



# *AMD Alchemy™ Au1100™ Processor Data Book*

---

*April 2006*

Publication ID: 30362D

---

© 2006 Advanced Micro Devices, Inc. All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

### **Trademarks**

AMD, the AMD Arrow logo, and combinations thereof, and AMD Alchemy and Au1100 are trademarks of Advanced Micro Devices, Inc.

MIPS32 is a trademark of MIPS Technologies.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other jurisdictions.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

# Contents

---

<b>List of Figures</b> .....	<b>5</b>
<b>List of Tables</b> .....	<b>7</b>
<b>1.0 Overview</b> .....	<b>11</b>
1.1 Product Description .....	11
1.2 Features .....	12
<b>2.0 CPU</b> .....	<b>13</b>
2.1 Core .....	13
2.2 Caches .....	15
2.3 Write Buffer .....	21
2.4 Virtual Memory .....	24
2.5 Exceptions .....	25
2.6 MIPS32™ Instruction Set .....	26
2.7 Coprocessor 0 .....	28
2.8 System Bus (SBUS) .....	40
2.9 EJTAG .....	41
<b>3.0 Memory Controllers</b> .....	<b>43</b>
3.1 SDRAM Memory Controller .....	44
3.2 Static Bus Controller .....	53
<b>4.0 DMA Controller</b> .....	<b>75</b>
4.1 DMA Configuration Registers .....	75
4.2 Using GPIO as External DMA Requests (DMA_REQn) .....	80
4.3 Programming Considerations .....	81
<b>5.0 Interrupt Controller</b> .....	<b>83</b>
5.1 Interrupt Controller Sources .....	83
5.2 Register Definitions .....	85
5.3 Hardware Considerations .....	88
5.4 Programming Considerations .....	88
<b>6.0 Peripheral Devices</b> .....	<b>89</b>
6.1 AC97 Controller .....	89
6.2 USB Host Controller .....	95
6.3 USB Device Controller .....	97
6.4 IrDA .....	109
6.5 Ethernet MAC Controller .....	124

6.6	I2S Controller	146
6.7	UART Interfaces	151
6.8	SSI Interfaces	160
6.9	LCD Controller	166
6.10	Secure Digital (SD) Controller	178
6.11	Secondary General Purpose I/O	189
<b>7.0</b>	<b>System Control</b>	<b>193</b>
7.1	Clocks	194
7.2	Time of Year Clock and Real Time Clock	204
7.3	Primary General Purpose I/O and Pin Functionality	209
7.4	Power Management	214
<b>8.0</b>	<b>Power-up, Reset and Boot</b>	<b>223</b>
8.1	Power-up Sequence	223
8.2	Reset	223
8.3	Boot	225
<b>9.0</b>	<b>EJTAG</b>	<b>227</b>
9.1	EJTAG Instructions	227
9.2	Debug Exceptions	227
9.3	Coprocessor 0 Registers	228
9.4	EJTAG Memory Range	230
<b>10.0</b>	<b>Signal Descriptions</b>	<b>243</b>
<b>11.0</b>	<b>Electrical and Thermal Specifications</b>	<b>261</b>
11.1	Absolute Maximum Ratings	261
11.2	Thermal Characteristics	262
11.3	DC Parameters	263
11.4	AC Parameters	267
11.5	Power-up and Reset Timing	279
11.6	Asynchronous Signals	282
11.7	External Clock Specifications	282
11.8	Crystal Specifications	283
11.9	System Design Considerations	284
<b>12.0</b>	<b>Packaging, Pin Assignment and Ordering Information</b>	<b>285</b>
12.1	Mechanical Package	286
12.2	Pin Assignments	288
12.3	Ordering Information	301
<b>Appendix A</b>	<b>Support Documentation</b>	<b>303</b>
A.1	Memory Map	303
A.2	Differences between Au1100™ and Au1000™ Processors	310
A.3	Data Book Notations	311
A.4	Data Book Revision History	312

# List of Figures

Figure 1-1.	Block Diagram . . . . .	11
Figure 2-1.	Au1 Core Diagram . . . . .	13
Figure 2-2.	Cache Organization . . . . .	15
Figure 2-3.	Au1 Write Buffer . . . . .	21
Figure 2-4.	SBUS Arbitration . . . . .	40
Figure 3-1.	SDRAM Typical Read Timing . . . . .	50
Figure 3-2.	SDRAM Typical Write Timing . . . . .	51
Figure 3-3.	SDRAM Refresh Timing . . . . .	51
Figure 3-4.	Static Memory Read Timing (Single Read Followed by Burst) . . . . .	62
Figure 3-5.	Static Memory Read EWAIT# Timing . . . . .	62
Figure 3-6.	Static Memory Write Timing . . . . .	63
Figure 3-7.	Static Memory Write EWAIT# Timing . . . . .	63
Figure 3-8.	One Card PCMCIA Interface . . . . .	65
Figure 3-9.	Two Card PCMCIA Interface . . . . .	66
Figure 3-10.	PCMCIA Memory Read Timing . . . . .	67
Figure 3-11.	PCMCIA Memory Read PWAIT# Timing . . . . .	67
Figure 3-12.	PCMCIA Memory Write Timing . . . . .	68
Figure 3-13.	PCMCIA Memory Write PWAIT# Timing . . . . .	68
Figure 3-14.	PCMCIA I/O Read Timing . . . . .	68
Figure 3-15.	PCMCIA I/O Read PWAIT# Timing . . . . .	69
Figure 3-16.	PCMCIA I/O Write Timing . . . . .	69
Figure 3-17.	PCMCIA I/O Write PWAIT# Timing . . . . .	69
Figure 3-18.	LCD Controller Timing . . . . .	71
Figure 3-19.	LCD Read LWAIT# Timing . . . . .	71
Figure 3-20.	LCD Write LWAIT# Timing . . . . .	71
Figure 3-21.	16-Bit Chip Select Little-Endian Data Format (Default) . . . . .	72
Figure 3-22.	Big-Endian Au1 Core and Little-Endian 16-Bit Chip Select . . . . .	73
Figure 3-23.	Big-Endian Au1 Core and Big-Endian 16-Bit Chip Select . . . . .	73
Figure 5-1.	Interrupt Controller Logic . . . . .	85
Figure 6-1.	Endpoint Configuration Data Structure . . . . .	99
Figure 6-2.	Transmit Ring Buffer Entry Format . . . . .	120
Figure 6-3.	Receive Ring Buffer Entry Format . . . . .	122
Figure 6-4.	Typical Write Transaction Timing . . . . .	160
Figure 6-5.	Typical Read Transaction Timing . . . . .	160
Figure 6-6.	STN (Passive Mode) Timing . . . . .	176
Figure 6-7.	TFT (Active Mode) Timing . . . . .	177
Figure 6-8.	Logic for Interrupt Source Number 29 on Interrupt Controller 0 . . . . .	191
Figure 7-1.	Clocking Topology . . . . .	194
Figure 7-2.	Frequency Generator and Clock Source Block Diagram . . . . .	196
Figure 7-3.	Frequency Generator and Clock Source Mapping . . . . .	197
Figure 7-4.	TOY and RTC Block Diagram . . . . .	204
Figure 7-5.	GPIO Logic Diagram . . . . .	211
Figure 7-6.	Sleep and Idle Flow Diagram . . . . .	214
Figure 7-7.	Sleep Sequence . . . . .	217
Figure 8-1.	Power-up Sequence . . . . .	223

Figure 8-2.	Hardware Reset Sequence	224
Figure 8-3.	Runtime Reset Sequence	224
Figure 10-1.	Au1100™ Processor External Signals	244
Figure 11-1.	Voltage Undershoot Tolerances for Input and I/O Pins	262
Figure 11-2.	Voltage Overshoot Tolerances for Input and I/O Pins	262
Figure 11-3.	SDRAM Timing	268
Figure 11-4.	Static RAM, I/O Device and Flash Timing (Asynchronous Mode)	269
Figure 11-5.	PCMCIA Host Adapter Timing	270
Figure 11-6.	Static RAM, I/O Device and Flash Timing (Synchronous Mode)	271
Figure 11-7.	Wait Signal Recognition Timing for the Synchronous Static Bus	271
Figure 11-8.	GPIO Interrupt Timing	272
Figure 11-9.	Ethernet MII Timing Diagram	273
Figure 11-10.	I <sup>2</sup> S Timing Diagram	274
Figure 11-11.	AC-Link Timing Diagram	275
Figure 11-12.	SSI Timing Diagram	276
Figure 11-13.	LCD Controller Timing Diagram	277
Figure 11-14.	EJTAG Timing Diagram	278
Figure 11-15.	Power-up Sequence	279
Figure 11-16.	Hardware Reset Sequence	280
Figure 11-17.	Runtime Reset Sequence	281
Figure 12-1.	Package Dimensions	286
Figure 12-2.	Connection Diagram — Top View	288
Figure 12-3.	OPN Example	301

# List of Tables

Table 2-1.	Cache Line Allocation Behavior	16
Table 2-2.	Cache Operations	17
Table 2-3.	Cache Coherency Attributes (CCA)	17
Table 2-4.	Values for Page Size and PageMask Register	24
Table 2-5.	Cause[ExcCode] Encodings	25
Table 2-6.	CPU Interrupt Sources	26
Table 2-7.	Coprocessor 0 Register Definitions	28
Table 3-1.	Memory Controller Block Base Address	43
Table 3-2.	SDRAM Configuration Registers	45
Table 3-3.	SDRAM Signals	52
Table 3-4.	Static Bus Controller Configuration Registers	53
Table 3-5.	Device Type Encoding	56
Table 3-6.	Burst Size Mapping	56
Table 3-7.	Actual Number of Clocks for Timing Parameters (Except Tcsh)	57
Table 3-8.	Actual Number of Clocks for Tcsh	57
Table 3-9.	Static RAM, I/O Device and Flash Control Signals	61
Table 3-10.	PCMCIA Memory Mapping	64
Table 3-11.	PCMCIA Interface Signals	64
Table 3-12.	LCD Controller Interface Signals	70
Table 4-1.	DMA Channel Base Addresses	75
Table 4-2.	DMA Channel Configuration Registers	76
Table 4-3.	Peripheral Addresses and Selectors	76
Table 5-1.	Interrupt Controller Connections to the CPU	83
Table 5-2.	Interrupt Sources	84
Table 5-3.	Interrupt Controller Base Addresses	85
Table 5-4.	Interrupt Controller Registers	86
Table 5-5.	Interrupt Configuration Register Function	88
Table 6-1.	AC97 Base Address	89
Table 6-2.	AC97 Registers	89
Table 6-3.	AC-Link Signals	93
Table 6-4.	USB Host Base Address	95
Table 6-5.	USB Host Signals	96
Table 6-6.	USB Device Base Address	97
Table 6-7.	USB Device Register Block	97
Table 6-8.	Endpoint Configuration Field Descriptions	100
Table 6-9.	Example Endpoint Configuration Data Block	101
Table 6-10.	USB Device Signals	108
Table 6-11.	IrDA Modes Supported	109
Table 6-12.	IrDA Base Address	109
Table 6-13.	IrDA Registers	109
Table 6-14.	IrDA Hardware Connections	117
Table 6-15.	IrDA PHY Configuration Table	118
Table 6-16.	Fast Infrared Mode (FIR)	118
Table 6-17.	Medium Infrared Mode (MIR)	119
Table 6-18.	Slow Infrared Mode (SIR)	120

Table 6-19.	Transmit Ring Buffer Entry Format Description . . . . .	121
Table 6-20.	Receive Ring Buffer Entry Format Description . . . . .	122
Table 6-21.	Ethernet Base Addresses . . . . .	125
Table 6-22.	MAC Register Descriptions . . . . .	125
Table 6-23.	MAC DMA Entries . . . . .	135
Table 6-24.	MAC DMA Receive Entry Registers . . . . .	135
Table 6-25.	MAC DMA Transmit Entry Registers . . . . .	135
Table 6-26.	MAC DMA Block Indexed Address Bit Definitions . . . . .	136
Table 6-27.	Ethernet Signals . . . . .	144
Table 6-28.	I <sup>2</sup> S Base Address . . . . .	146
Table 6-29.	I <sup>2</sup> S Interface Register Block . . . . .	146
Table 6-30.	I <sup>2</sup> S Signals . . . . .	149
Table 6-31.	UART Register Base Addresses . . . . .	151
Table 6-32.	UART Registers . . . . .	151
Table 6-33.	Interrupt Cause Encoding . . . . .	153
Table 6-34.	UART Signals . . . . .	159
Table 6-35.	SSI Base Addresses . . . . .	161
Table 6-36.	SSI Registers . . . . .	161
Table 6-37.	SSI Signals . . . . .	165
Table 6-38.	LCD Base Address . . . . .	167
Table 6-39.	LCD Controller Registers . . . . .	167
Table 6-40.	Pixel Ordering . . . . .	169
Table 6-41.	LCD Controller Signals . . . . .	175
Table 6-42.	LCD Controller Data Pin Usage . . . . .	175
Table 6-43.	SD Base Address . . . . .	178
Table 6-44.	SD Registers . . . . .	178
Table 6-45.	Command Type Field Encodings . . . . .	184
Table 6-46.	SD Signals . . . . .	188
Table 6-47.	GPIO2 Register Base Addresses . . . . .	189
Table 6-48.	GPIO2 Registers . . . . .	189
Table 7-1.	System Control Block Base Address . . . . .	193
Table 7-2.	Clock Generation Registers . . . . .	195
Table 7-3.	Clock Mux Input Select Values . . . . .	201
Table 7-4.	Programmable Counter Registers . . . . .	205
Table 7-5.	GPIO Control Registers . . . . .	211
Table 7-6.	Peripheral Power Management . . . . .	215
Table 7-7.	Power Management Registers . . . . .	218
Table 8-1.	ROMSEL and ROMSIZE Boot Device . . . . .	225
Table 9-1.	Coprocessor 0 Registers for EJTAG . . . . .	228
Table 9-2.	EJTAG Memory Mapped Registers at 0xFF300000 . . . . .	231
Table 9-3.	EJTAG Instruction Register Values . . . . .	236
Table 9-4.	EJTAG Signals . . . . .	242
Table 10-1.	Signal Type Abbreviations for Table 10-3 . . . . .	245
Table 10-2.	Signal State Abbreviations for Table 10-3 . . . . .	245
Table 10-3.	Signal Descriptions . . . . .	246
Table 11-1.	Absolute Maximum Ratings . . . . .	261
Table 11-2.	Thermal Characteristics . . . . .	262
Table 11-3.	DC Parameters . . . . .	263
Table 11-4.	Voltage and Power Parameters for 333 MHz Part . . . . .	264
Table 11-5.	Voltage and Power Parameters for 400 MHz Part . . . . .	265
Table 11-6.	Voltage and Power Parameters for 500 MHz Part . . . . .	266
Table 11-7.	SDRAM Controller Interface . . . . .	268
Table 11-8.	Static RAM, I/O Device and Flash Timing (Asynchronous Mode) . . . . .	269
Table 11-9.	PCMCIA Timing . . . . .	270
Table 11-10.	Synchronous Static Bus . . . . .	271



Table 11-11.	GPIO Timing for Interrupts . . . . .	272
Table 11-12.	Ethernet MII Timing . . . . .	273
Table 11-13.	I <sup>2</sup> S Interface Timing . . . . .	274
Table 11-14.	AC-Link Interface Timing . . . . .	275
Table 11-15.	Synchronous Serial Interface Timing . . . . .	276
Table 11-16.	LCD Controller Timing . . . . .	277
Table 11-17.	EJTAG Interface Timing . . . . .	278
Table 11-18.	Power-up Timing Parameters . . . . .	279
Table 11-19.	Hardware Reset Timing Parameters . . . . .	280
Table 11-20.	Runtime Reset Timing Parameters . . . . .	281
Table 11-21.	External Clock EXTCLK[1:0] Specifications . . . . .	282
Table 11-22.	12 MHz Crystal Specification . . . . .	283
Table 11-23.	32.768 kHz Crystal Specification . . . . .	283
Table 12-1.	Pin Assignment — Sorted by Pin Number . . . . .	290
Table 12-2.	Pin Assignment — Sorted Alphabetically by Default Signal . . . . .	296
Table 12-3.	Pin Assignment — Alternate Signals Sorted Alphabetically . . . . .	300
Table 12-4.	Valid OPN Combinations . . . . .	301
Table A-1.	Basic Au1100™ Processor Physical Memory Map . . . . .	303
Table A-2.	System Bus Devices Physical Memory Map . . . . .	303
Table A-3.	Peripheral Bus Devices Physical Memory Map . . . . .	304
Table A-4.	Device Memory Map . . . . .	305
Table A-5.	Revision History . . . . .	312
Table A-6.	Edits to Current Revision . . . . .	312



# 1 Overview

The AMD Alchemy™ Au1100™ processor, a follow-on to the Au1000™ processor, is a high performance, low power, high integration system-on-a-chip (SOC) with the inclusion of a LCD controller and further reduction in power. The Au1100 is designed for use in mobile information appliances (IAs). These IAs include WebPADs, telematics, PDAs and multimedia handheld computing devices. Figure 1-1 is a block diagram of the Au1100 processor.

## 1.1 Product Description

The Au1100™ processor is a complete SOC based on the MIPS32™ instruction set. Designed for maximum performance at low power, the processor runs up to 500 MHz. Power dissipation is less than 0.25 watt for the 400-MHz version. Highly integrated with on-chip SDRAM, SRAM/Flash EPROM memory controllers, a LCD controller, 10/100 Ethernet Controller, USB 1.1 Host and Device, UARTs (3), and GPIOs (up to 48, 13 dedicated). The Au1100 processor runs a variety of operating systems, including Windows® CE.NET, Linux and VxWorks. Moreover, the integration of peripherals with the unique, high performance, MIPS-compatible core provides low system cost, small form factor, low system power requirement, simple designs at multiple performance points and thus, short design cycles.

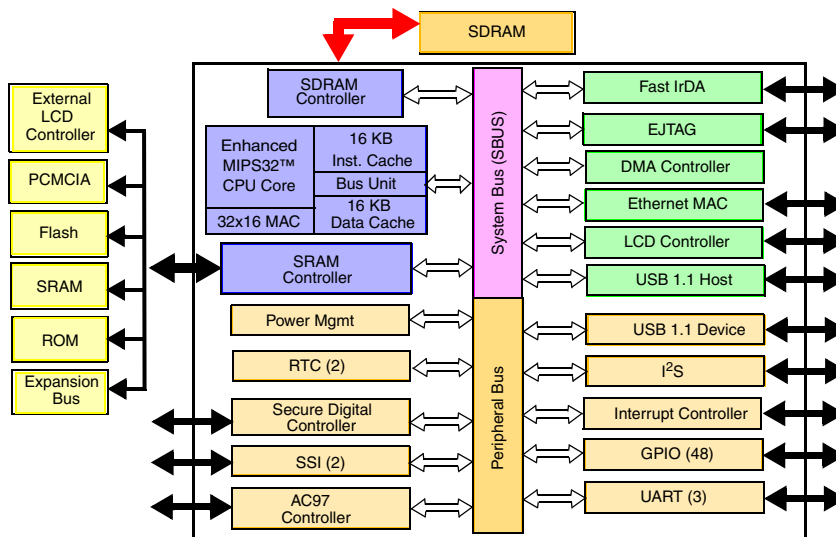


Figure 1-1. Block Diagram

## 1.2 Features

### High Speed MIPS CPU Core

- 333, 400, or 500 MHz
- MIPS32 instruction set 32-bit architecture
- 16KB instruction and 16KB data caches
- High speed multiply-accumulate (MAC) and divide unit
- 1.1-1.3V core
- 3.3V or 2.5- SDRAM I/O, 3.3V I/O

### Highly-Integrated System Peripherals

- GPIO (48 total, 13 dedicated for system use)
- 10/100 Ethernet MAC controller
- USB 1.1 device and host controllers
- Three UARTs
- IrDA controller
- AC97 controller
- I<sup>2</sup>S controller
- Two SSI controllers
- Two Secure Digital (SD) controllers
- LCD controller
- PCMCIA interface

### High-Bandwidth Memory Buses

- 100 MHz SDRAM controller (@400 MHz)
- SRAM/Flash EPROM controller

### Caches

- 16 KB non-blocking data cache
- 16 KB instruction cache
- Instruction and data caches are 4-way set associative
- Write-back with read-allocate
- Cache management features:
  - Programmable allocation policy
  - Line locking
- Prefetch instructions (instruction and data)
- High speed access to on-chip buses

### Core MicroArchitecture Highlights

- Pipeline
  - Scalar 5-stage pipeline
  - Load/store adder in I-stage (instr decode)
  - Scalar branch techniques optimized: Pipelined register file access in fetch stage
  - Zero penalty branch
- Multiply-Accumulate (MAC) and Divide Unit
  - Max issue rate of one 32x16 MAC per clock
  - Max issue rate of one 32x32 MAC per every other clock
  - Operates in parallel to CPU pipeline
  - Executes all integer multiply and divide instructions
  - 32 x 16-bit MAC hardware

### MMU

- Instruction and data watch registers for software breakpoints
- Separate interrupt exception vector
- TLB Features:
  - 32 dual-entry fully-associative
  - Variable page sizes: 4 KB to 16 MB
  - 4-entry ITB

### Low System Power

- Core / Power:
  - 333 MHz / < 200 mW
  - 400 MHz / 250 mW
  - 500 MHz / 400 mW
- Power Saving Modes:
  - Idle
  - Sleep
- Pseudo-static design to 0 Hz

### Package

- 399 BGA (Ball Grid Array), 17 x17 mm

### Operating System Support

- Microsoft® Windows® CE
- Linux
- VxWorks

### Development Tool Support

- Complete MIPS32™ Compatible Tool Set
- Numerous third party compilers, assemblers and debuggers

# 2 CPU

The Au1100 CPU core is a unique implementation of the MIPS32 instruction set architecture (ISA) designed for high frequency and low power. This chapter provides information on the implementation details of this MIPS32 compliant core.

The full description of the MIPS32 architecture is provided in the “MIPS32™ Architecture For Programmers” documentation, available from MIPS Technologies, Inc. The information contained in this chapter supplements the MIPS32 architecture documentation.

## 2.1 Core

The Au1100 CPU core (Au1) is a high performance, low power implementation of the MIPS32 architecture. Figure 2-1 shows a block diagram of the core.

The Au1 core contains a five-stage pipeline. All stages complete in a single cycle when data is present. All pipeline hazards and dependencies are enforced by hardware interlocks so that any sequence of instructions is guaranteed to execute correctly. Therefore, it is not necessary to pad legacy MIPS hazards (such as load delay slots and coprocessor accesses) with NOPs.

The general purpose register file has two read ports and one write port. The write port is shared with data cache loads and the pipeline Writeback stage.

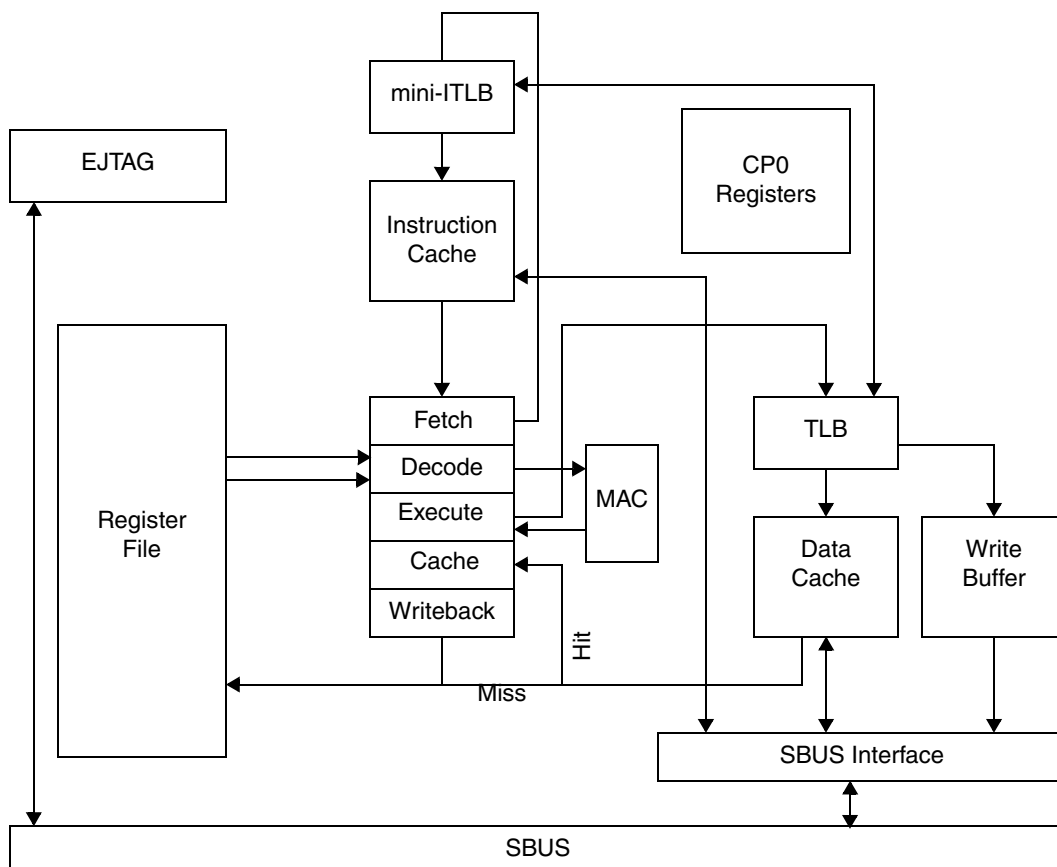


Figure 2-1. Au1 Core Diagram

### 2.1.1 Fetch Stage

The Fetch stage retrieves the next instruction from the instruction cache, where it is passed to the Decode stage. If the instruction is not present in the instruction cache, then the fetch address is forwarded to the virtual memory unit in order to fulfill the request. Instruction fetch stalls until the next instruction is available.

### 2.1.2 Decode Stage

The Decode stage prepares the pipeline for executing the instruction. In the Decode stage, the following occur in parallel:

- The instruction is decoded.
- Control for the instruction is generated.
- Register data is read.
- The branch target address is generated.
- The load/store address is generated.

Instructions stall in the Decode stage if dependent data or resources are not yet available. At the end of the Decode stage a new program counter value is sent to the Fetch stage for the next instruction fetch cycle.

### 2.1.3 Execute Stage

In the Execute stage, instructions that do not access memory are processed in hardware (shifters, adders, logical, comparators, etc.). Most instructions complete in a single cycle, but a few require multiple cycles (**CLO**, **CLZ**, **MUL**).

The virtual address calculation begins in the Decode stage so that physical address calculation can complete in the Execute stage, in time to initiate the access to the data cache in the Execute stage. If the physical address misses in the TLB, a TLB exception is posted.

Multiplies and divides are forwarded to the Multiply Accumulate unit. These instructions require multiple cycles to execute and operate mostly independent of the main five-stage pipeline.

All exception conditions (arithmetic, TLB, interrupt, etc.) are posted by the end of the Execute stage so that exceptions can be signalled in the Cache stage.

### 2.1.4 Cache Stage

In the Cache stage, load and store accesses complete.

Loads that hit in the data cache obtain the data in the Cache stage. If a load misses in the data cache, or is to a non-cacheable location, then the request is sent to the System Bus (SBUS) to be fulfilled. Load data is forwarded to dependent instructions in the pipeline.

Stores that hit in the data cache are written into the cache array. If a store misses in the data cache, or is to a non-cacheable location, then the store is sent to the write buffer.

If any exceptions are posted, an exception is signaled and the Au1 core is directed to fetch instructions at the appropriate exception vector address.

### 2.1.5 Writeback Stage

In the Writeback stage, results are posted to the general purpose register file, and forwarded to other stages as needed.

### 2.1.6 Multiply Accumulate Unit

The Multiply Accumulate unit (MAC) executes all multiply and divide instructions. The MAC is composed of a 32x16 bit pipelined array multiplier that supports early out detection, divide block, and the HI and LO registers used in calculations.

The MAC operates in parallel with the main five-stage pipeline. Instructions in the main pipeline that do not have dependencies on the MAC calculations execute simultaneously with instructions in the MAC unit.

A multiply calculation of 16x16 or 32x16 bits can complete in one cycle. The 32x16 bit multiply must have the sign-extended 16-bit value in register operand rt of the instruction.

32x32 bit multiplies may be started every other CPU cycle. The 32x32 multiplies will complete in two cycles if the results are written to the general purpose registers.

If the results are written to the HI/LO registers then three cycles are required for 16x16 and 32x16 bits multiplies. 32x32 bit multiplies that use HI/LO will complete in 4 cycles.

Divide instructions complete in a maximum of 35 cycles.

## 2.2 Caches

The Au1 core contains independent, on-chip 16KB instruction and data caches. As shown in Figure 2-2, each cache contains 128 sets and is four-way set associative with 32 bytes per way (cache line).

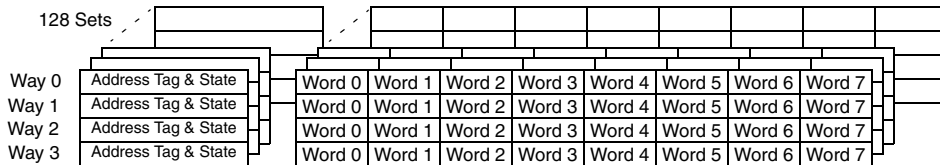
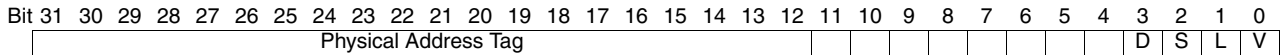


Figure 2-2. Cache Organization

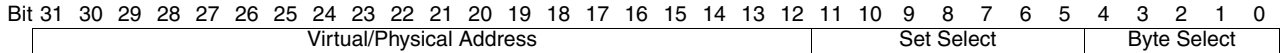
A cache line is tagged with a 20-bit physical address, a lock bit, and a valid bit. Data cache lines also include coherency and dirty status bits. The physical address tag contains bits [31:12] of the physical address; as such, physical addresses in which bits [35:32] are non-zero must be mapped non-cacheable.

### Cache Line State



A cache line address is always 32-byte aligned. The cache is indexed with the lower, untranslated bits (bits [11:5]) of the virtual address, allowing the virtual-to-physical address translation and the cache access to occur in parallel.

### Cache Address Decode



#### 2.2.1 Cache Line Replacement Policy

In general, the caches implement a least recently used (LRU) replacement policy. Each cache set maintains true LRU status bits (MRU, nMRU and LRU) to determine which cache line is selected for replacement. However, software can influence which cache line is replaced by marking memory pages as *streaming*, or by *locking* lines in the cache.

#### 2.2.2 Cache Line Locking Support

The **CACHE** instruction is used to lock individual lines in the cache. A locked line is not subject to replacement. All four lines in a set can not be locked at once; at least one line is always available for replacement. To *unlock* individual cache lines use the **CACHE** instruction with a 'hit invalidate' command opcode. See Section 2.2.5 "Cache Management" on page 17 for further discussion of the **CACHE** instruction.

#### 2.2.3 Cache Streaming Support

Streaming is typically characterized as the processing of a large amount of transient instructions and/or data. In traditional cache implementations (without explicit support for streaming), transient instructions and/or data quickly displace useful, recently used items in the cache. This yields poor utilization of the cache and results in poor system performance.

The Au1 caches explicitly support streaming by placing instructions and/or data marked as streaming into way 0 of the cache. This method ensures that streaming does not purge the cache(s) of useful, recently used items, while permitting transient instructions and/or data to be cached. The CCA bits in the TLB entry indicate if a page contains streaming instructions and/or data. In addition, the **PREF** instruction is available to software to allow data to be placed in the data cache in advance of its use.

## 2.2.4 Cache Line Allocation Behavior

When an instruction fetch misses in the instruction cache, or a data load misses in the data cache, a burst fill operation is performed to fill the cache line from memory. The cache line is selected by the following algorithm:

```
MRU is most recently used
nMRU is next most recently used
nLRU is next least recently used
LRU is least recently used
```

Cache Miss:

```
if (Streaming CCA=6) then Replacement = 0,
else if (LRU is !Valid or !Locked) then Replacement = LRU
else if (nLRU is !Valid or !Locked) then Replacement = nLRU
else if (nMRU is !Valid or !Locked) then Replacement = nMRU
else Replacement = MRU
```

Cache Hit:

```
new MRU = Hit Way
```

In short, the LRU selection is true LRU but with the following priorities:

- 1) Streaming: cache misses are forced to way 0.
- 2) Locking: cache misses follow policy above and set Lock bit.
- 3) Normal: true LRU replacement.

Table 2-1 summarizes cache line allocation for misses, as well as cache hit behavior. The table also shows how prefetching and cache locking affect the cache for hits and misses.

**Table 2-1. Cache Line Allocation Behavior**

Operation	Hit	Miss
<b>NORMAL</b>		
Data load, Instruction fetch	Read data from whichever cache line contains the address.	Allocate and fill cache line; clear Lock bit; return read data.
Data store	Write data to whichever cache line contains the address.	Send to the write buffer.
<b>STREAMING (CCA=6)</b>		
Data load, Instruction fetch	Read data from whichever cache line contains the address.	Allocate and fill cache line in Way 0; maintain Lock bit; return read data.
Data store	Write data to whichever cache line contains the address.	Send to the write buffer.
PREF (data prefetch instruction with 0x4 hint)	No action taken—data remains in current cache line.	Allocate and fill cache line in Way 0; maintain Lock bit.
<b>LOCKING</b>		
CACHE 0x1D/0x1C (cache management instruction with Lock opcode)	Set Lock bit in whichever cache line contains the address.	Allocate and fill cache line; set Lock bit.



### 2.2.5 Cache Management

The caches are managed with the **CACHE** instruction. The effect of the **CACHE** instruction is immediately visible to subsequent data accesses. Table 2-2 shows the cache operations, including the opcode to direct the operation to either the instruction cache or data cache. (An “n/a” indicates that the operation is not applicable.)

**Table 2-2. Cache Operations**

Operation	CACHE[20..18] Encoding	Opcode for Instruction Cache	Opcode for Data Cache
Index Invalidate	000	0x00	0x01 (with writeback)
Index Load Tag	001	0x04	0x05
Index Store Tag	010	0x08	0x09
Hit Invalidate	100	0x10	0x11
Fill	101	0x14	N/A
Hit Writeback and Invalidate	101	N/A	0x15
Hit Writeback	110	N/A	0x19
Fetch and Lock	111	0x1C	0x1D

These cache operations permit initialization, locking/unlocking and management of the caches.

### 2.2.6 Cache Coherency Attributes (CCA)

The cache coherency attributes (CCA) field in Config0[K0] and in the TLB determine the cache-ability of accesses to memory. Cached accesses are performed critical-word-first to improve performance. The Au1 implements the following:

**Table 2-3. Cache Coherency Attributes (CCA)**

CCA	CCA (3 Bits)	Description
0, 1	00x	Reserved (undefined).
2	010	Uncached, non-mergeable, non-gatherable. Required by the MIPS32 architecture. In addition, data is not merged within the write buffer to achieve a truly uncached effect.
3	011	Cached, mergeable, gatherable.
4	100	Reserved (undefined).
5	101	Cached, mergeable, gatherable.
6	110	Cached, mergeable, gatherable, streaming. Instructions and/or data are placed into way 0.
7	111	Uncached, mergeable, gatherable. Even though data is not cached, data stores sent to the write buffer are subject to merging and gathering in the write buffer.

## 2.2.7 Instruction Cache

The instruction cache is a 16-KB, four-way set associative cache. The instruction cache services instruction fetch requests from the Fetch stage of the pipeline.

An instruction cache line state consists of a 20-bit physical address tag, a lock bit (L) and a valid bit (V).

### Instruction Cache line state

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
Physical Address Tag																																																		L	V

### 2.2.7.1 Instruction Cache Initialization and Invalidation

Out of reset, all instruction cache lines are invalidated; thus the instruction cache is ready for use.

To invalidate the instruction cache in software, a loop of index invalidate **CACHE** instructions for each of the lines in the cache invalidates the cache.

```

li t0,(16*1024) # Cache size
li t1,32 # Line size
li t2,0x80000000 # First KSEG0 address
addu t3,t0,t2 # terminate address of loop
loop:
  cache 0,0(t2) # Icache indexed invalidate tag
  addu t2,t1 # compute next address
  bne t2,t3,loop
  nop

```

### 2.2.7.2 Instruction Cache Line Fills

If an instruction fetch address hits in the instruction cache, the instruction word is returned to the Fetch stage. If the fetch address misses in the cache, and the address is cacheable, then the instruction cache performs a burst transfer from the memory subsystem to fill a cache line, and returns the instruction word to the Fetch stage.

The instruction cache line is selected by the replacement policy described in Section 2.2.1 "Cache Line Replacement Policy" on page 15.

### 2.2.7.3 Instruction Cache Coherency

The instruction cache does not maintain coherency with the data cache. Coherency between the instruction cache and the data cache is the responsibility of software. However, the data cache snoops during instruction cache line fills.

Maintaining coherency is important when loading programs into memory, creating exception vector tables, or for self-modifying code. In these circumstances, memory is updated with new instructions using store instructions which places the new instructions in the data cache, but not in the instruction cache (thus the instruction cache may contain old instructions).

To maintain coherency, software must use the **CACHE** instruction to invalidate the modified range of program addresses in the instruction cache. Because the data cache snoops during instruction cache line fills, it is not necessary to writeback the data cache prior to invalidating the instruction cache. An instruction fetch to the newly loaded/modified program correctly fetches the new instructions.

### 2.2.7.4 Instruction Cache Control

The cache-ability of instructions is controlled by three mechanisms:

- Config0[K0] field
- The CCA bits in the TLB
- The **CACHE** instruction

The Config0[K0] field contains a cache coherency attribute (CCA) setting to control the cache-ability of KSEG0 region. At reset, this field defaults to CCA=3 (cacheable).

The CCA bits in the TLB entry control the cache-ability of the KUSEG, KSEG2, and KSEG3 regions. Each TLB entry specifies a CCA setting for the pages mapped by the TLB.

The **CACHE** instruction manages the caches, including the ability to lock lines in the cache. Valid instruction cache operations are the following:

- Index Invalidate
- Index Load Tag
- Index Store Tag
- Hit Invalidate
- Fill
- Fetch and Lock

The effect of the **CACHE** instruction is visible to subsequent instructions not already in the pipeline. Instructions already in the fetch and decode stages of the pipeline are not affected by a cache operation on the instruction cache.

## 2.2.8 Data Cache

The data cache is a 16KB four-way set associative write-back cache. Data cache accesses are distributed across the Execute and Cache pipeline stages.

A data cache line state consists the 20-bit physical address tag, a dirty bit (D), a coherency bit (S), a lock bit (L) and a valid bit (V).

### Data cache line state

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																									
Physical Address Tag																																																					D	S	L	V

The data cache employs a read-allocate policy. Cache lines can be replaced on loads, but not on stores. Stores that miss in the data cache are forwarded to the write buffer.

The data cache supports hit-under-miss for one outstanding miss. If an access misses in the data cache, the data cache services the next access while the memory subsystem provides the data for the missed access. If the next access hits in the data cache, the data is available immediately; otherwise the cache stalls the access until the first access completes.

### 2.2.8.1 Data Cache Initialization and Invalidation

Out of reset, all data cache lines are invalidated; thus the data cache is ready for use.

To invalidate the data cache in software, a loop of indexed writeback invalidate **CACHE** instructions for each of the lines in the cache invalidates the cache.

```

li t0,(16*1024) # Cache size
li t1,32 # Line size
li t2,0x80000000 # First KSEG0 address
addu t3,t0,t2 # terminate address of loop
loop:
cache 1,0(t2) # Dcache indexed invalidate tag
addu t2,t1 # compute next address
bne t2,t3,loop
nop

```

### 2.2.8.2 Data Cache Line Fills

A data cache access is initiated in the Execute stage which allows a cache hit or miss indication and all exceptions to be signaled early in the Cache stage. If the data address hits in the data cache, the data is available in the Cache stage. If the data address misses in the data cache, and the address is cacheable, the data cache performs a burst fill to a cache line, forwarding the critical word to the Cache stage.

The data cache line is selected by the replacement policy described in Section 2.2.1 on page 15. If the line selected contains modified data (cache line is valid and has its dirty bit set by a store hit), then the cache line is moved to a cast-out buffer, the cache line is filled from memory and the load request fulfilled, and then the cast-out buffer is written to memory.

### 2.2.8.3 Data Cache Coherency

The data cache snoops coherent SBUS transactions to maintain data coherency with other SBUS masters (i.e. DMA). If a coherent read transaction on the SBUS hits in the data cache, the data cache provides the data. If a coherent write transaction on the SBUS hits in the data cache, the data cache updates its internal array with the data. If a coherent transaction (read or write) misses in the data cache, the data cache array is unchanged by the transaction.

Loads and stores which hit in the data cache can bypass previous stores in cacheable regions. The read-allocate data cache policy forwards store-misses to the write buffer. Subsequent loads and stores which hit in the data cache, and to a different cache line address than store-misses, are fulfilled immediately (while store-misses may still be in the write buffer). However, if a load address hits in a cache-line address of an item in the write buffer, the load is stalled until the write buffer commits the corresponding store.

The data cache also maintains coherency with other caching masters. When a load is serviced from another caching master, both caching masters set the shared bit for the affected cache line. Then if a store occurs to a data cache line with the shared bit set, the cache line address is broadcast on the SBUS to invalidate cache lines in other caching masters that contain the same address.

The data cache is single-ported; therefore transactions on the SBUS are prioritized over accesses by the core. However, the data cache design prevents the SBUS from saturating the data cache indefinitely, which ensures that the core can make forward progress.

When changing the CCA encoding in Config0[K0] or the TLB to a different CCA encoding, software must ensure that data integrity is not compromised by first pushing modified (dirty) data to memory within the page. This is especially important when changing from a coherent CCA encoding to a non-coherent CCA encoding.

### 2.2.8.4 Data Cache Control

The cache-ability of data accesses is controlled by four mechanisms:

- Config0[K0] field
- The CCA bits in the TLB
- The **CACHE** instruction
- The **PREF** instruction

The Config0[K0] field contains a cache coherency attribute (CCA) setting to control the cache-ability of KSEG0 region. At reset, this field defaults to 011b, cacheable.

The CCA bits in the TLB entry control the cache-ability of the KUSEG, KSEG2, and KSEG3 regions. Each TLB entry specifies a CCA setting for the pages mapped by the TLB.

The **CACHE** instruction manages the caches; including the ability to lock lines in the cache. Valid data cache operations are:

- Index Writeback Invalidate
- Index Load Tag
- Index Store Tag
- Hit Invalidate (unlocks)
- Hit Writeback and Invalidate
- Hit Writeback
- Fetch and Lock

The effect of the **CACHE** instruction is immediately visible to subsequent data accesses.

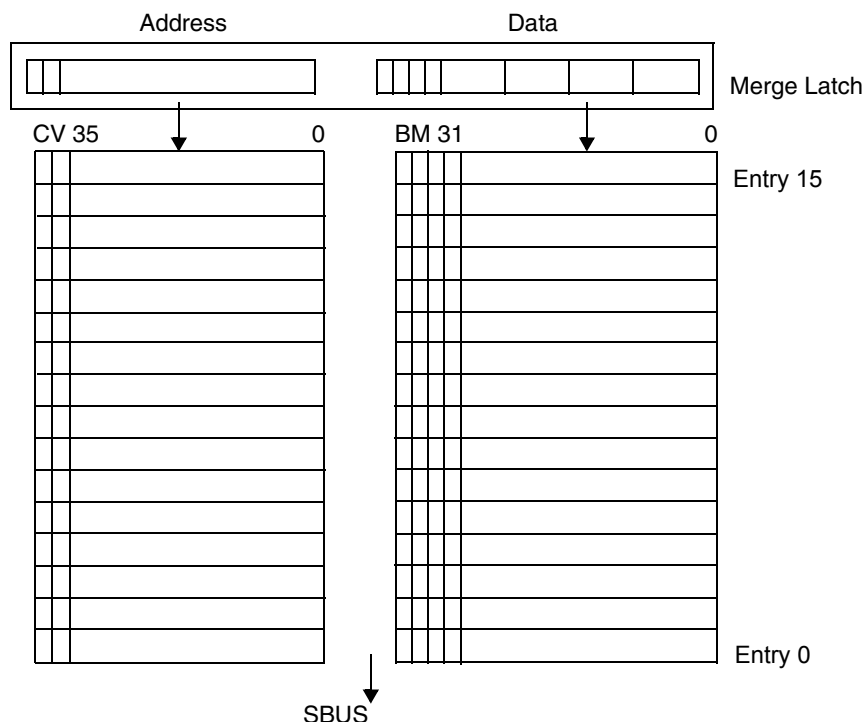
The **PREF** instruction places data into the data cache. The following prefetch hints are implemented:

- 0x00 - Normal load
- 0x04 - Streaming load

The streaming load hint directs the data be placed into way 0 of the data cache (even if the line is locked), thus permitting transient data to be cached and non-transient data to remain in the cache for improved performance. Data cache streaming support combined with the **PREF** instruction enhances multimedia processing.

## 2.3 Write Buffer

The Au1 write buffer is depicted in Figure 2-3. All non-cacheable processor stores and data cache store-misses (the data cache is a read-allocate policy) are routed through the write buffer.



**Figure 2-3. Au1 Write Buffer**

The write buffer is a 16-word deep first-in-first-out (FIFO) queue. All processor stores arrive first at the merge latch, where merging and gathering decisions are performed, and then travel through the queue. The write buffer arbitrates for the SBUS to perform consolidated transfers to the main memory.

A write buffer FIFO entry contains the address (word address), the data and associated byte masks (BM), and two control bits. The four BM bits indicate which bytes within the word contain valid data. The two control bits are the Valid bit which indicates if the entry is valid, and the Closed (C) bit. When a C bit is set, the write buffer initiates a request to the SBUS so that it can transfer data to memory. The circumstances for which the C bit is set are described below.

The write buffer is capable of variable-length burst writes to memory. The length can vary from one word to eight words, and is determined by the C bits in the write buffer. During each beat of the burst, the appropriate bytes to write are selected from the corresponding byte masks. As each entry is written to memory, it is popped from the FIFO, advancing each entry in the FIFO by one. In other words, entry 0 is always presented to the SBUS for writing.

When the write buffer has at least one empty entry, processor stores do not stall, thus improving processor performance.

The write buffer is disabled by setting Config0[WD] to 1. In this instance, all non-cacheable and data cache store-misses stall until the write completes. The remaining description of the write buffer operation assumes Config0[WD] is 0. Out of reset, Config0[WD] is 0.

### 2.3.1 Merge Latch

All processor stores first arrive at the merge latch. Logic within the merge latch decides what action to take with the incoming data.

- 1) The incoming address is the *same word address* as the merge latch address. This case is for Merging, which occurs within the merge latch itself.
- 2) The incoming word address is *sequentially adjacent* to the merge latch word address (incoming address is merge latch address + 4). This case is for Gathering. The merge latch contents are propagated to the FIFO with the C bit cleared for this entry.
- 3) Neither 1 nor 2 is true. The merge latch contents are propagated to the FIFO with the C bit set for this entry.

If the merge latch contents are propagated to the FIFO, the incoming address and data are placed in the merge latch for future comparisons. Furthermore, if the incoming address is the last word address of the maximum burst line size (the least significant 5 bits are 0x1C), then the C bit is set.

### 2.3.2 Write Buffer Merging

Write buffer merging combines stores destined for the *same word address*. Merging places the incoming data into the appropriate data byte(s) within the merge latch.

Write buffer merging is particularly useful for sequential, incremental address write operations, such as string operations. With write buffer merging, the writes are merged into 32-bit writes which reduces the number of accesses to the memory and increases the effective throughput to main memory.

This example demonstrates merging: these five byte writes occur in sequence:

```
0x00001000 = 0xAB
0x00001001 = 0xCD
0x00001002 = 0xDE
0x00001003 = 0xEF
0x00001002 = 0xBE
```

After the first four writes, the data in the merge latch contains 0xABCDEEF. However, after the fifth write, the merge latch data now contains 0xABCDBEEF.

So long as the incoming word address is the same as the merge latch word address, the data can change without a processor stall or access to memory.

Write buffer merging is controlled by the Config[NM] bit and the TLB[CCA] setting. When Config0[NM] is 1 or TLB[CCA] is 2, the merge latch does not perform merging. Conversely, Config0[NM] is 0 or TLB[CCA] is not 2 enables merging. Out of reset, Config0[NM] is 0.

**Note:** Merging takes place *only* in the merge latch. As such, writes to an address which are in the FIFO (but not in the merge latch) do not merge. In the example below, writes to 0x00001000 and 0x00001002 do not merge because the intervening write to address 0x00001005 is not in the same word address which caused 0x00001000 to leave the merge latch.

```
0x00001000 = 0xAB
0x00001005 = 0xCD
0x00001002 = 0xDE
```

### 2.3.3 Write Buffer Gathering

Write buffer gathering combines *sequentially adjacent* word addresses for burst transfers to the main memory. When a C bit is set, all queue entries from zero (0) up to and including the entry with its C bit set (N) are written to main memory in a single burst.

Write buffer gathering is particularly useful for sequential, incremental address store operations, such as string operations. With write buffer gathering, the stores are combined into bursts up to 32-bytes (eight words) in length which reduces the number of accesses to the memory and increases effective throughput.

Here is an example of an eight-word burst. The burst could result from code which sequentially writes words (optimized memcopy(), for example). These eight word writes occur in sequence:

```
0x00001000
0x00001004
0x00001008
0x0000100C
0x00001010
0x00001014
0x00001018
0x0000101C
```

The entries corresponding to word addresses 0x00001000 through 0x00001018 have C bit set to zero. When address 0x0000101C arrives, its C bit is set. When the write buffer is granted the SBUS, it bursts all eight entries to main memory.

Here is an example of two-word burst. This burst may be typical of application software. These four word writes occur in sequence:

```
0x00001000
0x00001004
0x0000100C
0x00001008
```

The C bit is cleared for the 0x00001000 entry and is set for the 0x00001004 entry. These two words are then burst to main memory. The 0x0000100C entry also has its C bit set, and is written to memory. The 0x00001008 will reside in the merge latch until displaced by a subsequent store.

### 2.3.4 Write Buffer Reads

When a read from memory is initiated, the read cache-line address (A35..A5) is compared against all cache-line addresses in the write buffer. If the read cache-line address matches a write buffer cache-line address, the read is stalled. The write buffer then flushes entries to memory until the read address no longer matches a write buffer cache-line address. The read is then allowed to complete. The write buffer ensures data integrity by not allowing reads to bypass writes.

### 2.3.5 Write Buffer Coherency

Non-cacheable stores and/or data cache store-misses reside in the write buffer, possibly indefinitely. Furthermore, the write buffer does not snoop SBUS transactions (e.g., integrated peripheral DMA engines). To ensure the write buffer contents are committed to memory, a **SYNC** instruction must be issued.

Issuing a **SYNC** instruction prior to enabling each DMA transfer from memory buffers and/or structures is necessary. Without the **SYNC**, the DMA engine may retrieve incomplete buffers and/or structures (the remainder of which may be in the write buffer).

Issuing a **SYNC** instruction after a store to an I/O region where stores have side effects is necessary. Without the **SYNC** instruction, the store may not leave the write buffer to achieve the side effects (e.g. clearing an interrupt acknowledge bit).

Note that a read access does not guarantee a complete write buffer flush since the write buffer flushes as few entries as necessary until the read address no longer matches an address in the write buffer.

## 2.4 Virtual Memory

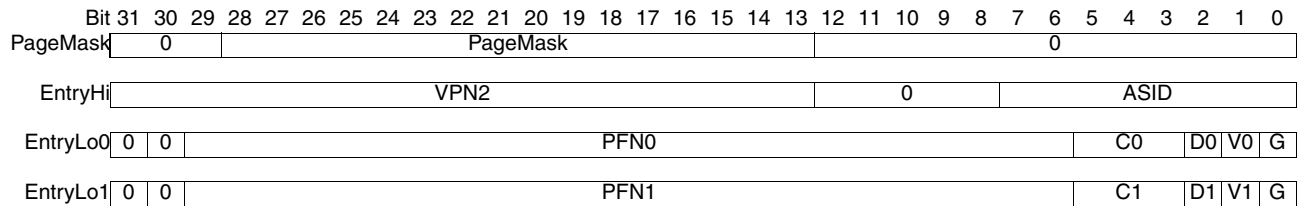
The Au1 implements a TLB-based virtual address translation unit which is compliant with the MIPS32 specification. This scheme is similar to the R4000 TLB and CP0 implementation. The “MIPS32 Architecture For Programmers Volume III” contains all the information relevant to a TLB-based virtual address translation unit.

The virtual address translation architecture is composed of a main 32-entry fully associative TLB array. To improve instruction fetch performance, a 4-entry fully associative instruction TLB is implemented. This miniature instruction TLB is fully coherent with the main TLB array and is completely transparent to software.

Each TLB entry maps a 32-bit virtual address to a pair of 36-bit physical addresses. The page size of a TLB entry is variable under software control, from 4 KB to 16 MB.

A TLB entry is described below.

### TLB Entry



The size of the page(s) that the TLB entry translates is determined by PageMask. The valid values for PageMask range from 4 KB to 16 MB, according to Table 2-4.

**Table 2-4. Values for Page Size and PageMask Register**

Page Size	PageMask Register	Bits [28:13]	PFN Select Bit
4 KB	0x00000000	0000000000000000	12
16 KB	0x00006000	0000000000000011	14
64 KB	0x0001E000	0000000000001111	16
256 KB	0x0007E000	0000000001111111	18
1 MB	0x001FE000	0000000011111111	20
4 MB	0x007FE000	0000001111111111	22
16 MB	0x01FFE000	0000111111111111	24

The PageMask determines the number of significant bits in the 32-bit address generated by the program (either as a load/store address or an instruction fetch address). The upper, significant bits of the program address are compared against the upper, significant bits of VPN2. When an address match occurs, the even/odd PFN select bit of the program address selects either PFN0 (even) or PFN1 (odd) as the upper bits of the resulting 36-bit physical address.

The TLB mechanism permits mapping a larger, 36-bit physical address space into the smaller 32-bit program address space. The Au1 implements an internal 36-bit physical address SBUS which is then decoded by integrated peripherals, and by chip-selects for external memories and peripherals.

The cache coherency attributes (CCA) of the physical page are controlled by the TLB entry. The valid values are described in Table 2-3 on page 17. In general, I/O spaces require a non-cacheable setting, whereas memory can utilize a cacheable setting.

**Note:** Physical addresses in which address bits [35:32] are non-zero must be mapped non-cached (CCA = 2).

The TLB array is managed completely by software. Software can implement a TLB replacement algorithm that is either random (via the **TLBWR** instruction) or deterministic (via the **TLBWI** instruction). Hardware is available to segment the TLB via the Wired register so different replacement strategies can be used for different areas of the TLB.



## 2.5 Exceptions

The Au1 core implements a MIPS32 compliant exception scheme. The scheme consists of the exception vector entry points in both KSEG0 and KSEG1, and the exception code (ExcCode) encodings to determine the nature of the exception.

### 2.5.1 Exception Causes

The nature of an exception is reported in the Cause[ExcCode] field. The Au1 core can generate the following exceptions:

**Table 2-5. Cause[ExcCode] Encodings**

ExcCode	Mnemonic	Description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	IBE	Bus error exception (instruction fetch)
7	DBE	Bus error exception (data reference: load or store)
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved instruction exception
11	CpU	Coprocessor Unusable exception
12	Ov	Arithmetic Overflow exception
13	Tr	Trap exception
23	WATCH	Reference to Watchpoint address
24	MCheck	Machine Check (duplicate TLB entry)

The Au1 core does not implement hardware floating-point. As a result, all floating-point instructions generate the Reserved Instruction (RI) exception, which permits floating-point operations to be emulated in software.

In addition, the Au1 core does not recognize Soft Reset, Non-Maskable Interrupt (NMI), or Cache Error exception conditions.

## 2.5.2 Interrupt Architecture

The Au1 core implements a MIPS32 compliant interrupt mechanism in which eight interrupt sources are presented to the core. Each interrupt source is individually maskable to either enable or disable the core from detecting the interrupt. Interrupts are generated by software, integrated interrupt controllers, performance counters and timers, as noted in Table 2-6.

**Table 2-6. CPU Interrupt Sources**

Interrupt Source	CP0 Cause Register Bit	CP0 Status Register Bit
Software Interrupt 0	8	8
Software Interrupt 1	9	9
Interrupt Controller 0 Request 0	10	10
Request 1	11	11
Interrupt Controller 1 Request 0	12	12
Request 1	13	13
Performance Counters	14	14
Count/Compare	15	15

All interrupt sources are equal in priority; that is, the interrupt sources are not prioritized in hardware. As a result, software determines the relative priority of the interrupt sources. When Cause[ExcCode]=0, software must examine the Cause[IP] bits to determine which interrupt source is requesting the interrupt.

For more information on Interrupt Controller 0 and 1 see Section 5.0 on page 83.

## 2.6 MIPS32™ Instruction Set

The Au1 core implements the instruction set defined in “MIPS32 Architecture For Programmers Volume II: The MIPS32 Instruction Set”. The floating-point instructions are not implemented in the Au1 core, but may be emulated in software.

The MIPS32 ISA is characterized as a combination of the R3000 user level instructions (MIPSII) and the R4000 memory management and kernel mode instructions (32-bit MIPSIII).

### 2.6.1 CACHE Instruction

The **CACHE** instruction permits management of the Au1 instruction and data caches. The valid operations are listed in Table 2-2 "Cache Operations" on page 17.

For *data* cache operations, the effect of the **CACHE** instruction is immediately visible to subsequent data accesses. However, for *instruction* cache operations, the effect of the **CACHE** instruction is not visible to subsequent instructions already in the Au1 core pipeline. Therefore, care should be exercised if modifying instruction cache lines containing the **CACHE** and subsequent instructions.

When issuing the **CACHE** instruction with indexed operations (Index Invalidate, Index Load Tag and Index Store Tag) the format of the effective address is as follows:

#### CACHE Index Operation Address Decode

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8000																Way				Set/Index				Byte Select							

The effective address base should be 0x80000000 (KSEG0) to avoid possible TLB exceptions, and place zeros in the remainder of the effective address. The format correlates to a 16 KB cache that is 4-way set associative with 128 sets and 32-byte line size.

Software must not use the Index Store Tag **CACHE** operation to change the Dirty, Lock and Shared state bits. To set the Lock bit, software must use the Fetch and Lock **CACHE** operation.

The Index Load Tag and Index Store Tag **CACHE** operations utilize CP0 registers DTag, DData, ITag and IData. The format of data for Index Tag operations is depicted in the description of these registers.

**CACHE** operations that require an effective address (i.e. not the Index operations) do not generate the Address Error Exception or trigger data watchpoint exceptions.

### 2.6.2 PREF Instruction

The **PREF** instruction prefetches data into the data cache. Data is prefetched to improve algorithm performance by placing the data in the cache in advance of its use, thus minimizing stalls due to data cache load misses. See also Section 2.2.8.4 "Data Cache Control" on page 20 for more on how to use **PREF**.

If the effective address computed by the **PREF** instruction does not translate in the TLB (i.e., the address would cause a TLBL exception), no exception is generated and the cache is unchanged.

The Au1 core implements the following **PREF** instruction hints:

- 0x00 - Normal load
- 0x04 - Streaming load

A **PREF** instruction using any other hint value becomes a **NOP** for the Au1 core.

### 2.6.3 WAIT Instruction

The **WAIT** instruction places the Au1 core in one of two low power modes: IDLE0 and IDLE1. The low power mode is encoded in the **WAIT** instruction bits 24:6 (implementation-dependent code). A value of 0 selects IDLE0, and the value 1 selects IDLE1. Other values are not supported and must not be used.

In the IDLE0 low power mode, the Au1 core stops clocks to all possible core units but continues to snoop the SBUS to maintain data coherency.

In the IDLE1 low power mode, the Au1 core stops clocks to all possible core units, including the data cache, so data coherency is no longer maintained.

In either Idle mode, the general purpose registers and the CP0 registers are preserved, so that when Idle mode is exited by an appropriate event, the Au1 core resumes processing instructions in exactly the same context as prior to entering Idle mode.

To enter the low power mode, the **WAIT** instruction must be followed by at least four NOPs, and the entire instruction sequence must be fetched from the instruction cache. More specifically, if the core fetches the **WAIT** and **NOP** instructions from main memory, then the mechanisms for accessing memory will prevent the core from entering low power mode. This is the recommended code sequence:

```
.global aul_wait
aul_wait:
    la t0, aul_wait # obtain address of aul_wait
    cache 0x14, 0(t0) # fill icache with first 8 insns
    cache 0x14, 32(t0) # fill icache with next 8 insns
    sync
    nop
    wait 0
    nop
    nop
    nop
    nop
    j ra
```

When the Au1 core is in Idle mode, the Count register increments at an unpredictable rate; therefore the Count/Compare registers can not be used as the system timer tick when using the **WAIT** instruction to enter an Idle mode.

## 2.7 Coprocessor 0

Coprocessor 0 (CP0) is responsible for virtual memory, cache and system control.

The MIPS32 ISA provides for differentiation of the CP0 implementation. The Au1 core has a unique CP0 that is compliant with MIPS32 specification.

The Au1 CP0 registers are listed in Table 2-7.

**Table 2-7. Coprocessor 0 Register Definitions**

Register Number	Sel	Register Name	Description	Compliance (Note 1)
0	0	Index	Pointer into TLB array	Required
1	0	Random	Pseudo-random TLB pointer	Required
2	0	EntryLo0	Low half of TLB entry for even pages	Required
3	0	EntryLo1	Low half of TLB entry of odd pages	Required
4	0	Context	Pointer to a page table entry	Required
5	0	PageMask	Variable page size select	Required
6	0	Wired	Number of locked TLB entries	Required
7	0		Reserved	Reserved
8	0	BadVAddr	Bad virtual address	Required
9	0	Count	CPU cycle count	Required
10	0	EntryHi	High half of TLB entries	Required
11	0	Compare	CPU cycle count interrupt comparator	Required
12	0	Status	Status	Required
13	0	Cause	Reason for last exception	Required
14	0	EPC	Program Counter of last exception	Required
15	0	PRId	Processor ID and Revision	Required
16	0	Config	Configuration Registers (aka Config0)	Required
	1	Config1	Configuration Register 1	Required
17	0	LLAddr	Load Link Address	Optional
18	0	WatchLo	Data memory break point low bits	Optional
	1	IWatchLo	Instruction fetch breakpoint low bits	Optional
19	0	WatchHi	Data memory break point high bits	Optional
	1	IWatchHi	Instruction fetch breakpoint high bits	Optional
20-21	0		Reserved	Reserved
22	0	Scratch	Scratch register	Au1
23	0	Debug	EJTAG control register	Optional
24	0	DEPC	PC of EJTAG debug exception	Optional
25	0	Reserved	Reserved	Au1 Reserved
	1	Reserved	Reserved	Au1 Reserved
26-27	0		Reserved	Reserved
28	0	DTag	Data cache tag value	Au1
	1	DData	Data cache data value	Au1
29	0	ITag	Instruction cache tag value	Au1
	1	IData	Instruction cache data value	Au1
30	0	ErrorEPC	Program counter at last error	Required
31	0	DESave	EJTAG debug exception save register	Optional

Note 1. A compliance of “Required” denotes a register required by the MIPS32 architecture. “Optional” denotes an optional register in the MIPS32 architecture which is implemented in the Au1 core. “Au1” denotes an optional register unique to the Au1 core. “Reserved” denotes a register that is not implemented.

### 2.7.1 Index Register (CP0 Register 0, Select 0)

The Index register is required for TLB-based virtual address translation units.

#### Index

#### CP0 Register 0, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	P	0																									Index										
Def.	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X

Bits	Name	Description	R/W	Default
31	P	Probe Failure.	R	UNPRED
30:5	Reserved	Must always write zeros, always reads zeros	R	0
4:0	Index	TLB Index	R/W	UNPRED

### 2.7.2 Random Register (CP0 Register 1, Select 0)

The Random register is required for TLB-based virtual address translation units.

#### Random

#### CP0 Register 1, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
	0																									Random													
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bits	Name	Description	R/W	Default
31:5	Reserved	Must always write zeros, always reads zeros	R	0
4:0	Random	TLB Random Index	R	31

### 2.7.3 EntryLo0, EntryLo1 Register (CP0 Registers 2 and 3, Select 0)

The EntryLo0 and EntryLo1 registers are required for TLB-based virtual address translation units.

#### EntryLo0, EntryLo1

#### CP0 Registers 2 and 3, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	0	PFN																									C	D	V	G							
Def.	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:30	Reserved	Ignored on writes, returns zero on read	R	0
29:6	PFN	Page Frame Number. Corresponds to physical address bits 35..12.	R/W	UNPRED
5:3	C	Cache coherency attribute of the page. See Table 2-3 "Cache Coherency Attributes (CCA)" on page 17.	R/W	UNPRED
2	D	Dirty bit.	R/W	UNPRED
1	V	Valid bit	R/W	UNPRED
0	G	Global bit	R/W	UNPRED

### 2.7.4 Context Register (CP0 Register 4, Select 0)

The Context register is required for TLB-based virtual address translation units.

Context		CP0 Register 4, Select 0																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	PTEBase									BadVPN2									0																
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0

Bits	Name	Description	R/W	Default
31:23	PTEBase	Used by the operating system as a pointer into the current PTA array in memory.	R/W	UNPRED
22:4	BadVPN2	Contains virtual address bits 31..13 upon a TLB exception.	R	UNPRED
3:0	Reserved	Reserved	R	0

### 2.7.5 PageMask Register (CP0 Register 5, Select 0)

The PageMask register is required for TLB-based virtual address translation units.

PageMask		CP0 Register 5, Select 0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0			Mask													0																
Def.	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:29	Reserved	Ignored on write, returns zero on read.	R	0
28:13	Mask	The Mask field is a bit mask in which a “1” bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. See Table 2-4 "Values for Page Size and PageMask Register" on page 24.	R/W	UNPRED
12:0	Reserved	Ignored on write, returns zero on read.	R	0

### 2.7.6 Wired Register (CP0 Register 6, Select 0)

The Wired register is required for TLB-based virtual address translation units.

Wired		CP0 Register 6, Select 0																														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0															Wired																
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:5	Reserved	Ignored on write, returns zero on read.	R	0
4:0	Wired	TLB wired boundary.	R/W	0

### 2.7.7 BadVAddr Register (CP0 Register 8, Select 0)

The BadVAddr register is required for TLB-based virtual address translation units.

BadVAddr		CP0 Register 8, Select 0															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	BadVAddr																
Def.	X X																

Bits	Name	Description	R/W	Default
31:0	BadVAddr	Bad virtual address.	R	UNPRED

### 2.7.8 Count Register (CP0 Register 9, Select 0)

The Count register is a required register for a constant rate timer. This counter increments 1:1 with the core frequency.

During IDLE0 or IDLE1 mode, the Count register increments at an unpredictable rate; therefore the Count/Compare registers can not be used as the system timer tick when using the **WAIT** instruction to enter an Idle mode.

During Sleep mode, this register will not increment.

Count		CP0 Register 9, Select 0															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	Count																
Def.	0 0																

Bits	Name	Description	R/W	Default
31:0	Count	Interval counter.	R/W	0

### 2.7.9 EntryHi Register (CP0 Register 10, Select 0)

The Index register is required for TLB-based virtual address translation units.

EntryHi		CP0 Register 10, Select 0																			
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																				
	VPN2													0			ASID				
Def.	X X																				

Bits	Name	Description	R/W	Default
31:13	VPN2	Virtual address bits 31..13.	R/W	UNPRED
12:8	Reserved	Ignored on write, returns zero on read.	R	0
7:0	ASID	Address space identifier.	R/W	UNPRED

### 2.7.10 Compare Register (CP0 Register 11, Select 0)

The Compare register is a required register for generating an interrupt from the constant rate timer.

Compare		CP0 Register 11, Select 0															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	Compare																
Def.	X X																

Bits	Name	Description	R/W	Default
31:0	Compare	Interval counter compare value.	R/W	UNPRED

### 2.7.11 Status Register (CP0 Register 12, Select 0)

The Status register is a required register for general control of the processor.

#### Status

#### CP0 Register 12, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	CU0	RP	0	RE	0	0	BEV	0	SR	NMI	0	0	0	IM					0	0	0	0	0	0	UM	0	ERL	EXL	IE	
Def.	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	R/W	Default
31	CU3	This bit is zero. Coprocessor 3 is not implemented.	R	0
30	CU2	This bit is zero. Coprocessor 2 is not implemented.	R	0
29	CU1	This bit is zero. Coprocessor 1 is not implemented.	R	0
28	CU0	Controls access to coprocessor 0.	R/W	0
27	RP	Reduced power. This bit has no effect.	R/W	0
26	RSVD	Reserved.	R	0
25	RE	Reverse-endian.	R/W	0
24:23	RSVD	Reserved.	R	0
22	BEV	Boot exception vectors.	R/W	1
21	RSVD	Reserved.	R	0
20	SR	Soft reset.	R/W	0
19	NMI	Non-maskable interrupt	R/W	0
18:16	RSVD	Reserved.	R	0
15:8	IM	Interrupt mask	R/W	0
7:5	RSVD	Reserved.	R	0
4	UM	User-mode.	R/W	0
3	R0	This bit is zero; Supervisor-mode not implemented	R	0
2	ERL	Error Level	R/W	1
1	EXL	Exception Level	R/W	0
0	IE	Interrupt Enable	R/W	0

### 2.7.12 Cause Register (CP0 Register 13, Select 0)

The Cause register is a required register for general exception processing.

#### Cause

#### CP0 Register 13, Select 0

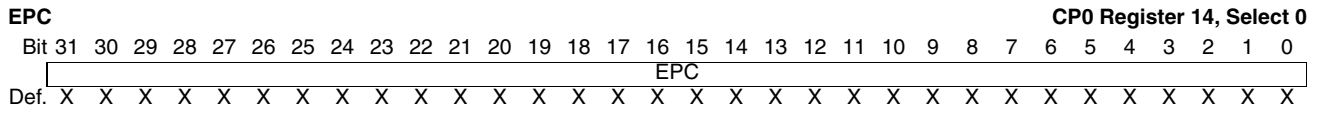
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	BD	0	CE	0			IV	WP	0					IP					0	ExcCode			0										
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	BD	Exception in branch delay slot	R	0
29:28	CE	Coprocessor error	R	0
27:24	RSVD	Reserved.	R	0
23	IV	Interrupt vector	R/W	0
22	WP	Watchpoint exception deferred	R/W	0
21:16	RSVD	Reserved.	R	0
15:10	IP[7:2]	Hardware interrupts pending	R	0x20
9:8	IP[1:0]	Software interrupts pending	R/W	0
7	RSVD	Reserved.	R	0
6:2	ExcCode	Exception Code	R	0
1:0	RSVD	Reserved.	R	0



### 2.7.13 Exception Program Counter (CP0 Register 14, Select 0)

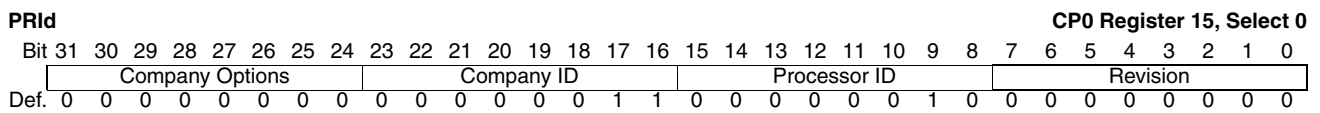
The Exception Program Counter (EPC) register is a required register for general exception processing.



Bits	Name	Description	R/W	Default
31:0	EPC	Exception Program Counter	R/W	UNPRED

### 2.7.14 Processor Identification (CP0 Register 15, Select 0)

The PRID register is a required register for processor identification.



Bits	Name	Description	R/W	Default
31:24	Company Options	System-on-a-chip (SOC) identification: 0 Au1000 1 Au1500 2 Au1100	R	2
23:16	Company ID	Company ID assigned by MIPS Technologies. AMD's ID = 3.	R	3
15:8	Processor Core ID	Identifies the core revision: 0 Reserved 1 Au1 revision 1 2 Au1 revision 2	R	2
7:0	Revision	Contains a manufacturing-specific revision level. 1 Silicon stepping BA; silicon revision 1.0 2 Silicon stepping BC; silicon revision 1.1 3 Silicon stepping BD; silicon revision 1.2 4 Silicon stepping BE; silicon revision 1.3	R	SOC specific

**2.7.15 Configuration Register 0 (CP0 Register 16, Select 0)**

The Config0 register is a required register for various processor configuration and capability.

**Config0**

**CP0 Register 16, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	M	CT				DD	CD	UM	WD	NM	SM	OD	0	0	TM	BE	AT	AR	MT	0	K0				0	1	1	0	0	0	0	0	0	1	1
Def.	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	

Bits	Name	Description	R/W	Default
31	M	Denotes Config1 register available at select 1	R	1
30:26	CT	Reserved, must write 0	R/W	0
25	DD	Reserved, must write 0	R/W	0
24	CD	Reserved, must write 0	R/W	0
23	UM	Reserved, must write 0	R/W	0
22	WD	Reserved, must write 0	R/W	0
21	NM	Reserved, must write 0	R/W	0
20	SM	Reserved, must write 0	R/W	0
19	OD	Reserved, must write 0	R/W	0
16	TM	Reserved, must write 0	R/W	0
15	BE	Indicates the endian mode.	R	1
14:13	AT	Architecture type is MIPS32.	R	0
12:10	AR	Architecture revision is Revision 1.	R	0
9:7	MT	MMU type is standard TLB.	R	1
2:0	K0	KSEG0 is cacheable, coherent.	R/W	3

**2.7.16 Configuration Register 1 (CP0 Register 16, Select 1)**

The Config1 register is a required register for various processor configuration and capability.

**Config1**

**CP0 Register 16, Select 1**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	MMU Size - 1				IS	IL	IA	DS	DL	DA	C2	MD	PC	WR	CA	EP	FP	0	0	0	0	0	1	1	0	0	0	1	0	1	0	
Def.	0	0	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	1	0	0	0	1	0	1	0

Bits	Name	Description	R/W	Default
30:25	MMU Size - 1	Number of entries in the TLB minus one. The TLB has 32 entries.	R	31
24:22	IS	Instruction cache sets per way is 128.	R	1
21:19	IL	Instruction cache line size is 32 bytes.	R	4
18:16	IA	Instruction cache associativity is 4-way.	R	3
15:13	DS	Data cache sets per way is 128.	R	1
12:10	DL	Data cache line size is 32 bytes.	R	4
9:7	DA	Data cache associativity is 4-way.	R	3
6	C2	Coprocessor 2 is not implemented.	R	0
5	MD	Always returns zero on read.	R	0
4	PC	Performance Counter registers are not implemented.	R	0
3	WR	Watchpoint registers are implemented.	R	1
2	CA	Code compression is not implemented.	R	0
1	EP	EJTAG is implemented.	R	1
0	FP	FPU is not implemented.	R	0

### 2.7.17 Load Linked Address Register (CP0 Register 17, Select 0)

The LLAddr register provides the physical address of the most recent Load Linked instruction.

<b>LLAddr</b>																		<b>CP0 Register 17, Select 0</b>															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	LLAddr																																
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	LLAddr	Load Linked Address	R	UNPRED

### 2.7.18 Data WatchLo Register (CP0 Register 18, Select 0)

The WatchLo and WatchHi registers are the interface to the data watchpoint facility.

<b>WatchLo</b>																		<b>CP0 Register 18, Select 0</b>																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	VAddr																												0	R	W			
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0

Bits	Name	Description	R/W	Default
31:3	VAddr	The virtual address to match	R/W	UNPRED
1	R	If this bit is a one, then watch exceptions are enabled for loads that match the address.	R/W	0
0	W	If this bit is a one, then watch exceptions are enabled for stores that match the address.	R/W	0

### 2.7.19 Instruction WatchLo Register (CP0 Register 18, Select 1)

The IWatchLo and IWatchHi registers are the interface to the instruction watchpoint facility.

<b>IWatchLo</b>																		<b>CP0 Register 18, Select 1</b>															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	VAddr																												I	0	0		
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0

Bits	Name	Description	R/W	Default
31:3	VAddr	The virtual address to match	R/W	UNPRED
2	I	If this bit is a one, then watch exceptions are enabled for instruction accesses that match the address.	R/W	0

**2.7.20 Data WatchHi Register (CP0 Register 19, Select 0)**

The WatchLo and WatchHi registerS are the interface to the data watchpoint facility.

**WatchHi**

**CP0 Register 19, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	M	G	0						ASID						0						Mask						0						
Def.	1	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0

Bits	Name	Description	R/W	Default
31	M	another pair of Watch registers is implemented at the next Select index.	R	1
30	G	If this bit is one, then the ASID field is ignored and any address that matches causes a watch exception.	R	UNPRED
23:16	ASID	ASID value which is required to match that in the EntryHi register if the G bit is zero in the WatchHi register.	R/W	UNPRED
11:3	Mask	Any bit in this field that is a one inhibits the corresponding address bit from participating in the address match.	R/W	UNPRED

**2.7.21 Instruction WatchHi Register (CP0 Register 19, Select 1)**

The IWatchLo and IWatchHi registers are the interface to the instruction watchpoint facility.

**IWatchHi**

**CP0 Register 19, Select 1**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	G	0						ASID						0						Mask						0						
Def.	0	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	X	X	X	X	X	X	X	X	X	X	0	0	0

Bits	Name	Description	R/W	Default
30	G	If this bit is one, then the ASID field is ignored and any address that matches causes a watch exception.	R	UNPRED
23:16	ASID	ASID value which is required to match that in the EntryHi register if the G bit is zero in the WatchHi register.	R/W	UNPRED
11:3	Mask	Any bit in this field that is a one inhibits the corresponding address bit from participating in the address match.	R/W	UNPRED

**2.7.22 Scratch Register (CP0 Register 22, Select 0)**

The Scratch register exists for the convenience of software. Upon a read, this register returns the value last written into it.

**Scratch**

**CP0 Register 22, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Scratch																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	Scratch	This register is present for the convenience of software.	R/W	UNPRED

### 2.7.23 Debug Register (CP0 Register 23, Select 0)

The Debug register is part of the interface to the EJTAG facility.

#### Debug

#### CP0 Register 23, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DB	DM	0	LS	0						001			DExcCode					0	SSt	0	0	DIN	0			DB	DS				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	DBD	Debug exception in branch delay slot.	R	UNPRED
30	DM	If this bit is a one, then in debug mode.	R	0
28	LSNM	Load/stores are performed in the normal fashion when in debug mode.	R/W	0
17:15	001	EJTAG version 2.5	R	001
14:10	DExcCode	Cause[ExcCode] for normal exceptions in debug mode.	R	UNPRED
8	SSt	Enable single step mode	R/W	0
5	DINT	Last debug exception was asynchronous debug interrupt	R	0
1	DBp	Last debug exception was an SDBPP instruction	R	0
0	DSS	Last debug exception was a single step	R	0

### 2.7.24 DEPC Register (CP0 Register 24, Select 0)

The DEPC register is part of the interface to the EJTAG facility.

#### DEPC

#### CP0 Register 24, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DEPC																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	DEPC	Debug exception program counter.	R/W	UNPRED

### 2.7.25 Data Cache Tag Register (CP0 Register 28, Select 0)

The DTag and DData registers are the interface to the data cache array. This cache interface is unique to the Au1.

**Note:** This register corresponds to the TagLo register in the MIPS32 ISA specification.

#### DTag

#### CP0 Register 28, Select 0

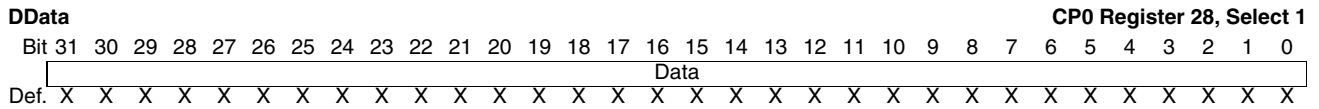
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAG																		MRU	NMRU	LRU	0	0	D	S	L	V					
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	X	X	X	X

Bits	Name	Description	R/W	Default
31:12	TAG	TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero.	R/W	UNPRED
11:10	MRU	Most recently used way.	R/W	UNPRED
9:8	NMRU	Next most recently used way.	R/W	UNPRED
7:6	LRU	Least recently used way.	R/W	UNPRED
3	D	Cache line is dirty (modified).	R/W	UNPRED
2	S	Cache line is shared (for data cache snoops).	R/W	UNPRED
1	L	Locked. This bit is set by the user to prevent overwriting of the cache line.	R/W	UNPRED
0	V	Cache line valid.	R/W	UNPRED

**2.7.26 Data Cache Data Register (CP0 Register 28, Select 1)**

The DTag and DData registers are the interface to the data cache array.

**Note:** This register corresponds to the DataLo register in the MIPS32 ISA specification.

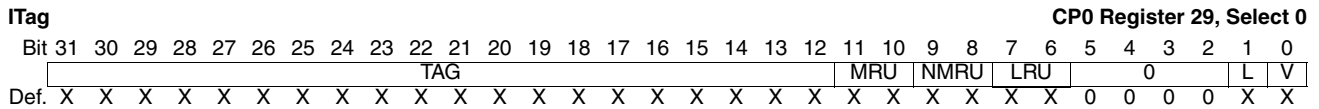


Bits	Name	Description	R/W	Default
31:0	Data	Data from the data cache line.	R	UNPRED

**2.7.27 Instruction Cache Tag Register (CP0 Register 29, Select 0)**

The ITag and IData registers are the interface to the instruction cache array. This cache interface is unique to the Au1.

**Note:** This register corresponds to the TagHi register in the MIPS32 ISA specification.

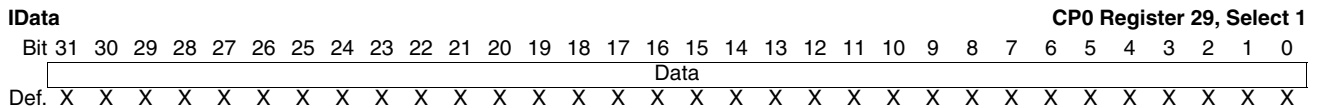


Bits	Name	Description	R/W	Default
31:12	TAG	TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero.	R/W	UNPRED
11:10	MRU	Most recently used way.	R/W	UNPRED
9:8	NMRU	Next most recently used way.	R/W	UNPRED
7:6	LRU	Least recently used way.	R/W	UNPRED
1	L	Locked. This bit is set by the user to prevent overwriting of the cache line.	R/W	UNPRED
0	V	Cache line valid.	R/W	UNPRED

**2.7.28 Instruction Cache Data Register (CP0 Register 29, Select 1)**

The ITag and IData registers are the interface to the instruction cache array.

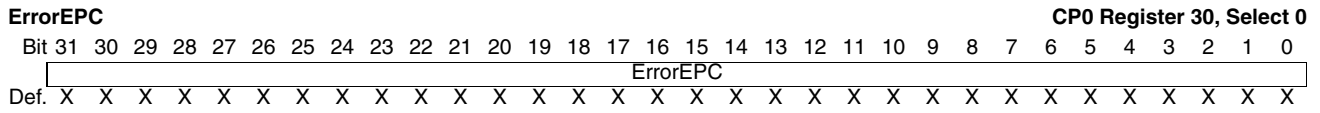
**Note:** This register corresponds to the DataHi register in the MIPS32 ISA specification.



Bits	Name	Description	R/W	Default
31:0	Data	Data from the instruction cache line.	R	UNPRED

### 2.7.29 ErrorEPC Register (CP0 Register 30, Select 0)

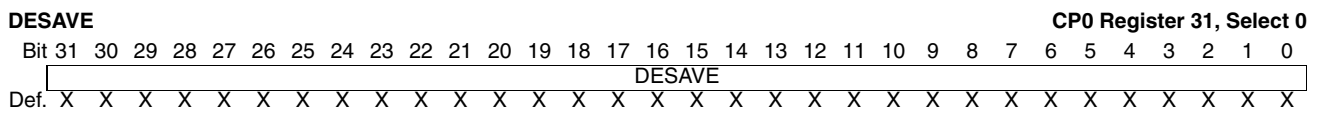
The ErrorEPC register is a required register for exception processing.



Bits	Name	Description	R/W	Default
31:0	ErrorEPC	Error Exception Program Counter	R/W	UNPRED

### 2.7.30 DESAVE Register (CP0 Register 31, Select 0)

The DESAVE register is part of the interface to the EJTAG facility.



Bits	Name	Description	R/W	Default
31:0	DESAVE	Debug save scratch register, for debug handlers.	R/W	UNPRED

## 2.8 System Bus (SBUS)

The Au1 core communicates with memories and peripherals via the System Bus (SBUS). The SBUS is a 36-bit physical address and 32-bit data bus which is internal to the Au1100 processor. The SBUS is the coherency point within the Au1100 processor.

### 2.8.1 SBUS Arbitration

The SBUS supports multiple masters - the Au1 core and peripheral DMA engines. The SBUS is granted to the masters in a least-recently-used/fair scheme. This scheme prevents two or more masters from consuming the entire SBUS bandwidth, while permitting low latency access to the SBUS for masters which request the bus infrequently (such as peripherals).

The SBUS requestors in the Au1100 processor are:

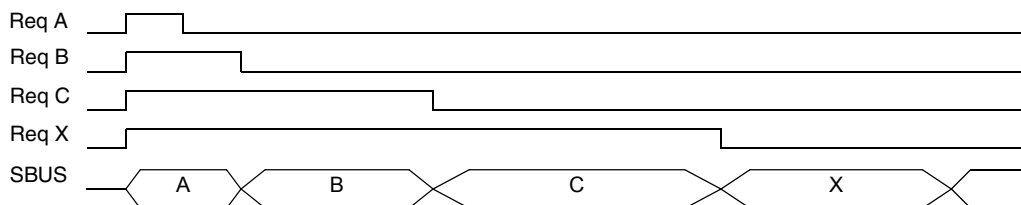
- Au1 core
- Ethernet MAC controller
- USB Host controller
- IrDA controller
- LCD Controller
- DMA controller

The Au1 presents a single request to the SBUS arbiter for the three possible requestors: the data cache, the instruction cache and the write buffer. The data cache has the highest priority and the write buffer the lowest priority among the three requests. However, the write buffer priority becomes the highest when the data cache requests a load to an address in the write buffer to allow the write buffer to empty prior to fulfilling the data cache load.

The SBUS arbiter has four bus arbitration slots for handling the SBUS masters:

- Slot 0: Au1 core (data cache, instruction cache, write buffer)
- Slot 1: LCD controller
- Slot 2: Ethernet MAC controller and DMA controller
- Slot 3: USB host controller and IrDA controller

The arbitration scheme for the SBUS is round-robin; that is, each bus master slot has an equal opportunity to obtain access to the SBUS. For a particular SBUS master X, if no other SBUS masters request the bus, then bus master X immediately wins the SBUS. By contrast, if all other SBUS masters request the bus, then bus master X must wait for three other SBUS master slots to transfer before it wins the SBUS, as shown in Figure 2-4.



**Figure 2-4. SBUS Arbitration**

When a SBUS master wins arbitration of the SBUS, it performs transfers to/from the integrated peripherals, SDRAM, or the Static bus.



### 2.8.2 SBUS Coherency Model

The SBUS is the coherency point within the Au1100 processor. An SBUS master (i.e., Au1 core or peripheral DMA engine) marks each SBUS transaction as either coherent or non-coherent. SBUS transactions marked as coherent are then snooped by all caching masters (i.e. Au1 data cache). An SBUS transaction that is marked non-coherent is not snooped by caching masters.

The Au1 core is a coherent, caching master. The Au1 data cache snoops SBUS transactions; if a read transaction hits in the data cache then the data cache provides the data, if a write transaction hits in the data cache then the data cache array is updated with the new data.

The integrated peripherals (with DMA engines) can be configured for coherent or non-coherent operation. The 'C' bit in the peripheral/module enable register directs whether peripheral SBUS transactions are to be marked coherent or non-coherent. If a peripheral is configured for coherent operation, then it is not necessary to writeback and invalidate Au1 data cache lines which hit in the memory buffers used by DMA engines. If, on the other hand, the peripheral is configured for non-coherent operation, then software must ensure that memory buffers used by the DMA engines are not in the data cache (else the data cache and/or the memory buffer may contain old, stale data).

The decision to use, or not use, coherent SBUS transactions is left to the application. However, peripheral device drivers using coherent SBUS transactions will perform better than drivers not using coherent SBUS transactions since the need to writeback the data cache is eliminated.

## 2.9 EJTAG

EJTAG is supported per the MIPS EJTAG Rev. 2.5 specification. EJTAG provides for CPU and board level bring-up and debug.



# Memory Controllers

# 3

The Au1100 processor contains two memory controllers, one for SDRAM and one for static devices.

The SDRAM controller supports SDRAM, SMROM and Sync Flash. The controller has its own independent voltage I/O that may be programmed to supply various output voltages such as 2.5V and 3.3V.

The static device controller supports SRAM, Flash, ROM, page mode ROM, PCMCIA/Compact Flash devices, and an external LCD controller interface.

Both memory controllers support software configurable memory address spaces. This allows designers to keep memory regions contiguous. For example, a system with 4 MB initially installed would locate the memory at physical address 0. Normally, adding 16 MB would create a 12 MB gap in the memory map. With the address configuration options in the Au1100 the 4 MB can be relocated to start at 16 MB, and the new memory can be located at 0 to allow a 20 MB contiguous memory pool.

All registers in the Memory Controller block are located off of the base address shown in Table 3-1.

**Table 3-1. Memory Controller Block Base Address**

Name	Physical Base Address	KSEG1 Base Address
mem	0x0 1400 0000	0xB400 0000

The system designer has the choice of booting from 32-bit Flash, 16-bit Flash, 32-bit SMROM, and 32-bit SyncFlash. The ROMSEL and ROMSIZE configuration is discussed in more detail in Section 8.3 "Boot" on page 225. Table 8-1 on page 225 shows how the state of ROMSEL and ROMSIZE determines where the processor boots from.

### 3.1 SDRAM Memory Controller

The SDRAM memory controller of the Au1100 processor is designed for glueless interface to one, two, or three ranks of SDRAM or SMROM. SDRAM and SyncFlash are run at 1/2 the internal System Bus (SBUS) speed. The SBUS defaults to 1/2 the processor clock speed so that SDRAM or SyncFlash will run at 99 MHz with a 396-MHz Au1100. SMROM operates at 1/4 the speed of the SBUS. The SBUS divider is programmable, see Section 7.4.3 "Device Power Management - Sleep" on page 216 for more information.

The SDRAM interface supports three chip selects (SDCS[2:0]#), corresponding to three ranks of SDRAM. Each chip select can be configured to support either SDRAM or SMROM. In addition, chip select 0 can be configured for SyncFlash (no other chip selects can be used to support SyncFlash). For chip selects configured as SDRAM or SyncFlash (on chip select 0) the controller keeps one row open for up to four banks per chip select allowing fast accesses and reducing the need to issue precharge cycles.

**Note:** The SDRAM memory controller supports a maximum of two loads per chip select.

When RESETIN# is negated, code is fetched from SMROM/SyncFlash if SMROM/SyncFlash boot is selected. When using SMROM or SyncFlash for boot, the SMROMCKE signal should be used for the SMROM/SyncFlash CKE. If SMROM or SyncFlash are being used (but not for boot), SDCKE should be used for the clock enable.

After boot internal configuration registers can be written to enable SDRAM chip selects. When a chip select is enabled the SDCKE is driven asserted and clocks are started. Software must wait 10  $\mu$ s for the SDRAM clock to stabilize before any device specific initialization steps.

All SDRAM/SMROM ranks must be 32 bits wide. Support is included for SDRAM with 2 or 4 banks, 11 to 13 row address bits, and 7 to 11 column address bits. It is also possible to send explicit commands to the SDRAM, under software control, for diagnostic, initialization, or power management purposes.

SDRAM clocks keep running during a runtime reset to allow any transaction in progress to complete. This avoids the possibility of bus contention when the part is brought out of reset.

The SDRAM controller assumes the following external SDRAM configuration:

- Burst Length = 8
- Addressing Mode = Sequential
- Write Mode = Burst Read and Write

### 3.1.1 SDRAM Controller Programming Model

The SDRAM controller contains a number of registers which configure the operation of the interface. All registers in the SDRAM controller block are located off of the base address shown in Table 3-1 on page 43. Table 3-2 shows the memory map of the register block.

**Table 3-2. SDRAM Configuration Registers**

Offset (Note 1)	Register Name	Description
0x0000	mem_sdmode0	SDRAM chip select n (SDCSn#) mode configuration register (timing and functionality)
0x0004	mem_sdmode1	
0x0008	mem_sdmode2	
0x000C	mem_sdaddr0	SDCSn# address configuration and enable
0x0010	mem_sdaddr1	
0x0014	mem_sdaddr2	
0x0018	mem_sdrefcfg	Refresh Configuration and Timing
0x001C	mem_sdprecmd	Issue PRECHARGE to all enabled SDRAM chip selects
0x0020	mem_sdautoref	Issue AUTO REFRESH to all enabled SDRAM chip selects
0x0024	mem_sdwrmd0	Write data to SDCSn# SDRAM mode configuration register
0x0028	mem_sdwrmd1	
0x002C	mem_sdwrmd2	
0x0030	mem_sdsleep	Force SDRAM into self refresh mode
0x0034	mem_sdsmcke	Toggle SMROMCKE pin

Note 1. See Table 3-1 on page 43 for base address.

### 3.1.2 SDRAM Registers

Each chip select is configured by two registers, a mode register and an address configuration register.

#### 3.1.2.1 Chip Select Mode Configuration Registers

The format and reset values of the chip select mode configuration registers is shown in the following figure. The timing parameters (Tcl, Tcrd, Trp, Twr, Tmrd, and Tras) correspond directly to times shown in the SDRAM timing diagrams. Times are measured in SDRAM/SMROM clock cycles.

The default values for chip select zero correspond to values for SMROM operation. Chip select 1 and 2 are configured with the slowest timing values at reset.

Reserved fields should be written as zeros and ignored on read to preserve compatibility with future versions of the product.

#### mem\_sdmode0 - CS0 Mode Configuration

Offset = 0x0000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	[Reserved]																SF	F	SR	BS	RS	CS	Tras	Tmrd	Twr	Trp	Trcd	Tcl						
Def.	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0

#### mem\_sdmode1 - CS1 Mode Configuration

Offset = 0x0004

#### mem\_sdmode2 - CS2 Mode Configuration

Offset = 0x0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	[Reserved]																F	SR	BS	RS	CS	Tras	Tmrd	Twr	Trp	Trcd	Tcl						
Def.	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W	Default
31:24	—	Reserved, should be cleared.	R	0
23	SF	Selects SyncFlash operation. SyncFlash is available only on SDCS0#. For other chip selects, this bit is reserved and should be cleared. 0 SyncFlash is not being used. 1 SyncFlash is being used.	R/W	0
22	F	Setting the F bit allows the SDRAM controller to assume that no caching master except the core will access this memory space. This allows accesses to begin sooner.  Note that the CPU core is the only possible caching master, so it is safe for the system designer to set this bit.	R/W	0
21	SR	Chip select operating mode 0 SDRAM/SyncFlash Operation 1 SMROM Operation	R/W	See above
20	BS	Select Number of Banks 0 Chip select controls 2-bank SDRAM 1 Chip select controls 4-bank SDRAM  Note: This bit must be cleared for SMROM support.	R/W	See above
19:18	RS	This field sets the number of bits in the row address as shown below: <u>RS Row Address Size</u> 00 11 01 12 10 13 11 Reserved	R/W	See above
17:15	CS	This field sets the number of bits in the column address as shown below: <u>CS Column Size</u> 000 7 001 8 010 9 011 10 100 11  All other values are reserved.	R/W	See above
14:11	Tras	This field designates the minimum delay from a activate to a precharge command.  (Tras + 1) is the actual number of clock cycles.	R/W	15
10:9	Tmrd	This field sets the required delay from an external load of the SDRAM mode register (not the chip select mode register) to an activate command.  (Tmrd + 1) is the actual number of clock cycles.	R/W	3
8:7	Twr	The Twr field sets the write recovery time. This is the last data for a write to a precharge. This field is sometimes referred to a Tdpl.  (Twr + 1) is the actual number of clock cycles.	R/W	3
6:5	Trp	This field sets the time from precharge to the next activate command.  (Trp + 1) is the actual number of clock cycles.	R/W	3
4:3	Trcd	This field sets the RAS to CAS delay.  (Trcd + 1) is the actual number of clock cycles.	R/W	See Above
2:0	Tcl	This field sets the minimum CAS latency timing. This is the time from CAS to DATA on reads.  (Tcl + 1) is the actual number of clock cycles.	R/W	See Above

### 3.1.2.2 SDRAM Chip Select Address Configuration Registers (*mem\_sdaddrn*)

The SDRAM chip-select address configuration registers (*mem\_sdaddrn*) assign an address range for each chip select. As shown below, each register contains a base address, an address comparison mask, and an enable bit.

#### *mem\_sdaddr0* - SDCS0# Address Configuration

Offset = 0x000C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Rs											CSBA									CSMASK												
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

#### *mem\_sdaddr1* - SDCS1# Address Configuration

Offset = 0x0010

#### *mem\_sdaddr2* - SDCS2# Address Configuration

Offset = 0x0014

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Rs											CSBA									CSMASK											
Def.	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W	Default
31:21	—	Reserved, should be cleared.	R	0
20	E	Enable. 0 Chip select is disabled. 1 Chip select is enabled.	R/W	0, except for <b>mem_sdaddr0</b> (Note 1)
19:10	CSBA	Chip select base address. Specifies bits 31:22 of the physical base address for this chip select. (The lower bits of the base address are zero.)	R/W	0x3FF, except for <b>mem_sdaddr0</b> where the default value is 0x7F.
9:0	CSMASK	Chip select address mask. Specifies which bits of CSBA are used to decode this chip select.	R/W	0x3FF

Note 1. The E bits for the chip selects SDCS1# and SDCS2# are automatically cleared (disabled) coming out of a runtime or hardware reset. For SDCS0#, however, the reset value of the E bit depends on ROMSEL and ROMSIZE: SDCS0#'s E bit is set when the ROMSEL and ROMSIZE pins indicate that the SMROM/SyncFlash should be used for the boot vector (ROMSEL==1, ROMSIZE==0). Also see Section 8.3 "Boot" on page 225.

Once enabled (E bit set), a chip select is asserted when the following condition is met:

$$(\text{phys\_addr} \& \text{addr\_mask}) == \text{base\_addr}$$

where:

**phys\_addr**: 32-bit physical address output on the internal SBUS (from the TLB for memory-mapped regions) (Bits 35:32 of the physical address are zeros.)

**addr\_mask**: address comparison mask taken from CSMASK

**base\_addr**: chip select base address taken from CSBA

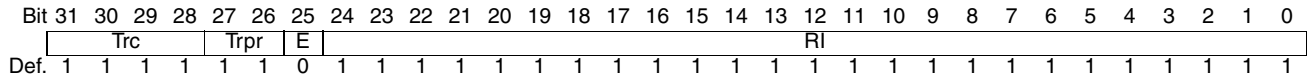
Chip select regions must be programmed so that each chip select occupies a unique area of the physical address space. Programming overlapping chip select regions results in undefined operation.

### 3.1.2.3 Refresh Configuration Register

The refresh configuration register sets the timing of SDRAM refresh for all chip selects. Since the timing for these signals apply to all chip selects, if different types of SDRAM is used the worst case timing must be applied. The format of the refresh configuration register is as follows:

**mem\_sdrefcfg - Refresh Configuration**

Offset = 0x0018



Bits	Name	Description	R/W	Default
31:28	Trc	The Trc field specifies the minimum time from the start of an auto refresh cycle to an activate command for all SDRAM chip selects. (Trc + 1) is the actual number of clock cycles.	R/W	0xf
27:26	Trpm	This field specifies the minimum time from a precharge to the start of a refresh cycle for all SDRAM chip selects. This is used because a pre-charge all command is automatically initiated before an auto refresh command. This value should be programmed with the worst case Trp from the <b>sdr_csmoden</b> registers. (Trpm + 1) is the actual number of clock cycles.	R/W	3
25	E	When this bit is set, refresh is enabled for all chip selects configured as SDRAM.	R/W	0
24:0	RI	Refresh Interval - This field specifies the maximum refresh interval in SBUS clocks for all SDRAM ranks.  The refresh interval is for each individual refresh so for a system with a row address size of 12 (4096 rows) and memory with a refresh time of 64 ms (all rows), the individual refresh interval will be 15.7 $\mu$ s (64 ms/4096). With a SBUS clock of 198 MHz, the RI value should be 0xC24 (15.7 $\mu$ s / (1/198 MHz)).	R/W	0x1FFFFFF

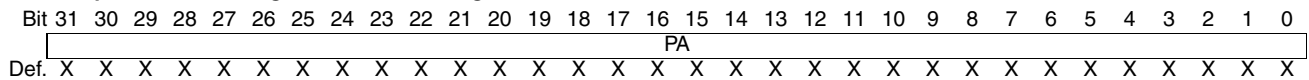
### 3.1.2.4 Precharge All Command Register

Writing any value to the **mem\_sdprecmd** register issues a precharge all command to all enabled SDRAM chip selects. This can be used for initialization sequences that require certain operations to be performed in a deterministic order.

Reading from the **mem\_sdprecmd** register is unpredictable.

**mem\_sdprecmd - Precharge All Command Reg**

Offset = 0x001C



Bits	Name	Description	R/W	Default
31:0	PA	Writing any value to PA will cause a precharge command to be issued to all enabled SDRAM chip selects.	W	UNPRED



### 3.1.2.5 Auto Refresh Command Register

Writing to the `mem_sdautoref` register performs an auto refresh command on all enabled SDRAM chip selects. This can be used for initialization sequences that require specific operations to be performed in a deterministic order. To insure future compatibility the value written should always be zero.

Reading from the `mem_sdautoref` register will return the current value of the refresh timer.

#### `mem_sdautoref` - Auto Refresh Command

Offset = 0x0020

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AR																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	AR	Writing a value to AR causes an AUTO REFRESH command to be issued to all enabled SDRAM chip selects.	R/W	UNPRED

### 3.1.2.6 External SDRAM Mode Register Access

The `mem_sdwrmd0`, `mem_sdwrmd1`, and `mem_sdwrmd2` command registers allow software to directly write to the mode registers in SDRAM connected to each chip select. This can be used in initialization sequences that require certain operations be performed in a deterministic order.

#### `mem_sdwrmd0` - Write CS0 SDRAM Mode

Offset = 0x0024

#### `mem_sdwrmd1` - Write CS1 SDRAM Mode

Offset = 0x0028

#### `mem_sdwrmd2` - Write CS2 SDRAM Mode

Offset = 0x002C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BA																WM															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:30	BA[1:0]	Bank address. These bits are reflected on the SDBA[1:0] signals. They can be used to write to the <i>extended</i> mode register (for synchronous Flash and battery RAM, for example). These bits must be cleared otherwise.	W	UNPRED
29:0	WM	The value written to this register is written to the external SDRAM mode register for the corresponding chip select.	W	UNPRED

### 3.1.2.7 SDRAM Sleep/Self Refresh Command Register

Writing any value to this register performs sends a self refresh command on all enabled SDRAM chip selects. This command can be used for the SDRAM power- down sequence which requires specific commands to be performed in a deterministic order.

After performing self refresh the SDRAM controller will hold SDCKE low and wait until a Sleep exit sequence or reset is performed. For this reason nothing should access the SDRAM after this command has been issued.

#### `mem_sdsleep` -SDRAM Sleep

Offset = 0x0030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	SL															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

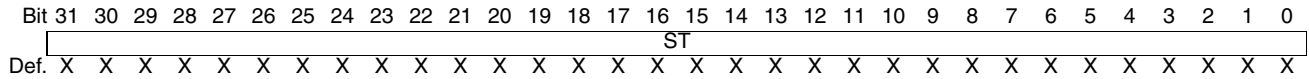
Bits	Name	Description	R/W	Default
31:0	SL	Writing any value to SL will issue a self refresh command on all enabled chip selects.	W	UNPRED

### 3.1.2.8 SMROMCKE Toggle Register

Writing to this register causes the state of the SMROMCKE signal to change. SMROMCKE will default to high when booting from SMROM or Sync Flash. This is used during power-up configuration to change the SMROM burst size from 4 to 8 beats. This command register does not affect the SDRAM SDCKE signal.

#### mem\_sdsmcke -SMROMCKE Toggle

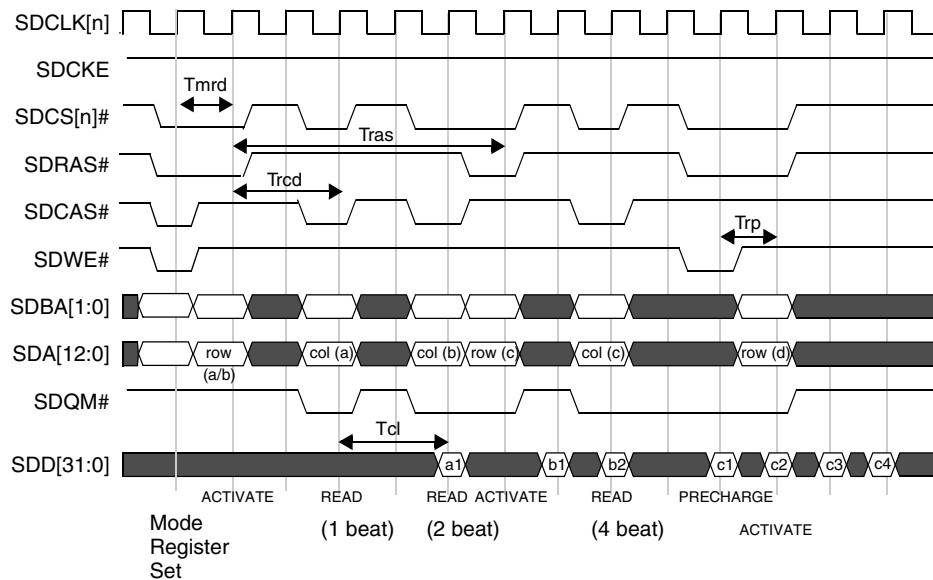
Offset = 0x0034



Bits	Name	Description	R/W	Default
31:0	ST	Writing to ST (regardless of the value written) inverts the current state of SMROMCKE.	W	UNPRED

### 3.1.3 SDRAM Timing

The following figures show examples of typical read, typical write and refresh timing.

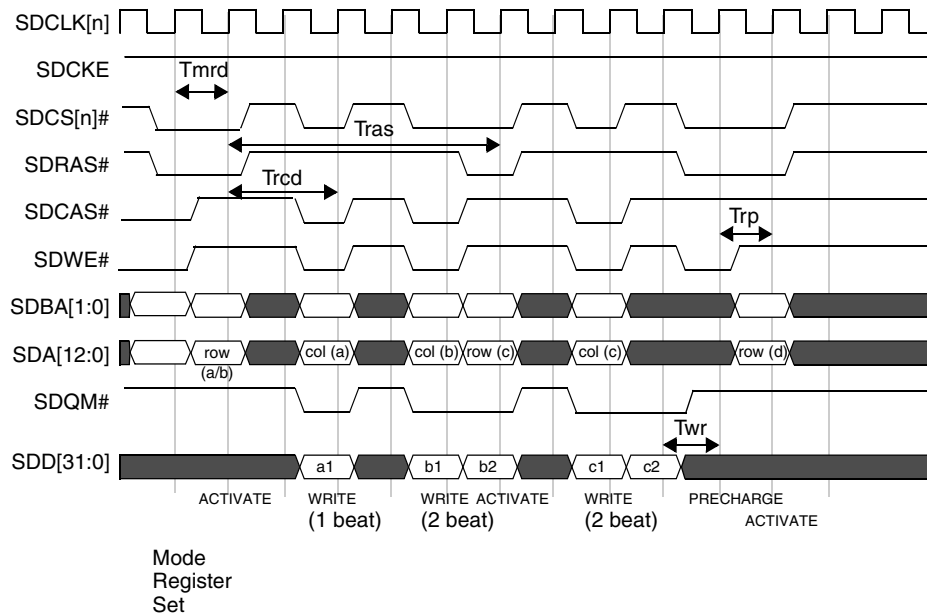


The above timing represents the following:

- 1)  $T_{ras} = 4$  (5 SDRAM clock cycles)
- 2)  $T_{rp} = 0$  (1 SDRAM clock cycles)
- 3)  $T_{rcd} = 1$  (2 SDRAM clock cycles)
- 4)  $T_{cl} = 1$  (2 SDRAM clock cycles)
- 5)  $T_{mrd} = 0$  (2 SDRAM clock cycles)

The above timing is presented to concisely display the different SDRAM timing parameters. The functional bus behavior may differ from that displayed.

**Figure 3-1. SDRAM Typical Read Timing**

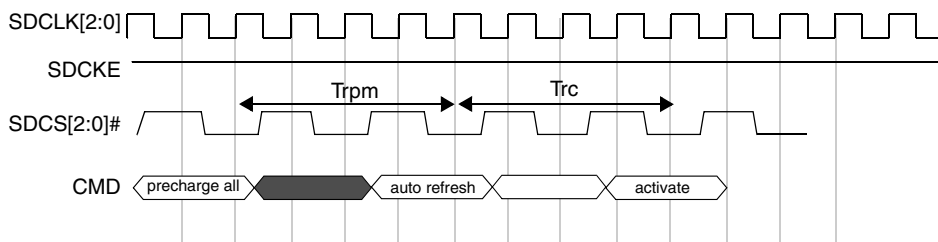


The above timing represents the following:

- 1)  $T_{ras} = 4$  (5 SDRAM clock cycles)
- 2)  $T_{rp} = 0$  (1 SDRAM clock cycles)
- 3)  $T_{rcd} = 1$  (2 SDRAM clock cycles)
- 4)  $T_{mr} = 0$  (1 SDRAM clock cycles)

The above timing is presented to concisely display the different SDRAM timing parameters. The functional bus behavior may differ from that displayed.

**Figure 3-2. SDRAM Typical Write Timing**



This example assumes that all SDCLK ranks ([2:0]) are enabled.

The above timing represents the following:

- 1)  $T_{rpm} = 3$  (4 SDRAM clock cycles)
- 2)  $T_{rc} = 3$  (4 SDRAM clock cycles)

**Figure 3-3. SDRAM Refresh Timing**

### 3.1.4 SDRAM Hardware Considerations

Table 3-3 shows the signals associated with the SDRAM interface.

**Table 3-3. SDRAM Signals**

Signal	Input/Output	Description
SDA[12:0]	O	Address Outputs: A0-A12 are driven during the ACTIVE command (row-address A0-A12) and READ/WRITE command to select one location out of the memory array in the respective bank. The address outputs also provide the opcode during a LOAD MODE REGISTER command.
SDBA[1:0]	O	Bank Address Outputs: SDBA1 and SDBA0 define to which bank the ACTIVE, READ, WRITE or PRECHARGE command is being applied. The SDBA signal values are programmed in <code>mem_sdwrmdn[BA]</code> .
SDD[31:0]	IO	SDRAM data bus
SDQM[3:0]#	O	Input/Output Mask: SDQM is a mask signal for write accesses and an output enable signal for read accesses. SDQM0 masks SDD[7:0], SDQM1 masks SDD[15:8], SDQM2 masks SDD[23:16], SDQM3 masks SDD[31:24].
SDRAS#	O	Command Outputs.
SDCAS#	O	SDRAS#, SDCAS# and SDWE# (along with SDCSn#) define the command being sent to the SDRAM rank.
SDWE	O	
SDCLK[2:0]	O	Clock output corresponding to each of the three chip selects. Clock speed is 1/2 SBUS frequency when corresponding SDCSn# is set to SDRAM or SyncFlash, 1/4 SBUS frequency when corresponding SDCSn# is set to SMROM.
SDCS[2:0]#	O	Programmable chip selects (3 ranks)
SDCKE	O	Clock enable for SDRAM
SMROMCKE	O	Synchronous Mask ROM Clock Enable. This signal must be pulled high if the system is booting from SMROM.  Muxed with GPIO[6]. If ROMSEL and ROMSIZE are configured to boot from Synchronous Mask ROM, SMROMCKE will control the pin out of reset, else GPIO[6] will control the pin out of reset.

## 3.2 Static Bus Controller

The static bus controller provides a general purpose interface to a variety of external peripherals and memory devices. Each of the four static bus chip selects may be programmed to support standard Flash memory, ROM, Page Mode Flash/ROM, SRAM, I/O peripherals, PCMCIA/Compact Flash devices, or an LCD controller. Because of the similarity of Compact Flash and PCMCIA, references to PCMCIA should be taken as applicable to Compact Flash except where noted.

The Au1100 processor allows control of different device types by reconfiguring what control signals chip select *n* manages based on how the device type field (DTY) is encoded in the **mem\_stcfgn** register. All device types use the same address and data bus signals, RAD[31:0] and RD[31:0].

Descriptions of all device types are provided in Section 3.2.2 "Static RAM, I/O Device and Flash Device Types" on page 61, Section 3.2.3 "PCMCIA/Compact Flash Device Type" on page 64, and Section 3.2.4 "LCD Controller Device Type" on page 70.

A read to the static bus causes a 32-bit access. This can cause a potential problem with volatile devices because a single 16-bit read results in two 16-bit reads on the external bus.

Chip selects may be programmed for fixed access times or an external wait signal may be used to provide a variable delay per access.

The static bus controller is a synchronous device, with timing derived from the System Bus (SBUS) clock. However, an external clock (LCLK) to reference the control signals is available only in synchronous mode.

### 3.2.1 Static Controller Programming Model

The properties of each static controller chip select are determined by a set of registers. All registers in the Static Controller block are located off of the base address shown in Table 3-1 on page 43. Table 3-4 shows the registers and offsets for the static bus controller.

After modifying the configuration of a chip select, software must issue a **SYNC** instruction before write accesses to the chip select are allowed.

**Table 3-4. Static Bus Controller Configuration Registers**

Offset (Note 1)	Register Name	Description
0x1000	mem_stcfg0	Configuration for RCS0#
0x1004	mem_sttime0	Timing parameters for RCS0#
0x1008	mem_staddr0	Address region control for RCS0#
0x1010	mem_stcfg1	Configuration for RCS1#
0x1014	mem_sttime1	Timing parameters for RCS1#
0x1018	mem_staddr1	Address region control for RCS1#
0x1020	mem_stcfg2	Configuration for RCS2#
0x1024	mem_sttime2	Timing parameters for RCS2#
0x1028	mem_staddr2	Address region control for RCS2#
0x1030	mem_stcfg3	Configuration for RCS3#
0x1034	mem_sttime3	Timing parameters for RCS3#
0x1038	mem_staddr3	Address region control for RCS3#

Note 1. See Table 3-1 on page 43 for base address.

### 3.2.1.1 Static Bus Configuration Registers

The static bus configuration registers (**mem\_stcfgn**) configure the basic properties of each chip select. Support is included for static RAM, Flash, ROM, PCMCIA, LCD, and other types of I/O devices.

When programming a chip select as an I/O, LCD, or PCMCIA device the address comparison mask will expect an address with the upper nibble set as shown in Table 3-5 on page 56 for the different device types. The TLB must be set up accordingly to map addresses to the memory region captured by the associated chip select.

For example, to configure the TLB for use with an LCD controller, bits 29:26 of CoProcessor register Entry Lo must be 0b1110 (in addition to the other steps necessary to set up the TLB). These bits represent address bits 35:32 of the physical address which must be 0xE in order for the address to match successfully when a chip select is enabled as an LCD device.

Since the RAM and Flash have an upper nibble of zero, it is not necessary to use the TLB to access devices set up with these types.

**mem\_stcfg0** **Offset = 0x1000**

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
										AS	S	DE	PH	V	TA	DIV			BV	AV			BE	TS	EW	H	BS	PM	RO	DTY		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Rs	0	0	0	0	1	1

<b>mem_stcfg1</b>	<b>Offset = 0x1010</b>
<b>mem_stcfg2</b>	<b>Offset = 0x1020</b>
<b>mem_stcfg3</b>	<b>Offset = 0x1030</b>

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
										AS	S	DE	PH	V											BE	TS	EW	H	BS	PM	RO	DTY	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	Name	Description	R/W	Default
31:22	—	Reserved, should be cleared.	R	0
21	AS	Setup address before output enable on reads. The setup duration is programmed in <b>mem_sttimen</b> [T1, T0] and is shown as Tcs_oe in Figure 3-4 on page 62. 0 Do not setup address. 1 Setup address.	R/W	0
20	S	Synchronous mode: when this bit is set all static bus signals are synchronized to LCLK; values in the <b>mem_sttimen</b> register are programmed in LCLK clock resolution. Note: Synchronous mode cannot be used when configured for PCMCIA.	R/W	0
19	DE	Deassert chip select, output enable, write enable, byte enable, read indicators LRD[1:0] and write indicators LWR[1:0] between each beat during bursts. The deassert time is programmable via <b>mem_sttimen</b> [Tcsoff]. Note: A one-word (32-bit) transfer to a 16-bit bus is treated as a burst.	R/W	0
18	PH	Block phantom mode. Negate the chip select, the output enable, and the write enable signals when RBE[1:0]# are not asserted. PH affects only 16-bit mode. 0 RBE[3:0]# are asserted only for valid bytes. 1 RBE[3:0]# are asserted for all bytes.	R/W	0
17	V	Volatile. For single word reads, RBE[3:0]# are asserted for all bytes. This is valid for both 16- and 32-bit modes. 0 RBE[3:0]# are asserted only for valid bytes. 1 RBE[3:0]# are asserted for all bytes. Note: Block phantom mode overrides volatile; that is, V is valid only when (PH = 0).	R/W	0
16	TA	Tcsh application. 0 Apply Tcsh after reads only. 1 Apply Tcsh after both writes and reads. This bit is a global attribute and is present only in <b>mem_stcfg0</b> .	R/W	0

Bits	Name	Description	R/W	Default
15:13	DIV	Adjusts the divisor for the LCLK output clock. The clock frequency is set by $LCLK = (SBus\ Clock / 2) / (DIV + 1)$ Note: LCLK must not exceed 33.33 MHz. This bit is a global attribute and is present only in <b>mem_stcfg0</b> .	R/W	0
12	BV	Burst size visible. When this bit is set the burst size for static transfers will be output for chip selects not configured as LCD or PCMCIA. The burst size output is one less than the number of 32-bit words to be transferred. For 16-bit chip selects twice as many beats will occur. The mapping of the burst size to pins is shown in Table 3-6 on page 56. This bit is a global attribute and is present only in <b>mem_stcfg0</b> .	R/W	0
11	—	Reserved, should be cleared.	R/W	0
10	AV	Address visible. Setting this bit will place the address for all internal accesses to the SBus on the static address bus. This is intended to be used as a debug aid and should not be used during normal operation as it will increase system power usage. This bit is a global attribute and is present only in <b>mem_stcfg0</b> .	R/W	0
9	BE	Endianness. 0 Little Endian 1 Big Endian Program this bit to match the endianness of the processor. This bit should not be set for PCMCIA.	R/W	0
8	TS	Time scale for chip select timing parameters. 0 Do not scale the timing parameters. 1 Multiply the timing parameters by a factor of four. This option allows for longer access times.	R/W	0
7	EW	When the EW bit is set the EWAIT# input is allowed to stretch the bus access time. The EW bit does not apply to chip selects operating in LCD or PCMCIA mode because they have different wait mechanisms.	R/W	0
6	H	Half Bus. Selects the data bus width for the chip select. 0 32-bit bus 1 16-bit bus using bits 15:0 of the data bus. For PCMCIA device type, clear this bit. For LCD device type, set this bit.	R/W	0, except for <b>mem_stcfg0</b> where the default value is determined by ROMSEL and ROMSIZE out of reset. See Table 8-1 on page 225.
5	BS	Burst Size for Page Mode Accesses. Selects the burst size for page mode accesses. Valid only in page mode (PM=1). 0 4 beats 1 8 beats	R/W	0
4	PM	If the PM bit is set the chip select will operate in page mode. This allows quick access to sequential locations in memory. Page mode applies only to reads. See Section 3.2.5.1 "Page Mode Transfers" on page 72.	R/W	0
3	RO	If the RO bit is set the chip select will operate in read only mode. This will inhibit the generation of write cycles to the chip select. Any attempt to write to the address region controlled by a read only chip select will be ignored.	R/W	0
2:0	DTY	Device type. Selects the type of device controlled by the static controller chip select. A list of device types and encodings is shown in Table 3-5. Programming multiple chip selects as LCD or PCMCIA is illegal. Only one of each is supported.	R/W	0 (SRAM), except for <b>mem_stcfg0</b> where the default value is 3 (Flash).

**Table 3-5. Device Type Encoding**

<b>DTY</b>	<b>Chip Select Function</b>	<b>PFN[35:32] (upper nibble of physical address)</b>	<b>Reference</b>
0	Static RAM	0x0	Section 3.2.2
1	I/O Device	0xD	Section 3.2.2
2	PCMCIA Device/Compact Flash	0xF	Section 3.2.3
3	Flash Memory	0x0	Section 3.2.2
4	LCD Device (RCS2# only)	0xE	Section 3.2.4
5–7	Reserved		

**Table 3-6. Burst Size Mapping**

<b>Signal</b>	<b>Pin</b>
burst_size[2]	LWR0#
burst_size[1]	LRD1#
burst_size[0]	LRD0#



### 3.2.1.2 Static Timing Registers

The static timing registers allow software to control the timing of each phase of a static bus access. The names of the timing parameters correspond directly to timing parameters shown on the timing diagrams.

All timing parameters are expressed as a number of clock cycles. Which clock base is used depends on the interface:

- For asynchronous (**mem\_stcfgn**[S]=0) chip selects, the clock is the SBUS clock.
- For synchronous (**mem\_stcfgn**[S]=1) chip selects, the clock is LCLK.

The actual number of clocks for each timing parameter ( $T_{\text{parameter}}$ ) is shown in Table 3-7. Note that the timing behavior for Tcsh is different and is shown in Table 3-8.

**Table 3-7. Actual Number of Clocks for Timing Parameters (Except Tcsh)**

Device Type	TS = 0	TS = 1
Static RAM, I/O, Flash	$T_{\text{parameter}} + 1$	$(4 * T_{\text{parameter}}) + 1$
PCMCIA Device/Compact Flash	$T_{\text{parameter}} + 2$	$(4 * T_{\text{parameter}}) + 2$

**Table 3-8. Actual Number of Clocks for Tcsh**

Tcsh Value	Synchronous ( <b>mem_stcfg</b> [S] = 1)		Asynchronous ( <b>mem_stcfg</b> [S] = 0)	
	TS = 0	TS = 1	TS = 0	TS = 1
0000	2	2	3	3
0001	2	4	3	6
0010	2	6	6	12
0011	4	8	6	15
0100	4	10	6	18
0101	4	12	9	24
0110	4	14	9	27
0111	6	16	9	30
1000	6	18	12	36
1001	6	20	12	39
1010	6	22	12	42
1011	8	24	15	48
1100	8	26	15	51
1101	8	28	15	54
1110	8	30	18	60
1111	10	33	18	63

**mem\_sttime0 (I/O, Flash, SRAM config)**

Offset = 0x1004

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
T1		Twcs		Tcsh			T0	Tcsoff		Twp				Tcsw			Tpm		Ta													
Def.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1

**mem\_sttime1 (I/O, Flash, SRAM config)**

Offset = 0x1014

**mem\_sttime2 (I/O, Flash, SRAM and LCD config)**

Offset = 0x1024

**mem\_sttime3 (I/O, Flash, SRAM config)**

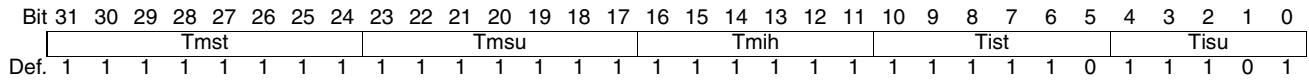
Offset = 0x1034

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1		Twcs		Tcsh			T0	Tcsoff		Twp				Tcsw			Tpm		Ta												
Def.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W	Default
31	T1	Most significant bit of Tcs_oe[1:0]. Tcs_oe represents the number of clocks needed to setup the address before asserting output enable. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0
30:28	Twcs	This field specifies the required chip select hold time after a write pulse. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x3
27:24	Tcsh	Chip select hold-off. Specifies the minimum number of cycles that the chip select must remain inactive between accesses. The next transaction through the static bus controller is held off until the Tcsh parameter is satisfied. If this next access falls within another chip select's memory region, the new set of timing parameters associated with the controlling chip select take effect once the new transaction begins.  Note that the SBUS can arbitrarily extend the time between accesses for internal operations. This can add up to about five additional clocks to the programmed time.  See Table 3-8 on page 57 for the actual number of clock cycles.	R/W	0xF
23	T0	Least significant bit of Tcs_oe[1:0]. Tcs_oe represents the number of clocks to setup the address before asserting output enable. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	1
22:20	Tcsoff	This field specifies the required number of cycles that the chip select, output enable, write enable, byte enable, read indicators LRD[1:0] and write indicators LWR[1:0] must remain deasserted between beats. Valid only when <b>mem_stcfgn[DE]</b> is set. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x7
19:14	Twp	This field specifies the duration of the write enable. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x3F
13:10	Tcsw	Chip select to write. Defines the delay from the assertion of chip select until the write strobe and byte enables are asserted. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0xF
9:6	Tpm	This field determines the number of cycles required from a burst address change until read data is valid if the PM bit is set in the <b>mem_stcfgn</b> register. See Table 3-7 on page 57 for the actual number of clock cycles. Ta determines the access time for the <i>first</i> beat of each burst.	R/W	0xF
5:0	Ta	The Ta parameter determines the number of cycles required for the assertion of the chip select. For page mode accesses Ta determines the access time up to the first beat of each burst, or the first beat after a page mode wrap. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x3F, except for <b>mem_sttime0</b> where the default value is 0x1D.

**mem\_sttime0 (PCMCIA config)**

Offset = 0x1004



**mem\_sttime1 (PCMCIA config)**

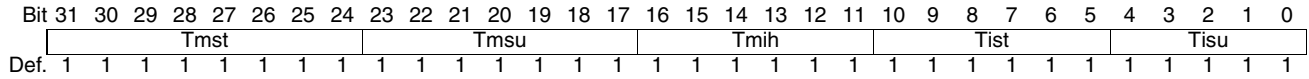
Offset = 0x1014

**mem\_sttime2 (PCMCIA config)**

Offset = 0x1024

**mem\_sttime3 (PCMCIA config)**

Offset = 0x1034



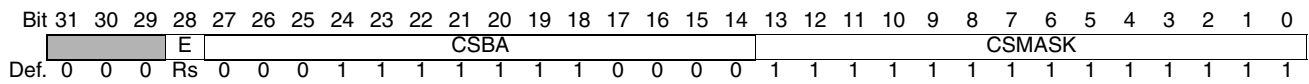
Bits	Name	Description	R/W	Default
31:24	Tmst	This field specifies the strobe width during memory accesses to PCMCIA chip selects.  The timing duration depends on the time-scale option <b>mem_stcfgn[TS]</b> : When TS=0, (Tmst + 2) is the number of cycles to the end of the strobe; however, the read occurs at (Tmst + 1). When TS=1, [(4 * Tmst) + 2] is the number of cycles to the end of the strobe; however, the read occurs at [(4 * Tmst) + 1].	R/W	0xFF
23:17	Tmsu	This field specifies the setup time from chip select to strobe during memory accesses to PCMCIA chip selects. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x7F
16:11	Tmih	This field specifies the hold time for address, data, and chip selects from the end of the strobe for both memory and I/O cycles to PCMCIA chip selects. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x3F
10:5	Tist	This field specifies the strobe width for I/O accesses for a chip select configured for PCMCIA.  The timing duration depends on the time-scale option <b>mem_stcfgn[TS]</b> : When TS=0, (Tmst + 2) is the number of cycles to the end of the strobe; however, the read occurs at (Tmst + 1). When TS=1, [(4 * Tmst) + 2] is the number of cycles to the end of the strobe; however, the read occurs at [(4 * Tmst) + 1].	R/W	0x3F, except for <b>mem_sttime0</b> where the default value is 0x3E.
4:0	Tisu	This field specifies the setup time from chip select to strobe during I/O accesses for PCMCIA. See Table 3-7 on page 57 for the actual number of clock cycles.	R/W	0x1F, except for <b>mem_sttime0</b> where the default value is 0x1D.

**3.2.1.3 Static Chip Select Address Configuration Registers (mem\_staddrn)**

The static memory chip-select address configuration registers (**mem\_staddrn**) assign an address range for each chip select. As shown below, each register contains a base address, an address comparison mask, and an enable bit.

**mem\_staddr0 - RCS0# Address Configuration**

Offset = 0x1008



**mem\_staddr1 - RCS1# Address Configuration**

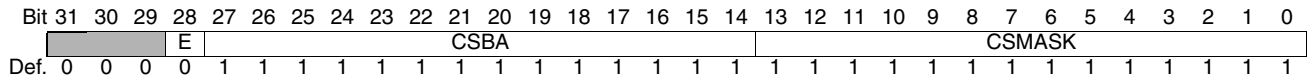
Offset = 0x1018

**mem\_staddr2 - RCS2# Address Configuration**

Offset = 0x1028

**mem\_staddr3 - RCS3# Address Configuration**

Offset = 0x1038



Bits	Name	Description	R/W	Default
31:29	—	Reserved, should be cleared.	R	0
28	E	Enable. 0 Chip select is disabled. 1 Chip select is enabled.	R/W	0, except for <b>mem_staddr0</b> (Note 1)
27:14	CSBA	Chip select base address. Specifies bits 31:18 of the physical base address for this chip select. The upper nibble of the chip select address is determined by the device type selected in <b>mem_stcfgn[DTY]</b> . The lower bits of the base address are zeros.	R/W	0x3FFF, except for <b>mem_staddr0</b> where the default value is 0x7F0.
13:0	CSMASK	Chip select address mask. Specifies bits 31:18 of the address comparison mask used to decode this chip select. (The upper nibble of the address comparison mask is determined by <b>mem_stcfgn[DTY]</b> . The lower bits of the mask are zeros.)	R/W	0x3FFF

Note 1. The enable (E) bits for chip selects RCS1#, RCS2#, and RCS3# are automatically cleared (disabled) coming out of a runtime or hardware reset. For RCS0#, however, the reset value of the E bit depends on ROMSEL: Holding ROMSEL low indicates that ROM should be used for the boot vector (and RCS0#'s E bit is set); otherwise, RCS0# is disabled. See also Section 8.3 "Boot" on page 225.

Once enabled, a chip select is asserted when the following condition is met:

$$(phys\_addr \& \text{addr\_mask}) == \text{base\_addr}$$

where:

- phys\_addr: 36-bit physical address output on the internal SBUS (from the TLB for memory-mapped regions)
- addr\_mask: address comparison mask taken from CSMASK
- base\_addr: chip select base address taken from CSBA

The chip select regions must be programmed so that each chip select occupies a unique area of the physical address space. Programming overlapping chip select regions results in undefined operation.

### 3.2.2 Static RAM, I/O Device and Flash Device Types

This section describes the static RAM interface which is implemented when the device type (**mem\_stcfn**[DTY]) is programmed to 0, 1 or 3. (Section 3.2.1.1 "Static Bus Configuration Registers" on page 54)

The static RAM, I/O device and Flash device types are all similar. The I/O device type is identical to the static RAM type except that it expects the upper nibble of the system address (bits 35:32) to be 0xD. The only difference between the Flash device type and the static RAM device type is that the Flash timing allows for a chip select hold time after a write pulse using **mem\_sttmem**[Twcs].

Other than these differences, the static RAM, I/O device and Flash device types share the same timing and control signals. The control signals are shown in Table 3-9.

**Table 3-9. Static RAM, I/O Device and Flash Control Signals**

Signal Name	Input/Output	Description
RAD[31:0]	O	Address bus
RD[31:0]	IO	Data bus
RBE[3:0]#	O	Byte enables: RBE0# corresponds to RD[7:0]. RBE1# is for RD[15:8]. RBE2 #is for RD[23:16]. RBE3# is for RD[31:24].
RWE#	O	Write enable
ROE#	O	Output enable
RCS[3:0]#	O	Programmable Chip Selects (4 banks). RCS[n]# is not used when configured as a PCMCIA device.
EWAIT#	I	Can be used to stretch the bus access time when enabled through <b>mem_stcfn</b> [EW].

#### 3.2.2.1 Static Memory Timing

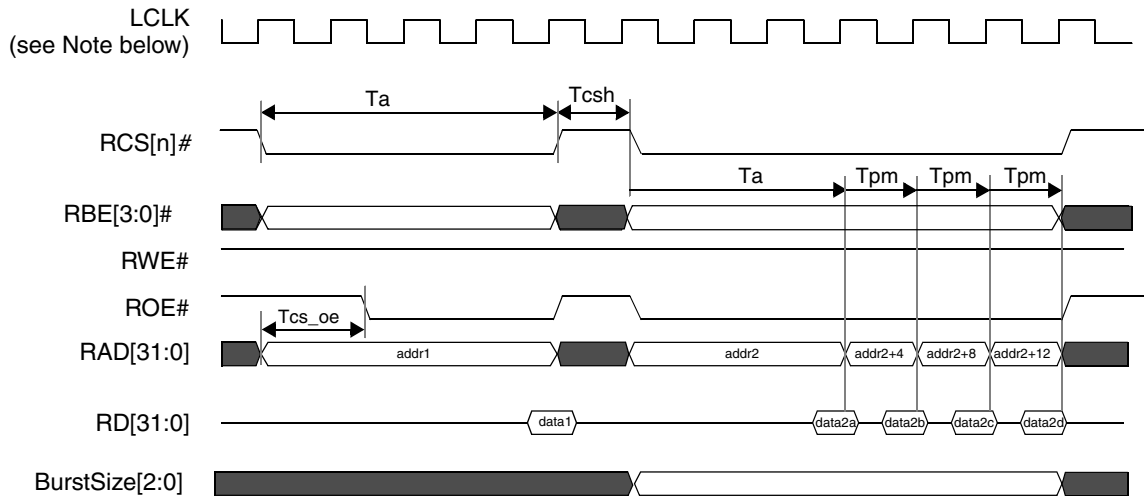
The following figures show static memory timing. Figure 3-4 on page 62 illustrates static memory read timing, and Figure 3-6 on page 63 illustrates static memory write timing. The EWAIT# timing diagrams are presented to show how EWAIT# will hold the cycle past  $T_a$  for reads and  $T_{wp}$  for writes.

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 11.0 "Electrical and Thermal Specifications" on page 261. (See Section 11.4.2 "Asynchronous Static Bus Controller Timing" on page 269 and Section 11.4.3 "Synchronous Static Bus Timing" on page 271.)

Timing parameters do not take into account SBUS overhead which may add inter-access delays. These delays are dependent on system design and are affected by the number of bus masters and the ability of other devices to hold the bus.

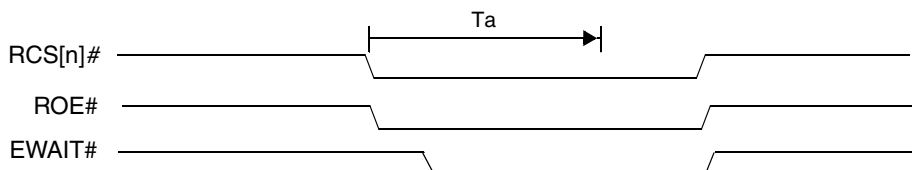
**Read Timing**

Read accesses to the static bus always retrieve 32-bits of data. As such, all four byte enables are asserted during the 32-bit access or the two 16-bit beats. The control signals (RCS[n]#, ROE#, and RBE[1:0]#) span both accesses. The only signal that changes state to indicate the start of the second beat is RADDR[1].



**Note:** The external clock LCLK is available only in synchronous mode (`mem_stcfg[S]=1`). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.1 "Static Bus Configuration Registers" on page 54.

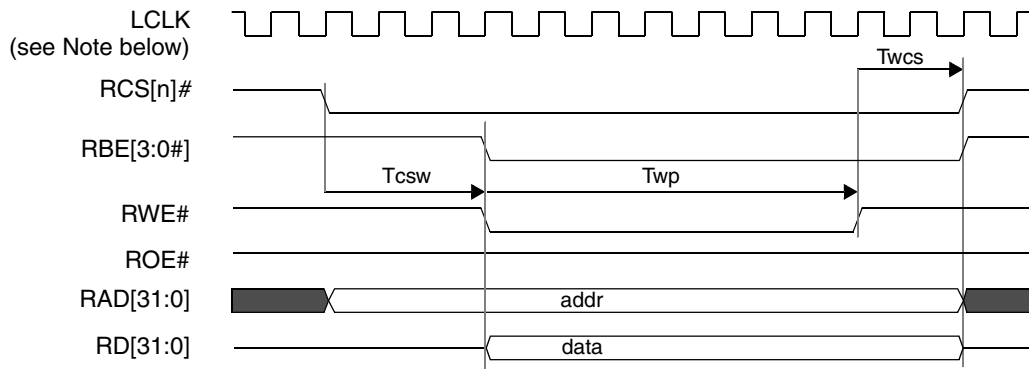
**Figure 3-4. Static Memory Read Timing (Single Read Followed by Burst)**



**Figure 3-5. Static Memory Read EWAIT# Timing**

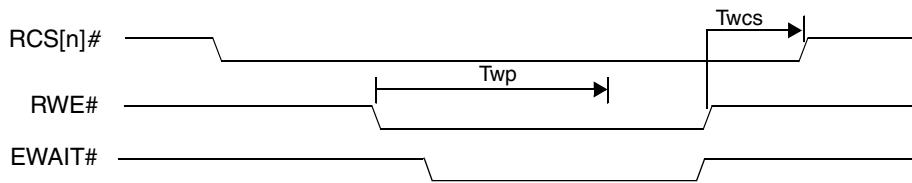
**Write Timing**

The timing diagrams below show the static bus write timing for I/O and SRAM device types. Figure 3-6 shows a single 32-bit write on a 32-bit chip select.



**Note:** The external clock LCLK is available only in synchronous mode (`mem_stcfg[S]=1`). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.1 "Static Bus Configuration Registers" on page 54.

**Figure 3-6. Static Memory Write Timing**



**Figure 3-7. Static Memory Write EWAIT# Timing**

### 3.2.3 PCMCIA/Compact Flash Device Type

Because of the similarity of Compact Flash and PCMCIA, references to PCMCIA should be taken as applicable to Compact Flash except where noted. The PCMCIA peripheral is designed to the PCMCIA2.1 specification—but only for the bus transactions as described in this section.

The Au1100 processor provides a PCMCIA host adapter when the device type is programmed for PCMCIA. The static controller interface provides the required bus signals necessary to control a PCMCIA interface. Auxiliary signals, such as card detect and voltage sense, can be implemented with GPIOs if desired.

The PCMCIA host interface adapter will support memory, attribute and I/O transactions. External logic can be added to support DMA transfers. The Au1100 processor supports only 8- and 16-bit load and store instructions (byte and halfword instructions) to PCMCIA devices. 32-bit accesses are not supported.

The PCMCIA interface provides control signals defined for PCMCIA devices. If two devices are required then external logic must be added to allow for both cards to share the bus. Note that when a chip select is programmed as a PCMCIA device that the associated RCS[n]# is not used.

The PCMCIA interface occupies a 36-bit address space with the upper 4 bits equal to 0xF. The TLB is required to generate addresses that will activate a chip select with a device type of "PCMCIA".

I/O, Memory and Attribute spaces are differentiated by addr[31:30]. Table 3-10 shows the mapping.

**Table 3-10. PCMCIA Memory Mapping**

Physical Address	PCMCIA Mapping
0xF 0xxx xxxx	I/O
0xF 4xxx xxxx	Attribute Memory
0xF 8xxx xxxx	Memory

Note: Each of the PCMCIA physical address spaces have a maximum size of 64 MB. Any access beyond the 64 MB space will alias back into the defined region.

Table 3-11 enumerates the signals to support the PCMCIA interface.

**Table 3-11. PCMCIA Interface Signals**

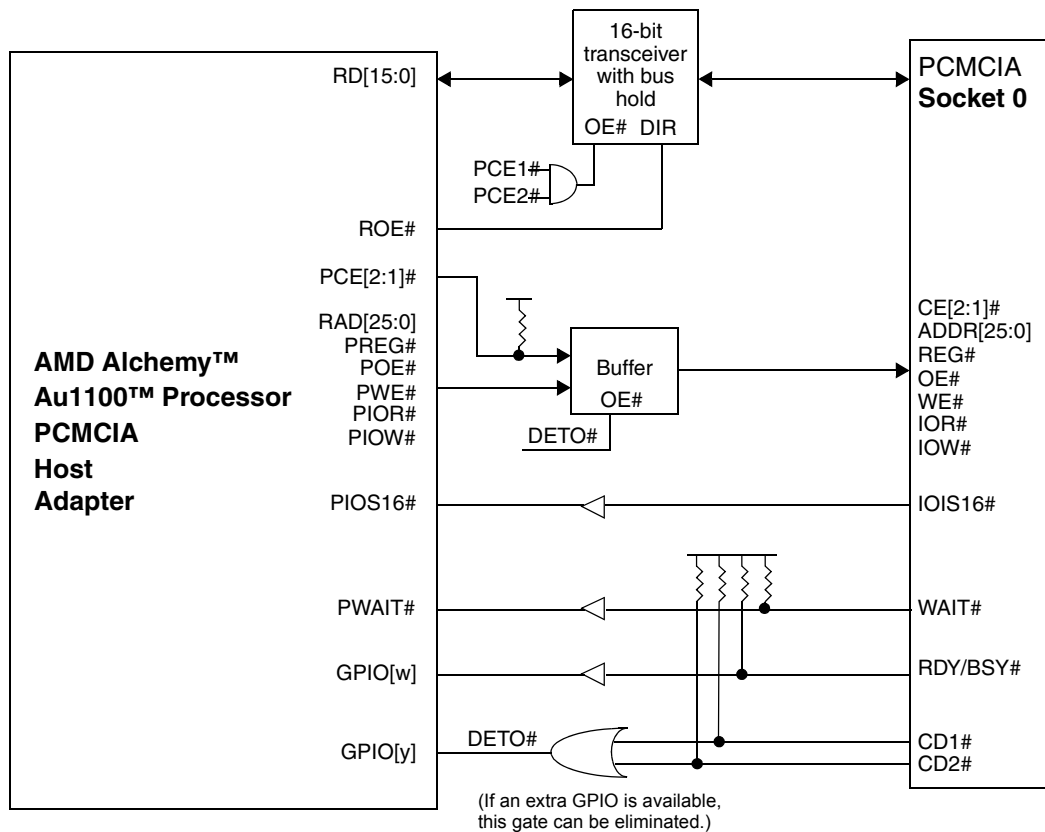
Signal Name	Input/Output	Description
RAD[31:0]	O	Address Bus.
RD[15:0]	IO	Data Bus.
PREG#	O	When this signal is asserted card access is limited to attribute memory when a memory access occurs and to I/O ports when an I/O access occurs. Muxed with GPIO[204] which controls the pin out of hardware reset, runtime reset and Sleep.
PCE[2:1]#	O	Card Enables. Muxed with GPIO[206:205] which controls the pins out of hardware reset, runtime reset and Sleep. Note that the card enables need pull-up resistors.
POE#	O	Memory Output Enable.
PWE#	O	Memory Write Enable. Muxed with GPIO[207] which controls the pin out of hardware reset, runtime reset and Sleep.
PIOR#	O	I/O Read Cycle Indication.
PIOW#	O	I/O Write Cycle Indication.
PWAIT#	I	This signal is asserted by the card to delay completion of a pending cycle. Note that this signal should be tied high through a resistor when the PCMCIA interface is not used.



**Table 3-11. PCMCIA Interface Signals (Continued)**

Signal Name	Input/Output	Description
PIOS16#	I	16-bit Port Select. Note that this signal should be tied high through a resistor when the PCMCIA interface is not used.
ROE#	O	Output Enable - This output enable is intended to be used as a data transceiver control. During a PCMCIA transaction, ROE# remains asserted (low) as configured in the timing registers ( <b>mem_sttmem</b> ) for reads and negated (high) for writes.

Figure 3-8 and Figure 3-9 on page 66 show a one and two card PCMCIA implementation. For the two card implementation RAD26 is used as a card select signal. Both figures assume that the PCMCIA card can be hot swapped at any time—note the use of isolation buffers on the shared bus. If the card is fixed in the system much of the interface logic can be removed. A Compact Flash implementation is very similar to the PCMCIA implementation except that the number of address lines used is fewer.



**Figure 3-8. One Card PCMCIA Interface**

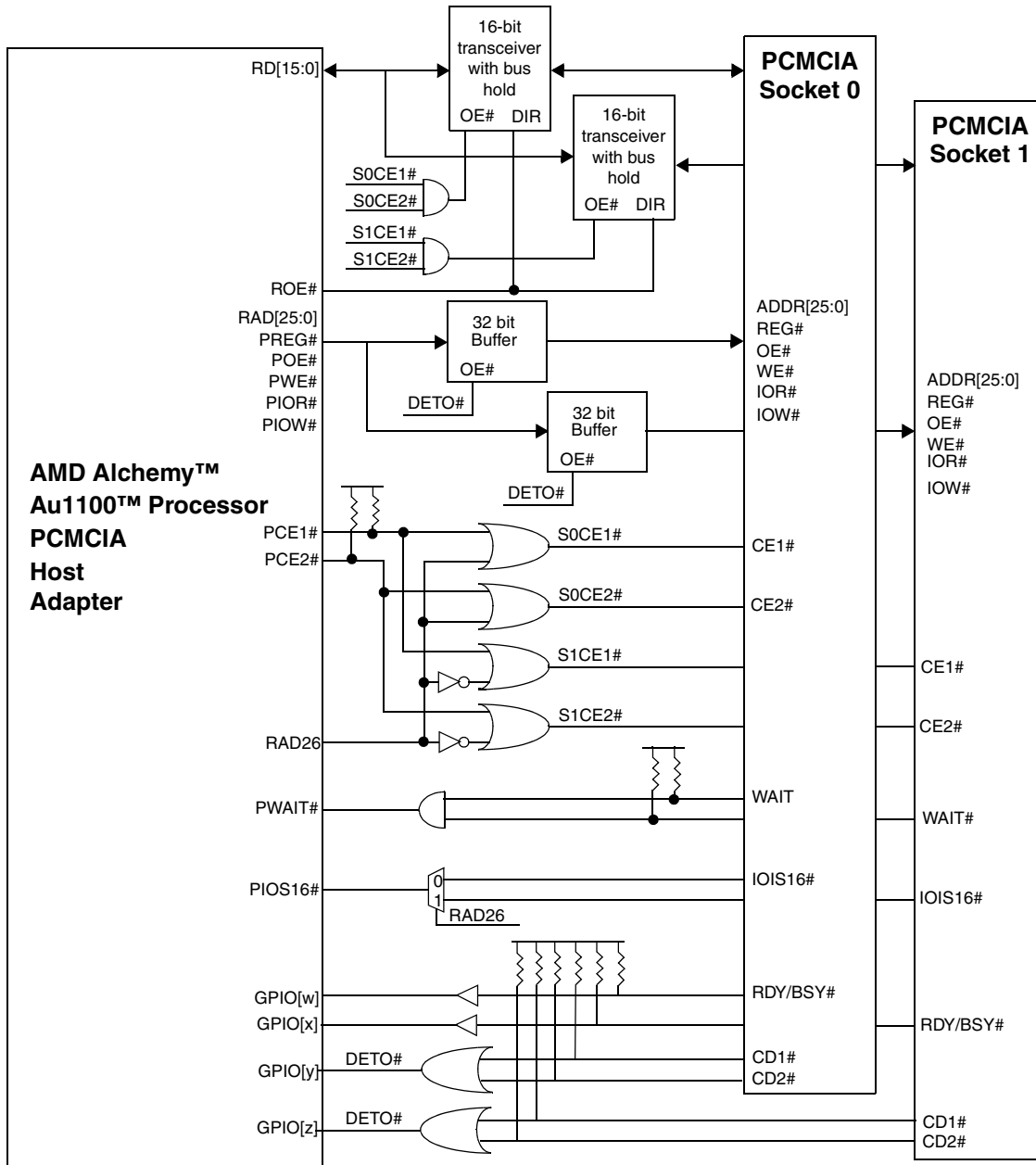


Figure 3-9. Two Card PCMCIA Interface

### 3.2.3.1 PCMCIA/CompactFlash Interface

The figures on the following pages illustrate the functional timing of the PCMCIA interface, including memory read timing, memory write timing, I/O read timing, and I/O write timing. The PWAIT# timing diagrams are presented to show how PWAIT# will hold the cycle past Tmst for memory reads and writes and Tist for I/O reads and writes.

Setup and hold time requirements are presented in Section 11.4.2 "Asynchronous Static Bus Controller Timing" on page 269.

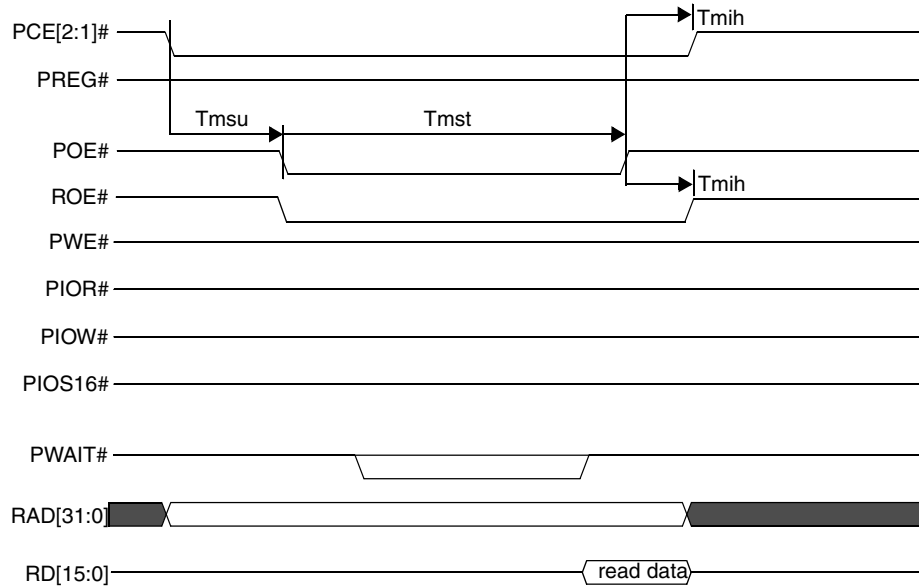


Figure 3-10. PCMCIA Memory Read Timing

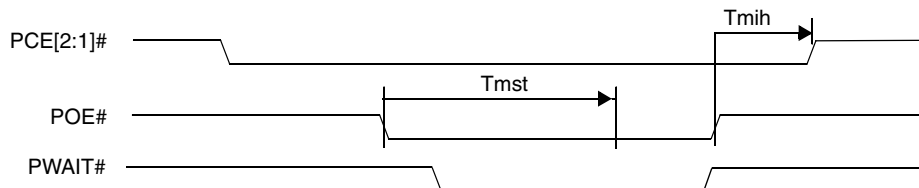


Figure 3-11. PCMCIA Memory Read PWAIT# Timing

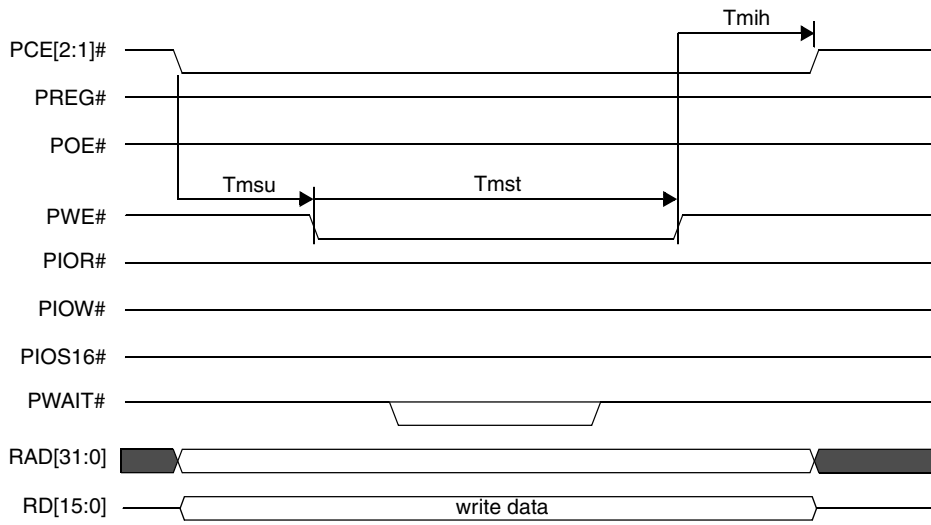


Figure 3-12. PCMCIA Memory Write Timing

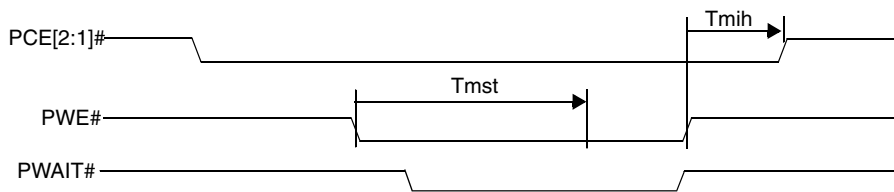


Figure 3-13. PCMCIA Memory Write PWAIT# Timing

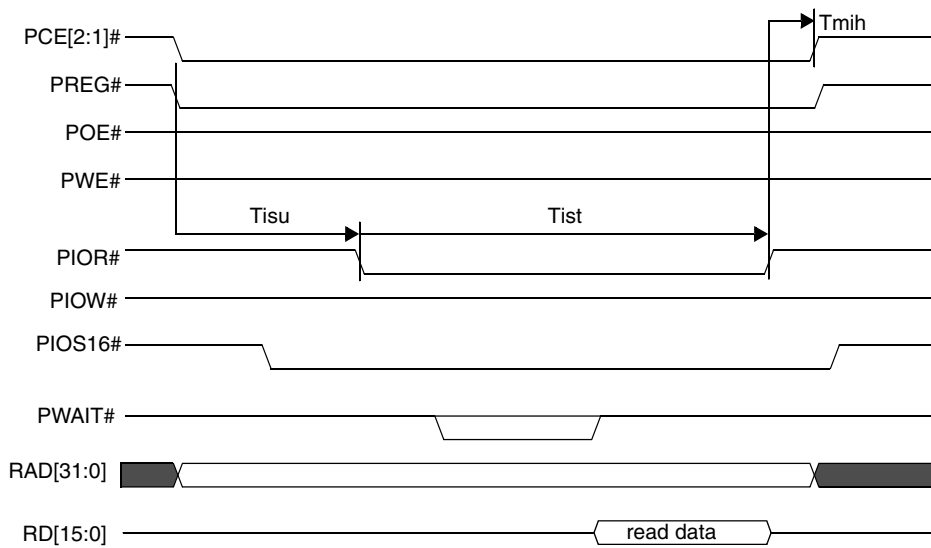


Figure 3-14. PCMCIA I/O Read Timing

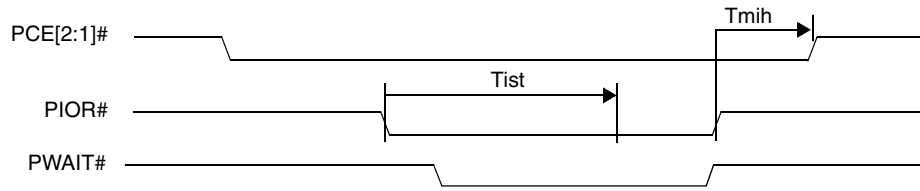


Figure 3-15. PCMCIA I/O Read PWAIT# Timing

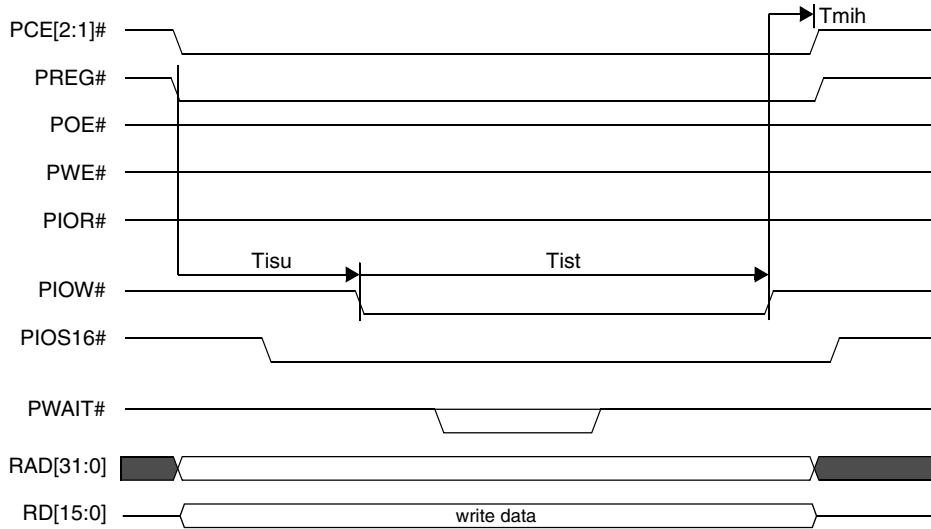


Figure 3-16. PCMCIA I/O Write Timing

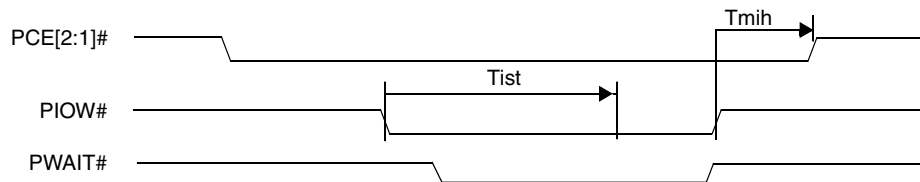


Figure 3-17. PCMCIA I/O Write PWAIT# Timing

### 3.2.4 LCD Controller Device Type

The Au1100 processor provides a LCD controller host adapter when the device type is programmed for an LCD. The static controller interface provides the bus signals necessary to interface to most LCD controllers.

A dedicated clock LCLK is provided for the LCD interface. The LCLK rate is the SBUS rate divided by a factor programmed in `mem_stcfg0[DIV]`; see Section 3.2 "Static Bus Controller" on page 53.

The Au1100 supports 8-, 16-, and 32-bit load and store instructions (byte, halfword, and word instructions) to the LCD controller interface.

The LCD controller occupies 36 bit address space with the upper 4 bits equal to 0xE. The MMU is required to generate addresses that will generate a chip select with a device type of "LCD".

Table 3-12 lists the control signals to support the LCD controller.

**Table 3-12. LCD Controller Interface Signals**

Signal	Input/Output	Description
RAD[31:0]	O	Address bus
RD[15:0]	IO	Data bus
RCS[3:0]#	O	Chip Selects
LCLK	O	Interface Clock
LWAIT#	I	Extend Cycle
LRD[1:0]#	O	Read Indicators. Muxed with GPIO[201:200] which controls the pins out of hardware reset, runtime reset and Sleep.
LWR[1:0]#	O	Write Indicators. Muxed with GPIO[203:202] which controls the pins out of hardware reset, runtime reset and Sleep.

### 3.2.4.1 LCD Controller Interface Timing

The following figures shows the LCD timing. The LWAIT# timing diagrams are presented to show how LWAIT# will hold the cycle past  $T_a$  for memory reads and  $T_{wp}$  for memory writes.

LWAIT timing requirements as well as setup and hold times are presented in Section 11.4.2 "Asynchronous Static Bus Controller Timing" on page 269.

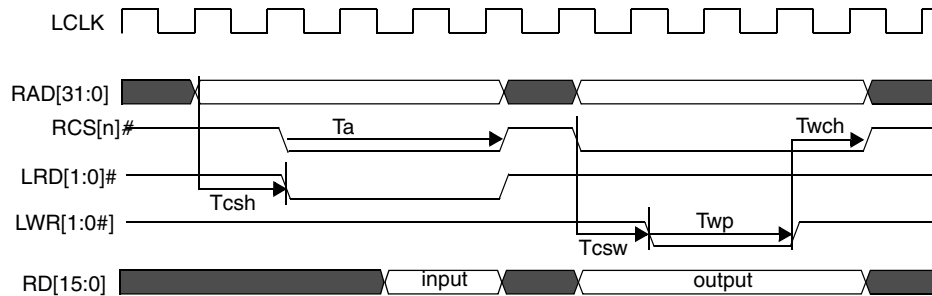


Figure 3-18. LCD Controller Timing

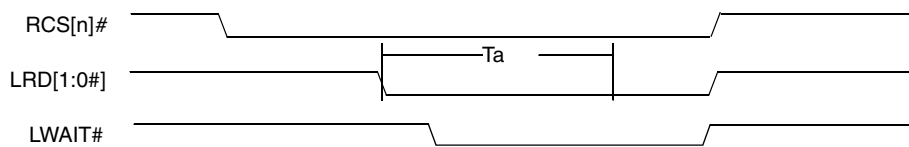


Figure 3-19. LCD Read LWAIT# Timing

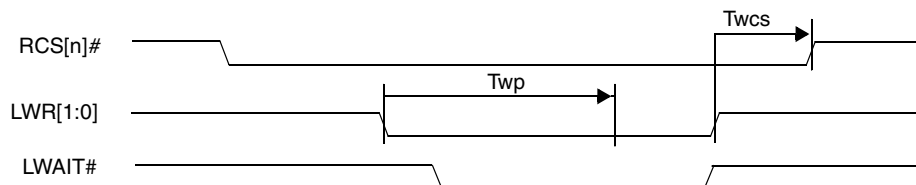


Figure 3-20. LCD Write LWAIT# Timing

### 3.2.5 Static Bus Controller Programming Considerations

#### 3.2.5.1 Page Mode Transfers

The static bus controller provides a page mode for quick read access to sequential locations in memory. Setting **mem\_stcfn[PM]** selects page mode operation for the chip select. The burst size (4 or 8 beats) for page mode transfers is programmed in **mem\_stcfn[BS]**.

Depending on the speed of the external memory device, the system designer can adjust two timing parameters in **mem\_sttmem** for page mode transfers:

- *Ta* is the time from chip select assertion to the first beat of valid data. *Ta* is the time required for the initial access to a peripheral device. *Ta* must allow time for the peripheral device to load its read buffer or activate the next page. Note that the page size depends on the peripheral device.
- *Tpm* is the time between beats.

Figure 3-4 on page 62 shows an example page mode read with the timing parameters *Ta* and *Tpm*.

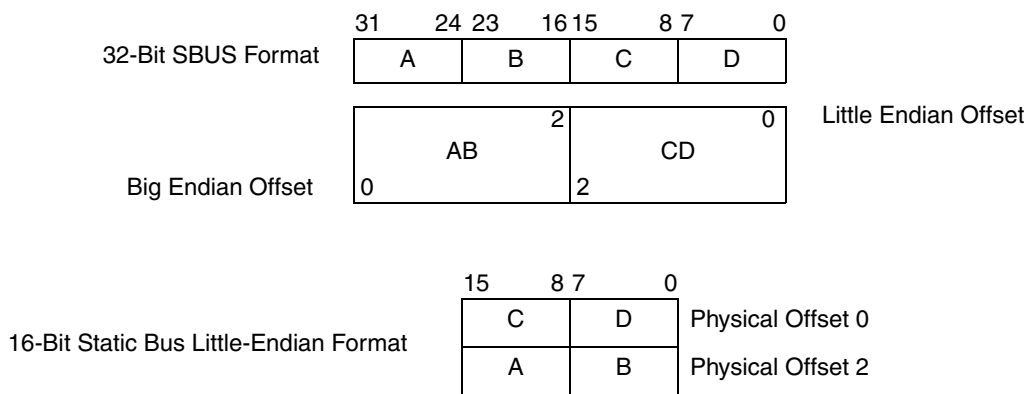
The static bus controller does not check for page boundaries during page mode reads. The addressing is sequential regardless of alignment. An access which crosses a page boundary may return invalid data if *Tpm* does not allow enough time for the external memory device to update its read buffer or activate the next page. If the system designer cannot ensure adequate address alignment to avoid crossing page boundaries, *Tpm* must be long enough to accommodate potential page updates.

In general, page-boundary timing issues do not arise for instruction fetches because they are always accessed first-word-first and therefore are properly aligned. Data fetches, however, may have page-boundary timing issues because they are accessed critical-word-first.

Note that **EWAIT#** can delay only the start of the burst (extend the *Ta* timing). That is, **EWAIT#** cannot be used to account for varying timing between beats (extend the *Tpm* timing) that may occur even for transfers within a page.

#### Halfword Ordering and 16-bit Chip Selects

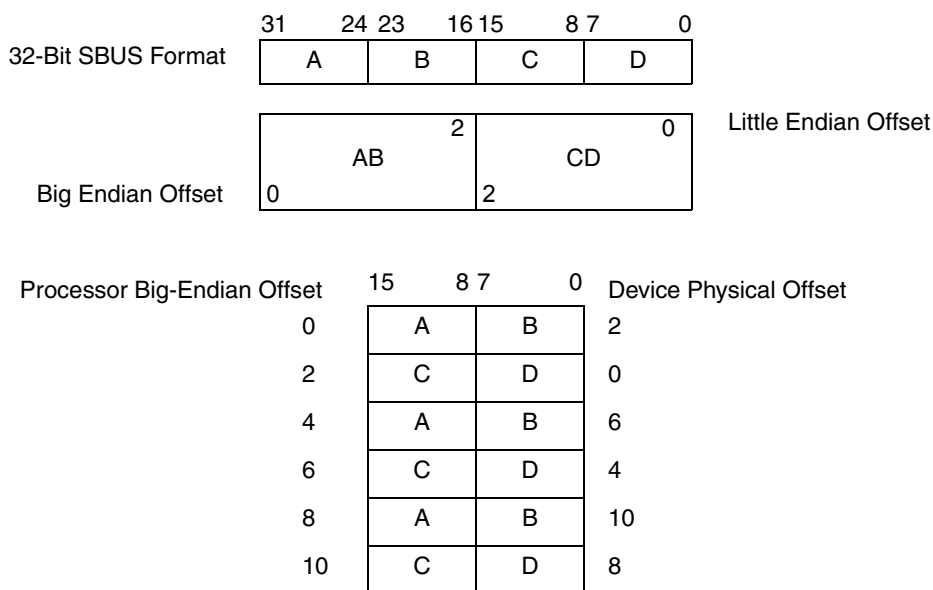
Because the static bus controller is not aware of the endian mode of the Au1 core, potential halfword swapping conflicts can arise. Upon reset, chip selects default to little-endian byte ordering (**mem\_stcfn[BE]** = 0). Figure 3-21 shows the data formats for the 32-bit SBUS and for a little-endian 16-bit chip select.



**Figure 3-21. 16-Bit Chip Select Little-Endian Data Format (Default)**

When a 16-bit chip select is in little-endian mode, the static bus controller accesses the least-significant halfword CD at physical offset 0 and accesses the most-significant halfword AB at physical offset 2. When the Au1 core is also in little-endian mode, the requested Au1 core offsets match the physical offsets of the 16-bit device. That is, the static bus controller and the Au1 core have the same view of memory. However, when the processor core is in big-endian mode, the default ordering of the static bus controller effectively reverses the ordering of the halfwords from what the big-endian Au1 core expects, as shown in Figure 3-22.



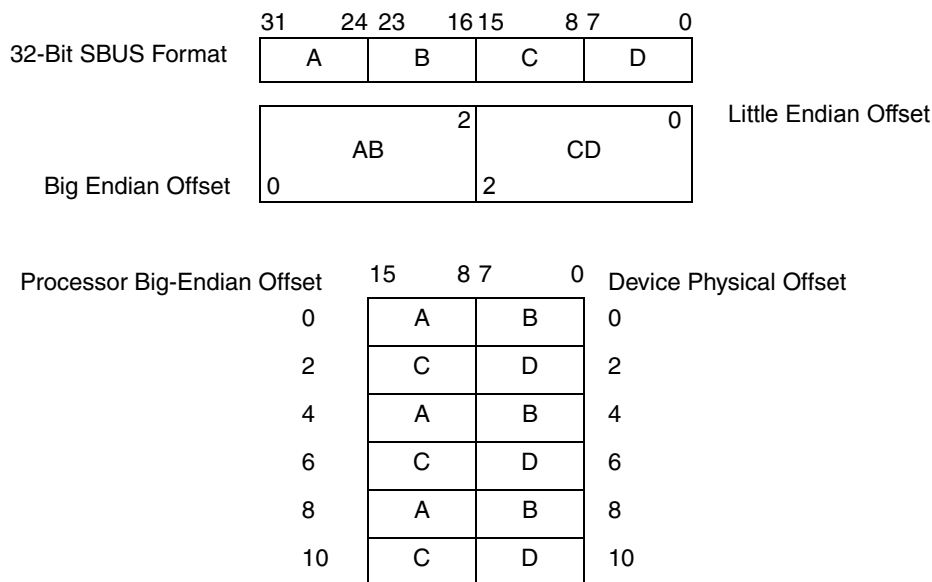


**Figure 3-22. Big-Endian Au1 Core and Little-Endian 16-Bit Chip Select**

For RAM memories, the halfword swapping has no side-effects because reads and writes are consistent. However, for ROM, Flash memories, and peripherals, be aware of the following side effects:

- For ROM and Flash, the memory contents are halfword-swapped throughout the entire 16-bit device memory.
- For Flash and peripherals, the programming register offsets are also halfword-swapped.

To prevent halfword swapping, configure the chip select for big-endian mode (**mem\_stcfg**[BE] = 1) before accessing the memory. (If booting from static memory, see Section 8.3.1 "Endianness and 16-Bit Static Bus Boot" on page 225.) The static bus controller inverts RAD1 for transfers on 16-bit chip selects in big-endian mode, as shown in Figure 3-23 on page 73.



**Figure 3-23. Big-Endian Au1 Core and Big-Endian 16-Bit Chip Select**



# DMA Controller

The Au1100 processor contains an eight-channel DMA controller. Each channel is capable of transferring data between memory and any of the integrated peripherals or between memory and a memory-mapped FIFO through the Static Controller using a GPIO as a request.

Note that memory-to-memory transfers are not supported by the DMA controller. That is, one side of the DMA transfer must have an incrementing address (memory buffer), while the other side must have a *fixed* address (FIFO).

GPIO[4] and GPIO[5] can be programmed to act as external DMA request signals. When configured for this special system function, the pins are labeled as follows:

- GPIO[4] becomes DMA\_REQ0.
- GPIO[5] becomes DMA\_REQ1.

See Section 4.2 "Using GPIO as External DMA Requests (DMA\_REQn)" on page 80 to configure these GPIO signals to act as DMA requests.

## 4.1 DMA Configuration Registers

Each channel of the DMA is configured by a register block. A channel register block contains seven registers. The 36-bit physical base address of the register block for each channel is shown in Table 4-1. Each register block contains the registers shown in Table 4-2 on page 76.

**Table 4-1. DMA Channel Base Addresses**

DMA Channel	Physical Base Address	KSEG1 Base Address	Priority
dma0	0x0 1400 2000	0xB400 2000	0 (highest)
dma1	0x0 1400 2100	0xB400 2100	1
dma2	0x0 1400 2200	0xB400 2200	2
dma3	0x0 1400 2300	0xB400 2300	3
dma4	0x0 1400 2400	0xB400 2400	4
dma5	0x0 1400 2500	0xB400 2500	5
dma6	0x0 1400 2600	0xB400 2600	6
dma7	0x0 1400 2700	0xB400 2700	7 (lowest)

**Table 4-2. DMA Channel Configuration Registers**

Offset (Note 1)	Register Name	Description
0x0000	dma_moderead	Read channel mode register
0x0000	dma_modeset	Set bits in channel mode register
0x0004	dma_modeclr	Clear bits in channel mode register
0x0008	dma_peraddr	Address of peripheral FIFO
0x000C	dma_buf0addr	Starting address of buffer 0
0x0010	dma_buf0size	Transfer size and remaining transfer count for buffer 0
0x0014	dma_buf1addr	Starting address of buffer 1
0x0018	dma_buf1size	Transfer Size and remaining transfer count for buffer 1

Note 1. See Table 4-1 on page 75 for base address.

Table 4-3 shows the different peripherals that are capable of DMA. The device ID, transfer size, and transfer width (device FIFO width) are configurable fields in the **dma\_mode** register. The FIFO address is a physical address whose address should be programmed in the **dma\_peraddr** register and in the DAH field of the **dma\_mode** register.

Enabling multiple DMA channels with the same device ID is undefined.

**Table 4-3. Peripheral Addresses and Selectors**

Peripheral Device	Device ID Select	Device ID	Transfer Size	Device FIFO Width (bits)	FIFO Physical Address
UART 0 Transmit	0	0	Programmable	8	0x0 1110 0004
UART 0 Receive	0	1	Programmable	8	0x0 1110 0000
DMA_REQ0 (GPIO[4])	0	2	Programmable	Programmable	Programmable
DMA_REQ1 (GPIO[5])	0	3	Programmable	Programmable	Programmable
AC97 Transmit	0	4	4	16	0x0 1000 0008
AC97 Receive	0	5	4	16	0x0 1000 0008
UART3 Transmit	0	6	Programmable	8	0x0 1140 0004
UART3 Receive	0	7	Programmable	8	0x0 1140 0000
USB Device Endpoint 0 Receive	0	8	4	8	0x0 1020 0000
USB Device Endpoint 0 Transmit	0	9	4	8	0x0 1020 0004
USB Device Endpoint 1 Transmit	0	10	4	8	0x0 1020 0008
USB Device Endpoint 2 Transmit	0	11	4	8	0x0 1020 000C
USB Device Endpoint 3 Receive	0	12	4	8	0x0 1020 0010
USB Device Endpoint 4 Receive	0	13	4	8	0x0 1020 0014
I <sup>2</sup> S Transmit	0	14	4	Programmable	0x0 1100 0000
I <sup>2</sup> S Receive	0	15	4	Programmable	0x0 1100 0000
SD 0 Transmit	1	0	Programmable	8	0x0 1060 0000
SD 0 Receive	1	1	Programmable	8	0x0 1060 0004
SD 1 Transmit	1	2	Programmable	8	0x0 1068 0000
SD 1 Receive	1	3	Programmable	8	0x0 1068 0004
Reserved	1	4-15	N/A	N/A	N/A

### 4.1.1 DMA Channel Mode Registers

Each DMA channel is controlled by a mode register. The current value of the register can be read from the **dma\_moderead** register but can not be set to an arbitrary value in a single operation. Instead, the configuration register is controlled by two registers: **dma\_modeset** and **dma\_modeclr**.

- The **dma\_modeset** register *sets* bits in the channel mode register when the corresponding bit is written as a one. (Bits written as zero do not affect the corresponding mode bit.)
- The **dma\_modeclr** register *clears* bits in the channel mode register when the corresponding bit is written as a one. (Bits written as zero do not affect the corresponding mode bit.)

The Au1100 processor has been designed to simplify the DMA control process by removing the need for a semaphore to control access to the registers. This is because there is no need to read, modify, write, as there are separate registers for setting and clearing a bit. In this way a function can freely manipulate the DMA channels associated with that function.

An arbitrary value may be written to a field within the register with the following sequence:

```
dma_modeset = new_value & field_mask;
dma_modeclr = ~new_value & field_mask;
```

The Transfer Size and Device Width fields must be programmed to match the FIFO of the peripheral chosen with the DID field according to Table 4-3 on page 76.

For the UART FIFOs the transfer size is programmable. It is the programmers responsibility to insure that the Transfer Size matches the trigger depth set in the UART FIFO control register. See Section 6.7 "UART Interfaces" on page 151 for more information.

For the I<sup>2</sup>S FIFOs the transfer width is programmable. It is the programmers responsibility to insure that the Transfer Width field matches the word size in the I<sup>2</sup>S configuration register and that memory is packed accordingly. See Section 6.6 "I<sup>2</sup>S Controller" on page 146 for more information.

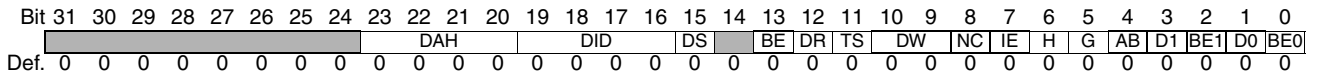
For external DMA using GPIO signals as requests (DMA\_REQn), the system designer must ensure that the Transfer Size and Device Width match the external FIFO and that memory is packed accordingly.

Note that before issuing a DMA request, the receiving or transmitting FIFO must be prepared to complete a full transaction (4 or 8 datums, depending on **dma\_mode**[TS]) without risking overflow or underflow.

**dma\_moderead - Read DMA Mode Register** **Offset = 0x0000**

**dma\_modeset - Set DMA Mode Register** **Offset = 0x0000**

**dma\_modeclr - Clear DMA Mode Register** **Offset = 0x0004**



Bits	Name	Description	R/W	Default
31:24	—	Reserved, should be cleared.	R	0
23:20	DAH	Device Address High. Provides the most significant 4 bits of physical device address.	R/W	0
19:16	DID	Device ID. Identifies the peripheral device to act as source or destination. This ID is used in combination with the Device ID Select bit (see Table 4-3 on page 76).	R/W	0
15	DS	Device ID Select. This bit selects between two banks of Device IDs. It is used in combination with DID (see Table 4-3 on page 76).	R/W	0
14	—	Reserved, should be cleared.	R	0
13	BE	Big Endian. 0 Little Endian byte order 1 Big Endian byte order	R/W	0
12	DR	Device Read. 0 Data is transferred from memory to device. 1 Data is transferred from device to memory.	R/W	0

Bits	Name	Description	R/W	Default
11	TS	Transfer Size. Number of datums transferred per transaction. The device width is programmed in DW. 0 4 datums. (Valid for all device widths.) 1 8 datums. (Valid for 8-bit and 16-bit device widths only.)	R/W	0
10:9	DW	Device FIFO Width. 00 Transfer width is 8 bits. 01 Transfer width is 16 bits. 10 Transfer width is 32 bits. (Not valid for TS=1.) 11 Reserved	RW	0
8	NC	Not Coherent. 0 Memory reads and writes are marked coherent on the SBUS. 1 Memory reads and writes are marked non coherent on the SBUS. For more information on coherency see Section 2.8.2 "SBUS Coherency Model" on page 41 for more information on coherency.	R/W	0
7	IE	Interrupt Enable. 0 No interrupts will be generated. 1 Interrupts are generated when either D1 or D0 is set.	R/W	0
6	H	Channel Halted. 0 Channel is active. 1 Channel is halted. This bit should be used to determine if the channel has been halted after the G bit has been cleared.	R	0
5	G	Channel Go. Setting the channel go bit enables the channel. When this bit is cleared the DMA controller does not arbitrate for this channel regardless of the state of the buffer enable bits. When the go bit is cleared by software the channel configuration should not be modified until the DMA controller sets the halt bit to indicate that the channel is inactive and therefore safe to be reconfigured.	R/W	0
4	AB	Active Buffer. 0 Buffer 0 is currently in use by the DMA. 1 Buffer 1 is currently in use by the DMA. This field can be read to determine what buffer the DMA will service next if there is not a DMA transaction in progress. During a DMA transaction this bit will reflect the buffer currently being used. Note that the DMA alternates between the two buffers. In other words, it is not possible to only use one buffer, DMA transactions must be switched between each buffer.	R	0
3	D1	Done 1. The D1 bit is set by the DMA controller to indicate that a transfer to or from buffer 1 is complete. This bit must be cleared by the processor.	R/W	0
2	BE1	The BE1 bit enables buffer 1. This bit is set by the processor and cleared by the DMA controller when the buffer has been filled or emptied. This bit may be cleared by the processor only when the H bit is set.	R/W	0
1	D0	Done 0. The D0 bit is set by the DMA controller to indicate that a transfer to or from buffer 0 is complete. This bit must be cleared by the processor.	R/W	0
0	BE0	The BE0 bit enables buffer 0. This bit is set by the processor and cleared by the DMA controller when the buffer has been filled or emptied. This bit may be cleared by the processor only when the H bit is set.	R/W	0

### 4.1.2 DMA Peripheral Device Address

The peripheral device address register contains a pointer to the peripheral FIFO to be used as a source or destination. Software is responsible for matching the peripheral address to the correct value of the Device ID (DID) field in the mode register. The correspondence between FIFO address and DID values is shown in Table 4-3 on page 76. The physical address of the FIFO must be used.

The DAH field from the **dma\_mode** register is used as the most significant four bits of the FIFO physical address.

#### dma\_peraddr - DMA Peripheral Address Register

Offset = 0x0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	ADDR																																	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	ADDR	Peripheral FIFO address	R/W	0

### 4.1.3 DMA Buffer Starting Address Registers

Each DMA channel has two buffers, labeled buffer0 and buffer1. The starting address of each buffer should be written to the **dma\_buf0addr** and **dma\_buf1addr** registers respectively. The starting address must be cache line (32 bytes) aligned.

The four most significant bits of the buffer address are held in the *BAH* field of the **dma\_buf0size** and **dma\_buf1size** registers.

The starting address must explicitly be written before each DMA transaction, even if the address has not changed from the previous, as **dma\_bufnaddr** will change during the DMA transaction.

Note that the DMA alternates between the two buffers. In other words, it is not possible to use only one buffer—DMA transactions must be switched between each buffer. The AB bit in the **dma\_mode** register can be used to determine the active buffer.

#### dma\_buf0addr - Buffer0 Starting Address

Offset = 0x000C

#### dma\_buf1addr - Buffer1 Starting Address

Offset = 0x0014

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	ADDR																																
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	ADDR	Lower 32 bits of the physical starting address of the DMA memory buffer.	R/W	0

#### 4.1.4 DMA Channel Buffer Size Registers

The size of each DMA buffer is given by the **dma\_buf0size** and **dma\_buf1size** registers. The buffer size registers also contribute the most significant four bits of the buffer physical address.

This register should be programmed with the block size of the buffer in datums. While a DMA transaction is in progress, it indicates the number of datums remaining in the transfer.

Note that the DMA alternates between the two buffers. In other words, it is not possible to only use one buffer, DMA transactions must be switched between each buffer. The active-buffer bit **dma\_mode[AB]** can be used to determine which buffer is active at a given time.

**dma\_buf0size - Buffer 0 Size**

Offset = 0x0010

**dma\_buf1size - Buffer 1 Size**

Offset = 0x0018

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	BAH												SIZE																				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:20	—	Reserved, should be cleared.	R/W	0
19:16	BAH	Buffer Address High. Provides the 4 most significant bits of the buffer address.	R/W	0
15:0	SIZE	Buffer Size and Count Remaining. Indicates the number of datums remaining in the current transfer.	R/W	0

## 4.2 Using GPIO as External DMA Requests (DMA\_REQ $n$ )

To use GPIO[4] or GPIO[5] as an external DMA request (DMA\_REQ $n$ ) follow these steps:

- 1) Write the **sys\_pinputen** to enable the GPIO to be used as an input. See Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209 for more information.
- 2) TRI-STATE the GPIO to make it an input through the **sys\_triout** register. See Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209 for more information.
- 3) Set the **dma\_peraddr** register to point to the external device data port. The Static Bus Controller must be configured correctly to recognize this address.
- 4) Program the mode register to match the direction of transfer and peripheral attributes.

The DMA\_REQ $n$  signal must be driven high to request a DMA transfer and must remain high until the DMA transaction is started. Once started, the DMA transaction continues until finished regardless of the DMA request signal state. A DMA transaction refers to a DMA transfer of one transfer size as defined in the DMA mode register (**dma\_mode[TS]**).

DMA\_REQ $n$  should be tied to the external FIFO threshold indicator. In this way the DMA\_REQ $n$  signal asserts when the FIFO threshold is reached and remains asserted until the FIFO fills or empties past the threshold (after the DMA transaction starts). DMA\_REQ $n$  should then negate after the FIFO threshold is met from the opposite direction (approaches full for a transmit or approaches empty for a read). The threshold should be designed such that a complete DMA transaction (4 or 8 datums) can occur without risking overflow or underflow.



### 4.3 Programming Considerations

The following pseudo code is for setting up and servicing a DMA channel:

```

SetupDMA() {
    Make sure interrupts are enabled globally (CP0 Reg 12, bit 0)
    Enable interrupt controller for this DMA channel, high/level
    Program the DMA controller {
        dma_modeclr = 0xFFFFFFFF
        dma_buf0size = buffer size (up to 65535)
        dma_buf1size = buffer size (up to 65535)
        dma_peraddr = Address of peripheral FIFO
        dma_buf0addr = physical base address of buffer
        dma_buf1addr = physical base address of next buffer

        Write the dma_modeset register {
            Enable interrupt
            Enable both buffers
            Set endianness
            Set data width, 8-bit, 16-bit, 32-bit
            Set Device ID
            Set transfer size, 4-datum burst, 8-datum burst
            set coherency = 0 (memory is coherent)
            set go = 1
        }
    }

    InterruptHandler() {
    Note: This routine assumes it is called from context save/restore routine at
        0x80000200.

    Check for hardware interrupt from interrupt controller 0,request 0:
        (CP0 Reg 13, bit 10) = 1

    Read interrupt controller 0 ic_req0int (interrupt status)and check if source is
        from this DMA channel, bits[13:6]

    if it is this DMA channel {

        Check dma_moderead to see which buffer is done: D0 or D1

        if (D0 is set) {
            write dma_modeclr bit D0 = 1 to clear interrupt
            if there is another buffer to send {
                dma_buf0addr = physical base address of buffer
                dma_buf0size = buffer size (up to 65535)
                write dma_modeset bit BE0 = 1 to enable buffer
            }
        }

        if (D1 is set) {
            write dma_modeclr bit D1 = 1 to clear interrupt
            if there is another buffer to send {
                dma_buf1addr = physical base address of buffer
                dma_buf1size = buffer size (up to 65535)
                write dma_modeset bit BE1 = 1 to enable buffer
            }
        }

    }
    Issue sync
}

```



# 5 Interrupt Controller

There are two interrupt controllers in the Au1100 processor. Each interrupt controller supports 32 interrupt sources. Interrupts can generate a signal to bring the Au1100 processor out of an IDLE0 or IDLE1 state and generate a CPU interrupt.

Each interrupt controller has two outputs referred to as requests 0 and 1. Each of these outputs are connected to the CPU core. See Section 2.5 "Exceptions" on page 25 for a complete Au1100 processor interrupt architecture discussion. Table 5-1 shows the interrupt controller connections to the CPU.

**Table 5-1. Interrupt Controller Connections to the CPU**

Interrupt Source	CP0 Cause Register Bit
Interrupt Controller 0:	
Request 0	10
Request 1	11
Interrupt Controller 1:	
Request 0	12
Request 1	13

## 5.1 Interrupt Controller Sources

Table 5-2 on page 84 shows the mapping of interrupt sources for Interrupt Controller 0 and 1.

Care should be taken to select the correct interrupt type (level or edge triggered) so that an interrupt is not missed. In general, level interrupts are chosen when multiple sources from a single peripheral might cause an interrupt. In this way the programmer will not miss a subsequent interrupt from a particular source while servicing the previous one.

Edge triggered interrupts can be used when there is only a single source for an interrupt. Edge triggered interrupts must be used when an interrupt is caused by an internal event and not tied to a register bit where it is latched and held until cleared by the programmer.

Details about the interrupt sources can be found in the respective peripheral sections.

Table 5-2. Interrupt Sources

Controller	Interrupt Number	Source	Type
0	0	UART0	High Level
0	1	UART1	High Level
0	2	SD0 or SD1	High Level
0	3	UART3	High Level
0	4	SSI0	High Level
0	5	SSI1	High Level
0	6	DMA0	High Level
0	7	DMA1	High Level
0	8	DMA2	High Level
0	9	DMA3	High Level
0	10	DMA4	High Level
0	11	DMA5	High Level
0	12	DMA6	High Level
0	13	DMA7	High Level
0	14	TOY (tick)	Rising Edge
0	15	TOY Match 0	Rising Edge
0	16	TOY Match 1	Rising Edge
0	17	TOY Match 2	Rising Edge
0	18	RTC (tick)	Rising Edge
0	19	RTC Match 0	Rising Edge
0	20	RTC Match 1	Rising Edge
0	21	RTC Match 2	Rising Edge
0	22	IrDA Transmit	High Level
0	23	IrDA Receive	High Level
0	24	USB Device Interrupt Request	High Level
0	25	USB Device Suspend Interrupt	Rising/Falling Edge
0	26	USB Host	Low Level
0	27	AC97 ACSYNC	Rising Edge
0	28	MAC 0 DMA Done	High Level
0	29	Logical OR of GPIO[215:208] (See Section 6.11.2.4 "Interrupt Enable Register" on page 191.)	System Dependent
0	30	LCD Controller	High Level
0	31	AC97 Command Done	Rising Edge
1	n = 0..31	GPIO[n]	System Dependent

Figure 5-1 shows the Interrupt Controller logic diagram. Where applicable, the names in the diagram correspond to bit *n* in the relative control register.

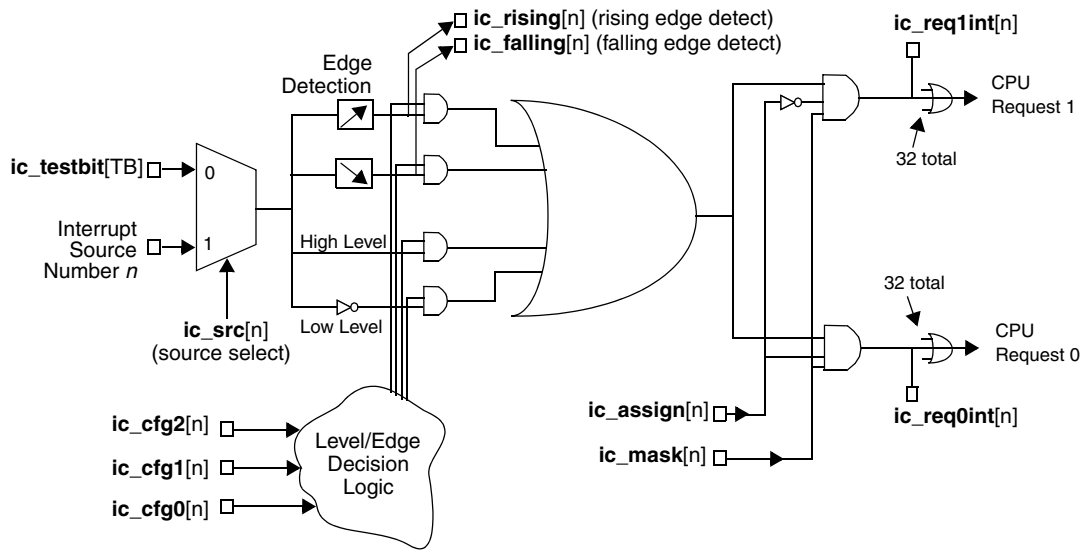


Figure 5-1. Interrupt Controller Logic

## 5.2 Register Definitions

The design of the software interface to the interrupt controller is based on the premise that software tasks should be able to access the value and control of an individual port without blocking other tasks from accessing ports of interest to them. This interrupt controller design removes the need to arbitrate via a semaphore access to the interrupt controller registers. The result is faster and simpler interrupt controller accessing.

Table 5-3 shows the base address for each interrupt controller.

Table 5-3. Interrupt Controller Base Addresses

Name	Physical Base Address	KSEG1 Base Address
ic0_base	0x0 1040 0000	0xB040 0000
ic1_base	0x0 1180 0000	0xB180 0000

Each interrupt controller has an identical set of registers that controls its set of 32 interrupts. Table 5-4 on page 86 shows the interrupt controller registers and their associated offsets. Certain offsets are shared but address different internal registers depending on whether the access is a read or a write. The register description details the functionality of the register. Bit *n* of a particular register is associated with interrupt *n* of the corresponding controller.

Table 5-4. Interrupt Controller Registers

Offset (Note 1)	Register Name	Type	Register Description	Default
0x0040	ic_cfg0rd	R	Configuration 0 register.	UNPRED
0x0040	ic_cfg0set	W	Configuration 1 register.	
0x0044	ic_cfg0clr	W	Configuration 2 register.	
0x0048	ic_cfg1rd	R	The combined field consisting of <b>ic_cfg2[n]</b> , <b>ic_cfg1[n]</b> , and <b>ic_cfg0[n]</b> specifies the trigger characteristics for interrupt <i>n</i> as shown in Table 5-5 on page 88.	UNPRED
0x0048	ic_cfg1set	W		
0x004C	ic_cfg1clr	W		
0x0050	ic_cfg2rd	R		UNPRED
0x0050	ic_cfg2set	W		
0x0054	ic_cfg2clr	W		
0x0054	ic_req0int	R	Shows active interrupts on request 0. Used by host software to determine the source of the interrupt.	0x0000 0000
0x0058	ic_srcrd	R	Selects the source of the interrupt between a test bit and the designated source.  0 The test bit ( <b>ic_testbit</b> [TB]) is used as interrupt source. 1 Peripheral interrupt (controller 0) or GPIO signal (controller 1) is used for interrupt source.	UNPRED
0x0058	ic_srcset	W		
0x005C	ic_srcclr	W		
0x005C	ic_req1int	R	Shows active interrupts on request 1. Used by host software to determine the source of the interrupt.	0x0000 0000
0x0060	ic_assignrd	R	Assigns the interrupt to one of the CPU requests.  0 Assign interrupt to request 1. 1 Assign interrupt to request 0.	UNPRED
0x0060	ic_assignset	W		
0x0064	ic_assignclr	W		
0x0068	ic_wakerd	R	Controls whether the interrupt can cause a wakeup from IDLE0 or IDLE1.  0 No wakeup from Idle. 1 Interrupt will cause wakeup from Idle.  The associated interrupt must still be enabled to wake from Idle.	0x0000 0000
0x0068	ic_wakeset	W		
0x006C	ic_wakeclr	W		
0x0070	ic_maskrd	R	Interrupt enable.  0 Disable the interrupt. 1 Enable the interrupt.	0x0000 0000
0x0070	ic_maskset	W		
0x0074	ic_maskclr	W		
0x0078	ic_risingrd	R	Designates active rising edge interrupts. If an interrupt is generated off of a rising edge, the associated rising edge detection bit must be cleared after detection.	UNPRED
0x0078	ic_risingclr	W		
0x007C	ic_fallingrd	R	Designates active falling edge interrupts. If an interrupt is generated off of a falling edge, the associated falling edge detection bit must be cleared after detection.	UNPRED
0x007C	ic_fallingclr	W		
0x0080	ic_testbit	R/W	This is a single bit register that is mapped to all the source select inputs for testing purposes.	UNPRED

Note 1. See Table 5-4 on page 86 for base address.

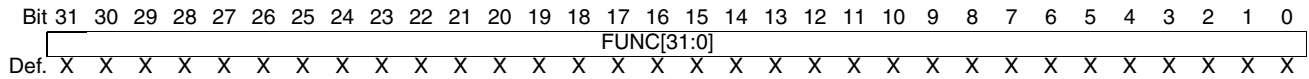
### 5.2.1 Interrupt Controller Registers

Each register (except the test-bit register) is 32 bits wide with bit *n* in each register affecting interrupt *n* in the corresponding controller.

\*rd

\*set

\*clr

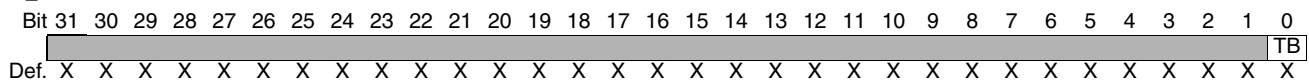


Bits	Name	Description	R/W	Default
31:0	FUNC[n]	The function of each register i'tctionality of interrupt <i>n</i> in the corresponding controller.	*rd - read only *set - write only *clr - write only See the following explanation.	See Table 5-4 on page 86.

The test-bit register contains the test bit which can be used as a test source for each interrupt. Figure 5-1 on page 85 shows how the test bit connects to the interrupt source-select logic.

Offset = 0x0080

ic\_testbit



Bits	Name	Description	R/W	Default
31:1	—	Reserved, should be cleared.	R/W	UNPRED
0	TB	Test bit value used as an alternate interrupt source.	R/W	UNPRED

Certain interrupt controller registers have the same offset but offer different functionality. This is by design. Care should be taken when programming the registers because a read from one location may reference something different from a write to the same location.

Registers ending in \*rd, \*set and \*clr have the following functionality:

- \*rd registers are read only registers will read back the current value of the register.
- \*set registers are write only registers and will set to 1 all bits that are written 1. Writing a value of 0 will have no impact on the corresponding bit.
- \*clr registers are write only registers and will clear to zero all bits that are written 1. Writing a value of 0 will have no impact on the corresponding bit.

The three configuration registers have a special functionality in that the value associated with ic\_cfg2[n], ic\_cfg1[n], ic\_cfg0[n] uniquely control interrupt *n*'s functionality as shown in Table 5-5 on page 88.

Table 5-5. Interrupt Configuration Register Function

ic_cfg2[n]	ic_cfg1[n]	ic_cfg0[n]	Function
0	0	0	Interrupts Disabled
0	0	1	Rising Edge Enabled
0	1	0	Falling Edge Enabled
0	1	1	Rising and Falling Edge Enabled
1	0	0	Interrupts Disabled
1	0	1	High Level Enabled
1	1	0	Low Level Enabled
1	1	1	Both Levels and Both Edges Enabled

### 5.3 Hardware Considerations

When using a GPIO or peripheral as an interrupt source, it is important that the associated pin functionality has been enabled in the **sys\_pinfunc** register. In addition when using a GPIO, the GPIO must first be enabled as an input. See Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209 for more information.

### 5.4 Programming Considerations

The Au1100 has been designed to simplify the interrupt control process by removing the need for a semaphore to control access to the registers. This is because there is no need to read, modify, write, as there are separate registers for setting and clearing a bit. In this way a function can freely manipulate the interrupts associated with that function.

If using edge triggered interrupts, it is important to clear the associated edge detection bit or future interrupts will not be seen.

Programming an interrupt controller can be broken into the following steps (the **set**, **clr**, and **rd** portion of the register name has been omitted):

- 1) Identify the interrupt number, *n*, with the associated peripheral or GPIO.
- 2) Use **ic\_src[n]** to assign the interrupt to the associated peripheral/GPIO (or the test bit can be used if testing the interrupt).
- 3) Set the **ic\_cfg2[n]**, **ic\_cfg1[n]** and **ic\_cfg0[n]** bits to the correct configuration for the corresponding interrupt (edge, level, polarity).
- 4) Assign the interrupt to a CPU request using **ic\_assign[n]**.
- 5) Use **ic\_wake[n]** to assign the interrupt to wake the processor from Idle if necessary or clear this register bit to keep the interrupt from waking the processor from Idle.
- 6) If the interrupt is an edge triggered interrupt, clear the edge detect register (**ic\_risingclr** or **ic\_fallingclr**) before enabling.
- 7) Finally, enable the interrupt through **ic\_mask[n]**.

When taking an interrupt the following steps should be taken:

- 1) Read **ic\_req0int** and **ic\_req1int** to determine the interrupt number *n*.
- 2) Use **ic\_fallingrd** and **ic\_risingrd** to determine if the interrupt was edge triggered. If the interrupt is edge triggered, use **ic\_fallingclr[n]** or **ic\_risingclr[n]** to clear the edge detection circuitry.
- 3) If the interrupt is to be disabled write **ic\_maskclr[n]**.
- 4) Service the interrupt.



# Peripheral Devices



This section provides descriptions of the peripheral devices of the Au1100 processor. This includes an AC97 controller, LCD controller, two SD controllers, USB Host and Device interfaces, IrDA, one 10/100 Ethernet MAC, I<sup>2</sup>S, three UARTs and two synchronous serial interfaces.

Each peripheral contains an enable register. All other registers within each peripheral's register block should not be accessed until the enable register is written the correct sequence to bring the peripheral out of reset. Accessing the peripheral register block before a peripheral is enabled will result in undefined results.

## 6.1 AC97 Controller

The Au1100 processor contains an AC97 controller which incorporates an AC-link capable of bridging to an AC97 compliant codec.

### 6.1.1 AC97 Registers

The AC97 controller is controlled by a register block whose physical base address is shown in Table 6-1. The register block consists of 5 registers as shown in Table 6-2.

**Table 6-1. AC97 Base Address**

Name	Physical Base Address	KSEG1 Base Address
ac97_base	0x0 1000 0000	0xB000 0000

**Table 6-2. AC97 Registers**

Offset (Note 1)	Register Name	Description
0x0000	ac97_config	AC-link Configuration
0x0004	ac97_status	Controller Status
0x0008	ac97_data	TX/RX Data
0x000C	ac97_cmmd	Codec Command
0x000C	ac97_cmmdresp	Codec Command Response
0x0010	ac97_enable	AC97 Block Control

Note 1. See Table 6-1 for base address.



Bits	Name	Description	R/W	Default
6	CP	Command Pending. This bit indicates that there is a command pending on the AC-link. A write to the Codec Command register will cause this bit to be set until the command is completed. The command is completed for a write when the data has been written out on slot 2. The command is completed for a read request when the status data has been read from the corresponding read request. (This means that a read request could be pending for more than 1 cycle depending on the latency of the read.) The Command register should not be written until the CP bit is clear. An interrupt can be enabled to indicate when a command is done. The source of this interrupt is an internal pulse so either rising edge or falling edge interrupt should be used for this interrupt.	R	0
5	—	Reserved	R	UNPRED
4	TE	Transmit Empty. When set this bit indicates that the transmit FIFO is empty.	R	0
3	TF	Transmit Full. When set this bit indicates the transmit FIFO is full.	R	0
2	—	Reserved	R	UNPRED
1	RE	Receive Empty. When set this bit indicates that the receive FIFO is empty.	R	0
0	RF	Receive Full. When set this bit indicates that the receive FIFO is full.	R	0

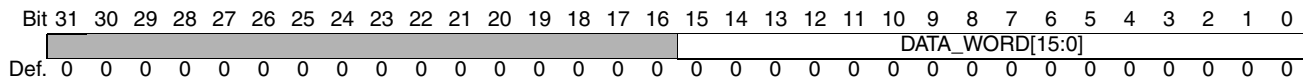
**6.1.1.3 TX/RX Data**

The TX/RX Data register is the transmit FIFO's input to the when written to and the receive FIFO's output when read from. Each FIFO has twelve 16-bit entries. Care should be taken to monitor the status register to insure that there is room for data in the FIFO for a read or write transaction. This will be taken care of automatically if using DMA (see Section 6.1.3 "Programming Considerations" on page 94).

The number of bits set in XMIT\_SLOTS will correspond with how many samples are pulled out of the FIFO and aligned in the respective slots. The number of bits set in RECV\_SLOTS will correspond with the number of samples placed in the FIFO from the respective slots in SDATA\_IN.

**ac97\_data - TX/RX Data**

Offset = 0x0008



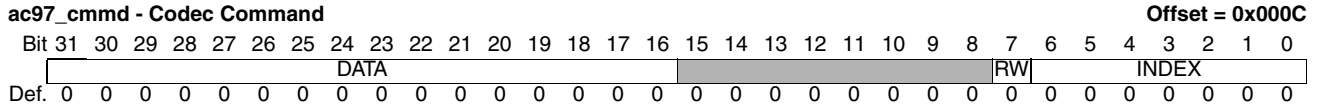
Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	DATA_WORD	Data Word. This is where data is written to or read from the FIFO. Each data word is 16 bits.	R/W	0

**6.1.1.4 Codec Command**

The Codec Command and Command Response registers share the same physical address.

The Codec Command register is used to send read and write commands to the codec. For write commands, the DATA field will be written to the register indicated by the INDEX field. For read commands, the DATA field should be written zero. The value read from the register indicated by INDEX will appear in the Codec Response register when the Command Pending bit in the status register (**ac97\_status[CP]**) returns to 0.

The Codec Command register should only be written if **ac97\_status[CP]** is 0.

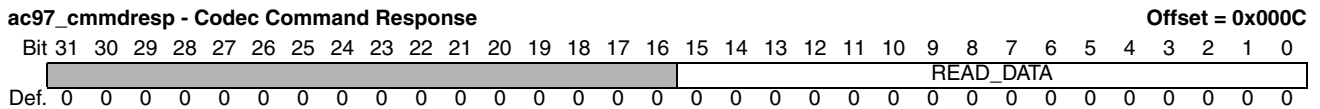


Bits	Name	Description	R/W	Default
31:16	DATA	Data. These bits will be the actual 16-bit word written to the register indicated by INDEX if RW is a 0. If RW is set (indicating a read), these bits should be written 0.	W	0
15:8	—	Reserved, should be cleared.	W	0
7	RW	Read/Write# Bit (1=read, 0=write). This bit maps to the Read/Write# bit in the command address and designates whether the current operation will be a read or a write.	W	0
6:0	INDEX	Codec Register Index. These bits will address the specific register to be read or written to inside the codec.	W	0

**6.1.1.5 Codec Command Response**

The Codec Command and Response registers share the same physical address.

After a read command is sent through the Codec Command register, the response can be read from the Codec Response register. The command response becomes valid when the Command Pending bit in the status register (**ac97\_status[CP]**) is cleared; however, the response remains valid for only one AC97 frame length in duration (20.8 μs).



Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	READ_DATA	Read Data. These bits will be the response to the last read command sent to the codec. The read data becomes valid after the read command is completed ( <b>ac97_status[CP]</b> = 0).  Note that this bit remains valid for only <i>one</i> AC97 frame (20.8 μs) and should therefore be read immediately after <b>ac97_status[CP]</b> is cleared.	R	0

### 6.1.1.6 AC97 Enable

The AC97 Enable register is used to enable and reset the entire AC97 Controller block. The routine for bringing the AC97 controller out of reset is as follows:

- 1) Set the CE bit to enable clocks while leaving the block disabled (D=1).
- 2) Clear the D bit to enable the peripheral.

#### ac97\_enable - AC97 Block Control

Offset = 0x0010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bits	Name	Description	R/W	Default
31:2	—	Reserved, should be cleared.	W	0
1	D	AC97 Controller Disable. Setting this bit will reset the AC97 block. After enabling the clock with CE, this bit should be cleared for normal operation.	W	1
0	CE	Clock Enable. This bit should be set to enable the clock driving the AC97 Controller. It can be cleared to disable the clock for power considerations.	W	0

### 6.1.2 Hardware Considerations

The AC-link consists of the signals listed in Table 6-3.

**Table 6-3. AC-Link Signals**

Signal	Input/Output	Definition
ACSYNC	O	Fixed rate sample sync. Muxed with S1DOUT.
ACBCLK	I	Serial data clock. Muxed with S1DIN.
ACDO	O	TDM output stream. Muxed with S1CLK.
ACDI	I	TDM input stream.
ACRST#	O	Codec reset. Muxed with S1DEN.

For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

### 6.1.3 Programming Considerations

To use the AC97 controller the AC97 bit in the **sys\_pifunc[A97]** register (see Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209) must be cleared. This enables the associated pins for AC97 use.

The AC97 block supports DMA transfers and interrupts. The use of the DMA or interrupts is program dependent and is not required to use the AC97 controller.

To use DMA for AC97 memory transfers the transmit and receive functions will each need a dedicated DMA channel. The DMA peripheral address register (**dma\_peraddr**) in the DMA configuration registers will be set to point to the AC97 **ac97\_data** register. The DMA mode register (**dma\_mode**) will need to be set up with the correct Device ID (DID). The Device Read bit (DR) will depend on whether the channel is being used for receive or transmit. Typically the Device Width (DW) should be set to 16 bits and the transfer size bit (TS) should be cleared because the FIFO threshold indicators correspond to four-datum transfers. This assumes that the audio samples are aligned in memory on a 16-bit audio sample boundary. The DMA will automatically monitor the transmit and receive request bits and feed data accordingly.

An interrupt ("AC97 Command Done" in interrupt controller 0) can be enabled to indicate when a command is completed. The source of this interrupt is an internal pulse so either rising edge or falling edge interrupt should be used for this interrupt.

When the AC97 ACSYNC interrupt is enabled in interrupt controller 0, an interrupt will occur corresponding to the rising edge of the ACSYNC signal. Internally a pulse is generated from the rising edge of the ACSYNC signal and fed to the interrupt controller. Regardless of the edge enabled in the interrupt controller the interrupt will come after the rising edge of ACSYNC. Enabling a rising edge interrupt will interrupt the processor closest to the rising edge of ACSYNC.

The output FIFO for the AC-link is shared for all slots so care should be taken that there is a correspondence with the number of valid bits being set and the number of valid samples written to the transmit FIFO or aligned in memory for DMA or erroneous results will occur. It is the programmer's responsibility to ensure that the number of samples written to the FIFO corresponds with the number of valid slots enabled. Data will automatically be pulled out of the FIFO in the order of what slots are enabled. In other words if slots 3, 4, 6 and 9 are enabled, the programmer should write samples corresponding to data for slots 3, 4, 6, and 9, in that order, to the FIFO.

To insure against underflow at least  $x$  words should be written per audio frame where  $x$  is the number of slots enabled. This is a mean rate over time and the actual write rate may differ depending on latency requirements, DMA buffer size, and the number of slots enabled.

Care should be taken that there is a correspondence with the number of valid bits that have been set and the number of valid samples read from the receive FIFO or erroneous results will occur.

The input FIFO for the AC-link is shared for all slots so care should be taken that there is a correspondence with the number of valid bits that are set and the number of samples read from the receive FIFO or erroneous results will occur. It is the programmer's responsibility to ensure that the number of samples read from the FIFO corresponds with the number of valid slots enabled. Data will automatically be put in the FIFO in the order of what slots are enabled. In other words if slots 3 and 4, are enabled, the programmer should read samples corresponding to data for slots 3 and 4, in that order, from the FIFO.

To insure against overflow at least  $x$  words should be read per audio frame where  $x$  is the number of slots enabled. This is a mean rate over time and the actual read rate may differ depending on latency requirements, DMA buffer size, and the number of slots enabled.

## 6.2 USB Host Controller

The Au1100 processor USB host controller conforms to the Open HCI interface specification, revision 1.0, and is USB 1.1 compliant. Two root hub ports, port 0 and port 1, are provided. The base of the Open HCI register block is shown in Table 6-4.

**Table 6-4. USB Host Base Address**

Name	Physical Base Address	KSEG1 Base Address
usbh_base	0x0 1010 0000	0xB010 0000

Only 32-bit accesses are allowed to the Open HCI registers.

All interrupts as described in the Open HCI specification are supported. These interrupts are combined when brought to the interrupt controller into one active-low interrupt (negative-edge triggered does not work). The interrupt controller should be programmed to reflect this by setting the USB Host interrupt to low level. See Section 5.0 "Interrupt Controller" on page 83 for details.

### 6.2.1 USB Host Controller Registers

#### 6.2.1.1 USB Host Enable Register

This register is not part of the OpenHCI registers; however, it shares the same base address. The **usbh\_enable** register controls the reset and clocks to the USB Host controller. When initializing the USB Host controller the programmer should first enable clocks, then enable the module (remove from reset), then wait for the RD bit to be set before performing OpenHCI initialization.

The correct routine for bringing the USB Host Controller out of reset is as follows:

- 1) Set the CE bit to enable clocks.
- 2) Set the E bit to enable the peripheral (at this time the C and BE bits should be configured appropriately for the system).
- 3) Clear the HCFS bit in the HcControl register to reset the OHCI state.
- 4) Wait for the RD bit to be set before issuing any commands to the OpenHCI controller.

To put the USB Host Controller into reset the following steps should be taken:

- 1) Set the HCFS bit in the HcControl register.
- 2) Clear the E and CE bits.

**usbh\_enable**

Offset = 0x7FFFC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											RD	CE	E	C	BE	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	Name	Description	R/W	Default
31:5	—	Reserved, must be cleared.	R/W	0
4	RD	Reset Done. Wait for this bit to be set before issuing any commands to the OpenHCI controller. Note: When writing to the <b>usbh_enable</b> register, this bit position must be 0.	R	0
3	CE	Clock Enable. When this bit is set, clocks are enabled to the USB Host controller.	R/W	0
2	E	Enable. This bit enables the USB Host controller. When this bit is clear the controller is held in reset.	R/W	0

Bits	Name	Description	R/W	Default
1	C	Coherent. If this bit is set memory accesses by the controller will be marked coherent on the SBUS. When this bit is clear memory accesses by the USB Host controller are non coherent.  For more information on coherency see Section 2.8.2 "SBUS Coherency Model" on page 41 for more information on coherency.	R/W	0
0	BE	Big Endian. When this bit is set the controller interprets data buffers in Big Endian byte order. When this bit is clear the controller interprets data buffers in Little Endian byte order.  Setting the BE bit does not swap the control structures defined in the OHCI specification. Endpoint descriptors (section 4.2), transfer descriptors (section 4.3), and the HCCA (host controller communications area, section 4.4) should always be written as words to ensure proper operation.	R/W	0

## 6.2.2 USB Host Signals

Table 6-5 shows the signals associated with the two USB host root hub ports. The USB root hub port pins have USB 1.1 compliant drivers with the addition of the external circuitry noted in the signal description.

**Table 6-5. USB Host Signals**

Signal	Input/Output	Description
USBH0P	IO	Positive signal of differential USB host port 0 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part.  Muxed with USBDP which controls the pin out of reset.
USBH0M	IO	Negative signal of differential USB host port 0 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part.  Muxed with USBDM which controls the pin out of reset.
USBH1P	IO	Positive signal of differential USB host port 1 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part.
USBH1M	IO	Negative signal of differential USB host port 1 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part.

For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.



## 6.3 USB Device Controller

The Au1100 processor USB device controller supports endpoints 0, 1, 2, 3, and 4. Endpoint 0 is always configured as a bidirectional control endpoint. Endpoints 1 and 2 are always IN endpoints and endpoints 3 and 4 are always OUT endpoints.

IN is from device to host. From the device perspective these endpoints are written, so the associated registers are tagged with write or wr.

OUT is from host to device. From the device perspective these endpoints are read, so the associated registers are tagged with read or rd.

The USB device registers are located off of the base address shown in Table 6-6.

**Table 6-6. USB Device Base Address**

Name	Physical Base Address	KSEG1 Base Address
usbd_base	0x0 1020 0000	0xB020 0000

### 6.3.1 USB Device Controller Registers

Table 6-7 shows the offsets of each register from the register base.

**Table 6-7. USB Device Register Block**

Offset (Note 1)	Register Name	Description
0x0000	usbd_ep0rd	Read from Endpoint 0
0x0004	usbd_ep0wr	Write to Endpoint 0
0x0008	usbd_ep1wr	Write to Endpoint 1
0x000c	usbd_ep2wr	Write to Endpoint 2
0x0010	usbd_ep3rd	Read from Endpoint 3
0x0014	usbd_ep4rd	Read from Endpoint 4
0x0018	usbd_inten	Interrupt Enable Register
0x001c	usbd_intstat	Interrupt Status Register
0x0020	usbd_config	Write Configuration Data
0x0024	usbd_ep0cs	Endpoint 0 Control and Status
0x0028	usbd_ep1cs	Endpoint 1 Control and Status
0x002c	usbd_ep2cs	Endpoint 2 Control and Status
0x0030	usbd_ep3cs	Endpoint 3 Control and Status
0x0034	usbd_ep4cs	Endpoint 4 Control and Status
0x0038	usbd_framenum	Current Frame Number
0x0040	usbd_ep0rdstat	EP0 Read FIFO Status
0x0044	usbd_ep0wrstat	EP0 Write FIFO Status
0x0048	usbd_ep1wrstat	EP1 Write FIFO Status
0x004c	usbd_ep2wrstat	EP2 Write FIFO Status
0x0050	usbd_ep3rdstat	EP3 Read FIFO Status
0x0054	usbd_ep4rdstat	EP4 Read FIFO Status
0x0058	usbd_enable	USB Device Controller Enable

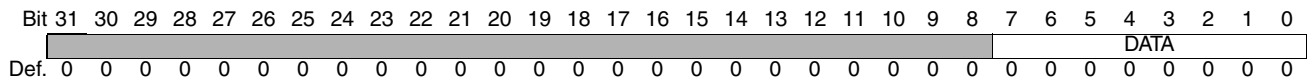
Note 1. See Table 6-6 for base address.

**6.3.1.1 Endpoint FIFO Read and Write Registers**

The endpoint FIFO read and write registers provide access to the endpoint FIFOs. Each endpoint FIFO is unidirectional. FIFO read registers may not be written, and FIFO write registers return unpredictable results if read.

Only the least significant byte of the FIFO registers contain data.

- usbd\_ep0rd** Offset = 0x0000
- usbd\_ep0wr** Offset = 0x0004
- usbd\_ep1wr** Offset = 0x0008
- usbd\_ep2wr** Offset = 0x000C
- usbd\_ep3rd** Offset = 0x0010
- usbd\_ep4rd** Offset = 0x0014



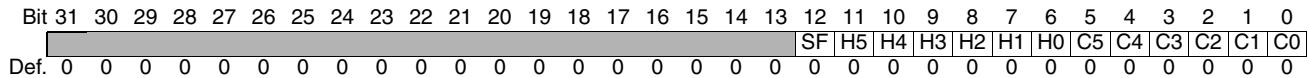
Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R/W	0
7:0	DATA	Data. Byte of data to be written to, or read from the endpoint FIFO.	R/W	0

**6.3.1.2 Interrupt Registers**

Each endpoint has an interrupt enable register and an interrupt status register. The two registers have identical formats. When a condition becomes true the corresponding bit is set in the **usbd\_intstat** register. If a bit is set in the interrupt enable register and the corresponding condition becomes true, then an interrupt is issued. The interrupt for the USB device should be programmed to high level.

Interrupts and pending conditions must be cleared by writing a 1 to the corresponding bit in the **usbd\_intstat** register.

- usbd\_inten** Offset = 0x0018
- usbd\_intstat** Offset = 0x001C



Bits	Name	Description	R/W	Default
31:13	—	Reserved, should be cleared.	R/W	0
12	SF	Start of Frame. This interrupt issues when an SOF token is received.	R/W	0
11:6	H5:H0	FIFO Half Full. These interrupts issue when the corresponding FIFO reaches the half full/half empty mark.  The bits correspond as follows: H0 - ep0rd H1 - ep0wr H2 - ep1wr H3 - ep2wr H4 - ep3rd H5 - ep4rd	R/W	0

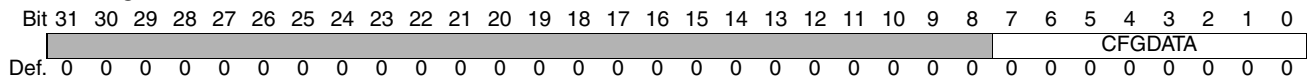
Bits	Name	Description	R/W	Default
5:0	C5:C0	Complete. These interrupts issue when a transmission or reception completes on the corresponding FIFO. For the read FIFOs (ep0rd, ep3rd, and ep4rd) these interrupts indicate the reception of a DATA0 or DATA1 packet, or a SETUP packet (ep0rd FIFO only). For the write FIFOs (ep0wr, ep1wr, and ep2wr) these interrupts indicate the transmission of a DATA0 or DATA1 packet.  The bits correspond as follows: C0 - ep0rd C1 - ep0wr C2 - ep1wr C3 - ep2wr C4 - ep3rd C5 - ep4rd	R/W	0

### 6.3.1.3 Device Configuration Register

The device configuration register allows configuration data to be loaded to the controller after reset.

usbdc\_config

Offset = 0x0020



Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R/W	0
7:0	CFGDATA	Configuration data byte. Use this field to write the configuration data block to the controller one byte at a time.	R/W	0

The device configuration data is a 25-byte block which contains the configuration information for the five supported endpoints. Each endpoint requires five configuration bytes in the format shown in Figure 6-1.

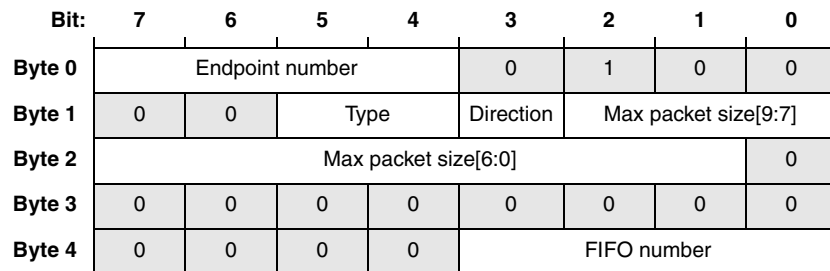


Figure 6-1. Endpoint Configuration Data Structure

The configuration fields are described in Table 6-8.

**Table 6-8. Endpoint Configuration Field Descriptions**

Field	Description
Endpoint number	Although the endpoint number ranges from 0 to 15, only endpoints 0, 1, 2, 3, and 4 are supported. It is highly recommended that the example values in Table 6-9 on page 101 be used for this field.
Type	Endpoint type. 00 Control 01 Isochronous 10 Bulk 11 Interrupt
Direction	Endpoint direction. (Does not apply to control endpoints.) 0 Out 1 In
Max packet size	Maximum packet size (in bytes). Note that for control, bulk, and interrupt endpoints, the maximum packet size is limited to 64 bytes. Only isochronous endpoints can accept packets up to 1023 bytes. 000 0000 000 = 0 bytes 000 0000 001 = 1 byte ... 111 1111 111 = 1023 bytes
FIFO number	This field designates which FIFO the endpoint uses. For endpoint 0 this field is ignored since endpoint 0 always uses FIFOs 0 and 1. It is highly recommended that the example values in Table 6-9 on page 101 be used for this field.

After the controller is removed from reset, the device configuration data must be written to the **usbd\_config** register in order beginning with byte 0. Bytes are written individually using unsigned 32-bit words, as shown the following example code:

```
for (i=0; i<25; i++)
    *usbd_config = (unsigned int) cfg_data_bytes[i];
```

An example configuration data block is shown in Table 6-9.

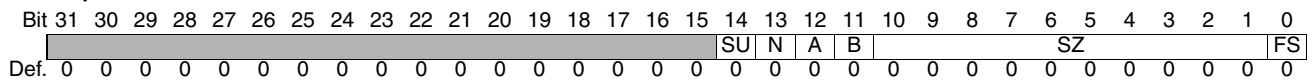
**Table 6-9. Example Endpoint Configuration Data Block**

Byte	Value	Description
0	0000 0100	Endpoint number = 0
1	0000 0000	Type = control
2	1000 0000	Direction = bidirectional
3	0000 0000	Max packet size = 64 bytes
4	0000 0000	FIFOs 0 and 1
5	0001 0100	Endpoint number = 1
6	0011 1000	Type = interrupt
7	1000 0000	Direction = in
8	0000 0000	Max packet size = 64 bytes
9	0000 0010	FIFO 2
10	0010 0100	Endpoint number = 2
11	0010 1000	Type = bulk
12	1000 0000	Direction = in
13	0000 0000	Max packet size = 64 bytes
14	0000 0011	FIFO 3
15	0011 0100	Endpoint number = 3
16	0010 0000	Type = bulk
17	1000 0000	Direction = out
18	0000 0000	Max packet size = 64 bytes
19	0000 0100	FIFO 4
20	0100 0100	Endpoint number = 4
21	0010 0000	Type = bulk
22	1000 0000	Direction = out
23	0000 0000	Max packet size = 64 bytes
24	0000 0101	FIFO 5

6.3.1.4 Endpoint Control Registers

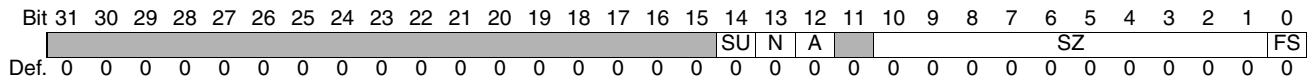
The endpoint control registers define parameters and reflect operational conditions for each endpoint.

**usbd\_ep0cs** Offset = 0x0024



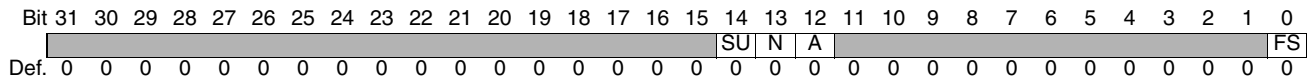
**usbd\_ep1cs** Offset = 0x0028

**usbd\_ep2cs** Offset = 0x002C



**usbd\_ep3cs** Offset = 0x0030

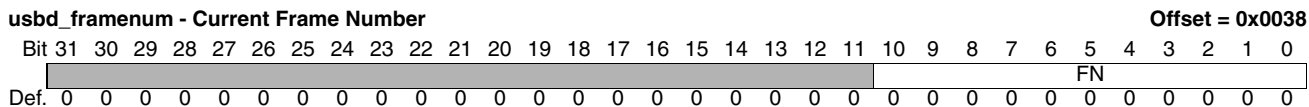
**usbd\_ep4cs** Offset = 0x0034



Bits	Name	Description	R/W	Default
31:15	—	Reserved, should be cleared.	R/W	0
14	SU	Setup Received. This bit is set when a SETUP packet is received from the host. It is only valid for EP 0.	R	0
13	N	NAK. This bit is set when an operation does not complete successfully or when data in a receive FIFO should be ignored. For most cases this implies a returned NAK in response to a DATA packet or an incorrect CRC.	R	0
12	A	ACK. This bit is set when an operation completes successfully. Most of the time this means that the Host returned an ACK to a DATA (or SETUP) packet or that a packet was received correctly and an ACK returned to the Host.  Isochronous DATA and SETUP packets deviate from this model. For these types of packets the A bit indicates successful transmission or reception but no ACK is returned or expected.	R	0
11	B	Alternate ACK. Set when a DATA frame is correctly received on endpoint 0. (B is not present on other endpoints.)	R	0
10:1	SZ	Size. The SZ field specifies the data size of an IN transfer. The SZ field applies only to endpoints 0, 1, and 2.	R/W	0
0	FS	Force Stall. Setting this bit places the endpoint in a stalled condition. Any transaction directed to the endpoint is answered with a STALL response. STALL is typically used to indicate that the endpoint has halted. Note that a Clear Feature command received via the USB does not clear a stall condition forced by this bit.	R/W	0

### 6.3.1.5 Current Frame Number

This register provides the current frame number from the start of frame packet.



Bits	Name	Description	R/W	Default
31:11	—	Reserved, should be cleared.	R	0
10:0	FN	Frame Number This field contains the frame number from the start of frame packet.	R	0

### 6.3.1.6 FIFO Status Registers

Each FIFO has a status register that indicates the current state and any error conditions. The USB FIFOs are 1-byte wide and eight bytes deep.

**usb\_d\_ep0rdstat** Offset = 0x0040

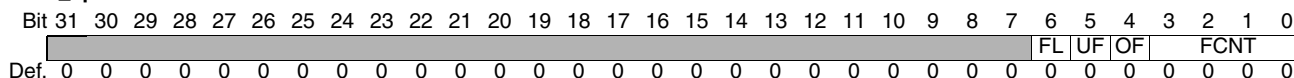
**usb\_d\_ep0wrstat** Offset = 0x0044

**usb\_d\_ep1wrstat** Offset = 0x0048

**usb\_d\_ep2wrstat** Offset = 0x004C

**usb\_d\_ep3rdstat** Offset = 0x0050

**usb\_d\_ep4rdstat** Offset = 0x0054



Bits	Name	Description	R/W	Default
31:7	—	Reserved, should be cleared.	R/W	0
6	FL	Flush FIFO. Setting this bit flushes the corresponding FIFO and discards any data contained in it.	W	0
5	UF	Underflow Flag. Set when attempting a read from an empty FIFO. Clear this flag by writing a 1 to it.	R/W	0
4	OF	Overflow Flag. Set if a byte is written to a full FIFO. Clear this flag by writing a 1 to it.	R/W	0
3:0	FCNT	FIFO Count. Reflects the current number of bytes (0 to 8) in the corresponding FIFO.	R	

### 6.3.1.7 Device Controller Enable Register

The USB device controller enable register (**usbd\_enable**) controls the clocks and reset to the device controller. The programmer should first enable clocks before enabling the device controller. To bring the USB device out of reset, follow these steps:

- 1) Set the CE bit to enable clocks.
- 2) Delay for a period greater than or equal to 1  $\mu$ s.
- 3) Set the E bit to enable the peripheral.
- 4) Delay at least 1  $\mu$ s before programming any registers in the peripheral.

**usbd\_enable**

Offset = 0x0058

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:3	—	Reserved, should be cleared.	W	0
2	SI	Streaming Isochronous mode. Clearing this bit allows isochronous endpoints to service IN and OUT transactions when the endpoint interrupt is pending. This mode is enabled by default. Setting this bit (not recommended) forces ISO endpoints to wait for pending interrupts to be cleared before accepting further data.	W	0
1	CE	Clock Enable. Clearing this bit disables all clocks to the USB Device core. Setting this bit allows normal operation.	W	0
0	E	Enable. When this bit is cleared the Device Controller will be held in reset. Setting this bit enables normal operation.	W	0

## 6.3.2 Programming Considerations

### 6.3.2.1 Removing the Controller from Reset

The following sequence of operations must be applied to remove the controller from reset.

- 1) Write a 0x0002 to the **usbd\_enable** register to enable the clocks.
- 2) Wait 1  $\mu$ s.
- 3) Write a 0x0003 to the **usbd\_enable** register to remove the controller from reset.
- 4) Wait 1  $\mu$ s.
- 5) Write 25 bytes of configuration data to the **usbd\_config** register.

There are no special constraints on *entering* the reset state: One write to the **usbd\_enable** register may be used to turn the clocks off and reset the controller.

Note: Accessing the endpoint control registers (**usbd\_epnics**), frame number register (**usbd\_framenum**) or configuration data register (**usbd\_config**) while the USB device is in suspend mode will result in a System Bus (SBUS) deadlock. This will inhibit any further operation of the CPU, including EJTAG debugger operation.

### 6.3.2.2 Latency Requirements

The time from reception of a token such as IN or OUT until the controller must source the corresponding DATA frame is very short. It is not practical to wait for a token before preparing the buffer for the response. Buffers must be posted before the token is received.

The token itself is not passed to the buffer—only DATA and SETUP frames are transferred. When a DATA or SETUP frame is received the difference between an OUT and a SETUP can be determined by examining the SU bit in the **usbd\_ep0cs** register. (Only endpoint 0 should receive SETUP packets.)

If an IN endpoint is enabled and no data is available in the FIFO the endpoint will NAK. Underrunning the FIFO during a transfer (after the first byte has been written to the FIFO) will result in a bit stuff error.



### 6.3.2.3 Using DMA

DMA should be used for all transfers with the exception of the FIFO cleanout described below for OUT transactions.

For IN transactions the size in the **usbd\_epnrcs** register should be set to MAXPACKET for all but the last buffer and to the actual remaining transfer size for the last packet. The DMA size must be set to match **usbd\_epnrcs** before the DMA is enabled for proper frame transmission.

If the last buffer of an IN series is a full MAXPACKET in length it may be necessary to set the size in the **usbd\_epnrcs** register to zero and write a byte to the FIFO to enable the transmission of a zero length DATA frame since this is often the indicator for end-of-transfer. In this case the FIFO must be cleared before the next buffer is set up.

For OUT endpoints the DMA may be programmed to a larger size than a transfer will use. When the endpoint completes the FIFO should be examined to see if there are any remaining bytes available. These bytes must be read from the FIFO under program control since the DMA will not receive a request when less than 4 bytes are in the FIFO.

For endpoint zero it is necessary to keep a DMA buffer enabled at all times since SETUP and OUT transactions can come at any time. The user should implement a circular buffer and extract transactions from this buffer in software, rather than trying to have the DMA place transactions into separate buffers.

### 6.3.2.4 Servicing Interrupts

When an interrupt is received the **usbd\_intstat** register should be read to determine the cause of the interrupt. Once the interrupt has been serviced the **usbd\_intstat** register should be written with the same value to clear the interrupt.

When an IN or OUT transaction is completed the device controller will NAK all further IN/OUT tokens until the interrupt is cleared by writing the **usbd\_intstat** register. This allows the interrupt service routine time to drain the FIFO and set up for the next transaction rather than concatenating data from separate transactions. This automatic wait can be disabled for isochronous endpoints by clearing **usbd\_enable[SI]**. SETUP packets can never be delayed with NAK.

For bulk, isochronous, and interrupt endpoints the A and N flags are somewhat redundant. Only one of them should be set for a given transaction.

For the control endpoint the flags are broken out to provide separate feedback for various phases of control transactions. This is necessary since only the IN and OUT phases can be paused with NAKs. SETUP packets must be absorbed gracefully at all times.

The A flag (ACK) is set to indicate successful reception of OUT or SETUP packets. The B flag (alternate ACK) is set to indicate successful reception of OUT data only. (SETUP packets do not affect this flag.) The N flag (NAK) is set to indicate an unsuccessful attempt to send data in response to an IN token.

This combination of flags allows all situations to be decoded. The most complex of these is when a SETUP packet immediately follows an OUT phase used to acknowledge the previous transaction. Without this separation the acknowledgement would be lost.

### 6.3.2.5 Automatic Execution of Commands

Some standard setup commands directed at endpoint zero are automatically serviced by the USB device hardware. These commands are still passed to the memory buffer. No further action is required to service these commands although they may be used to signal state changes within the software.

The following commands are automatically serviced:

- Set Address
- Set/Clear Feature
- Set/Get Configuration
- Set/Get Interface
- Get Status

### 6.3.2.6 Detecting USB Reset

The USB device controller does not provide a way to detect reset on the USB. It is recommended that if a device needs to change state on reset it should use the reception of a **Set Address** command to indicate that a reset has occurred.

### 6.3.2.7 Automatic Suspension

If the USB device is idle for more than 5 ms, the device controller enters a suspend state. In this state the device controller does not consume data. A rising edge suspend interrupt is provided to inform the CPU when this occurs. The suspend interrupt may also be used to detect the exit from suspend by using the falling edge of the interrupt.

**Note:** Because the USB device controller will suspend itself if left idle, the USB device configuration routine (including programming the interrupt controller to recognize request and suspend interrupts from the USB device) must be *fully* completed within 5 ms of bringing the peripheral out of its reset state.

### 6.3.2.8 Re-establishing a Connection after Reset

During software initialization of the USB device controller, the USBDP and USBDM signals do not automatically enter a disconnect-bus state in which both signals go low for more than 2.5  $\mu$ s. Instead, after a runtime or hardware reset of the system, the signals stay in a connect-bus state in which USBDP remains high and USBDM remains low. This prevents the USB host from recognizing the need to establish a new bus enumeration, and the logical communication flow remains disrupted.

To re-establish logical communication after reset, system initialization software can control a GPIO signal to temporarily (more than 2.5  $\mu$ s) disable power to USBDP. It is recommended to use the GPIO to toggle an LDO (low drop-out) voltage regulator placed between the USB power supply (VBUS) and the pull-up resistor attached to USBDP.

## 6.3.3 Programming Examples for USB Device

### 6.3.3.1 Initialization

- 1) Configure 48 MHz USB device clock from AUX PLL.

```
sys_auxpll = 16;           // set the AUX PLL to 192 MHz (12 MHz x 16)
sys_freqctrl0 |= 0x3;     // enable FREQ0 and select
                           // AUX PLL as the FREQ0 source
sys_clksrc |= 0xB;       // divide FREQ0 by 4 to obtain
                           // 48 MHz, and select FREQ0 as the USB clock
```

- 2) Enable USB Device Controller.

```
usbd_enable = 0x02;      // enable USB D clocks
wait at least 1 us;
usbd_enable = 0x03;      // remove reset from USB D Controller
wait at least 1 us;
```

- 3) Write 25-byte configuration data to the Configuration Register.

```
for( i = 0; i < 25; ++i )
{
    usbd_config = (unsigned int) config_data_bytes[i];
}
wait at least 1 us;
```

- 4) Set up Endpoint Control Registers (example).

```
usbd_ep0cs = 64 << 1;    // set endpoint 0 MAXPACKET
usbd_ep1cs = 8 << 1;     // set endpoint 1 MAXPACKET
usbd_ep2cs = 8 << 1;     // set endpoint 2 MAXPACKET
usbd_ep3cs = 8 << 1;     // set endpoint 3 MAXPACKET
usbd_ep4cs = 8 << 1;     // set endpoint 4 MAXPACKET
```

- 5) Clear FIFO Status Registers.

```
// clear Overflow Flag, Underflow Flag, Flush FIFO
usbd_ep0rdstat = 0x70;
usbd_ep0wrstat = 0x70;
usbd_ep1wrstat = 0x70;
usbd_ep2wrstat = 0x70;
usbd_ep3rdstat = 0x70;
usbd_ep4rdstat = 0x70;
```

## 6) Configure DMA channels.

```
// assign a DMA channel for endpoint 0 receive and build
// multiple buffer descriptors
// assign a DMA channel for endpoint 0 transmit and build
// multiple buffer descriptors
// assign a DMA channel for endpoint 1 transmit and build
// multiple buffer descriptors, if necessary
// assign a DMA channel for endpoint 2 transmit and build
// multiple buffer descriptors, if necessary
// assign a DMA channel for endpoint 3 receive and build
// multiple buffer descriptors, if necessary
// assign a DMA channel for endpoint 4 receive and build
// multiple buffer descriptors, if necessary
```

## 7) Configure the interrupt type for the USB device request (interrupt controller 0, number 24) as high-level.

## 8) Configure the interrupt type for the USB device suspend (interrupt controller 0, number 25) as rising-edge.

## 9) Start the Endpoint 0 receive DMA.

## 10) Enable USB Interrupts.

```
usb_d_inten = 0x0000003f; // enable transfer-complete interrupts
```

**6.3.3.2 Interrupt Handler**

The steps to handle an interrupt are shown below. This example handler is for a general USB application and may not be sufficient for a specific application. The handler must be installed before interrupts are enabled.

## 1) Obtain the USBBD interrupt status.

```
status = usb_d_intstat // obtain usb_d_intstat
```

## 2) Execute each interrupt condition.

```
// check if endpoint transfer complete
{
    // if ep0rd completed, execute the process for ep0rd.
    // if ep0wr completed, execute the process for ep0wr.
    // if ep1wr completed, execute the process for ep1wr.
    // if ep2wr completed, execute the process for ep2wr.
    // if ep3rd completed, execute the process for ep3rd.
    // if ep4rd completed, execute the process for ep4rd.
}
```

## 3) Clear the USBBD interrupt status.

```
usb_d_intstat = status; // clear interrupts
```

### 6.3.4 Hardware Considerations

Table 6-10 shows the signals associated with the USB device. The USB root hub port pins have USB 1.1 compliant drivers with the addition of the external circuitry noted in the signal description. The USB device implementation is full speed with the required termination noted in Table 6-10. Low speed is not supported.

**Table 6-10. USB Device Signals**

Signal	Input/Output	Description
USBDP	IO	Positive signal of differential USB device driver. Requires a 1.5 kohm pull-up resistor to denote a full speed device. Also requires an external ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant. Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part. Muxed with USBH0P.
USBDM	IO	Negative signal of differential USB device driver. Requires an external ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant. Termination Note: Requires an external 20 ohm resistor placed in series within 0.5 inches of the part. Muxed with USBH0M.

For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

## 6.4 IrDA

The IrDA (Infrared Data Association) peripheral is a serial device that uses an infrared serial bus. Features of this peripheral are:

- FIR, MIR, and SIR modes supported
- Integrated physical layer (PHY) implementation - only an infrared transceiver is needed.
- Integrated DMA for block transfer of packet data to/from memory
- Support for both Big Endian and Little Endian memory addressing
- 16-bit or 32-bit hardware CRC generation and detection
- Interrupt support on send and receive of buffer

The operating modes and standards supported are listed in Table 6-11.

**Table 6-11. IrDA Modes Supported**

Mode	Speed	Compliance
SIR	2.4 to 115.2 kbps	IrDA 1.0
MIR	1.152 Mbps	IrDA 1.1 with error detection
FIR	4.0 Mbps	IrDA 1.1 with error detection

### 6.4.1 IrDA Registers

The IrDA peripheral is programmed via a block of registers with a base address as shown in Table 6-12. The register set index is described in Table 6-13.

**Table 6-12. IrDA Base Address**

Name	Physical Base Address	KSEG1 Base Address
irda_base	0x0 1030 0000	0xB030 0000

**Table 6-13. IrDA Registers**

Offset (Note 1)	Register Name	Description
0x0000	ir_rngptrstat	Infrared Ring Pointer Status
0x0004	ir_rngbsadrh	Infrared Ring Base Address High Register
0x0008	ir_rngbsadrl	Infrared Ring Base Address Low Register
0x000C	ir_ringsize	Infrared Ring Size Register
0x0010	ir_rngprompt	Infrared Ring Prompt Register
0x0014	ir_rngadrcmp	Infrared Ring Address Compare Register
0x0018	ir_intclear	IrDA interrupt clear register
0x0020	ir_config1	Infrared Configuration 1 Register
0x0024	ir_sirflags	Infrared SIR Flags Register
0x0028	ir_statusen	Infrared Status/Enable Register
0x002C	ir_rdphycfg	Infrared Read PHY Configuration Register
0x0030	ir_wrphycfg	Infrared Write PHY Configuration Register
0x0034	ir_maxpktlen	Infrared Maximum Packet Length Register
0x0038	ir_rxbytecnt	Infrared Received Byte Count Register
0x003C	ir_config2	Infrared Configuration Register 2
0x0040	ir_enable	Infrared Interface Configuration Register

Note 1. See Table 6-12 for base address.

### 6.4.1.1 Infrared Ring Pointer Status Register

This read-only register gives the current indices for both the transmit and receive ring buffer pointers. The ring buffers form one contiguous memory block with the receive ring buffer beginning at the ring base address and the transmit ring buffer following afterward.

#### ir\_rngptrstat - Infrared Ring Pointer Status

Offset = 0x0000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:15	—	Reserved, read as 0.	R	0
14	—	Reserved, read as 1.	R	1
13:8	TRPI	Transmit Ring Pointer Index. Gives the current pointer location in the transmit ring buffer.	R	0
7:6	—	This bits are reserved and are always read as 0.	R	0
5:0	RRPI	Receive Ring Pointer Index. Gives the current pointer location in the receive ring buffer.	R	0

### 6.4.1.2 Infrared Ring Base Address High Register

This register defines the base address of the transmit and receive ring buffers. The receive ring buffer begins at the specified base address; the transmit ring buffer begins at base address + 512 bytes.

#### ir\_rngbsadrh - Infrared Ring Base Address High Register

Offset = 0x0004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:6	—	Reserved, read/written as 0.	R/W	0
5:0	RBAH	Ring buffer base address bits [31:26].	R/W	0

### 6.4.1.3 Infrared Ring Base Address Low Register

This register defines the base address of the transmit and receive ring buffers. The receive ring buffer begins at the specified base address; the transmit ring buffer begins at base address + 512 bytes.

#### ir\_rngbsadrl - Infrared Ring Base Address Low Register

Offset = 0x0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R/W	0
15:0	RBAL	Ring buffer base address bits [25:10].	R/W	0

#### 6.4.1.4 Infrared Ring Size Register

This register defines the size for both the transmit and receive ring buffers.

##### ir\_ringsize - Infrared Ring Size Register

Offset = 0x000C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved. Read/written as 0.	R/W	0
15:12	TRBS	Transmit Ring Buffer Size and Receive Ring Buffer Size. Each ring buffer size is programmed as follows: 0000 4 entries (default) 0001 8 entries 0011 16 entries 0111 32 entries 1111 64 entries All other values are not valid.	R/W	0
11:8	RRBS		R/W	0
7:0	—	Reserved. Read/written as 0.	R/W	0

#### 6.4.1.5 Infrared Ring Prompt Register

Writing this register forces the infrared controller to read the ownership bits of the transmit and receive ring buffers. Reading this register returns a value of 0x0000FFFF.

##### ir\_rngprompt - Infrared Ring Prompt Register

Offset = 0x0010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:16	—	These bits are reserved and should be written as 0.	W	UNPRED
15:0	D/C	Don't care.	W	UNPRED

#### 6.4.1.6 Infrared Ring Address Compare Register

Setting the address field in this register will define which IrDA packets to accept.

**Note:** This feature must be enabled by setting EN = 1.

##### ir\_rngadrcmp - Infrared Ring Address Compare Register

Offset = 0x0014

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

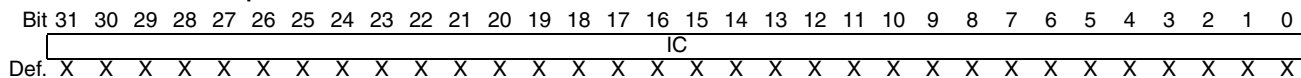
Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R	0
15	EN	Address comparison enable. 0 Comparison disabled. 1 Comparison enabled.	R/W	0
14:8	—	Reserved, read/written as 0.	R/W	0
7:0	ADDR	IrDA packet address to compare.	R/W	0

### 6.4.1.7 IrDA Interrupt Clear

Writing to this register will clear all pending IrDA interrupts.

#### ir\_intclear - IrDA Interrupt Clear

Offset = 0x0018



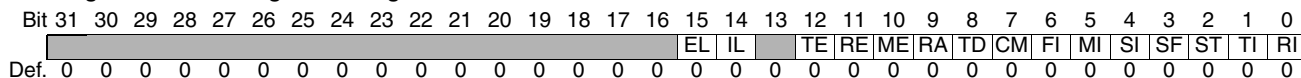
Bits	Name	Description	R/W	Default
31:0	IC	Interrupt Clear. Any write to this register will clear all pending IrDA interrupts.	W	UNPRED

### 6.4.1.8 Infrared Configuration Register 1

This register defines general setup parameters for the IrDA controller.

#### ir\_config1 - Infrared Configuration Register 1

Offset = 0x0020



Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R/W	0
15	EL	Enable external transmit while in loopback.	R/W	0
14	IL	Enable internal loopback (FIR only).	R/W	0
13	—	Reserved, read/written as 0.	R/W	0
12	TE	Transmit enable. Unless in loopback mode, only one transfer direction (transmit or receive) can be enabled at one time.	R/W	0
11	RE	Receive enable. Unless in loopback mode, only one transfer direction (transmit or receive) can be enabled at one time.	R/W	0
10	ME	DMA Enable; when set ME allows DMA access to system memory by the IrDA controller. The IrDA has its own DMA controller. This bit should always be set for normal operation.	R/W	0
9	RA	Receive all small/run packets of size less than 4 bytes (SIR mode only).	R/W	0
8	TD	Transparency destuffing disable for SIR receive filter 0 Destuffing enabled. 1 Destuffing disabled.	R/W	0
7	CM	Cyclical Redundancy Check (CRC) mode 0 32-bit CRC 1 16-bit CRC	R/W	0
6	FI	Fast infrared mode enable (FIR) When this bit is set the IRFIRSEL output will be a 1	R/W	0
5	MI	Medium infrared mode enable (MIR) When this bit is set the IRFIRSEL output will be a 1	R/W	0
4	SI	Slow infrared mode enable (SIR) When this bit is set the IRFIRSEL output will be a 0	R/W	0
3	SF	Enable SIR byte filter on the receiver (SIR mode only).	R/W	0
2	ST	Enable SIR filter when not in SIR mode (test).	R/W	0
1	TI	Invert transmit LED signal	R/W	0
0	RI	Invert receive LED signal	R/W	0



### 6.4.1.9 Infrared SIR Flags Register

This register returns bit sequences for start-of-frame and end-of-frame of an IrDA packet.

#### ir\_sirflags - Infrared SIR Flags Register

Offset = 0x0024

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read as 0.	R	0
15:8	FS	Footer bit sequence for end-of-frame.	R	0xC1
7:0	HS	Header bit sequence for start-of-frame.	R	0xC0

### 6.4.1.10 Infrared Status/Enable Register

This register defines enabling/disabling of the physical (PHY) layer and gives programming status for the IrDA controller as defined by Infrared Configuration Register 1 (**ir\_config1**).

#### ir\_statusen - Infrared Status/Enable Register

Offset = 0x0028

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R/W	0
15	E	Enable PHY layer.	R/W	0
14	CE	Configuration Error. This bit is set when more than one operating mode (SIR, MIR, or FIR) is enabled simultaneously.	R	0
13	FV	Valid FIR mode configuration.	R	0
12	MV	Valid MIR mode configuration.	R	0
11	SV	Valid SIR mode configuration.	R	0
10	TS	Status of transmit enable (TE) bit.	R	0
9	RS	Status of receive enable.	R	0
8	CS	Status of Cyclical Redundancy Check mode (CM) bit.	R	0
7:0	—	Reserved, read as 1.	R	0xFF

#### 6.4.1.11 Infrared Read PHY Configuration Register

This register returns the settings of the the last value in **ir\_wrphycfg** when bit 15 (Enable) of the **ir\_statusen** register is 0. When Enable is set, a write to **ir\_wrphycfg** will not update into **ir\_rdphycfg** until enable is 0 again.

##### ir\_rdphycfg - Infrared Read PHY Configuration Register

Offset = 0x002C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	[Reserved]																BR			PW			P									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read as 0.	R	0
15:10	BR	Baud rate (see Section 6.4.3 "Programming Considerations" on page 117).	R	0
9:5	PW	Pulse width (see Section 6.4.3 "Programming Considerations" on page 117).	R	0
4:0	P	This register will determine the number of preamble bytes to send for FIR, or start flags for MIR.  It should be interpreted as 1 less than the actual number of preamble bytes/start flags required (i.e. setting this field to 1 will cause 2 start flags to be sent in MIR mode).  This field does not apply to SIR.	R	0

#### 6.4.1.12 Infrared Write PHY Configuration Register

This register defines the settings of the physical layer (PHY) interface. When read this register returns the last value written to it.

The status of these values may be read by the Infrared Read PHY Configuration Register, **ir\_rdphycfg** when bit 15 (Enable) of the **ir\_statusen** register is 0.

##### ir\_wrphycfg - Infrared Write PHY Configuration Register

Offset = 0x0030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	[Reserved]																BR			PW			P									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R/W	0
15:10	BR	Baud rate (see Section 6.4.3 "Programming Considerations" on page 117).	R/W	0
9:5	PW	Pulse width (see Section 6.4.3 "Programming Considerations" on page 117).	R/W	0
4:0	P	This register will determine the number of preamble bytes to send for FIR, or start flags for MIR.  It should be interpreted as 1 less than the actual number of preamble bytes/start flags required (i.e. setting this field to 1 will cause 2 start flags to be sent in MIR mode).  This field does not apply to SIR.	R/W	0

### 6.4.1.13 Infrared Maximum Packet Length Register

This register defines the maximum length of a received IrDA packet.

#### ir\_maxpktlen - Infrared Maximum Packet Length Register

Offset = 0x0034

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:13	—	Reserved, read/written as 0.	R/W	0
12:0	ML	Maximum received packet length.	R/W	0

### 6.4.1.14 Infrared Receive Byte Count Register

This register returns the current number of received bytes.

#### ir\_rxbytecnt - Infrared Receive Byte Count Register

Offset = 0x0038

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:13	—	Reserved, read as 0.	R	0
12:0	RBCR	Received byte count.	R	0

### 6.4.1.15 Infrared Configuration Register 2

This register defines general setup parameters for the IrDA controller.

#### ir\_config2 - Infrared Configuration Register 2

Offset = 0x003C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1

Bits	Name	Description	R/W	Default
31:16	—	Reserved, read/written as 0.	R/W	0
15:9	—	Reserved, read as 1.	R	0x7F
8	IE	Interrupt Enable. Setting this bit will allow interrupts to be generated when a ring buffer has been transmitted or received. Writing to the ir_intclear register will clear all pending interrupts.	R/W	0
7:6	FS	Filter selection for finite impulse response DPLL. 00 Highest filter 01 Medium high filter 10 Medium low filter 11 Lowest filter	R/W	0x0
5	DA	Disable adjacent pulse width packet circuit in the FIR DPLL. 0 Circuit enabled 1 Circuit disabled	R/W	0
4	DP	Disable pulse width adjustment circuit in the FIR DPLL.	R/W	0

Bits	Name	Description	R/W	Default
3:2	CS	PHY layer clock speed. 00 40 MHz 01 48 MHz 10 56 MHz 11 64 MHz Note that the IrDA clock must be configured to match value set in CS. The IrDA clock is programmed from the clock generator; see Section 7.1 "Clocks" on page 194.	R/W	0x0
1	P	One receive pin mode. 0 Two pins for receive 1 One pin each for receive and speed select (slow or fast)	R/W	0
0	MI	Mode inversion (when P=1). 0 Fast speed is chosen by asserting speed select low. 1 Fast speed is chosen by asserting speed select high.	R/W	0

**6.4.1.16 Infrared Enable Register**

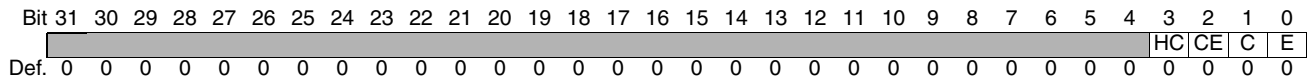
This register defines the IrDA peripheral interface setup and has a bit to enable clocks to the IrDA module.

The correct routine for bringing the IrDA out of reset is as follows:

- 1) Set the CE bit to enable clocks with the HC, C, and E bit set appropriately.

**ir\_enable - Infrared Enable Register**

Offset = 0x0040



Bits	Name	Description	R/W	Default
31:4	—	Reserved, read/written as 0.	R/W	0
3	HC	Half clock speed for IrDA clock. 0 Clock runs at full SBUS frequency. 1 Clock runs at one-half SBUS frequency. Note that HC is not just for power savings. HC must be set when the SBUS is greater than 100 MHz.	R/W	0
2	CE	Clock enable for the IrDA module. 0 Disable clocks. 1 Enable clocks.	R/W	0
1	C	Coherent. 0 Memory accesses are marked non coherent 1 Memory accesses are marked coherent For more information on coherency see Section 2.8.2 "SBUS Coherency Model" on page 41 for more information on coherency.	R/W	1
0	E	Endian mode. 0 Little Endian 1 Big Endian	R/W	1

## 6.4.2 Hardware Considerations

Table 6-14 describes the connection between the IrDA peripheral and the external transceiver.

**Table 6-14. IrDA Hardware Connections**

Signal	Input/Output	Description
IRDATX	O	Serial IrDA output Muxed with GPIO[211] which controls the pin out of hardware reset, runtime reset and Sleep.
IRDARX	I	Serial IrDA input
IRFIRSEL	O	Output which will signal at which speed the IrDA is currently set. This signal is not necessary for IrDA operation. This pin will be driven high when IrDA is configured for FIR or MIR. This pin will be driven low for SIR mode. Muxed with GPIO[15] which controls the pin out of hardware reset, runtime reset and Sleep.

For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

## 6.4.3 Programming Considerations

### 6.4.3.1 Initialization

First the IrDA clock must be set to match the CS setting in the **ir\_config2** register. Please see Section 7.1 "Clocks" on page 194 for more information.

Second, enable peripheral logic by programming the **ir\_enable** register: HC should be set to 1 for low power or if the System Bus (SBUS) is greater than 100 MHz, CE must be set to 1 to enable the peripheral logic, C should be set to 1 for dcache to respond to irda accesses on the SBUS if it has the data, and E should be set for the appropriate endianness.

Next, the **sys\_pinfunc** register bits must be set to the alternate (IrDA) function: IRF can optionally be set to 1 to enable IrDA to drive the FIRSEL pin (this pin is not required if external logic takes care of setting the transceiver speed). IRD must be set to 0 to enable data transmission through the IRTXD pin.

### 6.4.3.2 Power Management

The HC bit in the **ir\_enable** register can be used to run the IRDA at half the SBUS. The CE should be disabled when not using the IRDA to gate clocks from this peripheral.

### 6.4.3.3 Programming Notes

IrDA can be operated at speeds ranging from 2400 bps to 4 Mbps. Table 6-15 shows the proper parameters to configure communications speed and IrDA mode.

**Table 6-15. IrDA PHY Configuration Table**

Mode	Speed (bps)	Baud Rate	Pulse Width			Preamble/Start Flags
			Min	Nom	Max	
SIR	2400	47	0	12	12	N/A
SIR	9600	11	0	12	12	N/A
SIR	19200	5	1	12	12	N/A
SIR	38400	2	3	12	14	N/A
SIR	57600	1	5	12	16	N/A
SIR	115200	0	11	12	20	N/A
MIR	1150000	0	N/A	8	N/A	2 (P field = 1)
FIR	4000000	0	N/A	N/A	N/A	16 (P field = 15)

Table 6-16, Table 6-17 and Table 6-18 show the ordered steps for programming the IrDA peripheral for each mode.

**Table 6-16. Fast Infrared Mode (FIR)**

Step	Register	Value	Notes
1	ir_enable	0x000E	Enable half clock speed (HC), clocks (CE), coherency (C), and little endian (E).
2	ir_statusen	0x0000	Clear bit E to allow peripheral programming (disable IrDA).
3	ir_maxpktlen	0x0020	32 bytes maximum per packet
4	ir_wrphyscfg	0x000F	16 preamble bytes (P field requires 1 less than number needed)
5	ir_config1	0x1C40	Enable transmitter (TE), receiver (RE), memory scheduler (ME), and fast infrared mode (FI). <b>Note:</b> Set pin inversion bits (TI and/or RI) accordingly for proper transceiver operation.
6	ir_rngbsadrl	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
7	ir_rngbsadrh	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
8	ir_ringsize	user defined	Write the desired ring size.
9	ir_config2	0x0004	Set the PHY clock speed to 48 MHz.
10	ir_statusen	0x8000	Set bit E to enable the peripheral, then read register again for correct status (should equal 0xA6FF).
11	ir_rngprompt	0x0000	Write a zero to this register to start the IrDA transfers.

Table 6-17. Medium Infrared Mode (MIR)

Step	Register	Value	Notes
1	ir_enable	0x000E	Enable half clock speed (HC), clocks (CE), coherency (C), and little endian (E).
2	ir_statusen	0x0000	Clear bit E to allow peripheral programming (disable IrDA).
3	ir_maxpktlen	0x0020	32 bytes maximum per packet
4	ir_wrphycfg	0x0101	1 preamble byte (P field requires one less than number needed), Pulse Width = 8
5	ir_config1	0x1C20	Enable transmitter (TE), receiver (RE), memory scheduler (ME), and medium infrared mode (MI). <b>Note:</b> Set pin inversion bits (TI and/or RI) accordingly for proper transceiver operation.
6	ir_rngbsadrl	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
7	ir_rngbsadrh	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
8	ir_ringsize	user defined	Write the desired ring size.
9	ir_config2	0x0004	Set the PHY clock speed to 48 MHz.
10	ir_statusen	0x8000	Set bit E to enable the peripheral, then read register again for correct status (should equal 0x96FF).
11	ir_rngprompt	0x0000	Write a zero to this register to start the IrDA transfers.

Table 6-18. Slow Infrared Mode (SIR)

Step	Register	Value	Notes
1	ir_enable	0x000E	Enable half clock speed (HC), clocks (CE), coherency (C), and little endian (E).
2	ir_statusen	0x0000	Clear bit E to allow peripheral programming (disable IrDA).
3	ir_maxpktlen	0x0020	32 bytes maximum per packet
4	ir_wrphycfg	0x0180	Baudrate = 0 (115200), Pulse width = 12
5	ir_config1	0x1E10	Enable transmitter (TE), receiver (RE), memory scheduler (ME), receive all runt packets (RA), and slow infrared mode (SI). Note: set pin inversion bits (TI and/or RI) accordingly for proper transceiver operation.
6	ir_rngbsadrl	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
7	ir_rngbsadrh	user defined	Write the physical address of ring buffer memory. <b>Note:</b> The final address must have zeros for address bits 9:0 (i.e. the address must reside on a 1 KByte boundary).
8	ir_ringsize	user defined	Write the desired ring size.
9	ir_config2	0x0004	Set the PHY clock speed to 48 MHz.
10	ir_statusen	0x8000	Set bit E to enable the peripheral, then read register again for correct status (should equal 0x8EFF).
11	ir_rngprompt	0x0000	Write a zero to this register to start the IrDA transfers.

#### 6.4.3.4 Ring Buffers

The IrDA controller is designed to allow the CPU to access the IR media through a system of “rings” set up in memory. Each ring entry corresponds to a LAN packet and stores information and status about that packet as well as the physical address of where the data for that packet is stored. The ring area is split into two areas: Transmit and Receive. The receive ring starts at the Base Address location (specified by the contents of the ring base address registers) and the transmit ring starts at the Base Address + 512 bytes (decimal). Each ring entry contains 8 bytes with a maximum of 64 ring entries in each of the transmit and/or receive ring areas. The actual number of entries used is programmed via the **ir\_ringsize** register.

The format for each transmit ring entry is shown in Figure 6-2.

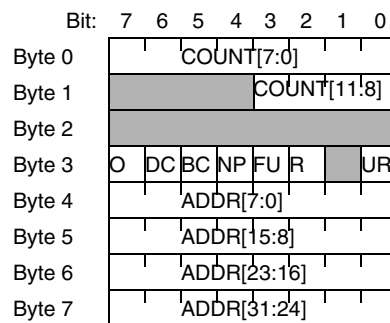


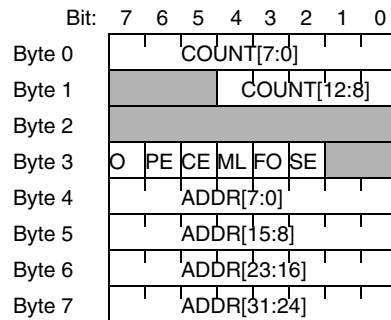
Figure 6-2. Transmit Ring Buffer Entry Format



Table 6-19. Transmit Ring Buffer Entry Format Description

Bits	Name	Description	R/W	Default
Byte 0: bits 7:0	COUNT[7:0]	Number of bytes to transmit (lowest 8 bits).	R/W	User Assigned
Byte 1: bits 7:4	—	Reserved, read/written as 0.	R/W	0
Byte 1: bits 3:0	COUNT[11:8]	Number of bytes to transmit (upper 4 bits).	R/W	User Assigned
Byte 2: bits 7:0	—	Reserved, read/written as 0.	R/W	0
Byte 3: bit 7	O	Ownership flag. 0 User has ownership of the packet. 1 Hardware has ownership of the packet and is sending the packet data to the transmitter. Hardware clears this bit when the packet has been sent.	R/W	User Assigned
Byte 3: bit 6	DC	Disable the transmit CRC. 0 Used for synchronous packet operation. 1 Used by IrDA SIR mode. Hardware clears this bit when the packet has been sent.	R/W	User Assigned
Byte 3: bit 5	BC	Force a bad CRC. 0 Normal CRC operation. 1 Send an 'invalid' CRC flag in the packet. Used to test receiver CRC checking. Hardware clears this bit when the packet has been sent.	R/W	User Assigned
Byte 3: bit 4	NP	Need an indication pulse. 0 Normal operation. 1 Transmit an indication pulse after the packet has been transmitted. Hardware clears this bit when the packet has been sent.	R/W	User Assigned
Byte 3: bit 3	FU	Force an underrun condition. 0 Normal operation. 1 Force an underrun on this packet. Packet size must be greater than 18 bytes. Used for testing only.	R/W	User Assigned
Byte 3: bit 2	R	Request to disable transmitter. 0 Normal operation. 1 Hardware will clear <b>ir_config1</b> transmit enable (TE) bit after this packet has been transmitted. Used to shut down the transmitter immediately after the last packet.	R/W	User Assigned
Byte 3: bit 1	—	Reserved, read/written as 0.	R/W	0
Byte 3: bit 0	UR	Hardware Underrun error. This bit is set if a hardware underrun occurs during transmission of a packet. Used only to find hardware errors.	R	0
Byte 4: bits 7:0	ADDR[7:0]	Address of data to transmit (bits [7:0]).	R/W	User Assigned
Byte 5: bits 7:0	ADDR[15:8]	Address of data to transmit (bits [15:8]).	R/W	User Assigned
Byte 6: bits 7:0	ADDR[23:16]	Address of data to transmit (bits [23:16]).	R/W	User Assigned
Byte 7: bits 7:0	ADDR[31:24]	Address of data to transmit (bits [31:24]).	R/W	User Assigned

The format for each receive ring entry is described in Figure 6-3.



**Figure 6-3. Receive Ring Buffer Entry Format**

**Table 6-20. Receive Ring Buffer Entry Format Description**

Bits	Name	Description	R/W	Default
Byte 0: bits 7:0	COUNT[7:0]	Number of bytes received (lowest 8 bits)	R/W	User Assigned
Byte 1: bits 7:5	—	Reserved, read/written as 0.	R/W	0
Byte 1: bits 4:0	COUNT[12:8]	Number of bytes received (upper 5 bits)	R/W	User Assigned
Byte 2: bits 7:0	—	Reserved, read/written as 0.	R/W	0
Byte 3: bit 7	O	Ownership flag. 0 User has ownership of the packet. 1 Hardware has ownership of the packet and is writing packet data from the receiver to memory. Hardware clears this bit when the packet has been received.	R/W	User Assigned
Byte 3: bit 6	PE	PHY layer error detected.	R/W	User Assigned
Byte 3: bit 5	CE	CRC error detected. Valid for FIR and MIR modes only.	R/W	User Assigned
Byte 3: bit 4	ML	Maximum packet length reached. For SIR mode, data will continue to be received in adjacent packets. However, for FIR and MIR modes, subsequent data will be dropped.	R/W	User Assigned
Byte 3: bit 3	FO	Internal hardware FIFO overflow. This should not occur under normal operation.	R/W	User Assigned
Byte 3: bit 2	SE	SIR error detected. If the SIR filter is enabled, this flag will be set if an end flag is not received.	R/W	User Assigned
Byte 3: bits 1:0	—	Reserved, read/written as 0.	R/W	0
Byte 4: bits 7:0	ADDR[7:0]	Address of data to receive (bits [7:0]).	R/W	User Assigned
Byte 5: bits 7:0	ADDR[15:8]	Address of data to receive (bits [15:8]).	R/W	User Assigned
Byte 6: bits 7:0	ADDR[23:16]	Address of data to receive (bits [23:16]).	R/W	User Assigned
Byte 7: bits 7:0	ADDR[31:24]	Address of data to receive (bits [31:24]).	R/W	User Assigned

On the transmit side the descriptors are set up and point to the data associated with them. Each buffer has an ownership bit that tells the hardware it has been given control of that buffer. When the hardware has finished with a buffer it will clear the 'O' bit. If polling this is how software can tell whether a receive or transit is done. When using interrupts, when the hardware is finished either transmitting or receiving an interrupt will be generated if they are enabled in the **ir\_config2** register. See Section 5.0 "Interrupt Controller" on page 83.

Buffers are in a ring structure and are always accessed in sequence. Once the controller reaches a buffer in which the ownership bit is not set, it will stop the chaining at that point and will require the processor to "PROMPT" it to look at the buffer again and restart the chaining.

## 6.5 Ethernet MAC Controller

The Au1100 processor contains one Ethernet MAC device. The MAC provides the interface between the host application and the PHY layer through the Media Independent Interface (MII). The PHY layer device is external to the processor.

The MAC supports the protocol requirements to meet the Ethernet/IEEE 802.3 specification. The MAC operates in both half and full duplex modes. In half duplex mode the MAC is compliant with section 4 of ISO/IEC 8802-3 (ANSI/IEEE Standard) and ANSI/IEEE 802.3.

The MAC provides programmable enhanced features designed to minimize host supervision, bus utilization and pre/post message processing. These features include ability to disable retries after a collision, dynamic FCS generation on a frame by frame basis, automatic pad field insertion and deletion to enforce minimum frame size attributes, automatic retransmission and detection of collision frames. The MAC can sustain transmission or reception of minimal size back to back packets at full line speed with an inter-packet gap of 9.6  $\mu$ s for 10 Mbps and 0.96  $\mu$ s for 100 Mbps.

A dedicated DMA engine is implemented to support the MAC so that the general purpose DMA is not required.

The primary attributes of the MAC are:

- Transmit and receive message data encapsulation with framing and error detection.
- Frame boundaries are delimited and frames are synchronized. Error detection is done at the physical medium transmission level.
- Media access management is supported through medium allocation and contention resolution. This is accomplished through collision avoidance and handling. The MAC handles collision per the ISO 8802.3 specification.
- Support for flow control during full duplex mode is accomplished by decoding of control frames and disabling the transmitter in conjunction with generation of control frames.
- The serial control interface supports the MII protocol to interface to an MII based PHY.

The MAC features are:

- IEEE 802.3, 802.3u, 803.3x specification compliance
- 10/100 Mbps data transfer rates
- IEEE 802.3 compliant MII interface to talk to an external PHY
- Full and half duplex
- CSMA/CD in half duplex
- Flow control support for full duplex
- Collision detection and auto retransmit on collisions in half duplex
- Preamble generation and removal
- Automatic 32-bit CRC generation and checking
- Optional automatic Pad stripping on the receive packets.
- Loopback support on the MII
- Filtering modes supported on the Ethernet side:
  - - One 48-bit perfect address
  - - 64 hash-filtered multicast addresses
  - - Pass all multicast addresses
  - - Promiscuous Mode
  - - Pass all incoming packets with a status report
  - - Toss bad packets
- Separate 32 bit status returned for transmit and receive packets
  - Jumbo packet (0x2800 bytes)
  - Big/Little Endian data format support
- The following PHY interfaces are supported:
  - MII - Ethernet 4-bit parallel PHY per IEEE 802.3u spec
  - MII Management - 2-wire bus to control and receive status from PHY
  - HPNA 1.0 support across MII

The control registers for the MAC are used for address filtering, packet filter for good and bad frames, 48-bit MAC address with a local station address, a multicast table for filtering multicast frames and more. Each register is 32 bits wide.

### 6.5.1 Ethernet Base Address Registers

The Ethernet MAC contained in the Au1100 processor is located at the base address shown in Table 6-21. In addition, the base addresses for the enable register and the MAC DMA registers are shown.

**Table 6-21. Ethernet Base Addresses**

Name	Physical Base Address	KSEG1 Base Address
mac0_base	0x0 1050 0000	0xB050 0000
macen_base	0x0 1052 0000	0xB052 0000
macdma0_base	0x0 1400 4000	0xB400 4000

### 6.5.2 MAC Configuration Registers

The Ethernet MAC registers (offset from **mac0\_base**) are listed in Table 6-22.

**Table 6-22. MAC Register Descriptions**

Offset (Note 1)	Register Name	Description
0x0000	mac_control	Operation Mode and address filter
0x0004	mac_addrhigh	High 16 bits of the MAC physical address
0x0008	mac_addrlow	Lower 32 bits of the MAC physical address
0x000C	mac_hashhigh	High 32 bits of the Multicast hash address
0x0010	mac_hashlow	Low 32 bits of the Multicast hash address
0x0014	mac_miictrl	Control of PHY management interface
0x0018	mac_miidata	Data to be written or read from PHY over control interface
0x001C	mac_flowctrl	Control Frame Generation Control
0x0020	mac_vlan1	VLAN1 Tag
0x0024	mac_vlan2	VLAN2 Tag

Note 1. See Table 6-21 for base address.

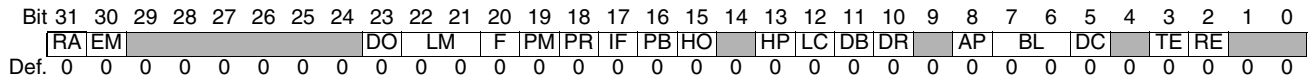
6.5.2.1 MAC Control Register

The MAC Control Register establishes the receive and transmit operating modes and controls for address filtering and packet filtering.

Note that the PM, PR, IF, HP and HO bits in the MAC Control register will determine the address filtering mode. The RA, DB, PC and PB bits will determine the packet filter mode. The first bit of the destination address will determine if the address is a physical address (first bit = 0) or a multicast address (first bit = 1). If all bits in the destination address are set to 1 then the address is a broadcast address.

mac\_control

Offset = 0x0000



Bits	Name	Description	R/W	Default
31	RA	Receive All. 0 Normal operation. 1 All incoming packets will be received regardless of the destination address. The address filter status is reported in Receive Status bit Filtering Fail. The Packet Filter bit in the Receive Status is set for all error-free frames regardless of the Destination Address field.	R/W	0
30	EM	Endian mode for data buffers. 0 Little endian. 1 Big endian.	R/W	0
29:24	—	Reserved, should be cleared.	R	0
23	DO	Disable Receive Own. 0 The MAC receives all packets that are given by the PHY. 1 The MAC disables reception of frames when the TXEN is asserted. The MAC ignores any loop backed receive packets. This bit should be cleared when the full duplex mode bit is set or the Operating Mode is set to other than Normal Mode.	R/W	0
22:21	LM	Loopback Operating Mode. 00 Normal mode. 01 Internal loopback. 10 External loopback. 11 Reserved.	R/W	00
20	F	Full Duplex Mode. 0 Half duplex mode. 1 Full duplex mode. <b>Note:</b> Be sure to disable both the transmitter and receiver before changing duplex modes.	R/W	0
19	PM	Pass All Multicast. 0 Normal 1 All incoming frames with a multicast destination address (first bit in the destination address field is '1') are received and the Filter Fail bit reset. Incoming frames with physical address destinations are filtered according to HP (bit 13) and HO (bit 15).	R/W	0
18	PR	Promiscuous Mode. 0 Normal operation. 1 Any incoming valid frame is received regardless of its destination address. The Filter Fail bit is always cleared in Promiscuous Mode.	R/W	1

Bits	Name	Description	R/W	Default
17	IF	<p>Inverse Filtering.</p> <p>0 Normal operation.</p> <p>1 Physical addresses are checked with inverse filtering. In other words if the address passes a perfect address filter, the frame is not passed; if the address fails a perfect filter, the frame is passed.</p> <p>This is valid only during perfect filtering mode.</p>	R/W	0
16	PB	<p>Pass Bad Frames.</p> <p>0 Normal operation.</p> <p>1 All incoming frames that passed the address filtering are received including runt frames, collided frames, or truncated frames caused by buffer overflow.</p> <p>The Packet Filter bit is set for error frames that pass the Address filtering. If all received bad frames are required, promiscuous mode (bit 18) should be set.</p>	R/W	0
15	HO	<p>Hash Only Filtering Mode.</p> <p>0 Perfect address filtering mode for physical addresses</p> <p>1 Imperfect address filtering mode both for physical and multicast addresses</p> <p>Setting this bit is valid only if HP=1.</p>	R/W	0
14	—	Reserved, should be cleared.	R	0
13	HP	<p>Hash/Perfect Filtering Mode.</p> <p>0 Address Check block does a perfect address filter of incoming frames according the address specified in the MAC Address register.</p> <p>1 Address Check block does imperfect address filtering of multicast incoming frames according to the hash table specified in the multicast Hash Table Register. If the Hash Only (HO) bit is set, then physical addresses are imperfectly filtered too. If the Hash Only bit (HO) is reset, then physical addresses are perfect address filtered according to the MAC Address Register.</p>	R/W	0
12	LC	<p>Late Collision Control.</p> <p>0 Abort frame transmission on a late collision.</p> <p>1 Enable the retransmission of the collided frame even after the collision period (late collision).</p> <p>In either case the Late Collision Status is appropriately updated in the Transmit Packet Status.</p> <p>This bit is valid only when operating in half duplex mode.</p>	R/W	0
11	DB	<p>Disable Broadcast Frames.</p> <p>0 Forward all the broadcast frames to the application. (Packet Filter bit is set.)</p> <p>1 Disable the reception of broadcast frames. (Packet Filter bit is cleared.)</p>	R/W	0
10	DR	<p>Disable Retry.</p> <p>0 The MAC will attempt 16 transmissions before signaling a retry error.</p> <p>1 The MAC will attempt transmission of a frame only once. When a collision is seen on the bus, the MAC will ignore the current frame and go to the next frame and a retry error will be reported in the Transmit Status.</p> <p>This bit is valid only when operating in half duplex mode.</p>	R/W	0
9	—	Reserved, should be cleared.	R	0

Bits	Name	Description	R/W	Default
8	AP	<p>Automatic Pad Stripping.</p> <p>0 Pass all the incoming frames to the host unmodified.</p> <p>1 Strip the pad field on all the incoming frames if the length field is less than 46 bytes. The FCS field is also stripped, because it is computed at the transmitting station based on the data and pad field characters and will therefore be invalid for a receive frame that has had the pad characters stripped. Receive frames which have a length field of 46 bytes or greater will be passed to the host unmodified (FCS is not stripped).</p> <p>Pad stripping is done only on the IEEE 802.3 formatted frames (frames with Length field).</p>	R/W	0
7:6	BL	<p>Backoff Limit. The Backoff limit determines the integer number of slot times the MAC waits before rescheduling a transmission attempt (during retries after a collision).</p>	R/W	00
5	DC	<p>Deferral Check.</p> <p>0 The deferral check is disabled in the MAC and the MAC defers indefinitely.</p> <p>1 The deferral check is enabled in the MAC. The MAC will abort the transmission attempt if it has deferred for more than 24,288 bit times. Deferring starts when the transmitter is ready to transmit, but is prevented from doing so because CRS is active. Defer time is not cumulative. In other words, if the transmitter defers, then transmits, collides, backs off, and then has to defer again after completion of backoff, the deferral timer resets to 0 and restarts.</p> <p>This bit is valid only when operating in half duplex mode.</p>	R/W	0
4	—	Reserved, should be cleared.	R	0
3	TE	<p>Transmitter Enable.</p> <p>0 The MAC transmitter is disabled and will not transmit any frames on the MII interface.</p> <p>1 The MAC transmitter is enabled and it will transmit frames from the buffer on to the MII interface.</p>	R/W	0
2	RE	<p>Receiver Enable.</p> <p>0 The MAC receiver is disabled and will not receive any frames from the MII interface.</p> <p>1 The MAC receiver is enabled and will receive frames from the MII interface.</p>	R/W	0
1:0	—	Reserved, should be cleared.	R	0



### 6.5.2.2 MAC Address High and Low Registers

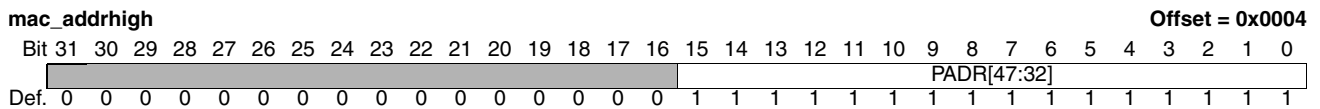
The MAC Address High Register contains the upper 16 bits of the physical address of the MAC. The MAC Address Low Register contains the lower 32 bits of the physical address of the MAC.

It is the responsibility of the system designer to provide the MAC address for the system.

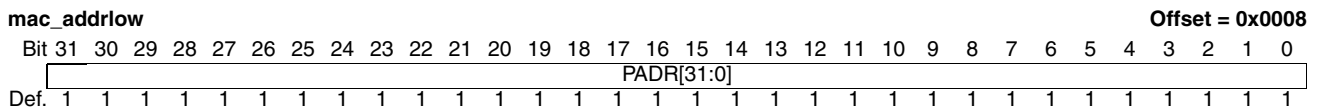
The MAC address will be compared with the destination address from the incoming frame with PADR[0] (bit 0 of the Mac Address Low register) being compared with the first bit of the destination address and PADR[47] (bit 15 of the MAC Address High register) compared with the 48th bit of the destination address.

Example: To program the MAC address 00.50.c2.0c.20.10 the MAC address registers should be programmed as follows:

```
mac_addrhigh = 0x00001020
mac_addrlow = 0x0CC25000
```



Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	PADR[47:32]	Physical Address [47:32]. Contains the upper 16 bits (47 to 32) of the Physical Address of the MAC.	R/W	0xFFFF

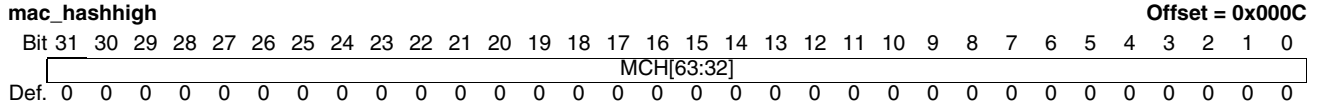


Bits	Name	Description	R/W	Default
31:0	PADR[31:0]	Physical Address [31:0]. Contains the lower 32 bits (31 to 0) of the Physical Address of the MAC.	R/W	0xFFFFFFFF

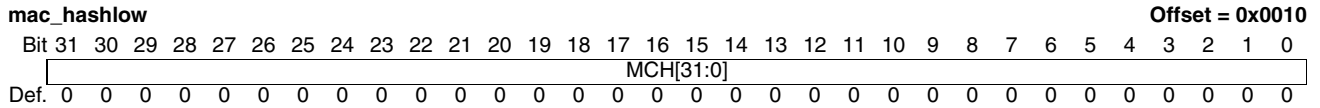
**6.5.2.3 Multicast Address High Hash and Low Hash Table Register**

The 64-bit multicast address hash table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (1 = Hi, 0 = Low), while the other five bits determine the bit within the register. A value of '00000' selects the bit 0 of the selected register and a value of '11111' selects the bit 31 of the selected register.

If the corresponding bit in the hash table is '1', then the multicast frame is accepted, otherwise it is rejected. If the Pass All Multicast is set, then all multi-cast frames are accepted regardless of the multi-cast hash values. The Multi Cast Hash Table High Register contains the higher 32 bits of the hash table and the Multi Cast Hash Table Low Register contains the lower 32 bits of the hash table.



Bits	Name	Description	R/W	Default
31:0	MCH[63:32]	Multicast Address Hash Table High. These bits map to the upper 32 bits of the 64-bit hash table.	R/W	0x00000000



Bits	Name	Description	R/W	Default
31:0	MCH[31:0]	Multicast Address Hash Table Low. These bits map to the lower 32 bits of the 64-bit hash table.	R/W	0x00000000

### 6.5.2.4 MII Control Register

The MII Address Register is used to control and generate the Management cycles to the External PHY Controller chip. A write to this register will generate a read/write access on the MII Management Interface (MDIO/MDC) bus to an external PHY device.

**mac\_miictrl**

**Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	[Reserved]																PHYADDR[4:0]					MIIREG[4:0]				MW		MB				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:11	PHYADDR	PHY Address. These bits tell which of the 32 possible PHY devices are being accessed.	R/W	00000
10:6	MIIREG	MII register. These bits select the desired MII register in the selected PHY device.	R/W	00000
5:2	—	Reserved, should be cleared.	R	0
1	MW	MII Write. 0 Operation will be a read (data read is placed in MII Data Register) 1 Operation will be a write (data to be written is taken from MII Data Register)	R/W	0
0	MB	MII Busy. This bit should read a logic 0 before writing to the MII address and MII data registers. This bit should be reset to 0 when writing to the MII address register.  This bit will be set by the MAC to signify that a read or write access to the external PHY is in progress. For a write operation the data register should be kept valid until this bit is cleared by the MAC. For a read operation the MII data register is invalid until this bit is cleared by the MAC.  The MII address register should not be modified until this bit is cleared. The MAC clears this bit after the PHY access is done.	R/W	0

### 6.5.2.5 MII Data Register

The MII Data Register contains the data to be written to the PHY register specified in the MII address register, or it contains the read data from the PHY register whose address is specified in the MII address register.

**mac\_miidata**

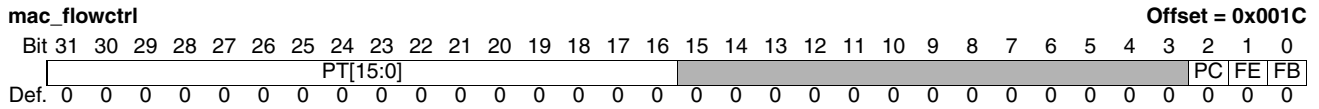
**Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	[Reserved]																MIIDATA[15:0]															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	MIIDATA	MII Data. 16-bit value read from the PHY after a MII read operation, or the 16-bit data value to be written to the PHY before a MII write operation.	R/W	0x0000

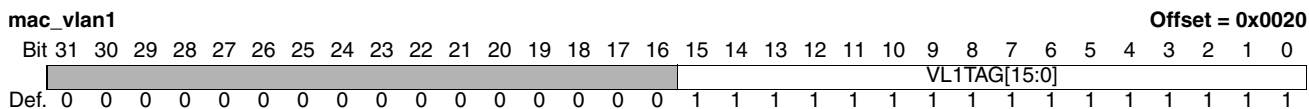
**6.5.2.6 Flow Control Register**

This register is used to control the generation and reception of the Control (PAUSE Command) frames by the MAC's Flow control block. A write to this register with the busy bit set to '1' triggers the Flow Control block to generate a PAUSE Control frame. The fields of the control frame are selected as specified in the 802.3x specification with the Pause Time field from this register used in the "Pause Time" field of the control frame. The Busy bit will remain set until the control frame is transmitted. The Host has to insure that the Busy bit is cleared before writing to the register. The Pass Control Frames bit indicates to the MAC whether or not to pass the control frame to the Host. The Flow Control Enable bit enables the receive portion of the Flow Control block.



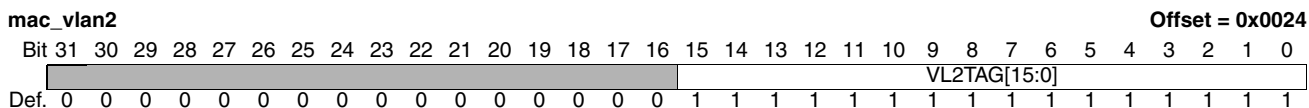
Bits	Name	Description	R/W	Default
31:16	PT	Pause Time. This field will be used in the PAUSE TIME field in the generation of the PAUSE control frame.	R/W	0x0000
15:3	—	Reserved, should be cleared.	R	0
2	PC	Pass Control Frame 0 The MAC decodes the control frames but does not pass the frames to the Host. The Control Frame bit in the Receive Status (bit 25) is set and the Transmitter Pause Mode signal gives the current status of the Transmitter, but the PacketFilter bit in the Receive Status is reset to signal the application to flush the frame. 1 Control frames are passed to the Host. The MAC decodes the control frame (PAUSE) and disables the transmitter for the specified amount of time. The Control Frame bit in the Receive Status (bit 25) is set, and the Transmitter Pause Mode signal indicates the current state of the MAC Transmitter.	R/W	0
1	FE	Flow Control Enable 0 The flow control operation in the MAC is disabled, and the MAC does not decode the frames for control frames. 1 The MAC is enabled for flow control operation and it will decode all the incoming frames for control frames. If the MAC receives a valid control frame (PAUSE command), it will disable the transmitter for the specified time. This bit is valid only in full duplex mode.	R/W	0
0	FB	Flow Control Busy Status. This bit should read a logic 0 before writing to the Flow Control register. To initiate a PAUSE control frame the host must set this bit. During a transfer of Control Frame, this bit remains set to signify that a frame transmission is in progress. After the completion of PAUSE control frame transmission, the MAC clears FB.	R/W	0

### 6.5.2.7 VLAN1 Tag Register



Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	VL1TAG	VLAN 1 Tag Identifier This field will be compared with the 13th and 14th bytes of the incoming frame. If a nonzero match occurs the VLAN 1 Frame bit will be set in the receiver status packet. In addition, the legal length of a frame is increased from 1518 bytes to 1522 bytes.	R/W	0xFFFF

### 6.5.2.8 VLAN2 Tag Register



Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	VL2TAG	VLAN 2 Tag Identifier This field will be compared with the 13th and 14th bytes of the incoming frame. If a nonzero match occurs the VLAN 2 Frame bit will be set in the receiver status packet. In addition the legal length of a frame is increased from 1518 bytes to 1538 bytes.	R/W	0xFFFF

### 6.5.3 MAC Enable Registers

The Ethernet MAC has an enable register located off of the **macen\_base** shown in Table 6-21 on page 125.

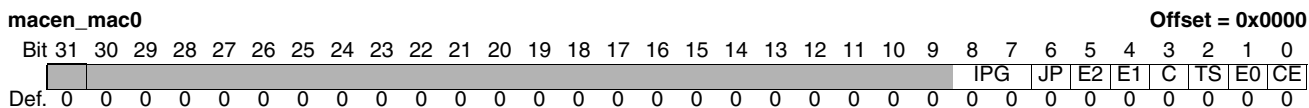
#### 6.5.3.1 MAC0 Enable

The enable register contains a bit that enables the entire block. The block should be disabled if not in use to minimize power consumption. In addition, the enable register contains a toss bit (TS) which prevents frames that do not pass the address filter from being put into memory.

The process for bringing the MAC out of reset is as follows:

- 1) Enable clocks (CE=1).
- 2) Bring E[2:0] high together with the other bits configured as desired (keeping clocks enabled).

**Note:** MAC clocks must be running before the internal MAC registers are accessed.



Bits	Name	Description	R/W	Default
31:9	—	Reserved, should be cleared.	W	0
8:7	IPG	Inter-Packet Gap. Determines the inter-packet gap measured in bit times. For example, the minimum IPG for 100 Mbps full-duplex Ethernet is 960 bit times (0.960 $\mu$ s). 00 1080 (Default) 01 1040 10 1000 11 960 (for IEEE 802.3 minimum IPG compliance)	W	0
6	JP	Jumbo Packet Enable 0 Normal (Max packet length = 0x800 bytes). 1 Enable Jumbo Packet (Max packet length = 0x2800 bytes).	W	0
5:4	E[2:1]	Enable field bits 2 and 1. Together with E0, this field resets and enables the MAC. 000 Reset. 111 Enable. All other combinations are invalid.	W	00
3	C	Coherent 0 Memory accesses are marked coherent on SBUS. 1 Memory accesses are marked non coherent on SBUS. For more information on coherency see Section 2.8.2 "SBUS Coherency Model" on page 41.	W	0
2	TS	Disable Toss. 0 Only frames passing the address filter are passed to memory. Frames which fail length error, CRC error, or other non-address filter failures are still passed to memory. Frames are not passed to memory if the filter fail bit is set, or the frame is a broadcast frame and broadcast frames have been disabled. In promiscuous mode all frames are passed to memory unless the disable broadcast bit is set which prevents broadcast frames from being passed to memory. Frames that are not passed to memory are transparent to software—no status or indication informs software. 1 All frames are passed to memory, regardless of address filter result.	W	UNPRED
1	E[0]	Enable field bit 0. See description for E[2:1].	W	0
0	CE	Clock Enable. 0 Clocks disabled to MAC. 1 Clocks enabled to MAC.	W	0

### 6.5.4 MAC DMA Registers

The MAC has four DMA buffers for both receive and transmit (four for RX, four for TX). The DMA buffers are serviced in a round-robin fashion. The MAC has a 32-word FIFO for both transmit and receive. The transfer size for the MAC DMA is eight words. Both the FIFO size and transfer size are taken care of automatically by the MAC DMA and are transparent to the programmer except that all memory buffers must be implemented on a cache line boundary (32 bytes).

The MAC DMA base address contains eight entries which correspond to four transmit buffer entries and four receive buffer entries as shown in Table 6-23.

**Table 6-23. MAC DMA Entries**

Offset (Note 1)	Entry Prefix	Entry Name
0x000	tx0	Transmit Buffer 0
0x010	tx1	Transmit Buffer 1
0x020	tx2	Transmit Buffer 2
0x030	tx3	Transmit Buffer 3
0x100	rx0	Receive Buffer 0
0x110	rx1	Receive Buffer 1
0x120	rx2	Receive Buffer 2
0x130	rx3	Receive Buffer 3

Note 1. See Table 6-21 on page 125 for base address.

Within each receive entry there are two registers implemented as shown in Table 6-24. (The third and fourth reserved entries are shown for completeness but are not used.)

**Table 6-24. MAC DMA Receive Entry Registers**

Offset (Note 1)	Receive Entry Register	Description
0x0	stat	Status register
0x4	addr	Address/enable register
0x8	Reserved	Nothing is implemented at this offset.
0xC	Reserved	Nothing is implemented at this offset.

Note 1. See Table 6-21 on page 125 for base address.

Within each transmit entry, there are three registers implemented as shown in Table 6-25. (The fourth reserved entry is shown for completeness but is not used.)

**Table 6-25. MAC DMA Transmit Entry Registers**

Offset (Note 1)	Transmit Entry Register	Description
0x0	stat	Status register
0x4	addr	Address/enable register
0x8	len	Length register
0xC	Reserved	Nothing is implemented at this offset.

Note 1. See Table 6-21 on page 125 for base address.

To calculate the address of a specific MAC DMA buffer all offsets should be combined. For example the physical address of the MAC0 receive buffer 3 address register is calculated as follows:

$$\begin{aligned} \text{macdma0\_rx3addr} &= \text{macdma0\_base} + \text{rx3} + \text{addr} \\ &= 0x0\ 1400\ 4000 + 0x130 + 0x4 \\ &= 0x0\ 1400\ 4134 \end{aligned}$$

Another way to look at the DMA register addresses is to view them as built off of the base address using an indexed approach to build the address for each unique register within the block. In other words, each bit (or set of bits) within the address will select a parameter of the DMA Register (TX/RX, Buffer number, Status/Address/Length register) until a unique address is formed selecting a single register.

To build the address for a unique register the bits should be set according to the definitions in Table 6-26.

**Table 6-26. MAC DMA Block Indexed Address Bit Definitions**

AddrBits	Description
8	TX/RX. 0 Transmit Block. 1 Receive Block.
7:6	These bits should be cleared.
5:4	MAC DMA Buffer. 00 Buffer 0. 01 Buffer 1. 10 Buffer 2. 11 Buffer 3.
3:2	Register Select. 00 Status Register. 01 Address/Enable Register. 10 Length Register (valid for transmit only). 11 Reserved.
1:0	These bits should be cleared because the registers are aligned on a word boundary.

The enumerated DMA registers are shown in Table A-4 "Device Memory Map" on page 305.



### MAC DMA Receive Registers

There are two receive registers for each DMA channel associated with each MAC: the status register and the address/enable register. The length register is not applicable to the receive DMA channel, as the length will be determined by the size of the received packet (typically the size of a frame for a complete, successful reception). The receive memory buffers should be 0x800 bytes when Jumbo Packets are not enabled and to 0x2800 when Jumbo packets are enabled. This will allow for the worst case maximum reception length.

#### 6.5.4.1 Receive Status

This register contains the receive packet status bits sent by the MAC after receiving a frame. This register is only valid after a receive transaction has been enabled by the host and the done bit has been set by the MAC in the Address/Enable Register to indicate that the transaction is complete.

The MI bit should be checked by software after receiving a frame to verify that the received frame is valid.

macdma0\_rxnstat

offset = 0x0

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
	MI	PF	FF	BF	MF	UC	CF	LE	V2	V1	CR	DB	ME	FT	CS	FL	RF	WT	L[13:0]																		
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					

Bits	Name	Description	R/W	Default
31	MI	Missed Frame. 0 The frame is received normally by the Application without any latency or error violations 1 Indicates that a frame was missed due to an internal FIFO overrun.	R	UNPRED
30	PF	Packet Filter. 0 Indicates that the current frame failed the packet filter. 1 Indicates that the current frame passed the packet filter that is implemented in the MAC.  Packet Filter will indicate failed frame when any of the following conditions happens. a. FF = 0 and frame is not a Broadcast or RA is 1 b. Frame is Broadcast and DB is 0 c. Frame is not Control Frame or PC is 1 d. No Error Status or PB Frames is 1 e. Unsupported Control Frame is 0  The Application can use this bit to decide whether to keep the packet in the memory or flush the packet from the memory.  Note that frames with length greater than max Ethernet size (1500 bytes-normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2) will create an error status thus failing the packet filter. The frames may still be valid with failure only due to frame size.	R	UNPRED
29	FF	Filtering Fail. 0 Current frame passed address filtering 1 Destination Address field in the current frame failed the Address filtering.	R	UNPRED
28	BF	Broadcast Frame. 0 Destination address is not Broadcast. 1 Destination address is all 1's indicating broadcast address.	R	UNPRED
27	MF	Multicast Frame. 0 Destination address is not multicast. 1 Destination address is multicast (the first bit is 1).	R	UNPRED

Bits	Name	Description	R/W	Default
26	UC	<p>Unsupported Control Frame.</p> <p>0 If the Control Frame bit is set, this bit indicates a supported control frame has been received (Pause Frame).</p> <p>1 The MAC observed an unsupported Control Frame. This is set when a control frame is received and the opcode field is unsupported, or the length is not equal to minFrameSize (64 bytes). This bit is set only when the MAC is operating in the full-duplex mode.</p>	R	UNPRED
25	CF	<p>Control Frame..</p> <p>0 Current frame is not a control frame.</p> <p>1 Current frame is a control frame. This bit is only set when operating in Full Duplex mode.</p>	R	UNPRED
24	LE	<p>Length Error</p> <p>0 No length error occurred.</p> <p>1 The current frame Length value is inconsistent with the total number of bytes received in the current frame. When the number of bytes received in the data field are more than what indicated in the Length/Type field, the additional bytes are assumed to be PAD bytes and the Length Error bit is not set. When the number of bytes received in the data field is less than what was indicated in the Length/Type field, the Length Error bit is set. This is valid when the Frame Type is set to '0' (802.3 Frame).</p> <p>This bit is not applicable for frame lengths greater than max Ethernet size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2).</p>	R	UNPRED
23	V2	<p>VLAN2 ID.</p> <p>0 No match with VLAN2 tag</p> <p>1 The current frame is tagged with a VLAN2 ID. The thirteenth and fourteenth bytes of the frame were a nonzero match with the VLAN2 tag register.</p>	R	UNPRED
22	V1	<p>VLAN1 ID.</p> <p>0 No match with VLAN1 tag</p> <p>1 The current frame is tagged with a VLAN1 ID. The thirteenth and fourteenth bytes of the frame were a nonzero match with the VLAN1 tag register.</p>	R	UNPRED
21	CR	<p>CRC Error.</p> <p>0 No CRC error in current frame</p> <p>1 CRC error occurred in received frame.</p> <p>This bit is not applicable for frame lengths greater than max Ethernet size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2). If a CRC check is required it must be done in software.</p>	R	UNPRED
20	DB	<p>Dribbling Bit.</p> <p>0 An integer multiple of eight bits was received.</p> <p>1 A non-integer multiple of eight bits was received. This bit is not valid if either the Collision Seen bit or Runt Frame bit is set. If this bit is set and the CRC Error bit is 0, then the packet is still valid.</p>	R	UNPRED
19	ME	<p>MII Error.</p> <p>0 No MII error</p> <p>1 MII error during frame reception</p>	R	UNPRED
18	FT	<p>Frame Type.</p> <p>0 IEEE 802.3 Frame</p> <p>1 Ethernet-type frame (frame length field is greater than max Ethernet size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2).</p> <p>This bit is still applicable when Jumbo packets are enabled.</p> <p>This bit is not valid for runt frames of less than 14 bytes.</p>	R	UNPRED

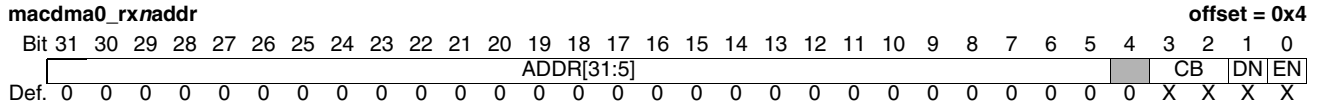
Bits	Name	Description	R/W	Default
17	CS	<p>Collision Seen.</p> <p>0 No collision seen during frame reception.</p> <p>1 The frame was damaged by a collision that occurred after the 64 bytes following the start of frame delimiter (SFD). This is a late collision.</p>	R	UNPRED
16	FL	<p>Frame Too Long.</p> <p>0 Frame size is less than or equal to max Ethernet frame size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2).</p> <p>1 Frame size is greater than the maximum Ethernet specified size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2). This also applies when Jumbo packets are enabled.</p> <p>Frame too long is only a length indication and does not cause frame truncation.</p>	R	UNPRED
15	RF	<p>Runt Frame.</p> <p>0 Frame was not damaged in collision window.</p> <p>1 Frame was damaged by a collision or premature termination before the collision window passed.</p>	R	UNPRED
14	WT	<p>Watchdog Timeout.</p> <p>0 Frame was received before timeout occurred.</p> <p>1 The receive watchdog timer expired while receiving the frame. The watchdog timer inside the MAC is programmed to be twice the Max-FrameLength. When set, the Frame Length field is invalid.</p> <p>Any time the max frame length is exceeded (0x800 bytes for normal mode, 0x2800 with Jumbo packets enabled) the WT bit will be set.</p>	R	UNPRED
13:0	L[13:0]	<p>Frame Length. Indicates length in bytes of the received frame. The host should take into account how the Automatic Pad Stripping (AP) bit in the corresponding MAC control register is set, as this will affect how the length field and frame contents should be interpreted.</p>	R	UNPRED

**6.5.4.2 Receive Buffer Address/Enable Register**

This register contains the starting address for the receive buffer. The host should ensure that the memory buffer is set up to accommodate the worst case largest frame size to be able to handle all received packets. At worst case the MAC will receive 0x800 bytes before aborting a receive in normal mode or 0x2800 bytes when Jumbo packets have been enabled in the **macen\_macn** register.

After the transaction has been enabled this register should not be written until the DN bit has been set.

The buffer for the DMA must be cache line aligned so the lowest 5 bits are not used as part of the address. These bits have been employed as done and enable bits that are exclusive of the address.



Bits	Name	Description	R/W	Default
31:5	ADDR	Buffer Address. Upper 27 bits of the starting physical address for the DMA buffer. This address must be cache line (32 bytes) aligned so only 27 bits are used. This address must be written for each DMA transaction (the address will not remain after the transaction is enabled)	R/W	0
4	—	Reserved, should be cleared.	R	0
3:2	CB	Current Buffer. Current DMA Receive Buffer	R	UNPRED
1	DN	Transaction Done. This bit will be set by hardware to indicate that the receive transaction has been completed and that the receive packet status is valid.  If the respective MAC DMA interrupt is enabled (see Section 5.0 "Interrupt Controller" on page 83), an interrupt will be generated when this bit is set. Done bits for all TX and RX buffers are or'ed together for this interrupt so a high level interrupt should be used.  This bit must be cleared explicitly by software after checking for done. This will also clear the interrupt.	R/W	UNPRED
0	EN	MAC DMA Enable. When set, this bit enables a DMA receive transaction to the memory location designated in ADDR.	R/W	UNPRED

**MAC DMA Transmit Registers**

There are three transmit registers, including the status register, the address/enable register, and the length register.

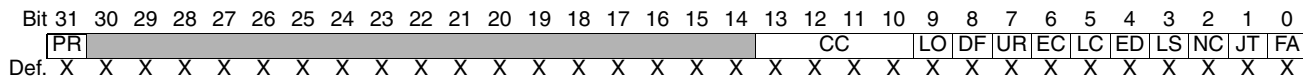
**6.5.4.3 Transmit Packet Status Register**

This register contains the transmit packet status bits sent by the MAC after transmitting a frame. This register is valid after a transmit transaction has been enabled by the host and the done bit has been set by the MAC in the Address/Enable Register to signify that the transmit transaction is complete.

If either PR (bit 31) or FA (bit 0) is set then the frame was not sent successfully and the application should resend the frame.

macdma0\_txrstat

offset = 0x0



Bits	Name	Description	R/W	Default
31	PR	Packet Retry. 0 Transmission of current packet is complete. 1 The Application has to restart the transmission of the frame (packet) when this bit is set to '1'. The successful/unsuccessful completion of the frame's transmission is indicated by the Frame Aborted bit (bit 0).	R	UNPRED
30:14	—	These bits are reserved.	R	UNPRED
13:10	CC	Collision Count. This 4-bit count indicates the number of collisions that occurred before the frame was transmitted. This bit is not valid when the Excessive Collisions bit is set. This bit is valid only when the MAC is operating in half-duplex mode.	R	UNPRED
9	LO	Late Collision Observed. 0 No late collision observed during transmission. 1 Indicates that the MAC observed a late collision (collision after 64 bytes into transmission of frame), but retransmitted the frame in the next retransmission attempt. This bit will be set when the Late Collision bit is set. This bit is valid only when the MAC is operating in half-duplex mode.	R	UNPRED
8	DF	Deferred. 0 Transmitter did not defer when transmitting. 1 The transmitter had to defer while ready to transmit a frame. This bit is valid only when operating in half-duplex mode.	R	UNPRED
7	UR	Under Run. 0 No data under run. 1 The transmitter aborted the message because of data under run during the frame's transmission.	R	UNPRED
6	EC	Excessive Collisions. 0 Transmission did not abort due to excessive collisions. 1 Transmission aborted after 16 successive collisions. If the Disable Retry bit is set, this bit is set after the first collision and the transmission of the frame will be aborted. This bit is valid only when operating in half-duplex mode.	R	UNPRED
5	LC	Late Collision. 0 No late collision. 1 Transmission was aborted due to collision occurring after the collision window of 64 bytes. This bit is not valid if under run error is set. This bit is valid only when operating in half-duplex mode.	R	UNPRED

Bits	Name	Description	R/W	Default
4	ED	Excessive Deferral. 0 No excessive deferral. 1 Transmission has ended because of excessive deferral of over 24,288 bit times during the transmission, if the defer bit is set high in the control register.  This bit is valid only when operating in half-duplex mode.	R	UNPRED
3	LS	Loss of Carrier. 0 No loss of carrier. 1 The loss of carrier occurred during the frame's transmission (i.e., the CRS input was inactive for one or more bit times when the frame is being transmitted).  This bit is valid only when operating in half-duplex mode.	R	UNPRED
2	NC	No Carrier. 0 Carrier present. 1 The carrier signal from the transceiver was not present during transmission.  This bit is valid only when operating in half-duplex mode.	R	UNPRED
1	JT	Jabber Timeout. 0 No jabber timeout. 1 The MAC transmitter has been active for an abnormally long time (twice the Ethernet maxFrameLength size).	R	UNPRED
0	FA	Frame Aborted. 0 Current frame was successfully transmitted. 1 The transmission of the current frame has been aborted by the MAC because of one or more of the following conditions:  Jabber Timeout (bit 1). No Carrier (bit 2). Loss of Carrier (bit 3). Excessive Deferral (bit 4). Late Collision (bit 5). Retry Count exceeds the attempt limit (bit 6). Data under run (bit 7).	R	UNPRED

#### 6.5.4.4 Transmit Buffer Address/Enable Register

This register contains the starting address for the transmit memory buffer. The MAC DMA transfers the number of bytes designated in the Length register.

The buffer for the DMA must be cache line aligned so the lowest 5 bits are not used as part of the address. These bits have been employed as done and enable bits and are exclusive of the address.

`macdma0_txnaddr`

offset = 0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	ADDR[31:5]																											CB	DN	EN			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0

Bits	Name	Description	R/W	Default
31:5	ADDR	Buffer Address. Upper 27 bits of the starting physical address for the DMA buffer—but not including the most significant nibble address bits 35:32. This address must be cache line (32 bytes) aligned so only 27 bits are used.  Note: This address must be written for each DMA transaction (the address will not remain after the transaction is enabled).	R/W	0
4	—	Reserved, should be cleared.	R	0
3:2	CB	Current Buffer. Current DMA transmit buffer.	R	UNPRED
1	DN	Transaction Done. This bit will be set by hardware to indicate that the transmit transaction has been completed and that the transmit packet status is valid.  If the respective MAC DMA interrupt is enabled (see Section 5.0 "Interrupt Controller" on page 83), an interrupt will be generated when this bit is set. Done bits for all TX and RX buffers are or'ed together for this interrupt so a high level interrupt should be used.  This bit must be cleared explicitly by software after checking for done. This will also clear the interrupt.	R/W	UNPRED
0	EN	MAC DMA Enable. When set, this bit enables a DMA transmit transaction from the memory location designated in ADDR.	R/W	

#### 6.5.4.5 Transmit Buffer Length Register

This register contains the length of the memory buffer in bytes to be transmitted.

`macdma0_txnlen`

offset = 0x8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESERVED														LEN[13:0]																	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:14	—	Reserved, should be cleared.	R	0
13:0	LEN	Buffer Length. This field sets the length of the memory buffer (in bytes).  When the normal bit is set the length can only be up to 0x800 bytes.  When the Jumbo packets are enabled in the enable register the length can be set up to 0x2800 bytes.	R/W	0

### 6.5.5 Hardware Connections

Table 6-27 shows the signals associated with the two Ethernet MAC MII interfaces.

**Table 6-27. Ethernet Signals**

Signal	Input/Output	Description
<b>Ethernet Controller 0 (MAC0)</b>		
N0TXCLK	I	Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100 Mbps and 2.5 MHz when operating at 10 Mbps.
N0TXEN	O	Indicates that the data nibble on N0TXD[3:0] is valid. Muxed with GPIO[24]. GPIO[24] is the default signal coming out of hardware reset, runtime reset, and Sleep.
N0TXD[3:0]	O	Nibble wide data bus synchronous to N0TXCLK. For each N0TXCLK period in which N0TXEN is asserted, TXD[3:0] will have the data to be accepted by the PHY. While N0TXEN is de-asserted the data presented on TXD[3:0] should be ignored. Muxed with GPIO[28:25]. GPIO[28:25] are the default signals coming out of hardware reset, runtime reset, and Sleep.
N0RXCLK	I	Continuous clock that provides the timing reference for the data transfer from the PHY to the MAC. N0RXCLK is sourced by the PHY. The N0RXCLK shall have a frequency equal to 25% of the data rate of the received signal data stream (typically 25 MHz at 100 Mb/s and 2.5 MHz at 10 Mb/s).
N0RXDV	I	Active high. Indicates that a receive frame is in process and that the data on N0RXD[3:0] is valid.
N0RXD[3:0]	I	RXD[3:0] is a nibble wide data bus driven by the PHY to the MAC synchronous with N0RXCLK. For each N0RXCLK period in which N0RXDV is asserted, RXD[3:0] will transfer four bits of recovered data from the PHY to the MAC. While N0RXDV is de-asserted, RXD[3:0] will have no effect on the MAC.
N0CRS	I	N0CRS shall be asserted by the PHY when either transmit or receive medium is non-idle. N0CRS shall be deasserted by the PHY when both the transmit and receive medium are idle. N0CRS is an asynchronous input.
N0COL	I	N0COL shall be asserted by the PHY upon detection of a collision on the medium, and shall remain asserted while the collision condition persists. N0COL is an asynchronous input. The N0COL signal is ignored by the MAC when operating in the full duplex mode.
N0MDC	O	N0MDC is sourced by the MAC to the PHY as the timing reference for transfer of information on the N0MDIO signal. N0MDC is an aperiodic signal that has no maximum high or low times. The N0MDC frequency is fixed at SBUS clock divided by 160. Muxed with GPIO[215]. GPIO[215] is the default signal coming out of hardware reset, runtime reset, and Sleep.
N0MDIO	IO	N0MDIO is the bidirectional data signal between the MAC and the PHY that is clocked by N0MDC.

MAC0 shares its pins with GPIO[28:24] and GPIO[215]; these pins must be assigned to MAC0 in order to use MAC0. Please see Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209 for more information.



## 6.5.6 Programming Considerations

The Ethernet MAC is designed such that the application could use a pool of memory buffers for both the transmit and receive functions.

The lowest level device driver would respond to the MAC DMA interrupt and swap out the filled DMA buffers for those that are empty for the receive case. For the transmit case the driver should provide ready to transmit buffers to the DMA while reclaiming empty buffers. Four transmit and receive DMA buffers are allocated for each MAC to allow for latency to service the lowest level MAC DMA interrupt.

At the next level in software the device driver can parse the valid data out of the frame for receive, or build the frame for transmit. The number of memory buffers needed in the pool will depend on how fast the parsing can occur for worst case receive bursts, and any minimum transmit latency requirements.

From this level the application or protocol stack can take the data and apply it as needed.

### 6.5.6.1 Initialization

This section demonstrates the functional requirements for getting the MAC running. This assumes that the programmer has already performed the Au1100 bringup.

- 1) Interrupt Controller - a high level interrupt should be used as the interrupt is triggered with an OR'ing of the DN (Done) bits.
- 2) DMA Controller Setup
- 3) MAC Registers - It is the system designer's responsibility to set up addresses.
- 4) Memory - Depending on how the system is built, there could be a pool of memory buffers which can be used for parsing and building of frames. Individual buffers would be swapped in and out of the 4 active receive and transmit DMA buffers as needed. This strategy would require some sort of minimal memory management within the Ethernet driver to insure chronology of Ethernet frames.

The following is a transmit example in a basic form. Typically this would be split between an interrupt handler and another higher layer.

- 1) Construct Frame
- 2) Set length in **macdma0\_txrlen** register
- 3) Set address of memory buffer and enable transmit. During this time the physical memory buffer and address and length registers should not be disrupted or transmit contents will be undefined.
- 4) Wait for done. This can be done by waiting for the interrupt handler or polling the done signal in the **macdma0\_txraddr** register.
- 5) Read status. Its validity is signaled by the reception of the done signal.

The following is a basic receive example:

- 1) Enable all receive buffers with four different memory buffer addresses.
- 2) Wait for interrupt. Conversely the done bit could be polled. During this time the physical memory buffer and address registers should not be disrupted or receive contents will be undefined.
- 3) Replace all full buffers with empty memory buffers.
- 4) Read Status for full buffers.
- 5) Parse frames.

## 6.6 I<sup>2</sup>S Controller

The Au1100 contains an I<sup>2</sup>S controller capable of interfacing with a codec or a discreet DAC and ADC. The I<sup>2</sup>S interface works in two different modes: unidirectional data mode and bidirectional data mode.

In unidirectional data mode the I2SDI signal is not used. In this mode the I2SDIO signal can be configured as an input or an output and can be used with either an ADC or a DAC at any one time.

In bidirectional mode the I2SDIO signal is configured as an output and used in conjunction with I2SDI to interface the port to a codec or discreet ADC and DAC.

The port will only support one input at any one time. In other words, I2SDIO can not be enabled as an input at the same time I2SDI is being used.

### 6.6.1 I<sup>2</sup>S Register Descriptions

The I<sup>2</sup>S interface is controlled by a register block whose physical base address is shown in Table 6-28. The I<sup>2</sup>S register block consists of three registers as shown in Table 6-29.

**Table 6-28. I<sup>2</sup>S Base Address**

Name	Physical Base Address	KSEG1 Base Address
i2s_base	0x0 1100 0000	0xB100 0000

**Table 6-29. I<sup>2</sup>S Interface Register Block**

Offset (Note 1)	Register Name	Description
0x0000	i2s_data	Input and output from data FIFOs
0x0004	i2s_config	Configuration and status register
0x0008	i2s_enable	Allows port to be enabled and disabled

Note 1. See Table 6-28 for base address.

#### 6.6.1.1 I<sup>2</sup>S Data

The I<sup>2</sup>S Data register is the input to the transmit FIFO when written to and the output from the receive FIFO when read from. Each FIFO is 12 words deep.

Care should be taken to monitor the status register to insure that there is room for data for a write or data in the FIFO for a read transaction.

The FIFO is for both the left and the right channels. For this reason data should be read from and written to the FIFO in pairs. The programmer should insure that data is written to the FIFO corresponding to how the Justification, Initial Channel, and Size is configured. If the sample size being written or read is different than the size being configured, the programmer should justify the data accordingly.

#### i2s\_data - TX/RX Data

Offset = 0x0000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	—	Reserved, should be cleared.	R	0
23:0	DATA	Data word up to 24 bits. When written, this field is the transmit data. When read, this field is the receive data.	R/W	

### 6.6.1.2 Configuration and Status Register

The I<sup>2</sup>S Interface Configuration and Status register contains status bits for the transmit and receive FIFOs, and configuration bits for the interface.

#### i2s\_config - Configuration and Status

Offset = 0x0004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
							XU	XO	RU	RO	TR	TE	TF	RR	RE	RF							ICK	PD	LB	IC	FM	TN	RN	SZ				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:26	—	Reserved, should be cleared.	R	0
25	XU	Transmit FIFO underflow status. 0 No underflow. 1 Underflow error condition.  This sticky bit is cleared by writing a '0' to it. Because this register is also used for configuration, mask this bit with a '1' to preserve its value if needed.	R/W	0
24	XO	Transmit FIFO overflow status. 0 No overflow. 1 Overflow error condition.  This sticky bit is cleared by writing a '0' to it. Because this register is also used for configuration, mask this bit with a '1' to preserve its value if needed.	R/W	0
23	RU	Receive FIFO underflow status. 0 No underflow. 1 Underflow error condition.  This sticky bit is cleared by writing a '0' to it. Because this register is also used for configuration, mask this bit with a '1' to preserve its value if needed.	R/W	0
22	RO	Receive FIFO overflow status. 0 No overflow. 1 Overflow error condition.  This sticky bit is cleared by writing a '0' to it. Because this register is also used for configuration, mask this bit with a '1' to preserve its value if needed.	R/W	0
21	TR	Transmit Request. This bit indicates that the transmit FIFO has at least 4 samples of space to accommodate a burst write.	R	0
20	TE	Transmit Empty. This bit indicates that the transmit FIFO is empty.	R	0
19	TF	Transmit Full. This bit indicates the transmit FIFO is full.	R	0
18	RR	Receive Request. This bit indicates that the receive FIFO has at least 4 samples in it to accommodate a burst read.	R	0
17	RE	Receive Empty. This bit indicates that the receive FIFO is empty.	R	0
16	RF	Receive Full. This bit indicates that the receive FIFO is full.	R	0
15:13	—	Reserved, should be cleared.	R	0
12	ICK	Invert Clock. 0 Data is valid on falling edge. 1 Data is valid on rising edge.	R/W	0

Bits	Name	Description	R/W	Default
11	PD	<p>Pin Direction. Configures the direction of the I2SDIO signal.</p> <p>0 I2SDIO is an output—for unidirectional or bidirectional operation. For the unidirectional transmit case, do not use I2SDI. Note also in this case that the GPIO[8] function (which is muxed with I2SDI) can be used only as an output unless the I<sup>2</sup>S receive function is disabled (RN=0).</p> <p>1 I2SDIO is an input—for unidirectional operation only. Do not use I2SDI. Note also in this unidirectional receive case that the GPIO[8] function (which is muxed with I2SDI) can be used only as an <i>output</i>.</p>	R/W	0
10	LB	<p>Loopback. When set this bit will enable a loop back mode where data coming on the input will be presented on the output.</p>	R/W	0
9	IC	<p>Initial Channel.</p> <p>0 The left sub channel is the first presented. This means that data will not be presented until the word clock is the correct polarity for left as described in Format.</p> <p>1 The right sub channel is the first presented. This means that data will not be presented until the word clock is the correct polarity for right (as described in the Format section of this table).</p>	R/W	0
8:7	FM	<p>Format. The following formats are supported:</p> <p>00 I<sup>2</sup>S mode. In this mode the first bit of a sample word will be presented after one I2SCLK delay from the transition of I2SWORD. The left sample data will be presented when the word clock is low. The data is presented MSB first.</p> <p>01 Left Justified mode. In this mode the first bit of a sample word will be presented on the first I2SCLK after an I2SWORD transition. The left sample data will be presented when the word clock is high. The data is presented MSB first.</p> <p>10 Right Justified mode. In this mode the first bit of a sample word will be presented on the first I2SCLK after an I2SWORD transition. The left sample data will be presented when the word clock is high. The data is presented LSB first.</p> <p>11 Reserved.</p>	R/W	00
6	TN	<p>Transmit Enable. This will enable the transmit FIFO and must be enabled if the output is being used.</p> <p>0 Disable transmit FIFO.</p> <p>1 Enable transmit FIFO.</p>	R/W	0
5	RN	<p>Receive Enable. This will enable the receive FIFO and must be enabled if either of the inputs are being used.</p> <p>0 Disable receive FIFO.</p> <p>1 Enable receive FIFO.</p>	R/W	1
4:0	SZ	<p>Size. These bits will set the size of the sample word. The following combinations are valid:</p> <p>01000 8-bit words  10000 16-bit words  10010 18-bit words  10100 20-bit words  11000 24-bit words</p> <p>If using DMA it is important that memory is packed consistently with the transfer width programmed for the DMA channel and the Size field.</p>	R/W	10010

### 6.6.1.3 I<sup>2</sup>S Enable

The I<sup>2</sup>S Block Control register is used to enable clocks to and reset the entire I<sup>2</sup>S block.

The suggested power on reset is as follows:

- 1) Set both CE and D.
- Clear D for to enable the peripheral.

#### i2s\_enable - I<sup>2</sup>S Block Control

Offset = 0x0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bits	Name	Description	R/W	Default
31:2	—	Reserved, should be cleared.	W	0
1	D	Disable. Setting this bit will disable the I <sup>2</sup> S block. After enabling the clock with CE, this bit should be cleared for normal operation.	W	1
0	CE	Clock Enable. This bit should be set to enable the clock driving the I <sup>2</sup> S block. It can be cleared to disable the clock for power considerations.	W	0

### 6.6.2 Hardware Considerations

Table 6-30 shows the signals associated with this port.

Table 6-30. I<sup>2</sup>S Signals

Signal	Input/Output	Description
I2SCLK	O	Serial bit clock. Muxed with GPIO[30]. GPIO[30] is the default signal coming out of hardware reset, runtime reset, and Sleep.
I2SWORD	O	Word clock typically configured to the sampling frequency (Fs). Muxed with GPIO[31]. GPIO[31] is the default signal coming out of hardware reset, runtime reset, and Sleep.
I2SDI	I	Serial data input sampled on the rising edge of I2SCLK. Note that I2SDI is used as the input for <i>bidirectional</i> operation only, in which case it is used in conjunction with I2SDIO as the output ( <i>i2s_config</i> [PD]=0). Muxed with GPIO[8]. GPIO[8] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the I <sup>2</sup> S interface for <i>unidirectional</i> operation (I2SDI not used), the GPIO[8] function is available but with the following restrictions: <ul style="list-style-type: none"> <li>• When I2SDIO is configured as an <i>input</i>, GPIO[8] can be used only as an output.</li> <li>• When I2SDIO is configured as an <i>output</i>, the I<sup>2</sup>S receive function must be disabled if GPIO[8] is to be used as an input.</li> </ul>
I2SDIO	IO	Configurable as input or output. As input, data should be presented on rising edge. As output, data is valid on the rising edge. Muxed with GPIO[29]. GPIO[29] is the default signal coming out of hardware reset, runtime reset, and Sleep.
EXTCLK <sub>n</sub>	O	This is the system audio clock and typically is programmed to 256 * Fs ( <i>where</i> Fs is the sampling frequency for the system). The system audio clock should be taken from EXTCLK0 or EXTCLK1 because these signals are synchronous to I2SCLK and I2SWORD. These clocks are programmed individually; see Section 7.1 "Clocks" on page 194.

For changing pin functionality, see the **sys\_pinfunc** register description in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

### 6.6.3 Programming Considerations

It is the programmer's responsibility to set up DMA channels, and the clocks (I2SCLK and the EXTCLK $n$ ) to be used with the system.

The I2SWORD clock, which is typically equal to the sampling frequency, is a function of the word width and the I2SCLK frequency. I2SCLK and EXTCLK $n$  are programmable as described in Section 7.1 "Clocks" on page 194. The EXTCLK $n$  signals are the external clocks available on the pins shared with GPIO[2] and GPIO[3].

## 6.7 UART Interfaces

The Au1100 contains three UART interfaces. Each UART has the following features:

- 5 - 8 Data Bits
- 1 - 2 Stop Bits
- Even, Odd, Mark, or No Parity
- 16 Byte Transmit and Receive FIFOs
- Interrupts for Receive FIFO Full, Half Full, and Not Empty
- Interrupts for Transmit FIFO Empty
- False Start Bit Detection
- Full Modem Control Signals on UART3
- Capable of speeds up to 1.5 Mbps to enable connections with Bluetooth and other peripherals through a UART interface
- Similar to personal computer industry standard 16550 UART

### 6.7.1 Programming Model

Each UART is controlled by a register block. Table 6-31 lists the base address for each UART register block. UART0 and UART3 are capable of being used with DMA. See Figure 4 "DMA Controller" on page 75 for more information.

**Table 6-31. UART Register Base Addresses**

Name	Physical Base Address	KSEG1 Base Address
uart0_base	0x0 1110 0000	0xB110 0000
uart1_base	0x0 1120 0000	0xB120 0000
uart3_base	0x0 1140 0000	0xB140 0000

### 6.7.2 UART Registers

Each register block contains the registers listed in Table 6-32.

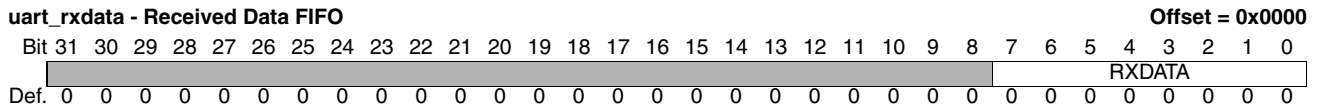
**Table 6-32. UART Registers**

Offset (Note 1)	Register Name	Description
0x0000	uart_rxddata	Received Data FIFO
0x0004	uart_txddata	Transmit Data FIFO
0x0008	uart_inten	Interrupt Enable Register
0x000C	uart_intcause	Pending Interrupt Cause Register
0x0010	uart_fifoctrl	FIFO Control Register
0x0014	uart_linectrl	Line Control Register
0x0018	uart_mdmctrl	Modem Line Control Register (UART3 only)
0x001C	uart_linestat	Line Status Register
0x0020	uart_mdmstat	Modem Line Status Register (UART3 only)
0x0024	uart_autoflow	Automatic Hardware Flow Control (UART3 only)
0x0028	uart_clkdiv	Baud Rate Clock Divider
0x0100	uart_enable	Module Enable Register

Note 1. See Table 6-31 for base address.

**6.7.2.1 Received Data FIFO**

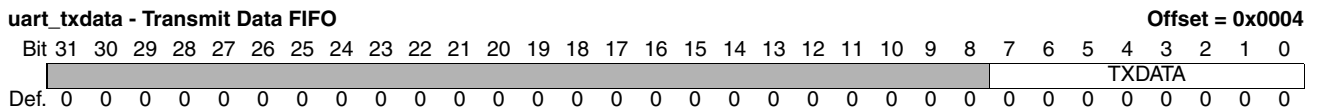
The `uart_rxddata` register contains the next entry in the received data FIFO. This register is read only.



Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7:0	RXDATA	Receive Data.	R	0

**6.7.2.2 Transmit Data FIFO**

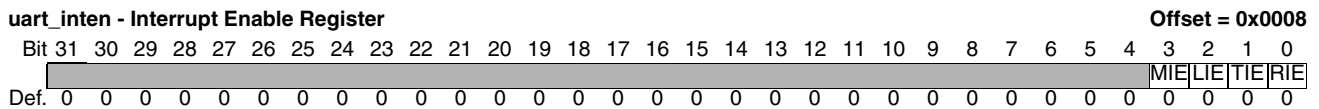
The `uart_txddata` register provides access to the transmit data FIFO. This register is write only.



Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7:0	TXDATA	Transmit Data	R	0

**6.7.2.3 Interrupt Enable Register**

The `uart_inten` register contains bits which enable interrupts under certain operational conditions.



Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3	MIE	Modem Status Interrupt Enable (UART3 only). When the MIE bit is set an interrupt is generated when changes occur in the state of the optional modem control signals available with UART3.  <b>System Note:</b> For systems that use the UART3 interface but do <i>not</i> use the optional modem control signals ( <code>sys_pinctrl[UR3]=0</code> ), the modem status interrupts must be disabled ( <code>MIE=0</code> ) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as a general-purpose system input.	R/W	0
2	LIE	Line Status Interrupt Enable. When the LIE bit is set an interrupt is generated when errors (overrun, framing, stop bits) or break conditions occur.	R/W	0
1	TIE	Transmit Interrupt Enable. When the TIE bit is set an interrupt is generated when the transmit FIFO is not full.	R/W	0
0	RIE	Receive Interrupt Enable. When the RIE bit is set the UART will generate an interrupt on received data ready ( <code>DR</code> bit in the <code>uart_linestat</code> register) or a character time out.	R/W	0

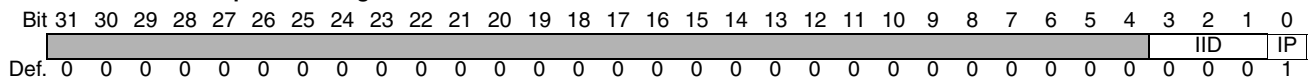


### 6.7.2.4 Interrupt Cause Register

The `uart_intcause` register contains information about the cause of the current interrupt.

#### uart\_intcause - Interrupt Cause Register

Offset = 0x000c



Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3:1	IID	Interrupt Identifier. The IID field identifies the highest priority current interrupt condition. Table 6-33 lists the priorities and encodings of each interrupt condition.	R	0
0	IP	No interrupt pending. 0 An interrupt is pending. 1 No interrupts are pending.	R	1

Table 6-33 contains information about the interrupt cause encoding.

**Table 6-33. Interrupt Cause Encoding**

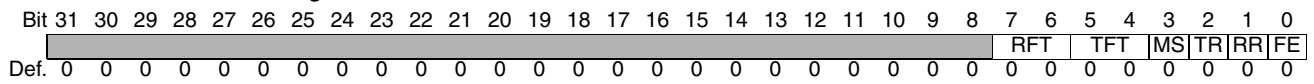
IID	Priority	Type	Source
0	5 (lowest)	Modem Status	DD, TRI, DR or DC of <code>uart_mdmstat</code>
1	4	Transmit Buffer Available	TT of <code>uart_linestat</code>
2	3	Receive Data Available	The receive FIFO having greater than RFT (of <code>uart_fifoctrl</code> ) bytes in it if FIFOs are enabled. DR of <code>uart_linestat</code> if FIFOs are disabled.
3	1 (highest)	Receive Line Status	OE, PE, FE, BI in <code>uart_linestat</code> register
4		Reserved	
5		Reserved	
6	2	Character Time Out	Character has been in receive FIFO for 0x300 UART clocks (set by <code>uart_clkdiv</code> )
7		Reserved	

**6.7.2.5 FIFO Control Register**

The `uart_fifoctrl` register provides control of character buffering options.

**uart\_fifoctrl - FIFO Control Register**

Offset = 0x0010



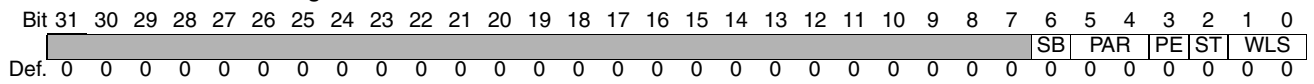
Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7:6	RFT	Receive FIFO Threshold. A receive threshold interrupt is generated when the number of characters in the receiver FIFO is greater than or equal to the trigger level listed below: 00 Trigger depth = 1 01 Trigger depth = 4 10 Trigger depth = 8 11 Trigger depth = 14 If using DMA it is important that the receive FIFO threshold and transmit FIFO threshold are the same and programmed consistently with the transfer size for the DMA channel being used. See Section 4.0 "DMA Controller" on page 75 for more information.	R/W	0
5:4	TFT	Transmit FIFO Threshold. A transmit threshold interrupt is generated if the number of valid characters contained in the transmit FIFO is less than or equal to the trigger depth. The encoding of trigger depth for each value of TFT is shown below: 00 Trigger depth = 0 01 Trigger depth = 4 10 Trigger depth = 8 11 Trigger depth = 12 If using DMA it is important that the receive FIFO threshold and transmit FIFO threshold are the same and programmed consistently with the transfer size for the DMA channel being used. See Section 4.0 "DMA Controller" on page 75 for more information.	R/W	0
3	MS	Mode Select. If the MS bit is clear interrupts are generated by the receiver when any data is available and by the transmitter when there is no data to transmit. Setting the MS bit causes interrupts to be generated based on FIFO threshold levels.	R/W	0
2	TR	Transmitter Reset. Writing a one to the TR bit will clear the transmit FIFO and reset the transmitter. The transmit shift register is not cleared.	R/W	0
1	RR	Receiver Reset. Writing a one to the RR bit will clear the receiver FIFO and reset the receiver. The receiver shift register is not cleared.	R/W	0
0	FE	FIFO Enable. The FE bit enables the 16 byte FIFOs on transmit and receive. When the FE bit is clear both FIFOs will have an effective depth of 1 byte.	R/W	0

### 6.7.2.6 Line Control Register

The `uart_linectrl` register provides control over the data format and parity options.

#### uart\_linectrl - Line Control Register

Offset = 0x0014



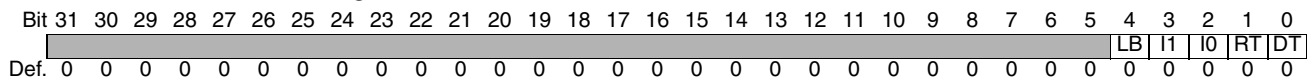
Bits	Name	Description	R/W	Default
31:7	—	Reserved, should be cleared.	R	0
6	SB	Send Break. Setting the SB bit will force the transmitter output to zero.	R/W	0
5:4	PAR	Parity Select. Selects the parity encoding for the transmitter and receiver. 00 Odd parity 01 Even parity 10 Mark parity 11 Zero parity	R/W	0
3	PE	Parity Enable. If the PE bit is clear parity will not be sent or expected. If the PE bit is set parity is selected according to the PAR field.	R/W	0
2	ST	Stop Bits. If the ST bit is clear one stop bit is sent and expected. Setting the ST bit selects 1.5 stop bits for 5 bit characters and 2 stop bits for all other character lengths.	R/W	0
1:0	WLS	Word Length Select. The WLS field selects the number of data bits in each character. The number of bits is WLS+5.	R/W	0

### 6.7.2.7 Modem Control Register

The `uart_mdmcctl` register allows the state of the output modem control signals to be set. The external modem signals are only available on UART3.

#### uart\_mdmcctl - Modem Control Register

Offset = 0x0018



Bits	Name	Description	R/W	Default												
31:5	—	Reserved, should be cleared.	R	0												
4	LB	Loopback. 0 No loopback (normal operation). 1 Enable loopback for self-test. Establish the internal connections shown below:  <table border="0" style="margin-left: 20px;"> <tr> <td>Output Signal</td> <td>Looped Back To</td> </tr> <tr> <td>TXD</td> <td>RXD</td> </tr> <tr> <td>DTR#</td> <td>DSR#</td> </tr> <tr> <td>RTS#</td> <td>CTS#</td> </tr> <tr> <td>I0#</td> <td>RI#</td> </tr> <tr> <td>I1#</td> <td>DCD#</td> </tr> </table>	Output Signal	Looped Back To	TXD	RXD	DTR#	DSR#	RTS#	CTS#	I0#	RI#	I1#	DCD#	R/W	0
Output Signal	Looped Back To															
TXD	RXD															
DTR#	DSR#															
RTS#	CTS#															
I0#	RI#															
I1#	DCD#															
3	I1	Internal Line 1 State. When the I1 bit is set the internal I1# line for this port is driven low. This can be used in loopback mode.	R/W	0												
2	I0	Internal Line 0 State. When the I0 bit is set the external I0# line for this port is driven low. This can be used in loopback mode.	R/W	0												
1	RT	Request To Send. When the RT bit is set the external RTS# line for this port is driven low.  NOTE: This bit has no effect if <code>uart_autoflow[AE]</code> is set.	R/W	0												
0	DT	Data Terminal Ready. When the DT bit is set the external DTR# line for this port is driven low.	R/W	0												

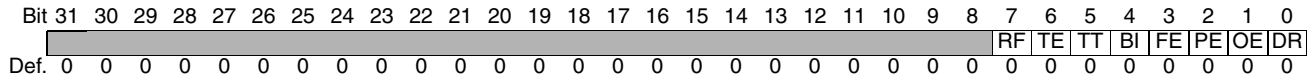
**6.7.2.8 Line Status Register**

The `uart_linestat` register reflects the state of the interface.

Bits in this register are set when the listed condition and cleared when this register is read.

**uart\_linestat - Line Status Register**

Offset = 0x001C



Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7	RF	Receiver FIFO Contains Error. This bit is set when one of the characters in the receive FIFO contains a parity error, framing error, or break indication.	R	0
6	TE	Transmit Shift Register Empty. This bit is set when the transmit shift register is empty and there are no more characters in the FIFO.	R	0
5	TT	Transmit Threshold. This bit is set when the transmitter FIFO depth is less than or equal to the value of the TFT field in the FIFO control register. When FIFOs are not enabled this bit is set when the transmitter data register is empty	R	0
4	BI	Break Indication. This bit is set if a break is received. When a break is detected a single zero character is received. The BI bit is valid when the zero character is at the top of the receive FIFO. This bit must be cleared with a read to <code>uart_linestat</code> before more characters are received.	R	0
3	FE	Framing Error. The FE bit is set when a valid stop bit is not detected. This bit reflects the state of the character at the top of the receive FIFO. The FE bit is cleared by a read to <code>uart_linestat</code> .	R	0
2	PE	Parity Error. The PE bit is set when the received character at the top of the FIFO contains a parity error. This bit is cleared by reading <code>uart_linestat</code> .	R	0
1	OE	Overrun Error. The OE bit is set when a receiver overrun occurs. This bit is cleared when <code>uart_linestat</code> is read.	R	0
0	DR	Data Ready. The DR bit is set when the receive FIFO contains valid characters.	R	0

### 6.7.2.9 Modem Status Register

The `uart_mdmstat` register reflects the state of the external modem signals. Reading this register will clear any delta indications and the corresponding interrupt. The external modem signals are optional and are present only on UART3.

#### uart3\_mdmstat - Modem Status Register

Offset = 0x0020

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									CD	RI	DS	CT	DD	TRI	DR	DC
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7	CD	Data Carrier Detect. The CD bit reflects the status of the external DCD# pin.	R	0
6	RI	Ring Indication. The RI bit reflects the status of the external RI# pin.	R	0
5	DS	Data Set Ready. The DS bit reflects the status of the external DSR# pin.	R	0
4	CT	Clear To Send. The CT bit reflects the status of the external CTS# pin.	R	0
3	DD	Delta DCD. The DD bit is set when a change occurs in the state of the external DCD# pin.	R	0
2	TRI	Terminate Ring Indication. The TRI bit is set when a positive edge occurs in the state of the external RI# pin.	R	0
1	DR	Delta DSR. The DR bit is set when a change occurs in the state of the external DSR# pin.	R	0
0	DC	Delta CTS. The DC bit is set when a change occurs in the state of the external CTS# pin.	R	0

### 6.7.2.10 Automatic Hardware Flow Control Register

The `uart Autoflow` register controls automatic hardware flow control using modem control signals CTS# and RTS#. Upon enabling this mode, internal logic controls the output signal RTS# based upon the data register state and threshold levels. The internal logic asserts RTS# (low) to request data until the internal receive FIFO reaches its preset threshold. In this mode RTS# cannot be controlled with the `uart_mdmctrl[RT]` bit. The input signal CTS# controls the transmission of data by loading the transmit shift register from the data register only while CTS# is asserted (low). Once the transmit shift register is loaded with data, it sends the entire character regardless of the CTS# signal state.

#### uart Autoflow - Automatic Hardware Flow Control Register

Offset = 0x0024

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
																																AE	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:1	—	Reserved, should be cleared.	R	0
0	AE	Autoflow Enable. Setting this bit enables automatic hardware flow control on UART3. Enabling this mode overrides software control of the signals.	R/W	0

**6.7.2.11 Clock Divider Register**

The `uart_clkdiv` contains the divider used to generate the baud rate clock. The input to the UART clock divider is the internal peripheral bus clock. The actual baud rate of the interface will be as follows:

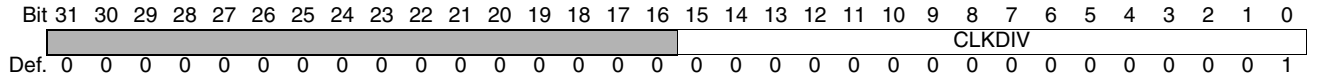
$$\text{Baud rate} = \text{CPU} / (\text{SD} * 2 * \text{CLKDIV} * 16)$$

CPU = CPU clock

SD = System bus (SBUS) divider (See Section 7.4 "Power Management" on page 214 information on changing SD.)

**uart\_clkdiv - Clock Divider Register**

Offset = 0x0028



**6.7.2.12 UART Enable**

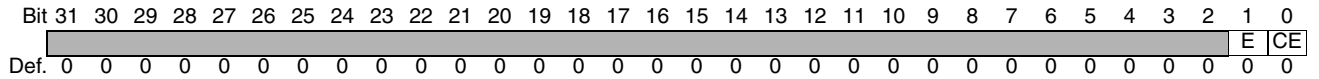
The `uart_enable` register controls reset and clock enable to the UART

The correct routine for bringing the USB Device out of reset is as follows:

- 1) Set the CE bit to enable clocks.
- 2) Set the E bit to enable the peripheral.

**uart\_enable - UART Enable Register**

Offset = 0x0100



Bits	Name	Description	R/W	Default
31:2	—	Reserved, should be cleared.	R	0
1	E	Enable. When the E bit is clear the entire module is held in reset. After enabling clocks, this bit should be set to enable normal operation.	R/W	0
0	CE	Clock Enable. When the CE bit is clear the module clock source is inhibited. This can be used to place the module in a low power Standby state. The CE bit should be set before the module is enabled for proper bringup.	R/W	0

### 6.7.3 Hardware Considerations

The UART signals are listed in Table 6-34. For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

**Table 6-34. UART Signals**

Signal	Input/Output	Definition
<b>UART0</b>		
U0TXD	O	UART0 Transmit. Muxed with GPIO[212]. GPIO[212] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U0RXD	I	UART0 Receive.
<b>UART1</b>		
U1TXD	O	UART1 Transmit. Muxed with GPIO[213]. GPIO[213] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U1RXD	I	UART1 Receive.
<b>UART3</b>		
U3TXD	O	UART3 Transmit. Muxed with GPIO[214]. GPIO[214] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U3RXD	I	UART3 Receive.
U3CTS#	I	Clear to Send (optional). Muxed with GPIO[9]. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.
U3DSR#	I	Data Set Ready (optional). Muxed with GPIO[10]. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.
U3DCD#	I	Data Carrier Detect (optional). Muxed with GPIO[11]. GPIO[11] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.
U3RI#	I	Ring Indication (optional). Muxed with GPIO[12]. GPIO[12] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.
U3RTS#	O	Request to Send (optional). Muxed with GPIO[13]. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U3DTR#	O	Data Terminal Ready (optional). Muxed with GPIO[14]. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.

## 6.8 SSI Interfaces

The Au1100 processor contains two synchronous serial interfaces (SSIs) designed to provide a simple connection to external serial devices. These serial channels support the SSI protocol and a subset of the SPI protocol.

Each serial channel is independently programmable for various address and data lengths, clock rates, and behavior.

Each channel has a data in pin, data out pin, a clock pin, and an enable pin. Only master mode is supported. The Au1100 processor drives the clock and enable pins when the interface is enabled.

The data out pin will tristate during a read, thus the data out pin and data in pin can be tied together for a bidirectional data pin.

### 6.8.1 Operation

The SSI generates the clock output SCLK. The clock is derived from the peripheral bus clock by a divider controlled by the **ssi\_clkdiv** register. The clock only transitions when a transaction is in progress.

The SSI contains a status register that reflects the current state. A busy bit is set when a transfer is initiated and cleared when SSI returns to Idle. A done bit is set when the transfer is complete. The done bit may be used to signal an interrupt.

#### 6.8.1.1 Write Transactions

Write transactions transfer data from the Au1100 processor to a peripheral device attached to the SSI. The transaction consists of a data field and optional address and direction fields. The order of the address and direction fields is configurable. The address and direction fields may also be omitted from the transaction. The data field is always the last field transmitted. The order of bit transmission within a field (MSb first or LSb first) is also configurable.

The SDEN output presents an envelope around the transaction. Figure 6-4 shows a typical write transaction. For this transaction the address field is 3 bits long, data is 8 bits, and direction precedes address.

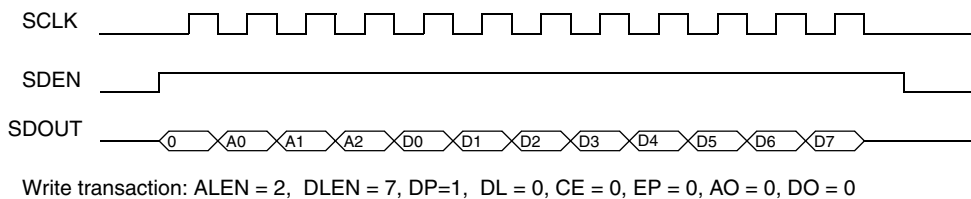


Figure 6-4. Typical Write Transaction Timing

#### 6.8.1.2 Read Transactions

A read transaction is initiated by writing the address and direction to the **ssi\_adata** register (the data field is ignored). The busy status bit will be set and will remain set until the done bit is set to indicate completion. Once the transaction is complete the data may be read from the data field in **ssi\_adata**.

An extra clock cycle is inserted between the direction/address transmission by the processor and the data field transmission by the external device to avoid contention. The behavior of SCLK may be changed during this extra cycle by programming the BM field in the **ssi\_config** register.

Figure 6-5 shows a typical read transaction where the bus mode is set to hold SCLK high during the bus turnaround.

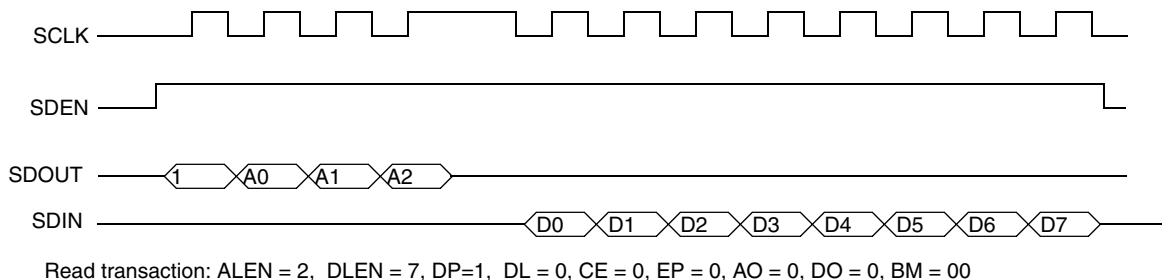


Figure 6-5. Typical Read Transaction Timing



### 6.8.2 Register Description

Each SSI contains a register block used to configure the interface and to pass data. All registers must be written and read as 32 bit words. The locations of the register blocks for each SSI are shown in the table below. Table 6-36 shows the offset and function of each register.

**Table 6-35. SSI Base Addresses**

Name	Physical Base Address	KSEG1 Base Address
ssi0_base	0x0 1160 0000	0xB160 0000
ssi1_base	0x0 1168 0000	0xB168 0000

**Table 6-36. SSI Registers**

Offset (Note 1)	Register Name	Description
0x0000	ssi_status	SSI Status Register
0x0004	ssi_int	SSI Interrupt Pending Register
0x0008	ssi_inten	SSI Interrupt Enable Register
0x0020	ssi_config	SSI Configuration Register
0x0024	ssi_adata	SSI Address/Data Register
0x0028	ssi_clkdiv	SSI Clock Divider Register
0x0100	ssi_enable	SSI Channel Enable Register

Note 1. See Table 6-35 for base address.

#### 6.8.2.1 SSI Interface Status Register

The **ssi\_status** register reflects the current status of the interface.

**ssi\_status - SSI Interface Status**

Offset = 0x0000

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											BF	OF	UF	D	B
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

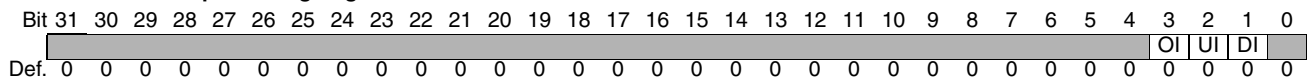
Bits	Name	Description	R/W	Default
31:5	—	Reserved, should be cleared.	R	0
4	BF	Buffer Full. This bit indicates that the data buffer is currently full. It is set by either receiving a buffer from the serial interface or a write by the processor. It is cleared by either a transmit on the serial interface or a read by the processor.	R	0
3	OF	Overflow. This bit is set when the serial data register is written multiple times without completing an intervening transfer. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R/W	0
2	UF	Underflow. This bit is set when the serial data register is read multiple times without an intervening serial transfer. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R/W	0
1	D	Done. This bit is set at the completion of an SSI transfer. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R/W	0
0	B	Busy. This bit is set if an SSI transfer is in progress	R	0

### 6.8.2.2 Interrupt Pending Register

The `ssi_int` register shows which interrupt indications are currently active.

#### ssi\_int - SSI Interrupt Pending Register

Offset = 0x0004



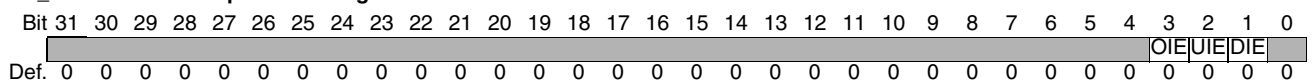
Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3	OI	The OI bit indicates that the current interrupt is being generated by an overflow condition. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R	0
2	UI	The UI bit indicates that the current interrupt is being generated by an underflow condition. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R	0
1	DI	The DI bit indicates that the current interrupt is being generated by a done condition. This bit is sticky. Once set high it must be written a '1' to clear the bit.	R	0
0	—	Reserved, should be cleared.	R	0

### 6.8.2.3 SSI Interrupt Enable Register

The `ssi_inten` register is writable by the processor and enables certain conditions on the SSI to generate an interrupt. The interrupt will be generated (and indicated in `ssi_int`) when the corresponding bits are both set in `ssi_inten` and `ssi_status`.

#### ssi\_inten - SSI Interrupt Enable Register

Offset = 0x0008



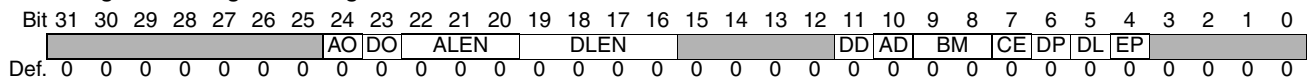
Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3	OIE	This bit enables interrupts on an overflow condition.	R/W	0
2	UIE	This bit enables interrupts on an underflow condition.	R/W	0
1	DIE	This bit enables interrupts on a done condition.	R/W	0
0	—	Reserved, should be cleared.	R	0

### 6.8.2.4 SSI Configuration Register

The `ssi_config` register contains fields which configure the operational parameters of the serial interface.

#### ssi\_config - SSI Configuration Register

Offset = 0x0020



Bits	Name	Description	R/W	Default
31:25	—	These bits are reserved and should be written as 0.	R	0
24	AO	Address Field Order. The AO bit selects the bit order of the address field. If AO is cleared the address field is set LSB first. If AO is set the address field is sent MSB first.	R/W	0
23	DO	Data Field Order. The DO field selects the transmission order for the data field. If DO is cleared the data field is sent LSB first. If DO is set the data field is sent MSB first.	R/W	0
22:20	ALEN	Address Field Length. The ALEN field selects the length of the address field in the serial stream. The number of bits in the address field will be ALEN+1.	R/W	0
19:16	DLEN	Data Field Length. The DLEN field selects the length of the data field in the serial stream. The number of bits in the data field will be DLEN+1. Values of DLEN that result in a length greater than 12 are reserved and will result in undefined behavior.	R/W	0
15:12	—	These bits are reserved and should be written as 0.	R/W	0
11	DD	Direction Bit Disable. If the DD bit is set the direction bit will not be sent.	R/W	0
10	AD	Address Field Disable. If the AD bit is set the address field will not be sent.	R/W	0
9:8	BM	Bus Mode. Determines the turnaround behavior for read cycles. 00 SCLK held high during turnaround. 01 SCLK held low during turnaround. 10 SCLK cycles during turnaround. 11 Reserved	R/W	0
7	CE	The CE bit determines which clock edge is active for SCLK. If CE is cleared data and address will be clocked out on the negative edge and captured at the positive edge. If CE is set data and address will be clocked out on the positive edge and captured on the negative edge.	R/W	0
6	DP	Direction Polarity. Determines whether a write is indicated by an active-high or active-low direction bit. 0 A write is indicated by an active-high direction bit. 1 A write is indicated by an active-low direction bit.	R/W	0
5	DL	Direction Bit Location. If the DL bit is clear the direction bit is sent before the address bits in the serial stream. If DL is set the direction bit will follow the address field.	R/W	0
4	EP	Enable Polarity. Selects the polarity of the enable signal on the interface. 0 Enable is active high. 1 Enable is active low.	R/W	0
3:0	—	Reserved, should be cleared.	R/W	0

### 6.8.2.5 SSI Address/Data Register

The **ssi\_adata** register contains the address, data, and direction fields. Writing to **ssi\_adata** will initiate a transfer. The type of transfer (read or write) is determined by the D (direction) bit.

#### ssi\_adata - SSI address/data Register

Offset = 0x0024

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	D								ADDR								DATA																	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:25	—	Reserved, should be cleared.	R	0
24	D	Direction Bit 0 The transaction is a read, and the data field contains the value of the serial input at the end of the transaction. 1 The transaction is a write.	R/W	0
23:16	ADDR	Address Field	R/W	0
15:12	—	Reserved, should be cleared.	R	0
11:0	DATA	Data Field	R/W	0

### 6.8.2.6 SSI Clock Divider Register

The **ssi\_clkdiv** register determines the baud rate of the serial port. The baud rate is defined as follows:

$$\text{Baudrate} = \text{CPU} / (\text{SD} * 4 * (\text{CLKDIV} + 1))$$

CPU = CPU clock

SD = System Bus (SBUS) divider (See Section 7.4 "Power Management" on page 214 for information on SD.)

#### ssi\_clkdiv - SSI clock divider

Offset = 0x0028

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	CLKDIV																																
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	CLKDIV	The CLKDIV field determines the baud rate of the interface.	R/W	0

### 6.8.2.7 SSI Enable Register

The **ssi\_enable** register allows the serial interface to be disabled or placed in a low power mode. The correct routine for bringing the SSI block out of reset is as follows:

- 1) Clear the CD bit to enable clocks.
- 2) Set the E bit to enable the peripheral.

#### ssi\_enable - SSI Enable Register

Offset = 0x0100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	CD	E														
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bits	Name	Description	R/W	Default
31:2	—	Reserved, should be cleared.	R	0
1	CD	Clock Disable. 0 Enable the clock to the SSI block. 1 Disable (disconnect) the clock to the SSI block.	W	1
0	E	Enable. 0 Hold the SSI block in reset. 1 Enable the SSI block.	W	0

### 6.8.3 Hardware Considerations

The SSI ports consist of the signals listed in Table 6-37. For changing pin functionality please refer to the **sys\_pinfunc** register in Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.

**Table 6-37. SSI Signals**

Signal	Input/Output	Definition
<b>SSI0</b>		
S0CLK	O	Master only clock output. The speed and polarity of clock edge is programmable. Muxed with GPIO[209] which controls the pin out of hardware reset, runtime reset and Sleep.
S0DIN	I	Serial Data Input. This signal may be tied with S0DOUT to create a single bidirectional data signal.
S0DOUT	O	Serial Data Output. This signal is tristated during a read and thus may be tied to S0DIN to create a single bidirectional data signal. Muxed with GPIO[208] which controls the pin out of hardware reset, runtime reset and Sleep.
S0DEN	O	Enable signal which frames transaction. The polarity is programmable. Muxed with GPIO[210] which controls the pin out of hardware reset, runtime reset and Sleep.
<b>SSI1</b>		
S1CLK	O	Master only clock output. The speed and polarity of clock edge is programmable. Muxed with ACDO which controls the pin out of hardware reset, runtime reset and Sleep.
S1DIN	I	Serial Data Input. This signal may be tied with S1DOUT to create a single bidirectional data signal. Muxed with ACBCLK which controls the pin out of hardware reset, runtime reset and Sleep.
S1DOUT	O	Serial Data Output. This signal is tristated during a read and thus may be tied to S1DIN to create a single bidirectional data signal. Muxed with ACSYNC which controls the pin out of hardware reset, runtime reset and Sleep.
S1DEN	O	Enable signal which frames transaction. The polarity is programmable. Muxed with ACRST# which controls the pin out of hardware reset, runtime reset and Sleep.

## 6.9 LCD Controller

The integrated LCD controller on the Au1100 processor contains the essential elements required to drive the latest industry standard 1-4 bit grayscale or 4-18 bit color LCD panels. The controller performs the basic memory based frame buffer to LCD panel data transfer through use of a dedicated DMA controller with double buffering support. It also supports hardware rotation (for up to 320x240 pixel displays) and spatio-temporal dithering (frame rate modulation) for STN type LCD panels.

The controller is capable of driving both active (TFT) and passive (STN) LCD panels through multiplexed signal pins. Color palette support is accomplished with an on-chip 256 entry 16-bit grayscale palette. TFT 16-bit mode allows the display of up to 65,536 simultaneous colors. A wide variety of LCD panels are supported through the use of user-programmable vertical and horizontal synchronization signals, bias signals and pixel clock rates.

The main features of the LCD controller include the following:

### Panel Support

- 4/8-bit mono single passive matrix STN panels
- 8-bit color single passive matrix STN panels
- 16-bit color dual passive matrix STN panels
- 12/16-bit TFT panels
- 18-bit TFT panels (up to 65,536 colors)
- Panel sizes up to 800x600 are supported

### Display Modes

- 1/2/4/8 bpp paletized TFT
- 12/16 bpp non-paletized TFT
- 1/2/4 bpp mono STN
- 1/2/4/8 bpp paletized color STN
- 12 bpp non-paletized color STN

### Miscellaneous Features

- Double buffering support
- Hardware Swivel (90, 180, and 270 degrees) for up to 320x240 pixel displays
- Two pulse width modulation (PWM) clocks to support digital control of contrast and brightness voltages (requires external filter circuits).

### 6.9.1 LCD Controller Registers

The LCD controller is controlled by a register block whose physical base address is shown in Table 6-38. The register block consists of the registers as shown in Table 6-39.

**Table 6-38. LCD Base Address**

Name	Physical Base Address	KSEG1 Base Address
lcd_base	0x0 1500 0000	0xB500 0000

**Table 6-39. LCD Controller Registers**

Offset (Note 1)	Register Name	Description
0x0000	lcd_control	Control Register
0x0004	lcd_intstatus	Interrupt Status Register
0x0008	lcd_intenable	Interrupt Enable Register
0x000C	lcd_horztiming	Horizontal Timing Register
0x0010	lcd_verttiming	Vertical Timing Register
0x0014	lcd_clkcontrol	Clock Control Register
0x0018	lcd_dmaaddr0	DMA Start Address 0
0x001C	lcd_dmaaddr1	DMA Start Address 1
0x0020	lcd_words	Frame Buffer Words
0x0024	lcd_pwmdiv	Pulse Width Modulation Frequency Divider
0x0028	lcd_pwmhi	Pulse Width Modulation High Time
0x0400	lcd_palettebase	Palette Interface Registers

Note 1. See Table 6-38 for base address.

#### 6.9.1.1 LCD Control Register

The LCD Control Register contains bits necessary to configure the LCD Controller. With the exception of the GO bit and the Enable White Data bit no fields of the Control Register should be written while the controller is active.

##### lcd\_control - LCD Control

Offset = 0x0000

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
									SBB	SBPPF		WP	WD	C	SM	DB	CCO	DP	PO	MPI	PT	PC			BPP	GO						
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:23	—	Reserved.	R	0
22:21	SBB	SBUS Bandwidth. Increases the bandwidth of the LCD controller by allowing multiple LCD transactions per SBUS access. 00 1 transaction per access 01 2 transactions per access 10 3 transactions per access 11 4 transactions per access	R/W	00

Bits	Name	Description	R/W	Default
20:18	SBPPF	Sixteen Bits Per Pixel Data Format. 000 6-bit Red, 5-bit Green, 5-bit Blue. 001 5-bit Red, 6-bit Green, 5-bit Blue. 010 5-bit Red, 5-bit Green, 6-bit Blue. 011 1-bit Intensity, 5-bit Red, 5-bit Green, 5-bit Blue. 100 5-bit Red, 5-bit Green, 5-bit Blue, 1-bit Intensity. All other values are reserved.	R/W	000
17	WP	White Data Polarity. This is the value which LCD_D[15:0] pins are set to when WD bit is set high.	R/W	0
16	WD	Enable White Data. When this bit is high LCD_D[15:0] pins are given the value programmed in the White Data Polarity (WP) bit. This bit is used during the startup and shutdown sequence of some LCD panels. This bit may be written at any time.	R/W	0
15	C	Coherent. 0LCD transactions are marked as non-coherent on the SBUS 1 LCD transactions are marked as coherent on the SBUS.	R/W	0
14:13	SM	Swivel Mode. 00Normal portrait 01 90 degree rotate (only supported for panels up to 320x240 pixels) 10 180 degree rotate 11 270 degree rotate (only supported for panels up to 320x240 pixels)	R/W	00
12	DB	TFT Data bits. This bit is used in paletized TFT modes to indicate how many LCD_DATA pins are to be used. 0 16 data pins 1 12 data pins	R/W	0
11	CCO	Color Channel Orientation. 0 RGB Channel Format 1 BGR Channel Format	R/W	0
10	DP	STN Panel Type. 0 Single Panel 1 Dual Panel	R/W	0
9:8	PO	Pixel Order. These bits show the order that pixels are packed into words in the frame buffer. See Table 6-40 on page 169.	R/W	0
7	MPI	Monochrome Panel Interface. 0 4 bit monochrome panel. 1 8 bit monochrome panel.	R/W	0
6	PT	Panel Type. 0 STN Passive: frame rate modulation algorithm used. 1 TFT Active.	R/W	0
5	PC	Panel Color. 0 Monochrome. 1 Color.	R/W	0
4	—	Reserved	R	0
3:1	BPP	Bits Per Pixel. 000 1 bit per pixel. 001 2 bits per pixel. 010 4 bits per pixel. 011 8 bits per pixel. 100 12 bits per pixel. 101 16 bits per pixel—not supported for DSTN panels. 11x Reserved.	R/W	000



Bits	Name	Description	R/W	Default
0	GO	LCD Go. When this bit is written high the LCD Controller's DMA engine starts fetching data for the frame. When data has been received it will begin sending this data along with the proper timing signals to the LCD panel. When this bit is written low the LCD controller will complete scanning out the current frame before shutting down. After completion of the last frame the SD bit in the interrupt status register will go high signaling that the LCD controller is now shutdown and can be reconfigured.	R/W	0

**Table 6-40. Pixel Ordering**

**PO = 00**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

**bpp**

1	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p15		p14		p13		p12		p11		p10		p9		p8		p7		p6		p5		p4		p3		p2		p1		p0	
4	p7				p6				p5				p4				p3				p2				p1				p0			
8	p3								p2								p1								p0							
12	p1								p0																							
16	p1																p0															

**PO = 01**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

**bpp**

1	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
2	p0		p1		p2		p3		p4		p5		p6		p7		p8		p9		p10		p11		p12		p13		p14		p15	
4	p0				p1				p2				p3				p4				p5				p6				p7			
8	p0								p1								p2								p3							
12	p0								p1																							
16	p0																p1															

**PO = 10**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

**bpp**

1	p24	p25	p26	p27	p28	p29	p30	p31	p16	p17	p18	p19	p20	p21	p22	p23	p8	p9	p10	p11	p12	p13	p14	p15	p0	p1	p2	p3	p4	p5	p6	p7
2	p12		p13		p14		p15		p8		p9		p10		p11		p4		p5		p6		p7		p0		p1		p2		p3	
4	p6				p7				p4				p5				p2				p3				p0				p1			
8	p3								p2								p1								p0							
12	p1								p0																							
16	p1																p0															

**PO = 11**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

**bpp**

1	p7	p6	p5	p4	p3	p2	p1	p0	p15	p14	p13	p12	p11	p10	p9	p8	p23	p22	p21	p20	p19	p18	p17	p16	p31	p30	p29	p28	p27	p26	p25	p24
2	p3		p2		p1		p0		p7		p6		p5		p4		p11		p10		p9		p8		p15		p14		p13		p12	
4	p1				p0				p3				p2				p5				p4				p7				p6			
8	p0								p1								p3								p2							
12	p0								p1																							
16	p0																p1															

**6.9.1.2 Interrupt Registers**

The interrupt status (**lcd\_status**) and interrupt enable (**lcd\_enable**) registers have identical formats. If a bit is set in **lcd\_enable** and the corresponding condition becomes true, an interrupt is issued with the corresponding bit in **lcd\_status** set. The interrupt for the LCD controller should be programmed as a high level type; see Section 5.1 "Interrupt Controller Sources" on page 83.

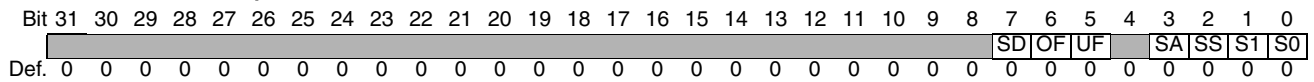
All interrupts except for shutdown must be cleared by writing a 1 to the corresponding bit in **lcd\_status**. These registers may be read and written while the LCD controller is active.

**lcd\_status - LCD Interrupt Status**

Offset = 0x0004

**lcd\_enable - LCD Interrupt Enable**

Offset = 0x0008



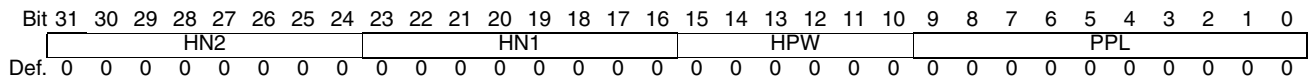
Bits	Name	Description	R/W	Default
31:8	—	Reserved.	R	0
7	SD	Shutdown. This condition occurs when the controller's GO bit is written low and the controller has finished displaying the last frame. After this bit goes high all registers of the controller can be written.	R/W	0
6	OF	Output FIFO Overflow.	R/W	0
5	UF	Output FIFO Underflow. This can occur when there is too much traffic on the SBUS, causing the LCD Controller to be unable to fetch data fast enough to refresh the LCD panel.	R/W	0
4	—	Reserved.	R	0
3	SA	Start Of Active Video. Occurs at the end of the vertical retrace	R/W	0
2	SS	Start Vertical Sync Period.	R/W	0
1	S1	Start Address 1 Latched. This interrupt is used for "double buffering". When this interrupt occurs it means that the LCD controller has latched DMA Start Address 1 and software is now free to change it. In this way software can be writing to one frame buffer while the controller is reading from the other. Start Address 1 is only used with Dual Panel STN displays.	R/W	0
0	S0	Start Address 0 Latched. This interrupt is used for "double buffering". When this interrupt occurs it means that the LCD controller has latched DMA Start Address 0 and software is now free to change it. In this way software can be writing to one frame buffer while the controller is reading from the other.	R/W	0

**6.9.1.3 Horizontal Timing Register**

See Figure 6-6 on page 176 and Figure 6-7 on page 177 for a graphical description of the LCD timing parameters.

**lcd\_horztiming - LCD Horizontal Timing**

Offset = 0x000C



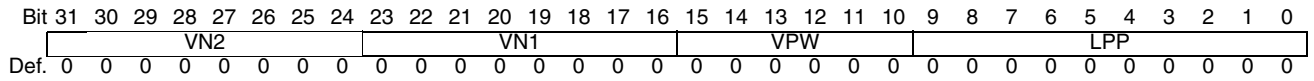
Bits	Name	Description	R/W	Default
31:24	HN2	Horizontal Non Display Period 2 (in pixels). Value programmed is one pixel less than actual value.	R/W	0
23:16	HN1	Horizontal Non Display Period 1 (in pixels). Value programmed is one pixel less than actual value.	R/W	0
15:10	HPW	Horizontal Sync Pulse Width (in pixels). Value programmed is one pixel less than actual value.	R/W	0
9:0	PPL	Pixels Per Line (in pixels). Value programmed is one pixel less than actual value.	R/W	0

### 6.9.1.4 Vertical Timing Register

See Figure 6-6 on page 176 and Figure 6-7 on page 177 for a graphical description of the LCD timing parameters. The “vertical retrace” time (STN: VN1, TFT: VN1+VN2+VPW) must be large enough for the LCD Controller’s DMA engine to fetch the start of the next frame. The number of lines which is required is system dependent. The number of lines required is typically larger in 90 & 270 degree swivel modes.

#### lcd\_vertiming- LCD Vertical Timing

Offset = 0x0010



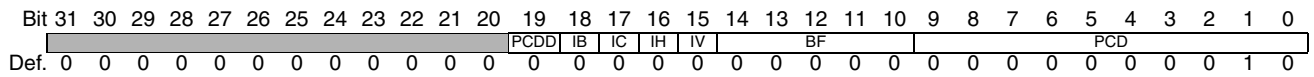
Bits	Name	Description	R/W	Default
31:24	VN2	Vertical Non Display Period 2 (in lines). Value programmed is one line less than actual value. This parameter is not used with STN panels.	R/W	0
23:16	VN1	Vertical Non Display Period 1 (in lines). Value programmed is one line less than actual value.	R/W	0
15:10	VPW	Vertical Sync Pulse Width (in lines). Value programmed is one line less than actual value. This value is not used with STN panels.	R/W	0
9:0	LPP	Lines Per Panel (in lines). Value programmed is one line less than actual value. For DSTN panels, program LPP to be one less than half the total number of lines per panel.	R/W	0

### 6.9.1.5 LCD Clock Control Register

The LCD Clock Control Register defines the parameters associated with the LCD pins.

#### lcd\_clkcontrol - LCD Clock Control

Offset = 0x0014



Bits	Name	Description	R/W	Default
31:20	—	Reserved	R	0
19	PCDD	Pixel Clock Divisor Disable. 0 Use the pixel clock divisor (PCD). See the PCD description below. 1 Disable the pixel clock divisor. (Pixel clock runs at same frequency as the LCD clock.)	R/W	0
18	IB	Invert Bias. 0 Do not invert the signal. 1 Invert the signal.	R/W	0
17	IC	Invert Pixel Clock. 0 Data is launched on the rising edge of pixel clock 1 Data ia launched on the falling edge of pixel clock	R/W	0
16	IH	Invert Line Clock	R/W	0
15	IV	Invert Frame Clock. 0 Do not invert the clock. 1 Invert the clock.	R/W	0
14:10	BF	BIAS Signal Frequency. Used only with STN panels. The Bias signal will toggle every BF Line clocks. The value programmed is one line less than the actual number of lines.	R/W	0

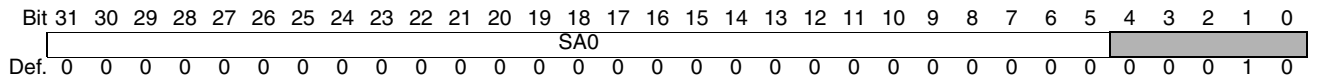
Bits	Name	Description	R/W	Default
9:0	PCD	Pixel Clock Divisor. Determines the pixel clock frequency derived from the LCD controller clock where: $F_{pck} = \text{LCD Clock} / (2 * (\text{PCD}+1))$ . For STN mono 8-bit panels PCD must be greater than 2. For STN mono 4-bit panels PCD must be greater than zero. For STN color panels PCD must be greater than zero.  Note: To run the pixel clock at the same frequency as the LCD clock, disable the divisor by setting PCDD (bit 19).	R/W	0

**6.9.1.6 LCD DMA Start Address 0 Register**

This address represents the DMA frame buffer base address for single panel STN or TFT panels. For dual STN panels this is the upper frame buffer start address.

**lcd\_dmaaddr0 - LCD DMA Start Address 0**

Offset = 0x0018



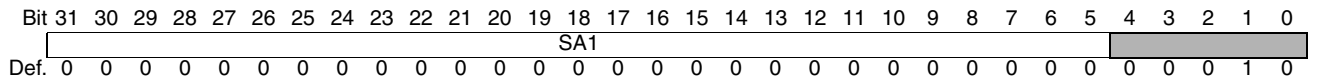
Bits	Name	Description	Read/Write	Default
31:5	SA0	Frame Buffer Start Address 0. This is a physical address and must be cache line aligned.	R/W	0
4:0	—	Reserved, should be cleared.	R/W	0

**6.9.1.7 LCD DMA Start Address 1 Register**

This address represents the DMA frame buffer base address for the lower frame buffer on dual STN panels. This is not used with TFT panels.

**lcd\_dmaaddr1 - LCD DMA Start Address 1**

Offset = 0x001C



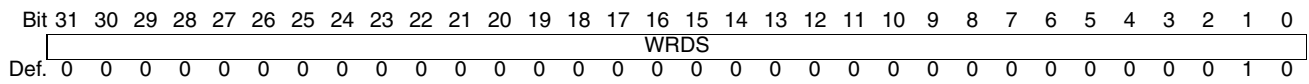
Bits	Name	Description	R/W	Default
31:5	SA1	Frame Buffer Start Address 1. This is a physical address and must be cache line aligned.	R/W	0
4:0	—	Reserved, should be cleared.	R/W	0

### 6.9.1.8 Frame Buffer Words Register

This register represents the number of words in the frame buffer.

#### lcd\_words - Frame Buffer Words

Offset = 0x0020



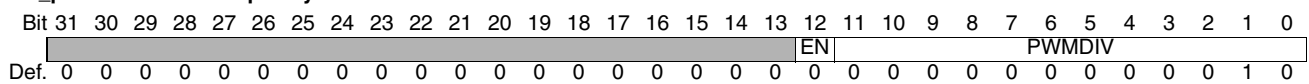
Bits	Name	Description	R/W	Default
31:0	WRDS	Frame Buffer Words. 90 & 270 degree swivel: Words per frame buffer line. The value programmed is 1 less than the actual value. 0 & 180 degree swivel: Words in entire frame buffer. In 180 degree swivel this value must be evenly divisible by 8. The value programmed is 1 less than the actual value.	R/W	0

### 6.9.1.9 Pulse Width Modulation Frequency Divider

This register controls the frequency of the two pulse width modulation (PWM) clocks LCD\_PWM[1:0]. The PWM clocks are based off the LCD clock.

#### lcd\_pwmdiv - PWM Frequency Divider

Offset = 0x0024



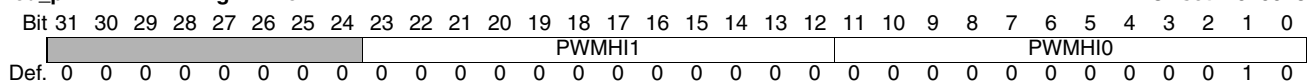
Bits	Name	Description	R/W	Default
31:11	—	Reserved.	R	0
12	EN	Enable. 0 Disable PWM clocks 1 Enable PWM clocks	R/W	0
11:0	PWMDIV	PWM Frequency Divider. Determines the frequency for the PWM clocks: $F_{PWM} = LCD\ Clock / (PWMDIV + 1)$	R/W	0

### 6.9.1.10 Pulse Width Modulation High Time

This register controls the duty cycle of the 2 PWM clocks.

#### lcd\_pwmhi - PWM High Time

Offset = 0x0028



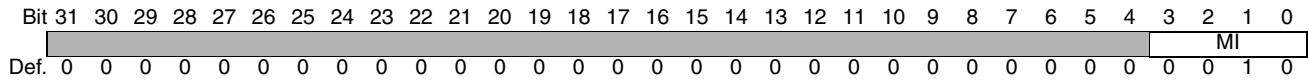
Bits	Name	Description	R/W	Default
31:24	—	Reserved	R	0
23:12	PWMHI1	PWM high time for clock 1. Duty Cycle = (PWMHI1 + 1) / (PWMDIV + 1)	R/W	0
11:0	PWMHI0	PWM high time for clock 0. Duty Cycle = (PWMHI0 + 1) / (PWMDIV + 1)	R/W	0

**6.9.1.11 LCD Palette Interface Registers**

The 256 color palette entries in the controller are read and written through the following 32-bit palette interface registers mapped to offset range 0x400 - 0x7FC. All registers must be accessed as words.

**lcd\_palettebase MONOCHROME MODE**

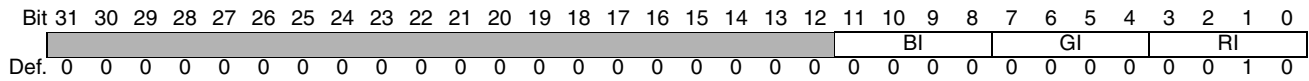
Offset Mapped = 0x0400 - 0x07FC



Bits	Name	Description	R/W	Default
31:4	—	Reserved	R	0
3:0	MI	Monochromatic Panel Intensity	R/W	0

**lcd\_palettebase COLOR STN MODE**

Offset Mapped = 0x0400 - 0x07FC

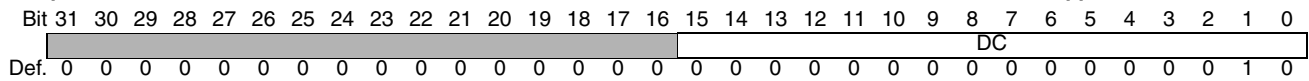


Bits	Name	Description	R/W	Default
31:12	—	Reserved	R	0
11:8	RI	Red (Note 1) channel intensity	R/W	0
7:4	GI	Green channel intensity	R/W	0
3:0	BI	Blue (Note 1) channel intensity	R/W	0

Note 1. These values are swapped when **lcd\_control**[CCO] is set.

**lcd\_palettebase COLOR TFT PALLETIZED**

Offset Mapped = 0x0400 - 0x07FC



Bits	Name	Description	R/W	Default
31:16	—	Reserved	R	0
15:0	DC	16-bit Direct True Color Value. The bit fields of this value are described by SBPPF.	R/W	0

## 6.9.2 Hardware Considerations

The LCD controller interface consists of the signals listed in Table 6-41.

**Table 6-41. LCD Controller Signals**

Signal	Input/Output	Definition
LCD_FCK	O	Frame Clock
LCD_LCK	O	Line Clock
LCD_PCK	O	Pixel Clock
LCD_D[15:0]	O	LCD Data Bus
LCD_BIAS	O	BIAS Clock
LCD_LEND	O	Line End
LCD_PWM0	O	Pulse Width Modulation Clock 0
LCD_PWM1	O	Pulse Width Modulation Clock 1

The usage of the 16 LCD\_D pins is summarized in Table 6-42.

**Table 6-42. LCD Controller Data Pin Usage**

LCD Pin Name	Mono STN Panel		Color STN Panel		Color TFT Panel (Note 1)	
	4-bit	8-bit	Single	Dual	12-bit	18-bit
LCD_D[0]	D0	D0	D0	D0	R0	R1
LCD_D[1]	D1	D1	D1	D1	R1	R2
LCD_D[2]	D2	D2	D2	D2	R2	R3
LCD_D[3]	D3	D3	D3	D3	R3	R4
LCD_D[4]	driven low	D4	D4	D4	G0	R5
LCD_D[5]	driven low	D5	D5	D5	G1	G0
LCD_D[6]	driven low	D6	D6	D6	G2	G1
LCD_D[7]	driven low	D7	D7	D7	G3	G2
LCD_D[8]	driven low	driven low	driven low	D8	B0	G3
LCD_D[9]	driven low	driven low	driven low	D9	B1	G4
LCD_D[10]	driven low	driven low	driven low	D10	B2	G5
LCD_D[11]	driven low	driven low	driven low	D11	B3	B1
LCD_D[12]	driven low	driven low	driven low	D12	driven low	B2
LCD_D[13]	driven low	driven low	driven low	D13	driven low	B3
LCD_D[14]	driven low	driven low	driven low	D14	driven low	B4
LCD_D[15]	driven low	driven low	driven low	D15	driven low	B5

Note 1. For TFT panels the R and B pins are reversed if **lcd\_control**[CCO] is set.

### 6.9.3 Programming Considerations

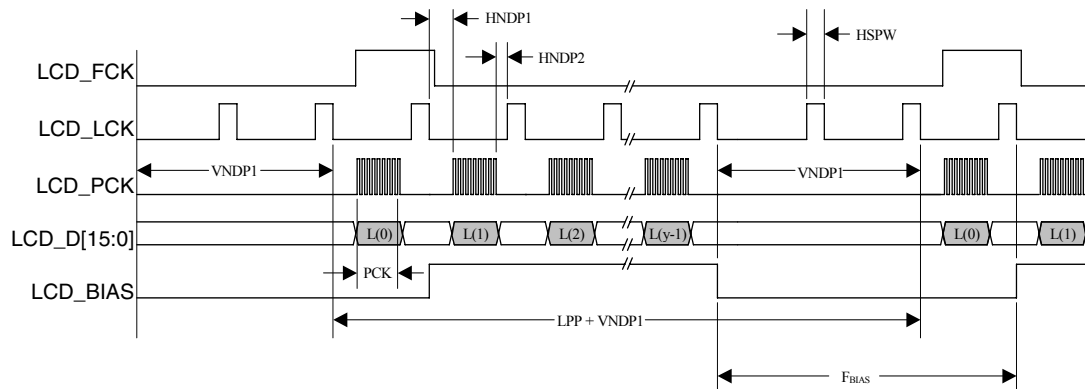
#### 6.9.3.1 Enabling the LCD Controller

The first step in enabling the LCD controller is to program the LCD clock source generator to the desired frequency. (See Section 7.1.2 "Clock Generation" on page 195.)

The LCD controller's configuration should not be changed while the controller is enabled. When starting the LCD controller the configuration for the panel should be programmed then the GO bit should be written high. In order to disable the controller the GO bit should be written low then software should wait for the SD bit to go high before re-configuring the controller.

#### 6.9.3.2 Definition of Timing Parameters

The timing diagrams shown in Figure 6-6 and Figure 6-7 on page 177 show the definitions of the timing registers plus an example for each mode.



**Notes:**

$$PCK_{color} = (PixelsPerLine * 3) / DataBusWidth$$

$$PCK_{mono} = (PixelsPerLine) / DataBusWidth$$

$$F_{PCK} = (LCD\ Clock) / (2 * (PCD + 1))$$

**In this diagram:**

$$F_{BIAS} = 3 \quad (lcd\_clkcontrol[BF] = 0b00010)$$

$$HNDP1 = 4 \quad (lcd\_horztiming[HN1] = 0b00000011)$$

$$HNDP2 = 2 \quad (lcd\_horztiming[HN2] = 0b00000001)$$

$$HSPW = 3 \quad (lcd\_horztiming[HPW] = 0b000010)$$

$$VNDP1 = 2 \quad (lcd\_verttiming[VN1] = 0b000001)$$

$$LPP = y$$

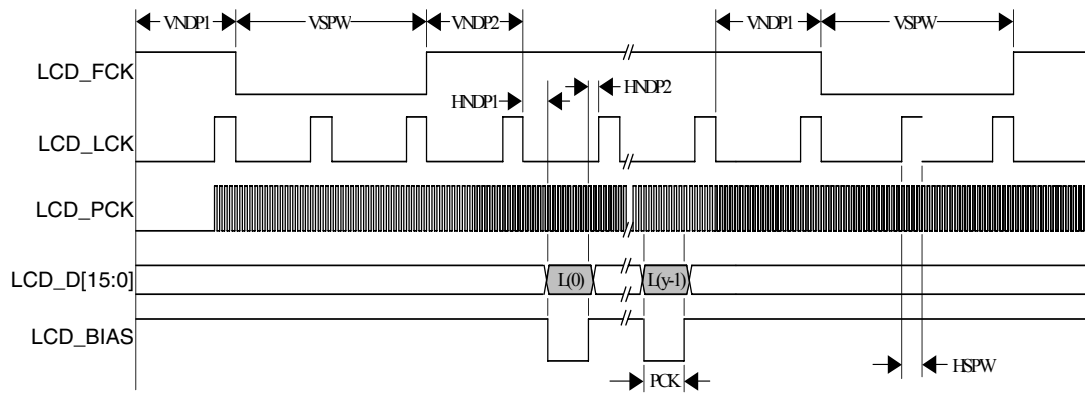
FRAME ↑ transitions at the same time as first PCLK ↑

FRAME ↓ transitions one PCLK period after LCLK ↓

FRAME, LCLK, PCLK shown here with  
 $lcd\_clkcontrol[IC:IH:IV] = 0b000$

**Figure 6-6. STN (Passive Mode) Timing**





**Notes:**

**In this diagram:**

- PCK = PixelsPerLine
- $F_{PCK} = (\text{LCD Clock}) / (2 * (\text{PCD} + 1))$
- $F_{BIAS} = \text{NA}$  (`lcd_clkcontrol[BF] = 0bXXXXXX`)
- HNDP1 = 5 (`lcd_horztiming[HN1] = 0b00000100`)
- HNDP2 = 2 (`lcd_horztiming[HN2] = 0b00000001`)
- HSPW = 4 (`lcd_horztiming[HPW] = 0b000011`)
- VNDP1 = 1 (`lcd_verttiming[VN1] = 0b00000000`)
- VNDP2 = 1 (`lcd_verttiming[VN2] = 0b00000000`)
- VSPW = 2 (`lcd_verttiming[VPW] = 0b0000001`)
- LPP = y
- FRAME transitions at the same time as LCLK goes inactive
- FRAME, LCLK, PCLK, and BIAS shown here with `lcd_clkcontrol[IB:IC:IH:IV] = 0b1001`

**Figure 6-7. TFT (Active Mode) Timing**

## 6.10 Secure Digital (SD) Controller

The Au1100 has two Secure Digital (SD) controllers which incorporate both SD and SDIO interfaces. The peripheral bus clock is used as the clock reference for the SD controllers.

The SD controllers comply with version 1.1 of the SD card specification. References in this section are to that version of the specification.

### 6.10.1 SD Registers

Each SD controller (SD0 and SD1) has its own block of control and configuration registers with physical base address shown in Table 6-43. The register block consists of the registers shown in Table 6-44.

**Table 6-43. SD Base Address**

Name	Physical Base Address	KSEG1 Base Address
sd0_base	0x0 1060 0000	0xB060 0000
sd1_base	0x0 1068 0000	0xB068 0000

**Table 6-44. SD Registers**

Offset (Note 1)	Register Name	Description
0x0000	sd_txport	Destination data port for PIO or DMA writes
0x0004	sd_rxport	Source data port for PIO or DMA reads
0x0008	sd_config	Interrupt and Clock Configuration
0x000C	sd_enable	SD Peripheral Control
0x0010	sd_config2	Protocol and data transfer mode configuration
0x0014	sd_blksize	data block transfer size
0x0018	sd_status	Interrupt Status
0x001C	sd_debug	Debug Info
0x0020	sd_cmd	SD Command Register
0x0024	sd_cmdarg	SD Command Argument Register
0x0028	sd_resp3	SD Report Response 3
0x002C	sd_resp2	SD Report Response 2
0x0030	sd_resp1	SD Report Response 1
0x0034	sd_resp0	SD Report Response 0
0x0038	sd_timeout	SD NAC Timeout Value

Note 1. See Table 6-43 for base address.

### 6.10.1.1 SD Transmit Data Port Register

The transmit data port register (**sdn\_txport**) is used to send data to the SD interface for either PIO or DMA write modes. A write to **sdn\_txport** pushes an entry into the 8-byte transmit FIFO.

sdn_txport		Offset = 0x0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Def.	0 0	TXD

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7:0	TXD	Transmit data.	W	0

### 6.10.1.2 SD Receive Data Port Register

The receive data port register (**sdn\_rxport**) is used to read data from the SD interface from either PIO or DMA read modes. A read from **sdn\_rxport** pops an entry from the 8-byte receive FIFO.

sdn_rxport		Offset = 0x0004
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Def.	0 0	RXD

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7:0	RXD	Receive data.	R	0

### 6.10.1.3 SD Configuration Register

The SD configuration register (**sdn\_config**) is used to enable interrupts and configure SD clocks.

sdn_config		Offset = 0x0008
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Def.	0 0	DIV

Bits	Name	Description	R/W	Default
31	SI	SDIO device interrupt enable.	R/W	0
30	CD	Card insertion/removal detect interrupt enable.	R/W	0
29	RF	RX buffer full interrupt enable.	R/W	0
28	RA	RX buffer almost full (all but one entry) interrupt enable.	R/W	0
27	RH	RX buffer at least half full interrupt enable.	R/W	0
26	TA	TX buffer almost empty interrupt enable.	R/W	0
25	TE	TX buffer empty interrupt enable.	R/W	0
24	TH	TX buffer at most half empty interrupt enable.	R/W	0
23	—	Reserved.	—	—
22	WC	Write CRC error interrupt enable.	R/W	0
21	RC	Read CRC error interrupt enable.	R/W	0
20	SC	Response CRC error interrupt enable.	R/W	0
19	DT	Data access timeout interrupt enable (NAC).	R/W	0
18	DD	Data transfer done interrupt enable.	R/W	0
17	RAT	Command-response response access timeout interrupt enable (NCR).	R/W	0

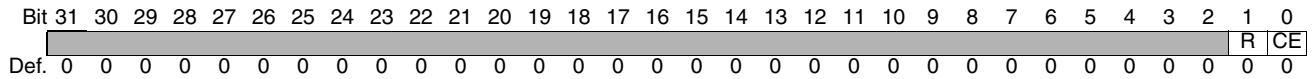
Bits	Name	Description	R/W	Default
16	CR	Command-response transfer done interrupt enable (or command only if the command does not require a response).	R/W	0
15	I	Master interrupt enable. 0 Disable all SD interrupts. 1 Enable SD interrupts. Individual interrupt enables must still be used.	R/W	0
14	RO	RX FIFO overrun interrupt enable.	R/W	0
13	RU	RX FIFO underrun interrupt enable.	R/W	0
12	TO	TX FIFO overrun interrupt enable.	R/W	0
11	TU	TX FIFO underrun interrupt enable.	R/W	0
10	NE	RX FIFO not empty interrupt enable.	R/W	0
9	DE	Divider write enable. 0 Do not change the clock divider, regardless of value written in DIV field. 1 Change the clock divider to the value written in the DIV field.	R/W	0
8:0	DIV	Clock divider. The SD clock is derived from the peripheral bus clock as follows: $SD\ clock = Peripheral\ clock / [2 * (DIV+1)]$ For example, if DIV = 0, the clock is divided by 2; if DIV = 0x1FF, the clock is divided by 1024. Note: This value must be written simultaneously with DE to take effect.	R/W	0

**6.10.1.4 SD Enable Register**

The SD enable register (**sdn\_enable**) contains bits to enable clocks and reset the SD interface.

**sdn\_enable**

Offset = 0x000C



Bits	Name	Description	R/W	Default
31:2	—	Reserved.	—	—
1	R	Peripheral reset. Clearing this bit (with the peripheral clock running) resets the entire peripheral. Set this bit for normal operation. SD reset sequence: First write 0b01 to enable/maintain the clock while resetting. Then write 0b11 to take the peripheral out of reset.	R/W	0
0	CE	Peripheral clock enable. Set this bit for normal operation. Clear this bit to disable the clock and conserve power.	R/W	0

### 6.10.1.5 SD Configuration 2 Register

The SD configuration 2 register (**sdn\_config2**) is used to set up the PIO or DMA mode and state machine master enable.

#### sdn\_config2

Offset = 0x0010

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
 Def. 0

Bits	Name	Description	R/W	Default
31:10	—	Reserved	—	—
9	RW	Read wait enable. When this bit is set serial clock-based flow control is not used. This bit is valid for SDIO mode only.	R/W	0
8	WB	Wide bus transfer mode. 0 One wire data transfer. 1 Four wire data transfer.	R/W	0
7:5	—	Reserved	—	—
4	DC	Disable hardware timeout counter. 0 Normal hardware timeout. 1 No hardware timeout (software timeout).	R/W	0
3	DF	Disable clock freezing for flow control. 0 Enable clock freezing. 1 Disable clock freezing.	R/W	0
2	—	Reserved.	R/W	0
1	FF	Force FIFO flush and reset. This bit is sticky and must be manually cleared to resume normal operation.	R/W	0
0	EN	Serial interface state machine and FIFO master enable. This bit must be set to enable the SD controller.	R/W	0

### 6.10.1.6 SD Block Size Register

The SD block size register (**sdn\_blksize**) defines the size and number of blocks to be transmitted by the SD controller.

#### sdn\_blksize

Offset = 0x0014

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
 Def. 0

Bits	Name	Description	R/W	Default
31:25	—	Reserved.	—	—
24:16	BC	Block I/O count. This field is used for a known number of SDIO R/W blocks. The number of blocks to be transferred is (BC + 1).	R/W	0
15:11	—	Reserved.	—	—
10:0	BS	Block size in bytes. The value programmed is one less than the actual number of bytes in the block. (Block size = BS + 1)	R/W	0

### 6.10.1.7 SD Status Register

The SD status register (**sdn\_status**) reports pending interrupts and the cause of the interrupts. Each field has a description of the interrupt type: level triggered (LT) or edge triggered (ET). To clear an edge-triggered interrupt, write a '1' to the appropriate bit. A level-triggered interrupt bit is cleared when the triggering event no longer applies. This register also contains the CRC status word resulting from a block write. The **sdn\_status** bits always reflect the current status, regardless of the corresponding **sdn\_config** bits.

**sdn\_status**

Offset = 0x0018

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SI	CD	RA	RF	RH	TA	TE	TH		WC	RC	SC	DT	DD	RAT	CR	I	RO	RU	TO	TU	NE		D3	CF	DB	CB			DCRCW		
Def. 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Level/Edge	R/W	Default
31	SI	SDIO device interrupt.	LT	R/W	0
30	CD	Card insertion/removal detect interrupt.	ET	R/W	0
29	RF	RX buffer full interrupt.	LT	R/W	0
28	RA	RX buffer almost full (all but one entry) interrupt.	LT	R/W	0
27	RH	RX buffer at least half full interrupt.	LT	R/W	0
26	TA	TX buffer almost empty interrupt.	LT	R/W	0
25	TE	TX buffer empty interrupt.	LT	R/W	0
24	TH	TX buffer at most half empty interrupt.	LT	R/W	0
23	—	Reserved.	—	—	—
22	WC	Write CRC error interrupt.	ET	R/W	0
21	RC	Read CRC error interrupt.	ET	R/W	0
20	SC	Response CRC error interrupt.	ET	R/W	0
19	DT	Data access timeout interrupt (NAC).	ET	R/W	0
18	DD	Data transfer done interrupt	ET	R/W	0
17	RAT	Command-response response access timeout interrupt (NCR).	ET	R/W	0
16	CR	Command-response transfer done interrupt (or command only if the command does not require a response).	ET	R/W	0
15	I	Master interrupt. Reads a 1 when any unmasked (enabled) interrupt is taken. This bit is cleared automatically once all unmasked interrupts are cleared.	—	R	0
14	RO	RX FIFO overrun interrupt.	ET	R/W	0
13	RU	RX FIFO underrun interrupt.	ET	R/W	0
12	TO	TX FIFO overrun interrupt.	ET	R/W	0
11	TU	TX FIFO underrun interrupt.	ET	R/W	0
10	NE	RX FIFO not empty interrupt.	LT	R/W	0
9:8	—	Reserved.	—	—	—
7	D3	Real-time direct sample of SDMS <sub>n</sub> _DAT[3] provided for software debouncing.	—	R/W	0
6	CF	Clock freezing status. 0 Clock frozen (potential overrun/underrun) 1 Normal clocking	—	R	0
5	DB	SD data-response busy status.	—	R	0
4	CB	SD command-response busy status.	—	R	0
3	—	Reserved.	—	—	—
2:0	DCRCW	Device CRC status word. 010 No error. 101 Transmission error. 111 No CRC response. All other bit combinations are undefined.	—	R	0

### 6.10.1.8 SD Debug Register

The SD debug register (**sdn\_debug**) is used for read-only access to the read and write pointers for both transmit and receive FIFOs. The pointers contain the entry number for the active FIFO entry.

#### sdn\_debug

Offset = 0x001C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
																RXR		RXW		TXR		TXW												
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	Name	Description	R/W	Default
31:15	—	Reserved.	—	—
14:12	RXR	Receive FIFO read pointer.	R	0
11	—	Reserved.	—	—
10:8	RXW	Receive FIFO write pointer.	R	0
7	—	Reserved.	—	—
6:4	TXR	Transmit FIFO read pointer.	R	0
3	—	Reserved.	—	—
2:0	TXW	Transmit FIFO write pointer.	R	0

### 6.10.1.9 SD Command Register

The SD command register (**sdn\_cmd**) contains fields used to build an SD command sequence.

#### sdn\_cmd

Offset = 0x0020

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
									RT				CI				CT				RY	GO											
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:16	RT	Response type. 0x00 No response. 0x01 R1 response (48 bits). 0x02 R2 response (136 bits). 0x03 R3 response (48 bits). 0x04 R4 response (48 bits)—SDIO only. 0x05 R5 response (48 bits)—SDIO only. 0x06 R6 response (48 bits). 0x81 R1b response (48 bits). All other values are reserved.	R/W	0
15:8	CI	Command index. See SD specification for command listing.	R/W	0
7:4	CT	Command type. Must be written with each command. See Table 6-45 on page 184 for valid encoding and descriptions.	R/W	0
3:2	—	Reserved.	—	—
1	RY	Response ready. This bit is set by the SD block once the command-response sequence is finished and automatically cleared once <b>sdn_resp0</b> is read.	R	0
0	GO	Command go/busy. This bit is set to initiate a command. The bit is automatically cleared once the last bit of the command argument is transmitted.	R/W	0

Table 6-45. Command Type Field Encodings

CT[3:0]	Action applied to SD Memory	Action applied to SDIO
0000	Non-data-write, non-data-read, non-data-stop, non-io-abort commands.	Non-data-write, non-data-read, non-data-stop, non-io-abort commands.
0001	Single block write. Use when doing a WRITE_BLOCK (CMD24) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored.	Single block IO write. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 0 (byte mode). The block size is defined in Byte/Block Count. A 0x0 value in Byte/Block Count is considered to be 256 bytes (see p.18 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored.
0010	Single block read. Use when doing a READ_SINGLE_BLOCK (CMD17) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored.	Single block IO read. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 0 (byte mode). the block size is defined in Byte/Block Count. A 0x0 value in Byte/Block Count is considered to be 256 bytes (see p.18 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored.
0011	Multiple block write requiring STOP command to end transfer. Use when doing a WRITE_MULTIPLE_BLOCK (CMD25) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored. The transfer has to be terminated by issuing a STOP_TRANSMISSION (CMD12) command.	Multiple block IO write requiring writing to CCCR to end transfer. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 1 (block mode) and Byte/Block Count = 0x0 (infinite block count) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored. The transfer has to be terminated by issuing a IO_RW_DIRECT (CMD52) command to write to the abort register in CCCR (bits [2:0] of register 6) (see p.23 of SDIO spec).
0100	Multiple block read requiring STOP command to end transfer. Use when doing a READ_MULTIPLE_BLOCK (CMD18) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored. The transfer has to be terminated by issuing a STOP_TRANSMISSION (CMD12) command.	Multiple block IO read requiring writing to CCCR to end transfer. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 1 (block mode) and Byte/Block Count = 0x0 (infinite block count) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. <b>sdn_blksize</b> [BC] is ignored. The transfer has to be terminated by issuing a IO_RW_DIRECT (CMD52) command to write to the abort register in CCCR (bits [2:0] of register 6) (see p.23 of SDIO spec).



Table 6-45. Command Type Field Encodings (Continued)

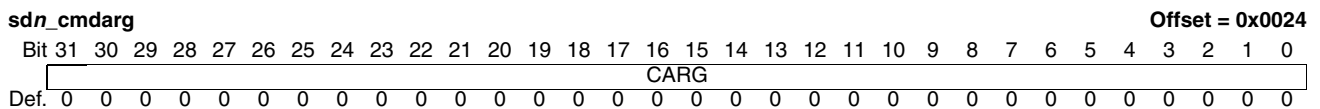
CT[3:0]	Action applied to SD Memory	Action applied to SDIO
0101	Not applicable	Multiple block IO write with fixed number of blocks. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 1 (block mode) and Byte/Block Count set to the desired number of blocks to transfer (must be non-zero) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. Based on the Byte/Block Count, <b>sdn_blksize</b> [BC] is programmed with the correct number of blocks. Using either 1-bit or 4-bit wire will not affect this number because in the 4-bit wire case, 1 block of data is split into 4 sub-blocks (each 1/4 of the original block size) on each data wire. The start and stop bits still define the boundary of a block. The transfer will be terminated when the correct number of blocks have been transmitted. No abort action is required.
0110	Not applicable	Multiple block IO read with fixed number of blocks. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 1 (block mode) and Byte/Block Count set to the desired number of blocks to transfer (must be non-zero) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in <b>sdn_blksize</b> [BS]. Based on the Byte/Block Count, <b>sdn_blksize</b> [BC] is programmed with the correct number of blocks. Using either 1-bit or 4-bit wire will not affect this number because in the 4-bit wire case, 1 block of data is split into 4 sub-blocks (each 1/4 of the original block size) on each data wire. The start and stop bits still define the boundary of a block. The transfer will be terminated when the correct number of blocks have been transmitted. No abort action is required.
0111	Terminate transfer of a multiple block write or read. Use when doing a STOP_TRANSMISSION (CMD12) command (see p.41 of SD spec).	Not applicable.
1000	Not applicable	Terminate transfer of a multiple block IO write or read without a fixed desired number of block count. Use when issuing a IO_RW_DIRECT (CMD52) command to write to the abort register in CCCR (bits [2:0] of register 6) to stop the transfer (see p.23 of SDIO spec).

**Table 6-45. Command Type Field Encodings (Continued)**

CT[3:0]	Action applied to SD Memory	Action applied to SDIO
1001	Not applicable	<p>Suspend current data transfer. Use when issuing an IO_RW_DIRECT (CMD52) command to set BR=1 (see pp.37-39 of SDIO spec).</p> <p>The host needs to check the response for BS=0 to know that the data transfer is suspended. The host is free to do other commands. To resume, the host has to issue the command again. For example, if the host suspends a multiple write with undefined number of blocks, the same command has to issue again for the “resume”. If the host suspends a multiple read with defined number of blocks, the host has to find out the number of remaining blocks of data yet to be transferred and set it up in the multiple read command for the “resume”.</p> <p>A suspension can be done only at a block boundary when (i) in a read wait state waiting for the start bit of the next block when the device will hold off data when receiving the suspend command; (ii) during the write busy state when the device is sending logic low on data bit 0.</p>
1010 to 1111	Reserved	Reserved

**6.10.1.10 SD Command Argument Register**

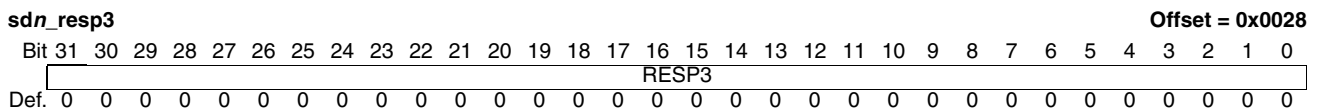
The SD command argument register (**sdn\_cmdarg**) holds the argument used in an SD command-response sequence.



Bits	Name	Description	R/W	Default
31:0	CARG	Command argument. Must write this register first, then write <b>sdn_cmd</b> [BY] to issue the command.	R/W	0

**6.10.1.11 SD Response 3 Register**

The SD response 3 register (**sdn\_resp3**) contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits.



Bits	Name	Description	R/W	Default
31:0	RESP3	Bits 127:96 of response from device.	R	0

### 6.10.1.12 SD Response 2 Register

The SD response 2 register (**sdn\_resp2**) contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits.

<b>sdn_resp2</b>		<b>Offset = 0x002C</b>															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	RESP2																
Def.	0 0																

Bits	Name	Description	R/W	Default
31:0	RESP2	Bits 95:64 of response from device.	R	0

### 6.10.1.13 SD Response 1 Register

The SD response 1 register (**sdn\_resp1**) contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits or (6 + 32) bits.

<b>sdn_resp1</b>		<b>Offset = 0x0030</b>															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	RESP1																
Def.	0 0																

Bits	Name	Description	R/W	Default
31:0	RESP1	Bits 63:32 of response from device.	R	0

### 6.10.1.14 SD Response 0 Register

The SD response 0 register (**sdn\_resp0**) contains the response from an issued command-response sequence. Valid for all the expected response lengths.

<b>sdn_resp0</b>		<b>Offset = 0x0034</b>															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	RESP0																
Def.	0 0																

Bits	Name	Description	R/W	Default
31:0	RESP0	Bits 31:0 of response from device.	R	0

### 6.10.1.15 SD Timeout Register

The SD timeout register (**sdn\_timeout**) defines the timeout value for NAC.

<b>sdn_timeout</b>		<b>Offset = 0x0038</b>															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																
	TMAX																
Def.	0 0																

Bits	Name	Description	R/W	Default
31:21	—	Reserved.	—	—
20:0	TMAX	Maximum timeout value for NAC where: NAC = TAAC + NSAC (See the SD specification.) If using a 25MHz clock, the maximum timeout value is 81.02 ms.	R/W	0

### 6.10.2 Hardware Considerations

The SD interface consists of the signals listed in Table 6-46.

**Table 6-46. SD Signals**

Signal	Input/Output	Definition
SDMS_MS_EN	I	Reserved for future use. Must be 0.
SDMS0_CLK	O	SD Card 0 Interface Clock.
SDMS0_CMD	I/O	SD Card 0 Half Duplex Command and Response.
SDMS0_DAT[3:0]	I/O	SD Card 0 Data Bus.
SDMS1_CLK	O	SD Card 1 Interface Clock.
SDMS1_CMD	I/O	SD Card 1 Half Duplex Command and Response.
SDMS1_DAT[3:0]	I/O	SD Card 1 Data Bus.

## 6.11 Secondary General Purpose I/O

The Au1100 processor contains two GPIO blocks (primary and secondary). This section describes the programming model of the *secondary* GPIO block which corresponds to signals labeled GPIO[200] through GPIO[215]. (For a description of the primary GPIO block refer to Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209 in the system control block description. Note that some GPIO2 signals share pins with internal peripherals; Section 7.3 also covers how to configure the pin functionality for these signals.)

### 6.11.1 GPIO2 Programming Model

The secondary GPIO (GPIO2) logic block is controlled by a register block referenced from the base address described in Table 6-47.

**Table 6-47. GPIO2 Register Base Addresses**

Name	Physical Base Address	KSEG1 Base Address
gpio2_base	0x0 1170 0000	0xB170 0000

### 6.11.2 GPIO2 Registers

The secondary GPIO register block is shown in Table 6-48.

**Table 6-48. GPIO2 Registers**

Offset (Note 1)	Register Name	Description
0x0000	gpio2_dir	GPIO2 Direction
0x0004	—	Reserved
0x0008	gpio2_output	GPIO2 Data Output
0x000C	gpio2_pinstate	GPIO2 Pin State
0x0010	gpio2_inten	GPIO2 Interrupt Enable (for GPIO[215:208])
0x0014	gpio2_enable	GPIO2 Enable

Note 1. See Table 6-47 for base address.

#### 6.11.2.1 Direction Register

The **gpio2\_dir** register controls the direction of each GPIO2 signal. Note that this register only controls the output enable for the output buffer. Clearing a bit in this register disables the output for the corresponding pin making it possible to read an externally driven input. Output enable control can also be used to emulate an open drain driver.

##### gpio2\_dir - Direction Register

Offset = 0x0000

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIR															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

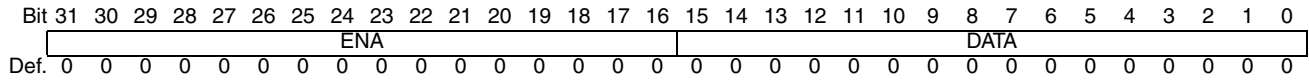
Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	DIR	Direction Control. Each bit controls the I/O direction of one GPIO signal in the secondary block. Bits 15:0 correspond to GPIO[215:200]. 0 Pin is an input (output disabled). 1 Pin is an output.	R/W	0

### 6.11.2.2 Data Output Register

The **gpio2\_output** register controls the output data for the secondary GPIOs. Data bits 15:0 are output to the corresponding GPIO when the enable bit is set for that bit during a write to this register. For example, to output a '1' on GPIO[200] and a '0' on GPIO[201] without changing the output of any other GPIOs, write the value 0x00030001 to **gpio2\_output**.

#### gpio2\_output - Data Output Register

Offset = 0x0008



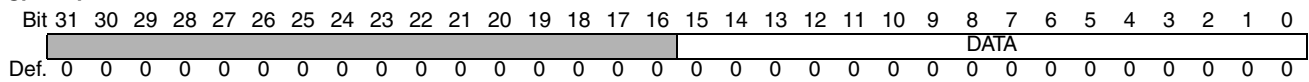
Bits	Name	Description	R/W	Default
31:16	ENA[15:0]	Data Output Write Enable. ENA[15:0] corresponds to DATA[15:0]. Note that ENA is write-only and should be ignored on reads. 0 Disable modifications to corresponding bit in DATA[15:0]. 1 Enable modifications to corresponding bit in DATA[15:0].	W	0
15:0	DATA[15:0]	Output Data. DATA[15:0] corresponds to GPIO[215:200]. The DATA bit values are reflected in the corresponding GPIO output signal level. When modifying a bit in DATA[15:0], the corresponding bit in ENA[15:0] must be set to allow the write. This mechanism allows individual data bits to be modified without affecting DATA[15:0] as a whole.	R/W	0

### 6.11.2.3 Pin State Register

The **gpio2\_pinstate** register reflects the current state of the corresponding secondary GPIO pin.

#### gpio2\_pinstate - Pin State

Offset = 0x000C



Bits	Name	Description	R/W	Default
31:16	—	These bits are reserved and will be read 0.	R	0
15:0	DATA	Current Pin State for GPIO[215:200]	R	0

### 6.11.2.4 Interrupt Enable Register

The **gpio2\_inten** register contains bits which enable interrupts under certain operational conditions. Note that **gpio2\_inten** applies only to interrupts on GPIO[215:208]. The GPIO[215:208] signals are OR'd together to create *one* interrupt source (source number 29 on interrupt controller 0), as shown in Figure 6-8. See Section 5.0 "Interrupt Controller" on page 83 for more information on how to program interrupts.

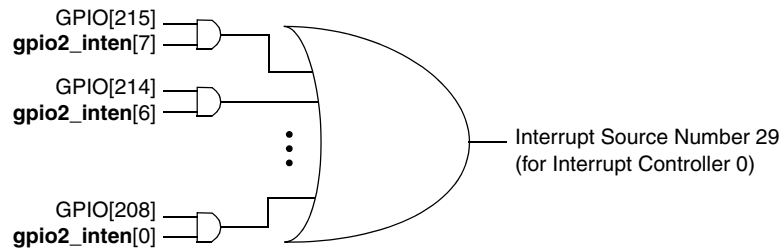
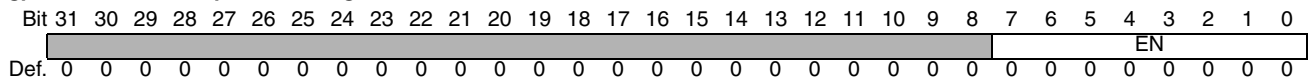


Figure 6-8. Logic for Interrupt Source Number 29 on Interrupt Controller 0

#### gpio2\_inten - Interrupt Enable Register

Offset = 0x0010



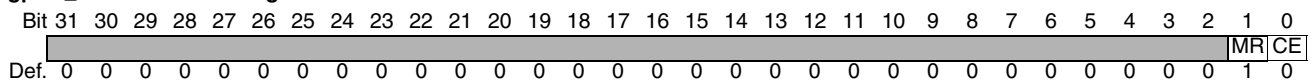
Bits	Name	Description	R/W	Default
31:8	—	Reserved, should be cleared.	R	0
7:0	EN	Interrupt enable bits [7:0] correspond to GPIO[215:208]. Setting a bit enables the signal's OR'd contribution to interrupt source number 29 (on interrupt controller 0).	R/W	0

### 6.11.2.5 Enable Register

The **gpio2\_enable** register controls the clocks and reset to the secondary GPIO block.

#### gpio2\_enable - Enable Register

Offset = 0x0014



Bits	Name	Description	R/W	Default
31:2	—	Reserved, should be cleared.	R	0
1	MR	Module Reset. When this bit is set the module is held in reset.	R/W	1
0	CE	Clock Enable. When this bit is clear the module clocks are disabled.	R/W	0





# System Control 7

The Au1100 processor contains a robust system control strategy that includes the means to control the following:

- Clocking (See Section 7.1 "Clocks" on page 194.)
- Time of Year and Real Time Clock counters (See Section 7.2 "Time of Year Clock and Real Time Clock" on page 204.)
- GPIO control (See Section 7.3 "Primary General Purpose I/O and Pin Functionality" on page 209.)
- Power management (See Section 7.4 "Power Management" on page 214.)

All registers in the system control block are located off of the base address shown in Table 7-1.

**Table 7-1. System Control Block Base Address**

Name	Physical Base Address	KSEG1 Base Address
sys_base	0x0 1190 0000	0xB190 0000

The registers in the system control block are affected differently by events such as power-on hardware reset, Sleep and runtime reset (see Section 8.0 "Power-up, Reset and Boot" on page 223 for a discussion on the different reset types). Each register is documented with how it will be affected by the different system states. Care should be taken by the system designer to observe what registers will and will not revert to defaults when the different events occur.

## 7.1 Clocks

The Au1100 processor supports two oscillator inputs: 12 MHz and 32.768 kHz. This section documents the clock domains driven directly and indirectly by the 12 MHz input. The 32.768 kHz clock input drives the two programmable counters intended for use as a real time clock (RTC) and time of year clock (TOY). The programmable counters are documented in Section 7.2 "Time of Year Clock and Real Time Clock" on page 204. (See Section 11.8 "Crystal Specifications" on page 283 for the specifications of both crystals.)

The Au1100 processor contains two PLLs driven by the 12-MHz oscillator and a clocking block from which the following are derived:

- CPU Clock
- Core Cycle Counter register clocked by the CPU clock
- System Bus (SBUS) clock
- Peripheral Bus Clock
- SDRAM Bus Clock
- Programmable clocks needed by certain peripherals
- Programmable clocks EXTCLK[1:0] for external use (provided on pins shared with the GPIO[3:2] signals)

Figure 7-1 shows the basic clocking topology and the relationship between the CPU Clock, the SBUS clock and the Peripheral Clock. As shown, the SBUS frequency is derived by dividing the CPU Clock by the value SD programmed in the **sys\_powerctrl** register. (See Section 7.4 "Power Management" on page 214 for the **sys\_powerctrl** register definition.) The Peripheral Bus clock and the SDRAM bus are fixed at the SBUS frequency divided by 2. Figure 7-1 also shows the peripheral blocks driven by clock sources derived from the programmable clock generator logic (as described in Section 7.1.2 "Clock Generation" on page 195).

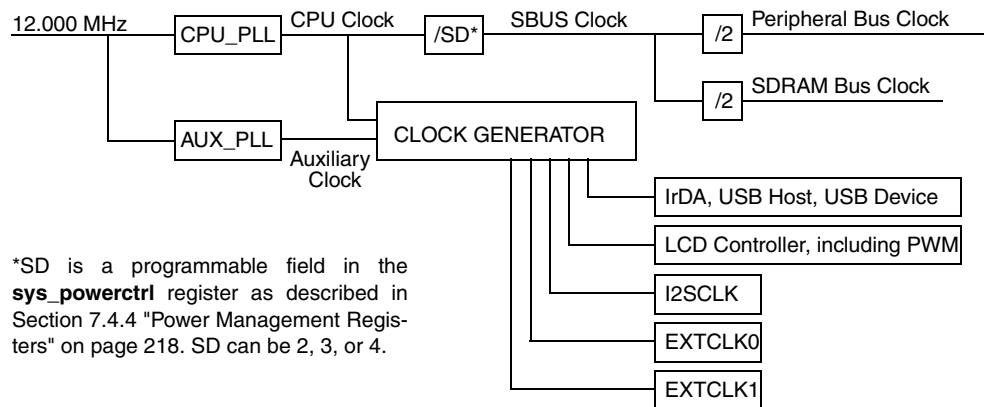


Figure 7-1. Clocking Topology

### 7.1.1 Clock Register Descriptions

The clock manager registers and their associated offsets are listed in Table 7-2.

**Table 7-2. Clock Generation Registers**

Offset (Note 1)	Register Name	Description	Reset Type
0x0020	sys_freqctrl0	Controls (source, enable, and divider) frequency generators 0, 1, and 2	Hardware
0x0024	sys_freqctrl1	Controls (source, enable, and divider) frequency generators 3, 4, and 5	Hardware
0x0028	sys_clksrc	Controls (source and divider) the derived clocks	Hardware
0x0060	sys_cpupll	Changes CPU PLL frequency	Hardware
0x0064	sys_auxpll	Changes Auxiliary PLL frequency	Hardware & Runtime

Note 1. See Table 7-1 on page 193 for base address.

### 7.1.2 Clock Generation

This section documents registers for the clock generation block which provides clocks to some peripheral devices and as well as two externally available clocks. The clock generation subsystem is split into two sets of distinct blocks which allows up to six distinct frequencies to drive up to six clock sources. Figure 7-2 on page 196 shows a logical representation of one of the six identical frequency generators and how the six frequency sources are mapped to one of the six identical internal clock sources. The names in the figure correspond to the bit names in the control registers. Figure 7-3 on page 197 shows a pictorial representation of the relationship between the frequency generator blocks to the clock source blocks.

Each peripheral has clock restrictions as follows. If these restrictions are not met, the peripheral will not operate correctly.

The USB Host Clocks, USB Device Clock and IrDA Clock must be programmed to 48 MHz. Additionally, the **ir\_config2[CS]** bit that specifies the PHY layer clock speed field must be set to match 48 MHz. See Section 6.4 "IrDA" on page 109 for more information.

The I2SCLK must be set to match the effective bit rate which will be determined by the sampling frequency (system dependent) times the bit rate ( $2 * SZ$ ). **SZ** is the Size field in the **i2s\_config** register.

The EXTCLK[1:0] clocks can be programmed for system use. If the I<sup>2</sup>S peripheral is being used, typically one of these clocks will be programmed to provide the system oversampling clock for the codec (i.e., 128Fs, 256Fs, or 512Fs where Fs is the system sampling frequency).

Note that the EXTCLK[1:0] clocks have a maximum frequency rating of ( $F_{max} / 16$ ), where  $F_{max}$  is the maximum frequency rating for the part. For example, for a 400 MHz part be sure the EXTCLK[1:0] clocks are programmed to run at no more than 25 MHz. (See also Section 11.7 "External Clock Specifications" on page 282.)

Note also that the EXTCLK[1:0] clocks are multiplexed signals and require programming of the **sys\_pifunc** register (see Section 7.3.1.1 "Pin Function" on page 209) as follows:

- EXTCLK0 shares a pin with GPIO[2]. If EXTCLK0 is to be used, **sys\_pifunc[EX0]** must be set to allow the clock to drive this pin. In addition, **sys\_pifunc[CS]** must be cleared.
- EXTCLK1 shares a pin with GPIO[3]. If EXTCLK1 is to be used, **sys\_pifunc[EX1]** must be set to allow the clock to drive this pin.

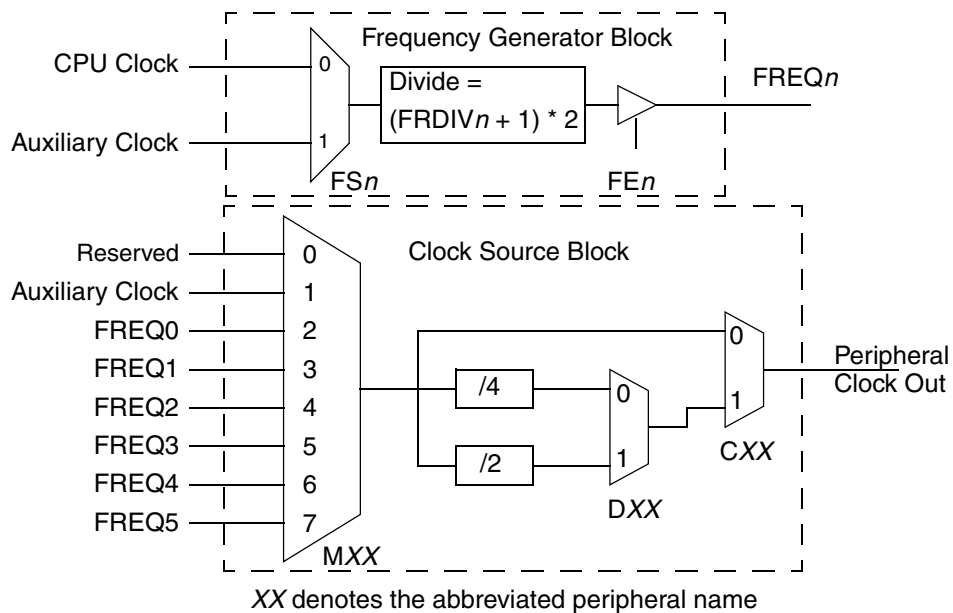


Figure 7-2. Frequency Generator and Clock Source Block Diagram

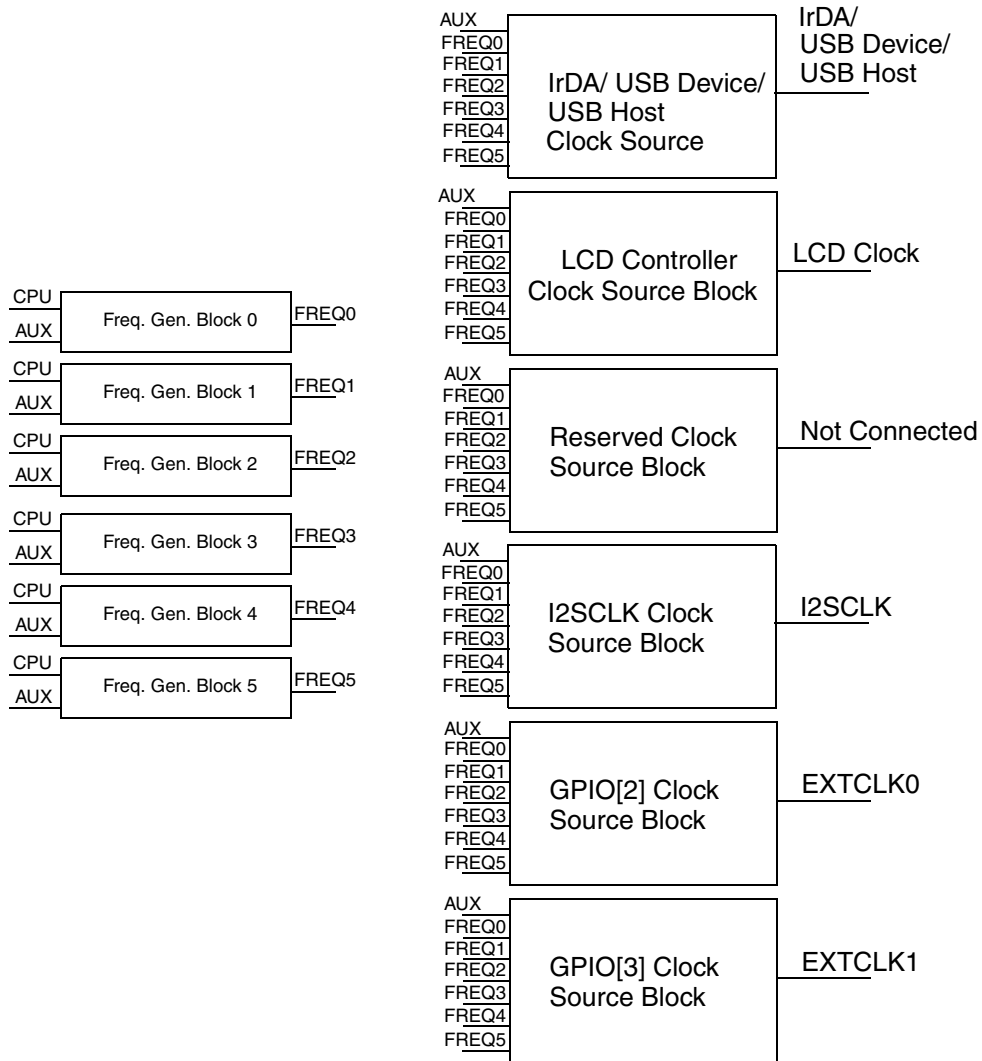


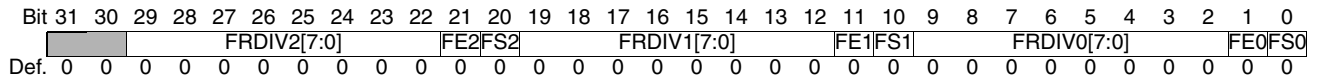
Figure 7-3. Frequency Generator and Clock Source Mapping

**7.1.2.1 Frequency Control 0**

This register controls the frequency generator block for output frequencies 0, 1, and 2. This register will reset to defaults only on a hardware reset. During a runtime reset and during Sleep this register will retain its value.

**sys\_freqctrl0**

**Offset = 0x0020**



Bits	Name	Description	R/W	Default
31:30	—	Reserved, should be cleared.	R	0
29:22	FRDIV2	Divider for Frequency Generator 2. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
21	FE2	Frequency Generator Output Enable 2 0 Disable output. 1 Enable output .	R/W	0
20	FS2	Frequency Generator 2 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0
19:12	FRDIV1	Divider for Frequency Generator 1. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
11	FE1	Frequency Generator 1 Output Enable. 0 Disable output. 1 Enable output .	R/W	0
10	FS1	Frequency Generator 1 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0
9:2	FRDIV0	Divider for Frequency Generator 0. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
1	FE0	Frequency Generator 0 Output Enable. 0 Disable output. 1 Enable output.	R/W	0
0	FS0	Frequency Generator 0 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0

### 7.1.2.2 Frequency Control 1

This register controls the frequency generator block for output frequencies 3, 4, and 5. This register will reset to defaults only on a hardware reset. During a runtime reset and during Sleep this register will retain its value.

**sys\_freqctrl1**

**Offset = 0x0024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
			FRDIV5[7:0]							FE5		FS5		FRDIV4[7:0]							FE4		FS4		FRDIV3[7:0]							FE3		FS3	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Bits	Name	Description	R/W	Default
31:30	—	Reserved, should be cleared.	R	0
29:22	FRDIV5	Divider for Frequency Generator 5. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
21	FE5	Frequency Generator 5 Output Enable. 0 Disable output. 1 Enable output .	R/W	0
20	FS5	Frequency Generator 5 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0
19:12	FRDIV4	Divider for Frequency Generator 4. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
11	FE4	Frequency Generator 4 Output Enable. 0 Disable output. 1 Enable output .	R/W	0
10	FS4	Frequency Generator 4 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0
9:2	FRDIV3	Divider for Frequency Generator 3. The frequency divider is $(FRDIV + 1) * 2$ , where FRDIV is the value programmed in this field.	R/W	0
1	FE3	Frequency Generator 3 Output Enable. 0 Disable output. 1 Enable output .	R/W	0
0	FS3	Frequency Generator 3 Source. 0 CPU Core clock. 1 Auxiliary clock.	R/W	0

7.1.2.3 Clock Source Control

This register controls the clock source for all output clocks. This register will reset to defaults only on a hardware reset. During a runtime reset and during Sleep this register will retain its value.

sys\_clksrc

Offset = 0x0028

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ME1[2:0]	DE1	CG3	ME0[2:0]	DE0	CE0	MI2[2:0]	DI2	CI2									ML[2:0]	DL	CL	MIR[2:0]	DIR	CIR				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:30	—	Reserved, should be cleared.	R	0
29:27	ME1	EXTCLK1 Clock Mux input select. See Table 7-3 on page 201.	R/W	000
26	DE1	EXTCLK1 Clock Divider Select. 0 Divide by 4. 1 Divide by 2.	R/W	0
25	CE1	EXTCLK1 Clock Select. 0 Clock is taken directly from mux. (The divider select bit DE1 has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0
24:22	ME0	EXTCLK0 Clock Mux input select. See Table 7-3 on page 201.	R/W	000
21	DE0	EXTCLK0 Clock Divider Select. 0 Divide by 4. 1 Divide by 2.	R/W	0
20	CE0	EXTCLK0 Clock Select. 0 Clock is taken directly from mux. (The divider select bit DE0 has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0
19:17	MI2	I <sup>2</sup> S Clock Mux input select. See Table 7-3 on page 201.	R/W	000
16	DI2	I <sup>2</sup> S Clock Divider Select. 0 Divide by 4 1 Divide by 2	R/W	0
15	CI2	I <sup>2</sup> S Clock Select. 0 Clock is taken directly from mux. (The divider select bit DI2 has no effect.) 1 clock is taken from 2/4 divider	R/W	0
14:10	—	Reserved		
9:7	ML	LCD Controller Clock Mux input select. See Table 7-3 on page 201.	R/W	000
6	DL	LCD Controller Clock Divider Select. 0 Divide by 4. 1 Divide by 2.	R/W	0
5	CL	LCD Controller Clock Select. 0 Clock is taken directly from mux. (The divider select bit DL has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0
4:2	MIR	IrDA/ USB Host/ USB Device Clock Mux input select. See Table 7-3 on page 201.	R/W	000
1	DIR	IrDA/ USB Host/ USB Device Clock Divider Select. 0 Divide by 4. 1 Divide by 2.	R/W	0
0	CIR	Clock Select for IrDA/ USB Host/ USB Device Clock. 0 Clock is taken directly from mux. (The divider select bit DIR has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0



The specific values written to the 3-bit clock-mux-input-select fields are shown in Table 7-3. The  $FREQ_n$  selections come from the output of the corresponding frequency generators, as shown in Figure 7-2 on page 196.

**Table 7-3. Clock Mux Input Select Values**

Value	Meaning
000	No clocking
001	Auxiliary Clock
010	FREQ0
011	FREQ1
100	FREQ2
101	FREQ3
110	FREQ4
111	FREQ5

### 7.1.3 PLL Control

There are two registers for controlling the two PLLs integrated into the Au1100 processor. Each PLL is independently programmable. Note that when programming the PLL control registers, the system designer must not violate the rated frequency limits of the Au1100 processor. Configuring the PLLs outside this frequency range causes undefined behavior.

The Core Cycle Counter register located at CP0 register 9 can be used to count core cycles. Please see Section 2.7 "Coprocessor 0" on page 28, for more information.

The two PLLs in the Au1100 processor drive the CPU clock and the auxiliary clock. The default PLL multiplier value is 16 for the CPU clock and 0 for the AUXPLL which has the following implications assuming a 12 MHz crystal on XTI12 and XTO12:

- CPU Clock = 192 MHz
- Auxiliary Clock = Disabled
- SBUS Clock = 96 MHz (SD - SBUS divider - defaults to 2)
- Peripheral Bus = 48 MHz
- SDRAM Bus = 48 MHz.

When modifying the CPU clock frequency approximately 20  $\mu$ s elapse while the CPU and bus clocks shut off and the CPU PLL locks to the new frequency. During this period instructions are not executed and interrupts are not serviced. Interrupts are serviced once execution begins again at the new frequency.

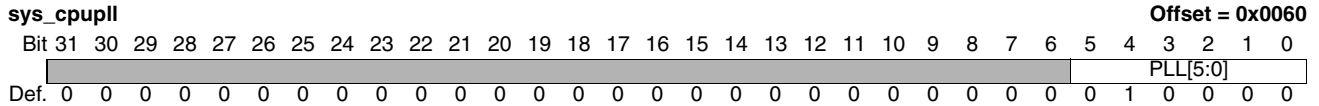
**7.1.3.1 CPU PLL Control**

The CPU PLL control register (**sys\_cpupll**) resets to its default value only for a hardware reset. That is, after Sleep, and during a runtime reset the CPU PLL retains its frequency.

Note that when programming the CPU PLL control register the system designer must not violate the rated frequency limits of the Au1100 processor. Configuring the PLL outside this frequency range causes undefined behavior.

This register is read/write, but the value read is valid only after initialization. After coming out of reset, hardware reset or Sleep, this register must first be written for the value read back to be valid. For this reason it is suggested that this register be initialized at boot time regardless if the value is changed from default.

After writing to the **sys\_cpupll** register, the system automatically halts for 20  $\mu$ s to allow for the PLL to relock and clocks to become stable.



Bits	Name	Description	R/W	Default
31:6	—	Reserved, should be cleared.	R/W	0
5:0	PLL	CPU PLL Multiplier. Determines the integer multiplier used to multiply the CPU PLL to generate the CPU clock. For example, with the default of 16 and a 12 MHz OSC frequency, the CPU clock frequency is 192 MHz. Note that PLL multiplier values that place the clock frequency outside of rated limits are invalid. 0–15: Reserved and undefined 16–(n-1): Valid PLL multiplier n–63: Reserved and undefined where <i>n</i> is the smallest PLL multiplier that would cause the CPU clock frequency to exceed the rated frequency limits of the part.	R/W	0x10

### 7.1.3.2 Auxiliary PLL Control

The auxiliary PLL control register (**sys\_auxpll**) resets to its default value on hardware reset, after Sleep, and during a runtime reset. This register is read/write, but the value read is valid only after initialization. For this reason it is recommended that system software initialize this register at hardware reset, runtime reset and Sleep, even if programming its default value.

Note that when programming the auxiliary PLL control register the system designer must not violate the rated frequency limits of the Au1100 processor. Configuring the PLL outside this frequency range causes undefined behavior.

Unlike the **sys\_cpupll** register, writing **sys\_auxpll** does not cause the system to halt. As a consequence, clocks taken from the AUX PLL may be unstable for up to 20  $\mu$ s. To ensure stable clocks during AUX PLL lock time, the **sys\_cpupll** register can be written with its current value to force the system to halt for 20  $\mu$ s.

<b>sys_auxpll</b>																<b>Offset = 0x0064</b>																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PLL[5:0]																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:6	—	Reserved, should be cleared.	R/W	0
5:0	PLL	<p>Auxiliary PLL Multiplier. Determines the integer multiplier used to multiply the auxiliary PLL to generate the auxiliary clock.</p> <p>For example, with a value of 12 and a 12-MHz OSC frequency, the auxiliary clock frequency will be 144 MHz.</p> <p>Note that PLL multiplier values that place the clock frequency outside of rated limits are invalid.</p> <p>0: Disable the auxiliary PLL.</p> <p>1–7: Reserved and undefined</p> <p>8–(<i>n</i>-1): Valid PLL multiplier</p> <p><i>n</i>–63: Reserved and undefined</p> <p>where <i>n</i> is the smallest PLL multiplier that would cause the auxiliary clock frequency to exceed the rated frequency limits of the part.</p>	R/W	0x00

### 7.1.4 Hardware Considerations

Note also that the EXTCLK[1:0] clocks are multiplexed signals and require programming of the **sys\_pinctrl** register (see Section 7.3.1.1 "Pin Function" on page 209) as follows:

When using the external clocks from the clock generation block, the **sys\_pinctrl** register must be programmed such that GPIO[2] and/or GPIO[3] are configured to be driven by EXTCLK0 and/or EXTCLK1.

Section 11.8 "Crystal Specifications" on page 283, define the crystal specifications.

### 7.1.5 Programming Considerations

When changing the CPU PLL value through the **sys\_cpupll** register, the system automatically halts for 20  $\mu$ s to allow clocks to stabilize. During this time no interrupts are serviced, potentially affecting real-time systems. However, modifying the **sys\_auxpll** register does *not* cause the system to halt, and therefore clocks taken from the AUX PLL may be unstable for up to 20  $\mu$ s. To ensure stable clocks while the AUX PLL locks, the **sys\_cpupll** register can be written with its current value to force the system to halt for 20  $\mu$ s.

## 7.2 Time of Year Clock and Real Time Clock

The Au1100 processor contains two programmable counters designed for use as a time of year clock (TOY) and real time clock (RTC). Because the TOY continues counting through Sleep, a TOY counter match can be used as a wake-up source. The RTC, however, will power-down in Sleep mode.

Note that both the TOY and RTC counters are driven by the 32.768-kHz clock input. The clock input source can be a crystal or external clock. (See Section 11.8 "Crystal Specifications" on page 283 for crystal details.)

Each programmable counter employs a register to initialize the counter or load a new value, a trim divider to adjust the incoming 32.768-kHz clock, and three match registers which have associated interrupts that trigger on a match. Each counter is also able to generate an interrupt on every tick. All interrupts are maintained through the interrupt controller. Both programmable counters share a status register.

Figure 7-4 shows the functional block diagram of both the TOY and the RTC. The registers used to implement the block, including the counter control register (**sys\_cntrctrl**), are described in the following section.

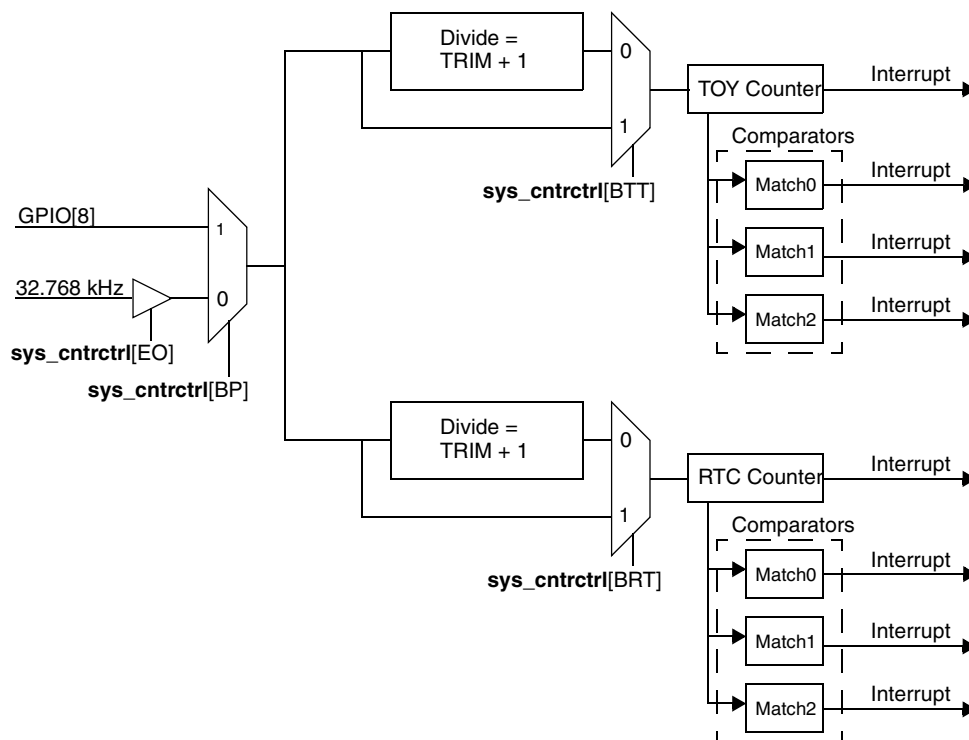


Figure 7-4. TOY and RTC Block Diagram

### 7.2.1 Time of Year Clock and Real Time Clock Registers

Each counter operates identically with the only difference being that the TOY continues counting through Sleep and the RTC does not.

The programmable counter control registers and their associated offsets are listed in Table 7-4. When functionality is identical for registers in the different programmable counters, only one general register description is presented with offsets pointing to the specific registers.

**Table 7-4. Programmable Counter Registers**

Offset (Note 1)	Register Name	Description	Reset Type
0x0000	sys_toytrim	Trim value for 32.768-kHz clock source for TOY	Hardware
0x0004	sys_toywrite	TOY counter value is written through this register	Hardware
0x0008	sys_toymatch0	TOY match 0 value for interrupt generation	Hardware
0x000C	sys_toymatch1	TOY match 1 value for interrupt generation	Hardware
0x0010	sys_toymatch2	TOY match 2 value for interrupt generation	Hardware
0x0014	sys_cntrctrl	Control register for TOY and RTC	Hardware
0x0040	sys_toyread	TOY counter value is read from this register	Hardware
0x0044	sys_rtctrim	Trim value for 32.768-kHz clock source for RTC	Hardware
0x0048	sys_rtcwrite	RTC counter value is written through this register	Hardware
0x004C	sys_rtcmatch0	RTC match 0 value for interrupt generation	Hardware
0x0050	sys_rtcmatch1	RTC match 1 value for interrupt generation	Hardware
0x0054	sys_rtcmatch2	RTC match 2 value for interrupt generation	Hardware
0x0058	sys_rtcread	RTC counter value is read from this register	Hardware

Note 1. See Table 7-1 on page 193 for base address.

#### 7.2.1.1 Trim Register

The TOY trim write status bit (**sys\_cntrctrl**[TTS]) must be clear before writing **sys\_toytrim**. It is set upon writing this register and is cleared by hardware when the write takes effect.

The RTC trim write status bit (**sys\_cntrctrl**[RTS]) must be clear before writing **sys\_rtctrim**. It is set upon writing this register and is cleared by hardware when the write takes effect.

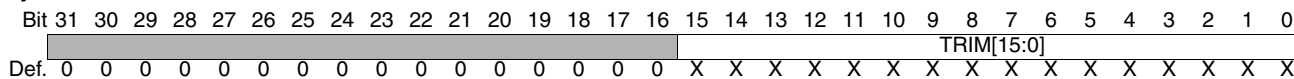
This register is unpredictable at power-on. During a runtime reset and during Sleep this register retains its value.

**sys\_toytrim - TOY Trim**

**Offset = 0x0000**

**sys\_rtctrim - RTC Trim**

**Offset = 0x0044**



Bits	Name	Description	R/W	Default
31:16	—	Reserved, should be cleared.	R	0
15:0	TRIM	Divide value for 32.768kHz input. Divide = TRIM + 1	R/W	UNPRED

### 7.2.1.2 Counter Write

The TOY value write status bit (**sys\_cntrctrl**[TS]) must be clear before writing **sys\_toywrite**. It is set upon writing this register and is cleared by hardware when the write takes effect.

The RTC value write status bit (**sys\_cntrctrl**[RS]) must be clear before writing **sys\_rtcwrite**. It is set upon writing this register and is cleared by hardware when the write takes effect.

This register is unpredictable at power-on. During a runtime reset and during Sleep this register retains its value.

**sys\_toywrite** - TOY counter value write

Offset = 0x0004

**sys\_rtcwrite** - RTC counter value write

Offset = 0x0048

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	COUNT[31:0]																																	
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		

Bits	Name	Description	R/W	Default
31:0	COUNT	Counter Write. The respective counter will be updated with the value written to this register at the next trimmed clock.	W	UNPRED

### 7.2.1.3 Match Registers

The corresponding write status bit (**sys\_cntrctrl**[TM $n$ ] or **sys\_cntrctrl**[RM $n$ ]) must be clear before writing the below registers. It is set upon writing the register and is cleared by hardware when the write takes effect.

Each match register is capable of causing an interrupt as shown in Section 5.0 "Interrupt Controller" on page 83. The **sys\_toymatch2** can be used to wake up from Sleep; see Section 7.4.4.2 "Wakeup Source Mask Register" on page 219. See also Section 7.2.2 "Programming Considerations" on page 208.

These registers are unpredictable at power-on. During a runtime reset and during Sleep these registers retain their value.

**sys\_toymatch0** - TOY Match 0

Offset = 0x0008

**sys\_toymatch1** - TOY Match 1

Offset = 0x000C

**sys\_toymatch2** - TOY Match 2

Offset = 0x0010

**sys\_rtcmatch0** - RTC Match 0

Offset = 0x004C

**sys\_rtcmatch1** - RTC Match 1

Offset = 0x0050

**sys\_rtcmatch2** - RTC Match 2

Offset = 0x0054

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	MATCH[31:0]																																	
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		

Bits	Name	Description	Read/Write	Default
31:0	MATCH	A match with the counter and the value in this register causes an interrupt.	R/W	UNPRED

### 7.2.1.4 TOY and RTC Counter Control

The TOY and RTC counter control register (**sys\_cntrctrl**) contains control bits and status bits to configure and control both programmable counters.

Write Status Bits: These bits indicate the status of the latest update to the respective register/field. When the corresponding register/field is written, this bit is set indicating that there is a write pending. When this bit is cleared the write has taken place. Software should poll the correct bit and insure that it is 0 before updating the respective register/field.

This register resets to default values only on a hardware reset. During a runtime reset and during Sleep this register retains its value.

**sys\_cntrctrl**

Offset = 0x0014

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
													RTS	RM2	RM1	RM0	RS		BP		BRT		BTT		EO	CCS		32S	TT	TM2	TM1	TM0	TS
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	

Bits	Name	Description	R/W	Default
31:21	—	Reserved, should be cleared.	R	0
20	RTS	<b>sys_rtctrim</b> write status	R	0
19	RM2	<b>sys_rtcmatch2</b> write status	R	0
18	RM1	<b>sys_rtcmatch1</b> write status	R	0
17	RM0	<b>sys_rtcmatch0</b> write status	R	0
16	RS	<b>sys_rtcwrite</b> write status 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	R
15	—	Reserved, should be cleared.	R	0
14	BP	Bypass the 32.768 kHz OSC. 0 Select Oscillator Input (XTI32, XTO32). 1 GPIO[8] drives the counters. This is a test mode where GPIO[8] can drive the counters from an external source or through software using the GPIO controller.	R/W	0
13	—	Reserved, should be cleared.	R/W	0
12	BRT	Bypass RTC Trim. 0 Normal operation. 1 The RTC is driven directly by the 32.768 kHz clock, bypassing the trim.	R/W	0
11	—	Reserved, should be cleared.	R/W	0
10	BTT	Bypass TOY Trim. 0 Normal operation. 1 The TOY is driven directly by the 32.768 kHz clock, bypassing the trim.	R/W	0
9	—	Reserved, should be cleared.	R	0
8	EO	Enable 32.768-kHz Oscillator. Enables the clock for the RTC/TOY block. 0 Disable the clock. 1 Enable the clock.  Regardless of the clock source (crystal or overdriven clock through XTI32/XTO32, or bypass through GPIO[8]), the EO bit must be set to enable the RTC/TOY counters. After enabling the clock by setting EO, poll the oscillator status bit (32S) until it returns a '1'. Once 32S is set, wait an additional one second to allow for frequency stabilization within the block before accessing other RTC/TOY registers (not including <b>sys_cntrctrl</b> ).  Note: If the oscillator is being overdriven or bypassed through GPIO[8], be sure to set EO only <i>after</i> a stable clock is being driven into the part.	R/W	0
7	CCS	<b>sys_cntrcntrl</b> write status.	R	0
6	—	Reserved, should be cleared.	R	0
5	32S	32.768-kHz Oscillator Status. Detects two consecutive 32 kHz cycles from the clock source for the RTC/TOY block. 0 Clock is not running. 1 Clock is running.  Note: Be sure to wait 1 second after 32S is set to allow for frequency stabilization within the block before accessing RTC/TOY registers.	R	UNPRED

Bits	Name	Description	R/W	Default
4	TTS	<b>sys_toytrim</b> write status	R	0
3	TM2	<b>sys_toymatch2</b> write status	R	0
2	TM1	<b>sys_toymatch1</b> write status	R	0
1	TM0	<b>sys_toymatch0</b> write status	R	0
0	TS	<b>sys_toywrite</b> write status 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0

### 7.2.2 Programming Considerations

To change the values of the counter and match registers, software must poll the state of the corresponding status bit in **sys\_cntrctrl**. When the corresponding write status bit (**sys\_cntrctrl**[TTS, TM $n$ , TS] or **sys\_cntrctrl**[RTS, RM $n$ , RS]) is 0 it is okay to write a new value. Once the new value is written to the register the status bit will change to a 1. When the write status bit is 1 the new value is being updated in supporting hardware. When the write status changes to a 0 then the new value is active in the device.



### 7.3 Primary General Purpose I/O and Pin Functionality

The Au1100 processor contains two separate GPIO blocks (primary and secondary). This section covers the programming model for the primary general purpose I/O (GPIO) signals. The Au1100 processor supports 48 GPIOs, 32 of which are controlled by the primary GPIO block. For a description of the programming model for the secondary GPIO block see Section 6.11 "Secondary General Purpose I/O" on page 189.

This section also documents how to change the functionality of multiplexed pins. These pins can function at the system level as a GPIO signal, or they can be assigned a signal function dedicated to an integrated peripheral device. Note that this includes some GPIO2 signals that share pins with internal peripherals.

Each GPIO can be configured as either an input or an output. The GPIO ports also can be connected to the internal interrupt controllers to generate an interrupt from input signals. See Section 5.0 "Interrupt Controller" on page 83 for information on interrupts.

#### 7.3.1 Pin Functionality

To maximize the functionality of the Au1100 processor, many of the pins have multiple uses. Note that if a pin is programmed for a certain use, any other functionality associated with that pin can not be utilized at the same time. In other words, a pin can not be used as a GPIO at the same time it is assigned to a peripheral device.

(For reference, Figure 10-1 on page 244 shows a block diagram of all external signals. Signals that are multiplexed on one pin will show the shared function in parentheses.)

##### 7.3.1.1 Pin Function

This register resets to its default state at hardware reset, runtime reset and Sleep.

sys\_pinfunc

Offset = 0x002C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														PC	LCD	CS	USB	U3		U1	SRC	EX1	EX0	RF	UR3	I2D	I2S	NI	U0	RD	A97	SO
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	0	1

Bits	Name	Description	R/W	Default
31:19	—	Reserved, should be cleared.	R	all 0s
18	PC	PCMCIA/GPIO[207:204]. 0 PREG#, PCE1#, PCE2# and PWE# will drive pins. 1 GPIO[207:204] will drive pins.	R/W	1
17	LCD	External LCD Controller/GPIO[203:200]. 0 LRD, LRD1, LWR0 and LWR1 will drive pins. 1 GPIO[203:200] will drive pins.	R/W	1
16	CS	Clock Select. Applies only when EX0 = 1. 0 EXTCLK0 will drive pin. 1 32-kHz OSC clock will drive pin.	R/W	0
15	USB	USB Functionality. 0 USBDP and USBDM will drive pins (pins are connected to USB device module). 1 USBH0P and USBH0M will drive pins (pins are connected to USB host port 0).	R/W	0
14	U3	UART3/GPIO[214]. 0 U3TXD drives pin. 1 Pin is configured for GPIO[214].	R/W	1
13	—	This bit is reserved and should be written as 1.	R/W	1
12	U1	UART1/GPIO[213]. 0 U1TXD drives pin. 1 Pin is configured for GPIO[213].	R/W	1
11	SRC	GPIO[6]/SMROMCKE. 0 Pin is configured for GPIO[6]. 1 SMROMCKE drives pin.	R/W	0

Bits	Name	Description	R/W	Default
10	EX1	GPIO[3]/EXTCLK1. 0 Pin is configured for GPIO[3]. 1 EXTCLK1 will drive pin.	R/W	0
9	EX0	GPIO[2] / (EXTCLK0 or 32kHz OSC). 0 Pin is configured for GPIO[2]. 1 Pin is configured for EXTCLK0 or 32kHz OSC. CS (bit 16) selects whether EXTCLK0 or the 32-kHz OSC drives the pin.	R/W	0
8	IRF	GPIO[15]/IRFIRSEL. 0 Pin is configured for GPIO[15]. 1 IRFIRSEL will drive pin.	R/W	0
7	UR3	GPIO[14:9]/UART3. 0 Pins are configured as GPIO[14:9]. 1 Pins are configured for optional UART3 flow control. U3DTR#, U3RTS#, U3RI#, U3DCD#, U3DSR#, and U3CTS# will drive pins. <b>System Note:</b> For systems that use the UART3 interface but do <i>not</i> use the optional modem control signals (UR3=0), the modem status interrupts must be disabled ( <code>uart3_inten[MIE]=0</code> ) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	R/W	0
6	I2D	GPIO[8]/I2SDI. 0 Pin is configured for GPIO[8]. 1 Pin is configured as I2SDI. <b>System Note:</b> For systems that use the I <sup>2</sup> S interface for <i>unidirectional</i> operation (I2SDI not used), the GPIO[8] function is available but with the following restrictions: -- When I2SDIO is configured as an <i>input</i> , GPIO[8] can be used only as an output. -- When I2SDIO is configured as an <i>output</i> , the I <sup>2</sup> S receive function must be disabled if GPIO[8] is to be used as an input.	R/W	0
5	I2S	I2S/GPIO[31:29]. 0 Pins are configured for I <sup>2</sup> S mode. I2SWORD, I2SCLK, I2SDIO will drive pins. 1 Pins are configured as GPIO[31:29].	R/W	1
4	NI	MAC0/GPIO. 0 Pins are configured as Ethernet port 0. N0TXD[3:0], N0TXEN and N0MDC will drive port. 1 Pins are configured as GPIO[28:24] and GPIO[215].	R/W	1
3	U0	UART0/GPIO[212]. 0 Pin is configured for U0TXD (necessary for UART0 operation). 1 Pin is configured as GPIO[212].	R/W	1
2	IRD	IrDA/GPIO[211]. 0 Pin is configured for IRTXD (necessary for IrDA operation). 1 Pin is configured as GPIO[211].	R/W	1
1	A97	AC97/SSI_1. 0 Pins are configured for AC97 mode. ACSYNC, ACBCLK, ACDO, ACRST# will drive pins. 1 Pins are configured for SSI_1 mode. S1DOUT, S1DIN, S1CLK and S1DEN will drive pins.	R/W	0
0	S0	SSI_0/GPIO[210:208]. 0 Pins are configured for SSI_0 mode. S0CLK, S0DOUT and S0DEN will drive pins. 1 Pins are configured as GPIO[210:208].	R/W	1

### 7.3.2 Primary GPIO Control Registers

The primary GPIOs on the Au1100 processor have been designed to simplify the GPIO control process by removing the need for a semaphore to control access to the registers. This is because there is no need to read, modify, write, as there are separate registers for setting and clearing a bit. In this way a function can freely manipulate its associated GPIOs without interfering with other functions.

Figure 7-5 shows the logical implementation of each GPIO. The names represent bit *n* of the corresponding register which affect GPIO[n].

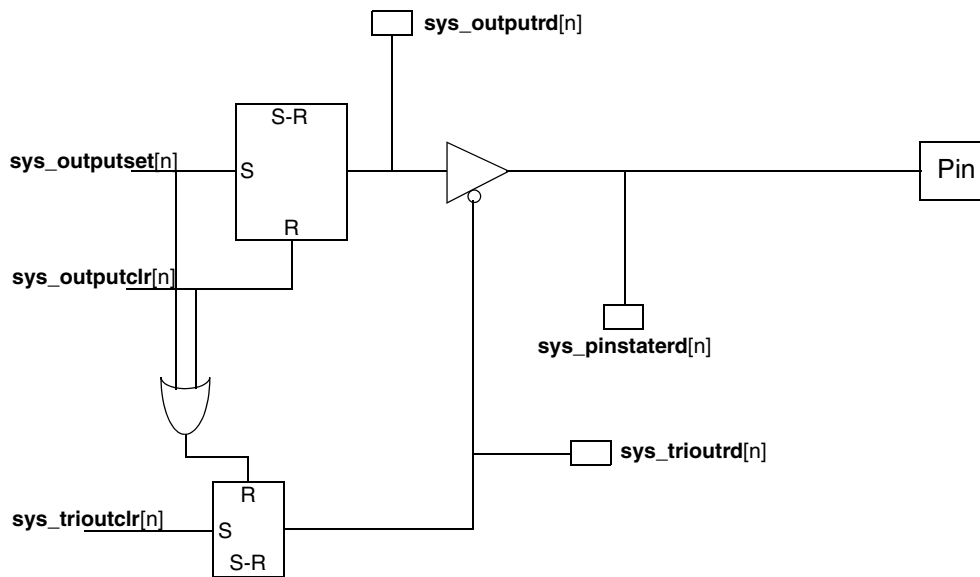


Figure 7-5. GPIO Logic Diagram

The following table shows the GPIO control registers and the associated offsets from **sys\_base**. Certain registers share offsets and have different functionality depending on whether the access is a read or a write. The register descriptions detail the functionality of each register. Bit *n* of a particular register should be associated with GPIO[n] for all registers except **sys\_pininputen**.

Table 7-5. GPIO Control Registers

Offset (Note 1)	Register Name	Register Description	Default
0x0100	sys_trioletrd	The TRI-STATE/Output state register shows the current state of the GPIO. 0 GPIO[n] is TRI-STATED. TRI-STATING GPIO[n] is accomplished by setting the corresponding bit in the <b>sys_trioletclr</b> register. 1 Output is enabled. Enabling GPIO[n] as an output is accomplished by programming GPIO[n] as a 0 or 1 using the <b>sys_outputclr[n]</b> or <b>sys_outputset[n]</b> registers. If the pin is not an output, it should be in TRI-STATE.	0x00000000 (all GPIOs are in TRI-STATE)
0x0100	sys_trioletclr		
0x0108	sys_outputrd	Controls the state of the GPIO[n] as an output. 0 To output a low level, set <b>sys_outputclr[n]</b> . 1 To output a high level, set <b>sys_outputset[n]</b> . Programming a bit value in the output register brings the pin out of tristate mode and enables the output.	UNPRED
0x0108	sys_outputset		
0x010C	sys_outputclr		

**Table 7-5. GPIO Control Registers (Continued)**

Offset (Note 1)	Register Name	Register Description	Default
0x0110	sys_pinstaterd	Allows the pin state to be read when an input. This register will also give the output state.	UNPRED
0x0110	sys_pinputen	Any write to this register allows GPIO[31:0] to be used as inputs. This register must be written before any GPIO can be used as an input, an interrupt source, or for use as a wake up source.	UNPRED

Note 1. See Table 7-1 on page 193 for base address.

### 7.3.2.1 GPIO Control Registers

Each GPIO control register is 32 bits wide with bit *n* in each register affecting GPIO[*n*].

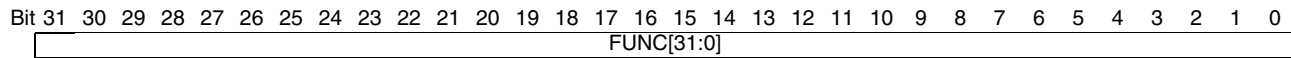
These registers will reset to defaults only on a hardware reset. During a runtime reset and during Sleep this register will retain its value.

See Table 7-5 for the default values at hardware reset.

\*rd

\*set

\*clr



Bits	Name	Description	R/W	Default
31:0	FUNC[ <i>n</i> ]	The function of each register is given in the previous table. FUNC[ <i>n</i> ] controls the functionality of GPIO[ <i>n</i> ].	*_read - read only *_set - write only *_clear - write only See the following text.	0

Certain registers in the list have the same offset but offer different functionality depending on whether a read or a write is being performed.

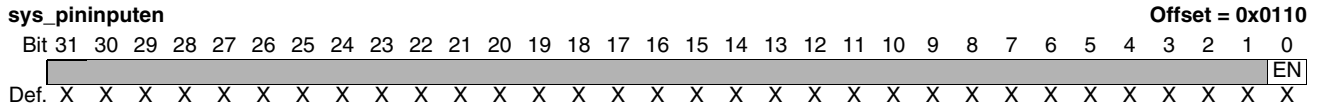
Registers ending in \*rd, \*set and \*clr have the following functionality:

- \*rd registers are read only registers will read back the current value of the register.
- \*set registers are write only registers and will set to 1 all bits that are written 1. Writing a value of 0 will have no impact on the corresponding bit.
- \*clr registers are write only registers and will clear to zero all bits that are written 1. Writing a value of 0 will have no impact on the corresponding bit.

### 7.3.2.2 GPIO Input Enable

The **sys\_pininputen** is a 32-bit, write-only register. When this register is written, the input functionality of all GPIOs is enabled. This register enables GPIOs for use as an input but does not explicitly configure all GPIOs as inputs. The value of the GPIO control registers and the pin function register will define the state of each GPIO.

GPIOs cannot be used as inputs until this register is written. This write is required only once per hardware reset (i.e., Sleep and a runtime reset will not require another write to this register).



Bits	Name	Description	R/W	Default
31:1	—	Reserved, should be cleared.	W	N/A
0	EN	A write to this bit (0 or 1) enables all GPIOs to be used as inputs.	W	N/A

### 7.3.3 Hardware Considerations

The system pin function register (**sys\_pinfunc**) controls the functionality of many GPIO/peripheral pins. If a pin is programmed for a certain functionality, all other functionality associated with that pin is disabled.

For example, if **sys\_pinfunc**[U3] is cleared configuring the pin as U3TXD, GPIO[214] can not be used as a GPIO nor can the GPIO be configured as an interrupt. Conversely if **sys\_pinfunc**[U3] is set configuring the pin as GPIO[214], U3TXD (and thus the UART3 interface) is not usable. GPIO[214] can be used as a GPIO and to generate interrupts.

### 7.3.4 Using GPIO for External DMA Requests

See Section 4.2 "Using GPIO as External DMA Requests (DMA\_REQn)" on page 80 for information.

## 7.4 Power Management

The Au1100 processor contains a robust power management scheme allowing multiple levels of power conservation to enable the system designer options depending on whether power conservation or system responsiveness is more critical.

In the Au1100 processor, power management can be broken into three different areas:

- CPU
- Peripherals
- Device

The lowest power state consists of putting the entire device into a Sleep state. The CPU also supports two Idle states that differ as to whether bus snooping is supported. In addition each peripheral can have its clocks disabled when not in use thus significantly reducing the power draw by those blocks not in use.

The flow chart in Figure 7-6 shows the different stages of power management for the CPU (IDLE0,1) and the device (SLEEP) and how each state is entered and left. Note that any interrupt can be used to bring the CPU out of either Idle state while only a GPIO[7:0] or `sys_toymatch2` interrupt can be enabled (in `sys_wakemsk`) to bring the device out of Sleep.

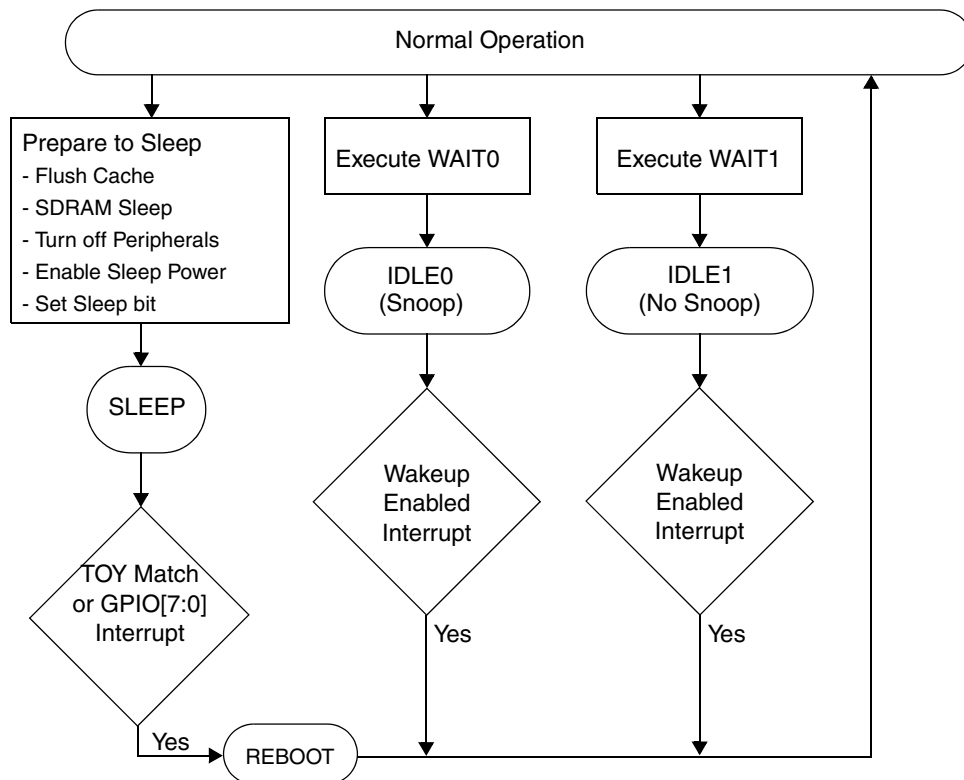


Figure 7-6. Sleep and Idle Flow Diagram

### 7.4.1 CPU Power Management - Idle

The CPU can be put into two different low-power Idle modes (IDLE0 and IDLE1) by using the wait instruction:

- 1) In the IDLE0 state the CPU snoops the bus and cache coherency is maintained.
- 2) In the IDLE1 state the CPU does not snoop the bus and cache coherency is lost.

The wait instruction and at least four instructions following it must be in the cache for the wait to occur. See Section 2.6.3 "WAIT Instruction" on page 27 for more information.

At all times the MMU, data cache, execution and multiply-and-accumulate blocks are placed in a low power state if they are not being used.

#### 7.4.1.1 Returning from Idle

The processor wakes from the Idle state (IDLE0 or IDLE1) upon receiving an interrupt. The time required for the processor core to return to normal execution is as follows:

- Five to ten CPU clocks are needed to restart clocks to the CPU.
- It takes an additional ten CPU clocks for the core to recognize the interrupt and begin fetching the interrupt service routine.

Therefore, a maximum of 20 CPU clocks are required to resume normal instruction pipeline execution. If the interrupt service routine is in the instruction cache, the instruction returns immediately; otherwise, there is an additional delay while fetching the instruction from memory.

### 7.4.2 Peripheral Power Management

Peripheral power management is handled through clock management and disabling of unused peripherals. Table 7-6 lists the peripherals and their related power management registers. The actual register descriptions should be referred to for programming details.

Note that when separate reset/peripheral enable and clock-enable bits are provided, the reset must be applied first, and then the clocks should be disabled. This will simplify programming, as the suggested bring up sequence is typically to first enable clocks and then subsequently to bring the peripheral out of reset.

**Table 7-6. Peripheral Power Management**

Peripheral	Power Management Register	Power Management Strategy
USB Host	usbh_enable	When the USB host is not in use the E bit can be cleared to disable the host. The CE bit should also be cleared to disable clocks to the block.
USB Device	usbd_enable	When the USB device is not in use the E bit can be cleared to disable the host. The CE bit should also be cleared to disable clocks to the block.
Ethernet MAC	macen_mac0	When this block is not being used, the E[2:0] bits should be cleared to disable the MAC, and the CE bit should be cleared to gate clocks to the MAC.
UART <sub>n</sub>	uartn_enable	When a UART is not being used, the E bit should be cleared to hold the part in reset and the CE bit should be cleared to disable clocks to the block.
SSIn	ssin_enable	When an SSI is not being used, clear the E bit to hold the block in reset, and set the CD bit to disable clocks to the block.
IrDA	ir_enable	The HC bit can be used to run the IrDA at half the SBUS. The CE bit should be disabled when not using the IrDA to disable clocks to this peripheral.

Table 7-6. Peripheral Power Management (Continued)

Peripheral	Power Management Register	Power Management Strategy
Primary General Purpose I/O (GPIO) Controller	sys_trioutclr	Although there is not a specific low-power configuration for the primary GPIOs, tristating the unused GPIOs minimizes their power usage.
Secondary General Purpose I/O (GPIO2) Controller	gpio2_enable	If no GPIO2 signals are being used, the GPIO2 module reset (MR) bit should be set to place the module in reset. Also, clear the CE bit to disable clocks to the block. (By default, the GPIO2 module is disabled coming out of reset.)
Programmable Counters (TOY and RTC)	sys_cntrctrl	If both the TOY and RTC are not being used, then disable the oscillator.
AC97 Controller	ac97_enable	If the AC97 block is not in use, the D bit should be used to disable the module and the CE bit should be disabled to gate clocks from the block.
I <sup>2</sup> S	i2s_enable	If the I <sup>2</sup> S block is not in use, the E bit should be used to place the block in reset and the CE bit should be disabled to gate clocks from the block.
LCD Controller	lcd_control	To disable the LCD controller, clear the GO bit in the LCD control register. Also, the LCD controller source clock (a dedicated frequency generator, for example) should be disabled if possible; see Section 7.1 "Clocks" on page 194 for more information.
SD <sub>n</sub> Controller	sdn_enable	If the SD block is not in use, clear the R bit to place the block in reset, and clear the CE bit to disable the clock.

### 7.4.3 Device Power Management - Sleep

The Sleep state of the Au1100 processor puts the entire device into a low-power state. Sleep is the lowest power state of the part and requires a complete system initialization on wakeup. There are multiple steps to take when going into Sleep and waking up to insure data integrity. During this state all registers values outside the system control block are lost and cache coherency is not maintained.

The programmable counter 0 (intended for TOY) continues clocking and remains functional during Sleep. However, the programmable counter 1, as well as other clocks throughout the Au1100, are disabled during Sleep.

When coming out of Sleep there is a programmable delay defined by **sys\_powerctrl**[VPUT]. This is the time that the system designer has to ensure  $V_{DD1}$  is stable from the rising edge of PWR\_EN.

To enter Sleep the following steps should be taken. This code should be run from Flash, or conversely the system programmer should guarantee that this code will run from cache because after SDRAM is put into auto-refresh mode, memory accesses will no longer work.

- 1) Enable Sleep Power by writing to the **sys\_slppwr** register.
- 2) Turn off all peripherals. (Explicitly turning off all peripherals in use ensures a graceful transition to Sleep mode.)
- 3) Push dirty data out of the cache. (During Sleep cached data is lost.)
- 4) If SDRAM contents are to be kept through Sleep, SDRAM should be put into auto-refresh mode. See Section 3.1 "SDRAM Memory Controller" on page 44 for more information. If SDRAM is not needed to be maintained through Sleep, disable the SDRAM.
- 5) If using one of GPIO[7:0] as a wakeup source, **sys\_pinputen** must be written to enable the GPIO as an input if this has not already been done at system startup.
- 6) The **sys\_wakemsk** register should be set with the appropriate value according to what signal(s) should wake the processor.
- 7) The **sys\_wakesrc** register should be written to explicitly clear any pending wake interrupts.



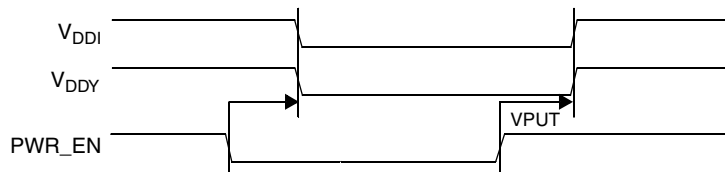
- 8) Enable Sleep by writing to the **sys\_sleep** register. This step puts the system to Sleep.
- 9) As the system enters Sleep mode, the PWR\_EN signal is negated. This can be used to disable  $V_{DDI}$  and  $V_{DDY}$  if needed. Note that on initial power-on, PWR\_EN asserts as soon as VDDXOK asserts. The system should assert VDDXOK only after *both*  $V_{DDX}$  and  $V_{DDY}$  have ramped up. For Sleep wakeup, however, PWR\_EN can be used to ramp  $V_{DDY}$ . Thus, the logic equation for the system's  $V_{DDY}$  power-enable signal needs to be  $PWR\_EN\_for\_V_{DDY} = PWR\_EN \text{ OR (initial power-on ramp)}$ .

When the processor takes a Sleep interrupt to wake up, the following steps should be taken:

- 1) After the Sleep interrupt is taken, the PWR\_EN signal is asserted by hardware. Within the time indicated by **sys\_powerctrl**[VPUT], the system must ensure that  $V_{DDI}$  is stable. If  $V_{DDY}$  has been disabled during Sleep it must also be stable within this time.
- 2) The processor will then boot from physical address 0x1FC0 0000 as normal.
- 3) If Sleep is to be used by the system and a different flow should be followed when coming out of Sleep the **sys\_wakesrc** should be read to determine if the processor is coming out of Sleep and what caused the wakeup. The system should then write the **sys\_wakesrc** register to clear this information.
- 4) The processor will need to perform complete system initialization. All registers except those described as otherwise in the System Control Block will be at their default values.

#### 7.4.3.1 Sleep Sequence and Timing

As the processor enters Sleep mode, the system designer has the option of disabling  $V_{DDI}$  and  $V_{DDY}$  to conserve power. The PWR\_EN signal defines the Sleep window. Figure 7-7 shows the Sleep sequence.



**Figure 7-7. Sleep Sequence**

The system designer must ensure  $V_{DDI}$  and  $V_{DDY}$  are stable from the rising edge of PWR\_EN within the time period as programmed in **sys\_powerctrl**[VPUT]. Note that VDDXOK (not shown) remains asserted during the Sleep sequence.

### 7.4.4 Power Management Registers

The power management registers and their associated offsets are listed in Table 7-7. These registers are located off of the base shown in Table 7-1 on page 193.

**Table 7-7. Power Management Registers**

Offset (Note 1)	Register Name	Description	Reset Type
0x0018	sys_scratch0	User-defined register that retains its value through Sleep.	Hardware
0x001C	sys_scratch1	User-defined register that retains its value through Sleep.	Hardware
0x0034	sys_wakemsk	Sets which GPIO or whether TOY match can cause Sleep wakeup.	Hardware
0x0038	sys_endian	Sets Big or Little Endian.	Hardware & Runtime
0x003C	sys_powerctrl	Sets SBUS divider and power-up time.	Mixed - see register description
0x005C	sys_wakesrc	Gives source of Sleep wakeup.	Hardware
0x0078	sys_slppwr	Initiates power state for Sleep mode.	Hardware
0x007C	sys_sleep	Initiates Sleep mode.	Hardware

Note 1. See Table 7-1 on page 193 for base address.

#### 7.4.4.1 Scratch Registers

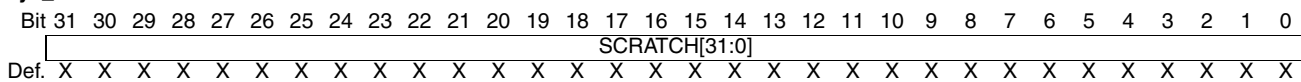
The scratch registers keep their values through Sleep and runtime resets. These registers allow the system programmer to save user-defined state information or a pointer to a context so that the previous context can be restored when coming out of Sleep, if needed. Note that the scratch registers have unpredictable default values after a hardware reset.

**sys\_scratch0**

**Offset = 0x0018**

**sys\_scratch1**

**Offset = 0x001C**



Bits	Name	Description	R/W	Default
31:0	SCRATCH	User-defined information.	R/W	UNPRED

#### 7.4.4.2 Wakeup Source Mask Register

For each individual bit that is set, the corresponding signal or event (for the case of the TOY match) can be used to cause a Sleep wakeup.

A high level on the enabled GPIO will cause the interrupt to trigger.

This register will reset to defaults only on a hardware reset. During a runtime reset and during Sleep this register retains its value.

##### sys\_wakemsk

Offset = 0x0034

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	M2	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:9	—	Reserved, should be cleared.	R	0
8	M2	Setting this bit enables the programmable TOY Counter Match Register 2 ( <b>sys_toymatch2</b> ) to cause a wakeup interrupt. See Section 7.2.1.3 "Match Registers" on page 206.	R/W	0
7:0	GPIO[7:0]	Setting bit <i>n</i> causes GPIO[ <i>n</i> ] to cause a Sleep wakeup.	R/W	0

#### 7.4.4.3 Endianness Register

To change the endianness of the Au1100 processor is a three step process as follows:

- 1) Program the endianness bit in the system endianness register (**sys\_endian**[EN]).
- 2) Read the **sys\_endian** register. (This is required to ensure the final write to the CP0 register will update the endian value.)
- 3) Read the CP0 register **Config0**. (See Section 2.7.15 "Configuration Register 0 (CP0 Register 16, Select 0)" on page 34.)
- 4) Write the value read back into the CP0 **Config0** register. The act of writing the CP0 register will put the processor into the endian state as programmed in **sys\_endian**[EN].

This register as well as the processor endianness will reset to big endian after a hardware reset, runtime reset and after Sleep.

##### sys\_endian

Offset = 0x0038

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	EN	0

Bits	Name	Description	R/W	Default
31:1	—	Reserved, should be cleared.	R	UNPRED
0	EN	Endianness. 0 Big Endian. 1 Little Endian.	R/W	0

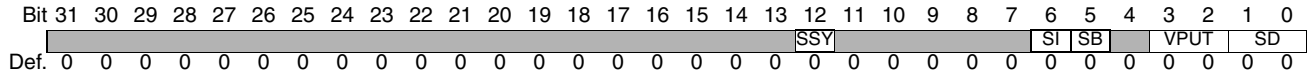
**7.4.4.4 Power Control Register**

Bits[6:5] of this register are reset to default values for a hardware reset, runtime reset and after Sleep.

Bits[12, 4:0] of this register reset to default values only on a hardware reset. During a runtime reset and during Sleep these bits retain their values.

**sys\_powerctrl**

**Offset = 0x003C**



Bits	Name	Description	R/W	Default
31:13	—	Reserved, should be cleared.	R	0
12	SSY	Sleep state for V <sub>DDY</sub> -domain signals. Takes effect only when the part is in Sleep mode. 0 SDRAM interface signals hold their last values if driven (normal Sleep values). 1 Force the SDRAM interface signals to the values they would take during a hardware reset. Note that the system should bring down V <sub>DDY</sub> during Sleep when using this option (SSY=1).	RW	0
11:7	—	Reserved, should be cleared.	R	0
6	SI	Idle State SBUS Clock Divider Enable. 0 The Idle state SBUS clock divider is disabled. 1 Enable the SBUS clock to be divided by an additional factor of 2 when the processor is in an Idle state (taken through the WAIT instruction). All peripheral bus clocks (such as the SDRAM and UART controllers) will be internally compensated with no programmer intervention required. NOTE: SD must be programmed to 00 (divide by two) when SI is set.	R/W	0
5	SB	SBUS Clock Divider Enable. 0 The SBUS clock divider is disabled. 1 Enable the SBUS clock to be divided by an additional factor of 2 when there is no bus activity. All clocks derived from the peripheral bus clock (such as the SDRAM and UART controllers) will be internally compensated with no programmer intervention required. NOTE: SD must be programmed to 00 (divide by two) when SB is set.	R/W	0
4	—	Reserved, should be cleared.	R	0
3:2	VPUT	V <sub>DDI</sub> Power-up Time. 00 20 ms 01 5 ms 10 100 ms 11 2 μs	R/W	Hardware Reset 00
1:0	SD	SBUS Clock Divider. 00 2 01 3 10 4 11 Reserved	R/W	Hardware Reset 00

#### 7.4.4.5 Wakeup Cause Register

Before setting the Sleep bit this register should be cleared. This register will retain pending interrupts according to the setting in the **sys\_wakemsk** register even if those events did not occur during Sleep. In other words if a GPIO's functionality is multiplexed between multiple functions, a high level could cause the associated **sys\_wakesrc** bit to be set even if the action did not occur during Sleep.

The bits in this register must be explicitly cleared as they will hold their values through Sleep and a runtime reset.

All bits in this register are set by hardware and cleared by any write to this register.

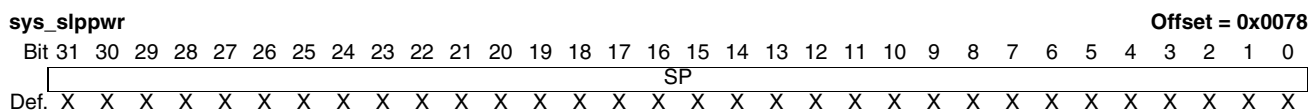
##### sys\_wakesrc

Offset = 0x005C

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
 Def. 0 0 0 0 0 0 0 Rs 0 Rs

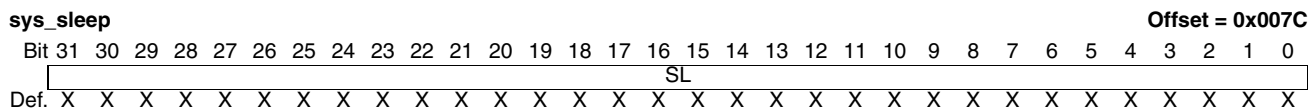
Bits	Name	Description	R/W	Default
31:25	—	Reserved, should be cleared.	R/W	0
24	M2	Programmable TOY Match 2 caused wakeup from Sleep. Set by hardware on Sleep wakeup due to TOY match. Cleared by hardware on VDDXOK assertion.  This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	0
23	GP7	GPIO[n] caused wakeup from Sleep. Set by hardware on Sleep wakeup due to GPIO[n].  This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	0
22	GP6		R/W	0
21	GP5		R/W	0
20	GP4		R/W	0
19	GP3		R/W	0
18	GP2		R/W	0
17	GP1		R/W	0
16	GP0	R/W	0	
15:2	—	Reserved, should be cleared.	R/W	0
1	SW	Sleep Wakeup. This bit is set by hardware on a Sleep wakeup and cleared by software by a write to this register.  A runtime reset can be detected if both SW and IP are 0 at boot.  This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	0
0	IP	Initial Power-up. This bit is set by hardware on a hardware reset and cleared by software by a write to this register.  A runtime reset can be detected if both SW and IP are 0 at boot.  This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	1

7.4.4.6 Sleep Power Register



Bits	Name	Description	R/W	Default
31:0	SP	A write to this register prepares the internal power supply for going to Sleep.	W	UNPRED

7.4.4.7 Sleep Register



Bits	Name	Description	R/W	Default
31:0	SL	A write to this register puts system to Sleep.	W	UNPRED

# Power-up, Reset and Boot

# 8

This section presents the power-up, hardware reset and runtime reset sequence for the Au1100 processor. In addition the boot vector is described.

## 8.1 Power-up Sequence

The Au1100 processor power structure is designed such that the external I/O voltage ( $V_{DDX}$  and  $V_{DDY}$ ) is driven separately from the core voltage ( $V_{DDI}$ ). In this way the core voltage can be sourced at lower voltages saving power. In addition the Au1100 processor is designed to allow the system designer to remove the core voltage during Sleep to maximize power efficiency.

Two signals  $VDDXOK$  and  $PWR\_EN$  are used to facilitate this power strategy.  $VDDXOK$  is used as a signal to the processor that power on  $V_{DDX}$  is stable. Stable is defined as having reached 90% of its nominal value.  $PWR\_EN$  is an output from the Au1100 that is asserted after  $VDDXOK$  is asserted and can be used as an enable to the regulator that is providing the core voltage,  $V_{DDI}$ .

The following describes the power-up sequence for the Au1100 processor:

- 1) Apply  $V_{DDX}$  and  $V_{DDY}$  (3.3V I/O power).
- 2) When  $V_{DDX}$  and  $V_{DDY}$  have reached 90% of nominal, assert  $VDDXOK$ .
- 3) The Au1100 processor then asserts  $PWR\_EN$  which can be used to enable the regulator driving  $V_{DDI}$  (CPU power).

Figure 8-1 shows the power-up sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 11.5.1 "Power-up Sequence Timing" on page 279.

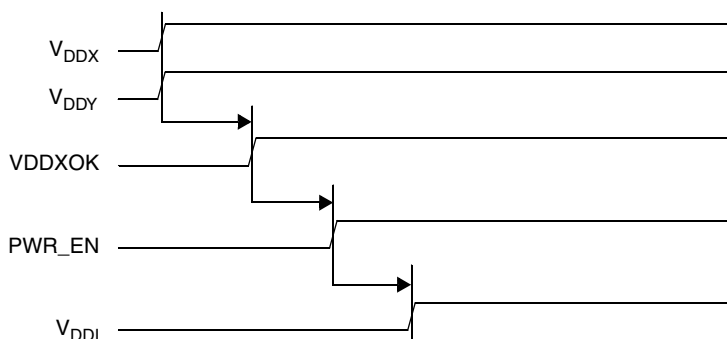


Figure 8-1. Power-up Sequence

## 8.2 Reset

A hardware reset is defined as a reset in which both  $VDDXOK$  and  $RESETIN\#$  are toggled. Typically this happens only at power-on, but a system designer can choose to tie  $VDDXOK$  and  $RESETIN\#$  together in which case all resets will be hardware resets.

For a runtime reset, power remains applied and only the  $RESETIN\#$  signal is toggled. Note that certain registers, specifically some of those in the system control block, are not affected by this type of reset. See the register description for the register in question for more information. If a register is not reset to defaults by both hardware reset and runtime reset, it is noted in the register description.

### 8.2.1 Hardware Reset

For a hardware reset, VDDXOK makes a transition from low to high followed by RESETIN# negating (transitioning from low to high). The following sequence describes a hardware reset:

- 1) ROMSEL and ROMSIZE should be terminated in the design so the appropriate boot type occurs. These values should not change during runtime.
- 2) At the same time or after VDDXOK is asserted, RESETIN# can be negated. In other words, RESETIN# can not be negated before VDDXOK is asserted. This allows VDDXOK and RESETIN# to be tied together.
- 3) RESETOUT# is negated after RESETIN# is negated.

Figure 8-2 shows the hardware reset sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 11.5.2 "Hardware Reset Timing" on page 280.

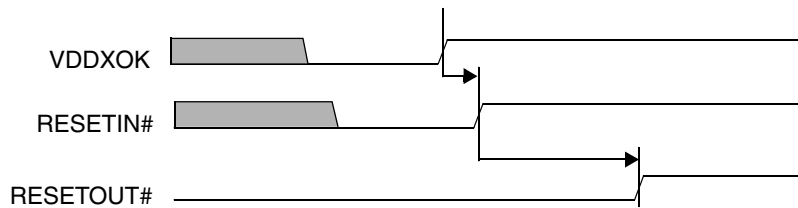


Figure 8-2. Hardware Reset Sequence

### 8.2.2 Runtime Reset

During runtime (after power is stable) the reset sequence can be broken down as follows:

- 1) During a runtime reset it is assumed that  $V_{DDX}$  and  $V_{DDI}$  remain at their nominal voltage. In addition, VDDXOK must remain asserted; otherwise, a hardware reset will occur. PWR\_EN remains asserted by the Au1100 processor.
- 2) RESETIN# is held asserted long enough to be recognized as a valid reset.
- 3) The processor acknowledges the reset by asserting RESETOUT#.
- 4) After RESETIN# is released, the processor signals the end of the reset by negating RESETOUT#.

Note that certain registers (specifically those in the system control block) are not affected by a runtime reset. Note also that ROMSEL and ROMSIZE should already be terminated in the design so the appropriate boot type occurs—these values should not change during runtime.

Figure 8-3 shows the runtime reset sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 11.5.3 "Runtime Reset Timing" on page 281.

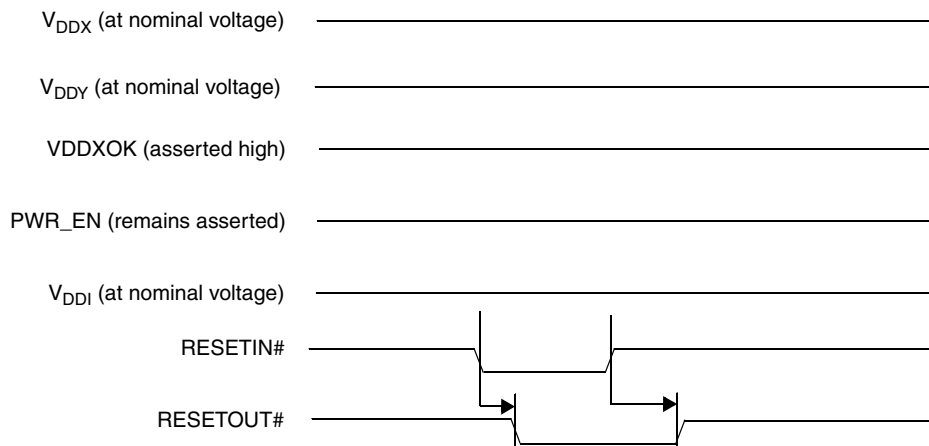


Figure 8-3. Runtime Reset Sequence



## 8.3 Boot

For both hardware and runtime resets, the CPU boots from KSEG1 address 0xBFC0 0000 which is translated to physical address 0x0 1FC0 0000; therefore, the system designer should place the start of the boot code at 0x0 1FC0 0000.

The ROMSEL and ROMSIZE signals determine the boot device type and width according to Table 8-1. The system designer should configure ROMSEL and ROMSIZE appropriately. Note that ROMSEL and ROMSIZE should not change during runtime.

**Table 8-1. ROMSEL and ROMSIZE Boot Device**

ROMSEL	ROMSIZE	Boot Device Type and Width
0	0	Boot from 32-bit ROM interface
0	1	Boot from 16-bit ROM interface
1	0	Boot from 32-bit SMROM interface and Sync Flash boot
1	1	Reserved

RCS0 is configured to be enabled for 0x0 1FC0 0000 at default when booting from a ROM device (ROMSEL = 0, ROMSIZE = x). See Section 3.2 "Static Bus Controller" on page 53, for more information about the default timing and size of the address enabled at reset.

SDCS0 is configured to be enabled for 0x0 1FC0 0000 at default when booting from a SMROM device (ROMSEL = 1, ROMSIZE = 0). See Section 3.1 "SDRAM Memory Controller" on page 44, for more information about the default timing and size of the address enabled at reset.

### 8.3.1 Endianness and 16-Bit Static Bus Boot

When booting from a 16-bit chip select on the static bus, the system designer must be sure the data format (endianness) is consistent across the Au1 core, the static bus controller, and the software image itself. This section describes how to make endianness consistent for both little- and big-endian systems.

For more on how the endian mode affects the behavior of 16-bit static bus chip selects, see "Halfword Ordering and 16-bit Chip Selects" on page 72.

**Note:** When programming ROM or Flash devices with a part programmer, take care to ensure that the programmer is not swapping bytes or halfwords erroneously. The configuration of the part programmer is often a source of error when initially bringing-up a new design.

#### 8.3.1.1 16-Bit Boot for Little-Endian System

Booting from 16-bit ROM or Flash in a system that is intended to run the Au1 core in little-endian mode is very straightforward. Generally speaking, the boot code and/or the application is compiled for little-endian. Because the the Au1 core defaults to big-endian mode, the boot code must change the Au1 core endianness to little-endian *before* any data accesses (to the 16-bit chip-select). The resulting boot code and/or application image is placed in the ROM/Flash memory in the little-endian format.

Even though the Au1 core starts in big-endian mode, the static bus controller properly retrieves instructions needed to boot the system since the application image is in little-endian format and the static bus controller defaults to little-endian ordering out of reset.

### 8.3.1.2 16-Bit Boot for Big-Endian System

Booting from 16-bit ROM or Flash in a system that is intended to run the Au1 core in big-endian mode is very straightforward, but does need one extra, important step.

Generally speaking, the boot code and/or the application is compiled for big-endian. The boot code must set the **mem\_stcfg[BE]** bit before it can properly fetch/reference the big-endian image. The resulting boot code and/or application image is placed in the ROM/Flash memory in the big-endian format.

In this situation, there is the dilemma that, out of reset, the Static Bus controller defaults to little-endian ordering, but the application image itself is in big-endian format. The solution is to place the following code at the reset exception vector (KSEG1 address 0xBFC0 0000, physical address 0x0 1FC0 0000):

```
.long 0xb4003c08 # lui      t0,0xb400
.long 0x10003508 # ori      t0,t0,0x1000
.long 0x00008d09 # lw       t1,0(t0)
.long 0x02003529 # ori      t1,t1,0x2000
.long 0x0000ad09 # sw       t1,0(t0)
.long 0x00000000 # nop
.long 0x00000000 # nop
.long 0x00000000 # nop
.long 0x00000000 # nop
```

The code does a read-modify-write of register **mem\_stcfg0** to set the BE bit. The values in the .long statements above are the halfword-swapped opcodes of the instructions in the comments to the right. With this technique, these first few instructions are actually in the little-endian format to match the static bus controller out of reset, and set **mem\_stcfg[BE]** which in turns allows the remainder of the big-endian memory contents to be accessed properly. The NOPs are necessary to ensure that the Au1 core pipeline does not contain incorrectly [halfword swapped] prefetched instructions. Note too that the NOP opcode 0x00000000 is the same instruction regardless of endian ordering.

NOTE: The boot code should set **mem\_stcfg0[BE]** as early as possible, preferably as the first activity. It is especially important to ensure that no cachable accesses take place to the 16-bit device, else the cache will contain the halfword swapped contents of the 16-bit memory.

### 8.3.2 System Boot

For system debug, the processor can be configured to boot from the EJTAG probe through the EJTAG port; see Section 9.0 "EJTAG" on page 227 for more information.



The Au1100 processor implements EJTAG following the MIPS' EJTAG 2.5 Specification. This section presents the EJTAG implementation on the Au1100 processor while concentrating on those features from the EJTAG 2.5 specification which are implementation specific. In addition, those features which have not been implemented or any differences in the Au1100 processor implementation of EJTAG from the rev 2.5 specification are also noted.

It is assumed that the EJTAG 2.5 specification will be referenced for implementation details not covered here. If a particular bit is not implemented it can be assumed that the functionality associated with the bit is not implemented or not applicable unless otherwise noted.

The following features comprise the EJTAG implementation on the Au1100 processor:

- Extended instructions SDBBP and DERET
- Debug exceptions
- Extended CP0 registers DEBUG, DEPC and DESAVE
- EJTAG memory range 0xFF200000 to 0xFF3FFFFFF
- Instruction/data breakpoints through the watch exception (specific to Au1100)
- Processor bus breakpoints (from EJTAG 2.0)
- Memory overlay (from EJTAG 2.0)
- EJTAG tap per IEEE1149.1

**Note:** The optional data and instruction breakpoint features from the EJTAG 2.5 specification are not implemented.

## 9.1 EJTAG Instructions

Both SDBBP and DERET are supported by the Au1100 processor:

- SDBBP causes a Debug Breakpoint exception.
- DERET is used to return from a Debug Exception.

## 9.2 Debug Exceptions

The following exceptions will cause entry into debug mode.

- DSS - debug single step
- DINT - debug interrupt, processor bus break
- DBp - execution of SDBBP instruction
- DWATCH - debug watch exception. Au1100 processor-specific implementation allowing CPU watch exception to cause debug exception. See description of the "EJWatch Register (TAP Instruction EJWATCH)" on page 241 register.

Note that other normal exceptions, when taken in debug mode, will be handled by the debug exception handler.

### 9.3 Coprocessor 0 Registers

The Coprocessor 0 Registers for EJTAG are shown in Table 9-1.

**Table 9-1. Coprocessor 0 Registers for EJTAG**

Register Number	Select	Name	Description
23	0	debug	Debug indications and controls for the processor.
24	0	depc	Program Counter at last debug exception or exception in debug mode.
31	0	desave	Debug exception save register.

#### 9.3.1 Debug Register (CP0 Register 23, Select 0)

The Debug register contains the cause of the most recent debug exception and exception in Debug Mode. It also controls single stepping. Only the DM bit and the EJTAGver field are valid when read from the Debug register in Non-Debug Mode; the value of all other bits and fields is UNPREDICTABLE.

The following bits and fields are updated only on debug exceptions and/or exceptions in Debug Mode:

- DSS, DBp, DINT are updated on both debug exceptions and on exceptions in Debug Modes.
- DExcCode is updated on normal exceptions in Debug Mode, and is undefined after a debug exception.
- DBD is updated on both debug and on normal exceptions in Debug Modes.

debug

CP0 Register 23, Select 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	x	x	x	x	x	0	0	0	0	0	x	0	0	0	x	x
	DD	DM	ND	LS			CD									VER			DEXCOSE		NS	SS			DI			DB	DS				

Bits	Name	Description	R/W	Default
31	DD	DBD. Indicates whether the last debug exception or exception in Debug Mode occurred in a branch or jump delay slot. 0 Not in delay slot. 1 In delay slot.	R	UNPRED
30	DM	Indicates that the processor is operating in Debug Mode. 0 Processor is operating in Non-Debug Mode. 1 Processor is operating in Debug Mode.	R	0
29	ND	NoDCR. 0 DSEG is present. 1 DSEG is not present.	R	0
28	LS	LSNM. Controls access of loads/stores between dseg and remaining memory when dseg is present and while in debug mode. 0 Loads/stores in dseg address range go to dseg 1 Loads/stores in dseg address range go to system memory	R/W	0
27	—	Reserved, should be cleared. <i>This bit is called Doze in the EJTAG 2.5 specification and was not implemented.</i>	R	0
26	—	Reserved, should be cleared. <i>This bit is called Halt in the EJTAG 2.5 specification and was not implemented.</i>	R	0
25	CD	CountDM. This bit is 0, indicating that the counter will be stopped in debug mode.	R	0
24	—	Reserved, should be cleared. <i>This bit is called IBusEP in the EJTAG 2.5 specification and was not implemented.</i>	R	0
23	—	Reserved, should be cleared. <i>This bit is called MCheckP in the EJTAG 2.5 specification and was not implemented.</i>	R	0

Bits	Name	Description	R/W	Default
22	—	Reserved, should be cleared. <i>This bit is called CacheEP in the EJTAG 2.5 specification and was not implemented.</i>	R	0
21	—	Reserved, should be cleared. <i>This bit is called DBusEP in the EJTAG 2.5 specification and was not implemented.</i>	R	0
20	—	Reserved, should be cleared. <i>This bit is called IEXI in the EJTAG 2.5 specification and was not implemented.</i>	R	0
19	—	Reserved, should be cleared. <i>This bit is called DDBSImpr in the EJTAG 2.5 specification and was not implemented.</i>	R	0
18	—	Reserved, should be cleared. <i>This bit is called DDBLImpr in the EJTAG 2.5 specification and was not implemented.</i>	R	0
17:15	VER	EJTAGver. 1 EJTAG Version 2.5.	R	1
14:10	DEXCODE	DExcCode. Indicates the cause of the latest exception in Debug Mode. The field is encoded as the ExcCode field in the Cause register for those exceptions that can occur in Debug Mode (the encoding is shown in the MIPS32 specification), with addition of code 30 with the mnemonic CacheErr for cache errors. This value is undefined after a debug exception.	R	UNPRED
9	NS	NoSSt. 0 Single step is implemented.	R	0
8	SS	SSSt. Controls whether single-step feature is enabled: 0 No enable of single-step feature. 1 Single-step feature enabled.	R/W	0
7:6	—	Reserved, should be cleared.	R	0
5	DI	DINT. Indicates that a Debug Interrupt exception occurred. This could be either a Processor Bus Break (indicated by BS0 in the Processor Bus Break Status Register) or EJTAG break. The BS0 bit should be checked to see what caused the exception. Cleared on exception in Debug Mode. 0 No Debug Interrupt exception. 1 Debug Interrupt exception.	R	UNPRED
4	—	Reserved, should be cleared. <i>This bit is called DIB in the EJTAG 2.5 specification and was not implemented.</i>	R	0
3	—	Reserved, should be cleared. <i>This bit is called DDBS in the EJTAG 2.5 specification and was not implemented.</i>	R	0
2	—	Reserved, should be cleared. <i>This bit is called DDBL in the EJTAG 2.5 specification and was not implemented.</i>	R	0
1	DB	DBp. Indicates that a Debug Breakpoint exception occurred. Cleared on exception in Debug Mode. 0 No Debug Breakpoint exception. 1 Debug Breakpoint exception.	R	UNPRED
0	DS	DSS. Indicates that a Debug Single Step exception occurred. Cleared on exception in Debug Mode. 0 No debug single-step exception. 1 Debug single-step exception.	R	UNPRED

### 9.3.2 Debug Exception Program Counter Register

The Debug Exception Program Counter (DEPC) register is a read/write register that contains the address at which processing resumes after the exception has been serviced.

Hardware updates this register on debug exceptions and exceptions in Debug Mode.

For precise debug exceptions and precise exceptions in Debug Mode, the DEPC register contains either:

- the virtual address of the instruction that was the direct cause of the exception; or
- the virtual address of the immediately preceding branch or jump instruction, when the exception-causing instruction is in a branch delay slot, and the Debug Branch Delay (BDB) bit in the Debug register is set.

For imprecise debug exceptions and imprecise exceptions in Debug Mode, the DEPC register contains the address at which execution is resumed when returning to Non-Debug Mode.

depc - Debug Exception Program Counter																	CP0 Register 24, Select 0																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	DEPC	Debug Exception Program Counter	R/W	UNPRED

### 9.3.3 Debug Exception Save Register - DESAVE

The Debug Exception Save (DESAVE) register is a read/write register that functions as a simple scratchpad register.

The debug exception handler uses this to save one of the GPRs, which is then used to save the rest of the context to a pre-determined memory area, for example, in the dmseg. This register allows the safe debugging of exception handlers and other types of code where the existence of a valid stack for context saving cannot be assumed.

desave - Debug Exception Save Register																	CP0 Register 31, Select 0															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Bits	Name	Description	R/W	Default
31:0	DESAVE	Debug Exception Save contents	R/W	UNPRED

## 9.4 EJTAG Memory Range

In debug mode accesses to virtual 0xFF200000 to 0xFF3FFFFFF bypass translation.

The debug memory is split into two logical divisions:

- dmseg: 0xFF200000 - 0xFF2FFFFFF
- drseg: 0xFF300000 - 0xFF3FFFFFF

**Note:** The physical address addr[35:32] of this range is zero.

Dmseg is the memory range that will be serviced by the probe TAP in debug mode for all instruction accesses to this virtual address range and for data accesses if the LSNM in the Debug Register is 0.

Drseg is the memory range containing the EJTAG memory mapped registers and is accessible when LSNM in the Debug Register is 0.

### 9.4.1 EJTAG Memory Mapped Registers

Table 9-2 shows the EJTAG memory mapped registers located in drseg.

**Table 9-2. EJTAG Memory Mapped Registers at 0xFF300000**

Offset	Register	Description
0x0000	dcr	Debug Control Register
0x000C	pbs	Processor Break Status
0x0300	pab	Processor Address Bus Break
0x0304	pdb	Processor Data Break
0x0308	pdm	Processor Data Mask
0x030C	pbcam	Processor Control/Address Mask
0x0310	phab	Processor High Address Break
0x0314	pham	Processor High Address Mask

The EJTAG implementation in the Au1100 processor does not employ data breakpoints and instruction breakpoints as described in the EJTAG 2.5 specification. Instead it offers Processor breakpoints as described in the EJTAG 2.0.0 specification.

The Processor Bus Match registers monitor the bus interface of the MIPS CPU and provide debug exception or trace trigger for a given physical address and data.

In addition, the implementation allows the CPU watchpoints to cause a debug exception. This functionality is enabled through the EJTAG TAP port. Please see “EJWatch Register (TAP Instruction EJWATCH)” on page 241 for details.

#### 9.4.1.1 Debug Control Register

The Debug Control Register (DCR) controls and provides information about debug issues. The width of the register is 32 bits. The DCR is located in the drseg at offset 0x0000.

##### dcr - Debug Control Register

Offset = 0x0000

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		EN												DB	IB													IE	NE	NP	SR	PE
Def.	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0

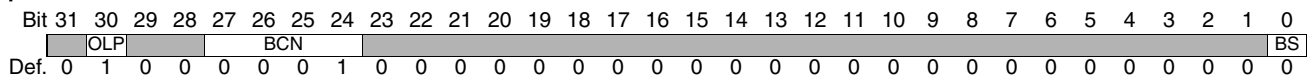
Bits	Name	Description	R/W	Default
31:30	—	Reserved, should be cleared.	R	0
29	EN	ENM. 1 Processor is big Endian in both debug and kernel mode.	R	1
28:18	—	Reserved, should be cleared.	R	0
17	DB	DataBrk. 0 No data hardware breakpoints implemented.	R	0
16	IB	InstBrk. 0 No instruction hardware breakpoints implemented.	R	0
15:5	—	Reserved, should be cleared.	R	0
4	IE	IntE. 1 Interrupt enabled in debug mode depending on other enabling mechanisms.	R	1
3	NE	NMIE. 1 Non-Maskable Interrupt is enable for non-debug mode. The NMI is not implemented in the Au1100 so this bit has no applicability.	R	1

Bits	Name	Description	R/W	Default
2	NP	NMIPend. 0 No NMI pending The NMI is not implemented in the Au1100 so this bit has no applicability.	R	0
1	SR	SRstE. 1 Soft reset is fully enabled. Soft Reset is not implemented in the Au1100 so this bit has no applicability.	R	1
0	PE	ProbEn. Indicates value of the ProbEn value in the ECR register. 0 No access should occur to dmseg. 1 Probe services accesses to dmseg.	R	Same value as ProbEN in ECR

**9.4.1.2 Processor Bus Break Status Register**

**pbs - Processor Bus Break Status**

Offset = 0x000C



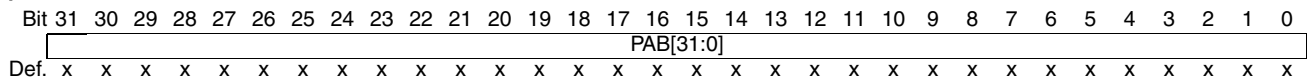
Bits	Name	Description	R/W	Default
31	—	Reserved, should be cleared.	R	0
30	OLP	1 Memory overlay functionality is implemented for processor breaks.	R	1
29:28	—	Reserved, should be cleared.	R	0
27:24	BCN	Number of Processor Breaks. 1 One Channel has been implemented for the Processor Bus Break.	R	1
23:15	—	Reserved, should be cleared.	R	0
14:1	—	Reserved, should be cleared. <i>These bits are the Bsn bits in the EJTAG 2.0.0 specification and are not needed since only one break is implemented.</i>	R	0
0	BS	Break Status. This bit, when set, indicates that a processor bus break or processor bus trigger has occurred. BS can be cleared by activating PrRst (EJTAG CONTROL Register), hard reset and also by writing a '0' to it. The Debug handler must clear this bit before returning from debug mode.	R/W	0

**9.4.1.3 Processor Address Bus Break**

This register contains the bits of the physical Processor Address Bus Break.

**pab - Processor Address Bus Break**

Offset = 0x0300



Bits	Name	Description	R/W	Default
31:0	PAB	Processor Address Bus Break 0. This index contains the lower 32 bits of the physical address. In combination with the high order address bits, these bits make up the break address.	R	UNPRED

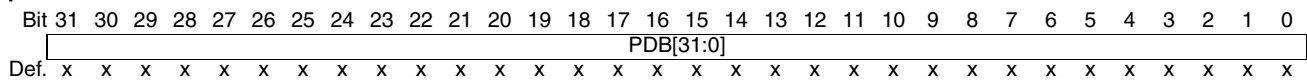


#### 9.4.1.4 Processor Data Bus Break

This register specifies the data value for the Processor Data Bus match.

##### pdb - Processor Data Bus Break

Offset = 0x0304



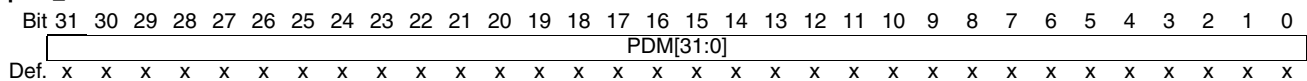
Bits	Name	Description	R/W	Default
31:0	PDB	Processor Data Bus Break 0. This index contains the 32 bits of the data bus match.	R	UNPRED

#### 9.4.1.5 Processor Data Mask/Upper Overlay Address Mask

This register is dual purpose depending on the value of the Overlay Enable bit in the Processor Bus Break Control and Address Mask. This register specifies the mask value for the Processor Data Mask register. Each bit corresponds to a bit in the Data register.

##### pdm\_uoam - Processor Data Mask

Offset = 0x0308



Bits	Name	Description	R/W	Default
31:0	PDM	Applies only when OE not enabled. Processor Data Mask 0. When OE in the pbcam register is not enabled. 0 Data bit is not masked, data bit is compared. 1 Data bit is masked, data bit is not compared.	R	UNPRED
31:24'	UOAM	Applies only when OE is enabled. Upper Overlay Address Mask. These bits represent bits 31:24 of the address mask and are combined with the LAM and HAM fields to create a complete 36 bit address mask. 0 Address bit is not masked, address bit is compared. 1 Address bit is masked, address bit is not compared. Note that bits 23:0 are not used when OE is set and should be written 0.	R/W	UNPRED

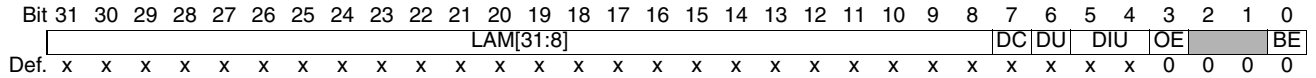
**9.4.1.6 Processor Bus Break Control and Address Mask**

This register selects the Processor Bus match function to enable debug break or trace trigger. It also includes control bits to enable comparison as well as mask bits to exclude address bits from comparison.

**Note:** All processor break exceptions are imprecise.

**pbcam - Bus Break Control and Address Mask**

Offset = 0x030C



Bits	Name	Description	R/W	Default
31:8	LAM	Address Mask. These bits specify the mask value for the 24 lower bits of the Processor Address register (PBA0[23..0]). Each bit corresponds to the same bit in PBA0. 0 Address bit is not masked, address bit is compared. 1 Address bit is masked, address bit is not compared.	R/W	UNPRED
7	DC	Data Store to Cached Area. This bit enables the comparison on Processor Address and Data Bus for Data Store to the Cached area. 0 Processor Address and Data is not compared for storing data to the Cached area. 1 Processor Address and Data is compared for storing data to the Cached area.	R/W	UNPRED
6	DU	Data Store To Uncached Area. This bit enables the comparison on Processor Address and Data Bus for Data Store to the uncached area. 0 Processor Address and Data is not compared for storing data into the un-cached area. 1 Processor Address and Data is compared for storing data into the un-cached area.	R/W	UNPRED
5:4	DIU	Data or Instruction fetch or load from Uncached Area. These bits enable the comparison on Processor Address and Data Bus for Data or Instruction load and fetch from the un-cached area. 00 Processor Address and Data is not compared for loading data or fetching instruction from the un-cached area. 11 Processor Address and Data is compared for loading data or fetching instruction from the un-cached area. Bits 5 and 4 were named ILUC and DFUC in the EJTAG 2.0.0 specification and were implemented separately for instruction and data fetches.	R	UNPRED
3	OE	Overlay Enable. When this bit is 1 and the processor physical address, masked by the HAM, UOAM and the LAM fields (all 36 bits of the address mask), matches the PHAB and PAB registers, then the memory request is redirected to the EJTAG Probe.  The processor bus break can not be used for normal break, function if the OLE bit is set, so BE must be set to 0. The behavior is otherwise undefined.  Overlay is only valid for memory regions. It is not valid for I/O or debug space and the behavior is unpredictable if addresses within this space are used.	R/W	0
2	—	Reserved, should be cleared. <i>This bit is called TE in the EJTAG 2.0.0 specification and was not implemented.</i>	R	0
1	—	Reserved, should be cleared. <i>This bit is called CBE in the EJTAG 2.0.0 specification and was not implemented.</i>	R	0

Bits	Name	Description	R/W	Default
0	BE	<p>Break Enable. This bit enables the Processor Bus break function.</p> <p>0 Processor Bus break function is disabled.</p> <p>1 Processor Bus break function is enabled.</p> <p>If Break Enable is set and the processor physical address, masked by the HAM and the LAM fields (UOAM is only for overlay so bits 31:24 are not masked here), matches the PHAB and PAB registers, and the processor data bus matches the PDB register (masked by PDM), then a debug exception to the processor is generated.</p> <p>The BS bit in the Processor Bus Break Status register is set and the DINT bit in the Debug Register is set. If the debug exception handler is already running (DM='1'), then the debug exception will not be taken until DM = 0.</p> <p>This functionality is mutually exclusive to OLE so only one of OLE or BE should be set at any time.</p>	R/W	0

#### 9.4.1.7 Processor High Address Bus Break

This register specifies the high order address for the processor address bus break.

##### pha - Processor High Address Bus Break

Offset = 0x0310

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X

Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3:0	HA	These bits map to the high physical address bits 35:31.	R/W	UNPRED

#### 9.4.1.8 Processor High Address Mask

This register specifies the high order address mask for the processor address bus break.

##### pham - Processor High Address Mask

Offset = 0x0314

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X

Bits	Name	Description	R/W	Default
31:4	—	Reserved, should be cleared.	R	0
3:0	HAM	<p>High Address Mask for address bits 35:31</p> <p>0 Data bit is not masked, data bit is compared</p> <p>1 Data bit is masked, data bit is not compared</p>	R/W	UNPRED

### 9.4.2 EJTAG Test Access Port (TAP)

The EJTAG TAP contains the five TAP pins and a 16 state controller with a 5-bit instruction register.

Table 9-3 shows the 5-bit instructions supported by the Au1100.

**Table 9-3. EJTAG Instruction Register Values**

Hex Value	Instruction	Function
0x00	EXTEST	Boundary Scan
0x01	IDCODE	Selects ID Register
0x02	SAMPLE	Boundary Scan Sample/Preload (IEEE JTAG Instruction)
0x03	IMPCODE	Selects Implementation Register
0x04	—	Reserved
0x05	—	This reserved register is for test mode HIZ - Tristate all output pins and Select Bypass register.
0x06	—	This reserved register is for test mode CLAMP - IEEE Clamp pins and select bypass register.
0x07	—	Reserved
0x08	ADDRESS	Selects Address Register.
0x09	DATA	Selects Data Register.
0x0A	CONTROL	Selects EJTAG Control Register.
0x0B	ALL	Selects the Address, Data and EJTAG Control registers.
0x0C	EJTAGBOOT	Makes the processor take a debug exception after reset.
0x0D	NORMALBOOT	Makes the processor execute the reset handler after reset.
0x0E-0x1B	—	Reserved
0x1C	EJWATCH	Selects Watch register
0x1D-0x1E	—	Reserved
0x1F	BYPASS	Bypass mode

#### 9.4.2.1 Device Identification (ID) Register

The Device ID register is a 32-bit read-only register that identifies the specific device implementing EJTAG.

##### IDCODE - Device Identification

##### TAP Instruction IDCODE

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	VER				PNUM																MANID												
Def.	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1

Bits	Name	Description	R/W	Default
31:28	VER	Identifies the version of the device.	R	0
27:12	PNUM	Identifies the part number of the device.	R	0x03E8
11:1	MANID	Identifies the manufacturer ID code for the device. MANID[6:0] are derived from the last byte of the JEDEC code with the parity bit discarded. MANID[10:7] provides a binary count of the number of bytes in the JEDEC code that contain the continuation character (0x7F). When the number of continuations characters exceeds 15, these four bits contain the modulo-16 count.	R	0x147
0	—	This bit is reserved and should be written a 1.	R	1

### 9.4.2.2 Implementation Register

The Implementation register is a 32-bit read-only register that identifies features implemented in this EJTAG compliant processor, mainly those accessible from the TAP.

#### IMPCODE - Implementation

IMPCODE - Implementation																TAP Instruction IMPCODE																
Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
VER		R3						DI		AS						M16		ND		M32												
Def.	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:29	EJTAGver	1 EJTAG version 2.5.	R	1
28	R3	0 R3k privileged environment.	R	0
27:25	—	Reserved, should be cleared.	R	0
24	DI	0 DINT signal from the probe is not supported.	R	0
23	—	Reserved, should be cleared.	R	0
22:21	AS	10 8-bit ASID.	R	10
20:17	—	Reserved, should be cleared.	R	0
16	M16	0 No MIPS16 support.	R	0
15	—	Reserved, should be cleared.	R	0
14	ND	1 No EJTAG DMA support.	R	1
13:1	—	Reserved, should be cleared.	R	0
0	MIPS32/64	0 32-bit processor.	R	0

### 9.4.2.3 Data Register

The read/write Data register is used for opcode and data transfers during processor accesses. The width of the Data register is 32 bits.

The value read in the Data register is valid only if a processor access for a write is pending, in which case the data register holds the store value. The value written to the Data register is only used if a processor access for a pending read is finished afterwards, in which case the data value written is the value for the fetch or load. This behavior implies that the Data register is not a memory location where a previously written value can be read afterwards.

#### DATA

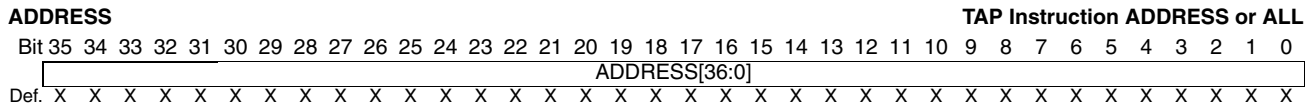
DATA																TAP Instruction DATA or ALL																
Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DATA[31:0]																																
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	DATA	Data used by processor access	R/W	UNPRED

**9.4.2.4 Address Register**

The read-only Address register provides the address for a processor access. The width of the register is 36 bits.

The value read in the register is valid if a processor access is pending, otherwise the value is undefined. The two LSBs of the register are used with the Psz field from the EJTAG Control register to indicate the size and data position of the pending processor access transfer. These bits are not taken directly from the address referenced by the load/store (i.e. these bits are encoded with Psz).



Bits	Name	Description	R/W	Default
35:0	Address	Address used by processor access.	R	UNPRED

**9.4.2.5 EJTAG Control Register (ECR)**

The 32-bit EJTAG Control Register (ECR) handles processor reset, Debug Mode indication, access start, finish, and size and read/write indication. The ECR also:

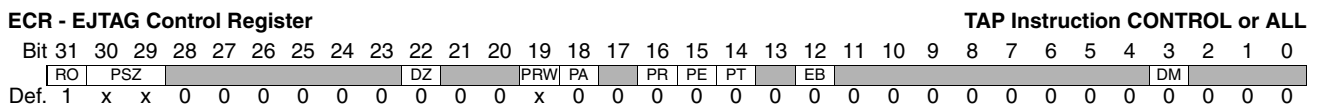
- Controls debug vector location and indication of serviced processor accesses.
- Allows debug interrupt request.
- Indicates processor low-power mode.

The EJTAG Control register is not updated/written in the Update-DR state unless the Reset occurred; that is RO (bit 31) is either already 0 or is written to 0 at the same time. This condition ensures proper handling of processor accesses after a reset.

Bits that are R/W in the register return their written value on a subsequent read, unless other behavior is defined. Internal synchronization hardware thus ensures that a written value is updated for reading immediately afterwards, even when the TAP controller takes the shortest path from the Update-DR to Capture-DR state.

Note: To ensure a write is successful to the PE, PT and EB bits when the processor is undergoing a clock change (for PLL lock/relock), the host must continue writing these bits until the write is verified by reading the change. Failure to do this could result in the write of these bits being lost.

Reset of the processor can be indicated in the TCK domain a number of TCK cycles after it is removed in the processor clock domain in order to allow for proper synchronization between the two clock domains.



Bits	Name	Description	R/W	Default
31	RO	Indicates if a processor reset has occurred since the bit was cleared: 0 No reset occurred 1 Reset occurred The RO bit stays set as long as reset is applied. This bit must be cleared to acknowledge that the reset was detected. The EJTAG Control register is not updated in the Update-DR state unless RO is 0 or written to 0 at the same time. This is in order to ensure correct handling of the processor access after reset.	R/W0	1

Bits	Name	Description	R/W	Default
30:29	PSZ	Indicates the size of a pending processor access, in combination with the Address register. 00 Byte 01 Halfword 10 Word 11 Triple This field is valid only when a processor access is pending; otherwise, the read value is undefined.	R	UNPRED
28:23	—	Reserved, should be cleared.	R	0
22	DZ	Doze. Indicates if the processor is in a WAIT state: 0 Processor is not in a wait state. 1 Processor is in a wait state.	R	0
21	—	Reserved, should be cleared. <i>This bit is called Halt in the EJTAG 2.0.0 specification and was not implemented.</i>	R	0
20	—	Reserved, should be cleared. <i>This bit is called PerRst in the EJTAG 2.0.0 specification and was not implemented.</i>	R	0
19	PRW	Indicates read or write of a pending processor access. 0 Read processor access, for a fetch/load access 1 Write processor access, for a store access This value is defined only when a processor access is pending.	R	UNPRED
18	PA	Indicates a pending processor access and controls finishing of a pending processor access. When read: 0 No pending processor access 1 Pending processor access A write of 0 finishes a processor access if pending; otherwise operation of the processor is UNDEFINED if the bit is written to 0 when no processor access is pending. A write of 1 is ignored.	R/W0	0
17	—	Reserved, should be cleared.	R	0
16	PR	Controls the processor reset. 0 No processor reset applied 1 Processor reset applied Setting this bit to 1 will apply a processor reset. When this bit is read back it will always read a 0. Note that startup latencies should be observed when applying reset.	R/W	0
15	PE	Controls indication to the processor of whether the probe expects to handle accesses to EJTAG memory through servicing of processor accesses. 0 Probe does not service processor accesses 1 Probe will service processor accesses The ProbEn bit is reflected as a read-only bit in the Debug Control Register (DCR) bit 0. When a read from this bit shows a change, the new value has taken effect in the DCR. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain. However, a change of the ProbEn prior to setting the EhtagBrk bit is ensured to affect execution of the debug handler due to the debug exception. Not all combinations of ProbEn and ProbTrap are allowed. Please see the previous note about writing this bit (in “EJTAG Control Register (ECR)” on page 238).	R/W	Determined by EJTAG-BOOT

Bits	Name	Description	R/W	Default
14	PT	<p>Controls location of the debug exception vector:</p> <p>0 Normal memory 0xBFC0 0480</p> <p>1 EJTAG memory 0xFF20 0200</p> <p>When a read from this bit shows a change, the new value has taken effect in the DCR. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain.</p> <p>However, a change of the ProbTrap prior to setting the EhtagBrk bit is ensured to affect execution of the debug handler due to the debug exception.</p> <p>Not all combinations of ProbEn and ProbTrap are allowed.</p> <p>Please see the previous note about writing this bit (in “EJTAG Control Register (ECR)” on page 238).</p>	R/W	Determined by EJTAG-BOOT
13	—	Reserved, should be cleared.	R	0
12	EB	<p>Requests a debug interrupt exception to the processor when this bit is written as 1. This bit is cleared by hardware when the processor enters debug mode. If software then sets EB while the processor is already in debug, the request is not ignored but is delayed. That is, once the processor returns to normal mode, the pending debug exception request immediately sends the processor back into debug.</p> <p>A write of 0 is ignored. The debug request restarts the processor clock if the processor was in a wait mode, which stopped the processor clock. The read value indicates a pending Debug Interrupt exception requested through this bit:</p> <p>0 No pending Debug Interrupt exception requested through this bit</p> <p>1 Pending Debug Interrupt exception</p> <p>The read value can, but is not required to, indicate other pending DINT debug requests (for example, through the DINT signal).</p> <p>Please see the previous note about writing this bit (in “EJTAG Control Register (ECR)” on page 238).</p>	R/W1	Determined by EJTAG-BOOT
11:4	—	Reserved, should be cleared	R	0
3	DM	<p>Indicates if the processor is in Debug Mode.</p> <p>0 Processor is in Non-Debug Mode.</p> <p>1 Processor is in Debug Mode.</p>	R	0
2:0	—	Reserved, should be cleared.	R	0



#### 9.4.2.6 EJWatch Register (TAP Instruction EJWATCH)

The EJWatch register is used to enable CPU watchpoints to cause a debug exception. This functionality is unique to the Au1100.

EJWATCH		TAP Instruction EJWATCH							
Bit	7	6	5	4	3	2	1	0	
Def.	0	0	0	0	0	0	0	0	WATCH

Bits	Name	Description	R/W	Default
7:3	—	Reserved, should be cleared.	R	0
2	—	Reserved, should be cleared. This bit is the Global Scan test bit.	R	0
1	—	Reserved, should be cleared. This bit is a Test Mode bit.	R	0
0	WATCH	<p>This bit controls the debug functionality of the CPU watch register.</p> <p>0 Normal Watch Exception Mode</p> <p>1 Debug Watch Exception Mode</p> <ul style="list-style-type: none"> <li>- Blocks writes to Watch register in non-debug mode</li> <li>- Watch Exception will become debug exceptions with DEXCODE=23</li> <li>- The PC will be saved in the DEPC (not in the EPC as with a normal watch exception).</li> </ul> <p>Note that the Status, Cause, and EPC will not be affected by a debug watch exception when this bit is enabled.</p>	R/W	0

#### 9.4.2.7 Bypass Register (TAP Instruction BYPASS)

The Bypass register is a one-bit read-only register, which provides a minimum shift path through the TAP. This register is also defined in IEEE 1149.1.

BYPASS		TAP Instruction BYPASS	
Bit			0
Def.			BP
			0

Bits	Name	Description	R/W	Default
0	BP	Ignored on writes; returns zeros on reads.	R	0

### 9.4.3 EJTAG TAP Hardware Considerations

The EJTAG interface consists of the signals listed in Table 9-4.

**Table 9-4. EJTAG Signals**

Signal	Input/Output	Definition
TRST#	I	Asynchronous TAP reset
TDI	I	Test data input to the instruction or selected data registers. This signal will be sampled on the rising edge of TCK
TDO	O	Test data output from the instruction or data register. This signal will transition on the falling edge (valid on rising edge) of TCK
TMS	I	Control signal for TAP controller. This signal is sampled on the rising edge of TCK.
TCK	I	Control clock for updating TAP controller and shifting data through instruction or selected data register.

Note that the EJTAG TAP signal TCK must always be less than 1/4 the System Bus (SBUS) clock speed for proper operation. In addition, termination as shown in EJTAG 2.5 spec must be followed.

# 10 Signal Descriptions

---

This section describes the external signals on the Au1100 processor.

In order to maximize the functionality of the Au1100 processor, many of the pins have multiple uses. Note that if a pin is configured for one use, any other functionality associated with that pin can not be utilized at the same time. In other words a pin can not be used as a general-purpose I/O signal at the same time it is assigned to a peripheral device. (See Section 7.3.1 "Pin Functionality" on page 209.)

Figure 10-1 on page 244 shows the external signals of the Au1100 processor. All signals are grouped according to their functional block. Signals that share a pin are listed with the multiplexed signal name in parentheses—the signal name shown in **bold** is the default.

**Note:** A signal with an “#” is active-low; that is, the signal is considered asserted (active) when low and negated when high. Active-high signals (no overbar) are considered asserted when high and negated when low.

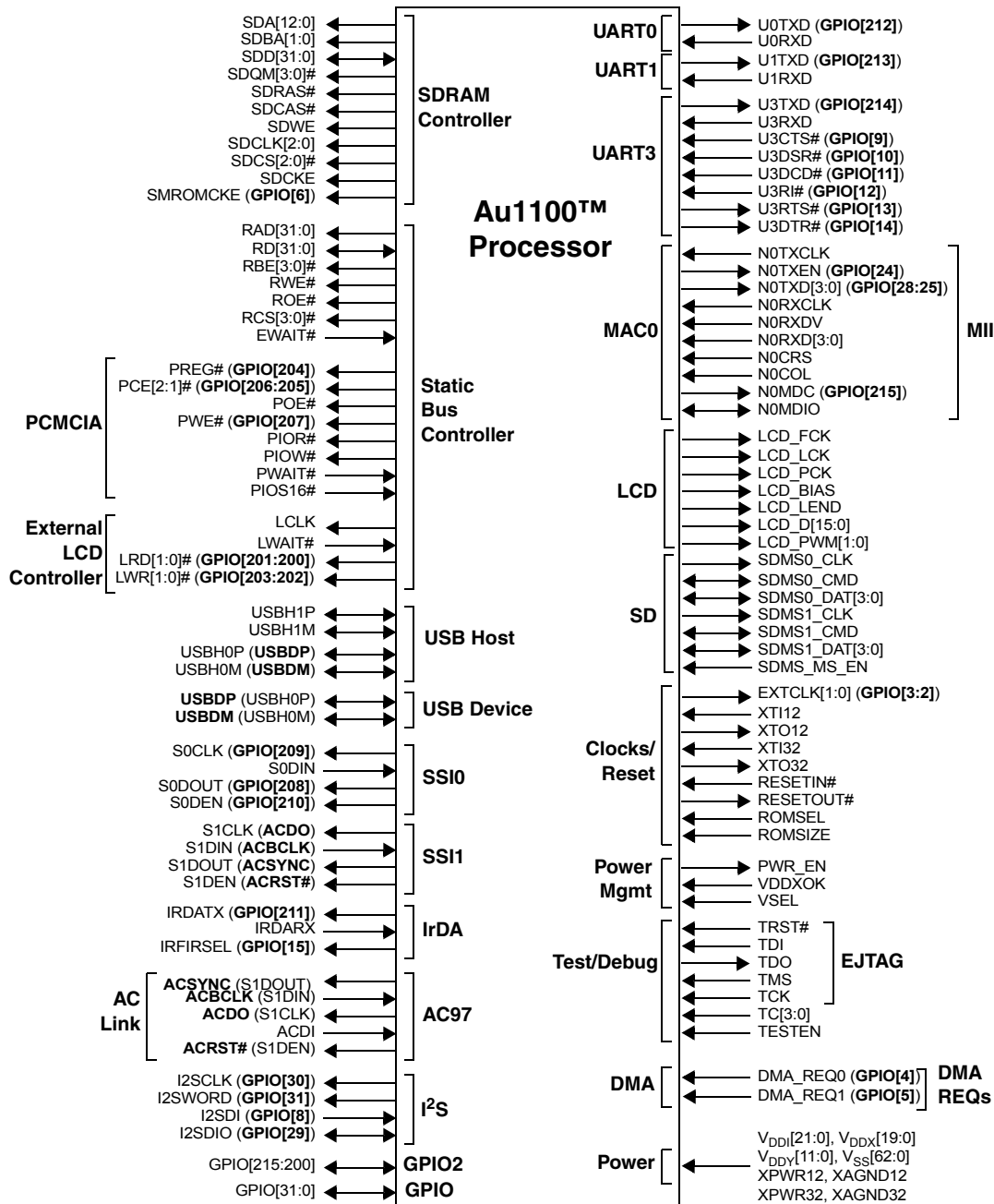


Figure 10-1. Au1100™ Processor External Signals

Table 10-3 gives a description of all external signals on the Au1100 processor. The signals have been grouped by functional block. Signals that require external termination are noted in the description. Table 10-3 also defines the default state of the signals during a hardware reset, a runtime reset, and Sleep. The abbreviations used for the signal types and the signal states are defined in Table 10-1 on page 245 and Table 10-2 on page 245.

**Table 10-1. Signal Type Abbreviations for Table 10-3**

Signal Type	Definition
I	Input. Note that all <i>unused</i> input pins should be terminated low or high via direct connection to either ground or power.
O	Output
IO	Bidirectional
Z	Tristatable
P	Power
G	Ground

**Table 10-2. Signal State Abbreviations for Table 10-3**

Signal State	Definition
0	Driven low
1	Driven high
IN	Signal is a input.
LV	If driven, an output signal continues to be driven at the last value before a reset or entering Sleep.
HIZ	TRI-STATE
ON	Clock remains on <i>if already enabled</i> .
DEP	Depends (Signal-specific explanations are provided in table footnotes.)
UN	Unpredictable
NC	Not connected
NA	Does not apply because this signal is not the default function coming out of a hardware or runtime reset.

Note for Table 10-3 that the signal states shown in the far-right column are valid during Sleep. When waking from Sleep, the processor performs an internal system reset that produces the same signal behavior as a runtime reset with two exceptions:

- SDRAM interface behavior. During and after a runtime reset the SDRAM configuration *mode* registers retain their values to allow a transaction in progress to complete; the remaining SDRAM configuration registers revert to their default values. Waking from Sleep, however, *all* SDRAM configuration registers revert to their default values, and the interface behaves the same as when coming out of a *hardware* reset.
- PWR\_EN behavior. During a runtime reset PWR\_EN remains asserted. During Sleep, PWR\_EN is negated. Waking from Sleep, PWR\_EN is asserted according to the timing specified in Section 7.4.3.1 "Sleep Sequence and Timing" on page 217.

Table 10-3. Signal Descriptions

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
<b>SDRAM Interface</b>					
SDA[12:0]	O	Address Outputs: A0-A12 are sampled during the ACTIVE command (row-address A0-A12) and READ/ WRITE command to select one location out of the memory array in the respective bank. The address outputs also provide the opcode during a LOAD MODE REGISTER command.	UN	UN	DEP (Note 1)
SDBA[1:0]	O	Bank Address Outputs: SDBA1 and SDBA0 define to which bank the ACTIVE, READ, WRITE, or PRECHARGE command is being applied.	UN	UN	DEP (Note 1)
SDD[31:0]	IO	SDRAM Data Bus. During a hardware reset the SDRAM data bus cycles from low voltage to hi-Z and then low as follows: 0 after VDDXOK is asserted. TRI-STATE when V <sub>DDI</sub> is on and RESETOUT# is asserted. 0 after hardware reset sequence is complete.	(See description at left.)	HIZ	DEP (Note 1)
SDQM[3:0]#	O	Input/Output Mask: SDQM is an input mask signal for write accesses and an output enable signal for read accesses.  SDQM0# masks SDD[7:0]. SDQM1# masks SDD[15:8]. SDQM2# masks SDD[23:16]. SDQM3# masks SDD[31:24].	1	1	DEP (Note 1)
SDRAS#	O	Command Outputs. SDRAS#, SDCAS#, and SDWE# (along with SDCSn#) define the command being sent to the SDRAM rank.	1	1	DEP (Note 1)
SDCAS#	O		1	1	DEP (Note 1)
SDWE#	O		1	1	DEP (Note 1)
SDCLK[2:0]	O	Clock output corresponding to each of the three chip selects. Clock speed is 1/2 SBUS frequency when corresponding SDCSn# is set to SDRAM, 1/4 SBUS frequency when corresponding SDCSn# is set to SMROM.	0	ON	DEP (Note 1) DEP (Note 1)
SDCS[2:0]#	O	Programmable chip selects	1	1	DEP (Note 1)
SDCKE	O	Clock enable for SDRAM	0	1	DEP (Note 1)
SMROMCKE	O	Synchronous Mask ROM Clock Enable. Valid only when ROMSEL=1 and ROMSIZE=0. Must be pulled high if the system is booting from SMROM.  Muxed with GPIO[6]. If ROMSEL and ROMSIZE are configured to boot from synchronous mask ROM, the SMROMCKE signal is selected for the pin coming out of reset; otherwise, GPIO[6] is selected.	1	1	DEP (Note 1)

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
<b>Static Bus (SRAM/IO/PCMCIA/Flash/ROM/LCD) Interface - Common Signals</b>					
RAD[31:0]	O	Address Bus.	UN	UN	LV
RD[31:0]	IO	Data Bus.	0	UN	LV
RBE[3:0]#	O	Byte Enable. RBE0# corresponds to RD[7:0], RBE1# corresponds to RD[15:8], RBE2# corresponds to RD[23:16], RBE3# corresponds to RD[31:24].	1	1	LV
RWE#	O	Write Enable.	1	1	LV
ROE#	O	Output Enable.	1	1	LV
RCS[3:0]#	O	Programmable Chip Selects. RCS[n]# is not used when configured as a PCMCIA device.	1	1	LV
EWAIT#	I	Can be used to stretch the bus access time when enabled. This input is not recognized for chip selects configured as LCD or PCMCIA because these buses have their own wait mechanisms.	IN	IN	LV
<b>PCMCIA</b>					
PREG#	O	Register-only access signal. Muxed with GPIO[204]. GPIO[204] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
PCE[2:1]#	O	Card Enables. Note that the card enables need pull-up resistors. Muxed with GPIO[206:205]. GPIO[206:205] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
POE#	O	Output Enable.	1	1	LV
PWE	O	Write Enable. Muxed with GPIO[207]. GPIO[207] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
PIOR#	O	Read Cycle Indication.	1	1	LV
PIOW#	O	Write Cycle Indication.	1	1	LV
PWAIT#	I	Extend Cycle. Note that this signal should be tied high through a resistor when the PCMCIA interface is not used.	IN	IN	LV
PIOS16#	I	16-bit port select. Note that this signal should be tied high through a resistor when the PCMCIA interface is not used.	IN	IN	LV
<b>LCD Controller Chip Interface</b>					
LCLK	O	Interface Clock.	0	0	LV
LWAIT#	I	Extend Cycle. Note that this signal should be tied high through a resistor when not used.	IN	IN	LV
LRD[1:0]#	O	Read Indicators. Muxed with GPIO[201:200]. GPIO[201:200] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
LWR[1:0]#	O	Write Indicators. Muxed with GPIO[203:202]. GPIO[203:202] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
<b>USB Host</b>					
USBH1P	IO	Positive signal of differential USB host port 1 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.	IN	IN	LV
USBH1M	IO	Negative signal of differential USB host port 1 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.	IN	IN	LV
USBH0P	IO	Positive signal of differential USB host port 0 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.  Muxed with USBDP. USBDP is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
USBH0M	IO	Negative signal of differential USB host port 0 driver. Requires an external 15 kohm pull-down resistor and ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.  Muxed with USBDM. USBDM is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>USB Device</b>					
USBDP	IO	Positive signal of differential USB device driver. Requires a 1.5 kohm pull-up resistor to denote a full speed device. Also requires an external ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.  Muxed with USBH0P. USBDP is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV



Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
USBDM	IO	Negative signal of differential USB device driver. Requires an external ESD protection diode (transient voltage suppressor) to be USB 1.1 compliant.  Termination Note: Requires an external 20 ohm series resistor placed within 0.5 inches of the part.  Muxed with USBH0M. USBDM is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
<b>SSIO</b>					
S0CLK	O	Master only clock output. The speed and polarity of clock edge is programmable.  Muxed with GPIO[209]. GPIO[209] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
S0DIN	I	Serial Data Input. May be tied to S0DOUT to create a single bidirectional data signal.	IN	IN	LV
S0DOUT	O	Serial Data Output. TRI-STATE during a read and thus may be tied to S0DIN to create a single bidirectional data signal.  Muxed with GPIO[208]. GPIO[208] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
S0DEN	O	Enable signal which frames transaction. The polarity is programmable.  Muxed with GPIO[210]. GPIO[210] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>SSI1</b>					
S1CLK	O	Master only clock output. The speed and polarity of clock edge is programmable.  Muxed with ACDO. ACDO is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
S1DIN	I	Serial Data Input. This signal may be tied to S1DOUT to create a single bidirectional data signal.  Muxed with ACBCLK. ACBCLK is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
S1DOUT	O	Serial Data Output. This signal is tristated during a read and thus may be tied to S1DIN to create a single bidirectional data signal.  Muxed with ACSYNC. ACSYNC is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
S1DEN	O	Enable signal which frames transaction. The polarity is programmable. Muxed with ACRST#. ACRST# is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>IrDA</b>					
IRDATX	O	Serial IrDA Output. Muxed with GPIO[211]. GPIO[211] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
IRDARX	I	Serial IrDA input.	IN	IN	LV
IRFIRSEL	O	Output which will signal at which speed the IrDA is currently set. This signal is not necessary for IrDA operation. This pin will be driven high when IrDA is configured for FIR or MIR. This pin will be driven low for SIR mode. Muxed with GPIO[15]. GPIO[15] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>UART0</b>					
U0TXD	O	UART0 Transmit. Muxed with GPIO[212]. GPIO[212] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
U0RXD	I	UART0 Receive.	IN	IN	IN
<b>UART1</b>					
U1TXD	O	UART1 Transmit. Muxed with GPIO[213]. GPIO[213] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
U1RXD	I	UART1 Receive.	IN	IN	LV
<b>UART3</b>					
U3TXD	O	UART3 Transmit. Muxed with GPIO[214]. GPIO[214] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
U3RXD	I	UART3 Receive.	IN	IN	LV
U3CTS#	I	Clear to Send. Muxed with GPIO[9]. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep. <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	NA	NA	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
U3DSR#	I	Data Set Ready. Muxed with GPIO[10]. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	NA	NA	LV
U3DCD#	I	Data Carrier Detect. Muxed with GPIO[11]. GPIO[11] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	NA	NA	LV
U3RI#	I	Ring Indication. Muxed with GPIO[12]. GPIO[12] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pinfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	NA	NA	LV
U3RTS#	O	Request to Send. Muxed with GPIO[13]. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
U3DTR#	O	Data Terminal Ready. Muxed with GPIO[14]. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
<b>Ethernet Controller 0</b>					
N0TXCLK	I	Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100 Mbps and 2.5 MHz when operating at 10 Mbps.	IN	IN	LV
N0TXEN	O	Indicates that the data nibble on N0TXD[3:0] is valid. Muxed with GPIO[24]. GPIO[24] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
N0TXD[3:0]	O	Nibble wide data bus synchronous to N0TXCLK. For each N0TXCLK period in which N0TXEN is asserted, TXD[3:0] will have the data to be accepted by the PHY. While N0TXEN is de-asserted the data presented on TXD[3:0] should be ignored.  Muxed with GPIO[28:25]. GPIO[28:25] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
N0RXCLK	I	Continuous clock that provides the timing reference for the data transfer from the PHY to the MAC. N0RXCLK is sourced by the PHY. The N0RXCLK shall have a frequency equal to 25% of the data rate of the received signal data stream (typically 25 MHz at 100 Mbps and 2.5 MHz at 10-Mbps).	IN	IN	LV
N0RXDV	I	Indicates that a receive frame is in process and that the data on N0RXD[3:0] is valid.	IN	IN	LV
N0RXD[3:0]	I	RXD[3:0] is a nibble wide data bus driven by the PHY to the MAC synchronous with N0RXCLK. For each N0RXCLK period in which N0RXDV is asserted, RXD[3:0] will transfer four bits of recovered data from the PHY to the MAC. While N0RXDV is de-asserted, RXD[3:0] will have no effect on the MAC.	IN	IN	LV
N0CRS	I	N0CRS shall be asserted by the PHY when either transmit or receive medium is non-idle. N0CRS shall be deasserted by the PHY when both the transmit and receive medium are idle. N0CRS is an asynchronous input.	IN	IN	LV
N0COL	I	N0COL shall be asserted by the PHY upon detection of a collision on the medium, and shall remain asserted while the collision condition persists. N0COL is an asynchronous input. The N0COL signal is ignored by the MAC when operating in the full duplex mode.	IN	IN	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
N0MDC	O	N0MDC is sourced by the MAC to the PHY as the timing reference for transfer of information on the N0MDIO signal. N0MDC is an aperiodic signal that has no maximum high or low times. The minimum high and low times for N0MDC will be 160 ns each, and the minimum period for N0MDC will be 400 ns.  Muxed with GPIO[215]. GPIO[215] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
N0MDIO	IO	N0MDIO is the bidirectional data signal between the MAC and the PHY that is clocked by N0MDC.	HIZ	UN	LV
<b>LCD Controller</b>					
LCD_PWM1	O	Pulse Width Modulation Clock 1.	0	0	LV
LCD_PWM0	O	Pulse Width Modulation Clock 0.	0	0	LV
LCD_BIAS	O	Bias Clock.	0	0	LV
LCD_FCK	O	Frame Clock.	0	0	LV
LCD_LCK	O	Line Clock.	0	0	LV
LCD_LEND	O	Line End.	0	0	LV
LCD_PCK	O	Pixel Clock.	0	0	LV
LCD_D[15:0]	O	LCD Data.	0	0	LV
<b>Secure Digital Controller</b>					
SDMS_MS_EN	I	Reserved for future use. Must be tied to ground.	IN	IN	LV
SDMS0_CLK	O	SD Card 0 Interface Clock.	1	0	LV
SDMS0_CMD	I/O	SD Card 0 Half Duplex Command and Response.	HIZ	HIZ	LV
SDMS0_DAT[3:0]	I/O	SD Card 0 Data Bus.	HIZ	HIZ	LV
SDMS1_CLK	O	SD Card 1 Interface Clock.	1	0	LV
SDMS1_CMD	I/O	SD Card 1 Half Duplex Command and Response.	HIZ	HIZ	LV
SDMS1_DAT[3:0]	I/O	SD Card 1 Data Bus.	HIZ	HIZ	LV
<b>I<sup>2</sup>S</b>					
I2SCLK	O	Serial Bit Clock. Muxed with GPIO[30]. GPIO[30] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
I2SWORD	O	Word clock typically configured to the sampling frequency (Fs).  Muxed with GPIO[31]. GPIO[31] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
I2SDI	I	Serial data input sampled on the rising edge of I2SCLK. Note that I2SDI is used as the input for <i>bidirectional</i> operation only, in which case it is used in conjunction with I2SDIO as the output ( <b>i2s_config</b> [PD]=0).  Muxed with GPIO[8]. GPIO[8] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the I <sup>2</sup> S interface for <i>unidirectional</i> operation (I2SDI not used), the GPIO[8] function is available but with the following restrictions: <ul style="list-style-type: none"> <li>When I2SDIO is configured as an <i>input</i>, GPIO[8] can be used only as an output.</li> <li>When I2SDIO is configured as an <i>output</i>, the I<sup>2</sup>S receive function must be disabled if GPIO[8] is to be used as an input.</li> </ul>	NA	NA	LV
I2SDIO	IO	Configurable as input or output. As input data should be presented on rising edge. As output, data will be valid on rising edge.  Muxed with GPIO[29]. GPIO[29] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>AC-Link</b>					
ACSYNC	O	Fixed Rate Sample Sync. Muxed with S1DOUT. ACSYNC is the default signal coming out of hardware reset, runtime reset, and Sleep.	0	0	LV
ACBCLK	I	Serial Data Clock. Muxed with S1DIN. ACBCLK is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
ACDO	O	TDM Output Stream. Muxed with S1CLK. ACDO is the default signal coming out of hardware reset, runtime reset, and Sleep.	0	0	LV
ACDI	I	TDM Input Stream.	IN	IN	LV
ACRST#	O	Codec Reset. Muxed with S1DEN. ACRST# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	0	LV
<b>EJTAG</b>					
TRST#	I	Asynchronous TAP Reset.	IN	IN	LV
TDI	I	Test data input to the instruction or selected data registers. Sampled on the rising edge of TCK.	IN	IN	LV
TDO	O	Test data output from the instruction or data register. Transitions occur on the falling edge (valid on rising edge) of TCK.	HIZ	UN	LV
TMS	I	Control signal for TAP controller. Sampled on the rising edge of TCK.	IN	IN	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
TCK	I	Control clock for updating TAP controller and shifting data through instruction or selected data register.	IN	IN	LV
<b>Test</b>					
TC[3:0]	I	Test clock inputs (not used in typical application). Should be pulled low for normal operation.	IN	IN	LV
TESTEN	I	Test Enable (not used in typical applications). Should be pulled low for normal operation.	IN	IN	LV
<b>Reserved</b>					
RESVD[5:4]	I	Reserved, should be pulled low.	IN	IN	LV
RESVD[3]	O	Reserved, should be left open (not connected).	0	0	LV
RESVD[2:0]	I	Reserved, should be pulled low.	IN	IN	LV
<b>GPIO</b>					
GPIO[1:0]	IOZ	General Purpose IO.	HIZ	DEP (Note 2)	LV
GPIO[3:2]	IOZ	General Purpose IO. Muxed with EXTCLK[1:0]. GPIO[3:2] are the default signals coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[4]	IOZ	General Purpose IO. Can be configured as DMA_REQ0.	HIZ	DEP (Note 2)	LV
GPIO[5]	IOZ	General Purpose IO. Can be configured as DMA_REQ1.	HIZ	DEP (Note 2)	LV
GPIO[6]	IOZ	General Purpose IO. Muxed with SMROMCKE. If ROMSEL and ROMSIZE are configured to boot from synchronous mask ROM, the SMROMCKE signal is selected for the pin coming out of reset; otherwise, GPIO[6] is selected.	HIZ	DEP (Note 2)	LV
GPIO[7]	IOZ	General Purpose IO.	HIZ	DEP (Note 2)	LV
GPIO[8]	IOZ	General Purpose IO. Muxed with I2SDI. GPIO[8] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the I <sup>2</sup> S interface for <i>unidirectional</i> operation (I2SDI not used), the GPIO[8] function is available but with the following restrictions: <ul style="list-style-type: none"> <li>When I2SDIO is configured as an <i>input</i>, GPIO[8] can be used only as an output.</li> <li>When I2SDIO is configured as an <i>output</i>, the I<sup>2</sup>S receive function must be disabled if GPIO[8] is to be used as an input.</li> </ul>	HIZ	DEP (Note 2)	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
GPIO[9]	IOZ	General Purpose IO. Muxed with U3CTS#. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	HIZ	DEP (Note 2)	LV
GPIO[10]	IOZ	General Purpose IO. Muxed with U3DSR#. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	HIZ	DEP (Note 2)	LV
GPIO[11]	IOZ	General Purpose IO. Muxed with U3DCD#. GPIO[11] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	HIZ	DEP (Note 2)	LV
GPIO[12]	IOZ	General Purpose IO. Muxed with U3RI#. GPIO[12] is the default signal coming out of hardware reset, runtime reset, and Sleep.  <b>System Note:</b> For systems that use the UART3 interface without the optional modem control signals ( <b>sys_pfunc</b> [UR3]=0), the modem status interrupts must be disabled ( <b>uart3_inten</b> [MIE]=0) to avoid false UART3 interrupts when using GPIO[9], GPIO[10], GPIO[11], or GPIO[12] as an input.	HIZ	DEP (Note 2)	LV
GPIO[13]	IOZ	General Purpose IO. Muxed with U3RTS#. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[14]	IOZ	General Purpose IO. Muxed with U3DTR#. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[15]	IOZ	General Purpose IO. Muxed with IRFIRSEL. GPIO[15] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[23:16]	IOZ	General Purpose IO.	HIZ	DEP (Note 2)	LV



Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
GPIO[24]	IOZ	General Purpose IO. Muxed with N0TXEN. GPIO[24] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[28:25]	IOZ	General Purpose IO. Muxed with N0TXD[3:0]. GPIO[28:25] are the default signals coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[29]	IOZ	General Purpose IO. Muxed with I2SDIO. GPIO[29] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[30]	IOZ	General Purpose IO. Muxed with I2SCLK. GPIO[30] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[31]	IOZ	General Purpose IO. Muxed with I2SWORD. GPIO[31] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	DEP (Note 2)	LV
GPIO[200]	IOZ	General Purpose IO. Muxed with LRD0#. GPIO[200] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[201]	IOZ	General Purpose IO. Muxed with LRD1#. GPIO[201] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[202]	IOZ	General Purpose IO. Muxed with LWR0#. GPIO[202] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[203]	IOZ	General Purpose IO. Muxed with LWR1#. GPIO[203] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[204]	IOZ	General Purpose IO. Muxed with PREG#. GPIO[204] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[205]	IOZ	General Purpose IO. Muxed with PCE1#. GPIO[205] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[206]	IOZ	General Purpose IO. Muxed with PCE2#. GPIO[206] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[207]	IOZ	General Purpose IO. Muxed with PWE#. GPIO[207] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[208]	IOZ	General Purpose IO. Muxed with S0DOUT. GPIO[208] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[209]	IOZ	General Purpose IO. Muxed with S0CLK. GPIO[209] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
GPIO[210]	IOZ	General Purpose IO. Muxed with S0DE. GPIO[210] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[211]	IOZ	General Purpose IO. Muxed with IRDATX. GPIO[211] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[212]	IOZ	General Purpose IO. Muxed with U0TXD. GPIO[212] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[213]	IOZ	General Purpose IO. Muxed with U1TXD. GPIO[213] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[214]	IOZ	General Purpose IO. Muxed with U3TXD. GPIO[214] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[215]	IOZ	General Purpose IO. Muxed with N0MDC. GPIO[215] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
<b>External Clocks</b>					
EXTCLK[1:0]	O	General-purpose external clocks. One of these clock outputs can be used as an oversampled audio clock (AUDCLK or MCLK) output with I <sup>2</sup> S port as it is synchronous to I2SCLK and I2SWORD. Typically it should be programmed to 128*Fs, 256*Fs, 384*Fs or 512*Fs for this application.  Muxed with GPIO[3:2]. GPIO[3:2] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
<b>System DMA Requests</b>					
DMA_REQ0 (GPIO[4])	I	GPIO[4] can be configured as an external, system DMA request input.	HIZ	HIZ	LV
DMA_REQ1 (GPIO[5])	I	GPIO[5] can be configured as an external, system DMA request input.	HIZ	HIZ	LV
<b>System Clocks and Reset</b>					
XTI12	I	Internally compensated 12-MHz (typical) crystal input.  Termination Note: The termination depends on the application as follows:  Crystal—Connect crystal between XTI12 and XTO12.  Overdriven—Connect to external 12-MHz clock source and drive complementary to XTO12.			

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
XTO12	O	Internally compensated 12 MHz (typical) crystal output. Termination Note: The termination depends on the application as follows: Crystal—Connect crystal between XT112 and XTO12. Overdriven—Connect to external 12 MHz clock source and drive complementary to XT112.			
XTI32	I	Internally compensated 32.768 kHz (typical) crystal input. Termination Note: The termination depends on the application as follows: Crystal—Connect crystal between XTI32 and XTO32. Overdriven—Connect to external 32.768 kHz clock source through a series 10 kom resistor and drive complementary to XTO32. Not used—Connect to $V_{DDX}$ .			
XTO32	O	Internally compensated 32.768 kHz (typical) crystal output Termination Note: The termination depends on the application as follows: Crystal—Connect crystal between XTI32 and XTO32. Overdriven—Connect to external 32.768 kHz clock source through a series 10 kohm resistor and drive complementary to XTI32. Not used—Connect to $V_{DDX}$ .			
RESETIN#	I	CPU Reset Input.	IN	IN	LV
RESETOUT#	O	Buffered output of CPU reset input (RESETIN#).	0	0	0
ROMSEL	I	Determines if boot is from ROM or SMROM. ROMSEL should be terminated appropriately as these signals should not change during runtime.	IN	IN	LV
ROMSIZE	I	Latched at the rising edge of reset to determine if ROM width is 16 or 32 bits. ROMSIZE should be terminated appropriately as these signals should not change during runtime.	IN	IN	LV
<b>Power Management</b>					
PWR_EN	O	Power enable output. This signal is intended to be used as the regulator enable for $V_{DDI}$ (core power).	1	1	0
VDDXOK	I	Input to signal that $V_{DDX}$ (and $V_{DDY}$ on power-up) is stable.	IN	IN	LV

Table 10-3. Signal Descriptions (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run Time	
<b>Power/Ground</b>					
V <sub>DDI</sub>	P	Internal core voltage. <b>Note:</b> Follow the power supply layout guidelines in Section 11.9.2 "Decoupling Recommendations" on page 284.			
V <sub>DDX</sub>	P	External I/O voltage. <b>Note:</b> Follow the power supply layout guidelines in Section 11.9.2 "Decoupling Recommendations" on page 284.			
V <sub>DDY</sub>	P	External I/O voltage for SDRAM only. <b>Note:</b> Follow the power supply layout guidelines in Section 11.9.2 "Decoupling Recommendations" on page 284.			
VSEL	I	External SDRAM voltage type. 0 2.5V 1 3.3V	IN	IN	LV
V <sub>SS</sub>	G	Ground			
XPWR12	P	12 MHz (typical) oscillator and PLL power. <b>Note:</b> Connect to V <sub>DDX</sub> through a 10 ohm resistor. In addition a 22 $\mu$ F capacitor in parallel with a 0.01 $\mu$ F capacitor should be placed from this pin to XAGND12.			
XAGND12	G	12 MHz (typical) oscillator and PLL ground.			
XPWR32	P	32.768 kHz (typical) oscillator and PLL power. Because XPWR32 powers other circuitry also, it should be connected even if the oscillator is not used. <b>Note:</b> Connect to V <sub>DDX</sub> through a 10 ohm resistor. In addition a 22 $\mu$ F capacitor in parallel with a 0.01 $\mu$ F capacitor should be placed from this pin to XAGND32.			
XAGND32	G	32.768 kHz (typical) oscillator and PLL ground.			

Note 1. Depends on **sys\_powerctrl[SSY]**. If SSY=0, the SDRAM interface signals hold their last values if driven (LV); if SSY=1, the SDRAM signals are forced to their hardware reset values (as shown in the 'Hardware' column).

Note 2. Depends on **sys\_trioutrd** and **sys\_outputset**. During a runtime reset, **sys\_pinfunc** returns to its default value, but the GPIO control registers **sys\_trioutrd** and **sys\_outputset** remain unchanged.

# 11 Electrical and Thermal Specifications

This chapter provides electrical specifications for the Au1100 processor, including the following:

- Absolute Maximum Ratings
- Thermal Characteristics
- DC Parameters
- AC Parameters
- Power-up, Reset, Sleep, and Idle Timing
- External Clock Specifications
- Crystal Specifications
- System Design Considerations

## 11.1 Absolute Maximum Ratings

Table 11-1 shows the absolute maximum ratings for the Au1100 processor. These ratings are stress ratings, operating at or beyond these ratings for extended periods of time may result in damage to the Au1100 processor.

Unless otherwise designated all voltages are relative to  $V_{SS}$ .

**Table 11-1. Absolute Maximum Ratings**

Parameter	Description	Min	Max	Unit
$V_{DDI}$	Core Voltage	$V_{SS} - 0.5$	1.32	V
$V_{DDX}$	I/O Voltage	$V_{SS} - 0.5$	3.6	V
$V_{DDY}$	I/O Voltage	$V_{SS} - 0.5$	3.6	V
XPWR12, XPWR32	Oscillator Voltage	$V_{SS} - 0.5$	3.6	V
$V_{IN}$	Voltage Applied to Any Pin	$V_{SS} - 0.5$	$V_{DDX,Y} + 0.5$	V
$T_{case}$ Commerical	Package Operating Temperature	0	85	°C
$T_{CASE}$ Industrial (333 MHz part only)	Package Operating Temperature	-40	100	°C
$T_S$	Storage Temperature	-40	125	°C

### 11.1.1 Undershoot

The minimum DC voltage on input or I/O pins is  $-0.5\text{V}$ . However, during voltage transitions, the device can tolerate undershoot to  $-2.0\text{V}$  for up to  $20\text{ ns}$ , as shown in Figure 11-1.

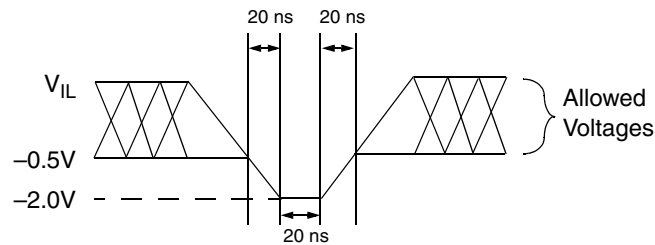


Figure 11-1. Voltage Undershoot Tolerances for Input and I/O Pins

### 11.1.2 Overshoot

The maximum DC voltage on input or I/O pins is  $(V_{DDX,Y} + 0.5)\text{ V}$ . However, during voltage transitions, the device can tolerate overshooting  $V_{DDX,Y}$  to  $(V_{DDX,Y} + 2.0)\text{ V}$  for up to  $20\text{ ns}$ , as shown in Figure 11-2.

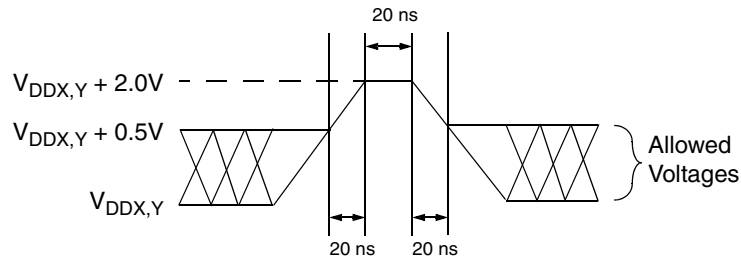


Figure 11-2. Voltage Overshoot Tolerances for Input and I/O Pins

## 11.2 Thermal Characteristics

Table 11-2 shows the thermal characteristics for the Au1100 processor.

Table 11-2. Thermal Characteristics

Parameter	Description	Value	Unit
$\Theta_{JA}$	Thermal resistance from device junction to ambient.	37.8 (Note 1)	$^{\circ}\text{C}/\text{W}$
$Y_{JT}$	Thermal characterization parameter measured from device junction to top center of package. (See JESD51-2, Sec. 4.)	5.0	$^{\circ}\text{C}/\text{W}$

Note 1. Measured without forced air—natural convection only.

### 11.3 DC Parameters

Table 11-3 shows the DC parameters for the Au1100 processor. Unless otherwise designated all voltages are relative to  $V_{SS}$ .

The operating requirements for the power supply voltages ( $V_{DDX}$ ,  $V_{DDI}$ , and  $V_{DDY}$ ) are given in the sections describing the DC characteristics for the different operating frequencies, beginning with Section 11.3.1 "Power and Voltage for 333, 400, and 500 MHz Rated Parts" on page 264.

**Table 11-3. DC Parameters**

Parameter	Description	Min	Nom	Max	Unit
$V_{IHx}$	Input High Voltage (except SDRAM signals)	2.4			V
$V_{ILx}$	Input Low Voltage (except SDRAM signals)			$0.2 * V_{DDX}$	V
$V_{OHx} @ 2 \text{ mA}$	Output High Voltage (except SDRAM signals)	$0.8 * V_{DDX}$			V
$V_{OLx} @ 2 \text{ mA}$	Output Low Voltage (except SDRAM signals)			$0.2 * V_{DDX}$	V
$V_{IHY} (VSEL=1)$	SDRAM Input High Voltage	2.4			V
$V_{IHY} (VSEL=0)$		$0.8 * V_{DDY}$			V
$V_{ILY}$	SDRAM Input Low Voltage			$0.2 * V_{DDY}$	V
$V_{OHY} @ 2 \text{ mA}$	SDRAM Output High Voltage	$0.8 * V_{DDY}$			V
$V_{OLY} @ 2 \text{ mA}$	SDRAM Output Low Voltage			$0.2 * V_{DDY}$	V
$I_I$	Input Leakage Current			5	$\mu\text{A}$
$C_{IN}$	Input Capacitance (Note 1)		5		pF
$I_{XPWR12}$ (Note 2)	XPWR12 Current		1	3	mA
$I_{XPWR32}$ (Note 2)	XPWR32 Current		1	3	mA

Note 1. This parameter is by design and not tested.

Note 2. Does not apply during Sleep.

### 11.3.1 Power and Voltage for 333, 400, and 500 MHz Rated Parts

The tables that follow give the voltage and power parameters for the individual MHz rated parts.

**Table 11-4. Voltage and Power Parameters for 333 MHz Part**

Parameter	Min	Typ	Max	Unit
$V_{DDI}$	1.12	1.22	1.32	V
$V_{DDX}$ , XPWR12 (Note 1), XPWR32 (Note 1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=0)	2.4	2.5	2.6	V
Power: $V_{DDI}$		155	300 (Note 2)	mW
Power: $V_{DDX}$		20	50 (Note 2)	mW
Power: $V_{DDY}$ where $V_{DDY}=2.5V$		25	135 (Note 2)	mW
IDLE Power (Note 3)		117		mW
Sleep Current ( $V_{DDI} = V_{SS}$ )			50	$\mu A$

Note 1. XPWR12 and XPWR32 should be connected to  $V_{DDX}$ . For a description of this circuit connection, see the entries for XPWR12 and XPWR32 in Table 10-3 on page 246.

Note 2. While the maximum power numbers should be used when specifying a regulator for a system, the numbers are well above the typical power consumption because none of the power-saving design features (such as IDLE, or the automatic SBUS divider) are enabled. Note that because the particular application software and external loading affect the power consumption on a given system design, certain conditions may exist which could cause the maximum power consumption to be different than shown.

Note 3. *IDLE Power* is the power measured when the processor core is in the IDLE0 state. (IDLE0 maintains cache coherency by snooping the SBUS; IDLE1 does not snoop the bus. Because caches are turned off during the IDLE1 state, IDLE1 consumes less power than IDLE0.) Typically the IDLE state is entered during an operating system's wait loop in which the core has no processes to run. While the processor core is in IDLE, clocks to the core are gated off; however, all registers retain their values, and the peripherals, DMA engine, and the interrupts remain active so that the system is still functional.



Table 11-5. Voltage and Power Parameters for 400 MHz Part

Parameter	Min	Typical	Max	Unit
$V_{DDI}$	1.12	1.22	1.32	V
$V_{DDX}$ , XPWR12 (Note 1), XPWR32 (Note 1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=0)	2.4	2.5	2.6	V
Power: $V_{DDI}$		185	350 (Note 2)	mW
Power: $V_{DDX}$		25	50 (Note 2)	mW
Power: $V_{DDY}$ where $V_{DDY}=2.5V$		40	165 (Note 2)	mW
IDLE Power (Note 3)		126		mW
Sleep Current ( $V_{DDI} = V_{SS}$ )			50	uA

- Note 1. XPWR12 and XPWR32 should be connected to  $V_{DDX}$ . For a description of this circuit connection, see the entries for XPWR12 and XPWR32 in Table 10-3 on page 246.
- Note 2. While the maximum power numbers should be used when specifying a regulator for a system, the numbers are well above the typical power consumption because none of the power-saving design features (such as IDLE, or the automatic SBUS divider) are enabled. Note that because the particular application software and external loading affect the power consumption on a given system design, certain conditions may exist which could cause the maximum power consumption to be different than shown.
- Note 3. *IDLE Power* is the power measured when the processor core is in the IDLE0 state. (IDLE0 maintains cache coherency by snooping the SBUS; IDLE1 does not snoop the bus. Because caches are turned off during the IDLE1 state, IDLE1 consumes less power than IDLE0.) Typically the IDLE state is entered during an operating system's wait loop in which the core has no processes to run. While the processor core is in IDLE, clocks to the core are gated off; however, all registers retain their values, and the peripherals, DMA engine, and the interrupts remain active so that the system is still functional.

Table 11-6. Voltage and Power Parameters for 500 MHz Part

Parameter	Min	Typical	Max	Unit
$V_{DDI}$	1.12	1.22	1.32	V
$V_{DDX}$ , XPWR12 (Note 1), XPWR32 (Note 1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=1)	3.0	3.3	3.6	V
$V_{DDY}$ (VSEL=0)	2.4	2.5	2.6	V
Power: $V_{DDI}$		316	435 (Note 2)	mW
Power: $V_{DDX}$		32	55 (Note 2)	mW
Power: $V_{DDY}$ where $V_{DDY}=2.5V$		52	215 (Note 2)	mW
IDLE Power (Note 3)		150		mW
Sleep Current ( $V_{DDI} = V_{SS}$ )			50	uA

Note 1. XPWR12 and XPWR32 should be connected to  $V_{DDX}$ . For a description of this circuit connection, see the entries for XPWR12 and XPWR32 in Table 10-3 on page 246.

Note 2. While the maximum power numbers should be used when specifying a regulator for a system, the numbers are well above the typical power consumption because none of the power-saving design features (such as IDLE, or the automatic SBUS divider) are enabled. Note that because the particular application software and external loading affect the power consumption on a given system design, certain conditions may exist which could cause the maximum power consumption to be different than shown.

Note 3. *IDLE Power* is the power measured when the processor core is in the IDLE0 state. (IDLE0 maintains cache coherency by snooping the SBUS; IDLE1 does not snoop the bus. Because caches are turned off during the IDLE1 state, IDLE1 consumes less power than IDLE0.) Typically the IDLE state is entered during an operating system's wait loop in which the core has no processes to run. While the processor core is in IDLE, clocks to the core are gated off; however, all registers retain their values, and the peripherals, DMA engine, and the interrupts remain active so that the system is still functional.

## 11.4 AC Parameters

This section describes the AC parameters for I/O devices in the Au1100 processor. Each class of output signal has different capacitive loads. As the capacitance on the load increases the propagation delay will increase. These specifications assume the maximum capacitive load to be 50 pF for all I/O signals other than the SDRAM interface.

The timing of those signals which have synchronous relationships or have a defined requirement are given. The timing diagrams are shown to illustrate the timing only and should not necessarily be interpreted as the functional timing of the port.

It is assumed that the timing and/or functionality of the protocol related to the port is adhered to by the external system. The protocol timing is not necessarily presented here and the appropriate section or specification should be referenced for complete functional timing parameters.

Timing measurements are made from 50% threshold to 50% threshold.

Certain timing parameters are based off of the internal System Bus (SBUS) clock. When this is the case the symbol  $T_{sys}$  is used.  $T_{sys}$  is defined in nanoseconds as:

$$T_{sys} = SD/CPU$$

The symbol CPU should be interpreted as the CPU clock speed in MHz as set by the CPU PLL. See Section 7.1 "Clocks" on page 194 for details. The symbol SD is the SBUS divider. See Section 7.4 "Power Management" on page 214 for details.

### 11.4.1 SDRAM Timing and Loading

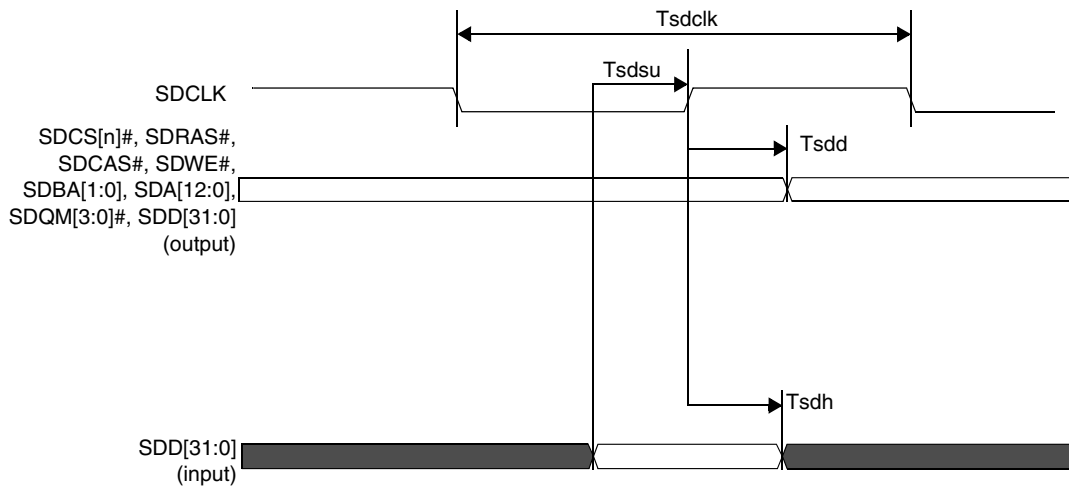
The SDRAM controller loading limits are as follows:

- SDRAM outputs *excluding the clocks and chip-selects* can support a maximum capacitive load of 35 pF (six 5 pF gate loads and 5 pF representing the trace).
- Each *clock* and each *chip-select* supports a maximum capacitive load of 15 pF (two 5 pF gate loads and 5 pF representing the trace).

The SDRAM is a high speed interface. Reflection and propagation delays should be accounted for in the system design. As a general rule of thumb, unterminated etches should be kept to 6 inches or less.

**Table 11-7. SDRAM Controller Interface**

Signal	Symbol	Parameter	Min	Max	Unit
SDCLK[n]	Tsdclk	SDCLK[n] Clock Cycle	$2 \times T_{sys}$		ns
SDCS[n]#, SDRAS#, SDCAS#, SDWE, SDBA[1:0], SDA[12:0], SDQM[3:0]#, SDD[31:0] (output)	Tsdd	Delay from SDCLK[n]	$\frac{T_{sdclk}}{4} - 1.5$	$\frac{T_{sdclk}}{4} + 2$	ns
SDD[31:0] (input)	Tdsu	Data setup to SDCLK[n]	3		ns
SDD[31:0] (input)	Tsdh	Data hold from SDCLK[n]	2		ns



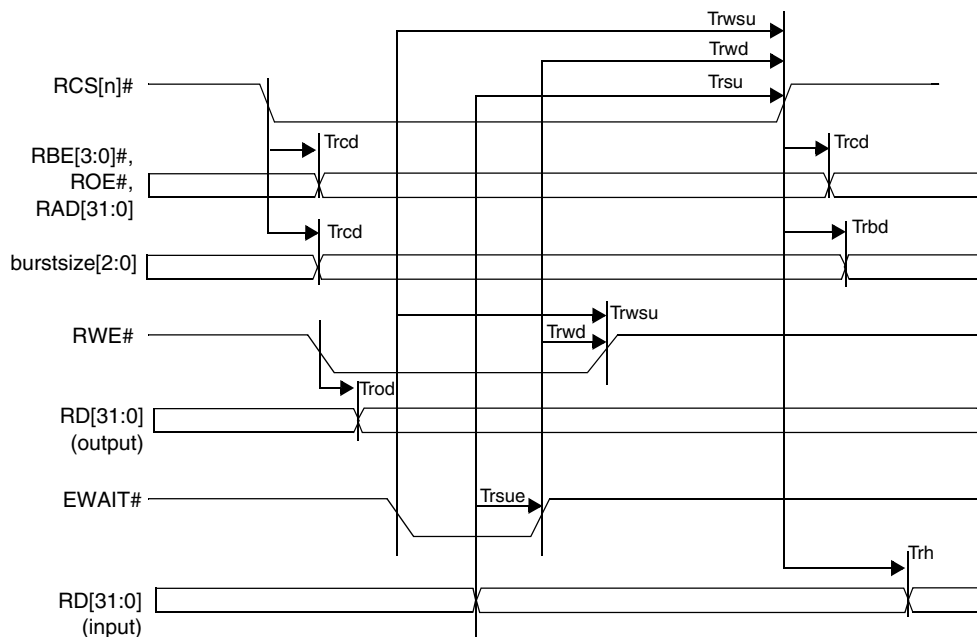
**Figure 11-3. SDRAM Timing**

### 11.4.2 Asynchronous Static Bus Controller Timing

The timing presented in registers `mem_sttmem` are not presented here. The parameters in these registers are presented in a certain number of clock cycles and are accurate to within  $\pm 2$  ns.

**Table 11-8. Static RAM, I/O Device and Flash Timing (Asynchronous Mode)**

Signal	Symbol	Parameter	Min	Max	Unit
RBE[3:0]#, ROE#, RAD[31:0], burstsize[2:0]	Trcd	Delay from RCS[n]#.	-2	+2	ns
RD[31:0] (read)	Trsu	Data setup to RCS[n]#. Note that Trsu does not apply when EWAIT# is used to extend the cycle.	15		ns
RD[31:0] (read)	Trsue	Data setup to EWAIT#. Note that Trsue applies only when EWAIT# is used to extend the cycle.	0		ns
RD[31:0] (read)	Trh	Data hold from RCS[n]#.	0		ns
RD[31:0] (write)	Trod	Delay from RWE# to data out.	-2	2	ns
EWAIT#	Trwsu	EWAIT# setup to RCS[n]# for reads, or RWE# for writes. If EWAIT# does not meet this setup time the cycle will not be held.	$3 * T_{sys} + 15$		ns
RCS[n]# (reads), RWE# (writes)	Trwd	Delay from EWAIT#.	$2 * T_{sys}$	$3 * T_{sys} + 15$	
burstsize[2:0]	Trbd	Delay from RCS[n]#.		$T_{sys} + 2$	



**Figure 11-4. Static RAM, I/O Device and Flash Timing (Asynchronous Mode)**

Table 11-9. PCMCIA Timing

Signal	Symbol	Parameter	Min	Max	Unit
PREG#, RAD[31:0], RD[31:0] (output)	Tpcd	Delay from PCE[n]#	-2	+2	ns
PIOS16#	Tpios	PIOS16# setup to PIOR#, PIOW#	4 * Tsys + 15		ns
PIOS16#	Tpioh	PIOS16# hold from PIOR#, PIOW#	0		ns
ROE#	Tpoed	ROE# delay from POE#, PIOR#	-2	+2	ns
RD[15:0] (input)	Tpsu	Data setup to POE#, PIOR#. Note that Tpsu does not apply when PWAIT# is used to extend the cycle.	Tsys + 15		ns
RD[15:0] (input)	Tpsup	Data setup to PWAIT#. Note that Tpsup applies only when PWAIT# is used to extend the cycle.	0		ns
RD[31:0]	Tph	Data hold from POE#, PIOR#	0		ns
PWAIT#	Tpwsu	PWAIT# setup to POE#, PWE#, PIOR#, PIOW# If PWAIT# does not meet this setup time the cycle will not be held	4 * Tsys + 15		ns
POE#, PWE#, PIOR#, PIOW#	Tpwd	POE#, PWE#, PIOR#, PIOW# delay from PWAIT#.	3 * Tsys	4 * Tsys + 15	ns

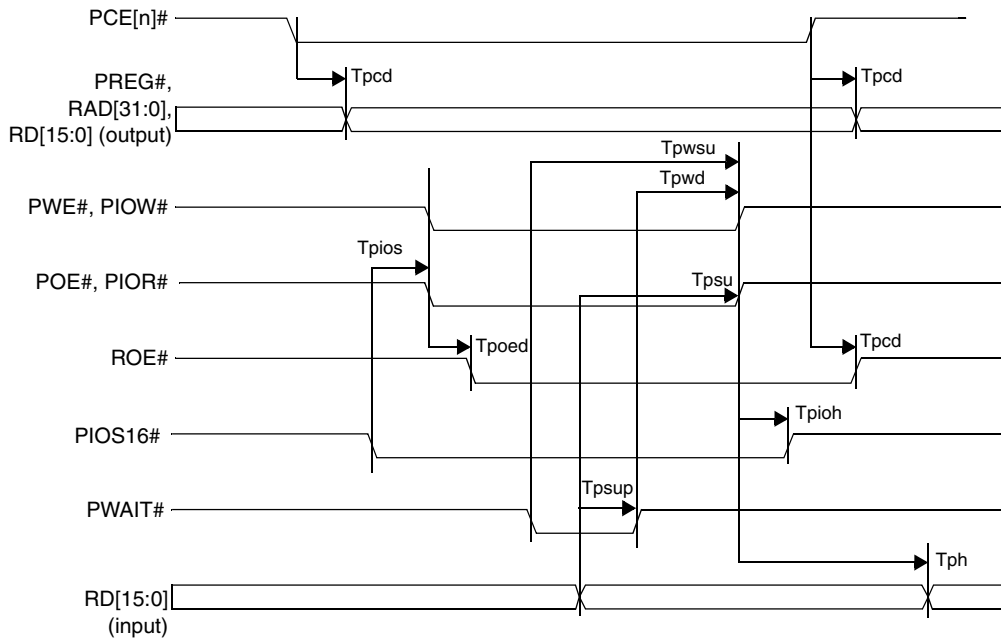


Figure 11-5. PCMCIA Host Adapter Timing

### 11.4.3 Synchronous Static Bus Timing

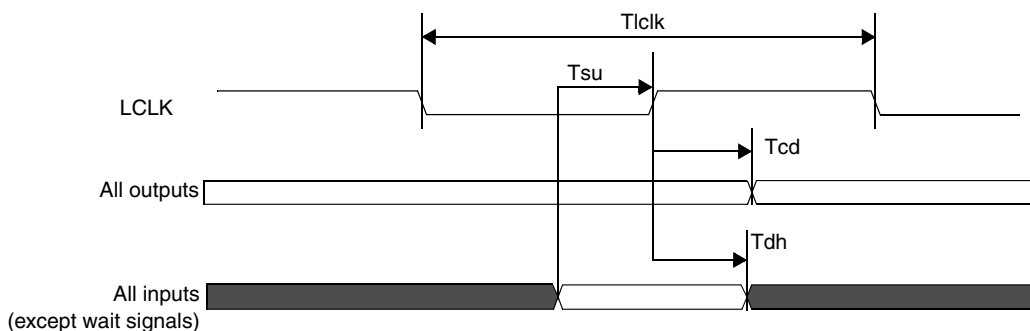
When the static bus is operating in synchronous mode all timing is referenced with respect to LCLK.

**Table 11-10. Synchronous Static Bus**

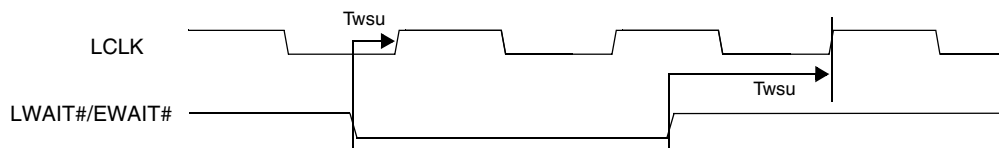
Signal	Symbol	Parameter	Min	Max	Unit
LCLK	Tlclk	LCLK clock period This parameter is programmed in <b>mem_stcfg0</b> [DIV].	MAX[30, Tsys * 2]	Tsys * 16	ns
All outputs (Note 1)	Tcd	Delay in output change from LCLK	Tsys	Tsys + 15	ns
All inputs (Note 2) (except wait signals)	Tsu	Data setup to LCLK	15		ns
	Tdh	Data hold from LCLK	10		ns
LWAIT# and EWAIT#	Twsu	Wait signal setup to LCLK for recognition of a state change in LWAIT#/EWAIT#.	Tsys + 15		ns

Note 1. Output signals: RAD[31:0], RD[31:0], RBE[3:0]#, ROE#, RWE#, RCS[3:0]#, LRD[1:0]#, LWR[1:0]#.

Note 2. Input signals: RD[31:0] (or RD[15:0] for 16-bit interfaces).



**Figure 11-6. Static RAM, I/O Device and Flash Timing (Synchronous Mode)**



**Figure 11-7. Wait Signal Recognition Timing for the Synchronous Static Bus**

## 11.4.4 GPIO Input Timing Requirements

### 11.4.4.1 GPIO Input Edge Rate

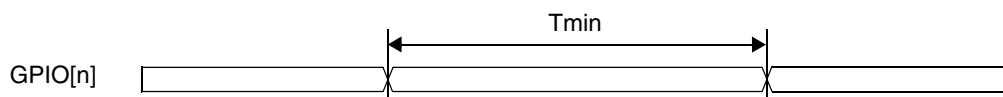
For level-sensitive GPIO inputs, edge rates as slow as 5 ms can be used. Note that no hysteresis is used on the inputs so for edge-sensitive inputs (such as clocks and edge-triggered interrupts) use a 20 ns (or faster) edge rate to ensure that noise does not cause false edges as the signal transitions through the threshold region.

### 11.4.4.2 GPIO Interrupt Timing

For system designs using GPIO signals as level-triggered interrupts, the signal level must be stable for at least 10 ns in order for a signal state change to be detected. See Table 11-11 and Figure 11-8.

**Table 11-11. GPIO Timing for Interrupts**

Signal	Symbol	Parameter	Min	Max	Unit
GPIO[n]	Tmin	Minimum high or low time for interrupt. The level is programmable. This timing reflects the minimum active period for the level programmed.	10		ns



**Figure 11-8. GPIO Interrupt Timing**



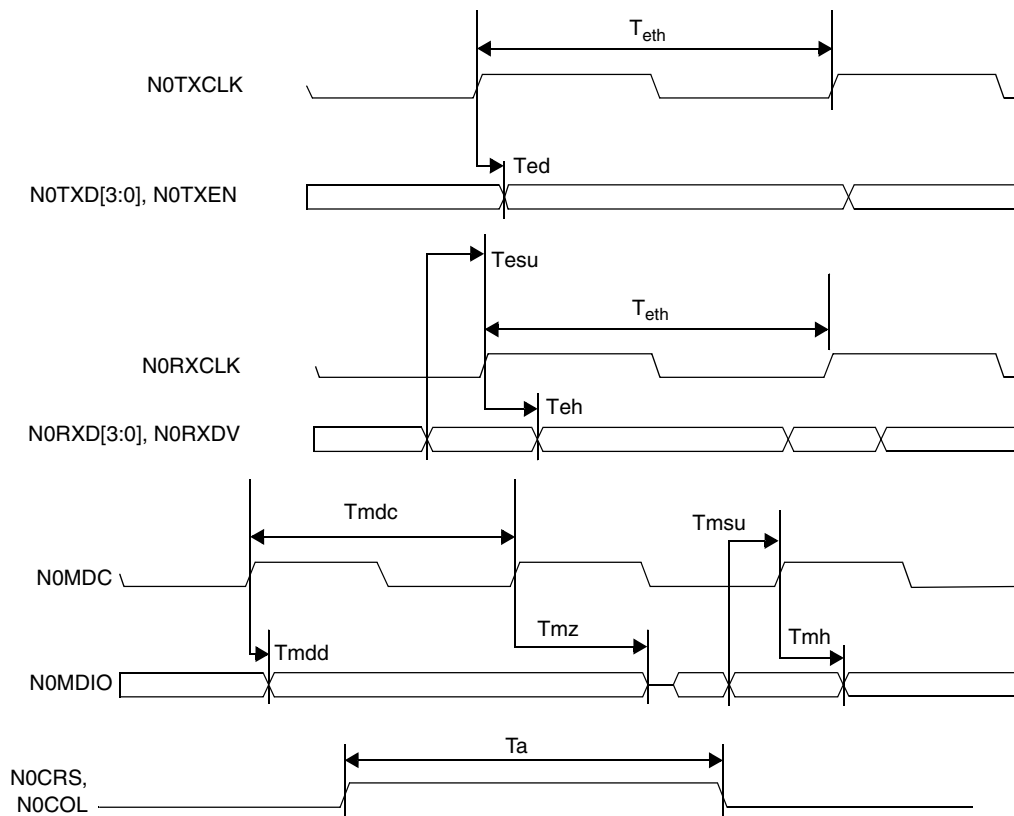
### 11.4.5 Peripheral Timing

This section contains the electrical timing specifications for the integrated peripherals.

#### 11.4.5.1 Ethernet MII Timing

**Table 11-12. Ethernet MII Timing**

Signal	Symbol	Parameter	Min	Max	Unit
N0TXCLK N0RXCLK	$T_{eth}$	Ethernet transmit/receive clock cycle time (25% of data rate)	40 ±100 ppm (100 Mbps) 400 ±100 ppm (10 Mbps)		ns
		Ethernet transmit/ receive clock duty cycle	35	65	%
N0TXEN, N0TXD[3:0]	$T_{ed}$	Delay from TXCLK to TXEN, TXD[3:0]	0	25	ns
N0RXD[3:0] N0RXDV	$T_{esu}$	Setup time before RXCCLK for RXD, and RXDV	10		ns
	$T_{eh}$	Hold time from RXCLK for RXD, and RXDV	10		ns
N0MDC	$T_{mdc}$	MDC cycle time	SBUS Clock / 160		
		MDC duty cycle	40	60	%
N0MDIO	$T_{mdd}$	Delay from MDC to MDIO	0	300	ns
	$T_{msu}$	Setup time before MDC for MDIO	10		ns
	$T_{mh}$	Hold time from MDC for MDIO	10		ns
	$T_{mz}$	Delay from MDC to MDIO TRI-STATE	0	300	ns
N0CRS, N0COL	$T_a$	Minimum active time			



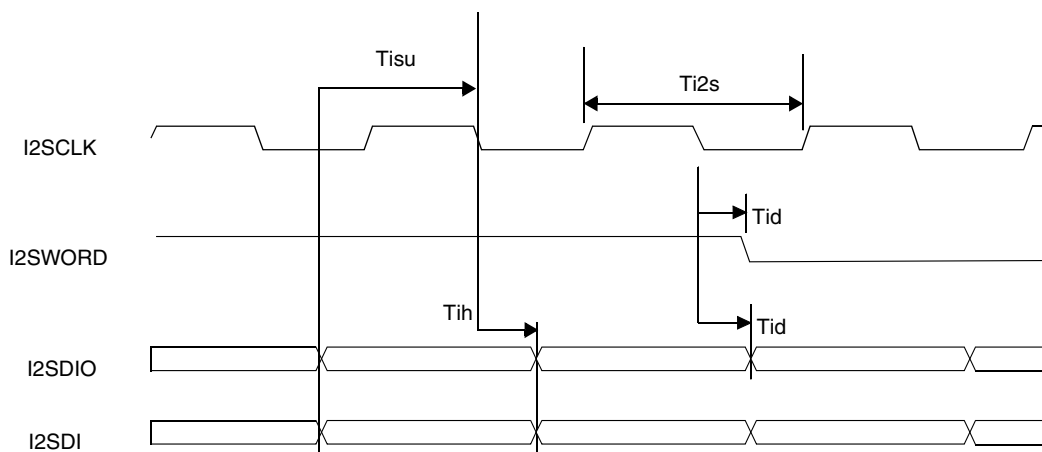
**Figure 11-9. Ethernet MII Timing Diagram**

11.4.5.2 I<sup>2</sup>S TimingTable 11-13. I<sup>2</sup>S Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
I2SCLK	Ti2s	I <sup>2</sup> S interface clock cycle time	40		ns
		I <sup>2</sup> S clock duty cycle	40	60	%
I2SDI, I2SDIO, I2SWORD	Tid	Delay from I2SCLK to I2SDIO and I2SWORD on output (I2SDIO programmed as output)	0	10	ns
	Tisu	Setup before I2SCLK on input (I2SDIO programmed as input)	20		ns
	Tih	Hold after I2SCLK on input (I2SDIO programmed as input)	0		ns

**Note:** Note that I2SDI and I2SDIO (as an input) are shown to have a 0 ns hold time relative to the falling edge. This design allows for the data source to transition data from the falling edge to the next data value.

I2SDIO input and output timing is shown on the same signal. In practice, the signal direction can be programmed as only one or the other.

Figure 11-10. I<sup>2</sup>S Timing Diagram

11.4.5.3 AC97 Timing

Table 11-14. AC-Link Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
ACBCLK	Tabc	AC97 bit clock cycle time	12.288 (typical)		MHz
	Tabh	AC97 bit clock high time	36	45	ns
	Tabl	AC97 bit clock low time	36	45	ns
ACSYNC	Tacs	AC97 sync cycle	48 (typical)		kHz
	Tacsh	AC97 sync high time	1.3 (typical)		µs
	Tacsl	AC97 sync low time	19.5 (typical)		µs
ACSYNC ACDO ACDI	Tad	Delay from ACBCLK to ACSYNC and ACDO on output		15	ns
	Tasu	Setup before ACBCLK for ACDI	10		ns
	Tah	Hold after ACBCLK for ACDI	10		ns

**Note:** ACRST# is an asynchronous signal controlled by software through the register `ac97_config`. It has no relationship to the other AC97 signals.

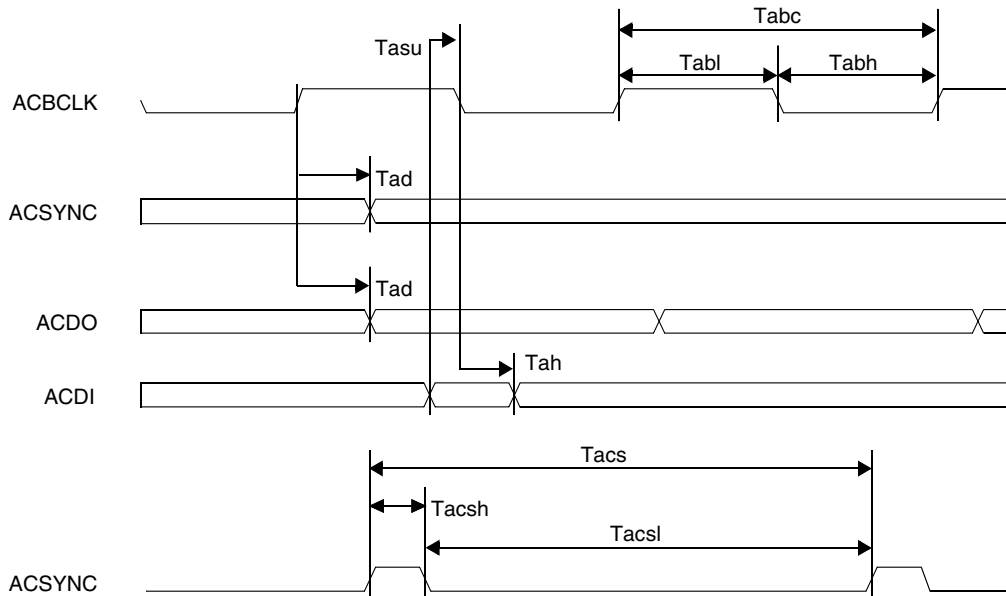


Figure 11-11. AC-Link Timing Diagram

## 11.4.5.4 SSI Timing

Table 11-15. Synchronous Serial Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
SnCLK	Tclk	Clock Period. This period is programmable. See Section 6.8 "SSI Interfaces" on page 160 for more information.	200		ns
SnCLK	Tscd	SnDEN to clock delay		Tclk + 10 ns	
SnDEN	Tsed	SnCLK to SnDEN delay		Tclk + 10 ns	
SnDIN	Tssu	SnDIN setup to active edge of SnCLK	30		ns
SnDIN	Tsh	SnDIN hold from active edge of SnCLK	10		ns
SnDOUT	Tsd	SnDOUT delay from inactive edge of SnCLK		20	ns

**Note:** The timing diagrams shown are for rising edge active SnCLK and active low SnDEN. Both parameters are programmable. Timing will apply to the relative active or inactive edge as stated in the timing table.

The timing diagram is to represent timing only, it is not intended to represent the functionality of the port.

Timing parameters for both SSI ports are identical. Only one set of timing is presented with the *n* representing either 0 or 1.

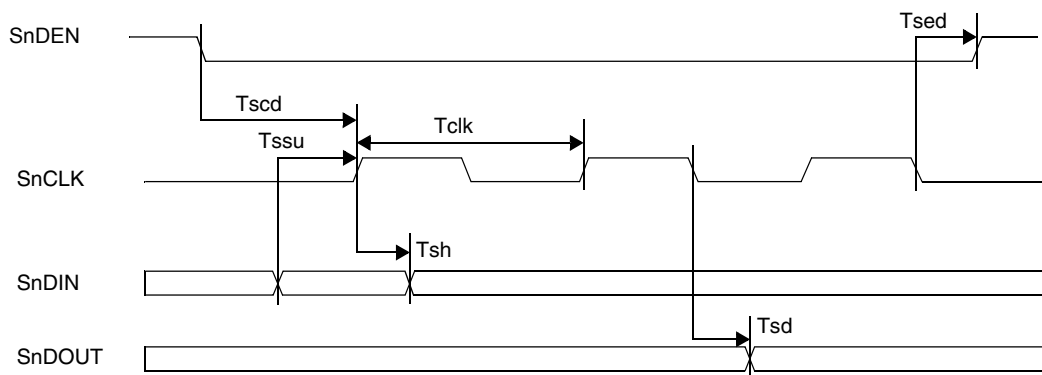


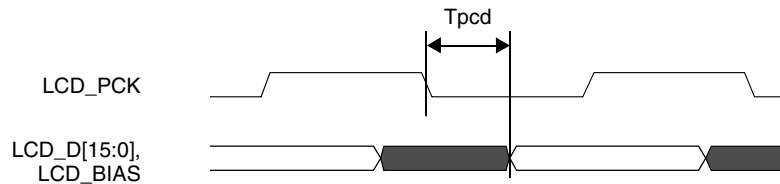
Figure 11-12. SSI Timing Diagram

**11.4.5.5 LCD Controller Timing**

The LCD controller timing is shown in Table 11-16 and Figure 11-13. The LCD\_BIAS signal and LCD data signals change state on the inactive edge of the pixel clock (LCD\_PCK). Note that the diagram assumes `lcd_clkcontrol[IC] = 0`. (See the invert-pixel-clock bit description in Section 6.9.1.1 "LCD Control Register" on page 167.)

**Table 11-16. LCD Controller Timing**

Signal	Symbol	Parameter	Min	Max	Unit
LCD_PCK, LCD_D[15:0], LCD_BIAS	Tpcd	Pixel clock inactive edge to LCD_D[15:0] and LCD_BIAS valid.	-1.5	5	ns



**Figure 11-13. LCD Controller Timing Diagram**

## 11.4.5.6 EJTAG Interface Timing

Table 11-17. EJTAG Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
TCK	Tec	EJTAG TCK cycle time	40		ns
	Tech	TCK high time	10		ns
	TecL	TCK low time	10		ns
TMS, TDI	Tesu	Setup before TCK for TMS and TDI	5		ns
	Teh	Hold after TCK for TMS and TDI	3		ns
TDO	Teco	Delay from TCK to TDO on output		15	ns
	Tecz	Delay from TCK to TDO tristate		15	ns
TRST#	TrstL	TRST# low time	25		ns

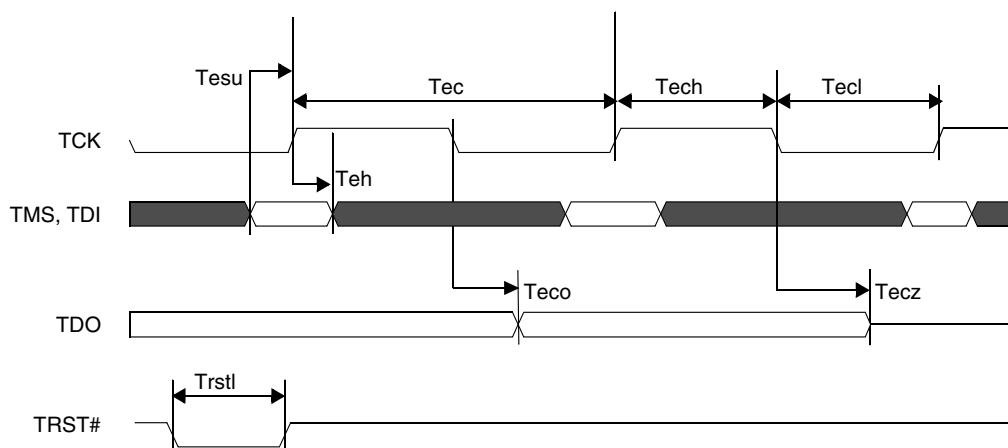


Figure 11-14. EJTAG Timing Diagram

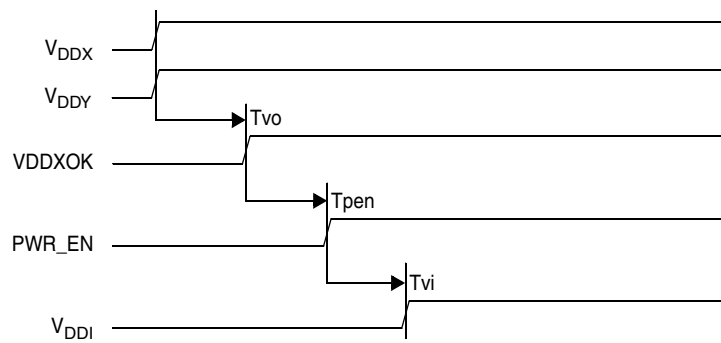
## 11.5 Power-up and Reset Timing

This section provides the timing specifications for the power-up sequence, and the hardware and runtime reset sequences. (See Section 8.0 "Power-up, Reset and Boot" on page 223 for functional descriptions of the sequences.)

### 11.5.1 Power-up Sequence Timing

**Table 11-18. Power-up Timing Parameters**

Parameter	Description	Min	Max
$T_{vo}$	$V_{DDX}$ at 90% of nominal to VDDXOK asserted	0 ns	
$T_{pen}$	VDDXOK asserted to PWR_EN driven high		30 ns
$T_{vi}$	PWR_EN to $V_{DDI}$ stable		20 ms



**Figure 11-15. Power-up Sequence**

11.5.2 Hardware Reset Timing

Table 11-19. Hardware Reset Timing Parameters

Parameter	Description	Min	Typ	Max
$T_{vxr}$	VDDXOK asserted to RESETIN# deasserted	0 ns		System Dependent
$T_{vl}$	VDDXOK low time	1 $\mu$ s		
$T_{rstl}$	RESETIN# low time	1 $\mu$ s		
$T_{vro}$	RESETIN# to RESETOUT# delay. MAX = max[750 ns, 170 ms - $T_{vxr}$ ]	600 ns		see desc.
$T_{rocs}$	RESETOUT# to RCS0#/SDCS0# asserted.		135 ns	1 $\mu$ s

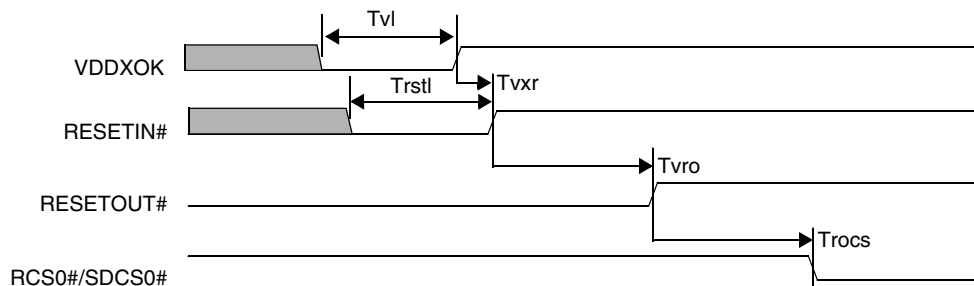


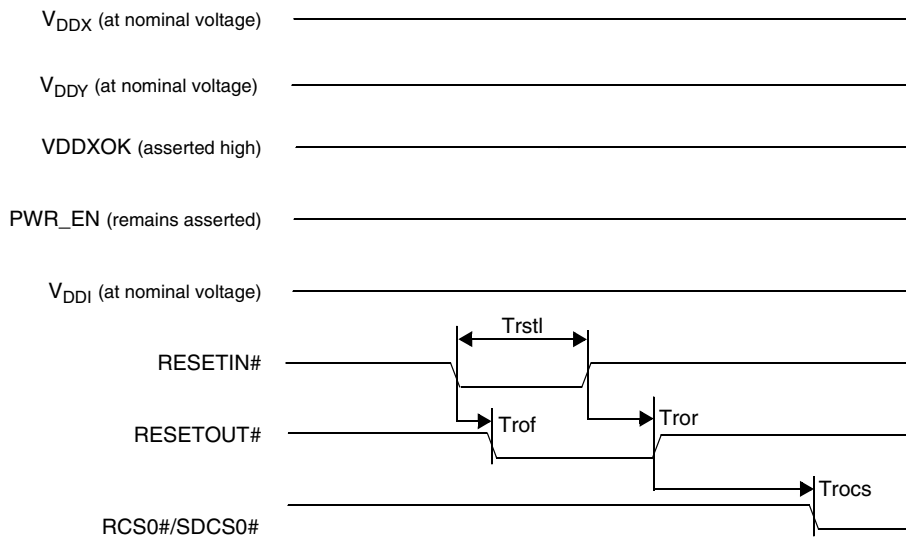
Figure 11-16. Hardware Reset Sequence



### 11.5.3 Runtime Reset Timing

**Table 11-20. Runtime Reset Timing Parameters**

Parameter	Description	Min	Typ	Max
T <sub>rstl</sub>	RESETIN# low time	1 $\mu$ s		
T <sub>rof</sub>	RESETIN# falling to RESETOUT# falling MAX: 25 ns + (0.5 * (CPU Clock/2))			see desc.
T <sub>ror</sub>	RESETIN# rising to RESETOUT# rising MAX: 25 ns + (0.5 * (CPU Clock/2)) + (120 * CPU Clock)	120 CPU clocks		see desc.
T <sub>rocs</sub>	RESETOUT# to RCS0#/SDCS0# asserted. Note that the timing values shown assume a 400 MHz CPU clock.		65 ns	500 ns



**Figure 11-17. Runtime Reset Sequence**

## 11.6 Asynchronous Signals

**GPIO** - The GPIO signals are driven by software. Note, however, when GPIO signals are used as inputs, there are timing requirements to ensure signal state changes are recognized cleanly; see Section 11.4.4 "GPIO Input Timing Requirements" on page 272.

**UART** - All UART signals are asynchronous to other external signals.

**USB** - All USB signals are asynchronous to other external signals. The USB protocol should be followed for appropriate operation.

## 11.7 External Clock Specifications

The EXTCLK[1:0] external clocks have a maximum frequency rating of  $(F_{\max} / 16)$ , where  $F_{\max}$  is the maximum frequency rating for the part. Table 11-21 provides the EXTCLK[1:0] specifications.

**Table 11-21. External Clock EXTCLK[1:0] Specifications**

Characteristic	333 MHz		400 MHz		500 MHz		Unit
	Min	Max	Min	Max	Min	Max	
Frequency		20.81		25		31.25	MHz
Frequency jitter		4		4		4	%
Duty cycle	40	60	40	60	40	60	%

## 11.8 Crystal Specifications

Note that load capacitors for the external oscillators are integrated into the Au1100 processor so no external circuitry is required when using the specified crystal. For design layout considerations concerning the crystals, see Section 11.9.1 "Crystal Layout" on page 284.

Table 11-22 provides the specification for the parallel resonant 12 MHz crystal to be placed between XT112 and XTO12 and Table 11-23 provides the specification for the parallel resonant 32 kHz crystal to be placed between XT132 and XTO32.

**Table 11-22. 12 MHz Crystal Specification**

Specification	Min	Typ	Max	Unit
Resonant Frequency	11	12	15	MHz
Frequency Stability			±100	ppm
Motional Resistance			60	Ohms
Shunt Capacitance		<5	7	pF
Load Capacitance (Note 1)	8	12	20	pF
Drive Level			100	μW
Crystal Type	AT Cut			

Note 1. This capacitance is integrated on the Au1100.

**Table 11-23. 32.768 kHz Crystal Specification**

Specification	Min	Typ	Max	Unit
Resonant Frequency		32.768		kHz
Equivalent Series Resistance			50k	Ohms
Shunt Capacitance		1.5	2.0	pF
Load Capacitance (Note 1)	6		12	pF
Motional Capacitance		3	4	fF
Drive Level			1	μW
Quality Factor	40k			
Crystal Type	Tuning Fork			

Note 1. This capacitance is integrated on the Au1100.

## 11.9 System Design Considerations

This section provides information for system-level design issues.

### 11.9.1 Crystal Layout

The crystal layouts are critical. Without using vias, place traces directly over a ground plane on the top layer with keep-outs on all surrounding sides. Trace lengths should be less than 0.5 inches, and trace widths should be set to the minimum signal trace width for the design. Be sure not to allow other signals to come within 0.025 inches of these sensitive analog signals.

### 11.9.2 Decoupling Recommendations

This section provides recommendations for minimizing noise in a system. Note that specific decoupling requirements are system dependent.

To filter noise on the power supplies,  $V_{DDX}$ ,  $V_{DDY}$  and  $V_{DDI}$ , as well as XPWR12 and XPWR32, should be bypassed to ground using 10- $\mu$ F capacitors: For each of the four sides of the package, place a capacitor within 0.5 inches.

To filter high-frequency noise, capacitors in the 10 nF range should be placed under the package:

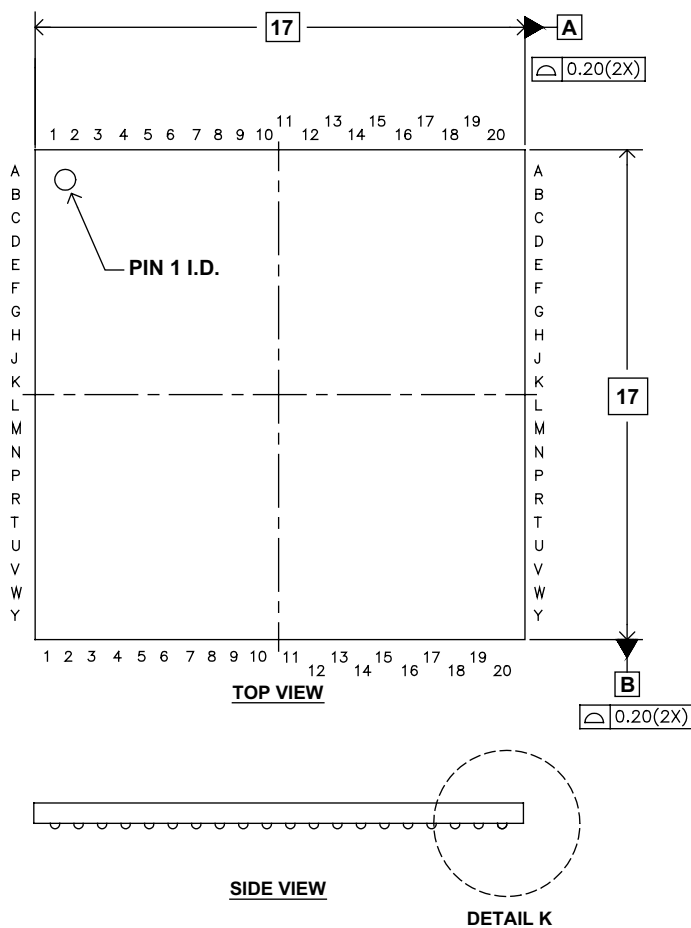
- For minimal high-frequency decoupling, use six to eight 10 nF capacitors.
- For systems requiring a broader spectrum of high-frequency noise be filtered, use four 15 nF and four 6.8 nF capacitors.

# Packaging, Pin Assignment and Ordering Information

This chapter provides information about the Au1100 processor package and pin assignment, as well as providing ordering information. The contents of the chapter are organized as follows:

- The package dimensions are shown in Figure 12-1 starting on page 286. The Au1100 is packaged in a 399-pin LF-PBGA device.
- Table 12-2 (starting on page 288) is the connection diagram showing the pin and signal placement on the package. For pins that provide multiple signal functions, the default signal is shown first followed by the alternate signal in parentheses. Note that the black square in the upper-left hand corner indicates where the device is keyed.
- The pin assignment listing ordered by pin number starts on page 290.
- The pin assignment listing sorted by default signal starts on page 296.
- The pin assignment listing sorted by alternate signal is on page 300.
- Ordering information is supplied on page 301.

### 12.1 Mechanical Package



**NOTES**

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M-1994 .
2. ALL DIMENSIONS ARE IN MILLIMETERS .
3. BALL POSITION DESIGNATION PER JESD 95-1, SPP-010.
4. BALL DIAMETER IS MEASURED AT ITS MAXIMUM DIMENSION IN A PLANE PARALLEL TO DATUM C.
5. THIS PACKAGE IS DIMENSIONED IN THE MANNER OF JEDEC OUTLINE MO-205 REV E, VARIATION AM.

Figure 12-1. Package Dimensions

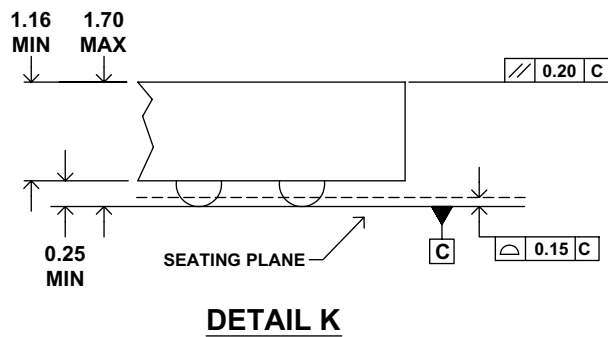
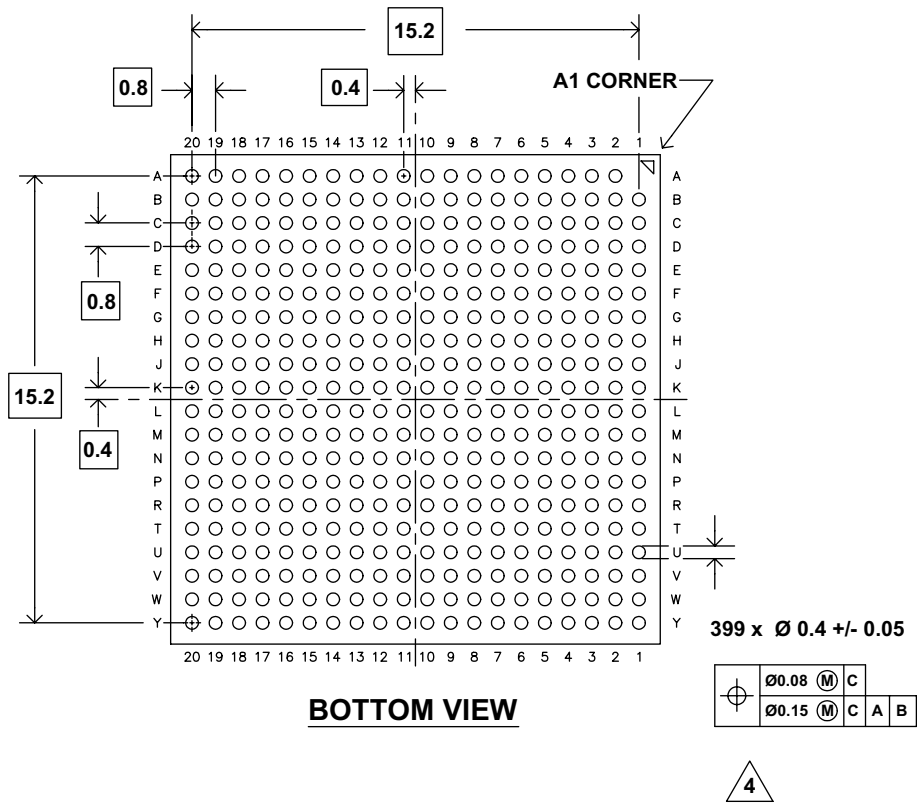


Figure 12-1. Package Dimensions (Continued)

## 12.2 Pin Assignments

Figure 12-2. Connection Diagram — Top View

	1	2	3	4	5	6	7	8	9
A	—	XTI32	XPWR12	XTO12	XTI12	TC3	GPIO[207] (PWE#)	GPIO[204] (PREG#)	PWAIT#
B	XPWR32	XAGND32	XTO32	ROMSEL	XAGND12	TC0	PIOW#	POE#	PIOS16#
C	VSEL	TMS	TESTEN	TRST#	VDDXOK	GPIO[0]	TC1	GPIO[5] (DMA_REQ1)	PIOR#
D	SDA11	RESETOUT#	SDA12	ROMSIZE	RESETIN#	GPIO[1]	GPIO[4] (DMA_REQ0)	TC2	GPIO[7]
E	SDA5	SDA6	SDA9	SDA8	SDA10	PWR_EN	GPIO[2] (EXTCLK0)	GPIO[3] (EXTCLK1)	GPIO[6] (SMROMCKE)
F	SDA3	SDA1	SDA2	SDA4	SDA7	V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	V <sub>SS</sub>
G	SDQM3#	SDQM2#	SDA0	SDBA0	SDBA1	V <sub>SS</sub>	V <sub>DDY</sub>	V <sub>DDY</sub>	V <sub>DDX</sub>
H	SDWE#	SDQM1#	SDQM0#	SDCAS#	SDRAS#	V <sub>DDI</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
J	SDCS2#	SDCKE	V <sub>DDY</sub>	SDCLK2	V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
K	SDCS0#	SDCS1#	V <sub>SS</sub>	SDCLK1	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
L	SDD31	SDD28	V <sub>DDY</sub>	SDCLK0	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
M	SDD30	SDD27	SDD24	SDD25	SDD26	V <sub>DDI</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
N	SDD29	SDD21	SDD20	SDD22	SDD23	V <sub>DDI</sub>	V <sub>DDY</sub>	V <sub>SS</sub>	V <sub>SS</sub>
P	SDD17	SDD15	SDD18	SDD19	SDD14	V <sub>SS</sub>	V <sub>DDY</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>
R	SDD16	SDD11	SDD12	SDD13	SDD10	V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	V <sub>SS</sub>
T	SDD9	SDD7	SDD8	SDMS1_DAT1	V <sub>SS</sub>	SDMS0_DAT0	SDMS0_CLK	RESVD[2]	GPIO[19]
U	SDD6	SDD1	SDD2	SDD3	SDMS_MS_EN	LCD_PWM1	GPIO[22]	N0TXCLK	GPIO[27] (N0TXD2)
V	SDD5	SDD0	SDMS1_DAT3	SDMS0_DAT1	SDMS0_CMD	LCD_PWM0	GPIO[23]	GPIO[21]	GPIO[28] (N0TXD3)
W	SDD4	SDMS1_DAT2	SDMS1_CMD	SDMS1_CLK	RESVD[4]	GPIO[17]	RESVD[1]	RESVD[0]	GPIO[215] (N0MDC)
Y	SDMS1_DAT0	SDMS0_DAT3	SDMS0_DAT2	RESVD[5]	RESVD[3]	GPIO[18]	GPIO[16]	GPIO[20]	N0MDIO
	1	2	3	4	5	6	7	8	9



Figure 12-2. Connection Diagram — Top View (Continued)

10	11	12	13	14	15	16	17	18	19	20	
EWAIT#	GPIO[203] (LWR1#)	RD4	RD9	RD11	RD16	RD17	RD24	RD25	RD26	RD28	<b>A</b>
GPIO[206] (PCE2#)	GPIO[200] (LRD0#)	RD0	RD6	RD10	RD15	RD18	RD22	RD27	RD31	RAD2	<b>B</b>
GPIO[201] (LRD1#)	GPIO[202] (LWR0#)	RD5	RD8	RD12	RD19	RD21	RD23	RD29	RAD1	RAD10	<b>C</b>
GPIO[205] (PCE1#)	V <sub>SS</sub>	LWAIT#	RD2	RD3	RD13	RD20	RD30	RAD0	RAD5	RAD11	<b>D</b>
V <sub>SS</sub>	LCLK	RD1	RD14	RD7	V <sub>DDI</sub>	V <sub>SS</sub>	RAD3	RAD4	RAD9	RAD15	<b>E</b>
V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	RAD6	RAD8	RAD16	RAD19	<b>F</b>
V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDI</sub>	RAD12	RAD7	RAD14	RAD20	RAD21	<b>G</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>SS</sub>	RAD18	RAD13	RAD17	RAD22	RAD27	<b>H</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>SS</sub>	RAD25	RAD23	RAD24	RAD26	RAD28	<b>J</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>DDI</sub>	RBE0#	RAD31	RAD30	RBE2#	RAD29	<b>K</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>DDI</sub>	V <sub>DDX</sub>	RBE3#	RWE#	ROE#	RBE1#	<b>L</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>SS</sub>	LCD_PCK	LCD_LCK	LCD_D5	RCS1#	RCS0#	<b>M</b>
V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDX</sub>	V <sub>SS</sub>	V <sub>SS</sub>	LCD_D4	LCD_D3	LCD_D2	RCS2#	<b>N</b>
V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDX</sub>	V <sub>DDI</sub>	LCD_D6	LCD_FCK	LCD_D11	LCD_D10	RCS3#	<b>P</b>
V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>DDI</sub>	V <sub>DDI</sub>	LCD_D7	LCD_D8	LCD_BIAS	LCD_D0	<b>R</b>
N0RXD3	GPIO[14] (U3DTR#)	N0RXD2	ACDI	U1RXD	V <sub>DDI</sub>	V <sub>SS</sub>	LCD_LEND	LCD_D9	LCD_D12	LCD_D1	<b>T</b>
GPIO[24] (N0TXEN)	N0RXD1	GPIO[30] (I2SCLK)	ACBCLK (S1DIN)	ACRST# (S1DEN)	GPIO[210] (S0DEN)	GPIO[8] (I2SDI)	GPIO[9] (U3CTS#)	TDO	TDI	LCD_D13	<b>U</b>
GPIO[26] (N0TXD1)	GPIO[15] (IRFIRSEL)	GPIO[213] (U1TXD)	GPIO[31] (I2SWORD)	ACSYNC (S1DOUT)	GPIO[211] (IRDATX)	GPIO[12] (U3Ri#)	GPIO[11] (U3DCC#)	TCK	USBH1M	LCD_D14	<b>V</b>
N0COL	N0RXCLK	GPIO[13] (U3RTS#)	N0RXD0	ACDO (S1CLK)	GPIO[29] (I2SDIO)	GPIO[209] (S0CLK)	USBDP (USBH0P)	S0DIN	U0RXD	LCD_D15	<b>W</b>
N0RXDV	GPIO[25] (N0TXD0)	GPIO[214] (U3TXD)	GPIO[212] (U0TXD)	N0CRS	U3RXD	IRDARX	GPIO[208] (S0DOUT)	USBDM (USBH0M)	GPIO[10] (U3DSR#)	USBH1P	<b>Y</b>
<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	

Table 12-1. Pin Assignment — Sorted by Pin Number

Pin Number	Default Signal	Alternate Signal
A1	—	
A2	XTI32	
A3	XPWR12	
A4	XTO12	
A5	XTI12	
A6	TC3	
A7	GPIO[207]	PWE#
A8	GPIO[204]	PREG#
A9	PWAIT#	
A10	EWAIT#	
A11	GPIO[203]	LWR1#
A12	RD4	
A13	RD9	
A14	RD11	
A15	RD16	
A16	RD17	
A17	RD24	
A18	RD25	
A19	RD26	
A20	RD28	
B1	XPWR32	
B2	XAGND32	
B3	XTO32	
B4	ROMSEL	
B5	XAGND12	
B6	TC0	
B7	PIOW#	
B8	POE#	
B9	PIOS16#	
B10	GPIO[206]	PCE2#
B11	GPIO[200]	LRD0#
B12	RD0	
B13	RD6	
B14	RD10	
B15	RD15	
B16	RD18	
B17	RD22	
B18	RD27	
B19	RD31	

Pin Number	Default Signal	Alternate Signal
B20	RAD2	
C1	VSEL	
C2	TMS	
C3	TESTEN	
C4	TRST#	
C5	VDDXOK	
C6	GPIO[0]	
C7	TC1	
C8	GPIO[5]	DMA_REQ1
C9	PIOR#	
C10	GPIO[201]	LRD1#
C11	GPIO[202]	LWR0#
C12	RD5	
C13	RD8	
C14	RD12	
C15	RD19	
C16	RD21	
C17	RD23	
C18	RD29	
C19	RAD1	
C20	RAD10	
D1	SDA11	
D2	RESETOUT#	
D3	SDA12	
D4	ROMSIZE	
D5	RESETIN#	
D6	GPIO[1]	
D7	GPIO[4]	DMA_REQ0
D8	TC2	
D9	GPIO[7]	
D10	GPIO[205]	PCE1#
D11	V <sub>SS</sub>	
D12	LWAIT#	
D13	RD2	
D14	RD3	
D15	RD13	
D16	RD20	
D17	RD30	
D18	RAD0	

Table 12-1. Pin Assignment — Sorted by Pin Number (Continued)

Pin Number	Default Signal	Alternate Signal
D19	RAD5	
D20	RAD11	
E1	SDA5	
E2	SDA6	
E3	SDA9	
E4	SDA8	
E5	SDA10	
E6	PWR_EN	
E7	GPIO[2]	EXTCLK0
E8	GPIO[3]	EXTCLK1
E9	GPIO[6]	SMROMCKE
E10	V <sub>SS</sub>	
E11	LCLK	
E12	RD1	
E13	RD14	
E14	RD7	
E15	V <sub>DDI</sub>	
E16	V <sub>SS</sub>	
E17	RAD3	
E18	RAD4	
E19	RAD9	
E20	RAD15	
F1	SDA3	
F2	SDA1	
F3	SDA2	
F4	SDA4	
F5	SDA7	
F6	V <sub>SS</sub>	
F7	V <sub>DDI</sub>	
F8	V <sub>DDI</sub>	
F9	V <sub>SS</sub>	
F10	V <sub>SS</sub>	
F11	V <sub>DDI</sub>	
F12	V <sub>DDI</sub>	
F13	V <sub>SS</sub>	
F14	V <sub>SS</sub>	
F15	V <sub>DDI</sub>	
F16	V <sub>DDI</sub>	

Pin Number	Default Signal	Alternate Signal
F17	RAD6	
F18	RAD8	
F19	RAD16	
F20	RAD19	
G1	SDQM3#	
G2	SDQM2#	
G3	SDA0	
G4	SDBA0	
G5	SDBA1	
G6	V <sub>SS</sub>	
G7	V <sub>DDY</sub>	
G8	V <sub>DDY</sub>	
G9	V <sub>DDX</sub>	
G10	V <sub>DDX</sub>	
G11	V <sub>DDX</sub>	
G12	V <sub>DDX</sub>	
G13	V <sub>DDX</sub>	
G14	V <sub>DDX</sub>	
G15	V <sub>DDI</sub>	
G16	RAD12	
G17	RAD7	
G18	RAD14	
G19	RAD20	
G20	RAD21	
H1	SDWE#	
H2	SDQM1#	
H3	SDQM0#	
H4	SDCAS#	
H5	SDRAS	
H6	V <sub>DDI</sub>	
H7	V <sub>DDY</sub>	
H8	V <sub>SS</sub>	
H9	V <sub>SS</sub>	
H10	V <sub>SS</sub>	
H11	V <sub>SS</sub>	
H12	V <sub>SS</sub>	
H13	V <sub>SS</sub>	

Table 12-1. Pin Assignment — Sorted by Pin Number (Continued)

Pin Number	Default Signal	Alternate Signal
H14	V <sub>DDX</sub>	
H15	V <sub>SS</sub>	
H16	RAD18	
H17	RAD13	
H18	RAD17	
H19	RAD22	
H20	RAD27	
J1	SDCS2#	
J2	SDCKE	
J3	V <sub>DDY</sub>	
J4	SDCLK2	
J5	V <sub>SS</sub>	
J6	V <sub>DDI</sub>	
J7	V <sub>DDY</sub>	
J8	V <sub>SS</sub>	
J9	V <sub>SS</sub>	
J10	V <sub>SS</sub>	
J11	V <sub>SS</sub>	
J12	V <sub>SS</sub>	
J13	V <sub>SS</sub>	
J14	V <sub>DDX</sub>	
J15	V <sub>SS</sub>	
J16	RAD25	
J17	RAD23	
J18	RAD24	
J19	RAD26	
J20	RAD28	
K1	SDCS0#	
K2	SDCS1#	
K3	V <sub>SS</sub>	
K4	SDCLK1	
K5	V <sub>DDY</sub>	
K6	V <sub>SS</sub>	
K7	V <sub>DDY</sub>	
K8	V <sub>SS</sub>	
K9	V <sub>SS</sub>	
K10	V <sub>SS</sub>	

Pin Number	Default Signal	Alternate Signal
K11	V <sub>SS</sub>	
K12	V <sub>SS</sub>	
K13	V <sub>SS</sub>	
K14	V <sub>DDX</sub>	
K15	V <sub>DDI</sub>	
K16	RBE0#	
K17	RAD31	
K18	RAD30	
K19	RBE2#	
K20	RAD29	
L1	SDD31	
L2	SDD28	
L3	V <sub>DDY</sub>	
L4	SDCLK0	
L5	V <sub>SS</sub>	
L6	V <sub>SS</sub>	
L7	V <sub>DDY</sub>	
L8	V <sub>SS</sub>	
L9	V <sub>SS</sub>	
L10	V <sub>SS</sub>	
L11	V <sub>SS</sub>	
L12	V <sub>SS</sub>	
L13	V <sub>SS</sub>	
L14	V <sub>DDX</sub>	
L15	V <sub>DDI</sub>	
L16	V <sub>DDX</sub>	
L17	RBE3#	
L18	RWE#	
L19	ROE#	
L20	RBE1#	
M1	SDD30	
M2	SDD27	
M3	SDD24	
M4	SDD25	
M5	SDD26	
M6	V <sub>DDI</sub>	
M7	V <sub>DDY</sub>	

Table 12-1. Pin Assignment — Sorted by Pin Number (Continued)

Pin Number	Default Signal	Alternate Signal
M8	V <sub>SS</sub>	
M9	V <sub>SS</sub>	
M10	V <sub>SS</sub>	
M11	V <sub>SS</sub>	
M12	V <sub>SS</sub>	
M13	V <sub>SS</sub>	
M14	V <sub>DDX</sub>	
M15	V <sub>SS</sub>	
M16	LCD_PCK	
M17	LCD_LCK	
M18	LCD_D5	
M19	RCS1#	
M20	RCS0#	
N1	SDD29	
N2	SDD21	
N3	SDD20	
N4	SDD22	
N5	SDD23	
N6	V <sub>DDI</sub>	
N7	V <sub>DDY</sub>	
N8	V <sub>SS</sub>	
N9	V <sub>SS</sub>	
N10	V <sub>SS</sub>	
N11	V <sub>SS</sub>	
N12	V <sub>SS</sub>	
N13	V <sub>SS</sub>	
N14	V <sub>DDX</sub>	
N15	V <sub>SS</sub>	
N16	V <sub>SS</sub>	
N17	LCD_D4	
N18	LCD_D3	
N19	LCD_D2	
N20	RCS2#	
P1	SDD17	
P2	SDD15	
P3	SDD18	
P4	SDD19	

Pin Number	Default Signal	Alternate Signal
P5	SDD14	
P6	V <sub>SS</sub>	
P7	V <sub>DDY</sub>	
P8	V <sub>DDX</sub>	
P9	V <sub>DDX</sub>	
P10	V <sub>DDX</sub>	
P11	V <sub>DDX</sub>	
P12	V <sub>DDX</sub>	
P13	V <sub>DDX</sub>	
P14	V <sub>DDX</sub>	
P15	V <sub>DDI</sub>	
P16	LCD_D6	
P17	LCD_FCK	
P18	LCD_D11	
P19	LCD_D10	
P20	RCS3#	
R1	SDD16	
R2	SDD11	
R3	SDD12	
R4	SDD13	
R5	SDD10	
R6	V <sub>SS</sub>	
R7	V <sub>DDI</sub>	
R8	V <sub>DDI</sub>	
R9	V <sub>SS</sub>	
R10	V <sub>SS</sub>	
R11	V <sub>DDI</sub>	
R12	V <sub>DDI</sub>	
R13	V <sub>SS</sub>	
R14	V <sub>SS</sub>	
R15	V <sub>DDI</sub>	
R16	V <sub>DDI</sub>	
R17	LCD_D7	
R18	LCD_D8	
R19	LCD_BIAS	
R20	LCD_D0	
T1	SDD9	

Table 12-1. Pin Assignment — Sorted by Pin Number (Continued)

Pin Number	Default Signal	Alternate Signal
T2	SDD7	
T3	SDD8	
T4	SDMS1_DAT1	
T5	V <sub>SS</sub>	
T6	SDMS0_DAT0	
T7	SDMS0_CLK	
T8	RESVD[2]	
T9	GPIO[19]	
T10	N0RXD3	
T11	GPIO[14]	U3DTR#
T12	N0RXD2	
T13	ACDI	
T14	U1RXD	
T15	V <sub>DDI</sub>	
T16	V <sub>SS</sub>	
T17	LCD_LEND	
T18	LCD_D9	
T19	LCD_D12	
T20	LCD_D1	
U1	SDD6	
U2	SDD1	
U3	SDD2	
U4	SDD3	
U5	SDMS_MS_EN	
U6	LCD_PWM1	
U7	GPIO[22]	
U8	N0TXCLK	
U9	GPIO[27]	N0TXD2
U10	GPIO[24]	N0TXEN
U11	N0RXD1	
U12	GPIO[30]	I2SCLK
U13	ACBCLK	S1DIN
U14	ACRST#	S1DEN
U15	GPIO[210]	S0DEN
U16	GPIO[8]	I2SDI
U17	GPIO[9]	U3CTS#
U18	TDO	
U19	TDI	
U20	LCD_D13	

Pin Number	Default Signal	Alternate Signal
V1	SDD5	
V2	SDD0	
V3	SDMS1_DAT3	
V4	SDMS0_DAT1	
V5	SDMS0_CMD	
V6	LCD_PWM0	
V7	GPIO[23]	
V8	GPIO[21]	
V9	GPIO[28]	N0TXD3
V10	GPIO[26]	N0TXD1
V11	GPIO[15]	IRFIRSEL
V12	GPIO[213]	U1TXD
V13	GPIO[31]	I2SWORD
V14	ACSYNC	S1DOUT
V15	GPIO[211]	IRDATX
V16	GPIO[12]	U3RI#
V17	GPIO[11]	U3DCD#
V18	TCK	
V19	USBH1M	
V20	LCD_D14	
W1	SDD4	
W2	SDMS1_DAT2	
W3	SDMS1_CMD	
W4	SDMS1_CLK	
W5	RESVD[4]	
W6	GPIO[17]	
W7	RESVD[1]	
W8	RESVD[0]	
W9	GPIO[215]	N0MDC
W10	N0COL	
W11	N0RXCLK	
W12	GPIO[13]	U3RTS#
W13	N0RXD0	
W14	ACDO	S1CLK
W15	GPIO[29]	I2SDIO
W16	GPIO[209]	S0CLK
W17	USBDP	USBH0P
W18	S0DIN	
W19	U0RXD	

Table 12-1. Pin Assignment — Sorted by Pin Number (Continued)

Pin Number	Default Signal	Alternate Signal
W20	LCD_D15	
Y1	SDMS1_DAT0	
Y2	SDMS0_DAT3	
Y3	SDMS0_DAT2	
Y4	RESVD[5]	
Y5	RESVD[3]	
Y6	GPIO[18]	
Y7	GPIO[16]	
Y8	GPIO[20]	
Y9	N0MDIO	
Y10	N0RXDV	

Pin Number	Default Signal	Alternate Signal
Y11	GPIO[25]	N0TXD0
Y12	GPIO[214]	U3TXD
Y13	GPIO[212]	U0TXD
Y14	N0CRS	
Y15	U3RXD	
Y16	IRDARX	
Y17	GPIO[208]	S0DOUT
Y18	USBDM	USBH0M
Y19	GPIO[10]	U3DSR#
Y20	USBH1P	

Table 12-2. Pin Assignment — Sorted Alphabetically by Default Signal

Default Signal	Alternate Signal	Pin Number	Default Signal	Alternate Signal	Pin Number
ACBCLK	S1DIN	U13	GPIO[201]	LRD1#	C10
ACDI		T13	GPIO[202]	LWR0#	C11
ACDO	S1CLK	W14	GPIO[203]	LWR1#	A11
ACRST#	S1DEN	U14	GPIO[204]	PREG#	A8
ACSYNC	S1DOUT	V14	GPIO[205]	PCE1#	D10
EWAIT#		A10	GPIO[206]	PCE2#	B10
GPIO[0]		C6	GPIO[207]	PWE#	A7
GPIO[1]		D6	GPIO[208]	S0DOUT	Y17
GPIO[2]	EXTCLK0	E7	GPIO[209]	S0CLK	W16
GPIO[3]	EXTCLK1	E8	GPIO[210]	S0DEN	U15
GPIO[4]	DMA_REQ0	D7	GPIO[211]	IRDATX	V15
GPIO[5]	DMA_REQ1	C8	GPIO[212]	U0TXD	Y13
GPIO[6]	SMROMCKE	E9	GPIO[213]	U1TXD	V12
GPIO[7]		D9	GPIO[214]	U3TXD	Y12
GPIO[8]	I2SDI	U16	GPIO[215]	N0MDC	W9
GPIO[9]	U3CTS#	U17	IRDARX		Y16
GPIO[10]	U3DSR#	Y19	LCD_BIAS		R19
GPIO[11]	U3DCD#	V17	LCD_D0		R20
GPIO[12]	U3RI#	V16	LCD_D1		T20
GPIO[13]	U3RTS#	W12	LCD_D2		N19
GPIO[14]	U3DTR#	T11	LCD_D3		N18
GPIO[15]	IRFIRSEL	V11	LCD_D4		N17
GPIO[16]		Y7	LCD_D5		M18
GPIO[17]		W6	LCD_D6		P16
GPIO[18]		Y6	LCD_D7		R17
GPIO[19]		T9	LCD_D8		R18
GPIO[20]		Y8	LCD_D9		T18
GPIO[21]		V8	LCD_D10		P19
GPIO[22]		U7	LCD_D11		P18
GPIO[23]		V7	LCD_D12		T19
GPIO[24]	N0TXEN	U10	LCD_D13		U20
GPIO[25]	N0TXD0	Y11	LCD_D14		V20
GPIO[26]	N0TXD1	V10	LCD_D15		W20
GPIO[27]	N0TXD2	U9	LCD_FCK		P17
GPIO[28]	N0TXD3	V9	LCD_LCK		M17
GPIO[29]	I2SDIO	W15	LCD_LEND		T17
GPIO[30]	I2SCLK	U12	LCD_PCK		M16
GPIO[31]	I2SWORD	V13	LCD_PWM0		V6
GPIO[200]	LRD0#	B11	LCD_PWM1		U6



Table 12-2. Pin Assignment — Sorted Alphabetically by Default Signal (Continued)

Default Signal	Alternate Signal	Pin Number
LCLK		E11
LWAIT#		D12
N0COL		W10
N0CRS		Y14
N0MDIO		Y9
N0RXCLK		W11
N0RXD0		W13
N0RXD1		U11
N0RXD2		T12
N0RXD3		T10
N0RXDV		Y10
N0TXCLK		U8
PIOR#		C9
PIOS16#		B9
PIOW#		B7
POE#		B8
PWAIT#		A9
PWR_EN		E6
RAD0		D18
RAD1		C19
RAD2		B20
RAD3		E17
RAD4		E18
RAD5		D19
RAD6		F17
RAD7		G17
RAD8		F18
RAD9		E19
RAD10		C20
RAD11		D20
RAD12		G16
RAD13		H17
RAD14		G18
RAD15		E20
RAD16		F19
RAD17		H18
RAD18		H16
RAD19		F20
RAD20		G19

Default Signal	Alternate Signal	Pin Number
RAD21		G20
RAD22		H19
RAD23		J17
RAD24		J18
RAD25		J16
RAD26		J19
RAD27		H20
RAD28		J20
RAD29		K20
RAD30		K18
RAD31		K17
RBE0#		K16
RBE1#		L20
RBE2#		K19
RBE3#		L17
RCS0#		M20
RCS1#		M19
RCS2#		N20
RCS3#		P20
RD0		B12
RD1		E12
RD2		D13
RD3		D14
RD4		A12
RD5		C12
RD6		B13
RD7		E14
RD8		C13
RD9		A13
RD10		B14
RD11		A14
RD12		C14
RD13		D15
RD14		E13
RD15		B15
RD16		A15
RD17		A16
RD18		B16
RD19		C15

Table 12-2. Pin Assignment — Sorted Alphabetically by Default Signal (Continued)

Default Signal	Alternate Signal	Pin Number	Default Signal	Alternate Signal	Pin Number
RD20		D16	SDBA1		G5
RD21		C16	SDCAS#		H4
RD22		B17	SDCKE		J2
RD23		C17	SDCLK0		L4
RD24		A17	SDCLK1		K4
RD25		A18	SDCLK2		J4
RD26		A19	SDCS0#		K1
RD27		B18	SDCS1#		K2
RD28		A20	SDCS2#		J1
RD29		C18	SDD0		V2
RD30		D17	SDD1		U2
RD31		B19	SDD2		U3
RESETIN#		D5	SDD3		U4
RESETOUT#		D2	SDD4		W1
RESVD[0]		W8	SDD5		V1
RESVD[1]		W7	SDD6		U1
RESVD[2]		T8	SDD7		T2
RESVD[3]		Y5	SDD8		T3
RESVD[4]		W5	SDD9		T1
RESVD[5]		Y4	SDD10		R5
ROE#		L19	SDD11		R2
ROMSEL		B4	SDD12		R3
ROMSIZE		D4	SDD13		R4
RWE#		L18	SDD14		P5
S0DIN		W18	SDD15		P2
SDA0		G3	SDD16		R1
SDA1		F2	SDD17		P1
SDA2		F3	SDD18		P3
SDA3		F1	SDD19		P4
SDA4		F4	SDD20		N3
SDA5		E1	SDD21		N2
SDA6		E2	SDD22		N4
SDA7		F5	SDD23		N5
SDA8		E4	SDD24		M3
SDA9		E3	SDD25		M4
SDA10		E5	SDD26		M5
SDA11		D1	SDD27		M2
SDA12		D3	SDD28		L2
SDBA0		G4	SDD29		N1

Table 12-2. Pin Assignment — Sorted Alphabetically by Default Signal (Continued)

Default Signal	Alternate Signal	Pin Number
SDD30		M1
SDD31		L1
SDMS_MS_EN		U5
SDMS0_CLK		T7
SDMS0_CMD		V5
SDMS0_DAT0		T6
SDMS0_DAT1		V4
SDMS0_DAT2		Y3
SDMS0_DAT3		Y2
SDMS1_CLK		W4
SDMS1_CMD		W3
SDMS1_DAT0		Y1
SDMS1_DAT1		T4
SDMS1_DAT2		W2
SDMS1_DAT3		V3
SDQM0#		H3
SDQM1#		H2
SDQM2#		G2
SDQM3#		G1
SDRAS		H5
SDWE#		H1
TC0		B6
TC1		C7
TC2		D8
TC3		A6
TCK		V18
TDI		U19
TDO		U18
TESTEN		C3
TMS		C2
TRST#		C4
U0RXD		W19
U1RXD		T14
U3RXD		Y15
USBDM	USBH0M	Y18
USBDP	USBH0P	W17
USBH1M		V19
USBH1P		Y20

Default Signal	Alternate Signal	Pin Number
V <sub>DDI</sub> (Total of 22)		E15, F7, F8, F11, F12, F15, F16, G15, H6, J6, K15, L15, M6, N6, P15, R7, R8, R11, R12, R15, R16, T15
V <sub>DDX</sub> (Total of 20)		G9, G10, G11, G12, G13, G14, H14, J14, K14, L14, L16, M14, N14, P8, P9, P10, P11, P12, P13, P14
VDDXOK		C5
V <sub>DDY</sub> (Total of 12)		G7, G8, H7, J3, J7, K5, K7, L3, L7, M7, N7, P7
VSEL		C1
V <sub>SS</sub> (Total of 63)		D11, E10, E16, F6, F9, F10, F13, F14, G6, H8, H9, H10, H11, H12, H13, H15, J5, J8, J9, J10, J11, J12, J13, J15, K3, K6, K8, K9, K10, K11, K12, K13, L5, L6, L8, L9, L10, L11, L12, L13, M8, M9, M10, M11, M12, M13, M15, N8, N9, N10, N11, N12, N13, N15, N16, P6, R6, R9, R10, R13, R14, T5, T16
XAGND12		B5
XAGND32		B2
XPWR12		A3
XPWR32		B1
XTI12		A5
XTI32		A2
XTO12		A4
XTO32		B3

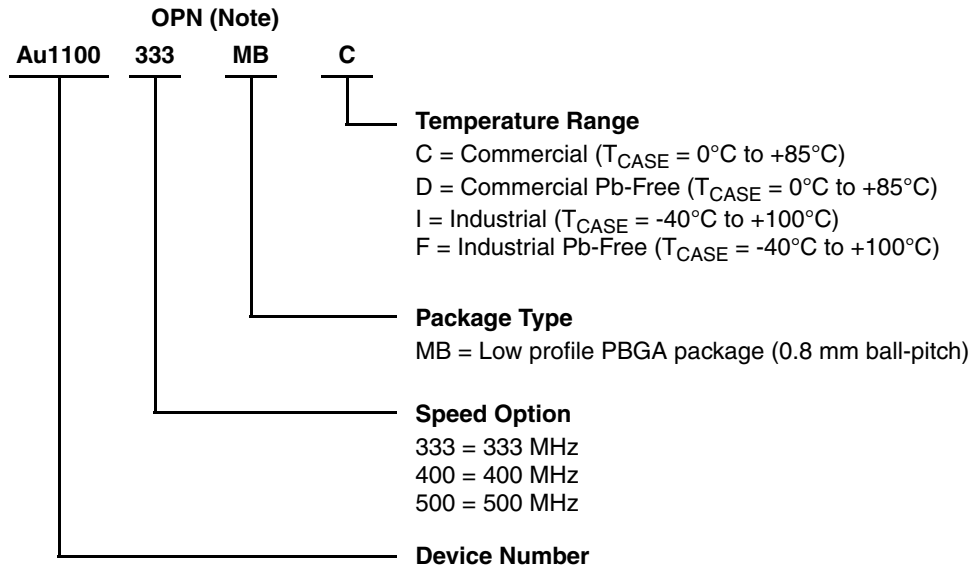
Table 12-3. Pin Assignment — Alternate Signals Sorted Alphabetically

Alternate Signal	Default Signal	Pin Number
DMA_REQ0	GPIO[4]	D7
DMA_REQ1	GPIO[5]	C8
EXTCLK0	GPIO[2]	E7
EXTCLK1	GPIO[3]	E8
I2SCLK	GPIO[30]	U12
I2SDI	GPIO[8]	U16
I2SDIO	GPIO[29]	W15
I2SWORD	GPIO[31]	V13
IRDATX	GPIO[211]	V15
IRFIRSEL	GPIO[15]	V11
LRD0#	GPIO[200]	B11
LRD1#	GPIO[201]	C10
LWR0#	GPIO[202]	C11
LWR1#	GPIO[203]	A11
N0MDC	GPIO[215]	W9
N0TXD0	GPIO[25]	Y11
N0TXD1	GPIO[26]	V10
N0TXD2	GPIO[27]	U9
N0TXD3	GPIO[28]	V9
N0TXEN	GPIO[24]	U10
PCE1#	GPIO[205]	D10
PCE2#	GPIO[206]	B10

Alternate Signal	Default Signal	Pin Number
PREG#	GPIO[204]	A8
PWE#	GPIO[207]	A7
S0CLK	GPIO[209]	W16
S0DEN	GPIO[210]	U15
S0DOUT	GPIO[208]	Y17
S1CLK	ACDO	W14
S1DEN	ACRST#	U14
S1DIN	ACBCLK	U13
S1DOUT	ACSYNC	V14
SMROMCKE	GPIO[6]	E9
U0TXD	GPIO[212]	Y13
U1TXD	GPIO[213]	V12
U3CTS#	GPIO[9]	U17
U3DCD#	GPIO[11]	V17
U3DSR#	GPIO[10]	Y19
U3DTR#	GPIO[14]	T11
U3RI#	GPIO[12]	V16
U3RTS#	GPIO[13]	W12
U3TXD	GPIO[214]	Y12
USBH0M	USBDM	Y18
USBH0P	USBDP	W17

### 12.3 Ordering Information

Ordering information for the AMD Alchemy™ Au1100™ processor is contained in this section. The ordering part number (OPN) is formed by a combination of elements. An example of the OPN is shown in Figure 12-3. Valid OPN combinations are provided in Table 12-4.



**Note:** Spaces are added to the ordering number shown above for viewing clarity only.

**Figure 12-3. OPN Example**

**Table 12-4. Valid OPN Combinations**

Device Number	Speed Option	Package Type	Temperature Range
Au1100	333	MB	C
			D
			I
			F
Au1100	400	MB	C
			D
Au1100	500	MB	C
			D

**Note:** Consult your local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations possibly not listed.



# Support Documentation

## A.1 Memory Map

The peripheral devices on the Au1100 processor contain memory-mapped registers visible to software. Table A-1 contains the memory map for the peripheral devices and physical memory. The addresses are 36 bits wide.

**Table A-1. Basic Au1100™ Processor Physical Memory Map**

Start Address	End Address	Size (MB)	Function
0x0 0000 0000	0x0 0FFF FFFF	256	Memory KSEG 0/1
0x0 1000 0000	0x0 11FF FFFF	32	I/O Devices on Peripheral Bus
0x0 1200 0000	0x0 13FF FFFF	32	Reserved
0x0 1400 0000	0x0 17FF FFFF	64	I/O Devices on System Bus
0x0 1800 0000	0x0 1FFF FFFF	128	Memory Mapped. 0x0 1FC0 0000 must contain the boot vector so this is typically where Flash or ROM is located.
0x0 2000 0000	0x0 7FFF FFFF	1536	Memory Mapped
0x0 8000 0000	0x0 EFFF FFFF	1792	Memory Mapped. Currently this space is memory mapped, but it should be considered reserved for future use.
0x0 F000 0000	0x0 FFFF FFFF	256	Debug Probe
0x1 0000 0000	0xC FFFF FFFF	4096 * 12	Reserved
0xD 0000 0000	0xD FFFF FFFF	4096	I/O Device
0xE 0000 0000	0xE FFFF FFFF	4096	External LCD Controller Interface
0xF 0000 0000	0xF FFFF FFFF	4096	PCMCIA Interface

The Au1100 processor system bus devices are mapped at the addresses based at 0x0 1400 0000. See Table A-2 for complete addresses.

**Table A-2. System Bus Devices Physical Memory Map**

Start Address	End Address	Size	Function
0x0 1400 0000	0x0 1400 0FFF	4 KB	SDRAM Memory Controller
0x0 1400 1000	0x0 1400 1FFF	4 KB	SRAM/FLASH Memory Controller
0x0 1400 2000	0x0 1400 2FFF	4 KB	DMA
0x0 1400 4000	0x0 1400 4FFF	4 KB	Ethernet DMA
0x0 1500 0000	0x0 1500 07FF	2 KB	LCD Controller

The Au1100 processor peripheral bus devices are based at 0x0 1100 0000. The individual memory spaces of the devices are defined in Table A-3.

**Table A-3. Peripheral Bus Devices Physical Memory Map**

Start Address	End Address	Size	Function
0x0 1000 0000	0x0 100F FFFF	1 MB	AC97 Controller
0x0 1010 0000	0x0 101F FFFF	1 MB	USB Host
0x0 1020 0000	0x0 102F FFFF	1 MB	USB Device
0x0 1030 0000	0x0 103F FFFF	1 MB	IrDA
0x0 1040 0000	0x0 104F FFFF	1 MB	Interrupt Controller 0
0x0 1050 0000	0x0 105F FFFF	1 MB	Ethernet MAC
0x0 1060 0000	0x0 106F FFFF	1 MB	SD Controller
0x0 1070 0000	0x0 10FF FFFF	9 MB	
0x0 1100 0000	0x0 110F FFFF	1 MB	I <sup>2</sup> S
0x0 1110 0000	0x0 111F FFFF	1 MB	UART0
0x0 1120 0000	0x0 112F FFFF	1 MB	UART1
0x0 1130 0000	0x0 113F FFFF	1 MB	
0x0 1140 0000	0x0 114F FFFF	1 MB	UART3
0x0 1150 0000	0x0 115F FFFF	1 MB	
0x0 1160 0000	0x0 116F FFFF	1 MB	SSI
0x0 1170 0000	0x0 117F FFFF	1 MB	Secondary GPIO
0x0 1180 0000	0x0 118F FFFF	1 MB	Interrupt Controller 1
0x0 1190 0000	0x0 119F FFFF	1 MB	System Control: RTC, TOY, Timers, Primary GPIO, Power Management



### A.1.1 Device Memory Map

Table A-4 lists all of the devices which are memory mapped to the Au1100 processor core. These devices are all mapped within KSEG1 (non-cached, non-TLB). All 32-bit addresses are translated into 36-bit addresses by changing bits 31:29 to zero and adding bits 35:32 which are set to zero.

**Table A-4. Device Memory Map**

Register	KSEG1 Address	Physical Address	Register	KSEG1 Address	Physical Address
<b>AC97 Controller - Section 6.1.1 on page 89</b>					
ac97_config	0xB000 0000	0x0 1000 0000	ir_wrphycfg	0xB030 0030	0x0 1030 0030
ac97_status	0xB000 0004	0x0 1000 0004	ir_maxpktlen	0xB030 0034	0x0 1030 0034
ac97_data	0xB000 0008	0x0 1000 0008	ir_rxbytecnt	0xB030 0038	0x0 1030 0038
ac97_cmmd	0xB000 000C	0x0 1000 000C	ir_config2	0xB030 003C	0x0 1030 003C
ac97_cmmdresp	0xB000 000C	0x0 1000 000C	ir_enable	0xB030 0040	0x0 1030 0040
ac97_control	0xB000 0010	0x0 1000 0010	<b>Interrupt Controller 0 - Section 5.2.1 on page 87</b>		
<b>USB Host Controller - Section 6.2.1 on page 95</b>			ic0_cfg0rd	0xB040 0040	0x0 1040 0040
Open HCI Register Set Base	0xB010 0000	0x0 1010 0000	ic0_cfg0set	0xB040 0040	0x0 1040 0040
usbh_enable	0xB017 FFFC	0x0 1017 FFFC	ic0_cfg0clr	0xB040 0044	0x0 1040 0044
<b>USB Device Controller - Section 6.3.1 on page 97</b>			ic0_cfg1rd	0xB040 0048	0x0 1040 0048
usbd_ep0rd	0xB020 0000	0x0 1020 0000	ic0_cfg1set	0xB040 0048	0x0 1040 0048
usbd_ep0wr	0xB020 0004	0x0 1020 0004	ic0_cfg1clr	0xB040 004C	0x0 1040 004C
usbd_ep1wr	0xB020 0008	0x0 1020 0008	ic0_cfg2rd	0xB040 0050	0x0 1040 0050
usbd_ep2wr	0xB020 000C	0x0 1020 000c	ic0_cfg2set	0xB040 0050	0x0 1040 0050
usbd_ep3rd	0xB020 0010	0x0 1020 0010	ic0_cfg2clr	0xB040 0054	0x0 1040 0054
usbd_ep4rd	0xB020 0014	0x0 1020 0014	ic0_req0int	0xB040 0054	0x0 1040 0054
usbd_inten	0xB020 0018	0x0 1020 0018	ic0_srcrd	0xB040 0058	0x0 1040 0058
usbd_intstat	0xB020 001C	0x0 1020 001C	ic0_srcset	0xB040 0058	0x0 1040 0058
usbd_config	0xB020 0020	0x0 1020 0020	ic0_srcclr	0xB040 005C	0x0 1040 005C
usbd_ep0cs	0xB020 0024	0x0 1020 0024	ic0_req1int	0xB040 005C	0x0 1040 005C
usbd_ep1cs	0xB020 0028	0x0 1020 0028	ic0_assignrd	0xB040 0060	0x0 1040 0060
usbd_ep2cs	0xB020 002C	0x0 1020 002C	ic0_assignset	0xB040 0060	0x0 1040 0060
usbd_ep3cs	0xB020 0030	0x0 1020 0030	ic0_assignclr	0xB040 0064	0x0 1040 0064
usbd_ep4cs	0xB020 0034	0x0 1020 0034	ic0_wakerd	0xB040 0068	0x0 1040 0068
usbd_ep0rdstat	0xB020 0040	0x0 1020 0040	ic0_wakeset	0xB040 006C	0x0 1040 006C
usbd_ep0wrstat	0xB020 0044	0x0 1020 0044	ic0_wakeclr	0xB040 0070	0x0 1040 0070
usbd_ep1wrstat	0xB020 0048	0x0 1020 0048	ic0_maskrd	0xB040 0070	0x0 1040 0070
usbd_ep2wrstat	0xB020 004C	0x0 1020 004C	ic0_maskset	0xB040 0074	0x0 1040 0074
usbd_ep3rdstat	0xB020 0050	0x0 1020 0050	ic0_maskclr	0xB040 0078	0x0 1040 0078
usbd_ep4rdstat	0xB020 0054	0x0 1020 0054	ic0_risingrd	0xB040 0078	0x0 1040 0078
usbd_enable	0xB020 0058	0x0 1020 0058	ic0_risingclr	0xB040 007C	0x0 1040 007C
<b>IrDA Controller - Section 6.4.1 on page 109</b>			ic0_fallingrd	0xB040 007C	0x0 1040 007C
ir_rngptrstat	0xB030 0000	0x0 1030 0000	ic0_fallingclr	0xB040 0080	0x0 1040 0080
ir_rngbsadrh	0xB030 0004	0x0 1030 0004	<b>Ethernet Controller MAC0 - Section 6.5.2 on page 125</b>		
ir_rngbsadrl	0xB030 0008	0x0 1030 0008	mac0_control	0xB050 0000	0x0 1050 0000
ir_ringsize	0xB030 000C	0x0 1030 000C	mac0_addrhigh	0xB050 0004	0x0 1050 0004
ir_rngprompt	0xB030 0010	0x0 1030 0010	mac0_addrlow	0xB050 0008	0x0 1050 0008
ir_rngadrcmp	0xB030 0014	0x0 1030 0014	mac0_hashhigh	0xB050 000C	0x0 1050 000C
ir_intclear	0xB030 0018	0x0 1030 0018	mac0_hashlow	0xB050 0010	0x0 1050 0010
ir_config1	0xB030 0020	0x0 1030 0020	mac0_miictrl	0xB050 0014	0x0 1050 0014
ir_sirflags	0xB030 0024	0x0 1030 0024	mac0_miidata	0xB050 0018	0x0 1050 0018
ir_statusen	0xB030 0028	0x0 1030 0028	mac0_flowctrl	0xB050 001C	0x0 1050 001C
ir_rdphycfg	0xB030 002C	0x0 1030 002C	mac0_vlan1	0xB050 0020	0x0 1050 0020
			mac0_vlan2	0xB050 0024	0x0 1050 0024

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
<b>Ethernet Controller Enable - Section 6.5.3 on page 133</b>		
macen_mac0	0xB052 0000	0x0 1052 0000
<b>SD Controller 0 - Section 6.10.1 on page 178</b>		
sd0_txport	0xB060 0000	0x0 1060 0000
sd0_rxport	0xB060 0004	0x0 1060 0004
sd0_config	0xB060 0008	0x0 1060 0008
sd0_enable	0xB060 000C	0x0 1060 000C
sd0_config2	0xB060 0010	0x0 1060 0010
sd0_blksize	0xB060 0014	0x0 1060 0014
sd0_status	0xB060 0018	0x0 1060 0018
sd0_debug	0xB060 001C	0x0 1060 001C
sd0_cmd	0xB060 0020	0x0 1060 0020
sd0_cmdarg	0xB060 0024	0x0 1060 0024
sd0_resp3	0xB060 0028	0x0 1060 0028
sd0_resp2	0xB060 002C	0x0 1060 002C
sd0_resp1	0xB060 0030	0x0 1060 0030
sd0_resp0	0xB060 0034	0x0 1060 0034
sd0_timeout	0xB060 0038	0x0 1060 0038
<b>SD Controller 1 - Section 6.10.1 on page 178</b>		
sd1_txport	0xB068 0000	0x0 1068 0000
sd1_rxport	0xB068 0004	0x0 1068 0004
sd1_config	0xB068 0008	0x0 1068 0008
sd1_enable	0xB068 000C	0x0 1068 000C
sd1_config2	0xB068 0010	0x0 1068 0010
sd1_blksize	0xB068 0014	0x0 1068 0014
sd1_status	0xB068 0018	0x0 1068 0018
sd1_debug	0xB068 001C	0x0 1068 001C
sd1_cmd	0xB068 0020	0x0 1068 0020
sd1_cmdarg	0xB068 0024	0x0 1068 0024
sd1_resp3	0xB068 0028	0x0 1068 0028
sd1_resp2	0xB068 002C	0x0 1068 002C
sd1_resp1	0xB068 0030	0x0 1068 0030
sd1_resp0	0xB068 0034	0x0 1068 0034
sd1_timeout	0xB068 0038	0x0 1068 0038
<b>i<sup>2</sup>S Controller - Section 6.6.1 on page 146</b>		
i2s_data	0xB100 0000	0x0 1100 0000
i2s_config	0xB100 0004	0x0 1100 0004
i2s_enable	0xB100 0008	0x0 1100 0008
<b>UART0 - Section 6.7.2 on page 151</b>		
uart0_rxddata	0xB110 0000	0x0 1110 0000
uart0_txddata	0xB110 0004	0x0 1110 0004
uart0_inten	0xB110 0008	0x0 1110 0008
uart0_intcause	0xB110 000C	0x0 1110 000C
uart0_fifoctrl	0xB110 0010	0x0 1110 0010
uart0_linectrl	0xB110 0014	0x0 1110 0014
—	0xB110 0018	0x0 1110 0018
uart0_linestat	0xB110 001C	0x0 1110 001C
—	0xB110 0020	0x0 1110 0020
uart0_clkdiv	0xB110 0028	0x0 1110 0028

Register	KSEG1 Address	Physical Address
uart0_enable	0xB110 0100	0x0 1110 0100
<b>UART1 - Section 6.7.2 on page 151</b>		
uart1_rxddata	0xB120 0000	0x0 1120 0000
uart1_txddata	0xB120 0004	0x0 1120 0004
uart1_inten	0xB120 0008	0x0 1120 0008
uart1_intcause	0xB120 000C	0x0 1120 000C
uart1_fifoctrl	0xB120 0010	0x0 1120 0010
uart1_linectrl	0xB120 0014	0x0 1120 0014
—	0xB120 0018	0x0 1120 0018
uart1_linestat	0xB120 001C	0x0 1120 001C
—	0xB120 0020	0x0 1120 0020
uart1_clkdiv	0xB120 0028	0x0 1120 0028
uart1_enable	0xB120 0100	0x0 1120 0100
<b>UART3 - Section 6.7.2 on page 151</b>		
uart3_rxddata	0xB140 0000	0x0 1140 0000
uart3_txddata	0xB140 0004	0x0 1140 0004
uart3_inten	0xB140 0008	0x0 1140 0008
uart3_intcause	0xB140 000C	0x0 1140 000C
uart3_fifoctrl	0xB140 0010	0x0 1140 0010
uart3_linectrl	0xB140 0014	0x0 1140 0014
uart3_mdmctrl	0xB140 0018	0x0 1140 0018
uart3_linestat	0xB140 001C	0x0 1140 001C
uart3_mdmstat	0xB140 0020	0x0 1140 0020
uart3_autoflow	0xB140 0024	0x0 1140 0024
uart3_clkdiv	0xB140 0028	0x0 1140 0028
uart3_enable	0xB140 0100	0x0 1140 0100
<b>SSIO - Section 6.8.2 on page 161</b>		
ssi0_status	0xB160 0000	0x0 1160 0000
ssi0_int	0xB160 0004	0x0 1160 0004
ssi0_inten	0xB160 0008	0x0 1160 0008
ssi0_config	0xB160 0020	0x0 1160 0020
ssi0_adata	0xB160 0024	0x0 1160 0024
ssi0_clkdiv	0xB160 0028	0x0 1160 0028
ssi0_enable	0xB160 0100	0x0 1160 0100
<b>SSI1 - Section 6.8.2 on page 161</b>		
ssi1_status	0xB168 0000	0x0 1168 0000
ssi1_int	0xB168 0004	0x0 1168 0004
ssi1_inten	0xB168 0008	0x0 1168 0008
ssi1_config	0xB168 0020	0x0 1168 0020
ssi1_adata	0xB168 0024	0x0 1168 0024
ssi1_clkdiv	0xB168 0028	0x0 1168 0028
ssi1_enable	0xB168 0100	0x0 1168 0100
<b>Secondary GPIO - Section 6.11.2 on page 189</b>		
gpio2_dir	0xB170 0000	0x0 1170 0000
reserved	0xB170 0004	0x0 1170 0004
gpio2_output	0xB170 0008	0x0 1170 0008
gpio2_pinstat	0xB170 000C	0x0 1170 000C
gpio2_inten	0xB170 0010	0x0 1170 0010
gpio2_enable	0xB170 0014	0x0 1170 0014

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
<b>Interrupt Controller 1 - Section 5.2.1 on page 87</b>		
ic1_cfg0rd	0xB180 0040	0x0 1180 0040
ic1_cfg0set	0xB180 0040	0x0 1180 0040
ic1_cfg0clr	0xB180 0044	0x0 1180 0044
ic1_cfg1rd	0xB180 0048	0x0 1180 0048
ic1_cfg1set	0xB180 0048	0x0 1180 0048
ic1_cfg1clr	0xB180 004C	0x0 1180 004C
ic1_cfg2rd	0xB180 0050	0x0 1180 0050
ic1_cfg2set	0xB180 0050	0x0 1180 0050
ic1_cfg2clr	0xB180 0054	0x0 1180 0054
ic1_req0int	0xB180 0054	0x0 1180 0054
ic1_srcrd	0xB180 0058	0x0 1180 0058
ic1_srcset	0xB180 0058	0x0 1180 0058
ic1_srcclr	0xB180 005C	0x0 1180 005C
ic1_req1int	0xB180 005C	0x0 1180 005C
ic1_assignrd	0xB180 0060	0x0 1180 0060
ic1_assignset	0xB180 0060	0x0 1180 0060
ic1_assignclr	0xB180 0064	0x0 1180 0064
ic1_wakerd	0xB180 0068	0x0 1180 0068
ic1_wakeset	0xB180 006C	0x0 1180 006C
ic1_wakeclr	0xB180 0070	0x0 1180 0070
ic1_maskrd	0xB180 0070	0x0 1180 0070
ic1_maskset	0xB180 0074	0x0 1180 0074
ic1_maskclr	0xB180 0078	0x0 1180 0078
ic1_risingrd	0xB180 0078	0x0 1180 0078
ic1_risingclr	0xB180 007C	0x0 1180 007C
ic1_fallingrd	0xB180 007C	0x0 1180 007C
ic1_fallingclr	0xB180 0080	0x0 1180 0080
<b>Clock Controller - Section 7.1.1 on page 195</b>		
sys_freqctrl0	0xB190 0020	0x0 1190 0020
sys_freqctrl1	0xB190 0024	0x0 1190 0024
sys_clksrc	0xB190 0028	0x0 1190 0028
sys_cpupll	0xB190 0060	0x0 1190 0060
sys_auxpll	0xB190 0064	0x0 1190 0064
<b>TOY &amp; RTC - Section 7.2.1 on page 205</b>		
sys_toytrim	0xB190 0000	0x0 1190 0000
sys_toywrite	0xB190 0004	0x0 1190 0004
sys_matchtoy0	0xB190 0008	0x0 1190 0008
sys_matchtoy1	0xB190 000C	0x0 1190 000C
sys_matchtoy2	0xB190 0010	0x0 1190 0010
sys_cntrctrl	0xB190 0014	0x0 1190 0014
sys_toyread	0xB190 0040	0x0 1190 0040
sys_rtctrim	0xB190 0044	0x0 1190 0044
sys_rtcwrite	0xB190 0048	0x0 1190 0048
sys_rtcmatch0	0xB190 004C	0x0 1190 004C
sys_rtcmatch1	0xB190 0050	0x0 1190 0050
sys_rtcmatch2	0xB190 0054	0x0 1190 0054
sys_rtcread	0xB190 0058	0x0 1190 0058

Register	KSEG1 Address	Physical Address
<b>Primary GPIO - Section 7.3.2 on page 211</b>		
sys_pinfunc	0xB190 002C	0x0 1190 002C
sys_trioutrd	0xB190 0100	0x0 1190 0100
sys_trioutclr	0xB190 0100	0x0 1190 0100
sys_outputrd	0xB190 0108	0x0 1190 0108
sys_outputset	0xB190 0108	0x0 1190 0108
sys_outputclr	0xB190 010C	0x0 1190 010C
sys_pinsterd	0xB190 0110	0x0 1190 0110
sys_pinputen	0xB190 0110	0x0 1190 0110
<b>Power Management - Section 7.4.4 on page 218</b>		
sys_scratch0	0xB190 0018	0x0 1190 0018
sys_scratch1	0xB190 001C	0x0 1190 001C
sys_wakemsk	0xB190 0034	0x0 1190 0034
sys_endian	0xB190 0038	0x0 1190 0038
sys_powerctrl	0xB190 003C	0x0 1190 003C
sys_wakesrc	0xB190 005C	0x0 1190 005C
sys_slppwr	0xB190 0078	0x0 1190 0078
sys_sleep	0xB190 007C	0x0 1190 007C
<b>SDRAM Controller - Section 3.1.2 on page 45</b>		
mem_sdmode0	0xB400 0000	0x0 1400 0000
mem_sdmode1	0xB400 0004	0x0 1400 0004
mem_sdmode2	0xB400 0008	0x0 1400 0008
mem_sdaddr0	0xB400 000c	0x0 1400 000c
mem_sdaddr1	0xB400 0010	0x0 1400 0010
mem_sdaddr2	0xB400 0014	0x0 1400 0014
mem_sdrefcfg	0xB400 0018	0x0 1400 0018
mem_sdprecmd	0xB400 001c	0x0 1400 001c
mem_sdautoref	0xB400 0020	0x0 1400 0020
mem_sdwrmd0	0xB400 0024	0x0 1400 0024
mem_sdwrmd1	0xB400 0028	0x0 1400 0028
mem_sdwrmd2	0xB400 002C	0x0 1400 002C
mem_sdsleep	0xB400 0030	0x0 1400 0030
mem_sdsckcke	0xB400 0034	0x0 1400 0034
<b>Static Bus Controller - Section 3.2.1 on page 53</b>		
mem_stcfg0	0xB400 1000	0x0 1400 1000
mem_sttime0	0xB400 1004	0x0 1400 1004
mem_staddr0	0xB400 1008	0x0 1400 1008
mem_stcfg1	0xB400 1010	0x0 1400 1010
mem_sttime1	0xB400 1014	0x0 1400 1014
mem_staddr1	0xB400 1018	0x0 1400 1018
mem_stcfg2	0xB400 1020	0x0 1400 1020
mem_sttime2	0xB400 1024	0x0 1400 1024
mem_staddr2	0xB400 1028	0x0 1400 1028
mem_stcfg3	0xB400 1030	0x0 1400 1030
mem_sttime3	0xB400 1034	0x0 1400 1034
mem_staddr3	0xB400 1038	0x0 1400 1038

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
<b>DMA Controller 0 - Section 4.1 on page 75</b>		
dma0_moderead	0xB400 2000	0x0 1400 2000
dma0_modeset	0xB400 2000	0x0 1400 2000
dma0_modeclr	0xB400 2004	0x0 1400 2004
dma0_peraddr	0xB400 2008	0x0 1400 2008
dma0_buf0addr	0xB400 200C	0x0 1400 200C
dma0_buf0size	0xB400 2010	0x0 1400 2010
dma0_buf1addr	0xB400 2014	0x0 1400 2014
dma0_buf1size	0xB400 2018	0x0 1400 2018
<b>DMA Controller 1 - Section 4.1 on page 75</b>		
dma1_moderead	0xB400 2100	0x0 1400 2100
dma1_modeset	0xB400 2100	0x0 1400 2100
dma1_modeclr	0xB400 2104	0x0 1400 2104
dma1_peraddr	0xB400 2108	0x0 1400 2108
dma1_buf0addr	0xB400 210c	0x0 1400 210c
dma1_buf0size	0xB400 2110	0x0 1400 2110
dma1_buf1addr	0xB400 2114	0x0 1400 2114
dma1_buf1size	0xB400 2118	0x0 1400 2118
<b>DMA Controller 2 - Section 4.1 on page 75</b>		
dma2_moderead	0xB400 2200	0x0 1400 2200
dma2_modeset	0xB400 2200	0x0 1400 2200
dma2_modeclr	0xB400 2204	0x0 1400 2204
dma2_peraddr	0xB400 2208	0x0 1400 2208
dma2_buf0addr	0xB400 220C	0x0 1400 220C
dma2_buf0size	0xB400 2210	0x0 1400 2210
dma2_buf1addr	0xB400 2214	0x0 1400 2214
dma2_buf1size	0xB400 2218	0x0 1400 2218
<b>DMA Controller 3 - Section 4.1 on page 75</b>		
dma3_moderead	0xB400 2300	0x0 1400 2300
dma3_modeset	0xB400 2300	0x0 1400 2300
dma3_modeclr	0xB400 2304	0x0 1400 2304
dma3_peraddr	0xB400 2308	0x0 1400 2308
dma3_buf0addr	0xB400 230C	0x0 1400 230C
dma3_buf0size	0xB400 2310	0x0 1400 2310
dma3_buf1addr	0xB400 2314	0x0 1400 2314
dma3_buf1size	0xB400 2318	0x0 1400 2318
<b>DMA Controller 4 - Section 4.1 on page 75</b>		
dma4_moderead	0xB400 2400	0x0 1400 2400
dma4_modeset	0xB400 2400	0x0 1400 2400
dma4_modeclr	0xB400 2404	0x0 1400 2404
dma4_peraddr	0xB400 2408	0x0 1400 2408
dma4_buf0addr	0xB400 240C	0x0 1400 240C
dma4_buf0size	0xB400 2410	0x0 1400 2410
dma4_buf1addr	0xB400 2414	0x0 1400 2414
dma4_buf1size	0xB400 2418	0x0 1400 2418

Register	KSEG1 Address	Physical Address
<b>DMA Controller 5 - Section 4.1 on page 75</b>		
dma5_moderead	0xB400 2500	0x0 1400 2500
dma5_modeset	0xB400 2500	0x0 1400 2500
dma5_modeclr	0xB400 2504	0x0 1400 2504
dma5_peraddr	0xB400 2508	0x0 1400 2508
dma5_buf0addr	0xB400 250C	0x0 1400 250C
dma5_buf0size	0xB400 2510	0x0 1400 2510
dma5_buf1addr	0xB400 2514	0x0 1400 2514
dma5_buf1size	0xB400 2518	0x0 1400 2518
<b>DMA Controller 6 - Section 4.1 on page 75</b>		
dma6_moderead	0xB400 2600	0x0 1400 2600
dma6_modeset	0xB400 2600	0x0 1400 2600
dma6_modeclr	0xB400 2604	0x0 1400 2604
dma6_peraddr	0xB400 2608	0x0 1400 2608
dma6_buf0addr	0xB400 260C	0x0 1400 260C
dma6_buf0size	0xB400 2610	0x0 1400 2610
dma6_buf1addr	0xB400 2614	0x0 1400 2614
dma6_buf1size	0xB400 2618	0x0 1400 2618
<b>DMA Controller 7 - Section 4.1 on page 75</b>		
dma7_moderead	0xB400 2700	0x0 1400 2700
dma7_modeset	0xB400 2700	0x0 1400 2700
dma7_modeclr	0xB400 2704	0x0 1400 2704
dma7_peraddr	0xB400 2708	0x0 1400 2708
dma7_buf0addr	0xB400 270C	0x0 1400 270C
dma7_buf0size	0xB400 2710	0x0 1400 2710
dma7_buf1addr	0xB400 2714	0x0 1400 2714
dma7_buf1size	0xB400 2718	0x0 1400 2718
<b>Ethernet Controller DMA Channels - Section 6.5.4 on page 135</b>		
macdma0_tx0stat	0xB400 4000	0x0 1400 4000
macdma0_tx0addr	0xB400 4004	0x0 1400 4004
macdma0_tx0len	0xB400 4008	0x0 1400 4008
macdma0_tx1stat	0xB400 4010	0x0 1400 4010
macdma0_tx1addr	0xB400 4014	0x0 1400 4014
macdma0_tx1len	0xB400 4018	0x0 1400 4018
macdma0_tx2stat	0xB400 4020	0x0 1400 4020
macdma0_tx2addr	0xB400 4024	0x0 1400 4024
macdma0_tx2len	0xB400 4028	0x0 1400 4028
macdma0_tx3stat	0xB400 4030	0x0 1400 4030
macdma0_tx3addr	0xB400 4034	0x0 1400 4034
macdma0_tx3len	0xB400 4038	0x0 1400 4038
macdma0_rx0stat	0xB400 4100	0x0 1400 4100
macdma0_rx0addr	0xB400 4104	0x0 1400 4104
macdma0_rx1stat	0xB400 4110	0x0 1400 4110
macdma0_rx1addr	0xB400 4114	0x0 1400 4114
macdma0_rx2stat	0xB400 4120	0x0 1400 4120
macdma0_rx2addr	0xB400 4124	0x0 1400 4124
macdma0_rx3stat	0xB400 4130	0x0 1400 4130
macdma0_rx3addr	0xB400 4134	0x0 1400 4134

**Table A-4. Device Memory Map (Continued)**

Register	KSEG1 Address	Physical Address
<b>LCD Controller - Section 6.9.1 on page 167</b>		
lcd_control	0xB500 0000	0x0 1500 0000
lcd_intstatus	0xB500 0004	0x0 1500 0004
lcd_intenable	0xB500 0008	0x0 1500 0008
lcd_horztiming	0xB500 000C	0x0 1500 000C
lcd_verttiming	0xB500 0010	0x0 1500 0010
lcd_clkcontrol	0xB500 0014	0x0 1500 0014
lcd_dmaaddr0	0xB500 0018	0x0 1500 0018
lcd_dmaaddr1	0xB500 001C	0x0 1500 001C
lcd_words	0xB500 0020	0x0 1500 0020
lcd_pwmdiv	0xB500 0024	0x0 1500 0024
lcd_pwmhi	0xB500 0028	0x0 1500 0028
lcd_palettebase	0xB500 0400	0x0 1500 0400

**A.1.2 Programming Tips****A.1.2.1 Memory Mapped Registers**

Peripheral, or system device registers should all be marked with the CCA bits to non-cacheable. Access must be on 32-bit boundaries, one 32-bit value at a time. See Section 2.2 "Caches" on page 15 for more information.

## A.2 Differences between Au1100™ and Au1000™ Processors

### A.2.1 Peripherals

The Au1100 processor does not have the following peripherals that are present on the Au1000 processor:

- Ethernet MAC 1
- UART2

The Au1100 processor has added these functions not present on the Au1000 processor:

- Integrated LCD controller
- Two Secure Digital controllers
- General-purpose I/O (GPIO): 48 total, 13 dedicated. (Au1000 has 32 total, 5 dedicated.)

Additionally, the SDRAM memory controller has an independent I/O power supply ( $V_{DDY}$ ) and can support both 3.3V and 2.5V devices.

### A.2.2 Miscellaneous

Some inputs to the interrupt controller have changed due to the addition/removal of blocks. Refer to the interrupt controller section for the Au1100 processor interrupt map.

The DMA channel encoding provides up to 32 Device IDs to the controller. New channels for SD data transfer have been added.

## A.3 Data Book Notations

This section addresses some of the terminology used in this book.

### A.3.1 Unpredictable and Undefined

The terms UNPREDICTABLE and UNDEFINED are used throughout this book to describe the behavior of the processor in certain cases. UNDEFINED behavior or operations can occur only as the result of executing instructions in a privileged mode (i.e., in Kernel Mode or Debug Mode, or with the CP0 usable bit set in the Status register). Unprivileged software can never cause UNDEFINED behavior or operations. Conversely, both privileged and unprivileged software can cause UNPREDICTABLE results or operations.

### A.3.2 Unpredictable

UNPREDICTABLE results may vary from processor implementation to implementation, instruction to instruction, or as a function of time on the same implementation or instruction. Software can never depend on results that are UNPREDICTABLE. UNPREDICTABLE operations may cause a result to be generated or not. If a result is generated, it is UNPREDICTABLE. UNPREDICTABLE operations may cause arbitrary exceptions.

UNPREDICTABLE results or operations have several implementation restrictions:

- Implementations of operations generating UNPREDICTABLE results must not depend on any data source (memory or internal state) which is inaccessible in the current processor mode
  - UNPREDICTABLE operations must not read, write, or modify the contents of memory or internal state which is inaccessible in the current processor mode. For example, UNPREDICTABLE operations executed in user mode must not access memory or internal state that is only accessible in Kernel Mode or Debug Mode or in another process
  - UNPREDICTABLE operations must not halt or hang the processor
- UNPRED used to describe the default state of registers should be taken as meaning UNPREDICATABLE.

### A.3.3 Undefined

UNDEFINED operations or behavior may vary from processor implementation to implementation, instruction to instruction, or as a function of time on the same implementation or instruction. UNDEFINED operations or behavior may vary from nothing to creating an environment in which execution can no longer continue. UNDEFINED operations or behavior may cause data loss.

UNDEFINED operations or behavior has one implementation restriction:

- UNDEFINED operations or behavior must not cause the processor to hang (that is, enter a state from which there is no exit other than powering down the processor). The assertion of any of the reset signals must restore the processor to an operational state.

### A.3.4 Register Fields

In general, fields marked as reserved should be considered unpredictable. In other words, these fields should be written zeros and ignored on read to preserve future compatibility.

## A.4 Data Book Revision History

This document is a report of the revision/creation process of the data book for the Au1100 processor. Any revisions (i.e., additions, deletions, parameter corrections, etc.) are recorded in the table(s) below.

**Table A-5. Revision History**

Revision (Date)	Description
A	See the <i>AMD Alchemy™ Au1100™ Processor Specification Update</i> (publication ID 27353).
B	
C (June 2005)	See revision C for details.
D (April 2006)	Removed “preliminary”, reformatted to bring page count down (from 421 to 314,) and corrected some minor errors. See Table A-6 for details.

**Table A-6. Edits to Current Revision**

Section	Revisions / Comments
<b>All Sections / General</b>	<ul style="list-style-type: none"> <li>Reformatted document for page, figure, table and section titles. <ul style="list-style-type: none"> <li>All registers now have either a Heading 3 or 4 associated with it so the electronic PDF will be more useful.</li> </ul> </li> <li>Changed active low signals to use and “#” instead of an overbar (e.g., <math>\overline{ACRST}</math> changed to ACRST#).</li> <li>Omitted Index and added back cover page.</li> <li>Removed “Preliminary”.</li> </ul>
<b>Section 1.0 "Overview"</b>	<ul style="list-style-type: none"> <li>Figure 1-1 "Block Diagram" on page 11: <ul style="list-style-type: none"> <li>Changed pages (moved forward to first page of Section 1.0).</li> </ul> </li> <li>Section 1.2 "Features" on page 12: <ul style="list-style-type: none"> <li>Modified second bullet under High-Bandwidth Memory Buses (removed “with NAND/NOR Flash support”).</li> </ul> </li> <li>Moved what was Section 1.3 “Data Book Notations” and Section 1.4 “Differences between Au1100™ and Au1000™ Processors” in rev c to the Appendix, Sections A.2 and A.3, respectively.</li> </ul>
<b>Section 2.0 "CPU"</b>	<ul style="list-style-type: none"> <li>Section 2.3.2 "Write Buffer Merging" on page 22: <ul style="list-style-type: none"> <li>Fixed addresses in the “Note” (i.e., changed 0x00010000 to 0x00001000 and 0x0001002 to 0x00001002).</li> </ul> </li> </ul>
<b>Section 4.0 "DMA Controller"</b>	<ul style="list-style-type: none"> <li>Table 4-1 "DMA Channel Base Addresses" on page 75: <ul style="list-style-type: none"> <li>Corrected cell heading from KSEG0 to KSEG1.</li> </ul> </li> <li>Section 4.1.4 "DMA Channel Buffer Size Registers" on page 80: <ul style="list-style-type: none"> <li>PDF of rev C was messed up and could not be read. Fixed.</li> </ul> </li> </ul>
<b>Section 6.0 "Peripheral Devices"</b>	<ul style="list-style-type: none"> <li>Section 6.5.3.1 "MAC0 Enable" on page 133: <ul style="list-style-type: none"> <li>Added IPG (bits [8:7]) to bit format and bit description tables.</li> </ul> </li> </ul>
<b>Section 8.0 "Power-up, Reset and Boot"</b>	<ul style="list-style-type: none"> <li>Section 8.3 "Boot" on page 225: <ul style="list-style-type: none"> <li>Corrected all references of 0x1FC0 0000 to 0x0 1FC0 0000.</li> </ul> </li> <li>Section 8.3.1.2 "16-Bit Boot for Big-Endian System" on page 226: <ul style="list-style-type: none"> <li>Corrected all references to 0x1FC00000 to 0x0 1FC0 0000.</li> </ul> </li> </ul>



**Table A-6. Edits to Current Revision**

Section	Revisions / Comments
<b>Section 12.0 "Packaging, Pin Assignment and Ordering Infor- mation"</b>	<ul style="list-style-type: none"><li>• Section 12.3 "Ordering Information" on page 301:<ul style="list-style-type: none"><li>— Updated figure and table with Pb-Free ordering information.</li></ul></li></ul>



[www.amd.com](http://www.amd.com)