TMS320DM36x Digital Media System-on-Chip (DMSoC) ARM Subsystem

User's Guide

Literature Number: SPRUFG5A
April 2009 – Revised August 2009

List of Figures

List of Tables

Read This First

About This Manual

This document describes the operation of the ARM subsystem in the TMS320DM36x 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 TMS320DM36x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the internet at www.ti.com.

SPRUFG5 — *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* This document describes the ARM Subsystem in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the components of the ARM Subsystem, the peripherals, and the external memories.

SPRUFG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) Users Guide* This document describes the Video Processing Front End (VPFE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFG9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Users Guide* This document describes the Video Processing Back End (VPBE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) 64-bit Timer Users Guide* This document describes the operation of the software-programmable 64-bit timers in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Serial Peripheral Interface (SPI) Users Guide* This document describes the serial peripheral interface (SPI) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.

SPRUFH2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Asynchronous Receiver/Transmitter (UART) Users Guide* This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

SPRUFH3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Inter-Integrated Circuit (I2C) Peripheral Users Guide* This document describes the inter-integrated circuit (I2C) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The I2C peripheral provides an interface between the DMSoC and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus.

SPRUFH5 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multimedia Card (MMC)/Secure Digital (SD) Card Controller Users Guide* This document describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH6 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Pulse-Width Modulator (PWM) Users Guide* This document describes the pulse-width modulator (PWM) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH7 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Real-Time Out (RTO) Controller Users Guide* This document describes the Real Time Out (RTO) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) General-Purpose Input/Output (GPIO) Users Guide* This document describes the general-purpose input/output (GPIO) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

SPRUFH9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Users Guide* This document describes the universal serial bus (USB) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.

SPRUFI0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Enhanced Direct Memory Access (EDMA) Controller Users Guide* This document describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EDMA controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.

SPRUFI1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (EMIF) Users Guide* This document describes the asynchronous external memory interface (EMIF) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.

SPRUFI2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Users Guide* This document describes the DDR2/mDDR memory controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The DDR2/mDDR memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM and mobile DDR devices.

SPRUFI3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multibuffered Serial Port Interface (McBSP) User's Guide* This document describes the operation of the multibuffered serial host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the McBSP are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP supports general serial port receive and transmit operation.

SPRUFI4 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Host Port Interface (UHPI) User's Guide* This document describes the operation of the universal host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI5 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Ethernet Media Access Controller (EMAC) User's Guide* This document describes the operation of the ethernet media access controller interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI7 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Analog to Digital Converter (ADC) User's Guide* This document describes the operation of the analog to digital conversion in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Key Scan User's Guide* This document describes the key scan peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Voice Codec User's Guide* This document describes the voice codec peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). This module can access ADC/DAC data with internal FIFO (Read FIFO/Write FIFO). The CPU communicates to the voice codec module using 32-bit-wide control registers accessible via the internal peripheral bus.

SPRUFJ0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Power Management and Real-Time Clock Subsystem (PRTCSS) User's Guide* This document provides a functional description of the Power Management and Real-Time Clock Subsystem (PRTCSS) in the TMS320DM36x Digital Media System-on-Chip (DMSoC) and PRTC interface (PRTCIF).

SPRUGG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Face Detection User's Guide*This document describes the face detection capabilities for the TMS320DM36x Digital Media System-on-Chip (DMSoC).

Introduction

1 Device Overview

Developers can now deliver pixel-perfect images in their digital video designs without concerns of video format support, constrained network bandwidth, limited system storage capacity or cost, with the new TMS320DM36x digital media processor based on DaVinci technology from Texas Instruments Incorporated (TI). Along with multi-format HD video, the device also features an integrated image signal processing (ISP) solution and a suite of peripherals which saves developers on system costs. This ARM9-based DM36x device supports video accelerators that offload compression needs from the ARM core so that developers can utilize the most performance from the ARM for their application. Video surveillance designers achieve greater compression efficiency and provide more storage without straining the network bandwidth. Developers of media playback and camera-driven applications, such as video doorbells, digital signage, digital video recorders, portable media players and more, can ensure interoperability as well as product scalability by taking advantage of the full suite of codecs supported on the device.

DM36x enables a seamless interface to most additional external devices required for video applications. The image sensor interface is flexible enough to support CCD, CMOS, and various other interfaces such as BT.565 and BT1120. The device also offers a high level of integration with HD display support including, three built-in 10-bit HD analog video digital to analog converters (DACs), DDR2/mDDR, Ethernet MAC, USB 2.0, integrated audio, host port interface (HPI), analog to digital converter, and many more features, saving developers on overall system costs as well as real estate on their circuit boards allowing for a slimmer, sleeker design.

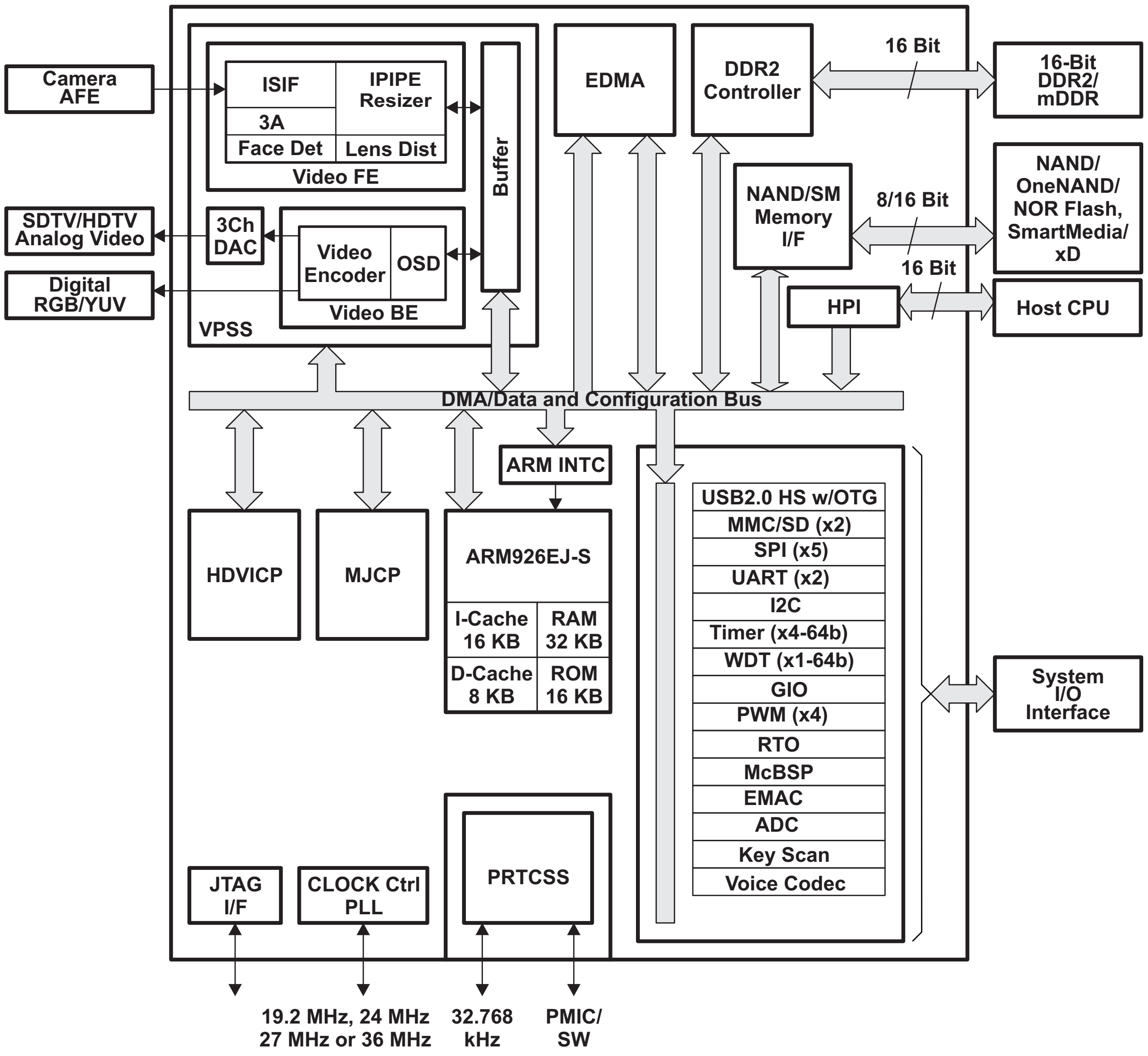
1.1 Block Diagram

The device consists of the following primary components and subsystems:

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

The detailed block diagram is shown in Figure 1.

Figure 1. Functional Block Diagram



2 ARM Subsystem

The ARM926EJ-S 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. Section 2.2 describes the ARMSS components and system control functions that the ARM core performs.

2.1 Purpose of the ARM Subsystem

The ARM subsystem contains components required to provide master control of the overall system to the ARM926EJ-S (ARM), including control over the VPSS subsystem, the peripherals, and external memories.

This subsystem is also responsible for handling system functions such as system-level initialization, configuration, user interface, user command execution, connectivity functions, etc. The ARM is master and performs these functions because it has a large program memory space and fast context switching capability, and is thus suitable for complex, multi-tasking, and general-purpose control tasks.

2.2 Components of the ARM Subsystem

The ARM subsystem (ARMSS) consists of the following components:

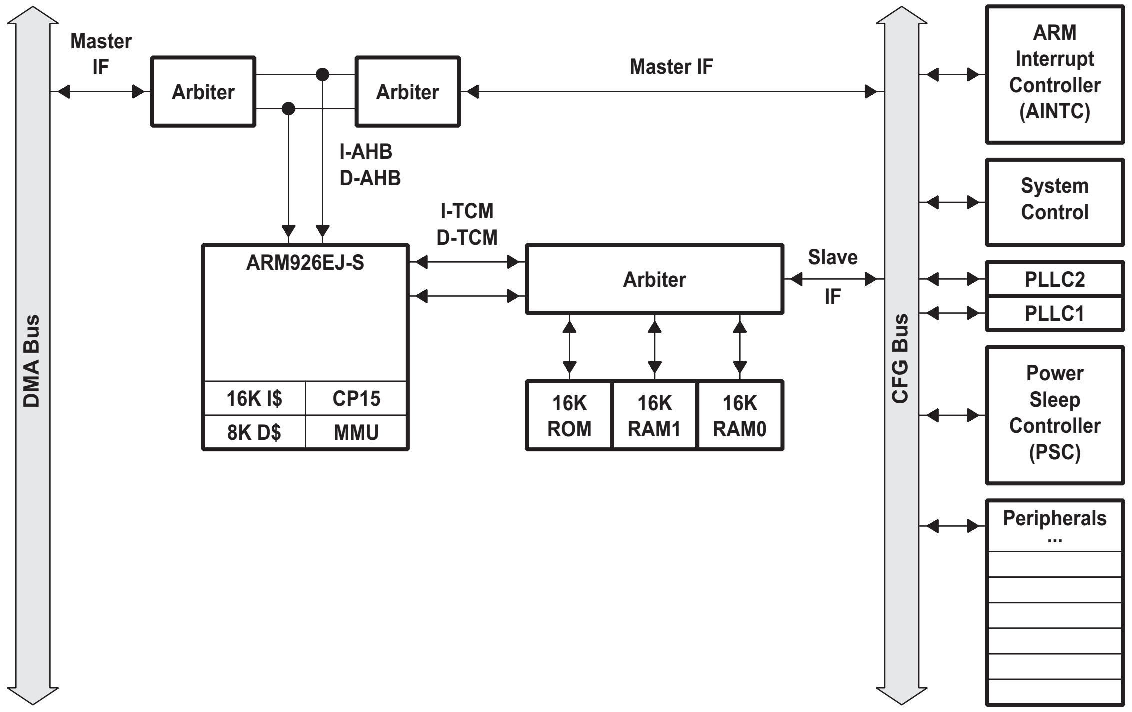
- ARM926EJ-S RISC processor, including:
  - Coprocessor 15 (CP15)
  - MMU
  - 16KB Instruction cache
  - 8KB Data cache
  - Write Buffer
  - Java accelerator
- ARM Internal Memories
  - 32KB Internal RAM (32-bit wide access)
  - 16KB Internal ROM (ARM bootloader for non-AEMIF boot options)
- Embedded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

The ARMSS also manages/controls the following peripherals:

- DDR2/mDDR EMIF controller
- AEMIF Controller
- Enhanced DMA (EDMA)
- Universal Asynchronous Receiver/Transmitter (UART)
- Timers
- Real-Time Out (RTO)
- Pulse Width Modulator (PWM)
- Inter-IC Communication (I2C)
- Multimedia Card/Secure Digital (MMC/SD)
- Multichannel Buffered Serial Port (McBSP)
- Universal Serial Bus Controller (USB)
- Ethernet Media Access Controller (EMAC)
- Management Data Input/Output (MDIO)
- Host Port Interface (HPI) Peripheral
- Key Scan
- Analog-to-Digital Converter (ADC)
- Voice Codec
- Serial Port Interface (SPI)
- Video Processing Front End (VPFE)
- Video Processing Back End (VPBE)

Figure 2 shows the functional block diagram of the ARM subsystem.

Figure 2. ARM Subsystem Block Diagram



2.3 References

See the following related documents for more information:

- TMS320DM36x peripheral users guides other than the ARM Core, found in .
- For more detailed information about the ARM processor core, see the ARM926EJ-S Technical Reference Manual found at the ARM Ltd. web site:
  - http://www.arm.com/documentation/ARMProcessor_Cores/index.html

3 ARM Core

3.1 Introduction

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

- ARM926EJ-S - 32-bit RISC processor
- 16-KB instruction cache
- 8-KB data cache
- MMU
- CP15 to control MMU, cache, write buffer, etc.
- Java accelerator
- ARM Internal Memory
  - 32-KB built-in RAM
  - 16-KB built-in ROM (boot ROM)
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB). Features include:
  - 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 ARM instructions

For maximum operating clock frequency see the device-specific data manual.

The ARM926EJ-S processor is a member of the ARM9 family of general-purpose microprocessors, and targets multi-tasking applications where full memory management, high performance, low die size, and low power are all important. The processor also supports the 32-bit ARM and the 16-bit THUMB instruction sets, enabling a trade-off between high-performance and high-code density. This includes features for efficient execution of Java byte codes and provides 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 Harvard architecture included in this processor 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

Additional features about the ARM926EJ-S processor are provided below:

- Implements ARM architecture version 5TEJ.
- Includes new signal processing extensions to enhance 16-bit fixed-point performance using a single-cycle 32 x 16 multiply-accumulate (MAC) unit. The ARM subsystem also has 32KB of internal RAM and 16KB 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 DM36x 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, you must generate a software interrupt (SWI). An IRQ interrupt causes the processor to enter the IRQ mode, while 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.

NOTE: See the ARM926EJ-S TRM, downloadable from http://www.arm.com for more detailed information.

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. PSR [7:0] are the processor control bits, PSR [27:8] are reserved bits, and PSR [31:28] are status bits. The control bits, PSR[7:0], are defined as follows:

- 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

The status bits, PSR[31:28], reflect the result of the most recent ALU operation. The status bits are defined as follows:

- 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 the Programmer’s Model of the ARM926EJ-S TRM, downloadable from http://www.arm.com 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; 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 0x00000000 (VINTH = 0) or at address 0xFFFF0000 (VINTH = 1). This is a feature of the ARM926EJ-S core. However, in this DMSoC there is no memory in the address region starting at 0xFFFF0000, so do not set VINTH.

The default vector table is shown in Table 1.

NOTE: See ARM926EJ-S TRM, downloadable from http://www.arm.com, for more detailed information.

Table 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
04h	Undefined instruction	Undefined	Set	Unchanged
08h	Software interrupt	Supervisor	Set	Unchanged
0Ch	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-S 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. Additionally, 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

The 16-bit instructions operate with the standard 32-bit register configuration, allowing excellent interoperability 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 (e.g., code that performs character string handling), and some instructions (like branches) do not process any data at all. If a 16-bit architecture has only 16-bit instructions, and a 32-bit architecture has only 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 a 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.

NOTE: See the ARM926EJ-S TRM, downloadable from http://www.arm.com for more detailed information.

3.6 Coprocessor 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

Table 2 displays the three different types of addresses that exist in an ARM926EJ-S system.

Table 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 1. Address Manipulation.

Example 1.

Here are some guidelines concerning 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 for more detailed information.

3.6.2 Memory Management Unit

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

The MMU features are as follows:

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

NOTE: See the Memory Management Unit of the ARM926EJ-S TRM, downloadable from http://www.arm.com 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 8KB, instruction cache is 16KB, 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 support for 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
- Ability to perform critical-word first cache refilling
- Cache lockdown registers that enable control over which cache ways are used for allocation on a linefill, providing a mechanism for both lockdown and controlling cache pollution
- Dcache that 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:
  – Entire Dcache or Icache
  – Regions of the Dcache or Icache
  – Entire Dcache
  – Regions of virtual memory
- Provide operations for efficient cleaning and invalidation of the following:
  – Entire Dcache
  – Regions of the Dcache
  – Regions of virtual memory
- Write buffer 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.
- Main write buffer with a 16-word data buffer and a four-address buffer
- Dcache write-back with eight data word entries and a single address entry
- MCR drain write buffer that enables both write buffers to be drained under software control
- MCR wait for interrupt which 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 for more detailed information.

3.7 Tightly Coupled Memory

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

The DM36x processor supports both instruction TCM (I-TCM) and data TCM (D-TCM). The instruction TCM is located at 0x0000:0000 to 0x0000:7FFF. The data TCM is located at 0x0001:0000 to 0x0001:7FFF, as shown in Table 3.

Table 3. ITCM/DTCM Memory Map

I-TCM Address	D-TCM Address	Size (Bytes)	Description
0x0000 :0000 - 0x0000 :3FFF	0x0001 :0000 - 0x0001 :3FFF	16K	IRAM0
0x0000 :4000 - 0x0000 :7FFF	0x0001 :4000 - 0x0001 :7FFF	16K	IRAM1
0x0000 :8000 - 0x0000 :BFFF	0x0001 :8000 - 0x0001 :BFFF	16K	ROM
0x0000 :C000 - 0x0000 :FFFF	0x0001 :C000 - 0x000F :FFFF	16K	Reserved

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

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

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

The format of the data in the TCM register is as shown below:

Figure 3. TCM Status Register

31–17	16
SBZ/UNP	DTCM

15–1	0
SBZ/UNP	ITCM

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

If the DTCM bit is 0, Data TCM is not present and if the DTCM bit is 1, Data TCM is present.

If the ITCM bit is 0, Instruction TCM is not present and if the ITCM bit is 1, Instruction TCM is present.

Use the ITCM / DTCM region registers to enable ITCM and DTCM.

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

```
MRC p15, #0, Rd, c9, c0, #0 ; read DTCM region register MCR p15, #0, Rd, c9, c0, #0 ; write DTCM
region register MRC p15, #0, Rd, c9, c0, #1 ; read ITCM region register MCR p15, #0, Rd, c9, c0,
#1 ; write ITCM region register
```

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

The format of the data in the TCM register is shown below:

Figure 4. TCM Register

31–16
ADDRESS

15–12	11–6	5–2	1	0
ADDRESS	SBZ/UNP	SIZE	SBZ/UNP	ENB
				R/W-1

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

Write 0 to the ENB bit to enable ITCM and DTCM. Write 1 to the ENB bit to enable it. The physical address of the memory should be set to the ADDRESS field. The SIZE field reflects the size. The size encoding is given below in Table 4.

Table 4. ITCM/DTCM Size Encoding

Binary Code	Size
0000	0 KB / absent
0001,0010	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

NOTE: See device clocking of the Tightly-Coupled Memory Interface of the ARM926EJ-S TRM, downloadable from http://www.arm.com for more detailed information.

Use the values 0x00010019 to enable DTCM for DM36x: 0x00010000 (base address) | 0b0110 << 2 (size) | 1 (enable)

3.8 *Embedded Trace Support*

To support real-time trace, the ARM926EJ-S processor provides an interface to enable connection of an embedded trace macrocell (ETM). The ARM926ES-J subsystem also includes the embedded trace buffer (ETB).

The ETM consists of two parts: the trace port and triggering facilities. The two ETM parts are shown in Table 5.

NOTE: **Note:** The device's trace port is not pinned out. Instead, it is connected to a 4KB embedded trace buffer. ETB-enabled debug tools are required to read/interpret the captured trace data.

Table 5. ETM Part Descriptions

ETM Part	Description
Trace Port	The trace port allows you to debug the processor. The trace port has a protocol that has been developed to provide a real-time trace capability for processor cores that are deeply embedded in large ASIC designs. This is beneficial to developers and manufacturers when it is not possible to determine how the processor core is operating by only observing the pins of the ASIC.
Triggering Facilities	An extensible ETM specification exists to specify the exact set of trigger resources required for a particular application. Resources include address and data comparators, counters, and sequencers.

The ETM is used to compress the trace information and export it through a narrow trace port. An external trace port analyzer (TPA) is used to capture the trace information.

NOTE: See the embedded trace macro-cell support of the ARM926EJ-S TRM, downloadable from http://www.arm.com for more detailed information.

4 Memory Mapping

4.1 Memory Map

The memory map is shown in Table 6 and Table 7 depicts the expanded map of the configuration space (0x01C0 0000 through 0x01FF FFFF). The multiple columns in the table represent the memory map of each of the masters on the chip. The device has multiple on-chip memories associated with its processor and various subsystems. The various bus masters in the device are the ARM, EDMA, EMAC, USB, HPI, MJCP, HDVICP, and VPSS range.

Table 6. Memory Map

Start Address	End Address	Size (Bytes)	ARM Mem Map	EDMA Mem Map	Master Periph Mem Map	VPSS Mem Map
0x0000 0000	0x0000 3FFF	16K	ARM RAM0 (Instruction)	Reserved	Reserved	Reserved
0x0000 4000	0x0000 7FFF	16K	ARM RAM1 (Instruction)	Reserved	Reserved	Reserved
0x0000 8000	0x0000 BFFF	16K	ARM ROM (Instruction)	Reserved	Reserved	Reserved
0x0000 C000	0x0000 FFFF	16k	Reserved	Reserved	Reserved	Reserved
0x0001 0000	0x0001 3FFF	16K	ARM RAM0 (Data)	ARM RAM0	ARM RAM0	Reserved
0x0001 4000	0x0001 7FFF	16K	ARM RAM1 (Data)	ARM RAM1	ARM RAM1	Reserved
0x0001 8000	0x0001 BFFF	16K	ARM ROM (Data)	ARM ROM	ARM ROM	Reserved
0x0001 C000	0x000F FFFF	912K	Reserved	Reserved	Reserved	Reserved
0x0010 0000	0x01BB FFFF	26M	Reserved	Reserved	Reserved	Reserved
0x01BC 0000	0x01BC 0FFF	4K	ARM ETB Mem	Reserved	Reserved	Reserved
0x01BC 1000	0x01BC 17FF	2K	ARM ETB Reg	Reserved	Reserved	Reserved
0x01BC 1800	0x01BC 18FF	256	ARM IceCrusher	Reserved	Reserved	Reserved
0x01BC 1900	0x01BC FFFF	59136	Reserved	Reserved	Reserved	Reserved
0x01BD 0000	0x01BF FFFF	192K	Reserved	Reserved	Reserved	Reserved
0x01C0 0000	0x01FF FFFF	4M	CFG Bus Peripherals	CFG Bus Peripherals	CFG Bus Peripherals	Reserved
0x0200 0000	0x09FF FFFF	128M	ASYNC EMIF (Data)	ASYNC EMIF (Data)	Reserved	Reserved
0x0A00 0000	0x11EF FFFF	127M - 16K	Reserved	Reserved	Reserved	Reserved
0x11F0 0000	0x11F1 FFFF	128K	MJCP DMA Port	MJCP DMA Port	Reserved	Reserved
0x11F2 0000	0x11FF FFFF	896K	Reserved	Reserved	Reserved	Reserved
0x1200 0000	0x1207 FFFF	512K	HDVICP DMA Port1	HDVICP DMA Port1	Reserved	HDVICP DMAPort1
0x1208 0000	0x120F FFFF	512K	Reserved	HDVICP DMA Port2	Reserved	Reserved
0x1210 0000	0x1217 FFFF	512k	Reserved	HDVICP DMA Port3	Reserved	Reserved
0x1218 0000	0x1FFF FFFF	2225M	Reserved	Reserved	Reserved	Reserved
0x2000 0000	0x2000 7FFF	32K	DDR2 EMIF Control Regs	DDR2 EMIF Control Regs	Reserved	Reserved
0x2000 8000	0x41FF FFFF	544M-32K	Reserved	Reserved	Reserved	Reserved
0x4200 0000	0x49FF FFFF	128M	Reserved	Reserved	Reserved	Reserved
0x4A00 0000	0x7FFF FFFF	864M	Reserved	Reserved	Reserved	Reserved
0x8000 0000	0x8FFF FFFF	256M	DDR2 EMIF	DDR2 EMIF	DDR2 EMIF	DDR2 EMIF
0x9000 0000	0xFFFF FFFF	1792M	Reserved	Reserved	Reserved	Reserved

4.1.1 ARM Internal Memories

The ARM has access to the following ARM internal memories:

- 32KB ARM Internal RAM on TCM interface, logically separated into two 16KB pages to allow simultaneous access on any given cycle, if there are separate accesses for code (I-TCM) and data (D-TCM) to the different memory regions.
- 16KB ARM Internal ROM

4.1.2 External Memories

The ARM has access to the following external memories:

- DDR2/mDDR Synchronous DRAM
- Asynchronous EMIF/OneNand
- NAND Flash
- NOR Flash
- Flash card devices:
  - MMC/SD
  - XD
  - SmartMedia

4.1.3 MPEG/JPEG Coprocessor (MJCP)

The device's performance is enhanced by its dedicated hard-wired MPEG/JPEG coprocessor (MJCP). The MJCP performs all the computational operations required for JPEG and MPEG4 compression. These operations can be invoked using the xDM (xDAIS for Digital Media) APIs. For more information, refer to the *xDAIS-DM (Digital Media) User's Guide* (SPRUEC8).

4.1.4 Peripherals

The ARM and EDMA have access to the configuration registers and memories of the following peripherals (see Table 7):

- EDMA Controller
- Universal Asynchronous Receiver/Transmitter (UART)
- Inter-IC Communication (I2C)
- Timers and One WDT(Watch Dog Timer)
- Pulse Width Modulator (PWM)
- Universal Serial Bus Controller (USB)
- Multichannel Buffered Serial Port (McBSP)
- Ethernet Media Access Controller (EMAC)
- Management Data Input/Output (MDIO)
- Host Port Interface (HPI)
- Key Scan
- Multimedia Card/Secure Digital (MMC/SD)
- Analog-to-Digital Converter (ADC)
- Voice Codec
- SPI Serial Interfaces
- General-Purpose Input/Output (GPIO)
- Video Processing Subsystem (VPSS)
- Asynchronous EMIF (AEMIF) Controller
- Real-Time Out (RTO)
- MPEG/JPEG Coprocessor (MJCP)
- Power Management and Real-Time Clock Subsystem (PRTCSS)
- High-Definition Video Image Coprocessors (HDVICP)

The ARM and EDMA also have access to the following internal peripherals:

- ETM/ETB
- ICEcrusher
- System Module
- PLL Controllers
- Power Sleep Controller
- ARM Interrupt Controller

Table 7. ARM Configuration Bus Access to Peripherals

	Address		
Region	**Start**	**End**	**Size**
EDMA CC	0x01C0 0000	0x01C0 FFFF	64K
EDMA TC0	0x01C1 0000	0x01C1 03FF	1K
EDMA TC1	0x01C1 0400	0x01C1 07FF	1K
EDMA TC2	0x01C1 0800	0x01C1 0BFF	1K
EDMA TC3	0x01C1 0C00	0x01C1 0FFF	1K
Reserved	0x01C1 1000	0x01C1 FFFF	60K
UART0	0x01C2 0000	0x01C2 03FF	1K
Reserved	0x01C2 0400	0x01C2 07FF	1K
Timer3	0x01C2 0800	0x01C2 0BFF	1K
Real-time out	0x01C2 0C00	0x01C2 0FFF	1K
I2C	0x01C2 1000	0x01C2 13FF	1K
Timer0	0x01C2 1400	0x01C2 17FF	1K
Timer1	0x01C2 1800	0x01C2 1BFF	1K
Timer2/WDT	0x01C2 1C00	0x01C2 1FFF	1K
PWM0	0x01C2 2000	0x01C2 23FF	1K
PWM1	0x01C2 2400	0x01C2 27FF	1K
PWM2	0x01C2 2800	0x01C2 2BFF	1K
PWM3	0x01C2 2C00	0x01C2 2FFF	1K
SPI4	0x01C2 3000	0x01C2 37FF	2K
Timer4	0x01C2 3800	0x01C2 3BFF	1K
ADC	0x01C2 3C00	0x01C2 3FFF	1K
Reserved	0x01C2 4000	0x01C3 4FFF	112K
System Module	0x01C4 0000	0x01C4 07FF	2K
PLL Controller 1	0x01C4 0800	0x01C4 0BFF	1K
PLL Controller 2	0x01C4 0C00	0x01C4 0FFF	1K
Power/Sleep Controller	0x01C4 1000	0x01C4 1FFF	4K
Reserved	0x01C4 2000	0x01C4 7FFF	24K
ARM Interrupt Controller	0x01C4 8000	0x01C4 83FF	1K
Reserved	0x01C4 8400	0x01C6 3FFF	111K
USB OTG 2.0 Regs / RAM	0x01C6 4000	0x01C6 5FFF	8K
SPI0	0x01C6 6000	0x01C6 67FF	2K
SPI1	0x01C6 6800	0x01C6 6FFF	2K
GPIO	0x01C6 7000	0x01C6 77FF	2K
SPI2	0x01C6 7800	0x01C6 7FFF	2K
SPI3	0x01C6 8000	0x01C6 87FF	2K
Reserved	0x01C6 8800	0x01C6 87FF	2K
PRTCIF	0x01C6 9000	0x01C6 93FF	1K
KEYSCAN	0x01C6 9400	0x01C6 97FF	1K

Table 7. ARM Configuration Bus Access to Peripherals (continued)

	Address		
HPI	0x01C6 9800	0x01C6 9FFF	2K
Reserved	0x01C6 A000	0x01C6 FFFF	24K
VPSS Subsystem	0x01C7 0000	0x01C7 FFFF	64K
Reserved	0x01C8 0000	0x01C9 FFFF	128K
MJCP	0x01CA 0000	0x01CB FFFF	128K
Reserved	0x01CC 0000	0x01CF FFFF	256K
MMC/SD1	0x01D0 0000	0x01D0 1FFF	8K
McBSP	0x01D0 2000	0x01D0 3FFF	8K
Reserved	0x01D0 4000	0x01D0 5FFF	8K
UART1	0x01D0 6000	0x01D0 63FF	1K
Reserved	0x01D0 6400	0x01D0 6FFF	3K
EMAC Control Regs	0x01D0 7000	0x01D0 7FFF	4K
EMAC Wrap RAM	0x01D0 8000	0x01D0 9FFF	8K
EMAC Wrap Control Regs	0x01D0 A000	0x01D0 AFFF	4K
EMAC_MDIO	0x01D0 B000	0x01D0 B7FF	2K
VoiceCodec	0x01D0 C000	0x01D0 C3FF	1K
Reserved	0x01D0 C400	0x01D0 FFFF	17K
ASYNC EMIF Control	0x01D1 0000	0x01D1 0FFF	4K
MMC/SD0	0x01D1 1000	0x01D1 FFFF	60K
Reserved	0x01D2 0000	0x01DF FFFF	896K
HDVICP	0x01E0 0000	0x01FF FFFF	2M
ASYNC EMIF Data (CE0)	0x0200 0000	0x03FF FFFF	32M
ASYNC EMIF Data (CE1)	0x0400 0000	0x05FF FFFF	32M
Reserved	0x0600 0000	0x09FF FFFF	64M
Reserved	0x0A00 0000	0x0FFF FFFF	96M

4.2 Memory Interfaces

This section describes the types of memory interfaces supported by the device. These include :

- DDR2/mDDR Synchronous DRAM
- Asynchronous EMIF (NAND Flash, NOR flash, OneNAND Flash functions, and SRAM)

4.2.1 DDR2 EMIF

The DDR2 EMIF interface is a dedicated interface to DDR2 and mDDR SDRAM. It supports only 16-bit wide DDR2 SDRAM devices which are JESD79D-2A standard-compliant. The DDR2 SDRAM device also plays a key role in this system, which 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 VPFE
- Numerous OSD display buffers
- Intermediate buffering for large raw Bayer data image files while performing still camera processing functions
- Buffering for intermediate data while performing video encode and decode functions
- Storage of executable firmware for the ARM

The asynchronous external memory interface (AEMIF) provides an 8-bit or 16-bit data bus, an address bus width of up to 23 bits for 16-bit and 8-bit, and two dedicated chip selects, along with memory control signals. The EMIF module supports:

- NAND flash memories
- OneNAND/NOR flash memories

4.2.2 Asynchronous EMIF (AEMIF)

The asynchronous EMIF (AEMIF) mode supports these features:

- SRAM up to two asynchronous chip selects
- 8-bit or 16-bit data bus widths
- Programmable asynchronous cycle timings
- Extended waits
- Select strobe mode
- Booting of the ARM processor from CE0 (e.g., SRAM) via direct execution
- NOR flash

4.2.3 Asynchronous EMIF Features for NAND Device

The AEMIF supports the following features when interfaced to a NAND:

- NAND Flash up to two asynchronous chip selects
- 8-bit and 16-bit data bus widths
- Programmable cycle timings
- 1-bit and 4-bit ECC calculation (does not perform error correction)
- SmartMedia/SSFDC (Solid State Floppy Disk Controller) and xD memory cards
- Booting of the ARM processor from NAND-Flash located at CE0

4.2.4 Asynchronous EMIF Features for oneNAND Device

The AEMIF supports the following features when interfaced to a oneNAND device:

- oneNAND flash up to two chip selects
- Only 16-bit data bus widths
- Asynchronous writes and reads
- Synchronous reads with continuous linear burst mode
- Programmable cycle timings for each chip select in asynchronous mode
- Booting of the DM36x ARM processor from oneNAND Flash located at CE0 via direct execution

5 Device Clocking

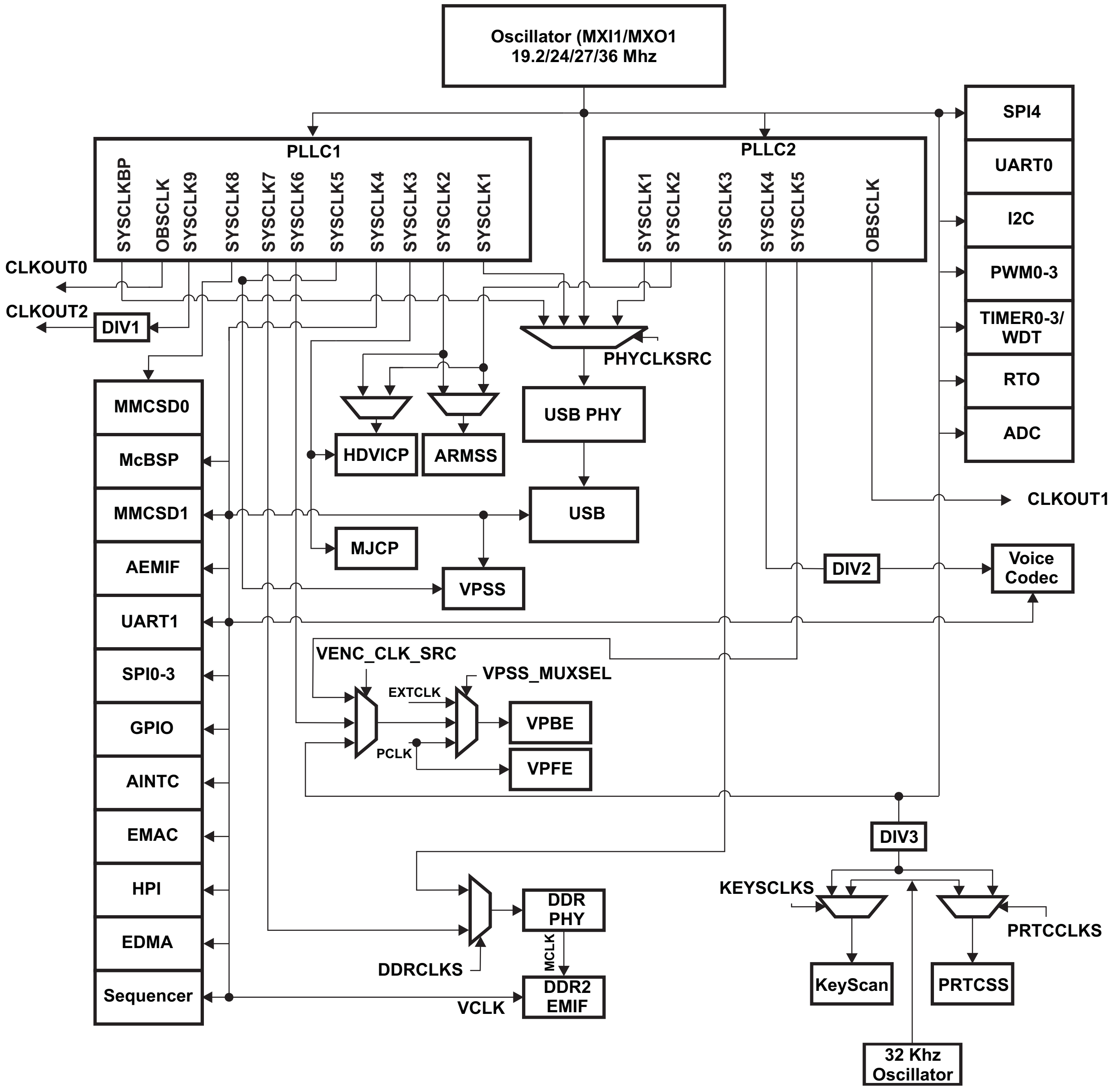
5.1 Overview

The device requires one primary reference clock at the MXI1/MXO1 pins, and drives two separate PLL controllers: PLLC1 and PLLC2. PLLC1 generates the clocks required by the ARM, EDMA, VPSS, and the rest of the peripherals as shown in Table 8. PLLC2 generates the clock required by the DDR2 PHY interface and is also capable of providing clocks to the ARM, USB, VPSS, or voice codec modules as shown in Table 9.

Figure 5 represents the clocking architecture for the ARM subsystem. For more information on the system PLL controller see Section 6.

Refer to the device-specific data manual for information on supported device clocking configurations (e.g., supported PLL configurations).

Figure 5. Clocking Architecture



5.2 Peripheral Clocking Considerations

5.2.1 Video Processing Back End Clocking

The Video Processing Back End (VPBE) is a submodule of the Video Processing Subsystem (VPSS). See the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Users Guide* (SPRUFG9) for complete information on VPBE clocking.

5.2.2 USB Clocking

Details of USB clocking are listed below:

- The USB PHY clock source is selected from PLLC1SYSCLK1, PLLC1SYSCLKBP, PLLC1AUXCLK, or PLLC2SYCLK1.
- The USB controller is driven by two clocks: an output clock of the PLLC1 or PLLC2 controllers and an output clock of the USB PHY.
- The USB PHY takes an input clock that is configurable by the USB PHY clock source bits (PHYCLKSRC) in the USB PHY control register (USB_PHY_CTRL) as described in Section 9.12.14.
- When a 24 MHz/19.2 MHz crystal is used at MXI1/MXO1, the PHYCLKSRC can be set to 0. This will provide a 24 MHz/19.2 MHz from the PLLC1AUXCLK clock domain to the USB PHY.
- When a 36 MHz crystal is used at MXI1/MXO1, the PHYCLKSRC can be set to 1. This will provide a 12 MHz clock from SYSCLKBP (36 MHz divided internally by three) to the USB PHY. The USB PHY is capable of accepting only 24 MHz /12 MHz/19.2 MHz.
- If a 27 MHz crystal is used and if the 24 MHz can be generated from the PLLC1 controller, the PHYCLKSRC can be set to 2. In this case, the PLLC1SYSCLK1 clock domain will be provided to the USB PHY.
- If the 24 MHz is generated from PLLC2, the PHYCLKSRC can be set to 3. In this case, the PLLC2SYSCLK1 clock domain will be provided to the USB PHY.
- The USB module will not work in PLL bypass mode.
- The PHYCLKFREQ can be programmed to the value corresponding to the input clock frequency of the USB PHY.

5.2.3 Key Scan

The keyscan module can operate from two clock sources. Clock source selection for the keyscan module is based on the progammation of the KEYSCLKS field of the PERI_CLKCTL register. The clocking options are as follows:

- A divided clock from PLLC1 AUXCLK: DIV3 in the PERI_CLKCTL register configures the divider ratio for this clock
  - KeyScan clock frequency = PLLC1AUXCLK / (DIV3+1)
- 32.768 kHz clock from RTCXI

5.2.4 Voice Codec

The voice codec module supports the sampling frequency (Fs) from 16 KHz to 48 KHz. The voice codec clock is configured by setting PLLC1SYSCLK4 and PERI_CLKCTL.DIV2 divider values in order for the voice codec clock to be same or close to 256xFs.

- Voice codec clock frequency = PLLC2SYSCLK4/(DIV2+1)

5.2.5 HDVICP

The HDVICP block uses two different clock domains. For its processing logic it uses either the PLLC1SYSCLK2 or PLLC2SYSCLK2 clock sources (selected by the appropriate HDVICPCLKS value in the PERI_CLKCTL register), and for its bus interface it uses the PLLC1SYCLK3 clock domain.

5.2.6 PRTC Subsystem

The PRTC subsystem (PRTCSS) can operate from two clock sources. Clock source selection for the PRTCSS module is based on the programmation of the PRTCCLKS field of the PERI_CLKCTL register. The clocking options are as follows:

- A divided Clock from PLLC1 AUXCLK: DIV3 in PERI_CLKCTL register configures the divider ratio for this clock.
  - PRTCSS clock frequency = PLLC1AUXCLK / (DIV3+1)
- 32.768 kHz clock from RTCXI

5.2.7 MPEG/JPEG Coprocessor (MJCP)

The MJCP block uses two different clock domains. For its processing logic it uses PLLC1SYSCLK4, and for its bus interface it uses the PLLC1SYCLK3 clock domain.

5.2.8 ARM926EJ-S

The ARM subsystem uses either the PLLC1SYSCLK2 or the PLLC2SYSCLK2 clock domain as its source clock, which can be selected by programming the ARMCLKS bit field in the PERI_CLKCTL register.

5.2.9 DDR2 EMIF Clocking

The DDR2/mDDR EMIF uses either PLLC1SYSCLK7 or PLLC2SYSCLK3 clock domains depending on what the selection is by programming the DDRCLKS bit field in the PERI_CLKCTL register. The reference clock for the PLLC1 or PLLC2 can either be a 24 MHz, 36 MHz, 19.2 MHz, or a 27 MHz crystal input. The PLLC1SYSCLK7 or the PLLC2SYSCLK3 clock domain selected as the source for the DDR2 clock needs to supply the DDR2 PHY module with a clock rate frequency of 2X the desired DDR2 clock rate frequency. For example, if the DDR2 interface is required to work at 216 MHz from the PLLC1 controller, then the PLLC1 must provide a 432 MHz clock to the DDR2 PHY. The PHY will divide the 432 MHz by 2 and generate the 216 MHz interface clock which will be used by the DDR2/mDDR controller module.

The PLLC1 and PLLC2 controllers have programmable dividers that can be used to divide-down their PLL output clocks to provide the required DDR2 PHY clock. If DDR2/mDDR EMIF uses the PLLC1SYSCLK7 clock domain, this divider value is configured in the PLLC1.PLLDIV7 register. Alternatively, if DDR2/mDDR EMIF uses the PLLC2SYSCLK3 clock domain, then this divider value is configured in the PLLC2.PLLDIV3 register. These dividers ensure that the frequency output from the PLLC1 or PLLC2 controllers lies within the optimal frequency range for DDR2 PHY operation.

5.2.10 Auxiliary Clock (AUXCLK)

The AUXCLK clock domain operates at the same clock rate as the oscillator input frequency, bypassing both PLL controllers. The reference clock for the AUXCLK domain frequency can be 24 MHz, 36 MHz, 19.2 MHz, or 27 MHz, depending on the crystal input used. The following peripherals' clock source is driven from the AUXCLK clock domain: ADC, RTO, Timer0-4, WDT, UART0, I2C, SPI4, and PWM0-3.

6 PLL Controllers (PLLCs)

6.1 PLL Controller Module

Two PLL controllers provide clocks to different components of the chip. The PLL controller 1 (PLLC1) provides clocks to most of the components of the chip. The PLL controller 2 (PLLC2) provides clocks to the DDR PHY and is also capable of providing clocks to the ARM, USB, VPSS, or the voice codec modules.

The PLL module provides the following:

- Glitch-free transitions (on changing PLL settings)
- Domain clocks alignment
- Clock gating
- PLL bypass
- PLL power down

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

- Domain clocks: SYSCLKn
- Bypass domain clock: SYSCLKBP
- Auxiliary clock from reference clock: AUXCLK

Various dividers that can be used are as follows:

- Pre-PLL divider: PREDIV
- Post-PLL divider: POSTDIV
- SYSCLK divider: PLLDIV1, …, PLLDIVn
- SYSCLKBP divider: BPDIV
- OBSCLK divider: OSCDIV1

The multiplier values supported are handled by:

- PLL multiplier control: PLLM

NOTE: PLLCxSYSCLKy is used to denote post divide clock output SYSCLKy from PLL controller x

x= denotes PLL Controller values 1 and 2

y = denotes post divide clock outputs values from 1 to 9 for PLLC1 and from 1 to 5 for PLLC2

6.2 PLLC1 Controller

The PLLC1 controller provides most of the DM36x clocks and also generates the frequencies needed for the ARM, Video Processing Subsystem (VPSS), MJCP block, EDMA, HDVICP, USB, MMC/SD and the rest of the DM36x peripherals. Software controls the PLLC1 operation through the PLLC1 registers. The following list in Table 8 and Figure 6 describe the customizations of the PLLC1 controller.

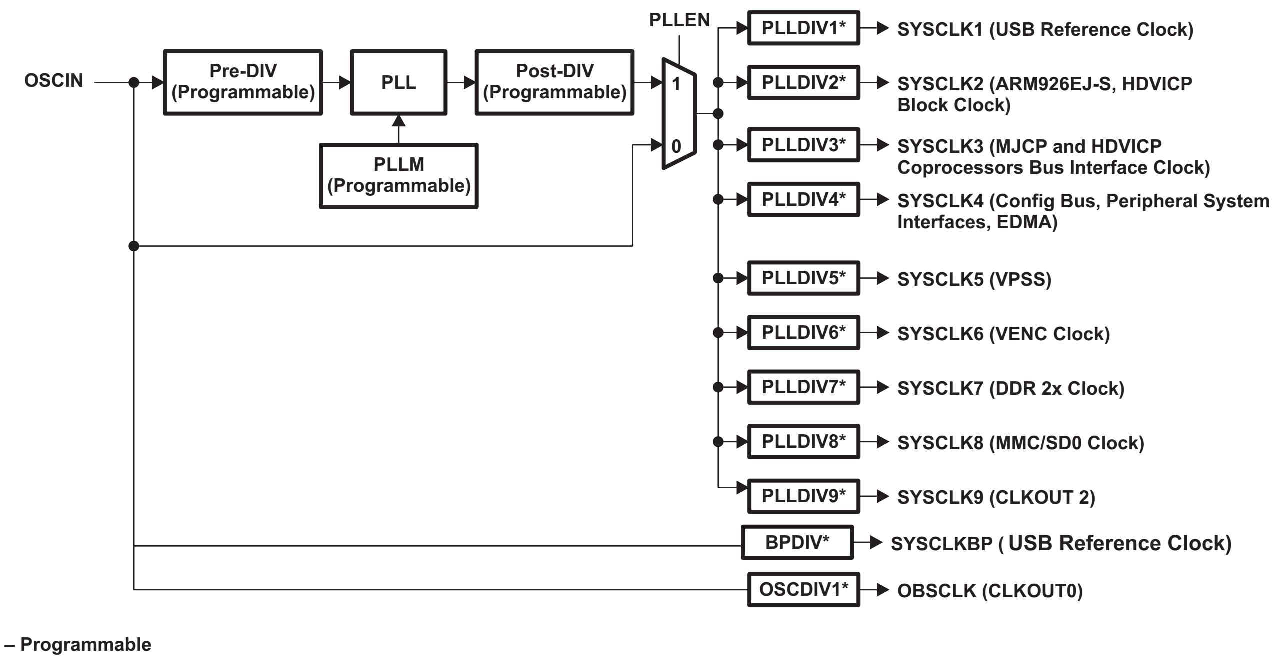
- Provides primary system clock
- Software configurable
- PLL pre-divider value is programmable
- PLL multiplier value is programmable
- PLL post-divider value is programmable
- SYSCLK [9:1] outputs are provided

Table 8. PLLC1 Output Clocks

PLLC1SYSCLKy	Used By	PLLDIV Divider
PLLC1SYSCLK1	USB reference clock[1]	Programmable
PLLC1SYSCLK2	ARM926EJ-S, HDVICP block clock [1]	Programmable
PLLC1SYSCLK3	MJCP and HDVICP bus interface clock	Programmable
PLLC1SYSCLK4	Configuration bus clock, peripheral system interfaces, EDMA	Programmable
PLLC1SYSCLK5	VPSS clock	Programmable
PLLC1SYSCLK6	VENC clock[1]	Programmable
PLLC1SYSCLK7	DDR 2x clock[1]	Programmable
PLLC1SYSCLK8	MMC/SD0 clock	Programmable
PLLC1SYSCLK9	CLKOUT2	Programmable
PLLC1OBSCLK	CLKOUT0	Programmable
PLLC1SYSCLKBP	USB reference clock[1]	Programmable

[1] These clock outputs are multiplexed with other clocks. For clock source selection, refer to the peripheral-specific clock selection in Section 5.

Figure 6. PLLC1 Configuration



6.3 PLLC2 Controller

The PLLC2 controller provides the USB reference clock, ARM926EJ-S, DDR 2x clock, voice codec clock, and VENC 27 MHz, 74.25 MHz clock. The functionality of the PLLC2 controller can be programmed via the PLLC2 registers. The following list in Table 9 and Figure 7 describe the customizations of the PLLC2 controller.

The PLLC2 customization includes the following features:

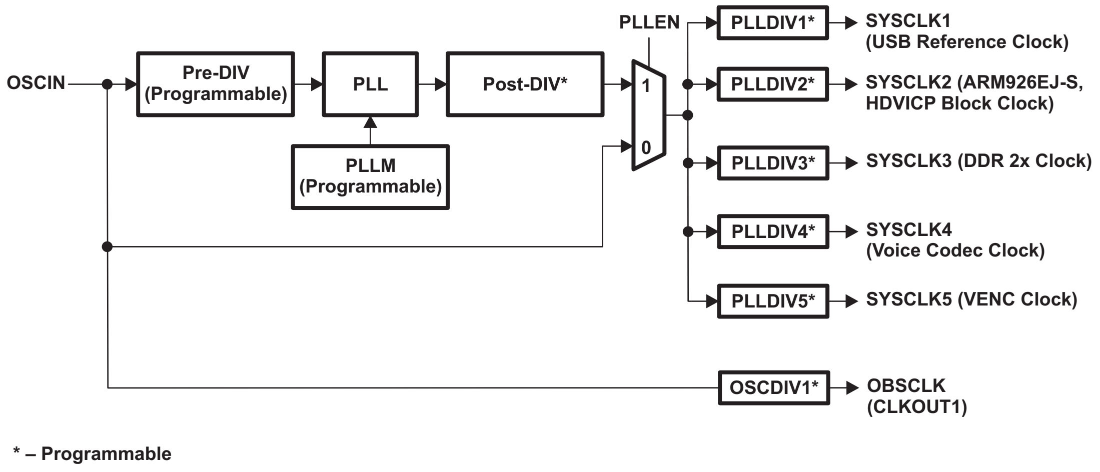
- Is software configurable
- PLL pre-divider value is programmable
- PLL multiplier value is programmable
- PLL post-divider value is programmable
- SYSCLK [5:1] clock outputs are used

Table 9. PLLC2 Output Clocks

PLLC2SYSCLKy	Used by	PLLDIV Divider
PLLC2SYSCLK1	USB reference clock[1]	Programmable
PLLC2SYSCLK2	ARM926EJ-S, HDVICP block clock [1]	Programmable
PLLC2SYSCLK3	DDR 2x clock [1]	Programmable
PLLC2SYSCLK4	Voice Codec clock	Programmable
PLLC2SYSCLK5	VENC clock [1]	Programmable
PLLC2OBSCLK	CLKOUT1	Programmable

[1] These clock outputs are multiplexed with other clocks. For clock source selection refer to the peripheral specific clock selection in Section 5.

Figure 7. PLLC2 Configuration



6.4 PLLC Functional Description

This section describes the multiplier and dividers in the PLL controller as well as the bypass and PLL modes of operation.

6.4.1 Multipliers and Dividers

The PLL controller is programmed through the PLL multiplier control register (PLLM), PLL pre-divider control register (PREDIV), PLL post-divider control register (POSTDIV), and the PLL system clock divider control registers (PLLDIVn). The dividers are programmable and may be enabled or disabled. When a divider is disabled, no clock is output from that clock divider, so a divider only outputs a clock when it is enabled in its corresponding divider control register.

6.4.2 Bypass Mode

The PLLM multiplier, pre-divider (PREDIV), and post-divider (POSTDIV) registers of the PLL controller may be bypassed altogether. The PLL enable bit (PLLEN) in the PLL control/status register (PLLCTL) determines the PLL controller mode.

- When PLLEN = 1, the PLL mode is enabled and PLLM, PREDIV, and POSTDIV registers of the PLL are used.
- When PLLEN = 0, the bypass mode is enabled and the PLLM, PREDIV, and POSTDIV registers of the PLL are bypassed. When bypass mode is enabled, the input reference clock is directly fed to the system clock dividers (PLLDIVn). The PLL controller defaults to bypass mode after device reset.

6.4.3 PLL Mode

When in PLL mode (PLLEN = 1), the input reference clock is supplied to the pre-divider (PREDIV). PREDIV must be enabled (PREDEN = 1) in PLL mode. When it is enabled, the input reference clock is divided-down by the value in the PLL divider ratio bits (RATIO) in PREDIV. The output from PREDIV is input to the PLL.

The PLL multiplies the clock by 2x the value in the PLL multiplier bits (PLLM) in the PLL multiplier control register (PLLM). The output from the PLL (PLLOUT) is input to POSTDIV. POSTDIV must be enabled (POSTEN = 1) in PLL mode. When it is enabled, the output from PLLOUT is divided-down by the value in the PLL divider ratio bits (RATIO) in POSTDIV.

The output from POSTDIV is input to the system clock dividers (PLLDIVn). When enabled (bit DnEN = 1), a PLLDIVn divides-down the output clock of the PLL by the value in the PLL divider ratio bits (RATIO) in PLLDIVn. The system clock dividers generate 50% duty cycle output clocks on the SYSCLKn clock domains.

6.5 PLL Configuration

This section describes the procedures for initializing and configuring the PLL controller.

6.5.1 PLL Mode and Bypass Mode

6.5.1.1 PLL Mode (PLLEN = 1)

The sequence for PLL mode are as follows:

1. In PLLCTL, write PLLPWRDN = 0 (power up the PLL).
2. In PLLCTL, write PLLENSRC = 0 (enable PLLEN). The bit PLLEN in PLLCTL has no effect unless you write PLLENSRC = 0.
3. In PLLCTL, write PLLEN = 0 (bypass mode).
4. Wait at least four reference clock cycles for the bypass mode to take effect.
5. In PLLCTL, write PLLRST = 1 (assert PLL reset).
6. Wait at least five micro-seconds for the PLL reset.
7. In PLLCTL, write PLLRST = 0 (de-assert PLL reset).
8. Write PREDIV, POSTDIV, and PLLM registers with the required divider and multiplier values to obtain the desired frequency.
9. Program the sequence in PLLSECCTL (TENABLE, TENABLEDIV, and TINITZ) for the multipliers and pre-dividers to take effect.
10. Write PLLDIV to set the PLLDIVn dividers. You can apply the GO operation to change these dividers to new ratios.
11. Poll LOCK1, LOCK2, and LOCK3 bits in the PLLCx_CONFIG register.
12. In PLLCTL, write PLLEN = 1 to switch from bypass mode to PLL mode.

NOTE: If ARM926 needs to be run from PLLC2SYSCLK2, then set the PERI_CLKCTL.ARMCLKS bit to 1 after the PLLC1 and PLLC2 configuration is completed.

The following sequence is required in PLLSECCTL for the multiplier and pre-divider values to take effect :

1. Assert TENABLE = 1, TENABLEDIV=1, TINITZ = 1
2. Assert TENABLE = 1, TENABLEDIV=1, TINITZ = 0
3. Assert TENABLE = 0, TENABLEDIV=0, TINITZ = 0
4. Assert TENABLE = 0, TENABLEDIV=0, TINITZ = 1

6.5.1.2 Bypass Mode (PLLEN = 0)

The sequence for bypass mode are as follows:

1. In PLLCTL, write PLLEN = 0 (bypass mode).
2. Wait at least four reference clock cycles for the PLLEN mux to change.
3. In PLLCTL, write PLLRST = 1 (assert PLL reset).
4. Do not program PREDIV, PLLM, and POSTDIV because they have no effect in bypass mode.
5. Program PLLDIVn if necessary. You can apply the GO operation to change these dividers to new ratios. See Section 6.5.2.1.

6.5.2 Changing Divider / Multiplier Ratios

This section describes how to change the divider and multiplier values.

6.5.2.1 PLLDIVn and GO Operation

The GO operation is required to change the divider ratios of the PLLDIVn registers. Section Section 6.5.2.1.1 discusses the GO operation. Section 6.5.2.1.2 gives the software steps required to change the divider ratios.

6.5.2.1.1 GO Operation

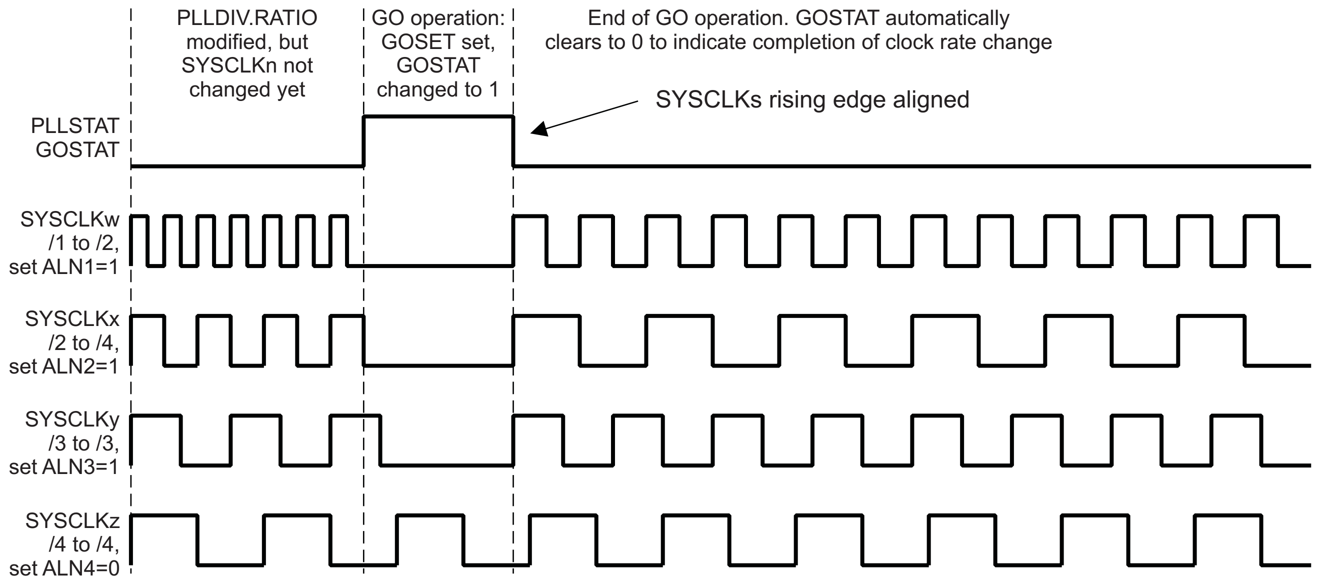
This section discusses the GO operation and the alignment of the SYSCLKs. Writes to the RATIO field in the PLLDIVn registers do not immediately change the dividers' actual divide ratios. The PLLDIVn dividers change to the new RATIO rates only during a GO operation. The PLL controller clock align control register (ALNCTL) determines which SYSCLKs must be aligned. Before a GO operation, you must program ALNCTL so that the appropriate clocks are aligned during the GO operation. In this device, you must always program ALNCTL so that all SYSCLKs are aligned. A GO operation is initiated by setting the GOSET bit in PLLCMD to 1.

During a GO operation:

- Any SYSCLKn with the corresponding ALNn bit in ALNCTL set to 1 is paused at the low edge. Then the PLL controller restarts all these SYSCLKs simultaneously, aligned at the rising edge. When the SYSCLKs are restarted, SYSCLKn toggles at the rate programmed in the RATIO field in PLLDIVn.
- Any SYSCLKn with the corresponding ALNn bit in ALNCTL cleared to 0 remains free-running during a GO operation. SYSCLKn is not modified to the new RATIO rate in PLLDIVn. SYSCLKn is not aligned to other SYSCLKs. In this device, do not program any ALNn bit in ALNCTL to 0; always program ALNCTL so that all SYSCLKs are aligned.
- The GOSTAT bit in PLLSTAT is set to 1 throughout the duration of a GO operation.

Figure 8 shows how the clocks' rising-edges are aligned during a GO operation. Notice that although the SYSCLKy ratio remains the same, it is still stopped, since ALN3 = 1 in ALNCTL.

Figure 8. Clock Ratio Change and Alignment with Go Operation


PLLDIV.RATIO modified, but SYSCLKn not changed yet
GO operation: GOSET set, GOSTAT changed to 1
End of GO operation. GOSTAT automatically clears to 0 to indicate completion of clock rate change
SYSCLKs rising edge aligned
PLLSTAT GOSTAT
SYSCLKw /1 to /2, set ALN1=1
SYSCLKx /2 to /4, set ALN2=1
SYSCLKy /3 to /3, set ALN3=1
SYSCLKz /4 to /4, set ALN4=0

6.5.2.1.2 Software Steps to Modify PLLDIVn Ratios

To modify the PLLDIVn ratios, perform the following steps:

1. Check that the GOSTAT bit in PLLSTAT is cleared to 0 to show that no GO operation is currently in progress.
2. Program the RATIO field in PLLDIVn to the desired new divide-down rate. If the RATIO field changes, the PLL controller will flag the change in the corresponding bit of DCHANGE.
3. Set the respective ALNn bits in ALNCTL to 1 to align any SYSCLKs after the GO operation.
4. Set the GOSET bit in PLLCMD to 1 to initiate the GO operation to change the divide values, and to align the SYSCLKs as programmed.
5. Read the GOSTAT bit in PLLSTAT to make sure the bit goes back to 0 to indicate that the GO operation has completed.

6.5.2.2 Pre-Divider (PREDIV), PLL Multiplier (PLLM), and Post-Divider (POSTDIV)

To change the values of PREDIV, PLLM, or POSTDIV, the PLL controller must first be placed in bypass mode. Perform the following steps to modify PREDIV, PLLM, or POSTDIV ratios:

1. In PLLCTL, write PLLEN = 0 to place the PLL in bypass mode.
2. Wait at least four reference clock cycles for the bypass mode to take effect.
3. In PLLCTL, write PLLRST = 1 (assert PLL).
4. Modify PREDIV, PLLM, and/or POSTDIV ratios.
5. Program the sequence in PLLSECCTL(TENABLE, TENABLEDIV, TINITZ) for the multipliers and pre-dividers to take effect.
6. Write PLLDIV to set PLLDIVn dividers. Apply the GO operation to change these dividers to new ratios.
7. Poll LOCK1, LOCK2, and LOCK3 bits in the PLLCx_CONFIG register.
8. In PLLCTL, write PLLEN = 1 to switch from bypass mode to PLL mode.

The following sequence is required in PLLSECCTL for multiplier and pre-divider values to take effect:

1. Assert TENABLE = 1, TENABLEDIV=1, TINITZ = 1
2. Assert TENABLE = 1, TENABLEDIV=1, TINITZ = 0
3. Assert TENABLE = 0, TENABLEDIV=0, TINITZ = 0
4. Assert TENABLE = 0, TENABLEDIV=0, TINITZ = 1

6.5.3 PLL Power Down and Wakeup

The PLL may be powered-down, in which case the PLL controller is in bypass mode and the device runs from the input reference clock. The device is able to run when the PLL is powered-down because it is still being clocked by the bypass clock.

Perform the following procedure to power-down the PLL:

1. In PLLCTL, write PLLEN = 0 (bypass mode).
2. Wait at least four reference clock cycles for the PLLEN mux to change.
3. In PLLCTL, write PLLPWRDN = 1 to power-down the PLL.

To wake up the PLL from its power-down mode, follow the PLL sequence described in Section 6.5.1.1.

6.6 PLL Controller Register Map

6.6.1 Introduction

Table 10 lists the base addresses for the PLLC1 and PLLC2 registers. Table 11 lists the memory-mapped registers for PLLC1 and PLLC2.

Table 10. PLL Controller Module Instance Table

Instance ID	Base Address	End Address	Size
PLLC1	0x01C4 0800	0x01C4 0BFF	0x400
PLLC2	0x01C4 0C00	0x01C4 0FFF	0x400

Table 11. PLLC Registers

Offset	Acronym	Register Description	Section
00h	PID	Peripheral ID and revision information	Section 6.6.2
E4h	RSTYPE[1]	Reset Type Status Register	Section 6.6.3
100h	PLLCTL	Controls PLL operations	Section 6.6.4
104h	OCSEL	OBSCLK Select Register	Section 6.6.5
108h	PLLSECCTL	PLL Secondary Control Register	Section 6.6.6
110h	PLLM	PLL Multiplier Control	Section 6.6.7
114h	PREDIV	Pre-divider control	Section 6.6.8
118h	PLLDIV1	PLL Controller Divider 1 Register (SYSCLK1)	Section 6.6.9
11Ch	PLLDIV2	PLL Controller Divider 2 Register (SYSCLK2)	Section 6.6.10
120h	PLLDIV3	PLL Controller Divider 3 Register (SYSCLK3)	Section 6.6.11
124h	OSCDIV1	Oscillator Divider 1 Register	Section 6.6.12
128h	POSTDIV	Post-divider control	Section 6.6.13
12Ch	BPDIV	Bypass divider control	Section 6.6.14
138h	PLLCMD	PLL Controller Command register	Section 6.6.15
13Ch	PLLSTAT	PLL Controller Status register	Section 6.6.16
140h	ALNCTL	Align control register	Section 6.6.17
144h	DCHANGE	PLL divider ratio change status register	Section 6.6.18
148h	CKEN[1]	Clock enable control AUXCLK	Section 6.6.19
14Ch	CKSTAT	Clock status for SYSCLKBP and AUXCLK	Section 6.6.20
150h	SYSTAT	Clock status for SYSCLKn clocks	Section 6.6.21
160h	PLLDIV4	PLL Controller Divider 4 Register (SYSCLK4)	Section 6.6.22
164h	PLLDIV5	PLL Controller Divider 5 Register (SYSCLK5)	Section 6.6.23
168h	PLLDIV6[1]	PLL Controller Divider 6 Register (SYSCLK6)	Section 6.6.24
16Ch	PLLDIV7[1]	PLL Controller Divider 7 Register (SYSCLK7)	Section 6.6.25
170h	PLLDIV8[1]	PLL Controller Divider 8 Register (SYSCLK8)	Section 6.6.26
174h	PLLDIV9[1]	PLL Controller Divider 9 Register (SYSCLK9)	Section 6.6.27

(1) Available only for PLLC1 and not for PLLC2. Unless explicitly stated, the bit fields described are applicable to both PLLC1 and PLLC2 throughout the section.

6.6.2 Peripheral ID Register (PID) Register

The peripheral ID register (PID) is shown in Figure 9 and described in Table 12.

Figure 9. Peripheral ID (PID) Register

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

15 8	7 0
CLASS	REV
R-0x8	R-0xE

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

Table 12. Peripheral ID (PID) Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Any writes to these bit(s) must always have a value of 0.
23-16	TYPE	1	Peripheral Type: 0x01 identifies as PLLC
15-8	CLASS	0x08	Peripheral Class: 0x08
7-0	REV	0x0E	Peripheral Revision

6.6.3 Reset Type Status (RSTYPE) Register

The reset type status (RSTYPE) register is shown in Figure 10 and described in Table 13 for PLLC1. It latches because 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 10. Reset Type Status (RSTYPE) Register

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

Table 13. Reset Type Status (RSTYPE) Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.
3	SRST		System Reset
		0	System reset (srst) was not the last reset to occur
		1	System reset (srst) was the last reset to occur
2	MRST		Maximum Reset
		0	Maximum reset (mrst) was not the last reset to occur
		1	Maximum reset (mrst) was the last reset to occur
1	XWRST		External Warm Reset
		0	External warm reset (xwrst) was not the last reset to occur
		1	External warm reset (xwrst) was the last reset to occur
0	POR		Power On Reset
		0	POR (xpor) was not the last reset to occur
		1	POR was the last reset to occur

6.6.4 PLL Control (PLLCTL) Register

The PLL control register is shown in Figure 11 and described in Table 14 for PLLC1 and PLLC2.

Figure 11. PLL Control (PLLCTL) Register

31–16
Reserved
R-0

15–9	8
Reserved	Reserved
R-0	R-*0/**1

7	6	5	4	3	2	1	0
Reserved	Reserved	PLLENSRC	Reserved	PLLRST	Reserved	PLLPWRDN	PLLEN
R-0	R-1	R/W-*1/**0	R-1	R/W-1	R-0	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n*=Reset value of field for both PLLC1 and PLLC2 registers.
* = Reset value of the field of register belonging to PLLC1
** = Reset value of the field of register belonging to PLLC2

Table 14. PLL Control (PLLCTL) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Any writes to these bit(s) must always have a value of 0.
5	PLLENSRC		PLL enable source. This bit must be cleared to 0 before PLLEN will have any effect.
		0	PLL enable is controlled by the register bit PLLEN
		1	PLL enable is controlled by internal test hardware
4	Reserved		Any writes to these bit(s) must always have a value of 0.
3	PLLRST		PLL reset
		0	PLL reset de-assert
		1	PLL reset assert
2	Reserved		Any writes to these bit(s) must always have a value of 0.
1	PLLPWRDN		PLL power-down
		0	PLL operating, not powered down
		1	PLL power-down
0	PLLEN		PLL Mode Enable. Bit PLLENSRC must be cleared to 0 before PLLEN will have any effect.
		0	Bypass mode
		1	PLL mode, not bypassed

6.6.5 OBSCLK Select (OCSEL) Register

The OBSCLK select (OCSEL) register is shown in Figure 12 and described in Table 15 for PLLC1 and PLLC2.

Figure 12. OBSCLK Select (OCSEL) Register

31–16
Reserved
R-0

15–5	4	3–0
Reserved	OCSRC	Reserved
R-0	R-0	R-0

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

Table 15. OBSCLK Select (OCSEL) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4	OCSRC		OBSCLK source.
		0	Oscillator divider output enabled
		1	Oscillator divider output disabled
3-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

6.6.6 PLL Secondary Control (PLLSECCTL) Register

The PLL secondary control (PLLSECCTL) register is shown in Figure 13 and described in Table 16.

Figure 13. PLL Secondary Control (PLLSECCTL) Register

31		24
	Reserved	
	R-0	

23	22	21	20	19	18	17	16
LIMIT RECALEN	STOP MODE	LOW CURRSTDBY	SLOW CLKLOCK	DRIFT GUARDEN	TENABLEDIV	TENABLE	TINITZ
R/W-0	R/W-1	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1

15		0
	Reserved	
	R-0	

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

Table 16. PLL Secondary Control (PLLSECCTL) Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Any writes to these bit(s) must always have a value of 0.
23	LIMITRECALEN		Force recalibration on code limits (active high)
		0	Disabled
		1	Enabled
22	STOPMODE		Stop /Limp select (active high/low)
		0	Disabled
		1	Enabled
21	LOWCURRSTDBY		Low current stand by select(active high)
		0	Disabled
		1	Enabled
20	SLOWCLKLOCK		Low input frequency control (active high)
		0	Disabled
		1	Enabled
19	DRIFTGUARDEN		Temperature, Drift recalibration enable (active high)
		0	Disabled
		1	Enabled
18	TENABLEDIV		Core Register M2/N2 load enable (low-high)
		0	Disabled
		1	Enabled
17	TENABLE		Core Register M/N load enable (low - high)
		0	Disabled
		1	Enabled
16	TINITZ		Core soft reset lock sequence initialization (high-low-high)
		0	Disabled
		1	Enabled
15-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.
For normal usage, bits 23 to 19 can be "01000" and use TENABLE, TENABLEDIV, and TINITZ for multiplier and pre-divider loading.			

6.6.7 PLL Multiplier Control (PLLM) Register

The PLL multiplier control (PLLM) register is shown in Figure 14 and described in Table 17 for PLLC1 and PLLC2. The default multiplier value is 11. The multiplier value may range from 1 to 1023.

Figure 14. PLL Multiplier Control (PLLM) Register

31 – 10	9 – 0
Reserved	PLLM
R-0	R/W-0x0b

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

Table 17. PLL Multiplier Control (PLLM) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Any writes to these bit(s) must always have a value of 0.
9-0	PLLM	0x0b	PLL Multiplier. Multiplier value = 2xPLLM

6.6.8 PLL Pre-Divider Control (PREDIV) Register

The PLL pre-divider control (PREDIV) register is shown in Figure 15 and described in Table 18 for PLLC1 and PLLC2.

Figure 15. PLL Pre-Divider (PREDIV) Control Register

31 5	4 0
Reserved	RATIO
R-0	R/W-0

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

Table 18. PLL Pre-Divider Control (PREDIV) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for pre divider. Ratio value = RATIO + 1

6.6.9 PLL Controller Divider 1 (PLLDIV1) Register

The PLL controller divider 1 (PLLDIV1) register is shown in Figure 16 and described in Table 19 for PLLC1 and PLLC2. PLLDIV1 controls the divider for PLLC1SYSCLK1 and PLLC2SYSCLK1.

Figure 16. PLL Controller Divider 1 (PLLDIV1) Register

31–16
Reserved
R-0

15	14–5	4–0
D1EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 19. PLL Controller Divider 1 (PLLDIV1) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D1EN		Divider enable for SYSCLK1. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved		Any writes to these bit(s) must always have a value of 0.
4-0	RATIO		Divider ratio for SYSCLK1. Ratio value = RATIO + 1

6.6.10 PLL Controller Divider 2 (PLLDIV2) Register

The PLL controller divider 2 (PLLDIV2) register is shown in Figure 17 and described in Table 20 for PLLC1 and PLLC2. PLLDIV2 controls the divider for PLLC1SYSCLK2 and PLLC2SYSCLK2.

Figure 17. PLL Controller Divider 2 (PLLDIV2) Register

31–16
Reserved
R-0

15	14–5	4–0
D2EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 20. PLL Controller Divider 2 (PLLDIV2) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D2EN		Divider enable for SYSCLK2. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK2. Ratio value = RATIO + 1

6.6.11 PLL Controller Divider 3 (PLLDIV3) Register

The PLL controller divider 3 (PLLDIV3) register is shown in Figure 18 and described in Table 21 for PLLC1 and PLLC2. PLLDIV3 controls the divider for PLLC1SYSCLK3 and PLLC2SYSCLK3.

Figure 18. PLL Controller Divider 3 (PLLDIV3) Register

31–16
Reserved
R-0

15	14–5	4–0
D3EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 21. PLL Controller Divider 3 (PLLDIV3) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D3EN		Divider enable for SYSCLK3. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK3. Ratio value = RATIO + 1

6.6.12 Oscillator Divider 1 (OSCDIV1) Register for OBSCLK

The oscillator divider 1 (OSCDIV1) register for OBSCLK is shown in Figure 19 and described in Table 22 for PLL1 and PLL2. It does not go through the PLL path. The OBSCLK will be oscillator input clock/(RATIO+1).

Figure 19. Oscillator Divider 1 (OSCDIV1) for OBSCLK Register

31–16
Reserved
R-0

15	14–5	4–0
OD1EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 22. Oscillator Divider 1 (OSCDIV1) for OBSCLK Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	OD1EN		Oscillator Divider OD1 Enable
		0	Oscillator Divider 1 Disabled
		1	Oscillator Divider 1 Enabled
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for OBSCLK divider. Ratio value = RATIO + 1

6.6.13 PLL Post-Divider Control (POSTDIV) Register

The PLL post-divider control (POSTDIV) register is shown in Figure 20 and described in Table 23 for PLLC1 and PLLC2.

Figure 20. PLL Post-Divider Control (POSTDIV) Register

31 – 16
Reserved
R-0

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

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

Table 23. PLL Post-Divider Control (POSTDIV) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	POSTDEN		Post-divider enable. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for post divider. Ratio value = RATIO + 1

6.6.14 Bypass Divider (BPDIV) Register

The bypass divider (BPDIV) register is shown in Figure 21 and described in Table 24 for PLLC1 and PLLC2. BPDIV controls the divider for SYSCLKBP. The divider must always be enabled (BPDEN=1).

Figure 21. Bypass Divider (BPDIV) Register

31	16
Reserved	
R-0	

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

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

Table 24. Bypass Divider (BPDIV) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Any writes to these bit(s) must always have a value of 0.
15	BPDEN		Divider enable for bypass clock. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO		Divider ratio for bypass clock

6.6.15 PLL Controller Command (PLLCMD) Register

The PLL controller command (PLLCMD) register is shown in Figure 22 and described in Table 25 for PLLC1 and PLLC2. PLLCMD is used to initiate a GO operation for SYSCLKn ratio change and/or phase alignment.

Figure 22. PLL Controller Command (PLLCMD) Register

31 … 1	0
Reserved	GOSET
R-0	R/W-0

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

Table 25. PLL Controller Command (PLLCMD) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Any writes to these bit(s) must always have a value of 0.
0	GOSET		GO operation command for SYSCLKn ratio change and/or phase alignment. Before setting this bit to 1 to initiate a GO operation, check the GOSTAT bit in the PLLSTAT register to ensure all previous GO operations have completed.
		0	Clear bit. Write of 0 clears bit to 0.
		1	Initiates GO operation. Write of 1 initiates GO operation. Once set, GOSET remains set but further writes of 1 can initiate the GO operation.

6.6.16 PLL Controller Status (PLLSTAT) Register

The PLL controller status register (PLLSTAT) is shown in Figure 23 and described in Table 26 for PLLC1 and PLLC2.

Figure 23. PLL Controller Status (PLLSTAT) Register

31–16
Reserved
R-0

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

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

Table 26. PLL Controller Status (PLLSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2	STABLE		OSCIN Stable
		0	Resetclk counter not finished counting
		1	OSCIN/CLKIN is assumed to be stable
1	LOCK		PLL Core STATUS
		0	PLL core not locked
		1	PLL core locked
0	GOSTAT		GO status
		0	GO operation is not in progress. SYSCLK divider ratios and/or phase alignment are not being changed.
		1	GO operation is in progress. SYSCLK divider ratios and/or phase alignment are changing.

6.6.17 PLLC Clock Align Control (ALNCTL) Register

The PLLC clock align control (ALNCTL) register is shown in Figure 24 and described in Table 27 for PLLC1 and shown in Figure 25 and described in Table 28 for PLLC2. ALNCTL controls SYSCLK alignment when the GOSET bit in PLLCMD is set to 1. In this device, all SYSCLKn must be aligned.

Figure 24. PLLC1 Clock Align Control (ALNCTL) Register

31 … 9	8 … 0
Reserved	ALN[8:0]
R-0	R/W-0x1C

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

Figure 25. PLLC2 Clock Align Control (ALNCTL) Register

31 … 5	4 … 0
Reserved	ALN[4:0]
R-0	R/W-0

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

Table 27. PLLC1 Clock Align Control (ALNCTL) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Any writes to these bit(s) must always have a value of 0.
8-0	ALN[8:0]		SYSCLKy needs to be aligned with other clocks selected in this register.
		0	Do not need to align SYSCLKy to other clocks
		1	Align SYSCLKy to other clocks selected in this register

Table 28. PLLC2 Clock Align Control (ALNCTL) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Any writes to these bit(s) must always have a value of 0.
4-0	ALN[0:4]		SYSCLKy needs to be aligned with other clocks selected in this register
		0	Do not need to align SYSCLKy to other clocks
		1	Align SYSCLKy to other clocks selected in this register

6.6.18 PLLDIV Ratio Change Status (DCHANGE) Register

The PLLDIV ratio change status (DCHANGE) register is shown in Figure 26 and described in Table 29 for PLLC1 and PLLC2. Whenever a different ratio is written to the PLLDIVn registers, the SYS flags in DCHANGE change during the GO operation.

Figure 26. PLLDIV Ratio Change Status (DCHANGE) Register

31 9	8 0
Reserved	SYS[9:1]
R-0	R-0

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

Table 29. PLLDIV Ratio Change Status (DCHANGE) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Any writes to these bit(s) must always have a value of 0.
8:0	SYS[9:1]		SYSCLKy divide ratio has been modified/not modified. SYS[9:6] reserved for PLLC2.
		0	SYSCLKy ratio has not been modified
		1	SYSCLKy ratio has been modified

6.6.19 Clock Enable Control (CKEN) Register

The clock enable control (CKEN) register is shown in Figure 27 and described in Table 30 for PLLC1. The CKEN register is used to enable the PLL auxiliary clock (AUXCLK). The auxiliary clock should always be enabled and CKEN must always be set to 1. PLLC2 does not use the auxiliary clock, so the CKEN register is not applicable to PLLC2.

Figure 27. Clock Enable Control (CKEN) Register

31–16
Reserved
R-0

15–2	1	0
Reserved	OBSCLK	AUXEN
R-0	R/W-0	R/W-1

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

Table 30. Clock Enable Control (CKEN) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Any writes to these bit(s) must always have a value of 0.
1	OBSCLK		OBSCLK Enable
		0	Clock disabled
		1	Clock enabled
0	AUXEN		Auxiliary clock (AUXCLK) enable
		0	Disable
		1	Enable

6.6.20 Clock Status (CKSTAT) Register

The clock status (CKSTAT) register is shown in Figure 28 and described in Table 31 for PLLC1 and PLLC2. CKSTAT shows the on/off status of the bypass clock (SYSCLKBP) for PLLC1 and PLLC2 and the auxiliary clock (AUXCLK) for PLLC1.

Figure 28. Clock Status (CKSTAT) Register

31–16
Reserved
R-0

15–4	3	2	1	0
Reserved	BPON	Reserved	OBSON	AUXEN
R-0	R-1	R-0	R-0	R-1*/0**

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset for fields of both PLLC1 and PLLC2 registers.
* = Reset value of the field of register belonging to PLLC1
** = Reset value of the field of register belonging to PLLC2

Table 31. Clock Status (CKSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.
3	BPON		SYSCLKBP status. Shows the clock on/off status for SYSCLKBP
		0	Bypass clock is off
		1	Bypass clock is on
2	Reserved	0	Any writes to these bit(s) must always have a value of 0.
1	OBSON		OBSCLK on
		0	Clock is gated
		1	Clock is on (not gated)
0	AUXEN		AUXCLK status. Shows the clock on/off status for AUXCLK(PLLC1 only)
		0	AUX clock is off
		1	AUX clock is on

6.6.21 SYSCLK Status (SYSTAT) Register

The SYSCLK status (SYSTAT) register is shown in Figure 29 and described in Table 32 for PLLC1 and PLLC2. SYSTAT shows the on/off status of the SYSCLKn clocks.

Figure 29. SYSCLK Status (SYSTAT) Register

31–16
Reserved
R-0

15–9	8
Reserved	SYS9ON
R-0	R-0x1FF*/1F**

7	6	5	4	3	2	1	0
SYS8ON	SYS7ON	SYS6ON	SYS5ON	SYS4ON	SYS3ON	SYS2ON	SYS1ON

R-0x1FF*/1F**

LEGEND: R/W = Read/Write; R = Read only; -*n* =Reset value of fields for both PLLC1 and PLLC2 registers.
* = Reset value of the field of register belonging to PLLC1
** = Reset value of the field of register belonging to PLLC2

Table 32. SYSCLK Status (SYSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Any writes to these bit(s) must always have a value of 0.
8	SYS9ON		SYSCLK9 on (This bit is reserved for PLLC2)
		0	SYSCLK9 is off
		1	SYSCLK9 is on
7	SYS8ON		SYSCLK8 on (This bit is reserved for PLLC2)
		0	SYSCLK8 is off
		1	SYSCLK8 is on
6	SYS7ON		SYSCLK7 on (This bit is reserved for PLLC2)
		0	SYSCLK7 is off
		1	SYSCLK7 is on
5	SYS6ON		SYSCLK6 on (This bit is reserved for PLLC2)
		0	SYSCLK6 is off
		1	SYSCLK6 is on
4	SYS5ON		SYSCLK5 on
		0	SYSCLK5 is off
		1	SYSCLK5 is on
3	SYS4ON		SYSCLK4 on
		0	SYSCLK4 is off
		1	SYSCLK4 is on
2	SYS3ON		SYSCLK3 on
		0	SYSCLK3 is off
		1	SYSCLK3 is on
1	SYS2ON		SYSCLK2 on
		0	SYSCLK2 is off
		1	SYSCLK2 is on
0	SYS1ON		SYSCLK1 on
		0	SYSCLK1 is off
		1	SYSCLK1 is on

6.6.22 PLL Controller Divider 4 (PLLDIV4) Register

The PLL controller divider 4 (PLLDIV4) register is shown in Figure 30 and described in Table 33 for PLLC1 and PLLC2. PLLDIV4 controls the divider for PLLC1SYSCLK4 and PLLC2SYSCLK4. The divider must always be enabled (bit D4EN=1).

Figure 30. PLL Controller Divider 4 (PLLDIV4) Register

31-16
Reserved
R-0

15	14-5	4-0
D4EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 33. PLL Controller Divider 4 (PLLDIV4) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Any writes to these bit(s) must always have a value of 0.
15	D4EN		Divider enable for SYSCLK4
		0	Disable
		1	Enable
14-5	Reserved		Any writes to these bit(s) must always have a value of 0.
4-0	RATIO		Divider ratio for SYSCLK4. Ratio value = RATIO + 1

6.6.23 PLL Controller Divider 5 (PLLDIV5) Register

The PLL controller divider 5 (PLLDIV5) register is shown in Figure 30 and described in Table 33 for PLLC1 and PLLC2. PLLDIV5 controls the divider for PLLC1SYSCLK5 and PLLC2SYSCLK5. The divider must always be enabled (bit D5EN=1).

Figure 31. PLL Controller Divider 5 (PLLDIV5) Register

31	16
Reserved	
R-0	

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

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

Table 34. PLL Controller Divider 5 (PLLDIV5) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D5EN		Divider enable for SYSCLK5. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK5. Ratio value = RATIO + 1

6.6.24 PLL Controller Divider 6 (PLLDIV6) Register

The PLL controller divider 6 (PLLDIV6) register is shown in Figure 32 and described in Table 35 for PLLC1. PLLDIV6 controls the divider for PLLC1SYSCLK6. For PLLC1, the divider must always be enabled (bit D6EN=1). The PLLDIV6 register is not applicable to PLLC2.

Figure 32. PLL Controller Divider 6 (PLLDIV6) Register

31 – 16
Reserved
R-0

15	14 – 5	4 – 0
D6EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 35. PLL Controller Divider 6 (PLLDIV6) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D6EN		Divider enable for SYSCLK6. For PLLC1, this bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK6. Ratio value = RATIO + 1

6.6.25 PLL Controller Divider 7 (PLLDIV7) Register

The PLL controller divider 7 (PLLDIV7) register is shown in Figure 33 and described in Table 36 for PLLC1. PLLDIV7 controls the divider for PLLC1SYSCLK7. For PLLC1, the divider must always be enabled (bit D7EN=1). The PLLDIV7 register is not applicable to PLLC2.

Figure 33. PLL Controller Divider 7 (PLLDIV7) Register

31–16
Reserved
R-0

15	14–5	4–0
D7EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 36. PLL Controller Divider 7 (PLLDIV7) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D7EN		Divider enable for SYSCLK7
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK7. Ratio value = RATIO + 1

6.6.26 PLL Controller Divider 8 (PLLDIV8) Register

The PLL controller divider 8 (PLLDIV8) register is shown in Figure 34 and described in Table 37 for PLLC1. PLLDIV8 controls the divider for PLLC1SYSCLK8. For PLLC1, the divider must always be enabled (bit D8EN=1). The PLLDIV8 register is not applicable to PLLC2.

Figure 34. PLL Controller Divider 8 (PLLDIV8) Register

31-16
Reserved
R-0

15	14-5	4-0
D8EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 37. PLL Controller Divider 8 (PLLDIV8) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D8EN		Divider enable for SYSCLK8
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK8. Ratio value = RATIO + 1

6.6.27 PLL Controller Divider 9 (PLLDIV9) Register

The PLL controller divider 9 (PLLDIV9) register is shown in Figure 35 and described in Table 37 for PLLC1. PLLDIV9 controls the divider for PLLC1SYSCLK9. For PLLC1, the divider must always be enabled (bit D9EN=1). The PLLDIV9 register is not applicable to PLLC2.

Figure 35. PLL Controller Divider 9 (PLLDIV9) Register

31 16
Reserved
R-0

15	14 5	4 0
D9EN	Reserved	RATIO
R/W-1	R-0	R/W-0

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

Table 38. PLL Controller Divider 9 (PLLDIV9) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	D9EN		Divider enable for SYSCLK9.
		0	Disable
		1	Enable
14-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	RATIO	0	Divider ratio for SYSCLK9. Ratio value = RATIO + 1

6.6.28 PLL Formula and Example Calculation

The PLLCx controller-generated frequency (VCO frequency) = Oscillator frequency * (2 * M) / (N+1) where M = PLLCx_PLLM and should be programmed with a multiplier value, and
N = PLLCx_PLLN which is the PLLx pre-divider ratio value and should be programmed with a value of 1 less than the intended divider.

PLLCx_PREDIV ratio = N+1, where N is the value to be programmed.

For instance, if the oscillator frequency is 24 MHz and if the intended VCO frequency for PLLC1 is 432 MHz, then the following would be the setting for M and N:
M = 9 and N = 0, then 2*M = 2 * 9 = 18.

The VCO frequency (for PLLC1) = oscillator frequency * (2 * M) / (N+1) = 24 MHz * (18 / (0+1)) = 432 MHz.

Similarly, if an intended VCO frequency for PLLC2 is 270 MHz, then the following be the setting for M and N:
M = 45 and N = 7, then 2*M = 2 * 45 = 90

The VCO frequency (for PLLC2) = oscillator frequency *(2 * M) / (N+1) = 24MHz * (90/(7+1)) = 270 MHz.

This is the way the VCO frequency is derived. It is then fed to the PLLCx post-divider and then to the SYSCLKx-dividers before feeding to individual modules.

7 Power and Sleep Controller

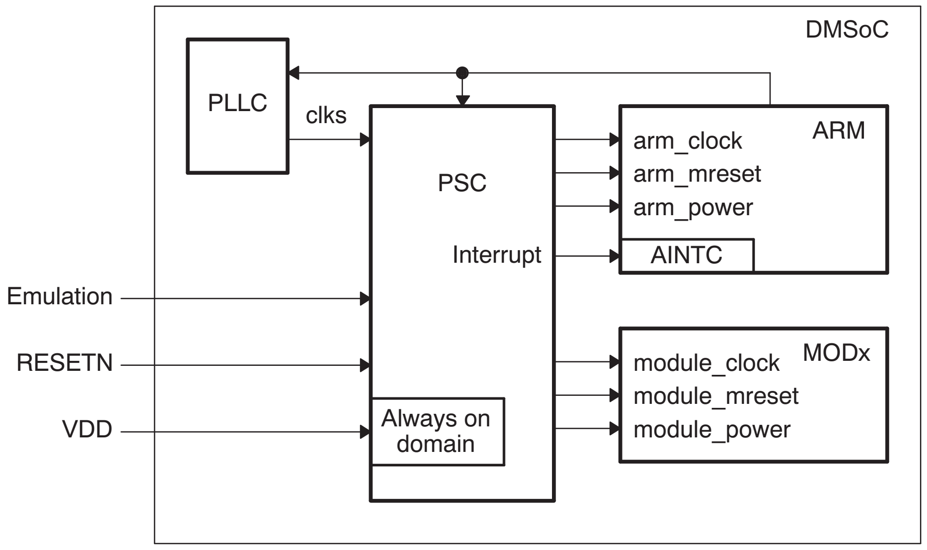
7.1 Introduction

In the DM36x system, the Power and Sleep Controller (PSC) is responsible for managing the transitions of system power on/off, clock on/off, and reset. A block diagram of the PSC is shown in Figure 36. Many of the operations of the PSC (e.g., power-on-reset operations) are transparent to software. However, the PSC provides you with an interface to control several important clock and reset operations. The clock and reset operations are the focus of this chapter.

The PSC includes the following features:

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

Figure 36. DM36x Power and Sleep Controller (PSC)



7.2 DM36x Power Domain and Module Topology

The DM36x system includes one power domain and 52 separate modules, as shown in Figure 37 and summarized in Table 39. The system's power domain is always on when the chip is on, and it is referred to as the AlwaysOn power domain. The AlwaysOn domain is powered by the V_{DD} pins of the DM36x chip (see the device data manual). All of the DM36x modules lie within the AlwaysOn power domain.

Table 39 shows the default state of each module after reset (power-on-reset, warm reset, and max reset). These states are defined in the following sections. The default state of some modules is determined by the boot select pins BTSEL[2:0]. For example, if UART boot mode is selected (BTSEL[2:0]=011), then the default state for the UART module is enabled. See Section 10 and Section 11 for additional information on reset, default configuration, and booting.

Figure 37. DM36x Power Domain and Module Topology

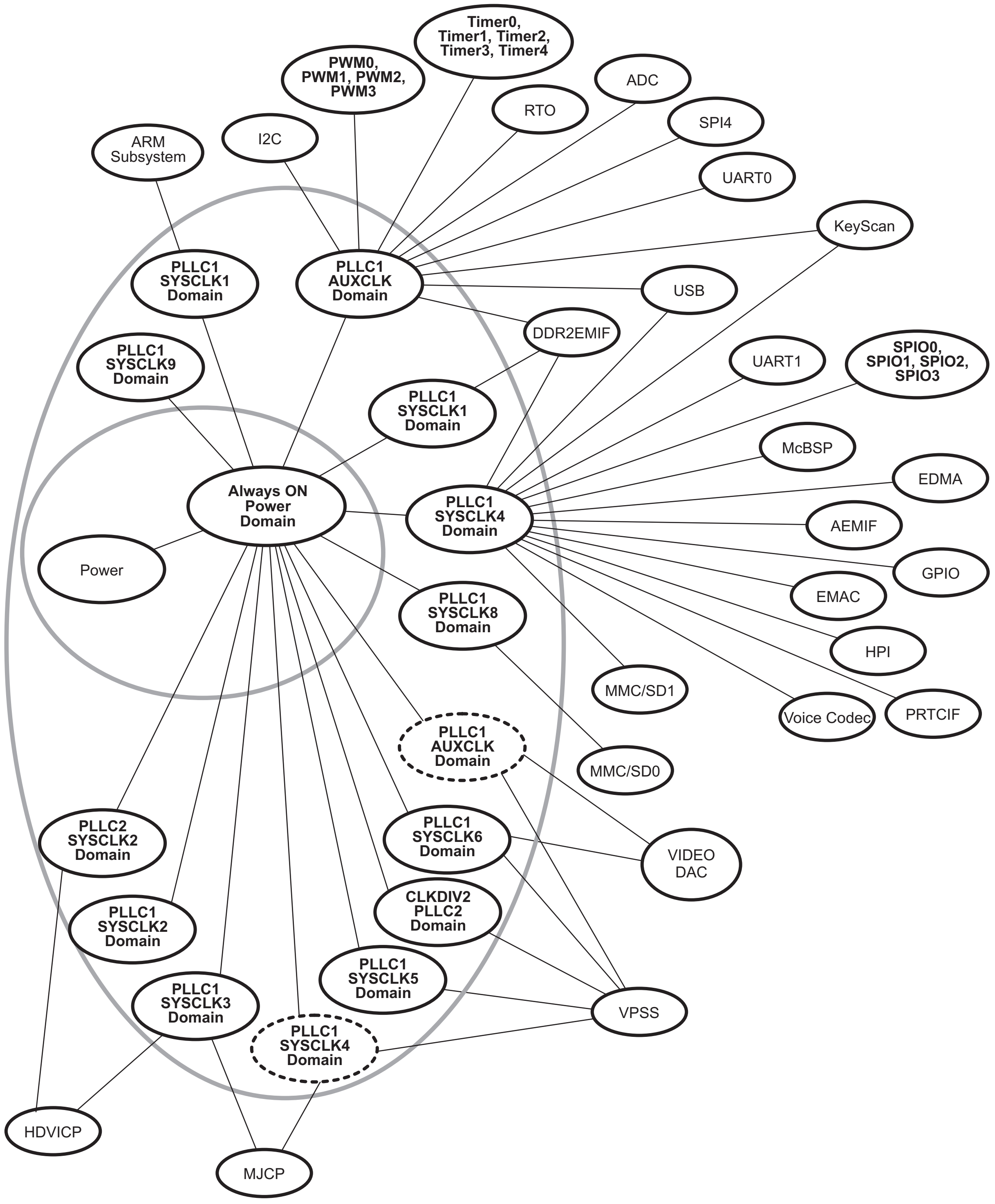


Table 39. Module Configuration

LPSC	Module Name	Power Domain	Power Domain State	Boot Modes (BTSEL[2:0])							
				000 ROM (NAND)	001 ROM (AEMIF)	010 ROM (SD)	011 ROM (UART)	100 ROM (USB)	101 ROM (SPI)	110 ROM (EMAC)	111 ROM (HPI)
0	EDMA CC	AlwaysOn	ON	On	On	**	**	On	**	On	**
1	EDMA TC0	AlwaysOn	ON	On	On	**	**	On	**	On	**
2	EDMA TC1	AlwaysOn	ON	**	**	**	**	**	**	**	**
3	EDMA TC2	AlwaysOn	ON	**	**	**	**	**	**	**	**
4	EDMA TC3	AlwaysOn	ON	**	**	**	**	**	**	**	**
5	TIMER3	AlwaysOn	ON	**	**	**	**	**	**	**	**
6	SPI1	AlwaysOn	ON	**	**	**	**	**	**	**	**
7	MMC/SD1	AlwaysOn	ON	**	**	**	**	**	**	**	**
8	McBSP	AlwaysOn	ON	**	**	**	**	**	**	**	**
9	USB	AlwaysOn	ON	**	**	**	**	On	**	**	**
10	PWM3	AlwaysOn	ON	**	**	**	**	**	**	**	**
11	SPI2	AlwaysOn	ON	**	**	**	**	**	**	**	**
12	RTO	AlwaysOn	ON	**	**	**	**	**	**	**	**
13	DDR2 EMIF	AlwaysOn	ON	**	**	**	**	**	**	**	**
14	AEMIF	AlwaysOn	ON	On	On	**	**	**	**	**	**
15	MMC/SD0	AlwaysOn	ON	**	**	On	**	**	**	**	**
16	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
17	TIMER4	AlwaysOn	ON	**	**	**	**	**	**	**	**
18	I2C	AlwaysOn	ON	**	**	**	**	**	**	**	**
19	UART0	AlwaysOn	ON	**	**	**	On	**	**	**	**
20	UART1	AlwaysOn	ON	**	**	**	**	**	**	**	**
21	HPI	AlwaysOn	ON	**	**	**	**	**	**	**	On
22	SPI0	AlwaysOn	ON	**	**	**	**	**	On	**	**
23	PWM0	AlwaysOn	ON	**	**	**	**	**	**	**	**
24	PWM1	AlwaysOn	ON	**	**	**	**	**	**	**	**
25	PWM2	AlwaysOn	ON	**	**	**	**	**	**	**	**
26	GPIO	AlwaysOn	ON	**	**	**	**	**	**	**	**
27	TIMER0	AlwaysOn	ON	On	On	On	On	On	On	On	On
28	TIMER1	AlwaysOn	ON	**	**	**	**	**	**	**	**
29	TIMER2	AlwaysOn	ON	On	On	On	On	On	On	On	On
30	System	AlwaysOn	ON	On	On	On	On	On	On	On	On
31	ARM	AlwaysOn	ON	On	On	On	On	On	On	On	On
32	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
33	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
34	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
35	Emulation	AlwaysOn	ON	On	On	On	On	On	On	On	On
36	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
37	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
38	SPI3	AlwaysOn	ON	**	**	**	**	**	**	**	**
39	SPI4	AlwaysOn	ON	**	**	**	**	**	**	**	**
40	EMAC	AlwaysOn	ON	**	**	**	**	**	**	On	**
41	PRTCIF	AlwaysOn	ON	On	On	On	On	On	On	On	On
42	KEYSCAN	AlwaysOn	ON	**	**	**	**	**	**	**	**
43	ADC	AlwaysOn	ON	**	**	**	**	**	**	**	**

Table 39. Module Configuration (continued)

LPSC	Module Name	Power Domain	Power Domain State	Boot Modes (BTSEL[2:0])							
44	Voice Codec	AlwaysOn	ON	**	**	**	**	**	**	**	**
45	VDAC CLKREC	AlwaysOn	ON	**	**	**	**	**	**	**	**
46	VDAC CLK	AlwaysOn	ON	**	**	**	**	**	**	**	**
47	VPSSMASTER	AlwaysOn	ON	**	**	**	**	**	**	**	**
48	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
49	Reserved	AlwaysOn	ON	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd
50	MJCP	AlwaysOn	ON	**	**	**	**	**	**	**	**
51	HDVICP	AlwaysOn	ON	**	**	**	**	**	**	**	**
Note: ** = Sync Reset; Rsvd=Reserved											

7.3 Power Domain and Module States Defined

7.3.1 Power Domain States

A power domain can only be in the ON state or the OFF state:

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

The AlwaysOn power domain is always in the ON state when the chip is powered-on.

7.3.2 Module States

A module can be in one of four states: Disable, Enable, SwResetDisable, or SyncReset. These four states correspond to combinations of module reset asserted or de-asserted and module clock on or off, as shown in Table 40.

Reset of a module is defined to completely reset the module hardware, such that all module hardware returns to its default state. See Section 10 and Section 11 for more information on module reset.

Table 40. Module States

Module State	Module Reset	Module Clock
Enable	De-asserted	On
Disable	De-asserted	Off
Sync Reset	Asserted	On
SwRestDisable	Asserted	Off

The module states are defined as follows:

Module State	Module State Definition
Enable	A module in the enable state has its module reset de-asserted and its clock on. This is the normal run-time state for a given module.
Disable	A module in the disable state has its module reset de-asserted and its clock off. This state is typically used for disabling a module clock to save static power. The DM36x is designed in full static CMOS, so when you stop a module clock, it retains the module's state. When the clock is restarted, the module resumes operating from the stopping point.
SyncReset	A module in the SyncReset state has its module reset asserted and its clock on. After initial power-on, most modules are in the SyncReset state by default (see Table 39). Generally, software is not expected to initiate this state.
SwResetDisable	A module in the SwResetDisable state has its module reset asserted and it has its clock set to off. Generally, software is not expected to initiate this state.

7.4 Executing State Transitions

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

7.4.1 Power Domain State Transitions

The AlwaysOn Power Domain is automatically transitioned to the ON state upon power-on-reset. No software intervention is required; the transition is automatically handled by the hardware.

7.4.2 Module State Transitions

The procedure for module state transitions is as follows ('x' corresponds to the module):

- Wait for the GOSTATx bit in PTSTAT to clear to 0x0. You must wait for any previously initiated transitions to finish before initiating a new transition.
- Set the NEXT bit in MDCTL[x] to SwResetDisable (0x0), SyncReset (0x1), Disable (0x2), or Enable (0x3).

  Note: You may set transitions in multiple NEXT bits in MDCTL[x] in this step.
- Set the GOx bit in PTCMD to 0x1 to initiate the transition(s).
- Wait for the GOSTATx bit in PTSTAT to clear to 0x0. The module is in the new state after the GOSTATx bit in PTSTAT clears to 0x0.

7.5 IcePick Emulation Support in the PSC

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

Table 41. 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.

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

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

7.6 PSC Interrupts

The PSC has an interrupt that is tied to the ARM interrupt controller. This interrupt is named PSCINT in the ARM interrupt map. The PSC interrupt is generated when certain IcePick emulation events occur.

7.6.1 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 42 and described in more detail in this section.

Table 42. PSC Interrupt Events

Interrupt Enable Bits		Interrupt Condition
Control Register	**Status Bit**	
MDCTLx	EMUIHB	Interrupt occurs when the emulation alters the module's state
MDCTLx	EMURST	Interrupt occurs when the emulation alters the module's local reset

7.6.1.1 Module State Emulation Events

A module state emulation event occurs when emulation alters the state of a module. The status is reflected in the EMUIHB bit in the MDSTAT[x]. In particular, a module state emulation event occurs under the following conditions:

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

7.6.1.2 Local Reset Emulation Events

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

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

7.6.2 Interrupt Registers

The PSC interrupt enable bits are: the EMUIHB bit in MDCTL[x], and the EMURSTIE bit in MDCTL[x].

Note: To interrupt the ARM, the ARM’s power and sleep controller interrupt (PSCINT) must also be enabled in the ARM interrupt controller. See Section 8 for more information on the ARM’s power and sleep controller interrupt and the ARM interrupt controller.

The PSC interrupt status bits are the Mx bit in the MERRPR0 register, the Mx bit in MERRPR1 register, the EMUIHB bit in the MDSTAT[x] register, and the EMURST bit in the MDSTAT[x] register. The status bits in the MERRPR0 and MERRPR1 registers are read by software to determine which module has generated an emulation interrupt, and then software can read the corresponding status bits in the MDSTATx register to determine which event caused the interrupt.

The PSC interrupt clear bits are the Mx bits in the MERRCRx register. The PSC interrupt evaluation bit is the ALLEV bit in the INTEVAL register. 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 the INTEVAL register is set to 0x1, the PSCINT is reasserted to the ARM interrupt controller. You must set the ALLEV bit in the INTEVAL register before exiting the PSCINT interrupt service routine to ensure that you do not miss any PSC interrupts while the ARM interrupts are globally disabled.

See Section 7.7 for complete descriptions of all PSC registers.

7.6.3 Interrupt Handling

The procedure for handling the PSC interrupts is described here.

- First, enable the interrupt.
  1. Set the EMUIHB bit in the MDCTL[x] register, and/or the EMURSTIE bit in the MDCTL[x] register to enable the interrupt events that you want.
     Note: There is no enable bit for the external power control pending interrupt event, so this event is always enabled. The PSC interrupt is sent to the ARM interrupt controller when at least one enabled event becomes active.
  2. Enable the ARM's power and sleep controller interrupt (PSCINT) in the ARM interrupt controller. To interrupt the ARM, PSCINT must be enabled in the ARM interrupt controller. See Section 8 for more information.
- The ARM enters the interrupt service routine (ISR) when it receives the interrupt.
  1. Read the Mx bit in the MERRPR0 register, and/or the Mx bit in the MERRPR1 register 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 MDSTAT[x] register, depending on the status bits read in the previous step to determine the event that caused the interrupt.
     - Service the interrupt as required by your application.
     - Write the Mx bit in the MERRCRx register to clear corresponding status.
     - Set the ALLEV bit in the INTEVAL register. Setting this bit reasserts the PSCINT to the ARM's interrupt controller, if there are still any active interrupt events.

7.7 PSC Registers

Table 43 lists the memory-mapped registers for the PSC. See the device memory map Table 7 for the memory address of these registers. The default PSC configurations (after reset), are shown in Table 39.

NOTE: Do not read or write reserved PSC register fields. In particular, registers associated with module 39 are reserved and must not be read or written.

Table 43. PSC Registers

Offset	Register	Description	Section
0h	PID	Peripheral Revision and Class Information	Section 7.7.1
18h	INTEVAL	Interrupt Evaluation Register	Section 7.7.2
40h	MERRPR0	Module Error Pending Register 0	Section 7.7.3
44h	MERRPR1	Module Error Pending Register 1	Section 7.7.4
50h	MERRCR0	Module Error Clear Register 0	Section 7.7.5
54h	MERRCR1	Module Error Clear Register 1	Section 7.7.6
120h	PTCMD	Power Domain Transition Command Register	Section 7.7.7
128h	PTSTAT	Power Domain Transition Status Register	Section 7.7.8
800h	MDSTAT[52]	Module Status Registers	Section 7.7.9
A00h	MDCTL[52]	Module Control Registers	Section 7.7.10

NOTE: Default PSC configurations (after reset) are shown in Table 39 .

7.7.1 Peripheral Revision and Class Information (PID) Register

The peripheral revision and class information (PID) register is shown in Figure 38 and described in Table 44.

Figure 38. Peripheral Revision and Class Information (PID) Register

31-30	29-28	27-16
SCHEME	Reserved	FUNC
R-0x1	R-00	R-0x482

15-11	10-8	7-6	5-0
RTL	MAJOR	CUSTOM	MINOR
R-0x28	R-0x1	R-0	R-0x5

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

Table 44. Peripheral Revision and Class Information (PID) Register Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	0x1h	Scheme
29-28	Reserved	0	Any writes to these bit(s) must always have a value of 0.
27-16	FUNC	0x482	Software compatible
15-11	RTL	0x28	RTL Version
10-8	MAJOR	0x1	Major revision
7-6	CUSTOM	0	Indicates a special version for a particular device
5-0	MINOR	0x5	Minor revision

7.7.2 Interrupt Evaluation (INTEVAL) Register

The interrupt evaluation (INTEVAL) register is shown in Figure 39 and described in Table 45.

Figure 39. Interrupt Evaluation (INTEVAL) Register

31 … 1	0
Reserved	ALLEV
R-0	W-0

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

Table 45. Interrupt Evaluation (INTEVAL) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Any writes to these bit(s) must always have a value of 0.
0	ALLEV		Re-evaluate PSC interrupt
		0	A write of 0 has no effect
		1	Write 1 to re-evaluate the interrupt condition

7.7.3 Module Error Pending Register 0 (MERRPR0) for Modules 0-31

The module error pending register 0 (MERRPR0) for modules 0-31 is shown in Figure 40 and described in Table 46.

Figure 40. Module Error Pending Register 0 (MERRPR0) for Modules 0-31

31 ... 0
M0[32]
R-0

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

Table 46. Module Error Pending Register 0 (MERRPR0) for Modules 0-31 Field Descriptions

Bit	Field	Value	Description
31-0	M0[32]		Module interrupt status bit for modules 0-31
		0	Module interrupt is not active
		1	Module interrupt is active

7.7.4 Module Error Pending Register 1 (MERRPR1) for Modules 32-51

The module error pending register 1 (MERRPR1) for modules 32-51 is shown in Figure 41 and described in Table 47.

Figure 41. Module Error Pending Register 1 (MERRPR1) for Modules 32-51

31 – 19	18 – 0
Reserved	M0[19]
R-0	R-0

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

Table 47. Module Error Pending Register 1 (MERRPR1) for Modules 32-51 Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-0	M0[19]		Module interrupt status bit for modules 32-51
		0	Module interrupt is not active
		1	Module interrupt is active

7.7.5 Module Error Clear Register 0 (MERRCR0) for Modules 0-31

The module error clear register 0 (MERRCR0) for modules 0-31 is shown in Figure 42 and described in Table 48.

Figure 42. Module Error Clear Register 0 (MERRCR0) for Modules 0-31

31 ... 0
M0[32]
R-0

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

Table 48. Module Error Clear Register 0 (MERRCR0) for Modules 0-31 Field Descriptions

Bit	Field	Value	Description
31-0	M0[32]		Clears the interrupt bit set in the corresponding MERRPR0 and the MDSTAT interrupt bit fields. This pertains to module 0-31.
		0	A write of 0 has no effect
		1	Clears module interrupt

7.7.6 Module Error Clear Register 1 (MERRCR1) for Modules 32-51

The module error clear register 1 (MERRCR1) for modules 32-51 is shown in Figure 43 and described in Table 49.

Figure 43. Module Error Clear Register 1 (MERRCR1) for Modules 32-51

31 – 19	18 – 0
Reserved	M0[19]
R-0	R/W-0

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

Table 49. Module Error Clear Register 1 (MERRCR1) for Modules 32-51 Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-0	M0[19]		Clears the interrupt bit set in the corresponding MERRPR1 register bit field and the MDSTAT interrupt bit fields. This pertains to module 32-51.
		0	A write of 0 has no effect
		1	Clears module interrupt

7.7.7 Power Domain Transition Command (PTCMD) Register

The power domain transition command (PTCMD) register is shown in Figure 44 and described in Table 50.

Figure 44. Power Domain Transition Command (PTCMD) Register

31 1	0
Reserved	GO
R-0	W-0

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

Table 50. Power Domain Transition Command (PTCMD) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Any writes to these bit(s) must always have a value of 0.
0	GO		Power domain GO transition command
		0	A write of 0 has no effect
		1	Writing 1 causes the state transition interrupt generation block to evaluate the new PTNEXT and the NEXT states in MDCTL as the desired states of the application.

7.7.8 Power Domain Transition Status (PTSTAT) Register

The power domain transition status register (PTSTAT) is shown in Figure 45 and described in Table 51 .

Figure 45. Power Domain Transition Status (PTSTAT) Register

31 1	0
Reserved	GOSTAT
R-0	R-0

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

Table 51. Power Domain Transition Status (PTSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Any writes to these bit(s) must always have a value of 0.
0	GOSTAT		Power domain transition status
		0	No transition in progress
		1	Power domain is transitioning (i.e., either the power domain is transitioning or modules in this power domain are transitioning).

7.7.9 Module Status Registers (MDSTATn) where n represents modules 0-51

The module status (MDSTATn) registers' format is shown in Figure 46 and described in Table 52. There are a total of 52 32-bit MDSTAT registers and they all have the same format. For the reserved LPSC modules, listed in Table 39, the corresponding MDSTATn register value is also reserved.

Figure 46. Module Status Registers (MDSTATn) where n represents modules 0-51

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

15–13	12	11	10	9	8
Reserved	MCKOUT	MRSTDONE	MRST	LRSTDONE	LRST
R-0	R-0	R-1	R-1	R-1	R-1

7–6	5–0
Reserved	STATE
R-0	R-0x3

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

Table 52. Module Status Registers (MDSTATn) where n represents modules 0-51 Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Any writes to these bit(s) must always have a value of 0.
17	EMUIHB		Emulation alters modules state interrupt active
		0	Interrupt not active
		1	Interrupt active
16	EMURST		Emulation alters module reset interrupt active
		0	Interrupt not active
		1	Interrupt active
15-13	Reserved	0	Any writes to these bit(s) must always have a value of 0.
12	MCKOUT		Module clock output status. Shows status of module clock ON / OFF.
		0	Module clock is off
		1	Module clock is on
11	MRSTDONE		Module reset done. Software is responsible for checking that mode reset is done before accessing the module.
		0	Module reset is not done
		1	Module reset is done
10	MRST		Module reset status. Reflects actual state of module reset.
		0	Module reset is asserted
		1	Module reset is de-asserted.
9	LRSTDONE		Module local reset initialization done status
		0	Local reset initialization not done
		1	Local reset initialization done
8	LRST		Module local reset actual status
		0	Local reset mod_lrst asserted
		1	Local reset mod_lrst de-asserted
7-6	Reserved	0	Any writes to these bit(s) must always have a value of 0.

Table 52. Module Status Registers (MDSTATn) where n represents modules 0-51 Field Descriptions (continued)

Bit	Field	Value	Description
5-0	STATE		Module state status: indicates current module status
		0	SwResetDisable state
		1	SyncReset state
		2	Disable state
		3	Enable state
		4h-3Fh	Indicates a transition

7.7.10 Module Control Registers (MDCTLn) where n represents modules 0-51

The module control (MDCTLn) registers' format is shown in Figure 47 and described in Table 53. There are a total of 52 32-bit MDCTL registers and they all have the same format. For the reserved LPSC modules, listed in Table 39, the corresponding MDCTLn register value is also reserved.

Figure 47. Module Control Registers (MDCTLn) where n represents modules 0-51

31-16
Reserved
R-0

15-11	10	9	8	7-5	4-0
Reserved	EMUIHBIE	EMURSTIE	LRST	Reserved	NEXT
R-0	R/W-1	R/W-1	R/W-1	R-0	R/W-0x3

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

Table 53. Module Control Registers (MDCTLn) where n represents modules 0-51 Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10	EMUIHBIE		Interrupt enable for emulation alters module state
		0	Disable interrupt
		1	Enable interrupt
9	EMURSTIE		Interrupt enable for emulation alters reset
		0	Disable interrupt
		1	Enable interrupt
8	LRST		Module local reset control
		0	Assert local reset
		1	Deassert local reset
7-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4-0	NEXT		Module next state
		0	SwResetDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

8 Interrupt Controller

8.1 Introduction

The DM36x 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. Each interrupt channel is mappable to an FIQ or to an IRQ type of interrupt, and each channel can be enabled or disabled. The INTC 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 INTC'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 AINTC takes up to 64 ARM device interrupts and maps them to either the IRQ or to the FIQ of the ARM. Each interrupt is also assigned one of eight priority levels (two for FIQ, six for IRQ). For interrupts with the same priority level, the priority is determined by the hardware interrupt number (the lowest number has the highest priority).

Table 54. AINTC Interrupt Connections

Interrupt Number	Interrupt Name	Source	Interrupt Number	Interrupt Name	Source
0	VPSSINT0	VPSS - INT0	32	TINT0	Timer 0 - TINT12
1	VPSSINT1	VPSS - INT1	33	TINT1	Timer 0 - TINT34
2	VPSSINT2	VPSS - INT2	34	TINT2	Timer 1 - TINT12
3	VPSSINT3	VPSS - INT3	35	TINT3	Timer 1 - TINT34
4	VPSSINT4	VPSS - INT4	36	PWM0INT	PWM0
5	VPSSINT5	VPSS - INT5	37	PWM1INT	PWM1
6	VPSSINT6	VPSS - INT6	38	PWM2INT or TINT8	PWM2 or Timer 4 - TINT12
7	VPSSINT7 or NSFINT	VPSS - INT7 or MJCP NSFINT	39	I2CINT	I2C
8	VPSSINT8 or IMX1INT	VPSS - INT8 or MJCP IMX1INT	40	UART0INT	UART0
9	SEQINT	MJCP SEQINT	41	UART1INT	UART1
10	IMX0INT or HDVICP_ARMINT	MJCP IMX0INT or HDVICP_ARMINT	42	SPI0INT0	SPI0 - SPIINT0
11	MJCPINT	MJCP	43	SPI0INT1 or SPI3INT0	SPI0 - SPIINT1 or SPI3 - SPIINT0
12	USBINT	USB OTG Controller	44	GIO0	GPIO0
13	RTOINT or TINT4	RTO or Timer 2 - TINT12	45	GIO1	GPIO1
14	TINT5	Timer 2 - TINT34	46	GIO2	GPIO2
15	TINT6	Timer 3 - TINT12	47	GIO3	GPIO3
16	EDMA CC_INT0	EDMA CC Region 0	48	GIO4	GPIO4

Table 54. AINTC Interrupt Connections (continued)

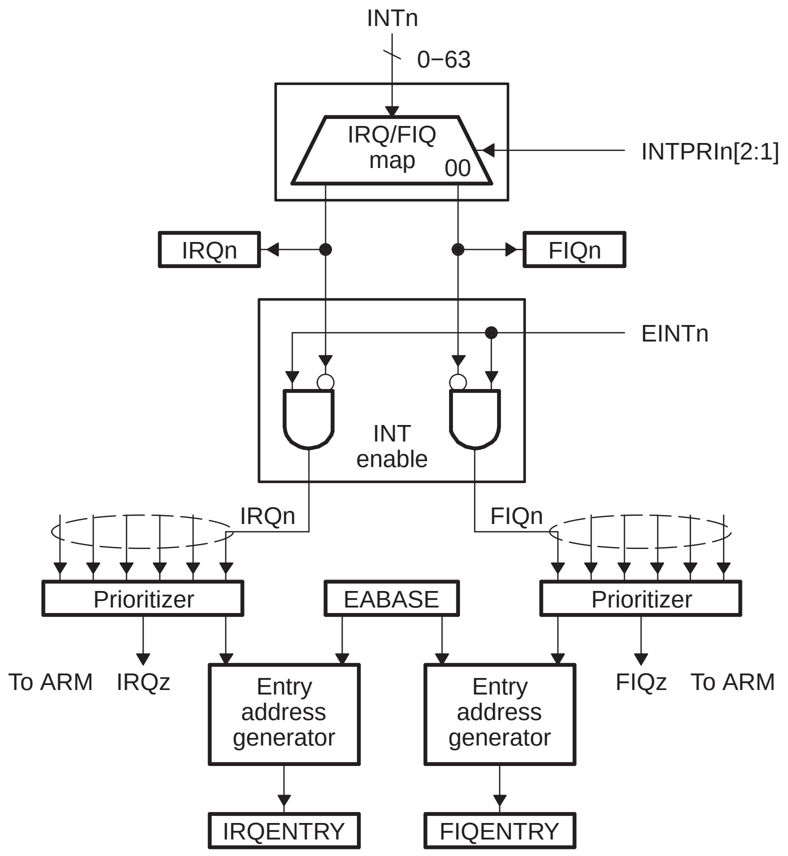
Interrupt Number	Interrupt Name	Source	Interrupt Number	Interrupt Name	Source
17	SPI1INT0 or CCERRINT	SPI1 - SPIINT0 or EDMA CC Error Interrupt	49	GIO5	GPIO5
18	SPI1INT1 or EDMA TC0_ERRINT	SPI1 - SPIINT1 or EDMA TC0 Error Interrupt	50	GIO6	GPIO6
19	SPI2INT0 or EDMA TC1_ERRINT	SPI2 - SPIINT0 or EDMA TC1 Error Interrupt	51	GIO7	GPIO7
20	PSCINT or TVINT	PSC - ALLINT or TVINT from VDAC	52	GIO8 or EMACRXTHREESH	GPIO8 or EMAC
21	SPI2INT1	SPI2 - SPIINT1	53	GIO9 or EMACRXPULSE	GPIO9 or EMAC
22	TINT7	Timer 3 - TINT34	54	GIO10 or EMACTXPULSE	GPIO10 or EMAC
23	SDIO0INT	MMC/SD0	55	GIO11 or EMACMISCPULSE	GPIO11 or EMAC
24	XINT or VCINT	McBSP or Voice Codec	56	GIO12 or PWRGIO0	GPIO12or PWRCTROGIO0
25	RINT	McBSP	57	GIO13 or PWRGIO1	GPIO13 or PWRCTROGIO1
26	MMC0INT	MMC/SD0	58	GIO14 or PWRGIO2	GPIO14 or PWRCTROGIO2
27	MMC1INT	MMC/SD1	59	GIO15 or ADCINT	GPIO15 or ADC
28	PWM3INT or TINT9	PWM3 or Timer 4 – TINT34	60	KEYINT	Keyscan
29	DDRINT or RTCINT	DDR2 EMIF or PRTCSS	61	COMMTX or EDMA TC2_ERRINT	ARMSS or EDMA TC2 Error Interrupt
30	AEMIFINT or HINT	Async EMIF or HPI	62	COMMRX or EDMA TC3_ERRINT	ARMSS or EDMA TC3 Error Interrupt
31	SDIO1INT	MMC/SD1	63	EMUINT	E2ICE

8.3 INTC Methodology

INTC methodology is illustrated in Figure 48 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 is generated to the ARM926 core (depending on whether the interrupt channel is mapped to IRQ or FIQ interrupt). The ARM then branches to the IRQ or FIQ interrupt routine.
- The INTC 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 the 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 48. 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 0x0 or 0x1 are designated as FIQs. Those with priorities of 0x2-0x7 are designated as IRQs. The appropriate IRQ/FIQ registers capture interrupt events. Each event causes an IRQ or FIQ to be generated 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 eight 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.

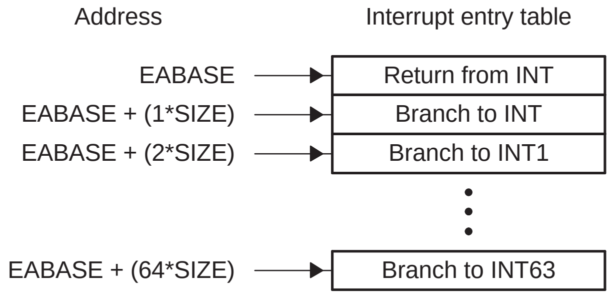
8.3.3 Vector Table Entry Address Generation

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 49.

Figure 49. 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 (enabled or not).

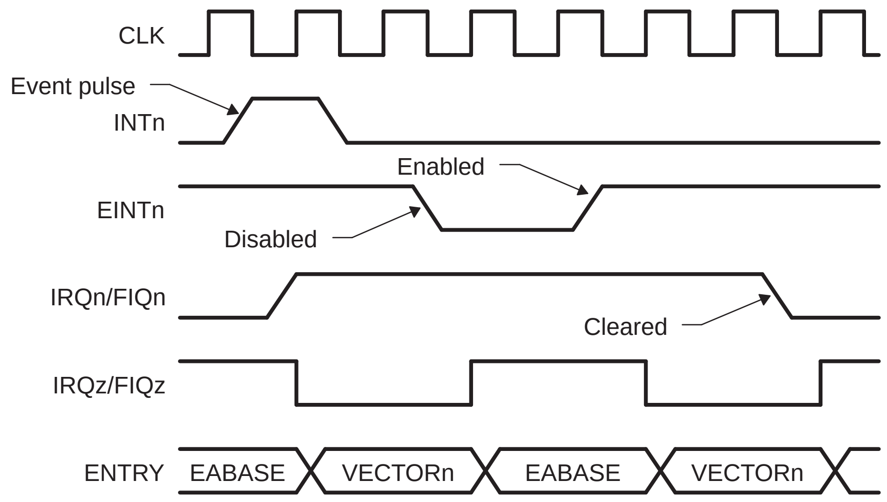
8.3.4 Clearing Interrupts

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

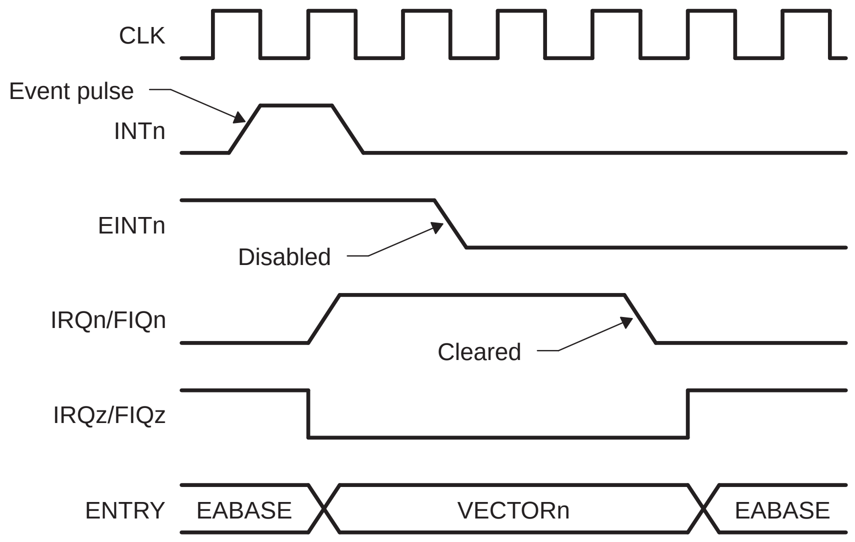
The AINTC has two methods for enabling and disabling interrupts, immediate or delayed, based on the setting of the IDMODE bit in the INTCTL register. When the bit is set to 0 (default), clearing an interrupt's EINT bit has an immediate effect. The prioritizer removes the disabled interrupt from consideration and adjusts the IRQ/FIQENTRY value correspondingly. If no other interrupts are pending, 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 50.

Figure 50. 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 51.

Figure 51. Delayed Interrupt Disable



8.4 INTC Registers

Table 55 lists the memory-mapped registers for the INTC. See the device memory map (Table 7) for the memory address of these registers.

Table 55. Interrupt Controller (INTC) Registers

Offset	Acronym	Register Description	Section
00h	FIQ0	Fast Interrupt Request Status 0 Register	Section 8.4.1
04h	FIQ1	Fast Interrupt Request Status 1 Register	Section 8.4.2
08h	IRQ0	Interrupt Request Status 0 Register	Section 8.4.3
0Ch	IRQ1	Interrupt Request Status 1 Register	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	Section 8.4.10
30h	INTPRI0	Interrupt Priority 0 Register	Section 8.4.11
34h	INTPRI1	Interrupt Priority 1 Register	Section 8.4.12
38h	INTPRI2	Interrupt Priority 2 Register	Section 8.4.13
3Ch	INTPRI3	Interrupt Priority 3 Register	Section 8.4.14
40h	INTPRI4	Interrupt Priority 4 Register	Section 8.4.15
44h	INTPRI5	Interrupt Priority 5 Register	Section 8.4.16
48h	INTPRI6	Interrupt Priority 6 Register	Section 8.4.17
4Ch	INTPRI7	Interrupt Priority 7 Register	Section 8.4.18

8.4.1 Fast Interrupt Request Status 0 (FIQ0) Register

The fast interrupt request status 0 (FIQ0) of INT[31:0] (if mapped to FIQ0) register is shown in Figure 52 and described in Table 56.

Figure 52. Fast Interrupt Request 0 (FIQ0) Register

31 ... 0
FIQ[31:0]
R/W-1

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

Table 56. Fast Interrupt Request 0 (FIQ0) Register Field Descriptions

Bit	Field	Value	Description
31-0	FIQ[31:0]		Interrupt status of INTx, if mapped to FIQ.
		0	Read: Interrupt occurred
		1	Write: Acknowledge interrupt

8.4.2 Fast Interrupt Request Status 1 (FIQ1) Register

The fast interrupt request status 1 (FIQ1) of INT[63:32] (if mapped to FIQ1) register is shown in Figure 53 and described in Table 57.

Figure 53. Fast Interrupt Request Status 1 (FIQ1) Register

31 0
FIQ[63:32]
R/W-1

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

Table 57. Fast Interrupt Request Status 1 (FIQ1) Register Field Descriptions

Bit	Field	Value	Description
31-0	FIQ[63:32]		Interrupt status of INTx, if mapped to FIQ.
		0	Read: Interrupt occurred
		1	Write: Acknowledge interrupt

8.4.3 Interrupt Request Status 0 (IRQ0) Register

The interrupt request status of INT[31:0] (if mapped to IRQ0) register is shown in Figure 54 and described in Table 58.

Figure 54. Interrupt Request Status 0 (IRQ0) Register

31	0
IRQ[31:0]	
R/W-1	

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

Table 58. Interrupt Request Status 0 (IRQ0) Register Field Descriptions

Bit	Field	Value	Description
31-0	IRQ[31:0]		Interrupt status of INTx, if mapped to IRQ
		0	Read: Interrupt occurred
		1	Write: Acknowledge interrupt

8.4.4 Interrupt Request Status 1 (IRQ1) Register

The interrupt request status of INT[63:32] (if mapped to IRQ1) register is shown in Figure 55 and described in Table 59.

Figure 55. Interrupt Request Status 1 (IRQ1) Register

31 ... 0
IRQ[63:32]
R/W-1

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

Table 59. Interrupt Request Status 1 (IRQ1) Register Field Descriptions

Bit	Field	Value	Description
31-0	IRQ		Interrupt status of INTx, if mapped to IRQ
		0	Read: Interrupt occurred
		1	Write: Acknowledge interrupt

8.4.5 Fast Interrupt Request Entry Address (FIQENTRY) Register

The fast interrupt request entry address (FIQENTRY) register is shown in Figure 56 and described in Table 60.

Figure 56. Fast Interrupt Request Entry Address (FIQENTRY) Register

31 ... 0
FIQENTRY
R-0

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

Table 60. Fast Interrupt Request Entry Address (FIQENTRY) Register Field Descriptions

Bit	Field	Value	Description
31-0	FIQENTRY	0	Interrupt entry table address of the current highest-priority FIQ

8.4.6 Interrupt Request Entry Address (IRQENTRY) Register

The interrupt request entry address (IRQENTRY) register is shown in Figure 57 and described in Table 61.

Figure 57. Interrupt Request Entry Address (IRQENTRY) Register

31	0
IRQENTRY	
R-0	

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

Table 61. Interrupt Request Entry Address (IRQENTRY) Register Field Descriptions

Bit	Field	Value	Description
31-0	IRQENTRY	0	Interrupt entry table address of the current highest-priority IRQ

8.4.7 Interrupt Enable Register 0 (EINT0)

The interrupt enable register 0 (EINT0) for INT[31:0] is shown in Figure 58 and described in Table 62.

Figure 58. Interrupt Enable Register 0 (EINT0)

31	0
EINT[31:0]	
R/W-0	

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

Table 62. Interrupt Enable Register 0 (EINT0) Field Descriptions

Bit	Field	Value	Description
31-0	EINT[31:0]		Interrupt enable for INTx
		0	Mask interrupt
		1	Enable interrupt

8.4.8 Interrupt Enable Register 1 (EINT1)

The interrupt enable register 1 (EINT1) for INT[63:32] is shown in Figure 59 and described in Table 63.

Figure 59. Interrupt Enable Register 1 (EINT1)

31	0
EINT[63:32]	
R/W-0	

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

Table 63. Interrupt Enable Register 1 (EINT1) Field Descriptions

Bit	Field	Value	Description
31-0	EINT		Interrupt enable for INTx
		0	Mask interrupt
		1	Enable interrupt

8.4.9 Interrupt Operation Control (INTCTL) Register

The interrupt operation control (INTCTL) register is shown in Figure 60 and described in Table 64.

Figure 60. Interrupt Operation Control (INTCTL) Register

31–16
Reserved
R-0

15–3	2	1	0
Reserved	IDMODE	IERAW	FERAW
R-0	R/W-0	R/W-0	R/W-0

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

Table 64. Interrupt Operation Control (INTCTL) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2	IDMODE		Interrupt disable mode
		0	Disable immediately
		1	Disable after ack
1	IERAW		Masked interrupt reflected in the IRQENTRY register
		0	Disable reflect
		1	Enable reflect
0	FERAW		Masked interrupt reflect in FIQENTRY register
		0	Disable reflect
		1	Enable reflect

8.4.10 EABASE Register

The EABASE register is shown in Figure 61 and described in Table 65.

Figure 61. EABASE Register

31–16
EABASE
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 65. EABASE Register Field Descriptions

Bit	Field	Value	Description
31-3	EABASE	0	Interrupt entry table base address (8-byte aligned)
2	Reserved		Any writes to these bit(s) must always have a value of 0.
1-0	SIZE		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 Priority0 (INTPRI0) Register

The interrupt priority0 (INTPRI0) register is shown in Figure 62 and described in Table 66.

Figure 62. Interrupt Priority0 (INTPRI0) Register

31	30	29	28	27	26	25	24
Reserved	INT7			Reserved	INT6		
R-0	R/W-7			R-0	R/W-7		

23	22	21	20	19	18	17	16
Reserved	INT5			Reserved	INT4		
R-0	R/W-7			R-0	R/W-7		

15	14	13	12	11	10	9	8
Reserved	INT3			Reserved	INT2		
R-0	R/W-7			R-0	R/W-7		

7	6	5	4	3	2		0
Reserved	INT1			Reserved	INT0		
R-0	R/W-7			R-0	R/W-7		

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

Table 66. Interrupt Priority 0 (INTPRI0) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT7	0x7	Selects INT7 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT6	0x7	Selects INT6 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT5	0x7	Selects INT5 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT4	0x7	Selects INT4 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT3	0x7	Selects INT3 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT2	0x7	Selects INT2 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT1	0x7	Selects INT1 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT0	0x7	Selects INT0 priority level

8.4.12 Interrupt Priority1 (INTPRI1) Register

The interrupt priority1 (INTPRI1) register is shown in Figure 63 and described in Table 67.

Figure 63. Interrupt Priority1 (INTPRI1) Register

31	30-28	27	26-24
Reserved	INT15	Reserved	INT14
R-0	R/W-7	R-0	R/W-7

23	22-20	19	18-16
Reserved	INT13	Reserved	INT12
R-0	R/W-7	R-0	R/W-7

15	14-12	11	10-8
Reserved	INT11	Reserved	INT10
R-0	R/W-7	R-0	R/W-7

7	6-4	3	2-0
Reserved	INT9	Reserved	INT8
R-0	R/W-7	R-0	R/W-7

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

Table 67. Interrupt Priority1 (INTPRI1) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT15	0x7	Selects INT15 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT14	0x7	Selects INT14 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT13	0x7	Selects INT13 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT12	0x7	Selects INT12 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT11	0x7	Selects INT11 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT10	0x7	Selects INT10 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT9	0x7	Selects INT9 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT8	0x7	Selects INT8 priority level

8.4.13 Interrupt Priority 2 (INTPRI2) Register

The interrupt priority 2 (INTPRI2) register is shown in Figure 64 and described in Table 68.

Figure 64. Interrupt Priority2 (INTPRI2) Register

31	30–28	27	26–24
Reserved	INT23	Reserved	INT22
R-0	R/W-7	R-0	R/W-7

23	22–20	19	18–16
Reserved	INT21	Reserved	INT20
R-0	R/W-7	R-0	R/W-7

15	14–12	11	10–8
Reserved	INT19	Reserved	INT18
R-0	R/W-7	R-0	R/W-7

7	6–4	3	2–0
Reserved	INT17	Reserved	INT16
R-0	R/W-7	R-0	R/W-7

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

Table 68. Interrupt Priority 2 (INTPRI2) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT23	0x7	Selects INT23 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT22	0x7	Selects INT22 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT21	0x7	Selects INT21 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT20	0x7	Selects INT20 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT19	0x7	Selects INT19 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT18	0x7	Selects INT18 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT17	0x7	Selects INT17 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT16	0x7	Selects INT16 priority level

8.4.14 Interrupt Priority 3 (INTPRI3) Register

The interrupt priority 3 (INTPRI3) register is shown in Figure 65 and described in Table 69.

Figure 65. Interrupt Priority 3 (INTPRI3) Register

31	30	28	27	26	24
Reserved	INT31		Reserved	INT30	
R-0	R/W-7		R-0	R/W-7	
23	22	20	19	18	16
Reserved	INT29		Reserved	INT28	
R-0	R/W-7		R-0	R/W-7	
15	14	12	11	10	8
Reserved	INT27		Reserved	INT26	
R-0	R/W-7		R-0	R/W-7	
7	6	4	3	2	0
Reserved	INT25		Reserved	INT24	
R-0	R/W-7		R-0	R/W-7	

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

Table 69. Interrupt Priority 3 (INTPRI3) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT31	0x7	Selects INT31 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT30	0x7	Selects INT30 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT29	0x7	Selects INT29 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT28	0x7	Selects INT28 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT27	0x7	Selects INT27 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT26	0x7	Selects INT26 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT25	0x7	Selects INT25 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT24	0x7	Selects INT24 priority level

8.4.15 Interrupt Priority 4 (INTPRI4) Register

The interrupt priority4 (INTPRI4) register is shown in Figure 66 and described in Table 70.

Figure 66. Interrupt Priority 4 (INTPRI4) Register

31	30 28	27	26 24
Reserved	INT39	Reserved	INT38
R-0	R/W-7	R-0	R/W-7

23	22 21 20	19	18 16
Reserved	INT37	Reserved	INT36
R-0	R/W-7	R-0	R/W-7

15	14 12	11	10 8
Reserved	INT35	Reserved	INT34
R-0	R/W-7	R-0	R/W-7

7	6 4	3	2 0
Reserved	INT33	Reserved	INT32
R-0	R/W-7	R-0	R/W-7

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

Table 70. Interrupt Priority 4 (INTPRI4) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT39	0x7	Selects INT39 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT38	0x7	Selects INT38 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT37	0x7	Selects INT37 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT36	0x7	Selects INT36 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT35	0x7	Selects INT35 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT34	0x7	Selects INT34 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT33	0x7	Selects INT33 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT32	0x7	Selects INT32 priority level

8.4.16 Interrupt Priority 5 (INTPRI5) Register

The interrupt priority 5 (INTPRI5) register is shown in Figure 67 and described in Table 71.

Figure 67. Interrupt Priority 5 (INTPRI5) Register

31	30 28	27	26 24
Reserved	INT47	Reserved	INT46
R-0	R/W-7	R-0	R/W-7

23	22 20	19	18 16
Reserved	INT45	Reserved	INT44
R-0	R/W-7	R-0	R/W-7

15	14 12	11	10 8
Reserved	INT43	Reserved	INT42
R-0	R/W-7	R-0	R/W-7

7	6 4	3	2 0
Reserved	INT41	Reserved	INT40
R-0	R/W-7	R-0	R/W-7

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

Table 71. Interrupt Priority 5 (INTPRI5) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT47	0x7	Selects INT47 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT46	0x7	Selects INT46 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT45	0x7	Selects INT45 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT44	0x7	Selects INT44 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT43	0x7	Selects INT43 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT42	0x7	Selects INT42 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT41	0x7	Selects INT41 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT40	0x7	Selects INT40 priority level

8.4.17 Interrupt Priority 6 (INTPRI6) Register

The interrupt priority 6 (INTPRI6) register is shown in Figure 68 and described in Table 72.

Figure 68. Interrupt Priority 6 (INTPRI6) Register

31	30-28	27	26-24
Reserved	INT55	Reserved	INT54
R-0	R/W-7	R-0	R/W-7

23	22-20	19	18-16
Reserved	INT53	Reserved	INT52
R-0	R/W-7	R-0	R/W-7

15	14-12	11	10-8
Reserved	INT51	Reserved	INT50
R-0	R/W-7	R-0	R/W-7

7	6-4	3	2-0
Reserved	INT49	Reserved	INT48
R-0	R/W-7	R-0	R/W-7

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

Table 72. Interrupt Priority 6 (INTPRI6) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT55	0x7	Selects INT55 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT54	0x7	Selects INT54 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT53	0x7	Selects INT53 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT52	0x7	Selects INT52 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT51	0x7	Selects INT51 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT50	0x7	Selects INT50 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT49	0x7	Selects INT49 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT48	0x7	Selects INT48 priority level

8.4.18 Interrupt Priority 7 (INTPRI7) Register

The interrupt priority 7 (INTPRI7) register is shown in Figure 69 and described in Table 73.

Figure 69. Interrupt Priority 7 (INTPRI7) Register

31	30	28	27	26	24
Reserved	INT63		Reserved	INT62	
R-0	R/W-7		R-0	R/W-7	

23	22	20	19	18	16
Reserved	INT61		Reserved	INT60	
R-0	R/W-7		R-0	R/W-7	

15	14	12	11	10	8
Reserved	INT59		Reserved	INT58	
R-0	R/W-7		R-0	R/W-7	

7	6	4	3	2	0
Reserved	INT57		Reserved	INT56	
R-0	R/W-7		R-0	R/W-7	

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

Table 73. Interrupt Priority 7 (INTPRI7) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-28	INT63	0x7	Selects INT63 priority level
27	Reserved	0	Any writes to these bit(s) must always have a value of 0.
26-24	INT62	0x7	Selects INT62 priority level
23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	INT61	0x7	Selects INT61 priority level
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	INT60	0x7	Selects INT60 priority level
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-12	INT59	0x7	Selects INT59 priority level
11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	INT58	0x7	Selects INT58 priority level
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	INT57	0x7	Selects INT57 priority level
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	INT56	0x7	Selects INT56 priority level

9 System Control Module

9.1 Overview

The system control module consists of a set of status and control registers, accessible by the ARM and which support the following system features and operations:

- Device identification
- Device configuration
  - Pin multiplexing control
  - Device boot configuration status
- ARM Interrupt and EDMA event multiplexing control
- Special peripheral status and control
  - Timer Input control
  - USB PHY control
  - VPSS clock and Video DAC control and status
  - DDR2 VTP control
  - Clock out Control
  - GIO debounce control
  - HPI control
- Power management
  - Deep sleep control
- Bandwidth management
  - Bus master DMA priority control

9.2 Device Identification

The DEVICE_ID register of the system control module contains a software readable version of the JTAG ID device. Software can use this register to determine the version of the device. The register format and description are shown in a Figure 79 and Table 89, respectively.

9.3 Device Configuration

The device configuration contains registers for controlling pin multiplexing and the boot configuration status.

9.3.1 Pin Multiplexing Control

The device makes extensive use of pin multiplexing to accommodate the large number of peripheral functions in the smallest possible package. A combination of hardware configuration (configuration pins latched at device reset) and program controlled PINMUX registers control pin multiplexing. Hardware does not attempt to ensure that the proper pin multiplexing is selected for the peripherals, or for the interface which is being used.

9.3.1.1 Program Controlled Pin Multiplexing

All pin multiplexing options can be controlled by software using the five pin mux registers. The format of the registers and a description of the pins they control are found in their respective sections. A brief overview is shown in Table 74.

Table 74. PINMUX Register Tables

Register Name	Register	Register Description
PINMUX0 Register	Pin Mux 0 Register Figure 70	Table 79
PINMUX1 Register	Pin Mux 1 Register Figure 71	Table 80
PINMUX2 Register	Pin Mux 2 Register Figure 72	Table 81
PINMUX3 Register	Pin Mux 3 Register Figure 73	Table 82
PINMUX4 Register	Pin Mux 4 Register Figure 74	Table 83

9.3.2 Device Boot Configuration Status

The device boot configuration (the state of the BTSEL[2:0] and AECFG[2:0] signals are captured in the BOOTCFG register), as shown in Figure 75 and Table 84.

9.4 ARM Interrupt and EDMA Event Multiplexing Control

The ARM_INTMUX and EDMA_EVTMUX registers are registers that control the multiplexing for interrupts to ARM and events to the EDMA, respectively. These registers are discussed in Figure 76 and Figure 77.

9.5 Special Peripheral Status and Control

Several of the peripheral modules require additional system-level control logic. These registers are discussed in detail in Section 9.12.

9.5.1 Timer Input Control

The timer input control (TIMER64_CTL) register controls the GPIO input selection for input to Timers. See Figure 81 and Table 91.

9.5.2 USB PHY Control

The USB_PHY_CTL register controls various features of the USB PHY, as shown in Figure 82 and Table 92.

9.5.3 VPSS Clock and DAC Control and Status

Clocks for the video processing subsystem are controlled via the VPSS_CLK_CTRL register. Video DAC configuration is controlled by VDAC_CONFIG register as shown in Figure 80 and Table 90.

9.6 Clock Out Configuration Status

The device supports three clock output pins (CLKOUT[2:0]) for sourcing additional external components in a system. CLKOUT2 output is derived from PLLC1SYSCLK9 and a PLL clock divider DIV1 in the PERI_CLKCTL register .

- CLKOUT2 = PLLCL1SYSCLK9 / (DIV1+1)
- The CLKOUTx output is enabled by the CLKOUTxEN bit field in the PERI_CLKCTL register

9.7 GIO De-Bounce Control

The DEBOUNCE registers control whether GIO0-GIO7 pin inputs are de-bounced or not. The de-bounce logic cancels the chattering caused by mechanical switch or slow slope input. See Figure 89.

9.8 HPI Control

The HPI control register is available to configure the mode in which the HPI module operates and is shown in Figure 78 and Table 88.

9.9 Power Management

9.9.1 Deep Sleep Control

The DEEPSLEEP register contains bits for the Deep Sleep power mode. See Figure 88.

9.10 Bandwidth Management

9.10.1 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 DM36x master is shown in Table 75.

Table 75. Master IDs

MSTID	Master
0	ARM Instruction
1	ARM Data
2-7	Reserved
8	VPSS
9	MJCP
10	EDMA CC
11	HDVICP
12-15	Reserved
16	EDMA TC0 – read
17	EDMA TC0 – write
18	EDMA TC1 – read
19	EDMA TC1 – write
20	EDMA TC2 – read
21	EDMA TC2 – write
22	EDMA TC3 – read
23	EDMA TC3 – write
24-31	Reserved
32	EMAC
33	HPI
34	USB
35-63	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. The priority levels need to be tuned to obtain the best system performance for a particular application. Lower values indicate higher priority.

On DM36x, only VPSS and EDMA masters can generate priority values. For all other masters, the value is programmed in the chip-level MSTPRI0 and MSTPRI1 registers as shown in Table 78. The default priority level for each DM36x bus master is shown in Table 76. Application software can modify these values to obtain the desired system performance.

Table 76. Default Master Priorities

Master	Default Priority
VPSS[1]	0
EDMA TC0[2]	0
EDMA TC1[2]	0
EDMA TC2[2]	0
EDMA TC3[2]	0
ARM (DMA)	1
ARM (CFG)	1
USB	4
EMAC	4
HPI	4
HDVICP	5
MJCP	5

(1) Default value for VPSS is in the buffer control register (BCR) of the VPFE module
(2) Default value for EDMA CC is in the QUEPRI register of the EDMA module

9.11 HPI Pin Muxing

HPI pins are multiplexed with AEMIF pins. Table 77 shows the multiplexed HPI and AEMIF pins. When BTSEL[2:0] = 111b, then these pins will function as HPI pins. Otherwise, these pins will function as AEMIF pins.

Table 77. HPI Pin Muxing

AEMIF signal name	HPI signal name
$\overline{\text{EM\_CE0}}$	$\overline{\text{HCS}}$
$\overline{\text{EM\_CE1}}$	$\overline{\text{HAS}}$
EM_A1	$\overline{\text{HHWIL}}$
EM_ADV	HR/$\overline{\text{W}}$
EM_WAIT	$\overline{\text{HRDY}}$
$\overline{\text{EM\_OE}}$	$\overline{\text{HDS1}}$
$\overline{\text{EM\_WE}}$	$\overline{\text{HDS2}}$
EM_A2	HCNTLA
EM_A0	HCNTLB
EM_BA1	HINTN
EM_D[15:0]	HD[15:0]

9.12 System Control Register Descriptions

9.12.1 Introduction

Table 78 lists the memory-mapped registers for the system module. Refer to Table 7 for the memory address of these registers.

Table 78. System Module Registers

Offset	Acronym	Register Description	Section
0h	PINMUX0	Pin Mux 0 Register	Section 9.12.2
4h	PINMUX1	Pin Mux 1 Register	Section 9.12.3
8h	PINMUX2	Pin Mux 2 Register	Section 9.12.4
Ch	PINMUX3	Pin Mux 3 Register	Section 9.12.5
10h	PINMUX4	Pin Mux 4 Register	Section 9.12.6
14h	BOOTCFG	Boot Configuration Register	Section 9.12.7
18h	ARM_INTMUX	Multiplexing Control for ARM Interrupts Register	Section 9.12.8
1Ch	EDMA_EVTMUX	Multiplexing Control for EDMA Events Register	Section 9.12.9
24h	HPI_CTL	HPI Control Register	Section 9.12.10
28h	DEVICE_ID	Device ID Register	Section 9.12.11
2Ch	VDAC_CONFIG	Video DAC Configuration Register	Section 9.12.12
30h	TIMER64_CTL	Timer Input Control Register	Section 9.12.13
34h	USB_PHY_CTRL	USB PHY Control Register	Section 9.12.14
38h	MISC	Miscellaneous Control Register	Section 9.12.15
3Ch	MSTPRI0	Master Priorities 0 Register	Section 9.12.16
40h	MSTPRI1	Master Priorities 1 Register	Section 9.12.17
44h	VPSS_CLK_CTRL	VPSS Clock Mux Control Register	Section 9.12.18
48h	PERI_CLKCTL	Peripheral Clock Control Register	Section 9.12.19
4Ch	DEEPSLEEP	DEEPSLEEP Control Register	Section 9.12.20
54h	DEBOUNCE0	De-bounce GIO0 input Register	Section 9.12.21
58h	DEBOUNCE1	De-bounce GIO1 input Register	Section 9.12.21
5Ch	DEBOUNCE2	De-bounce GIO2 input Register	Section 9.12.21
60h	DEBOUNCE3	De-bounce GIO3 input Register	Section 9.12.21
64h	DEBOUNCE4	De-bounce GIO4 input Register	Section 9.12.21
68h	DEBOUNCE5	De-bounce GIO5 input Register	Section 9.12.21
6Ch	DEBOUNCE6	De-bounce GIO6 input Register	Section 9.12.21
70h	DEBOUNCE7	De-bounce GIO7 input Register	Section 9.12.21
74h	VTPIOCR	VTP IO Control Register	Section 9.12.22
78h	PUPDCTL0	IO cell pull up/down control 0 Register	Section 9.12.23
7Ch	PUPDCTL1	IO cell pull up/down control 1 Register	Section 9.12.24
80h	HDVICPBT	HDVICP boot Register	Section 9.12.25
84h	PLLC1_CONFIG	PLLC1 configuration Register	Section 9.12.26
88h	PLLC2_CONFIG	PLLC2 configuration Register	Section 9.12.27

9.12.2 Pin Mux 0 (PINMUX0) Register

The pin mux 0 (PINMUX0) register is shown in Figure 70 and described in Table 79. This register controls pin multiplexing for VPFE, MMC/SD0, MMC/SD1, McBSP, SPI3, GIO[49:43], and GIO[102:93].

NOTE: The Y input (YIN[7:0]) and C input (CIN[7:0]) buses can be swapped by programming the field bit YCINSWP in the VPFE CCD Configuration (CCDCFG) register (0x01C7 0136h). If the YCINSWP bit is 0 (default), YIN[7:0] = Y signal / CIN[7:0] = C signal. If the YCINSWP bit is 1, YIN[7:0] = C signal / CIN[7:0] = Y signal.

For more information, see the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) Users Guide* (SPRUFG8).

Figure 70. Pin Mux 0 (PINMUX0) Register

31						25	24
Reserved							MMC/SD0
R-0							R/W+0

23	22	21	20	19	18	17	16
GIO49	GIO48	GIO47	GIO46	GIO45	GIO44	GIO43	
R/W+0	R/W+0	R/W+0	R/W+0	R/W+0	R/W+0	R/W+00	

15	14	13	12	11	10	9	8
C_WE_FIELD		VD	HD	YIN0	YIN1	YIN2	YIN3
R/W+00		R/W+0	R/W+0	R/W+0	R/W+0	R/W+0	R/W+0

7	6	5	4	3	2	1	0
YIN4		YIN5		YIN6		YIN7	
R/W+00		R/W+00		R/W+00		R/W+00	

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

Table 79. Pin Mux 0 (PINMUX0) Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved		Any writes to these bit(s) must always have a value of 0.
24	MMC/SD0		MMC/SD0 pin multiplexing control
		0	MMCSD0_CLK; MMCSD0_CMD; MMCSD0_DATA3; MMCSD0_DATA2; MMCSD0_DATA1; MMCSD0_DATA0
		1	Reserved
23	GIO49		GIO49 and McBSP pin multiplexing control
		0	GIO49
		1	McBSP_DX
22	GIO48		GIO48 and McBSP pin multiplexing control
		0	GIO48
		1	McBSP_CLKX
21	GIO47		GIO47 and McBSP pin multiplexing control
		0	GIO47
		1	McBSP_FSX
20	GIO46		GIO46 and McBSP pin multiplexing control
		0	GIO46
		1	McBSP_DR
19	GIO45		GIO45 and McBSP pin multiplexing control
		0	GIO45
		1	McBSP_CLKR

Table 79. Pin Mux 0 (PINMUX0) Register Field Descriptions (continued)

Bit	Field	Value	Description
18	GIO44		GIO44 and McBSP pin multiplexing control
		0	GIO44
		1	McBSP_FSR
17-16	GIO43		GIO43, MMC/SD1, and AEMIF pin multiplexing control
		0	GIO43
		1	MMCSD1_CLK
		2	EM_A20
		3	Reserved
15-14	C_WE_FIELD		C_WE_FIELD (video in), GIO93 ,USB, and CLKOUT0 pin multiplexing control
		0	C_WE_FIELD
		1	GIO93
		2	CLKOUT0
		3	USBDRVVBUS
13	VD		VD (video in) and GIO94 pin multiplexing control
		0	VD
		1	GIO94
12	HD		HD (video in) and GIO95 pin multiplexing control
		0	HD
		1	GIO95
11	YIN0		YIN0 (video in) and GIO96 pin multiplexing control
		0	YIN0
		1	GIO96
10	YIN1		YIN1 (video in) and GIO97 pin multiplexing control
		0	YIN1
		1	GIO97
9	YIN2		YIN2 (video in) and GIO98 pin multiplexing control
		0	YIN2
		1	GIO98
8	YIN3		YIN3 (video in) and GIO99 pin multiplexing control
		0	YIN3
		1	GIO99
7-6	YIN4		YIN4 (video in), SPI3, and GIO100 pin multiplexing control
		0	YIN4
		1	GIO100
		2	SPI3_SOMI
		3	SPI3_SCS[1]
5-4	YIN5		YIN5 (video in), SPI3, and GIO101 pin multiplexing control
		0	YIN5
		1	GIO101
		2	SPI3_SCS[0]
		3	Reserved
3-2	YIN6		YIN6 (video in), SPI3, and GIO102 pin multiplexing control
		0	YIN6
		1	GIO102
		2	SPI3_SIMO
		3	Reserved

Table 79. Pin Mux 0 (PINMUX0) Register Field Descriptions (continued)

Bit	Field	Value	Description
1-0	YIN7		YIN7 (video in), SPI3, and GIO103 pin multiplexing control
		0	YIN7
		1	GIO103
		2	SPI3_SCLK
		3	Reserved

9.12.3 Pin Mux 1 (PINMUX1) Register

The pin mux 1 (PINMUX1) register is shown in Figure 71 and described in Table 80. This register controls pin multiplexing for PWM[3-0], RTO, VPBE, and GIO[92:79] pins.

Figure 71. Pin Mux 1 (PINMUX1) Register

31-24
Reserved
R-0

23	22	21-20	19-18	17	16
Reserved	VCLK	EXTCLK	FIELD	LCD_OE	HVSYNC
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15-14	13-12	11-10	9-8
COUT0	COUT1	COUT2	COUT3
R/W-0	R/W-0	R/W-0	R/W-0

7-6	5-4	3-2	1-0
COUT4	COUT5	COUT6	COUT7
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 80. Pin Mux 1 (PINMUX1) Register Field Descriptions

Bit	Field	Value	Description
31-23	Reserved		Any writes to these bit(s) must always have a value o f 0.
22	VCLK		VCLK (Video Out) and GIO79 pin multiplexing control
		0	VCLK
		1	GIO79
21-20	EXTCLK		EXTCLK (Video Out), PWM3, and GIO80 pin multiplexing control
		0	GIO80
		1	EXTCLK
		2	B2
		3	PWM3
19-18	FIELD		FIELD (Video Out), PWM3, and GIO81 pin multiplexing control
		0	GIO81
		1	FIELD
		2	R2
		3	PWM3
17	LCD_OE		LCD_OE (Video Out) and GIO82 pin multiplexing control
		0	LCD_OE
		1	GIO82
16	HVSYNC		HVSYNC/VSYNC (Video Out) and GIO[84:83] pin multiplexing control
		0	HSYNC / VSYNC
		1	GIO[84:83]
15-14	COUT0		COUT0 (Video Out), PWM3, and GIO85 pin multiplexing control
		0	GIO85
		1	COUT0
		2	PWM3
		3	Reserved

Table 80. Pin Mux 1 (PINMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
13-12	COUT1		COUT1 (Video Out), PWM3, STTRIG, and GIO86 pin multiplexing control
		0	GIO86
		1	COUT1
		2	PWM3
		3	STTRIG
11-10	COUT2		COUT2 (Video Out), PWM2, RTO3, and GIO87 pin multiplexing control
		0	GIO87
		1	COUT2
		2	PWM2
		3	RTO3
9-8	COUT3		COUT3 (Video Out), PWM2, RTO2, and GIO88 pin multiplexing control
		0	GIO88
		1	COUT3
		2	PWM2
		3	RTO2
7-6	COUT4		COUT4 (Video Out), PWM2, RTO1, and GIO89 pin multiplexing control
		0	GIO89
		1	COUT4
		2	PWM2
		3	RTO1
5-4	COUT5		COUT5(Video Out), PWM2, RTO0, and GIO90 pin multiplexing control
		0	GIO90
		1	COUT5
		2	PWM2
		3	RTO0
3-2	COUT6		COUT6 (Video Out), PWM, and GIO91 pin multiplexing control
		0	GIO91
		1	COUT6
		2	PWM1
		3	Reserved
1-0	COUT7		COUT7 (Video Out), PWM0, and GIO92 pin multiplexing control
		0	GIO92
		1	COUT7
		2	PWM0
		3	Reserved

9.12.4 Pin Mux 2 (PINMUX2) Register

The pin mux 2 (PINMUX2) register is shown in Figure 72 and described in Table 81. This register controls pin multiplexing for GIO[78:50] and AEMIF pins. Some of the register fields have default values set by external pins that allow control of the AEMIF configuration to match the boot mode. The AEMIF AECFG[2:0] configuration at boot time is shown in Table 85.

Figure 72. Pin Mux 2 (PINMUX2) Register

31 – 16
Reserved
R-0

15 – 13	12	11	10	9	8
Reserved	EM_CLK	EM_ADV	EM_WAIT	EM_WE_OE	$\overline{\text{EM\_CE1}}$
R-0	R/W+0	R/W+0	R/W+0	R/W+0	R/W+0

7	6	5 – 4	3 – 2	1 – 0
$\overline{\text{EM\_CE0}}$	EM_D15_8	EM_A7	EM_A3	EM_AR
R/W+0	R/W+0	R/W+00	R/W+00	R/W+00

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

Table 81. Pin Mux 2 (PINMUX2) Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Any writes to these bit(s) must always have a value of 0.
12	EM_CLK		EM_CLK (AEMIF) and GI050 pin multiplexing control
		0	EM_CLK
		1	GIO50
11	EM_ADV		EM_ADV (AEMIF) and GIO51 pin multiplexing control
		0	EM_ADV
		1	GIO51
10	EM_WAIT		EM_WAIT (AEMIF) and GIO52 pin multiplexing control
		0	EM_WAIT
		1	GIO52
9	EM_WE_OE		EM_WE_OE (AEMIF) and GIO[54:53] pin multiplexing control
		0	$\overline{\text{EM\_WE}}$ and $\overline{\text{EM\_OE}}$
		1	GIO[54:53]
8	$\overline{\text{EM\_CE1}}$		$\overline{\text{EM\_CE1}}$ (AEMIF) and GIO55 pin multiplexing control
		0	$\overline{\text{EM\_CE1}}$
		1	GIO55
7	$\overline{\text{EM\_CE0}}$		$\overline{\text{EM\_CE0}}$ (AEMIF) and GIO56 pin multiplexing control
		0	$\overline{\text{EM\_CE0}}$
		1	GIO56
6	EM_D[15:8]		EM_D[15:8] (AEMIF) and GIO[64:57] pin multiplexing control. Reset value is set by AECFG[2] which sets the AEMIF bus width for boot as follows: If PINMUX2 [6] = AECFG[2] = 0, then 8- bit data bus EM_D[7:0] is available If PINMUX2 [6] = AECFG[2] = 1, then 16-bit data bus EM_D[15:0] is available
		0	GIO[64:57]
		1	EM_D[15:8]

Table 81. Pin Mux 2 (PINMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
5-4	EM_A7		EM_A7 (AEMIF) and GIO72 pin multiplexing control. Reset value is set by AECFG[1:0] which sets EM_A3 accordingly. During device boot, PINMUX2[5:4] = AECFG[1:0] If AECFG[1:0] = 00, then EM_A7 = 00 If AECFG[1:0] = 01 or 10, then EM_A7 = 01 or 10
		0	GIO72
		1	EM_A7
		2	EM_A7
		3	KEYA3
3-2	EM_A3		EM_A3 (AEMIF) and GIO68 pin multiplexing control. During device boot PINMUX2 [3:2] = AECFG[1:0] If AECFG[1:0] = 00, then EM_A3 = 00 If AECFG[1:0] = 01 or 10, then EM_A3 = 01 or 10
		0	GIO68
		1	EM_A3
		2	EM_A3
		3	KEYB3
1-0	EM_AR		EM_AR (AEMIF), GIO[78:73] ,GIO[71:69] ,GIO[78:73] and GIO[67:65] pin multiplexing control. During device boot PINMUX2 [1:0] = AECFG[1:0] If AECFG[1:0] = 00, then EM_AR =00 If AECFG[1:0] = 01, then EM_AR= 01 If AECFG[1:0] = 10, then EM_AR= 10
		0	GIO[78:73] ,GIO[71:69] and GIO[67:65]
		1	EM_A[13:8], EM_A[6:4], EM_A0, EM_BA1 and EM_A14
		2	EM_A[13:8], EM_A[6:4], EM_A0, EM_BA1 and EM_BA0
		3	GIO[78:73], KEYA[2:0] and KEYB[2:0]

9.12.5 Pin Mux 3 (PINMUX3) Register

The pin mux 3 (PINMUX3) register is shown in Figure 73and described in Table 82. This register controls pin multiplexing for EMAC, SPI1, PWM1, UART[1:0], I2C and GIO[26:1] pins.

Figure 73. Pin Mux 3 (PINMUX3) Register

31	30	29	28	27	26	25	24
GIO26	GIO25		GIO24	GI023		GIO22	GIO21
R/W-0	R/W-0		R/W-0	R/W-0		R/W-0	R/W-0

23	22	21	20	19	18	17	16
GIO21	GIO20		GIO19	GIO18	GIO17		GIO16
R/W-0	R/W-0		R/W-0	R/W-0	R/W-0		R/W-0

15	14	13	12	11	10	9	8
GIO16	GIO15	GIO14	GIO13	GIO12	GIO11	GIO10	GIO9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
GIO8	GIO7	GIO6	GIO5	GIO4	GIO3	GIO2	GIO1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 82. Pin Mux 3 (PINMUX3) Register Field Descriptions

Bit	Field	Value	Description
31	GIO26		GIO26 (GPIO) and SPI1 pin multiplexing control
		0	GIO26
		1	SPI1_SIMO
30-29	GIO25		GIO25 (GPIO), SPI0, UART1, and PWM1 pin multiplexing control
		0	GIO25
		1	SPI0_SCS[0]
		2	PWM1
		3	UART1_TXD
28	GIO24		GI024 (GPIO) and SPI0 pin multiplexing control
		0	GIO24
		1	SPI0_SCLK
27-26	GIO23		GIO23 (GPIO), SPI0, and PWM0 pin multiplexing control
		0	GIO23
		1	SPI0_SOMI
		2	SPI0_SCS[1]
		3	PWM0
25	GIO22		GIO22 (GPIO) and SPI0 pin multiplexing control
		0	GIO22
		1	SPI0_SIMO
24-23	GIO21		GIO21 (GPIO), UART1, and I2C pin multiplexing control
		0	GIO21
		1	UART1_RTS
		2	I2C_SDA
		3	Reserved

Table 82. Pin Mux 3 (PINMUX3) Register Field Descriptions (continued)

Bit	Field	Value	Description
22-21	GIO20		GIO20 (GPIO), UART1, and I2C pin multiplexing control
		0	GIO20
		1	UART1_CTS
		2	I2C_SCL
		3	Reserved
20	GIO19		GIO19 (GPIO) and UART0 pin multiplexing control
		0	GIO19
		1	UART0_RXD
19	GIO18		GIO18 (GPIO) and UART0 pin multiplexing control
		0	GIO18
		1	UART0_TXD
18-17	GIO17		GIO17 (GPIO), EMAC and UART1 pin multiplexing control
		0	GIO17
		1	EMAC_TX_EN
		2	UART1_RXD
		3	Reserved
16-15	GIO16		GIO16 (GPIO), EMAC and UART1 pin multiplexing control
		0	GIO16
		1	EMAC_TX_CLK
		2	UART1_TXD
		3	Reserved
14	GIO15		GIO15 (GPIO) and EMAC pin multiplexing control
		0	GIO15
		1	EMAC_COL
13	GIO14		GIO14 (GPIO) and EMAC pin multiplexing control
		0	GIO14
		1	EMAC_TXD[3]
12	GIO13		GIO13 (GPIO) and EMAC pin multiplexing control
		0	GIO13
		1	EMAC_TXD[2]
11	GIO12		GIO12 (GPIO) and EMAC pin multiplexing control
		0	GIO12
		1	EMAC_TXD[1]
10	GIO11		GIO11 (GPIO) and EMAC pin multiplexing control
		0	GIO11
		1	EMAC_TXD[0]
9	GIO10		GIO10 (GPIO) and EMAC pin multiplexing control
		0	GIO10
		1	EMAC_RXD[3]
8	GIO9		GIO9 (GPIO) and EMAC pin multiplexing control
		0	GIO9
		1	EMAC_RXD[2]
7	GIO8		GIO8 (GPIO) and EMAC pin multiplexing control
		0	GIO8
		1	EMAC_RXD[1]

Table 82. Pin Mux 3 (PINMUX3) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	GIO7		GIO7 (GPIO) and EMAC pin multiplexing control
		0	GIO7
		1	EMAC_RXD[0]
5	GIO6		GIO6 (GPIO) and EMAC pin multiplexing control
		0	GIO6
		1	EMAC_RX_CLK
4	GIO5		GIO5 (GPIO) and EMAC pin multiplexing control
		0	GIO5
		1	EMAC_RX_DV
3	GIO4		GIO4 (GPIO) and EMAC pin multiplexing control
		0	GIO4
		1	EMAC_RX_ER
2	GIO3		GIO3 (GPIO) and EMAC pin multiplexing control
		0	GIO3
		1	EMAC_CRS
1	GIO2		GIO2 (GPIO) and EMAC_MDIO pin multiplexing control
		0	GIO2
		1	EMAC_MDIO
0	GIO1		GIO1 (GPIO) and EMAC_MDIO pin multiplexing control
		0	GIO1
		1	EMAC_MDCLK

9.12.6 Pin Mux 4 (PINMUX4) Register

The pin mux 4 (PINMUX4) register is shown in Figure 74and described in Table 83. This register controls pin multiplexing for MMC/SD1, PWM, McBSP, UART1 ,SPI4 ,SPI2, SPI1, AEMIF, VENC and GIO[42:27] pins.

Figure 74. Pin Mux 4 (PINMUX4) Register

31-30	29-28	27-26	25-24
GIO42	GIO41	GIO40	GIO39
R/W-0	R/W-0	R/W-0	R/W-0

23-22	21-20	19-18	17-16
GIO38	GIO37	GIO36	GIO35
R/W-0	R/W-0	R/W-0	R/W-0

15-14	13-12	11-10	9-8
GIO34	GIO33	GIO32	GIO31
R/W-0	R/W-0	R/W-0	R/W-0

7-6	5-4	3-2	1-0
GIO30	GIO29	GIO28	GIO27
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 83. Pin Mux 4 (PINMUX4) Register Field Descriptions

Bit	Field	Value	Description
31-30	GIO42		GIO42 (GPIO), MMC/SD1 and AEMIF pin multiplexing control
		0	GIO42
		1	MMCSD1_CMD
		2	EM_A19
		3	Reserved
29-28	GIO41		GIO41 (GPIO), MC/SD1 and AEMIF pin multiplexing control
		0	GIO41
		1	MMCSD1_DATA3
		2	EM_A18
		3	Reserved
27-26	GIO40		GIO40 (GPIO), MMC/SD1, and AEMIF pin multiplexing control
		0	GIO40
		1	MMCSD1_DATA2
		2	EM_A17
		3	Reserved
25-24	GIO39		GIO39 (GPIO), MMC/SD1, and AEMIF pin multiplexing control
		0	GIO39
		1	MMCSD1_DATA1
		2	EM_A16
		3	Reserved
23-22	GIO38		GIO38 (GPIO), MMC/SD1, and AEMIF pin multiplexing control
		0	GIO38
		1	MMCSD1_DATA0
		2	EM_A15
		3	Reserved

Table 83. Pin Mux 4 (PINMUX4) Register Field Descriptions (continued)

Bit	Field	Value	Description
21-20	GIO37		GIO37 (GPIO), SPI4, McBSP, and CLKOUT0 pin multiplexing control
		0	GIO37
		1	SPI4_SCS[0]
		2	McBSP_CLKS
		3	CLKOUT0
19-18	GIO36		GIO36 (GPIO), SPI4, and AEMIF pin multiplexing control
		0	GIO36
		1	SPI4_SCLK
		2	EM_A21
		3	EM_A14
17-16	GIO35		GIO35 (GPIO), SPI4, and CLKOUT1 pin multiplexing control
		0	GIO35
		1	SPI4_SOMI
		2	SPI4_SCS[1]
		3	CLKOUT1
15-14	GIO34		GIO34 (GPIO), SPI4, and UART1 pin multiplexing control
		0	GIO34
		1	SPI4_SIMO
		2	SPI4_SOMI
		3	UART1_RXD
13-12	GIO33		GIO33 (GPIO), SPI2, VENC, and USB pin multiplexing control
		0	GIO33
		1	SPI2_SCS[0]
		2	USBDRVVBUS
		3	R1
11-10	GIO32		GIO32 (GPIO), VENC, and SPI2 pin multiplexing control
		0	GIO32
		1	SPI2_SCLK
		2	Reserved
		3	R0
9-8	GIO31		GIO31 (GPIO), SPI2, and CLKOUT2 pin multiplexing control
		0	GIO31
		1	SPI2_SOMI
		2	SPI2_SCS[1]
		3	CLKOUT2
7-6	GIO30		GIO30 (GPIO) and SPI2 pin multiplexing control
		0	GIO30
		1	SPI2_SIMO
		2	Reserved
		3	G1
5-4	GIO29		GIO29 (GPIO), VENC, and SPI1 pin multiplexing control
		0	GIO29
		1	SPI1_SCS[0]
		2	Reserved
		3	G0

Table 83. Pin Mux 4 (PINMUX4) Register Field Descriptions (continued)

Bit	Field	Value	Description
3-2	GIO28		GIO28 (GPIO), VENC, and SPI1 pin multiplexing control
		0	GIO28
		1	SPI1_SCLK
		2	Reserved
		3	B1
1-0	GIO27		GIO27(GPIO), VENC, and SPI1 pin multiplexing control
		0	GIO27
		1	SPI1_SOMI
		2	SPI1_SCS[1]
		3	B0

9.12.7 Boot Configuration (BOOTCFG) Register

The boot configuration (BOOTCFG) register is shown in Figure 75 and described in Table 84. This register reflects the status of the device boot configuration pins at boot time (the state of the BTSEL[2:0] and AECFG[2:0] pins).

Figure 75. Boot Configuration (BOOTCFG) Register

31–16
Reserved
R-0

15–9	8	7–5	4–3	2–0
Reserved	GIO0_RESET	BTSEL	OSC_SW	AECFG
R-0	R-0	R-0	R-10	R-000

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

Table 84. Boot Configuration (BOOTCFG) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Any writes to these bit(s) must always have a value of 0.
8	GIO0_RESET	0	GIO0 value sampled at reset prior to debounce circuit.
7-5	BTSEL	0	Configuration of BTSEL[2:0] pins at boot time 000 = Boot from ROM (NAND) 001 = Boot from AEMIF 010 = Boot from ROM (SD) 011 = Boot from ROM (UART) 100 = Boot from ROM (USB) 101 = Boot from ROM (SPI) 110 = Boot from ROM (EMAC) 111 = Boot from ROM (HPI)
4:3	OSC_SW[1:2]		Oscillator frequency mode
		10	15-35MHz
		01	30-40MHz
2-0	AECFG		AEMIF address width configuration at boot time as shown in Table 85.

Table 85. Async EMIF Configuration (AECFG) Pin Mux Coding

000	001	010	100	101	110
GIO[65]	EM_A[14]	EM_BA[0]	GIO[65]	EM_A[14]	EM_BA[0]
GIO[66]	EM_BA[1]	EM_BA[1]	GIO[66]	EM_BA[1]	EM_BA[1]
GIO[67]	EM_A[0]	EM_A[0]	GIO[67]	EM_A[0]	EM_A[0]
EM_A[1]	EM_A[1]	EM_A[1]	EM_A[1]	EM_A[1]	EM_A[1]
EM_A[2]	EM_A[2]	EM_A[2]	EM_A[2]	EM_A[2]	EM_A[2]
GIO[68]	EM_A[3]	EM_A[3]	GIO[68]	EM_A[3]	EM_A[3]
GIO[69]	EM_A[4]	EM_A[4]	GIO[69]	EM_A[4]	EM_A[4]
GIO[70]	EM_A[5]	EM_A[5]	GIO[70]	EM_A[5]	EM_A[5]
GIO[71]	EM_A[6]	EM_A[6]	GIO[71]	EM_A[6]	EM_A[6]
GIO[72]	EM_A[7]	EM_A[7]	GIO[72]	EM_A[7]	EM_A[7]
GIO[73]	EM_A[8]	EM_A[8]	GIO[73]	EM_A[8]	EM_A[8]
GIO[74]	EM_A[9]	EM_A[9]	GIO[74]	EM_A[9]	EM_A[9]
GIO[75]	EM_A[10]	EM_A[10]	GIO[75]	EM_A[10]	EM_A[10]
GIO[76]	EM_A[11]	EM_A[11]	GIO[76]	EM_A[11]	EM_A[11]
GIO[77]	EM_A[12]	EM_A[12]	GIO[77]	EM_A[12]	EM_A[12]
GIO[78]	EM_A[13]	EM_A[13]	GIO[78]	EM_A[13]	EM_A[13]
GIO[57]	GIO[57]	GIO[57]	EM_D[8]	EM_D[8]	EM_D[8]
GIO[58]	GIO[58]	GIO[58]	EM_D[9]	EM_D[9]	EM_D[9]
GIO[59]	GIO[59]	GIO[59]	EM_D[10]	EM_D[10]	EM_D[10]
GIO[60]	GIO[60]	GIO[60]	EM_D[11]	EM_D[11]	EM_D[11]
GIO[61]	GIO[61]	GIO[61]	EM_D{12]	EM_D{12]	EM_D{12]
GIO[62]	GIO[62]	GIO[62]	EM_D[13]	EM_D[13]	EM_D[13]
GIO[63]	GIO[63]	GIO[63]	EM_D[14]	EM_D[14]	EM_D[14]
GIO[64]	GIO[64]	GIO[64]	EM_D[15]	EM_D[15]	EM_D[15]

9.12.8 ARM Interrupt Mux Control (ARM_INTMUX) Register

The ARM interrupt mux control (ARM_INTMUX) register is shown in Figure 76 and described in Table 86. It provides multiplexing control for interrupts to the interrupt controller block. The INTC can support only 64 discrete events, so different events are multiplexed and controlled by this register.

Figure 76. ARM Interrupt Mux (ARM_INTMUX) Control Register

31	30 26	25	24	23	22	21	20	19	18	17	16
INT0	Reserved	INT7	INT8	INT62	INT61	INT59	INT58	INT57	INT56	INT55	INT54
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
INT53	INT52	INT43	INT38	INT30	INT29	INT28	INT26
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INT24	Reserved	INT20	INT19	INT18	INT17	INT13	INT10
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 86. ARM Interrupt Mux (ARM_INTMUX) Control Register Field Descriptions

Bit	Field	Value	Source
31	INT0		VPSS_INT0
		0	VPSS_INT0
		1	Reserved
30-26	Reserved	0	Any writes to these bit(s) must always have a value of 0.
25	INT7		VPSS_INT7 or MJCP: NSFINT
		0	VPSS_INT7
		1	MPEG/JPEG Coprocessor NSFINT
24	INT8		VPSS_INT8 or IMXI1NT
		0	VPSS_INT8
		1	MPEG/JPEG Coprocessor IMX1INT
23	INT62		COMMRX or EDMA3 TC3_ERRINT
		0	COMMRX
		1	EDMA TC3 Error Interrupt
22	INT61		COMMTX or EDMA3 TC2_ERRINT
		0	COMMTX
		1	EDMA TC2 Error Interrupt
21	INT59		GPIO15 or ADCINT
		0	GPIO15
		1	ADCINT
20	INT58		GPIO14 or PWRGIO2
		0	GPIO14
		1	PWRGIO2
19	INT57		GPIO13 or PWRGIO1
		0	GPIO13
		1	PWRGIO1
18	INT56		GPIO12 or PWRGIO0
		0	GPIO12
		1	PWRGIO0

Table 86. ARM Interrupt Mux (ARM_INTMUX) Control Register Field Descriptions (continued)

Bit	Field	Value	Source
17	INT55		GPIO11 or EMACMISCPULSE
		0	GPIO11
		1	EMACMISCPULSE
16	INT54		GPIO10 or EMACTXPULSE
		0	GPIO10
		1	EMACTXPULSE
15	INT53		GPIO9 or EMACRXPULSE
		0	GPIO9
		1	EMACRXPULSE
14	INT52		GPIO8 or EMACRXTHREESH
		0	GPIO8
		1	EMACRXTHREESH
13	INT43		SPI0INT1 or SPI3INT0
		0	SPI0 - SPIINT1
		1	SPI3 - SPIINT0
12	INT38		PWM2 or Timer 4 : TINT8
		0	PWM2
		1	Timer 4 : TINT8
11	INT30		Async EMIF or HPI
		0	Async EMIF
		1	HPI
10	INT29		DDR2 EMIF or PRTCSS
		0	DDR2 EMIF
		1	PRTCSS
9	INT28		PWM3 or Timer 4 : TINT9
		0	PWM3
		1	Timer 4 : TINT9
8	INT26		MMC/SD0
		0	MMC0
		1	Reserved
7	INT24		McBSP XINT or Voice Codec
		0	McBSP XINT
		1	VCINT
6	Reserved	0	Any writes to these bit(s) must always have a value of 0
5	INT20		PSC or TVINT
		0	PSC
		1	TVINT
4	INT19		SPI2INT0 or EDMA3 TC1_ERRINT
		0	SPI2 - SPIINT0
		1	EDMA TC1 Error Interrupt
3	INT18		SPI1INT1 or EDMA3 TC0_ERRINT
		0	SPI1 - SPIINT1
		1	EDMA TC0 Error Interrupt
2	INT17		SPI1INT0 or EDMA3 CC_ERRINT
		0	SPI1 - SPIINT0
		1	EDMA CC Error Interrupt

Table 86. ARM Interrupt Mux (ARM_INTMUX) Control Register Field Descriptions (continued)

Bit	Field	Value	Source
1	INT13		RTO or Timer2:TINT4
		0	RTO
		1	Timer2:TINT4
0	INT10		IMX0INT or HDVICP:HDVICP_ARMINT
		0	MPEG/JPEG Coprocessor IMX0INT
		1	HDVICP_ARMINT

9.12.9 EDMA Event Mux (EDMA_EVTMUX) Control Register

The EDMA events (EDMA_EVTMUX) register is shown in Figure 77 and described in Table 87. This register controls multiplexing for EDMA events due to the limited number of events supported by the EDMA.

Figure 77. EDMA Event Mux (EDMA_EVTMUX) Control Register

31–24
Reserved
R-0

23	22	21	20	19	18	17	16
Reserved	EVT63	EVT62	EVT61	EVT60	EVT59	EVT58	EVT57
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11–10	9	8
EVT56	EVT55	EVT54	EVT53	Reserved	EVT43	EVT42
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
EVT41	EVT40	EVT26	EVT19	EVT18	EVT12	EVT3	EVT2
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 87. EDMA Event Mux (EDMA_EVTMUX) Control Register Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22	EVT63		MJCP : COPCINT or HDVICP : CP_ECDEND
		0	0 =MPEG/JPEG Coprocessor COPCINT Event
		1	1 = HDVICP CP_ECDEND Event
21	EVT62		MJCP : RCNTINT or HDVICP : CP_MC
		0	0 = MPEG/JPEG Coprocessor RCNTINT Event
		1	1 = HDVICP CP_MC Event
20	EVT61		MJCP : VLCDERRINT or HDVICP: CP_ LPF
		0	0 = MPEG/JPEG Coprocessor VLCDERRINT Event
		1	1 = HDVICP CP_LPF Event
19	EVT60		MJCP : BPSINT or HDVICP : CP_BS
		0	0 = MPEG/JPEG Coprocessor BPSINT Event
		1	1 = HDVICP CP_BS Event
18	EVT59		MJCP : QIQINT or HDVICP: CP_IPE
		0	0 = MPEG/JPEG Coprocessor QIQINT Event
		1	1 = HDVICPr CP_IPE Event
17	EVT58		MJCP: DCTINT or HDVICP : CP_CALC
		0	0 =MPEG/JPEG Coprocessor DCTINT Event
		1	1 = HDVICPr CP_CALC Event
16	EVT57		MJCP : BIMINT or HDVICP : CP_ME
		0	0 = MPEG/JPEG Coprocessor BIMINT Event
		1	1 = HDVICP CP_ME Event
15	EVT56		MJCP : VLCDINT or HDVICP : CP_ECDCMP
		0	0 =MPEG/JPEG Coprocessor VLCDINT Event
		1	1 =HDVICP CP_ECDCMP Event

Table 87. EDMA Event Mux (EDMA_EVTMUX) Control Register Field Descriptions (continued)

Bit	Field	Value	Description
14	EVT55		PWM3 or HDVICP : CP_UNDEF
		0	0 = PWM3
		1	1 = HDVICP CP_UNDEF Event
13	EVT54		PWM2 or MJCP : NSFINT
		0	0 = PWM2
		1	1 = MPEG/JPEG Coprocessor NSFINT Event
12	EVT53		PWM1 or MJCP : IMX1INT
		0	0 = PWM1
		1	1 = MPEG/JPEG Coprocessor IMX1INT interrupt
11-10	Reserved		Any writes to these bit(s) must always have a value of 0.
9	EVT43		GPIO : GPINT14 or EMACMISCTHREESH
		0	0 = GPINT14
		1	1 = EMACMISCTHREESH
8	EVT42		GPIO : GPINT14 or EMACTXPULSE
		0	0 = GPINT14
		1	1 = EMACTXPULSE
7	EVT41		GPIO : GPINT14 or EMACRXPULSE
		0	0 = GPINT14
		1	1 = EMACRXPULSE
6	EVT40		GPIO : GPINT14 or EMACRXTHREESH
		0	0 = GPINT14
		1	1 = EMACRXTHRESH
5	EVT26		MMC0 : RXEVT
		0	0 = MMC0 : RXEVT
		1	Reserved
4	EVT19		UART0 : UTXEVT0 or SPI3: SPI3REVT
		0	0 = UTXEVT0
		1	1 = SPI3REVT
3	EVT18		UART0 : URXEVT0 or SPI3: SPI3XEVT
		0	0 = URXEVT0
		1	1 = SPI3XEVT
2	EVT12		MJCP : IMX0INT or HDVICP : HDVICP_ARMINT
		0	0 = MPEG/JPEG Coprocessor IMX0INT Event
		1	1 = HDVICP HDVICP_ARMINT Event
1	EVT3		McBSP : REVT or VoiceCodec : VCREVT
		0	0 = McBSP: REVT
		1	1 = VCREVT
0	EVT2		McBSP : XEVT or VoiceCodec : VCXEVT
		0	0 = McBSP: XEVT
		1	1 = VCXEVT

9.12.10 HPI Control (HPI_CTL) Register

The HPI control (HPI_CTL) register is shown in Figure 78 and described in Table 88.

Figure 78. HPI Control (HPI_CTL) Register

31–16
Reserved
R-0

15–10	9	8	7–0
Reserved	CTLMODE	ADDMODE	TIMEOUT
R-0	RW,+0	RW,+0	RW,+1000 0000

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

Table 88. HPI Control (HPI_CTL) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Any writes to these bit(s) must always have a value of 0.
9	CTLMODE		HPIC register write access
		0	HOST
		1	DM36x (if ADDMODE = 1)
8	ADDMODE		HPIA register write access
		0	HOST
		1	DM36x
7-0	TIMEOUT		Host burst write timeout value

9.12.11 Device ID (DEVICE_ID) Register

The device identification (DEVICE_ID) register is shown in Figure 79 and described in Table 89. This register provides the identification information for the TI ARM processor.

Figure 79. Device ID (DEVICE_ID) Register

31–28	27–16
DEVREV	PARTNUM
R-0	R-0xB83E

15–12	11–1	0
PARTNUM	MFGR	Reserved
R-0xB83E	R-0x017	R-1

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

Table 89. Device ID (DEVICE_ID) Register Field Descriptions

Bit	Field	Value	Description
31-28	DEVREV	0	DM36x Silicon version
27-12	PARTNUM	0xB83E	Device Part Number (Unique JTAG ID) [27] - ARM Core ID = 0: ARM processor [26:24] - Capability = 111: ARM Processor with J extension - soft macrocell [23:20] - Family = 1001: 0x9 [19:12] - Device Number = 0010 0110: 0x26
11-1	MFGR	0x017	Manufacturer's JTAG ID
0	Reserved	1	Any writes to these bit(s) must always have a value of 0.

9.12.12 Video Dac Configuration (VDAC_CONFIG) Register

The video DAC configuration (VDAC_CONFIG) register is shown in Figure 80 and described in Table 90.

Figure 80. Video Dac Configuration (VDAC_CONFIG) Register

31	30	29-24
TVSHORT	TVINT	Reserved
R-0	R-0	R-0x001

23-20	19	18-16
Reserved	PDTVSHORTZ	Reserved
R-0x001	R/W-0	R-0x554

15-6
Reserved
R-0x554

5	4	3	2	1	0
XDMODE	PWDNZ_TVDETECT	PWDNBUFZ	PWD_C	PWD_B	PWD_A
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 90. Video Dac Configuration (VDAC_CONFIG) Register Field Descriptions

Bit	Field	Value	Description
31	TVSHORT	0	TVSHORT performs short detection for the video output. When the output of either channel is shorted to ground, the signal TVSHORT bit is set to 1.
30	TVINT	0	TVINT performs detection of connection and disconnection of video signal. Reading ‘1” shows the connection of video signal.
29-20	Reserved	0x001	Any writes to these bit(s) must always have a value of 0.
19	PDTVSHORTZ	0	Output interrupt signal when TVOUT shorts to ground. Active high.
18-6	Reserved	0x554	Any writes to these bit(s) must always have a value of 0.
5	XDMODE		Select HD DAC mode / SD Video Buffer mode for DAC CH-C
		0	SD Video Buffer mode
		1	HD DAC mode
4	PWDNZ_TVDETECT		TVINT circuit enable signal
		0	Disable
		1	Enable
3	PWDNBUFZ		Power Down control for SD Video Buffer
		0	Power Down
		1	Normal
2	PWD_C		Power Down mode control for CH-C (COMPPR), VDAC channel also used for TVOUT
		0	Power down
		1	Normal
1	PWD_B		Power Down mode control for CH-B (COMPPB)
		0	Power down
		1	Normal
0	PWD_A		Power Down mode control for CH-A (COMPY)
		0	Power down
		1	Normal
Recommended values for SD DAC and HD DAC: HD DAC -> 0x1019 41E7h ; SD DAC -> 0x1019 41DCh			

9.12.13 Timer Input Control (TIMER64_CTL) Register

The timer input control (TIMER64_CTL) register is shown in Figure 81 and described in Table 91.

Figure 81. Timer Input Control (TIMER64_CTL) Register

31–16
Reserved
R-0

15–2	1	0
Reserved	GIO3_4	GIO1_2
R-0	R/W-0	R/W-0

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

Table 91. Timer Input Control (TIMER64_CTL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Any writes to these bit(s) must always have a value of 0.
1	GIO3_4		GIO3 or GIO4 for input to the timer
		0	GIO3 for input
		1	GIO4 for input
0	GIO1_2		GIO1 or GIO2 for input to the timer
		0	GIO1for input
		1	GIO2 for input

9.12.14 USB PHY Control (USB_PHY_CTRL) Register

The USB PHY control (USB_PHY_CTL) register is shown in Figure 82 and described in Table 92.

Figure 82. USB PHY Control (USB_PHY_CTRL)

31							24
Reserved							
R-0							

23							16
Reserved							
R-0							

15			12	11	10	9	8
PHYCLKFREQ				DATAPOL	PHYCLKSRC		PHYCLKGD
R/W-0				R/W-0	R/W-0		R-0

7	6	5	4	3	2	1	0
SESNDEN	VBDTCTEN	VBUSENS	PHYPLLON	Reserved	Reserved	OTGPDWN	PHYPDWN
R/W-1	R/W-1	R-0	R/W-0	R-0	R/W-1	R/W-1	R/W-1

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

Table 92. USB PHY Control (USB_PHY_CTRL) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15-12	PHYCLKFREQ		USB PHY clock source
		1	12 MHz clock (after dividing down 36 MHz crystal by 3) from clock divide circuitry – from PLLC1SYSCLKBP
		2	24 MHz clock
		4	19.2 MHz clock
		0,3,5-15	All values are reserved
11	DATAPOL		USB PHY data polarity inversion
		0	No inversion
		1	Inversion
10-9	PHYCLKSRC		USB PHY input clock source
		0	Crystal directly – from PLLC1AUXCLK
		1	12 MHz input (after dividing down 36 MHz crystal by 3) from clock divide circuitry – from PLLC1SYSCLKBP
		2	PLLC1SYSCLK1
		3	PLLC2SYSCLK1
8	PHYCLKGD		USB PHY power and clock good
		0	PHY power not ramped or PLL not locked
		1	PHY power is good and PLL is locked
7	SESNDEN		Session end comparator enable
		0	Comparator disabled
		1	Comparator enabled
6	VBDTCTEN		VBUS comparator enable
		0	Comparators (except session end) disabled
		1	Comparators (except session end) enabled
5	VBUSENS		OTG analog block VBUSSENSE output status
		0	VBUS not present (<0.5V)
		1	VBUS present (>0.5V)

Table 92. USB PHY Control (USB_PHY_CTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	PHYPLLON		USB PHY PLL suspend override
		0	Normal PLL operation
		1	Override PLL suspend state
3-2	Reserved		Any writes to these bit(s) must always have a value of 0.
1	OTGPDWN		USB OTG analog block power down control
		0	OTG analog block powered
		1	OTG analog block power off
0	PHYPDWN		USB PHY power down control
		0	PHY powered
		1	PHY power off

9.12.15 Miscellaneous Control (MISC) Register

The miscellaneous control (MISC) register is shown in Figure 83 and described in Table 93.

Figure 83. Miscellaneous Control (MISC) Register

31-16
Reserved
R-0

15-12	11	10	9-8
Reserved	MMC/SD0_INHB	HDVICP_INHB	BOOTST
R-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3-1	0
EDMA TC1_BST	CLKSTP_BYP1	CLKSTP_BYP0	TIMER2_WDT	Reserved	AIM_WAIST
R/W-0	R/W-0	R/W-0	R/W-1	R-0	R/W-1

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

Table 93. Miscellaneous Control (MISC) Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Any writes to these bit(s) must always have a value of 0.
11	MMC/SD0_INHB		MMC/SD0 bus access control
		0	Normal operation
		1	Inhibits the bus access to MMC/SD0 (Need to set 1 before stopping MMC/SD0 clock and reset to 0 after enabling MMC/SD0 clock)
10	HDVICP_INHB		HDVICP bus access control
		0	Normal operation
		1	Inhibits the bus access to HDVICP (Need to set 1 before stopping HDVICP clock and reset to 0 after enabling HDVICP clock)
9-8	BOOTST		Boot up status
		0	Boot failed in the first and latest attempts
		1	Boot failed in the first and passed in the latest attempt
		2	Invalid state
		3	Boot passed in the first attempt itself
7	EDMA TC1_BST		Specifies default burst size of EDMA TC1
		0	32-byte burst
		1	64-byte burst
6	CLKSTP_BYP1		Bypassing the clock stop Req/Ack handshake for EDMA
		0	Normal mode
		1	Bypass mode
5	CLKSTP_BYP0		Bypassing the clock stop Req/Ack handshake for McBSP
		0	Normal mode
		1	Bypass mode. Setting '1' to this register bypasses the request/acknowledge handshake between the PSC module and the McBSP module and forces the enabling of the clock stop operation.
4	TIMER2_WDT		TIMER2 Definition (Normal vs. WDT)
		0	TIMER2 is normal Timer
		1	TIMER2 is WDT
3-1	Reserved	0	Any writes to these bit(s) must always have a value of 0.
0	AIM_WAIST		ARM Internal Memory Wait States
		0	1 wait state to IRAM
		1	0 wait state to IRAM

9.12.16 Master Priorities 0 (MSTPRI0) Register

The master priorities 0 (MSTPRI0) register is shown in Figure 84 and described in Table 94.

Figure 84. Master Priorities 0 (MSTPRI0) Register

31-23	22-20	19	18-16
Reserved	HDVICP	Reserved	MJCP
R-0	R-0x5	R-0	R-0x5

15-7	6-4	3	2-0
Reserved	ARM_CFGP	Reserved	ARM_DMAP
R-0	R/W-0x1	R-0	R/W-0x1

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

Table 94. Master Priorities 0 (MSTPRI0) Register Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Any writes to these bit(s) must always have a value of 0.
22-20	HDVICP	0x05	HDVICP processing priority
19	Reserved	0	Any writes to these bit(s) must always have a value of 0.
18-16	MJCP	0x05	MJCP processing priority
15-7	Reserved	0	Any writes to these bit(s) must always have a value of 0.
6-4	ARM_CFGP	0x01	ARM CFG bus priority
3	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2-0	ARM_DMAP	0x01	ARM DMA priority

9.12.17 Master Priorities 1 (MSTPRI1) Register

The master priorities 1 (MSTPRI1) register is shown in Figure 85 and described in Table 95 .

Figure 85. Master Priorities 1 (MSTPRI1) Register

31 – 16
Reserved
R-0

15 – 11	10 – 8	7 – 0
Reserved	PERIP	Reserved
R-0	R/W-0x4	R-0x44

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

Table 95. Master Priorities 1 (MSTPRI1) Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Any writes to these bit(s) must always have a value of 0.
10-8	PERIP	0x4	Peripheral bus priority
7-0	Reserved	0x044	Any writes to these bit(s) must always have a value of 0.

9.12.18 VPSS Clock Mux Control (VPSS_CLK_CTRL) Register

The VPSS clock mux control (VPSS_CLK_CTRL) register is shown in Figure 86 and described in Table 96.

Figure 86. VPSS Clock Mux Control (VPSS_CLK_CTRL) Register

31–16
Reserved
R-0

15–8
Reserved
R-0

7	6–5	4	3	2	1–0
VPSS_CLKMD	VENC_CLK_SRC	DACCLKEN	VENCLKEN	PCLK_INV	VPSS_MUXSEL
R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0

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

Table 96. VPSS Clock Mux Control (VPSS_CLK_CTRL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7	VPSS_CLKMD		Config/EDMA bus clock versus VPSS clock ratio
		0	1:2
		1	1:1
6-5	VENC_CLK_SRC		27MHz/74.25MHz clock source
		0	PLLC1 SYSCLK6
		1	PLLC2 SYSCLK5
		2	MXI oscillator
		3	MXI oscillator
4	DACCLKEN		Video DAC clock enable
		0	Disable
		1	Enable
3	VENCLKEN		Video encoder clock enable.
		0	Disable
		1	Enable
2	PCLK_INV		Video encoder PCLK polarity
		0	VENC clk mux receives normal PCLK
		1	VENC clk mux receives inverted PCLK
1-0	VPSS_MUXSEL		VPSS and DAC clock selection
		0	VENC_CLK_SRC selected clock
		1	VENC_CLK_SRC selected clock
		2	EXTCLK input
		3	PCLK (or ~PCLK)

9.12.19 Peripheral Clock Control (PERI_CLKCTL) Register

The peripheral clock control (PERI_CLKCTL) register is shown in Figure 87 and described in Table 97.

Figure 87. Peripheral Clock Control (PERI_CLKCTL) Register

31	30	29	28	27	26	25	24
Reserved	PRTCSSCLKS	ARMCLKS	KEYSCLKS	DDRCLKS	HDVICPCLKS	DIV3	
R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0x3FF	

25	16
DIV3	
R/W-0x3FF	

15	7
DIV2	
R/W-0x1FF	

6 3	2	1	0
DIV1	CLOCKOUT2EN	CLOCKOUT1EN	CLOCKOUT0EN
0xF	R/W-0	R/W-0	R/W-0

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

Table 97. Peripheral Clock Control (PERI_CLKCTL) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30	PRTCCLKS		PRTCSS clock source selection
		0	RTCXI (OSC)
		1	PLLC1AUXCLK clock Divider
29	ARMCLKS		ARM926 clock source selection When changing the source clock (either 0 to 1 or 1 to 0), ARM926 clock frequency must be ≥ CFG/DMA bus clock frequency (PLLC1SYSCLK4).
		0	PLLC1SYSCLK2
		1	PLLC2SYSCLK2
28	KEYSCLKS		KeyScan clock source selection
		0	RTCXI (MXI)
		1	PLLC1AUXCLK clock Divider
27	DDRCLKS		DDR2 clock source selection
		0	PLLC1SYSCLK7
		1	PLLC2SYSCLK3
26	HDVICPCLKS		HDVICP Processing logic clock source selection
		0	PLLC1SYSCLK2
		1	PLLC2SYSCLK2
25-16	DIV3	0x3FF	PLL clock divider for Key Scan and PRTCSS
			Key Scan and PRTCSS Peripheral Clock = PLLC1AUXCLK / (DIV3+1)
15-7	DIV2	0x1FF	PLL clock divider for Voice Codec
			Voice Codec Peripheral Clock = PLLC2SYSCLK4 / (DIV2+1)
6-3	DIV1	0xF	PLL clock divider for CLKOUT2
			CLKOUT2 = PLLCL1SYSCLK9 / (DIV1+1)
2	CLOCKOUT2EN		CLOCKOUT2EN
		0	Output CLOCKOUT2 enable
		1	Output CLOCKOUT2 disable

Table 97. Peripheral Clock Control (PERI_CLKCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
1	CLOCKOUT1EN		CLOCKOUT1EN
		0	Output CLOCKOUT1 enable
		1	Output CLOCKOUT1 disable
0	CLOCKOUT0EN		CLOCKOUT0EN
		0	Output CLOCKOUT0 enable
		1	Output CLOCKOUT0 disable

9.12.20 Deep Sleep Mode Configuration (DEEPSLEEP) Register

The DEEPSLEEP register provides configuration for the Deep Sleep power-down mode and uses the GIO0 pin. For additional details on the DEEPSLEEP mode sequence, see Section 12.5.1. The DEEPSLEEP register is shown in Figure 88 and described in Table 98.

Figure 88. Deep Sleep Mode Configuration (DEEPSLEEP) Register

31	30	29 - 16
SLEEPENABLE	SLEEPCOMPLETE	Reserved
R/W-0	R-0	R-0

15 - 0
COUNTER
R/W-0x176B

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

Table 98. Deep Sleep Mode Configuration (DEEPSLEEP) Register Field Descriptions

Bit	Field	Value	Description
31	SLEEPENABLE		Enable Deep Sleep Mode Software must clear this bit to ‘0’ when the device is woken up from the deep sleep. NOTE: After wakeup, Deep Sleep Mode must be disabled to reset the SLEEPCOMPLETE bit.
		0	Disable Deep Sleep mode - normal operation
		1	Enable Deep Sleep Mode
30	SLEEPCOMPLETE		Status of the deep sleep complete mode Software polls this bit once the deep sleep process starts. When a '1' is read, software clears the SLEEPENABLE bit and continues operation. DEEPSLEEP mode sequence is described in detail in Section 12.5.1.
		0	Normal operation or still asleep
		1	Device is awake after Deep Sleep Mode
29-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15-0	COUNTER	0x176b	Wakeup Delay Counter Number of clock cycles to count prior to enabling clocks. This is to ensure oscillator is stable before enabling clocks.

9.12.21 De-bounce for GIO[n] Input (DEBOUNCE[n]) Register

The DEBOUNCE[n] (n is 0 to 7) is an array of eight registers that provide the controls for enabling and configuring Debounce for GIO[n] inputs.

Figure 89. De-bounce for GIO[n] Input (DEBOUNCE[n]) Register

31	30–21	20–16
ENABLE	Reserved	INTERVAL
R/W-0	R-0	R/W-0

15–0
INTERVAL
R/W-0

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

Table 99. De-bounce for GIO[n] Input (DEBOUNCE[n]) Register Field Descriptions

Bit	Field	Value	Description
31	ENABLE		Debounce Enable
		0	Debounce Enable
		1	Debounce Disable
30-21	Reserved	0	Any writes to these bit(s) must always have a value of 0.
20-0	INTERVAL	0	Interval count for the debounce circuit

9.12.22 VTP IO Control (VTPIOCR) Register

The VTP IO control register (VTPIOCR) is shown in Figure 90 and described in Table 100. Refer to the *TMS320DM36x Digital Media System-on-chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Users Guide* (SPRUFI2) for information on how to calibrate the DDR2/mDDR I/O using this register.

Figure 90. VTP IO Control (VTPIOCR) Register

31–24
Reserved
R-0

23–21	20	19	18	17–16
Reserved	DLLRSTZ	CLKRSTZ	VREFEN	VREFTAP
R-0	R/W-1	R/W-1	R/W-0	R/W-0

15	14	13	12	11	10	9	8
READY	IOPWRDN	CLRZ	FORCEDNP	FORCEDNN	FORCEUPP	FORCEUPN	PWRSAVE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
LOCK	PWRDN	D0	D1	D2	F0	F1	F2
R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1

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

Table 100. VTP IO Control (VTPIOCR) Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Any writes to these bit(s) must always have a value of 0.
20	DLLRSTZ	1	Active low reset is used to reset the DLL. This should code ‘1’ in normal operation.
19	CLKRSTZ	1	Active low reset is used to reset the clock divider of DDR2 address/data macro. This should code ‘1’ in normal operation.
18	VREFEN		Internal DDR2 IO Vref enable
		0	Connected to pad, external reference
		1	Connected to internal reference
17-16	VREFTAP		Selection for internal reference voltage level
		0	Vref = 50.0% of VDDS
		1	Vref = 47.5% of VDDS
		2	Vref = 52.5% of VDDS
		3	Vref = 50.0% of VDDS
15	READY		VTP Ready status
		0	VTP not ready
		1	VTP ready
14	IOPWRDN		Power down control enable for DDR2 input buffer
		0	Disable power down control by config_pwrdnen register
		1	Enable power down control by config_pwrdnen register
13	CLRZ	0	VTP clear. Write 0 to clear VTP flops.
12	FORCEDNP	0	Force decrease PFET drive
11	FORCEDNN	0	Force decrease NFET drive
10	FORCEUPP	0	Force increase PFET drive
9	FORCEUPN	0	Force increase PFET drive
8	PWRSAVE		VTP Power Save Mode
		0	Disable power save mode
		1	Enable power save mode

Table 100. VTP IO Control (VTPIOCR) Register Field Descriptions (continued)

Bit	Field	Value	Description
7	LOCK		VTP Impedance Lock
		0	Unlock impedance
		1	Lock impedance
6	PWRDN		VTP Power Down
		0	Disable power down
		1	Enable power down
5	D0	1	Drive strength control bit
4	D1	1	Drive strength control bit
3	D2	0	Drive strength control bit
2	F0	1	Digital filter control bit
1	F1	1	Digital filter control bit
0	F2	1	Digital filter control bit

9.12.23 Pullup/Down Control 0 (PUPDCTL0) Register

The pullup/down control 0 (PUPDCTL0) register is shown in Figure 91 and described in Table 101.

Figure 91. Pullup/Down Control 0 (PUPDCTL0) Register

31	30	29	28	27	26	25	24
GIO31	GIO30	GIO29	GIO28	GIO27	GIO26	GIO25	GIO24
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

23	22	21	20	19	18	17	16
GIO23	GIO22	GIO21	GIO20	GIO19	GIO18	GIO17	GIO16
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

15	14	13	12	11	10	9	8
GIO15	GIO14	GIO13	GIO12	GIO11	GIO10	GIO9	GIO8
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

7	6	5	4	3	2	1	0
GIO7	GIO6	GIO5	GIO4	GIO3	GIO2	GIO1	GIO0
R/W-1	R/W-1	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

Table 101. Pullup/Down Control 0 (PUPDCTL0) Register Field Descriptions

Bit	Field	Value	Description
31	GIO31		GIO31 Pull down enable
		0	Disable
		1	Enable
30	GIO30		GIO30 Pull down enable
		0	Disable
		1	Enable
29	GIO29		GIO29 Pull down enable
		0	Disable
		1	Enable
28	GIO28		GIO28 Pull down enable
		0	Disable
		1	Enable

Table 101. Pullup/Down Control 0 (PUPDCTL0) Register Field Descriptions (continued)

Bit	Field	Value	Description
27	GIO27		GIO27 Pull down enable
		0	Disable
		1	Enable
26	GIO26		GIO26 Pull down enable
		0	Disable
		1	Enable
25	GIO25		GIO25 Pull down enable
		0	Disable
		1	Enable
24	GIO24		GIO24 Pull down enable
		0	Disable
		1	Enable
23	GIO23		GIO23 Pull down enable
		0	Disable
		1	Enable
22	GIO22		GIO22 Pull down enable
		0	Disable
		1	Enable
21	GIO21		GIO21 Pull down enable
		0	Disable
		1	Enable
20	GIO20		GIO20 Pull down enable
		0	Disable
		1	Enable
19	GIO19		GIO19 Pull down enable
		0	Disable
		1	Enable
18	GIO18		GIO18 Pull down enable
		0	Disable
		1	Enable
17	GIO17		GIO17 Pull down enable
		0	Disable
		1	Enable
16	GIO16		GIO16 Pull down enable
		0	Disable
		1	Enable
15	GIO15		GIO15 Pull down enable
		0	Disable
		1	Enable
14	GIO14		GIO14 Pull down enable
		0	Disable
		1	Enable
13	GIO13		GIO13 Pull down enable
		0	Disable
		1	Enable

Table 101. Pullup/Down Control 0 (PUPDCTL0) Register Field Descriptions (continued)

Bit	Field	Value	Description
12	GIO12		GIO12 Pull down enable
		0	Disable
		1	Enable
11	GIO11		GIO11 Pull down enable
		0	Disable
		1	Enable
10	GIO10		GIO10 Pull down enable
		0	Disable
		1	Enable
9	GIO9		GIO9 Pull down enable
		0	Disable
		1	Enable
8	GIO8		GIO8 Pull down enable
		0	Disable
		1	Enable
7	GIO7		GIO7 Pull down enable
		0	Disable
		1	Enable
6	GIO6		GIO6 Pull down enable
		0	Disable
		1	Enable
5	GIO5		GIO5 Pull down enable
		0	Disable
		1	Enable
4	GIO4		GIO4 Pull down enable
		0	Disable
		1	Enable
3	GIO3		GIO3 Pull down enable
		0	Disable
		1	Enable
2	GIO2		GIO2 Pull down enable
		0	Disable
		1	Enable
1	GIO1		GIO1 Pull down enable
		0	Disable
		1	Enable
0	GIO0		GIO0 Pull down enable
		0	Disable
		1	Enable

9.12.24 Pullup/Down Control 1 (PUPDCTL1) Register

The pullup/down control 1 (PUPDCTL1) register is shown in Figure 92 and described in Table 102 .

Figure 92. Pullup/Down Control 1 (PUPDCTL1) Register

31	30	29	28	27	26	25	24
YIN	CIN	GIO95	GIO94	GIO93	GIO81	GIO80	GIO78
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-0
23	22	21	20	19	18	17	16
GIO77	GIO76	GIO75	GIO74	GIO73	GIO52	GIO49	GIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
GIO47	GIO46	GIO45	GIO44	GIO43	GIO42	GIO41	GIO40
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
GIO39	GIO38	GIO37	GIO36	GIO35	GIO34	GIO33	GIO32
R/W-1	R/W-1	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

Table 102. Pullup/Down Control 1 (PUPDCTL1) Register Field Descriptions

Bit	Field	Value	Description
31	YIN		YIN[7:0] Pull down enable
		0	Disable
		1	Enable
30	CIN		CIN[7:0] and PCLK Pull down enable
		0	Disable
		1	Enable
29	GIO95		GIO95 Pull down enable
		0	Disable
		1	Enable
28	GIO94		GIO94 Pull down enable
		0	Disable
		1	Enable
27	GIO93		GIO93 Pull down enable
		0	Disable
		1	Enable
26	GIO81		GIO81Pull down enable
		0	Disable
		1	Enable
25	GIO80		GIO80 Pull down enable
		0	Disable
		1	Enable
24	GIO78		GIO78 Pull down enable
		0	Disable
		1	Enable
23	GIO77		GIO77 Pull down enable
		0	Disable
		1	Enable
22	GIO76		GIO76 Pull down enable
		0	Disable
		1	Enable

Table 102. Pullup/Down Control 1 (PUPDCTL1) Register Field Descriptions (continued)

Bit	Field	Value	Description
21	GIO75		GIO75 Pull down enable
		0	Disable
		1	Enable
20	GIO74		GIO74 Pull down enable
		0	Disable
		1	Enable
19	GIO73		GIO73 Pull down enable
		0	Disable
		1	Enable
18	GIO52		GIO52 Pull down enable
		0	Disable
		1	Enable
17	GIO49		GIO49 Pull down enable
		0	Disable
		1	Enable
16	GIO48		GIO48 Pull down enable
		0	Disable
		1	Enable
15	GIO47		GIO47 Pull down enable
		0	Disable
		1	Enable
14	GIO46		GIO46 Pull down enable
		0	Disable
		1	Enable
13	GIO45		GIO45 Pull down enable
		0	Disable
		1	Enable
12	GIO44		GIO44 Pull down enable
		0	Disable
		1	Enable
11	GIO43		GIO43 Pull down enable
		0	Disable
		1	Enable
10	GIO42		GIO42 Pull down enable
		0	Disable
		1	Enable
9	GIO41		GIO41 Pull down enable
		0	Disable
		1	Enable
8	GIO40		GIO40 Pull down enable
		0	Disable
		1	Enable
7	GIO39		GIO39 Pull down enable
		0	Disable
		1	Enable

Table 102. Pullup/Down Control 1 (PUPDCTL1) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	GIO38		GIO38 Pull down enable
		0	Disable
		1	Enable
5	GIO37		GIO37 Pull down enable
		0	Disable
		1	Enable
4	GIO36		GIO36 Pull down enable
		0	Disable
		1	Enable
3	GIO35		GIO35 Pull down enable
		0	Disable
		1	Enable
2	GIO34		GIO34 Pull down enable
		0	Disable
		1	Enable
1	GIO33		GIO33 Pull down enable
		0	Disable
		1	Enable
0	GIO32		GIO32 Pull down enable
		0	Disable
		1	Enable

9.12.25 HDVICP Boot Register

The HDVICP boot register is shown in Figure 93 and described in Table 103.

Figure 93. HDVICP Boot Register

31–16
Reserved
R-0

15–5	4	3–1	0
Reserved	INITRAM	Reserved	COPHLT
R-0	R/W-0	R-0	R/W-1

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

Table 103. HDVICP Boot Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4	INITRAM		HDVICP INITRAM enable
		0	Internal TCM disable
		1	Internal TCM enable
3-1	Reserved	0	Any writes to these bit(s) must always have a value of 0.
0	COPHLT		HDVICP Fetch Halt
		0	CFG master port enable
		1	CFG master port stalled

9.12.26 PLLC1 Configuration (PLLC1_CONFIG) Register

The PLLC1 configuration (PLLC1_CONFIG) register is shown in Figure 94 and described in Table 104.

Figure 94. PLLC1 Configuration (PLLC1_CONFIG) Register

31–28	27–25	24
Reserved	LOCK1,2,3	Reserved
R-0	R-0	R-0

23–0
Reserved
R-0

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

Table 104. PLLC1 Configuration (PLLC1_CONFIG) Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Any writes to these bit(s) must always have a value of 0.
27-25	LOCK1,2,3		PLL in lock mode condition
		0	Not locked
		111b	Locked
24-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

9.12.27 PLLC2 Configuration (PLLC2_CONFIG) Register

The PLLC2 configuration (PLLC2_CONFIG) register is shown in Figure 95 and described in Table 105.

Figure 95. PLLC2 Configuration (PLLC2_CONFIG) Register

31–28	27–25	24
Reserved	LOCK1,2,3	Reserved
R-0	R-0	R-0

23–0
Reserved
R-0

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

Table 105. PLLC2 Configuration (PLLC2_CONFIG) Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Any writes to these bit(s) must always have a value of 0.
27-25	LOCK1,2,3		PLL in lock mode condition
		0	Not locked
		111b	Locked
24-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

10 Reset

10.1 Reset Overview

DM36x has two types of reset domains: the PRTC subsystem(PRTCSS) and the DM36x device. These resets differ by how they are initiated and/or by their effect on the device. Each type is briefly described in Table 106 and further described in the following sections.

10.1.1 PRTCSS Reset

PRTCSS has one power-on-reset (POR).

10.1.2 DM36x Device Reset

The various types of hardware, software, and pin-initiated resets are listed here.

Table 106. Reset Types

Type	Initiator	Effect
POR (Power-On-Reset)	RESETN pin low and TRSTN low	Total reset of the chip (cold reset). Resets all modules including memory and emulation.
Warm Reset	RESETN pin low and TRSTN high (initiated by ARM emulator).	Resets all modules including memory, except ARM emulation.
Max Reset	ARM emulator or Watchdog Timer (WDT).	Same effect as warm reset.
System Reset	ARM emulator	Resets all modules except memory and ARM emulation. It is a soft reset that maintains memory contents and does not affect or reset clocks or power states.
Module Reset	ARM software	Resets a specific module. Allows the ARM software to independently reset a module. Module reset is intended as a debug tool not as a tool to use in production.

10.2 Reset Pins

Power-on-reset (POR) and warm reset are initiated by the RESETN and TRSTN pins. The RESETN and TRSTN pins are briefly described in Table 107.

For more information, see the *TMS320DM365 Digital Media System-on-Chip Data Manual* (SPRS457).

Table 107. Reset Pins

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

10.3 Types of Reset

10.3.1 Power-On Reset (POR)

POR totally resets the chip, including all modules, memories, and emulation circuitry. The following steps describe the POR sequence:

1. Apply power and clocks to the chip and drive TRSTN and RESETN low to initiate POR.
2. Drive RESETN high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 10.3.6.1 for more information.
4. Hardware resets all of the modules, including memory and emulation circuitry.
5. POR finishes, all modules are now in their default configurations, and hardware begins the boot process.

See the *TMS320DM365 Digital Media System-on-Chip Data Manual* (SPRS457) for power sequencing and reset timing requirements.

10.3.2 Warm Reset

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

1. Emulator drives TRSTN high and RESETN low to initiate warm reset.
2. Emulator drives RESETN high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 10.3.6.1 below for more information.
4. Hardware resets all of the modules including memories, but not ARM emulation circuitry.
5. Warm reset finishes, all modules except ARM emulation are in their default configurations, and hardware begins the boot process.

See the *TMS320DM365 Digital Media System-on-Chip Data Manual* (SPRS457) for reset timing requirements.

10.3.3 Max Reset

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

The following steps describe the max reset sequence:

1. To initiate max reset, the WDT expires (indicating a runaway condition) or the ARM emulator initiates a max reset command via the IcePick emulation module.
2. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 10.3.6.1 for more information.
3. Hardware resets all modules including memories, but not ARM emulation circuitry.
4. Warm reset finishes, all modules except ARM emulation are in their default configurations, and hardware begins the boot process.

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

See the *TMS320DM36x Digital Media System-on-chip (DMSoC) Timer/Watchdog Timer User's Guide* (SPRUFH0) for information on the WDT. See Section 3 for information on IcePick emulation.

10.3.4 System Reset

The emulator initiates system reset via the ICECrusher emulation module. It is considered a soft reset (i.e., memory is not reset). None of the following modules are reset: DDR2 EMIF, PLL Controller (PLLC), Power and Sleep Controller (PSC), and emulation.

The following steps describe the system reset sequence:

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

10.3.5 Module Reset

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

The procedures for asserting and de-asserting module reset are fully described in Section 7.

10.3.6 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, and AEMIF.

Default configuration is the configuration immediately after POR, warm reset, and max reset and just before the boot process begins. The boot ROM updates the configuration. See Section 11 for more information on the boot process.

10.3.6.1 Device Configuration Pins

The device configuration pins are described in Table 108. The device configuration pins are latched at reset and allow you to configure the following options at reset:

- ARM Boot Mode
- Asynchronous EMIF pin configuration

These pins are described further in the following sections.

NOTE: The device configuration pins are multiplexed with AEMIF pins. After the device configuration pins are sampled at reset, they automatically change to function as AEMIF pins. Pin multiplexing is described in Section 9.

Table 108. Device Configuration

Device Configuration Input	Function	Sampled Pin	Default Setting (by internal pull-up/ pull-down)
BTSEL[2:0]	Selects ARM boot mode 000 = Boot from ROM (NAND) 001 = Boot from AEMIF 010 = Boot from ROM (MMC/SD) 011 = Boot from ROM (UART) 100 = Boot from ROM (USB) 101 = Boot from ROM (SPI) 110 = Boot from ROM (EMAC) 111 = Boot from ROM (HPI)	EM_A[13:11]	000 (Boot from ROM - NAND)

AECFG[2:0]	AEMIF Configuration[1]
	AECFG[2] = '0' for 8-bit AEMIF configuration
	AECFG[2] = '1' for 16-bit AEMIF configuration
		EM_A[10:8]	000
			(8-bit NAND)

OSCCFG	Oscillator Configuration
	OSCCFG = '0' for mode #1
	OSCCFG = '1' for mode #2
		GIO81	0
			(Mode #1)

[1] Other supported AECFG[2:0] combinations can be found in Table 85.

10.3.6.2 PLL 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 MXI1 (typically 24 MHz) drives the chip after reset. For more information, see Section 5 and Section 6. The default state of the PLLs is reflected by the default state of the register bits in the PLLC registers.

10.3.6.3 Module Configuration

Only a subset of modules are enabled after reset by default. Table 39 in Section 7 shows which modules are enabled after reset. Furthermore, as shown in Table 39, the following modules are enabled depending on the sampled state of the device configuration pins: EDMA (CC and TC0), AEMIF, MMC/SD0, UART0, and Timer0. For example, UART0 is enabled after reset when the device configuration pins (BTSEL[2:0] = 011 - Enable UART) select UART boot mode.

10.3.6.4 ARM Boot Mode Configuration

The input pins BTSEL[2:0] determine whether the ARM will boot from its ROM or from the Asynchronous EMIF (AEMIF). When ROM boot is selected, a jump to the start of internal ROM (address 0x0000: 8000) is forced into the first fetched instruction word. The embedded ROM bootloader code (RBL) then performs certain configuration steps, reads the BOOTCFG register to determine the desired boot method, and branches to the appropriate boot routine (i.e., EMAC, HPI, SPI, USB, NAND, MMC/SD, or UART loader routine) .

If AEMIF boot is selected (BTSEL[2:0] = 001), a jump to the start of AEMIF (address 0x0200: 0000) is forced into the first fetched instruction word. The ARM then continues executing from external asynchronous memory using the default AEMIF timings until modified by software.

NOTE: For AEMIF boot, OneNAND/NOR must be connected to the first AEMIF chip select space ($\overline{\text{EM\_CE0}}$). The AEMIF does not support direct execution from NAND Flash.

Boot modes are further described in Section 11.

10.3.6.5 AEMIF Configuration

This section discusses the pin and timing configurations for the AEMIF. For more information on the AEMIF, see the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (AEMIF) User's Guide* (SPRUFI1).

10.3.6.5.1 AEMIF Pin Configuration

The AECFG[2:0] input pins determine the AEMIF configuration immediately after reset. Use AECFG[2:0] to properly configure the AEMIF pins . Refer to the section on pin multiplexing in Section 9 and the device data manual for additional information on AEMIF pin configuration.

10.3.6.5.2 AEMIF Timing Configuration

When AEMIF is enabled, the wait state registers are reset to the slowest possible configuration, which is 88 cycles per access (16 cycles of setup, 64 cycles of strobe, and 8 cycles of hold). The AEMIF operates in the half-rate mode to meet frequency requirements. Thus, with a 24 MHz clock at MXI/MXO, the AEMIF is configured to run at (12 MHz)/(88) which equals approximately 136.36 kHz.

11 Boot Modes

11.1 Boot Modes Overview

The DM36x ARM can boot from either Async EMIF (OneNand/NOR) or from ARM ROM, as determined by the setting of the device configuration pins BTSEL[2:0]. The boot selection pins (BTSEL[2:0]) determine the ARM boot process. These ROM boot modes are described in more detail in the following sections.

After reset (POR, warm reset, or max reset), ARM program execution begins in ARM ROM at 0x0000: 8000, except when BTSEL[2:0] = 001, indicating AEMIF (OneNand/NOR) flash boot. See Section 10 for information on the boot selection pins.

11.1.1 Features

The ARM ROM bootloader (RBL) executes when the BOOTSEL[2:0] pins indicate a condition other than the normal ARM EMIF boot.

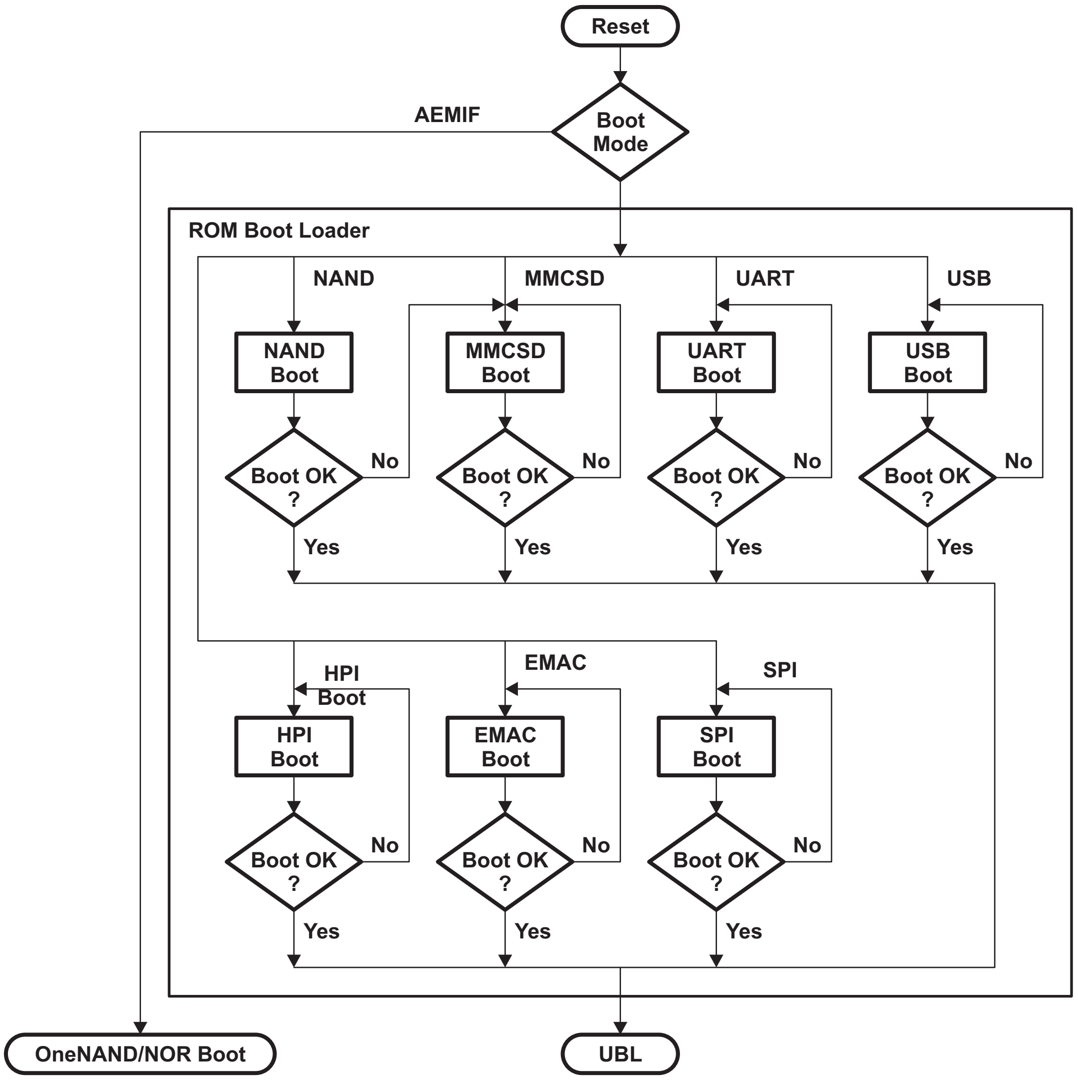
- If BTSEL[2:0] = 001 - Asynchronous EMIF (AEMIF boot). This mode is handled by hardware control and does not involve the ROM. In the case of OneNAND, you are responsible for putting any necessary boot code in the OneNAND's boot page. This code shall configure the AEMIF module for the OneNAND device. After the AEMIF module is configured, booting will continue immediately after the OneNAND’s boot page with the AEMIF module managing pages thereafter.
- The RBL supports seven distinct boot modes:
  - BTSEL[2:0] = 000 - ARM NAND Boot
  - BTSEL[2:0] = 010 - ARM MMC/SD Boot
  - BTSEL[2:0] = 011 - ARM UART Boot
  - BTSEL[2:0] = 100 - ARM USB Boot
  - BTSEL[2:0] = 101- ARM SPI Boot
  - BTSEL[2:0] = 110 - ARM EMAC Boot
  - BTSEL[2:0] = 111 - ARM HPI Boot
- If NAND boot fails, then MMC/SD mode is tried.
- If MMC/SD boot fails, then MMC/SD boot is tried again.
- If UART boot fails, then UART boot is tried again.
- If USB boot fails, then USB boot is tried again
- If SPI boot fails, then SPI boot is tried again
- If EMAC boot fails, then EMAC boot is tried again
- If HPI boot fails, then HPI boot is tried again
- RBL shall update boot status (PASS/FAIL) in MISC register bits 8 and 9 in the system control module
- ARM ROM Boot - NAND Mode
  - No support for a full firmware boot. Instead, copies a second stage user-bootloader (UBL) from NAND flash to ARM internal RAM (AIM) and transfers control to the user-defined UBL.
  - Support for NAND with page sizes up to 4096 bytes
  - Support for magic number error detection and retry (up to 24 times) when loading UBL
  - Support for up to 30KB UBL (32KB IRAM - ~2KB for RBL stack)
  - Optional, user-selectable, support for use of DMA and I-cache during RBL execution (i.e.,while loading UBL)
  - Supports booting from 8-bit NAND devices (16-bit NAND devices are not supported)
  - Supports 4-bit ECC (1-bit ECC is not supported)
  - Supports NAND flash that requires chip select to stay low during the tR read time
- ARM ROM Boot - MMC/SD Mode
  - No support for a full firmware boot. Instead, copies a second stage user bootloader (UBL) from MMC/SD to ARM Internal RAM (AIM) and transfers control to your software.
  - Support for MMC/SD native protocol (MMC/SD SPI protocol is not supported)
  - Support for descriptor error detection and retry (up to 24 times) when loading UBL
  - Support for up to 30KB UBL (32KB - ~2KB for RBL stack)

- ARM ROM Boot - UART mode
  - No support for a full firmware boot. Instead, loads a second stage user bootloader (UBL) via UART to ARM internal RAM (AIM) and transfers control to your software.
  - Support for up to 30KB UBL (32KB - ~2KB for RBL stack)
  - If the state of BTSEL[2:0] pins at reset is 011, then the UART boot mode executes. This mode enables a small program, referred to here as a user bootloader (UBL), to be downloaded to the on-chip ARM internal RAM via the on-chip serial UART and executed. A host program, referred to as serial host utility program, manages the interaction with RBL and provides a means for operator feedback and input. The UART boot mode execution assumes the following UART settings: Time-Out 500 ms, one-shot Serial RS-232 port 115.2 Kbps, 8-bit, no parity, one stop bit Command, data, and checksum format. Everything sent from the host to the DM365 UART RBL must be in ASCII format.
- ARM ROM Boot – USB Mode
  - No support for a full firmware boot. Instead, loads a second stage user bootloader (UBL) via USB to ARM Internal RAM (AIM) and transfers control to the your software.
- ARM ROM Boot – SPI Mode
  - Device will copy UBL to ARM Internal RAM (AIM) via SPI interface from a SPI peripheral like SPI EEPROM. RBL will then transfer control to the UBL.
- ARM ROM Boot – EMAC Mode
  - Device will send a boot request packet and the host/server will respond with the boot packets. RBL will wait for all boot packets to arrive and then transfer control to the UBL which is received via boot packets. In EMAC boot mode, an I2C EEPROM or SPI EEPROM is necessary for programming the EMAC descriptor (including the EMAC address for the device). If a magic number is not found in the EEPROM, then the EMAC boot mode will use a default MAC address. In this case there will be no magic number support.
- ARM ROM Boot– HPI Mode
  - The Host will copy UBL to ARM Internal RAM (AIM) via HPI interface and notify the ROM bootloader after Copy is finished. RBL will then transfer control to the UBL.

11.1.2 Functional Block Diagram

The general boot sequence is shown in Figure 96.

Figure 96. Boot Mode Functional Block Diagram



11.2 ARM ROM Boot Modes

The ARM ROM bootloader (RBL) executes when the BOOTSEL[2:0] pins indicate a condition other than the normal ARM EMIF boot (BTSEL[2:0] ≠ 001). In this case, control is passed to the ROM bootloader (RBL). The RBL then executes the proper mode after reading the state of the BTSEL[2:0] pins from the BOOTCFG register.

11.2.1 NAND Boot Mode

If the value in BTSEL[2:0] from the BOOTCFG register is 000, the NAND mode executes. The outline of operations followed in the NAND mode is described in Figure 97. The NAND boot mode assumes the NAND is located on the EM_CE0 interface, whose bus configuration is configured by the AECFG[2:0] pins. The AECFG[2:0] pins must be configured such that the proper EMIF signals are available for the NAND device.

First, the device ID of the NAND device is read from the device, and then any necessary information (such as the block and page sizes, etc.) are obtained from the device information table in the RBL. The device information in the RBL is based on the list of supported NAND devices. Next, the RBL searches for the UBL descriptor in page 0 of the block after the CIS/IDI block (block 1).

If a valid UBL is not found here, as determined by reading a valid UBL magic number, the next block is searched. Searching continues for up to 24 blocks. This provision for additional searching is made in case the first few consecutive blocks have been marked as bad (i.e., they have errors). Searching 24 blocks is sufficient to handle the errors found in virtually all NAND devices.

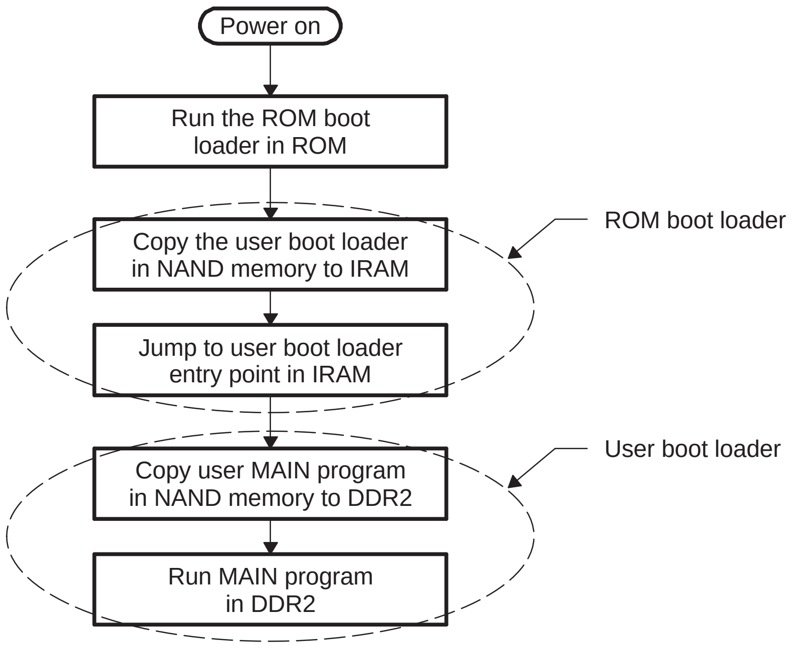
When a valid UBL signature is found, the corresponding block number (from 1 to 24) is written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). This feature is provided as a basic debug mechanism. By reading these 32 bits of memory, via JTAG for example, you can determine in which block the RBL found a valid UBL signature. If no valid UBL signature is found after searching 24 blocks, the RBL will try to boot via MMC/SD.

If a valid UBL is found, the UBL descriptor is read and processed. The descriptor gives the information required for loading and control transfer to the UBL. The UBL is then read and processed. The RBL may enable any combination of faster EMIF and I-Cache operations based on information in the UBL descriptor first. Additionally, the descriptor provides information on whether or not DMA should be used during UBL copying. Once the user-specified start-up conditions are set, the RBL copies the UBL into ARM internal RAM, starting at address 0x0000: 0020.

NOTE: The actual copying is performed on the lower 30KB of the TCM data area: 0x10020 - 0x1781F.

The NAND RBL uses the 4-bit ECC hardware to determine if a read error occurs while reading the UBL into ARM IRAM. If a 4-bit ECC read error is detected, the UBL will correct the error via the ECC correction algorithm. If the read fails for any other reason, the copy will immediately halt for that instance on the magic number. Then the RBL will continue to search the block following that in which the magic number was found for another instance of a magic number. When another magic number is found, the process is repeated. Using this retry process, the magic number and UBL can be duplicated up to 24 times, giving significant redundancy and error resilience to NAND read errors.

Figure 97. NAND Boot Flow


Power on
Run the ROM boot loader in ROM
Copy the user boot loader in NAND memory to IRAM
Jump to user boot loader entry point in IRAM
Copy user MAIN program in NAND memory to DDR2
Run MAIN program in DDR2
ROM boot loader
User boot loader

The NAND user bootloader UBL descriptor format is described in Table 109.

Table 109. NAND UBL Descriptor

Page 0 Address	32-Bits	Description
0	0xA1AC EDxx	Magic number (0xA1ACEDxx)
4	Entry Point Address of UBL	Entry point address for the user bootloader (absolute address)
8	Number of pages in UBL	Number of pages (size of user bootloader in number of pages)
12	Starting Block # of UBL	Block number where user bootloader is present
16	Starting Page # of UBL	Page number where user bootloader is present
20	PLL settings -M	PLL setting -Multiplier (only valid is Magic Number indicates PLL enable)
24	PLL settings -N	PLL setting -Divider (only valid is Magic Number indicates PLL enable)
28	Fast EMIF setting	Fast EMIF settings(only valid is Magic Number indicates fast EMIF boot)

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

Different NAND boot mode options can set different MAGIC IDs in the UBL descriptor. Table 110 lists the UBL signatures.

Table 110. UBL Signatures and Special Modes

Mode	Value	Description
UBL_MAGIC_SAFE	0x A1AC ED00	Safe boot mode
UBL_MAGIC_DMA	0x A1AC ED11	DMA boot mode
UBL_MAGIC_IC	0x A1AC ED22	Instruction Cache boot mode
UBL_MAGIC_FAST	0x A1AC ED33	Fast EMIF boot mode
UBL_MAGIC_DMA_IC	0x A1AC ED44	DMA +Instruction Cache boot mode
UBL_MAGIC_DMA_IC_FAST	0x A1AC ED55	DMA +Instruction Cache+ Fast EMIF boot mode
UBL_MAGIC_PLL	0x A1AC ED66	With PLL enabled to have higher ARM/DMA clocks
UBL_MAGIC_PLL_DMA	0x A1AC ED77	With PLL enabled +DMA
UBL_MAGIC_PLL_IC	0x A1AC ED88	With PLL enabled +Instruction Cache
UBL_MAGIC_PLL_FAST	0x A1AC ED99	With PLL enabled +Fast EMIF
UBL_MAGIC_PLL_DMA_IC	0x A1AC EDAA	With PLL enabled +DMA+Instruction Cache
UBL_MAGIC_PLL_DMA_IC_FAST	0x A1AC EDBB	With PLL enabled +DMA+Instruction Cache+ Fast EMIF
UBL_MAGIC_SAFE_LEGACY	0xA1ACEDCC	Safe boot mode with legacy

When NAND boot mode is not used with the PLL option, the safe boot mode option wherein PLL is in bypass mode, and DMA, iCache, and fast AEMIF are not enabled.

When NAND boot mode is used with PLL option, ARM frequency is stepped up based on multiplier and pre-divider values supplied in the UBL descriptor.

Fast AEMIF setting is a part of NAND UBL descriptor which is used in only a few of the NAND boot mode options (having the term FAST). It is a 32-bit value which sets Async1 Config Register (A1CR) of AEMIF. See the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (AEMIF) Users Guide* (SPRUF11) for more information on the AEMIF. The register is otherwise programmed as 0x3FFFFFFC in other NAND boot options. The values to be used for Fast AEMIF setting depend upon AEMIF frequency, as shown in Table 111 and Table 112.

Table 111. NAND BOOT MODE and SETTING

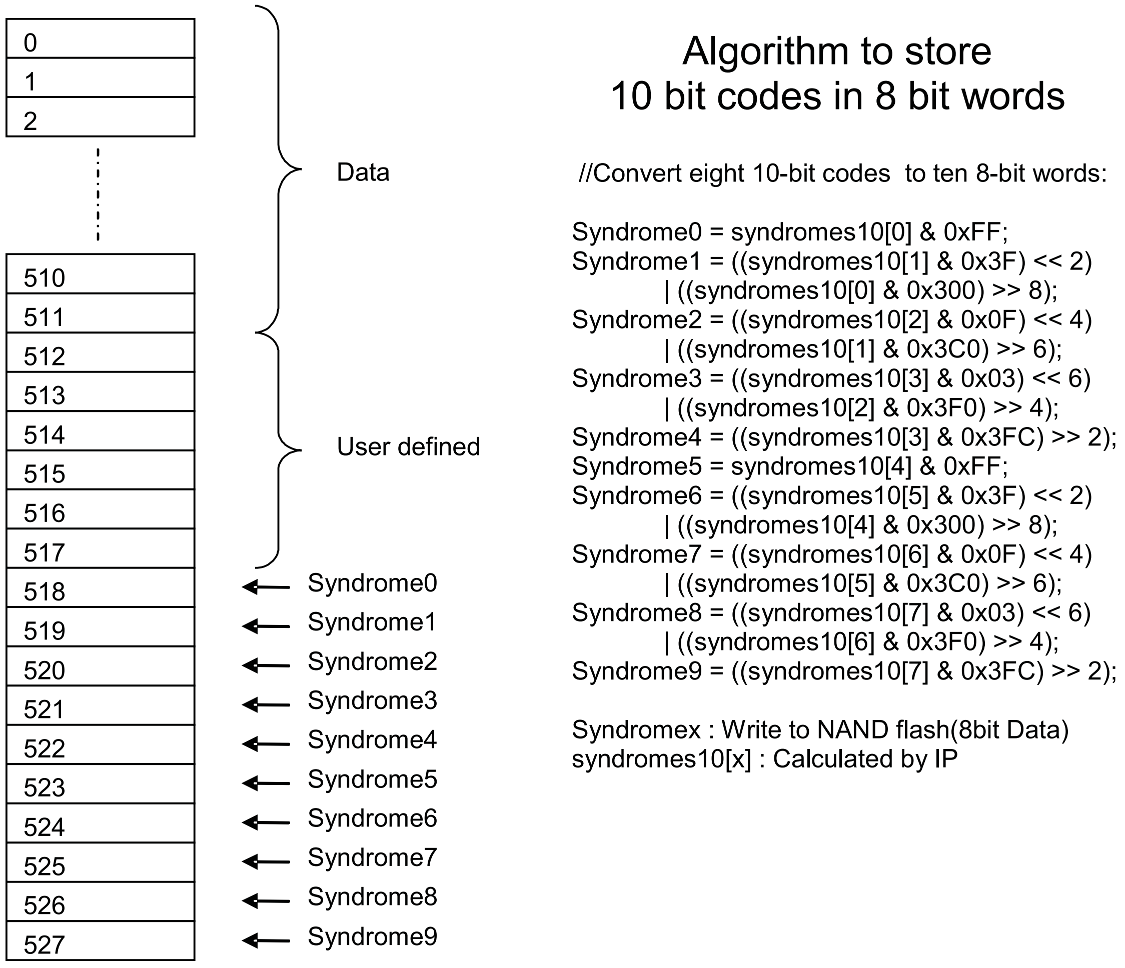
Mode Name	I cache	EDMA	AEMIF ACCESS	PLL	AEMIF CLOCK
UBL_MAGIC_SAFE	X	X	MAX	BYPASS	(Oscillator clock)/2
UBL_MAGIC_DMA	X	**	MAX	BYPASS	(Oscillator clock)/2
UBL_MAGIC_IC	**	X	MAX	BYPASS	(Oscillator clock)/2
UBL_MAGIC_FAST	X	X	MIN	BYPASS	(Oscillator clock)/2
UBL_MAGIC_DMA_IC	**	**	MAX	BYPASS	(Oscillator clock)/2
UBL_MAGIC_DMA_IC_FAST	**	**	MIN	BYPASS	(Oscillator clock)/2
UBL_MAGIC_PLL	X	X	MAX	ENABLED	(PLLC1SYCLK4)/2
UBL_MAGIC_PLL_DMA	X	**	MAX	ENABLED	(PLLC1SYCLK4)/2
UBL_MAGIC_PLL_IC	**	X	MAX	ENABLED	(PLLC1SYCLK4)/2
UBL_MAGIC_PLL_FAST	X	X	MIN	ENABLED	(PLLC1SYCLK4)/2
UBL_MAGIC_PLL_DMA_IC	**	**	MAX	ENABLED	(PLLC1SYCLK4)/2
UBL_MAGIC_PLL_DMA_IC_FAST	**	**	MIN	ENABLED	(PLLC1SYCLK4)/2
X= Disable; **= Enable					

Table 112. AEMIF ACCESS Timing (A1CR register setting)

AEMIF ACCESS	W_SETUP / R_SETUP	W_STROBE / R_STROBE	W_HOLD / R_HOLD
MAX	16 cycles	64 cycles	8 cycles
MIN	[1]	[1]	[1]

[1] Depends on NAND device to be used for MIN values.

Figure 98. 4-Bit ECC Format and Bit 10 to 8-Bit Compression Algorithm

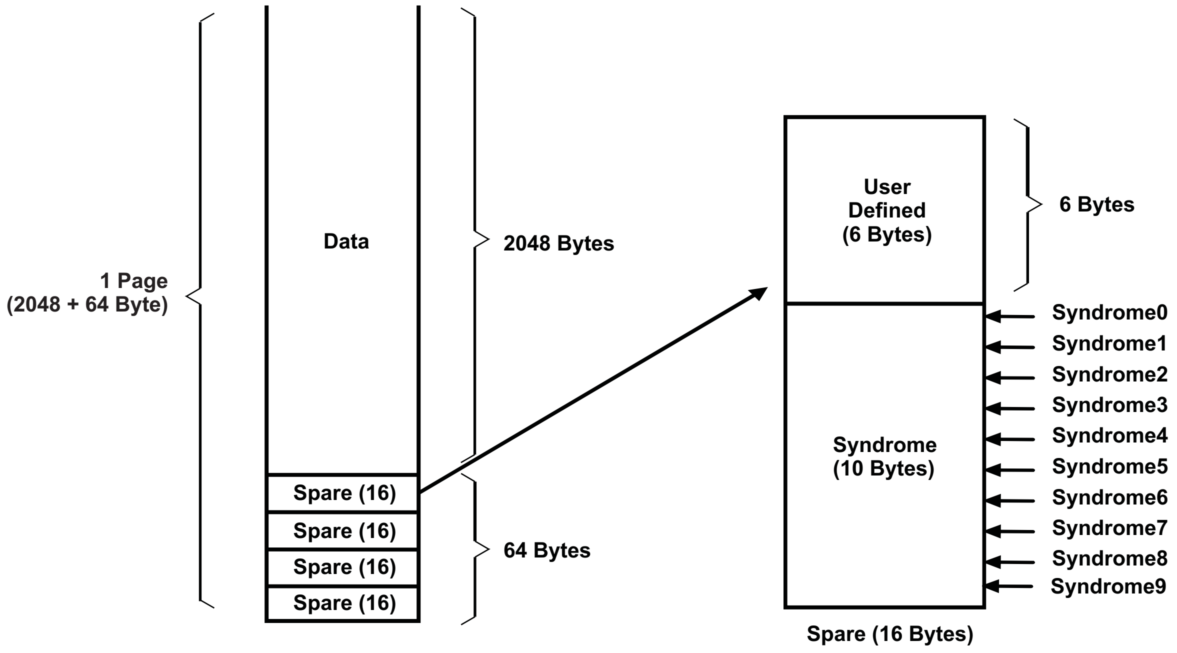


```
//Convert eight 10-bit codes  to ten 8-bit words:

Syndrome0 = syndromes10[0] & 0xFF;
Syndrome1 = ((syndromes10[1] & 0x3F) << 2)
            | ((syndromes10[0] & 0x300) >> 8);
Syndrome2 = ((syndromes10[2] & 0x0F) << 4)
            | ((syndromes10[1] & 0x3C0) >> 6);
Syndrome3 = ((syndromes10[3] & 0x03) << 6)
            | ((syndromes10[2] & 0x3F0) >> 4);
Syndrome4 = ((syndromes10[3] & 0x3FC) >> 2);
Syndrome5 = syndromes10[4] & 0xFF;
Syndrome6 = ((syndromes10[5] & 0x3F) << 2)
            | ((syndromes10[4] & 0x300) >> 8);
Syndrome7 = ((syndromes10[6] & 0x0F) << 4)
            | ((syndromes10[5] & 0x3C0) >> 6);
Syndrome8 = ((syndromes10[7] & 0x03) << 6)
            | ((syndromes10[6] & 0x3F0) >> 4);
Syndrome9 = ((syndromes10[7] & 0x3FC) >> 2);

Syndromex : Write to NAND flash(8bit Data)
syndromes10[x] : Calculated by IP
```

Figure 99. 4-Bit ECC Format for 2048+64 Byte Page Size



11.2.1.1 NAND Boot Detailed Flow

An overview of the NAND boot process is shown in the flow chart in Figure 100 and exemplified in Figure 101. The following steps describe the NAND boot process:

- Initialize the stack in the upper ~2K of RAM1 (RAM1 ' 0x7800-0x7FFF). Do not use the last 32-bits of IRAM (0x7ffc-0x8000) for stack, because these will be written with a valid block number.
- Disable all interrupts, IRQ and FIQ
- The external pin DEEPSLEEPZ/GIO0 must be driven high during chip reset in order for NAND boot mode to work.
- Read the device Id of NAND and get the parameters for NAND from a table in ROM.
- Initialize the NAND region according to the parameters for the NAND flash see the Table 113
- Search for the user bootloader magic number in the blocks after CIS/IDI page (CIS/IDI is generally block 0, page 0). See Figure 102. The magic number is detected based on reading 0xA1ACEDxx in the first 32-bits of page 0 in a block. Only Page 0 of blocks 1 to 24 will be read and searched for the magic number. The magic number for all blocks will be read to ascertain that the block is not an invalid block. For debug purposes, when a valid UBL magic number is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). The UBL Descriptor provides the necessary details of the user bootloader. See Table 109 and Table 110 for details of the UBL Descriptor.
- The UBL Descriptor consists of the following parameters (all UBL parameters are 32-bits wide):
  - Entry Point Address: absolute entry point AFTER loading UBL
    - Must be in range 0x0020 - 0x781C
  - Number of NAND pages in UBL:
    - Must be contiguous pages
    - May span multiple blocks
    - Total bytes must be less than or equal to 30KByte total (size of IRAM - ~2KB stack space)
  - Starting Block of UBL:
    - May be the same block as UBL descriptor
  - Starting Page of UBL
    - May not be the same page as UBL descriptor since full pages must be loaded
- Copy the user bootloader from NAND flash to IRAM with hardware ECC error detection enabled. If a 4-bit ECC read error is detected, the UBL will correct the error via the ECC correction algorithm. If the read fails due to any other error, the descriptor search process begins anew in the next block after that in which the UBL descriptor was found, for up to the first 24 blocks. If no valid UBL descriptor is found after searching 24 blocks, the RBL will try to boot via MMC/SD.
- Give control to user bootloader at UBL entry address.
- Safe boot mode for NAND boot mode is done in PLL bypass mode and it does not use fast EMIF, DMA or I-Cache. In other modes a combination of the above settings are used. For example, in UBL_MAGIC_PLL_DMA_IC_FAST mode all the four settings are activated and it is supposed to be the fastest mode of NAND boot.

Figure 100. NAND Boot Mode Flow Chart

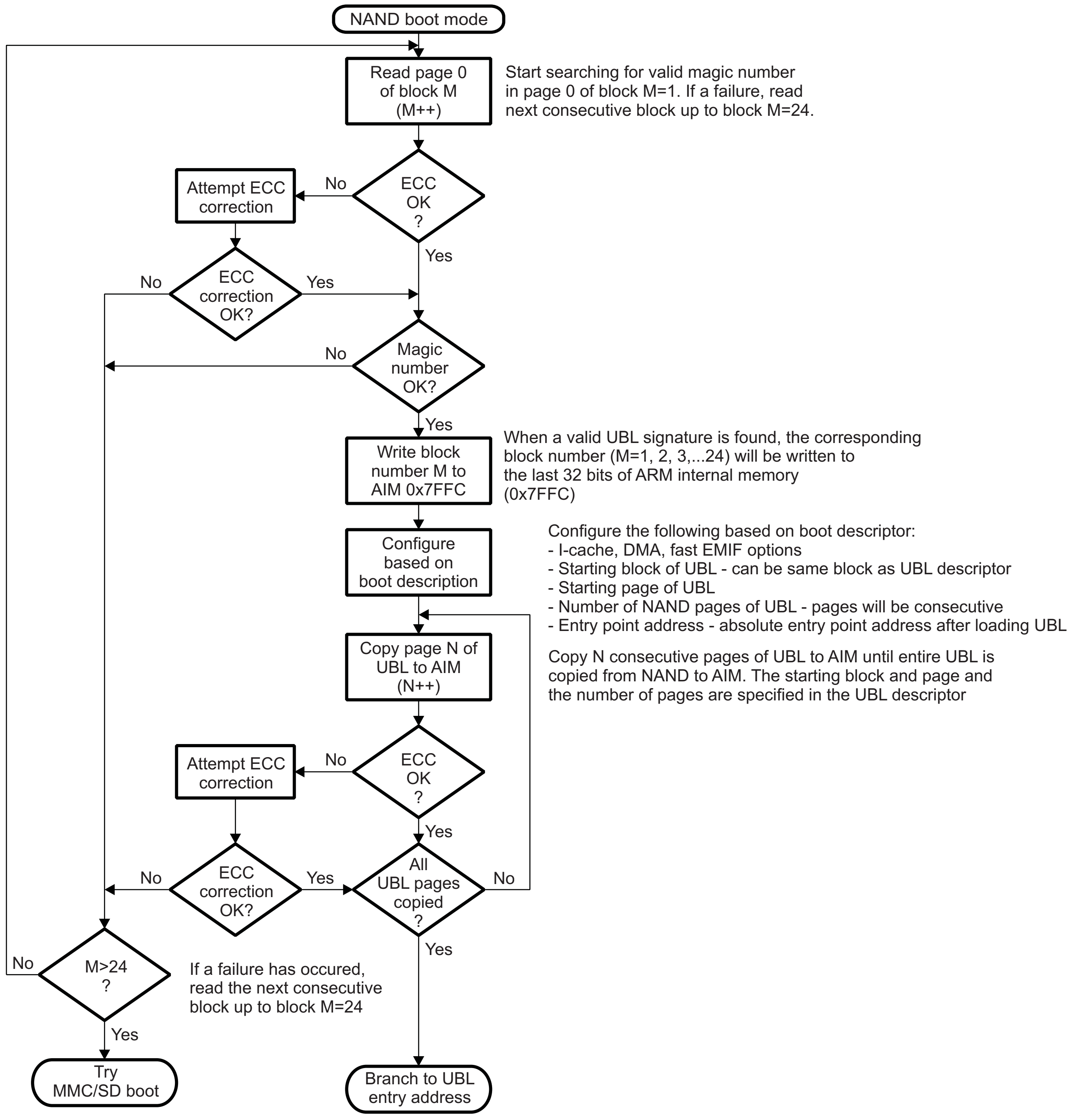

NAND boot mode
Read page 0 of block M (M++)
Start searching for valid magic number in page 0 of block M=1. If a failure, read next consecutive block up to block M=24.
ECC OK ?
No
Attempt ECC correction
Yes
ECC correction OK?
No
Yes
Magic number OK?
No
Yes
Write block number M to AIM 0x7FFC
When a valid UBL signature is found, the corresponding block number (M=1, 2, 3,...24) will be written to the last 32 bits of ARM internal memory (0x7FFC)
Configure based on boot description
Configure the following based on boot descriptor:
- I-cache, DMA, fast EMIF options
- Starting block of UBL - can be same block as UBL descriptor
- Starting page of UBL
- Number of NAND pages of UBL - pages will be consecutive
- Entry point address - absolute entry point address after loading UBL
Copy page N of UBL to AIM (N++)
Copy N consecutive pages of UBL to AIM until entire UBL is copied from NAND to AIM. The starting block and page and the number of pages are specified in the UBL descriptor
ECC OK ?
No
Attempt ECC correction
Yes
ECC correction OK?
No
Yes
All UBL pages copied ?
No
Yes
M>24 ?
No
Yes
If a failure has occured, read the next consecutive block up to block M=24
Try MMC/SD boot
Branch to UBL entry address

Figure 101. ARM NAND ROM bootloader Example

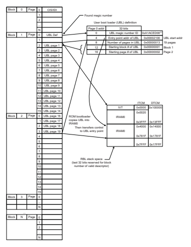
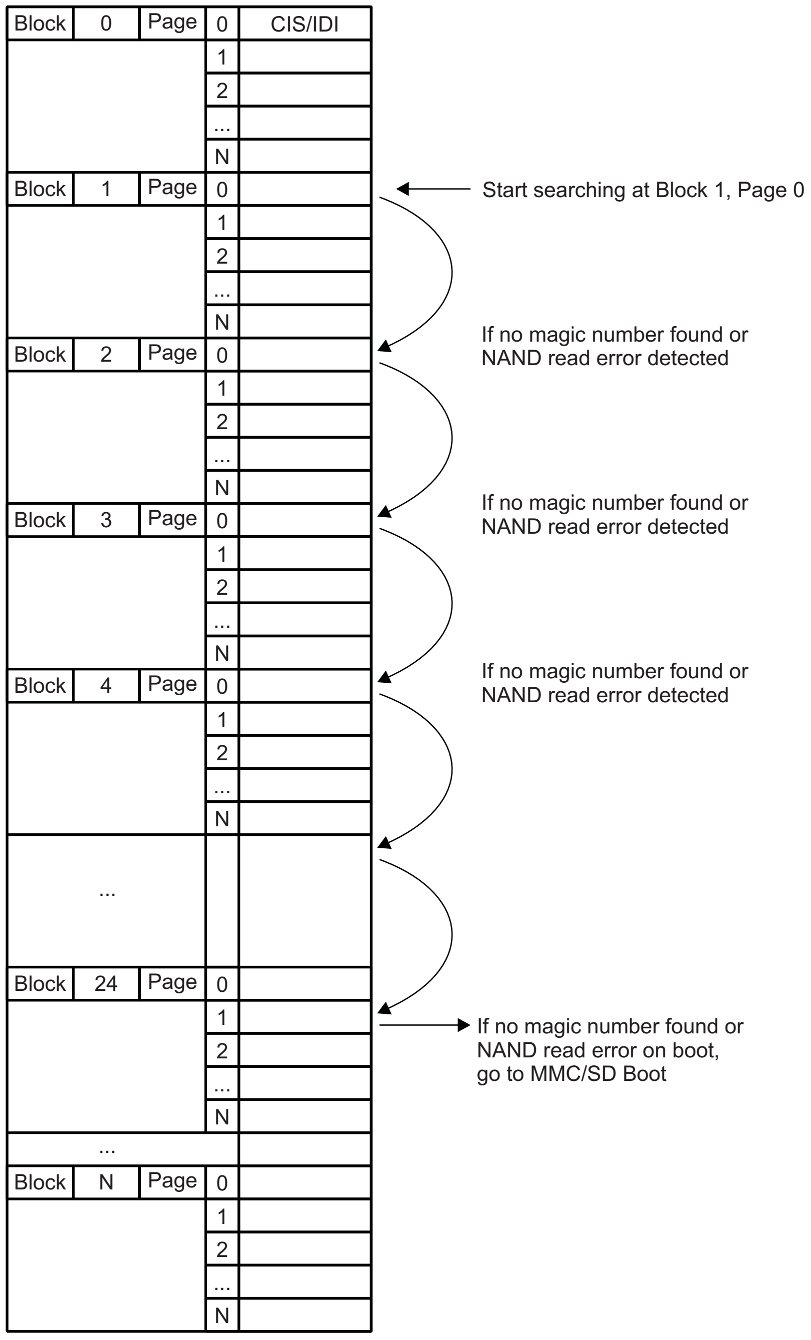


Figure 102. Descriptor Search for ARM NAND Boot Mode


Block 0 Page 0 CIS/IDI
1
2
...
N
Block 1 Page 0
Start searching at Block 1, Page 0
If no magic number found or NAND read error detected
Block 2 Page 0
If no magic number found or NAND read error detected
Block 3 Page 0
If no magic number found or NAND read error detected
Block 4 Page 0
...
Block 24 Page 0
If no magic number found or NAND read error on boot, go to MMC/SD Boot
Block N Page 0

11.2.1.2 NAND Device IDs Supported

The list of IDs supported by ROM bootloader, along with its characteristics, is shown in Table 113.

Table 113. NAND IDs Supported

Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (For address)	No. of address cycles
0x43	32	512+16	13	3
0x45	32	512+16	13	3
0x53	32	512+16	13	3
0x55	32	512+16	13	3
0x73	32	512+16	13	3
0x33	32	512+16	13	3
0x75	32	512+16	13	3
0x35	32	512+16	13	3
0x76	32	512+16	13	4
0x36	32	512+16	13	4
0x79	32	512+16	13	4
0x71	32	512+16	13	4
0x46	32	512+16	13	4
0x56	32	512+16	13	4
0x74	32	512+16	13	4
0xF1	64	2048+64	22	4
0xA1	64	2048+64	22	4
0x78	32	512+16	13	4
0x98DA	64	512+16	14	4
0x98DC	64	512+16	14	4

The UBL_MAGIC_SAFE_LEGACY mode will be used for big block NAND devices. The list of such NAND devices is given below. In UBL_MAGIC_SAFE_LEGACY mode NAND device IDs will be read from both Table 113 and Table 114. When not in UBL_MAGIC_SAFE_LEGACY mode NAND device IDs will only be read from Table 113.

Table 114. Device IDs of NANDs supported in BL_MAGIC_SAFE_LEGACY mode

Device ID / Manufacturer ID + Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (For address)	No. of address cycles
0xB1	64	2048+64	22	4
0xC1	64	2048+64	22	4
0xAA	64	2048+64	22	5
0x2CDA	64	2048+64	22	5
0xAC	64	2048+64	22	5
0x2CDC	64	2048+64	22	5
0xADD3	64	2048+64	22	5
0xECD3	128	2048+64	23	5
0xA3	64	2048+64	22	5
0xA5	64	2048+64	22	5
0xECD3	64	4096+128	22	5
0xECD5	64	4096+128	22	5

11.2.2 MMC/SD Boot Mode

The following list describes the process for MMC/SD boot mode.

If the value is BTSEL[2:0] from the BOOTCFG register is 010, the MMC/SD boot mode will be executed. The outline of operations followed in the MMC/SD mode is shown in Figure 103.

The MMC/SD card needs to be powered on for boot up. After the boot up is finished, a GIO may be used as a power switch to the MMC/SD card.

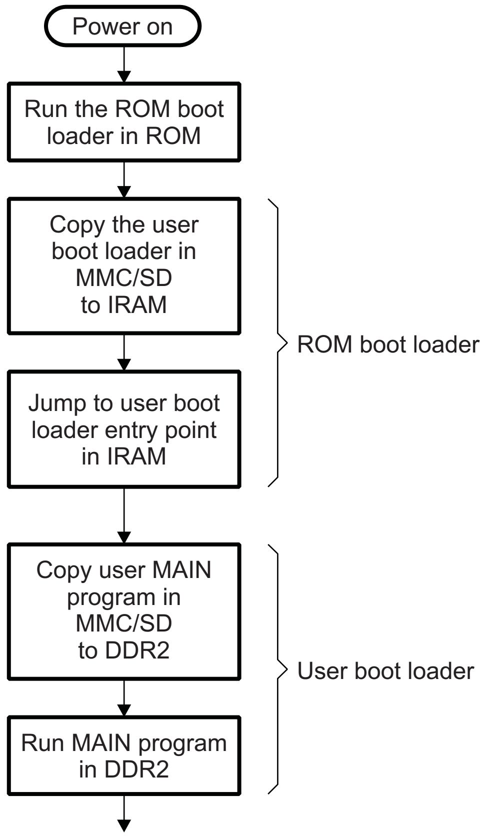
Initialization information, such as block size, is read from the CID and CSD registers of the MMC/SD device. The CID and CSD registers of the MMC/SD device are read by the MMC/SD module in native mode. All initialization and data transfers are done in native mode. SPI mode is not supported.

After performing the MMC/SD initialization sequence, the RBL searches for the UBL Descriptor starting in block 0. If a valid UBL is not found in block 0, as determined by reading a proper UBL magic number, the next block will be searched. Searching will continue for up to 24 blocks. This provision for additional searching is made in case the first few consecutive blocks have errors. When a valid UBL descriptor is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). This feature is provided as a basic debug mechanism. By reading these 32 bits of memory, via JTAG for example, you can determine in which block the RBL found a valid UBL signature. If no valid UBL signature is found after searching 24 blocks, the RBL will toggle GIO61 at 4Hz while repeating the MMC/SD boot process from the beginning.

The UBL descriptor, which gives the information required for loading and control transfer to the UBL, will then be read and processed. Based on information in the UBL descriptor, the RBL may first enable I-Cache operation. Once the specified startup conditions are set, the RBL will copy the UBL into ARM Internal RAM, starting at 0x0000:0020. Note that the actual copy will be done to the lower 30KB of the TCM Data area: 0x10020 0x1781F.

The MMC/SD RBL will use the hardware CRC error detection capability to determine if a read error occurs when reading the UBL including the UBL descriptor. If a read error occurs, the UBL copy will immediately halt for that instance of magic number but the RBL will continue to search the block following that block in which the magic number was found for another instance of a magic number. When a magic number is found, the process is repeated. Using this retry process, the magic number and UBL can be duplicated up to 24 times, which will give significant redundancy and error resilience to MMC/SD read errors.

Figure 103. MMC/SD Boot Mode Overview


Power on
Run the ROM boot loader in ROM
Copy the user boot loader in MMC/SD to IRAM
Jump to user boot loader entry point in IRAM
ROM boot loader
Copy user MAIN program in MMC/SD to DDR2
Run MAIN program in DDR2
User boot loader

The MMC/SD user bootloader UBL descriptor format is described in Table 115.

Table 115. MMC/SD UBL Descriptor

Page 0 Address	32-Bits	Description
0	0xA1AC EDxx	Magic number (0xA1ACEDxx)
4	Entry Point Address of UBL	Entry point address for the user bootloader (absolute address)
8	Number of blocks in UBL	Number of blocks (size of user bootloader in number of blocks)
12	Starting Block # of UBL	Block number where user bootloader is located

Table 116. MMC/SD UBL Signatures and Special Modes

Mode	Value	Description
UBL_MAGIC_SAFE	0x A1AC ED00	Safe boot mode
UBL_MAGIC_IC	0x A1AC ED22	I Cache boot mode

11.2.2.1 MMC/SD Boot Detailed Flow

An overview of the MMC/SD boot process is shown in the flow chart in Figure 104 and exemplified in Figure 105. The following steps describe the MMC/SD boot process. There are two instantiations of the MMC/SD Controller in the device: MMC/SD0 and MMCSD1. The MMC/SD boot shall use only MMC/SD0.

- Initialize the stack in the upper 2K of RAM1 in IRAM (RAM1 ' 0x7800-0x7FFF). Do not use the last 32-bits of IRAM (0x7ffc-0x8000) for stack, since these will be written with the block number corresponding to the valid block number.
- Disable all interrupts, IRQ and FIQ
- Read CID and CSD registers of MMC/SD and initialize the MMC/SD module in Native mode.
- Search for the magic number in the UBL descriptor starting in block 0 (see Figure 106). The magic number is of the format 0xA1ACEDxx and is in the first 32-bits of the block. CRC error detection shall be enabled when reading the UBL Descriptor. If a CRC read error is detected or the magic number is not valid, the descriptor search process shall begin anew in the next block after that in which the UBL descriptor was just searched for up to the first 24 blocks. When a valid UBL signature is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). The UBL Descriptor provides the needed details of the user bootloader. See Table 115 and Table 116 for details of the UBL Descriptor. The UBL Descriptor consists of the following parameters (all UBL parameters are 32 bits wide):
  – Entry Point Address: absolute entry point AFTER loading UBL
    - Must be in range 0x0020 - 0x781C
  – Number of MMC/SD blocks in UBL:
    - Must be contiguous blocks
    - Total bytes must be less than or equal to 30KByte total (size of IRAM - ~2KB stack space)
  – Number of MMC/SD blocks in UBL:
    - Starting Block of UBL cannot be same block as UBL Definition
- Copy the UBL from MMC/SD to IRAM with hardware CRC error detection enabled. If a CRC read error is detected, the descriptor search process begins anew in the next block after that in which the UBL descriptor was found for up to the first 24 blocks. Detected CRC errors will not be corrected. If no valid magic number is found after searching 24 blocks, giving significant redundancy and error resilience to MMC/SD read errors.
- Give control to user bootloader at UBL Entry Address.

Figure 104. MMC/SD Boot Mode Flow Chart

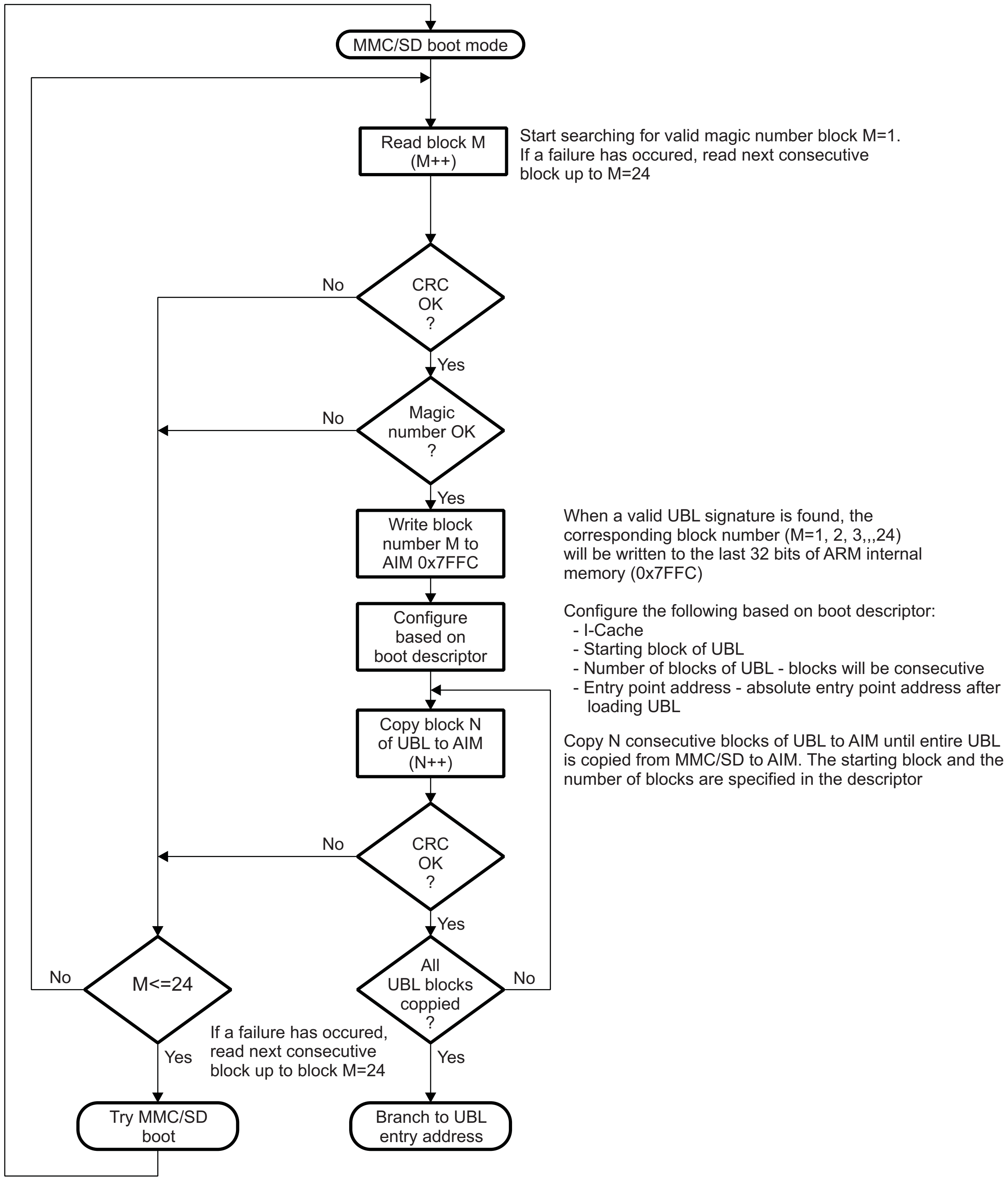

MMC/SD boot mode
Read block M (M++)
Start searching for valid magic number block M=1. If a failure has occured, read next consecutive block up to M=24
CRC OK ?
No
Yes
Magic number OK ?
No
Yes
Write block number M to AIM 0x7FFC
When a valid UBL signature is found, the corresponding block number (M=1, 2, 3,,,24) will be written to the last 32 bits of ARM internal memory (0x7FFC)
Configure based on boot descriptor
Configure the following based on boot descriptor:
- I-Cache
- Starting block of UBL
- Number of blocks of UBL - blocks will be consecutive
- Entry point address - absolute entry point address after loading UBL
Copy block N of UBL to AIM (N++)
Copy N consecutive blocks of UBL to AIM until entire UBL is copied from MMC/SD to AIM. The starting block and the number of blocks are specified in the descriptor
CRC OK ?
No
Yes
All UBL blocks coppied ?
No
Yes
M<=24
No
Yes
If a failure has occured, read next consecutive block up to block M=24
Try MMC/SD boot
Branch to UBL entry address

Figure 105. ARM MMC/SD ROM Bootloader Example

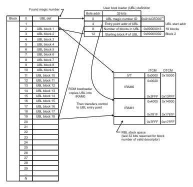
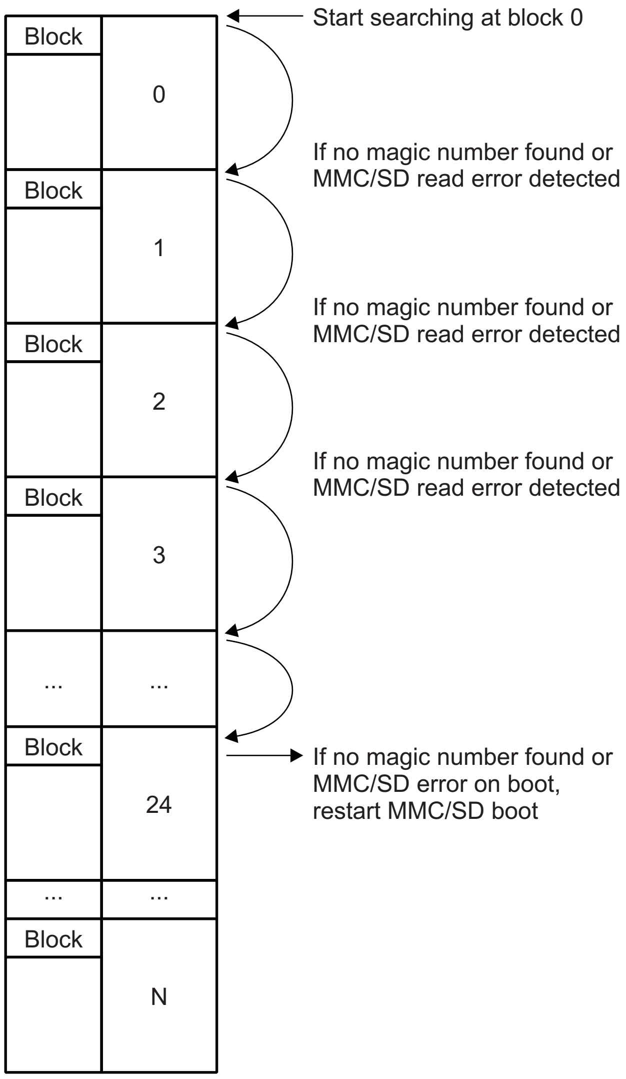


Figure 106. Descriptor Search for ARM MMC/SD Boot Mode


Block
0
Block
1
Block
2
Block
3
...
...
Block
24
...
...
Block
N
Start searching at block 0
If no magic number found or MMC/SD read error detected
If no magic number found or MMC/SD read error detected
If no magic number found or MMC/SD read error detected
If no magic number found or MMC/SD error on boot, restart MMC/SD boot

11.2.3 UART Boot Mode

If the state of BTSEL[2:0] pins at reset is 011, then the UART boot mode executes. This mode enables a small program, referred to here as a user bootloader (UBL), to be downloaded to the on-chip ARM internal RAM via the on-chip serial UART and executed. A host program, referred to as a serial host utility program, manages the interaction with the RBL and provides a means for operator feedback and input.

The UART boot mode execution assumes the following UART settings:

Time-Out	500 ms, one-shot
Serial RS-232 port	115.2 Kbps, 8-bit, no parity, one stop bit
Command, data, and checksum format	Everything sent from the host to the DM36x UART RBL must be in ASCII format.

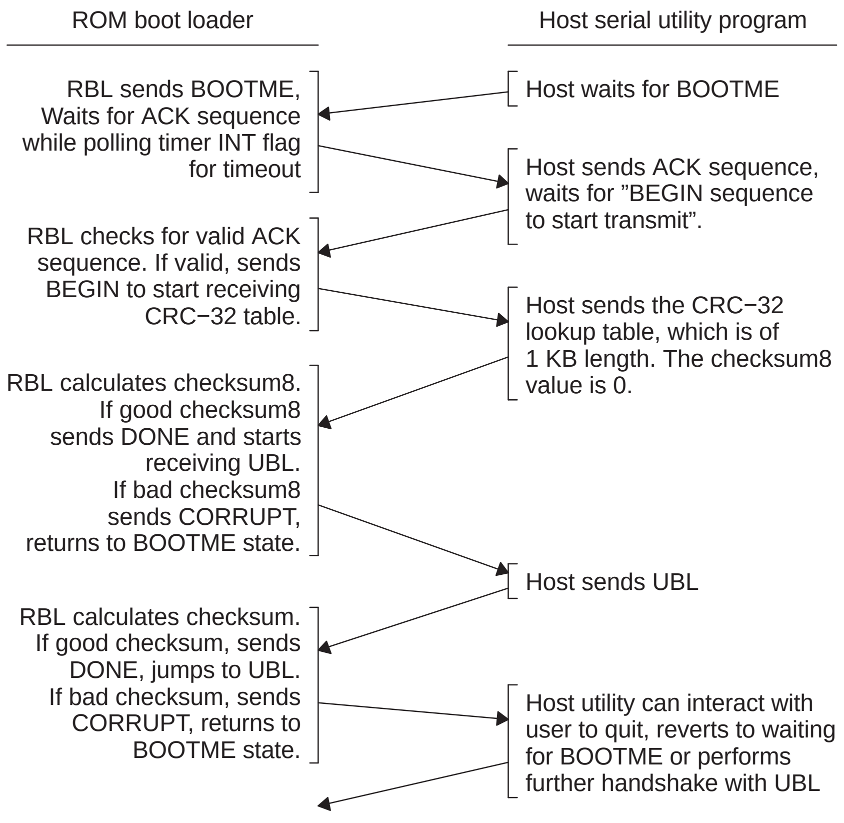
11.2.3.1 Serial Host Handshake

If the state of the BTSEL[2:0] pins reset is 011, then the UART boot mode executes as shown in Figure 107. The state of the BTSEL[2:0] pins at reset is captured and stored in the BTSEL bits in the BOOTCFG register in the System Control module. The RBL reads this register to determine whether to execute the UART boot. Figure 107 shows the handshake between the device and a serial host utility program. After initialization, there are three main receive sequences: ACK, 1KB CRC32 table, and user bootloader (UBL). For each receive sequence, a time-out check is done in the RBL. This means that if a timeout value is reached during the sequence, the serial boot mode restarts from the beginning at which the RBL sends out the BOOTME message. The error checking behavior for the UART receive mode is the same. For each byte received, if there is an error, RBL restarts from the beginning. In UART boot mode, the RBL copies the UBL code starting at address 0x0020.

The check sum method used for UBL data is CRC 32 check sum. The lookup table that is used for the CRC 32 calculation (1KB) must be sent by the host serial utility. Check sum8 is used as the check sum methodology for the CRC 32 lookup table.

When calculated, the check sum8 value for the lookup table results in a value of 0. Since this value remains the same, it is checked by the RBL before downloading the UBL data from the host serial utility. Whenever a wrong ACK, CRC 32 table, or UBL is received, the serial boot process restarts.

Figure 107. UART Boot Mode Handshake


ROM boot loader
Host serial utility program
RBL sends BOOTME, Waits for ACK sequence while polling timer INT flag for timeout
Host waits for BOOTME
Host sends ACK sequence, waits for ”BEGIN sequence to start transmit”.
RBL checks for valid ACK sequence. If valid, sends BEGIN to start receiving CRC−32 table.
Host sends the CRC−32 lookup table, which is of 1 KB length. The checksum8 value is 0.
RBL calculates checksum8. If good checksum8 sends DONE and starts receiving UBL. If bad checksum8 sends CORRUPT, returns to BOOTME state.
Host sends UBL
RBL calculates checksum. If good checksum, sends DONE, jumps to UBL. If bad checksum, sends CORRUPT, returns to BOOTME state.
Host utility can interact with user to quit, reverts to waiting for BOOTME or performs further handshake with UBL

11.2.3.2 UART Bootloader Data Sequences

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

Table 117 lists the values for the handshake sequences and header for UBL.

Table 117. UART Data Sequences

Sequence	Sequence	Usage
BOOTME	^BOOTME/0	Notify host utility serial boot mode begins. This is an 8-byte ASCII value. ^ is a space.
ACK	^^^^ACK/0	For the host utility to respond within the time out period by sending a 28-byte header to prepare for reception of user bootloader. The check sum is a 32-bit check sum. Note that the RBL jumps the program counter to the start address (i.e. UBL entry point) after the downloading process. If the sequence is 0001, no BEGIN is sent below; if it is 0000, BEGIN is sent.
	UBL 8-byte check sum	
	UBL 4-byte count	
	UBL 4-byte ARM physical start address 0000 or 0001	
BEGIN	^^BEGIN/0	RBL to signal host utility to begin transmission of user bootloader
DONE	^^^DONE/0	RBL to signal host utility that data received is OK and the transfer can be terminated
BAD ADDR	BADADDR/0	Bad start address received
BAD COUNT	^BADCNT/0	Bad count received
CORRUPT	CORRUPT/0	RBL to signal host utility that there is an error with the transmission. The host utility asks you to reset the board.
UBL	Variable	The format for UBL is the same as NAND boot.

The CRC 32 check sum value is calculated for the UBL data and passed by the host serial utility. The polynomial used for CRC32 is:

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

11.2.3.3 Host Utility Data Format

The RBL expects the data sent from the host utility to be in a particular format. This section describes the data format for the ACK, 1KB CRC32 table, and UBL sequences.

All data sent from the host to the RBL must be in ASCII format. The host utility must transfer the ACK sequence as shown in Table 118, CRC32 table as shown in Table 119, and UBL in the same format as the CRC32 table (Table 119). The check sum8 is calculated as shown in Table 119. The check sum8 is calculated after each transfer completes. Also note that since eight bytes are necessary to do the check sum8 calculation, the host utility must be sure to send a multiple of eight bytes of total data.

Example 2.

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

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

Table 118. Host Utility Data Format

"^^^^ACK/0" Transfer

	D0	D1	D2	D3	D4	D5	D6	D7
Start Byte ——→	"^"	"^"	"^"	"^"	"A"	"C"	"K"	"/0"

Table 118. Host Utility Data Format (continued)

"^^^^ACK/0" Transfer

D0	D1	D2	D3	D4	D5	D6	D7

Checksum "9af944c9" Transfer

D0	D1	D2	D3	D4	D5	D6	D7	
9	a	f	9	4	4	c	9	——→

Table 119. CRC32 Table Transfer

crc32_table[1024] = {0x01234567L, 0x89ABCDEFL....}

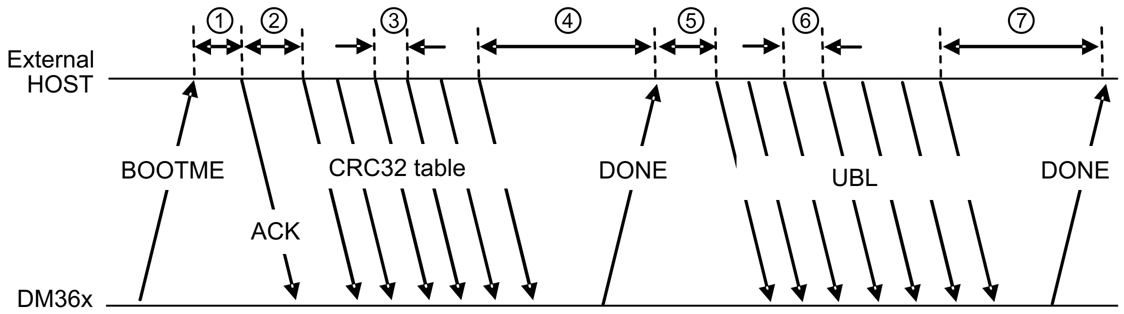
	D0	D1	D2	D3	D4	D5	D6	D7	
Start Byte ——→	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	

D0	D1	D2	D3	D4	D5	D6	D7	
"8"	"9"	"A"	"B"	"C"	"D"	"E"	"F:	——→
0x67 + 0x45 + 0x32 + 0x01 + 0xFE + 0xCD + 0xAB + 0x89 + ... =>checksum8								

11.2.3.5 Host Utility Timing Requirements

The UART host utility must allow the RBL time to process commands and data. When sending characters from the host to the UART RBL, the host utility must insert a delay between each byte character equal to 1 ms. Furthermore, 5 ms delay must be inserted for each of the timing parameters shown in Figure 108.

Figure 108. Host Utility Timing



(1) The delay time from "BOOTME" received until "^^^ACK" sent.

(2) The delay time from "ACK" sent to CRC32 table data sent.

(3) The delay time from CRC32-table data sent to next CRC32-table data sent.

(4) The delay time from final CRC32-table data sent to "DONE" or "CORREPT" received.

(5) The delay time from "DONE" or "CORREPT" received until UBL data sent.

(6) The delay time from UBL data sent until next UBL data sent.

(7) The delay time from final UBL data sent until "DONE" or "CORREPT" received.

11.2.4 ARM ROM USB Mode

The following are some specifics concerning the USB mode:

- If the value is BTSEL[2:0] from the BOOTCFG register is ‘100’, the USB boot mode will be executed.

- The RBL must enable the USB pin multiplexing if necessary, since pin multiplexing will not be enabled automatically by the BTSEL[2:0] settings. This mode enables the user bootloader (UBL) to be downloaded to the on-chip ARM internal RAM via the on-chip USB and executed.
- A host program, hereafter referred to as USB host utility program, manages the interaction with RBL and provides a mean for operator feedback and input.
- In the USB boot mode, the DM36x USB module is initialized as function (or device) with one out (receive) endpoint. The boot code is received from a USB host controller.
- The data format (entry address, length of section, address for section to store data and code, multiple sections) is as shown in Section 11.2.4.1.
- The USB already supports CRC on byte level with its hardware, therefore no checksum verification is required for USB boot loading.
- The USB module is put in self-powered mode and the RBL will not support the remote wakeup feature.
- Support for up to 30KB UBL (32KB IRAM – 2KB(for RBL stack)) in ARM TCM RAM.
- High speed mode will not be supported by the RBL.

11.2.4.1 USB Booting Detailed Flow

After the device is attached, the host performs the enumeration and GetConfiguration() to verify that the right chip is connected. Table 120 shows the USB definitions for enumeration.

Table 120. USB Definitions for Enumeration

USB Definition	Description
Manufacturer or Vendor ID	0x51, 0x04
String Descriptor	May be supported (subject to code space availability in the ROM)
Processor ID	0x55AA
Release (1.0)	0x0100

The Host sends the data and code in following format:

- Magic Word – 4 bytes
  - Send entry address (start address of bootloaded software) – 4 bytes
  - Send block length of data (in bytes) to be downloaded – 4 bytes
  - Send address where block of data has to be stored – 4 bytes
  - Send 4 bytes of zeroes (reserved word)
  - Download the block with ‘n’ bytes of code or data

Note that the first data packet transmitted by the host on Endpoint1 should be at least 20 bytes in size (containing the five words mentioned above).

1.The following information is returned to the USB host on the receipt of request for a device descriptor:

```
{ 0x12, /**< bLength */
0x01, /**< bDescriptorType */
0x00, 0x02, /**< bcdUSB */
0x00, /**< bDeviceClass */
0x00, /**< bSubDeviceClass */
0x00, /**< bDeviceProtocol */
0x40, /**< bMaxPacketSize0 */
0x51, 0x04, /**< idVendor */
0x55AA /**< idProduct */
0x00, 0x02, /**< bcdDevice */
usb_str_descr_manufacturer_index, /**< iManufacturer (1) */
usb_str_descr_product_index, /**< iProduct (2) */
usb_str_descr_serial_number_index, /**< iSerialNumber (3) */
0x01 /**< bNumConfigurations */ }
```

2. The following information is returned to the USB host on the receipt of request for a configuration descriptor :

```
{ 0x09, /**< bLength */
0x02, /**< bDescriptorType */
0x19, 0x00, /**< wTotalLength ( 9 + 9 + 7 = 25) */
0x01, /**< bNumInterfaces */
0x01, /**< bConfigurationValue */
0x00, /**< iConfiguration */
0xC0, /**< bmAttributes ( Self powered/No remote wakeup) */
0x00, /**< bMaxPower */
0x09, /**< bLength */
0x04, /**< bDescriptorType */
0x00, /**< bIntefaceNumber */
0x00, /**< bAlternateSetting */
0x01, /**< bNumEndpoints */
0x00, /**< bInterfaceClass */
0x00, /**< bInterfaceSubClass */
0x00, /**< bInterfaceProtocol */
0x00, /**< iInterface */
0x07, /**< bLength */
0x05, /**< bDescriptorType */
0x01, /**< bEndpointAddress. Endpoint 1 - OUT */
0x02, /**< bmAttributes.Data Endpoint - Bulk Transfer */
0x40, 0x00, /**< wMaxPacketSize. 64 */
0x00, /**< bInterval */ }
```

Please note that a request for a configuration descriptor returns the configuration descriptor, all interface descriptors (in our case, one) and endpoint descriptors for all the interfaces supported (in our case, one). The interface descriptor follows the configuration descriptor. The endpoint descriptor for the interface follows the interface descriptor.

3 String Descriptors :

String index 0 for all languages returns a string descriptor that contains an array of two-byte LANGID codes supported by the device.

The following information will be returned to the USB host in response to a request for string descriptor 0:

```
{ 0x04, /**< bLength (N + 2) */
0x03, /**< bDescriptorType */
0x09,0x04 /**< 0x0409 stands for English (United States) */ }
```

The following information will be returned to the USB host in response to a request for string descriptor 1 (manufacturer index) :

```
{ 0x24, /**< bLength (N + 2) */
0x03, /**< bDescriptorType */
'T', 0x00, 'e', 0x00, 'x', 0x00, 'a', 0x00, 's', 0x00, ' ', 0x00, 'I',
0x00, 'n', 0x00, 's', 0x00, 't', 0x00, 'r', 0x00, 'u', 0x00, 'm', 0x00,'e', 0x00, 'n', 0x00, 't',
0x00, 's', 0x00 }
```

The following information will be returned to the USB host in response to a request for string descriptor 2 (product index):

```
{ 0x10, /**< bLength (N + 2) */
0x03, /**< bDescriptorType */
'D', 0x00, 'a', 0x00, 'v', 0x00, 'I', 0x00, 'n', 0x00, 'c', 0x00, 'I', 0x00 }
```

The following information will be returned to the USB host in response to a request for string descriptor 3 (serial number index) :

```
{ 0x0C, /**< bLength (N + 2) */
0x03, /**< bDescriptorType */
'D', 0x00, 'M', 0x00, '3', 0x00, '6', 0x00, '0', 0x00 }
```

11.2.4.2 USB UBL Descriptor Format

Table 121 describes the USB UBL Descriptor format. The data follows the UBL descriptor at offset 0x0a.

Table 121. USB UBL Descriptor

Offset	Field	Size (in bytes)	Description
0x0	Magic Number	4	0xA1ACED00
0x4	UBL Entry Point	4	After boot loading, the RBL jumps to execute from this point
0x8	UBL Data size (N)	4	Length of bytes of data to be downloaded

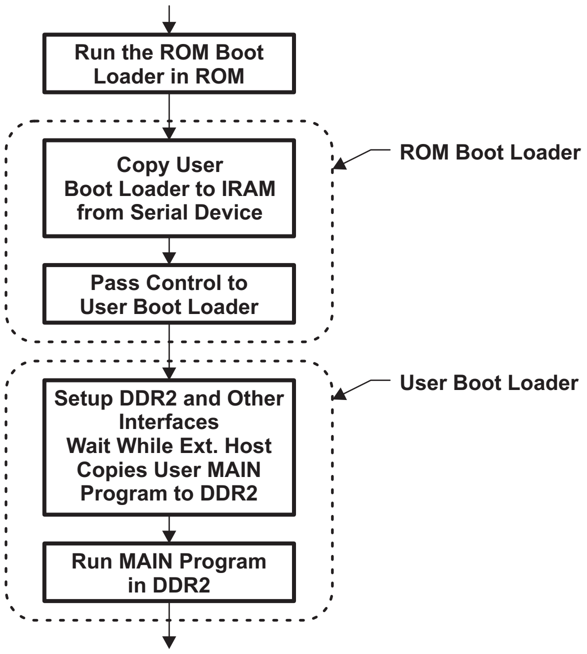
Table 121. USB UBL Descriptor (continued)

Offset	Field	Size (in bytes)	Description
0xC	UBL Load Address	4	Where the data should be stored (load address)
0x10	Rsvd	4	Send four bytes of zeros
0x14	Data	UBL code size	Data for UBL: N number of bytes

11.2.5 ARM ROM SPI Mode

If the value in BTSEL[2:0] from the BOOTCFG register is ‘101’, the SPI boot mode will be executed. The operations followed in the SPI boot mode are described in Figure 109.

Figure 109. SPI Boot Overview


Run the ROM Boot Loader in ROM
Copy User Boot Loader to IRAM from Serial Device
Pass Control to User Boot Loader
ROM Boot Loader
Setup DDR2 and Other Interfaces Wait While Ext. Host Copies User MAIN Program to DDR2
Run MAIN Program in DDR2
User Boot Loader

DM36x loads the UBL data in the following locations, ARM TCM RAM received via SPI0. The UBL data is received from a serial device like serial EEPROM.

11.2.5.1 SPI Key Features

The key features for SPI are as follows:

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

11.2.5.2 SPI Boot - Detailed Flow

The following list describes the flow of the SPI boot:

- RBL configures the pin-multiplexing settings to bring out the SPI0 signals
- RBL configures the EEPROM initially in 24-bit addressable mode and reads the first byte. Based on the first byte it will configure the EEPROM to 16-bit or 24-bit addressable modes.
- Bootloader reads entire UBL descriptor and finds out the properties of slave EEPROM The UBL descriptor contains the prescalar value, which is the divider used to generate the SPI clock. The

FAST_READ flag is used to indicate fast / normal mode, RBL will use FAST_READ command if the flag is set else uses standard READ command.

- RBL validates the other UBL header parameters
- Downloads the UBL to ARM internal memory
- RBL updates the boot status and then passes control to the entry point given in the UBL descriptor

Table 122. User Bootloader (UBL) Descriptor for SPI Mode

Byte Range	32-bits	Description
0-3	0xA1ACED0X	Magic number 0xA1ACED00 - 24 bit 0xA1ACED01 - 16 bit
4-7	0x0020-0x0781F	Entry point address for the user bootloader (absolute address) in ARM internal memory
8-11	UBL size	Size of UBL in bytes
12	Prescaler	Prescaler value to be used for dividing the clock for SPI
13	FASTREAD	Flag for enabling fast read (1 - fast read enables, 0- fast read disabled) Note: FAST READ option may not be valid for a specific EEPROM. Please read the EEPROM specifications before setting this parameter.
14-15	0X0000	Dummy bytes
16-19	Start address of UBL	Start address of UBL in EEPROM
20-23	0x0020-0x0781F	Load address of UBL in ARM internal memory

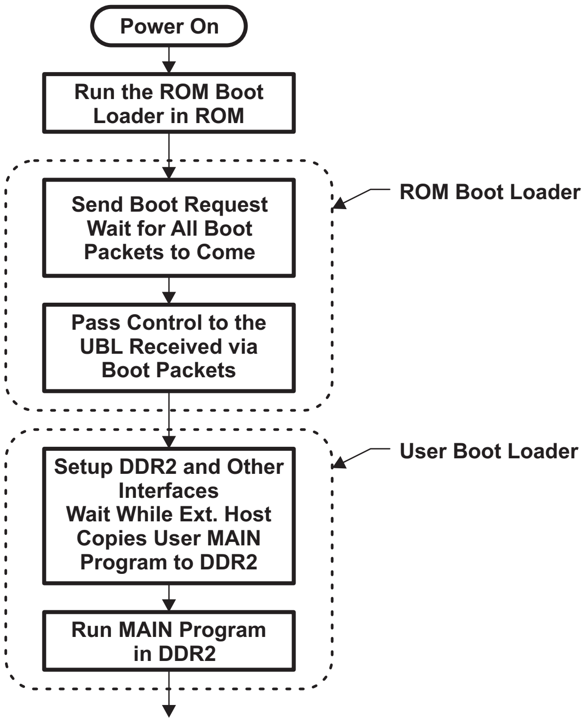
11.2.6 ARM ROM EMAC Mode

If the value in BTSEL[2:0] from the BOOTCFG register is ‘110’, the EMAC boot mode will be executed.

The operations followed in the EMAC boot mode are described here and illustrated in Figure 110.

1. The device sends a BOOTP request and the Host/Server will reply with boot packets.
2. The device will then wait for all the boot packets to arrive.
3. The boot packets are stored in ARM internal memory.
4. Once all the boot packets are received, the bootloader branches to the start of the UBL received via boot packets.

Figure 110. EMAC Boot Overview


Power On
Run the ROM Boot Loader in ROM
Send Boot Request Wait for All Boot Packets to Come
Pass Control to the UBL Received via Boot Packets
ROM Boot Loader
Setup DDR2 and Other Interfaces Wait While Ext. Host Copies User MAIN Program to DDR2
Run MAIN Program in DDR2
User Boot Loader

11.2.6.1 EMAC Key Features

The following are key features of EMAC:

- CPGMAC is initialized to operate in MII, full duplex mode with 10/100 Mbps mode rate. Also, its flow control is kept in disabled mode.
- CPGMAC operates with a single transfer and single receive channel.
- CPGMAC can optionally send Ethernet Ready Announcement Frame. This is done only once without any retransmission.
- Frames using both DIX and 802.3 MAC header formats will be accepted as will frames with and without VLAN tags.
- Any source MAC address is acceptable. But the destination MAC address or the M-MAC specified must be of this device.
- Transmit and Receive channel interrupts from CPGMAC are waited for in poll mode by ARM.

11.2.6.2 EMAC Address

Since the device does not have a unique MAC address programmed for each device, the MAC address needs to be obtained from external devices. The following protocol will be used for finding the MAC address:

- Check for a magic number (0xEADDA10x) in the I2C EEPROM. If found, the next 6 bytes will be the MAC address (in little endian format).
- If not found in I2C EEPROM, check for a magic number (0xEADDA10x) in the SPI EEPROM. If found, the next 6-bytes will be the MAC address (in little endian format).
- If not found, then the EMAC boot mode will use a default MAC address in the silicon. In this mode there is no magic number support. The EMAC boot happens with the following default values:
  – MAC Address: 08-00-28-32-C2-00
  – Source port : 0
  – Destination port: 9

- Mult and div Values : 45 and 7 respectively, to program ARM at 270 MHz and DDR at 216 MHz
- PacketQueueSize: 5. This allows a maximum of 17KB of UBL code size

11.2.6.3 EMAC Magic Numbers

Four valid magic numbers (0xEADDA10x) are used to read various EMAC boot parameters. The least significant byte of the magic word indicates how many bytes will be processed.

0: 4 bytes magic word and 6 bytes MAC address.

1: 4 bytes magic word, 6 bytes MAC address, 2 bytes source port and 2 bytes destination port. The default source and destination port values used are 0 and 9 respectively.

2: 4 bytes magic word, 6 bytes MAC address, 2 bytes source port, 2 bytes destination port, 2 bytes “mult” and 2 bytes “div” for PLL configuration.

3: 4 bytes magic word, 6 bytes MAC address, 2 bytes source port, 2 bytes destination port, 2 bytes “mult”, 2 bytes “div” for PLL configuration, and 2 bytes of “PacketQueueSize”. The default “PacketQueueSize” used is 10. This variable can be set to any value from 3 to 10. It is used to provide software flexibility for EMAC boot mode.

The bigger PacketQueueSize value has an adverse effect on the maximum UBL code size that could be downloaded. The available space for UBL code size + Packet Queue size is nearly 25KB (after considering memory allocation requirements for other buffers in the EMAC boot mode). Each packet allocation is 1600 bytes and if PacketQueueSize used is 10 then ~16KB is allocated to the packet queue. This puts an upper limit on UBL code size to be a maximum of 9KB. However, by keeping PacketQueueSize programmable to a minimum of 3 packets, the UBL code size can go as high as 20KB. The value of PacketQueueSize must be carefully chosen along with the PLL settings so that ARM processes the packets in the queue faster that CPGMAC receives them to avoid any overrun condition.

11.2.6.4 EMAC Boot - Detailed Flow

The bootloader configures the EMAC peripheral, configures the communications port programming interfaceRx (CPPI), and also routes the EMAC Rx interrupt to 53 and Tx interrupt to 54 in AINTC. Then the bootloader transmits an Ethernet Ready Frame out if it is selected. When an EMAC Rx interrupt happens, the bootloader processes the received packet and passes the boot tables into memory. When the end of the boot table is reached, it clears the Rx Channel 0 head description pointer to disable reception and exit boot and jumps to application. The steps of the flow are listed below:

1. Configure the pin-multiplexing settings for EMAC.
2. Initialize the default boot parameters.
3. Configure the CPGMAC based on boot parameters. Also configure AINTC to accept interrupts from CPGMAC.
4. Open the transmit channel, prepare Ethernet Ready Announcement Frame and send the packet.
5. Configure Receive CPPI and enable packet reception. Wait for an interrupt on the receive channel.
6. Process the receive packet and pass data to the boot table function.
7. Wait for another interrupt on the receive channel until the boot table is complete.
8. Clear the Receive HDP to disable packet reception and exit the bootloader.

The Ethernet peripheral is configured to accept a combination of a single MAC address; a single multicast address, and broadcast packets, as defined by the Ethernet boot parameter table. The peripheral rejects packets not matching the MAC addresses selected without a record of the drop. IRAM is allocated for the received packets. The 16KB starts from 0x20 are divided into 10 packet buffers each, of size 1600 bytes. The CPPI is allocated in the dedicated CPPI memory area.

11.2.6.5 EMAC Boot Table Frame Format

The Ethernet boot table frame format is shown in Table 123.

Table 123. EMAC Boot Table Frame Format

Ethernet Header, one of the following types: DIX Ethernet (DMAC, SMAC, type: 14 bytes) 802.3 w/ SNAP/LLC (DMAC, SMAC, len, LLC, SNAP: : 22 bytes) DIX Ethernet w/ VLAN (18 bytes) 802.3 w/ VLAN and SNAP/LLC (26 bytes)
IPV4 (20 to 84 bytes)
UDP (8 bytes)
Boot Table Frame Header (4 bytes)
Boot Table Frame Payload (min 4 bytes, max limited by max Ethernet frame – previous headers)

Table 124. Boot Table Header Format

Word address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Magic Number (0x544b)															
1	Opcode (0x01)								Sequence number							

The boot table format is encapsulated in Ethernet frames with IPV4 and UDP headers. The following points describe the Ethernet frames which are accepted. Frames not matching the criteria specified below are silently discarded and subsequent frames are processed.

Frames using both DIX and 802.3 MAC header formats are accepted as are frames with and without VLAN tags. Any source MAC address is acceptable. A destination MAC address of this device (as specified in boot params) or the M-MAC specified in the boot params is accepted. VLAN fields (other than type/len) are ignored. If 802.3 format MAC format is used, the SNAP/LLC header is verified and skipped. The type field selects IPV4 type (0x0800).

The IPV4 header validates the Version 4 (IPV6 is not supported), flag and fragment fields, and protocol (UDP) field. The header length field is parsed in order to properly skip header option words. Any source and destination IP addresses are accepted.

The UDP header validates that the source and destination port numbers match those specified in the boot parameters. If the boot parameter source port field is 0 then the source port is accepted. The UDP header length is sanity tested against the (appropriately adjusted) frame length. If the UDP length is too long for the frame or is not a multiple of 2, the frame is discarded. The UDP checksum is verified and the frame with incorrect UDP checksum is discarded if the UDP checksum field is non-zero.

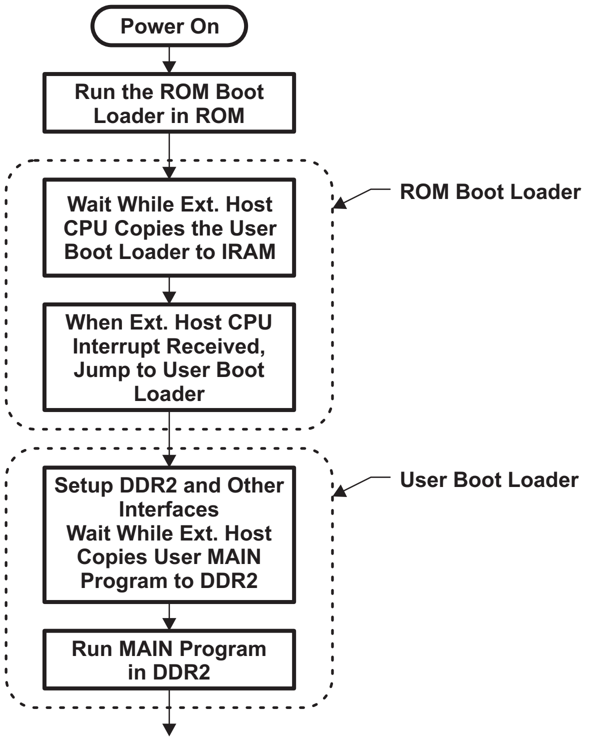
The following checks are performed on the boot table frame header. The magic number field and opcode fields are compared to the expected values. The sequence number field is compared to the expected value. The expected value for sequence number is 0 for the first frame processed and it increments by one for each processed frame. The sequence number must be allowed to flip over when the maximum is reached, meaning that the sequence number value 0 is expected after reception of sequence number 256.

The boot table frame payload (which is a multiple of 4 bytes in length) is processed by the boot-table processing function.

11.2.7 ARM ROM HPI Mode

If the value in BTSEL[2:0] from the BOOTCFG register is ‘111’, the HPI boot mode will be executed. The HOST can load the UBL data in the following locations, ARM DTCM RAM: (0x10020- 0x1781F). The overview of the HPI boot is shown in Figure 111.

Figure 111. HPI Boot Overview


Power On
Run the ROM Boot Loader in ROM
Wait While Ext. Host CPU Copies the User Boot Loader to IRAM
When Ext. Host CPU Interrupt Received, Jump to User Boot Loader
ROM Boot Loader
Setup DDR2 and Other Interfaces
Wait While Ext. Host Copies User MAIN Program to DDR2
Run MAIN Program in DDR2
User Boot Loader

11.2.7.1 HPI Key Features

The HPI includes the following features:

- It can load up to 30 KB size of UBL.
- It requires 50 seconds of wait time after sending the HINTN signal to the host.

11.2.7.2 HPI Boot - Detailed Flow

The detailed flow for HPI boot is as follows:

- The RBL assumes that HPI is enabled at power on in the HPI boot mode.
- The RBL signals the host that the device is ready via the host interrupt / HINTN signal.
- RBL drops into a polling loop, waiting for the external host interrupt to ACK its presence.
- If the Host does not ACK its presence within a timeout period of 50 seconds, the HPI boot will fail and the RBL will do a soft-reset of the HPI module and try again.
- If the Host does ACK via the interrupt, then the RBL will drop into a polling loop, waiting for the external host to load/copy the UBL and interrupt the device RBL. The RBL will wait forever until interrupted by the Host.
- In addition to the UBL, host will write the UBL info at location 0x7FE0 as given in Table 125.
- After the Host interrupt is received, the RBL will get the entry point for UBL from location 0x7FE0 and the control is transferred to the UBL.
- The RBL updates the boot status and then passes control to the entry point given in the UBL descriptor.

Table 125. User Bootloader (UBL) Descriptor for HPI mode

Address	32-bits	Description
0x7FE0	0xA1ACED00	Magic number
0x7FE4	Entry point addr of UBL	Entry point address for the user bootloader (absolute address in ARM internal memory)

12 Power Management

12.1 Overview

The device is designed for minimal power consumption. There are two components to power consumption: active power and leakage power. Active power is the power consumed to perform activity and scales with clock frequency and the amount of computations being performed. Active power can be reduced by controlling the clocks in such a way as to either operate at a clock setting just high enough to complete the required operation in the required timeline or to run at a clock setting until the activity is complete and then drastically cut the clocks (e.g., to PLL bypass mode) until additional activity must be performed. Leakage power is due to static current leakage and occurs regardless of the clock rate. Leakage, or standby power, is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely from a device or subsystem.

The DM36x includes several power management features which are briefly described in Table 126 and detailed in the following sections.

Table 126. Power Management Features

Power Management Features	Description
Clock Management	
Module clock disable	Module clocks can be disabled to reduce active power
Module clock frequency scaling	Module clock frequency can be scaled to reduce active power
PLL power-down	The PLLs can be powered-down when not in use to reduce active power
ARM Sleep Mode	
ARM Wait-for-Interrupt sleep mode	Disable ARM clock to reduce active power
System Sleep Modes	
Deep Sleep Mode	Stop all device clocks and power down internal oscillators to reduce active power to a minimum. Registers and memory contents are preserved.
PRTCSS Mode	Consumes the lowest possible power the device is not powered
I/O Management	
USB PHY power-down	The USB PHY can be powered-down to reduce USB I/O power
DAC power-down	The DAC's can be powered-down to reduce DAC power
DDR2 self-refresh and power down	The DDR2 device can be put in self-refresh and power down states
ADC power-down	The ADC will power-down automatically when inactive
Voice Codec power-down	The voice codec can be powered down to reduce voice codec power consumption when not in use

12.2 PSC and PLLC Overview

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

12.3 Clock Management

12.3.1 Module Clock Disable

The module clock disable feature allows software to disable individual module clocks, in order to reduce a module's active power consumption to 0. The device is designed in full static CMOS; thus, when a module clock stops, the module's state is writes to these bit(s) must always have a value of 0. When the clock is restarted, the module resumes operating from the stopping point. Stopping clocks to a module only affects active power consumption; it does not affect leakage power consumption.

The power and sleep controller (PSC) controls module clock disable/enable. The procedure to disable/enable module clocks is described in Section 7.

12.3.2 Module Clock Frequency Scaling

Module clock frequency is scalable by bypassing the PLLs or by programming its multiply and divide parameters. Reducing the clock frequency reduces the active power consumption linearly with frequency. It has no impact on leakage power consumption.

Section 5 and Section 6 describe how to bypass the PLLs and how to program PLL.

12.3.3 PLL Bypass and Power Down

You can bypass the PLLs in DM36x. Bypassing the PLLs sends the PLL reference clock to the post dividers of the PLLC instead of the PLL VCO output clock. The PLL reference clock is typically at 24 MHz; therefore, you can use this mode to reduce the core and module clock frequencies to very low levels without using the PLL during periods of very low system activity. Furthermore, you can power-down the PLL when bypassing it to save additional active power.

Section 5 and Section 6 describe PLL bypass and PLL power-down.

12.4 ARM Sleep Mode Management

12.4.1 ARM Wait-For-Interrupt Sleep Mode

The ARM module cannot have its clock turned off/on via the PSC module like other modules. 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 ARM926EJ-S 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 (e.g., 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 (e.g., 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 (e.g., GIO or WDT) must not be disabled, or the device will never wake up.

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

12.5 System Sleep Modes

12.5.1 Deep Sleep Mode

Deep sleep mode is a special power down mode in which all device clocks are stopped and the internal oscillators are powered down. Registers and software states are preserved and upon recovery, the program may continue from where it left off with minimal overhead involved. When enabled, driving GIO[0] low will initiate Deep Sleep and driving GIO[0] high will initiate wakeup from deep sleep.

The deep sleep power-down process works as follows:

- The ARM prepares for power down, typically after an external microcontroller notifies the ARM to prepare for power down via an interrupt or serial communication.
- The ARM puts DDR2 in its to self-refresh state. The program in DDR2 is preserved while DDR2 is in its self-refresh state. In the case of mDDR, you may utilize partial array self refresh (PASR) for additional power savings.
- To reduce the chip stand-by power, power-down all the analog blocks (PLL cores, DDR PHY DLL, DDR PHY, USB PHY and Video DAC).
- The ARM sets SLEEPENABLE in register DEEPSLEEP in the system module.
- The ARM informs the microcontroller that it has initiated deep sleep and begins polling SLEEPCOMPLETE in DEEPSLEEP. During the recovery process, the ARM will wake up and detect that SLEEPCOMPLETE has changed.
- The microcontroller transitions GIO0 from high to low and then continues to hold GIO0 low (for a minimum of 500 ns) until it desires to exit Deep Sleep mode. The transition of GIO0 from high to low creates a clock pulse advancing the deep sleep state machine. After this transition, all clocks are stopped and then the internal oscillators are powered down.
- At this point, the device is in deep sleep mode; power is reduced to a minimum.

The deep sleep wakeup process is as follows:

- To initiate the wakeup process, the microcontroller transitions GIO0 from low to high. This transition creates a clock pulse advancing the Deep Sleep state machine. After this transition, the oscillators are powered up and allowed to stabilize and then all clocks are restarted.
- The ARM detects that SLEEPCOMPLETE has changed
- The ARM clears SLEEPENABLE in register DEEPSLEEP
- The ARM brings DDR2 out of self refresh.
- At this point the ARM resumes normal operation. The ARM may branch to program code preserved in DDR2. Register states are preserved.

12.5.2 PRTCSS Mode

The PRTCSS mode consumes the lowest possible power except for switching off power altogether. In this mode only the PRTCSS is powered and the DM36x Device is not powered. For details refer to *TMS320DM36x Digital Media System-on-Chip (DMSoC) Power Management and Real-Time Clock Subsystem (PRTCSS) User's Guide* (SPRUFJ0).

12.6 I/O Management

12.6.1 USB PHY Power Down

You can power-down the USB PHY when it is not in use. The USB PHY is powered-down via the PHYPWDN bit in the USB_PHY_CTL register described in Section 9. Also see the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Users Guide* (SPRUFH9) for more information.

12.6.2 Video DAC Power Down

The VPBE includes three video digital-to-analog converters (DAC) to drive analog displays, such as NTSC and PAL television displays. The Video Encoder (VENC) module of the VPBE includes registers for enabling/disabling the DAC. You can use the VIE bit in VMOD to force the analog output of the DAC to a low level, regardless of the video signal. Furthermore, you can use the DACCLKEN in register VPSS_CLK_CTRL to disable each DAC clock. See the *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Users Guide* (SPRUFG9) for detailed information on DAC power-down.

12.6.3 DDR2 Self-Refresh and Power Down

The DDR2 controller supports self-refresh and power down. This allows you to put the DDR2 device in its self-refresh or power down states for power savings. See the *TMS320DM36x Digital Media system-on-Chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Users Guide* (SPRUFI2) for detailed information on to DDR2 self-refresh and power-down.

Appendix A Revision History

This document has been revised to include the following technical change(s).

Table 127. Revisions

Location	Modifications/Additions/Deletions
Global	All instances of EM_CE0 changed to $\overline{\text{EM\_CE0}}$
Global	All instances of EM_CE1 changed to $\overline{\text{EM\_CE1}}$
Section 11.2.3.1	Rewrote sentence beginning "In UART boot mode, the RBL copies the UBL code.."
Table 122	Replaced both "Entry Point Addr of UBL" and "Load address of UBL" with 0x0020-0x0781F
Section 11.2.5	For "DM36x loads the UBL data in the following locations, ARM TCM RAM (0x10020-0x1781F)..." Removed the address range (0x10020-0x1781F) from this sentence
Table 113 and Table 114.	Added this sentence between these two tables. "In UBL_MAGIC_SAFE_LEGACY mode NAND device IDs will be read from both tables 113 and 114. When not in UBL_MAGIC_SAFE_LEGACY mode NAND device IDs will only be read from table 113."
Figure 99	Replaced the graphic