 - NAND/SPI	If UART boot is selected or if primary boot selection fails and UART boot is executed, sets: SYS.PINMUX1.UART0 If NAND boot is selected, configures AEMIF CS2 to NAND interface.
EM_WIDTH	Selects AEMIF CS2 bus width 0 = 8-bit 1 = 16-bit	COUT2	0 (8-bit)	Sets AEMIF CS2 data bus width.

NOTE: Affects both AEMIF and NAND/SPI boot modes

AEAW[4:0]	AEMIF address bus width	YOUT[4:0]	00000 (All GPIO)	SYS.PINMUX0.AEAW
DSP_BT	DSP self-boot 0 = ARM boots DSP 1 = DSP self-boots	COUT3	0 (ARM Boot)	Powers up DSP and boots from AEMIF CS2.

NOTE: Incompatible with NAND/SPI boot option since DSP needs direct read access to AEMIF CS2 address space.

11.4.2 PLL and Clock Configuration

After POR, warm reset, and max reset, the PLLs and clocks are set to their default configurations.

The PLLs are in bypass mode and disabled by default. This means that the input reference clock at MXI (typically 27 MHz) drives the chip after reset. For more information, see Chapter 5 and Chapter 6. The default state of the PLLs is reflected by the default state of the register bits in the PLLC registers.

Only a subset of module clocks are enabled after reset by default. Table 7-1 shows which modules are enabled after reset. As shown in Table 7-1, the following modules are enabled, depending on the sampled state of the device configuration pins: EMDA, AEMIF, UART0, and DSP. For example, UART0 is enabled after reset when the device configuration pins (BTSEL[1:0] = 11, enable UART) select UART boot mode.

11.4.3 ARM Boot Mode Configuration

The BTSEL[1:0] inputs determine whether the ARM will boot from its ROM or from the Asynchronous EMIF (AEMIF). When ROM boot is selected (BTSEL[1:0] = 00, 10, or 11), a jump to the internal ROM (0000:4000h) is forced into the first fetched instruction word. The embedded ROM boot loader code (RBL) then performs certain configuration steps, reads the BOOTCFG register to determine the desired boot method, and branches to the appropriate boot function (that is, a NAND Flash loader utility or UART loader utility).

If AEMIF boot is selected (BTSEL[1:0] = 01), a jump to the lowest AEMIF address (0200:0000h) is forced into the first fetched instruction word. The ARM then continues executing from external memory using the default AEMIF timings until modified by software.

NOTE: Either NOR Flash or ROM must be connected to the first AEMIF chip select space (EM_CS2). The AEMIF does not support direct execution from NAND Flash.

Boot modes are further described in Chapter 12.

11.4.4 AEMIF Configuration

11.4.4.1 CE0 Bus Width Configuration

The EM_WIDTH input determines the default width of the first AEMIF chip select space (EM_CS2). If EM_WIDTH = 0, the space defaults to 8-bits wide. If EM_WIDTH = 1, it defaults to 16-bits wide. This allows the ARM to make full use of the width of the attached memory device if booting from AEMIF or NAND.

NOTE: EM_WIDTH only selects the default width and needs to be set depending on whether 8-bit or 16-bit AEMIF memory or NAND is used at boot time. After boot, the width of EM_CS2 can be changed.

The EM_WIDTH input affects only the first chip select space (EM_CS2). All other chip select spaces default to 8-bits wide and must be modified using the appropriate AEMIF control register if 16-bit operation is desired.

See the *TMS320DM644x DMSoC Asynchronous External Memory Interface (EMIF) User's Guide* (SPRUE20) for more information on the AEMIF.

11.4.4.2 AEMIF Address Width Configuration

The DM644x DMSoC pin multiplexing control logic allows all but three of the Asynchronous EMIF address pins to be used as GPIOs. If devices (such as NAND Flash) attached to the AEMIF require less than the 22 address pins provided, then you can configure the unused upper-order addresses as GPIOs. You must configure these pins at reset so that pins that the AEMIF drives with addresses do not cause bus contention with pins that the system drives as general purpose inputs.

By default, all address lines are multiplexed as GPIOs, except for EM_A[1] and EM_A[2]. These may be used as ALE and CLE signals for NAND Flash control if they are booting from internal ROM (NAND boot modes). If they are booting from NOR Flash, then the AEAW[4:0] inputs must enable the appropriate number of address outputs at reset. (See Chapter 10 for AEAW[4:0] encoding.)

11.4.4.3 AEMIF Timing Configuration

When AEMIF is enabled, the wait state registers are reset to the slowest possible configuration, which is 88 cycles per access (16 cycles of setup, 64 cycles of strobe, and 8 cycles of hold). Thus, with a 27 MHz clock at MXI, the AEMIF is configured to run at 4.5 MHz/88 which equals approximately 51 kHz by default. See the *TMS320DM644x DMSoC Asynchronous External Memory Interface (EMIF) User's Guide* (SPRUE20) for more information on the AEMIF.

11.4.5 DSP Boot Mode Configuration

The DSP_BT input determines DSP operation at reset. For most applications, the ARM is the master device and controls the reset and boot of the DSP (DSP_BT = 0). Under this scenario, the DSP remains powered-off after reset and the ARM is responsible for enabling power to DSP, ensuring that a valid DSP program image is available in program memory accessible by the DSP (DSP memory, AEMIF, DDR2), configuring the DSP boot address (DSPBOOTADR bit in SYS), and for releasing the DSP from reset.

When DSP_BT = 1, the DSP boots itself. Under this scenario, the DSP power domain is turned on and the DSP is released from reset without ARM intervention. The DSP boot address is set to an AEMIF address executable by the DSP (4220:0000h), which corresponds to 0220:0000h in ARM's EM_CS2 address map. Then, DSP begins execution with instruction (L1P) cache enabled.

For more information on the DSP Power Domain, see Chapter 7, Chapter 8, and Section 10.5.

Boot Modes

Topic Page

12.1 Boot Modes Overview

The TMS320DM644x DMSoC ARM can boot from either asynchronous EMIF/NOR Flash or from ARM ROM, as determined by the setting of the device configuration pins BTSEL[1:0]. The BTSEL[1:0] pins can define the ROM boot mode further as well. These ROM boot modes are described in the following sections. For a more detailed description of the ROM boot modes, see *Basic Application Loading Over the Serial Interface for the DaVinci DM644x* (SPRAAI0) and *Booting and Flashing via the DaVinci TMS320DM644x Serial Interface* (SPRAAI4).

The boot selection pins (BTSEL[1:0]) determine the ARM boot process. After reset (POR, warm reset, or max reset), ARM program execution begins in ARM ROM at 0000:4000h, except in the case of a GP (General Purpose) device and BTSEL[1:0] = 01, indicating a NOR Boot. See Chapter 11 for information on the boot selection pins.

12.1.1 Features

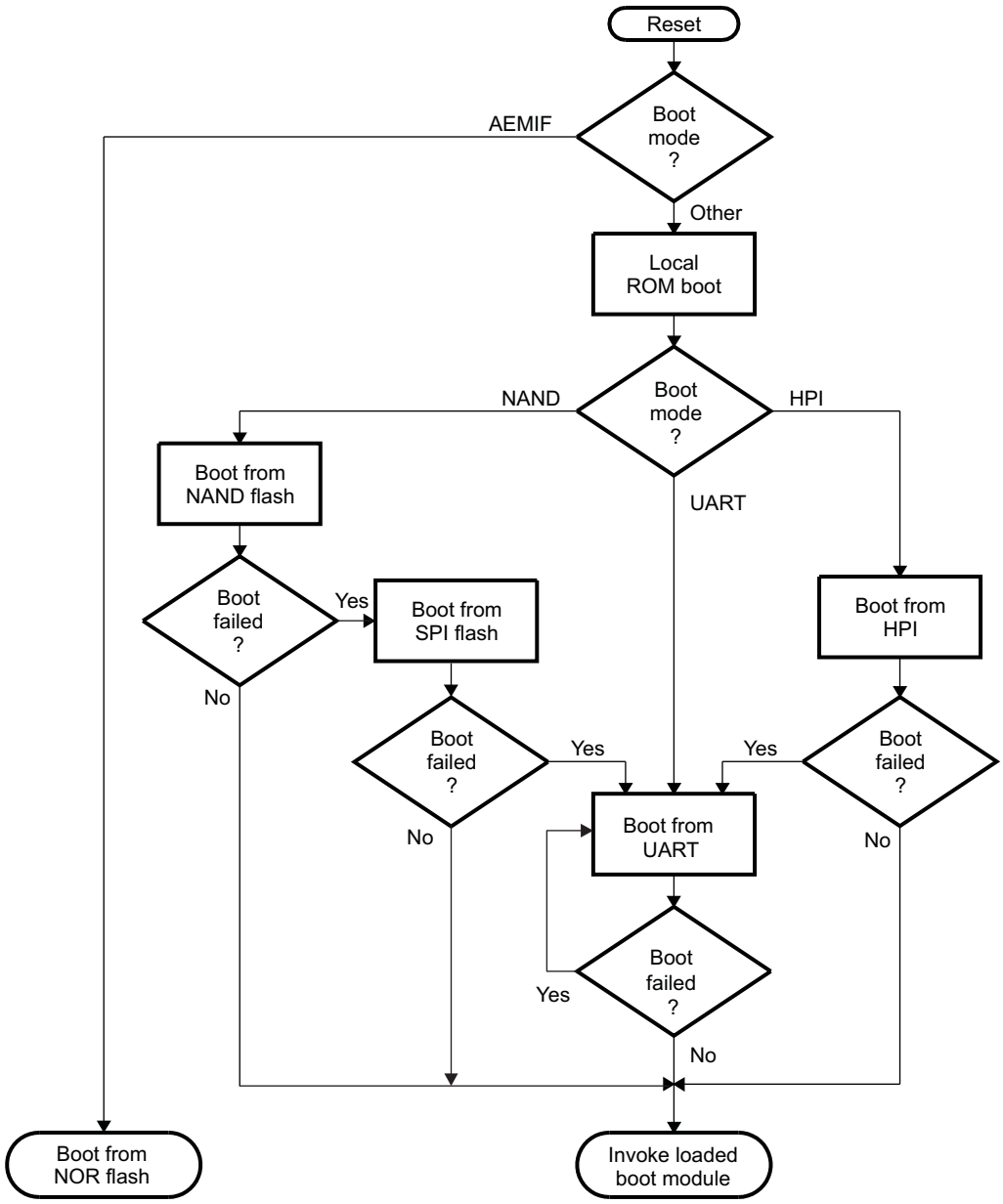
The DM644x DMSoC ARM ROM boot loader (RBL) executes when the BOOTSEL[1:0] pins indicate a condition other than the normal ARM EMIF boot.

- If BTSEL[1:0] = 01 - Asynchronous EMIF (AEMIF or NOR Flash) boot. This mode is handled by hardware control and does not involve the ROM.
- The RBL supports 3 distinct boot modes:
  - BTSEL[1:0] = 00 - ARM NAND/SPI Boot
  - BTSEL[1:0] = 11 - ARM UART Boot
  - BTSEL[1:0] = 10 - ARM HPI Boot
- ARM ROM Boot - NAND/SPI Mode
  - No support for a full firmware boot. Instead, copies a second stage user boot loader (UBL) from NAND flash to ARM internal RAM (AIM) and transfers control to the user-defined UBL.
  - Support for NAND with page sizes up to 2048 bytes.
  - Support for error detection and retry (up to 5 times) when loading UBL
  - Support for up to 14 kB UBL
  - Optional, user-selectable, support for use of DMA and I-cache during RBL execution (that is, while loading UBL)
  - Supports booting from both 8-bit and 16-bit NAND devices with selection determined by the EM_WIDTH pin which determines the NAND CE data bus width at boot.
- ARM ROM Boot - UART mode
  - No support for a full firmware boot. Instead, loads a second stage user boot loader (UBL) via UART to ARM internal RAM (AIM) and transfers control to the user software.
  - Support for up to 14 kB UBL.
- ARM ROM Boot - HPI mode
  - No support for a full firmware boot. Instead, waits for external host to load a second stage user boot loader (UBL) via HPI to ARM Internal RAM (AIM) and transfers control to the user software.
  - Support for up to 14 kB UBL.

12.1.2 Functional Block Diagram

The general boot sequence is shown in Figure 12-1.

Figure 12-1. Boot Mode Functional Block Diagram



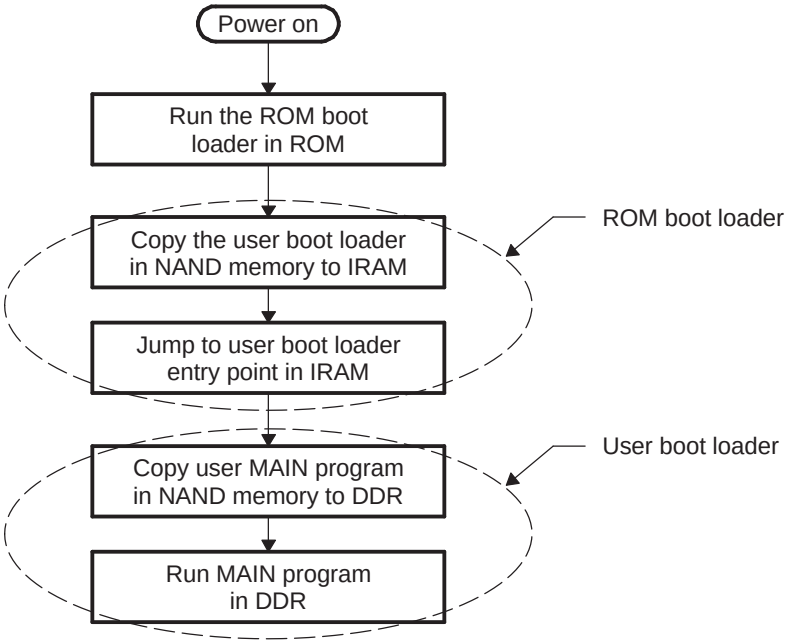
12.2 ARM ROM Boot Modes

The DM644x DMSoC ARM ROM boot loader (RBL) executes when the BOOTSEL[1:0] pins indicate a condition other than the normal ARM EMIF boot (BTSEL[1:0] ≠ 01). In this case, control is passed to the ROM boot loader (RBL). The RBL then executes the proper mode after reading the state of the BTSEL[1:0] pins from the BOOTCFG register.

12.2.1 NAND/SPI Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 00b, the NAND mode executes. The outline of operations followed in the NAND mode is described in Figure 12-2. The NAND boot mode assumes the NAND is located on the EM_CS2 interface, whose bus width is controlled by the external EM_WIDTH pin at reset. The RBL uses the state of the EM_WIDTH pin from the BOOTCFG register to determine the access size to be used when reading data from the NAND.

Figure 12-2. NAND Boot Flow



First the NAND geometry is determined. The following steps are used to determine NAND parameters (Figure 12-3):

- If the device is ONFI, read the parameters page; else command is sent to the NAND device requesting four bytes (called the NAND READ_ID) that contain the manufacturer, device, and 4th ID.
- The RBL contains an internal table with a list of known NAND devices.
- If the device ID is not found in the table, then the RBL uses the fourth byte of the NAND to decode this to obtain the necessary parameters. The manufacturer IDs and format supported are given in Table 12-1.
- If the manufacturer ID is not supported, default parameters (Page size: 2048 bytes, Page count: 64, Address cycles: 5, Page shift: 16, Block shift: 22) are used for NAND geometry.

Then, the RBL searches for the UBL descriptor in page 0 of the block after CIS/IDI block (block 1).

Figure 12-3. NAND Parameter Detection Flow

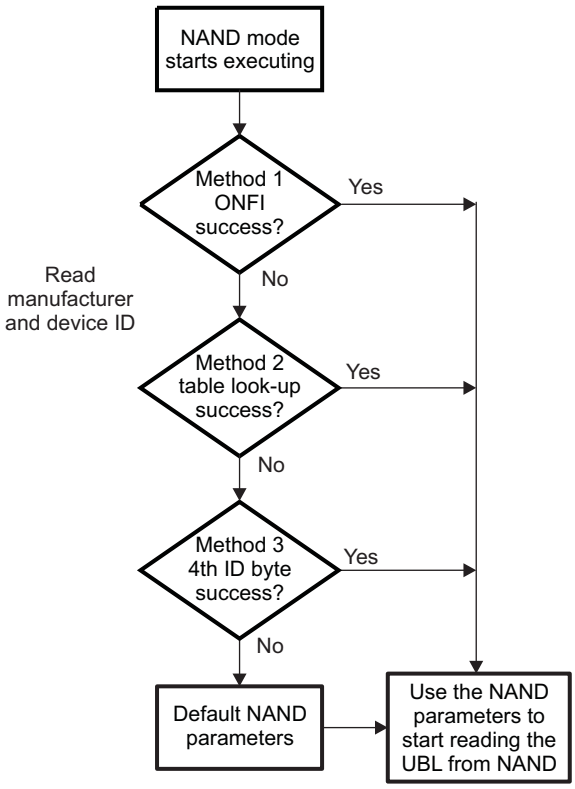


Table 12-1. NAND Manufacturer IDs and Format Supported

Manufacturer ID	Manufacturer	4th ID Format Supported
04h	Fujitsu	Bits 5 and 4 determine the block size:
07h	Renesas	Bits 5,4 = 00: 64 kB
20h	ST Micro or Numonyx	Bits 5,4 = 01: 128 kB
2Ch	Micron	Bits 5,4 = 10: 256 kB
8Fh	National	Bits 5,4 = 11: 512 kB
98h	Toshiba	Bits 1 and 0 determine the page size:
ADh	Hynix	Bits 1,0 = 00: 1 kB
		Bits 1,0 = 01: 2 kB
		Bits 1,0 = 10: 4 kB
		Bits 1,0 = 11: 8 kB
ECh	Samsung	Bits 5 and 4 determine the block size:
		Bits 5,4 = 00: 128 kB
		Bits 5,4 = 01: 256 kB
		Bits 5,4 = 10: 512 kB
		Bits 5,4 = 11: 1024 kB
		Bits 1 and 0 determine the page size:
		Bits 1,0 = 00: 2 kB
		Bits 1,0 = 01: 3 kB
		Bits 1,0 = 10: 4 kB
		Bits 1,0 = 11: reserved

If a valid UBL is not found here, as determined by reading a proper UBL signature, the next block is searched. Searching continues for up to 5 blocks. This provision for additional searching is made in case the first few consecutive blocks have been marked as bad (that is, they have errors). Searching 5 blocks is sufficient to handle the errors found in virtually all NAND devices. If no valid UBL signature is found in the search, the RBL reverts to the UART boot mode.

If a valid UBL is found, the UBL descriptor is read and processed. The descriptor gives the information required for loading and control transfer to the UBL. The UBL is then read and processed. The RBL may enable any combination of faster EMIF and I-Cache operations based on information in the UBL descriptor first. Additionally, the descriptor provides information on whether or not DMA should be used during UBL copying. Once the user-specified start-up conditions are set, the RBL copies the UBL into ARM internal RAM, starting at address 0000:0020h.

NOTE: The actual copying of the UBL is performed on the lower 14 kB of the TCM data area: 8020 to B81Fh.

The first 32-bytes of the ARM internal RAM (AIM) are the ARM's system interrupt vector table (IVT) (8 vectors, 4-bytes each). The UBL copy starts after the 32-byte IVT.

The NAND RBL attempts to verify a correct read by checking the ECC values when reading the UBL in the ARM IRAM. If a read error occurs, the UBL copy immediately halts for that instance, but the RBL continues to search the block following that in which the magic number was found for another instance of a magic number. When a magic number is found, the process repeats. Using this retry process, the magic number and UBL can be duplicated up to 5 times, giving significant redundancy and error resilience to NAND read errors.

As stated previously, when the RBL reads NAND memory pages, it attempts to verify a correct read by checking the ECC values. For every 512 bytes read from the NAND, the EMIF interface generates an ECC value in the NANDF*n*ECC register of the AEMIF peripheral. The RBL compares this value against the value stored in the spare bytes of the NAND page it is reading. These stored values should be placed in the spare bytes region when the page is written. Therefore, any program that writes a UBL header and UBL binary to the NAND flash must know where the RBL looks for the stored ECC values when it tries to load the UBL.

The exact location of where the RBL expects to find the stored ECC depends on the page size of the device. For 256-byte/page and 512-byte/page devices, the ECC value is stored in the first four bytes of the spare byte region (see Table 12-2 and Table 12-3). For 2048-byte/page devices, there are four ECC values (Table 12-4), one for each set of 512 bytes read, which should be stored at addresses 8h, 18h, 28h, and 38h of the spare bytes region. The 32-bit ECC values should be in big-endian order. The provided example UBL respects these ECC requirements, as can be seen by inspecting the source file, *nand.c*.

The NAND user boot loader descriptor format is described in Table 12-5.

Table 12-2. ECC Value Location in Spare Page Bytes of 256-Byte/Page NAND Devices

ECC[3]	ECC[2]	ECC[1]	ECC[0]	XX	XX	XX	XX
0	1h	2h	3h	4h	5h	6h	7h

Table 12-3. ECC Value Location in Spare Page Bytes of 512-Byte/Page NAND Devices

ECC[3]	ECC[2]	ECC[1]	ECC[0]	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh

Table 12-4. ECC Value Location in Spare Page Bytes of 2048-Byte/Page NAND Devices

XX	XX	XX	XX	XX	XX	XX	XX	ECC1[3]	ECC1[2]	ECC1[1]	ECC1[0]	XX	XX	XX	XX
0	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh
XX	XX	XX	XX	XX	XX	XX	XX	ECC2[3]	ECC2[2]	ECC2[1]	ECC2[0]	XX	XX	XX	XX
10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh
XX	XX	XX	XX	XX	XX	XX	XX	ECC3[3]	ECC3[2]	ECC3[1]	ECC3[0]	XX	XX	XX	XX
20h	21h	22h	23h	24h	25h	26h	27h	28h	29h	2Ah	2Bh	2Ch	2Dh	2Eh	2Fh
XX	XX	XX	XX	XX	XX	XX	XX	ECC4[3]	ECC4[2]	ECC4[1]	ECC4[0]	XX	XX	XX	XX
30h	31h	32h	33h	34h	35h	36h	37h	38h	39h	3Ah	3Bh	3Ch	3Dh	3Eh	3Fh

Table 12-5. NAND UBL Descriptor

Page 0 Address	32-Bits	Description
0	A1AC EDxxh	Magic number (A1AC EDxxh)
4	Entry Point Address of UBL	Entry point address for the user boot loader (absolute address)
8	Number of pages in UBL	Number of pages (size of user boot loader in number of pages)
12	Starting Block # of UBL	Block number where user boot loader is present
16	Starting Page # of UBL	Page number where user boot loader is present

12.2.1.1 NAND Device IDs Supported

The list of IDs supported by ROM boot loader is shown in Table 12-6 with its characteristics.

NOTE: The DM644x DMSoC does not support NAND flashes that require the chip select to stay low during the tR time for a read.

Table 12-6. NAND IDs Supported

Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (for address)	Number of address cycles
6Eh	16	256 + 8	12	3
68h	16	256 + 8	12	3
ECh	16	256 + 8	12	3
E8h	16	256 + 8	12	3
EAh	16	256 + 8	12	3
E3h	16	512 + 16	12	3
E5h	16	512 + 16	12	3
E6h	16	512 + 16	12	3
39h	16	512 + 16	12	3
6Bh	16	512 + 16	12	3
73h	32	512 + 16	13	3
33h	32	512 + 16	13	3
75h	32	512 + 16	13	3
35h	32	512 + 16	13	3
76h	32	512 + 16	13	4
36h	32	512 + 16	13	4
79h	32	512 + 16	13	4

Table 12-6. NAND IDs Supported (continued)

Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (for address)	Number of address cycles
71h	32	512 + 16	13	4
46h	32	512 + 16	13	4
56h	32	512 + 16	13	4
74h	32	512 + 16	13	4
F1h	64	2048 + 64	22	4
A1h	64	2048 + 64	22	4
AAh	64	2048 + 64	22	5
DAh	64	2048 + 64	22	5
ACh	64	2048 + 64	22	5
DCh	64	2048 + 64	22	5
ACh	64	2048 + 64	22	5
B1h	64	2048 + 64	22	4
C1h	64	2048 + 64	22	4

12.2.1.2 UBL Signature and Special Modes

Different NAND boot mode options are selected by setting different MAGIC IDs in the UBL descriptor. Table 12-7 lists the UBL signatures.

Table 12-7. UBL Signatures and Special Modes

Mode	Value	Description
UBL_MAGIC_SAFE	A1AC ED00h	Safe boot mode
UBL_MAGIC_DMA	A1AC ED11h	DMA boot mode
UBL_MAGIC_IC	A1AC ED22h	I Cache boot mode
UBL_MAGIC_FAST	A1AC ED33h	Fast EMIF boot mode
UBL_MAGIC_DMA_IC	A1AC ED44h	DMA + I Cache boot mode
UBL_MAGIC_DMA_IC_FAST	A1AC ED55h	DMA + I Cache + Fast EMIF boot mode

12.2.1.3 SPI Boot Mode

As shown in Figure 12-1, when the NAND boot fails, the SPI boot is automatically executed. The DM644x DMSoC loads the UBL data in the following locations, ARM TCM RAM received via SPI0. The UBL data is received from a serial device like a serial EEPROM.

12.2.1.3.1 SPI Key Features

The key features for SPI are:

- Master interface to a serial EEPROM/Flash for initial code load
- Support for fast boot mode through the UBL descriptor
- Support for prescaler through the UBL descriptor
- Support for 16-bit and 24-bit addressable EEPROMs through the UBL descriptor (see Table 12-8)
- Support for 4-pin SPI (CS, CLK, serial input, serial output)

Table 12-8. User Boot Loader (UBL) Descriptor for SPI Mode

Byte Range	32-Bits	Description
0-3	A1AC ED0xh	Magic number: A1AC ED00h = 24 bit A1AC ED01h = 16 bit
4-7	Entry Point	Entry point address for the user boot loader (absolute address) in ARM internal memory.
8-11	UBL size	Size of UBL in bytes.
12	Prescaler	Prescaler value to be used for dividing the clock for SPI.
13	FASTREAD	Flag for enabling fast read. FAST READ option may not be valid for a specific EEPROM. Note the EEPROM specifications before setting this parameter. 0 = fast read is disabled 1 = fast read is enabled
14-15	0000h	Dummy bytes
16-19	Start address of UBL	Start address of UBL in EEPROM.
20-23	Load address	Load address of UBL in ARM internal memory.

12.2.1.3.2 SPI Boot - Detailed Flow

The following list describes the flow of the SPI boot:

- RBL configures the pin-multiplexing settings to bring out the SPI0 signals.
- RBL configures the EEPROM initially in 24-bit addressable mode and reads the first byte. Based on the first byte it configures the EEPROM to 16-bit or 24-bit addressable modes.
- Boot loader reads entire UBL descriptor and finds out the properties of slave EEPROM. The UBL descriptor contains the prescalar value, which is the divider used to generate the SPI clock. The FAST_READ flag is used to indicate fast/normal mode. RBL uses the FAST_READ command if the flag is set; else, uses the standard READ command.
- RBL validates the other UBL header parameters.
- Downloads the UBL to the ARM internal memory.
- RBL updates the boot status and then passes control to the entry point given in the UBL descriptor.

12.2.2 UART Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 11b, the UART serial boot mode executes.

This mode enables a small program, referred to here as a user boot loader (UBL), to be downloaded to the on-chip ARM internal RAM via the on-chip serial UART and executed. A host program, (referred to as serial host utility program), manages the interaction with RBL and provides a means for operator feedback and input.

The UART boot mode execution assumes the following UART settings:

Time-Out	500 ms, one-shot
Serial RS-232 port	115.2 Kbps(1), 8-bit, no parity, one stop bit

(1) Specified for a 27-MHz MXI/CLKIN clock frequency. Buad rate changes as the frequency of MXI/CLKIN changes.

12.2.2.1 TMS320DM644x DMSoC and Serial Host Handshake

Figure 12-4 shows the handshake between the DM644x DMSoC and a serial host utility program. After initialization, there are three main receive sequences: ACK, 1 kB CRC32 table, and UBL. For each receive sequence, the time-out check is done in the RBL. If the timeout value is reached during the sequence, the serial boot mode restarts from the beginning which sends out the BOOTME message. The error checking behavior for the UART receive mode is the same. For each byte received, if there is an error, RBL restarts from the beginning.

The checksum method used for UBL data is CRC32 checksum. The lookup table that is used for the CRC32 calculation (1 kB) must be sent by the host serial utility. Checksum8 is used as the checksum methodology for the CRC32 lookup table.

The checksum8 value for the lookup table when calculated results in a value of 0. Since this value remains the same, it is checked by the RBL before downloading the UBL data from the host serial utility. Whenever a wrong ACK, CRC32 table or UBL is received, the serial boot process restarts.

12.2.2.2 UART Boot Loader Data Sequences

The serial boot loader data sequences consist of handshake messages, UBL header, and the UBL payload itself. The messages use a fixed 8-byte ASCII string including a null string terminator. Short messages have leading spaces besides the null.

Table 12-9 lists the values for the handshake sequences and header for UBL.

Table 12-9. UART Data Sequences

Sequence	Sequence	Usage
BOOTME	^BOOTME/0	Notify host utility serial boot mode begins. This is an 8-byte ASCII value. ^ is a space.
ACK	^^^^ACK/0	For the host utility to respond within the time out period by sending a 28-byte header to prepare for reception of user boot loader. The checksum is a 32-bit checksum.
	UBL 8-byte checksum	
	UBL 4-byte count	
	UBL 4-byte ARM physical start address	
	TBD 4-byte zeros	

NOTE: RBL jumps to the start address after the downloading process (that is, UBL entry point).

BEGIN	^^BEGIN/0	RBL to signal host utility to begin transmission of user boot loader
DONE	^^^DONE/0	RBL to signal host utility that data received is OK and the transfer can be terminated
BAD ADDR	BADADDR/0	Bad start address received
BAD COUNT	^BADCNT/0	Bad count received
CORRUPT	CORRUPT/0	RBL to signal host utility that there is an error with the transmission. The host utility asks you to reset the board.
UBL	Variable	The format for UBL is the same as NAND boot.

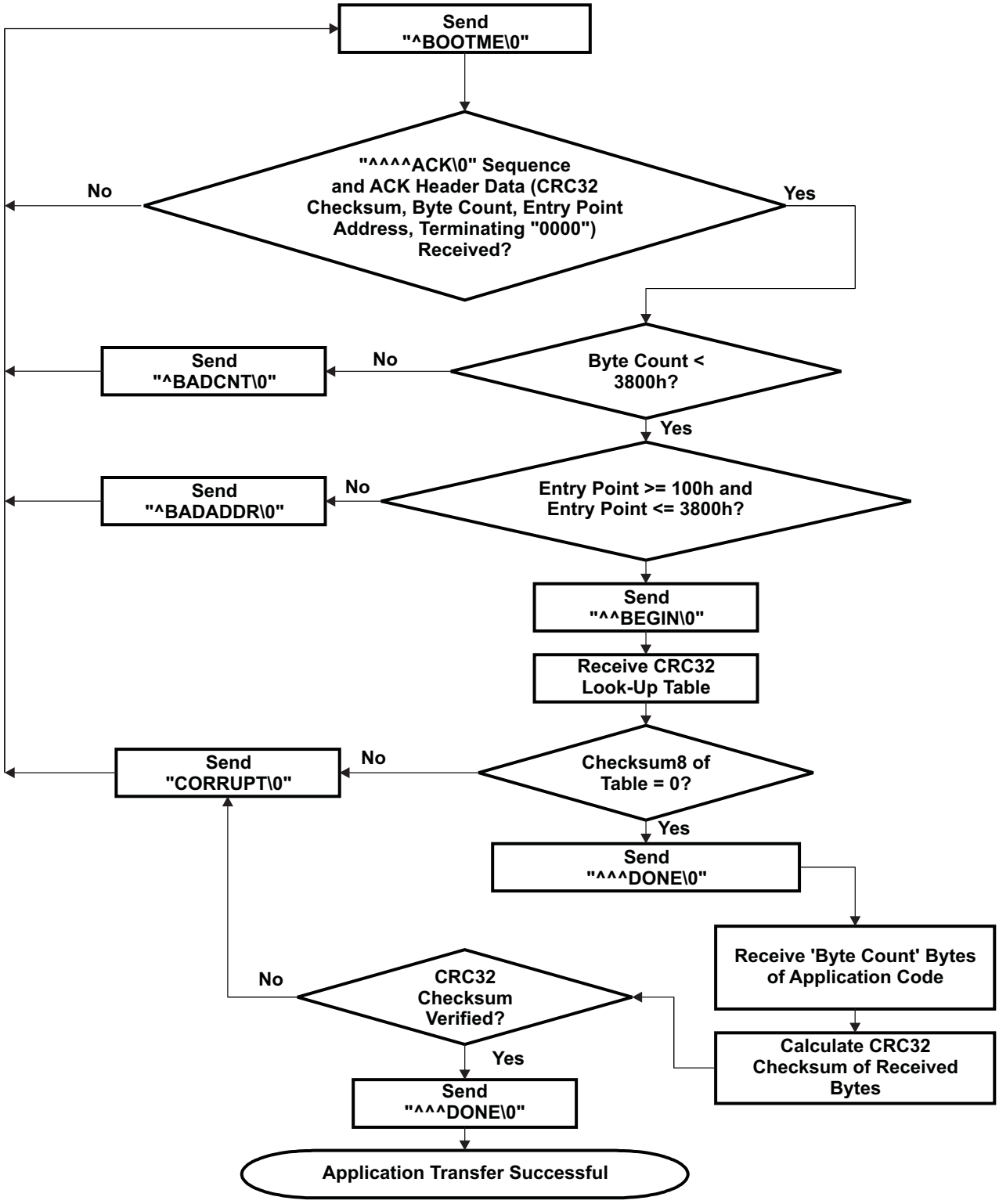
The CRC32 checksum value is calculated for the UBL data and passed by the host serial utility. The polynomial used for CRC32 is:

X^32 + X^26 + X^23 + X^22 + X^16 + X^12 + X^11 + X^10 + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + X^0.

Although the CRC32 results in a 32 bit value (4 bytes), the host serial utility transmits 8 characters (bytes). Example 12-1 illustrates.

For additional details describing the CRC32 checksum algorithm, see *Basic Application Loading Over the Serial Interface for the DaVinci DM644x* (SPRAAI0).

Figure 12-4. UART Boot Mode Handshake



NOTE: Messages are sent as ASCII 8-byte characters representing hexadecimal numbers.

Example 12-1. Host Serial Utility Transmission of Characters

For a given UBL data, let the checksum (CRC32) value calculated be 0x ffaa 10a1. Then, instead of the host utility transmitting “ascii (0xff) ascii (0xaa) ascii (0x10) ascii (a1)” , it will transmit “ffaa 10a1”. These 8 characters (bytes) are appropriately interpreted by the RBL.

You can generate the user boot loader using any ARM code generation tools, but the final format is expected in binary memory image format with no headers, etc.

The starting address of the UBL is at 0020h to allocate space for a 32-byte interrupt vector table.

12.2.3 HPI Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 10b, the HPI boot mode executes. The operations followed in the HPI boot mode are shown in Figure 12-5.

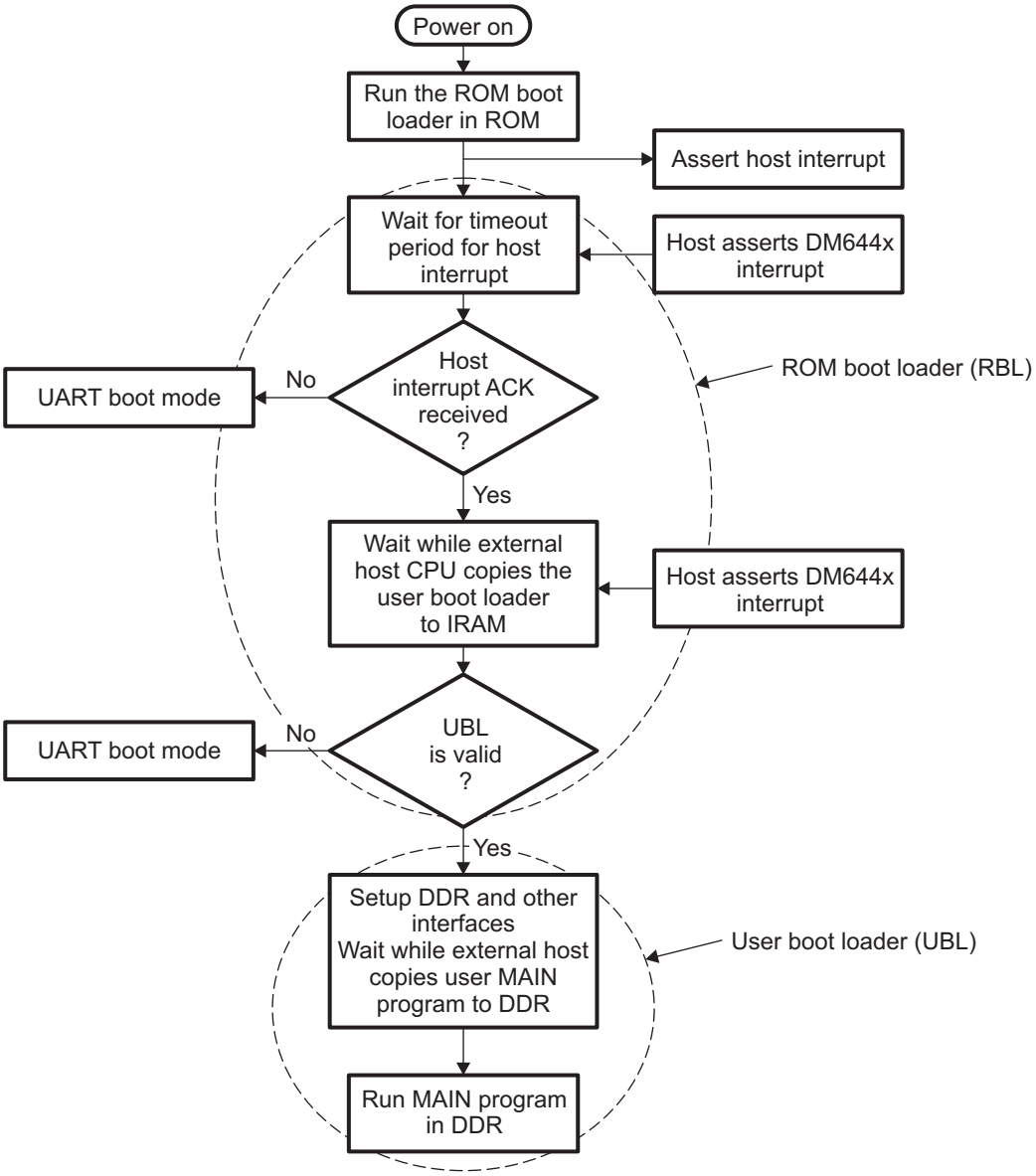
First, the DM644x DMSoC RBL signals the Host that it is ready via the Host interrupt/$\overline{\text{HINT}}$ signal. Once the interrupt has been sent to the Host, the RBL drops into a polling loop, waiting for the external Host interrupt to acknowledge (ACK) its presence. The RBL waits in this polling loop for a timeout period of approximately 50 seconds. After approximately 50 seconds if the Host has not ACK its presence, the RBL aborts the HPI boot and transfers control to the backup UART boot mode. If the Host does ACK via the interrupt, the RBL drops into a polling loop, waiting for the Host to load/copy the UBL. The RBL waits indefinitely for the Host to load/copy the UBL.

NOTE: The entry point must be specified in terms of the ARM instruction TCM area and should be in the range 0020h to 381Ch, if the UBL is in ARM IRAM.

When loading the UBL, the Host can write up to a 14kB image to 8020h to B81Fh. The UBL should be followed by a footer consisting of the magic number and entry point address (at B820h and B824h, respectively). The Host must not copy data to the interrupt vector table region (8000h to 8020h in data space or 0000h to 0020h in program space) or to the RBL stack region (BA00h to BFFFh in data space or 3A00h to 3FFFh in program space).

Once the Host has completed loading the UBL, it should signal the DM644x DMSoC that it is finished by setting the DSPINT bit in the HPIC register. After the Host interrupt is received, the RBL verifies the magic number at B820h, making sure the UBL is good. A valid magic number is A1AC ED00h. If the magic number is verified as valid, the RBL transfers control to the UBL entry point specified at B824h; if the magic number is invalid, the HPI boot fails and the RBL transfers control to the UART boot mode.

Figure 12-5. HPI Boot Sequence



ARM-DSP Integration

Topic **Page**

13.1 Introduction

The TMS320DM644x DMSoC integrates an ARM core for overall system control functions and a DSP subsystem for complex data and image/video processing functions. Figure 13-1 shows the interconnections between the ARM and the DSP cores and the shared resources. Both the ARM and the DSP have access to the EDMA and to the audio serial port (ASP) peripherals. Both the ARM and DSP have access to several blocks of shared memory, including ARM internal memory, DSP internal memory, and external memory of the DDR2 memory controller and Asynchronous EMIF (AEMIF). The system control module includes registers that allow the ARM to interrupt the DSP and conversely allow the DSP to interrupt the ARM. The power and sleep controller (PSC) and the system control module (SYS) provide the ARM with a set of registers to boot the DSP, power-on/off the DSP, enable/disable the DSP clock, and reset the DSP.

In summary, ARM-DSP integration includes all of the following features:

- Shared peripherals
  - ARM and DSP have access to EDMA
  - ARM and DSP have access to ASP
- Shared memory
  - ARM can access DSP internal memory (L1P, L1D, L2)
  - DSP can access ARM internal memory
  - ARM and DSP can access DDR2 memory controller and AEMIF
- ARM-DSP interrupts
  - ARM can interrupt the DSP (via 4 general interrupts and 1 NMI)
  - DSP can interrupt the ARM (via 2 general interrupts)
- ARM control of DSP power, clock, reset, and boot
  - ARM can boot the DSP
  - ARM can power-on/off the DSP [Note that DSP domain power-down is not supported on the DM644x DMSoC]
  - ARM can control DSP
  - Clock on/off
  - ARM can assert/de-assert DSP module and local resets

These features are described in the following sections.

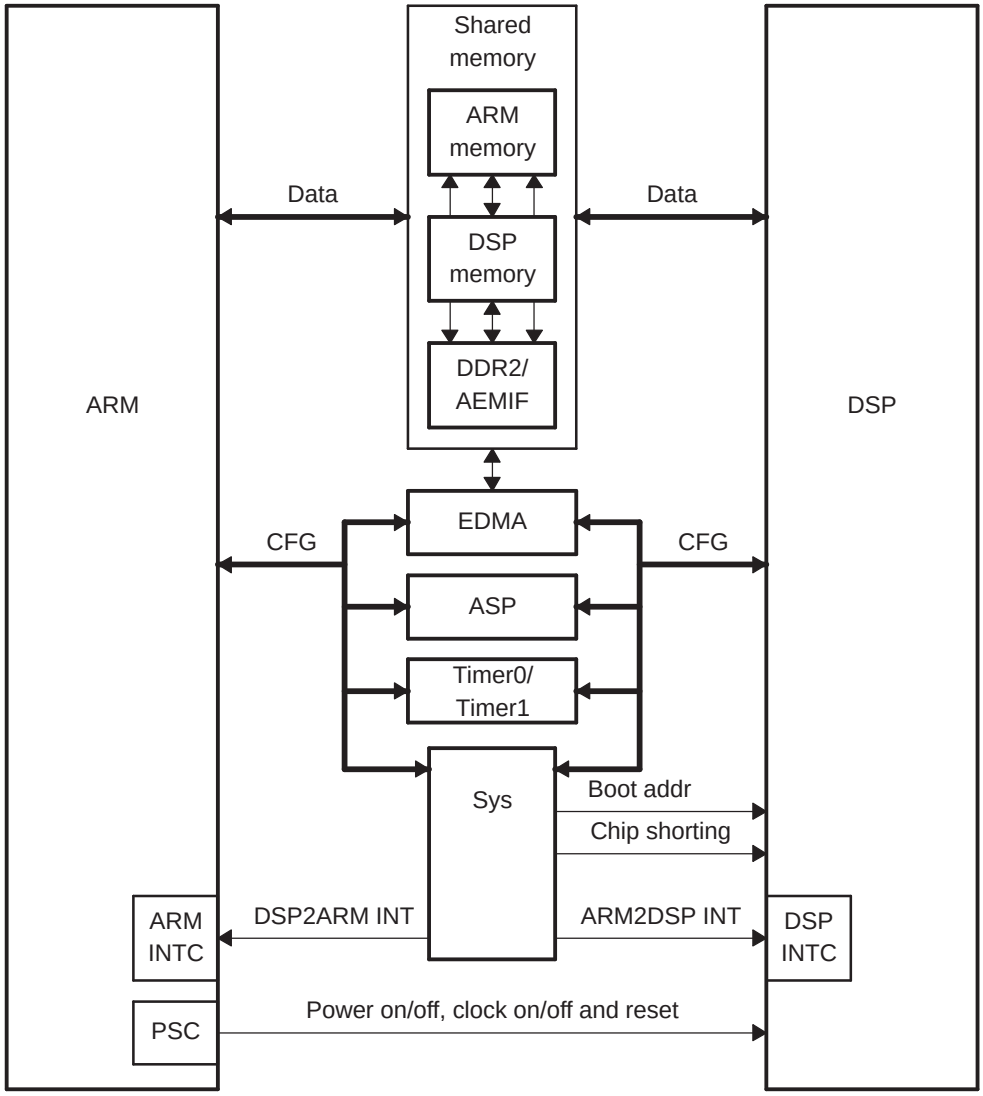
13.2 Shared Peripherals

The following peripherals are fully accessible by both the ARM and the DSP.

- EDMA
- ASP

Both the ARM and the DSP access these peripherals through the configuration bus. See Chapter 4 for information on the configuration bus.

Figure 13-1. ARM-DSP Integration



13.3 Shared Memory

The DM644x DMSoC memory-map is described in detail in Chapter 4. As noted in Chapter 4, ARM, DSP, and EDMA all have access to ARM internal memory, DSP internal memory, and external memory of the DDR2 memory controller and AEMIF. The EDMA can transfer data among shared memory without ARM or DSP intervention. See the *TMS320DM644x DMSoC Enhanced Direct Memory Access (EDMA) Controller User's Guide* (SPRUE23) for more information on the EDMA.

13.3.1 ARM Internal Memories

The ARM, DSP, and EDMA can access the ARM's internal memories:

- 16 kB ARM internal RAM
- 16 kB ARM internal ROM

13.3.2 DSP Memories

The ARM, DSP, and EDMA can access the DSP's internal memories:

- L1P RAM (32 kB)
- L1D RAM (80 kB)
- L2 RAM (64 kB)

This feature allows the ARM and EDMA to load DSP memories with program instructions and data.

NOTE: Portions of the above DSP memories are configurable as DSP cache memory. When configured as cache, neither the ARM nor the EDMA can access the cache portions. For more information on the DSP internal memories and cache configuration, see the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15).

13.3.3 External Memories

Both the ARM and the DSP have access to devices connected to the DDR2 memory controller and the AEMIF. This allows the ARM and DSP to access program and data from DDR on the DDR2 memory controller and from devices attached to the AEMIF, such as NOR flash or SRAM.

NOTE: The DSP can access the data space of the DDR2 memory controller and of the AEMIF. However, the DSP cannot access the control register space of these EMIFs. Therefore, it is the ARM'S responsibility to configure the control registers of the DDR2 memory controller and the AEMIF.

13.4 ARM-DSP Interrupts

The ARM can interrupt the DSP; conversely, the DSP can interrupt the ARM. These interrupts are generally used to allow the ARM and the DSP to coordinate. For example, the ARM may interrupt the DSP when it is ready to have the DSP process some data buffer in shared memory. A typical sequence is as follows:

- ARM writes command in shared memory
- ARM interrupts DSP
- DSP responds to interrupt and reads command in shared memory
- DSP executes a task based on the command
- DSP interrupts ARM upon completion of the task

This sequence is often referred to as ARM-DSP communication.

The ARM has access to five DSP interrupt events. These events are labeled ARM2DSP0, ARM2DSP1, ARM2DSP2, ARM2DSP3, and NMI in the DSP interrupt event map. The DSP has access to two ARM interrupts. These interrupts are labeled DSP2ARM0 and DSP2ARM1 in the ARM interrupt map. The ARM-DSP interrupts/events are summarized in Table 13-1.

Table 13-1. ARM-DSP Interrupt Mapping

ARM Interrupt Map			DSP Interrupt Event Map		
Interrupt Number	**Label**	**Source**	**Event #**	**Label**	**Source**
46	DSP2ARM0	DSP	16	ARM2DSP0	ARM
47	DSP2ARM1	DSP	17	ARM2DSP1	ARM
			18	ARM2DSP2	ARM
			19	ARM2DSP3	ARM
			n/a	NMI	ARM

The ARM and DSP use the INTGEN register in the system control module to interrupt each other. See Chapter 10 for more information on the INTGEN register. The ARM can interrupt the DSP by setting the INTDSP[3:0] bits or the INTNMI bit. Similarly, the DSP can interrupt the ARM by setting the INTARM[1:0] bits. Interrupt status is reflected in the corresponding status bits.

For more information on ARM interrupts, see Chapter 9. For more information on DSP interrupts, see the DSP interrupts section of the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15). For more information on the system control module, see Chapter 10.

13.5 ARM Control of DSP Boot, Power, Clock, and Reset

As system master, the ARM can control all of the following functions:

- Boot the DSP
- Power the DSP on/off
- Turn the clock on/off
- Reset the DSP

To initiate the above operations, firmware on the ARM must coordinate with the DSP and use the power and sleep controller (PSC) module. This section provides specific details on how to initiate these operations. For more information on the PSC and system control module, see Chapter 7 and Chapter 10.

NOTE: The DM644x DMSoC does not support powering down the DSP power domain. However, the information is included here because there are cases where the ARM will be responsible for powering on the DSP power domain after reset (depending on the boot mode).

13.5.1 DSP Boot

The DSP can boot in either of the following two modes:

- ARM boots DSP mode
- DSP self-boot mode

In the ARM boots DSP mode, the ARM is responsible for managing DSP boot after power-on/reset. In DSP self-boot mode, the DSP boots without intervention from the ARM immediately upon power-on/reset. The mode is determined by sampling the pin COUT3_BTSEL at power-on/reset (Table 13-2). See Chapter 11 and Chapter 12 for more information on boot and reset modes. This section describes the procedure to boot the DSP with the ARM, when in ARM boots DSP mode.

Table 13-2. DSP Boot Configuration

Device Configuration	Function	Sampled Pin	Default Setting
DSP boot	0 = ARM boots DSP 1 = DSP self-boots	COUT3_BTSEL	ARM boots DSP

To boot the DSP, the ARM must specify a boot address in the DSPBOOTADDR bit in SYS of the system control module and ensure that DSP program code is loaded properly into memory. If the DSP is powered-down or isolated from the rest of the DMSoC, the ARM must power-up the DSP and/or reconnect it to the rest of the DMSoC. When the ARM releases the DSP from reset, the DSP immediately begins code execution from the DSPBOOTADDR bit in SYS.

- ARM: Boot the DSP
  - Put the DSP power domain in the on state and put the DSP module in the enable state. Prior to beginning the DSP boot sequence, the DSP power domain must be on and the DSP module must be in the enable state. See Section 13.5.2 and Section 13.5.3 for information on how to execute power-on and module clock on.
  - Clear the LRSTZ bit in MDCTL39 to 0. This asserts the DSP's local reset. By default, after power-on reset or hard reset, the value of the LRSTZ bit in MDCTL39 is cleared to 0.
  - Set the DSP boot address in the DSPBOOTADDR bit in SYS. By default, after power-on or hard reset, the value in DSPBOOTADDR is 4220 0000h, which maps to the AEMIF. The ARM software can specify a boot address that maps to L1P internal DSP memory, AEMIF, or DDR2 memory controller. The DSP can execute program instructions from any of these memories.
  - Ensure that the DSP program code is loaded / stored with a reset vector at the DSP boot address specified in the previous step.
  - Set the LRSTZ bit in MDCTL39 to 1. This step de-asserts the DSP local reset. The DSP immediately executes program instructions starting at the boot address after reset is de-asserted.

13.5.2 DSP Power Domain ON/OFF

In the DM644x DMSoC, the DSP power domain can be turned completely off. Turning the DSP power domain completely off allows for maximal power savings across the DSP subsystem. This section describes the procedures that are necessary for DSP power-on/off. For more information describing DSP power down features, see Chapter 8 and the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15).

NOTE: The ARM typically masters DSP power management; therefore, power management is covered more fully in the this guide.

The DM644x DMSoC does not support powering down the DSP power domain. However, the information is included here because there are cases where the ARM will be responsible for powering on the DSP power domain after reset (depending on the boot mode).

13.5.2.1 DSP Domain Power On

Power to the DSP is turned on in the power-on state. The power-on state is the normal domain state when the DSP is running.

- ARM: Provide power and enable clock to the DSP
  - Apply power to the power pins (CV_{DDDSP}) of the DSP power domain. In this step, the ARM coordinates with an external device (for example, microcontroller) to supply power to the power pins. See the device-specific data manual for information on the power pins of the DSP power domain.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. You must wait for any previously initiated transitions to finish before initiating a new transition.
  - Set the NEXT bit in PDCTL1 to 1 to prepare the DSP power domain for an on transition.
  - Set the NEXT bit in MDCTL39 to 3h to prepare the DSP module for an enable transition.

    NOTE: The DSP clock is enabled in this step. Thus, this step is not needed to turn power on; however, in typical applications, it is efficient to enable the clock in this sequence rather than enabling the DSP clock in a separate sequence. For more information on enabling the DSP clock, see to Section 13.5.3.1.
  - Set the GO[1] bit in PTCMD to 1 to initiate the state transitions. After this bit is set, the PSC begins the process of de-activating isolation cells that exist around the DSP.
  - Wait for the EPC bit in EPCPR to change to 1 to indicate that the power has been applied. You can do this step by polling the EPC bit in EPCPR or by enabling the PSC's external power control pending interrupt. See Section 7.6 for detailed information on this interrupt.
  - Set the DSPPWRON bit in the chip shorting switch control register (CHP_SHRTSW) of the system control module to 1 to short the power rails of the AlwaysOn and DSP power domains. The DSPPWRON bit in CHP_SHRTSW controls power rail shorting. Setting this bit to 1 closes the switch; the switch must be closed for proper DSP power-on. The DSP_BT pin configuration determines the default switch value. If DSP self-boot is selected (DSP_BT = 1), then the DSP is powered-up and DSPPWRON defaults to 1. For ARM Boots DSP mode (DSP_BT = 0), the DSPPWRON bit in CHP_SHRTSW defaults to 0 and the ARM must set it after DSP domain power is turned on.
  - Set the EPCGOOD bit in PDCTL1 to 1 to indicate that power has been applied. The PSC proceeds with the transition after software sets the EPCGOOD bit in PDCTL1 to 1.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[1] bit is cleared to 0.

13.5.3 DSP Module Clock ON/OFF

13.5.3.1 DSP Module Clock ON

In the clock enable state, the DSP's module clock is enabled while DSP module reset is de-asserted. This is the state for normal DSP run-time.

- ARM: Enable clocks to the DSP.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  - Set the NEXT bit in MDCTL39 to 3h to prepare the DSP module for an enable transition.
  - Set the GO[1] bit in PTMCD to 1 to initiate the state transition.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[1] bit is cleared to 0.
  - Wait for the STATE bit in MDSTAT39 to change to 3h. The module is only safely in the new state after the STAT bit in MDSTAT39 changes to reflect the new state.

- ARM: Wake the DSP.
  - If transitioning from the disable state, trigger a DSP interrupt that has previously been configured as a wake-up interrupt, (for example, set INTDSP*n* or INTNMI in INTGEN).

NOTE: This step only applies if you are transitioning from the disable state. If previously in the disable state, a wake-up interrupt must be triggered in order to wake the DSP. This example assumes that the DSP enabled this interrupt before entering its IDLE state. If previously in the software reset disable or synchronous reset state, it is not necessary to wake the DSP because these states assert the DSP module reset. See Chapter 11 for information on the software reset disable and synchronous reset states. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on DSP interrupts.

13.5.3.2 DSP Module Clock Off

In the clock disable state, the DSP's module clock is disabled, while DSP reset remains de-asserted. This state is typically used to disable the DSP clock to save power.

- ARM: Notify the DSP to prepare for power-down.
- DSP: Prepare for power-down.
  - Set PDCCMD to 0001 5555h. PDCMD is a control register in the DSP power-down controller module.

NOTE: This register can only be written while the DSP is in supervisor mode. See the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15) for more information on the power-down controller.

  - Enable one of the ARM2DSP interrupts: ARM2DSP0, ARM2DSP1, ARM2DSP2, ARM2DSP3, or NMI. This interrupt wakes the DSP in the DSP clock-on sequence.
  - Execute the IDLE instruction. IDLE is a program instruction in the C64x+ CPU instruction set. When the CPU executes IDLE, the PDC is notified and initiates DSP power-down according to the bits that you set in the PDCCMD (0181 0000h) register. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the PDC and the IDLE instruction.
- ARM: Disable the DSP clock.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  - Set the NEXT bit in MDCTL39 to 2h to prepare the DSP module for a disable transition.
  - Set the GO[1] bit in PTCMD to 1 to initiate the state transition.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[1] bit is cleared to 0.
  - Wait for the STATE bit in MDSTAT39 to change to 2h. The module is only safely in the new state after the STATE bit in MDSTAT39 changes to reflect the new state.

13.5.4 DSP Reset

With access to the PSC registers, the ARM can assert and de-assert DSP local reset and DSP module reset. When DSP local reset is asserted, the DSP's internal memories (L1P, L1D, and L2) are still accessible. Local reset only resets the DSP CPU. Local reset is useful when the DSP module is in the enable or disable states, since module reset is asserted in the SyncReset and SwRstDisable states and module reset supersedes local reset. The intent of local reset is for the ARM to use local reset to reset the DSP during the DSP boot process. The intent of module reset is for it to completely reset the DSP (like hard reset). For more information on the PSC, see Chapter 7. For more information on local reset and on module reset, see Chapter 11 and Chapter 12. This section describes how to initiate DSP local reset and module reset.

13.5.4.1 DSP Local Reset

The following steps describe how to assert/de-assert local reset:

1. Clear the LRSTZ bit in MDCTL39 to 0 to assert DSP reset.
2. Set the LRSTZ bit in MDCTL39 to 1 to de-assert DSP reset.

13.5.4.2 DSP Module Reset

13.5.4.2.1 Software Reset Disable

In the software reset disable state, the DSP's module reset is asserted and its module clock is turned off. You can use this state to reset the DSP. The following steps describe how to put the DSP in the software reset disable state:

- ARM: Notify the DSP to prepare for power-down.
- DSP: Put the DSP in the IDLE state.
  - Set PDCCMD to 0001 5555h. PDCMD is a control register in the DSP power-down controller module.

    NOTE: This register can only be written while the DSP is in its supervisor mode. See the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15).
  - Execute the IDLE instruction if the DSP is in the enable state. IDLE is a program instruction in the C64x+ CPU instruction set. When the CPU executes IDLE, the PDC is notified and will initiate the DSP power-down according to the bits that you set in the PDCCMD (0181 0000h) register. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the PDC and the IDLE instruction.
- ARM: Software reset disable DSP.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  - Clear the NEXT bit in PDCTL1 to 0 to prepare the DSP module for a software reset disable transition.
  - Set the GO[1] bit in PTCMD to 1 to initiate the state transition.
  - Wait for GOSTAT[1] bit in PTSTAT to clear to 0. The domain is safely in the new state only after the GOSTAT[1] bit is cleared to 0.
  - Wait for the STATE bit in MDSTAT39 to change to 0. The module is safely in the new state only after the STATE bit in MDSTAT39 changes to reflect the new state.
- ARM: Assert the DSP local reset.
  - Clear the LRSTZ bit in MDCTL39 to 0. This step is optional. This step asserts the DSP local reset, and is included here so that the DSP does not start running immediately upon power-on/enable. Typically, software de-asserts local reset sometime after finishing the power-on and enable sequence.

13.5.4.2.2 Synchronous Reset

In the synchronous reset state, the DSP's module reset is asserted and its module clock is enabled. You can use this state to reset the DSP. The following steps describe how to put the DSP in the synchronous reset state:

- ARM: Notify the DSP to prepare for power-down.
- DSP: Put the DSP in the IDLE state.
  - Set PDCCMD to 0001 5555h. PDCMD is a control register in the DSP power-down controller module.

NOTE: This register can only be written while the DSP is in supervisor mode. See the *TMS320DM644x DMSoC DSP Subsystem Reference Guide* (SPRUE15) for more information on the power-down controller.

  - Execute the IDLE instruction.
- ARM: Sync reset DSP
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  - Set the NEXT bit in PDCTL to 1 to prepare the DSP module for a software reset disable transition.
  - Set the GO[1] bit in PTCMD to 1 to initiate the state transition.
  - Wait for the GOSTAT[1] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[1] bit is cleared to 0.
  - Wait for the STATE bit in MDSTAT39 to change to 1. The module is only safely in the new state after the STATE bit in MDSTAT39 changes to reflect the new state.
- ARM: Assert DSP local reset.
  - Clear the LRSTZ bit in MDCTL39 to 0. This step is optional. This step asserts the DSP local reset and is included here so that the DSP does not start running immediately upon power-on/enable. Typically, software de-asserts local reset some time after finishing the power-on and enable sequence.

Revision History

Table A-1 lists the changes made since the previous version of this document.

Table A-1. Document Revision History

Reference	Additions/Modifications/Deletions
Table 5-1	Changed table.
Table 5-2	Changed table.
Table 5-3	Added table. Subsequent tables renumbered.
Table 5-6	Changed frequency range of Video Processing Front End (VPFE).
	Changed frequency range of DDR2 Memory Controller.
Section 6.4	Deleted table.
Table 6-3	Added register addresses.
Table 6-4	Added table.
Table 7-1	Changed description of BTSEL = 00.
Table 7-5	Added register addresses.
	Changed PDSTAT*n*.
	Changed PDCTL*n*.
Table 9-2	Added register addresses.
Figure 9-7	Changed figure.
Table 9-5	Changed table.
Figure 9-12	Changed figure.
Table 9-10	Changed table.
Table 10-3	Added register addresses.
	Added HPI_CTL.
Table 11-3	Changed description of BTSEL[1:0] = 00.
	Changed Default Setting for BTSEL[1:0].
	Changed NOTES.
Section 12.1.1	Changed first sub-bullet in second bullet.
Figure 12-1	Changed figure.
Section 12.2.1	Changed subsection title.
	Changed second paragraph.
Figure 12-3	Added figure. Subsequent figures renumbered.
Table 12-1	Added table. Subsequent tables renumbered.
Section 12.2.1.3	Added subsection.
Table 12-8	Added table. Subsequent tables renumbered.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI 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 from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

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

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

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