

TMS320DM357 DMSoC ARM Subsystem

Reference Guide



Literature Number: SPRUG25
November 2008

Preface	11
1 Introduction	15
1.1 Block Diagram	16
1.2 ARM Subsystem in TMS320DM357 DMSoC.....	16
2 ARM Subsystem Overview	17
2.1 Purpose of the ARM Subsystem	18
2.2 Components of the ARM Subsystem	18
2.3 References	19
3 ARM Core	21
3.1 Introduction.....	22
3.2 Operating States/Modes.....	23
3.3 Processor Status Registers	23
3.4 Exceptions and Exception Vectors.....	24
3.5 The 16-BIS/32-BIS Concept	25
3.5.1 16-BIS/32-BIS Advantages	25
3.6 Co-Processor 15 (CP15)	26
3.6.1 Addresses in an ARM926EJ-S System	26
3.6.2 Memory Management Unit.....	26
3.6.3 Caches and Write Buffer	27
3.7 Tightly Coupled Memory	28
4 System Memory	31
4.1 Memory Map	32
4.1.1 ARM Internal Memories	32
4.1.2 External Memories	32
4.1.3 Peripherals	32
4.2 Memory Interfaces Overview	33
4.2.1 DDR2 EMIF	33
4.2.2 External Memory Interface	33
5 Device Clocking	35
5.1 Overview.....	36
5.2 Clock Domains.....	38
5.2.1 Core Domains	38
5.2.2 Frequency Flexibility	38
5.2.3 DDR2/EMIF Clock.....	39
5.2.4 I/O Domains.....	40
5.2.5 Video Processing Back End	41
6 PLL Controller	43
6.1 PLL Module	44
6.2 PLL1 Control.....	45

6.2.1	Device Clock Generation	46
6.2.2	Steps for Changing PLL1/Core Domain Frequency	46
6.3	PLL2 Control	48
6.3.1	Device Clock Generation	49
6.3.2	Steps for Changing PLL2 Frequency	49
6.4	PLL Controller Register Map.....	52
6.4.1	Peripheral ID Register (PID)	53
6.4.2	Reset Type Status Register (RSTYPE)	53
6.4.3	PLL Control Register (PLLCTL)	54
6.4.4	PLL Multiplier Control Register (PLLM)	55
6.4.5	PLL Controller Divider 1 Register (PLLDIV1).....	55
6.4.6	PLL Controller Divider 2 Register (PLLDIV2).....	56
6.4.7	PLL Controller Divider 3 Register (PLLDIV3).....	57
6.4.8	PLL Post-Divider Control Register (POSTDIV).....	58
6.4.9	Bypass Divider Register (BPDIV)	59
6.4.10	PLL Controller Command Register (PLLCMD).....	60
6.4.11	PLL Controller Status Register (PLLSTAT).....	61
6.4.12	PLL Controller Clock Align Control Register (ALNCTL).....	62
6.4.13	PLLDIV Ratio Change Status Register (DCHANGE).....	63
6.4.14	Clock Enable Control Register (CKEN)	64
6.4.15	Clock Status Register (CKSTAT)	65
6.4.16	SYSCLK Status Register (SYSTAT).....	66
6.4.17	PLL Controller Divider 5 Register (PLLDIV5).....	67
7	Power and Sleep Controller	69
7.1	Introduction.....	70
7.2	TMS320DM357 DMSoC Power Domain and Module Topology	71
7.3	Power Domain and Module States Defined	73
7.3.1	Power Domain States.....	73
7.3.2	Module States	73
7.4	Executing State Transitions.....	73
7.4.1	Module State Transitions	73
7.5	IcePick Emulation Support in the PSC	74
7.6	PSC Interrupts	75
7.6.1	Interrupt Events.....	75
7.6.2	Interrupt Registers.....	76
7.6.3	Interrupt Handling	76
7.7	PSC Registers	77
7.7.1	Peripheral Revision and Class Information Register (PID)	78
7.7.2	Interrupt Evaluation Register (INTEVAL).....	78
7.7.3	Module Error Pending Register 0 (mod 0-31) (MERRPR0)	79
7.7.4	Module Error Pending Register 1 (mod 32-40) (MERRPR1)	79
7.7.5	Module Error Clear Register 0 (mod 0-31) (MERRCR0).....	80
7.7.6	Module Error Clear Register 1 (mod 32-40) (MERRCR1)	80
7.7.7	Power Error Pending Register (PERRPR)	81
7.7.8	Power Error Clear Register (PERRCR)	81
7.7.9	Power Domain Transition Command Register (PTCMD)	82
7.7.10	Power Domain Transition Status Register (PTSTAT)	82
7.7.11	Power Domain Status n Register (PDSTATn).....	83

7.7.12	Power Domain Control n Register (PDCTLn)	84
7.7.13	Module Status n Register (MDSTAT0-MDSTAT40)	85
7.7.14	Module Control n Register (MDCTL0-MDCTL40)	86
8	Power Management	87
8.1	Overview	88
8.2	PSC and PLLC Overview	88
8.3	Clock Management	89
8.3.1	Module Clock ON/OFF	89
8.3.2	Module Clock Frequency Scaling	89
8.3.3	PLL Bypass and Power Down	89
8.4	ARM Sleep Mode Management	90
8.4.1	ARM Wait-For-Interrupt Sleep Mode	90
8.5	I/O Management	90
8.5.1	3.3 V I/O Power-Down	90
8.6	VDD3P3V_PWDN Register	91
8.7	USB Phy Power Down	91
8.8	Video DAC Power Down	91
9	Interrupt Controller	93
9.1	Introduction	94
9.2	Interrupt Mapping	94
9.3	INTC Methodology	95
9.3.1	Interrupt Mapping	96
9.3.2	Interrupt Prioritization	96
9.3.3	Vector Table Entry Address Generation	97
9.3.4	Clearing Interrupts	97
9.3.5	Enabling and Disabling Interrupts	98
9.4	INTC Registers	99
9.4.1	Fast Interrupt Request Status Register 0 (FIQ0)	100
9.4.2	Fast Interrupt Request Status Register 1 (FIQ1)	100
9.4.3	Interrupt Request Status Register 0 (IRQ0)	101
9.4.4	Interrupt Request Status Register 1 (IRQ1)	101
9.4.5	Fast Interrupt Request Entry Address Register (FIQENTRY)	102
9.4.6	Interrupt Request Entry Address Register (IRQENTRY)	102
9.4.7	Interrupt Enable Register 0 (EINT0)	103
9.4.8	Interrupt Enable Register 1 (EINT1)	103
9.4.9	Interrupt Operation Control Register (INTCTL)	104
9.4.10	Interrupt Entry Table Base Address Register (EABASE)	105
9.4.11	Interrupt Priority Register 0 (INTPRI0)	106
9.4.12	Interrupt Priority Register 1 (INTPRI1)	106
9.4.13	Interrupt Priority Register 2 (INTPRI2)	107
9.4.14	Interrupt Priority Register 3 (INTPRI3)	107
9.4.15	Interrupt Priority Register 4 (INTPRI4)	108
9.4.16	Interrupt Priority Register 5 (INTPRI5)	108
9.4.17	Interrupt Priority Register 6 (INTPRI6)	109
9.4.18	Interrupt Priority Register 7 (INTPRI7)	109
10	System Control Module	111
10.1	Overview of the System Control Module	112
10.2	Device Identification	112

10.3	Device Configuration	112
10.3.1	Pin Multiplexing Control	112
10.4	Device Boot Configuration Status	112
10.5	Power Management	113
10.5.1	DV _{DD} 3.3 V I/O Power-Down Control.....	113
10.6	Special Peripheral Status and Control	113
10.6.1	USB PHY Control.....	113
10.6.2	VPSS Clock and DAC Control	113
10.7	Bandwidth Management	113
10.7.1	Bus Master DMA Priority Control	113
10.8	System Control Register Descriptions	114
11	Reset	115
11.1	Reset Overview	116
11.2	Reset Pins	116
11.3	Types of Reset	117
11.3.1	Power-On Reset (POR).....	117
11.3.2	Warm Reset	117
11.3.3	Maximum Reset.....	117
11.3.4	Module Reset	118
11.4	Default Device Configurations	118
11.4.1	Device Configuration Pins	118
11.4.2	PLL and Clock Configuration.....	119
11.4.3	ARM Boot Mode Configuration	119
11.4.4	AEMIF Configuration	119
12	Boot Modes	121
12.1	Boot Modes Overview	122
12.1.1	Features.....	122
12.1.2	Functional Block Diagram	123
12.2	ARM ROM Boot Modes	124
12.2.1	NAND Boot Mode.....	124
12.2.2	UART Boot Mode	127
12.2.3	HPI Boot Mode.....	130

List of Figures

1-1	TMS320DM357 DMSoC Block Diagram	16
2-1	TMS320DM357 DMSoC ARM Subsystem Block Diagram	19
5-1	Overall Clocking Diagram	37
5-2	VPBE/DAC Clocking	41
6-1	PLL1 Structure in the TMS320DM357 DMSoC	45
6-2	PLL2 Structure in TMS320DM357 DMSoC	48
6-3	Peripheral ID Register (PID)	53
6-4	Reset Type Status Register (RSTYPE)	53
6-5	PLL Control Register (PLLCTL)	54
6-6	PLL Multiplier Control Register (PLLM)	55
6-7	PLL Controller Divider 1 Register (PLLDIV1)	55
6-8	PLL Controller Divider 2 Register (PLLDIV2)	56
6-9	PLL Controller Divider 3 Register (PLLDIV3)	57
6-10	PLL Post-Divider Control Register (POSTDIV)	58
6-11	Bypass Divider Register (BPDIV)	59
6-12	PLL Controller Command Register (PLLCMD)	60
6-13	PLL Controller Status Register (PLLSTAT)	61
6-14	PLL Controller Clock Align Control Register (ALNCTL)	62
6-15	PLLDIV Ratio Change Status Register (DCHANGE)	63
6-16	Clock Enable Control Register (CKEN)	64
6-17	Clock Status Register (CKSTAT)	65
6-18	SYSCLK Status Register (SYSTAT)	66
6-19	PLL Controller Divider 5 Register (PLLDIV5)	67
7-1	TMS320DM357 DMSoC Power and Sleep Controller (PSC)	70
7-2	TMS320DM357 DMSoC Power Domain and Module Topology	71
7-3	Peripheral Revision and Class Information Register (PID)	78
7-4	Interrupt Evaluation Register (INTEVAL)	78
7-5	Module Error Pending Register 0 (mod 0-31) (MERRPR0)	79
7-6	Module Error Pending Register 1 (mod 32-40) (MERRPR1)	79
7-7	Module Error Clear Register 0 (mod 0-31) (MERRCR0)	80
7-8	Module Error Clear Register 1 (mod 32-40) (MERRCR1)	80
7-9	Power Error Pending Register (PERRPR)	81
7-10	Power Error Clear Register (PERRCR)	81
7-11	Power Domain Transition Command Register (PTCMD)	82
7-12	Power Domain Transition Status Register (PTSTAT)	82
7-13	Power Domain Status <i>n</i> Register (PDSTAT n)	83
7-14	Power Domain Control <i>n</i> Register (PDCTL n)	84
7-15	Module Status <i>n</i> Register (MDSTAT n)	85
7-16	Module Control <i>n</i> Register (MDCTL n)	86
8-1	VDD3P3V_PWDN Register	91
9-1	AINTC Functional Diagram	96
9-2	Interrupt Entry Table	97
9-3	Immediate Interrupt Disable/Enable	98
9-4	Delayed Interrupt Disable	98
9-5	Fast Interrupt Request Status Register 0 (FIQ0)	100
9-6	Fast Interrupt Request Status Register 1 (FIQ1)	100
9-7	Interrupt Request Status Register 0 (IRQ0)	101
9-8	Interrupt Request Status Register 1 (IRQ1)	101
9-9	Fast Interrupt Request Entry Address Register (FIQENTRY)	102
9-10	Interrupt Request Entry Address Register (IRQENTRY)	102
9-11	Interrupt Enable Register 0 (EINT0)	103
9-12	Interrupt Enable Register 1 (EINT1)	103

9-13	Interrupt Operation Control Register (INTCTL)	104
9-14	Interrupt Entry Table Base Address Register (EABASE).....	105
9-15	Interrupt Priority Register 0 (INTPRI0)	106
9-16	Interrupt Priority Register 1 (INTPRI1)	106
9-17	Interrupt Priority Register 2 (INTPRI2)	107
9-18	Interrupt Priority Register 3 (INTPRI3)	107
9-19	Interrupt Priority Register 4 (INTPRI4)	108
9-20	Interrupt Priority Register 5 (INTPRI5)	108
9-21	Interrupt Priority Register 6 (INTPRI6)	109
9-22	Interrupt Priority Register 7 (INTPRI7)	109
12-1	Boot Mode Functional Block Diagram	123
12-2	NAND Boot Flow	125
12-3	UART Boot Mode Handshake.....	128
12-4	HPI Boot Sequence	131

List of Tables

3-1	Exception Vector Table for ARM	24
3-2	Different Address Types in ARM System	26
3-3	ITCM/DTCM Memory Map	28
3-4	TCM Status Register Field Descriptions	28
3-5	TCM Region Setup Register Field Descriptions.....	29
3-6	ITCM/DTCM Size Encoding.....	29
5-1	System Clock Modes and Fixed Ratios for Core Clock Domains.....	36
5-2	Example PLL1 Frequencies.....	38
5-3	Example PLL2 Frequencies (Core Voltage = 1.2V)	39
5-4	Peripherals	40
5-5	Possible Clocking Modes.....	41
6-1	System PLLC1 Output Clocks	46
6-2	DDR PLLC Output Clocks.....	49
6-3	PLL and Reset Controller Module Instance Table	52
6-4	PLL and Reset Controller Registers	52
6-5	Peripheral ID Register (PID) Field Descriptions.....	53
6-6	Reset Type Status Register (RSTYPE) Field Descriptions	53
6-7	PLL Control Register (PLLCTL) Field Descriptions.....	54
6-8	PLL Multiplier Control Register (PLLM) Field Descriptions	55
6-9	PLL Controller Divider 1 Register (PLLDIV1) Field Descriptions	55
6-10	PLL Controller Divider 2 Register (PLLDIV2) Field Descriptions	56
6-11	PLL Controller Divider 3 Register (PLLDIV3) Field Descriptions	57
6-12	PLL Post-Divider Control Register (POSTDIV) Field Descriptions	58
6-13	Bypass Divider Register (BPDIV) Field Descriptions	59
6-14	PLL Controller Command Register (PLLCMD) Field Descriptions.....	60
6-15	PLL Controller Status Register (PLLSTAT) Field Descriptions.....	61
6-16	PLL Controller Clock Align Control Register (ALNCTL) Field Descriptions	62
6-17	PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions	63
6-18	Clock Enable Control Register (CKEN) Field Descriptions	64
6-19	Clock Status Register (CKSTAT) Field Descriptions	65
6-20	SYSCCLK Status Register (SYSTAT) Field Descriptions	66
6-21	PLL Controller Divider 5 Register (PLLDIV5) Field Descriptions	67
7-1	Module Configuration	72
7-2	Module States	73
7-3	IcePick Emulation Commands.....	74
7-4	PSC Interrupt Events	75
7-5	Power and Sleep Controller (PSC) Registers	77
7-6	Peripheral Revision and Class Information Register (PID) Field Descriptions	78
7-7	Interrupt Evaluation Register (INTEVAL) Field Descriptions	78
7-8	Module Error Pending Register 0 (mod 0-31) (MERRPR0) Field Descriptions	79
7-9	Module Error Pending Register 1 (mod 32-40) (MERRPR1) Field Descriptions.....	79
7-10	Module Error Clear Register 0 (mod 0-31) (MERRCR0) Field Descriptions	80
7-11	Module Error Clear Register 1 (mod 32-40) (MERRCR1) Field Descriptions.....	80
7-12	Power Error Pending Register (PERRPR) Field Descriptions.....	81
7-13	Power Error Clear Register (PERRCR) Field Descriptions	81
7-14	Power Domain Transition Command Register (PTCMD) Field Descriptions.....	82
7-15	Power Domain Transition Status Register (PTSTAT) Field Descriptions.....	82
7-16	Power Domain Status <i>n</i> Register (PDSTAT <i>n</i>) Field Descriptions	83
7-17	Power Domain Control <i>n</i> Register (PDCTL <i>n</i>) Field Descriptions	84

7-18	Module Status n Register (MDSTAT n) Field Descriptions	85
7-19	Module Control n Register (MDCTL n) Field Descriptions.....	86
8-1	Power Management Features	88
8-2	VDD3P3V_PWDN Register Field Descriptions.....	91
9-1	AINTC Interrupt Connections	95
9-2	Interrupt Controller (INTC) Registers	99
9-3	Fast Interrupt Request Status Register 0 (FIQ0) Field Descriptions.....	100
9-4	Fast Interrupt Request Status Register 1 (FIQ1) Field Descriptions.....	100
9-5	Interrupt Request Status Register 0 (IRQ0) Field Descriptions.....	101
9-6	Interrupt Request Status Register 1 (IRQ1) Field Descriptions.....	101
9-7	Fast Interrupt Request Entry Address Register (FIQENTRY) Field Descriptions	102
9-8	Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions	102
9-9	Interrupt Enable Register 0 (EINT0) Field Descriptions	103
9-10	Interrupt Enable Register 1 (EINT1) Field Descriptions	103
9-11	Interrupt Operation Control Register (INTCTL) Field Descriptions	104
9-12	Interrupt Entry Table Base Address Register (EABASE) Field Descriptions	105
9-13	Interrupt Priority Register 0 (INTPRI0) Field Descriptions.....	106
9-14	Interrupt Priority Register 1 (INTPRI1) Field Descriptions.....	106
9-15	Interrupt Priority Register 2 (INTPRI2) Field Descriptions.....	107
9-16	Interrupt Priority Register 3 (INTPRI3) Field Descriptions.....	107
9-17	Interrupt Priority Register 4 (INTPRI4) Field Descriptions.....	108
9-18	Interrupt Priority Register 5 (INTPRI5) Field Descriptions.....	108
9-19	Interrupt Priority Register 6 (INTPRI6) Field Descriptions.....	109
9-20	Interrupt Priority Register 7 (INTPRI7) Field Descriptions.....	109
10-1	TMS320DM357 DMSoC Master IDs	113
10-2	TMS320DM357 DMSoC Default Master Priorities	114
10-3	System Control Registers.....	114
11-1	Reset Types	116
11-2	Reset Pins	116
11-3	Device Configuration	118
12-1	ECC Value Location in Spare Page Bytes of 256-Byte/Page NAND Devices.....	125
12-2	ECC Value Location in Spare Page Bytes of 512-Byte/Page NAND Devices.....	125
12-3	ECC Value Location in Spare Page Bytes of 2048-Byte/Page NAND Devices	125
12-4	NAND UBL Descriptor	126
12-5	NAND IDs Supported.....	126
12-6	UBL Signatures and Special Modes	127
12-7	UART Data Sequences	129

Read This First

About This Manual

This document describes the ARM subsystem in the TMS320DM357 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 TMS320DM357 Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

[SPRUG06](#) — ***TMS320DM357 DMSoC Video Processing Back End (VPBE) User's Guide***. Describes the video processing back end (VPBE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPBE is the video encoder, on-screen display, and digital LCD controller.

[SPRUG25](#) — ***TMS320DM357 DMSoC ARM Subsystem Reference Guide***. Describes the ARM subsystem in the TMS320DM357 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 video processing subsystem, and a majority of the peripherals and external memories.

[SPRUG26](#) — ***TMS320DM357 DMSoC Universal Asynchronous Receiver/Transmitter (UART) User's Guide***. This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM357 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.

[SPRUG27](#) — ***TMS320DM357 DMSoC Inter-Integrated Circuit (I2C) Peripheral User's Guide***. Describes the inter-integrated circuit (I2C) peripheral in the TMS320DM357 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. External components attached to this 2-wire serial bus can transmit and receive up to 8-bit wide data to and from the DMSoC through the I2C peripheral. This document assumes the reader is familiar with the I2C-bus specification.

- [SPRUG28](#)** — ***TMS320DM357 DMSoC 64-Bit Timer User's Guide***. Describes the operation of the software-programmable 64-bit timer in the TMS320DM357 Digital Media System-on-Chip (DMSoC). Timer 0 and Timer 1 are used as general-purpose (GP) timers and can be programmed in 64-bit mode, dual 32-bit unchained mode, or dual 32-bit chained mode; Timer 2 is used only as a watchdog timer. The GP timer modes can be used to generate periodic interrupts or enhanced direct memory access (EDMA) synchronization events. The watchdog timer mode is used to provide a recovery mechanism for the device in the event of a fault condition, such as a non-exiting code loop.
- [SPRUG29](#)** — ***TMS320DM357 DMSoC Serial Peripheral Interface (SPI) User's Guide***. Describes the serial peripheral interface (SPI) in the TMS320DM357 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.
- [SPRUG30](#)** — ***TMS320DM357 DMSoC Host Port Interface (HPI) Reference Guide***. This document describes the host port interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The HPI provides a parallel port interface through which an external host processor can directly access the TMS320DM357 DMSoC processor's resources (configuration and program/data memories).
- [SPRUG31](#)** — ***TMS320DM357 DMSoC General-Purpose Input/Output (GPIO) User's Guide***. Describes the general-purpose input/output (GPIO) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of the input by reading the state of an internal register. When configured as an output, you can write to an internal register to control the state driven on the output pin.
- [SPRUG32](#)** — ***TMS320DM357 DMSoC Multimedia Card (MMC)/Secure Digital (SD) Card Controller User's Guide***. Describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The MMC/SD card is used in a number of applications to provide removable data storage. The MMC/SD controller provides an interface to external MMC and SD cards. The communication between the MMC/SD controller and MMC/SD card(s) is performed by the MMC/SD protocol.
- [SPRUG33](#)** — ***TMS320DM357 DMSoC Asynchronous External Memory Interface (EMIF) User's Guide***. Describes the asynchronous external memory interface (EMIF) in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.
- [SPRUG34](#)** — ***TMS320DM357 DMSoC Enhanced Direct Memory Access (EDMA) Controller User's Guide***. Describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EDMA3 controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.
- [SPRUG35](#)** — ***TMS320DM357 DMSoC Audio Serial Port (ASP) User's Guide***. Describes the operation of the audio serial port (ASP) audio interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the ASP are the AC97 and IIS modes. In addition to the primary audio modes, the ASP supports general serial port receive and transmit operation, but is not intended to be used as a high-speed interface.
- [SPRUG36](#)** — ***TMS320DM357 DMSoC Ethernet Media Access Controller (EMAC)/Management Data Input/Output (MDIO) Module User's Guide***. Discusses the ethernet media access controller (EMAC) and physical layer (PHY) device management data input/output (MDIO) module in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMAC controls the flow of packet data from the DMSoC to the PHY. The MDIO module controls PHY configuration and status monitoring.

[SPRUG37](#) — TMS320DM357 DMSoC Pulse-Width Modulator (PWM) Peripheral User's Guide.

Describes the pulse-width modulator (PWM) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

[SPRUG38](#) — TMS320DM357 DMSoC DDR2 Memory Controller User's Guide. Describes the DDR2 memory controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The DDR2 memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM devices.

[SPRUG39](#) — TMS320DM357 DMSoC Video Processing Front End (VPFE) User's Guide. Describes the video processing front end (VPFE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPFE is the preview engine, CCD controller, resizer, histogram, and hardware 3A (H3A) statistic generator.

[SPRUGH2](#) — TMS320DM357 DMSoC Peripherals Overview Reference Guide. This document provides an overview of the peripherals in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

[SPRUGH3](#) — TMS320DM357 DMSoC Universal Serial Bus Controller User's Guide. This document describes the universal serial bus (USB) controller in the TMS320DM357 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.

Introduction

Topic	Page
1.1 Block Diagram	16
1.2 ARM Subsystem in TMS320DM357 DMSoC	16

1.1 Block Diagram

The TMS320DM357 Digital Media System-on-Chip (DMSoC) consists of the following primary components and sub-systems:

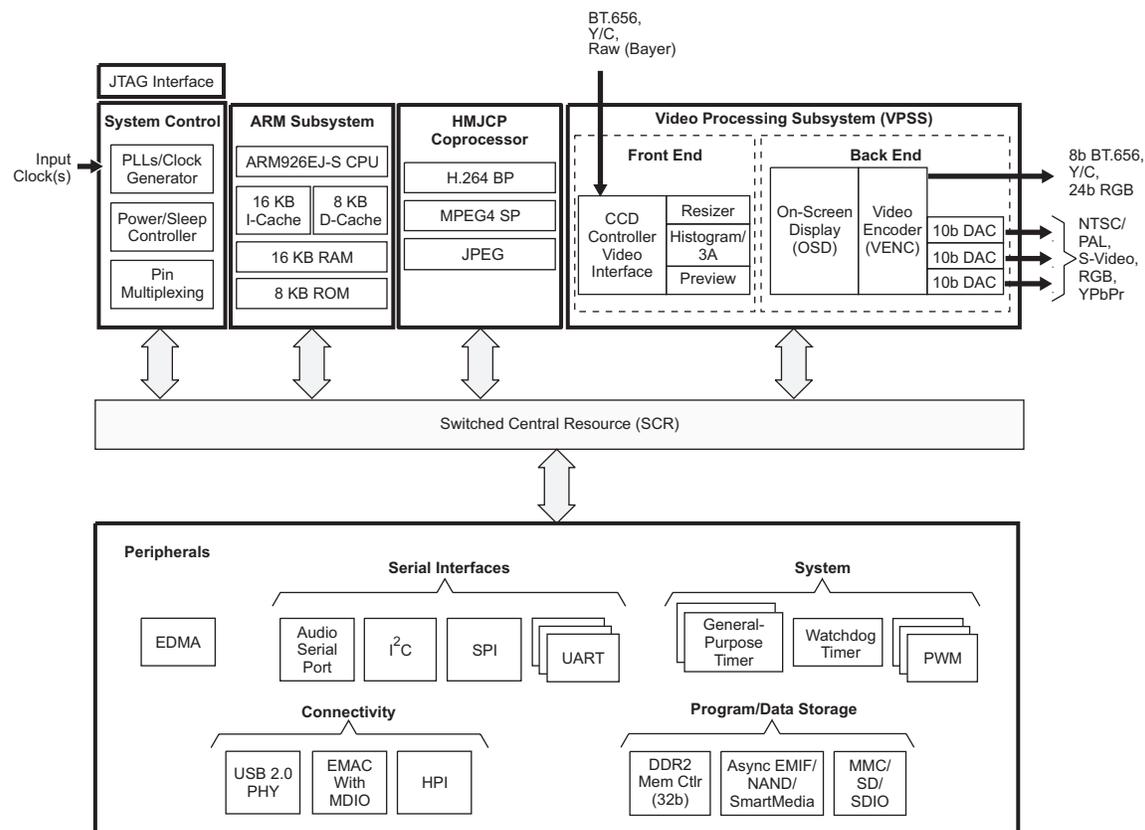
- ARM Subsystem (ARMSS), including the ARM926 RISC CPU core and associated memories
- H.264/MPEG4/JPEG coprocessor (HMJCP). A dedicated video and image coprocessor supporting MPEG4 Simple Profile (SP), D1, VGA, SIF encode/decode and JPEG encode/decode.
- Video Processing Subsystem (VPSS), including the Video Processing Front End (VPFE), Image Input and Image Processing Subsystem, and the Video Processing Back End (VPBE) Display Subsystem
- A set of I/O peripherals
- A powerful DMA Subsystem and DDR2 EMIF interface

An example block diagram (for the TMS320DM357 DMSoC) is shown in [Figure 1-1](#).

1.2 ARM Subsystem in TMS320DM357 DMSoC

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

Figure 1-1. TMS320DM357 DMSoC Block Diagram



ARM Subsystem Overview

Topic	Page
2.1 Purpose of the ARM Subsystem	18
2.2 Components of the ARM Subsystem	18
2.3 References	19

2.1 Purpose of the ARM Subsystem

The ARM Subsystem contains the components required to provide the ARM926EJ-S (ARM) master control of the TMS320DM357 DMSoC system. In general, the ARM is responsible for configuration and control of the overall DM357 DMSoC system, the VPSS Subsystem, and a majority of the peripherals and external memories.

In the DMSoC, the ARM is responsible for handling many system functions such as system-level initialization, configuration, user interface, user command execution, connectivity functions, interface, and overall system control. The ARM performs these functions because it has a larger program memory space and better context switching capability.

2.2 Components of the ARM Subsystem

The ARM Subsystem (ARMSS) in the DM357 DMSoC consists of the following components:

- ARM926EJ-S RISC processor, including:
 - Co-Processor 15 (CP15)
 - MMU
 - 16KB Instruction cache and 8KB Data cache
 - Write Buffer
- ARM Internal Memories
 - 16KB Internal RAM (32-bit wide access)
 - 8KB 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 ARM also manages/controls the following peripherals:

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

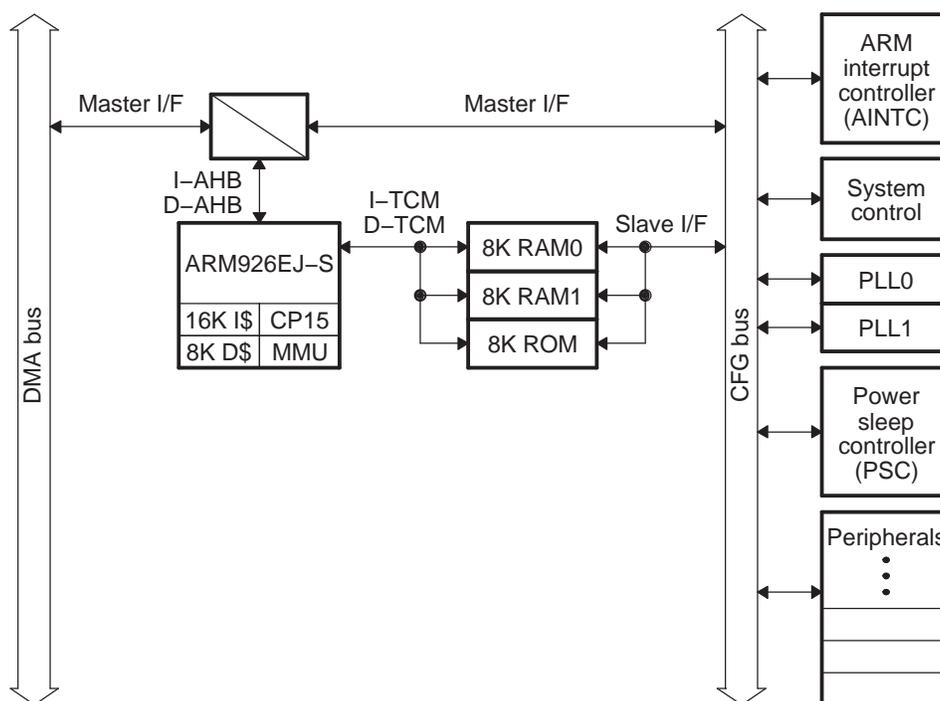
- Video Processing Back End (VPBE):
 - On-Screen Display (OSD)
 - Video Encoder Engine (VENC)

Figure 2-1 shows the functional block diagram of the DM357 DMSoC ARM Subsystem.

The DM357 DMSoC architecture uses two primary bus subsystems to transfer data within the system:

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

Figure 2-1. TMS320DM357 DMSoC ARM Subsystem Block Diagram



2.3 References

See the following DM357 DMSoC related documents for more information:

- For related documentation about the DM357 DMSoC other than the ARM core, see the *Related Documentation* section at the beginning of this document.
- For more detailed information about the ARM processor core, see ARM Ltd.'s web site (particularly, see the ARM926EJ-S Technical Reference Manual):
 - http://www.arm.com/documentation/ARMProcessor_Cores/index.html

ARM Core

Topic	Page
3.1 Introduction.....	22
3.2 Operating States/Modes.....	23
3.3 Processor Status Registers.....	23
3.4 Exceptions and Exception Vectors.....	24
3.5 The 16-BIS/32-BIS Concept	25
3.6 Co-Processor 15 (CP15).....	26
3.7 Tightly Coupled Memory	28

3.1 Introduction

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

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

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

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

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

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

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

The ARM926EJ-S core includes new signal processing extensions to enhance 16-bit fixed-point performance using a single-cycle 32×16 multiply-accumulate (MAC) unit. The ARM Subsystem also has 16 KB of internal RAM and 8 KB of internal ROM, accessible via the I-TCM and D-TCM interfaces through an arbiter. The same arbiter provides a slave DMA interface to the rest of the DM357 DMSoC. Furthermore, the ARM has DMA and CFG bus master ports via the AHB interface.

3.2 Operating States/Modes

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

The ARM can operate in the following modes:

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

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

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

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

3.3 Processor Status Registers

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

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

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

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

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

3.4 Exceptions and Exception Vectors

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

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

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

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

The default vector table is shown in [Table 3-1](#)

Table 3-1. Exception Vector Table for ARM

Vector Offset Address	Exception	Mode on entry	I Bit State on Entry	F Bit State on Entry
0h	Reset	Supervisor	Set	Set
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 processor has two instruction sets:

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

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

3.5.1 16-BIS/32-BIS Advantages

16-bit instructions operate with the standard 32-bit register configuration, allowing excellent inter-operability between 32-BIS and 16-BIS states. Each 16-bit instruction has a corresponding 32-bit instruction with the same effect on the processor model. The major advantage of a 32-bit architecture over a 16-bit architecture is its ability to manipulate 32-bit integers with single instructions, and to address a large address space efficiently. When processing 32-bit data, a 16-bit architecture takes at least two instructions to perform the same task as a single 32-bit instruction. However, not all of the code in a program processes 32-bit data (for example, code that performs character string handling), and some instructions (like branches) do not process any data at all.

If a 16-bit architecture only has 16-bit instructions, and a 32-bit architecture only has 32-bit instructions, then the 16-bit architecture has better code density overall, and has better than one half of the performance of the 32-bit architecture. Clearly, 32-bit performance comes at the cost of code density. The 16-bit instruction breaks this constraint by implementing a 16-bit instruction length on a 32-bit architecture, making the processing of 32-bit data efficient with compact instruction coding. This provides far better performance than a 16-bit architecture, with better code density than a 32-bit architecture.

The 16-BIS also has a major advantage over other 32-bit architectures with 16-bit instructions. The advantage is the ability to switch back to full 32-bit code and execute at full speed. Thus, critical loops for applications such as fast interrupts and ARM algorithms can be coded using the full 32-BIS and linked with 16-BIS code. The overhead of switching from 16-bit code to 32-bit code is folded into sub-routine entry time. Various portions of a system can be optimized for speed or for code density by switching between 16-BIS and 32-BIS execution, as appropriate.

3.6 Co-Processor 15 (CP15)

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

3.6.1 Addresses in an ARM926EJ-S System

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

Table 3-2. Different Address Types in ARM System

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

An example of the address manipulation that occurs when the ARM9EJ-S core requests an instruction is shown in [Example 3-1](#).

Example 3-1. Address Manipulation

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

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

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

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

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

3.6.2 Memory Management Unit

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

The MMU features are as follows:

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

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

3.6.3 Caches and Write Buffer

The ARM926EJ-S processor includes:

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

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

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

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

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

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

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

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

3.7 Tightly Coupled Memory

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

The DM357 DMSoC supports both instruction TCM (I-TCM) and data TCM (D-TCM). The instruction TCM is located at 0000:0000h to 0000:5FFFh. The data TCM is located at 0000:8000h to 0000:DFFFh, as shown in [Table 3-3](#).

Table 3-3. ITCM/DTCM Memory Map

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

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

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

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

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

31	Reserved	17	16	DTCM
15	Reserved	1	0	ITCM

Table 3-4. TCM Status Register Field Descriptions

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

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

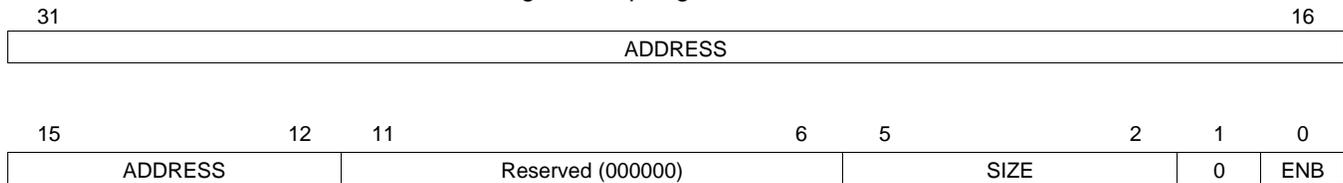


Table 3-5. TCM Region Setup Register Field Descriptions

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

Table 3-6. ITCM/DTCM Size Encoding

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

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

```
MRC p15, #0, Rd, c9, 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.

On DM357 devices, the base address of the ITCM is 0000 0000h and the size is 16KB. Hence, the address field of the ITCM register c9 should be programmed with 00000h. The memory block size field of the ITCM register c9 is fixed to the value of 5h. The memory block size field of the ITCM register c9 is read only and write has no effect.

On DM357 devices, the base address of the DTCM is 0000 8000h and the size is 32KB. The DM357 DTCM includes 16KB of RAM and 16KB of ROM. The address field of the DTCM register c9 should be programmed with 00008h. The memory block size field of the DTCM register c9 is fixed to the value of 6h. The memory block size field of the DTCM register c9 is read only and write has no effect.

Example 3-2. TMS320DM357 ITCM Register c9 Programming

```

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

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

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

```

Example 3-3. TMS320DM357 DTCM Register c9 Programming

```

DTCM_BASE_ADDR      .word      0x8000
DTCM_MASK           .word      0x0FFF

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

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

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

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

```

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

System Memory

Topic	Page
4.1 Memory Map.....	32
4.2 Memory Interfaces Overview	33

4.1 Memory Map

The TMS320DM357 DMSoC has multiple on-chip memories associated with its two processors and various subsystems. To help simplify software development, a unified memory map is used where possible to maintain a consistent view of device resources across all bus masters.

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

4.1.1 ARM Internal Memories

The ARM has access to the following ARM internal memories:

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

4.1.2 External Memories

The ARM has access to the following external memories:

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

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

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

4.1.3 Peripherals

The ARM has access to the following peripherals:

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

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

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

4.2 Memory Interfaces Overview

This section describes the different memory interfaces of DM357 DMSoC. The DM357 DMSoC supports several memory and external device interfaces, including the following:

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

4.2.1 DDR2 EMIF

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

DDR2 SDRAM plays a key role in a DM357 DMSoC-based system. Such a system is expected to require a significant amount of high-speed external memory for the following:

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

4.2.2 External Memory Interface

The DM357 DMSoC external memory interface (EMIF) provides an 8-bit or 16-bit data bus, an address bus width of up to 24-bits, and 4 dedicated chip selects, along with memory control signals. These signals are statically multiplexed between two primary memory interface modules. The primary memory interface modules are:

- AEMIF module - providing asynchronous EMIF (AEMIF) and NAND interfaces

Most of the EMIF address lines are configurable as GPIO signals if they are not required.

4.2.2.1 Asynchronous EMIF (AEMIF)

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

- The AEMIF Mode supports asynchronous devices (RAM, ROM, and NOR Flash)
- 128MB asynchronous address range over 4 chip selects (32MB each)
- Supports 8-bit or 16-bit data bus widths
- Programmable asynchronous cycle timings
- Supports extended waits
- Supports Select Strobe mode
- Supports TI HPI interface
- Supports booting DM357 DMSoC ARM processor from CS0 (SRAM / NOR Flash)

4.2.2.2 NAND (NAND, SmartMedia, xD)

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

- The NAND Mode supports NAND Flash on up to 4 asynchronous chip selects
- Supports 8-bit and 16-bit data bus widths
- Programmable cycle timings
- Performs ECC calculation
- NAND Mode also supports SmartMedia/SSFDC (Solid State Floppy Disk Controller) and xD memory cards
- ARM ROM supports booting of the DM357 DMSoC ARM processor from NAND-Flash located at CS0

Device Clocking

Topic	Page
5.1 Overview	36
5.2 Clock Domains	38

5.1 Overview

The TMS320DM357 device requires two primary reference clocks. The primary reference clocks can be either crystal input or driven by external oscillators. A 27 MHz crystal is recommended for the system PLLs, which generate the clocks for the ARM, coprocessors, peripherals, DMA, and imaging peripherals. The recommended 27 MHz input enables you to use the video DACs to drive NTSC/PAL television signals at the proper frequencies. A 24 MHz crystal is also required if you are going to use the USB peripheral.

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

There are two clocking modes:

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

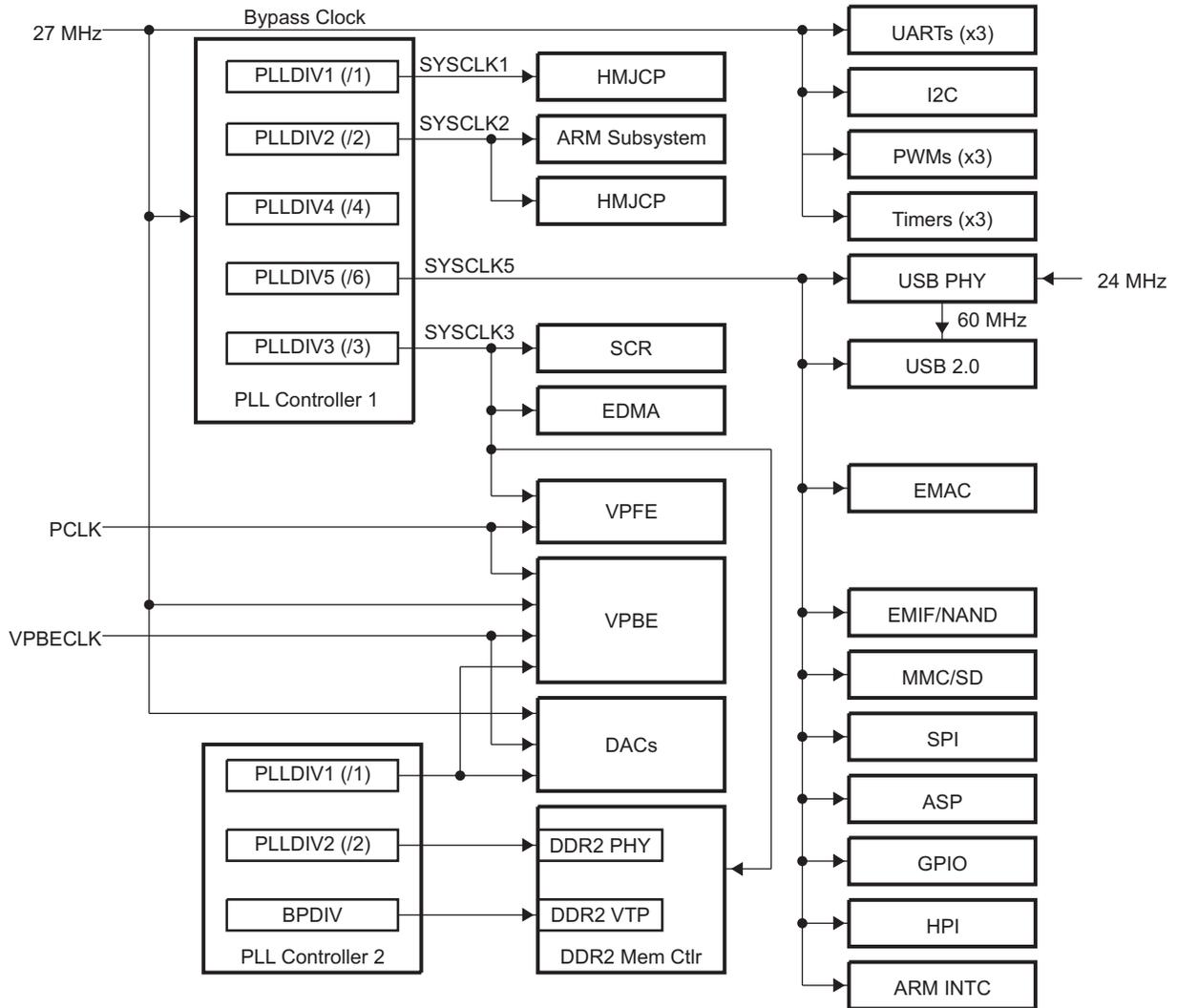
The clock of the major chip subsystems operate at fixed ratios of the primary system/PLL1 output clock frequency within each mode, as shown in [Table 5-1](#).

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

Subsystem	Core Clock Domain	Fixed Ratio vs. PLL1 Output Frequency
-	SYSCLK1	1:1
ARM	SYSCLK2	1:2
EDMA VPSS	SYSCLK3	1:3
Peripherals	SYSCLK5	1:6

Figure 5-1 shows the DM357 DMSoC clocking architecture.

Figure 5-1. Overall Clocking Diagram



5.2 Clock Domains

5.2.1 Core Domains

The core domains refer to the clock domains for all of the internal processing elements of the DM357 DMSoC, such as the EDMA. All of the core clock domains are synchronous to each other, come from a single PLL (PLL1), have aligned clock edges, and have fixed divide by ratios, as shown in [Figure 5-1](#) and [Table 5-1](#).

The entire ARM subsystem is in the SYSCLK2 domain.

5.2.2 Frequency Flexibility

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

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

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

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

Table 5-2. Example PLL1 Frequencies

PLL1 Multiplier	PLL1 VCO Frequency (MHZ)	PLL1 Output		
		Div1	Div2	Div3
15	405.0	405.0	202.5	135.0
16	432.0	432.0	216.0	144.0
17	459.0	459.0	229.5	153.0
18	486.0	486.0	243.0	162.0
19	513.0	513.0	256.5	171.0
20	540.0	540.0	270.0	180.0

5.2.3 DDR2/EMIF Clock

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

The DM357 DMSoC video DACs are capable of driving high quality progressive television displays, if driven by a 54-MHz input clock that is sourced by PLL2 (see the *TMS320DM357 DMSoC Video Processing Back End (VPBE) User's Guide (SPRUG06)* for more detailed information). This will limit the possible PLL2 settings to a multiple of 54 MHz so that the VPBE clock can be derived with a simple integer clock divider.

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

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

[Table 5-3](#) shows the possible PLL2/DDR2 clock rates and the settings that are also a multiple of 54 MHz.

Table 5-3. Example PLL2 Frequencies (Core Voltage = 1.2V)

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

5.2.4 I/O Domains

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

Peripherals can be divided into 4 groups, depending upon their clock requirements. They are shown in [Table 5-4](#).

Table 5-4. Peripherals

Peripheral Group	Peripheral Group Definition	Peripherals Contained within the Group	Frequency of Peripheral	Source of Peripheral
Fixed-Frequency Peripherals	As the name suggests, fixed-frequency peripherals have a fixed-frequency requirement. They are fed the fixed 27 MHz directly from the oscillator input.	UART I2C Timer/WDT PWM	- - - -	- - - -
Synchronous Peripherals	Synchronous peripherals have their frequencies derived from the core domain peripheral system clock frequency, which is PLL1+6. The peripheral system clock frequency changes accordingly, if the PLL1 frequency changes. Most synchronous peripherals have internal dividers so they can generate their required clock frequencies.	NAND/SM/Async EMIF - MMC/SD SPI GPIO	- - - - -	- - - - -
Asynchronous Peripherals	Asynchronous peripherals are peripherals that require an asynchronous interface due to unique clocking requirements.	Video Processing Front End (VPFE) Video Processing Back End (VPBE) USB - DDR2 Memory Controller	10-98 MHz 6.25-75 MHz 24 MHz <40 MHz 125-150 MHz	External External or Crystal or PLL2 Crystal External PLL2
Synchronous/Asynchronous Peripherals	Synchronous/Asynchronous peripherals can be run with either internally generated synchronous clocks, or externally generated asynchronous clocks, selectable by MMR bits in the peripheral.	ASP I2C	128 KHz-24.576 MHz Up to 400 KHz	Up to 25.5 MHz (PLL1+6 divided down) Up to 400 KHz (27 MHz divided down)

5.2.5 Video Processing Back End

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

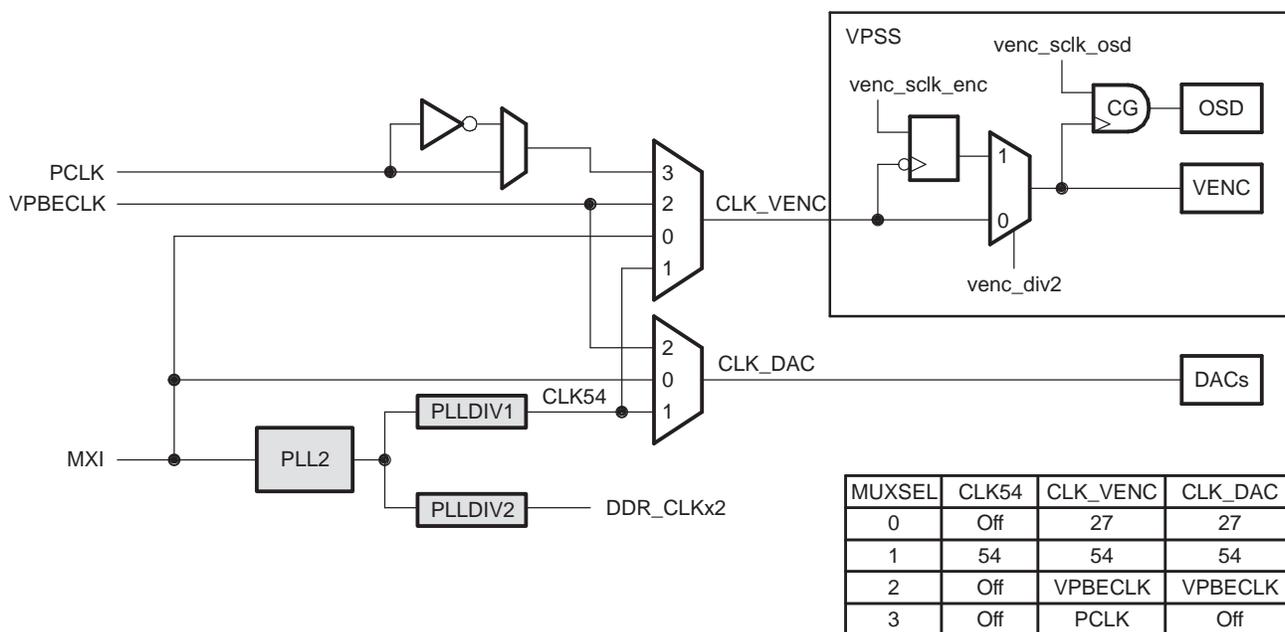
The VPBE must interface with a variety of LCDs, as well as the 4-channel DAC module. There are many different types of LCDs, which require many different specific frequencies. The range of frequencies that the pin interface needs to run is 6.25 MHz to 75 MHz.

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

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

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

Figure 5-2. VPBE/DAC Clocking



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

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

Table 5-5. Possible Clocking Modes

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

PLL Controller

Topic	Page
6.1 PLL Module	44
6.2 PLL1 Control	45
6.3 PLL2 Control	48
6.4 PLL Controller Register Map	52

6.1 PLL Module

The TMS320DM357 DMSoC has two PLL controllers that provide clocks to different parts of the system. PLL1 provides clocks (through various dividers) to most of the components of the DMSoC. PLL2 is dedicated to the DDR2 port and components for the VPSS. The reference clock is the 27 MHz crystal input, as mentioned in [Chapter 5](#).

The PLL controller provides the following:

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

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

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

Various dividers that can be used are as follows:

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

Various other controls supported are as follows:

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

6.2 PLL1 Control

PLL1 supplies the primary DM357 DMSoC system clock. Software controls the PLL1 operation through the system PLL controller 1 (PLLC1) registers (base address: 1C4 0800h). [Figure 6-1](#) shows the customization of PLL1 in the DM357 DMSoC.

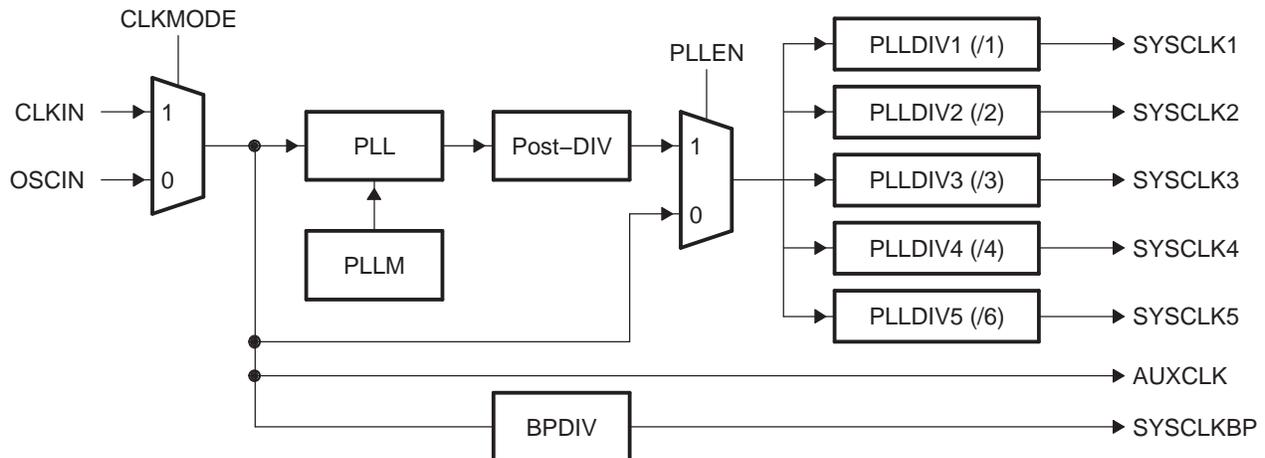
- The SYSCLK dividers are fixed (see [Table 6-1](#)).
- SYSCLKBP can be output to the CLK_OUT0 pin.
- AUXCLK is the clock provided to the fixed clock domains

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

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

Registers used in PLLC1 are listed in [Table 6-4](#).

Figure 6-1. PLL1 Structure in the TMS320DM357 DMSoC



6.2.1 Device Clock Generation

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

Table 6-1. System PLL1 Output Clocks

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

6.2.2 Steps for Changing PLL1/Core Domain Frequency

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

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

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

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

6.2.2.1 Initialization to PLL Mode from PLL Power Down

If the PLL is powered down (PLLWRDN bit in PLLCTL is set to 1), you must follow the procedure below to change PLL1 frequencies. The recommendation is to stop all peripheral operation before changing the PLL1 frequency, with the exception of the DDR2. The DDR2 operates off of the clock from PLL2.

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

13. Set the PLEN bit in PLLCTL to 1 to remove the PLL from bypass mode.

6.2.2.2 Changing PLL Multiplier

If the PLL is not powered down (PLLWRDN bit in PLLCTL is cleared to 0) and the PLL stabilization time is previously met (step 7 in [Section 6.2.2.1](#)), follow this procedure to change PLL1 multiplier. The recommendation is to stop all peripheral operation before changing the PLL multiplier, with the exception of DDR2. DDR2 operates off of the clock from PLLC2.

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

6.3 PLL2 Control

PLL2 provides the clock from which the DDR EMIF and optional VPBE clocks are derived. The DDR PLL controller (PLLC2) controls PLL2, which accepts the clock from the oscillator and also generates the various frequency clocks needed. [Figure 6-2](#) shows the customization of PLL2 in the DM357 DMSoC.

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

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

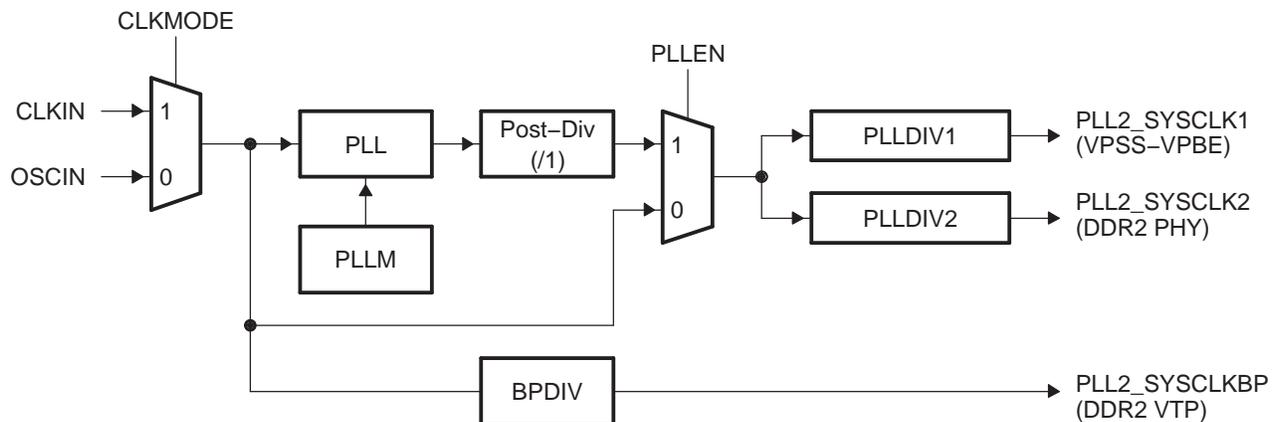
The PLL2 output clock must be divided-down to the DDR2 operating range.

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

Registers used in PLLC2 are listed in [Table 6-4](#).

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

Figure 6-2. PLL2 Structure in TMS320DM357 DMSoC



6.3.1 Device Clock Generation

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

Table 6-2. DDR PLLC Output Clocks

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

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

6.3.2 Steps for Changing PLL2 Frequency

The PLL2 is programmed similarly to the PLL1. Refer to the appropriate subsection on how to program the PLL2 clocks:

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

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

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

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

6.3.2.1 DDR2 Considerations When Modifying PLL2 Frequency

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

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

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

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

1. Leave the DDR2 memory controller in reset.
2. Program the PLL2 clocks by following the steps in the appropriate section: [Section 6.3.2.2](#), [Section 6.3.2.3](#), or [Section 6.3.2.4](#). (Discussion in [Section 6.3.2](#) explains which is the appropriate subsection).
3. Initialize the DDR2 memory controller. The steps for DDR2 memory controller initialization are found in the *TMS320DM357 DMSoC DDR2 Memory Controller User's Guide* ([SPRUG38](#)).

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

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

1. Stop DDR2 memory controller accesses and purge any outstanding requests.
2. Put the DDR2 memory in self-refresh mode and stop the DDR2 memory controller clock. The DDR2 memory controller clock shut down sequence is in the *TMS320DM357 DMSoC DDR2 Memory Controller User's Guide* ([SPRUG38](#)).
3. Program the PLL2 clocks by following the steps in the appropriate section: [Section 6.3.2.2](#), [Section 6.3.2.3](#), or [Section 6.3.2.4](#). (Discussion in [Section 6.3.2](#) explains which is the appropriate subsection).
4. Re-enable the DDR2 memory controller clock. The DDR2 memory controller clock on sequence is in the *TMS320DM357 DMSoC DDR2 Memory Controller User's Guide* ([SPRUG38](#)).

6.3.2.2 Initialization to PLL Mode from PLL Power Down

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

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

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

For information on initializing the DDR2 memory controller, see the *TMS320DM357 DMSoC DDR2 Memory Controller User's Guide* ([SPRUG38](#)).

6.3.2.3 Changing PLL Multiplier

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

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

6.3.2.4 Changing SYSCLK Dividers

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

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

6.4 PLL Controller Register Map

Table 6-3. PLL and Reset Controller Module Instance Table

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

Table 6-4 lists the memory-mapped registers for the PLL and Reset Controller. See the device-specific data manual for the memory address of these registers.

Table 6-4. PLL and Reset Controller Registers

Offset	Acronym	Register Description	Section
00h	PID	Peripheral ID Register	Section 6.4.1
E4h	RSTYPE	Reset Type Status Register	Section 6.4.2
100h	PLLCTL	PLL Control Register	Section 6.4.3
110h	PLLM	PLL Multiplier Control Register	Section 6.4.4
118h	PLLDIV1	PLL Controller Divider 1 Register	Section 6.4.5
11Ch	PLLDIV2	PLL Controller Divider 2 Register	Section 6.4.6
120h	PLLDIV3 ⁽¹⁾	PLL Controller Divider 3 Register	Section 6.4.7
128h	POSTDIV	PLL Post-Divider Control Register	Section 6.4.8
12Ch	BPDIV	Bypass Divider Register	Section 6.4.9
138h	PLLCMD	PLL Controller Command Register	Section 6.4.10
13Ch	PLLSTAT	PLL Controller Status Register	Section 6.4.11
140h	ALNCTL	PLL Controller Clock Align Control Register	Section 6.4.12
144h	DCHANGE	PLLDIV Ratio Change Status Register	Section 6.4.13
148h	CKEN	Clock Enable Control Register	Section 6.4.14
14Ch	CKSTAT	Clock Status Register	Section 6.4.15
150h	SYSTAT	SYSCLK Status Register	Section 6.4.16
164h	PLLDIV5 ⁽¹⁾	PLL Controller Divider 5 Register	Section 6.4.17

(1) not supported for PLL2

6.4.1 Peripheral ID Register (PID)

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

Figure 6-3. Peripheral ID Register (PID)

31	Reserved	24	23	16
	R-0		TYPE	R-1h
15	CLASS	8	7	0
	R-8h		REV	R-2h

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

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

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

6.4.2 Reset Type Status Register (RSTYPE)

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

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

31	Reserved				16			
	R-0							
15	Reserved			4	3	2	1	0
	R-0			SRST	MRST	XWRST	POR	
				R-0	R-0	R-0	R-0	

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

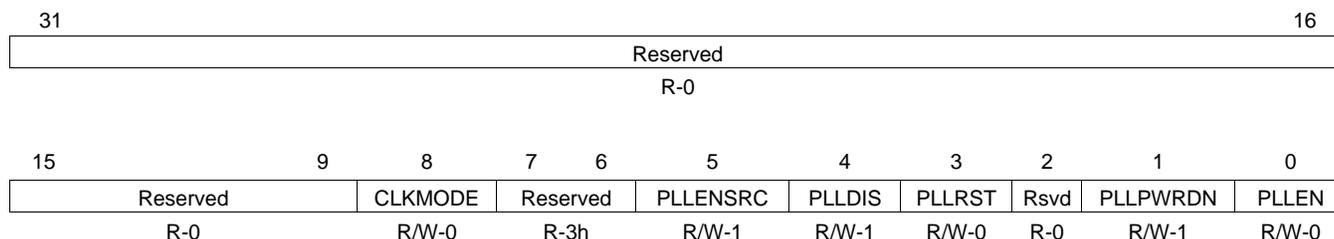
Table 6-6. Reset Type Status Register (RSTYPE) Field Descriptions

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

6.4.3 PLL Control Register (PLLCTL)

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

Figure 6-5. PLL Control Register (PLLCTL)



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

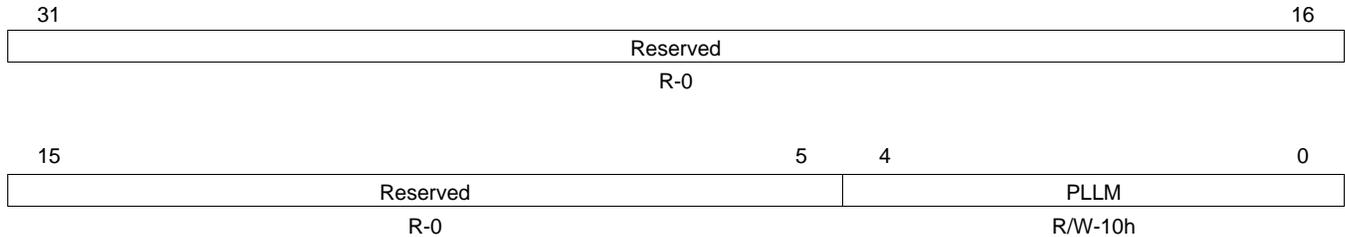
Table 6-7. PLL Control Register (PLLCTL) Field Descriptions

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

6.4.4 PLL Multiplier Control Register (PLLM)

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

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



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

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

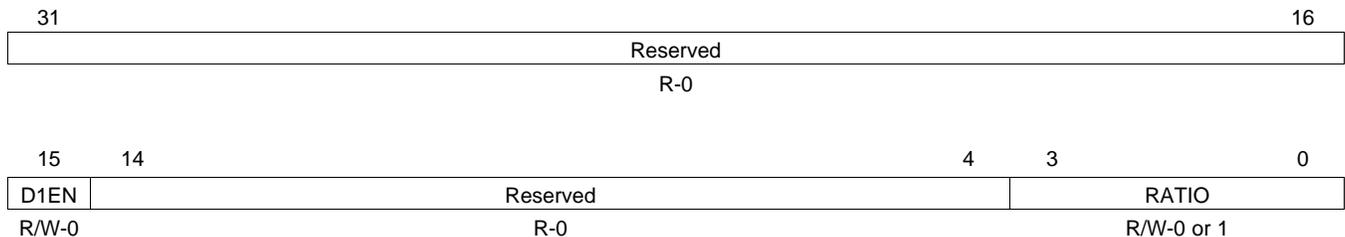
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PLLM	0-1Fh	PLL Multiplier Select. Multiplier Value = PLLM + 1. The valid range of multiplier values for a given MXI/CLKIN is defined by the minimum and maximum frequency limits on the PLL VCO frequency. See the device-specific data manual for PLL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

6.4.5 PLL Controller Divider 1 Register (PLLDIV1)

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

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

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



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

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

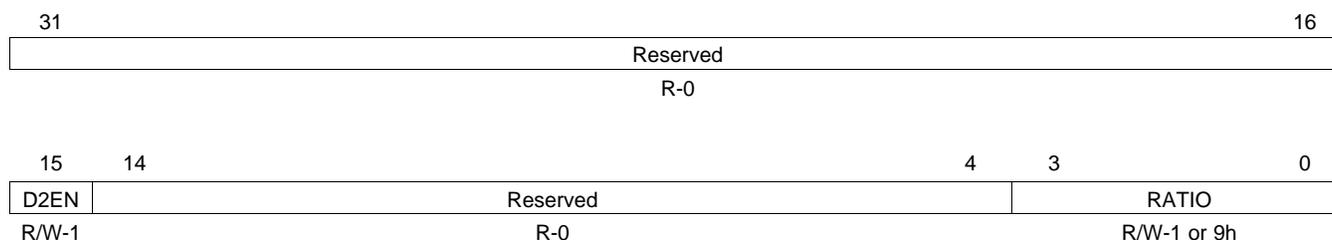
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN	0 1	Divider Enable. Disable Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 0 (PLL divide by 1); for PLL2, RATIO defaults to 1 (PLL2 divide by 2).

6.4.6 PLL Controller Divider 2 Register (PLLDIV2)

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

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

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



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

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

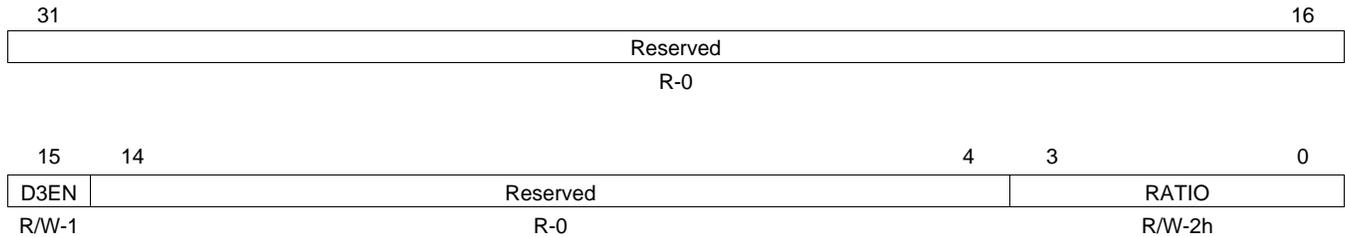
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN	0 1	Divider Enable. Disable Enable
14-4	Reserved	0	Reserved
3-0	RATIO	0-Fh	Divider ratio. Divider Value = RATIO + 1. For PLL1, RATIO defaults to 1 (PLL divide by 2); for PLL2, RATIO defaults to 9h (PLL2 divide by 10).

6.4.7 PLL Controller Divider 3 Register (PLLDIV3)

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

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

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



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

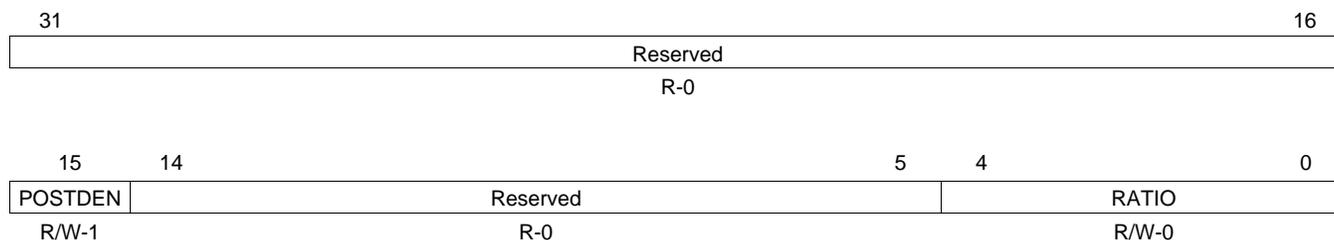
Table 6-11. PLL Controller Divider 3 Register (PLLDIV3) Field Descriptions

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

6.4.8 PLL Post-Divider Control Register (POSTDIV)

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

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



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

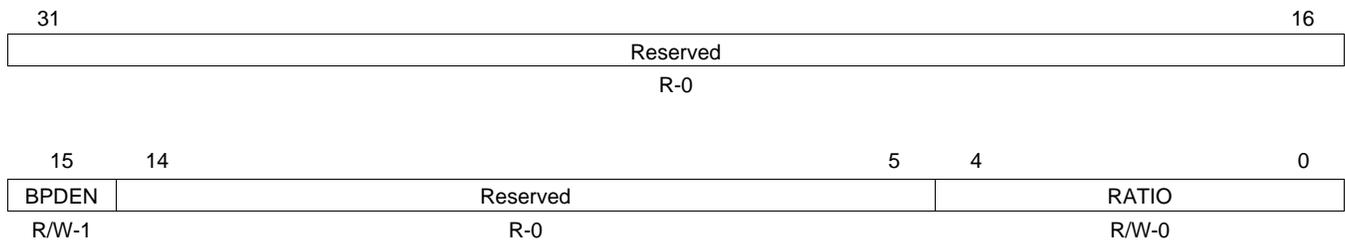
Table 6-12. PLL Post-Divider Control Register (POSTDIV) Field Descriptions

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

6.4.9 Bypass Divider Register (BPDIV)

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

Figure 6-11. Bypass Divider Register (BPDIV)



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

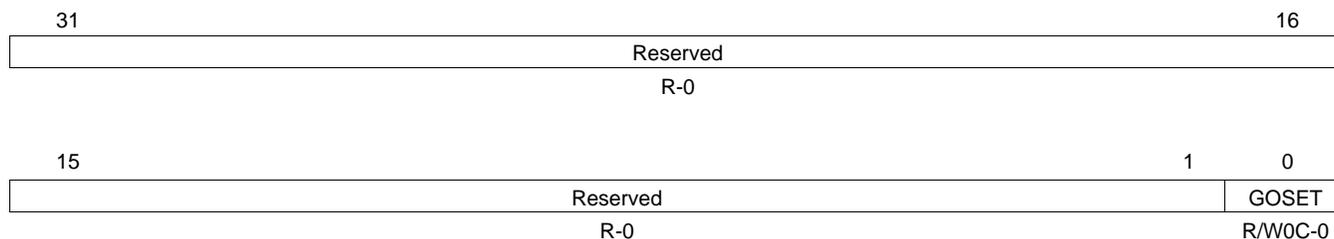
Table 6-13. Bypass Divider Register (BPDIV) Field Descriptions

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

6.4.10 PLL Controller Command Register (PLLCMD)

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

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



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

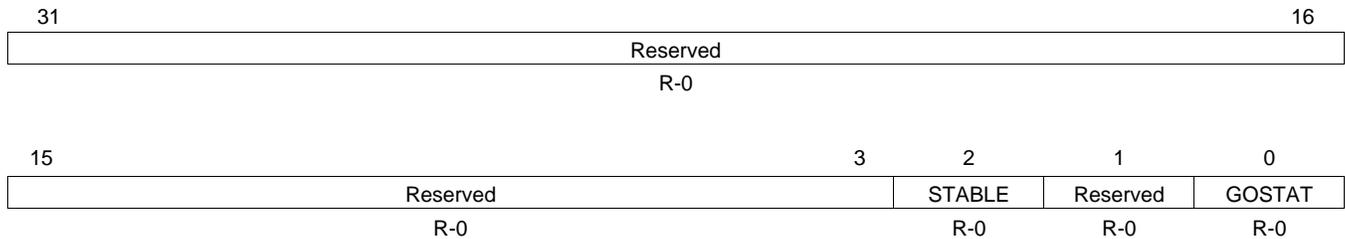
Table 6-14. PLL Controller Command Register (PLLCMD) Field Descriptions

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

6.4.11 PLL Controller Status Register (PLLSTAT)

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

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



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

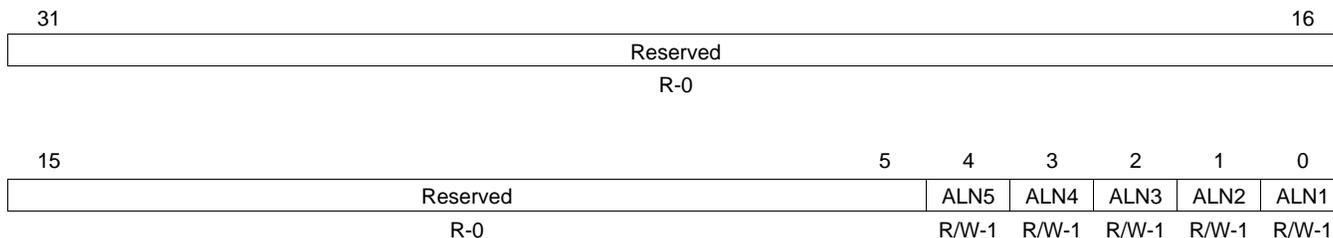
Table 6-15. PLL Controller Status Register (PLLSTAT) Field Descriptions

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

6.4.12 PLL Controller Clock Align Control Register (ALNCTL)

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

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



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

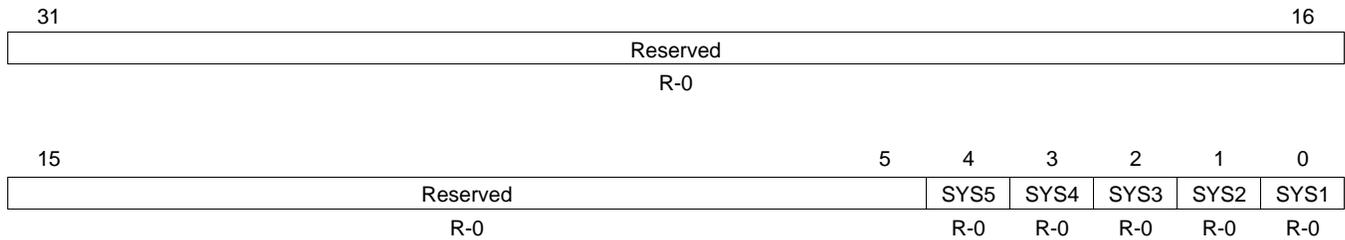
Table 6-16. PLL Controller Clock Align Control Register (ALNCTL) Field Descriptions

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

6.4.13 PLLDIV Ratio Change Status Register (DCHANGE)

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

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



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

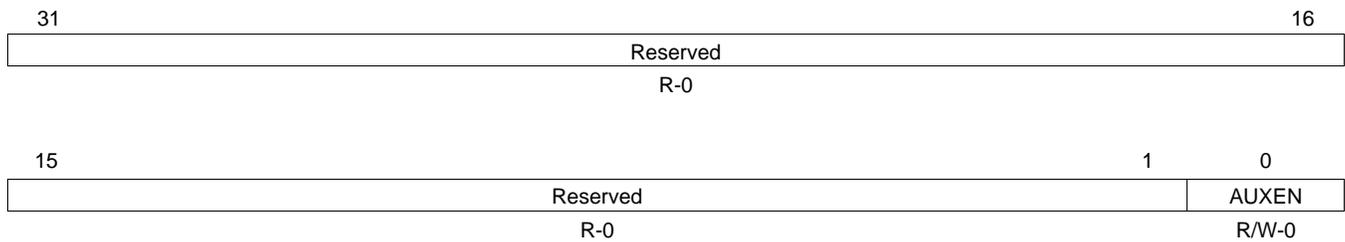
Table 6-17. PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions

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

6.4.14 Clock Enable Control Register (CKEN)

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

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



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

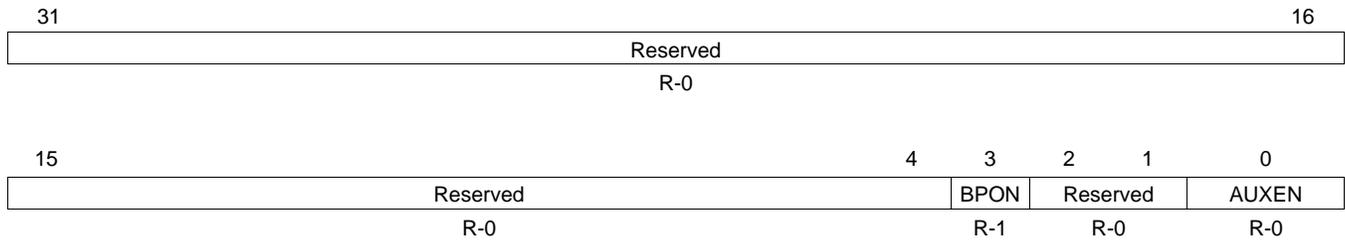
Table 6-18. Clock Enable Control Register (CKEN) Field Descriptions

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

6.4.15 Clock Status Register (CKSTAT)

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

Figure 6-17. Clock Status Register (CKSTAT)



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

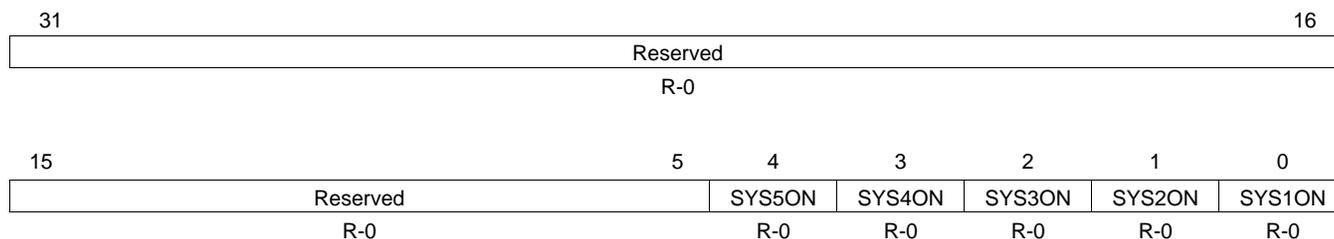
Table 6-19. Clock Status Register (CKSTAT) Field Descriptions

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

6.4.16 SYSCLK Status Register (SYSTAT)

The SYSCLK status register (SYSTAT) is shown in [Figure 6-18](#) and described in [Table 6-20](#). Indicates SYSCLK on/off status. Actual default is determined by actual clock on/off status, which depends on the DnEN bit in PLLDIVn default.

Figure 6-18. SYSCLK Status Register (SYSTAT)



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

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

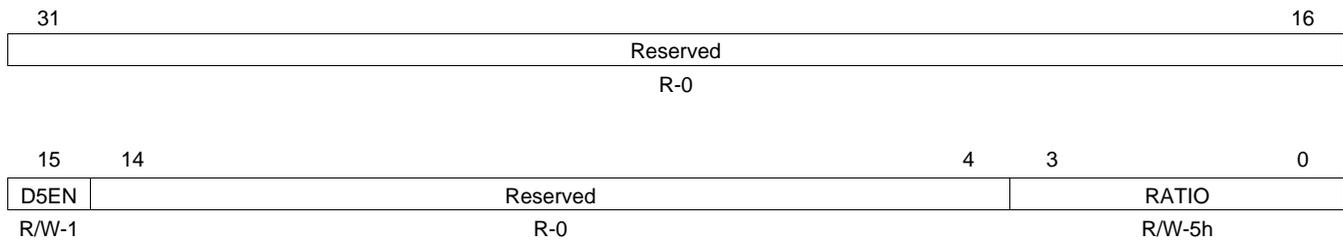
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	SYS5ON	0 1	SYSCLK5 on status. Off On
3	SYS4ON	0 1	SYSCLK4 on status. Off On
2	SYS3ON	0 1	SYSCLK3 on status. Off On
1	SYS2ON	0 1	SYSCLK2 on status. Off On
0	SYS1ON	0 1	SYSCLK1 on status. Off On

6.4.17 PLL Controller Divider 5 Register (PLLDIV5)

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

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

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



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

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

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

Power and Sleep Controller

Topic	Page
7.1 Introduction.....	70
7.2 TMS320DM357 DMSoC Power Domain and Module Topology.....	71
7.3 Power Domain and Module States Defined.....	73
7.4 Executing State Transitions	73
7.5 IcePick Emulation Support in the PSC	74
7.6 PSC Interrupts	75
7.7 PSC Registers	77

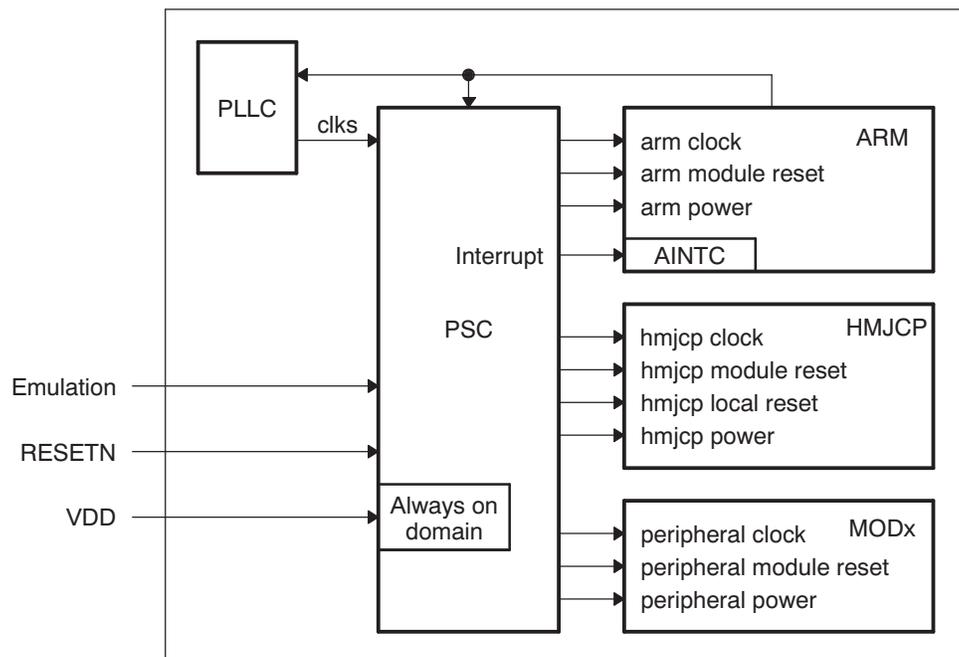
7.1 Introduction

In the TMS320DM357 DMSoC system, the Power and Sleep Controller (PSC) is responsible for managing transitions of system power on/off, clock on/off, and reset. A block diagram of the PSC is shown in [Figure 7-1](#). Many of the operations of the PSC are transparent to software, such as power-on and hard reset operations. However, the PSC provides you with an interface to control several important power, clock, and reset operations. The power, clock, and reset operations are the focus of this chapter.

The PSC includes the following features:

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

Figure 7-1. TMS320DM357 DMSoC Power and Sleep Controller (PSC)



7.2 TMS320DM357 DMSoC Power Domain and Module Topology

The DM357 DMSoC system includes two separate power domains and multiple separate modules, as shown in Figure 7-2 and summarized in Table 7-1. The AlwaysOn power domain is always on when the chip is on. The AlwaysOn domain is powered by the CV_{DD} pins of the DM357 DMSoC (see the device-specific data manual).

Figure 7-2. TMS320DM357 DMSoC Power Domain and Module Topology

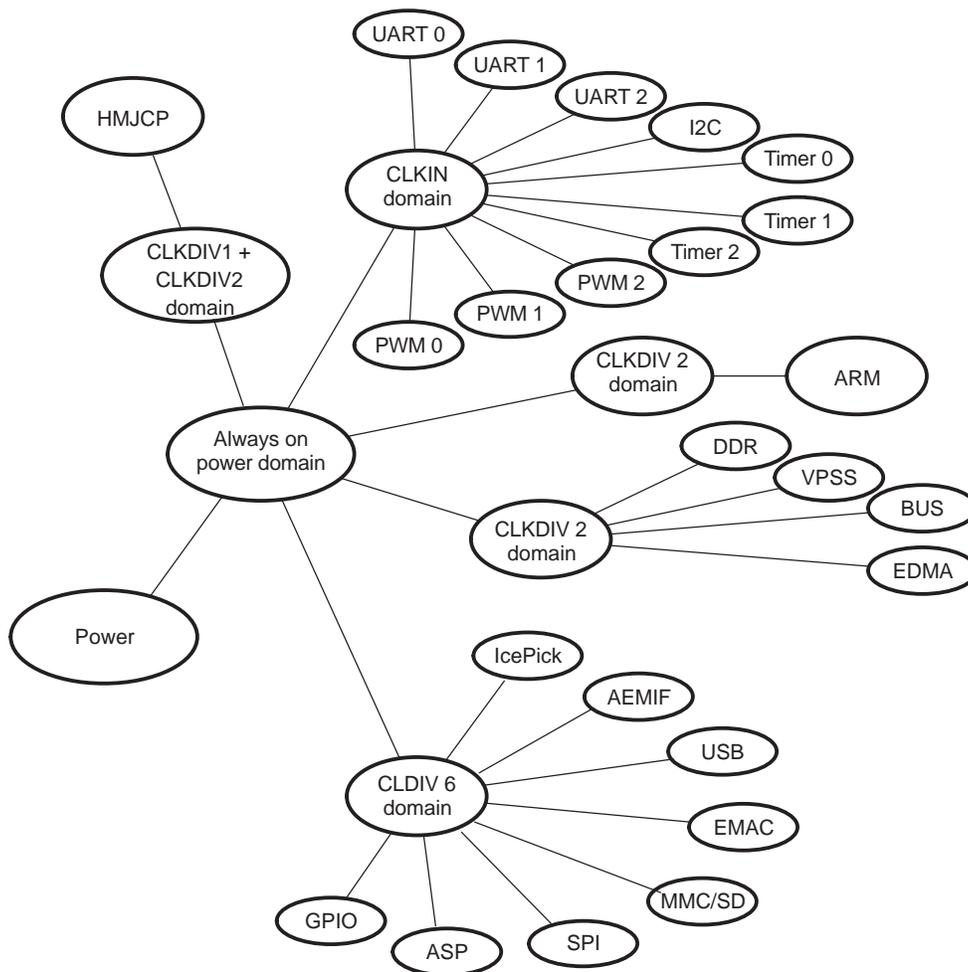


Table 7-1. Module Configuration

Module Number	Module Name	Power Domain	Default States		
			Power Domain State	Module State	Local Reset State
0	VPSS (master)	AlwaysOn	ON	Disable	-
1	VPSS (slave)	AlwaysOn	ON	Disable	-
2	CC	AlwaysOn	ON	BTSEL = 00: Enable (NAND) BTSEL = 01: Enable (NOR)	-
3	TC0	AlwaysOn	ON	BTSEL = 10: Reserved	-
4	TC1	AlwaysOn	ON	BTSEL = 11: Enable (UART)	-
5	EMAC	AlwaysOn	ON	Disable	-
6	EMAC	AlwaysOn	ON	Disable	-
7-8	Reserved	-	-	-	-
9	USB	AlwaysOn	ON	Disable	-
10-11	Reserved	-	-	-	-
12	HPI	AlwaysOn	ON	Disable	-
13	DDR EMIF	AlwaysOn	ON	BTSEL = 11: Disable (UART)	-
14	AEMIF	AlwaysOn	ON	BTSEL = 00: Enable (NAND) BTSEL = 01: Enable (NOR) BTSEL = 10: Reserved BTSEL = 11: Enable (UART)	-
15	MMC/SD	AlwaysOn	ON	Disable	-
17	ASP	AlwaysOn	ON	Disable	-
18	I2C	AlwaysOn	ON	Disable	-
19	UART0	AlwaysOn	ON	BTSEL = 00: Disable (NAND) BTSEL = 01: Disable (NOR) BTSEL = 10: Reserved BTSEL = 11: Enable (UART)	-
20	UART1	AlwaysOn	ON	Disable	-
21	UART2	AlwaysOn	ON	Disable	-
22	SPI	AlwaysOn	ON	Disable	-
23	PWM0	AlwaysOn	ON	Disable	-
24	PWM1	AlwaysOn	ON	Disable	-
25	PWM2	AlwaysOn	ON	Disable	-
26	GPIO	AlwaysOn	ON	Disable	-
27	TIMER0	AlwaysOn	ON	Enable	-
28	TIMER1	AlwaysOn	ON	Disable	-
29	TIMER2	AlwaysOn	ON	Enable	-
30	System Module	AlwaysOn	ON	Enable	-
31	ARM	AlwaysOn	ON	Enable	-
32 - 38	Internal Bus	AlwaysOn	ON	Enable	-
39-40	Reserved	-	-	-	-

7.3 Power Domain and Module States Defined

Table 7-2 shows the state of each module after chip power-on/hard reset. These states are defined in the following sections.

Table 7-2. Module States

Module State	Module Reset	Module Clock	Module State Definition
Enable	De-asserted	On	A module in the enable state has its module reset de-asserted and it has its clock on. This is the normal run-time state for a given module.
Disable	De-asserted	Off	A module in the disable state has its module reset de-asserted and it has its clock off. This state is typically used for disabling a module clock to save power. The DM357 DMSoC is designed in full static CMOS, so when you stop a module clock, it retains the module's state. When the clock is restarted, the module resumes operating from the stopping point.
SyncReset	Asserted	On	A module in the SyncReset state has its module reset asserted and it has its clock on. After initial power-on, most modules are in the SyncRst state by default (see Table 7-1). Generally, software is not expected to initiate this state.
SwRstDisable	Asserted	Off	A module in the SwResetDisable state has its module reset asserted and it has its clock set to off. Generally, software is not expected to initiate this state.

7.3.1 Power Domain States

In the DM357 DMSoC, 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, SyncReset, or SwRstDisable. These four states correspond to combinations of module reset asserted or de-asserted and module clock on or off, as shown in Table 7-2.

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

For more information on the DM357 DMSoC power management, see Chapter 8.

7.4 Executing State Transitions

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

7.4.1 Module State Transitions

This section describes the procedure for transitioning the module state.

Note: The following procedure is directly applicable for all modules.

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

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

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

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

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

Note: The EMURSTIE bit in MDCTL n is also used for PSC emulation features. The emulation features are described in [Section 7.5](#).

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

7.5 IcePick Emulation Support in the PSC

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

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

Table 7-3. IcePick Emulation Commands

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

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

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

7.6 PSC Interrupts

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

7.6.1 Interrupt Events

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

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

These interrupt events are summarized in [Table 7-4](#) and described in more detail in this section.

Table 7-4. PSC Interrupt Events

Interrupt Enable Bits		
Control Register	Enable Bit	Interrupt Condition
PDCTL n	EMUIHBIE	Interrupt occurs when the emulation alters the power domain state
MDCTL n	EMUIHBIE	Interrupt occurs when the emulation alters the module state
MDCTL n	EMURSTIE	Interrupt occurs when the emulation alters the module's local reset

7.6.1.1 Module State Emulation Events

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

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

7.6.1.2 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 n . In particular, a module local reset emulation event occurs under the following conditions:

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

7.6.2 Interrupt Registers

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

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

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

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

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

See [Section 7.7](#) for complete descriptions of all PSC registers.

7.6.3 Interrupt Handling

Handle the PSC interrupts as described in the following procedure:

First, enable the interrupt.

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

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

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

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

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

7.7 PSC Registers

[Table 7-5](#) lists the memory-mapped registers for the PSC. See the device-specific data manual for the memory address of these registers.

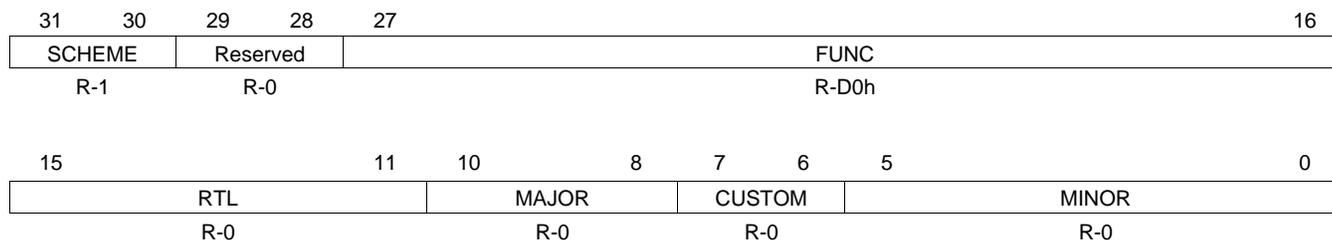
Table 7-5. Power and Sleep Controller (PSC) Registers

Offset	Register	Description	Section
0h	PID	Peripheral Revision and Class Information Register	Section 7.7.1
18h	INTEVAL	Interrupt Evaluation Register	Section 7.7.2
40h	MERRPR0	Module Error Pending Register 0 (module 0-31)	Section 7.7.3
44h	MERRPR1	Module Error Pending Register 1 (module 32-63)	Section 7.7.4
50h	MERRCR0	Module Error Clear Register 0 (module 0-31)	Section 7.7.5
54h	MERRCR1	Module Error Clear Register 1 (module 32-63)	Section 7.7.6
60h	PERRPR	Power Error Pending Register	Section 7.7.7
68h	PERRCR	Power Error Clear Register	Section 7.7.8
120h	PTCMD	Power Domain Transition Command Register	Section 7.7.9
128h	PTSTAT	Power Domain Transition Status Register	Section 7.7.10
200h	PDSTAT n	Power Domain Status n Register	Section 7.7.11
300h	PDCTL n	Power Domain Control n Register	Section 7.7.12
800h	MDSTAT n	Module Status n Register (modules 0-40)	Section 7.7.13
A00h	MDCTL n	Module Control n Register (modules 0-40)	Section 7.7.14

7.7.1 Peripheral Revision and Class Information Register (PID)

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

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



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

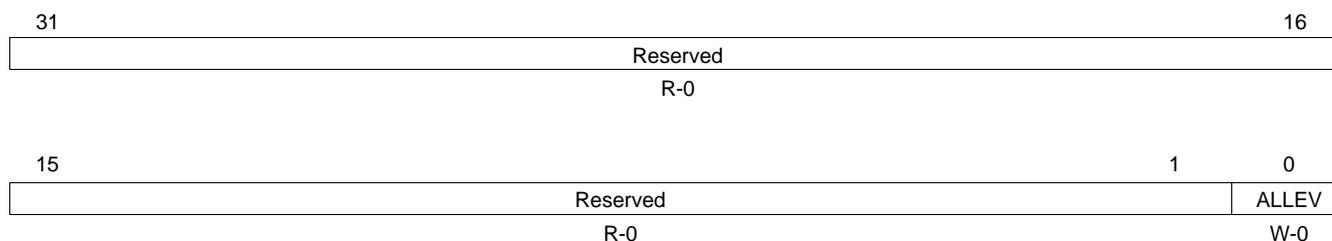
Table 7-6. Peripheral Revision and Class Information Register (PID) Field Descriptions

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

7.7.2 Interrupt Evaluation Register (INTEVAL)

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

Figure 7-4. Interrupt Evaluation Register (INTEVAL)



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

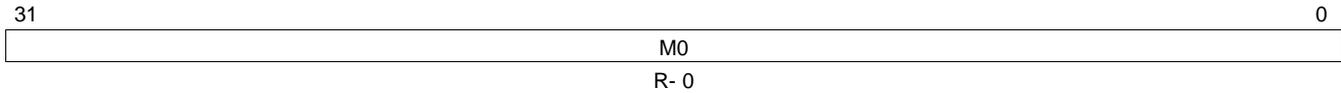
Table 7-7. Interrupt Evaluation Register (INTEVAL) Field Descriptions

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

7.7.3 Module Error Pending Register 0 (mod 0-31) (MERRPR0)

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

Figure 7-5. Module Error Pending Register 0 (mod 0-31) (MERRPR0)



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

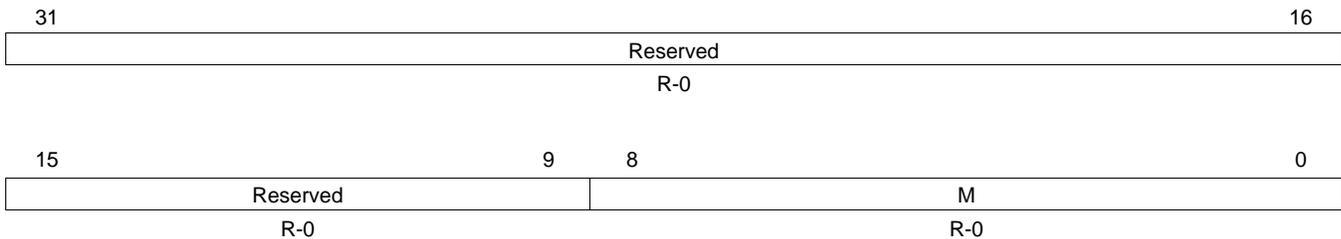
Table 7-8. Module Error Pending Register 0 (mod 0-31) (MERRPR0) Field Descriptions

Bit	Field	Value	Description
31-0	M0[n]	0	Module interrupt status bit for modules 0-31. Power domain interrupt <i>n</i> is not active.
		1	Power domain interrupt <i>n</i> is active.

7.7.4 Module Error Pending Register 1 (mod 32-40) (MERRPR1)

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

Figure 7-6. Module Error Pending Register 1 (mod 32-40) (MERRPR1)



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

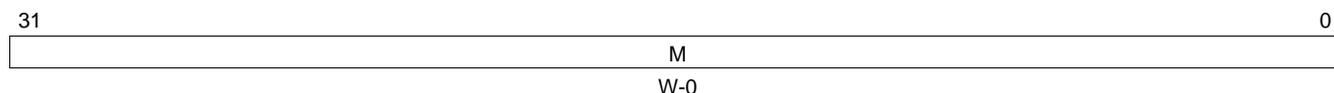
Table 7-9. Module Error Pending Register 1 (mod 32-40) (MERRPR1) Field Descriptions

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

7.7.5 Module Error Clear Register 0 (mod 0-31) (MERRCR0)

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

Figure 7-7. Module Error Clear Register 0 (mod 0-31) (MERRCR0)



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

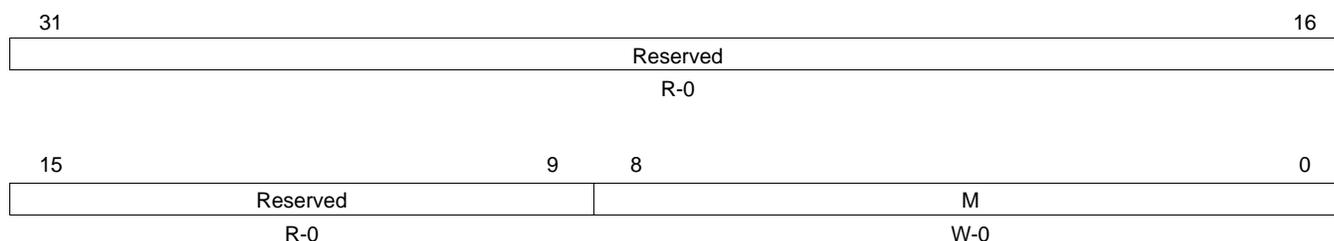
Table 7-10. Module Error Clear Register 0 (mod 0-31) (MERRCR0) Field Descriptions

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

7.7.6 Module Error Clear Register 1 (mod 32-40) (MERRCR1)

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

Figure 7-8. Module Error Clear Register 1 (mod 32-40) (MERRCR1)



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

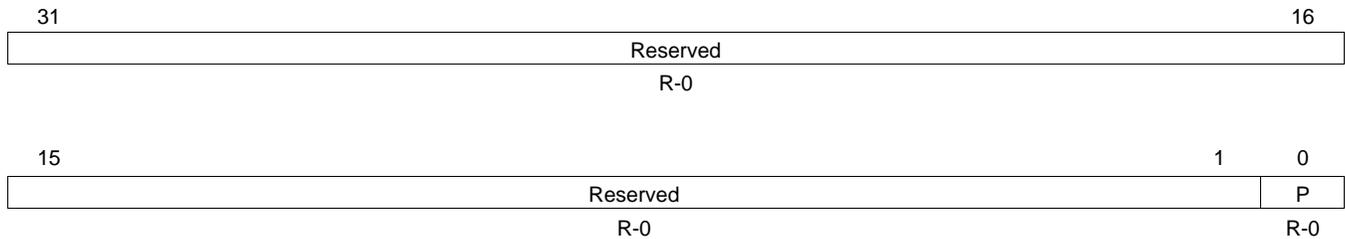
Table 7-11. Module Error Clear Register 1 (mod 32-40) (MERRCR1) Field Descriptions

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

7.7.7 Power Error Pending Register (PERRPR)

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

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



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

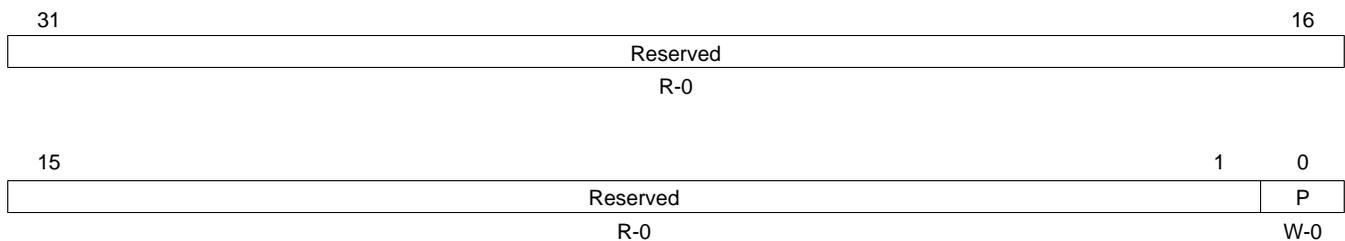
Table 7-12. Power Error Pending Register (PERRPR) Field Descriptions

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

7.7.8 Power Error Clear Register (PERRCR)

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

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



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

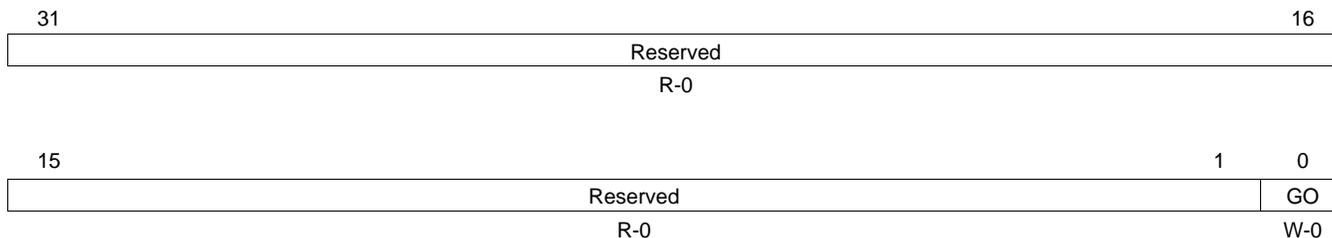
Table 7-13. Power Error Clear Register (PERRCR) Field Descriptions

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

7.7.9 Power Domain Transition Command Register (PTCMD)

The power domain transition command register (PTCMD) is shown in [Figure 7-11](#) and described in [Table 7-14](#).

Figure 7-11. Power Domain Transition Command Register (PTCMD)



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

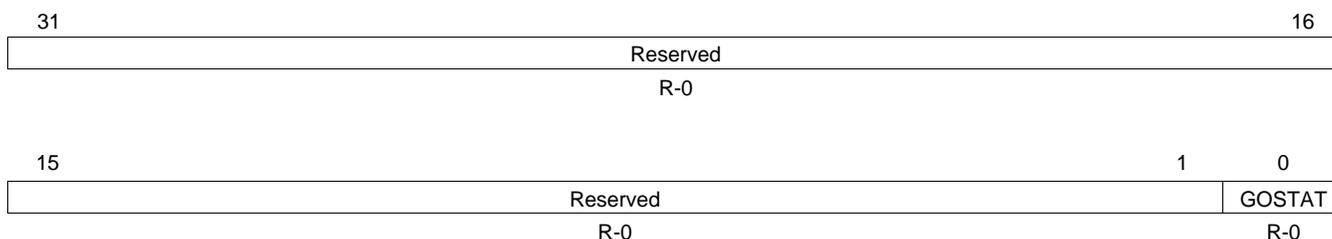
Table 7-14. Power Domain Transition Command Register (PTCMD) Field Descriptions

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

7.7.10 Power Domain Transition Status Register (PTSTAT)

The power domain transition status register (PTSTAT) is shown in [Figure 7-12](#) and described in [Table 7-15](#).

Figure 7-12. Power Domain Transition Status Register (PTSTAT)



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

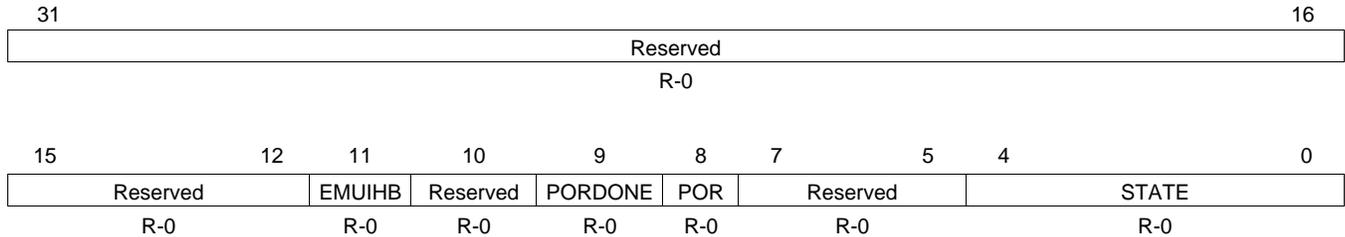
Table 7-15. Power Domain Transition Status Register (PTSTAT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSTAT	0 1	Always On Power domain transition status. No transition in progress. Modules in Always On power domain are transitioning. Always On Power domain is transitioning.

7.7.11 Power Domain Status *n* Register (PDSTAT*n*)

The power domain status *n* register (PDSTAT*n*) is shown in [Figure 7-13](#) and described in [Table 7-16](#).

Figure 7-13. Power Domain Status *n* Register (PDSTAT*n*)



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

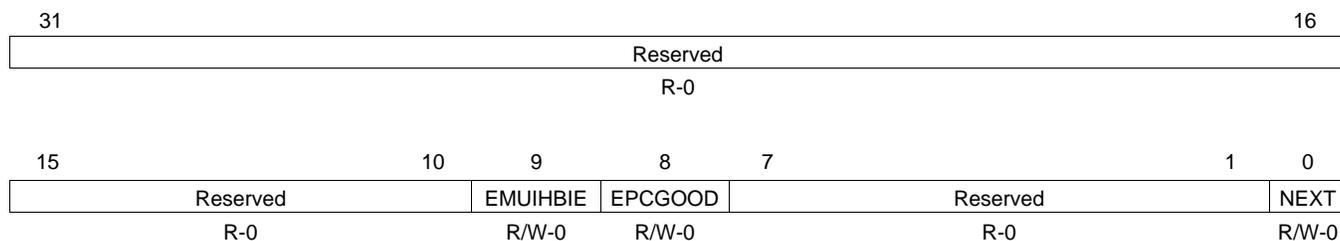
Table 7-16. Power Domain Status *n* Register (PDSTAT*n*) Field Descriptions

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

7.7.12 Power Domain Control n Register (PDCTL n)

The power domain control n register (PDCTL n) is shown in [Figure 7-14](#) and described in [Table 7-17](#).

Figure 7-14. Power Domain Control n Register (PDCTL n)



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

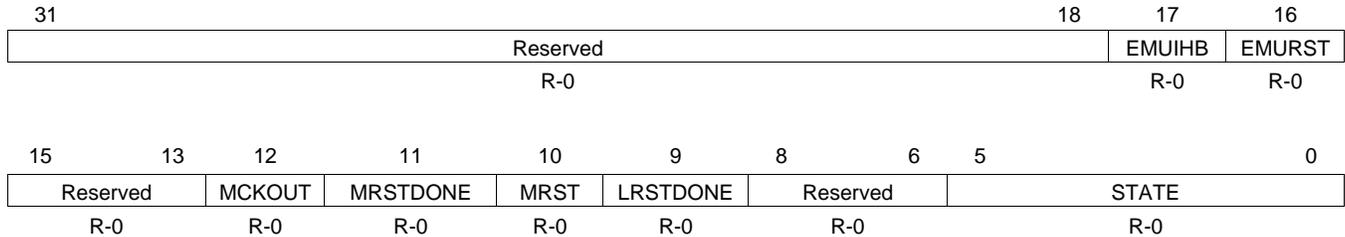
Table 7-17. Power Domain Control n Register (PDCTL n) Field Descriptions

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

7.7.13 Module Status n Register (MDSTAT0-MDSTAT40)

The module status n register (MDSTAT n) is shown in [Figure 7-15](#) and described in [Table 7-18](#).

Figure 7-15. Module Status n Register (MDSTAT n)



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

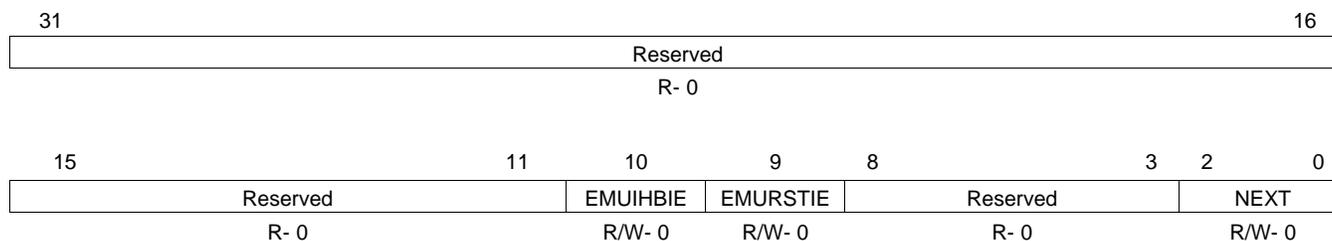
Table 7-18. Module Status n Register (MDSTAT n) Field Descriptions

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

7.7.14 Module Control n Register (MDCTL0-MDCTL40)

The module control n register (MDCTL n) is shown in [Figure 7-16](#) and described in [Table 7-19](#).

Figure 7-16. Module Control n Register (MDCTL n)



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

Table 7-19. Module Control n Register (MDCTL n) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	EMUIHBIE	0	Interrupt enable for emulation alters module state. Disable interrupt.
		1	Enable interrupt.
9	EMURSTIE	0	Interrupt enable for emulation alters reset. Disable interrupt.
		1	Enable interrupt.
8-3	Reserved	0	Reserved
2-0	NEXT	0-3h	Module next state.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

Power Management

Topic	Page
8.1 Overview	88
8.2 PSC and PLLC Overview	88
8.3 Clock Management	89
8.4 ARM Sleep Mode Management	90
8.5 I/O Management	90
8.6 VDD3P3V_PWDN Register	91
8.7 USB Phy Power Down	91
8.8 Video DAC Power Down.....	91

8.1 Overview

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

The TMS320DM357 DMSoC has several means of managing the power consumption, as detailed in the following sections. There is extensive use of automatic clock gating in the design as well as software-controlled module clock gating to not only reduce the clock tree power, but to also reduce module power by basically freezing its state while not operating. Clock management enables you to slow the clocks down on the chip in order to reduce switching power. In particular, the DM357 DMSoC includes all of the power management features described in [Table 8-1](#).

Table 8-1. Power Management Features

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

8.2 PSC and PLLC Overview

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

8.3 Clock Management

8.3.1 Module Clock ON/OFF

The module clock on/off feature allows software to disable clocks to module individually, in order to reduce the module's active power consumption to 0. The DM357 DMSoC is designed in full static CMOS; thus, when a module clock stops, the module's state is preserved. When the clock is restarted, the module resumes operating from the stopping point.

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

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

The power and sleep controller (PSC) controls module clock gating. The procedure to turn module clocks on/off is described in [Chapter 7](#).

8.3.2 Module Clock Frequency Scaling

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

[Chapter 5](#) and [Chapter 6](#) describe the how to program the PLL frequency and the frequency constraints.

8.3.3 PLL Bypass and Power Down

You can bypass the PLLs in the DM357 DMSoC. Bypassing the PLLs sends the PLL reference clock to the post dividers of the PLLC instead of to the PLL VCO output. The PLL reference clock is typically at 27 MHz; therefore, you can use this mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity. Furthermore, you can power-down the PLL when bypassing it to save additional active power.

[Chapter 5](#) and [Chapter 6](#) describe PLL bypass and PLL power down.

8.4 ARM Sleep Mode Management

8.4.1 ARM Wait-For-Interrupt Sleep Mode

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

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

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

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

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

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

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

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

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

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

8.5 I/O Management

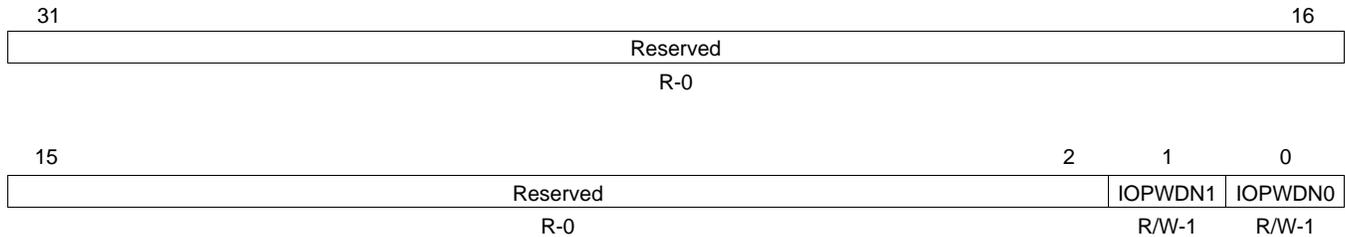
8.5.1 3.3 V I/O Power-Down

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

8.6 VDD3P3V_PWDN Register

The VDD3P3V_PWDN register is shown in [Figure 8-1](#) and described in [Table 8-2](#).

Figure 8-1. VDD3P3V_PWDN Register



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

Table 8-2. VDD3P3V_PWDN Register Field Descriptions

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

8.7 USB Phy Power Down

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

8.8 Video DAC Power Down

The DM357 DMSoC video processing back end (VPBE) includes four video digital-to-analog converters (DACs) to drive analog television displays. The Video Encoder (VENC) module of the VPBE includes registers for enabling/disabling the DACs. You can use the VIE bit in VMOD to force the analog output of the 4 DACs to a low level, regardless of the video signal. Furthermore, you can use the DAPD[3:0] bits in DACTST to disable each DAC independently. See the *TMS320DM357 DMSoC Video Processing Back End (VPBE) User's Guide* ([SPRUG06](#)) for register descriptions and more detailed information on DAC power-down.

Interrupt Controller

Topic	Page
9.1 Introduction.....	94
9.2 Interrupt Mapping.....	94
9.3 INTC Methodology.....	95
9.4 INTC Registers.....	99

9.1 Introduction

The TMS320DM357 DMSoC ARM interrupt controller (AINTC) has the following features:

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

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

9.2 Interrupt Mapping

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

[Table 9-1](#) shows the connection of device interrupts to the ARM.

Table 9-1. AINTC Interrupt Connections

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

9.3 INTC Methodology

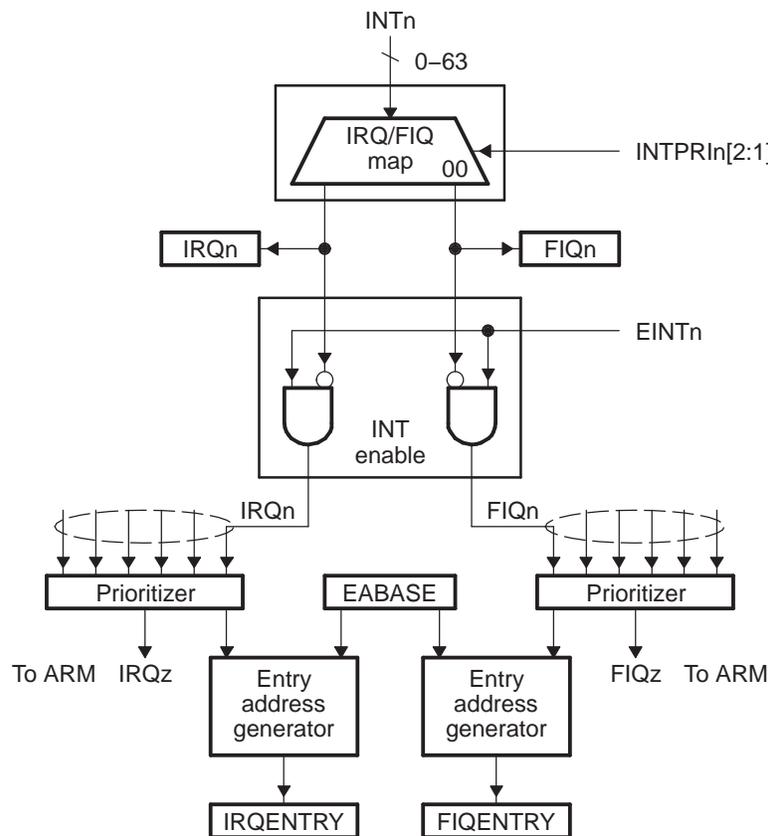
INTC methodology is illustrated in [Figure 9-1](#) and described below.

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

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

- When an interrupt from any interrupt channel occurs (for which interrupt is enabled), an IRQ or FIQ interrupt generates to the ARM926 core (depending on whether the interrupt channel is mapped to IRQ or FIQ interrupt). The ARM then branches to the IRQ or FIQ interrupt routine.

- The 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 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 9-1. AINTC Functional Diagram


9.3.1 Interrupt Mapping

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

9.3.2 Interrupt Prioritization

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

9.3.3 Vector Table Entry Address Generation

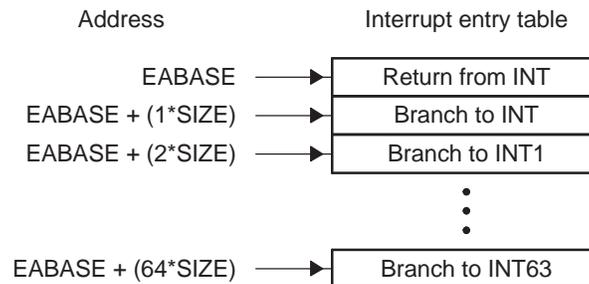
To help speed up the ISR, the AINTC provides two vectors into the ARM's interrupt entry table, which correspond to the highest priority effective IRQ and FIQ interrupts. This vector is generated by modifying a base address with a priority index. The priority index takes the size of each interrupt entry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

$$\text{IRQENTRY} = \text{EABASE} + ((\text{highest priority IRQ EVT\#} + 1) \times \text{SIZE})$$

$$\text{FIQENTRY} = \text{EABASE} + ((\text{highest priority FIQ EVT\#} + 1) \times \text{SIZE})$$

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

Figure 9-2. Interrupt Entry Table



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

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

1. For the FIQENTRY:

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

2. For the IRQENTRY:

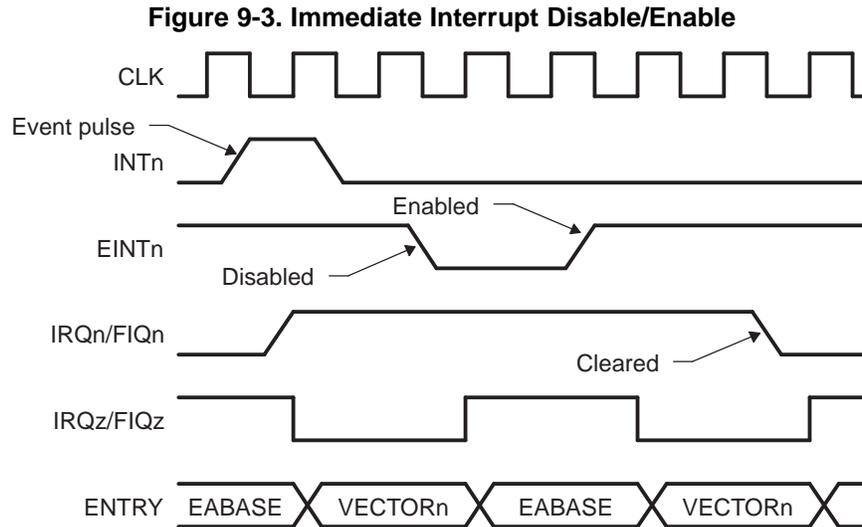
- If IERAW is 0, IRQENTRY reflects the state of the highest priority pending enabled IRQ interrupt. If the active IRQ interrupt is cleared in IRQn, then IRQENTRY is immediately updated with the vector of the next highest priority pending enabled IRQ interrupt.
- If IERAW is 1, IRQENTRY reflects the state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

9.3.4 Clearing Interrupts

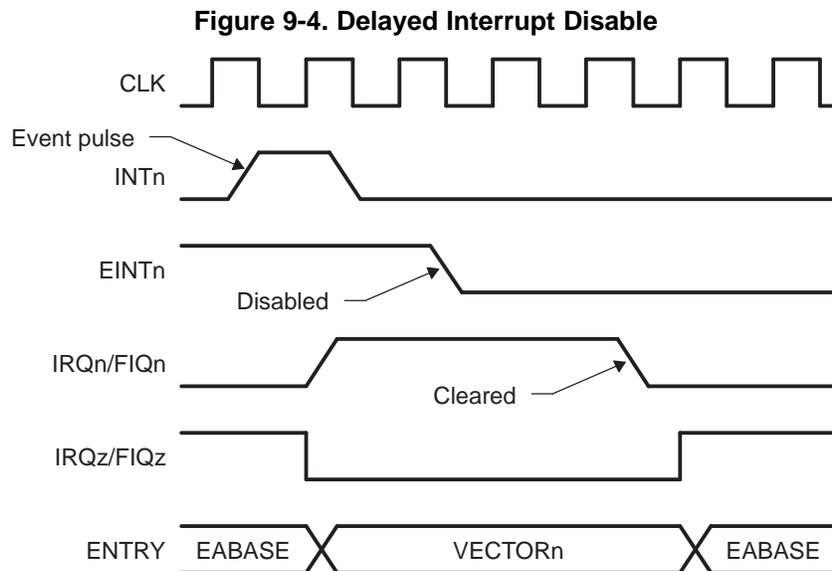
Events cause their matching bit in the FIQ or IRQ register (depending on the event priority) to be cleared to 0. An event is cleared by writing a 1 to the corresponding bit in the FIQ or IRQ register. Writing a 1 to the corresponding bit sets the bit back to a 1. Writing a 0 to an event bit does not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

9.3.5 Enabling and Disabling Interrupts

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



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



9.4 INTC Registers

[Table 9-2](#) lists the memory-mapped registers for the INTC. See the device-specific data manual for the memory address of these registers.

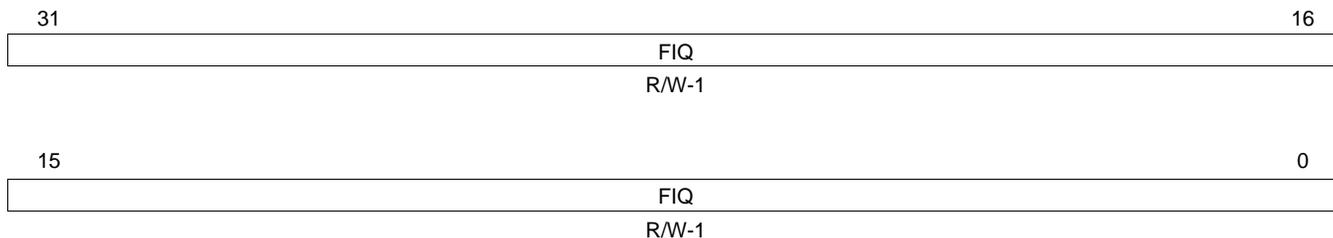
Table 9-2. Interrupt Controller (INTC) Registers

Offset	Acronym	Register Description	Section
00h	FIQ0	Fast Interrupt Request Status Register 0	Section 9.4.1
04h	FIQ1	Fast Interrupt Request Status Register 1	Section 9.4.2
08h	IRQ0	Interrupt Request Status Register 0	Section 9.4.3
0Ch	IRQ1	Interrupt Request Status Register 1	Section 9.4.4
10h	FIQENTRY	Fast Interrupt Request Entry Address Register	Section 9.4.5
14h	IRQENTRY	Interrupt Request Entry Address Register	Section 9.4.6
18h	EINT0	Interrupt Enable Register 0	Section 9.4.7
1Ch	EINT1	Interrupt Enable Register 1	Section 9.4.8
20h	INTCTL	Interrupt Operation Control Register	Section 9.4.9
24h	EABASE	Interrupt Entry Table Base Address Register	Section 9.4.10
30h	INTPRI0	Interrupt 0-7 Priority Register 0	Section 9.4.11
34h	INTPRI1	Interrupt 8-15 Priority Register 1	Section 9.4.12
38h	INTPRI2	Interrupt 16-23 Priority Register 2	Section 9.4.13
3Ch	INTPRI3	Interrupt 24-31 Priority Register 3	Section 9.4.14
40h	INTPRI4	Interrupt 32-39 Priority Register 4	Section 9.4.15
44h	INTPRI5	Interrupt 40-47 Priority Register 5	Section 9.4.16
48h	INTPRI6	Interrupt 48-55 Priority Register 6	Section 9.4.17
4Ch	INTPRI7	Interrupt 56-63 Priority Register 7	Section 9.4.18

9.4.1 Fast Interrupt Request Status Register 0 (FIQ0)

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

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



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

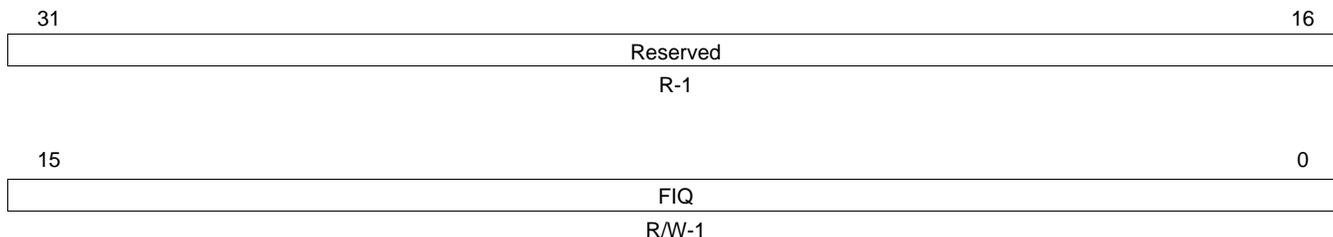
Table 9-3. Fast Interrupt Request Status Register 0 (FIQ0) Field Descriptions

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

9.4.2 Fast Interrupt Request Status Register 1 (FIQ1)

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

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



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

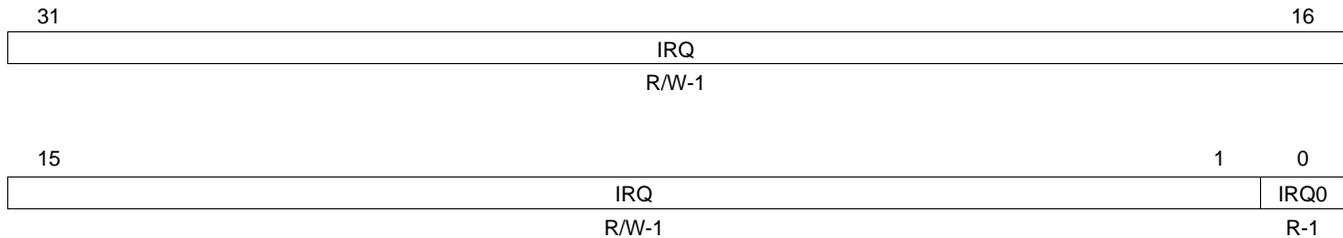
Table 9-4. Fast Interrupt Request Status Register 1 (FIQ1) Field Descriptions

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

9.4.3 Interrupt Request Status Register 0 (IRQ0)

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

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



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

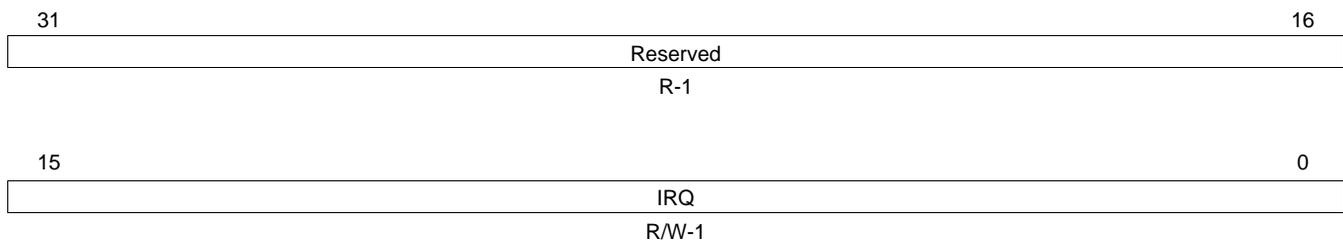
Table 9-5. Interrupt Request Status Register 0 (IRQ0) Field Descriptions

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

9.4.4 Interrupt Request Status Register 1 (IRQ1)

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

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



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

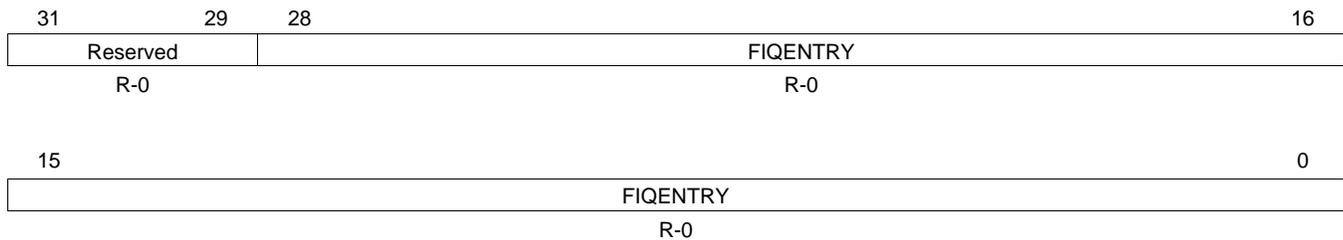
Table 9-6. Interrupt Request Status Register 1 (IRQ1) Field Descriptions

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

9.4.5 Fast Interrupt Request Entry Address Register (FIQENTRY)

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

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



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

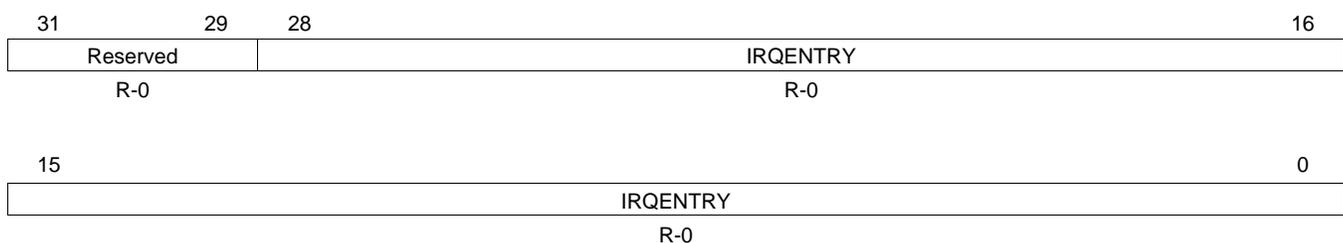
Table 9-7. Fast Interrupt Request Entry Address Register (FIQENTRY) Field Descriptions

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

9.4.6 Interrupt Request Entry Address Register (IRQENTRY)

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

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



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

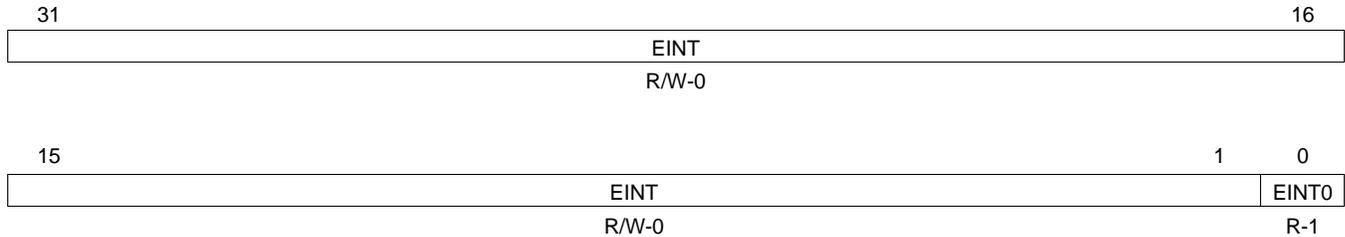
Table 9-8. Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions

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

9.4.7 Interrupt Enable Register 0 (EINT0)

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

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



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

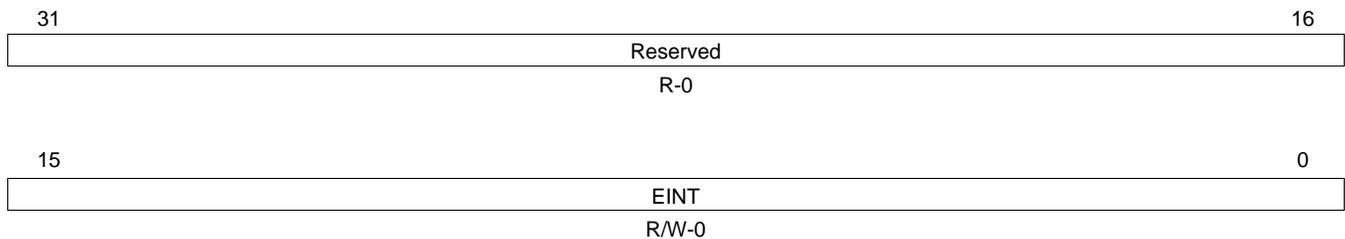
Table 9-9. Interrupt Enable Register 0 (EINT0) Field Descriptions

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

9.4.8 Interrupt Enable Register 1 (EINT1)

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

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



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

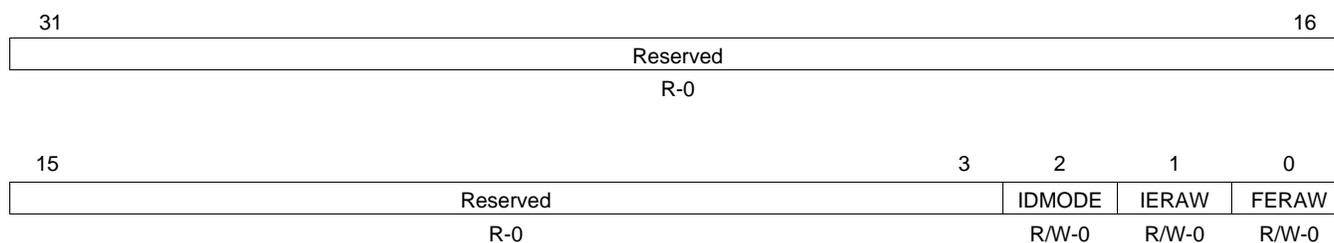
Table 9-10. Interrupt Enable Register 1 (EINT1) Field Descriptions

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

9.4.9 Interrupt Operation Control Register (INTCTL)

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

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



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

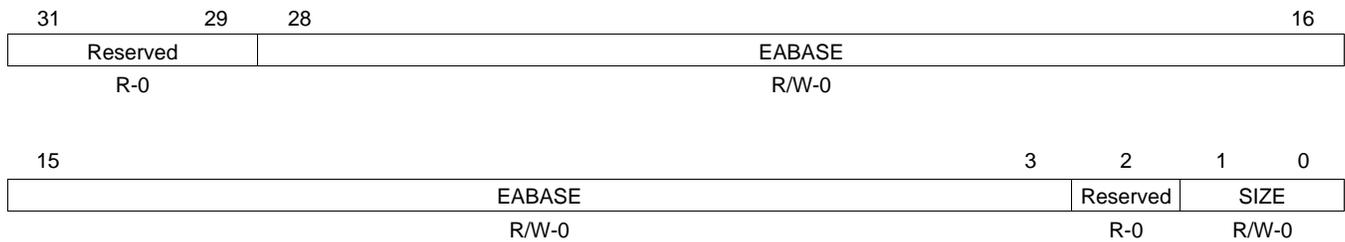
Table 9-11. Interrupt Operation Control Register (INTCTL) Field Descriptions

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

9.4.10 Interrupt Entry Table Base Address Register (EABASE)

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

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



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

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

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

9.4.11 Interrupt Priority Register 0 (INTPRI0)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT7	Reserved	INT6	Reserved	INT5	Reserved	INT4	Reserved	INT3	Reserved	INT2
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT3	Reserved	INT2	Reserved	INT1	Reserved	INT0	Reserved	INT0	Reserved	INT0
R-0	R/W-7h										

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

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

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

9.4.12 Interrupt Priority Register 1 (INTPRI1)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT15	Reserved	INT14	Reserved	INT13	Reserved	INT12	Reserved	INT11	Reserved	INT10
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT11	Reserved	INT10	Reserved	INT9	Reserved	INT8	Reserved	INT7	Reserved	INT6
R-0	R/W-7h										

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

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

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

9.4.13 Interrupt Priority Register 2 (INTPRI2)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT23	Reserved	Reserved	INT22	Reserved	Reserved	INT21	Reserved	Reserved	INT20	
R-0	R/W-7h	R-0	R-0	R/W-7h	R-0	R-0	R/W-7h	R-0	R-0	R/W-7h	
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT19	Reserved	Reserved	INT18	Reserved	Reserved	INT17	Reserved	Reserved	INT16	
R-0	R/W-7h	R-0	R-0	R/W-7h	R-0	R-0	R/W-7h	R-0	R-0	R/W-7h	

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

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

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

9.4.14 Interrupt Priority Register 3 (INTPRI3)

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

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

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

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

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

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

9.4.15 Interrupt Priority Register 4 (INTPRI4)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT39	Reserved	INT38	Reserved	INT37	Reserved	INT36	Reserved	INT35	Reserved	INT34
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT35	Reserved	INT34	Reserved	INT33	Reserved	INT32	Reserved	INT31	Reserved	INT30
R-0	R/W-7h										

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

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

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

9.4.16 Interrupt Priority Register 5 (INTPRI5)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT47	Reserved	INT46	Reserved	INT45	Reserved	INT44	Reserved	INT43	Reserved	INT42
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT43	Reserved	INT42	Reserved	INT41	Reserved	INT40	Reserved	INT39	Reserved	INT38
R-0	R/W-7h										

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

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

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

9.4.17 Interrupt Priority Register 6 (INTPRI6)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT55	Reserved	INT54	Reserved	INT53	Reserved	INT52	Reserved	INT51	Reserved	INT50
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT51	Reserved	INT50	Reserved	INT49	Reserved	INT48	Reserved	INT47	Reserved	INT46
R-0	R/W-7h										

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

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

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

9.4.18 Interrupt Priority Register 7 (INTPRI7)

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

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

31	30	28	27	26	24	23	22	20	19	18	16
Reserved	INT63	Reserved	INT62	Reserved	INT61	Reserved	INT60	Reserved	INT59	Reserved	INT58
R-0	R/W-7h										
15	14	12	11	10	8	7	6	4	3	2	0
Reserved	INT59	Reserved	INT58	Reserved	INT57	Reserved	INT56	Reserved	INT55	Reserved	INT54
R-0	R/W-7h										

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

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

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

System Control Module

Topic	Page
10.1 Overview of the System Control Module	112
10.2 Device Identification	112
10.3 Device Configuration	112
10.4 Device Boot Configuration Status	112
10.5 Power Management	113
10.6 Special Peripheral Status and Control	113
10.7 Bandwidth Management	113
10.8 System Control Register Descriptions	114

10.1 Overview of the System Control Module

The TMS320DM357 DMSoC system control module is a system-level module containing status and top-level control logic required by the device. The system control module consists of a set of status and control registers, accessible by the ARM, supporting all of the following system features and operations:

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

This chapter describes the system control module.

10.2 Device Identification

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

10.3 Device Configuration

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

10.3.1 Pin Multiplexing Control

The DM357 DMSoC makes extensive use of pin multiplexing to accommodate the large number of peripheral functions in the smallest possible package. A combination of hardware configuration (at device reset) and program control controls pin multiplexing to accomplish this. Hardware does not attempt to ensure that the proper pin multiplexing is selected for the peripherals or that interface mode is being used.

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

10.4 Device Boot Configuration Status

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

10.5 Power Management

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

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

10.6 Special Peripheral Status and Control

Several of the DM357 DMSoC peripheral modules require additional system-level control logic. Those registers are discussed in detail in this section.

10.6.1 USB PHY Control

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

10.6.2 VPSS Clock and DAC Control

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

10.7 Bandwidth Management

10.7.1 Bus Master DMA Priority Control

In order to determine allowed connections between masters and slaves, each master request source must have a unique master ID (mstid) associated with it. The master ID for each DM357 DMSoC master is shown in [Table 10-1](#).

Table 10-1. TMS320DM357 DMSoC Master IDs

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

Prioritization within each switched central resource (SCR) is selected to be either fixed or dynamic. Dynamic prioritization is based on an incoming priority signal from each master. On the DM357 DMSoC, only the VPSS and EDMA masters actually generate priority values. For all other masters, the value is programmed in the chip-level MSTRPRI registers. The default priority level for each DM357 DMSoC bus master is shown in [Table 10-2](#). Application software is expected to modify these values to obtain the desired system performance.

Table 10-2. TMS320DM357 DMSoC Default Master Priorities

Master	Default Priority
VPSS	0 ⁽¹⁾
EDMA Ch 0	0
EDMA Ch 1	0
ARM	1
ARM (CFG)	1
EMAC	4
USB	4
HPI	4

⁽¹⁾ Default value in VPSS PCR register

10.8 System Control Register Descriptions

[Table 10-3](#) lists the memory-mapped registers for the system control register.

See the device-specific data manual for the memory address of these registers and complete descriptions of their definition and use.

Table 10-3. System Control Registers

Offset	Acronym	Register Description
00h	PINMUX0	Pin multiplexing control 0
04h	PINMUX1	Pin multiplexing control 1
08h	-	Reserved
0Ch	-	Reserved
10h	-	Reserved
14h	BOOTCFG	Device boot configuration
28h	JTAGID	JTAG/Device ID
34h	USBPHY_CTL	USB PHY control
38h	CHP_SHRTSW	Chip shorting switch control
3Ch	MSTPRI0	Bus master priority control
40h	MSTPRI1	Bus master priority control
44h	VPSS_CLKCTL	VPSS clock mux control register
48h	VDD3P3V_PWDN	VDD 3.3V I/O powerdown control

Reset

Topic	Page
11.1 Reset Overview	116
11.2 Reset Pins	116
11.3 Types of Reset	117
11.4 Default Device Configurations	118

11.1 Reset Overview

There are six types of reset in the TMS320DM357 DMSoC. The types of reset differ by how they are initiated and/or by their effect on the chip. Each type is briefly described in [Table 11-1](#) and further described in the following sections.

Table 11-1. Reset Types

Type	Initiator	Effect
POR (Power-On-Reset)	RESETN pin low and TRSTN low	Total reset of the chip (cold reset). Resets all modules including memory and emulation.
Warm Reset	RESETN pin low and TRSTN high (expected to be the ARM emulator).	Resets all modules including memory, except ARM emulation.
Maximum (max) Reset	ARM emulator or Watchdog Timer (WDT).	Same effect as warm reset.
Module Reset	ARM software	Resets a specific module. Allows the ARM to independently reset any module. Module reset is intended as a debug tool, not necessarily as a tool to use in production.

11.2 Reset Pins

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

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

Table 11-2. Reset Pins

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

11.3 Types of Reset

11.3.1 Power-On Reset (POR)

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

The following steps describe the POR sequence:

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

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

11.3.2 Warm Reset

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

The following steps describe the warm reset sequence:

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

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

11.3.3 Maximum Reset

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

The following steps describe the max reset sequence:

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

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

See the *TMS320DM357 DMSoC 64-Bit Timer User's Guide* ([SPRUG28](#)) for information on the watchdog timer. See [Chapter 3](#) for information on emulation.

11.3.4 Module Reset

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

The procedures for asserting and de-asserting module reset are fully described in [Chapter 7](#).

11.4 Default Device Configurations

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

Note: Default configuration is the configuration before the boot process begins. The boot ROM updates the configuration. See [Chapter 12](#) for more information on the boot process.

11.4.1 Device Configuration Pins

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

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

These pins are described further in the following sections.

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

Table 11-3. Device Configuration

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

Note: Affects both AEMIF and NAND boot modes

AEAW[4:0]	AEMIF address bus width	YOUT[4:0]	00000 (All GPIO)	SYS.PINMUX0.AEAW
-----------	-------------------------	-----------	---------------------	------------------

11.4.2 PLL and Clock Configuration

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

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

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

11.4.3 ARM Boot Mode Configuration

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

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

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

Boot modes are further described in [Chapter 12](#).

11.4.4 AEMIF Configuration

11.4.4.1 CE0 Bus Width Configuration

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

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

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

11.4.4.2 AEMIF Address Width Configuration

The DM357 DMSoC pin multiplexing control logic allows all but three of the Asynchronous EMIF address pins to be used as GPIOs. If devices (such as NAND Flash) attached to the AEMIF require less than the 22 address pins provided, then you can configure the unused upper-order addresses as GPIOs. You must configure these pins at reset so that pins that the AEMIF drives with addresses do not cause bus contention with pins that the system drives as general purpose inputs.

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

11.4.4.3 AEMIF Timing Configuration

When AEMIF is enabled, the wait state registers are reset to the slowest possible configuration, which is 88 cycles per access (16 cycles of setup, 64 cycles of strobe, and 8 cycles of hold). Thus, with a 27 MHz clock at MXI, the AEMIF is configured to run at 4.5 MHz/88 which equals approximately 51 kHz by default. See the *TMS320DM357 DMSoC Asynchronous External Memory Interface (EMIF) User's Guide (SPRUG33)* for more information on the AEMIF.

Boot Modes

Topic	Page
12.1 Boot Modes Overview	122
12.2 ARM ROM Boot Modes.....	124

12.1 Boot Modes Overview

The TMS320DM357 DMSoC ARM can boot from either asynchronous EMIF/NOR Flash or from ARM ROM, as determined by the setting of the device configuration pins BTSEL[1:0]. The BTSEL[1:0] pins can define the ROM boot mode further as well. These ROM boot modes are described in the following sections.

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

12.1.1 Features

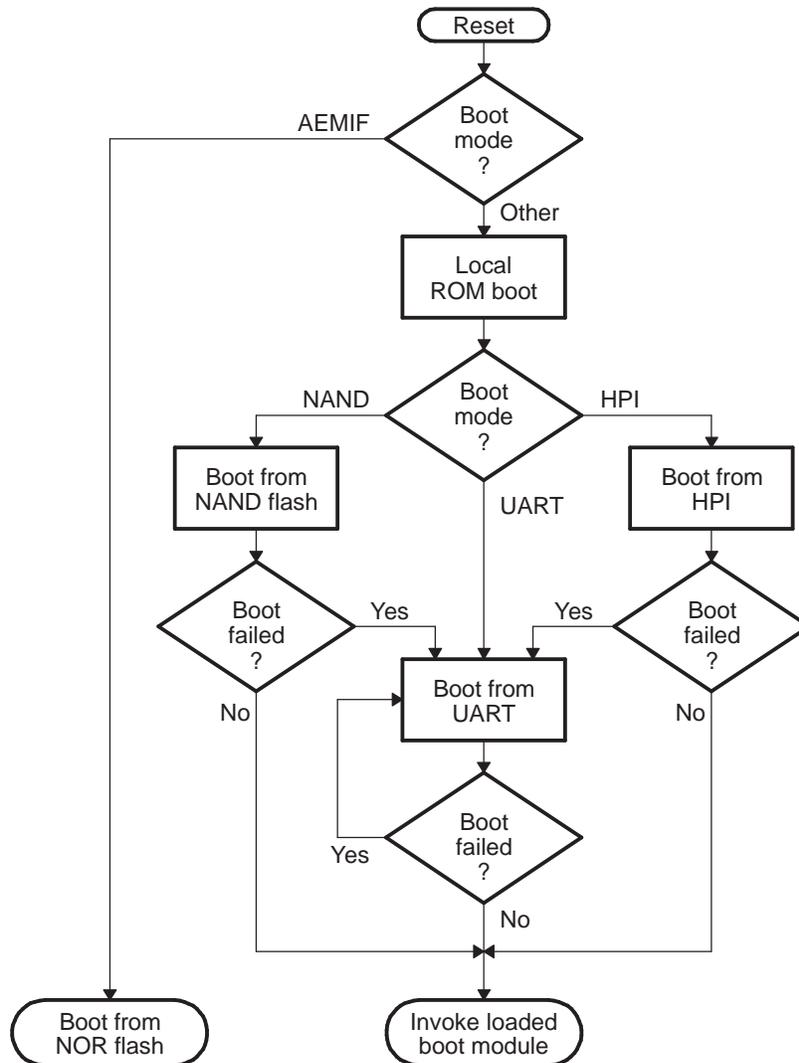
The DM357 DMSoC ARM ROM boot loader (RBL) executes when the BOOTSEL[1:0] pins indicate a condition other than the normal ARM EMIF boot.

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

12.1.2 Functional Block Diagram

The general boot sequence is shown in [Figure 12-1](#).

Figure 12-1. Boot Mode Functional Block Diagram



12.2 ARM ROM Boot Modes

The DM357 DMSoC ARM ROM boot loader (RBL) executes when the BOOTSEL[1:0] pins indicate a condition other than the normal ARM EMIF boot (BTSEL[1:0] ≠ 01). In this case, control is passed to the ROM boot loader (RBL). The RBL then executes the proper mode after reading the state of the BTSEL[1:0] pins from the BOOTCFG register.

12.2.1 NAND Boot Mode

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

First, the device ID of the NAND device is read from the device. Any needed access information (such as the block and page sizes, etc.) are obtained from the device information table in the RBL. Then, the RBL searches for the UBL descriptor in page 0 of the block after CIS/IDI block (block 1).

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

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

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

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

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

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

The exact location of where the RBL expects to find the stored ECC depends on the page size of the device. For 256-byte/page and 512-byte/page devices, the ECC value is stored in the first four bytes of the spare byte region (see [Table 12-1](#) and [Table 12-2](#)). For 2048-byte/page devices, there are four ECC values ([Table 12-3](#)), one for each set of 512 bytes read, which should be stored at addresses 8h, 18h, 28h, and 38h of the spare bytes region. The 32-bit ECC values should be in big-endian order. The provided example UBL respects these ECC requirements, as can be seen by inspecting the source file, *nand.c*.

The NAND user boot loader descriptor format is described in [Table 12-4](#).

Figure 12-2. NAND Boot Flow

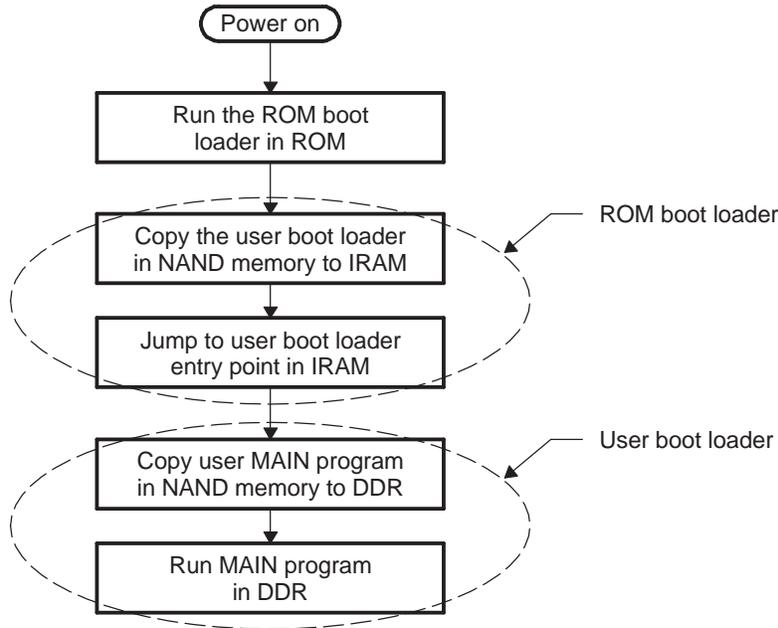


Table 12-1. ECC Value Location in Spare Page Bytes of 256-Byte/Page NAND Devices

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

Table 12-2. ECC Value Location in Spare Page Bytes of 512-Byte/Page NAND Devices

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

Table 12-3. ECC Value Location in Spare Page Bytes of 2048-Byte/Page NAND Devices

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

Table 12-4. NAND UBL Descriptor

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

12.2.1.1 NAND Device IDs Supported

The list of IDs supported by ROM boot loader is shown in [Table 12-5](#) with its characteristics.

Note: The DM357 DMSoC does not support NAND flashes that require the chip select to stay low during the tR time for a read.

Table 12-5. NAND IDs Supported

Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (for address)	Number of address cycles
6Eh	16	256 + 8	12	3
68h	16	256 + 8	12	3
ECh	16	256 + 8	12	3
E8h	16	256 + 8	12	3
EAh	16	256 + 8	12	3
E3h	16	512 + 16	12	3
E5h	16	512 + 16	12	3
E6h	16	512 + 16	12	3
39h	16	512 + 16	12	3
6Bh	16	512 + 16	12	3
73h	32	512 + 16	13	3
33h	32	512 + 16	13	3
75h	32	512 + 16	13	3
35h	32	512 + 16	13	3
76h	32	512 + 16	13	4
36h	32	512 + 16	13	4
79h	32	512 + 16	13	4
71h	32	512 + 16	13	4
46h	32	512 + 16	13	4
56h	32	512 + 16	13	4
74h	32	512 + 16	13	4
F1h	64	2048 + 64	22	4
A1h	64	2048 + 64	22	4
AAh	64	2048 + 64	22	5
DAh	64	2048 + 64	22	5
ACh	64	2048 + 64	22	5
DCh	64	2048 + 64	22	5
ACh	64	2048 + 64	22	5

Table 12-5. NAND IDs Supported (continued)

Device ID	Number of pages per block	Bytes per page (including extra data)	Block shift value (for address)	Number of address cycles
B1h	64	2048 + 64	22	4
C1h	64	2048 + 64	22	4

12.2.1.2 UBL Signature and Special Modes

Different NAND boot mode options are selected by setting different MAGIC IDs in the UBL descriptor. [Table 12-6](#) lists the UBL signatures.

Table 12-6. UBL Signatures and Special Modes

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

12.2.2 UART Boot Mode

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

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

The UART boot mode execution assumes the following UART settings:

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

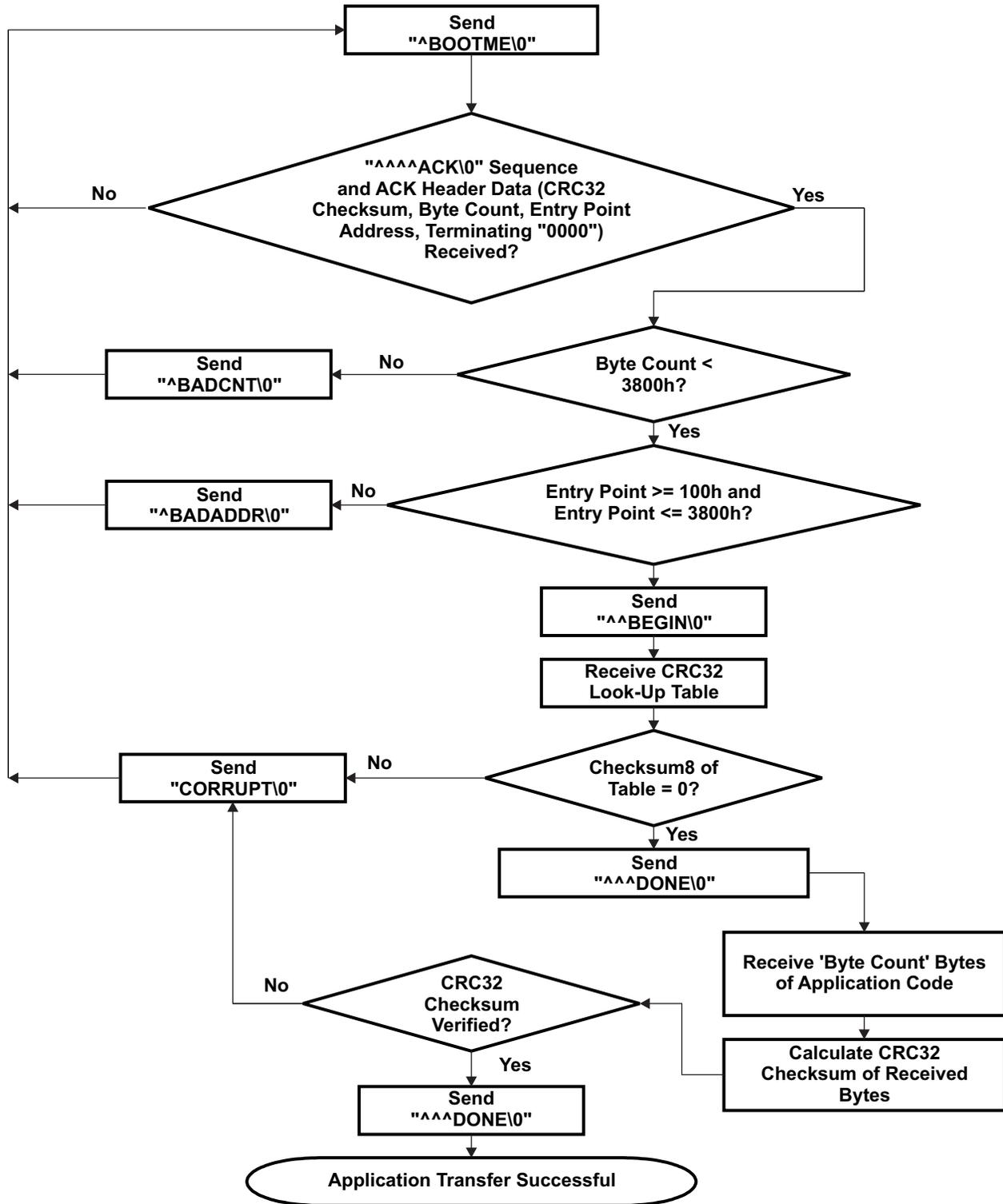
12.2.2.1 TMS320DM357 DMSoC and Serial Host Handshake

[Figure 12-3](#) shows the handshake between the DM357 DMSoC and a serial host utility program. After initialization, there are three main receive sequences: ACK, 1KB CRC32 table, and user boot loader (UBL). For each receive sequence, the time-out check is done in the RBL. If the timeout value is reached during the sequence, the serial boot mode restarts from the beginning which sends out the BOOTME message. The error checking behavior for the UART receive mode is the same. For each byte received, if there is an error, RBL restarts from the beginning.

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

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

Figure 12-3. UART Boot Mode Handshake



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

12.2.2.2 UART Boot Loader Data Sequences

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

Table 12-7 lists the values for the handshake sequences and header for UBL.

Table 12-7. UART Data Sequences

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

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

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

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

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

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

Example 12-1. Host Serial Utility Transmission of Characters

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

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

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

12.2.3 HPI Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 10b, the HPI boot mode executes. The operations followed in the HPI boot mode are shown in [Figure 12-4](#).

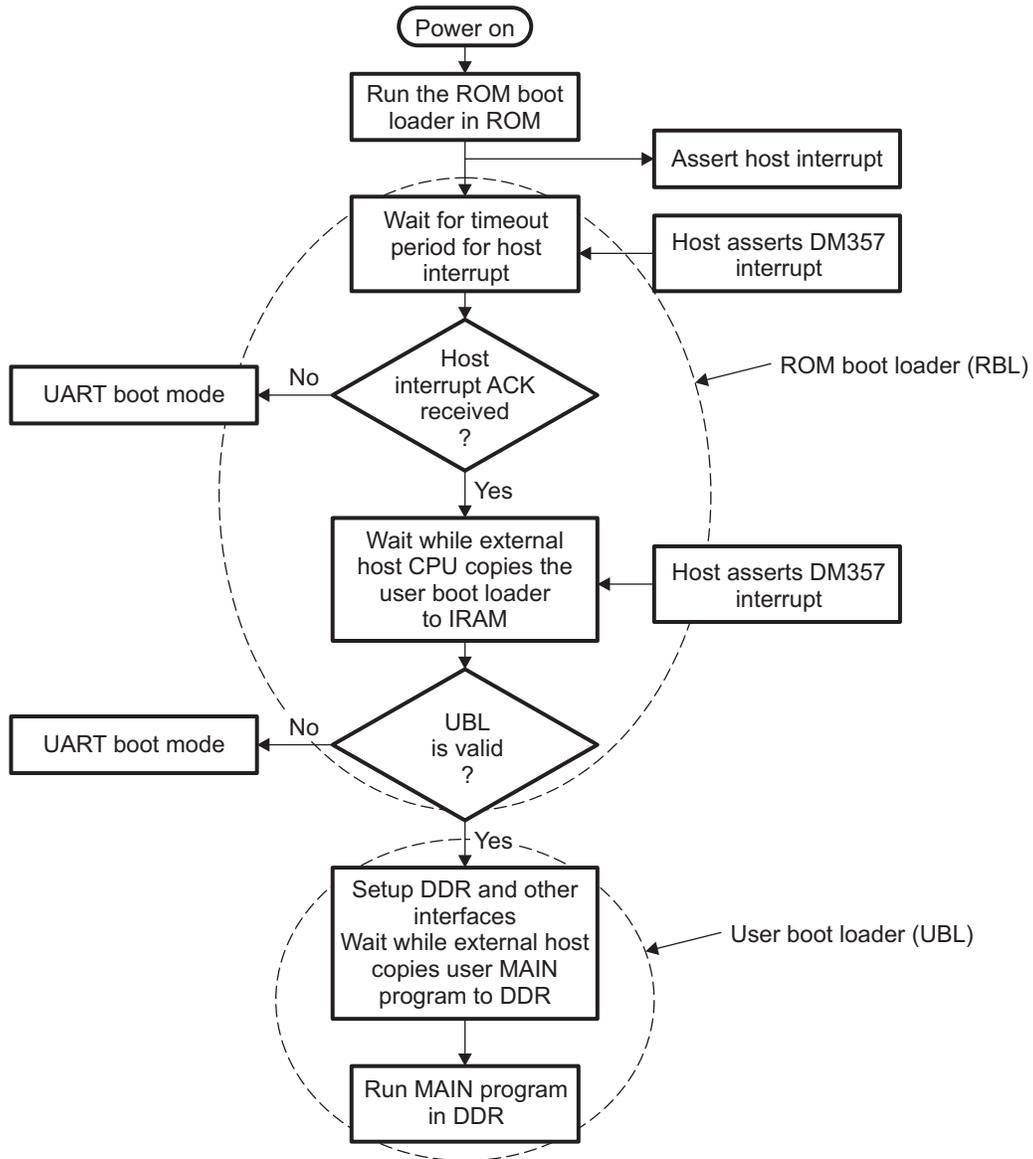
First, the DM357 DMSoC RBL signals the Host that it is ready via the Host interrupt/ $\overline{\text{HINT}}$ signal. Once the interrupt has been sent to the Host, the RBL drops into a polling loop, waiting for the external Host interrupt to acknowledge (ACK) its presence. The RBL waits in this polling loop for a timeout period of approximately 50 seconds. After approximately 50 seconds if the Host has not ACK its presence, the RBL aborts the HPI boot and transfers control to the backup UART boot mode. If the Host does ACK via the interrupt, the RBL drops into a polling loop, waiting for the Host to load/copy the UBL. The RBL waits indefinitely for the Host to load/copy the UBL.

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

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

Once the Host has completed loading the UBL, it should signal the DM357 DMSoC that it is finished via the Host interrupt/ $\overline{\text{HINT}}$ signal. After the Host interrupt is received, the RBL verifies the magic number at B820h, making sure the UBL is good. A valid magic number is A1AC ED00h. If the magic number is verified as valid, the RBL transfers control to the UBL entry point specified at B824h; if the magic number is invalid, the HPI boot fails and the RBL transfers control to the UART boot mode.

Figure 12-4. HPI Boot Sequence



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
RF/IF and ZigBee® Solutions	www.ti.com/lprf

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital Control	www.ti.com/digitalcontrol
Medical	www.ti.com/medical
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2008, Texas Instruments Incorporated