TMS320DM355 Digital Media System-on-Chip (DMSoC)
ARM Subsystem

User's Guide

Literature Number: SPRUFB3A
September 2007–Revised August 2010

Contents

Preface **15**

1 Introduction **19**
- 1.1 Device Overview 19
- 1.2 Block Diagram 19
- 1.3 ARM Subsystem 20

2 ARM Subsystem Overview **21**
- 2.1 Purpose of the ARM Subsystem 21
- 2.2 Components of the ARM Subsystem 21
- 2.3 References 22

3 ARM Core **23**
- 3.1 Introduction 23
- 3.2 Operating States/Modes 24
- 3.3 Processor Status Registers 24
- 3.4 Exceptions and Exception Vectors 25
- 3.5 The 16-BIS/32-BIS Concept 25
  - 3.5.1 16-BIS/32-BIS Advantages 25
- 3.6 Coprocessor 15 (CP15) 26
  - 3.6.1 Addresses in an ARM926EJ-S System 26
  - 3.6.2 Memory Management Unit 27
  - 3.6.3 Caches and Write Buffer 27
- 3.7 Tightly Coupled Memory 29
- 3.8 Embedded Trace Support 30

4 Memory Mapping **31**
- 4.1 Memory Map 31
  - 4.1.1 ARM Internal Memories 32
  - 4.1.2 External Memories 32
  - 4.1.3 MPEG/JPEG Coprocessor (MJCP) 32
  - 4.1.4 Peripherals 32
- 4.2 Memory Interfaces Overview 35
  - 4.2.1 DDR2 EMIF 35
  - 4.2.2 External Memory Interface 35

5 Device Clocking **37**
- 5.1 Overview 37
- 5.2 Peripheral Clocking Considerations 39
  - 5.2.1 Video Processing Back End Clocking 39
  - 5.2.2 USB Clocking 39

6 PLL Controllers (PLLCs) **41**
- 6.1 PLL Controller Module 41
- 6.2 PLLC1 42
- 6.3 PLLC2 43
- 6.4 PLLC Functional Description 44
  - 6.4.1 Multipliers and Dividers 44
  - 6.4.2 Bypass Mode 44
  - 6.4.3 PLL Mode 44

6.5 PLL Configuration 45
6.5.1 PLL Mode and Bypass Mode 45
6.5.2 Changing Divider / Multiplier Ratios 45
6.5.3 PLL Power Down and Wakeup 47
6.6 PLL Controller Registers 48
6.6.1 Peripheral ID Register (PID) 49
6.6.2 PLL Control (PLLCTL) 50
6.6.3 PLL Multiplier Control Register (PLLM) 51
6.6.4 PLL Pre-Divider Control Register (PREDIV) 52
6.6.5 PLL Controller Divider 1 Register (PLLDIV1) 53
6.6.6 PLL Controller Divider 2 Register (PLLDIV2) 54
6.6.7 PLL Controller Divider 3 Register (PLLDIV3) 55
6.6.8 PLL Post-Divider Control Register (POSTDIV) 56
6.6.9 Bypass Divider Register (BPDIV) 57
6.6.10 PLL Controller Command Register (PLLCMD) 58
6.6.11 PLL Controller Status Register (PLLSTAT) 59
6.6.12 PLL Controller Clock Align Control Register (ALNCTL) 60
6.6.13 PLLDIV Ratio Change Status Register (DCHANGE) 61
6.6.14 Clock Enable Control Register (CKEN) 62
6.6.15 Clock Status Register (CKSTAT) 63
6.6.16 SYSCLK Status Register (SYSTAT) 64
6.6.17 PLL Controller Divider 4 Register (PLLDIV4) 65

7 Power and Sleep Controller 67

7.1 Introduction 67
7.2 Power Domain and Module Topology 67
7.3 Power Domain and Module States Defined 70
7.3.1 Power Domain States 70
7.3.2 Module States 70
7.4 Executing State Transitions 71
7.4.1 Power Domain State Transitions 71
7.4.2 Module State Transitions 71
7.5 IcePick Emulation Support in the PSC 71
7.6 PSC Interrupts 72
7.6.1 Interrupt Events 72
7.6.2 Interrupt Registers 73
7.6.3 Interrupt Handling 74
7.7 PSC Registers 75
7.7.1 Peripheral Revision and Class Information (PID) 76
7.7.2 Interrupt Evaluation Register (INTEVAL) 77
7.7.3 Module Error Pending Register 0 (mod 0 - 31) (MERRPR0) 78
7.7.4 Module Error Pending Register 1 (mod 32-41) (MERRPR1) 79
7.7.5 Module Error Clear Register 0 (mod 0-31) (MERRCR0) 80
7.7.6 Module Error Clear Register 1 (mod 32-41) (MERRCR1) 81
7.7.7 Power Error Pending Register (PERRPR) 82
7.7.8 Power Error Clear Register (PERRCR) 83
7.7.9 External Power Control Pending Register (EPCPR) 84
7.7.10 External Power Control Clear Register (EPCCR) 85
7.7.11 Power Domain Transition Command Register (PTCMD) 86
7.7.12 Power Domain Transition Status Register (PTSTAT) 87
7.7.13 Power Domain Status Register (PDSTAT) 88
7.7.14 Power Domain Control Register (PDCTL) 89
7.7.15 Module Status n Register 0-32 (MDSTATn) 90
7.7.16 Module Control n Register 0-51 (MDCTLn) 91

8 Interrupt Controller 93
8.1 Introduction 93
8.2 Interrupt Mapping 93
8.3 INTC Methodology 94
8.3.1 Interrupt Mapping 95
8.3.2 Interrupt Prioritization 95
8.3.3 Vector Table Entry Address Generation 96
8.3.4 Clearing Interrupts 96
8.3.5 Enabling and Disabling Interrupts 97
8.4 INTC Registers 98
8.4.1 Fast Interrupt Request Status Register 0 (FIQ0) 99
8.4.2 Fast Interrupt Request Status Register 1 (FIQ1) 100
8.4.3 Interrupt Request Status Register 0 (IRQ0) 101
8.4.4 Interrupt Request Status Register 1 (IRQ1) 102
8.4.5 Fast Interrupt Request Entry Address Register (FIQENTRY) 103
8.4.6 Interrupt Request Entry Address Register (IRQENTRY) 104
8.4.7 Interrupt Enable Register 0 (EINT0) 105
8.4.8 Interrupt Enable Register 1 (EINT1) 106
8.4.9 Interrupt Operation Control Register (INTCTL) 107
8.4.10 EABASE 108
8.4.11 Interrupt Priority Register 0 (INTPRI0) 109
8.4.12 Interrupt Priority Register 1 (INTPRI1) 110
8.4.13 Interrupt Priority Register 2 (INTPRI2) 111
8.4.14 Interrupt Priority Register 3 (INTPRI3) 112
8.4.15 Interrupt Priority Register 4 (INTPRI4) 113
8.4.16 Interrupt Priority Register 5 (INTPRI5) 114
8.4.17 Interrupt Priority Register 6 (INTPRI6) 115
8.4.18 Interrupt Priority Register 7 (INTPRI7) 116

9 System Control Module 117
9.1 Overview of the System Control Module 117
9.2 Device Identification 117
9.3 Device Configuration 117
9.3.1 Pin Multiplexing Control 117
9.3.2 Device Boot Configuration Status 118
9.4 ARM Interrupt and EDMA Event Multiplexing Control 118
9.5 Special Peripheral Status and Control 118
9.5.1 Timer64+ Control 118
9.5.2 USB PHY Control 119
9.5.3 VPSS Clock and DAC Control and Status 119
9.5.4 DDR I/O Timing Control and Status 119
9.6 Clock Out Configuration Status 119
9.7 GIO De-Bounce Control 119
9.8 Power Management 119
9.8.1 Deep Sleep Control 119
9.9 Bandwidth Management 120
9.9.1 Bus Master DMA Priority Control 120
9.10 System Control Registers 122
9.10.1 PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register 123
9.10.2 PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register 125
9.10.3 PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register 127
9.10.4 PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register 129
9.10.5 PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register 131
9.10.6 BOOTCFG - Boot Configuration 132

9.10.7 ARM_INTMUX - ARM Interrupt Mux Control Register 133
9.10.8 EDMA_EVTMUX - EDMA Event Mux Control Register 134
9.10.9 DDR_SLEW - DDR Slew 135
9.10.10 CLKOUT - CLKOUT Divisor / Output Control 136
9.10.11 DEVICE_ID - Device ID 137
9.10.12 VDAC_CONFIG - Video Dac Configuration 138
9.10.13 TIMER64_CTL - Timer64+ Input Control 139
9.10.14 USB_PHY_CTRL - USB PHY Control 140
9.10.15 MISC - Miscellaneous Control 142
9.10.16 MSTPRI0 - Master Priorities 0 142
9.10.17 Master Priorities 1 (MSTPRI1) Register 142
9.10.18 VPSS_CLK_CTRL - VPSS Clock Mux Control 144
9.10.19 Deep Sleep Mode Configuration (DEEPSLEEP) Register 145
9.10.20 DEBOUNCE[8] - De-bounce for GIO[n] Input 145
9.10.21 VTPIOCR - VTP IO Control Register 147

10 Reset 149
10.1 Reset Overview 149
10.2 Reset Pins 149
10.3 Types of Reset 150
10.3.1 Power-On Reset (POR) 150
10.3.2 Warm Reset 150
10.3.3 Max Reset 151
10.3.4 System Reset 151
10.3.5 Module Reset 151
10.4 Default Device Configurations 151
10.4.1 Device Configuration Pins 151
10.4.2 PLL Configuration 152
10.4.3 Module Configuration 152
10.4.4 ARM Boot Mode Configuration 152
10.4.5 AEMIF Configuration 153

11 Boot Modes 155
11.1 Boot Modes Overview 155
11.1.1 Features 155
11.1.2 Functional Block Diagram 157
11.2 ARM ROM Boot Mode 157
11.2.1 SPI Boot Mode 158
11.2.2 NAND Boot Mode 159
11.2.3 MMC/SD Boot Mode 170
11.2.4 UART Boot Mode 175

12 Power Management 179
12.1 Overview 179
12.2 PSC and PLLC Overview 179
12.3 Clock Management 180
12.3.1 Module Clock Disable 180
12.3.2 Module Clock Frequency Scaling 180
12.3.3 PLL Bypass and Power Down 180
12.4 ARM Sleep Mode Management 180
12.4.1 ARM Wait-For-Interrupt Sleep Mode 180
12.5 System Sleep Modes 181
12.5.1 Deep Sleep Mode 181
12.6 I/O Management 181
12.6.1 USB Phy Power Down 181
12.6.2 Video DAC Power Down 182

12.6.3 DDR Self-Refresh and Power Down 182
A Revision History 183

List of Figures

1-1. Functional Block Diagram 20
2-1. ARM Subsystem Block Diagram 22
5-1. Clocking Architecture 38
6-1. PLLC1 Configuration 43
6-2. PLLC2 Configuration 44
6-3. Clock Ratio Change and Alignment with Go Operation 46
6-4. Peripheral ID Register (PID) 49
6-5. PLL Control Register (PLLCTL) 50
6-6. PLL Multiplier Control Register (PLLM) 51
6-7. PLL Pre-Divider Control Register (PREDIV) 52
6-8. PLL Controller Divider 1 Register (PLLDIV1) 53
6-9. PLL Controller Divider 2 Register (PLLDIV2) 54
6-10. PLL Controller Divider 3 Register (PLLDIV3) 55
6-11. PLL Post-Divider Control Register (POSTDIV) 56
6-12. Bypass Divider Register (BPDIV) 57
6-13. PLL Controller Command Register (PLLCMD) 58
6-14. PLL Controller Status Register (PLLSTAT) 59
6-15. PLL Controller Clock Align Control Register (ALNCTL) 60
6-16. PLLDIV Ratio Change Status (DCHANGE) 61
6-17. Clock Enable Control Register (CKEN) 62
6-18. Clock Status Register (CKSTAT) 63
6-19. SYSCLK Status Register (SYSTAT) 64
6-20. PLL Controller Divider 4 Register (PLLDIV4) 65
7-1. Power and Sleep Controller (PSC) 67
7-2. Power Domain and Module Topology 68
7-3. Peripheral Revision and Class Information Register (PID) 76
7-4. Interrupt Evaluation Register (INTEVAL) 77
7-5. Module Error Pending Register 0 (mod 0 - 31) (MERRPR0) 78
7-6. Module Error Pending Register 1 (mod 32-41) (MERRPR1) 79
7-7. Module Error Clear Register 0 (mod 0-31) (MERRCR0) 80
7-8. Module Error Clear Register 1 (mod 32-41) (MERRCR1) 81
7-9. Power Error Pending Register (PERRPR) 82
7-10. Power Error Clear Register (PERRCR) 83
7-11. External Power Control Pending Register (EPCPR) 84
7-12. External Power Control Clear Register (EPCCR) 85
7-13. Power Domain Transition Command Register (PTCMD) 86
7-14. Power Domain Transition Status Register (PTSTAT) 87
7-15. Power Domain Status Register (PDSTAT) 88
7-16. Power Domain Control Register (PDCTL) 89
7-17. Module Status n Register (MDSTATn) 90
7-18. Module Control n Register 0-41 (MDCTLn) 91
8-1. AINTC Functional Diagram 95
8-2. Interrupt Entry Table 96
8-3. Immediate Interrupt Disable / Enable 97
8-4. Delayed Interrupt Disable 97
8-5. Interrupt Status of INT[31:0] (if mapped to FIQ) 99
8-6. Interrupt Status of INT[63:32] (if mapped to FIQ) 100

8-7. Interrupt Status of INT[31:0] (if mapped to IRQ) 101
8-8. Interrupt Status of INT[31:0] (if mapped to IRQ) 102
8-9. Fast Interrupt Request Entry Address Register (FIQENTRY) 103
8-10. Interrupt Request Entry Address Register (IRQENTRY) 104
8-11. Interrupt Enable Register 0 (EINT0) 105
8-12. Interrupt Enable Register 1 (EINT1) 106
8-13. Interrupt Operation Control Register (INTCTL) 107
8-14. EABASE 108
8-15. Interrupt Priority Register 0 (INTPRI0) 109
8-16. Interrupt Priority Register 1 (INTPRI1) 110
8-17. Interrupt Priority Register 2 (INTPRI2) 111
8-18. Interrupt Priority Register 3 (INTPRI3) 112
8-19. Interrupt Priority Register 4 (INTPRI4) 113
8-20. Interrupt Priority Register 5 (INTPRI5) 114
8-21. Interrupt Priority Register 6 (INTPRI6) 115
8-22. Interrupt Priority Register 7 (INTPRI7) 116
9-1. PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register 123
9-2. PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register 125
9-3. PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register 127
9-4. PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register 129
9-5. PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register 131
9-6. BOOTCFG - Boot Configuration 132
9-7. ARM_INTMUX - ARM Interrupt Mux Control Register 133
9-8. EDMA_EVTMUX - EDMA Event Mux Control Register 134
9-9. DDR_SLEW - DDR Slew 135
9-10. CLKOUT - CLKOUT div/out Control 136
9-11. DEVICE_ID - Device ID 137
9-12. VDAC_CONFIG - Video Dac Configuration 138
9-13. TIMER64_CTL - Timer64+ Input Control 139
9-14. USB_PHY_CTRL - USB PHY Control 140
9-15. MISC - Miscellaneous Control 142
9-16. MSTPRI0 - Master Priorities 0 142
9-17. Master Priorities 1(MSTPRI1) Register 143
9-18. VPSS_CLK_CTRL - VPSS Clock Mux Control 144
9-19. Deep Sleep Mode Configuration (DEEPSLEEP) Register 145
9-20. DEBOUNCE[8] - De-bounce for GIO[n] Input 146
9-21. VTP IO Control Register (VTPIOCR) 147
11-1. Boot Modes Overview 156
11-2. Boot Mode Functional Block Diagram 157
11-3. SPI Boot Overview 158
11-4. NAND Boot Flow 160
11-5. 4-Bit ECC Format and Bit 10 to 8-Bit Compression Algorithm 162
11-6. 4-Bit ECC Format for 2048+64 Byte Page Size (in Compatibility Mode) 163
11-7. NAND Boot Mode Code Flow 164
11-8. NAND Boot Mode Flow Chart 167
11-9. ARM NAND ROM Boot Loader Example 168
11-10. Descriptor Search for ARM NAND Boot Mode 169
11-11. MMC/SD Boot Mode Overview 171
11-12. MMC/SD Boot Mode Flow Chart 173

11-13. ARM MMC/SD ROM Boot Loader Example 174
11-14. Descriptor Search for ARM MMC/SD Boot Mode 175
11-15. UART Boot Mode Handshake 176
11-16. Host Utility Timing 178

List of Tables

3-1. Exception Vector Table for ARM 25
3-2. Different Address Types in ARM System 26
3-3. ITCM/DTCM Memory Map 29
3-4. ITCM/DTCM Size Encoding 30
3-5. ETM Part Descriptions 30
4-1. Memory Map 31
4-2. ARM Configuration Bus Access to Peripherals 33
6-1. PLLC1 Output Clocks 42
6-2. PLLC2 Output Clocks 43
6-3. PLL and Reset Controller Module Instance Table 48
6-4. PLLC Registers 48
6-5. Peripheral ID Register (PID) Field Descriptions 49
6-6. PLL Control Register (PLLCTL) Field Descriptions 50
6-7. PLL Multiplier Control Register (PLLM) Field Descriptions 51
6-8. PLL Pre-Divider Control (PREDIV) Field Descriptions 52
6-9. PLL Controller Divider 1 Register (PLLDIV1) Field Descriptions 53
6-10. PLL Controller Divider 2 Register (PLLDIV2) Field Descriptions 54
6-11. PLL Controller Divider 3 Register (PLLDIV3) Field Descriptions 55
6-12. PLL Post-Divider Control (POSTDIV) Field Descriptions 56
6-13. Bypass Divider Register (BPDIV) Field Descriptions 57
6-14. PLL Controller Command Register (PLLCMD) Field Descriptions 58
6-15. PLL Controller Status (PLLSTAT) Field Descriptions 59
6-16. PLL Controller Clock Align Control (ALNCTL) Field Descriptions 60
6-17. PLLDIV Ratio Change Status (DCHANGE) Field Descriptions 61
6-18. Clock Enable Control Register (CKEN) Field Descriptions 62
6-19. Clock Status Register (CKSTAT) Field Descriptions 63
6-20. SYSCLK Status Register (SYSTAT) Field Descriptions 64
6-21. PLL Controller Divider 4 Register (PLLDIV4) Field Descriptions 65
7-1. Module Configuration 69
7-2. Module States 70
7-3. IcePick Emulation Commands 71
7-4. PSC Interrupt Events 72
7-5. PSC Registers 75
7-6. Peripheral Revision and Class Information Register (PID) Field Descriptions 76
7-7. Interrupt Evaluation Register (INTEVAL) Field Descriptions 77
7-8. Module Error Pending Register 0 (mod 0 - 31) (MERRPR0) Field Descriptions 78
7-9. Module Error Pending Register 1 (mod 32-41) (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 32-41) (MERRCR1) Field Descriptions 81
7-12. Power Error Pending Register (PERRPR) Field Descriptions 82
7-13. Power Error Clear Register (PERRCR) Field Descriptions 83
7-14. External Power Control Pending Register (EPCPR) Field Descriptions 84
7-15. External Power Control Clear Register (EPCCR) Field Descriptions 85
7-16. Power Domain Transition Command Register (PTCMD) Field Descriptions 86
7-17. Power Domain Transition Status Register (PTSTAT) Field Descriptions 87
7-18. Power Domain Status Register (PDSTAT) Field Descriptions 88
7-19. Power Domain Control Register (PDCTL) Field Descriptions 89

7-20. Module Status n Register 0-32 (MDSTATn) Field Descriptions 90
7-21. Module Control n Register 0-51 (MDCTLn) Field Descriptions 91
8-1. AINTC Interrupt Connections 93
8-2. Interrupt Controller (INTC) Registers 98
8-3. Interrupt Status of INT[31:0] (if mapped to FIQ) Field Descriptions 99
8-4. Interrupt Status of INT[63:32] (if mapped to FIQ) Field Descriptions 100
8-5. Interrupt Status of INT[31:0] (if mapped to IRQ) Field Descriptions 101
8-6. Interrupt Status of INT[31:0] (if mapped to IRQ) Field Descriptions 102
8-7. Fast Interrupt Request Entry Address Register (FIQENTRY) Field Descriptions 103
8-8. Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions 104
8-9. Interrupt Enable Register 0 (EINT0) Field Descriptions 105
8-10. Interrupt Enable Register 1 (EINT1) Field Descriptions 106
8-11. Interrupt Operation Control Register (INTCTL) Field Descriptions 107
8-12. EABASE Field Descriptions 108
8-13. Interrupt Priority Register 0 (INTPRI0) Field Descriptions 109
8-14. Interrupt Priority Register 1 (INTPRI1) Field Descriptions 110
8-15. Interrupt Priority Register 2 (INTPRI2) Field Descriptions 111
8-16. Interrupt Priority Register 3 (INTPRI3) Field Descriptions 112
8-17. Interrupt Priority Register 4 (INTPRI4) Field Descriptions 113
8-18. Interrupt Priority Register 5 (INTPRI5) Field Descriptions 114
8-19. Interrupt Priority Register 6 (INTPRI6) Field Descriptions 115
8-20. Interrupt Priority Register 7 (INTPRI7) Field Descriptions 116
9-1. Master IDs 120
9-2. Default Master Priorities 121
9-3. System Module (SYS) Registers 122
9-4. PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register Field Descriptions 123
9-5. PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register Field Descriptions 125
9-6. PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register Field Descriptions 127
9-7. PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register Field Descriptions 129
9-8. PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register Field Descriptions 131
9-9. BOOTCFG - Boot Configuration Field Descriptions 132
9-10. ARM_INTMUX - ARM Interrupt Mux Control Register Field Descriptions 133
9-11. EDMA_EVTMUX - EDMA Event Mux Control Register Field Descriptions 134
9-12. DDR_SLEW - DDR Slew Field Descriptions 135
9-13. CLKOUT - CLKOUT div/out Control Field Descriptions 136
9-14. DEVICE_ID - Device ID Field Descriptions 137
9-15. VDAC_CONFIG - Video Dac Configuration Field Descriptions 138
9-16. TIMER64_CTL - Timer64+ Input Control Field Descriptions 139
9-17. USB_PHY_CTRL - USB PHY Control Field Descriptions 140
9-18. MISC - Miscellaneous Control Field Descriptions 142
9-19. MSTPRI0 - Master Priorities 0 Field Descriptions 142
9-20. Master Priorities 1 (MSTPRI1) Register Field Descriptions 143
9-21. VPSS_CLK_CTRL - VPSS Clock Mux Control Field Descriptions 144
9-22. Deep Sleep Mode Configuration (DEEPSLEEP) Register Field Descriptions 145
9-23. DEBOUNCE[8] - De-bounce for GIO[n] Input Field Descriptions 146
9-24. VTPIOCR - VTP IO Control Field Descriptions 147
10-1. Reset Types 149
10-2. Reset Pins 149
10-3. Device Configuration 152

11-1. User Bootloader (UBL) Descriptor for SPI Mode 159
11-2. NAND Layout (Compatibility Mode) 159
11-3. NAND Layout (Standard Mode) 159
11-4. NAND UBL Descriptor 161
11-5. UBL Signatures and Special Modes 161
11-6. NAND Parameters 165
11-7. NAND Devices in NAND Device ID Table 169
11-8. MMC/SD UBL Descriptor 172
11-9. MMC/SD 172
11-10. UART Data Sequences 177
11-11. Host Utility Data Format 178
11-12. CRC32 Table Transfer 178
12-1. Power Management Features 179
A-1. Revision History 183

Read This First

About This Manual

This document describes the operation of the ARM subsystem in the TMS320DM355 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 TMS320DM35x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the internet at www.ti.com.

SPRS463 — *TMS320DM355 Digital Media System-on-Chip (DMSoC) Data Manual* This document describes the overall TMS320DM355 system, including device architecture and features, memory map, pin descriptions, timing characteristics and requirements, device mechanicals, etc.

SPRZ264 — *TMS320DM355 DMSoC Silicon Errata* Describes the known exceptions to the functional specifications for the TMS320DM355 DMSoC.

SPRUFB3 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* This document describes the ARM Subsystem in the TMS320DM35x Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the components of the ARM Subsystem, the peripherals, and the external memories.

SPRUED1 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (EMIF) Reference Guide* This document describes the asynchronous external memory interface (EMIF) in the TMS320DM35x Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.

SPRUED2 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Reference Guide* This document describes the universal serial bus (USB) controller in the TMS320DM35x 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.

SPRUED3 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Audio Serial Port (ASP) Reference Guide* This document describes the operation of the audio serial port (ASP) audio interface in the TMS320DM35x 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.

SPRUED4 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Serial Peripheral Interface (SPI) Reference Guide* This document describes the serial peripheral interface (SPI) in the TMS320DM35x 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.

SPRUED9 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Universal Asynchronous Receiver/Transmitter (UART) Reference Guide* This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM35x 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.

SPRUEE0 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Inter-Integrated Circuit (I2C) Peripheral Reference Guide* This document describes the inter-integrated circuit (I2C) peripheral in the TMS320DM35x 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.

SPRUEE2 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Multimedia Card (MMC)/Secure Digital (SD) Card Controller Reference Guide* This document describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM35x 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.

SPRUEE4 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Enhanced Direct Memory Access (EDMA) Controller Reference Guide* This document describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM35x Digital Media System-on-Chip (DMSoC). The EDMA controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.

SPRUEE5 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) 64-bit Timer Reference Guide* This document describes the operation of the software-programmable 64-bit timers in the TMS320DM35x Digital Media System-on-Chip (DMSoC). Timer 0, Timer 1, and Timer 3 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 and Real Time Output (RTO) events (Timer 3 only). 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.

SPRUEE6 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) General-Purpose Input/Output (GPIO) Reference Guide* This document describes the general-purpose input/output (GPIO) peripheral in the TMS320DM35x 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.

SPRUEE7 — *TMS320DM35x Digital Media System-on-Chip (DMSoC) Pulse-Width Modulator (PWM) Reference Guide* This document describes the pulse-width modulator (PWM) peripheral in the TMS320DM35x Digital Media System-on-Chip (DMSoC).

SPRUEH7 — ***TMS320DM35x Digital Media System-on-Chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Reference Guide*** This document describes the DDR2/mDDR memory controller in the TMS320DM35x Digital Media System-on-Chip (DMSoC). The DDR2/mDDR memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM and mobile DDR devices.

SPRUF71 — ***TMS320DM35x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) Reference Guide*** This document describes the Video Processing Front End (VPFE) in the TMS320DM35x Digital Media System-on-Chip (DMSoC).

SPRUF72 — ***TMS320DM35x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Reference Guide*** This document describes the Video Processing Back End (VPBE) in the TMS320DM35x Digital Media System-on-Chip (DMSoC).

SPRUF74 — ***TMS320DM35x Digital Media System-on-Chip (DMSoC) Real-Time Out (RTO) Controller Reference Guide*** This document describes the Real Time Out (RTO) controller in the TMS320DM35x Digital Media System-on-Chip (DMSoC).

SPRUFC8 — ***TMS320DM35x Digital Media System-on-Chip (DMSoC) Peripherals Overview Reference Guide*** This document provides an overview of the peripherals in the TMS320DM35x Digital Media System-on-Chip (DMSoC).

SPRAAR3 — ***Implementing DDR2/mDDR PCB Layout on the TMS320DM35x DMSoC*** This provides board design recommendations and guidelines for DDR2 and mobile DDR.

Introduction

1.1 Device Overview

The TMS320DM355 processor is a low cost, low power processor providing advanced graphical user interface for display applications that do not require video compression and decompression. Coupled with a video processing subsystem (VPSS) that provides up to 720p display, the DM355 processor is powered by an ARM926EJ-S core so developers can create feature rich graphical user interfaces allowing customers to interact with their portable, electronic devices such as video-enabled universal remote controls, Internet radio, e-books, video doorbells and digital telescopes. The DM355 is packed with a wide variety of peripherals including high speed USB 2.0 on-the-go, external memory interface (EMIF), mobile DDR/DDR2, two SDIO ports, three UART Ports, two Audio Serial Ports, three SPI Ports and SLC/MCL NAND Flash memory support. These peripherals help customers create DM355 processor-based designs that add video and audio excitement to a wide range of today's static user-interface applications while keeping silicon costs and power consumption low.

The device delivers a sophisticated suite of capabilities allowing for flexible image capture and display. Through its user interface technology, such as a four-level on-screen display, developers are able to create picture-within-picture and video-within-video as well as innovative graphic user interfaces. This is especially important for portable products that require the use of button or touch screen, such as portable karaoke, video surveillance and electronic gaming applications. Additional advanced capture and imaging technologies include support for CCD/CMOS image sensors, resize capability and video stabilization. The 1280-by-960-pixel digital LCD connection runs on a 75-MHz pixel clock and supports TV composite output for increased expandability. This highly integrated device is packaged in a 13 x 13 mm, 337 pin , 0.65 mm pitch BGA package.

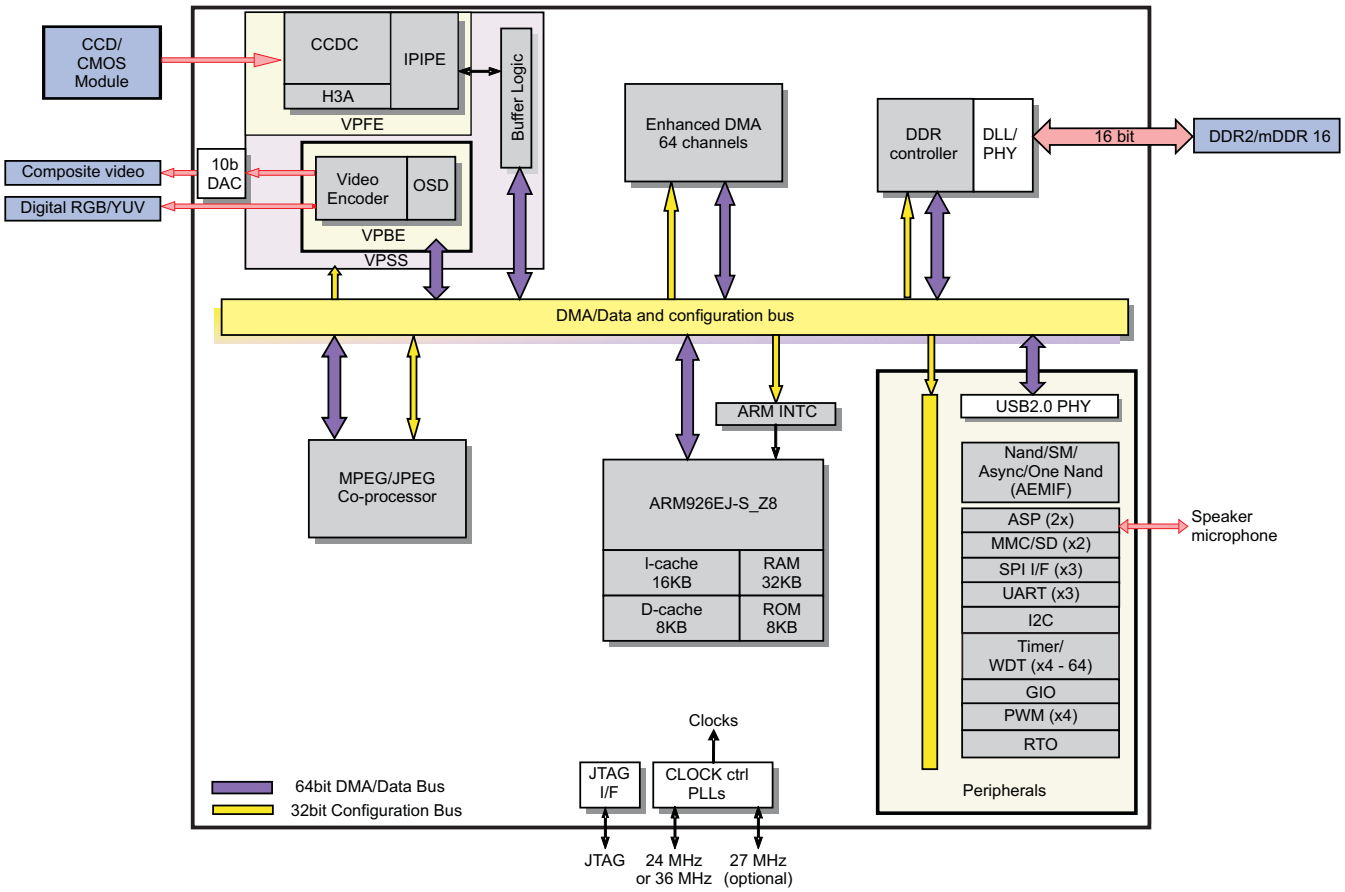
1.2 Block Diagram

The device consists of the following primary components and sub-systems:

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

The detailed block diagram is shown in Figure 1-1.

Figure 1-1. Functional Block Diagram



1.3 ARM Subsystem

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

ARM Subsystem Overview

2.1 Purpose of the ARM Subsystem

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

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

2.2 Components of the ARM Subsystem

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

- ARM926EJ-S RISC processor, including:
  - Coprocessor 15 (CP15)
  - MMU
  - 16KB Instruction cache
  - 8KB Data cache
  - Write Buffer
  - Java accelerator
- ARM Internal Memories
  - 32KB Internal RAM (32-bit wide access)
  - 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

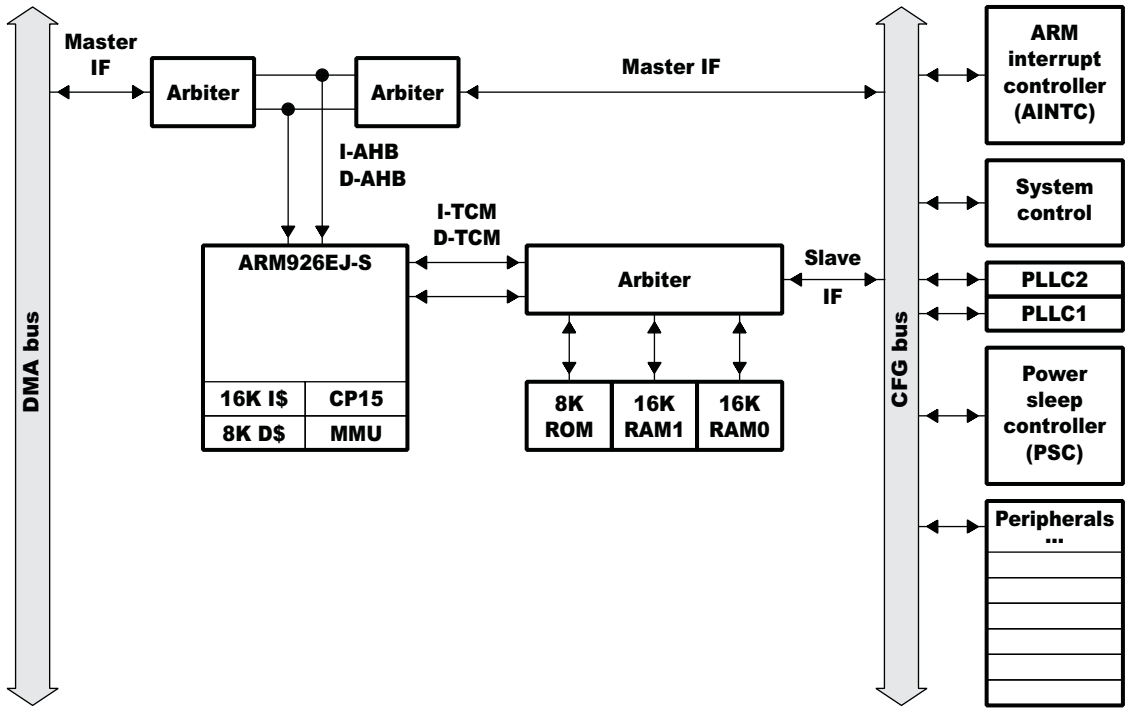
The ARM also manages/controls the following peripherals:

- DDR2 EMIF Controller
- AEMIF Controller, including the NAND flash interface
- Enhanced DMA (EDMA)
- UART (There are three UARTs supported in the device)
- Timers
- Real Time Out (RTO)
- Pulse Width Modulator (PWM)
- Inter-IC Communication (I2C)
- Multi-Media Card/Secure Digital (MMC/SD)
- Audio Serial Port (ASP)
- Universal Serial Bus Controller (USB)
- Serial Port Interface (SPI)

- Video Processing Front End (VPFE)
  - CCD Controller (CCDC)
  - Image Pipe (IPIPE)
  - H3A Engine (Hardware engine for computing Auto-focus, Auto white balance, and Auto exposure)
- Video Processing Back End (VPBE)
  - On Screen Display (OSD)
  - Video Encoder Engine (VENC)

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

Figure 2-1. ARM Subsystem Block Diagram



2.3 References

See the following related documents for more information:

- DM355 Data Manual (SPRS463): Provides a high-level overview of the DM355 system.
- Peripheral Reference Guides: For various peripherals on the device.
- For more detailed information about the ARM processor core, see ARM Ltd.'s web site:
  - http://www.arm.com/documentation/ARMProcessor_Cores/index.html
    - Particularly, see the ARM926EJ-S Technical Reference Manual

ARM Core

3.1 Introduction

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

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

The ARM926EJ-S processor is a member of the ARM9 family of general-purpose microprocessors. 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 x 16 multiply-accumulate (MAC) unit. The ARM subsystem also has 32 KB of internal RAM and 8 KB of internal ROM, accessible via the I-TCM and D-TCM interfaces through an arbiter. The same arbiter provides a slave DMA interface to the rest of the 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.

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

3.3 Processor Status Registers

The processor status register (PSR) controls the enabling and disabling of interrupts and setting the mode of operation of the processor. PSR [7:0] are the processor control bits, PSR [27:8] are reserved bits, and PSR [31:28] are status bits. The control bits, PSR[7:0], are defined as follows:

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

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

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

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

3.4 Exceptions and Exception Vectors

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

- Reset exception: processor reset
- FIQ interrupt: fast interrupt
- IRQ interrupt: normal interrupt
- Abort exception: abort indicates that the current memory access could not be completed. 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 0x00000000 (VINTH = 0) or at address 0xFFFF0000 (VINTH = 1). .

NOTE: This is a feature of the ARM926EJ-S core. However, in this DMSoC there is no memory in the address region starting at 0xFFFF0000, so do not set VINTH.

The default vector table is shown in Table 3-1

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

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-S processor has two instruction sets:

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

The 16-bit instruction length (16-BIS) allows the 16-BIS to approach twice the density of standard 32-BIS code while retaining most of the 32-BIS's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because 16-BIS code operates on the same 32-bit register set as 32-BIS code. 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 (e.g., code that performs character string handling), and some instructions (like branches) do not process any data at all. If a 16-bit architecture only has 16-bit instructions, and a 32-bit architecture only has 32-bit instructions, then the 16-bit architecture has better code density overall, and has better than one half of the performance of the 32-bit architecture. Clearly, 32-bit performance comes at the cost of code density. The 16-bit instruction breaks this constraint by implementing a 16-bit instruction length on a 32-bit architecture, making the processing of 32-bit data efficient with compact instruction coding. This provides far better performance than a 16-bit architecture, with better code density than a 32-bit architecture. The 16-BIS also has a major advantage over other 32-bit architectures with 16-bit instructions. The advantage is the ability to switch back to full 32-bit code and execute at full speed. Thus, critical loops for applications such as fast interrupts and DSP algorithms can be coded using the full 32-BIS and linked with 16-BIS code. The overhead of switching from 16-bit code to 32-bit code is folded into sub-routine entry time. Various portions of a system can be optimized for speed or for code density by switching between 16-BIS and 32-BIS execution, as appropriate.

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

3.6 Coprocessor 15 (CP15)

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

3.6.1 Addresses in an ARM926EJ-S System

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 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 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 linefill, 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 for more detailed information.

3.7 Tightly Coupled Memory

The ARM926EJ-Shas 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 device supports both instruction TCM (I-TCM) and data TCM (D-TCM). The instruction TCM is located at 0x0000:0000 to 0x0000:7FFF. The data TCM is located at 0x0001:0000 to 0x0001:7FFF, as shown in Table 3-3.

Table 3-3. ITCM/DTCM Memory Map

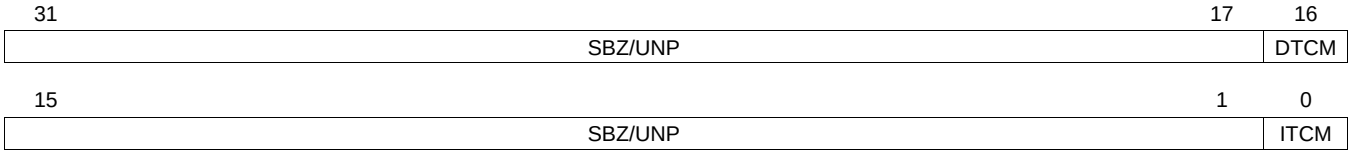
I-TCM Address	D-TCM Address	Size (Bytes)	Description
0x0000 :0000 - 0x0000 :3FFF	0x0001 :0000 - 0x0001 :3FFF	16K	IRAM0
0x0000 :4000 - 0x0000 :7FFF	0x0001 :4000 - 0x0001 :7FFF	16K	IRAM1
0x0000 :8000 - 0x0000 :BFFF	0x0001 :8000 - 0x0001 :8FFF	8K	ROM
0x0000 :C000 - 0x0000 :FFFF	0x0001 :C000 - 0x000F :FFFF	24K	Reserved

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

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

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

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



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

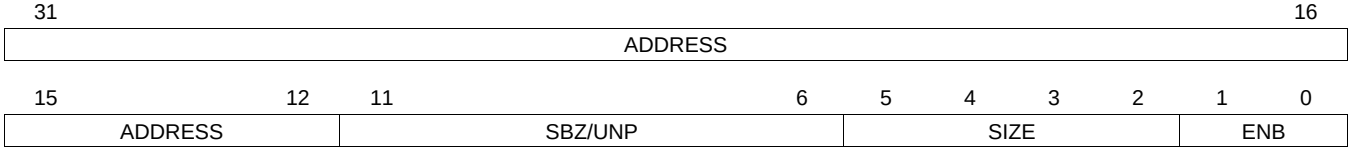
Use the ITCM / DTCM region registers to enable ITCM and DTCM.

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

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

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

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



Write 0 to the ENB bit to enable ITCM and DTCM. Write 1 to the ENB bit to enable it. The physical address of the memory should be set to the ADDRESS field. The SIZE field reflects the size. The size encoding is given below in Table 3-4.

Table 3-4. ITCM/DTCM Size Encoding

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

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

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

3.8 Embedded Trace Support

To support real-time trace, the ARM926EJ-S processor provides an interface to enable connection of an Embedded Trace Macrocell (ETM). The ARM926ES-J Subsystem in the device also includes the Embedded Trace Buffer (ETB).

The ETM consists of two parts: the trace port and triggering facilities. The two ETM parts are shown in Table 3-5.

NOTE: The device trace port is not pinned out. Instead, it is connected to a 4KB Embedded Trace Buffer. ETB enabled debug tools are required to read/interpret the captured trace data.

Table 3-5. ETM Part Descriptions

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

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

NOTE: See Chapter 10 of the Embedded Trace Macro-cell Support of the ARM926EJ-S TRM, downloadable from http://www.arm.com for more detailed information.

4.1 Memory Map

The device memory map is shown in Table 4-1. The multiple columns represent the memory map of each of the masters on the chip. The ARM, EDMA, USB, and VPSS are all masters with access to the regions shown in the table. The table's memory size column shows the size of the decoded block; it may not show the physical memory range. There may be replication of the contents within the block (as with the 8K ARM ROM) or unused address space within the block.

Table 4-1. Memory Map

Start Address	End Address	Size (Bytes)	ARM Mem Map	EDMA Mem Map	USB Mem Map	VPSS Mem Map
0x0000 0000	0x0000 3FFF	16K	ARM RAM0 (Instruction)	Reserved	Reserved	Reserved
0x0000 4000	0x0000 7FFF	16K	ARM RAM1 (Instruction)			
0x0000 8000	0x0000 FFFF	32K	ARM ROM (Instruction) - only 8K used			
0x0001 0000	0x0001 3FFF	16K	ARM RAM0 (Data)	ARM RAM0	ARM RAM0	
0x0001 4000	0x0001 7FFF	16K	ARM RAM1 (Data)	ARM RAM1	ARM RAM1	
0x0001 8000	0x0001 FFFF	32K	ARM ROM (Data) - only 8K used	ARM ROM	ARM ROM	
0x0002 0000	0x000F FFFF	896K	Reserved	Reserved	Reserved	
0x0010 0000	0x01BB FFFF	26M				
0x01BC 0000	0x01BC 0FFF	4K	ARM ETB Mem			
0x01BC 1000	0x01BC 17FF	2K	ARM ETB Reg			
0x01BC 1800	0x01BC 18FF	256	ARM IceCrusher			
0x01BC 1900	0x01BC FFFF	59136	Reserved			
0x01BD 0000	0x01BF FFFF	192K				
0x01C0 0000	0x01FF FFFF	4M	CFG Bus Peripherals	CFG Bus Peripherals		
0x0200 0000	0x09FF FFFF	128M	ASYNC EMIF (Data)	ASYNC EMIF (Data)		
0x0A00 0000	0x11EF FFFF	127M - 16K	Reserved	Reserved		
0x11F0 0000	0x11F1 FFFF	128K				
0x11F2 0000	0x1FFF FFFF	141M-64K				
0x2000 0000	0x2000 7FFF	32K	DDR EMIF Control Regs	DDR EMIF Control Regs		
0x2000 8000	0x41FF FFFF	544M-32K	Reserved	Reserved		
0x4200 0000	0x49FF FFFF	128M		Reserved		
0x4A00 0000	0x7FFF FFFF	864M		Reserved		
0x8000 0000	0x8FFF FFFF	256M	DDR EMIF	DDR EMIF	DDR EMIF	DDR EMIF
0x9000 0000	0xFFFF FFFF	1792M	Reserved	Reserved	Reserved	Reserved

4.1.1 ARM Internal Memories

The ARM has access to the following ARM internal memories:

- 32KB ARM Internal RAM on TCM interface, logically separated into two 16KB pages to allow simultaneous access on any given cycle, if there are separate accesses for code (I-TCM bus) and data (D-TCM) to the different memory regions.
- 8KB ARM Internal ROM

NOTE: By default, ARM access to internal memory is with one wait-state. However, if the ARM clock frequency is less than or equal to 150 MHz, you may configure ARM access to internal memory to be zero wait-state. To configure the wait-state use the bit AIM_WAIST in the Miscellaneous Control register (MISC) in the System Control Module. MISC is described in Section 9.10.15.

4.1.2 External Memories

The ARM has access to the following external memories:

- DDR2 / mDDR Synchronous DRAM
- Asynchronous EMIF/OneNand
- NAND Flash
- External host devices

Additionally, the ARM has access to the various common media storage card interfaces.

4.1.3 MPEG/JPEG Coprocessor (MJCP)

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

4.1.4 Peripherals

The ARM and EDMA have access to the registers and memories of the following peripherals (see Table 4-2):

- EDMA Controller
- Three UARTs
- I2C (Inter-IC Communication)
- Three 64-bit timers (each configurable as one 64-bit timer or two 32 bit timers) and one WDT
- PWM (Pulse Width Modulator)
- USB (Universal Serial Bus Controller)
- Two Audio Serial Ports (ASP)
- Three SPI serial interfaces
- General-Purpose Input/Output (GPIO)
- Video Processing Subsystem (VPSS)
- Asynchronous EMIF (AEMIF) Controller
- Real Time Out (RTO)

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

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

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

Table 4-2. ARM Configuration Bus Access to Peripherals

	Address			Accessibility	
Region	**Start**	**End**	**Size**	**ARM**	**EDMA**
EDMA CC	0x01C0 0000	0x01C0 FFFF	64K	√	√
EDMA TC0	0x01C1 0000	0x01C1 03FF	1K	√	√
EDMA TC1	0x01C1 0400	0x01C1 07FF	1K	√	√
Reserved	0x01C1 0800	0x01C1 9FFF	38K	√	√
Reserved	0x01C1 A000	0x01C1 FFFF	24K	√	√
UART0	0x01C2 0000	0x01C2 03FF	1K	√	√
UART1	0x01C2 0400	0x01C2 07FF	1K	√	√
Timer4/5	0x01C2 0800	0x01C2 0BFF	1K	√	√
Real-time out	0x01C2 0C00	0x01C2 0FFF	1K	√	√
I2C	0x01C2 1000	0x01C2 13FF	1K	√	√
Timer0/1	0x01C2 1400	0x01C2 17FF	1K	√	√
Timer2/3	0x01C2 1800	0x01C2 1BFF	1K	√	√
WatchDog Timer	0x01C2 1C00	0x01C2 1FFF	1K	√	√
PWM0	0x01C2 2000	0x01C2 23FF	1K	√	√
PWM1	0x01C2 2400	0x01C2 27FF	1K	√	√
PWM2	0x01C2 2800	0x01C2 2BFF	1K	√	√
PWM3	0x01C2 2C00	0x01C2 2FFF	1K	√	√
System Module	0x01C4 0000	0x01C4 07FF	2K	√	√
PLL Controller 0	0x01C4 0800	0x01C4 0BFF	1K	√	√
PLL Controller 1	0x01C4 0C00	0x01C4 0FFF	1K	√	√
Power/Sleep Controller	0x01C4 1000	0x01C4 1FFF	4K	√	√
Reserved	0x01C4 2000	0x01C4 7FFF	24K	√	√
ARM Interrupt Controller	0x01C4 8000	0x01C4 83FF	1K	√	√
Reserved	0x01C4 8400	0x01C6 3FFF	111K	√	√
USB OTG 2.0 Regs / RAM	0x01C6 4000	0x01C6 5FFF	8K	√	√
SPI0	0x01C6 6000	0x01C6 67FF	2K	√	√
SPI1	0x01C6 6800	0x01C6 6FFF	2K	√	√
GPIO	0x01C6 7000	0x01C6 77FF	2K	√	√
SPI2	0x01C6 7800	0x01C6 FFFF	2K	√	√
VPSS Subsystem	0x01C7 0000	0x01C7 FFFF	64K	√	√
VPSS Clock Control	0x01C7 0000	0x01C7 007F	128	√	√
Hardware 3A	0x01C7 0080	0x01C7 00FF	128	√	√
Image Pipe (IPIPE) Interface	0x01C7 0100	0x01C7 01FF	256	√	√
On Screen Display	0x01C7 0200	0x01C7 02FF	256	√	√
Reserved	0x01C7 0300	0x01C7 03FF	256	√	√
Video Encoder	0x01C7 0400	0x01C7 05FF	512	√	√
CCD Controller	0x01C7 0600	0x01C7 07FF	256	√	√
VPSS Buffer Logic	0x01C7 0800	0x01C7 08FF	256	√	√
Reserved	0x01C7 0900	0x01C7 09FF	256	√	√
Image Pipe (IPIPE)	0x01C7 1000	0x01C7 3FFF	12K	√	√
Reserved	0x01C7 4000	0x01CD FFFF	432K	√	√
Multimedia / SD 1	0x01E0 0000	0x01E0 1FFF	8K	√	√
ASP0	0x01E0 2000	0x01E0 3FFF	8K	√	√
ASP1	0x01E0 4000	0x01E0 5FFF	8K	√	√

Table 4-2. ARM Configuration Bus Access to Peripherals (continued)

	Address			Accessibility	
UART2	0x01E0 6000	0x01E0 63FF	1K	√	√
Reserved	0x01E0 6400	0x01E0 FFFF	39K	√	√
ASYNC EMIF Control	0x01E1 0000	0x01E1 0FFF	4K	√	√
Multimedia / SD 0	0x01E1 1000	0x01E1 FFFF	60K	√	√
Reserved	0x01E2 0000	0x01FF FFFF	1792K	√	√
ASYNC EMIF Data (CE0)	0x0200 0000	0x03FF FFFF	32M	√	√
ASYNC EMIF Data (CE1)	0x0400 0000	0x05FF FFFF	32M	√	√
Reserved	0x0600 0000	0x09FF FFFF	64M	√	√
Reserved	0x0A00 0000	0x0BFF FFFF	32M	√	√
Reserved	0x0C00 0000	0x0FFF FFFF	64M	√	√

4.2 Memory Interfaces Overview

This section describes the different memory interfaces.

The device supports several memory and external device interfaces, including the following:

- DDR2 / mDDR Synchronous DRAM
- Asynchronous EMIF
- NAND Flash
- OneNAND flash

4.2.1 DDR2 EMIF

The DDR2 EMIF interface, sometimes referred to as EMIF3.0 in the device documentation, is a dedicated interface to DDR and MDDR SDRAM. It supports JESD79D-2A standard compliant DDR2 SDRAM devices and supports only 16-bit interfaces.

DDR SDRAM plays a key role in a DM355-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 VPFE,
- Numerous OSD display buffers
- Intermediate buffering for large raw Bayer data image files while performing still camera processing functions
- Buffering for intermediate data while performing video encode and decode functions
- Storage of executable firmware for the ARM

4.2.2 External Memory Interface

The external memory interface (EMIF) provides an 8-bit or 16-bit data bus, an address bus width of up to 14-bits, and 2 dedicated chip selects, along with memory control signals. The EMIF module supports:

- Asynchronous memories (SRAM, Linear flash, etc.)
- NAND flash memories
- OneNAND flash memories

4.2.2.1 Asynchronous EMIF (AEMIF)

The asynchronous EMIF (AEMIF) mode supports the following features:

- SRAM on up to two asynchronous chip selects
- Supports 8-bit or 16-bit data bus widths
- Programmable asynchronous cycle timings
- Supports extended waits
- Supports Select Strobe mode
- Supports booting the device ARM processor from CE0 (e.g., SRAM) via direct execution

4.2.2.2 NAND (NAND, SmartMedia, xD)

The NAND mode supports the following features:

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

4.2.2.3 OneNAND

The OneNAND mode supports the following features:

- OneNAND Flash on up to two chip selects
- Supports only 16-bit data bus widths
- Supports asynchronous writes and reads
- Supports synchronous reads with continuous linear burst mode
- Does not support synchronous reads with wrap burst modes
- Programmable cycle timings for each chip select in asynchronous mode
- Supports booting of the device ARM processor from OneNAND-Flash located at CE0 via direct execution

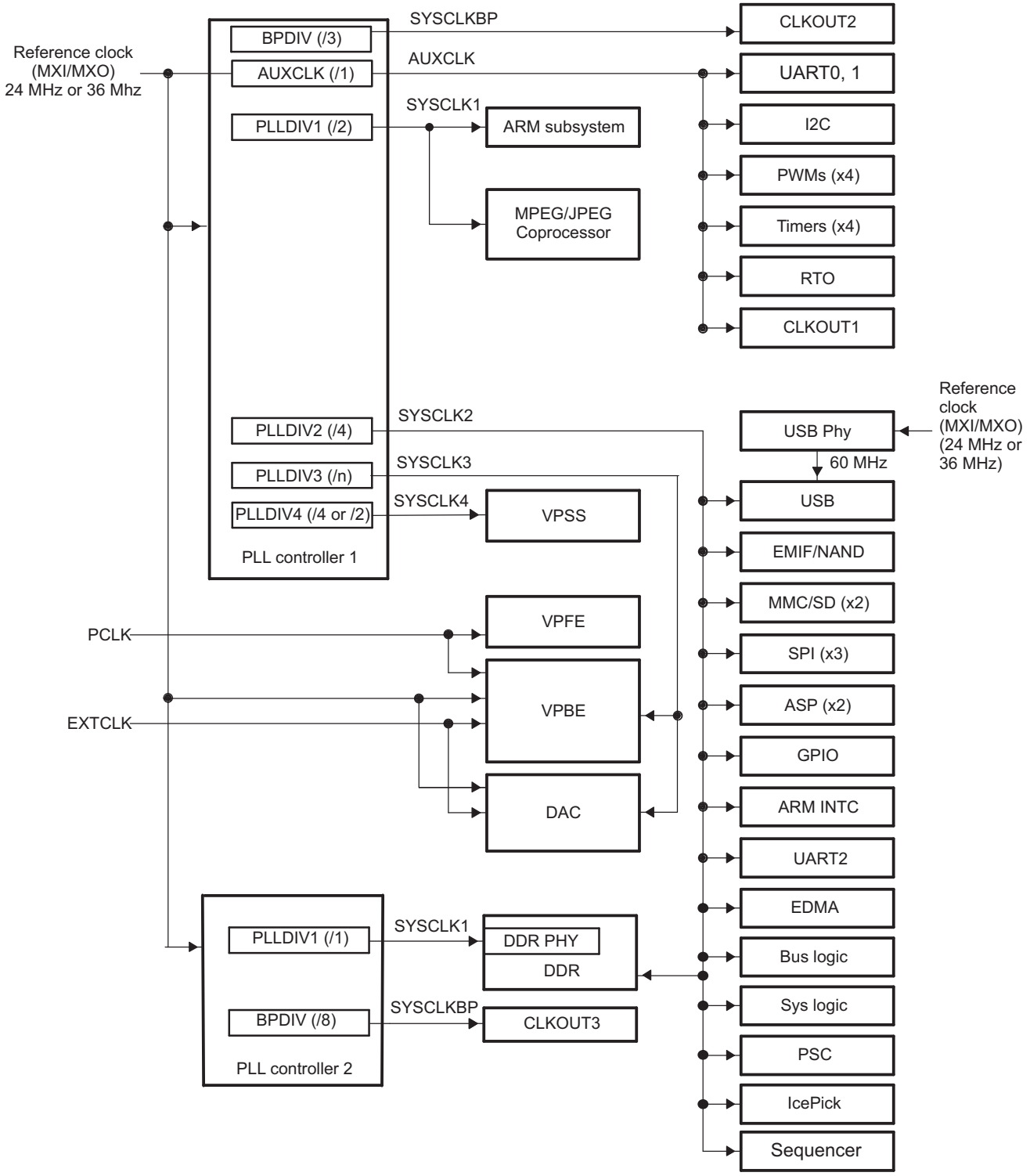
Device Clocking

5.1 Overview

The device requires one primary reference clock. The reference clock frequency may be generated either by crystal input or by external oscillator. The reference clock is the clock at the pins named MXI1/MXO1. The reference clock drives two separate PLL controllers (PLLC1 and PLLC2). PLLC1 generates the clocks required by the ARM, VPBE, VPSS, and peripherals. PLL2 generates the clock required by the DDR PHY. A block diagram of the clocking architecture is shown in Figure 5-1. The PLLs are described further in Chapter 6.

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

Figure 5-1. Clocking Architecture



5.2 Peripheral Clocking Considerations

5.2.1 Video Processing Back End Clocking

The Video Processing Back End (VPBE) is a sub-module of the Video Processing Subsystem (VPSS). The VPBE is designed to interface with a variety of LCDs and an internal DAC module. There are two asynchronous clock domains in the VPBE: an internal clock domain and an external clock domain. The internal clock domain is driven by the VPSS clock (PLL1 SYSCLK4). The external clock domain is configurable; you can select one of five sources:

- 24 MHz crystal input at MXI1
- 27 MHz crystal input at MXI2 (optional feature)
- PLL1 SYSCLK3
- EXTCLK pin (external VPBE clock input pin)
- PCLK pin (VPFE pixel clock input pin)

For complete information on VPBE clocking, see the *TMS320DM355 Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Reference Guide* (SPRUF72).

5.2.2 USB Clocking

The USB Controller is driven by two clocks: an output clock of PLL1 and an output clock of the USB Phy. The USB Phy clock is configurable by the USB Phy clock source bits (PHYCLKSRC) in the USB Phy control register (USB_PHY_CTL) in the System Control Module. USBPHY_CTL is described in Chapter 9. When a 24-MHz crystal is used at MXI1/MXO1, set PHYCLKSRC to 0. This will present a 24-MHz clock to the USB Phy. When a 36-MHz crystal is used at MXI1/MXO1, set PHYCLKSRC to 1. This will present a 12-MHz (36 MHz divided by three) crystal to the USB Phy. The USB Phy is capable of accepting only 24 MHz and 12 MHz; thus for USB support you must use either a 24-MHz or 36-MHz crystal at MXI1/MXO1. For more information, see the *TMS320DM355 Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Reference Guide* (SPRUED2).

PLL Controllers (PLLCs)

6.1 PLL Controller Module

Two PLL controllers provide clocks to different components of the chip. PLL controller 1 (PLLC1) provides clocks to most of the components of the chip. PLL controller 2 (PLLC2) provides clocks to the DDR PHY.

As a module, the PLL controller provides the following:

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

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

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

Various dividers that can be used are as follows:

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

Multipliers supported are as follows:

- PLL multiplier control: PLLM

6.2 PLLC1

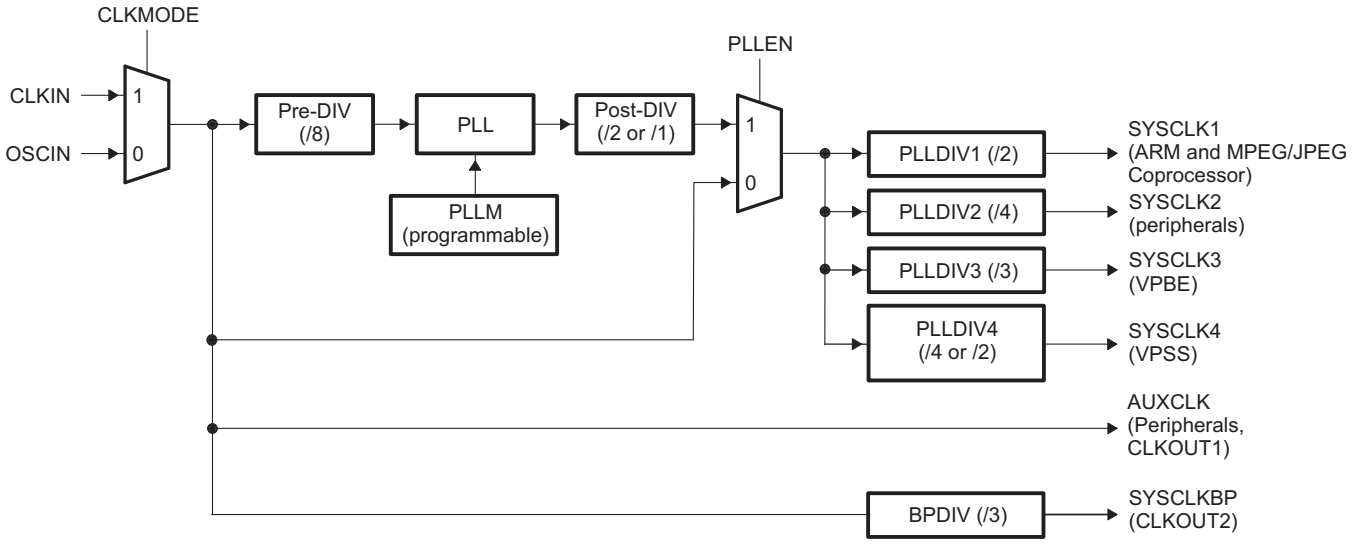
PLLC1 provides most of the device clocks. Software controls PLLC1 operation through the PLLC1 registers. The following list, Table 6-1, and Figure 6-1 describe the customizations of PLLC1.

- Provides primary device system clock
- Software configurable
- Accepts clock intput or internal oscillator input
- PLL pre-divider value is fixed to (/8)
- PLL multiplier value is programmable
- PLL post-divider value is programmable to either (/1) or (/2). See the device-specific data manual for all supported configurations.
- Only SYSCLK[4:1] are used
- SYSCLK1 divider value is fixed to (/2)
- SYSCLK2 divider value is fixed to (/4)
- SYSCLK3 divider value is programmable
- SYSCLK4 divider value is programmable (program to (/4) or (/2). See the device-specific data manual for all supported configurations.
- SYSCLKBP divider value is fixed to (/3)
- SYSCLK1 is routed to the ARM Subsystem
- SYSCLK2 is routed to peripherals
- SYSCLK3 is routed to the VPBE module
- SYSCLK4 is routed to the VPSS module
- AUXCLK is routed to peripherals with fixed clock domain and also to the output pin CLKOUT1
- SYSCLKBP is routed to the output pin CLKOUT2

Table 6-1. PLLC1 Output Clocks

Output Clock	Used By	PLLDIV Divider	Notes
SYSCLK1	ARM Subsystem	/2	Fixed divider
SYSCLK2	Peripherals	/4	Fixed divider
SYSCLK3	VPBE (VENC module)	/n	Programmable divider (used to get 27 MHz for VENC)
SYSCLK4	VPSS	/4 or /2	Programmable divider
AUXCLK	Peripherals, CLKOUT1	none	No divider
SYSCLKBP	CLKOUT2	/3	Fixed divider

Figure 6-1. PLLC1 Configuration



6.3 PLLC2

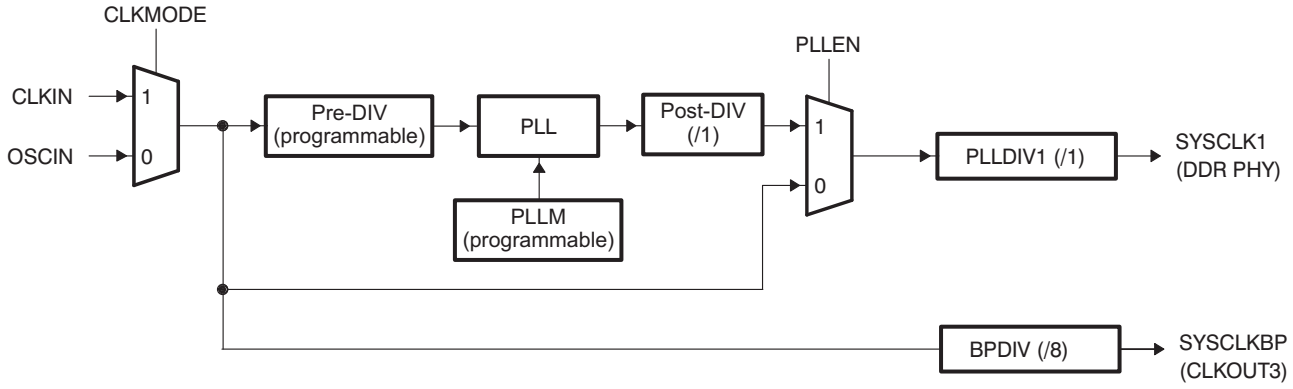
PLLC2 provides the DDR PHY clock and CLKOUT3. Software controls PLLC2 operation through the PLLC2 registers. The following list, Table 6-2, and Figure 6-2 describe the customizations of PLLC2.

- Provides DDR PHY clock and CLKOUT3
- Software configurable
- Accepts clock input or internal oscillator input (same input as PLLC1)
- PLL pre-divider value is programmable
- PLL multiplier value is programmable
- PLL post-divider value is fixed to (/1)
- Only SYSCLK[1] is used
- SYSCLK1 divider value is fixed to (/1)
- SYSCLKBP divider value is fixed to (/8)
- SYSCLK1 is routed to the DDR PHY
- SYSCLKBP is routed to the output pin CLKOUT3
- AUXCLK is not used.

Table 6-2. PLLC2 Output Clocks

Output Clock	Used by	PLLDIV Divider	Notes
SYSCLK1	DDR PHY	/1	Fixed divider
SYSCLKBP	CLKOUT3	/8	Fixed divider

Figure 6-2. PLLC2 Configuration



6.4 PLLC Functional Description

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

6.4.1 Multipliers and Dividers

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

6.4.2 Bypass Mode

The multiplier PLLM, pre-divider PREDIV, post-divider POSTDIV, and the PLL may be bypassed altogether. The PLL enable bit (PLLEN) in the PLL control/status register (PLLCTL) determines the PLL controller mode. When PLLEN = 1, PLL mode is enabled and PLLM, PREDIV, POSTDIV, and the PLL are used; when PLLEN = 0, bypass mode is enabled and PLLM, PREDIV, POSTDIV, and the PLL are bypassed. When bypass mode is enabled, the input reference clock is directly input to the system clock dividers (PLLDIVn). The PLL controller defaults, after reset, to bypass mode.

6.4.3 PLL Mode

When in PLL mode (PLLEN = 1), the input reference clock is supplied to divider PREDIV. Divider PREDIV must be enabled (PREDEN = 1) in PLL mode. When divider PREDIV is enabled, the input reference clock is divided down by the value in the PLL divider ratio bits (RATIO) in PREDIV. The output from divider PREDIV is input to the PLL. The PLL multiplies the clock by the value in the PLL multiplier bits (PLLM) in the PLL multiplier control register (PLLM). The output from the PLL (PLLOUT) is input to the divider POSTDIV. Divider POSTDIV must be enabled (POSTEN = 1) in PLL mode. When divider POSTDIV is enabled, the output from the PLL (PLLOUT) is divided down by the value in the PLL divider ratio bits (RATIO) in POSTDIV. The output from divider POSTDIV is input to the system clock dividers (PLLDIVn). When enabled (bit DnEN = 1), a system clock divider (PLLDIVn) divides-down the output clock of the PLL by the value in the PLL divider ratio bits (RATIO) in PLLDIVn. The system clock dividers generate 50% duty cycle output clocks SYSCLKn.

6.5 PLL Configuration

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

6.5.1 PLL Mode and Bypass Mode

6.5.1.1 PLL Mode (PLLEN = 1)

This section describes the sequence for PLL mode.

1. In PLLCTL, write CLKMODE = 0 (internal oscillator) or 1 (input clock) to select the type of reference clock. Write CLKMODE = 0 for internal oscillator. Write CLKMODE = 1 for square wave input clock.
2. In PLLCTL, write PLLENSRC = 0 (enable PLL enable). The bit PLLEN in PLLCTL has no effect unless you write PLLENSRC = 0.
3. In PLLCTL, write PLLEN = 0 (bypass mode).
4. Wait at least 4 reference clock cycles for the PLLEN mux to change.
5. In PLLCTL, write PLLRST = 1 (assert PLL reset).
6. In PLLCTL, write PLLDIS = 1 (assert PLL disable).
7. In PLLCTL, write PLLPWRDN = 0 (power up the PLL).
8. In PLLCTL, write PLLDIS = 0 (de-assert PLL disable).
9. If necessary, write PREDIV, POSTDIV, and PLLM to set divider and multiplier values.
10. If necessary, write PLLDIV to set PLLDIVn dividers. Note that you must apply the GO operation to change these dividers to new ratios. See Section 6.5.2.1.
11. Wait at least 5 miro-seconds for the PLL reset.
12. In PLLCTL, write PLLRST = 0 (de-assert PLL reset).
13. Wait at least 8000 reference clock cycles for the PLL to lock.
14. In PLLCTL, write PLLEN = 1 to (switch from bypass mode to PLL mode).

6.5.1.2 Bypass Mode (PLLEN = 0)

This section describes the sequence for bypass mode.

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

6.5.2 Changing Divider / Multiplier Ratios

This section describes how to change divider and multiplier values.

6.5.2.1 PLLDIVn and GO Operation

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

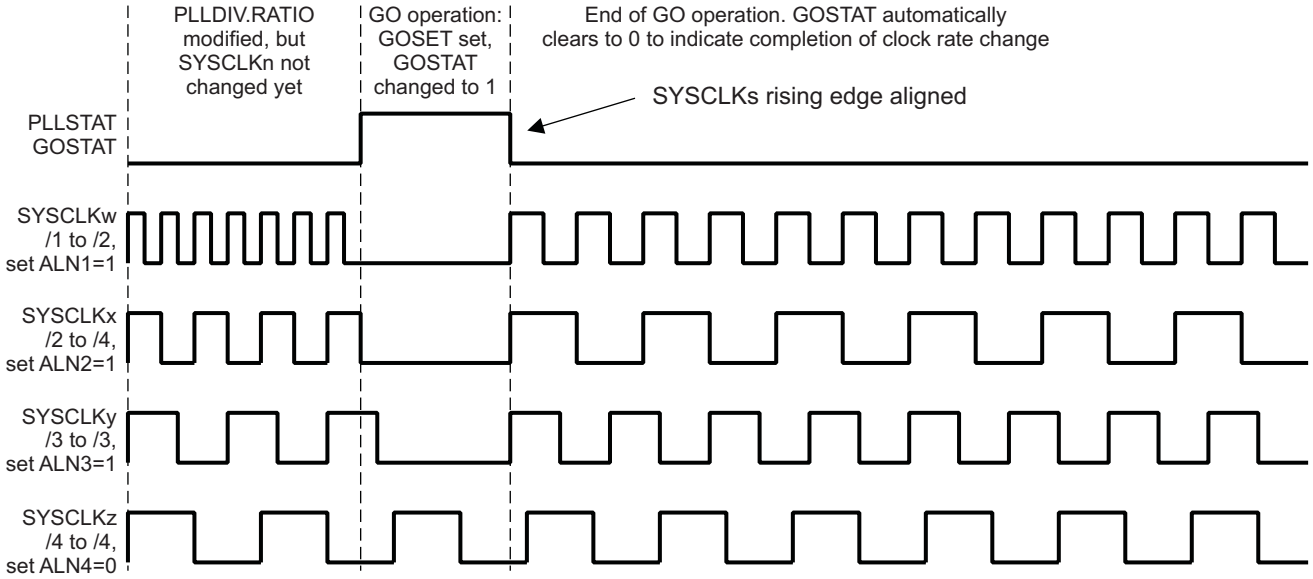
6.5.2.1.1 GO Operation

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

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

Figure 6-3 is an example showing how the clocks are rising-edge aligned during a GO operation. Notice that even though the SYSCLKy ratio remains the same, it is still stopped since ALN3 = 1 in ALNCTL.

Figure 6-3. Clock Ratio Change and Alignment with Go Operation



6.5.2.1.2 Software Steps to Modify PLLDIVn Ratios

To modify the PLLDIVn ratios, perform the following steps:

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

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

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

1. In PLLCTL, write PLLEN = 0 to place the PLL in bypass mode.
2. Wait at least 4 reference clock cycles for the PLLEN mux to change.
3. In PLLCTL, write PLLRST = 1 (assert PLL).
4. Modify PREDIV, PLLM, and/or POSTDIV ratios.
5. Wait at least 5 miro-seconds for the PLL reset.
6. In PLLCTL, write PLLRST = 0 (de-assert PLL reset)
7. Wait at least 8000 reference clock cycles for the PLL to lock.
8. In PLLCTL, write PLLEN = 1 (switch from bypass mode to PLL mode).

6.5.3 PLL Power Down and Wakeup

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

Perform the following procedure to power down the PLL.

1. In PLLCTL, write PLLEN = 0 (bypass mode).
2. Wait at least 4 reference clock cycles for the PLLEN mux to change.
3. In PLLCTL, write PLLPWRDN = 1 to power down the PLL.

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

6.6 PLL Controller Registers

Table 6-3 lists the base address for the PLLC1 and PLLC2 registers. Table 6-4 lists the memory-mapped registers for PLLC1 and PLLC2. Also, see the device memory map Table 4-2 for the base addresses of these registers.

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

Instance ID	Base Address	End Address	Size
PLLC1	0x1C4 0800	0x1C4 0BFF	0x 400
PLLC2	0x1C4 0C00	0x1C4 0FFF	0x 400

Table 6-4. PLLC Registers

Offset	Acronym	Register Description	Section
00h	PID	Contains peripheral ID and revision information	Section 6.6.1
100h	PLLCTL	Controls PLL operations	Section 6.6.2
110h	PLLM	PLL Multiplier Control	Section 6.6.3
114h	PREDIV	Pre-divider control	Section 6.6.4
118h	PLLDIV1	Divider 1 control-divider for SYSCLK1	Section 6.6.5
11Ch	PLLDIV2	Divider 2 control-divider for SYSCLK2	Section 6.6.6
120h	PLLDIV3	Divider 3 control-divider for SYSCLK3	Section 6.6.7
128h	POSTDIV	Post-divider control	Section 6.6.8
12Ch	BPDIV	Bypass divider control	Section 6.6.9
138h	PLLCMD	PLL controller command register	Section 6.6.10
13Ch	PLLSTAT	PLL controller status register	Section 6.6.11
140h	ALNCTL	SYSCLKn divider ratio change and align control register	Section 6.6.12
144h	DCHANGE	PLL divider ratio change status register	Section 6.6.13
148h	CKEN	Clock enable control AUXCLK	Section 6.6.14
14Ch	CKSTAT	Clock status for SYSCLKBP and AUXCLK	Section 6.6.15
150h	SYSTAT	Clock status for SYSCLKn clocks	Section 6.6.16
160h	PLLDIV4	Divider 4 control-divider for SYSCLK4	Section 6.6.17

6.6.1 Peripheral ID Register (PID)

The peripheral ID register (PID) is shown in Figure 6-4 and described in Table 6-5 for PLLC1 and PLLC2. Note that bit field descriptions shown in Figure 6-4 are given for PLLC1 (top) and PLLC2 (bottom). This format is used in the bit description figures throughout this section.

Figure 6-4. Peripheral ID Register (PID)

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

LEGEND: R = Read, W = Write, n = value at reset

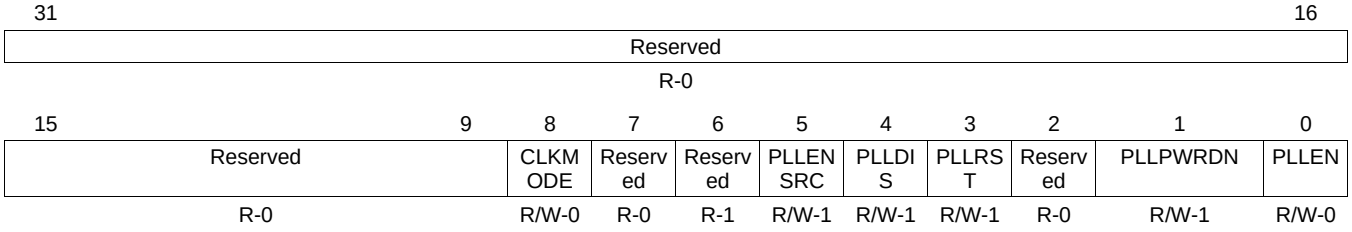
Table 6-5. Peripheral ID Register (PID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	TYPE	0-FFh	Peripheral Type: 0x01 to identify as PLLC
15-8	CLASS	0-FFh	Peripheral Class: 0x08
7-0	REV	0-FFh	Peripheral Revision: 0x0D

6.6.2 PLL Control (PLLCTL)

The PLL control register is shown in Figure 6-5 and described in Table 6-6 for PLLC1 and PLLC2.

Figure 6-5. PLL Control Register (PLLCTL)



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

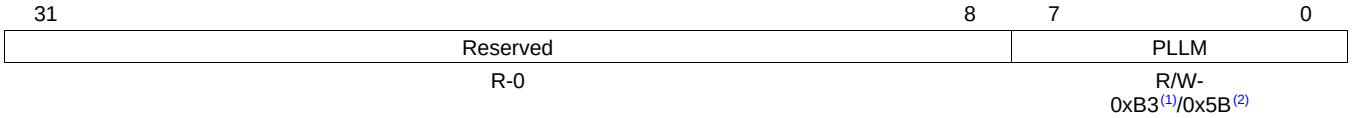
Table 6-6. PLL Control Register (PLLCTL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKMODE		Reference Clock Selection. This bit has no effect for PLLC2. In the device, a single oscillator or CLKIN square wave is input to both PLLC1 and PLLC2.
		0	Internal oscillator
		1	CLKIN square wave
7-6	Reserved	0	Reserved
5	PLLENSRC		PLL enable source. This bit must be cleared to 0 before PLLCTL.PLLEN will have any effect.
		0	PLL enable is controlled by the register bit PLLCTL.PLLEN
		1	PLL enable is controlled by internal test hardware
4	PLLDIS		PLL disable
		0	PLL disable de-assert
		1	PLL disable assert
3	PLLRST		PLL reset
		0	PLL reset de-assert
		1	PLL reset assert
2	Reserved	0	Reserved
1	PLLPWRDN		PLL power-down
		0	PLL operating, not powered down
		1	PLL power-down
0	PLLEN		PLL Mode Enable. Bit PLLCTL.PLLENSRC must be cleared to 0 before PLLCTL.PLLEN will have any effect.
		0	Bypass mode
		1	PLL mode, not bypassed

6.6.3 PLL Multiplier Control Register (PLLM)

The PLL multiplier control register (PLLM) is shown in Figure 6-6 and described in Table 6-7 for PLLC1 and PLLC2. For PLLC1, the default multiplier value is 180. For PLLC2, the default multiplier value is 92. You may change the multiplier value from 92 to 184.

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



LEGEND: R = Read, W = Write, n = value at reset

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	PLLM	5Bh-B7h	PLL Multiplier. Multiplier value = PLLM + 1

6.6.4 PLL Pre-Divider Control Register (PREDIV)

The PLL pre-divider control register (PREDIV) is shown in Figure 6-7 and described in Table 6-8 for PLLC1 and PLLC2. For PLLC1, the pre-divider ratio is fixed (cannot be changed) to 8. For PLLC2, the pre-divider ratio defaults to 8, however, it may be changed to allow for lower frequencies.

Figure 6-7. PLL Pre-Divider Control Register (PREDIV)

31	16	15	14	5	4	0
Reserved		PREDEN	Reserved		RATIO	
R-0		R-1	R-0		R-7[1]/R/W-7[2]	

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

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

Table 6-8. PLL Pre-Divider Control (PREDIV) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	PREDEN		Pre-divider enable. For PLLC1 and PLLC2, this bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for post divider. Ratio value = RATIO + 1

6.6.5 PLL Controller Divider 1 Register (PLLDIV1)

The PLL controller divider 1 register (PLLDIV1) is shown in Figure 6-8 and described in Table 6-9 for PLLC1 and PLLC2. PLLDIV1 controls the divider for SYSCLK1. The divider for PLLC1 SYSCLK1 is fixed (cannot be changed) to (/2). The divider for PLLC2 SYSCLK1 is fixed (cannot be changed) to (/1). For PLLC1 the divider must always be endabled (bit D1EN=1).

Figure 6-8. PLL Controller Divider 1 Register (PLLDIV1)

31	16	15	14	5	4	0
Reserved		D1EN	Reserved		RATIO	
R-0		R/W-1[1]/-0[2]	R-0		R-1[1]/-0[2]	

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

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN		Divider enable for SYSCLK1. For PLLC1, this bit must always be set to 1. For PLLC2, this bit may be set to 0 or 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for SYSCLK1. Ratio value = RATIO + 1

6.6.6 PLL Controller Divider 2 Register (PLLDIV2)

The PLL controller divider 2 register (PLLDIV2) is shown in Figure 6-9 and described in Table 6-10 for PLLC1 and PLLC2. PLLDIV2 controls the divider for SYSCLK2. The divider for PLLC1 SYSCLK2 is fixed (cannot be changed) to (/4). The divider for PLLC2 SYSCLK2 is fixed (cannot be changed) to (/2). For PLLC1 and PLLC2, the divider must always be enabled (bit D2EN=1).

Figure 6-9. PLL Controller Divider 2 Register (PLLDIV2)

31	16	15	14	5	4	0
Reserved		D2EN	Reserved		RATIO	
R-0		R/W-1	R-0		R-3[1]/-1[2]	

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

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN		Divider enable for SYSCLK2. For PLLC1 and PLLC2, this bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for SYSCLK2. Ratio value = RATIO + 1

6.6.7 PLL Controller Divider 3 Register (PLLDIV3)

The PLL controller divider 3 register (PLLDIV3) is shown in Figure 6-10 and described in Table 6-11 for PLLC1 and PLLC2. PLLDIV3 controls the divider for SYSCLK3. The divider for PLLC1 SYSCLK3 is programmable. The default value is (/10). For PLLC1, the divider must always be enabled (bit D3EN=1). The PLLDIV3 register is not applicable to PLLC2, therefore all PLLDIV3 bit fields are reserved for PLLC2.

Figure 6-10. PLL Controller Divider 3 Register (PLLDIV3)

31	16	15	14	5	4	0
Reserved		D3EN	Reserved		RATIO	
R-0		R/W-1	R-0		R/W-0x0F	

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

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

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN		Divider enable for SYSCLK3. For PLLC1, this bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for SYSCLK3. Ratio value = RATIO + 1

6.6.8 PLL Post-Divider Control Register (POSTDIV)

The PLL post-divider control register (POSTDIV) is shown in Figure 6-11 and described in Table 6-12 for PLLC1 and PLLC2. POSTDIV is a read only register. The post divider ratio for PLLC1 may be changed by the bit PLL1_POSTDIV in the miscellaneous control register (MISC) in the System Control module. See Section 9.10.15 for a description of MISC register. By default, PLL1_POSTDIV is configured such that the post divider is equal to 2. Therefore, in order to enable higher frequencies, you must change bit PLL1_POSTDIV such that the post divider is equal to 1. But for 135Mhz devices it is not possible to change the post divider from the default value of 2, and thus the frequencies are limited. The post divider for PLLC2 is always fixed (cannot be changed) to 1.

Figure 6-11. PLL Post-Divider Control Register (POSTDIV)

31	16	15	14	5	4	0
Reserved		POSTDEN	Reserved		RATIO	
R-0		R-1	R-0		R-0	

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

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

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	POSTDEN		Post-divider enable. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for post divider. Ratio value = RATIO + 1

6.6.9 Bypass Divider Register (BPDIV)

The bypass divider register (BPDIV) is shown in Figure 6-12 and described in Table 6-13 for PLLC1 and PLLC2. BPDIV controls the divider for SYSCLKBP. The divider for PLLC1 SYSCLKBP is fixed (cannot be changed) to 3. The divider for PLLC2 SYSCLKBP is fixed (cannot be changed) to 8. For PLLC1 and PLLC2, the divider must always be enabled (bit BPDEN=1).

Figure 6-12. Bypass Divider Register (BPDIV)

31	16	15	14	5	4	0
Reserved		BPDEN	Reserved		RATIO	
R-0		R/W-1	R-0		R-2[1]/-7[2]	

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

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	BPDEN		Divider enable for bypass clock. This bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for bypass clock

6.6.10 PLL Controller Command Register (PLLCMD)

The PLL controller command register (PLLCMD) is shown in Figure 6-13 and described in Table 6-14 for PLLC1 and PLLC2. PLLCMD is used to initiate a GO operation for SYSCLKn ratio change and/or phase alignment.

Figure 6-13. PLL Controller Command Register (PLLCMD)

31	1	0
Reserved		GOSET
R-0		R/W-0

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

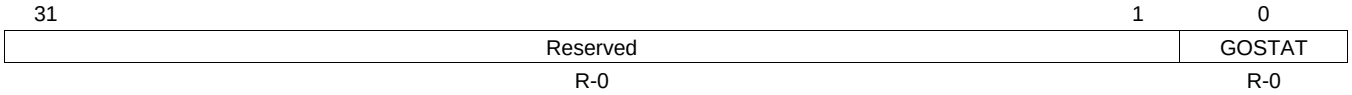
Table 6-14. PLL Controller Command Register (PLLCMD) Field Descriptions

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

6.6.11 PLL Controller Status Register (PLLSTAT)

The PLL controller status register (PLLSTAT) is shown in Figure 6-14 and described in Table 6-15 for PLLC1 and PLLC2. PLLSTAT shows the status of changing SYSCLKn divider ratios and/or phase alignment.

Figure 6-14. PLL Controller Status Register (PLLSTAT)



LEGEND: R = Read, n = value at reset

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

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSTAT		GO status
		0	GO operation is not in progress. SYSCLK divider ratios and/or phase alignment are not being changed.
		1	GO operation is in progress. SYSCLK divider ratios and/or phase alignment are changing.

6.6.12 PLL Controller Clock Align Control Register (ALNCTL)

The PLL controller clock align control register (ALNCTL) is shown in Figure 6-15 and described in Table 6-16 for PLLC1 and PLLC2. ALNCTL controls SYSCLK divider ratio change and alignment when GOSET bit in PLLCMD is set to 1. All SYSCLKn must be aligned. You should not change the defaults values of this register.

Figure 6-15. PLL Controller Clock Align Control Register (ALNCTL)

31	8	7	0
Reserved		ALNn	
R-0		R/W -0x1F[1]/-0x0[2]	

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

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	ALNn	0-1Fh	SYSCLKn divider ratio change and alignment enable. Do not change the default values of these fields. ALN0 is divider ratio change and alignment enable for SYSCLK1 ALN1 is divider ratio change and alignment enable for SYSCLK2 ALN2 is divider ratio change and alignment enable for SYSCLK3 (this bit is reserved for PLLC2) ALN3 is divider ratio change and alignment enable for SYSCLK4 (this bit is reserved for PLLC2) ALN[7:4] Reserved Do not change SYSCLKn divide ratio nor align SYSCLKn to other SYSCLKs during GO operation. SYSCLKn is left free-running when the GOSET bit in PLLCMD is set to 1. Change SYSCLKn ratio programmed in the RATIO bit in PLLDIVn and align SYSCLKn to other SYSCLKs selected in ALNCTL when the GOSET bit in PLLCMD is set to 1.

6.6.13 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLDIV ratio change status register (DCHANGE) is shown in Figure 6-16 and described in Table 6-17 for PLLC1 and PLLC2. Whenever a different ratio is written to the PLLDIVn registers, the PLLC flags the change in DCHANGE. During the GO operation, the PLL controller will only change the divide ratio of the SYSCLKs with the bit set in DCHANGE. Note that the ALNCTL register determines if that clock also needs to be aligned to other clocks.

Figure 6-16. PLLDIV Ratio Change Status (DCHANGE)

31	8	7	0
Reserved		SYSn	
R-0		R-0	

LEGEND: R = Read, n = value at reset

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

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SYSn	0-1Fh	SYSCLKn divider ratio has been modified status. When SYSn is 1, this bit indicates SYSCLKn ratio will be modified during GO operation. SYS0 shows divider ratio has been modified for SYSCLK1 SYS1 shows divider ratio has been modified for SYSCLK2 SYS2 shows divider ratio has been modified for SYSCLK3 (this bit is reserved for PLLC2) SYS3 shows divider ratio has been modified for SYSCLK4 (this bit is reserved for PLLC2) SYS[7:4] Reserved SYSCLKn divider ratio has not been modified. When PLLCMD.GOSET is set, SYSCLKn is not affected. SYSCLKn divider ratio has been modified. When PLLCMD.GOSET is set, SYSCLKn divider ratio will get updated.

6.6.14 Clock Enable Control Register (CKEN)

The clock enable control register (CKEN) is shown in Figure 6-17 and described in Table 6-18 for PLLC1 and PLLC2. The CKEN register is used to enable the PLL auxiliary clock (AUXCLK). The auxiliary clock should always be enabled, so you must always set this bit to 1. PLLC2 does not use the auxiliary clock, so the CKEN register is not applicable to PLLC2 and all CKEN bit fields are reserved for PLLC2.

Figure 6-17. Clock Enable Control Register (CKEN)

31	1	0
Reserved		AUXEN
R-0		R/W-1

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		Auxiliary clock (AUXCLK) enable. For PLLC1, this bit should always be set to 1.
		0	Disable
		1	Enable

6.6.15 Clock Status Register (CKSTAT)

The clock status (CKSTAT) register is shown in Figure 6-18 and described in Table 6-19 for PLLC1 and PLLC2. CKSTAT shows the on/off status of the bypass clock (SYSCLKBP) and the auxiliary clock (AUXCLK). PLLC2 does not use the auxiliary clock, so the AUXEN bit field is reserved for PLLC2.

Figure 6-18. Clock Status Register (CKSTAT)

31	4	3	2	1	0
Reserved		BPON	Reserved		AUXEN
R-0		R-1	R-0		R-1

LEGEND: R = Read, n = value at reset

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

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	BPON		SYSCLKBP status. Shows the clock on/off status for SYSCLKBP.
		0	Bypass clock is off
		1	Bypass clock is on
2-1	Reserved	0	Reserved
0	AUXEN		AUXCLK status. Shows the clock on/off status for AUXCLK.
		0	Aux clock is off
		1	Aux clock is on

6.6.16 SYSCLK Status Register (SYSTAT)

The SYSCLK status register (SYSTAT) is shown in Figure 6-19 and described in Table 6-20 for PLLC1 and PLLC2. SYSTAT shows the on/off status of the SYSCLKn clocks.

Figure 6-19. SYSCLK Status Register (SYSTAT)

31	8	7	0
Reserved		SYSONn	
R-0		R-7[1]/-2[2]	

LEGEND: R = Read, n = value at reset

(1) Reset value of the field of register belonging to PLLC1.
(2) Reset value of the field of register belonging to PLLC2.

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

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SYSONn	0-1Fh	SYSCLKn status. Shows the clock on/off status for SYSCLKn. SYSON0 shows clock on/off status for SYSCLK1 SYSON1 shows clock on/off status for SYSCLK2 SYSON2 shows clock on/off status for SYSCLK3 (this bit is reserved for PLLC2) SYSON3 shows clock on/off status for SYSCLK4 (this bit is reserved for PLLC2) SYSON[7:4] Reserved SYSCLKn is off SYSCLKn is on

6.6.17 PLL Controller Divider 4 Register (PLLDIV4)

The PLL controller divider 4 register (PLLDIV4) is shown in Figure 6-20 and described in Table 6-21 for PLLC1 and PLLC2. PLLDIV4 controls the divider for SYSCLK4. The divider for PLLC1 SYSCLK4 is programmable. See the device-specific data manual for all supported configurations. For PLLC1, the divider must always be enabled (bit D4EN=1). The PLLDIV4 register is not applicable to PLLC2, therefore all PLLDIV4 bit fields are reserved for PLLC2 .

Figure 6-20. PLL Controller Divider 4 Register (PLLDIV4)

31	16	15	14	5	4	0
Reserved		D4EN	Reserved		RATIO	
R-0		R/W-0	R-0		R/W-3	

LEGEND: R/W = Read/Write, R = Read; n = value at reset

Table 6-21. PLL Controller Divider 4 Register (PLLDIV4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D4EN		Divider enable for SYSCLK4. For PLLC1, this bit must always be set to 1.
		0	Disable
		1	Enable
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio for SYSCLK4. Ratio value = RATIO + 1

Power and Sleep Controller

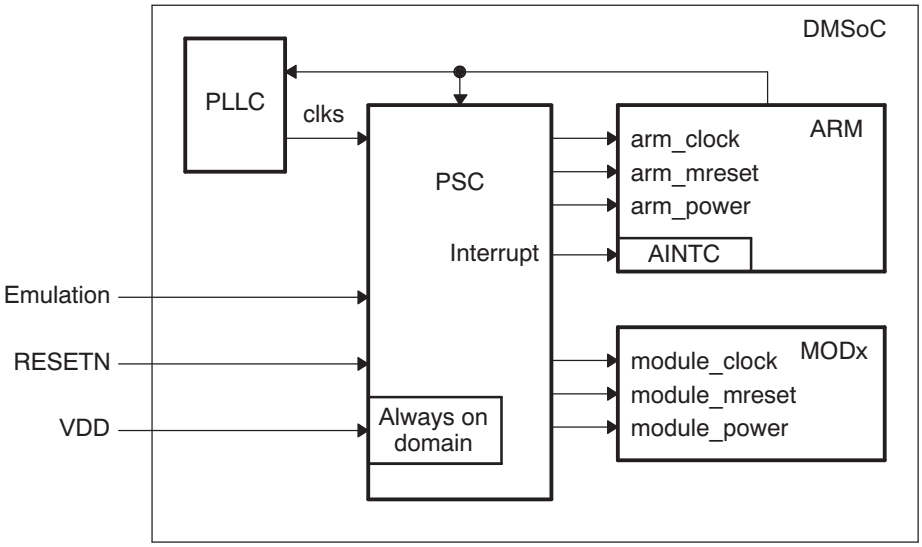
7.1 Introduction

In the DM355 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-reset operations. However, the PSC provides you with an interface to control several important clock and reset operations. The clock and reset operations are the focus of this chapter.

The PSC includes the following features:

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

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



7.2 Power Domain and Module Topology

The DM355 system includes one power domain and forty-one separate modules, as shown in Figure 7-2 and summarized in Table 7-1. The device's power domain is always on when the chip is on, and it is referred to as the AlwaysOn power domain. The AlwaysOn domain is powered by the V_{DD} pins of the device chip (see the device-specific data manual). All of the device modules lie within the AlwaysOn power domain.

Table 7-1 shows the default state of each module after reset (power-on-reset, warm reset, and max reset). These states are defined in the following sections. The default state of some modules is determined by the boot select pins BTSEL[1:0]. For example, if UART boot mode is selected (BTSEL[1:0]=11), then the default state for the UART module is Enable. See Chapter 10 and Chapter 11 for additional information on reset, default configurations, and booting.

Figure 7-2. Power Domain and Module Topology

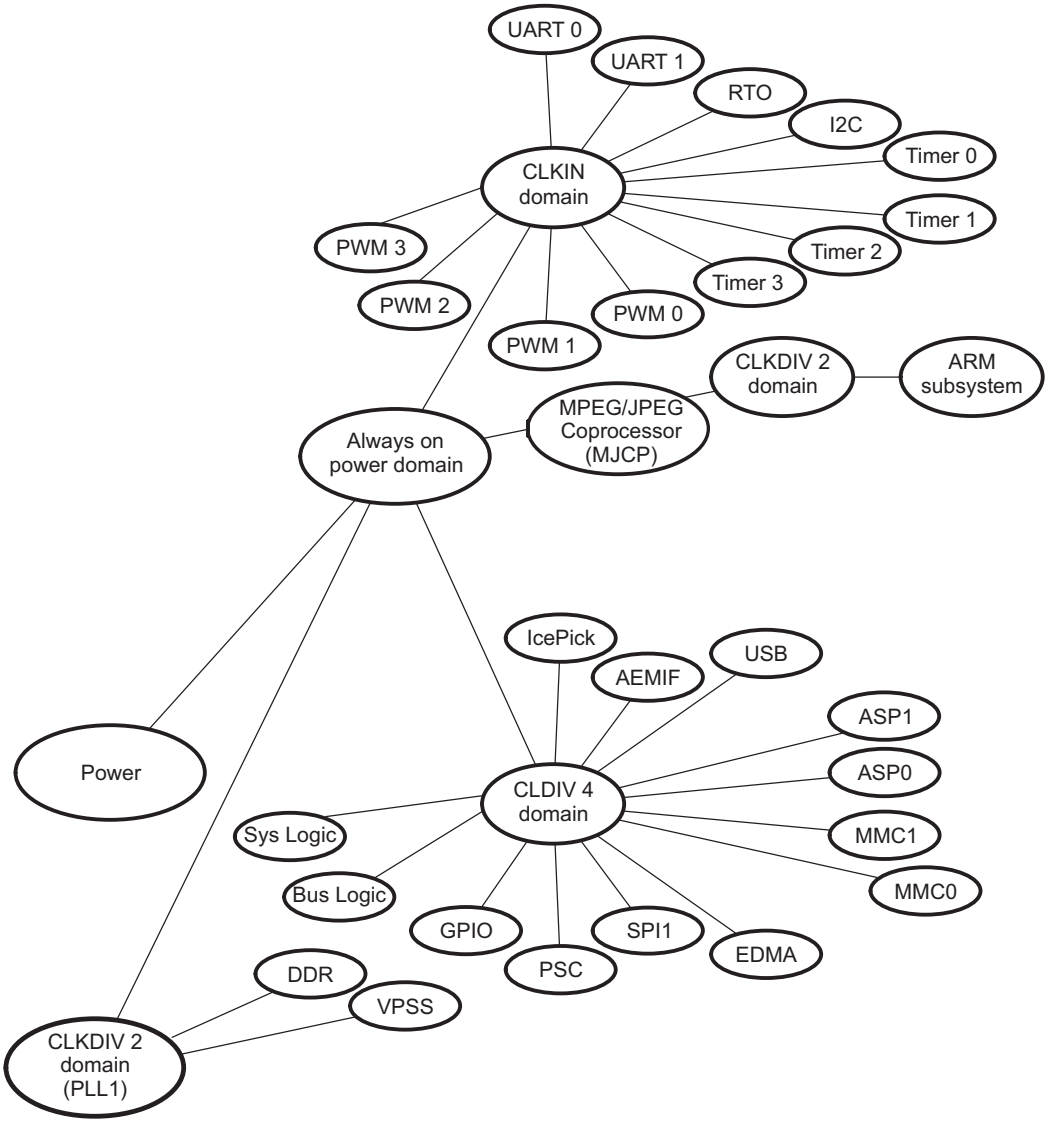


Table 7-1. Module Configuration

Module Number	Module Name	Power Domain	Default States	
			Power Domain State	Module State
0	VPSS Master	AlwaysOn	ON	SyncRst
1	VPSS Slave	AlwaysOn	ON	SyncRst
2	EDMA (CC)	AlwaysOn	ON	BTSEL[1:0] = 00 – Enable (NAND, SPI) BTSEL[1:0] = 01 – Enable (OneNAND) BTSEL[1:0] = 10 – SyncRst (MMC/SD) BTSEL[1:0] = 11 – Enable (UART)
3	EDMA (TC0)	AlwaysOn	ON	
4	EDMA (TC1)	AlwaysOn	ON	
5	Timer3	AlwaysOn	ON	SyncRst
6	SPI1	AlwaysOn	ON	SyncRst
7	MMC/SD1	AlwaysOn	ON	SyncRst
8	ASP1	AlwaysOn	ON	SyncRst
9	USB	AlwaysOn	ON	SyncRst
10	PWM3	AlwaysOn	ON	SyncRst
11	SPI2	AlwaysOn	ON	SyncRst
12	RTO	AlwaysOn	ON	SyncRst
13	DDR EMIF	AlwaysOn	ON	SyncRst
14	AEMIF	AlwaysOn	ON	BTSEL[1:0] = 00 – Enable (NAND, SPI) BTSEL[1:0] = 01 – Enable (OneNAND) BTSEL[1:0] = 10 – SyncRst (MMC/SD) BTSEL[1:0] = 11 – Enable (UART)
15	MMC/SD0	AlwaysOn	ON	BTSEL[1:0] = 00 – SyncRst (NAND, SPI) BTSEL[1:0] = 01 – SyncRst (OneNAND) BTSEL[1:0] = 10 – Enable (MMC/SD) BTSEL[1:0] = 11 – SyncRst (UART)
16	Reserved	Reserved	Reserved	Reserved
17	ASP	AlwaysOn	ON	SyncRst
18	I2C	AlwaysOn	ON	SyncRst
19	UART0	AlwaysOn	ON	BTSEL[1:0] = 00 – SyncRst (NAND, SPI) BTSEL[1:0] = 01 – SyncRst (OneNAND) BTSEL[1:0] = 10 – SyncRst (MMC/SD) BTSEL[1:0] = 11 – Enable (UART)
20	UART1	AlwaysOn	ON	SyncRst
21	UART2	AlwaysOn	ON	SyncRst
22	SPI0	AlwaysOn	ON	BTSEL[1:0] = 00 – Enable (NAND, SPI) BTSEL[1:0] = 01 – SyncRst (OneNAND) BTSEL[1:0] = 10 – Enable (MMC/SD) BTSEL[1:0] = 11 – Enable (UART)
23	PWM0	AlwaysOn	ON	SyncRst
24	PWM1	AlwaysOn	ON	SyncRst
25	PWM2	AlwaysOn	ON	SyncRst
26	GPIO	AlwaysOn	ON	SyncRst
27	TIMER0	AlwaysOn	ON	BTSEL[1:0] = 00 – Enable (NAND, SPI) BTSEL[1:0] = 01 – Enable (OneNAND) BTSEL[1:0] = 10 – Enable (MMC/SD) BTSEL[1:0] = 11 – Enable (UART)

Table 7-1. Module Configuration (continued)

			Default States	
28	TIMER1	AlwaysOn	ON	SyncRst
29	TIMER2	AlwaysOn	ON	Enable
30	System Module	AlwaysOn	ON	Enable
31	ARM	AlwaysOn	ON	Enable
32	BUS	AlwaysOn	ON	Enable
33	BUS	AlwaysOn	ON	Enable
34	BUS	AlwaysOn	ON	Enable
35	BUS	AlwaysOn	ON	Enable
36	BUS	AlwaysOn	ON	Enable
37	BUS	AlwaysOn	ON	Enable
38	BUS	AlwaysOn	ON	Enable
39	Reserved	Reserved	Reserved	Reserved
40	MPEG/JPEG Coprocessor (MJCP)	AlwaysOn	ON	SyncRst
41	VPSS DAC	AlwaysOn	ON	SyncRst

7.3 Power Domain and Module States Defined

7.3.1 Power Domain States

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

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

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

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

Table 7-2. Module States

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

The module states are defined as follows:

Module State	Module State Definition
Enable	A module in the enable state has its module reset de-asserted and it has its clock on. This is the normal run-time state for a given module.
Disable	A module in the disable state has its module reset de-asserted and it has its clock off. This state is typically used for disabling a module clock to save static power. The device 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.
SyncRst	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	A module in the SwResetDisable state has its module reset asserted and it has its clock set to off. Generally, software is not expected to initiate this state.

7.4 Executing State Transitions

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

7.4.1 Power Domain State Transitions

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

7.4.2 Module State Transitions

This section describes the procedure for transitioning the module state. The procedure for module state transitions is as follows ('x' corresponds to the module):

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

NOTE: You may set transitions in multiple NEXT bits in MDCTL[x] in this step.

- Set the GOx bit in PTCMD to 0x1 to initiate the transition(s).
- Wait for the GOSTATx bit in PTSTAT to clear to 0x0. The module is only safely in the new state after the GOSTATx bit in PTSTAT clears to 0x0.

7.5 IcePick Emulation Support in the PSC

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

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.

Table 7-3. IcePick Emulation Commands (continued)

Power On and Enable Features	Power On and Enable Descriptions	Reset Features	Reset Descriptions
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 and the NEXT bit in MDCTL register fields, as set by software.

7.6 PSC Interrupts

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

7.6.1 Interrupt Events

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

- 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		Interrupt Condition
Control Register	Status Bit	
PDCTL	EMUIHB	Interrupt occurs when the emulation alters the power domain state
MDCTLx	EMUIHB	Interrupt occurs when the emulation alters the module state
MDCTLx	EMURST	Interrupt occurs when the emulation alters the module's local reset
EPCPR	EPCx	Interrupt occurs during the power domain on/off sequence

7.6.1.1 Power Domain Emulation Events

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

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

7.6.1.2 Module State Emulation Events

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

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the enable state

- When force active is asserted by emulation and module is not already in the enable state

7.6.1.3 Local Reset Emulation Events

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

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

7.6.1.4 External Power Control Pending Event

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

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

7.6.2 Interrupt Registers

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

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

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

The PSC interrupt clear bits are the Mx bit in MERRCRx, the Mx bit in PERRCRx, and the EPx bit in EPCCR.

The PSC interrupt evaluation bit is the ALLEV bit in INTEVAL. When set, this bit forces the PSC interrupt logic to re-evaluate event status. If any events are still active (if any status bits are set) when the ALLEV bit in INTEVAL is set to 0x1, 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 EMUIHB bit in PDCTL, the EMUIHB bit in MDCTL[x], and / or the EMURSTIE bit in MDCTL[x] 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 8 for more information.

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

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

7.7 PSC Registers

Table 7-5 lists the memory-mapped registers for the PSC. See the device memory map Table 4-2 for the memory address of these registers. The default, after reset, PSC configurations are shown in Table 7-1 .

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

Table 7-5. PSC Registers

Offset	Register	Description	Section
0h	PID	Peripheral Revision and Class Information	Section 7.7.1
18h	INTEVAL	Interrupt Evaluation Register	Section 7.7.2
40h	MERRPR0	Module Error Pending Register 0	Section 7.7.3
44h	MERRPR1	Module Error Pending Register 1	Section 7.7.4
50h	MERRCR0	Module Error Clear Register 0	Section 7.7.5
54h	MERRCR1	Module Error Clear Register 1	Section 7.7.6
60h	PERRPR	Power Error Pending Register	Section 7.7.7
68h	PERRCR	Power Error Clear Register	Section 7.7.8
70h	EPCPR	External Power Error Pending Register	Section 7.7.9
78h	EPCCR	External Power Control Clear Register	Section 7.7.10
120h	PTCMD	Power Domain Transition Command Register	Section 7.7.11
128h	PTSTAT	Power Domain Transition Status Register	Section 7.7.12
200h	PDSTAT	Power Domain Status Register	Section 7.7.13
300h	PDCTL	Power Domain Control Register	Section 7.7.14
800h	MDSTAT[52]	Module Status Registers	Section 7.7.15
A00h	MDCTL[52]	Module Control Registers	Section 7.7.16

NOTE: After reset default PSC configurations are shown in Table 7-1 .

7.7.1 Peripheral Revision and Class Information (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)

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R- 208	

15	11	10	8	7	6	5	0
RTL		MAJOR		CUSTOM		MINOR	
R-0		R-1		R-0		R-5	

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31-30	SCHEME	0-3h	Scheme
29-28	Reserved	0	Reserved
27-16	FUNC	0-FFFh	Software compatible
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)

31	1	0
Reserved		ALLEV
R-0		W-0

LEGEND: R = Read, W = Write, n = value at reset

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

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ALLEV		Re-evaluate PSC interrupt.
		0	A write of 0 has no effect.
		1	Write 1 to re-evaluate the interrupt condition.

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

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

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

31	0
M0[32]	
R-0	

LEGEND: R = Read, n = value at reset

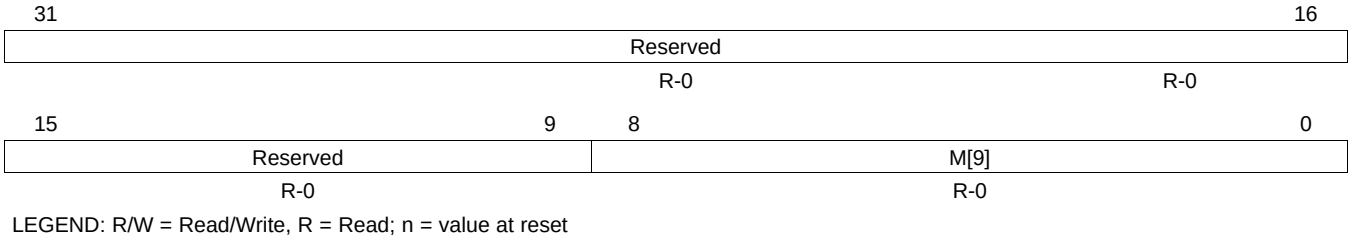
Table 7-8. Module Error Pending Register 0 (mod 0 - 31) (MERRPR0) Field Descriptions

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

7.7.4 Module Error Pending Register 1 (mod 32-41) (MERRPR1)

The module error pending register 1 (mod 32 - 41) (MERRPR1) is shown in Figure 7-6 and described in Table 7-9.

Figure 7-6. Module Error Pending Register 1 (mod 32-41) (MERRPR1)



LEGEND: R/W = Read/Write, R = Read; n = value at reset

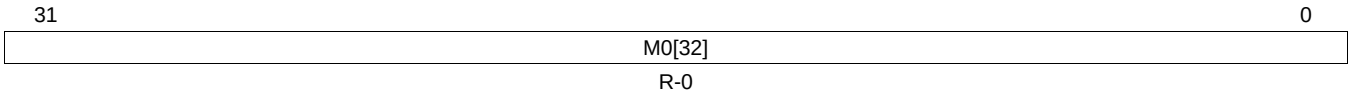
Table 7-9. Module Error Pending Register 1 (mod 32-41) (MERRPR1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8-0	M[9]		Module interrupt status bit for modules 32-41.
		0	Power domain interrupt is not active.
		1	Power domain interrupt is active.

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

The module error clear 0 (mod 0-31) register (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: R = Read, n = value at reset

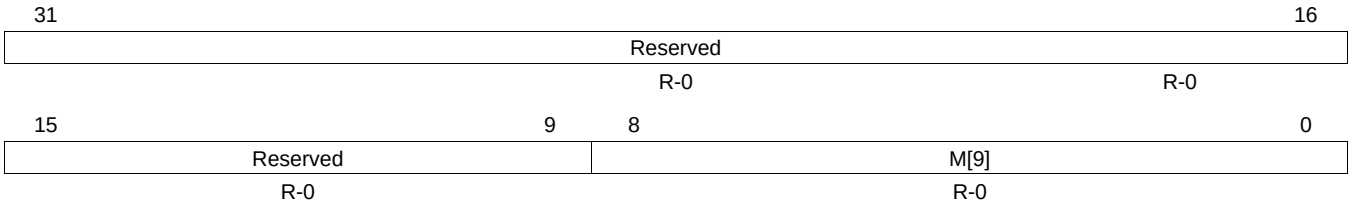
Table 7-10. Module Error Clear Register 0 (mod 0-31) (MERRCR0) Field Descriptions

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

7.7.6 Module Error Clear Register 1 (mod 32-41) (MERRCR1)

The module error clear 1 (mod 32-41) register (MERRCR1) is shown in Figure 7-8 and described in Table 7-11.

Figure 7-8. Module Error Clear Register 1 (mod 32-41) (MERRCR1)



LEGEND: R/W = Read/Write, R = Read; n = value at reset

Table 7-11. Module Error Clear Register 1 (mod 32-41) (MERRCR1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8-0	M[9]		Clears the interrupt bit set in the corresponding MERRPR1 register bit field and the MDSTAT interrupt bit fields. This pertains to modules 32-41.
		0	A write of 0 has no effect.
		1	Clears module interrupt.

7.7.7 Power Error Pending Register (PERRPR)

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

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

31	1	0
Reserved		P[1]
R-0		R-0

LEGEND: R = Read, n = value at reset

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

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

7.7.8 Power Error Clear Register (PERRCR)

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

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

31	1	0
Reserved		P[1]
R-0		W-0

LEGEND: R = Read, W = Write, n = value at reset

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

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

7.7.9 External Power Control Pending Register (EPCPR)

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

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

31	1	0
Reserved		EPC[1]
R-0		R-0

LEGEND: R = Read, n = value at reset

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

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

7.7.10 External Power Control Clear Register (EPCCR)

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

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

31	1	0
Reserved		EPC[1]
R-0		W-0

LEGEND: R = Read, W = Write, n = value at reset

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

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EPC[1]		External power control clear bit.
		0	A write of 0 has no effect.
		1	Set this bit to clear the EPCPR interrupt.

7.7.11 Power Domain Transition Command Register (PTCMD)

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

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

31	1	0
Reserved		GO[1]
R-0		W-0

LEGEND: R = Read, W = Write, n = value at reset

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

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GO[1]		Power domain GO transition command
		0	A write of 0 has no effect.
		1	Writing 1 causes the state transition interrupt generation block to evaluate the new PTNEXT and the NEXT states in MDCTL as the desired states of the application.

7.7.12 Power Domain Transition Status Register (PTSTAT)

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

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

31	1	0
Reserved		GOSTAT[1]
R-0		R-0

LEGEND: R = Read, n = value at reset

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

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSTAT[1]		Power domain transition status
		0	No transition in progress
		1	Power domain is transitioning (i.e., either the power domain is transitioning or modules in this power domain are transitioning).

7.7.13 Power Domain Status Register (PDSTAT)

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

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

31	16
Reserved	
R-0	

15	12	11	10	9	8	7	1	0
Reserved		EMUIHB	Reserved	PORDONE	POR	Reserved		STATE
R-0		R-0	R-0	R-0	R-0	R-0		R-0

LEGEND: R = Read; n = value at reset

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

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

7.7.14 Power Domain Control Register (PDCTL)

The power domain control register (PDCTL) is shown in Figure 7-16 and described in Table 7-19. In the device, only one PDCTL register is applicable because there is only one power domain, the AlwaysOn power domain. The AlwaysOn power domain is always on when voltage is applied to the device; it is not possible to use PDCTL turn off power to the AlwaysOn power domain.

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

31					16
Reserved					
R-0					
15	**10**	**9**	**8**	**7** **1**	**0**
Reserved		EMUIHBIE	EPCGOOD	Reserved	NEXT
R-0		R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

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

7.7.15 Module Status n Register 0-32 (MDSTATn)

The module status 0 register (MDSTATn) is shown in Figure 7-17 and described in Table 7-20. See Table 7-1 for after reset default module states.

Figure 7-17. Module Status n Register (MDSTATn)

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

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

LEGEND: R = Read; n = value at reset

Table 7-20. Module Status n Register 0-32 (MDSTATn) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	EMUIHB		Emulation alters modules state interrupt active
		0	Interrupt not active
		1	Interrupt active
16	EMURST		Emulation alters module reset interrupt active
		0	Interrupt not active
		1	Interrupt active
15-13	Reserved	0	Reserved
12	MCKOUT		Module clock output status. Shows status of module clock ON / OFF.
		0	Module clock is off.
		1	Module clock is on.
11	MRSTDONE		Module reset done. Software is responsible for checking that mode reset is done before accessing the module.
		0	Module reset is not done.
		1	Module reset is done.
10	MRST		Module reset status. Reflects actual state of module reset.
		0	Module reset is asserted.
		1	Module reset is de-asserted.
9-6	Reserved	0	Reserved
5-0	STATE		Module state status: indicates current module status.
		0	SwRstDisable state
		1	SyncReset state
		2	Disable state
		3	Enable state. Others: indicates transition.
		Others	Indicates a transition

7.7.16 Module Control n Register 0-51 (MDCTLn)

The module control n register 0-41 (MDCTLn) is shown in Figure 7-18 and described in Table 7-21. See Table 7-1 for after reset default module states. It is not possible to change the module states for modules 29 through 38. The states of these modules are taken care of by internal hardware and are primarily for internal chip infrastructure. Module 39 is reserved and you must not try to change the state of module 39. It is possible to change the state of other modules: 0-27, 40, and 41.

Figure 7-18. Module Control n Register 0-41 (MDCTLn)

31	16
Reserved	
R-0	

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

Table 7-21. Module Control n Register 0-51 (MDCTLn) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	EMUIHBIE		Interrupt enable for emulation alters module state.
		0	Disable interrupt.
		1	Enable interrupt.
9	EMURSTIE		Interrupt enable for emulation alters reset.
		0	Disable interrupt.
		1	Enable interrupt.
8-5	Reserved	0	Reserved
4-0	NEXT		Module next state
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

Interrupt Controller

8.1 Introduction

The device 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-S 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 INTC supports user-configurable interrupt-priority and interrupt entry addresses. Entry addresses minimize the time spent jumping to interrupt service routines (ISRs). When an interrupt occurs, the corresponding highest priority ISR's address is stored in the INTC's ENTRY register. The IRQ or FIQ interrupt routine can read the ENTRY register and jump to the corresponding ISR directly. Thus, the ARM does not require a software dispatcher to determine the asserted interrupt.

8.2 Interrupt Mapping

The AINTC takes up to 64 ARM device interrupts and maps them to either the IRQ or to the FIQ of the ARM. Each interrupt is also assigned one of 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 8-1 shows the connection of device interrupts to the ARM.

NOTE: The total number of interrupts in the device exceeds 64, which is the maximum value of the AINTC module. Therefore, several interrupts are multiplexed and you must use the register ARM_INTMUX in the system module to select the interrupt source for multiplexed interrupts. Refer to the Chapter 9 for more information on the system module register ARM_INTMUX.

Table 8-1. AINTC Interrupt Connections

Interrupt Number	Acronym	Source	Interrupt Number	Acronym	Source
0	VPSSINT0	VPSS - INT0, Configurable via VPSSBL register: INTSEL	32	TINT0	Timer 0 - TINT12
1	VPSSINT1	VPSS - INT1	33	TINT1	Timer 0 - TINT34
2	VPSSINT2	VPSS - INT2	34	TINT2	Timer 1 - TINT12
3	VPSSINT3	VPSS - INT3	35	TINT3	Timer 1 - TINT34
4	VPSSINT4	VPSS - INT4	36	PWMINT0	PWM0
5	VPSSINT5	VPSS - INT5	37	PWMINT1	PWM1

Table 8-1. AINTC Interrupt Connections (continued)

Interrupt Number	Acronym	Source	Interrupt Number	Acronym	Source
6	VPSSINT6	VPSS - INT6	38	PWMINT2	PWM2
7	VPSSINT7	VPSS - INT7	39	I2CINT	I2C
8	VPSSINT8	VPSS - INT8	40	UARTINT0	UART0
9	Reserved for use with the MJCP		41	UARTINT1	UART1
10	Reserved for use with the MJCP		42	SPINT0-0	SPI0
11	Reserved for use with the MJCP		43	SPINT0-1	SPI0
12	USBINT	USB OTG Collector	44	GPIO0	GPIO
13	RTOINT or TINT4	RTO or Timer 2 - TINT12 SYS.ARM_INTMUX	45	GPIO1	GPIO
14	UARTINT2 or TINT5	UART2 or Timer 2 - TINT34	46	GPIO2	GPIO
15	TINT6	Timer 3 TINT12	47	GPIO3	GPIO
16	CCINT0	EDMA CC Region 0	48	GPIO4	GPIO
17	SPINT1-0 or CCERRINT	SPI1 or EDMA CC Error	49	GPIO5	GPIO
18	SPINT1-1 or TCERRINT0	SPI1 or EDMA TC0 Error	50	GPIO6	GPIO
19	SPINT2-0 or TCERRINT1	SPI2 or EDMA TC1 Error	51	GPIO7	GPIO
20	PSCINT	PSC - ALLINT	52	GPIO8	GPIO
21	SPINT2-1	SPI2	53	GPIO9	GPIO
22	TINT7	Timer3 - TINT34	54	GPIOBNK0	GPIO
23	SDIOINT0	SDIO(MMC/SD0)	55	GPIOBNK1	GPIO
24	MBXINT0 or MBXINT1	ASP0 or ASP1	56	GPIOBNK2	GPIO
25	MBRINT0 or MBRINT1	ASP0 or ASP1	57	GPIOBNK3	GPIO
26	MMCINT0	MMC/SD0	58	GPIOBNK4	GPIO
27	MMCINT1	MMC/SD1	59	GPIOBNK5	GPIO
28	PWMINT3	PWM3	60	GPIOBNK6	GPIO
29	DDRINT	DDR EMIF	61	COMMTX	ARMSS
30	AEMIFINT	Async EMIF	62	COMMRX	ARMSS
31	SDIOINT1	SDIO(MMC/SD1)	63	EMUINT	E2ICE

8.3 INTC Methodology

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

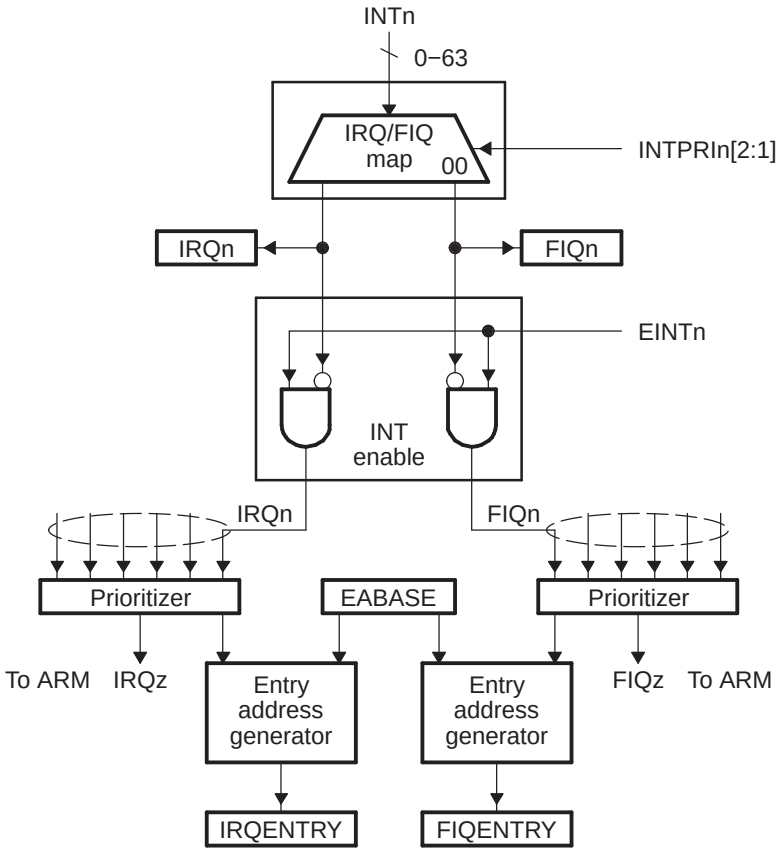
- When an interrupt occurs, the status is reflected in either the FIQn or the IRQn registers, depending upon the interrupt type selected.
- Interrupts are enabled or disabled (masked) by setting the EINTn register.

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

- When an interrupt from any interrupt channel occurs (for which interrupt is enabled), an IRQ or FIQ interrupt generates to the 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 8-1. AINTC Functional Diagram



8.3.1 Interrupt Mapping

Each event input is mapped to either the ARM IRQ or to the FIQ interrupt based on the priority level selected in the INTPRIn register. Events with a priority of 0x0 or 0x1 are designated as FIQs. Those with priorities of 0x2-0x7 are designated as IRQs. The appropriate IRQ / FIQ registers capture interrupt events. Each event causes an IRQ or FIQ to generate only if the corresponding EINT bit enables it. The EINT bit enables or disables the event regardless of whether it is mapped to IRQ or to FIQ. The IRQ / FIQ register always captures each event, regardless of whether the interrupt is actually enabled.

8.3.2 Interrupt Prioritization

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

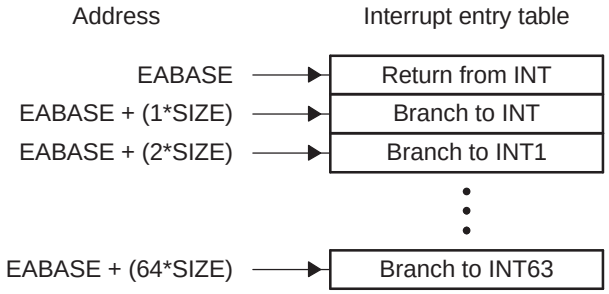
8.3.3 Vector Table Entry Address Generation

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

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

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

Figure 8-2. Interrupt Entry Table



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

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

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

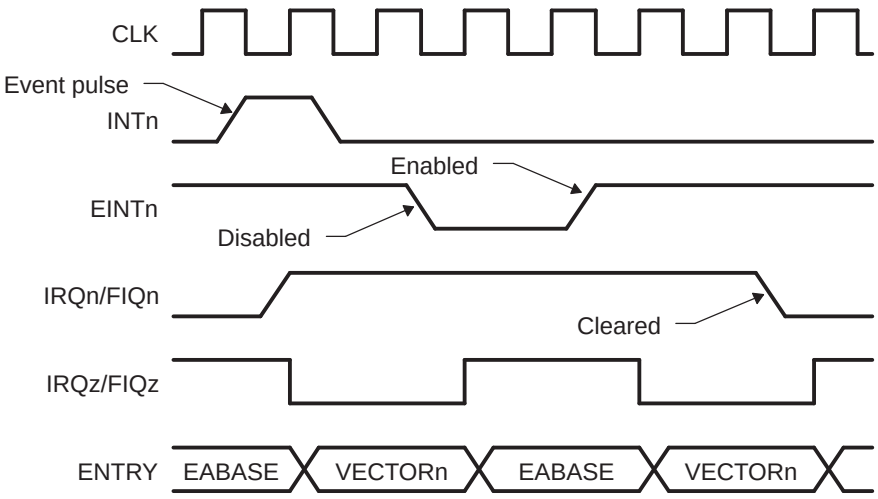
8.3.4 Clearing Interrupts

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

8.3.5 Enabling and Disabling Interrupts

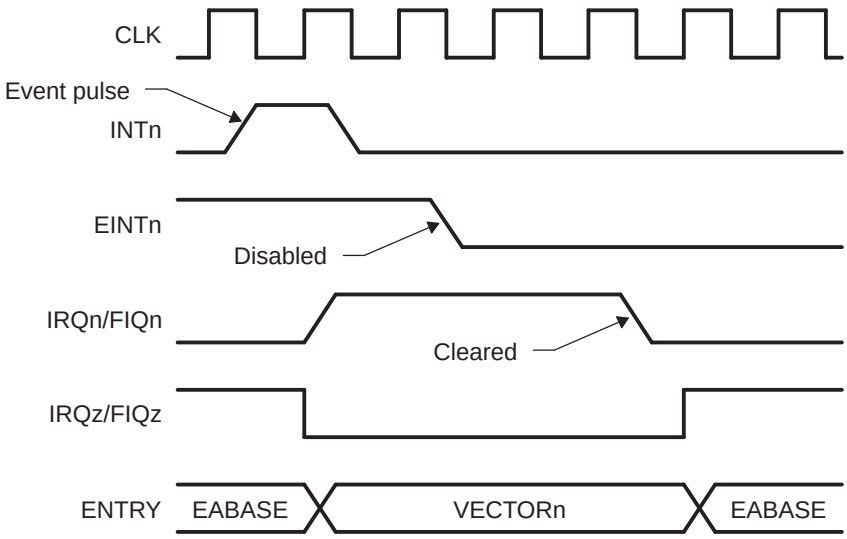
The AINTC has two methods for enabling and disabling interrupts: immediate or delayed, based on the setting of the IDMODE bit in the INTCTL register. When 0 (default), clearing an interrupt's EINT bit has an immediate effect. The prioritizer removes the disabled interrupt from consideration and adjusts the IRQ/FIQENTRY value correspondingly. If no other interrupts are pending, then the IRQz/FIQz output to the ARM may also go inactive. Enabling the interrupt if it is already pending takes immediate affect. This is shown in Figure 8-3.

Figure 8-3. Immediate Interrupt Disable / Enable



If IDMODE is 1, then the EINT effect is delayed. Essentially, the active interrupt status is latched until cleared by the ARM. If EINT is cleared, the prioritizer continues to use the interrupt and the IRQz/FIQz remains active. Once the ARM clears the pending interrupt, further interrupts are disabled. In the same way, setting EINT does not cause the previously pending interrupt event to become enabled until it has been cleared first. The disable operation is shown in Figure 8-4.

Figure 8-4. Delayed Interrupt Disable



8.4 INTC Registers

Table 8-2 lists the memory-mapped registers for the INTC. See the device memory mapTable 4-2 for the memory address of these registers.

Table 8-2. Interrupt Controller (INTC) Registers

Offset	Acronym	Register Description	Section
00h	FIQ0	Interrupt Status of INT [31:0] (if mapped to FIQ)	Section 8.4.1
04h	FIQ1	Interrupt Status of INT [63:32] (if mapped to FIQ)	Section 8.4.2
08h	IRQ0	Interrupt Status of INT [31:0] (if mapped to IRQ)	Section 8.4.3
0Ch	IRQ1	Interrupt Status of INT [63:32] (if mapped to IRQ)	Section 8.4.4
10h	FIQENTRY	Entry Address [28:0] for valid FIQ interrupt	Section 8.4.5
14h	IRQENTRY	Entry Address [28:0] for valid IRQ interrupt	Section 8.4.6
18h	EINT0	Interrupt Enable Register 0	Section 8.4.7
1Ch	EINT1	Interrupt Enable Register 1	Section 8.4.8
20h	INTCTL	Interrupt Operation Control Register	Section 8.4.9
24h	EABASE	Interrupt Entry Table Base Address	Section 8.4.10
30h	INTPRI0	Interrupt 0-7 Priority select	Section 8.4.11
34h	INTPRI1	Interrupt 8-15 Priority select	Section 8.4.12
38h	INTPRI2	Interrupt 16-23 Priority select	Section 8.4.13
3Ch	INTPRI3	Interrupt 24-31 Priority select	Section 8.4.14
40h	INTPRI4	Interrupt 32-29 Priority select	Section 8.4.15
44h	INTPRI5	Interrupt 40-47 Priority select	Section 8.4.16
48h	INTPRI6	Interrupt 48-55 Priority select	Section 8.4.17
4Ch	INTPRI7	Interrupt 56-63 Priority select	Section 8.4.18

8.4.1 Fast Interrupt Request Status Register 0 (FIQ0)

The fast interrupt request status register 0 (FIQ0) is shown in Figure 8-5 and described in Table 8-3.

Figure 8-5. Interrupt Status of INT[31:0] (if mapped to FIQ)

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

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

Table 8-3. Interrupt Status of INT[31:0] (if mapped to FIQ) Field Descriptions

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

8.4.2 Fast Interrupt Request Status Register 1 (FIQ1)

The fast interrupt request status register 1 (FIQ1) is shown in Figure 8-6 and described in Table 8-4.

Figure 8-6. Interrupt Status of INT[63:32] (if mapped to FIQ)

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

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

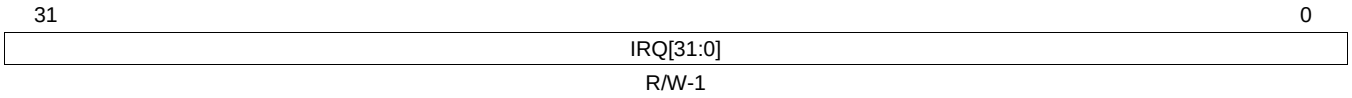
Table 8-4. Interrupt Status of INT[63:32] (if mapped to FIQ) Field Descriptions

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

8.4.3 Interrupt Request Status Register 0 (IRQ0)

The interrupt request status register 0 (IRQ0) is shown in Figure 8-7 and described in Table 8-5.

Figure 8-7. Interrupt Status of INT[31:0] (if mapped to IRQ)



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

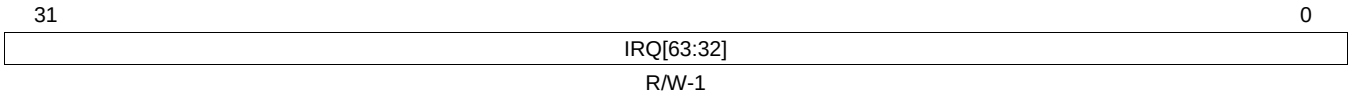
Table 8-5. Interrupt Status of INT[31:0] (if mapped to IRQ) Field Descriptions

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

8.4.4 Interrupt Request Status Register 1 (IRQ1)

The interrupt request status register 1 (IRQ1) is shown in Figure 8-8 and described in Table 8-6.

Figure 8-8. Interrupt Status of INT[31:0] (if mapped to IRQ)



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

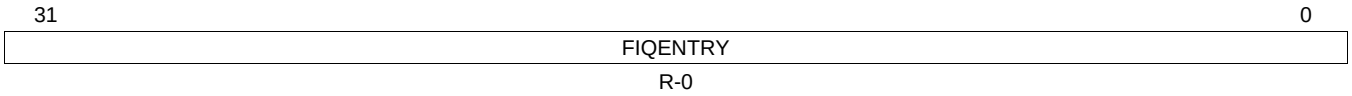
Table 8-6. Interrupt Status of INT[31:0] (if mapped to IRQ) Field Descriptions

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

8.4.5 Fast Interrupt Request Entry Address Register (FIQENTRY)

The fast interrupt request entry address register (FIQENTRY) is shown in Figure 8-9 and described in Table 8-7.

Figure 8-9. Fast Interrupt Request Entry Address Register (FIQENTRY)



LEGEND: R = Read only; n = value at reset

Table 8-7. Fast Interrupt Request Entry Address Register (FIQENTRY) Field Descriptions

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

8.4.6 Interrupt Request Entry Address Register (IRQENTRY)

The interrupt request entry address register (IRQENTRY) is shown in Figure 8-10 and described in Table 8-8.

Figure 8-10. Interrupt Request Entry Address Register (IRQENTRY)

31	0
IRQENTRY	
R-0	

LEGEND: R = Read only; n = value at reset

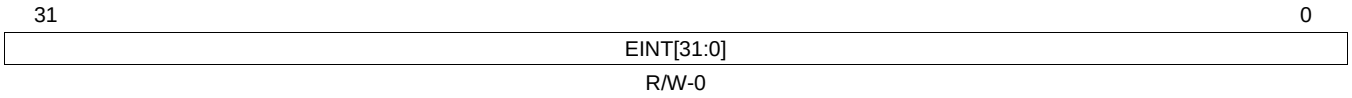
Table 8-8. Interrupt Request Entry Address Register (IRQENTRY) Field Descriptions

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

8.4.7 Interrupt Enable Register 0 (EINT0)

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

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



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

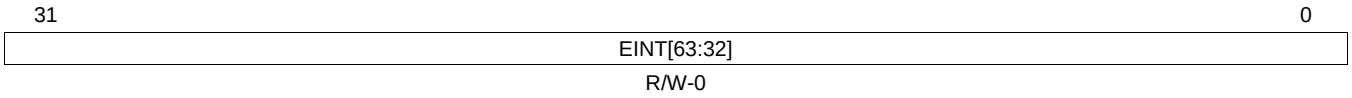
Table 8-9. Interrupt Enable Register 0 (EINT0) Field Descriptions

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

8.4.8 Interrupt Enable Register 1 (EINT1)

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

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



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

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

Bit	Field	Value	Description
31-0	EINT[63:32]		Interrupt enable for INTx.
		0	Mask interrupt
		1	Enable interrupt

8.4.9 Interrupt Operation Control Register (INTCTL)

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

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

31	3	2	1	0
Reserved		IDMODE	IERAW	FERAW
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 8-11. Interrupt Operation Control Register (INTCTL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IDMODE		Interrupt disable mode.
		0	Disable immediately.
		1	Disable after ack
1	IERAW		Masked interrupt reflected in the IRQENTRY register.
		0	Disable reflect.
		1	Enable reflect.
0	FERAW		Masked interrupt reflect in FIQENTRY register.
		0	Disable reflect.
		1	Enable reflect.

8.4.10 EABASE

The EABASE register is shown in Figure 8-14 and described in Table 8-12.

Figure 8-14. EABASE

31 29	28 3	2	1 0
Reserved	EABASE	Reserved	SIZE
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 8-12. 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		Size of each entry in the interrupt entry table.
		0	4 bytes
		1h	8 bytes
		2h	16 bytes
		3h	32 bytes

8.4.11 Interrupt Priority Register 0 (INTPRI0)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT7	0-7h	Selects INT7 priority level.
27	Reserved	0	Reserved
26-24	INT6	0-7h	Selects INT6 priority level.
23	Reserved	0	Reserved
22-20	INT5	0-7h	Selects INT5 priority level.
19	Reserved	0	Reserved
18-16	INT4	0-7h	Selects INT4 priority level.
15	Reserved	0	Reserved
14-12	INT3	0-7h	Selects INT3 priority level.
11	Reserved	0	Reserved
10-8	INT2	0-7h	Selects INT2 priority level.
7	Reserved	0	Reserved
6-4	INT1	0-7h	Selects INT1 priority level.
3	Reserved	0	Reserved
2-0	INT0	0-7h	Selects INT0 priority level.

8.4.12 Interrupt Priority Register 1 (INTPRI1)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT15	0-7h	Selects INT15 priority level.
27	Reserved	0	Reserved
26-24	INT14	0-7h	Selects INT14 priority level.
23	Reserved	0	Reserved
22-20	INT13	0-7h	Selects INT13 priority level.
19	Reserved	0	Reserved
18-16	INT12	0-7h	Selects INT12 priority level.
15	Reserved	0	Reserved
14-12	INT11	0-7h	Selects INT11 priority level.
11	Reserved	0	Reserved
10-8	INT10	0-7h	Selects INT10 priority level.
7	Reserved	0	Reserved
6-4	INT9	0-7h	Selects INT9 priority level.
3	Reserved	0	Reserved
2-0	INT8	0-7h	Selects INT8 priority level.

8.4.13 Interrupt Priority Register 2 (INTPRI2)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT23	0-7h	Selects INT23 priority level.
27	Reserved	0	Reserved
26-24	INT22	0-7h	Selects INT22 priority level.
23	Reserved	0	Reserved
22-20	INT21	0-7h	Selects INT21 priority level.
19	Reserved	0	Reserved
18-16	INT20	0-7h	Selects INT20 priority level.
15	Reserved	0	Reserved
14-12	INT19	0-7h	Selects INT19 priority level.
11	Reserved	0	Reserved
10-8	INT18	0-7h	Selects INT18 priority level.
7	Reserved	0	Reserved
6-4	INT17	0-7h	Selects INT17 priority level.
3	Reserved	0	Reserved
2-0	INT16	0-7h	Selects INT16 priority level.

8.4.14 Interrupt Priority Register 3 (INTPRI3)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT31	0-7h	Selects INT31 priority level.
27	Reserved	0	Reserved
26-24	INT30	0-7h	Selects INT30 priority level.
23	Reserved	0	Reserved
22-20	INT29	0-7h	Selects INT29 priority level.
19	Reserved	0	Reserved
18-16	INT28	0-7h	Selects INT28 priority level.
15	Reserved	0	Reserved
14-12	INT27	0-7h	Selects INT27 priority level.
11	Reserved	0	Reserved
10-8	INT26	0-7h	Selects INT26 priority level.
7	Reserved	0	Reserved
6-4	INT25	0-7h	Selects INT25 priority level.
3	Reserved	0	Reserved
2-0	INT24	0-7h	Selects INT24 priority level.

8.4.15 Interrupt Priority Register 4 (INTPRI4)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT39	0-7h	Selects INT39 priority level.
27	Reserved	0	Reserved
26-24	INT38	0-7h	Selects INT38 priority level.
23	Reserved	0	Reserved
22-20	INT37	0-7h	Selects INT37 priority level.
19	Reserved	0	Reserved
18-16	INT36	0-7h	Selects INT36 priority level.
15	Reserved	0	Reserved
14-12	INT35	0-7h	Selects INT35 priority level.
11	Reserved	0	Reserved
10-8	INT34	0-7h	Selects INT34 priority level.
7	Reserved	0	Reserved
6-4	INT33	0-7h	Selects INT33 priority level.
3	Reserved	0	Reserved
2-0	INT32	0-7h	Selects INT32 priority level.

8.4.16 Interrupt Priority Register 5 (INTPRI5)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT47	0-7h	Selects INT47 priority level.
27	Reserved	0	Reserved
26-24	INT46	0-7h	Selects INT46 priority level.
23	Reserved	0	Reserved
22-20	INT45	0-7h	Selects INT45 priority level.
19	Reserved	0	Reserved
18-16	INT44	0-7h	Selects INT44 priority level.
15	Reserved	0	Reserved
14-12	INT43	0-7h	Selects INT43 priority level.
11	Reserved	0	Reserved
10-8	INT42	0-7h	Selects INT42 priority level.
7	Reserved	0	Reserved
6-4	INT41	0-7h	Selects INT41 priority level.
3	Reserved	0	Reserved
2-0	INT40	0-7h	Selects INT40 priority level.

8.4.17 Interrupt Priority Register 6 (INTPRI6)

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

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

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

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

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT55	0-7h	Selects INT55 priority level.
27	Reserved	0	Reserved
26-24	INT54	0-7h	Selects INT54 priority level.
23	Reserved	0	Reserved
22-20	INT53	0-7h	Selects INT53 priority level.
19	Reserved	0	Reserved
18-16	INT52	0-7h	Selects INT52 priority level.
15	Reserved	0	Reserved
14-12	INT51	0-7h	Selects INT51 priority level.
11	Reserved	0	Reserved
10-8	INT50	0-7h	Selects INT50 priority level.
7	Reserved	0	Reserved
6-4	INT49	0-7h	Selects INT49 priority level.
3	Reserved	0	Reserved
2-0	INT48	0-7h	Selects INT48 priority level.

8.4.18 Interrupt Priority Register 7 (INTPRI7)

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

Figure 8-22. Interrupt Priority Register 7 (INTPRI7)

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

LEGEND: R/W = Read/Write, R = Read; n = value at reset

Table 8-20. Interrupt Priority Register 7 (INTPRI7) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	INT63	0-7h	Selects INT63 priority level.
27	Reserved	0	Reserved
26-24	INT62	0-7h	Selects INT62 priority level.
23	Reserved	0	Reserved
22-20	INT61	0-7h	Selects INT61 priority level.
19	Reserved	0	Reserved
18-16	INT60	0-7h	Selects INT60 priority level.
15	Reserved	0	Reserved
14-12	INT59	0-7h	Selects INT59 priority level.
11	Reserved	0	Reserved
10-8	INT58	0-7h	Selects INT58 priority level.
7	Reserved	0	Reserved
6-4	INT57	0-7h	Selects INT57 priority level.
3	Reserved	0	Reserved
2-0	INT56	0-7h	Selects INT56 priority level.

System Control Module

9.1 Overview of the System Control Module

The device's 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 miscellaneous set of status and control registers, accessible the ARM and supporting all of the following system features and operations:

- Device Identification
- Device Configuration
  - Pin multiplexing control
  - Device boot configuration status
- ARM Interrupt and EDMA Event multiplexing control
- Special Peripheral Status and Control
  - Timer64+ control
  - USB PHY control
  - VPSS clock and Video DAC control and status
  - DDR I/O timing control and status
  - DDR VTP control
  - Clock out circuitry
  - GIO de-bounce control
- Power Management
  - Deep Sleep Control
- Bandwidth Management
  - Bus master DMA priority control

This chapter describes the system control module.

9.2 Device Identification

The DEVICE_ID register of the System Control Module contains a software readable version of the JTAG ID device. Software can use this register to determine the version of the device on which it is executing. The register format and description are shown in Table 9-11 and Table 9-14, respectively.

9.3 Device Configuration

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

9.3.1 Pin Multiplexing Control

The device makes extensive use of pin multiplexing to accommodate the large number of peripheral functions in the smallest possible package. A combination of hardware configuration (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.

9.3.1.1 Hardware Controlled Default Pin Multiplexing

There are configuration input signals that can set some of the default pin mux and hardware configurations that may be needed for device boot. Use pins AECFG[3:0] to configure the pins of the AEMIF.

9.3.1.2 Program Controlled Pin Multiplexing

All pin multiplexing options other than those mentioned above are controlled by software via five pin mux registers. The format of the registers and a description of the pins they control are in the following sections.

9.3.1.2.1 PinMux0 Register

The PINMUX0 register controls pin multiplexing for the VPFE pins. The register format is shown in Table 9-4. A brief description of each field is shown in Table 9-4.

9.3.1.2.2 PinMux1 Register

The PINMUX1 register controls pin multiplexing for the VPBE pins. The register format is shown in Figure 9-2 with descriptions in Table 9-5.

9.3.1.2.3 PinMux2 Register

The PINMUX2 register controls pin multiplexing for the AEMIF pins. The register format is shown in Figure 9-3. A brief description of each field is shown in Table 9-6.

9.3.1.2.4 PinMux3 Register

The PINMUX3 register controls pin multiplexing for the GIO pins. The register format is shown in Figure 9-4. A brief description of each field is shown in Table 9-7.

9.3.1.2.5 PinMux4 Register

The PINMUX4 register controls pin multiplexing for SPI0 and MMC/SD0. The register format is shown in Figure 9-5. A brief description of each field is shown in Table 9-8.

9.3.2 Device Boot Configuration Status

The device boot configuration (the state of the BTSEL[1:0] and AECFG[3:0] signals are captured in the BOOTCFG register), as shown in Figure 9-6and Table 9-9.

9.4 ARM Interrupt and EDMA Event Multiplexing Control

The ARM_INTMUX and EDMA_EVTMUX registers are read/write registers containing the multiplexing control for interrupts and events to the ARM and EDMA, respectively. These registers are necessary because the total number of interrupts and events in the chip exceeds the maximum value of 64 which the AINTC and EDMA support. See Figure 9-7and Figure 9-8.

9.5 Special Peripheral Status and Control

Several of the device's peripheral modules require additional system-level control logic. Those registers are discussed in detail in Section 9.10.

9.5.1 Timer64+ Control

The TIMER64_CTL register controls the GIO input selection. See Figure 9-13 and Table 9-16.

9.5.2 USB PHY Control

The USB_PHY_CTL register controls various features of the USB PHY, as shown in Figure 9-14 and Table 9-17.

9.5.3 VPSS Clock and DAC Control and Status

Clocks for the video processing subsystem are controlled via the VPSS_CLK_CTRL register. Video DAC configuration is controlled by VDAC.

9.5.4 DDR I/O Timing Control and Status

The DDR_SLEW register reflects the DDR I/O timing, as programmed in the eFuse device. See Figure 9-9.

9.6 Clock Out Configuration Status

The device supports three clock out pins (CLKOUT[3:1]). The purpose of these pins is to provide input clock to external components which are CCD clock to the AFE/TG, audio clock, and clock for motor control. The CCD clock is the input crystal clock fed undivided, directly to the pin (CLKOUT1), the audio clock is a divide by 3 clock (CLKOUT2), and the motor control is a divide by 8 clock (CLKOUT3). The register CLKOUT is the CLK_OUT[3:1] divisor and output control register. In the device, this register is read only. See Figure 9-10.

9.7 GIO De-Bounce Control

The DEBOUNCE registers control whether GIO0-GIO7 pin inputs are de-bounced or not. The de-bounce logic cancels the chattering caused by mechanical switch or slow slope input. See Figure 9-20.

9.8 Power Management

9.8.1 Deep Sleep Control

Register DEEPSLEEP contains bits for the Deep Sleep power mode. See Figure 9-19.

9.9 Bandwidth Management

9.9.1 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 device master is shown in Table 9-1.

Table 9-1. Master IDs

MSTID	Master
0	ARM Instruction
1	ARM Data
2	Reserved
3	Reserved
4-7	Reserved
8	VPSS
9	MPEG/JPEG Coprocessor (MJCP)
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	Reserved
33	Reserved
34	USB
35	Reserved
36	Reserved
37	Reserved
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 device, 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 device bus master is shown in Table 9-2. Application software is expected to modify these values to obtain the desired system performance.

Table 9-2. Default Master Priorities

Master	Default Priority
VPSS	0[1]
EDMA Ch 0	0[2]
EDMA Ch 1	0[2]
ARM (DMA)	1
ARM (CFG)	1
Reserved	-
Reserved	-
Reserved	-
Reserved	-
USB	4
Reserved	-
Reserved	-
Reserved	-
MPEG/JPEG Coprocessor (MJCP)	-

(1) Default value in VPSS PCR register
(2) Default value for EDMA CC is in the QUEPRI register of the EDMA module.

9.10 System Control Registers

Table 9-3 lists the memory-mapped registers for the System Module (SYS). See the device memory map for Table 4-2 the memory address of these registers.

Table 9-3. System Module (SYS) Registers

Offset	Acronym	Register Description	Section
0h	PINMUX0	PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register	Section 9.10.1
4h	PINMUX1	PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register	Section 9.10.2
8h	PINMUX2	PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register	Section 9.10.3
Ch	PINMUX3	PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register	Section 9.10.4
10h	PINMUX4	PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register	Section 9.10.5
14h	BOOTCFG	Boot Configuration	Section 9.10.6
18h	ARM_INTMUX	Multiplexing Control for Interrupts	Section 9.10.6
1Ch	EDMA_EVTMUX	Multiplexing Control for EDMA Events	Section 9.10.8
20h	DDR_SLEW	DDR Slew Rate	Section 9.10.9
24h	CLKOUT	CLKOUT div/out Control	Section 9.10.10
28h	DEVICE_ID	Device ID	Section 9.10.11
2Ch	VDAC_CONFIG	Video DAC Configuration	Section 9.10.12
30h	TIMER64_CTL	TIMER64_CTL - Timer64+ Input Control	Section 9.10.13
34h	USB_PHY_CTRL	USB PHY Control	Section 9.10.14
38h	MISC	Miscellaneous Control	Section 9.10.15
3Ch	MSTPRI0	Master Priorities Reg0	Section 9.10.16
40h	MSTPRI1	Master Priorities Reg1	Section 9.10.17
44h	VPSS_CLK_CTRL	VPSS Clock Mux Control	Section 9.10.18
48h	DEEPSLEEP	DEEPSLEEP Configuration	Section 9.10.19
50h	DEBOUNCE0	DEBOUNCE - Debounce for GIO0 Input	Section 9.10.20
54h	DEBOUNCE1	DEBOUNCE - Debounce for GIO1 Input	Section 9.10.20
58h	DEBOUNCE2	DEBOUNCE - Debounce for GIO2 Input	Section 9.10.20
5Ch	DEBOUNCE3	DEBOUNCE - Debounce for GIO3 Input	Section 9.10.20
60h	DEBOUNCE4	DEBOUNCE - Debounce for GIO4 Input	Section 9.10.20
64h	DEBOUNCE5	DEBOUNCE - Debounce for GIO5 Input	Section 9.10.20
68h	DEBOUNCE6	DEBOUNCE - Debounce for GIO6 Input	Section 9.10.20
6Ch	DEBOUNCE7	DEBOUNCE - Debounce for GIO7 Input	Section 9.10.20
70h	VTPIOCR	VTP IO Control Register	Section 9.10.21

9.10.1 PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register

The PINMUX0 register controls pin multiplexing for the VPFE pins.

Figure 9-1. PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register

31	24
Reserved	
R-0	

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
Reserved	PCLK	CAM_WEN	CAM_VD	CAM_HD	YIN_70	CIN_10	CIN_32
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CIN_4		YCIN_5		CIN_6		CIN_7	
R/W-0		R/W-0		R/W-0		R/W-0	

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

Table 9-4. PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved.
14	PCLK		Enable the PCLK (Video In Pin Mux)
		0	GIO[82]
		1	PCLK
13	CAM_WEN		Enable the CAM_WEN (Video In Pin Mux)
		0	GIO[83]
		1	CAM_WEN
12	CAM_VD		Enable the CAM_VD (Video In Pin Mux)
		0	GIO[84]
		1	CAM_VD
11	CAM_HD		Enable the CAM_HD (Video In Pin Mux)
		0	GIO[85]
		1	CAM_HD
10	YIN_70		Enable the YIN[7:0] (Video In Pin Mux)
		0	GIO[93:86]
		1	YIN[7:0]
9	CIN_10		Enable the CIN[1:0] (Video In Pin Mux)
		0	GIO[95:94]
		1	CIN[1:0]
8	CIN_32		Enable the CIN[3:2] (Video In Pin Mux)
		0	GIO[97:96]
		1	CIN[3:2]
7-6	CIN_4		Enable the CIN[4] (Video In Pin Mux)
		0	GIO[98]
		1	CIN[4]
		2	SPI[2]_SDI
		3	SPI[2]_SDENA[1]

Table 9-4. PINMUX0 - Pin Mux 0 (Video In) Pin Mux Register Field Descriptions (continued)

Bit	Field	Value	Description
5-4	YCIN_5		Enable the CIN[5] (Video In Pin Mux)
		0	GIO[99]
		1	CIN[5]
		2	SPI[2]_SDENA[0]
		3	Reserved
3-2	CIN_6		Enable the CIN[6] (Video In Pin Mux)
		0	GIO[100]
		1	CIN[6]
		2	SPI[2]_SDO
		3	Reserved
1-0	CIN_7		Enable the CIN[7] (Video In Pin Mux)
		0	GIO[101]
		1	CIN[7]
		2	SPI[2]_SCLK
		3	Reserved

9.10.2 PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register

The PINMUX1 register controls pin multiplexing for the VPBE pins.

Figure 9-2. PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register

31	23	22	21	20	19	18	17	16
Reserved		VCLK	EXTCLK		FIELD		DLCD	HVSYNC
R-0		R/W-0	R/W-0		R/W-0		R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUT_0		COUT_1		COUT_2		COUT_3		COUT_4		COUT_5		COUT_6		COUT_7	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-5. PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register Field Descriptions

Bit	Field	Value	Description
31-23	RESERVED	0	Reserved Must be set to 0
22	VCLK		Enable VCLK (Video Out Pin Mux)
		0	VCLK
		1	GI0[68]
21-20	EXTCLK		Enable EXTCLK (Video Out Pin Mux)
		0	GIO[69]
		1	EXTCLK
		2	B2
		3	PWM3
19-18	FIELD		Enable FIELD (Video Out Pin Mux)
		0	GIO[70]
		1	FIELD
		2	R2
		3	PWM3
17	DLCD		Enable DLCD Signal (Video Out Pin Mux)
		0	LCD_OE or BRIGHT
		1	GIO[71]
16	HVSYNC		Enable HVSYNC (Video Out Pin Mux)
		0	HSYNC & VSYNC
		1	GIO[73:72]
15-14	COUT_0		Enable COUT[0] (Video Out Pin Mux)
		0	GIO[74]
		1	COUT[0]
		2	PWM3
		3	Reserved
13-12	COUT_1		Enable COUT[1] (Video Out Pin Mux)
		0	GIO[75]
		1	COUT[1]
		2	PWM3
		3	Reserved
11-10	COUT_2		Enable COUT[2] (Video Out Pin Mux)
		0	GIO[76]
		1	COUT[2]
		2	PWM2
		3	RTO3

Table 9-5. PINMUX1 - Pin Mux 1 (Video Out) Pin Mux Register Field Descriptions (continued)

Bit	Field	Value	Description
9-8	COUT_3		Enable COUT[3] (Video Out Pin Mux)
		0	GIO[77]
		1	COUT[3]
		2	PWM2
		3	RTO2
7-6	COUT_4		Enable COUT[4] (Video Out Pin Mux)
		0	GIO[78]
		1	COUT[4]
		2	PWM2
		3	RTO1
5-4	COUT_5		Enable COUT[5] (Video Out Pin Mux)
		0	GIO[79]
		1	COUT[5]
		2	PWM2
		3	RTO0
3-2	COUT_6		Enable COUT[6] (Video Out Pin Mux)
		0	GIO[80]
		1	COUT[6]
		2	PWM1
		3	Reserved
1-0	COUT_7		Enable COUT[7] (Video Out Pin Mux)
		0	GIO[81]
		1	COUT[7]
		2	PWM0
		3	Reserved

9.10.3 PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register

The PINMUX2 register controls pin multiplexing for the AEMIF pins. Some of the register fields have default values set by external pins that allow control of the AEMIF configuration to match the boot mode.

Figure 9-3. PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved				EM_CLK	EM_ADV	EM_WAIT	EM_WE_OE
R-0				R/W-0	R/W-1	R/W-0	R/W-0
7	6	5	4	3	2	1	0
EM_CE1	EM_CE0	EM_D7_0	EM_D15_8	EM_BA0		EM_A0_BA1	EM_A13_3
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	R/W-0

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

Table 9-6. PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved Must be set to 0.
11	EM_CLK		Enable EM_CLK (AEMIF Pin Mux)
		0	EM_CLK
		1	GIO[31]
10	EM_ADV		Enable EM_ADV (AEMIF Pin Mux)
		0	EM_ADV - Address Valid Detect for OneNAND
		1	GIO[32]
9	EM_WAIT		Enable EM_WAIT (AEMIF Pin Mux)
		0	EM_WAIT
		1	GIO[33]
8	EM_WE_OE		Enable EM_WE_OE (AEMIF Pin Mux)
		0	EM_WE & EM_OE
		1	GIO[35:34]
7	EM_CE1		Enable EM_CE1 (AEMIF Pin Mux)
		0	EM_AO
		1	GIO[36]
6	EM_CEO		Enable EM_CE0 (AEMIF Pin Mux)
		0	EM_CE0
		1	GIO[37]
5	EM_D7_0		Enable EM_D[7:0] (AEMIF Pin Mux)
		0	EM_D[7:0]
		1	GIO[45:38]

Table 9-6. PINMUX2 - Pin Mux 2 (AEMIF) Pin Mux Register Field Descriptions (continued)

Bit	Field	Value	Description
4	EM_D15_8		Enable EM_D[15:8] (AEMIF Pin Mux) Reset value set by AECFG[3] - sets AEMIF bus width for boot OneNAND operation requires PINMUX2[4:1] = AECFG[3:0] = 0010b; i.e., - 16_bit data bus, - full AEMIF address bus, - plus EM_A[14], EM_BA1 used as 16_bit address This puts the AEMIF module in "Half Rate" mode required for OneNAND
		0	EM_D[15:8]
		1	GIO[53:46]
3-2	EM_BA0		Enable EM_BA0 (AEMIF Pin Mux) Reset value set by AECFG[2:1] - sets AEMIF address usage for boot OneNAND operation requires PINMUX2[4:1] = AECFG[3:0] = 0010b; i.e., - 16_bit data bus, - full AEMIF address bus, - plus EM_A[14], EM_BA1 used as 16_bit address This puts the AEMIF module in "Half Rate" mode required for OneNAND
		0	EM_BA0 - Byte address for 8-bit data bus
		1	EM_A14 - Address MSB, required for OneNAND
		2	GIO[54]
		3	Reserved
1	EM_A0_BA1		Enable EM_A0 BA1 (AEMIF Pin Mux) Reset value set by AECFG[0] - sets AEMIF address width for boot OneNAND operation requires PINMUX2[4:1] = AECFG[3:0] = 0010b; i.e., - 16_bit data bus, - full AEMIF address bus, - plus EM_A[14], EM_BA1 used as 16_bit address This puts the AEMIF module in "Half Rate" mode required for OneNAND
		0	EM_A0 & EM_BA1
		1	GIO[56:55]
0	EM_A13_3		Enable EM_A13_3 (AEMIF Pin Mux) Reset value set by AECFG[0] - sets AEMIF address width for boot
		0	EM_A[13:3]
		1	GIO[64:57]

9.10.4 PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register

The PINMUX3 register controls pin multiplexing for the GIO pins.

Figure 9-4. PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register

31	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		GIO7	GIO8	GIO9		GIO10	GIO11	GIO12	GIO13	GIO14	GIO15	GIO16	GIO17	GIO18
R-0		R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIO19		GIO20		GIO21		GIO22		GIO23	GIO24	GIO25	GIO26	GIO27	GIO28	GIO29	GIO30
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 9-7. PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28	GIO7		Enable GIO[7] (GPIO Pin Mux)
		0	GIO[7]
		1	SPI0_SDENA[1]
27	GIO8		Enable GIO[8] (GPIO Pin Mux)
		0	GIO[8]
		1	SPI1_SDO
26-25	GIO9		Enable GIO[9] (GPIO Pin Mux)
		0	GIO[9]
		1	SPI0_SDI
		2	SPI0_SDENA[1]
		3	Reserved
24	GIO10		Enable GIO[10] (GPIO Pin Mux)
		0	GIO[10]
		1	SPI1_SCLK
23	GIO11		Enable GIO[11] (GPIO Pin Mux)
		0	GIO[11]
		1	SPI1_SDENA[0]
22	GIO12		Enable GIO[12] (GPIO Pin Mux)
		0	GIO[12]
		1	UART1_TXD
21	GIO13		Enable GIO[13] (GPIO Pin Mux)
		0	GIO[13]
		1	UART1_RXD
20	GIO14		Enable GIO[14] (GPIO Pin Mux)
		0	GIO[14]
		1	I2C_SCL
19	GIO15		Enable GIO[15] (GPIO Pin Mux)
		0	GIO[15]
		3	I2C_SDA
18	GIO16		Enable GIO[16] (GPIO Pin Mux)
		0	GIO[16]
		3	CLKOUT3
17	GIO17		Enable GIO[17](GPIO Pin Mux)
		0	GIO[17]
		1	CLKOUT2

Table 9-7. PINMUX3 - Pin Mux 3 (GIO/Misc) Pin Mux Register Field Descriptions (continued)

Bit	Field	Value	Description
16	GIO18		Enable GIO[18](GPIO Pin Mux)
		0	GIO[18]
		1	CLKOUT1
15-14	GIO19		Enable GIO[19](GPIO Pin Mux)
		0	GIO[19]
		1	SD1_DATA0
		2	UART2_TXD
		3	Reserved
13-12	GIO20		Enable GIO[20](GPIO Pin Mux)
		0	GIO[20]
		1	SD1_DATA1
		2	UART2_RXD
		3	Reserved
11-10	GIO21		Enable GIO[21](GPIO Pin Mux)
		0	GIO[21]
		1	SD1_DATA2
		2	UART2_CTS
		3	Reserved
9-8	GIO22		Enable GIO[22](GPIO Pin Mux)
		0	GIO[22]
		1	SD1_DATA3
		2	UART2_RTS
		3	Reserved
7	GIO23		Enable GIO[23](GPIO Pin Mux)
		0	GIO[23]
		1	SD1_CMD
6	GIO24		Enable GIO[24](GPIO Pin Mux)
		0	GIO[24]
		1	SD1_CLK
5	GIO25		Enable GIO[25](GPIO Pin Mux)
		0	GIO[25]
		1	ASP0_FSR
4	GIO26		Enable GIO[26](GPIO Pin Mux)
		0	GIO[26]
		1	ASP0_CLKR
3	GIO27		Enable GIO[27] (GPIO Pin Mux)
		0	GIO[27]
		1	ASP0_DR
2	GIO28		Enable GIO[28](GPIO Pin Mux)
		0	GIO[28]
		1	ASP0_FSX
1	GIO29		Enable GIO[29](GPIO Pin Mux)
		0	GIO[29]
		1	ASP0_CLKX
0	GIO30		Enable GIO[30](GPIO Pin Mux)
		0	GIO[30]
		1	ASP0_DX

9.10.5 PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register

The PINMUX4 register controls pin multiplexing for SPI0 and MMC/SD0.

Figure 9-5. PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register

31	3	2	1	0
Reserved		MMCSD0	SPI0_SDI	SPI0_SDENA
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-8. PINMUX4 - Pin Mux 4 (Misc) Pin Mux Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	MMCSD0		Enable MMCSD0
		0	MMC/SD[0] - SD0_CLK,SD0_CMD & SD0_DATA[3:0]
		1	Reserved
1	SPI0_SDI		Enable SPI0_SDI
		0	SPI0_SDI
		1	GIO[102]
0	SPI0_SDENA		Enable SPI0_SDENA0
		0	SPI0_SDENA[0]
		1	GIO[103]

9.10.6 BOOTCFG - Boot Configuration

The device boot configuration (the state of the BTSEL[1:0] and AECFG[3:0] signals are captured in the BOOTCFG register.

Figure 9-6. BOOTCFG - Boot Configuration

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	0
Reserved		GIO0_RESET	BTSEL		Reserved		AECFG	
R-0		R-0	R-0		R-0		R-1101	

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

Table 9-9. BOOTCFG - Boot Configuration Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	GIO0_RESET		GIO0 Value Sampled at Reset Sampled prior to debounce circuit
7-6	BTSEL[1:0]		Configuration at boot of BTSEL[1:0] pins
			Take care that AECFG[3:0] settings, which configure AEMIF pin mux settings in PINMUX2[4:0], are compatible with the boot mode:
			- OneNAND boot requires AECFG[3:0] = 0010b
			- Only 8_bit NAND boot is supported - AECFG[3:0] = 1XXXb
		0	Boot from ROM - NAND/SPI0 Flash boot mode
		1	Boot from AEMIF - OneNAND
		2	Boot from ROM - SD0 boot mode
		3	Boot from ROM - UART0 boot mode
5-4	Reserved	0	Reserved
3-0	AECFG[3:0]		AEMIF Configuration settings for boot by AECFG[3:0] pins
			[3] - AEMIF Data Bus width (0 = 16-bit, 1 = 8-bit)
			[2:1] - Configuration of EM_AN pin
			- 00 = BA0, needed for 8-bit ASYNC memories or devices
			- 01 = A[14], required for OneNAND operation and AEMIF in Half_Rate mode
			- 10 = GIO[54], usable in NAND mode only
			- 11 = reserved
			[0] - AEMIF Address bus width (0 = A{13:3] and A[0], BA1
			Take care that AECFG[3:0] settings, which configure AEMIF pin mux settings in PINMUX2[4:0], are compatible with the boot mode:
			- OneNAND boot requires AECFG[3:0] = 0010b
			- Only 8_bit NAND boot is supported _ AECFG[3:0] = 1XXXb

9.10.7 ARM_INTMUX - ARM Interrupt Mux Control Register

The ARM_INTMUX register provides multiplexing control for interrupts to the ARM since the Interrupt Controller (INTC) can only support 64 discrete events.

Figure 9-7. ARM_INTMUX - ARM Interrupt Mux Control Register

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		INT20	INT25	INT24	INT19	INT18	INT17	INT14	INT13
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 9-10. ARM_INTMUX - ARM Interrupt Mux Control Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	INT20		INT20 - PSC or Reserved
		0	Power Sleep Controller
		1	Reserved
6	INT25		INT25 - ASP0 RINT or ASP1 RINT
		0	ASP0 RINT
		1	ASP1 RINT
5	INT24		INT24 - ASP0 XINT or ASP1 XINT
		0	ASP0 XINT
		1	ASP1 XINT
4	INT19		INT19 - SPI2_INT0 or EDMA TC1 Error Interrupt
		0	SPINT2_0
		1	EDMA TC1 Error
3	INT18		INT18 - SPI1_INT1 or EDMA TC0 Error Interrupt
		0	SPINT1_1
		1	EDMA TC0 Error
2	INT17		INT17 - SPI1_INT0 or EDMA CC Error Interrupt
		0	SPI1 INT0
		1	EDMA CC Error
1	INT14		INT14 - UART2 or TIMER2:TINT5
		0	UART2
		1	TIMER2:TINT5
0	INT13		INT13 - RTO or Timer2:TINT4
		0	RTO
		1	TIMER2:TINT4

9.10.8 EDMA_EVTMUX - EDMA Event Mux Control Register

The EDMA_EVTMUX register controls multiplexing for EDMA Events due to the limited number of events supported by the EDMA.

Figure 9-8. EDMA_EVTMUX - EDMA Event Mux Control Register

31	3	2	1	0
Reserved		EVT26	EVT9	EVT8
R-0		R/W-0	R/W-0	R/W-0

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

Table 9-11. EDMA_EVTMUX - EDMA Event Mux Control Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2	EVT26		EVT26 - MMC/SD[0] Receive
		0	MMC/SD[0] Receive Event
		1	Reserved
1	EVT9		EVT9 - ASP1 Receive or Timer2:TINT5
		0	ASP1 Receive Event
		1	TIMER2:TINT5
0	EVT8		EVT9 - ASP1 transmit or Timer2:TINT4
		0	ASP1 Transmit Event
		1	TIMER2:TINT4

9.10.9 DDR_SLEW - DDR Slew

The DDR_SLEW register allows the CMOSEN field to configure the DDR I/O cells into an LVCMOS input buffer and makes it Mobile DDR compatible.

Figure 9-9. DDR_SLEW - DDR Slew

31	16
Reserved	
R-0	

15	5	4	3	0
Reserved		CMOSEN	Reserved	
R-038h		R-0	R-Fh	

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

Table 9-12. DDR_SLEW - DDR Slew Field Descriptions

Bit	Field	Value	Description
31-16	RESERVED	0	Reserved
15-5	RESERVED	038h	Reserved
4	CMOSEN		Selects Mobile DDR LVCMOS or SSTL18 differential Receiver. A '1' configures the DDR I/O cells into an LVCMOS input buffer and makes it Mobile DDR compatible.
		0	SSTL18 receiver
		1	LVCMOS receiver
3-0	RESERVED	0-Fh	Reserved

9.10.10 CLKOUT - CLKOUT Divisor / Output Control

The CLKOUT register provides control of divisors and output enables for CLKOUT[3:1]. In the device, this register is read only. The CLKOUT[3:1] pins are multiplexed. Use the PINMUX3 register in the system control module to control the pin multiplexing for CLKOUT[3:1]. See Section 9.3.1.2.4 for information on the PINMUX3 register.

Figure 9-10. CLKOUT - CLKOUT div/out Control

31	3	2	1	0
Reserved		CRYS_DIV8	CRYS_DIV3	CRYS_DIV1
R-0		R-1	R-1	R-1

LEGEND: R = Read only; n = value at reset

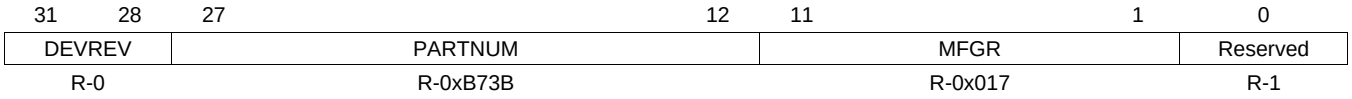
Table 9-13. CLKOUT - CLKOUT div/out Control Field Descriptions

Bit	Field	Value	Description
31-3	RESERVED	0	Reserved
2	CRYS_DIV8	0-1h	CLKOUT3 Enable (CLKOUT3 is crystal frequency (reference clock) divided by 8)
1	CRYS_DIV3	0-1h	CLKOUT2 Enable (CLKOUT2 is crystal frequency (reference clock) divided by 3)
0	CRYS_DIV1	0-1h	CLKOUT1 Enable (CLKOUT1 is crystal frequency (reference clock) divided by 1)

9.10.11 DEVICE_ID - Device ID

The DEVICE_ID register provides the identifying information for the TI ARM processor.

Figure 9-11. DEVICE_ID - Device ID



LEGEND: R = Read only; n = value at reset

Table 9-14. DEVICE_ID - Device ID Field Descriptions

Bit	Field	Value	Description
31-28	DEVREV	0-7h	Device Revision
27-12	PARTNUM	0-FFFFh	Part Number/Device JTAG ID (Uniquely Defined)
11-1	MFGR	0-7FFh	Manufacturer's JTAG ID Texas Instruments' Mfg ID
0	RESERVED	1h	Reserved (Always 1)

9.10.12 VDAC_CONFIG - Video Dac Configuration

the VDAC_CONFIG register provides control of the Video DAC.

Figure 9-12. VDAC_CONFIG - Video Dac Configuration

31	30	29	26	25	24
Reserved		TRESB4R4		TRESB4R2	
R-0		R-0	R/W-0xC	R/W-0x8	

23	22	21	18	17	16
TRESB4R2		TRESB4R1		TRIMBITS	
R/W-0x8		R/W-0xC		R/W-0x37	

15	11	10	9	8
TRIMBITS		PWD_BGZ	SPEED	TVINT
R/W-0x37		R/W-0	R/W-1	R-x

7	6	4	3	2	1	0
PWD_VBFUZ	VREFSET		ACCUP_EN	DINV	Reserved	Reserved
R/W-0	R/W-3		R/W-1	R/W-1	R-1	R-0

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

Table 9-15. VDAC_CONFIG - Video Dac Configuration Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-26	TRESB4R4	0-Fh	Resistance trimming control bit for VREF
25-22	TRESB4R2	0-Fh	Resistance trimming control bit for VREF
21-18	TRESB4R1	0-Fh	Resistance trimming control bit for VREF
17-11	TRIMBITS	0-7Fh	PNP transistor trimming control bit for VREF
10	PWD_BGZ		Power Down of VREFF
		0	Power down
		1	Power up
9	SPEED		Faster operation of VREF transfer
		0	Normal
		1	Faster
8	TVINT		TV cable connect status from DAC
		0	Cable connected
		1	Cable disconnected
7	PWD_VBFUZ		Power down of video buffer
		0	Power down
		1	Power up
6-4	VREFSET	0-7h	VREF setting to video buffer
3	ACCUP_EN		AC capacitor coupling externally to video buffer
		0	Disable the coupling
		1	Enable the coupling
2	DINV		Data invert from VENC (inside the DAC)
		0	No inversion - use only when VDAC is used without VREF and buffer
		1	Inversion - When VDAC is used with VFEF and buffer
1	Reserved	1	Reserved
0	Reserved	0	Reserved

9.10.13 TIMER64_CTL - Timer64+ Input Control

The TIMER64_CTL register provides Timer64+ input control.

Figure 9-13. TIMER64_CTL - Timer64+ Input Control

31	2	1	0
Reserved		GIO3_4	GIO1_2
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-16. TIMER64_CTL - Timer64+ Input Control Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GIO3_4		GIO3 OR GIO4 for input
		0	gio3 for input
		1	gio4 for input
0	GIO1_2		GIO1 OR GIO2 for input
		0	gio1for input
		1	gio2 for input

9.10.14 USB_PHY_CTRL - USB PHY Control

The USB_PHY_CTL register controls various features of the USB PHY.

Figure 9-14. USB_PHY_CTRL - USB PHY Control

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved				DATAPOL	PHYCLKSRC		PHYCLKGD
R-0				R/W-0	R/W-0		R-0

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

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

Table 9-17. USB_PHY_CTRL - USB PHY Control Field Descriptions

Bit	Field	Value	Description
31-12	RESERVED	0	Reserved
11	DATAPOL		USB PHY data polarity inversion
		0	USB PHY data polarity no inversion
		1	USB PHY data polarity inversion
10-9	PHYCLKSRC		USB PHY input clock source
		0	24MHz directly from crystal
		1	12MHz (after dividing 36 MHz crystal by 3)
		2	PLLC1.sysclk3 (backup in case 27MHz crystal is used)
		3	Reserved
8	PHYCLKGD		USB PHY Power and Clock Good
		0	Phy power not ramped or PLL not locked
		1	Phy power is good and PLL is locked
7	SESNDEN		Session End Comparator enable
		0	comparator disabled
		1	comparator enabled
6	VBDTCTEN		vbus comparator enable
		0	comparators (except session end) disabled
		1	comparators (except session end) enabled
5	VBUSENS		OTG analog block VBUSSENSE output status
		0	vbus not present (<0.5V)
		1	vbus present (>0.5V)
4	PHYPLLON		USB PHY PLL suspend override
		0	Normal PLL operation
		1	Override PLL suspend state
3	Reserved	0	Reserved
2	VPSS_OSCPDWN		VPSS oscillator power down control
		0	VPSS MXI2 powered
		1	VPSS MXI2 power off

Table 9-17. USB_PHY_CTRL - USB PHY Control Field Descriptions (continued)

Bit	Field	Value	Description
1	OTGPDWN		USB OTG analog block power down control
		0	OTG analog block powered
		1	OTG analog block power off
0	PHYPDWN		USB PHY power down control
		0	PHY powered
		1	PHY power off

9.10.15 MISC - Miscellaneous Control

The MISC register include miscellaneous control functions.

Figure 9-15. MISC - Miscellaneous Control

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved			TIMER2_WDT	Reserved		PLL1_POSTDIV	AIM_WAIST
R-0			R/W-1	R-0-3h		R/W-1	R/W-0

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

Table 9-18. MISC - Miscellaneous Control Field Descriptions

Bit	Field	Value	Description
31-5	RESERVED	0	Reserved.
4	TIMER2_WDT		TIMER2 Definition (Normal vs. WDT).
		0	TIMER2 is normal Timer.
		1	TIMER2 is WDT.
3-2	RESERVED	0-3h	Reserved.
1	PLL1_POSTDIV		PLL1 post-divider selection.
		0	Sets PLL1 post-divider equal to 1. Only used in >135Mhz speed grade devices.
		1	Sets PLL1 post-divider equal to 2.
0	AIM_WAIST		ARM Internal Memory Wait States.
		0	1 wait state to IRAM.
		1	0 wait state to IRAM. Set this bit for zero wait-state only if the ARM clock frequency is less than or equal to 150 MHz.

9.10.16 MSTPRI0 - Master Priorities 0

The MSTPRI0 registers provides control of the bus masters' DMA priorities.

Figure 9-16. MSTPRI0 - Master Priorities 0

31	7	6	4	3	2	0
Reserved		ARM_CFGP		Reserved	ARM_DMAP	
R-0		R/W-0x1		R-0	R/W-0x1	

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-19. MSTPRI0 - Master Priorities 0 Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-4	ARM_CFGP	0-7h	ARM CFG bus priority
3	Reserved	0	Reserved
2-0	ARM_DMAP	0-7h	ARM DMA priority

9.10.17 Master Priorities 1 (MSTPRI1) Register

The master priorities 1 (MSTPRI1) register is shown in Figure 9-17 and discussed in Table 9-20. It provides control of the bus masters' DMA priorities.

Figure 9-17. Master Priorities 1(MSTPRI1) Register

31	11	10	8	7	0
Reserved		USBP		Reserved	
R-0		R/W-0x4		R-0	

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

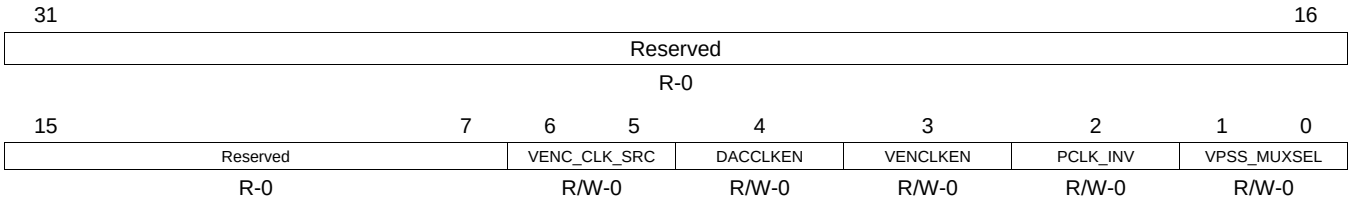
Table 9-20. Master Priorities 1 (MSTPRI1) Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10-8	USBP	0-7h	USB bus priority
7-0	Reserved	0	Reserved

9.10.18 VPSS_CLK_CTRL - VPSS Clock Mux Control

The VPSS Clock multiplexing control is provided by the VPSS_CLK_CTRL register.

Figure 9-18. VPSS_CLK_CTRL - VPSS Clock Mux Control



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

Table 9-21. VPSS_CLK_CTRL - VPSS Clock Mux Control Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-5	VENC_CLK_SRC		27MHz Input Source
		0	PLL1 divided down (SYSCLK3)
		1	External crystal 2 (MXI2/MXO2)
		2	External crystal 1 (MXI1/MXO1)
		3	Reserved
4	DACCLKEN		Video DAC clock enable.
		0	disable
		1	enable
3	VENCLKEN		Video Encoder clock enable.
		0	disable
		1	enable
2	PCLK_INV		Invert VPFE pixel clock (PCLK)
		0	VENC clk mux and CCDC receive normal PCLK
		1	VENC clk mux and CCDC receive inverted PCLK
1-0	VPSS_MUXSEL		VPSS clock selection.
		0	Use clock from PLL1 SYSCLK3, MXI2, or MXI1. Selection is determined by bit VENC_CLK_SRC. (DAC clock = selected clock: PLL1 SYSCLK3, MXI2, or MXI1).
		1	Reserved
		2	Use external VPBE clock input, EXTCLK pin (DAC clock = EXTCLK)
		3	Use pixel clock from VPFE, PCLK pin (DAC clock = off)

9.10.19 Deep Sleep Mode Configuration (DEEPSLEEP) Register

The deep sleep mode configuration (DEEPSLEEP) register is shown in Figure 9-19 and described in Table 9-22 .

Figure 9-19. Deep Sleep Mode Configuration (DEEPSLEEP) Register

31	30	29	16
SLEEPENABLE	SLEEPCOMPLETE	Reserved	
R/W-0	R-1	R-0	

15	4	3	2	1	0
COUNT		Reserved	DRVVBUS_FORCE	DRVVBUS_OVERRIDE	Reserved
R/W-0x176		R-1	R/W-0	R/W-1	R-1

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-22. Deep Sleep Mode Configuration (DEEPSLEEP) Register Field Descriptions

Bit	Field	Value	Description
31	SLEEPENABLE		Enable Deep Sleep Mode When enabled, driving GIO[0] low will initiate Deep Sleep and driving GIO[0] high will initiate wakeup from Deep Sleep. NOTE: After wakeup, Deep Sleep Mode must be disabled to reset the SLEEPCOMPLETE bit.
		0	Disable Deep Sleep mode - normal operation
		1	Enable Deep Sleep Mode
30	SLEEPCOMPLETE		Deep Sleep Wakeup Completed. This bit must be reset to 0 before enabling or initiating Deep Sleep. The ARM should 1) Prepare the device / system for shutdown by placing DDR in auto_refresh and other powerdown housekeeping as necessary and then 2) Enable Deep Sleep Mode (SLEEPENABLE=1) shut down 3) Inform the PMU/MCU it is ready for Deep Sleep 4) Go into a loop polling for this SLEEPCOMPLETE bit to be set, indicating it can proceed with restarting the DDR and other device modules. NOTE: After wakeup, Deep Sleep Mode must be disabled via SLEEPENABLE to reset this bit.
		0	Normal operation or still asleep
		1	Device is awake after Deep Sleep Mode
29-16	Reserved	0	Reserved
15-4	COUNT	0-FFFh	Wakeup Delay Counter. Number of clock cycles (x 16) to count prior to enabling clocks. Used to insure oscillator is stable before enabling clocks.
3	Reserved	0	Reserved
2	DRVVBUS_FORCE		USB_DRVVBUS Force Value When DRVVBUS_OVERRIDE is enabled
1	DRVVBUS_OVERRIDE		USB_DRVVBUS Override Overrides USB_DRVVBUS signal from USB controller and output DRVVBUS_FORCE instead
		0	(NORMAL) USB Controller outputs USB_DRVVBUS
		1	(OVERRIDE) Override USB_DRVVBUS
0	Reserved	0	Reserved

9.10.20 DEBOUNCE[8] - De-bounce for GIO[n] Input

The DEBOUNCE[8] array of registers provide the controls for enabling and configuring Debounce for GIO[7:0] inputs.

Figure 9-20. DEBOUNCE[8] - De-bounce for GIO[n] Input

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

LEGEND: R/W = Read/Write, R = Read only; n = value at reset

Table 9-23. DEBOUNCE[8] - De-bounce for GIO[n] Input Field Descriptions

Bit	Field	Value	Description
31	ENABLE		Debounce Enable
		0	Debounce Enable
		1	Debounce Disable
30-21	RESERVED	0	Reserved
20-0	INTERVAL	0-1F FFFFh	Interval count for the debounce circuit

9.10.21 VTPIOCR - VTP IO Control Register

VTPIOCR is used to calibrate the DDR2/mDDR I/O's. For information on how to calibrate the DDR2/mDDR I/O's using this register, refer to the *TMS320DM355 DDR2/mDDR Peripheral Reference Guide* (SPRUEH7).

Figure 9-21. VTP IO Control Register (VTPIOCR)

31									16
Reserved									
R-0									
15	14	13	12	9	8	7	6	5	0
READY	VTPIOREADY	CLR	Reserved		PWRSAVE	LOCK	PWRDN	Reserved	
R-0	RW-0	RW-1	N-0		R/W-0	RW-0	RW-1	N-0x37	

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

Table 9-24. VTPIOCR - VTP IO Control Field Descriptions

Bit	Field	Value	Description
31-16	RESERVED	0	Reserved
15	READY		VTP Ready Status
		0	VTP not ready
		1	VTP ready
14	VTPIOREADY		VTP IO Ready. Write 1 when VTP IO is ready.
		0	VTP IO not ready
		1	VTP IO ready
13	CLR		VTP Clear. Write 0 to clear VTP flops.
		0	Clear VTP
		1	Un-clear VTP
12-9	RESERVED	0	Reserved
8	PWRSAVE		VTP Power Save Mode
		0	Disable power save mode
		1	Enable power save mode
7	LOCK		VTP Impedance Lock
		0	Unlock impedance
		1	Lock impedance
6	PWRDN		VTP Power Down
		0	Disable power down
		1	Enable power down
5-0	RESERVED	0	Reserved

Reset

10.1 Reset Overview

There are five types of reset in the device. 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 10-1 and further described in the following sections.

Table 10-1. 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 (initiated by ARM emulator).	Resets all modules including memory, except ARM emulation.
Max Reset	ARM emulator or Watchdog Timer (WDT).	Same effect as warm reset.
System Reset	ARM emulator	Resets all modules except memory and ARM emulation. It is a soft reset that maintains memory contents and does not affect or reset clocks or power states.
Module Reset	ARM software	Resets a specific module. Allows the ARM software to independently reset a module. Module reset is intended as a debug tool not as a tool to use in production.

10.2 Reset Pins

Power-On-Reset (POR) and warm reset are initiated by the RESETN and TRSTN pins. The RESETN and TRSTN pins are briefly described in Table 10-2.

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

Table 10-2. Reset Pins

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

10.3 Types of Reset

10.3.1 Power-On Reset (POR)

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

The following steps describe the POR sequence:

1. Apply power and clocks to the chip and drive TRSTN and RESETN low to initiate POR.
2. Drive RESETN high after a required minimum number of MXI clock cycles.
3. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 10.4.1 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.

10.3.2 Warm Reset

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

The following steps describe the warm reset sequence:

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

For reset timing requirements, see the device-specific data manual.

10.3.3 Max Reset

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

The following steps describe the max reset sequence:

1. To initiate max reset, the WDT expires (indicating a runaway condition) or the ARM emulator initiates a max reset command via the IcePick emulation module.
2. Hardware latches the device configuration pins on the rising edge of RESETN. The device configuration pins allow you to set several options at reset. See Section 10.4.1 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.

For information on the WDT, see the *TMS320DM355 Digital Media System-on-Chip (DMSoC) Timer / Watchdog Timer Reference Guide* (SPRUEE5) . See Chapter 3 for information on IcePick emulation.

10.3.4 System Reset

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

The following steps describe the system reset sequence:

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

10.3.5 Module Reset

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

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

10.4 Default Device Configurations

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

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

10.4.1 Device Configuration Pins

The device configuration pins are described in Table 10-3. The device configuration pins are latched at reset and allow you to configure all of the following options at reset:

- ARM Boot Mode
- Asynchronous EMIF pin configuration

These pins are described further in the following sections.

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

Table 10-3. 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 with SPI EEPROM boot option) 01 = Boot from AEMIF 10 = Boot from ROM (MMC/SD) 11 = Boot from ROM (UART)	EM_A[13:12]	00 (NAND)	If any ROM boot mode is selected, GIO61 is used to indicated boot status. If NAND boot is selected, CE0 is used for NAND and SPI0 is used for SPI. Use AECFG[3:0] to configure AEMIF pins for NAND. If AEMIF boot is selected, CE0 is used for AEMIF device (OneNAND, ROM). Use AECFG[3:0] to configure AEMIF pins for NAND. If MMC/SD boot is selected, MMC/SD0 is used.
AECFG[3:0]	Selects the AEMIF pin configuration.	EM_A[11:8]	1101 (NAND)	AEMIF pin configuration. Refer to pin-muxing information in Chapter 9. Note that AECFG[3:0] affects pin configuration for both AEMIF (BTSEL[1:0]=01) and NAND (BTSEL[1:0]=00) boot modes.

10.4.2 PLL Configuration

After POR, warm reset, and max reset, the PLLs and clocks are set to their default configurations. The PLLs are in bypass mode and disabled by default. This means that the input reference clock at MXI1 (typically 24 MHz) drives the chip after reset. For more information, see 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.

10.4.3 Module Configuration

Only a subset of modules are enabled after reset by default. Table 7-1 in Chapter 7 shows which modules are enabled after reset. Furthermore, as shown in Table 7-1, the following modules are enabled depending on the sampled state of the device configuration pins: EMDA (CC and TC0), AEMIF, MMC/SD0, UART0, and Timer0. For example, UART0 is enabled after reset when the device configuration pins (BTSEL[1:0] = 11 - Enable UART) select UART boot mode.

10.4.4 ARM Boot Mode Configuration

The input pins BTSEL[1:0] 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 start of internal ROM (address 0x0000: 8000) 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 routine (i.e., a NAND, MMC/SD, or UART loader routine) .

If AEMIF boot is selected (BTSEL[1:0] = 01), a jump to the start of AEMIF (address 0x0200: 0000) is forced into the first fetched instruction word. The ARM then continues executing from external asynchronous memory using the default AEMIF timings until modified by software.

NOTE: For AEMIF boot, OneNAND must be connected to the first AEMIF chip select space (EM_CE0). The AEMIF does not support direct execution from NAND Flash.

Boot modes are further described in Chapter 11.

10.4.5 AEMIF Configuration

For more information on the AEMIF, see the *TMS320DM355 Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (EMIF) Reference Guide* (SPRUED1).

10.4.5.1 AEMIF Pin Configuration

The input pins AECFG[3:0] determine the AEMIF configuration immediately after reset. Use AECFG[3:0] to properly configure the pins of the AEMIF. Refer to the section on pin multiplexing in Chapter 9.

10.4.5.2 AEMIF Timing Configuration

When AEMIF is enabled, the wait state registers are reset to the slowest possible configuration, which is 88 cycles per access (16 cycles of setup, 64 cycles of strobe, and 8 cycles of hold). Thus, with a 24 MHz clock at MXI/MXO, the AEMIF is configured to run at (6 MHz)/(88) which equals approximately 68 kHz.

Boot Modes

11.1 Boot Modes Overview

The device ARM can boot from either Async EMIF (AEMIF/OneNand) 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 more detail 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 0x0000: 8000, except when BTSEL[1:0] = 01, indicating AEMIF (AEMIF/OneNand) boot. See Chapter 10 for information on the boot selection pins.

11.1.1 Features

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

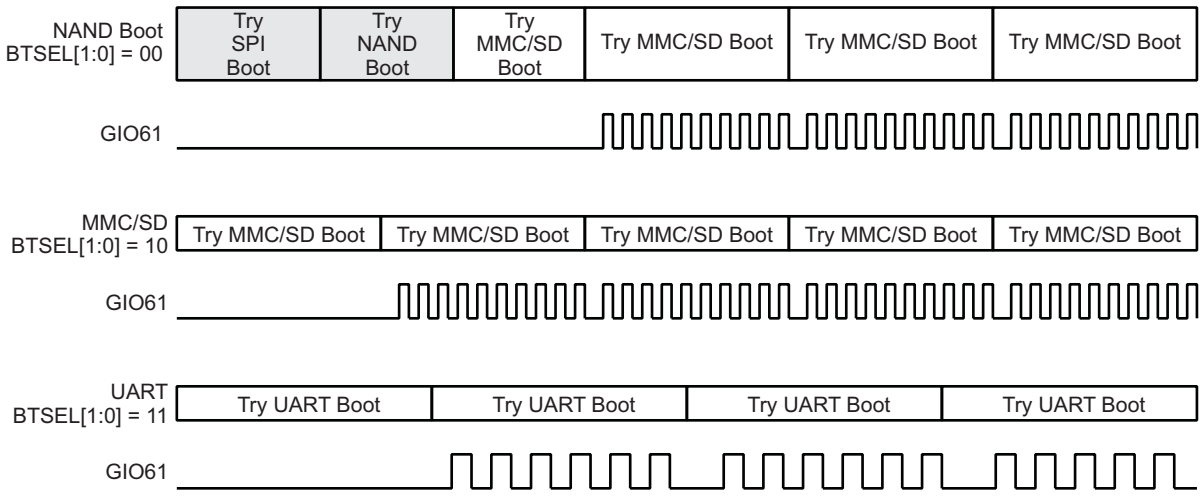
- If BTSEL[1:0] = 01 - Asynchronous EMIF (AEMIF boot. This mode is handled by hardware control and does not involve the ROM. In the case of OneNAND, the user is responsible for putting any necessary boot code in the OneNAND's boot page. This code shall configure the AEMIF module for the OneNAND device. After the AEMIF module is configured, booting will continue immediately after the OneNAND's boot page with the AEMIF module managing pages thereafter.
- The RBL supports three distinct boot modes:
  - BTSEL[1:0] = 00 - ARM NAND/SPI Boot
  - BTSEL[1:0] = 10 - ARM MMC/SD Boot
  - BTSEL[1:0] = 11 - ARM UART Boot
- In NAND boot mode if SPI boot fails, then NAND mode is tried.
- If NAND boot fails, then MMC/SD mode is tried.
- If MMC/SD boot fails, then MMC/SD boot is tried again.
- If UART boot fails, then UART boot is tried again.
- RBL uses GIO61 to indicate boot status (can use to blink LED):
  - After reset, GIO61 is initially driven low (e.g LED off)
  - If NAND boot fails and then MMC/SD boot fails, then GIO61 shall toggle at 4Hz while MMC/SD boot is retried.
  - If MMC/SD boot fails, then GIO61 shall toggle at 4Hz while MMC/SD boot is retried.
  - If UART boot fails, then GIO61 shall toggle at 2Hz while UART boot is retried.
  - When boot is successful, just before program control is given to UBL, GIO61 is driven high (e.g. LED on).
  - Timer 0 shall be used to accurately toggle GIO61 at 4 Hz and 2 Hz.
- ARM ROM Boot - SPI boot from ERPROM in NAND Boot Mode
  - No support for a full firmware boot. Instead, copies a second stage User Boot Loader (UBL) from SPI to ARM Internal RAM (AIM) and transfers control to the user software.
  - Support for 16 and 24 bit SPI EEPROMs.
  - Support for up to 30KB UBL (32KB - ~2KB for RBL stack).
  - RBL will copy UBL to ARM Internal RAM (AIM) via SPI interface from a SPI peripheral like SPI EEPROM. RBL will then transfer control to the UBL.
- ARM ROM Boot - NAND Mode
  The device supports two modes of operations for NAND boot mode: Standard mode and Compatibility mode. The mode of operation is automatically selected by the magic number read from the UBL

descriptor (see Table 11-4)

- 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 8192 bytes in Standard mode (2048 bytes in Compatibility mode).
  Note: At the time this document was prepared only 4K devices were available so 8K has not yet been tested but should function properly.
- Support for magic number error detection and retry (up to 24 times) when loading UBL.
- Support for up to 30KB UBL (32KB IRAM - ~2KB for RBL stack). For NANDs with page size 4KB, 28KB UBL size is supported and for NANDs with page size 8KB, 24KB UBL size is supported.
- Optional, user-selectable, support for use of DMA and I-cache during RBL execution (i.e.,while loading UBL)
- Supports booting from 8-bit NAND devices (16-bit NAND devices are not supported)
- Uses/Requires 4-bit HW ECC (NAND devices with ECC requirements ≤ 4 bits per 512 bytes are supported)
- Supports NAND flash that requires chip select to stay low during the tR read time

• ARM ROM Boot - MMC/SD Mode
  - No support for a full firmware boot. Instead, copies a second stage Uwer Boot Loader (UBL) from MMC/SD to ARm Internal RAM (AIM) and transfers control to the user software.
  - Support for MMC/SD Native protocol (MMC/SD SPI protocol is not supported)
  - Support for descriptor error detection and retry (up to 24 times) when loading UBL
  - Support for up to 30KB UBL (32KB - ~2KB for RBL stack)
• ARM ROM Boot - UART mode
  - No support for a full firmware boot. Instead, loads a second stage user boot loader (UBL) via UART to ARM internal RAM (AIM) and transfers control to the user software.
  - Support for up to 30KB UBL (32KB - ~2KB for RBL stack).

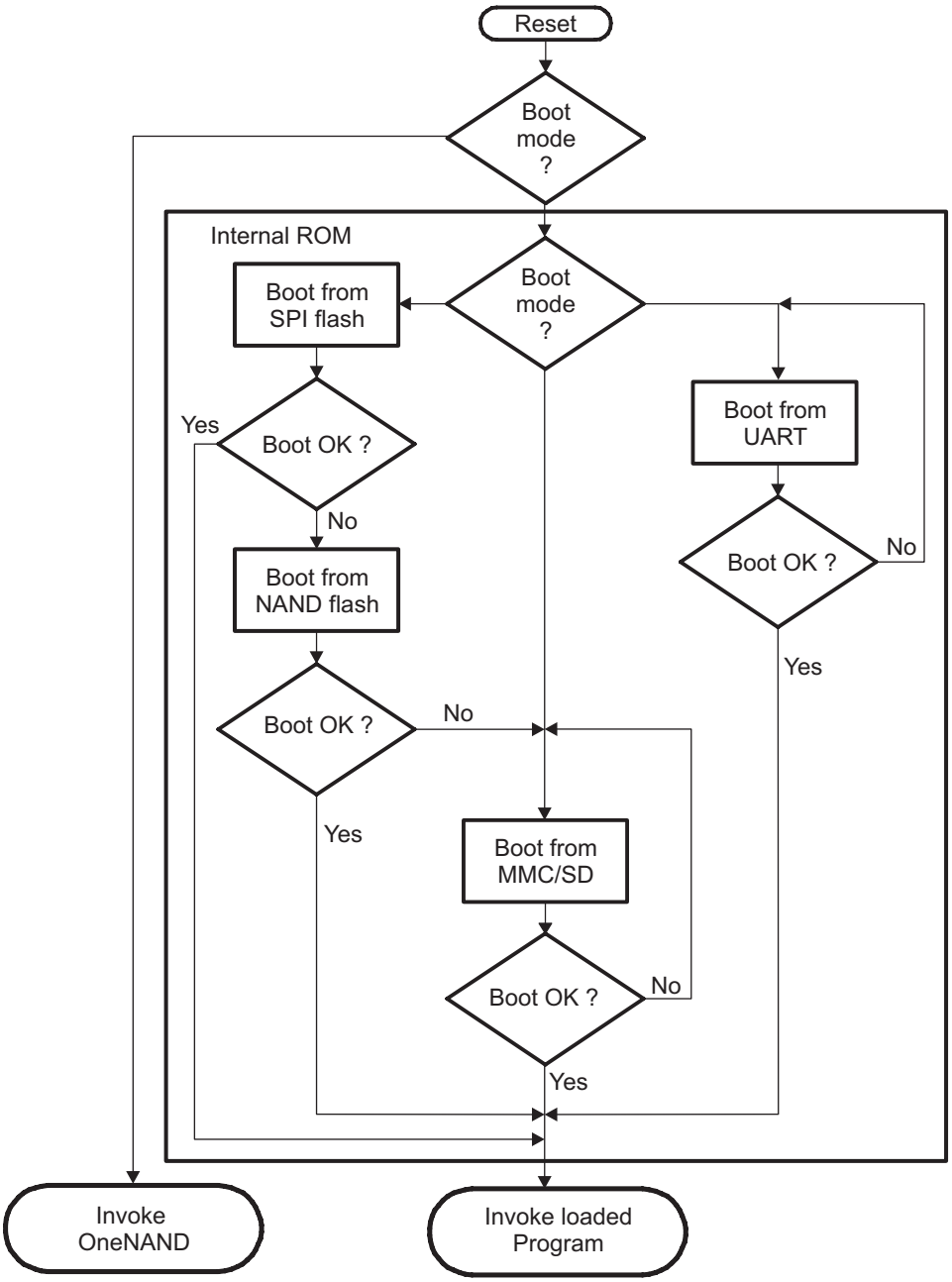
Figure 11-1. Boot Modes Overview



11.1.2 Functional Block Diagram

The general boot sequence is shown in Figure 11-2.

Figure 11-2. Boot Mode Functional Block Diagram



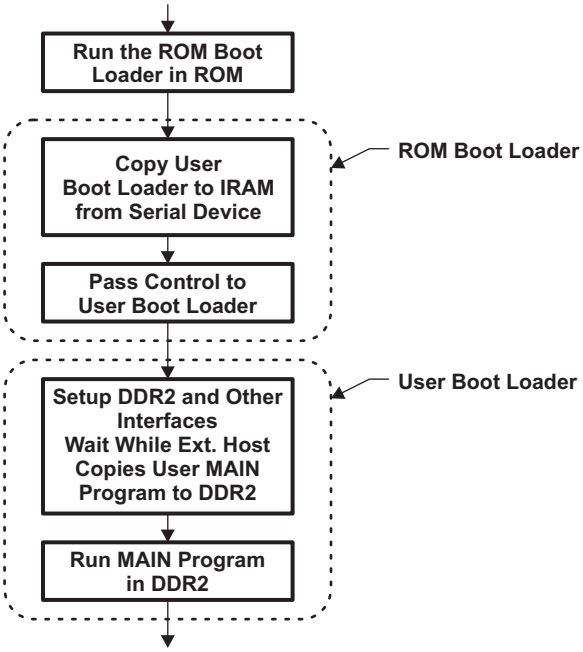
11.2 ARM ROM Boot Mode

DM355's 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.

11.2.1 SPI Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 00, the SPI mode executes (before NAND is executed if needed). The operations followed in the SPI boot mode are described in Figure 11-3.

Figure 11-3. SPI Boot Overview



DM355 loads the UBL data in the following locations, ARM TCM RAM received via SPI0. The UBL data is received from a serial device like serial EEPROM.

11.2.1.1 SPI Key Features

The key features for SPI are as follows:

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

11.2.1.2 SPI Boot - Detailed Flow

The following list describes the flow of the SPI boot:

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

Table 11-1. User Bootloader (UBL) Descriptor for SPI Mode[(1)]

Byte Range	32-bits	Description
8-11	0xA1AC ED0X	Magic number 0xA1ACED00 - 24 bit 0xA1ACED01 - 16 bit
12-15	Entry Point	Entry point address for the user bootloader (absolute address) in ARM internal memory
16-19	UBL size	Size of UBL in bytes
20	Prescaler	Prescaler value to be used for dividing the clock for SPI
21	FASTREAD	Flag for enabling fast read (1 - fast read enables, 0- fast read disabled) Note: FAST READ option may not be valid for a specific EEPROM. Please read the EEPROM specifications before setting this parameter.
22-23	0X0000	Dummy bytes
24-27	Start address of UBL	Start address of UBL in EEPROM
28-31	Load address	Load address of UBL in ARM internal memory

(1) The first 8 bytes might be used for MAC address and is not used by RBL/UBL

11.2.2 NAND Boot Mode

If the value in BTSEL[1:0] from the BOOTCFG register is 00 and the SPI boot fails, the NAND mode executes. The outline of operations followed in the NAND mode is described in Figure 11-2. The NAND boot mode assumes the NAND is located on the EM_CE0 interface, whose bus configuration is configure by the pins AECFG[3:0]. The pins AECFG[3:0] must be configured such that the proper EMIF signals are available for the NAND device.

For NAND boot there are two supported data layout modes: Standard mode and Compatibility mode. For details see Table 11-2 and Table 11-3. Both modes uses a variety of methods are used to determine NAND parameters/geometry. The methods to determine NAND Geometry are detailed in Section 11.2.2.1.

Table 11-2. NAND Layout (Compatibility Mode)

512 Byte Page Size	2048 Byte Page Size
512 bytes Data	512 bytes Data
16 bytes ECC Data	16 bytes ECC Data
	512 bytes Data
	16 bytes ECC Data
	512 bytes Data
	16 bytes ECC Data
	512 bytes Data
	16 bytes ECC Data

Table 11-3. NAND Layout (Standard Mode)

512 Byte Page Size	2048 Byte Page Size	4096 Byte Page Size
512 bytes Data	2048 bytes Data	4096 bytes Data
16 bytes ECC Data	64 bytes ECC Data	128 bytes ECC Data

After the NAND parameters/geometry is determined using any of the used methods, 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 valid UBL magic number, the next block is searched. Searching continues for up to 24 blocks. This provision for additional searching is made in case the first few consecutive blocks have been marked as bad (i.e., they have errors). Searching 24 blocks is sufficient to handle the errors found in virtually all NAND devices.

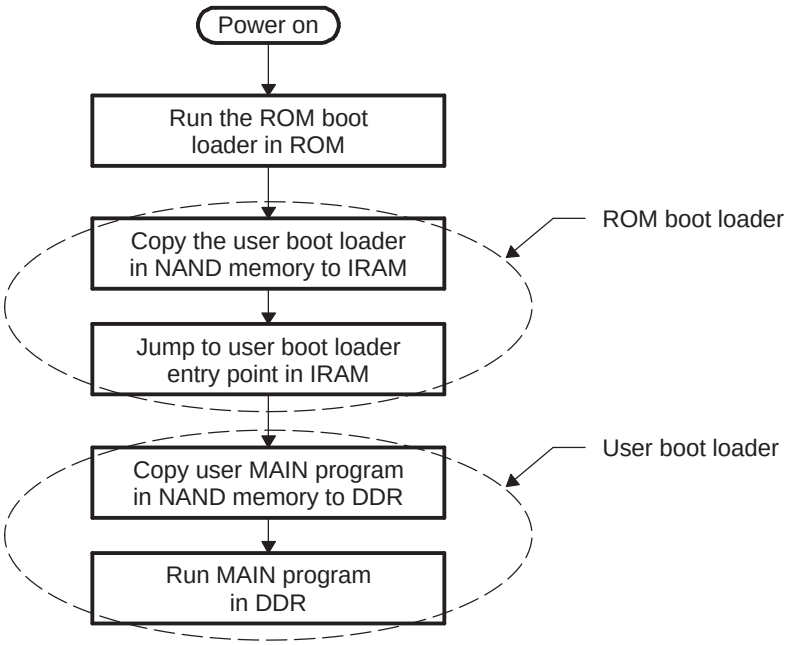
When a valid UBL signature is found, the corresponding block number (from 1 to 24) is written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). This feature is provided as a basic debug mechanism. By reading these 32 bits of memory, via JTAG for example, you can determine in which block the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

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

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

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

Figure 11-4. NAND Boot Flow



The NAND User boot loader UBL descriptor format is described in Table 11-4.

Table 11-4. NAND UBL Descriptor

Page 0 Address	32-Bits	Description
0	0xA1BC EDxx[1]	Magic number (0xA1BCEDxx)
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

(1) Will be 0xA1AC EDxx in Compatibility mode.

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

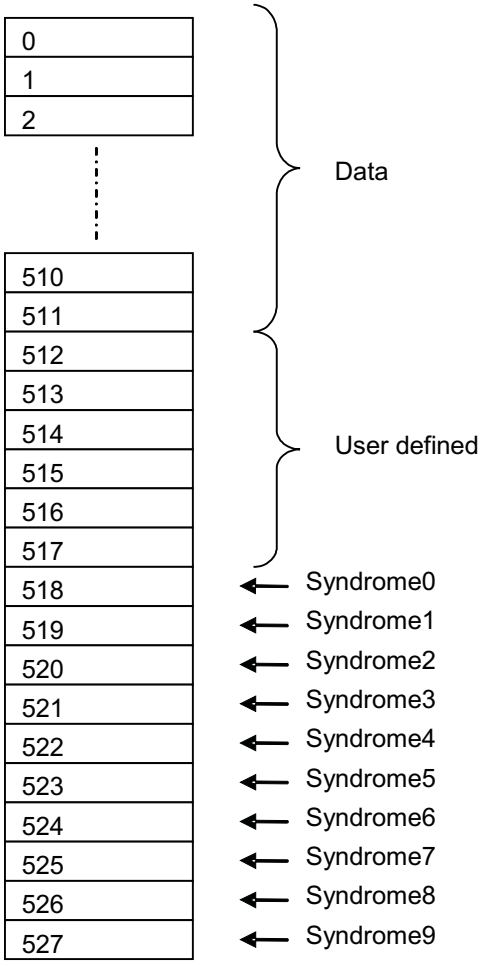
Different NAND boot mode options can be setting different MAGIC IDs in the UBL descriptor. Table 11-5 lists the UBL signatures.

Table 11-5. UBL Signatures and Special Modes

Mode	Value [1]	Description
UBL_MAGIC_SAFE	0x A1AC ED00	Safe boot mode
UBL_MAGIC_DMA	0x A1AC ED11	DMA boot mode
UBL_MAGIC_IC	0x A1AC ED22	I Cache boot mode
UBL_MAGIC_FAST	0x A1AC ED33	Fast EMIF boot mode
UBL_MAGIC_DMA_IC	0x A1AC ED44	DMA + I Cache boot mode
UBL_MAGIC_DMA_IC_FAST	0x A1AC ED55	DMA + I Cache + Fast EMIF boot mode
UBL_MAGIC_SPI_PARAMS	0xA1AC EDAA	NAND parameters from SPI EEPROM

(1) In Standard mode these numbers (except for UBL_MAGIC_SPI_PARAMS) will follow the form 0xA1BC EDxx.

Figure 11-5. 4-Bit ECC Format and Bit 10 to 8-Bit Compression Algorithm



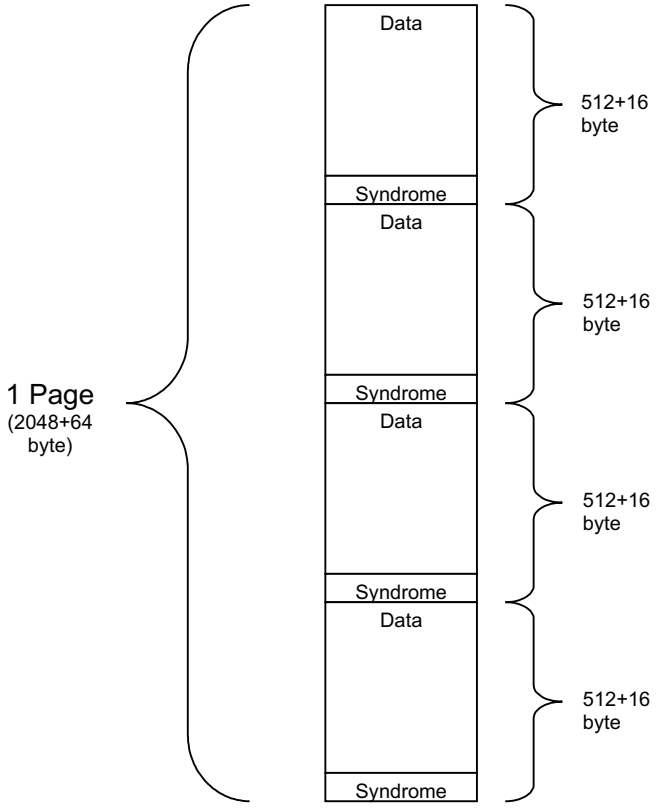
Algorithm to store 10 bit codes in 8 bit words

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

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

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

Figure 11-6. 4-Bit ECC Format for 2048+64 Byte Page Size (in Compatibility Mode)



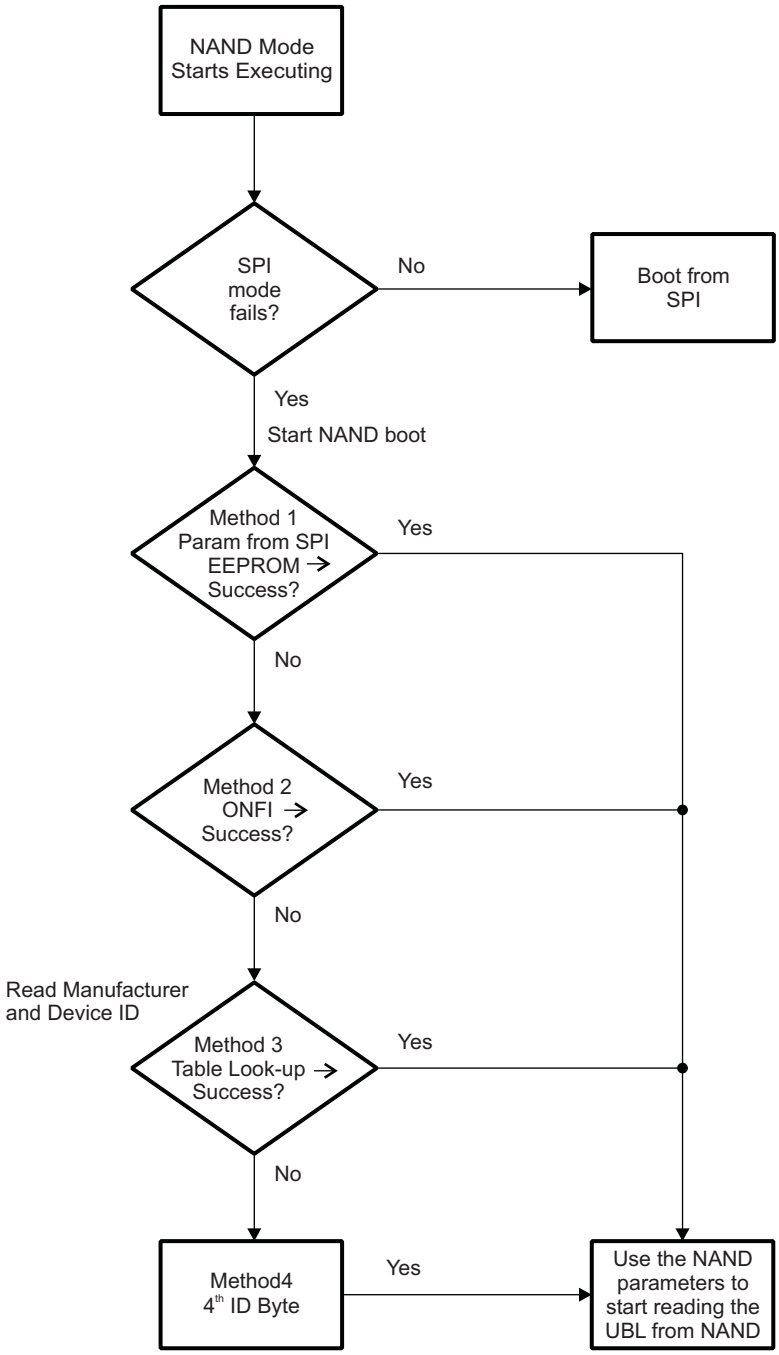
11.2.2.1 Methods to Determine NAND Geometry

A variety of methods are used to determine NAND parameters/geometry. The methods used are:

- NAND parameters from SPI EEPROM (see Section 11.2.2.1.1)
- ONFI (see Section 11.2.2.1.2)
- Table lookup
- 4th ID byte in legacy and Samsung format

The NAND boot mode code flow in silicon revision 1.4 is illustrated in Figure 11-7.

Figure 11-7. NAND Boot Mode Code Flow



11.2.2.1.1 NAND Parameters from SPI EEPROM

To provide more flexibility to obtain the correct NAND parameters, the NAND parameters can be read from a user written EEPROM over SPI. The format of the NAND parameters in EEPROM (as expected by RBL) is defined in Table 11-6.

Table 11-6. NAND Parameters

Offset (in bytes) from Base [1]	Parameter
8	Magic number (0xA1AC EDAA)
12	Page count (Number of pages per block)
16	Number of address cycles
20	Page size (Number of bytes in a page)
24	Spare size (Number of spare bytes in a page)
28	Block shift (Number of bits by which block address is to be shifted)

(1) NAND parameters format in EEPROM *Please note that the first 8 bytes might be used for MAC address and is not used by RBL/UBL.

11.2.2.1.2 ONFI Support

ONFI format is used by many NAND manufacturers including Micron, Numonyx and Hynix. It provides a standard parameter format as compared to the varying 4th ID byte across vendors and NAND generations. If a NAND is detected to be ONFI compatible, the information from the ONFI parameter page is used to interface with NAND.

11.2.2.1.3 4th ID Byte in Legacy and Samsung Format

If the NAND device is not found in the look-up table, then the RBL will read the fourth byte of the NAND ID table and attempt to decode this to obtain the necessary parameters.

For the purpose of determining NAND block size and page size the information from the fourth byte is considered as follows:

- Bits 5 and 4 determine the block size
  - Bits 5,4 = 00: 64KB
  - Bits 5,4 = 01: 128KB
  - Bits 5,4 = 10: 256KB
  - Bits 5,4 = 11: 512KB
- Bits 1 and 0 determine the page size
  - Bits 1,0 = 00: 1KB
  - Bits 1,0 = 01: 2KB
  - Bits 1,0 = 10: 4KB
  - Bits 1,0 = 11: 8KB

In silicon revision 1.4, the latest Samsung (manufacturer ID: 0xEC) 4th ID definition has been added which is as follows:

- Bits 5 and 4 determine the block size
  - Bits 5,4 = 00: 128KB
  - Bits 5,4 = 10: 256KB
  - Bits 5,4 = 01: 512KB
  - Bits 5,4 = 11: 1024KB
- Bits 1 and 0 determine the page size
  - Bits 1,0 = 00: 2KB
  - Bits 1,0 = 01: 3KB
  - Bits 1,0 = 10: 4KB
  - Bits 1,0 = 11: reserved

11.2.2.2 NAND Boot Detailed Flow

An overview of the NAND Boot process is shown in the flow chart in Figure 11-8 and exemplified in Figure 11-9. The following steps describe the NAND Boot process:

- Initialize the stack in the upper ~2K of RAM1 (RAM1 ' 0x7800-0x7FFF). Do not use the last 32-bits of IRAM (0x7ffc-0x8000) for stack, because these will be written with valid block number.

- Disable all interrupts, IRQ and FIQ
- The external pin DEEPSLEEPZ/GIO0 must be driven high during chip reset in order for NAND boot mode to work.
- SPI boot is tried. If the SPI boot fails, NAND boot starts.
- In NAND boot, a variety of methods are used to determine the NAND geometry/parameters (Section 11.2.2.1).
- Search for the User Bootloader magic number in the blocks after CIS/IDI page (CIS/IDI is generally block 0, page 0). See Figure 11-10. The magic number is detected based on reading 0xA1BC EDxx (0xA1AC EDxx for Compatibility mode) in the first 32-bits of page 0 in a block. Only Page 0 of blocks 1 to 24 will be read and searched for the magic number. The magic number for all blocks will be read to ascertain that the block is not an invalid block. For debug purposes, when a valid UBL magic number is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). The UBL Descriptor provides the necessary details of the user boot-loader. See Table 11-4 and Table 11-5 for details of the UBL Descriptor. The UBL Descriptor consists of the following parameters (all UBL parameters are 32-bits wide):
  - Entry Point Address: absolute entry point AFTER loading UBL
    - Must be in range 0x0020 - 0x781C
  - Number of NAND pages in UBL:
    - Must be contiguous pages
    - May span multiple blocks
    - Total bytes must be less than or equal to 30KByte total (size of IRAM - ~2KB stack space)
  - Starting Block of UBL:
    - May be the same block as UBL descriptor
  - Starting Page of UBL
    - May not be the same page as UBL descriptor since full pages must be loaded
- Copy the User boot-loader from NAND flash to IRAM with hardware ECC error detection enabled. If a 4-bit ECC read error is detected, the UBL will correct the error via the ECC correction algorithm. If the read fails due to any other error, the descriptor search process begins anew in the next block after that in which the UBL descriptor was found for up to the first 24 blocks. If no valid UBL descriptor is found after searching 24 blocks, the RBL will try to boot via MMC/SD.
- " Give control to User boot loader at UBL Entry Address. .

Figure 11-8. NAND Boot Mode Flow Chart

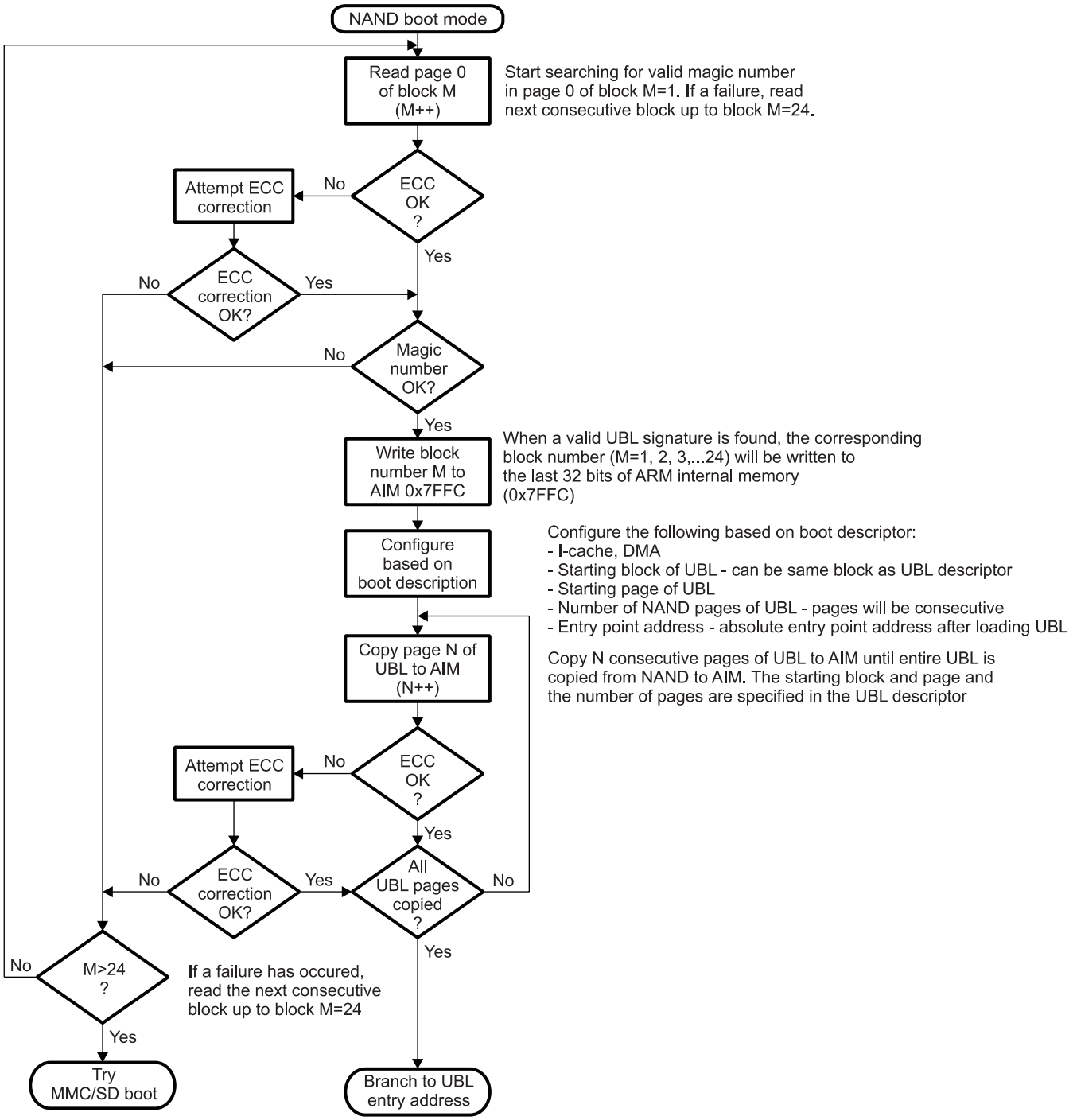
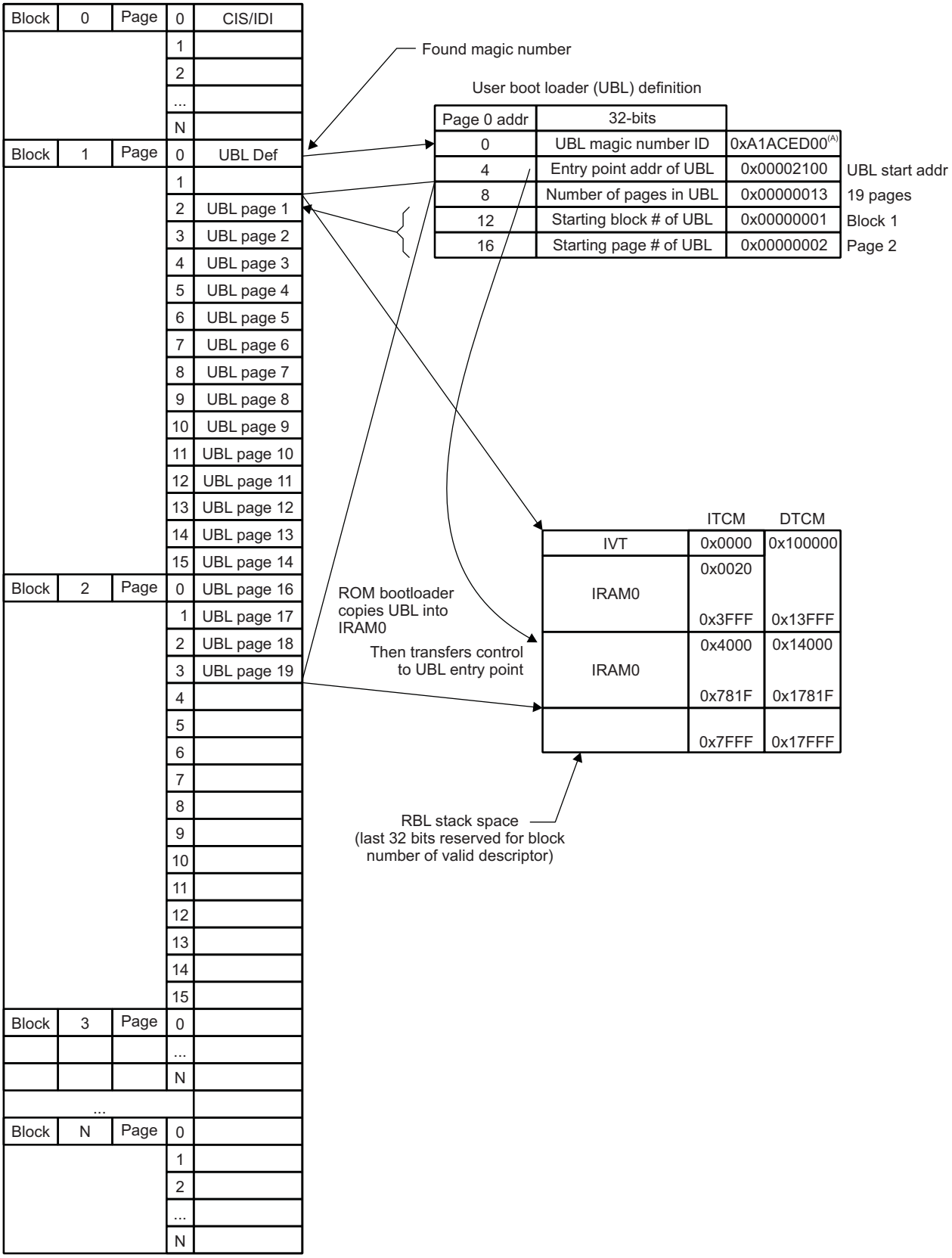
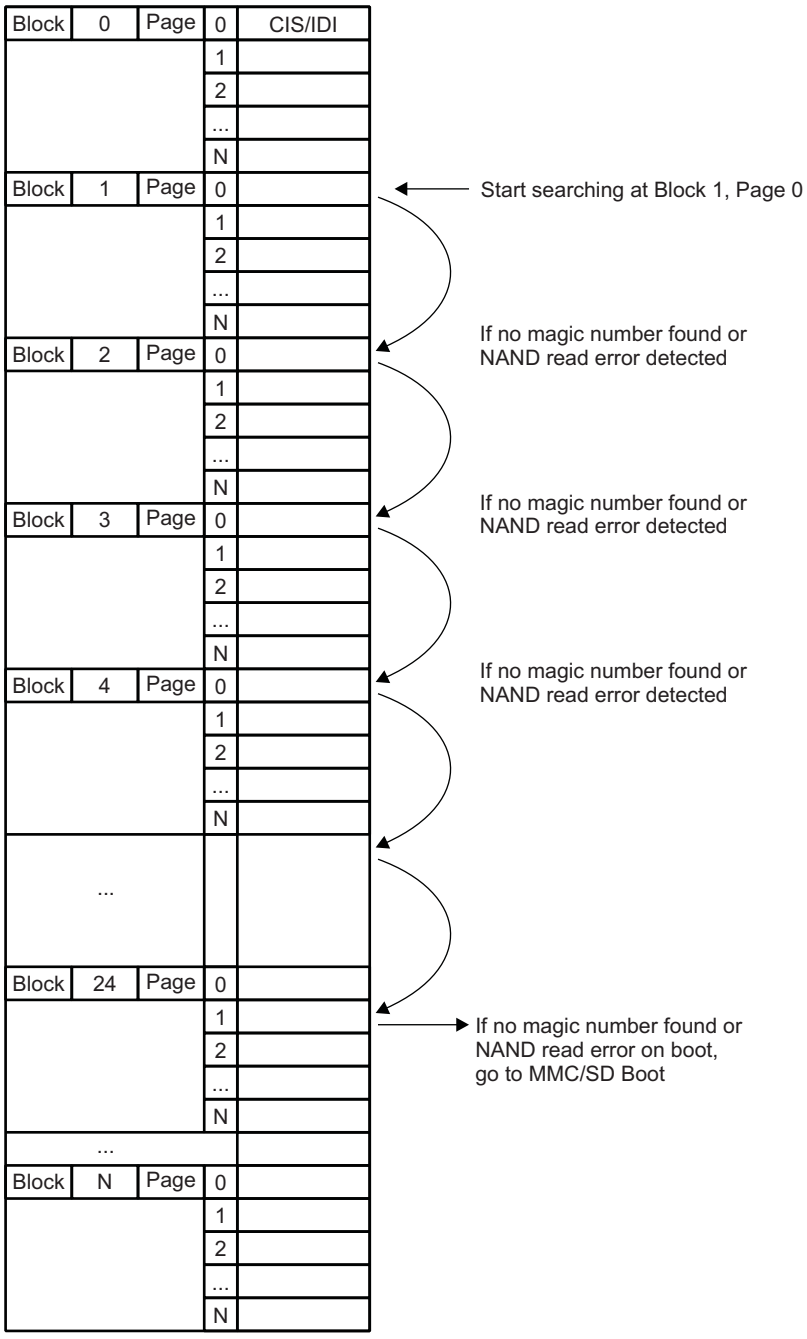


Figure 11-9. ARM NAND ROM Boot Loader Example



A The magic number is 0xA1AC EDxx compatibility mode and 0xA1BC EDxx for standard mode.

Figure 11-10. Descriptor Search for ARM NAND Boot Mode



11.2.2.3 NAND Device IDs Supported

The list of IDs supported by ROM boot loader is shown in Table 11-7 with its characteristics.

Table 11-7. NAND Devices in NAND Device ID Table

DEVICE ID	PAGES PER BLOCK	BYTES PER PAGE	BLOCK SHIFT VALUE FOR ADDRESS	NUMBER OF ADDRESS CYCLES
0xE3	16	512+16	12	3
0xE5	16	512+16	12	3
0xE6	16	512+16	12	3

Table 11-7. NAND Devices in NAND Device ID Table (continued)

0x39[1]	16	512+16	13	3
0x6B	16	512+16	13	3
0x73	32	512+16	13	3
0x33	32	512+16	13	3
0x75	32	512+16	13	3
0x35	32	512+16	13	3
0x43	32	512+16	13	4
0x45	32	512+16	13	4
0x53	32	512+16	13	4
0x55	32	512+16	13	4
0x76	32	512+16	13	4
0x36	32	512+16	13	4
0x79	32	512+16	13	4
0x71	32	512+16	13	4
0x46	32	512+16	13	4
0x56	32	512+16	13	4
0x74	32	512+16	13	4
0xF1	64	2048+64	22	4
0xA1	64	2048+64	22	4
0xAA	64	2048+64	22	5
0xDA	64	2048+64	22	5
0xAC	64	2048+64	22	5
0xDC	64	2048+64	22	5
0xB1	64	2048+64	22	5
0xC1	64	2048+64	22	5

(1) Present only on silicon revision 1.1.

11.2.3 MMC/SD Boot Mode

If the value is BTSEL[1:0] from the BOOTCFG register is '10', the MMC/SD Boot mode will be executed. The outline of operations followed in the MMC/SD mode is depicted in the figure below.

MMC/SD card needs to be powered on for boot up. After the boot up is finished, a GIO may be used as a power switch to the MMC/SD card.

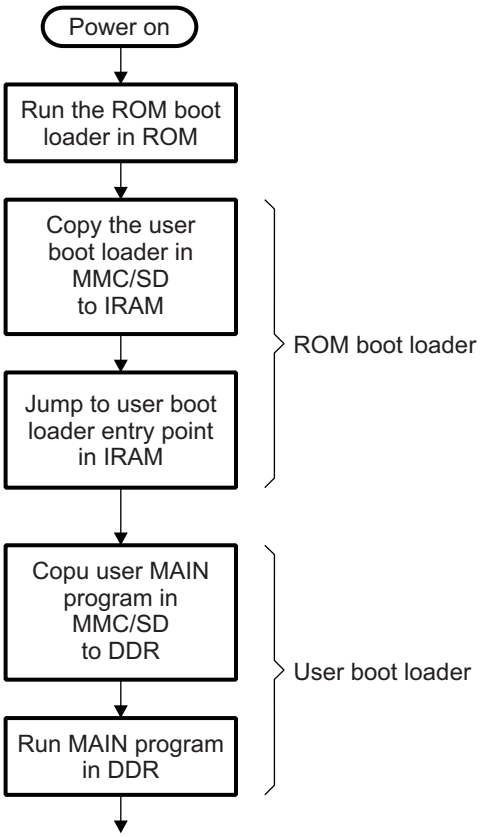
Initialization information, such as block size, is read from the CID and CSD registers of the MMC/SD device. The CID and CSD registers of the MMC/SD device are read by the MMC/SD module in native mode. All initialization and data transfers are done in native mode. SPI mode is not supported.

After performing the MMC/SD initialization sequence, the RBL searches for the UBL Descriptor starting in block 0. If a valid UBL is not found in block 0, as determined by reading a proper UBL magic number, the next block will be searched. Searching will continue for up to 24 blocks. This provision for additional searching is made in case the first few consecutive blocks have errors. When a valid UBL descriptor is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). This feature is provided as a basic debug mechanism. By reading these 32 bits of memory, via JTAG for example, you can determine in which block the RBL found a valid UBL signature. If no valid UBL signature is found after searching 24 blocks, MMC/SD boot will be tried.

The UBL descriptor, which gives the information required for loading and control transfer to the UBL, will then be read and processed. Based on information in the UBL descriptor, the RBL may first enable I-Cache operation. Once the user specified startup conditions are set, the RBL will copy the UBL into ARM Internal RAM, starting at 0x0000:0020. Note that the actual copy will be done to the lower 30KB of the TCM Data area: 0x10020 0x1781F.

The MMC/SD RBL will use the hardware CRC error detection capability to determine if a read error occurs when reading the UBL including the UBL descriptor. If a read error occurs, the UBL copy will immediately halt for that instance of magic number but the RBL will continue to search the block following that block in which the magic number was found for another instance of a magic number. When a magic number is found, the process is repeated. Using this retry process, the magic number and UBL can be duplicated up to 24 times, giving significant redundancy and error resilience to MMC/SD read errors.

Figure 11-11. MMC/SD Boot Mode Overview



The MMC/SD User boot loader UBL descriptor format is described in Table 11-8.

Table 11-8. MMC/SD UBL Descriptor

Page 0 Address	32-Bits	Description
0	0xA1BC EDxx[1]	Magic number (0xA1BCEDxx)
4	Entry Point Address of UBL	Entry point address for the user boot-loader (absolute address)
8	Number of blocks in UBL	Number of blocks (size of user boot-loader in number of blocks)
12	Starting Block # of UBL	Block number where user boot-loader is located

(1) Will be 0xA1AC EDxx in Compatibility Mode

Table 11-9. MMC/SD

Mode	Value	Description
UBL_MAGIC_SAFE	0x A1BC ED00	Safe boot mode
UBL_MAGIC_IC	0x A1BC ED22	I Cache boot mode
UBL_MAGIC_SAFE	0x A1AC ED00	Safe boot mode (compatiblity mode)
UBL_MAGIC_IC	0x A1AC ED22	I Cache boot mode (compatibility mode)

11.2.3.1 MMC/SD Boot Detailed Flow

An overview of the MMC/SD Boot process is shown in the flow chart in Figure 11-12 and exemplified in Figure 11-13. The following steps describe the MMC/SD Boot process:

- There are two instantiations of the MMC/SD Controller in the device: MMCSD0 and MMCSD1. MMC/SD boot shall use only MMCSD0.
- Initialize the stack in the upper 2K of RAM1 in IRAM (RAM1 ' 0x7800-0x7FFF). Do not use the last 32-bits of IRAM (0x7ffc-0x8000) for stack, since these will be written with the block number corresponding to the valid block number.
- Disable all interrupts, IRQ and FIQ
- Read CID and CSD registers of MMC/SD and initialize the MMC/SD module in Native mode.
- Search for the magic number in the UBL descriptor starting in block 0 (see Figure 11-14). The magic number is of the format 0xA1BCEDxx (or 0xA1AC EDxx when in compatibility mode) and is in the first 32-bits of the block. CRC error detection shall be enabled when reading the UBL Descriptor. If a CRC read error is detected or the magic number is not valid, the descriptor search process shall begin anew in the next block after that in which the UBL descriptor was just searched for up to the first 24 blocks. When a valid UBL signature is found, the corresponding block number (from 1 to 24) shall be written to the last 32 bits of ARM internal memory (0x7ffc-0x8000). The UBL Descriptor provides the needed details of the user boot-loader. See Figure 6 and Figure 7 for details of the UBL Descriptor. The UBL Descriptor consists of the following parameters (all UBL parameters are 32-bits wide):
  - Entry Point Address: absolute entry point AFTER loading UBL
    - Must be in range 0x0020 - 0x781C
  - Number of MMC/SD blocks in UBL:
    - Must be contiguous blocks
    - Total bytes must be less than or equal to 30KByte total (size of IRAM - ~2KB stack space)
  - Number of MMC/SD blocks in UBL:
    - Cannot be same block as UBL Definition
- Copy the User boot-loader from MMC/SD to IRAM with hardware CRC error detection enabled. If a CRC read error is detected, the descriptor search process begins anew in the next block after that in which the UBL descriptor was found for up to the first 24 blocks. Detected CRC errors will not be corrected. If no valid magic number is found after searching 24 blocks, MMC/SD boot will be tried.
- Give control to User boot loader at UBL Entry Address.

Figure 11-12. MMC/SD Boot Mode Flow Chart

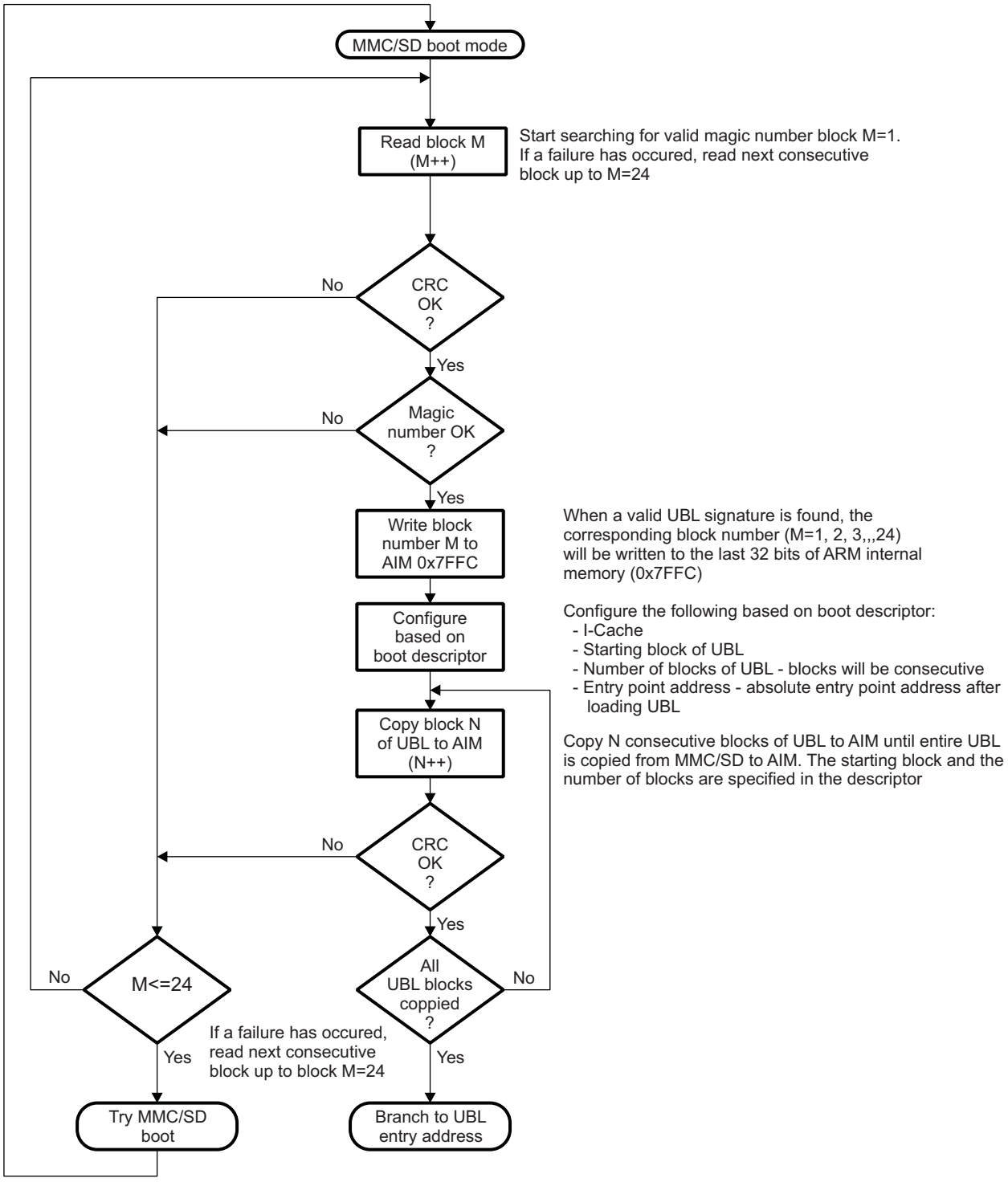
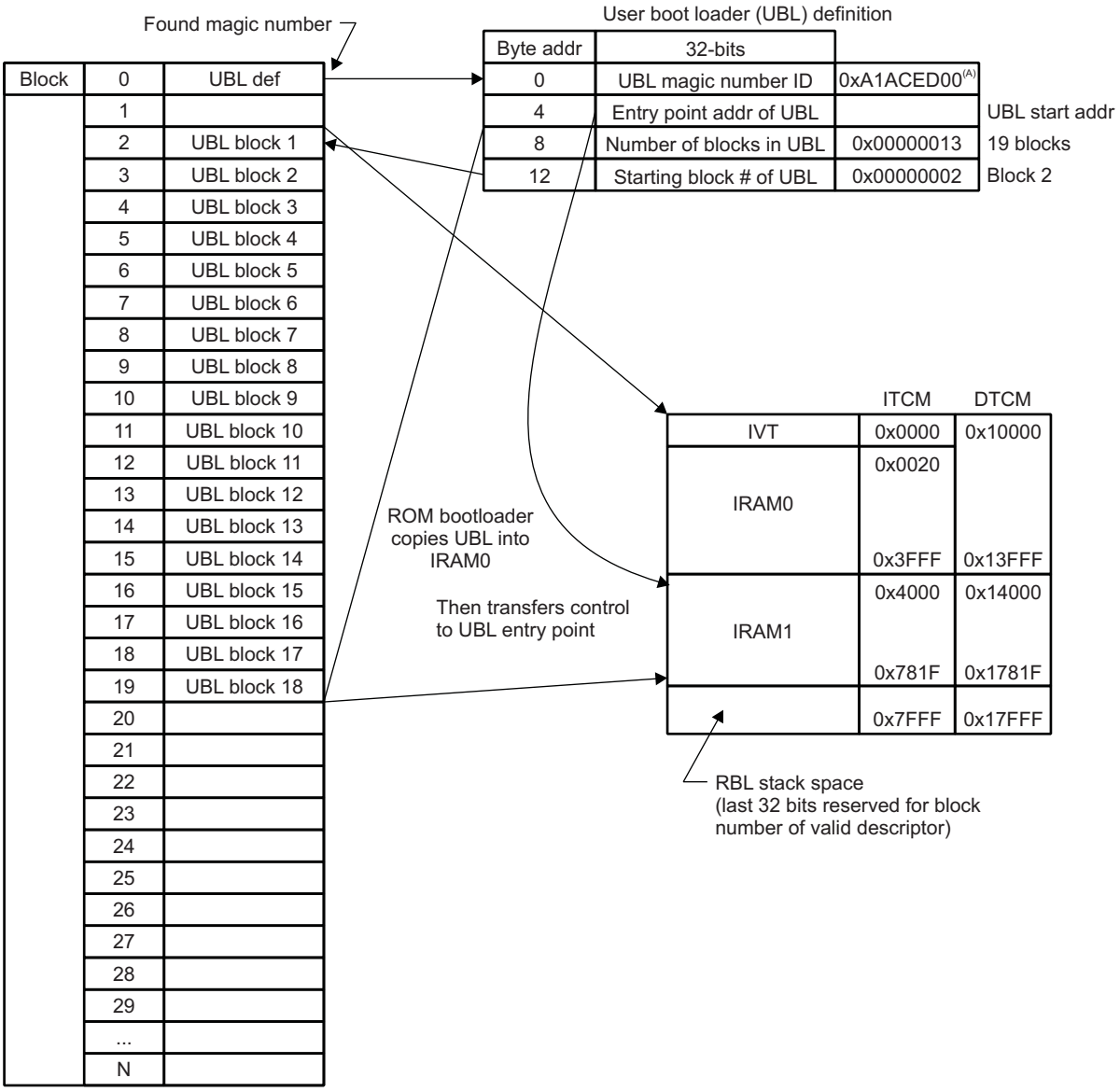
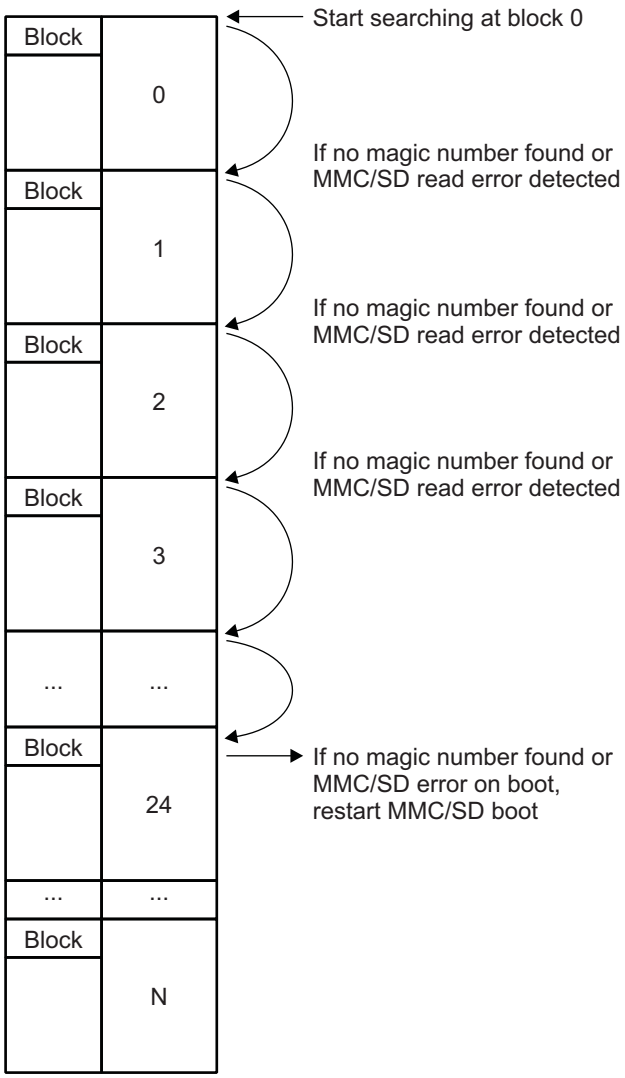


Figure 11-13. ARM MMC/SD ROM Boot Loader Example



A The magic number is 0xA1AC EDxx for compatibility mode and 0xA1BC EDxx for standard mode.

Figure 11-14. Descriptor Search for ARM MMC/SD Boot Mode



11.2.4 UART Boot Mode

If the state of BTSEL[1:0] pins at reset is 11, then the UART 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
Command, data, and checksum format	Everything sent from the host to the device UART RBL must be in ASCII format.

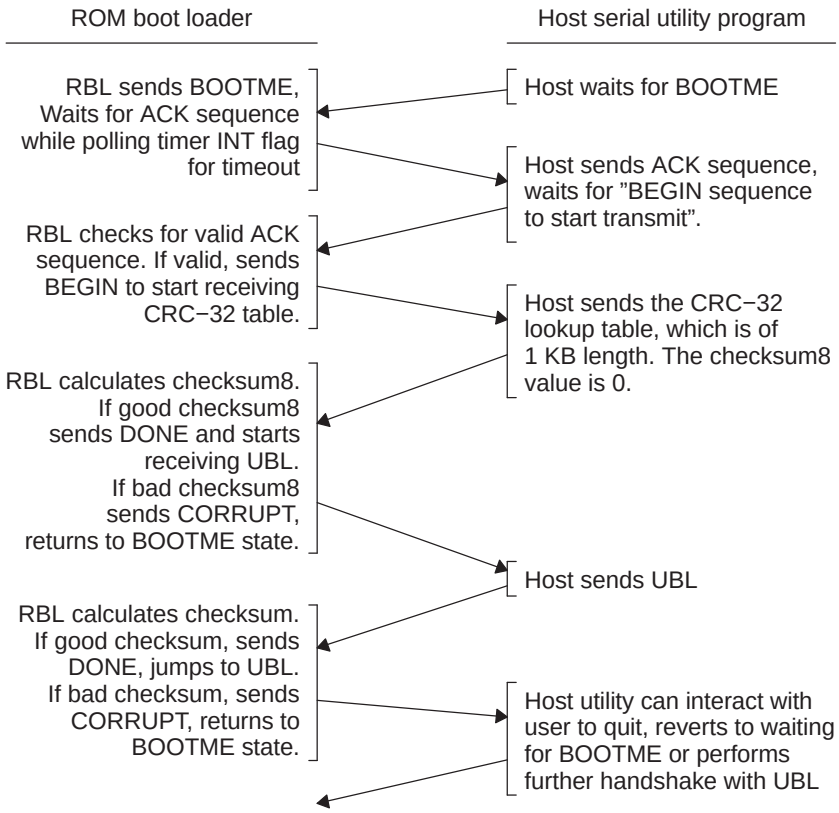
11.2.4.1 Serial Host Handshake

If the state of BTSEL[1:0] pins reset is 11, then the UART boot mode executes as shown in Figure 11-15. The state of BTSEL[1:0] pins at reset is captured and stored in the bits BTSEL in the BOOTCFG register in the System Control module. The RBL reads this register to determine if to execute UART boot. Figure 11-15 shows the handshake between the device and a serial host utility program. After initialization, there are three main receive sequences: ACK, 1KB CRC32 table, and user boot loader (UBL). For each receive sequence, a time-out check is done in the RBL. This means that if a timeout value is reached during the sequence, the serial boot mode restarts from the beginning at which the RBL sends out the BOOTME message. The error checking behavior for the UART receive mode is the same. For each byte received, if there is an error, RBL restarts from the beginning. In UART boot mode, the RBL copies the UBL code starting at address 0x0100. Note that for all other boot modes (e.g., NAND and MMCSD boot modes) the UBL code is copied starting at address 0x0020. So, UART boot mode is different in terms of starting copy address. The starting address of the UBL is at 0x0100.

The check sum method used for UBL data is CRC 32 check sum. The lookup table that is used for the CRC 32 calculation (1KB) must be sent by the host serial utility. Check sum8 is used as the check sum methodology for the CRC 32 lookup table.

The check sum8 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, CRC 32 table or UBL is received, the serial boot process restarts. GIO61 will toggle while retrying UART boot, as discussed previously.

Figure 11-15. UART Boot Mode Handshake



11.2.4.2 UART Boot Loader Data Sequences

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

Table 11-10 lists the values for the handshake sequences and header for UBL.

Table 11-10. 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 check sum is a 32-bit check sum. Note that the RBL jumps the program counter to the start address (i.e. UBL entry point) after the downloading process.
	UBL 8-byte check sum	
	UBL 4-byte count	
	UBL 4-byte ARM physical start address	
	TBD 4-byte zeros	
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 CRC 32 check sum value is calculated for the UBL data and passed by the host serial utility. The polynomial used for CRC32 is:

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

11.2.4.3 Host Utility Data Format

The RBL expects the data sent from the host utility to be in a particular format. This section describes the data format for the ACK, 1KB CRC32 table, and UBL sequences.

All data sent from the host to the RBL must be in ASCII format. The host utility must transfer the ACK sequence as shown in Table 11-11, CRC32 table as shown in Table 11-12, and UBL in the same format as the CRC32 table (Table 11-12). The check sum8 is calculated as shown in Table 11-12. The check sum8 is calculated after each transfer completes. Also note that since eight bytes are necessary to do the check sum8 calculation, the host utility must be sure to send a multiple of eight bytes of total data.

Example 11-1. Host Serial Utility Transmission of Characters

For a given UBL data, let the check sum (CRC32) value calculated be 0x FFAA 10A1. Then, instead of the host utility transmitting “ascii (0xff) ascii (0xaa) ascii (0x10) ascii (a1)” , it will transmit “ffaa 10a1”. These 8 characters (bytes) are appropriately interpreted by the RBL.

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

Table 11-11. Host Utility Data Format

"^^^^ACK/0" Transfer

	D0	D1	D2	D3	D4	D5	D6	D7	
Start Byte ——→	"^"	"^"	"^"	"^"	"A"	"C"	"K"	"/0"	

Checksum "9af944c9" Transfer

D0	D1	D2	D3	D4	D5	D6	D7	
9	a	f	9	4	4	c	9	——→

Table 11-12. CRC32 Table Transfer

crc32_table[1024] = {0x01234567L, 0x89ABCDEFL....}

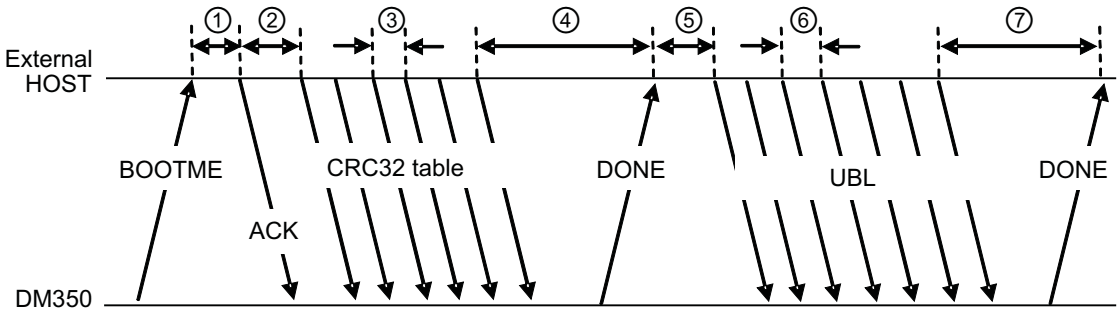
	D0	D1	D2	D3	D4	D5	D6	D7	
Start Byte ——→	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	

D0	D1	D2	D3	D4	D5	D6	D7	
"8"	"9"	"A"	"B"	"C"	"D"	"E"	"F:	——→

0x67 + 0x45 + 0x32 + 0x01 + 0xFE + 0xCD + 0xAB + 0x89 + ... =>checksum8

11.2.4.4 Host Utility Timing Requirements

The UART host utility must allow the RBL time to process commands and data. When sending characters from the host to the UART RBL, the host utility must insert a delay between each byte character equal to 1 ms. Furthermore, 5 ms delay must be inserted for each of the timing parameters shown in Figure 11-16.



(1) The delay time from "BOOTME" received until "^^^ACK" sent.
(2) The delay time from "ACK" sent to CRC32 table data sent.
(3) The delay time from CRC32-table data sent to next CRC32-table data sent.
(4) The delay time from final CRC32-table data sent to "DONE" or "CORREPT" received.
(5) The delay time from "DONE" or "CORREPT" received until UBL data sent.
(6) The delay time from UBL data sent until next UBL data sent.
(7) The delay time from final UBL data sent until "DONE" or "CORREPT" received.

Figure 11-16. Host Utility Timing

Power Management

12.1 Overview

The device is designed for minimal power consumption. There are two components to power consumption: active power and leakage power. Active power is the power consumed to perform work and scales with clock frequency and the amount of computations being performed. Active power can be reduced by controlling the clocks in such a way as to either operate at a clock setting just high enough to complete the required operation in the required timeline or to run at a clock setting until the work is complete and then drastically cut the clocks (e.g., 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 device includes several power management features which are briefly described in Table 12-1 and detailed in the following sections.

Table 12-1. Power Management Features

Power Management Features	Description
Clock Management	
Module clock disable	Module clocks can be disabled to reduce switching power
Module clock frequency scaling	Module clock frequency can be scaled to reduce switching power
PLL power-down	The PLLs can be powered-down when not in use to reduce switching power
ARM Sleep Mode	
ARM Wait-for-Interrupt sleep mode	Disable ARM clock to reduce active power
System Sleep Modes	
Deep Sleep Mode	Stop all device clocks and power down internal oscillators to reduce active power to a minimum. Registers and memory are preserved.
I/O Management	
USB Phy power-down	The USB Phy can be powered-down to reduce USB I/O power
DAC power-down	The DAC's can be powered-down to reduce DAC power
DDR self-refresh and power down	The DDR device can be put in self-refresh and power down states

12.2 PSC and PLLC Overview

The power and sleep controller (PSC) plays an important role in managing system power on/off, clock on/off, and reset. Similarly, the PLL controller (PLLC) plays an important role in device clock generation. 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.

12.3 Clock Management

12.3.1 Module Clock Disable

The module clock disable feature allows software to disable individual module clocks, in order to reduce a module's active power consumption to 0. The device is designed in full static CMOS; thus, when a module clock stops, the module's state is 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.

The power and sleep controller (PSC) controls module clock disable/enable. The procedure to disable/enable module clocks is described in Chapter 7.

12.3.2 Module Clock Frequency Scaling

Module clock frequency is scalable by bypassing the PLL's or 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 how to bypass the PLL's and how to program PLL frequency.

12.3.3 PLL Bypass and Power Down

You can bypass the PLLs in the device. Bypassing the PLLs sends the PLL reference clock to the post dividers of the PLLC instead of to the PLL VCO output clock. The PLL reference clock is typically at 24 MHz; therefore, you can use this mode to reduce the core and module clock frequencies to very low 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.

12.4 ARM Sleep Mode Management

12.4.1 ARM Wait-For-Interrupt Sleep Mode

The ARM module cannot have its clock turned off/on via the PSC module like other modules. However, the ARM includes a special sleep mode called “wait-for-interrupt”. When the wait-for-interrupt mode is enabled, the clock to the CPU core is shut off and the 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 (e.g., an external interrupt).
- Enable wait-for-interrupt mode using the following CP15 instruction:
  - mcr p15, #0, rd, c7, c0, #4

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

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

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

NOTE: The ARM interrupt controller and the module sourcing the wakeup interrupt (e.g., GIO or WDT) must not be disabled, or the device will never wake up.

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

12.5 System Sleep Modes

12.5.1 Deep Sleep Mode

Deep Sleep mode is a special power down mode in which all device clocks are stopped and the internal oscillators are powered down. Registers and software are preserved. Thus, upon recovery, the program may continue from where it left off with minimal overhead involved.

The Deep Sleep power down process works as follows:

- The ARM prepares for power down, typically after an external microcontroller notifies the ARM to prepare for power down via an interrupt or serial communication.
- The ARM puts DRR in its self-refresh state. Program in DDR is preserved while DDR is in its self-refresh state. In the case of mDDR, you may utilize Partial Array Self Refresh (PASR) for additional power savings.
- To reduce the chip stand by power, it is advised to power down all the analog blocks (PLL cores, DDR PHY DLL, DDR PHY, USB PHY, and Video DAC).
- The ARM sets SLEEPENABLE in register DEEPSLEEP in the System module.
- The ARM informs the microcontroller that it has initiated Deep Sleep and begins polling SLEEPCOMPLETE in DEEPSLEEP. During the recovery process, the ARM will wake up and detect that SLEEPCOMPLETE has changed.
- The microcontroller transitions GIO0 from high to low and then continues to hold GIO0 low (for a minimum of 500 ns) until it desires to exit Deep Sleep mode. The transition of GIO0 from high to low creates a clock pulse advancing the Deep Sleep state machine. After this transition, all clocks are stopped and then the internal oscillators are powered down.
- At this point, the device is in Deep Sleep mode; power is reduced to a minimum.

The Deep Sleep wake up process works as follows:

- To initiate the wake up process, the microcontroller transitions GIO0 from low to high. This transition creates a clock pulse advancing the Deep Sleep state machine. After this transition, the oscillators are powered up and allowed to stabilize and then all clocks are restarted.
- The ARM detects that SLEEPCOMPLETE has changed
- The ARM clears SLEEPENABLE in register DEEPSLEEP
- The ARM brings DDR out of self refresh.
- At this point the ARM resumes normal operation. The ARM may branch to program code preserved in DDR. Register states are preserved.

12.6 I/O Management

12.6.1 USB Phy Power Down

You can power-down the USB Phy when it is not in use. The USB Phy is powered-down via the PHYPWDN bit in the USB_PHY_CTL register of the system control module. USB_PHY_CTL is described in Chapter 9.

Also, see the *TMS320DM355 DMSoC Universal Serial Bus (USB) Controller Reference Guide* (SPRUED2) for more information.

12.6.2 Video DAC Power Down

The VPBE includes one video digital-to-analog converter (DAC) to drive analog displays, such as NTSC and PAL television displays. The Video Encoder (VENC) module of the VPBE includes registers for enabling/disabling the DAC. You can use the VIE bit in VMOD to force the analog output of the DAC to a low level, regardless of the video signal. Furthermore, you can use the DACCLKEN in register VPSS_CLK_CTRL to disable each DAC clock.

See the *TMS320DM355 Video Processing Back End (VPBE) Peripheral Reference Guide* (SPRUF72) for detailed information on DAC power-down.

12.6.3 DDR Self-Refresh and Power Down

The DDR controller supports self-refresh and power down. This allows you to put the DDR device in its self-refresh or power down states for power savings.

See the *TMS320DM355 DMSoC DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Reference Guide* (SPRUEH7) for detailed information on DDR self-refresh and power down.

Revision History

This document has been revised from SPRUFB3 to SPRUFB3A because of the following technical change(s).

Table A-1. Revision History

Location	Additions, Deletions, Changes
Global	Corrected Literature numbers for User Guides.
Global	Removed AEMIF FAST boot support.
Section 1.1	Updated section.
Figure 1-1	Corrected figure.
Section 2.2	Removed CFALD bullet.
Table 4-2	Updated table.
Section 6.6	Updated Reset values of some PLL Controller registers.
Figure 6-4	Changed REV bit Read value from R-2 to R-D in table.
Table 7-1	Fixed BSEL default states for RBL modules and added SPI boot mode support in NAND RBL mode.
Section 6.6.1	Updated PID register information.
Section 7.7.13	Updated PDSTAT register STATE field descriptions.
Section 7.7.14	Updated PDCTL register name.
Figure 8-6	Changed reset value on bit FIQ[63:32] from R to R/W.
Figure 8-12	Changed reset value on bit EINT[63:32] from R to RW.
Table 9-2	Added table note.
Section 9.10.2	Updated EXTCLK and FIELD bit description values.
Section 9.10.4	Added Reserved to GIO9 field description in GIO9 bit description and corrected the ASP signal names.
Section 9.10.5	Changed bit 2 name from MMCSD0_MS to MMCSD0.
Section 9.10.8	Corrected EVT26 bit value 1 description.
Section 9.10.9	Corrected register and added Mobile DDR LVCMOS enable bit..
Table 9-13	Added values to table.
Table 10-3	Added SPI boot option to NAND.
Table 9-14	Added value to Reserved bit 0.
Section 9.10.15	Changed bits 3-2 to Reserved.
Section 10.4.5.2	Updated frequency values.
Section 11.1.1	Changed bullets: • Added SPI to ARM NAND boot mode in 3rd bullet • Added 6th bullet • Added bullets 17, 18, 19, 20, and 21.
Figure 11-1	Updated figure.
Section 11.2.1	Added section and two subsections.
Section 11.2.2	Updated section.
Section 11.2.2.1	Added subsection.
Section 11.2.2.1.1	Added subsection.
Section 11.2.2.2	Corrected bullets 4 and 5.
Section 12.5.1	Removed CCP PHY from section.

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265