TMS320DM646x DMSoC
ARM Subsystem

Reference Guide

Literature Number: SPRUEP9E
August 2010

Contents

List of Figures

List of Tables

Read This First

About This Manual

This document describes the ARM subsystem in the TMS320DM646x 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 TMS320DM646x 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 DM646x DMSoC, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: www.ti.com/c6000.

SPRUEP8 — *TMS320DM646x DMSoC DSP Subsystem Reference Guide.* Describes the digital signal processor (DSP) subsystem in the TMS320DM646x Digital Media System-on-Chip (DMSoC).

SPRUEQ0 — *TMS320DM646x DMSoC Peripherals Overview Reference Guide.* Provides an overview and briefly describes the peripherals available on the TMS320DM646x 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.

SPRU656 — *TMS320C6000 DSP Cache User's Guide.* Explains the fundamentals of memory caches and describes how the two-level cache-based internal memory architecture in the TMS320C621x/C671x/C64x digital signal processors (DSPs) of the TMS320C6000 DSP family can be efficiently used in DSP applications. Shows how to maintain coherence with external memory, how to use DMA to reduce memory latencies, and how to optimize your code to improve cache

efficiency. The internal memory architecture in the C621x/C671x/C64x DSPs is organized in a two-level hierarchy consisting of a dedicated program cache (L1P) and a dedicated data cache (L1D) on the first level. Accesses by the CPU to the these first level caches can complete without CPU pipeline stalls. If the data requested by the CPU is not contained in cache, it is fetched from the next lower memory level, L2 or external memory.

SPRU862 — *TMS320C64x+ DSP Cache User's Guide.* Explains the fundamentals of memory caches and describes how the two-level cache-based internal memory architecture in the TMS320C64x+ digital signal processor (DSP) of the TMS320C6000 DSP family can be efficiently used in DSP applications. Shows how to maintain coherence with external memory, how to use DMA to reduce memory latencies, and how to optimize your code to improve cache efficiency. The internal memory architecture in the C64x+ DSP is organized in a two-level hierarchy consisting of a dedicated program cache (L1P) and a dedicated data cache (L1D) on the first level. Accesses by the CPU to the these first level caches can complete without CPU pipeline stalls. If the data requested by the CPU is not contained in cache, it is fetched from the next lower memory level, L2 or external memory.

Introduction

Topic **Page**

1.1 Overview

The TMS320DM646x 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:

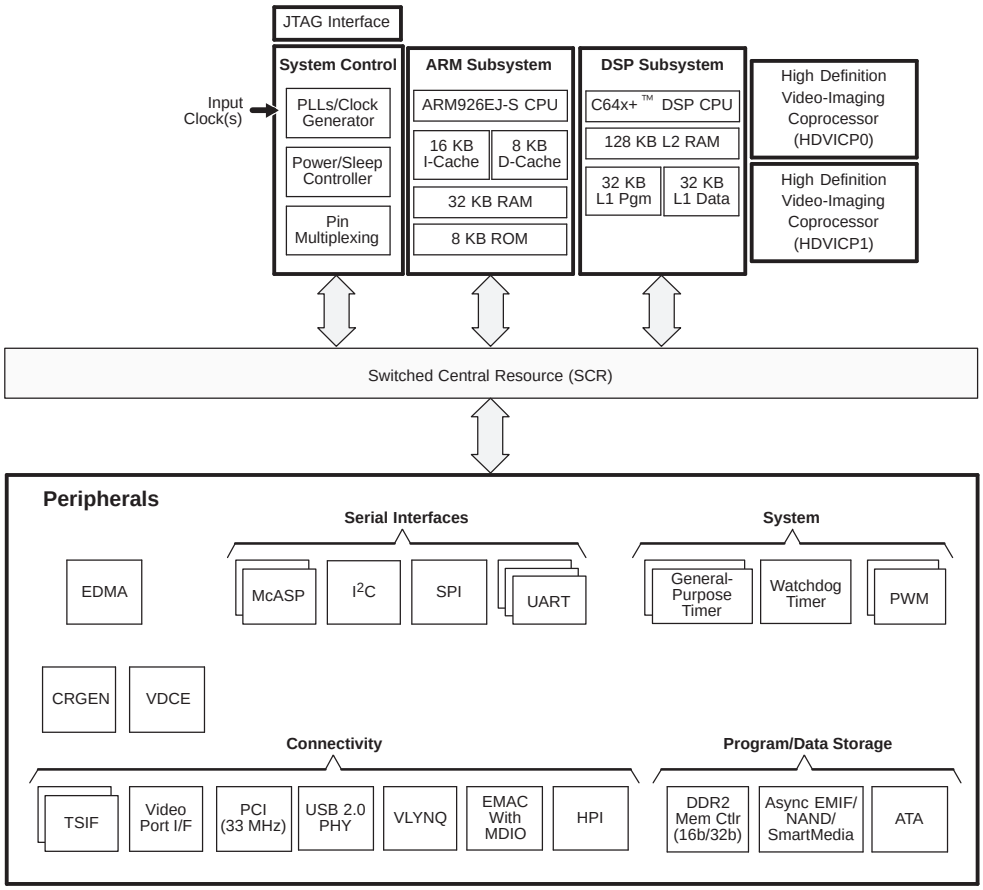
- ARM Subsystem (ARMSS), including the ARM926 RISC CPU core and associated memories
- DSP Subsystem (DSPSS), including the C64x+ DSP and associated memories
- Two programmable High-Definition Video Image Coprocessors (HDVICP)
- Video Data Conversion Engine (VDCE)
- Video Port Interface (VPIF)
- A set of I/O peripherals
- A powerful DMA Subsystem and DDR2 memory controller interface

An example block diagram (for the TMS320DM6467 DMSoC) is shown in Figure 1-1.

1.2 ARM Subsystem in TMS320DM646x 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. TMS320DM6467 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 TMS320DM646x DMSoC system. In general, the ARM is responsible for configuration and control of the overall DM646x DMSoC system, including the DSP Subsystem and a majority of the peripherals and external memories.

In the DMSoC, the ARM is responsible for handling 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 DM646x DMSoC consists of the following components:

- ARM926EJ-S RISC processor, including:
  - Co-Processor 15 (CP15)
  - MMU
  - 16KB Instruction cache and 8KB Data cache
  - Write Buffer
- ARM Internal Memories
  - 32 KB Internal RAM (32-bit wide access)
  - 8 KB Internal ROM (ARM bootloader for non-EMIFA 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:

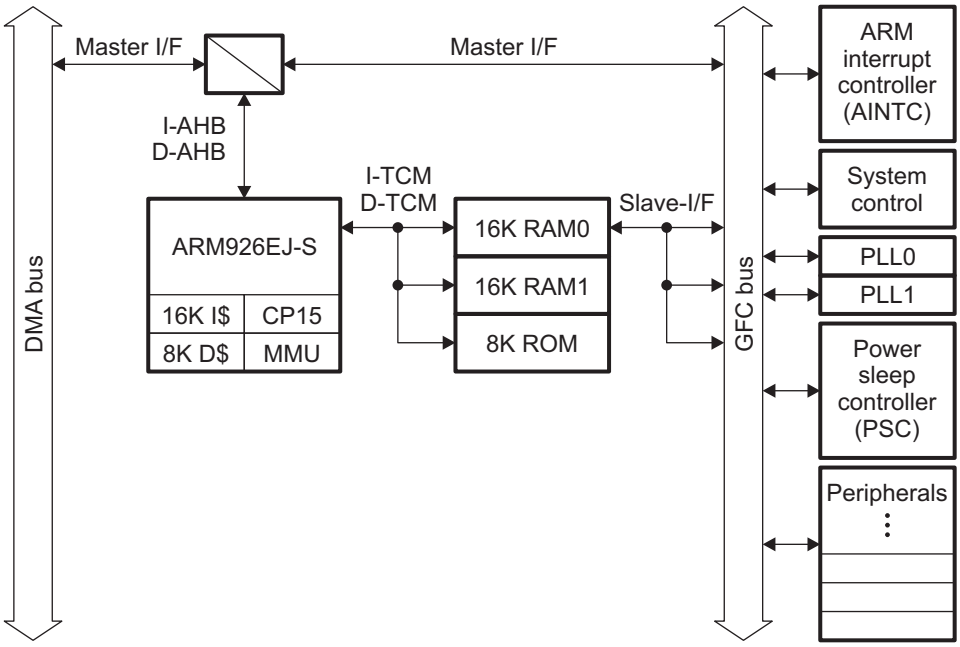
- Asynchronous EMIF (EMIFA), including the NAND flash interface
- ATA Controller
- Clock Reference Generator (CRGEN)
- DDR2 Memory Controller
- Enhanced DMA (EDMA) System - Channel Controller (CC) and Transfer Controllers (TCs)
- Ethernet Media Access Controller (EMAC)
- General-Purpose Input/Output (GPIO)
- Host Port Interface (HPI)
- Inter-IC Communication (I2C)
- Multichannel Audio Serial Port (McASP)
- Peripheral Component Interface (PCI)
- Pulse Width Modulator (PWM)
- Serial Port Interface (SPI)
- Timers
- Transport Stream Interface (TSIF)
- Universal Asynchronous Receiver/Transmitter (UART)
- Universal Serial Bus (USB) Controller
- Video Data Conversion Engine (VDCE)
- VLYNQ Interface
- Video Port Interface (VPIF)

Figure 2-1 shows the functional block diagram of the DM646x DMSoC ARM Subsystem.

The DM646x 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 read/write to peripheral registers in various modules for configuration.

Figure 2-1. TMS320DM646x DMSoC ARM Subsystem Block Diagram



2.3 References

See the following DM646x DMSoC related documents for more information:

- For related documentation about the DM646x 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
- Memory Management Unit (MMU)
- CP15 to control MMU, cache, etc.
- Java accelerator
- ARM Internal Memory
  - 32 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 32 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 DM646x 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 bits. 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 the 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 (MMU)

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-S 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 DM646x DMSoC supports both instruction TCM (I-TCM) and data TCM (D-TCM). The instruction TCM is located at 0000:0000h to 0000:9FFFh; the data TCM is located at 0001:0000h to 0001:9FFFh, as shown in Table 3-3.

Table 3-3. ITCM/DTCM Memory Map

I-TCM Address	D-TCM Address	Size (Bytes)	Description
0000:0000h - 0000:3FFFh	0001:0000h - 0001:3FFFh	16K	RAM0
0000:4000h - 0000:7FFFh	0001:4000h - 0001:7FFFh	16K	RAM1
0000:8000h - 0000:9FFFh	0001:8000h - 0001:9FFFh	8K	ROM
0000:A000h - 0000:FFFFh	0001:A000h - 0001:FFFFh	24K	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:

```
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 shown in Figure 3-1 and described in Table 3-4.

Figure 3-1. TCM Status Register

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 shown in Figure 3-2 and described in Table 3-5.

Figure 3-2. TCM Region Setup Register

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	0-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	0-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:

```
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 DM646x devices, the base address of the ITCM is 0000 0000h and the size is 32 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 6h. The memory block size field of the ITCM register c9 is read only and a write has no effect.

On DM646x devices, the base address of the DTCM is 0001 0000h and the size is 32 KB. The DM646x DTCM includes 32 KB of RAM and 8 KB of ROM. The address field of the DTCM register c9 should be programmed with 0 0010h. 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 a write has no effect.

Example 3-2. TMS320DM646x 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. TMS320DM646x DTCM Register c9 Programming

```
DTCM_BASE_ADDR          .word     0x10

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

;Create DTCM enable mask
LDR R0, DTCM_BASE_ADDR
MOV R0, R0, LSL #12
NOP
ORR R0, R0, #0x1;
ORR R0, R0, R3

; 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 TMS320DM646x 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, see the device-specific data manual.

4.1.1 ARM Internal Memories

The ARM has access to the following ARM internal memories:

- 32 KB ARM Internal RAM on TCM interface, logically separated into two 16-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

These memory interfaces are described in Section 1.1.

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:

- Asynchronous EMIF (EMIFA)
- ATA Controller
- 2 Clock Reference Generators (CRGEN)
- DDR2 Memory Controller
- Enhanced DMA (EDMA) Controller
- Ethernet Media Access Controller (EMAC)
- General-Purpose Input/Output (GPIO)
- Host Port Interface (HPI)
- Inter-IC Communication (I2C)
- 2 Multichannel Audio Serial Ports (McASP)
- Peripheral Component Interface (PCI)
- 2 Pulse Width Modulators (PWM)
- Serial Port Interface (SPI) up to 40 MHz with 2 chip selects
- 2 timers that are configurable as two 64-bit or four 32-bit timers and one 64-bit watchdog timer
- Transport Stream Interface (TSIF)
- 3 Universal Asynchronous Receiver/Transmitters (UART) (one with modem control)
- Universal Serial Bus (USB) Controller
- Video Data Conversion Engine (VDCE)

- VLYNQ Interface
- 2 Video Port Interfaces (VPIF)

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

- System Module
- PLL Controllers
- Power Sleep Controller (PSC)
- ARM Interrupt Controller (AINTC)

4.2 Memory Interfaces Overview

This section describes the different memory interfaces of DM646x DMSoC. The DM646x DMSoC supports several memory and external device interfaces, including the following:

- DDR2 synchronous DRAM
- Asynchronous EMIF/NOR/NAND Flash
- ATA

4.2.1 DDR2 Memory Controller

The DDR2 memory controller 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 DM646x 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 data conversion engine (VDCE)
- Video processing display buffers
- 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 DM646x 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 four parallel interface modules. The interface modules are:

- EMIF module - providing asynchronous EMIF (EMIFA) and NAND interfaces
- ATA – providing ATA/IDE drive support
- PCI
- HPI

Some of the control signals are configurable as GPIO signals if they are not required by the EMIFA, ATA, PCI, and HPI. See the device-specific data manual for more information on pin multiplexing.

See the device-specific data manual for more details on pin-muxing.

4.2.2.1 Asynchronous EMIF (EMIFA)

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

- The EMIFA mode supports asynchronous devices (RAM, ROM, and NOR Flash)
- 128MB asynchronous address range over 4 chip selects (32 MB 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 DM646x DMSoC ARM processor from CS2 (SRAM/NOR Flash)

4.2.2.2 NAND (NAND, SmartMedia, xD)

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

- The NAND Mode supports NAND Flash on up to 4 asynchronous chip selects
- Supports 8-bit data bus width
- 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 DM646x DMSoC ARM processor from NAND-Flash located at CS2

4.2.2.3 ATA Controller

The ATA controller provides the following capabilities:

- Supports PIO, multi-word DMA, and Ultra ATA 33/44/66/100
- Supports up to mode 4 timings on PIO mode
- Supports up to mode 2 timings on multi-word DMA
- Supports up to mode 5 timings on Ultra ATA
- Full scatter gather DMA capability
- Single channel capable of connecting up to two ATA/ATAPI devices
- Programmable timing features enable timing parameters to be reprogrammed to support any ATA timing mode at any clock frequency

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.

4.2.2.4 Peripheral Component Interface (PCI)

The PCI module allows communication with devices complaint to the *PCI Local Bus Specification* (revision 2.3) via a 32-bit address/data bus operating at speeds up to 33 MHz.

The PCI module supports the following features:

- PCI Local Bus Specification (revision 2.3) compliant
- Single function PCI interface provided
- 32-bit address/data bus width
- Operation up to 33 MHz and up to 66 MHz (for DM6467T devices only)
- Optimized burst behavior supported for system cache line sizes of 16, 32, 64 and 128 bytes
- PCI is only accessible from the ARM

4.2.2.5 Host Port Interface (HPI)

The HPI provides a parallel port interface through which an external host processor can directly access the TMS320DM646x DMSoC processor's resources (configuration and program/data memories). The external host device is asynchronous to the CPU clock and functions as a master to the HPI interface. The HPI enables a host device and the DM646x DMSoC processor to exchange information via internal or external memory. Dedicated address (HPIA) and data (HPID) registers within the HPI provide the data path between the external host interface and the processor resources. An HPI control register (HPIC) is available to the host and the CPU for various configuration and interrupt functions.

The HPI supports the following features:

- Multiplexed address/data
- Dual 16-bit halfword cycle access (internal data word is 32-bits wide)
- 16-bit-wide host data bus interface
- Internal data bursting using 8-word read and write first-in, first-out (FIFO) buffers
- HPI control register (HPIC) accessible by both the ARM CPU and the external host
- HPI address register (HPIA) accessible by both the ARM CPU and the external host
- Separate HPI address registers for read (HPIAR) and write (HPIAW) with configurable option for operating as a single HPI address register
- HPI data register (HPID)/FIFOs providing data-path between external host interface and CPU resources
- Multiple strobes and control signals to allow flexible host connection
- Asynchronous HRDY output to allow the HPI to insert wait states to the host
- Software control of data prefetching to the HPID/FIFOs
- Processor-to-Host interrupt output signal controlled by HPIC accesses
- Host-to-Processor interrupt controlled by HPIC accesses
- Register controlled HPIA and HPIC ownership and FIFO timeout
- Memory-mapped peripheral identification register (PID)
- Bus holders on host data and address buses (these are actually external to HPI module)
- 32-bit word cycle access (internal data word is 32-bits wide)
- 32-bit-wide host data bus interface

PLL Controller

Topic **Page**

5.1 PLL Module

The TMS320DM646x DMSoC has two PLL controllers that provide clocks to different parts of the system (see Figure 5-1). PLL1 provides clocks (though various dividers) to most of the components of the DMSoC. PLL2 is dedicated to the DDR2 memory controller. See the device-specific data manual for the supported input clocks.

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:

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

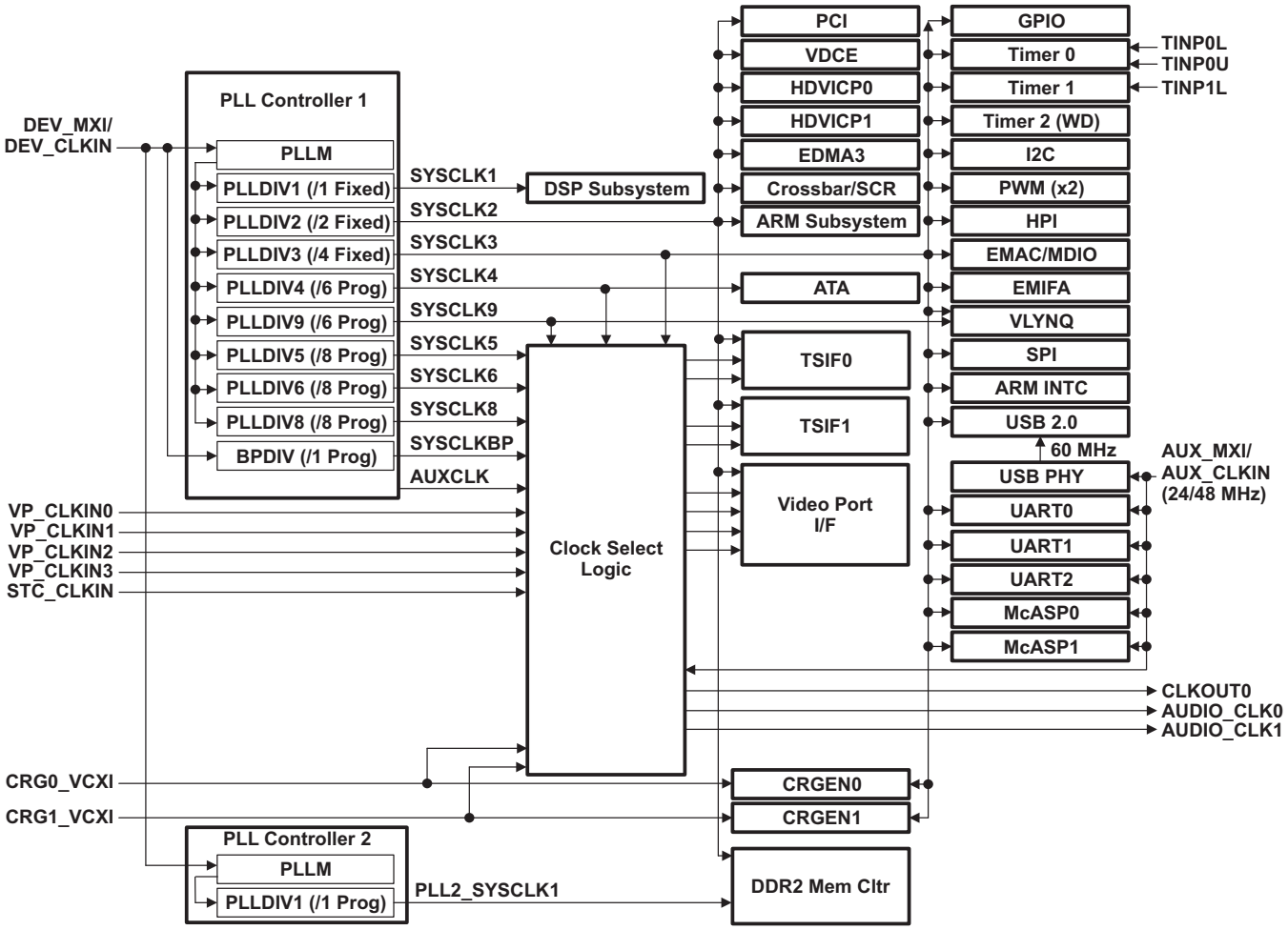
Various dividers that can be used are:

- SYSCLK Divider: D1, …, Dn
- SYSCLKBP Divider: BPDIV

Various other controls supported are:

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

Figure 5-1. PLL1 and PLL2 Clock Domain Block Diagram



5.2 PLL1 Control

PLL1 supplies the primary DM646x DMSoC system clock. Software controls the PLL1 operation through the system PLL controller 1 (PLLC1) registers (base address: 1C4 0800h). Figure 5-2 shows the customization of PLL1 in the DM646x DMSoC.

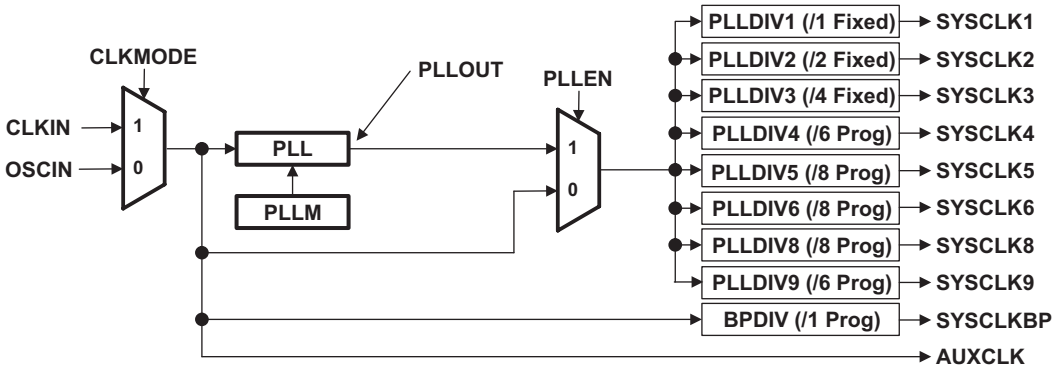
- The SYSCLK dividers are programmable (see Table 5-1).
- AUXCLK is the clock provided to the fixed clock domains.

The PLL1 multiplier is controlled by the PLLM bit in the PLL multiplier control register (PLLM) and is set to a default value of 15h at power-up, resulting in a PLL multiplier of 22×. This default setting yields a 594-MHz PLL output clock when using a 27-MHz clock source. The PLL1 multiplier may be modified by software (for example, set to 27× for a 729-MHz operation with a 27-MHz input clock source).

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 5-4.

Figure 5-2. PLL1 Structure in the TMS320DM646x DMSoC



5.2.1 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 5-1.

NOTE: The DM646x DMSoC supports multiple speed grade devices. Program SYSCLK*n* with the appropriate divider values depending on the speed of the device. See the device-specific data manual for different speed devices and corresponding SYSCLK*n* divider values.

Table 5-1. PLLC1 Output Clocks

Output Clock	Default Divider	Divider Type	Used by
SYSCLK1	/1	Fixed	DSP Subsystem
SYSCLK2	/2	Fixed	ARM Subsystem, EDMA, HDVICPs, DDR2 Memory Controller, PCI, VPIFs, TSIFs, VDCE
SYSCLK3	/4	Fixed	HPI, EMIFA, USB, VLYNQ, UARTs, McASPs, I2C, SPIs, PWMs, Timers, GPIO, EMAC, CRGEN, System Module
SYSCLK4	/6	Programmable	ATA
SYSCLK5	/8	Programmable	TSIF1
SYSCLK6	/8	Programmable	TSIF2
SYSCLK8	/8	Programmable	VPIF2
SYSCLK9	/6	Programmable	VLYNQ
SYSCLKBP	DEV_CLKIN/n	Programmable	TSIFs
AUXCLK	DEV_CLKIN	Fixed at DEV_CLKIN	TSIFs, VPIFs

5.2.2 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 5.2.2.1 to initialize the PLL.
- If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), follow the sequence in Section 5.2.2.2 to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and only the SYSCLK dividers need to be changed, follow the sequence in Section 5.3.2.4.

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

5.2.2.1 Initialization to PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), follow this procedure to change the PLL1 frequencies. The recommendation is to stop all peripheral operation before changing the PLL1 frequency, with the exception of the ARM and DDR2 memory controller. The ARM must be operational to program the PLL controller. The DDR2 memory controller 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 20 MXI clock cycles to ensure PLLC switches to bypass mode properly.
3. Set the PLLRST bit in PLLCTL to 1 (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. (4096 MXI clock cycles)
8. Program the required multiplier value in the PLL multiplier control register (PLLM).
9. Wait for PLL to reset properly. The PLL reset time is a minimum of 32 MXI clock cycles.
10. Clear the PLLRST bit in PLLCTL to 0 to bring the PLL out of reset.
11. Wait for 2000 MXI clock or reference clock cycles to allow PLL to lock.
12. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

5.2.2.2 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 5.2.2.1), 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 ARM and DDR2 memory controller. The ARM must be operational to program the PLL controller. The DDR2 memory controller 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 20 MXI clock cycles to ensure PLLC switches to bypass mode properly.
2. Set the PLLRST bit in PLLCTL to 1 (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 the PLL multiplier control register (PLLM).
5. Wait for PLL to reset properly. The PLL reset time is a minimum of 32 MXI clock cycles.
6. Clear the PLLRST bit in PLLCTL to 0 to bring the PLL out of reset.
7. Wait for 2000 MXI clock or reference clock cycles to allow PLL to lock.
8. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

5.2.2.3 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 the PLL controller status register (PLLSTAT) to clear to 0 to indicate that no GO operation is currently in progress.
2. Program the RATIO field in the PLL controller divider *n* register (PLLDIV4-PLLDIV6, PLLDIV8, PLLDIV9) with the desired divide factor.
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).

NOTE: The DM646x DMSoC supports multiple speed grade devices. Program SYSCLK*n* with the appropriate divider values depending on the speed of the device. See the device-specific data manual for different speed devices and corresponding SYSCLK*n* divider values.

5.3 PLL2 Control

PLL2 provides the clock from which the DDR2 memory controller clock is derived. 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 for DDR2. PLL2 must be configured to output a 2x clock to the DDR2 PHY interface.

The DDR2 PLL controller (PLLC2) controls PLL2, which accepts the clock from the oscillator and generates the various frequency clocks needed for the DDR2 memory controller. Figure 5-3 shows the customization of PLL2 in the DM646x DMSoC.

- The SYSCLK divider is programmable.
- AUXCLK and SYSCLKBP are not used.

PLL2 supplies the DDR2 memory controller clock. Software controls PLL2 operation through the 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 memory controller interface for best performance).

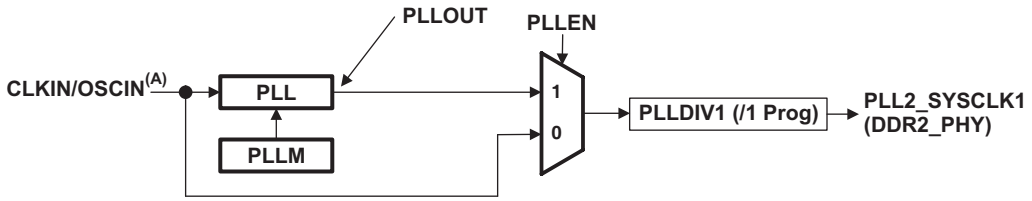
The PLL2 output clock must be divided-down to the DDR2 memory controller operating range.

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 DDR2 memory controller 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 5-4.

NOTE: PLLDIV1 defaults to /1 at reset and can be modified after reset. PLLDIV2 through PLLDIV9 are not supported on PLL2.

Figure 5-3. PLL2 Structure in TMS320DM646x DMSoC



(A) As selected by the PLL2 PLLCTL register

5.3.1 Device Clock Generation

PLLC2 generates the clock from the PLL2 output clock for use by the DDR2 memory controller, as shown in Table 5-2.

The SYSCLK1 output clock divider value defaults to /1. It can be modified by software (using the RATIO bit in PLLDIV1) in combination with other PLL multipliers to achieve the desired DDR2 memory controller clock rate.

Table 5-2. PLLC2 Output Clock

Output Clock	Default Divider	Divider Type	Used by
SYSCLK1	/1	Programmable	DSP Subsystem

5.3.2 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 5.3.2.2 to initialize the PLL.
- If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), follow the sequence in Section 5.3.2.3 to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and only the SYSCLK divider needs to be changed, follow the sequence in Section 5.3.2.4.

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. The DDR2 memory controller requires a special sequence to be followed before and after changing the PLL2 frequency. Follow the additional considerations for the DDR2 memory controller in Section 5.3.2.1, in order to not corrupt DDR2 memory controller operation.

5.3.2.1 DDR2 Memory Controller Considerations When Modifying PLL2 Frequency

Before changing PLL2 and/or PLLC2 frequency, take the DDR2 memory controller requirements into account. 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 memory controller operation.

- If the DDR2 memory controller is in reset when you desire to change the PLL2 frequency, follow the steps in Section 5.3.2.1.1.
- If the DDR2 memory controller is already out of reset when you desire to change the PLL2 frequency, follow the steps in Section 5.3.2.1.2.

5.3.2.1.1 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 ($\overline{\text{POR}}$), warm reset ($\overline{\text{RESET}}$), and max reset.

1. Leave the DDR2 memory controller in reset.
2. Program the PLL2 clocks by following the steps in the appropriate section: Section 5.3.2.2, Section 5.3.2.3, or Section 5.3.2.4. (Discussion in Section 5.3.2 explains which is the appropriate subsection).
3. Initialize the DDR2 memory controller. The steps for DDR2 memory controller initialization are found in the *TMS320DM646x DMSoC DDR2 Memory Controller User's Guide* (SPRUEQ4).

5.3.2.1.2 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 *TMS320DM646x DMSoC DDR2 Memory Controller User's Guide* (SPRUEQ4).
3. Program the PLL2 clocks by following the steps in the appropriate section: Section 5.3.2.2, Section 5.3.2.3, or Section 5.3.2.4. (Discussion in Section 5.3.2 explains which is the appropriate subsection).
4. Re-enable the DDR2 memory controller clock. The DDR2 memory controller clock on sequence is in the *TMS320DM646x DMSoC DDR2 Memory Controller User's Guide* (SPRUEQ4).

5.3.2.2 Initialization to PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), follow this procedure 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 20 MXI clock cycles to ensure PLLC switches to bypass mode properly.
3. Set the PLLRST bit in PLLCTL to 1 (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. (4096 MXI clock cycles)
8. Program the required multiplier value in the PLL multiplier control register (PLLM).
9. If necessary, program the PLL controller divider 1 register (PLLDIV1) to change the SYSCLK1 divide value:
   (a) Program the RATIO field in PLLDIV1 with the desired divide factor.
   (b) Set the GOSET bit in PLLCMD to 1 to initiate a new divider transition.
   (c) Wait for the GOSTAT bit in the PLL controller status register (PLLSTAT) to clear to 0 (completion of phase alignment).
10. Wait for PLL to reset properly. The PLL reset time is a minimum of 32 MXI clock cycles.
11. Clear the PLLRST bit in PLLCTL to 0 to bring the PLL out of reset.
12. Wait for 2000 MXI clock or reference clock cycles to allow PLL to lock.
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 *TMS320DM646x DMSoC DDR2 Memory Controller User's Guide* (SPRUEQ4).

5.3.2.3 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 5.3.2.2), 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 20 MXI clock cycles to ensure PLLC switches to bypass mode properly.
2. Set the PLLRST bit in PLLCTL to 1 (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 the PLL multiplier control register (PLLM).
5. If necessary, program the PLL controller divider 1 register (PLLDIV1) to change the SYSCLK1 divide value:
   (a) Program the RATIO field in PLLDIV1 with the desired divide factor.
   (b) Set the GOSET bit in PLLCMD to 1 to initiate a new divider transition.
   (c) Wait for the GOSTAT bit in the PLL controller status register (PLLSTAT) to clear to 0 (completion of phase alignment).
6. Wait for PLL to reset properly. The PLL reset time is a minimum of 32 MXI clock cycles.
7. Clear the PLLRST bit in PLLCTL to 0 to bring the PLL out of reset.
8. Wait for 2000 MXI clock or reference clock cycles to allow PLL to lock.
9. Set the PLLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

5.3.2.4 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 the PLL controller status register (PLLSTAT) to clear to 0 to indicate that no GO operation is currently in progress.
2. Program the RATIO field in the PLL controller divider 1 register (PLLDIV1) with the desired divide factor.
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).

NOTE: The DM646x DMSoC supports multiple speed grade devices. Program SYSCLK*n* with the appropriate divider values depending on the speed of the device. See the device-specific data manual for different speed devices and corresponding SYSCLK*n* divider values.

5.4 PLL Controller Registers

Table 5-3. PLL and Reset Controller Module Instance Table

Instance ID	Base Address	End Address	Size
0	1C4 0800h	1C4 0BFFh	400h
1	1C4 0C00h	1C4 0FFFh	400h

Table 5-4 lists the memory-mapped registers for the PLL and Reset Controller. See the device-specific data manual for the memory address of these registers.

Table 5-4. PLL and Reset Controller Registers

Offset	Acronym	Register Description	Section
00h	PID	Peripheral ID Register	Section 5.4.1
E4h	RSTYPE	Reset Type Status Register	Section 5.4.2
100h	PLLCTL	PLL Control Register	Section 5.4.3
110h	PLLM	PLL Multiplier Control Register	Section 5.4.4
118h	PLLDIV1	PLL Controller Divider 1 Register	Section 5.4.5
11Ch	PLLDIV2(1)	PLL Controller Divider 2 Register	Section 5.4.6
120h	PLLDIV3(1)	PLL Controller Divider 3 Register	Section 5.4.7
12Ch	BPDIV(1)	Bypass Divider Register	Section 5.4.8
138h	PLLCMD	PLL Controller Command Register	Section 5.4.9
13Ch	PLLSTAT	PLL Controller Status Register	Section 5.4.10
140h	ALNCTL	PLL Controller Clock Align Control Register	Section 5.4.11
144h	DCHANGE	PLLDIV Ratio Change Status Register	Section 5.4.12
148h	CKEN	Clock Enable Control Register	Section 5.4.13
14Ch	CKSTAT	Clock Status Register	Section 5.4.14
150h	SYSTAT	SYSCLK Status Register	Section 5.4.15
160h	PLLDIV4(1)	PLL Controller Divider 4 Register	Section 5.4.16
164h	PLLDIV5(1)	PLL Controller Divider 5 Register	Section 5.4.16
168h	PLLDIV6(1)	PLL Controller Divider 6 Register	Section 5.4.16
170h	PLLDIV8(1)	PLL Controller Divider 8 Register	Section 5.4.16
174h	PLLDIV9(1)	PLL Controller Divider 9 Register	Section 5.4.16

(1) For PLL1 only, not supported for PLL2

5.4.1 Peripheral ID Register (PID)

The peripheral ID register (PID) is shown in Figure 5-4 and described in Table 5-5.

Figure 5-4. 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 5-5. 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

5.4.2 Reset Type Status Register (RSTYPE)

The reset type status register (RSTYPE) is shown in Figure 5-5 and described in Table 5-6. 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 5-5. 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 5-6. 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 ($\overline{\text{RESET}}$) was the last reset to occur that is of highest priority.
0	POR	0-1	Power-on reset. If 1, the power-on reset ($\overline{\text{POR}}$) was the last reset to occur that is of highest priority.

5.4.3 PLL Control Register (PLLCTL)

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

Figure 5-6. 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

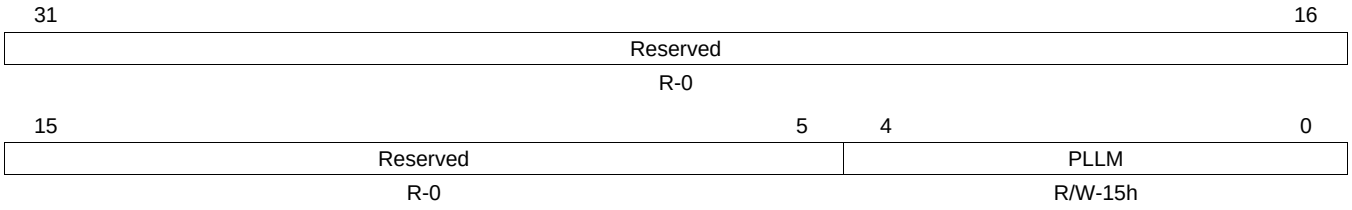
Table 5-7. PLL Control Register (PLLCTL) Field Descriptions

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 is de-asserted.
		1	PLL disable is 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

5.4.4 PLL Multiplier Control Register (PLLM)

The PLL multiplier control register (PLLM) is shown in Figure 5-7 and described in Table 5-8.

Figure 5-7. PLL Multiplier Control Register (PLLM)



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

Table 5-8. PLL Multiplier Control Register (PLLM) Field Descriptions

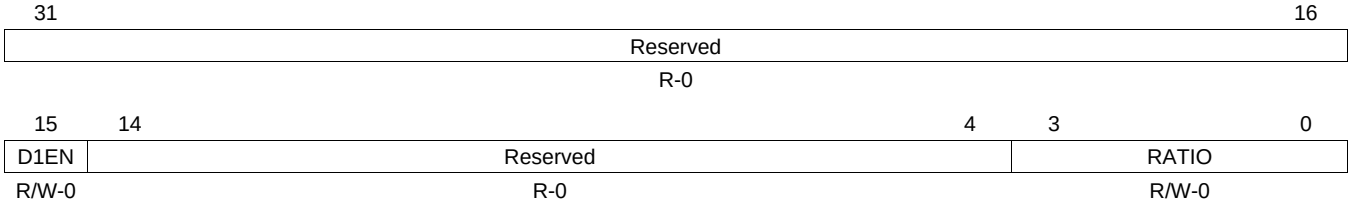
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PLLM	D-1Fh	PLL Multiplier select. Multiplier Value = PLLM + 1. The valid range of multiplier values for a given MXI/CLKIN is defined by the minimum and maximum frequency limits on the PLL VCO frequency. See the device-specific data manual for PLL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

5.4.5 PLL Controller Divider 1 Register (PLLDIV1)

The PLL controller divider 1 register (PLLDIV1) is shown in Figure 5-8 and described in Table 5-9. Divider 1 controls the divider for SYSCLK1.

NOTE: For PLL1, the SYSCLK1 divider value is fixed and is not programmable; for PLL2, the SYSCLK1 divider value is programmable.

Figure 5-8. PLL Controller Divider 1 Register (PLLDIV1)



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

Table 5-9. PLL Controller Divider 1 Register (PLLDIV1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN		Divider enable for SYSCLK1.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

5.4.6 PLL Controller Divider 2 Register (PLLDIV2)

The PLL controller divider 2 register (PLLDIV2) is shown in Figure 5-9 and described in Table 5-10. Divider 2 controls the divider for SYSCLK2. PLLDIV2 is not supported for PLL2.

NOTE: The SYSCLK2 divider value is fixed and is not programmable.

Figure 5-9. 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	

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

Table 5-10. PLL Controller Divider 2 Register (PLLDIV2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN		Divider enable for SYSCLK2.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL1 divide by 2).

5.4.7 PLL Controller Divider 3 Register (PLLDIV3)

The PLL controller divider 3 register (PLLDIV3) is shown in Figure 5-10 and described in Table 5-11. Divider 3 controls the divider for SYSCLK3. PLLDIV3 is not supported for PLL2.

NOTE: The SYSCLK3 divider value is fixed and is not programmable.

Figure 5-10. 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-3h	

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

Table 5-11. PLL Controller Divider 3 Register (PLLDIV3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN		Divider enable for SYSCLK3.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 3h (PLL divide by 4).

5.4.8 Bypass Divider Register (BPDIV)

The bypass divider register (BPDIV) is shown in Figure 5-11 and described in Table 5-12. Bypass divider controls the divider for SYSCLKBP. BPDIV is not supported for PLL2.

Figure 5-11. Bypass Divider Register (BPDIV)

31	16
Reserved	
R-0	

15	14	5	4	0
BPDEN	Reserved		RATIO	
R/W-1	R-0		R/W-0	

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

Table 5-12. 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. RATIO defaults to 0 (PLL divide by 1).

5.4.9 PLL Controller Command Register (PLLCMD)

The PLL controller command register (PLLCMD) is shown in Figure 5-12 and described in Table 5-13. Contains command bits for various operations. Writes of 1 initiate command; writes of 0 clear the bit, but have no effect.

Figure 5-12. 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

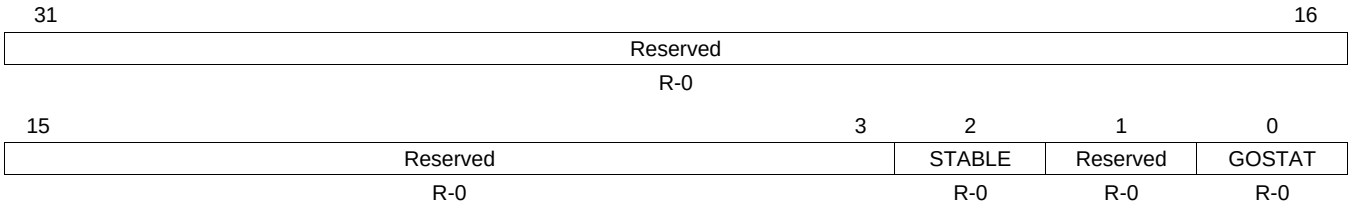
Table 5-13. PLL Controller Command Register (PLLCMD) Field Descriptions

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

5.4.10 PLL Controller Status Register (PLLSTAT)

The PLL controller status register (PLLSTAT) is shown in Figure 5-13 and described in Table 5-14.

Figure 5-13. PLL Controller Status Register (PLLSTAT)



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

Table 5-14. 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.

5.4.11 PLL Controller Clock Align Control Register (ALNCTL)

The PLL controller clock align control register (ALNCTL) is shown in Figure 5-14 and described in Table 5-15. Indicates which SYSCLKs need to be aligned for proper device operation.

Figure 5-14. PLL Controller Clock Align Control Register (ALNCTL)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		ALN9[1]	ALN8[1]	Rsvd	ALN6[1]	ALN5[1]	ALN4[1]	ALN3[1]	ALN2[1]	ALN1
R-0		R/W-1	R/W-1	R-0	R/W-1	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

(1) For PLL1 only, not supported for PLL2.

Table 5-15. PLL Controller Clock Align Control Register (ALNCTL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	ALN9		SYSCLK9 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
7	ALN8		SYSCLK8 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
6	Reserved	0	Reserved
5	ALN6		SYSCLK6 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
4	ALN5		SYSCLK5 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
3	ALN4		SYSCLK4 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
2	ALN3		SYSCLK3 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
1	ALN2		SYSCLK2 needs to be aligned to others selected in this register. For PLL1 only, not supported for PLL2.
		0	No
		1	Yes
0	ALN1		SYSCLK1 needs to be aligned to others selected in this register.
		0	No
		1	Yes

5.4.12 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLDIV ratio change status register (DCHANGE) is shown in Figure 5-15 and described in Table 5-16. Indicates if SYSCLK*n* divide ratio has been modified.

NOTE: The DM646x DMSoC supports multiple speed grade devices. Program SYSCLK*n* with the appropriate divider values depending on the speed of the device. See the device-specific data manual for different speed devices and corresponding SYSCLK*n* divider values.

Figure 5-15. PLLDIV Ratio Change Status Register (DCHANGE)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		SYS9[1]	SYS8[1]	Rsvd	SYS6[1]	SYS5[1]	SYS4[1]	SYS3[1]	SYS2[1]	SYS1
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

(1) For PLL1 only, not supported for PLL2.

Table 5-16. PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	SYS9		SYSCLK9 divide ratio is modified. SYSCLK9 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK9 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK9 changes to the new divide ratio.
7	SYS8		SYSCLK8 divide ratio is modified. SYSCLK8 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK8 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK8 changes to the new divide ratio.
6	Reserved	0	Reserved
5	SYS6		SYSCLK6 divide ratio is modified. SYSCLK6 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK6 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK6 changes to the new divide ratio.
4	SYS5		SYSCLK5 divide ratio is modified. SYSCLK5 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK5 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK5 changes to the new divide ratio.
3	SYS4		SYSCLK4 divide ratio is modified. SYSCLK4 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK4 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK4 changes to the new divide ratio.

Table 5-16. PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions (continued)

Bit	Field	Value	Description
2	SYS3		SYSCLK3 divide ratio is modified. SYSCLK3 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK3 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK3 changes to the new divide ratio.
1	SYS2		SYSCLK2 divide ratio is modified. SYSCLK2 divide ratio is changed during GO operation. For PLL1 only, not supported for PLL2.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK2 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK2 changes to the new divide ratio.
0	SYS1		SYSCLK1 divide ratio is modified. SYSCLK1 divide ratio is changed during GO operation.
		0	Ratio has not been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK1 is not affected.
		1	Ratio has been modified. When the GOSET bit in the PLL controller command register (PLLCMD) is set, SYSCLK1 changes to the new divide ratio.

5.4.13 Clock Enable Control Register (CKEN)

The clock enable control register (CKEN) is shown in Figure 5-16 and described in Table 5-17. Clock enable control for miscellaneous output clocks. CKEN is not used on PLL2.

Figure 5-16. 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 5-17. Clock Enable Control Register (CKEN) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	AUXEN		AUXCLK enable. The actual clock status is shown in the clock status register (CKSTAT).
		0	Clock is disabled.
		1	Clock is enabled.

5.4.14 Clock Status Register (CKSTAT)

The clock status register (CKSTAT) is shown in Figure 5-17 and described in Table 5-18. Clock status for all clocks, except SYSCLK*n*. CKSTAT is not used on PLL2.

Figure 5-17. 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 5-18. 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

5.4.15 SYSCLK Status Register (SYSTAT)

The SYSCLK status register (SYSTAT) is shown in Figure 5-18 and described in Table 5-19. Indicates SYSCLK*n* status. Actual default is determined by the actual clock status, which depends on the D*n*EN bit in the PLL controller divider *n* register (PLLDIV*n*).

Figure 5-18. SYSCLK Status Register (SYSTAT)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		SYS9ON(1)	SYS8ON(1)	Rsvd	SYS6ON(1)	SYS5ON(1)	SYS4ON(1)	SYS3ON(1)	SYS2ON(1)	SYS1ON
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

(1) For PLL1 only, not supported for PLL2.

Table 5-19. SYSCLK Status Register (SYSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	SYS9ON		SYSCLK9 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
7	SYS8ON		SYSCLK8 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
6	Reserved	0	Reserved
5	SYS6ON		SYSCLK6 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
4	SYS5ON		SYSCLK5 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
3	SYS4ON		SYSCLK4 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
2	SYS3ON		SYSCLK3 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
1	SYS2ON		SYSCLK2 on status. For PLL1 only, not supported for PLL2.
		0	Off
		1	On
0	SYS1ON		SYSCLK1 on status.
		0	Off
		1	On

5.4.16 PLL Controller Divider *n* Registers (PLLDIV4-PLLDIV6, PLLDIV8, PLLDIV9)

The PLL controller divider *n* register (PLLDIV*n*) is shown in Figure 5-19 and described in Table 5-20. PLLDIV4 controls the divider for SYSCLK4, PLLDIV5 controls the divider for SYSCLK5, PLLDIV6 controls the divider for SYSCLK6, PLLDIV8 controls the divider for SYSCLK8, and PLLDIV9 controls the divider for SYSCLK9. PLLDIV4-PLLDIV6, PLLDIV8, PLLDIV9 are not supported for PLL2.

NOTE: The DM646x DMSoC supports multiple speed grade devices. Program SYSCLK*n* with the appropriate divider values depending on the speed of the device. See the device-specific data manual for different speed devices and corresponding SYSCLK*n* divider values.

Figure 5-19. PLL Controller Divider *n* Register (PLLDIV*n*)

31		16
Reserved		
R-0		

15	14	4	3	0
D*n*EN	Reserved		RATIO	
R/W-1	R-0		R/W-5h or 7h[1]	

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

(1) For PLLDIV4 and PLLDIV9, RATIO defaults to 5 (PLL divide by 6); for PLLDIV5, PLLDIV6, and PLLDIV8, RATIO defaults to 7 (PLL divide by 8).

Table 5-20. PLL Controller Divider *n* Register (PLLDIV*n*) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D*n*EN		Divider enable for SYSCLK*n*.
		0	Disable
		1	Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1.

Power and Sleep Controller (PSC)

Topic **Page**

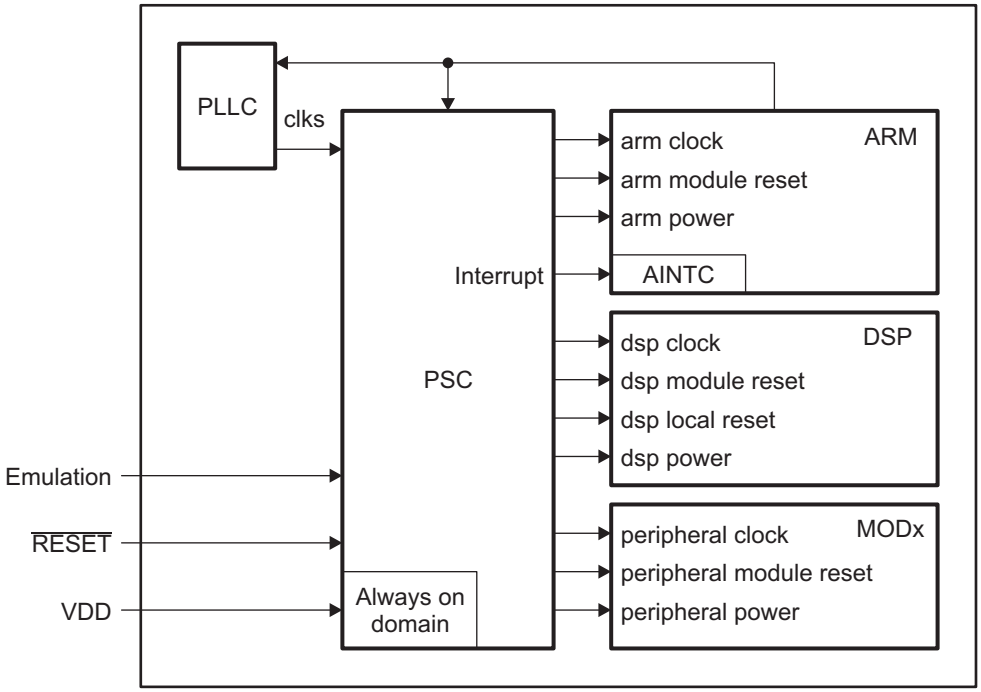
6.1 Introduction

In the TMS320DM646x DMSoC system, the Power and Sleep Controller (PSC) is responsible for managing transitions of clock on/off and reset. A block diagram of the PSC is shown in Figure 6-1. Many of the PSC operations are transparent to software, such as hard reset operations. However, the PSC provides you with an interface to control several important clock and reset operations. The clock and reset operations are described in this chapter.

The PSC includes the following features:

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

Figure 6-1. TMS320DM646x DMSoC Power and Sleep Controller (PSC)



6.2 Power Domain and Module Topology

The DM646x DMSoC system includes only one power domain (Always On) and multiple separate modules, as shown in Figure 6-2 and summarized in Table 6-1. The Always On power domain is always in the ON state when the chip is powered-on. The Always On domain is powered by the CV_{DD} pins of the DM646x DMSoC (see the device-specific data manual). All of the DM646x DMSoC modules reside within the Always On power domain. See the device-specific data manual for details on the clock domains.

Table 6-1 shows the state of each module after chip power-on/hard reset. The default state of the DSP module is determined by the DSP boot source (DSPBOOT) pin. If the DSP is selected to self-boot at reset via this signal, the DSP module is enabled by default. The ARM, Timer2, and System Module are always enabled.

Figure 6-2. TMS320DM646x DMSoC Power Domain and Module Topology

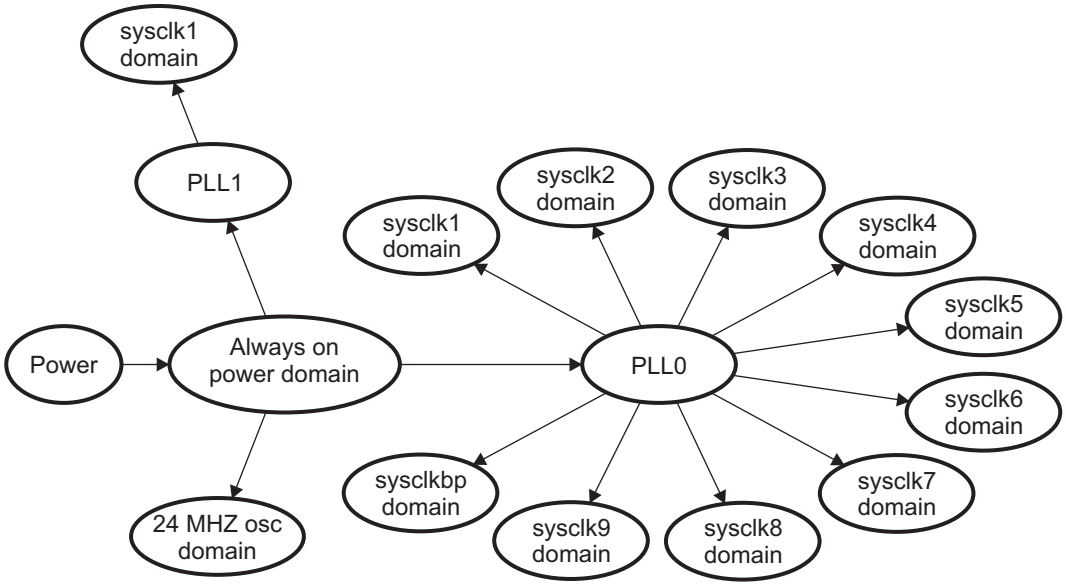


Table 6-1. Module Configuration

LPSC Number	Module Name	Default State	
		Module State [STATE bit in MDSTAT*n* Register]	**Local Reset State [LRST bit in MDSTAT*n* Register]**
0	ARM	Enable [3h]	-
1	DSP C64x+	If DSPBOOT = 0, then Enable [3h] If DSPBOOT = 1, then Enable [3h]	Asserted [0] Deasserted [1]
2	HDVICP0	SwRstDisable [0]	Asserted [0]
3	HDVICP1	SwRstDisable [0]	Asserted [0]
4	EDMA3CC	SwRstDisable [0]	-
5	EDMA3TC0	SwRstDisable [0]	-
6	EDMA3TC1	SwRstDisable [0]	-
7	EDMA3TC2	SwRstDisable [0]	-
8	EDMA3TC3	SwRstDisable [0]	-
9	USB2.0	SwRstDisable [0]	-
10	ATA	SwRstDisable [0]	-
11	VLYNQ	SwRstDisable [0]	-
12	HPI	SwRstDisable [0]	-
13	PCI	SwRstDisable [0]	-
14	EMAC/MDIO	SwRstDisable [0]	-
15	VDCE	SwRstDisable [0]	-
16-17	Video Port(1)	SwRstDisable [0]	-
18	TSIF0	SwRstDisable [0]	-
19	TSIF1	SwRstDisable [0]	-
20	DDR2 Memory Contoller	SwRstDisable [0]	-
21	EMIFA	If BTMODE[3:0] ≠ 0100 and DSPBOOT = 0, then SwRstDisable [0] If BTMODE[3:0] = 0100 or DSPBOOT = 1, then Enable [3h]	-
22	McASP0	SwRstDisable [0]	-
23	McASP1	SwRstDisable [0]	-
24	CRGEN0	SwRstDisable [0]	-
25	CRGEN1	SwRstDisable [0]	-
26	UART0	SwRstDisable [0]	-
27	UART1	SwRstDisable [0]	-
28	UART2	SwRstDisable [0]	-
29	PWM0	SwRstDisable [0]	-
30	PWM1	SwRstDisable [0]	-
31	I2C	SwRstDisable [0]	-
32	SPI	SwRstDisable [0]	-
33	GPIO	SwRstDisable [0]	-
34	Timer0	SwRstDisable [0]	-
35	Timer1	SwRstDisable [0]	-
36-44	Reserved	Reserved	-
45	ARM INTC	Enable [3h]	-

(1) The video port has a total of 5 clock inputs that can be controlled by LPSC. One LPSC can support a maximum of 4 clocks, so two LPSCs are assigned for the video port. Both video port LPSCs should be set to the same state.

6.2.1 Power Domain States

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

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

In the DM646x DMSoC, there is only one power domain (Always On) .The Always On power domain is always in the ON state when the chip is powered-on.

6.2.2 Module States

A module can be in one of four states: Disable, Enable, SyncReset, or SwRstDisable, as defined by the STATE bit in the module status *n* register (MDSTAT*n*) in the PSC. These four states correspond to combinations of module reset asserted or deasserted and module clock on or off, as shown in Table 6-2.

NOTE: Module Reset is defined to completely reset a given module, so that all hardware returns to its default state. See Chapter 10 for more information on module reset.

For more information on the DM646x DMSoC power management, see Chapter 7.

Table 6-2. Module States

STATE Bit in MDSTAT*n*	Module State	Module Reset	Module Clock	Module State Definition
0	SwRstDisable	Asserted	Off	A module in the SwResetDisable state has its module reset asserted and it has its clock set to off. After initial power-on, most modules are in the SwRstDisable state by default (see Table 6-1). Generally, software is not expected to initiate this state.
1	SyncReset	Asserted	On	A module in the SyncReset state has its module reset asserted and it has its clock on. Generally, software is not expected to initiate this state.
2h	Disable	Deasserted	Off	A module in the Disable state has its module reset deasserted and it has its clock off. This state is typically used for disabling a module clock to save power. The DM646x 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.
3h	Enable	Deasserted	On	A module in the Enable state has its module reset deasserted and it has its clock on. This is the normal run-time state for a given module.

6.2.3 DSP Local Reset

In addition to module reset (as described in Section 6.2.2), 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 would the DSP module reset. 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 Chapter 10 and Chapter 12 for more information on local reset, as well as DSP boot.

The procedures for asserting and deasserting DSP local reset are:

1. Clear the LRST bit in the module control 1 register (MDCTL1) in the PSC to 0. This asserts the DSP local reset. By default, after power-on reset or hard reset, the DSP boot source (DSPBOOT) pin determines the default state of the LRST bit in MDCTL1. See Chapter 10 for more information on this boot configuration pin.
2. Set the LRST bit in MDCTL1 to 1. This deasserts the DSP local reset. After reset is deasserted, if the DSP is in the Enable state, the DSP immediately begins code execution from the boot address programmed in the DSP boot address register (DSPBOOTADDR) in the System Module.

6.3 Executing Module State Transitions

The procedure for module state transitions is as follows (*n* corresponds to the module number, see Table 6-1 for the module numbers):

1. Wait for the GOSTAT[0] bit in the power domain transition status register (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 the module control *n* register (MDCTL*n*) to SwRstDisable (0), SyncReset (1), Disable (2h), or Enable (3h).

   NOTE: You may set transitions in multiple NEXT bits in multiple MDCTL*n* in this step. For the video port, the NEXT bits should be set for both video port LPSCs (MDCTL16 and MDCTL17) in this step.

3. This is a special step required for these specific modules. This step is not required for any module that is not listed. Set the EMURSTIE bit in MDCTL*n* to 1, if the module you want to transition is any of the following:
   - EMAC
   - USB
   - ATA
   - VLYNQ
   - DDR2 memory controller
   - Asynchronous EMIF
   - McASP
   - GPIO

   NOTE: The EMURSTIE bit in MDCTL*n* is also used for PSC emulation features. The emulation features are described in Section 6.4.

4. Set the GO[0] bit in the power domain transition command register (PTCMD) to 1 to initiate the transition(s).
5. Wait for the GOSTAT[0] bit in PTSTAT to clear to 0. The module is safely in the new state only after the GOSTAT[0] bit in PTSTAT is cleared to 0.
6. Wait for the STATE bit in the module status *n* register (MDSTAT*n*) to change to the required state, SwRstDisable (0), SyncReset (1), Disable (2h), or Enable (3h), that was set in MDCTL*n*.

   NOTE: You need to wait for the STATE bit in multiple MDSTAT*n* to change to the required state of all modules that were set in step 2. For the video port, wait for the STATE bit in MDSTAT16 and MDSTAT17 to change to the required state.

   To put the DSP in the software reset disable (SwRstDisable) state, see Section 12.5.3.2.1; in the synchronous reset (SyncReset) state, see Section 12.5.3.2.2; in the DSP module clock off (Disable) state, see Section 12.5.2.2; in the DSP module clock on (Enable) state, see Section 12.5.2.1.

6.4 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 6-3. 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: In the DM646x DMSoC, there is only one power domain (Always On), so all IcePick commands do not have any control on the power domain and only control module states and resets.

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

6.5 PSC Interrupts

The PSC has one interrupt (Table 6-4) to the ARM interrupt controller (AINTC). The PSC interrupt is generated when certain IcePick emulation events occur.

Table 6-4. PSC Interrupt

ARM Event	Acronym	Source
47	PSCINT	PSC

6.5.1 Interrupt Events

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

- Module State Emulation Event
- Module Local Reset Emulation Event

These interrupt events are summarized in Table 6-5 and described in more detail in this section.

Table 6-5. PSC Interrupt Events

Interrupt Enable Bits		
Control Register	Enable Bit	Interrupt Condition
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

6.5.1.1 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 module status *n* register (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 the module is not already in the enable state

6.5.1.2 Module Local Reset Emulation Events

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

- When assert reset is asserted by emulation although software deasserted 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

6.5.2 Interrupt Register Bits

The PSC interrupt enable bits are the EMUIHBIE bit and the EMURSTIE bit in the module control *n* register (MDCTL*n*).

NOTE: To interrupt the ARM, the ARM's power and sleep controller interrupt (PSCINT) must also be enabled in the ARM interrupt controller (AINTC). See Chapter 8 for more information on the ARM interrupt controller.

The PSC interrupt status bits are the M[*n*] bits in the module error pending register *n* (MERRPR*n*), and the EMUIHB bit and the EMURST bit in the module status *n* register (MDSTAT*n*). The status bits in MERRPR0 and MERRPR1 are read by software to determine which module has generated an emulation interrupt, and then software can read the corresponding status bits in MDSTAT*n* to determine which event caused the interrupt.

The PSC interrupt clear bits are the M[*n*] bits in the module error clear register *n* (MERRCR*n*).

The PSC interrupt evaluation bit is the ALLEV bit in the interrupt evaluation register (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 reasserted 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 6.6 for complete descriptions of all PSC registers.

6.5.3 Interrupt Handling

Handle the PSC interrupts as described in the following procedure:

Enable the interrupt.

1. Set the EMUIHBIE bit and the EMURSTIE bit in the module control *n* register (MDCTL*n*) to enable the interrupt events that you want.
2. Enable the ARM 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 8 for more information.

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

1. Read the M[*n*] bit in the module error pending register *n* (MERRPR*n*) to determine the source of the interrupt(s).
2. For each active event that you want to service:
   - Read the event status bits in the module status *n* register (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 a 1 to the M[*n*] bit in the module error clear register *n* (MERRCR*n*) to clear corresponding status.
   - Set the ALLEV bit in the interrupt evaluation register (INTEVAL). Setting this bit reasserts the PSCINT to the ARM interrupt controller, if there are still any active interrupt events.

6.6 PSC Registers

Table 6-6 lists the memory-mapped registers for the PSC. See the device-specific data manual for the memory address of these registers.

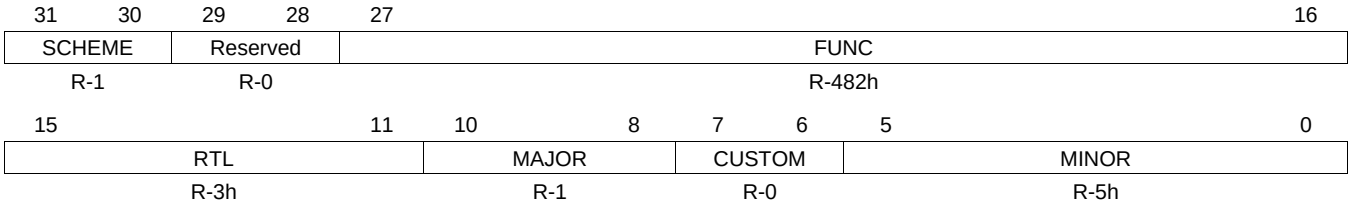
Table 6-6. Power and Sleep Controller (PSC) Registers

Offset	Register	Description	Section
0h	PID	Peripheral Revision and Class Information Register	Section 6.6.1
18h	INTEVAL	Interrupt Evaluation Register	Section 6.6.2
40h	MERRPR0	Module Error Pending Register 0 (modules 0-31)	Section 6.6.3
44h	MERRPR1	Module Error Pending Register 1 (modules 32-45)	Section 6.6.4
50h	MERRCR0	Module Error Clear Register 0 (modules 0-31)	Section 6.6.5
54h	MERRCR1	Module Error Clear Register 1 (modules 32-45)	Section 6.6.6
120h	PTCMD	Power Domain Transition Command Register	Section 6.6.7
128h	PTSTAT	Power Domain Transition Status Register	Section 6.6.8
200h	PDSTAT0	Power Domain Status Register	Section 6.6.9
300h	PDCTL0	Power Domain Control Register	Section 6.6.10
800h	MDSTAT*n*	Module Status *n* Register (modules 0-45)	Section 6.6.11
A00h	MDCTL*n*	Module Control *n* Register (modules 0-45)	Section 6.6.12

6.6.1 Peripheral Revision and Class Information Register (PID)

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

Figure 6-3. Peripheral Revision and Class Information Register (PID)



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

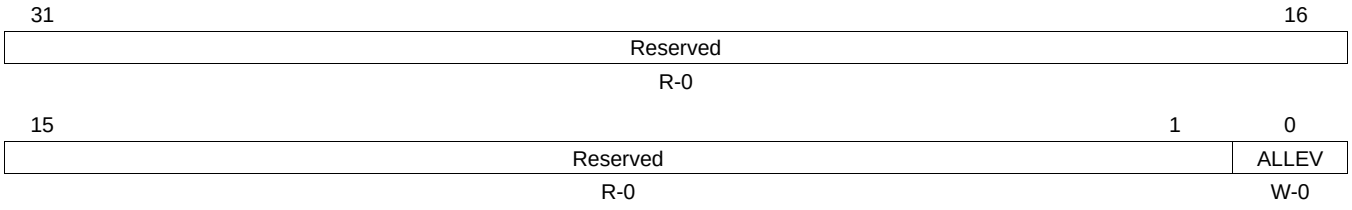
Table 6-7. Peripheral Revision and Class Information Register (PID) Field Descriptions

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.

6.6.2 Interrupt Evaluation Register (INTEVAL)

The interrupt evaluation register (INTEVAL) is shown in Figure 6-4 and described in Table 6-8.

Figure 6-4. Interrupt Evaluation Register (INTEVAL)



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

Table 6-8. 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.

6.6.3 Module Error Pending Register 0 (MERRPR0)

The module error pending register 0 (MERRPR0) records pending error conditions for modules 0-31. MERRPR0 is shown in Figure 6-5 and described in Table 6-9.

Figure 6-5. Module Error Pending Register 0 (MERRPR0)

31	0
M	
R- 0	

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

Table 6-9. 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	Module *n* does not have error condition.
		1	Module *n* has error condition. See the module status *n* register (MDSTAT*n*) for error type

6.6.4 Module Error Pending Register 1 (MERRPR1)

The module error pending register 1 (MERRPR1) records pending error conditions for modules 32-45. MERRPR1 is shown in Figure 6-6 and described in Table 6-10.

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

31	16
Reserved	
R-0	

15	14	13	12	4	3	0
Reserved		M	Reserved		M	
R-0		R-0	R-0		R-0	

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

Table 6-10. Module Error Pending Register 1 (MERRPR1) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	M[*n*]		Module interrupt status bit for module 45.
		0	Module *n* does not have error condition.
		1	Module *n* has error condition. See the module status *n* register (MDSTAT*n*) for error type.
12-4	Reserved	0	Reserved. (Modules 36-44 are reserved. See Table 6-1.)
3-0	M[*n*]		Module interrupt status bit for modules 32-35.
		0	Module *n* does not have error condition.
		1	Module *n* has error condition. See the module status *n* register (MDSTAT*n*) for error type.

6.6.5 Module Error Clear Register 0 (MERRCR0)

The module error clear register 0 (MERRCR0) clears the corresponding interrupt bit set (M[*n*]) in the module error pending register 0 (MERRPR0) and the module status *n* register (MDSTAT*n*) interrupt bit field for modules 0-31. MERRCR0 is shown in Figure 6-7 and described in Table 6-11.

Figure 6-7. Module Error Clear Register 0 (MERRCR0)

31	0
M	
W-0	

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

Table 6-11. Module Error Clear Register 0 (MERRCR0) Field Descriptions

Bit	Field	Value	Description
31-0	M[*n*]		Module interrupt clear bit for modules 0-31.
		0	A write of 0 has no effect.
		1	Clears module interrupt *n*.

6.6.6 Module Error Clear Register 1 (MERRCR1)

The module error clear register 1 (MERRCR1) clears the corresponding interrupt bit set (M[*n*]) in the module error pending register 1 (MERRPR1) and the module status *n* register (MDSTAT*n*) interrupt bit field for modules 32-45. MERRCR1 is shown in Figure 6-8 and described in Table 6-12.

Figure 6-8. Module Error Pending Register 1 (MERRCR1)

31															16
Reserved															
R-0															

15	14	13	12									4	3		0
Reserved		M	Reserved										M		
R-0		R-0	R-0										R-0		

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

Table 6-12. Module Error Clear Register 1 (MERRCR1) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	M[*n*]		Module interrupt clear bit for module 45.
		0	A write of 0 has no effect.
		1	Clears module interrupt *n*.
12-4	Reserved	0	Reserved. (Modules 36-44 are reserved. See Table 6-1.)
3-0	M[*n*]		Module interrupt clear bit for modules 32-35.
		0	A write of 0 has no effect.
		1	Clears module interrupt *n*.

6.6.7 Power Domain Transition Command Register (PTCMD)

The power domain transition command register (PTCMD) is shown in Figure 6-9 and described in Table 6-13.

Figure 6-9. Power Domain Transition Command Register (PTCMD)

31	16
Reserved	
R-0	

15	1	0
Reserved		GO
R-0		W-0

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

Table 6-13. Power Domain Transition Command Register (PTCMD) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
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 the NEXT bit in the module control *n* register (MDCTL*n*) for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (the STATE bit in the module status *n* register (MDSTAT*n*)), the PSC will transition those respective domain/modules to the new NEXT state.

6.6.8 Power Domain Transition Status Register (PTSTAT)

The power domain transition status register (PTSTAT) is shown in Figure 6-10 and described in Table 6-14 .

Figure 6-10. Power Domain Transition Status Register (PTSTAT)

31	16
Reserved	
R-0	

15	1	0
Reserved		GOSTAT
R-0		R-0

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

Table 6-14. Power Domain Transition Status Register (PTSTAT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
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.

6.6.9 Power Domain Status Register (PDSTAT0)

The power domain status register (PDSTAT0) is shown in Figure 6-11 and described in Table 6-15.

Figure 6-11. Power Domain Status Register (PDSTAT0)

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

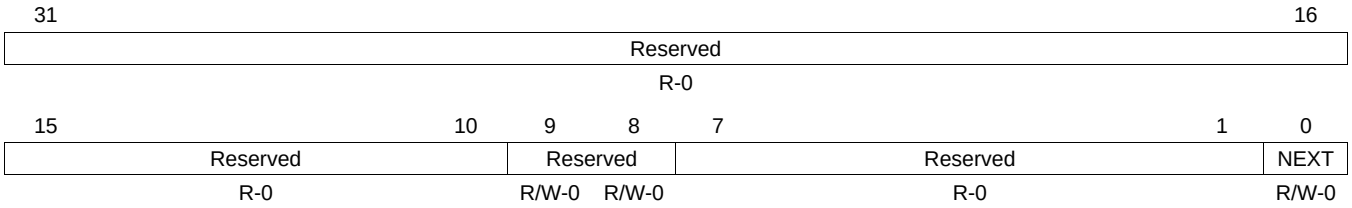
Table 6-15. Power Domain Status Register (PDSTAT0) Field Descriptions

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.

6.6.10 Power Domain Control Register (PDCTL0)

The power domain control register (PDCTL0) is shown in Figure 6-12 and described in Table 6-16.

Figure 6-12. Power Domain Control Register (PDCTL0)



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

Table 6-16. Power Domain Control Register (PDCTL0) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	Reserved	0	Reserved. Always write 0 to these bits.
7-1	Reserved	0	Reserved
0	NEXT		Power domain next state. In the DM646x DMSoC, there is only one power domain (Always On) .The Always On power domain is always in the ON state when the chip is powered-on. This field has no effect.
		0	Power domain off.
		1	Power domain on.

6.6.11 Module Status n Register (MDSTAT0-MDSTAT45)

The module status *n* register (MDSTAT*n*) shows the status of each module. Each module has one dedicated register. MDSTAT*n* is shown in Figure 6-13 and described in Table 6-17.

Figure 6-13. 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 6-17. Module Status *n* Register (MDSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	EMUIHB		Emulation alters module state interrupt status.
		0	Interrupt is not active.
		1	Interrupt is active.
16	EMURST		Emulation alters module reset interrupt status.
		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 deasserted.
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, HDVICP0 and HDVICP1 modules only).
		0	Local reset is asserted.
		1	Local reset is deasserted.
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

6.6.12 Module Control n Register (MDCTL0-MDCTL45)

The module control *n* register (MDCTL*n*) provides specific control for an individual module. Each module has one dedicated register. MDCTL*n* is shown in Figure 6-14 and described in Table 6-18.

Figure 6-14. 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 6-18. Module Control *n* Register (MDCTL*n*) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	EMUIHBIE		Emulation alters module state interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
9	EMURSTIE		Emulation alters module reset interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
8	LRST		Module local reset control (this bit applies to the DSP, HDVICP0 and HDVICP1 modules only.)
		0	Local reset is asserted.
		1	Local reset is deasserted.
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

7.1 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 either to 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.

The TMS320DM646x 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 allows you to slow down the clocks on the chip in order to reduce switching power. In particular, the DM646x DMSoC includes all of the power management features described in Table 7-1.

Table 7-1. 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

7.2 PSC and PLLC Overview

The power and sleep controller (PSC) plays an important role in managing system 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 6. For detailed information on the PLLC, see Chapter 5 and the device-specific data manual.

7.3 Clock Management

7.3.1 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 DM646x 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 6.

7.3.2 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 describes how to program the PLL frequency and the frequency constraints.

7.3.3 PLL Bypass and Power Down

The PLLs can be bypassed in the DM646x 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 the input clock source at DEV_MXI; therefore, this bypass mode can be used 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, the PLL can be powered down in bypass mode to save additional active power.

Chapter 5 describes PLL bypass and PLL power down details.

7.4 ARM and DSP Sleep Mode Management

7.4.1 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; otherwise, 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.

7.4.2 DSP Sleep Modes

The C64x+ megamodule of the DSP subsystem includes a power-down controller (PDC). 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 DM646x 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 *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8).

7.5 I/O Management

7.5.1 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. The VDD3P3V_PWDN register in the System Module controls power to the 3.3V I/O cells. The 3.3V I/Os are separated into functional groups for independent control different modules. For these groups, only the I/O cells needed for Host/AEMIF boot or power up operations are powered up by default (CLKOUT, boot configuration pins, PCI/HPI/AEMIF Blocks, GPIO Block). All other I/O cells are powered down by default to save power.

NOTE: If any of these I/O pins are needed for the application, be sure that application code starts out by programming the corresponding bits in VDD3P3V_PWDN to 0 to power up the I/O cells.

See the device-specific data manual for details on the VDD3P3V_PWDN register.

7.6 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 PHYPDWN bit in the USBCTL register of the system control module. USBCTL is described in the device-specific data manual.

ARM Interrupt Controller (AINTC)

Topic **Page**

8.1 Introduction

The TMS320DM646x 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.

8.2 Interrupt Mapping

The ARM926EJ CPU core supports 2 direct interrupts: FIQ and IRQ. The ARM interrupt controller (AINTC) prioritizes up to 64 interrupt requests from various peripherals and subsystems, listed in Table 8-1, and interrupts the ARM CPU. Each interrupt is programmable for up to 8 priority levels (6 levels for IRQ and 2 levels for FIQ). Interrupts at the same priority level are serviced in order by the ARM interrupt number, with the lowest number having the highest priority.

Table 8-1. ARM Interrupt Map

ARM Interrupt Number	Acronym	Source	ARM Interrupt Number	Acronym	Source
0	VP_VERTINT0	VPIF	32	TINTL0	Timer 0 lower – TINT12
1	VP_VERTINT1	VPIF	33	TINTH0	Timer 0 upper – TINT34
2	VP_VERTINT2	VPIF	34	TINTL1	Timer 1 lower – TINT12
3	VP_VERTINT3	VPIF	35	TINTH1	Timer 1 upper – TINT34
4	VP_ERRINT	VPIF	36	PWMINT0	PWM 0
5	-	Reserved	37	PWMINT1	PWM 1
6	-	Reserved	38	VLQINT	VLYNQ
7	WDINT	WD Timer (TIMER 2) – TINT12	39	I2CINT	I2C
8	CRGENINT0	CRGEN 0	40	UARTINT0	UART 0
9	CRGENINT1	CRGEN 1	41	UARTINT1	UART 1
10	TSINT0	TSIF 0	42	UARTINT2	UART 2
11	TSINT1	TSIF 1	43	SPINT0	SPI
12	VDCEINT	VDCE	44	SPINT1	SPI
13	USBINT	USB	45	DSP2ARM0	DSP Controller to ARM
14	USBDMAINT	USB DMA	46	-	Reserved
15	PCIINT	PCI	47	PSCINT	Power and Sleep Controller
16	CCINT0	EDMA CC Region 0	48	GPIO0	GPIO
17	CCERRINT	EDMA CC Error	49	GPIO1	GPIO
18	TCERRINT0	EDMA TC 0 Error	50	GPIO2	GPIO
19	TCERRINT1	EDMA TC 1 Error	51	GPIO3	GPIO
20	TCERRINT2	EDMA TC 2 Error	52	GPIO4	GPIO
21	TCERRINT3	EDMA TC 3 Error	53	GPIO5	GPIO
22	IDEINT	ATA	54	GPIO6	GPIO
23	HPIINT	HPI	55	GPIO7	GPIO
24	MAC_RXTH	EMAC Receive Threshold	56	GPIOBNK0	GPIO Bank 0
25	MAC_RX	EMAC Receive	57	GPIOBNK1	GPIO Bank 1
26	MAC_TX	EMAC Transmit	58	GPIOBNK2	GPIO Bank 2
27	MAC_MISC	EMAC Miscellaneous	59	DDRINT	DDR2 Memory Controller
28	AXINT0	McASP0 Transmit	60	EMIFAINT	EMIFA
29	ARINT0	McASP0 Receive	61	COMMTX	ARMSS
30	AXINT1	McASP1 Transmit	62	COMMRX	ARMSS
31	-	Reserved	63	EMUINT	E2ICE

8.3 AINTC Methodology

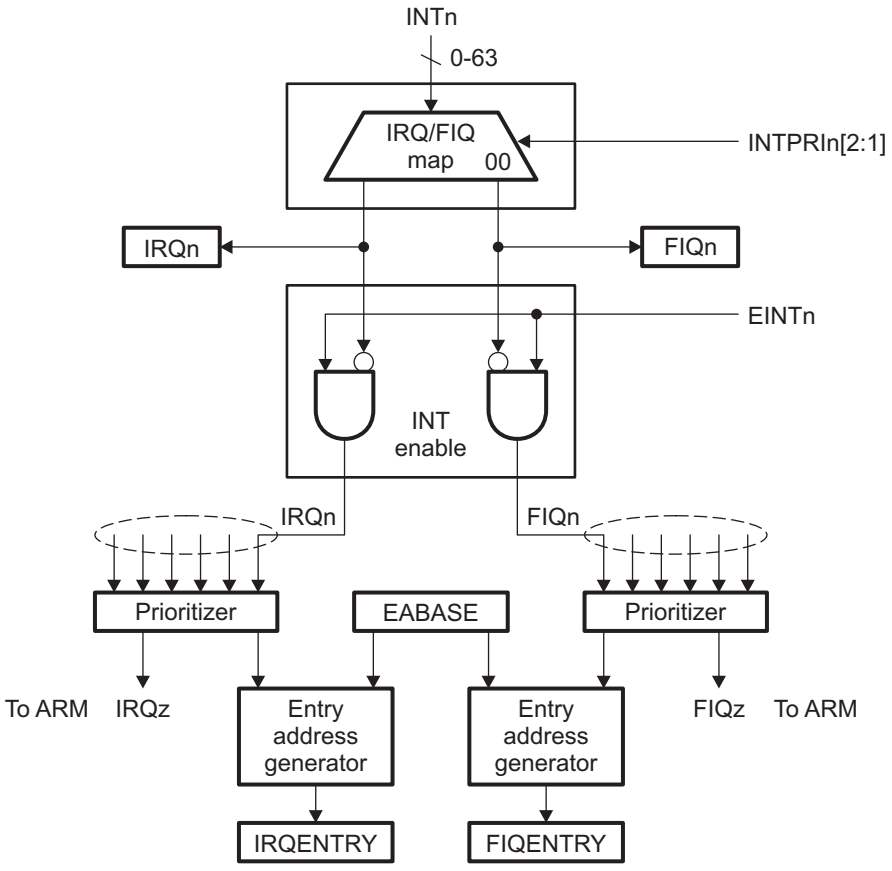
AINTC methodology is illustrated in Figure 8-1 and described below.

- 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 ARM926EJ 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.

Figure 8-1. AINTC Functional Diagram



8.3.1 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.

8.3.2 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.

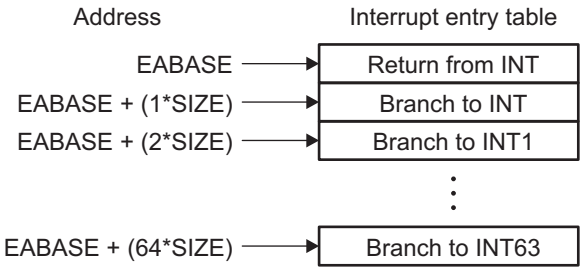
8.3.3 Vector Table Entry Address Generation

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 8-2.

Figure 8-2. 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).

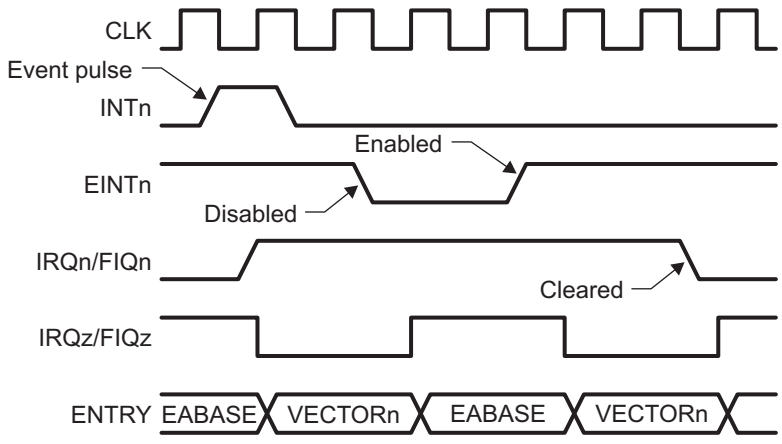
8.3.4 Clearing Interrupts

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.

8.3.5 Enabling and Disabling Interrupts

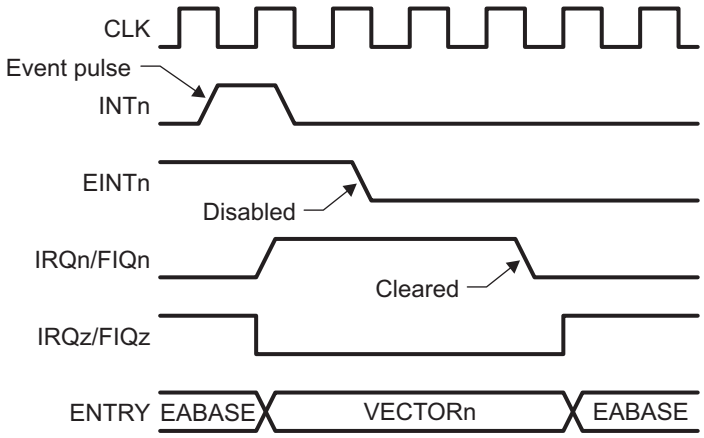
The AINTC has two methods for enabling and disabling interrupts: immediate or delayed, based on the setting of the IDMODE bit in the INTCTL register. When 0 (default), clearing an interrupt's EINT bit has an immediate effect. The prioritizer removes the disabled interrupt from consideration and adjusts 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 8-3.

Figure 8-3. 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 8-4.

Figure 8-4. Delayed Interrupt Disable



8.4 AINTC Registers

Table 8-2 lists the memory-mapped registers for the AINTC. See the device-specific data manual for the memory address of these registers.

Table 8-2. ARM Interrupt Controller (AINTC) Registers

Offset	Acronym	Register Description	Section
00h	FIQ0	Fast Interrupt Request Status Register 0	Section 8.4.1
04h	FIQ1	Fast Interrupt Request Status Register 1	Section 8.4.2
08h	IRQ0	Interrupt Request Status Register 0	Section 8.4.3
0Ch	IRQ1	Interrupt Request Status Register 1	Section 8.4.4
10h	FIQENTRY	Fast Interrupt Request Entry Address Register	Section 8.4.5
14h	IRQENTRY	Interrupt Request Entry Address Register	Section 8.4.6
18h	EINT0	Interrupt Enable Register 0	Section 8.4.7
1Ch	EINT1	Interrupt Enable Register 1	Section 8.4.8
20h	INTCTL	Interrupt Operation Control Register	Section 8.4.9
24h	EABASE	Interrupt Entry Table Base Address Register	Section 8.4.10
30h	INTPRI0	Interrupt 0-7 Priority Register 0	Section 8.4.11
34h	INTPRI1	Interrupt 8-15 Priority Register 1	Section 8.4.12
38h	INTPRI2	Interrupt 16-23 Priority Register 2	Section 8.4.13
3Ch	INTPRI3	Interrupt 24-31 Priority Register 3	Section 8.4.14
40h	INTPRI4	Interrupt 32-39 Priority Register 4	Section 8.4.15
44h	INTPRI5	Interrupt 40-47 Priority Register 5	Section 8.4.16
48h	INTPRI6	Interrupt 48-55 Priority Register 6	Section 8.4.17
4Ch	INTPRI7	Interrupt 56-63 Priority Register 7	Section 8.4.18

8.4.1 Fast Interrupt Request Status Register 0 (FIQ0)

The fast interrupt request status register 0 (FIQ0) is shown in Figure 8-5 and described in Table 8-3. Interrupt status of INT[31:0] (if mapped to FIQ).

Figure 8-5. Fast Interrupt Request Status Register 0 (FIQ0)

31	16
FIQ	
R/W-1	

15	0
FIQ	
R/W-1	

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

Table 8-3. 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.

8.4.2 Fast Interrupt Request Status Register 1 (FIQ1)

The fast interrupt request status register 1 (FIQ1) is shown in Figure 8-6 and described in Table 8-4. Interrupt status of INT[63:32] (if mapped to FIQ).

Figure 8-6. Fast Interrupt Request Status Register 1 (FIQ1)

31	16
Reserved	
R-1	

15	0
FIQ	
R/W-1	

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

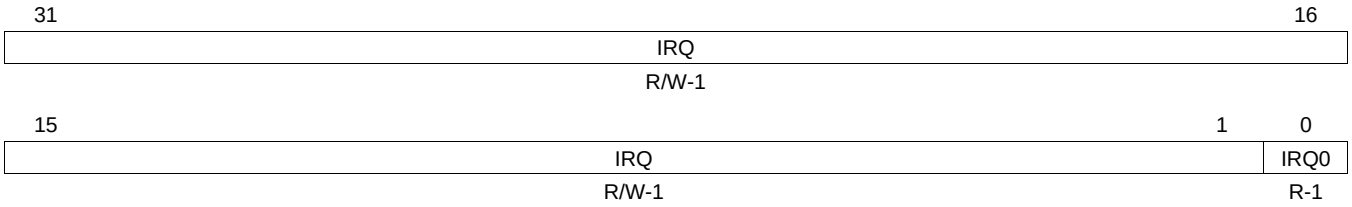
Table 8-4. Fast Interrupt Request Status Register 1 (FIQ1) Field Descriptions

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.

8.4.3 Interrupt Request Status Register 0 (IRQ0)

The interrupt request status register 0 (IRQ0) is shown in Figure 8-7 and described in Table 8-5. Interrupt status of INT[31:0] (if mapped to IRQ).

Figure 8-7. Interrupt Request Status Register 0 (IRQ0)



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

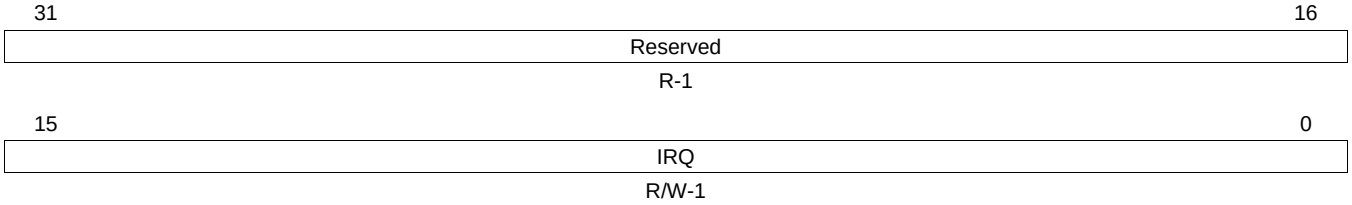
Table 8-5. Interrupt Request Status Register 0 (IRQ0) Field Descriptions

Bit	Field	Value	Description
31-1	IRQ[*n*]		Interrupt status of INT*n*, if mapped to interrupt request (IRQ31-1).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.
0	IRQ0	1	Interrupt 0 is always mapped to fast interrupt request (FIQ).

8.4.4 Interrupt Request Status Register 1 (IRQ1)

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

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



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

Table 8-6. Interrupt Request Status Register 1 (IRQ1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	1	Reserved
15-0	IRQ[*n*]		Interrupt status of INT*n*, if mapped to fast interrupt request (IRQ47-32).
		0	When reading bit, interrupt occurred.
		1	When writing bit, acknowledge interrupt.

8.4.5 Fast Interrupt Request Entry Address Register (FIQENTRY)

The fast interrupt request entry address register (FIQENTRY) is shown in Figure 8-9 and described in Table 8-7. Entry address [28:0] for valid FIQ interrupt.

Figure 8-9. 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 8-7. 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).

8.4.6 Interrupt Request Entry Address Register (IRQENTRY)

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

Figure 8-10. 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

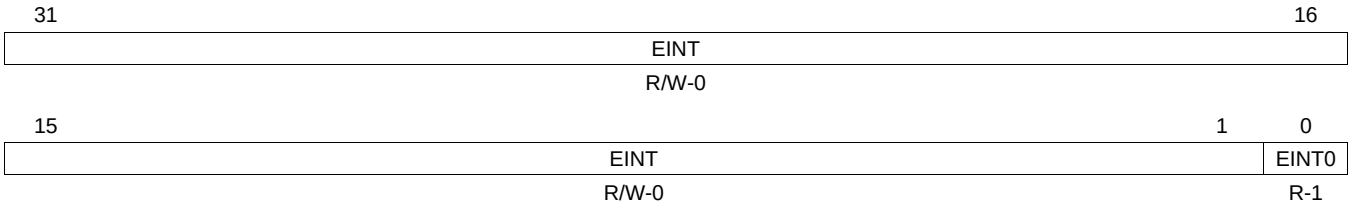
Table 8-8. Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions

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).

8.4.7 Interrupt Enable Register 0 (EINT0)

The interrupt enable register 0 (EINT0) is shown in Figure 8-11 and described in Table 8-9.

Figure 8-11. Interrupt Enable Register 0 (EINT0)



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

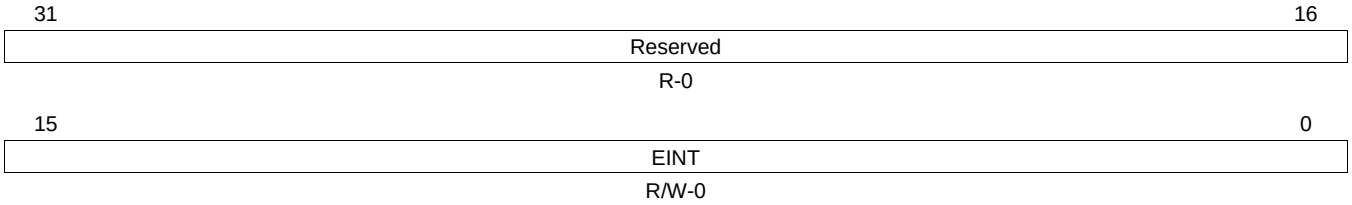
Table 8-9. Interrupt Enable Register 0 (EINT0) Field Descriptions

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.

8.4.8 Interrupt Enable Register 1 (EINT1)

The interrupt enable register 1 (EINT1) is shown in Figure 8-12 and described in Table 8-10.

Figure 8-12. Interrupt Enable Register 1 (EINT1)



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

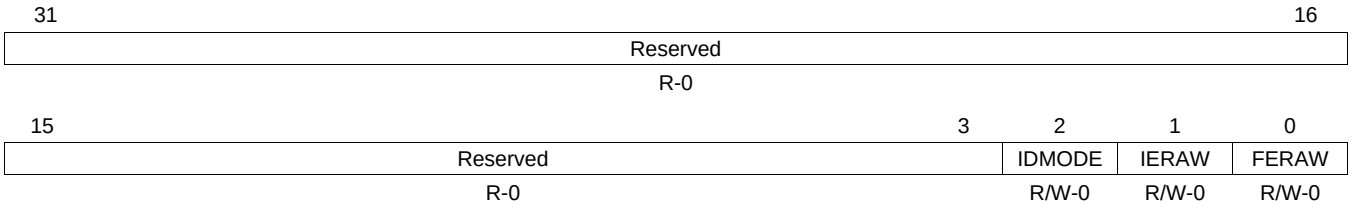
Table 8-10. Interrupt Enable Register 1 (EINT1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	EINT[*n*]		Interrupt enable for INT*n*. Bits 0 through 15 represent interrupts 32-47, respectively.
		0	Interrupt is disabled.
		1	Interrupt is enabled.

8.4.9 Interrupt Operation Control Register (INTCTL)

The interrupt operation control register (INTCTL) is shown in Figure 8-13 and described in Table 8-11.

Figure 8-13. Interrupt Operation Control Register (INTCTL)



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

Table 8-11. 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.

8.4.10 Interrupt Entry Table Base Address Register (EABASE)

The interrupt entry table base address register (EABASE) is shown in Figure 8-14 and described in Table 8-12.

Figure 8-14. 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 8-12. 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

8.4.11 Interrupt Priority Register 0 (INTPRI0)

The interrupt priority register 0 (INTPRI0) is shown in Figure 8-15 and described in Table 8-13.

Figure 8-15. 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 8-13. Interrupt Priority Register 0 (INTPRI0) Field Descriptions

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

8.4.12 Interrupt Priority Register 1 (INTPRI1)

The interrupt priority register 1 (INTPRI1) is shown in Figure 8-16 and described in Table 8-14.

Figure 8-16. 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 8-14. Interrupt Priority Register 1 (INTPRI1) Field Descriptions

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

8.4.13 Interrupt Priority Register 2 (INTPRI2)

The interrupt priority register 2 (INTPRI2) is shown in Figure 8-17 and described in Table 8-15.

Figure 8-17. 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 8-15. Interrupt Priority Register 2 (INTPRI2) Field Descriptions

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

8.4.14 Interrupt Priority Register 3 (INTPRI3)

The interrupt priority register 3 (INTPRI3) is shown in Figure 8-18 and described in Table 8-16.

Figure 8-18. 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 8-16. Interrupt Priority Register 3 (INTPRI3) Field Descriptions

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

8.4.15 Interrupt Priority Register 4 (INTPRI4)

The interrupt priority register 4 (INTPRI4) is shown in Figure 8-19 and described in Table 8-17.

Figure 8-19. 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 8-17. Interrupt Priority Register 4 (INTPRI4) Field Descriptions

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

8.4.16 Interrupt Priority Register 5 (INTPRI5)

The interrupt priority register 5 (INTPRI5) is shown in Figure 8-20 and described in Table 8-18.

Figure 8-20. 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 8-18. Interrupt Priority Register 5 (INTPRI5) Field Descriptions

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

8.4.17 Interrupt Priority Register 6 (INTPRI6)

The interrupt priority register 6 (INTPRI6) is shown in Figure 8-21 and described in Table 8-19.

Figure 8-21. 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 8-19. Interrupt Priority Register 6 (INTPRI6) Field Descriptions

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

8.4.18 Interrupt Priority Register 7 (INTPRI7)

The interrupt priority register 7 (INTPRI7) is shown in Figure 8-22 and described in Table 8-20.

Figure 8-22. 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 8-20. 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**

9.1 Overview of the System Control Module

The TMS320DM646x 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
  - Device boot process status
- ARM-DSP Integration
  - ARM-DSP interrupt control and status
  - DSP boot address control and status
- Power Management
  - V_{DD} 3.3 V I/O power-down control
- Special Peripheral Status and Control
  - Universal serial bus (USB) interface control
  - Host port interface (HPI) control
  - Video clock control
  - Transport stream interface (TSIF) control
  - Video and TSIF clock disable
  - PWM control
  - EDMA3 transfer controller (EDMA3TC) default burst size configuration
  - ARM memory wait state 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
- Clock and oscillator control

This chapter describes the system control module.

9.2 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.

9.3 Device Configuration

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

9.3.1 Pin Multiplexing Control

The DM646x 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.

9.3.2 Device Boot Configuration Status

The boot configuration status (BOOTMODE, CS2_BW, PCIEN, and DSP_BT bits) is captured in the boot configuration register (BOOTCFG) in the System Module. See the device-specific data manual for details on BOOTCFG.

9.3.3 Device Boot Process Status

The boot status register (BOOTSTAT) indicates the status of the device boot process (for example, boot error, boot complete, or watchdog timer reset). See the device-specific data manual for details on BOOTSTAT.

9.4 ARM-DSP Integration

9.4.1 ARM-DSP Interrupt Control and Status

The System Module includes registers for generating interrupts from the ARM to the DSP (DSPINT, DSPINTSET, and DSPINTCLR) and from the DSP to the ARM (DSPINT, DSPINTSET, and DSPINTCLR). See the device-specific data manual for details on these registers.

The ARM uses DSPINT, DSPINTSET, and DSPINTCLR to generate an interrupt to the DSP. The DSP interrupt status register (DSPINT) shows the status of the ARM-to-DSP interrupts. The ARM may generate an interrupt to the DSP by setting one of the four INTDSP*n* bits or the INTNMI bit in the DSP interrupt set register (DSPINTSET). The interrupt set (INTDSP*n*) bit then self-clears and the corresponding bit in DSPINT is automatically set to indicate that the interrupt was generated. After servicing the interrupt, the DSP clears the status bit in DSPINT by writing a 1 to the corresponding bit in the DSP interrupt clear register (DSPINTCLR). The ARM may poll the status bit in DSPINT to determine when the DSP has completed the interrupt service.

The DSP may generate an interrupt to the ARM in a similar manner using the ARM interrupt set register (ARMINTSET) and the ARM interrupt clear register (ARMINTCLR). The DSP can monitor the status of the DSP-to-ARM interrupts using the ARM interrupt status register (ARMINT). See Chapter 12 for more detailed information.

9.4.2 DSP Boot Address Control and Status

The DSP boot address register (DSPBOOTADDR) in the System Module contains the DSP reset vector. See the device-specific data manual for details on DSPBOOTADDR. The boot address defaults to 4220:0000h (EMIF 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 11.

9.5 Power Management

9.5.1 V_{DD} 3.3 V I/O Power-Down Control

The VDD 3.3V I/O power-down control register (VDD3P3V_PWDN) in the System Module 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 the device-specific data manual for details on VDD3P3V_PWDN.

9.6 Special Peripheral Status and Control

Several of the DM646x DMSoC peripherals require additional system-level control logic. Those registers are discussed in this section.

9.6.1 Universal Serial Bus (USB) Interface Control

The USB control register (USBCTL) controls various features of the USB interface. See the device-specific data manual for details on USBCTL.

9.6.2 Host Port Interface (HPI) Control

The HPI control register (HPICTL) controls write access to the HPI control and address registers and determines the host time-out value. HPICTL also determines the output mode of the $\overline{\text{HRDY}}$ signal. HPICTL is not reset by a soft reset, so that the HPI width remains correctly configured. See the device-specific data manual for details on HPICTL.

9.6.3 Video Clock Control and Disable

The video clock control register (VIDCLKCTL) allows you to select/control the clock multiplexing for the video channel (channels 1, 2, and 3) output clock source. See the device-specific data manual for details on VIDCLKCTL.

9.6.4 Transport Stream Interface (TSIF) Control

The TSIF control register (TSIFCTL) allows you to select/control the clock multiplexing for the counter and serial output of TSIF1 and the counter and parallel/serial output for TSIF0. See the device-specific data manual for details on TSIFCTL.

9.6.5 Video Source Clock Control and Disable

The video source clock disable register (VSCLKDIS) allows you to disable the selected video port interface (VPIF), transport stream interface (TSIF), and clock reference generator (CRGEN) module input clocks. See the device-specific data manual for details on VSCLKDIS.

NOTE: To ensure glitch-free operation, the clock should be disabled before changing the clock source frequency or multiplexing using the video clock control register (VIDCLKCTL) and the TSIF control register (TSIFCTL).

9.6.6 PWM Control

The PWM control register (PWMCTL) controls the chip-level connections of PWM0 and PWM1. See device-specific data manual for details on PWMCTL.

9.6.7 EDMA3 Transfer Controller (EDMA3TC) Burst Size Configuration

The EDMA transfer controller default burst size configuration register (EDMATCCFG) configures the default burst size (DBS) for the EDMA transfer controllers (EDMA3TC0, EDMA3TC1, EDMA3TC2, and EDMA3TC3). See the device-specific data manual for details on EDMATCCFG.

9.6.8 ARM Memory Wait State Control

The ARM memory wait state control register (ARMWAIT) is used to control ARM926 accesses to its TCM RAM. At normal ARM operating frequency, a wait state must be inserted when accessing TCM RAM. When the device is operating at lower speeds, performance may be increased by removing the wait state. Note that the TCM ROM will always operate with a wait state enabled. See the device-specific data manual for details on ARMWAIT.

9.7 Bandwidth Management

9.7.1 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 DM646x DMSoC master is shown in Table 9-1.

Table 9-1. TMS320DM646x DMSoC Master IDs

mstid	Master
0	ARM Instruction
1	ARM Data
2	DSP MDMA
3	DSP CFG
4-7	Reserved
8	HDVICP0 CFG
9	HDVICP1 CFG
10	EDMA3 CC TR
11-15	Reserved
16	EDMA3 TC0 read
17	EDMA3 TC0 write
18	EDMA3 TC1 read
19	EDMA3 TC1 write
20	EDMA3 TC2 read
21	EDMA3 TC2 write
22	EDMA3 TC3 read
23	EDMA3 TC3 write
24-31	Reserved
32	PCI
33	HPI
34	ATA
35	EMAC
36	USB
37	VLYNQ
38	VPIF mstr1 read
39	VPIF mstr0 write
40	TSIF0 read
41	TSIF0 write
42	TSIF1 read
43	TSIF1 write
44	VDCE write
45	VDCE read
46-63	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 DM646x DMSoC, all master peripherals are programmed in the bus master priority control registers (MSTRPRI*n*) in the System Module. The default priority level for each bus master is listed in Table 9-2. Application software is expected to modify these values to obtain the desired system performance.

Table 9-2. TMS320DM646x DMSoC Default Master Priorities

Default Priority Level	Bus Master	Priority Bit Field	Priority Control Register
1	VPIF Capture	VP0P	MSTPRI2 Register
1	VPIF Display	VP1P	MSTPRI2 Register
1	TSIF0	TSIF0P	MSTPRI2 Register
1	TSIF1	TSIF1P	MSTPRI2 Register
2	EDMA3TC0	EDMATC0P	MSTPRI2 Register(1)
2	EDMA3TC1	EDMATC1P	MSTPRI2 Register(1)
2	EDMA3TC2	EDMATC2P	MSTPRI2 Register(1)
2	EDMA3TC3	EDMATC3P	MSTPRI2 Register(1)
3	HDVICP0 (CFG)	HDVICP0P	MSTPRI0 Register
3	HDVICP1 (CFG)	HDVICP1P	MSTPRI0 Register
4	ARM926 (ARM Instruction)	ARMINSTP	MSTPRI0 Register
4	ARM926 (ARM Data)	ARMDATAP	MSTPRI0 Register
4	C64x+ DSP (DMA)	DSPDMAP	MSTPRI0 Register(2)
4	C64x+ DSP (CFG)	DSPCFGP	MSTPRI0 Register
4	VDCE	VDCEP	MSTPRI1 Register
5	EMAC	EMACP	MSTPRI1 Register
5	USB2.0	USBP	MSTPRI1 Register
5	ATA	ATAP	MSTPRI1 Register
5	VLYNQ	VLYNQP	MSTPRI1 Register
6	PCI	PCIP	MSTPRI1 Register
6	HPI	HPIP	MSTPRI1 Register

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

9.8 Emulation Control

9.8.1 Set Emulator Suspend Source

The flexibility of the DM646x 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 emulator suspend source register (SUSPSRC) 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.

9.9 Clock and Oscillator Control

The auxiliary (24 MHz) oscillator and the clock source of the CLKOUT, AUDIO_CLK0, and AUDIO_CLK1 outputs are controlled by the clock and oscillator control register (CLKCTL). See the device-specific data manual for details on CLKCTL.

9.10 System Control Register Descriptions

Table 9-3 lists the memory-mapped registers for the system control. See the device-specific data manual for the memory address of these registers and complete descriptions.

Table 9-3. System Control Registers

Offset	Acronym	Register Description
0h	PINMUX0	Pin Multiplexing Control 0
4h	PINMUX1	Pin Multiplexing Control 1
8h	DSPBOOTADDR	DSP Boot Address. Decoded by bootloader software for host boots.
Ch	SUSPSRC	Emulator Suspend Source
10h	BOOTSTAT	Boot Status
14h	BOOTCFG	Device Boot Configuration
24h	ARMBOOT	ARM926 Boot Control
28h	JTAGID	Device ID Number
30h	HPICTL	HPI Control
34h	USBCTL	USB Control
38h	VIDCLKCTL	Video Clock Control
3Ch	MSTPRI0	Bus Master Priority Control 0
40h	MSTPRI1	Bus Master Priority Control 1
44h	MSTPRI2	Bus Master Priority Control 2
48h	VDD3P3V_PWDN	V_{DD} 3.3-V I/O Powerdown Control
50h	TSIFCTL	TSIF Control
54h	PWMCTL	PWM Control
58h	EDMATCCFG	EDMA TC Configuration
5Ch	CLKCTL	Oscillator and Output Clock Control
60h	DSPINT	ARM to DSP Interrupt Status
64h	DSPINTSET	ARM to DSP Interrupt Set
68h	DSPINTCLR	ARM to DSP Interrupt Clear
6Ch	VSCLKDIS	Video and TSIF Clock Disable
70h	ARMINT	DSP to ARM Interrupt Status
74h	ARMINTSET	DSP to ARM Interrupt Set
78h	ARMINTCLR	DSP to ARM Interrupt Clear
7Ch	ARMWAIT	ARM Memory Wait State Control

Reset

Topic **Page**

10.1 Reset Overview

There are six types of reset in the TMS320DM646x DMSoC. The types of reset differ by how they are initiated and/or by their effect on the device. Each type is briefly described in Table 10-1 and further described in the following sections.

Table 10-1. Reset Types

Type	Initiator	Effect
Power-on reset (POR)	$\overline{\text{POR}}$ pin active low	Total reset of the chip (cold reset). Resets all modules including memory and emulation. Total reset of chip (cold reset). Activates the $\overline{\text{POR}}$ signal on-chip, which resets the entire chip including the emulation logic. $\overline{\text{POR}}$ assertion also causes internal $\overline{\text{TRST}}$ signal to be asserted. The power-on reset ($\overline{\text{POR}}$) must be driven low during power-ramp of the device. Device boot and configuration pins are latched.
Warm reset	$\overline{\text{RESET}}$ pin active low	Resets all modules including memory, except emulation logic. Deassertion of $\overline{\text{RESET}}$ causes latching of the device boot and configuration pins. Emulator stays alive during Warm reset.
Maximum (Max) reset	Emulator or Watchdog Timer (Timer2)	Same as Warm reset, except the device boot and configuration pins are not relatched.
System reset	Emulator	A soft reset. A soft reset maintains memory contents, and does not affect or reset clocks or power states. Does not reset emulation logic, nor relatch device boot and configuration pins.
Module reset	Software	Resets a specific module. Allows the software to independently reset any module.
DSP local reset	ARM software	Resets the DSP CPU. The DSP internal memories (L1P, L1D, and L2) are not reset. Allows the ARM to reset and boot the DSP.
Test reset (TRST)	$\overline{\text{TRST}}$ pin	Test reset on JTAG interface. Drive $\overline{\text{TRST}}$ pin low to reset the test and emulation logic ($\overline{\text{POR}}$ is also ANDed with the reset from this pin before going to the test reset targets). For proper DSP operation, $\overline{\text{TRST}}$ should always be 0 (active low) during normal functional mode of operation. Also, the $\overline{\text{TRST}}$ pin is required to be pulled high externally for proper ARM emulaton operation.

10.2 Reset Pins

Power-on reset, Warm reset, and Test reset are initiated by the $\overline{\text{POR}}$, $\overline{\text{RESET}}$, and $\overline{\text{TRST}}$ pins, respectively. These pins are briefly described in Table 10-2. For more information, see the device-specific data manual.

Table 10-2. Reset Pins

Pin Name	Type	Description
$\overline{\text{POR}}$	Input	Power-on reset
$\overline{\text{RESET}}$	Input	Active-low device reset
$\overline{\text{TRST}}$	Input	JTAG test-port reset

10.3 Types of Reset

10.3.1 Power-On Reset (POR)

Power-on reset (POR) is initiated by the $\overline{\text{POR}}$ pin and is used to reset the entire chip, including the test and emulation logic. Power-on reset is also referred to as a cold reset, since the device usually goes through a power-up cycle. During power-up, the $\overline{\text{POR}}$ pin must be asserted (driven low) until the power supplies have reached their normal operating conditions.

The following steps describe the POR sequence:

1. Apply power and clocks to the chip and drive $\overline{\text{POR}}$ low to initiate POR.
2. Wait for the power supplies to reach normal operating conditions while keeping the $\overline{\text{POR}}$ pin asserted (driven low).
3. Wait for the input clock source to be stable while keeping the $\overline{\text{POR}}$ pin asserted (driven low).
4. Once the power supplies and the input clock source are stable, the $\overline{\text{POR}}$ pin must remain asserted (low) for a minimum number of DEV_MXI cycles (see device-specific data manual for number of cycles).
5. Hardware latches the device configuration pins on the rising edge of $\overline{\text{POR}}$. The device configuration pins allow you to set several options at reset. See Section 10.4.1 for more information.
6. Hardware resets all of the modules, including memories and emulation circuitry.
7. POR finishes, all of the 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.

10.3.2 Warm Reset

A Warm reset is activated by driving the $\overline{\text{RESET}}$ pin active-low. This resets everything in the device, except the test or emulation logic. A DSP or ARM emulator session will stay alive during warm reset.

The following steps describe the Warm reset sequence:

1. Emulator drives $\overline{\text{RESET}}$ low to initiate Warm reset.
2. Emulator drives $\overline{\text{RESET}}$ high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of $\overline{\text{RESET}}$. The device configuration pins allow you to set several options at reset. See Section 10.4.1 for more information.
4. Hardware resets all of the modules including memories, but not the emulation circuitry.
5. Warm reset finishes, all of the modules except emulation logic are in their default configurations, and hardware begins the boot process.

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

10.3.3 Maximum (Max) Reset

A Maximum (Max) reset is initiated by the emulator or the watchdog timer (Timer2). The effects are the same as a Warm reset, except the device boot and configuration pins are not relatched. The emulator initiates a Max reset via the ICEPICK module. This ICEPICK-initiated reset is nonmaskable. When the watchdog timer counter reaches zero, this also initiates a Max reset to recover from a runaway condition.

For debug, Max reset allows an emulator to initiate chip reset using an emulation command, while remaining active during and after the reset sequence. To invoke the Max reset via the ICEPICK module, you can perform the following from the Code Composer Studio™ IDE menu: Debug→Advanced Resets→System Reset.

The following steps describe the Max reset sequence:

1. To initiate Max reset, the watchdog timer expires (indicating a runaway condition), or the emulator initiates a Max reset command via the ICEPICK module.
2. Hardware resets all of the modules including memories, but not the emulation circuitry. The device boot and configuration pins are not relatched.
3. Warm reset finishes, all of the modules except emulation logic 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 watchdog timer-initiated Max reset for debug purposes.

See the *TMS320DM646x DMSoC 64-Bit Timer User's Guide* (SPRUER5) for information on the watchdog timer.

10.3.4 System Reset

The emulator initiates System reset via special DSP emulation or ICECrusher. System reset is considered a soft reset (memory contents are maintained, clock logic and power control logic are not affected). None of the following modules are reset: DDR2 Memory Controller, PLL Controller (PLLC), Power and Sleep Controller (PSC), and emulation circuitry.

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.

10.3.5 Module Reset

Module reset allows the software to independently reset a module. Module reset can be used 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; rather than for general use, because if care is not taken arbitrarily setting a module can result in the switch fabric locking up.

The procedures for asserting and deasserting module reset are fully described in Chapter 6. Furthermore, special considerations for DSP module reset are described in Chapter 12.

10.3.6 DSP Local Reset

You can use DSP local reset to reset the DSP CPU. When the DSP local reset is asserted, the DSP 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 deasserted.

The procedures for asserting and deasserting DSP local reset are fully described in the Chapter 12 and Chapter 6.

10.3.7 Test and Emulation Reset ($\overline{TRST}$ pin)

This is the Test reset on JTAG interface. Drive the $\overline{TRST}$ pin low to keep the test and emulation logic in reset. $\overline{TRST}$ needs to be released (pulled high) whenever it is necessary to use a JTAG controller to debug the device. You can pull the $\overline{TRST}$ pin high on the board if emulation is required.

10.4 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, EMIFA, and DSP boot mode.

NOTE: Default configuration is the configuration before the boot process begins. The boot ROM updates the configuration. See Chapter 11 for more information on the boot process.

10.4.1 Device Configuration Pins

The device configuration pins are latched at the end of power-on reset or Warm reset.

The boot configuration register (BOOTCFG) in the System Module is a read-only register that indicates the value of the device configuration pins latched at the end of reset. During a hard reset ($\overline{POR}$ or $\overline{RESET}$ pin active [low]), the values of the device configuration pins (BTMODE[3:0], CS2BW, PCIEN, and DSPBOOT) are propagated through BOOTCFG to the Boot Controller. When $\overline{RESET}$ or $\overline{POR}$ is deasserted (raising edge), the value of the pins is latched. The BOOTCFG value does not change as a result of a soft reset, instead the value latched at the end of the previous global reset is retained. BOOTCFG is shown in Figure 10-1 and described in Table 10-3. See device-specific data manual for details on BOOTCFG.

The device configuration pins allow you to configure the following options at reset:

- Boot Mode (BTMODE[3:0] pins)
- EMIFA EM_CS2 default bus width (CS2BW pin)
- PCI enable (PCIEN pin)
- DSP Boot Mode (DSPBOOT pin)

NOTE: The device configuration pins are multiplexed with pins of the video port interface (VPIF). After the device configuration pins are latched at reset, they automatically change to function as VPIF pins. Pin multiplexing is described in Chapter 9.

Figure 10-1. Boot Configuration Register (BOOTCFG)

31	18	17	16
Reserved		DSP_BT	PCIEN
R-0		R-L	R-L

15	9	8	7	4	3	0
Reserved		CS2_BW	Reserved		BOOTMODE	
R-0		R-L	R-0		R-L	

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

Table 10-3. Boot Configuration Register (BOOTCFG) Field Descriptions

Bit	Field	Value	Description	Latched pin at the rising edge of $\overline{\text{RESET}}$ or $\overline{\text{POR}}$
31-18	Reserved	0	Reserved. Read returns 0.	
17	DSP_BT		DSP Boot. This bit causes the DSP to be released from reset automatically. The C64x+ boots from EMIFA (default DSPBOOTADDR address 0x4220 0000). If BOOTMODE = 2h or 3h, or PCIEN = 1, then the C64x+ self-boot will fail since EMIFA will be disabled.	DSPBOOT
		0	ARM boots C64x+ DSP.	
		1	C64x+ DSP self-boots.	
16	PCIEN		PCI Enable.	PCIEN
		0	PCI is disabled.	
		1	PCI is enabled. The internal pullup and pulldown resistors on the PCI pins are disabled and configures the pin multiplexing for PCI.	
15-9	Reserved	0	Reserved. Read returns 0.	
8	CS2_BW		EMIFA $\overline{\text{EM\_CS2}}$ Default Bus Width. This bit determines the default bus width of the EMIFA $\overline{\text{EM\_CS2}}$ memory space. This ensures that boot from EMIFA (ARM or DSP) correctly reads the attached memory.	CS2BW
		0	Default to 8-bit operation.	
		1	Default to 16-bit operation.	
7-4	Reserved	0	Reserved. Read returns 0.	
3-0	BOOTMODE	0-Fh	Boot Mode Configuration.	BTMODE[3:0]
		0	Emulation boot. (BOOT from ROM)	
		1h	Reserved	
		2h	HPI-16 (if PCIEN = 0). (BOOT from ROM) PCI without autoinitialization (if PCIEN = 1). (BOOT from ROM)	
		3h	HPI-32 (if PCIEN = 0). (BOOT from ROM) PCI with autoinitialization (if PCIEN = 1). (BOOT from ROM)	
		4h	EMIFA direct boot (ROM/NOR) (if PCIEN = 0; error if PCIEN = 1, defaults to UART0).	
		5h	Reserved	
		6h	I2C boot. (BOOT from ROM)	
		7h	NAND Flash boot (if PCIEN = 0; error if PCIEN = 1).	
		8h	UART0 boot. (BOOT from ROM)	
		9h	Reserved	
		Ah	VLYNQ boot. (BOOT from ROM)	
		Bh	Reserved	
		Ch-Dh	Reserved	
		Eh	SPI boot. (BOOT from ROM)	
		Fh	Reserved	

10.4.2 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 DEV_MXI drives the chip after reset. For more information, see Chapter 5 and the device-specific data manual. 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 6-1 shows which modules are enabled after reset. As shown in Table 6-1, the following modules are enabled by default: ARM, Timer2 (watchdog timer), System Module, and ARM interrupt controller. Some modules are enabled by default depending on the sampled state of the device configuration pins. For example, EMIFA is enabled after reset when the device configuration pins (BTMODE[3:0] = 0100, EMIFA direct boot and PCIEN = 0) select EMIFA boot mode.

10.4.3 ARM Boot Mode Configuration

The device configuration pins (BTMODE[3:0]) determine whether the ARM boots from its ROM or from the asynchronous EMIF (EMIFA).

When ROM boot is selected (BTMODE[3:0] != 0100), a jump to the internal TCM ROM (0000 8000h) is forced into the first fetched instruction word. The embedded ROM boot loader (RBL) code then performs certain configuration steps, reads the boot configuration register (BOOTCFG) in the System Module to determine the desired boot method, and branches to an appropriate secondary loader utility.

If EMIFA boot is selected (BTMODE[3:0] = 0100), a jump to the highest branch address (0200 0000h) is forced into the first fetched instruction word. ARM Instruction Address Modification logic inserts a 1 on bit 30 of the address bus to modify the access to address 4200 0000h, which is the start of the EMIFA CS2 memory region. The ARM then continues executing from external memory using the default EMIFA timings until modified by software. Code within the EMIFA memory should execute a branch to the actual EMIFA address, and then disable the Instruction Address Modification logic by clearing the ADDRMOD bit in the ARM boot configuration register (ARMBOOT) in the System Module.

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

The ARM boot configuration register (ARMBOOT) is used to control the ARM926 boot. The ARMBOOT value does not change as a result of a global soft reset, instead the last value written is retained. ARMBOOT is shown in Figure 10-2 and described in Table 10-4. See device-specific data manual for details on ARMBOOT.

Figure 10-2. ARM Boot Configuration Register (ARMBOOT)

31	16
Reserved	
R-0	

15	5	4	3	1	0
Reserved		ADDRMOD	Reserved		TRAMBOOT
R-0		R/W-C	R-0		R/W-0

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

Table 10-4. ARM Boot Configuration Register (ARMBOOT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved. Read returns 0.
4	ADDRMOD		IAHB Address Modification. The default value for this bit is determined by the BOOTMODE configuration bits (BTMODE[3:0]). If BTMODE[3:0] = 0100 [EMIFA direct boot (ROM/NOR)] , then ADDRMOD defaults to 1 so that instruction fetches from the ARM point to EMIFA CS2 memory space. For all other BTMODE[3:0] values, ADDRMOD defaults to 0 so the ARM boots from its TCM (ROM or RAM).
		0	No address modification.
		1	Address bit 30 is tied high to modify IAHB fetch address to point to EMIFA.
3-1	Reserved	0	Reserved. Read returns 0.
0	TRAMBOOT		ARM TCM RAM Boot. This is a "sticky" bit that can be used to force the ARM926 to boot from ITCM RAM. On $\overline{\text{POR}}$ reset, this bit is initialized to 0 because TCM RAM is not initialized; otherwise, the bit retains the value. After initializing ITCM RAM, software can set this bit so that subsequent Warm reset ($\overline{\text{RESET}}$) or soft reset boots from the ITCM.
		0	Use BTMODE[3:0] selected boot mode
		1	Boot from ITCM RAM

10.4.4 EMIFA Configuration

10.4.4.1 EMIFA CS2 Bus Width Configuration

The CS2BW pin determines the default width of the first EMIFA chip select space (EM_CS2):

- if CS2BW = 0, the space defaults to 8-bits wide
- if CS2BW = 1, the space defaults to 16-bits wide

This allows the ARM to make full use of the width of the attached memory device, if booting from EMIFA or NAND.

NOTE: CS2BW only selects the default width and needs to be set depending on whether 8-bit or 16-bit EMIFA memory or NAND is used at boot time. After boot, the width of CS2BW can be changed by software by accessing the appropriate EMIFA control register.

The CS2BW input affects only the first EMIFA chip select space (EM_CS2). All other chip select spaces default to 8-bits wide and must be modified using the appropriate EMIFA control register if 16-bit operation is desired.

See the *TMS320DM646x DMSoC Asynchronous External Memory Interface (EMIF) User's Guide* (SPRUEQ7) for more information on the EMIF.

10.4.4.2 EMIFA Timing Configuration

When EMIFA 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 EMIFA is configured to run at 4.5 MHz/88, which equals approximately 51 kHz by default. See the *TMS320DM646x DMSoC Asynchronous External Memory Interface (EMIF) User's Guide* (SPRUEQ7) for more information on the EMIF.

10.4.5 PCI Enable (PCIEN) Operation

The PCIEN is latched into the boot configuration register (BOOTCFG) in the System Module from the PCIEN configuration pin at the end of reset (that is, rising edge of RESET or POR).

The PCIEN configuration signal is used to select the default configuration of the HPI/PCI/EMIFA pins at reset. This allows the DM646x DMSoC to be PCI-compliant at reset when integrated into a PCI system. When PCIEN = 1, the PCI module disables the internal pullup and pulldown resistors on the PCI pins and configures pin multiplexing for PCI. PCIEN is also used in bootmode selection to differentiate between HPI (PCIEN = 0) and PCI (PCIEN = 1) modes. The PCIEN input must be 0 when the EMIFA boot is selected but need not be 0 for other boot modes. This allows the device to be part of a PCI system even if booted from UART, SPI, etc.

See the device-specific data manual for PCIEN pin multiplexing details in the pin multiplexing control 0 register (PINMUX0).

10.4.6 DSP Boot Mode (DSP_BT) Configuration

The DSP_BT input determines the DSP operation at reset. For most applications, the ARM is the master device and controls the reset and boot of the DSP. Under this scenario (DSP_BT = 0), the C64x+ DSP remains disabled (held in reset) after reset. The ARM is responsible for releasing the DSP from reset. Before releasing the DSP from reset, the ARM must transfer a valid DSP boot image to program memory accessible by the DSP (DSP memory, EMIFA or DDR2), and configure the DSP boot address in the DSP boot address register (DSPBOOTADDR) in the System Module from which the C64x+ DSP begins execution.

When DSP_BT = 1, the C64x+ DSP boots itself. Under this scenario, the C64x+ DSP is released from reset without ARM intervention. The DSP boot address is set to an EMIFA address (4220 0000h). The C64x+ DSP begins execution with instruction (L1P) cache enabled.

NOTE: The DSP_BT operation is overridden when ARM HPI or PCI boot is selected (BOOTMODE[3:0] = 001x). This is because the ARM HPI/PCI boot selection forces the HPIEN or PCIEN bit in the pin multiplexing control 0 register (PINMUX0) to 1. This enables HPI/PCI functions on the EMIFA control and data pins and prevents the DSP from using the EMIFA. DSP_BT is treated as 0 internally when BTMODE[3:0] = 001x, regardless of the value at the configuration pin (the actual pin value should still be latched in the boot configuration register (BOOTCFG) in the System Module).

Boot Modes

The TMS320DM646x DMSoC ARM can boot either from asynchronous EMIF/NOR Flash or from ARM ROM, as determined by the device configuration pins (BTMODE[3:0], CS2_BW, PCIEN, and DSP_BT) at reset. The PCIEN pin configuration is used to select the default configuration of the EMIFA/PCI/HPI pins at reset. This allows the DM646x DMSoC to be PCI-compliant at reset. PCIEN = 1 disables the internal pullup and pulldown resistors on the PCI pins and configures the pin muxing for PCI. For all other bootmodes (non-PCI bootmode), the PCIEN must be cleared to 0.

For a more detailed description of the ROM boot modes supported by the DM646x DMSoC, see *Using the TMS320DM646x DMSoC Bootloader Application Report* (SPRAAS0).

ARM-DSP Integration

Topic **Page**

12.1 Introduction

The TMS320DM646x DMSoC integrates an ARM core for overall system control functions and a DSP subsystem for complex data and image/video processing functions. Figure 12-1 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, McASP, Timer0, and Timer1 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 (EMIF). 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, 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 McASP
  - ARM and DSP have access to Timer0 and Timer1
- Shared memory
  - ARM has access to DSP internal memory (L1P, L1D, L2)
  - DSP has access to ARM internal memory
  - ARM and DSP have access to DDR2 memory controller and asynchronous EMIF
- ARM-DSP interrupts
  - ARM can interrupt the DSP (via 4 general interrupts and 1 NMI)
  - DSP can interrupt the ARM (via 1 general interrupt)
- ARM control of DSP clock, reset, and boot
  - ARM can boot the DSP
  - ARM can control the DSP
  - Clock on/off
  - ARM can assert/deassert DSP module and local resets

These features are described in the following sections.

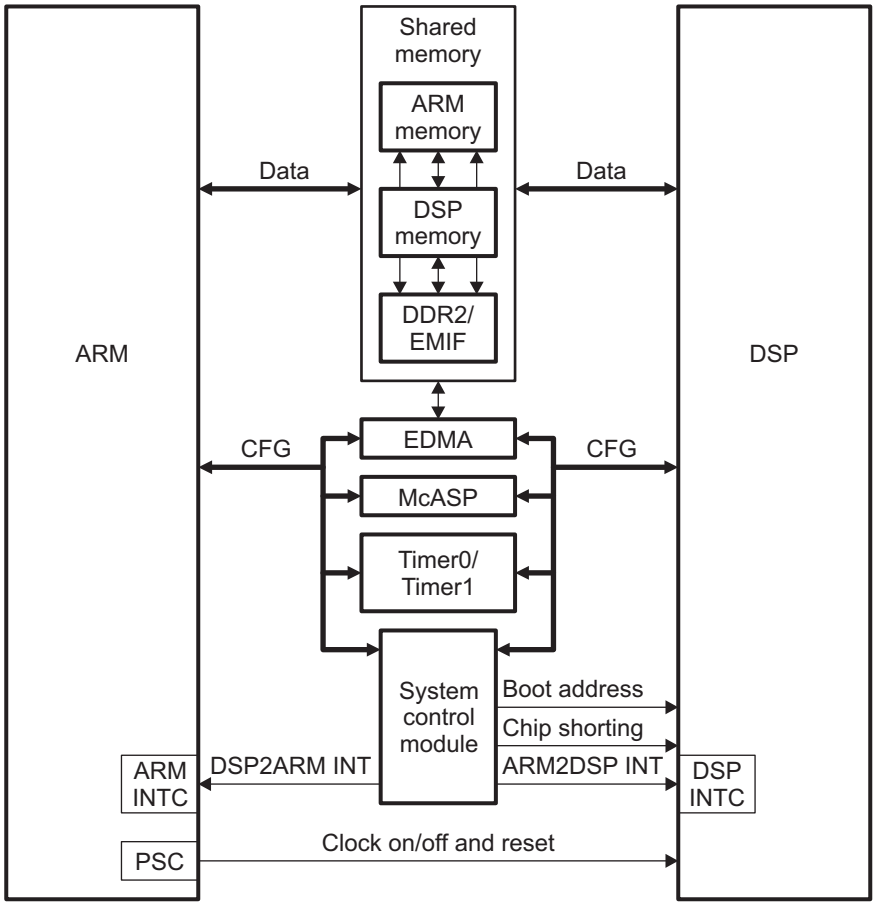
12.2 Shared Peripherals

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

- EDMA
- McASP
- Timer0 and Timer1

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

Figure 12-1. ARM-DSP Integration



12.3 Shared Memory

The DM646x 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 EMIFA. The EDMA can transfer data among shared memory without ARM or DSP intervention. See the *TMS320DM646x DMSoC Enhanced Direct Memory Access (EDMA) Controller User's Guide* (SPRUEQ5) for more information on the EDMA.

12.3.1 ARM Internal Memories

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

- 32 KB ARM internal RAM
- 8 KB ARM internal ROM

12.3.2 DSP Memories

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

- L1P RAM (32 KB)
- L1D RAM (32 KB)
- L2 RAM (128 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 *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8).

12.3.3 External Memories

Both the ARM and the DSP have access to devices connected to the DDR2 memory controller and the EMIFA. This allows the ARM and DSP to access program and data from DDR on the DDR2 memory controller and from devices attached to the EMIFA, such as NOR flash or SRAM.

NOTE: The DSP can access the data space of the DDR2 memory controller and of the EMIFA. 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 EMIFA.

12.4 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, ARM2DSP0, ARM2DSP1, ARM2DSP2, ARM2DSP3, and NMI, in the DSP interrupt event map. The DSP has access to one ARM interrupt event, DSP2ARM0, in the ARM interrupt event map. The ARM-DSP interrupts/events are summarized in Table 12-1.

Table 12-1. ARM-DSP Interrupt Mapping

		Source		Interrupt Number	
Name	**Description**	**ARM**	**DSP**	**AINTC**	**INTC**
DSP2ARM0	DSP Controller to ARM Interrupt	-	x	45	-
ARM2DSP0	ARM to DSP Controller 0 Interrupt	x	-	-	16
ARM2DSP1	ARM to DSP Controller 1 Interrupt	x	-	-	17
ARM2DSP2	ARM to DSP Controller 2 Interrupt	x	-	-	18
ARM2DSP3	ARM to DSP Controller 3 Interrupt	x	-	-	19
NMI	Nonmaskable Interrupt	x	-	-	-

The System Module includes registers for generating interrupts from the ARM to the DSP (DSPINT, DSPINTSET, and DSPINTCLR) and from the DSP to the ARM (DSPINT, DSPINTSET, and DSPINTCLR). See the device-specific data manual for details on these registers.

The ARM uses DSPINT, DSPINTSET, and DSPINTCLR to generate an interrupt to the DSP. The DSP interrupt status register (DSPINT) shows the status of the ARM-to-DSP interrupts. The ARM may generate an interrupt to the DSP by setting one of the four INTDSP*n* bits or the INTNMI bit in the DSP interrupt set register (DSPINTSET). The interrupt set (INTDSP*n*) bit then self-clears and the corresponding bit in DSPINT is automatically set to indicate that the interrupt was generated. After servicing the interrupt, the DSP clears the status bit in DSPINT by writing a 1 to the corresponding bit in the DSP interrupt clear register (DSPINTCLR). The ARM may poll the status bit in DSPINT to determine when the DSP has completed the interrupt service.

The DSP may generate an interrupt to the ARM in a similar manner using the ARM interrupt set register (ARMINTSET) and the ARM interrupt clear register (ARMINTCLR). The DSP can monitor the status of the DSP-to-ARM interrupts using the ARM interrupt status register (ARMINT).

For more information on ARM interrupts, see Chapter 8. For more information on DSP interrupts, see the DSP interrupts section of the *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8). For more information on the system control module, see Chapter 9.

12.5 ARM Control of DSP Boot, Clock, and Reset

As system master, the ARM can control all of the following functions:

- Boot the DSP
- Turn the clock on/off
- Reset the DSP

To initiate these 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 6 and Chapter 9.

NOTE: The DM646x DMSoC has only one power domain (Always On); therefore, the ARM has no control of the DSP power on/off.

12.5.1 DSP Boot

The DSP can boot in either of two modes: ARM boots DSP mode or DSP self-boot mode.

- In the ARM boots DSP mode, the ARM is responsible for managing the DSP boot after power-on/reset.
- In the DSP self-boot mode, the DSP boots without ARM intervention immediately upon power-on/reset.

The boot mode is determined by sampling the DSP boot source (DSPBOOT) pin at power-on/reset (Table 12-2). See Chapter 10 and Chapter 11 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 12-2. DSP Boot Configuration

Device Configuration	Function	Sampled Pin	Default Setting
DSP boot	0 = ARM boots DSP 1 = DSP self-boots	DSPBOOT	ARM boots DSP

To boot the DSP, the ARM must specify a boot address in the DSP boot address register (DSPBOOTADDR) in the System Module and ensure that the DSP program code is loaded properly into memory. When the ARM releases the DSP from reset, the DSP immediately begins code execution from the boot address programmed in DSPBOOTADDR. To boot the DSP:

1. Put the DSP module in the enable state. Prior to beginning the DSP boot sequence, the DSP module must be in the enable state. See Section 12.5.2 for information on how to execute DSP module clock on.
2. Clear the LRST bit in the module control 1 register (MDCTL1) in the PSC to 0. This asserts the DSP local reset. By default, after power-on reset or hard reset, the value of the LRST bit in MDCTL1 is cleared to 0.
3. Set the DSP boot address in DSPBOOTADDR. By default, after power-on or hard reset, the value in DSPBOOTADDR is 4220 0000h, which maps to the EMIFA. The ARM software can specify a boot address that maps to L1P internal DSP memory, EMIFA, or DDR2 memory controller. The DSP can execute program instructions from any of these memories.
4. Ensure that the DSP program code is loaded/stored with a reset vector at the DSP boot address specified in the previous step.
5. Set the LRST bit in MDCTL1 to 1. This deasserts the DSP local reset. After reset is deasserted, the DSP immediately begins code execution from the boot address programmed in DSPBOOTADDR.

12.5.2 DSP Module Clock ON/OFF

12.5.2.1 DSP Module Clock On (Enable)

In the clock enable state, the DSP's module clock is enabled while the DSP module reset is deasserted. This is the state for normal DSP run-time.

- ARM: Enable clocks to the DSP:
  1. Wait for the GOSTAT[0] bit in the power domain transition status register (PTSTAT) in the PSC to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  2. Set the NEXT bit in the module control 1 register (MDCTL1) in the PSC to 3h to prepare the DSP module for an enable transition.
  3. Set the GO[0] bit in the power domain transition command register (PTCMD) in the PSC to 1 to initiate the state transition.
  4. Wait for the GOSTAT[0] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[0] bit is cleared to 0.
  5. Wait for the STATE bit in the module status 1 register (MDSTAT1) in the PSC to change to 3h. The module is only safely in the new state after the STATE bit in MDSTAT1 changes to reflect the new state.
  6. Clocks are enabled to the DSP.
- ARM: Wake the DSP. If transitioning from the disable or idle state, trigger a DSP interrupt that has previously been configured as a wake-up interrupt, (for example, set the INTDSP*n* or INTNMI bit in the DSP interrupt set register (DSPINTSET) in the System Module.

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 10 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.

12.5.2.2 DSP Module Clock Off (Disable)

In the clock disable state, the DSP's module clock is disabled while the DSP module reset remains deasserted. 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:
  1. Set the power-down command register (PDCCMD) in the DSP power-down controller (PDC) module to 0001 5555h

  NOTE: PDCCMD can only be written while the DSP is in supervisor mode. See the *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8) and the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the power-down controller (PDC).

  2. Enable one of the interrupts: ARM2DSP0, ARM2DSP1, ARM2DSP2, ARM2DSP3, or NMI. This interrupt wakes the DSP in the DSP clock-on sequence.
  3. 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 set in PDCCMD. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the PDC and the IDLE instruction.
- ARM: Disable the DSP clock:
  1. Wait for the GOSTAT[0] bit in the power domain transition status register (PTSTAT) in the PSC to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  2. Set the NEXT bit in the module control 1 register (MDCTL1) in the PSC to 2h to prepare the DSP module for a disable transition.
  3. Set the GO[0] bit in the power domain transition command register (PTCMD) in the PSC to 1 to initiate the state transition.
  4. Wait for the GOSTAT[0] bit in PTSTAT to clear to 0. The domain is only safely in the new state after the GOSTAT[0] bit is cleared to 0.
  5. Wait for the STATE bit in the module status 1 register (MDSTAT1) in the PSC to change to 2h. The module is only safely in the new state after the STATE bit in MDSTAT1 changes to reflect the new state.
  6. Clocks to the DSP are disabled.

12.5.3 DSP Reset

With access to the PSC registers, the ARM can assert and deassert 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 6. For more information on local reset and on module reset, see Chapter 10. This section describes how to initiate DSP local reset and module reset.

12.5.3.1 DSP Local Reset

To assert/deassert local reset:

1. Clear the LRST bit in the module control 1 register (MDCTL1) in the PSC to 0 to assert DSP reset.
2. Set the LRST bit in MDCTL1 to 1 to deassert DSP reset.

12.5.3.2 DSP Module Reset

12.5.3.2.1 Software Reset Disable (SwRstDisable)

In the software reset disable (SwRstDisable) 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:
  1. Set the power-down command register (PDCCMD) in the DSP power-down controller (PDC) module to 0001 5555h.

     NOTE: PDCCMD can only be written while the DSP is in supervisor mode. See the *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8) and the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the power-down controller (PDC).

  2. 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 initiates DSP power-down according to the bits set in PDCCMD. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the PDC and the IDLE instruction.
- ARM: Software reset disable DSP:
  1. Wait for the GOSTAT[0] bit in the power domain transition status register (PTSTAT) in the PSC to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  2. Clear the NEXT bit in the module control 1 register (MDCTL1) in the PSC to 0 to prepare the DSP module for a software reset disable transition.
  3. Set the GO[0] bit in the power domain transition command register (PTCMD) in the PSC to 1 to initiate the state transition.
  4. Wait for GOSTAT[0] bit in PTSTAT to clear to 0. The domain is safely in the new state only after the GOSTAT[0] bit is cleared to 0.
  5. Wait for the STATE bit in the module status 1 register (MDSTAT1) in the PSC to change to 0. The module is safely in the new state only after the STATE bit in MDSTAT1 changes to reflect the new state.
- ARM: Assert the DSP local reset. Clear the LRST bit in the module control 1 register (MDCTL1) in the PSC 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 deasserts local reset sometime after finishing the enable sequence.

12.5.3.2.2 Synchronous Reset (SyncReset)

In the synchronous reset (SyncReset) 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:
  1. Set the power-down command register (PDCCMD) in the DSP power-down controller (PDC) module to 0001 5555h.

     NOTE: PDCCMD can only be written while the DSP is in supervisor mode. See the *TMS320DM646x DMSoC DSP Subsystem Reference Guide* (SPRUEP8) and the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the power-down controller (PDC).

  2. 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 initiates DSP power-down according to the bits set in PDCCMD. See the *TMS320C64x+ DSP Megamodule Reference Guide* (SPRU871) for more information on the PDC and the IDLE instruction.
- ARM: Sync reset DSP:
  1. Wait for the GOSTAT[0] bit in the power domain transition status register (PTSTAT) in the PSC to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
  2. Set the NEXT bit in the module control 1 register (MDCTL1) in the PSC to 1 to prepare the DSP module for a sync reset transition.
  3. Set the GO[0] bit in the power domain transition command register (PTCMD) in the PSC to 1 to initiate the state transition.
  4. Wait for GOSTAT[0] bit in PTSTAT to clear to 0. The domain is safely in the new state only after the GOSTAT[0] bit is cleared to 0.
  5. Wait for the STATE bit in the module status 1 register (MDSTAT1) in the PSC to change to 1. The module is safely in the new state only after the STATE bit in MDSTAT1 changes to reflect the new state.
- ARM: Assert DSP local reset. Clear the LRST bit in the module control 1 register (MDCTL1) in the PSC 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 deasserts local reset sometime after finishing the 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 5-8	Changed Value range for PLLM bit.

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