# *The Alchemy Au1100™ From AMD Internet Edge Processor Data Book*

**PRELIMINARY INFORMATION**

***April 2002***

**Important:** This document contains preliminary information regarding a product under characterization. A revised document will be available when the product is fully characterized.

**Document Number 2000-0001**
Document Revision 0.5



## Disclaimer



AMD
7800 Shoal Creek Blvd, Suite 222W
Austin, TX 78757
512.421.6200 phone
512.421.6262 fax
www.alchemysemi.com

Technical Support: support@alchemysemi.com

# Contents

# List of Figures

# List of Tables

# 1 The Alchemy Au1100™ From AMD Internet Edge Processor

## 1.1 Overview

The Au1100, a follow-on to the Au1000, is a high performance, low power, high integration systems-on-a-chip (SOC) with the inclusion of a LCD controller and further reduction in power. The Au1100 is targeted at the Mobile Information Appliances (IAs). These IAs include web pads, telematics, PDAs and multimedia handheld computing devices.

## 1.2 Product Description

The Alchemy Au1100 from AMD is a complete SOC based on the MIPS32™ instruction set. Designed for maximum performance at very low power, the Au1100 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 Host and Device, UARTs (3), and GPIOs (up to 48,24 dedicated). The Au1100 runs a variety of operating systems, including Windows® CE.NET,Linux® and VxWorks®. Moreover, the integration of peripherals with the unique, very high performance, MIPS-compatible core provides lower system cost, smaller form factor, lower system power requirement, simpler designs at multiple performance points and thus, shorter design cycles.

**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
1.0-1.2 V Core
3.3 V or 2.5 V SDRAM I/O
3.3 V I/O

**Highly-Integrated System Peripherals**

GPIO (48 total, 13 dedicated)
10/100 Ethernet MAC Controller
USB Device and Host Controller
Three UARTs
IrDA Controller
AC'97 Controller
$I^2S$ 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**

16KB Non-Blocking Data Cache
16KB Instruction Cache
Instruction/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
Scalar branch techniques optimized:
- Pipelined register file access in fetchstage

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:
- 32 dual-entry fully-associative
- Variable page sizes 4KB-16MB
- 4-entry ITB

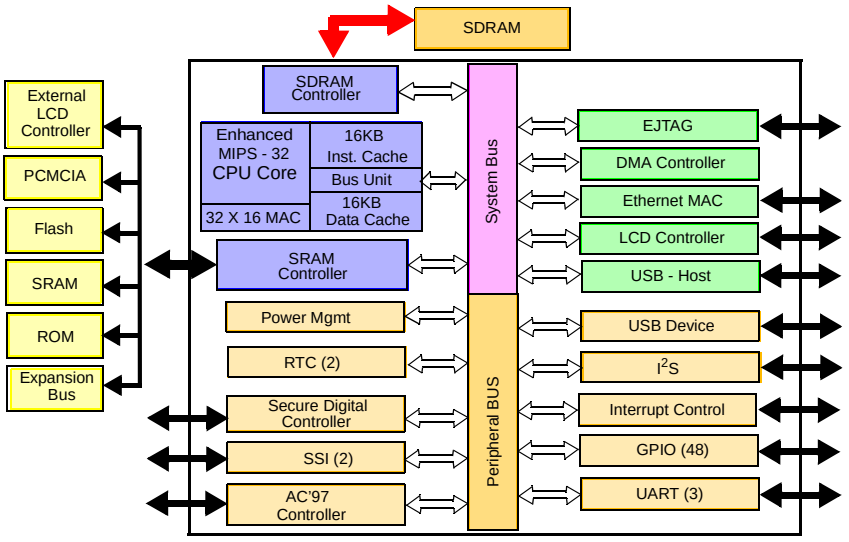



**FIGURE 1. Au1100 Internal Diagram**

### Low System Power

Core MHz Power
333 <200 mW
400 250 mW
500 500 mW
Power-Saving Modes:
- Idle
- Sleep

Static design to 0 Hz

### Package

399 BGA
17 mm x 17 mm

### Operating System Support

Microsoft Windows® CE
Linux®
VxWorks®

### Development Tool Support

Complete MIPS32-Compatible Tool Set
Numerous 3rd-Party Compilers, Assemblers and Debuggers

# 1.3 Databook Notations

This section addresses some of the terminology used in this book.

## 1.3.1 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.

## 1.3.2 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.

## 1.3.3 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

### 1.3.4 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.

## 1.4 Differences between Au1100 and Au1000

### 1.4.1 Peripherals

The Au1100 does not have the following peripherals that are present on the Au1000:

Ethernet MAC 1
UART2

The Au1100 has added these functions not present on the Au1000:

Integrated LCD Controller
Secure Digital Controller
13 Dedicated GPIO's (48 total)

Additionally, the SDRAM memory controller now has an independent I/O power supply (VDDY) and can support both 3.3 V and 2.5 V devices.

### 1.4.2 Miscellaneous

Some inputs to the interrupt controller have changed due to the addition/removal of blocks. Refer to the interrupt controller section for Au1100 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 new CCA encoding has been added to the Au1100. If CCA == 4, all system bus accesses will be cacheline aligned, i.e. no cacheline wrapping is supported.

# 2 CPU

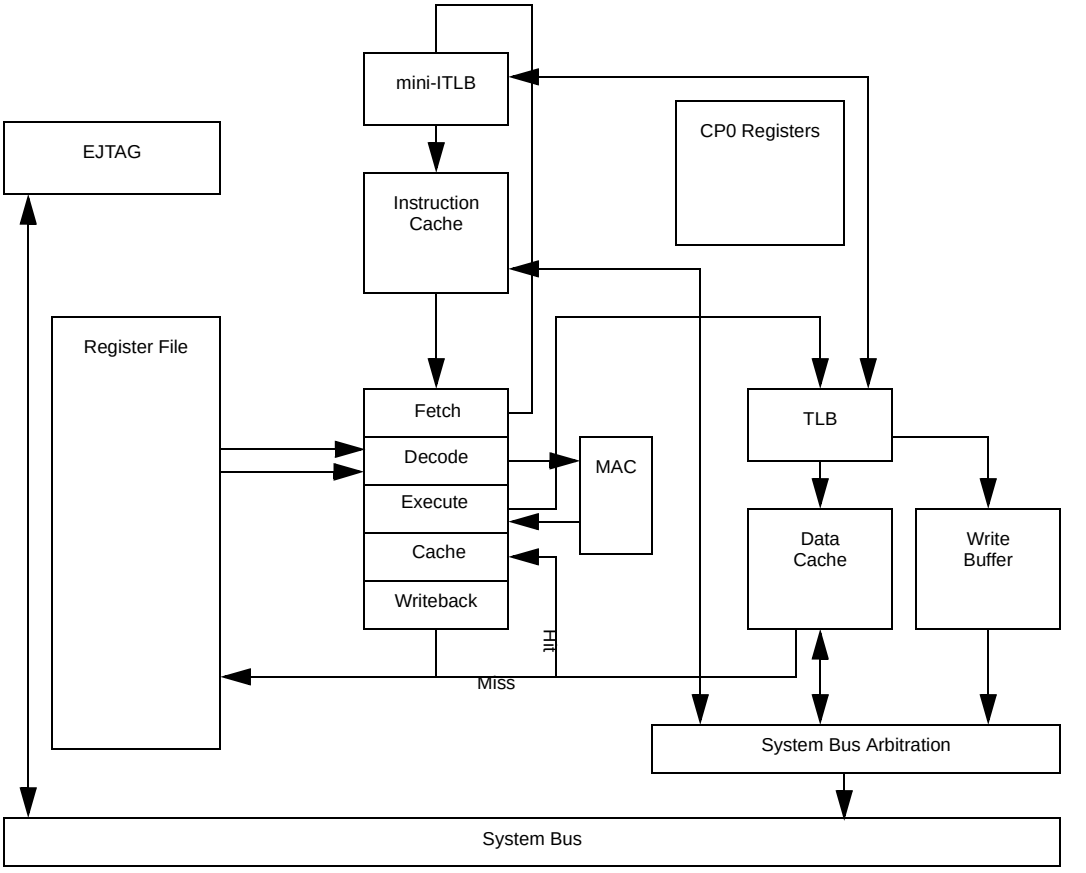
The Au1100 CPU core is a unique implementation of the MIPS32 instruction set architecture (ISA) designed for high frequency and low power. This chapter provides information on the implementation details of this MIPS32-compliant core.

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. Au1 Core Diagram**




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 unnecessary to pad load delay slots 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.

### 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 rd**).

The load/store hardware calculates addresses for data accesses. 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 to be fulfilled. Load data is forwarded to dependent instructions in the pipeline and the register file.

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 signalled 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.

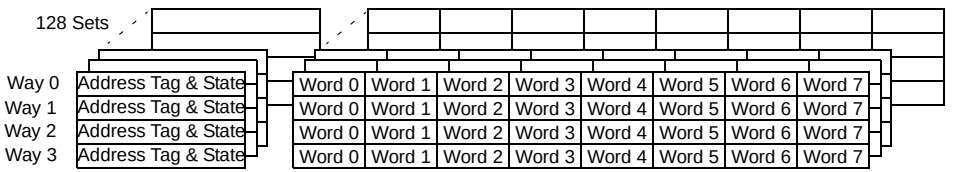
32x32 bit multiplies may be started every other CPU cycle. The 32x32 multiplies will complete in two cycles if the results are written to the general purpose registers.

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.

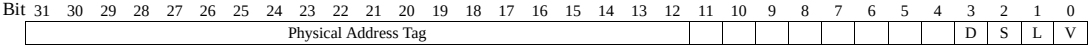
## 2.2 Caches

The Au1 core contains independent, on-chip 16KB instruction and data caches. Each cache is organized as four-way set associative, 128 sets per way, and 32 bytes per line.



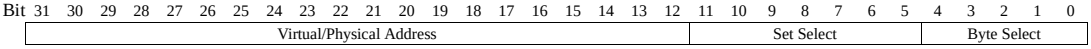
A cache line is tagged with a 20-bit physical address, a lock bit, and a valid bit. Data cache lines also include coherency and dirty status bits. The physical address tag contains bits 31:12 of the physical address; as such, physical addresses in which bits 35:32 are non-zero must be mapped non-cacheable.

**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.

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) 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 is a true LRU, but with the following priorities:

1. Streaming: cache misses are forced to way 0, cache hits remain unchanged.
2. Locking: cache misses follow policy above and set Lock bit, cache hits set Lock bit.

3. Normal LRU replacement policy.

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.

### 2.2.2 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.

### 2.2.3 Cache Management

The caches are managed with the **CACHE** instruction. Cache management operations are serializing and complete in order. The effect of the **CACHE** instruction is immediately visible to subsequent data accesses. Table 1 shows the Cache operations. An “X” indicates that the instruction or data cache supports the operation, an “N/A” indicates that the operation is not applicable.

**TABLE 1. Cache Operations**

| Operation | CACHE[20..18] Encoding | Instruction Cache | Data Cache |
|---|---|---|---|
| Index Invalidate | 000 | X | X (with writeback) |
| Index Load Tag | 001 | X | X |
| Index Store Tag | 010 | X | X |
| Hit Invalidate | 100 | X | X |
| Fill | 101 | X | N/A |
| Hit Writeback and Invalidate | 101 | N/A | X |
| Hit Writeback | 110 | N/A | X |
| Fetch and Lock | 111 | X | X |

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

### 2.2.4 Cache Coherency Attributes

The Cache Coherency Attributes (CCA) bits in Config0[K0] and in the TLB determine the cache-ability of accesses to memory. The Au1 implements the following:

**TABLE 2. CCA Values**

| CCA | Description |
|---|---|
| 000 - 0 | Reserved |
| 001 - 1 | Reserved |
| 010 - 2 | Uncached, non-mergable |
| 011 - 3 | Cacheable, coherent (critical word first) |
| 100 - 4 | Reserved |
| 101 - 5 | Cacheable, coherent (critical word first) |
| 110 - 6 | Cacheable, coherent, streaming (critical word first) |
| 111 - 7 | Uncached, mergable, gatherable |

CCA encodings 0 and 1 are unimplemented and must not be used.

CCA encoding 2 is uncached, as required by MIPS32. In addition, this encoding is non-mergable within the write buffer, to achieve a truly uncached effect.

CCA encoding 3 and 5 are cached and coherent. Coherent accesses are snooped to ensure data integrity. The access is performed as critical-word-first to improve performance.

CCA encoding 4 is unimplemented and must not be used.

CCA encoding 6 is cached and streaming. This encoding indicates that instructions and/or data are transient and are placed into way 0. The access is performed as critical word first.

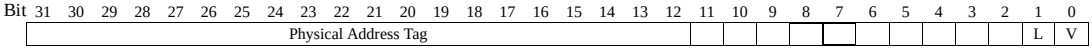
CCA encoding 7 is uncached, but mergable and gatherable. Data stores sent to the write buffer are subject to merging and gathering in the write buffer.

### 2.2.5 Instruction Cache

The instruction cache is a 16KB 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**




### 2.2.5.1 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
```

### 2.2.5.2 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 “Cache Line Replacement Policy” on page 11.

### 2.2.5.3 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 is snooped 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 may utilize the **CACHE** instruction to invalidate the modified range of program addresses in the instruction cache. The data cache is snooped during

instruction cache line fills; therefore it is not necessary to writeback the data cache prior to invalidating the instruction cache. Once complete, an instruction fetch to the newly loaded/ modified program will correctly fetch the new instructions (possibly from the data cache).

#### 2.2.5.4 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 011b, cacheable.

The CCA bits in the TLB entry control the cache-ability of the KUSEG region. 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:

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

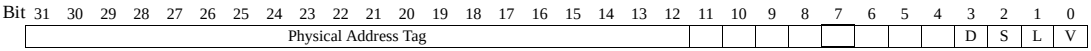
The **CACHE** instruction is serializing to guarantee that the effect of the cache operation is immediately visible to subsequent instructions.

### 2.2.6 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**




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.

#### 2.2.6.1 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
```

#### 2.2.6.2 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. 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.

#### 2.2.6.3 Data Cache Coherency

The data cache snoops coherent system bus transactions to maintain data coherency with other system bus masters (i.e. DMA). If a coherent read transaction on the system bus hits in the data cache, the data cache provides the data. If a coherent write transaction on the system bus 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 un-changed 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 system bus to invalidate cache lines in other caching masters that contain the same address.

The data cache is single-ported; therefore transactions on the system bus are prioritized over accesses by the core. However, the data cache design prevents the system bus from saturating the data cache indefinitely, which guarantees 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 comprised 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.

#### 2.2.6.4 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 region. 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
- Hit Writeback and Invalidate
- Hit Writeback
- Fetch and Lock

The **CACHE** instruction is serializing to guarantee that the effect of the cache operation is immediately visible to subsequent data accesses.

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

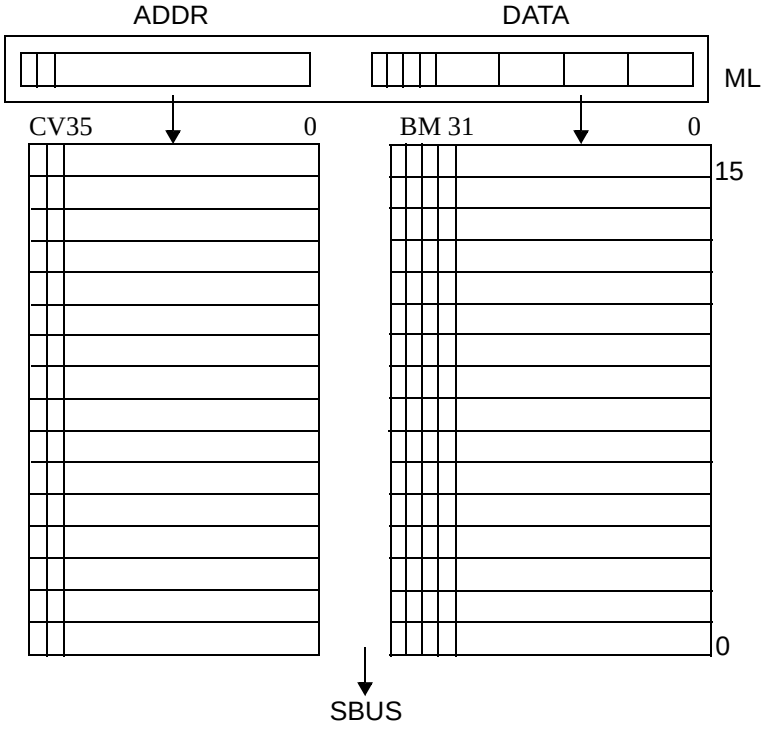
- 0x00 - Normal load
- 0x04 - Streaming load

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 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 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 (ML), where Merging and Gathering decisions are performed, and then travel through the queue. The write buffer arbitrates for the system bus 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 system bus 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 system bus for writing.

When the write buffer has at least one empty entry, processor stores are guaranteed to 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 (ML). 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 0xABCDDEEF. 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* *NOTE: Merging takes places only in the Merge Latch. As such, writes to an address which are in the FIFO (but not in the ML) do not merge. In the example below, writes to 0x0001000 and 0x0001002 do not merge because the intervening write to address 0x00001005 is not in the same word address which caused 0x00001000 to leave the ML.*

*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 memcpy(), 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 system bus, 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 zero for the 0x00001000 entry and is one 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 system bus transactions (e.g. inte-

grated 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

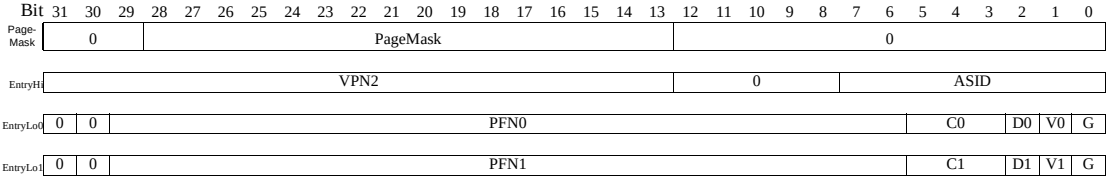
The Au1 implements a TLB-based virtual address translation unit which is compliant with the MIPS32 specification. This scheme is similar to the R4000 TLB and CP0 implementation. The “MIPS32 Architecture For Programmers Volume III” contains all the information relevant to a TLB-based virtual address translation 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 4KB to 16MB.

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 4KB to 16MB, according to Table 3.

**TABLE 3. Values for Page Size and PageMask Register**

| Page Size | PageSize Register | Bits 28:13 | PFN Bit |
|---|---|---|---|
| 4KB | 0x00000000 | 0000000000000000 | 12 |
| 16KB | 0x00006000 | 0000000000000011 | 14 |
| 64KB | 0x0001E000 | 0000000000001111 | 16 |
| 256KB | 0x0007E000 | 0000000000111111 | 18 |
| 1MB | 0x001FE000 | 0000000011111111 | 20 |
| 4MB | 0x007FE000 | 0000001111111111 | 22 |
| 16MB | 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 PFN bit of the program address selects either PFN0 or PFN1 as upper bits of the resulting 36-bit physical address.

The TLB mechanism permits mapping a smaller, 32-bit program address space into a larger 36-bit physical address space. The Au1 implements an internal 36-bit physical address system bus (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. 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 encoding 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 4. 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 5.

**TABLE 5. 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<br>Request 0<br>Request 1 | <br>10<br>11 | <br>10<br>11 |
| Interrupt Controller 1<br>Request 0<br>Request 1 | <br>12<br>13 | <br>12<br>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 Chapter 5.

## 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

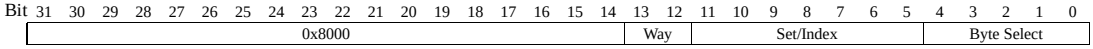
The **CACHE** instruction permits management of the Au1 instruction and data caches. The valid operations are listed in Table 1.

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**




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 16KB 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.

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.

## 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 system bus 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 that data coherency is no longer guaranteed.

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 au1_wait
au1_wait:
   la t0,au1_wait # obtain address of au1_wait
   cache 0x14,0(t0) # fill icache with first 8 insns
   cache 0x14,32(t0) # fill icache with next 8 insns
   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.

In Table 6 the Au1 CP0 registers are listed.

**TABLE 6. Coprocessor 0 Register Definitions**

| Register Number | Sel | Register Name | Description | Compliance |
|---|---|---|---|---|
| 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 |

**TABLE 6. Coprocessor 0 Register Definitions (Continued)**

| Register Number | Sel | Register Name | Description | Compliance |
|---|---|---|---|---|
| 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 |
| 16 | 1 | Config1 | Configuration Register 1 | Required |
| 17 | 0 | LLAddr | Load Link Address | Optional |
| 18 | 0 | WatchLo | Data memory break point low bits | Optional |
| 18 | 1 | IWatchLo | Instruction fetch breakpoint low bits | Optional |
| 19 | 0 | WatchHi | Data memory break point high bits | Optional |
| 19 | 1 | IWatchHi | Instruction fetch breakpoint high bits | Optional |
| 20 | 0 | | Reserved | Reserved |
| 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 | PerfCnt | Performance counter value(s) | Au1 |
| 25 | 1 | PerfCtrl | Performance counter control(s) | Au1 |
| 26 | 0 | | Reserved | Reserved |
| 27 | 0 | | Reserved | Reserved |

**TABLE 6. Coprocessor 0 Register Definitions (Continued)**

| Register Number | Sel | Register Name | Description | Compliance |
|---|---|---|---|---|
| 28 | 0 | DTag | Data cache tag value | Au1 |
| 28 | 1 | DData | Data cache data value | Au1 |
| 29 | 0 | ITag | Instruction cache tag value | Au1 |
| 29 | 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 |

A compliance of “Required” denotes a register required by the MIPS32 architecture.

A compliance of “Optional” denotes an optional register in the MIPS32 architecture which is implemented in the Au1 core.

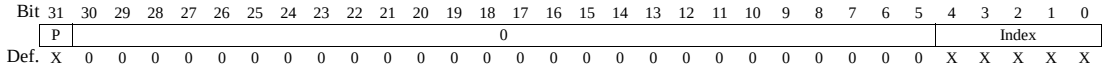
A compliance of “Au1” denotes an optional register unique to the Au1 core.

A compliance of “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**

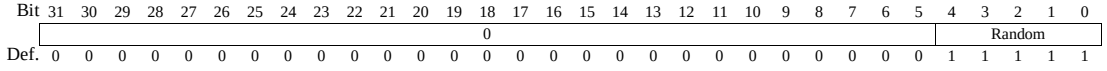



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | P | Probe Failure. | R | UNPRED |
| 30:5 | Reserved | Must always write zeros, always reads zeros | R | 0 |
| 4:0 | Index | TLB Index | R/W | UNPRED |

## 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**

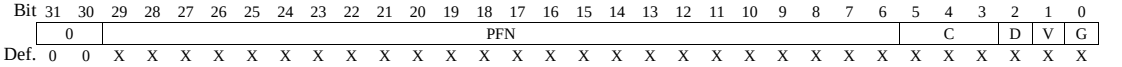


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:5 | Reserved | Must always write zeros, always reads zeros | R | 0 |
| 4:0 | Random | TLB Random Index | R | 31 |

## 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**

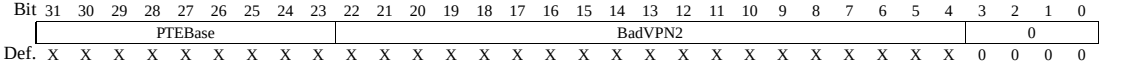



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:30 | Reserved | Ignored on writes, returns zero on read | R | 0 |
| 29:6 | PFN | Page Frame Number. Corresponds to physical address bits 35..12. | R/W | UNPRED |
| 5:3 | C | Cache coherency attribute of the page. See Table 2. | 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**

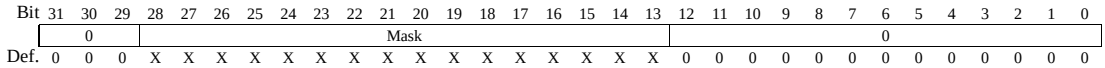


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:23 | PTEBase | Used by the operating system as a pointer into the current PTA array in memory. | R/W | UNPRED |
| 22:4 | BadVPN2 | Contains virtual address bits 31..13 upon a TLB exception. | R | UNPRED |
| 3:0 | Reserved | Reserved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

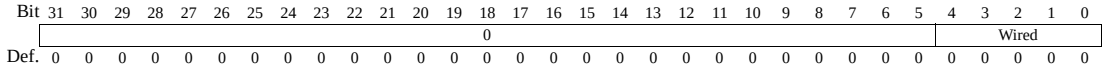



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:29 | Reserved | Ignored on write, returns zero on read. | R | 0 |
| 28:13 | Mask | The Mask field is a bit mask in which a “1” bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. See Table 3. | 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**

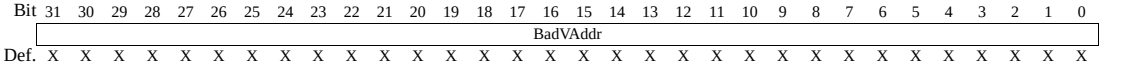



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:5 | Reserved | Ignored on write, returns zero on read. | R | 0 |
| 4:0 | Wired | TLB wired boundary | R/W | 0 |

## 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**

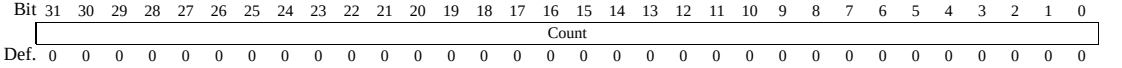



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | BadVAddr | Bad virtual address | R | UNPRED |

## 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.

**Count** **CP0 Register 9, Select 0**




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.

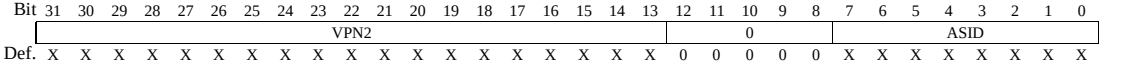
During Sleep mode, this register will not increment.

| Bit(s) | 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**

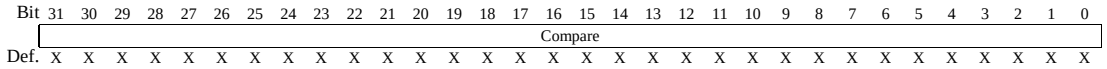


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | VPN2 | Virtual address bits 31..13. | R/W | UNPRED |
| 12:8 | Reserved | Ignored on write, returns zero on read. | R | 0 |
| 7:0 | ASID | Address space identifier | R/W | UNPRED |

## 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**

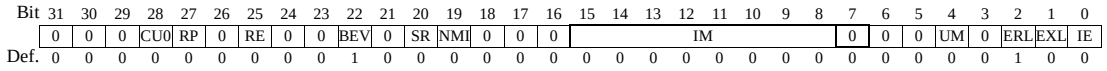



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | Compare | Interval counter compare value | R/W | UNPRED |

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

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

**Status** **CP0 Register 12, Select 0**



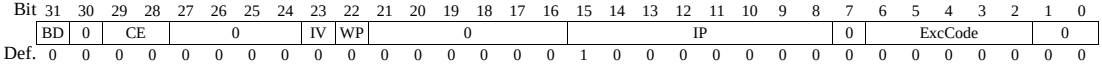

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | CU3 | This bit is zero. Coprocessor 3 is not implemented. | R | 0 |
| 30 | CU2 | This bit is zero. Coprocessor 2 is not implemented. | R | 0 |
| 29 | CU1 | This bit is zero. Coprocessor 1 is not implemented. | R | 0 |
| 28 | CU0 | Controls access to coprocessor 0. | R/W | 0 |
| 27 | RP | Reduced power. This bit has no effect. | R/W | 0 |
| 25 | RE | Reverse-endian. | R/W | 0 |
| 22 | BEV | Boot exception vectors. | R/W | 1 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 20 | SR | Soft reset. | R/W | 0 |
| 19 | NMI | Non-maskable interrupt | R/W | 0 |
| 15:8 | IM | Interrupt mask | R/W | 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**

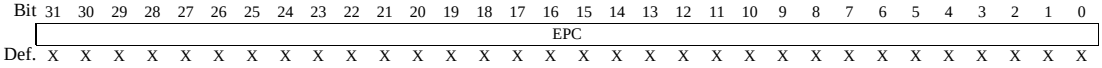



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | BD | Exception in branch delay slot | R | 0 |
| 29:28 | CE | Coprocessor error | R | 0 |
| 23 | IV | Interrupt vector | R/W | 0 |
| 22 | WP | Watchpoint exception deferred | R/W | 0 |
| 15:10 | IP[7:2] | Hardware interrupts pending | R | 0x20 |
| 9:8 | IP[1:0] | Software interrupts pending | R/W | 0 |
| 6:2 | ExcCode | Exception Code | R | 0 |

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

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

**EPC** **CP0 Register 14, Select 0**

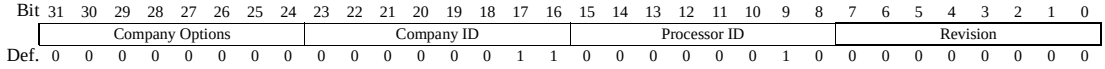


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | EPC | Exception Program Counter | R/W | UNPRED |

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

The PRId register is a required register for processor identification.

**PRId** **CP0 Register 15, Select 0**



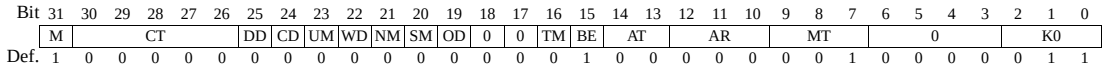

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:24 | Company Options | This field identifies the system-on-a-chip (SOC) identification:<br>0: Au1000<br>1: Au1500<br>2: Au1100 | R | 0 |
| 23:16 | Company ID | This field is assigned to AMD by MIPS Technologies, and contains the value 3. | R | 3 |
| 15:8 | Processor ID | This field identifies the core revision:<br>0: Reserved<br>1: Au1 revision 1<br>2: Au1 revision 2 | R | 2 |
| 7:0 | Revision | This field contains a manufacturing specific revision level. | R | SOC specific |

A complete listing of PRId values is available from AMD.

## 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**

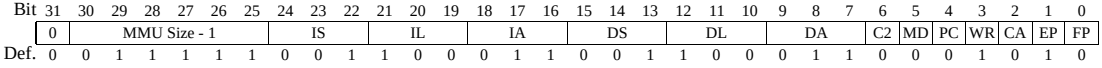


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | M | Denotes Config1 register available at select 1 | R | 1 |
| 30:26 | CT | Reserved, must write 0 | R/W | 0 |
| 25 | DD | Reserved, must write 0 | R/W | 0 |
| 24 | CD | Reserved, must write 0 | R/W | 0 |
| 23 | UM | Reserved, 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**



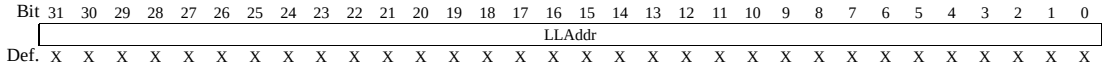

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 30:25 | MMU Size - 1 | Number of entries in the TLB minus one. The TLB has 32 entries. | R | 31 |
| 24:22 | IS | Instruction cache sets per way is 128. | R | 1 |
| 21:19 | IL | Instruction cache line size is 32 bytes. | R | 4 |
| 18:16 | IA | Instruction cache associativity is 4-way. | R | 3 |
| 15:13 | DS | Data cache sets per way is 128. | R | 1 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 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.

**LLAddr** **CP0 Register 17, Select 0**

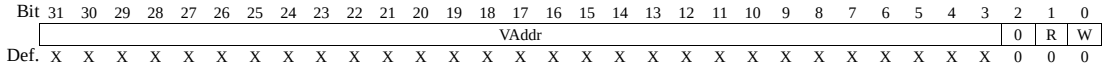



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | LLAddr | Load Linked Address | R | UNPRED |

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

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

**WatchLo** **CP0 Register 18, Select 0**

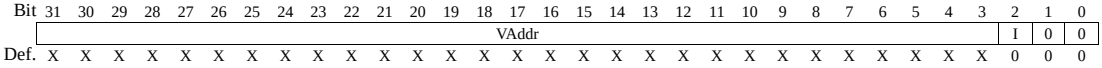


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:3 | VAddr | The virtual address to match | R/W | UNPRED |
| 1 | R | If this bit is a one, then watch exceptions are enabled for loads that match the address. | R/W | 0 |
| 0 | W | If this bit is a one, then watch exceptions are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

**IWatchLo** **CP0 Register 18, Select 1**

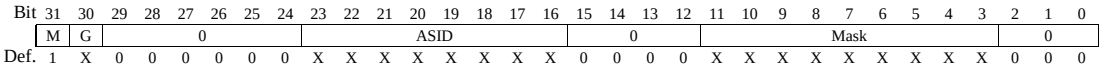



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:3 | VAddr | The virtual address to match | R/W | UNPRED |
| 2 | I | If this bit is a one, then watch exceptions are enabled for instruction accesses that match the address. | R/W | 0 |

## 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**



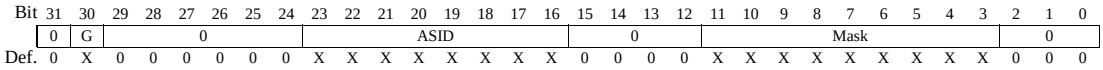

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | M | another pair of Watch registers is implemented at the next Select index. | R | 1 |
| 30 | G | If this bit is one, then the ASID field is ignored and any address that matches causes a watch exception. | R | UNPRED |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 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**

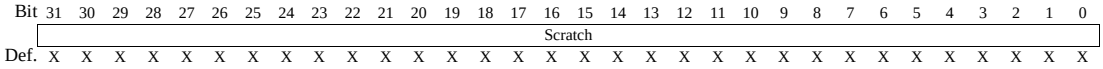



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 30 | G | If this bit is one, then the ASID field is ignored and any address that matches causes a watch exception. | R | UNPRED |
| 23:16 | ASID | ASID value which is required to match that in the EntryHi register if the G bit is 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**

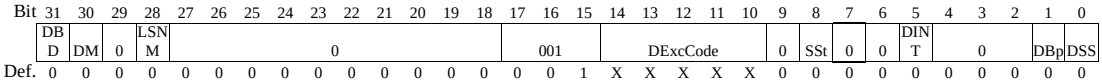


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | Scratch | This register is present for the convenience of software. | R/W | UNPRED |

## 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**

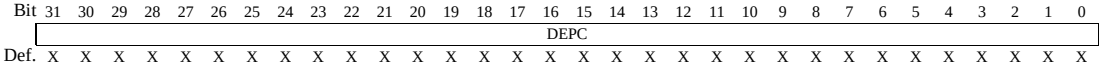



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | DBD | Debug exception in branch delay slot. | R | UNPRED |
| 30 | DM | If this bit is a one, then in debug mode. | R | 0 |
| 28 | LSNM | Load/stores are performed in the normal fashion when in debug mode. | R/W | 0 |
| 17:15 | 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**

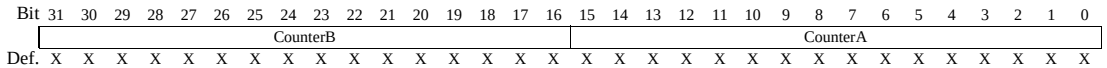


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | DEPC | Debug exception program counter. | R/W | UNPRED |

## 2.7.25 Performance Counter Register (CP0 Register 25, Select 0)

The PerfCnt and PerfCtrl registers are the interface to the performance counter facility. The performance counter implementation is unique to the Au1.

**PerfCnt** **CP0 Register 25, Select 0**

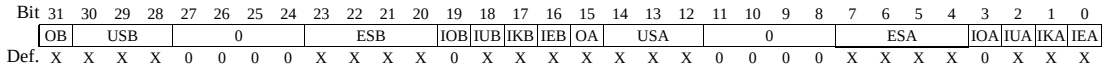



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | CounterB | Value of performance counter B. | R/W | UNPRED |
| 15:0 | CounterA | Value of performance counter A. | R/W | UNPRED |

## 2.7.26 Performance Control Register (CP0 Register 25, Select 1)

The PerfCnt and PerfCtrl registers are the interface to the performance counter facility.

**PerfCtrl** **CP0 Register 25, Select 1**




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | OB | Overflow detect for counter B. | R/W | UNPRED |
| 30:28 | USB | Unit select for 1 of 16 units as source of events for performance counter B. | R/W | UNPRED |
| 23:20 | ESB | Event select for 1 of 16 signals from selected unit for performance counter B. | R/W | UNPRED |
| 19 | IOB | Enable interrupt on overflow of counter B. | R/W | 0 |
| 18 | IUB | If this bit is one, event counting for performance counter B continues while in user mode (Status[ERL]=0b0 and Status[EXL]=0b0 and Status[UM]=0b1). | R/W | UNPRED |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 17 | IKB | If this bit is one, event counting for performance counter B continues while in kernel mode (Status[ERL]=0b0 and Status[EXL]=0b0 and Status[UM]=0b0). | R/W | UNPRED |
| 16 | IEB | If this bit is one, event counting for performance counter B continues during exception/interrupt processing (Status[ERL]=0b1 or Status[EXL]=0b1). | R/W | UNPRED |
| 15 | OA | Overflow detect for counter A. | R/W | UNPRED |
| 14:12 | USA | Unit select for 1 of 16 units as source of events for performance counter A. | R/W | UNPRED |
| 7:4 | ESA | Signal select for 1 of 16 signals from selected unit for performance counter A. | R/W | UNPRED |
| 3 | IOA | Enable interrupt on overflow of counter A. | R/W | 0 |
| 2 | IUA | If this bit is one, event counting for performance counter A continues while in user mode (Status[ERL]=0b0 and Status[EXL]=0b0 and Status[UM]=0b1). | R/W | UNPRED |
| 1 | IKA | If this bit is one, event counting for performance counter A continues while in kernel mode (Status[ERL]=0b0 and Status[EXL]=0b0 and Status[UM]=0b0). | R/W | UNPRED |
| 0 | IEA | If this bit is one, event counting for performance counter A continues during exception/interrupt processing (Status[ERL]=0b1 or Status[EXL]=0b1). | R/W | UNPRED |

The PerfCnt register provides read/write access to the current value of both performance counters A and B. The PerfCnt register concatenates the two 16-bit counters into a one 32-bit register. A read of this register returns the current count for each performance counter. A write to this register will preload the counters to new values. The performance counters are reset to zeros when a write to the PerfCtrl register is written.

NOTE: USx=0 and ESx=0 is constant so that the counter is effectively disabled and thus consumes no power.

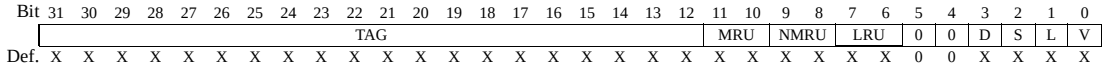
A listing of the valid units and events can be obtained from AMD.

### 2.7.27 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**



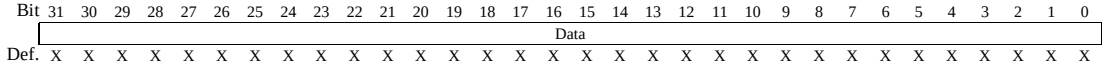

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:12 | TAG | TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero. | R/W | UNPRED |
| 11:10 | MRU | Most recently used way. | R/W | UNPRED |
| 9:8 | NMRU | Next most recently 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 over-writing of the cache block. | R/W | UNPRED |
| 0 | V | Cache block valid. | R/W | UNPRED |

## 2.7.28 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.

**DData** **CP0 Register 28, Select 1**



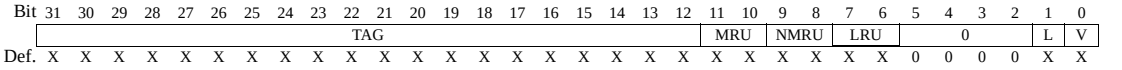

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | Data | Data from the data cache line. | R | UNPRED |

## 2.7.29 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.

**ITag** **CP0 Register 29, Select 0**



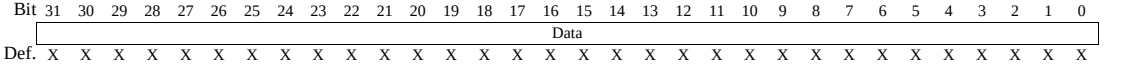

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:12 | TAG | TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero. | R/W | UNPRED |
| 11:10 | MRU | Most recently used way. | R/W | UNPRED |
| 9:8 | NMRU | Next most recently 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 over-writing of the cache block. | R/W | UNPRED |
| 0 | V | Cache block valid. | R/W | UNPRED |

## 2.7.30 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.

**IData** **CP0 Register 29, Select 1**

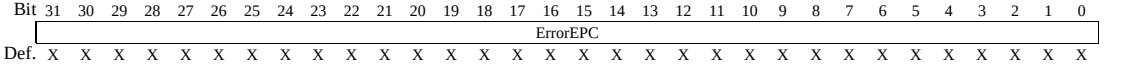



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | Data | Data from the instruction cache line. | R | UNPRED |

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

The ErrorEPC register is a required register for exception processing.

**ErrorEPC** **CP0 Register 30, Select 0**

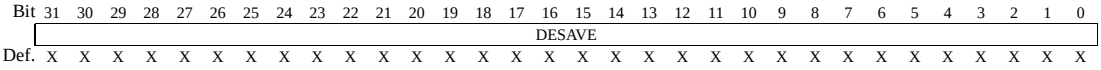


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | ErrorEPC | Error Exception Program Counter | R/W | UNPRED |

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

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

**DESAVE** **CP0 Register 31, Select 0**




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | DESAVE | Debug save scratch register, for debug handlers. | R/W | UNPRED |

## 2.8 System Bus

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

### 2.8.1 SBUS Arbitration

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

The system bus requestors in the Au1100 are:

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

- DMA channels (8)

The Au1 presents a single request to the system bus 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.

### 2.8.2 SBUS Coherency Model

The SBUS is the coherency point within the Au1100. 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.

# 3 Memory Controllers

The Au1100 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 2.5 V and 3.3 V.

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 4MB initially installed would locate the memory at physical address 0. Normally, adding 16MB would create a 12MB "hole" in the memory map. With the address configuration options in the Au1100 the 4MB can be relocated to start at 16M and the new memory can be located at 0 to allow a 20MB contiguous memory pool.

All registers in the Memory Controller block are located off of the base address shown in Table 7.

**TABLE 7. Memory Controller Block Base Address**

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

Table 83 shows how the state of $\overline{\text{ROMSEL}}$ and $\overline{\text{ROMSIZE}}$ will determine where the processor boots from. The system designer has the choice of booting from 32 bit flash, 16 bit flash, 32 bit SMROM or 32 bit SyncFlash. The $\overline{\text{ROMSEL}}$ and $\overline{\text{ROMSIZE}}$ configuration is discussed in more detail in Section 8.2, "Reset".

## 3.1 SDRAM Memory Controller

The SDRAM memory controller of the Au1100 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 speed. The system bus defaults to 1/2 the processor clock speed so that SDRAM or SyncFlash will run at 99MHz with a 396 MHz Au1100. SMROM operates at 1/4 the speed of the system bus. The system bus divider is programmable, see Section 7.4.3, "Device Power Management - Sleep" for more information.

The SDRAM interface supports three chip selects, 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 Sync-Flash). For chip selects configured as SDRAM or SyncFlash the controller will keep one row open for each chip select allowing faster accesses and reducing the need to issue precharge cycles.

When $\overline{\text{RESETIN}}$ is deasserted, code is fetched from SMROM/SyncFlash if SMROM/Sync-Flash boot is selected. When using SMROM or SyncFlash for boot, the SMROMCKE (muxed with GPIO6) should be used for the SMROM/SyncFlash CKE. If SMROM or Sync-Flash are being used, but not for boot, then SDCKE should be used for the clock enable. Please contact AMD technical support for detailed information on booting from SMROM or Syncflash.

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 100uS for the SDRAM clock to stablize 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 that the external SDRAM is configured to a burst of 8 mode.

### 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 7. Table 8 shows the memory map of the register block.

**TABLE 8. SDRAM Configuration Registers**

| Offset | Register Name | Description |
|---|---|---|
| 0x0000 | mem_sdmode0 | Chip Select 0 Mode Configuration (Timing and Functionality) |
| 0x0004 | mem_sdmode1 | Chip Select 1 Mode Configuration (Timing and Functionality) |
| 0x0008 | mem_sdmode2 | Chip Select 2 Mode Configuration (Timing and Functionality) |
| 0x000c | mem_sdaddr0 | Chip Select 0 Address Configuration and Enable |
| 0x0010 | mem_sdaddr1 | Chip Select 1 Address Configuration and Enable |
| 0x0014 | mem_sdaddr2 | Chip Select 2 Address Configuration and Enable |
| 0x0018 | mem_sdrefcfg | Refresh Configuration and Timing |
| 0x001c | mem_sdprecmd | Issue Precharge to all enabled chip selects |
| 0x0020 | mem_sdautoref | Issue Auto Refresh to all enabled chip selects |
| 0x0024 | mem_sdwrmd0 | Write data to CS0 SDRAM mode register |
| 0x0028 | mem_sdwrmd1 | Write data to CS1 SDRAM mode register |
| 0x002C | mem_sdwrmd2 | Write data to CS2 SDRAM mode register |
| 0x0030 | mem_sdsleep | Force SDRAM into self refresh mode |
| 0x0034 | mem_sdsmcke | Toggle SMROMCKE pin |

#### 3.1.1.1 SDRAM Registers

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

**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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | SF | F | SR | BS | RS | | | | CS | | | | Tras | | | | Tmrd | | Twr | | | Trp | | | Trcd | | | Tcl | | |
| Def. | 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 | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:24 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 23 | SF | Selects SyncFlash operation. SyncFlash is only available on CS0. For other chip selects this bit is reserved and should be written as 0. | R/W | 0 |
| 22 | F | Setting the F bit allows the SDRAM controller to assume that no cacheing master except the core will access this memory space. This allows accesses to begin sooner.<br>For the Au1100 the CPU core is the only possible cacheing master so this bit may always be set. | R/W | 0 |
| 21 | SR | Chip select operating mode<br>0: SDRAM Operation<br>1: SMROM Operation | R/W | See above |
| 20 | BS | Select Number of Banks<br>0: Chip select controls 2 bank SDRAM<br>1: Chip select controls 4 bank SDRAM<br>This bit must be set to 0 for SMROM support | R/W | See above |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 19:18 | RS | This field sets the number of bits in the row address as shown below:<br>RS Row Address Size<br>0 11<br>1 12<br>2 13<br>3 Reserved | R/W | See above |
| 17:15 | CS | This field sets the number of bits in the column address as shown below:<br>CS Column Size<br>0 7<br>1 8<br>2 9<br>3 10<br>4 11<br>5 Reserved<br>6 Reserved<br>7 Reserved | R/W | See above |
| 14:11 | Tras | This field designates the minimum delay from a activate to a precharge command.<br>The value in the Tras field must be one less than the minimum number of SDRAM clock cycles required. | 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.<br>The value in the Tmrd field must be one less than the required number of SDRAM 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.<br>The value in the Twr field must be one less than the required number of SDRAM clock cycles. | R/W | 3 |
| 6:5 | Trp | This field sets the time from precharge to the next activate command.<br>The value of the Trp field must be one less than the required number of SDRAM clock cycles. | R/W | 3 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 4:3 | Trcd | This field sets the RAS to CAS delay.<br>The value of the Trcd field must be one less than the required number of SDRAM 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.<br>The value in the Tcl field must be one less than the required number of SDRAM clock cycles. | R/W | See Above |

### Chip Select Address Configuration Registers

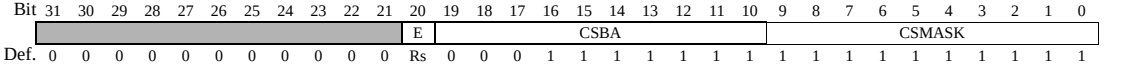
The chip select address configuration registers select the physical addresses for each chip select. Each register contains a base address and a mask. When the following equation is met

*(physical_addr & CSMASK) == CSBA*

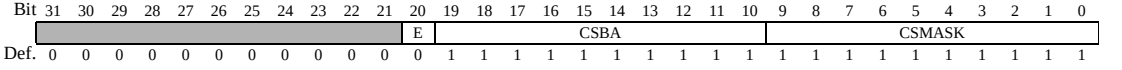
the given chip select is activated. *Physical_addr* is the actual address as output on the internal system bus (from the TLB for memory mapped regions), *CSMASK* is the mask set below and *CSBA* is the chip select base address as programmed below. Chip select regions must be programmed so that each chip select occupies a unique area of the physical address space. Programming overlapping chip select regions will result in undefined operation.

The E bits for chip selects CS1 and CS2 are initialized to zero. The E bit to CS0 is set non-zero when the ROM_SEL and ROM_SIZE pins indicate that the SMROM/SyncFlash should be used for the boot vector (ROM_SEL==1, ROM_SIZE==0).

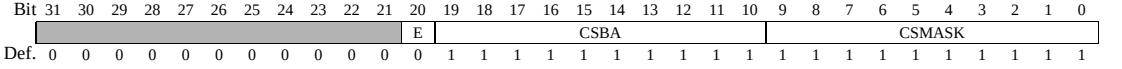
**mem_sdaddr0 - CS0 Address Configuration** Offset = 0x000C




**mem_sdaddr1 - CS1 Address Configuration** Offset = 0x0010




**mem_sdaddr2 - CS2 Address Configuration** Offset = 0x0014

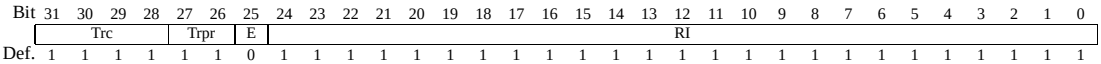


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:21 | RES | These bits are reserved and should be written a 0 | R | 0 |
| 20 | E | When the E bit is set the chip select is enabled. Clearing the E bit will prevent the chip select from becoming activated on a matching address compare. | R/W | See above |
| 19:10 | CSBA | The CSBA field becomes bits 31:22 of the chip select base address. The lower bits of the base address are zero. | R/W | See above |
| 9:0 | CSMASK | The bits in the CSMASK field become bits 31:22 of the chip select comparison mask. The lower bits of the mask are set to zero. | R/W | See above |

### 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




| Bit(s) | 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.<br>The value of the Trc field must be one less than the minimum number of SDRAM clock cycles. | R/W | 0xf |

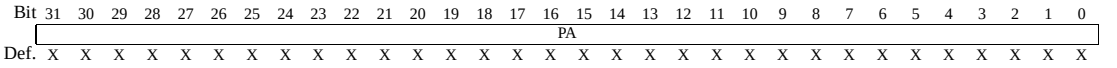
| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 27:26 | Trpm | This field specifies the minimum time from a precharge to the start of a refresh cycle for all SDRAM chip selects. This is used because a precharge all command is automatically initiated before an auto refresh command. This value should be programmed with the worst case Trp from the **sdr_csmode*n*** registers.<br>The value of the Trpr field must be one less than the minimum number of SDRAM 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 system bus clocks for all SDRAM ranks.<br>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 64ms (all rows), the individual refresh interval will be 15.7us (64ms/4096). With a system bus clock of 198MHz, the RI value should be 0xC24. (15.7us/(1/198MHz). | R/W | 0x1FFFFFF |

### 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



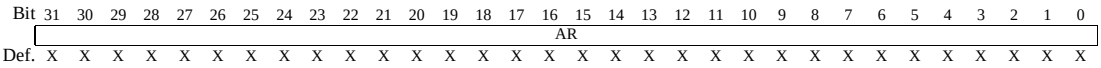
| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | PA | Writing any value to PA will cause a precharge command to be issued to all enabled SDRAM chip selects. | W | UNPRED |

### 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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | AR | Writing any value to AR will cause an Auto Refresh command to be issued to all enabled SDRAM chip selects. | R/W | UNPRED |

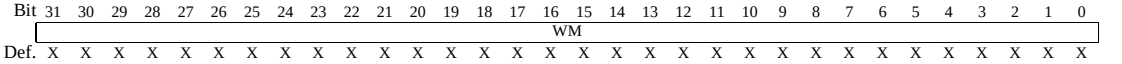
### External SDRAM Mode Register Access

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



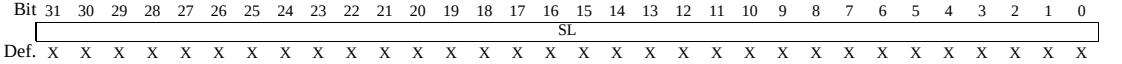

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | WM | The value written to this register will be written to the external SDRAM mode register for the corresponding chip select. | W | UNPRED |

**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

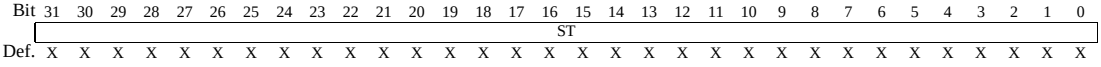



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | SL | Writing any value to SL will issue a self refresh command on all enabled chip selects. | W | UNPRED |

### SMROMCKE Toggle Register

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

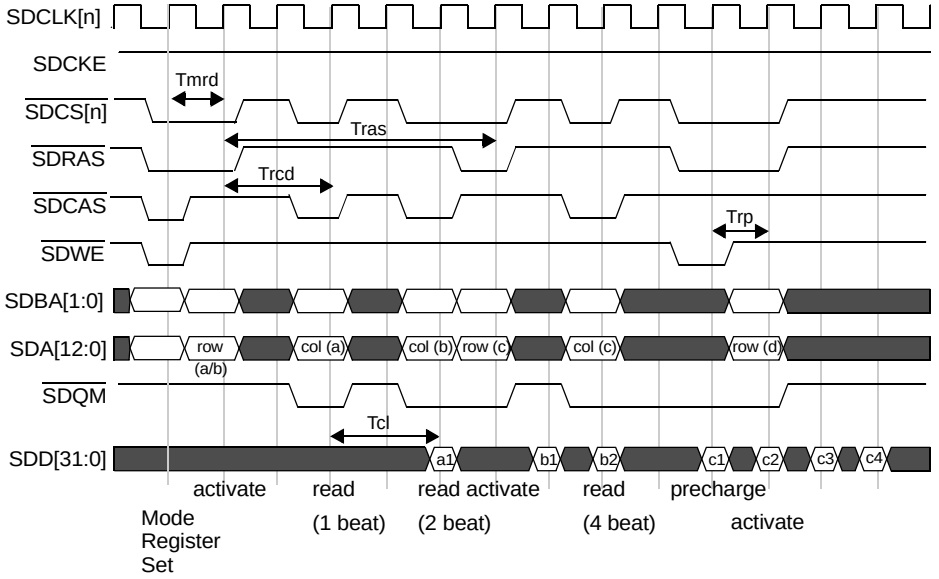
**mem_sdsmcke -SMROMCKE Toggle** Offset = 0x0034




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | ST | Writing any value to ST will invert the current state of the SMROMCKE pin. | W | UNPRED |

### 3.1.1.2 SDRAM Timing

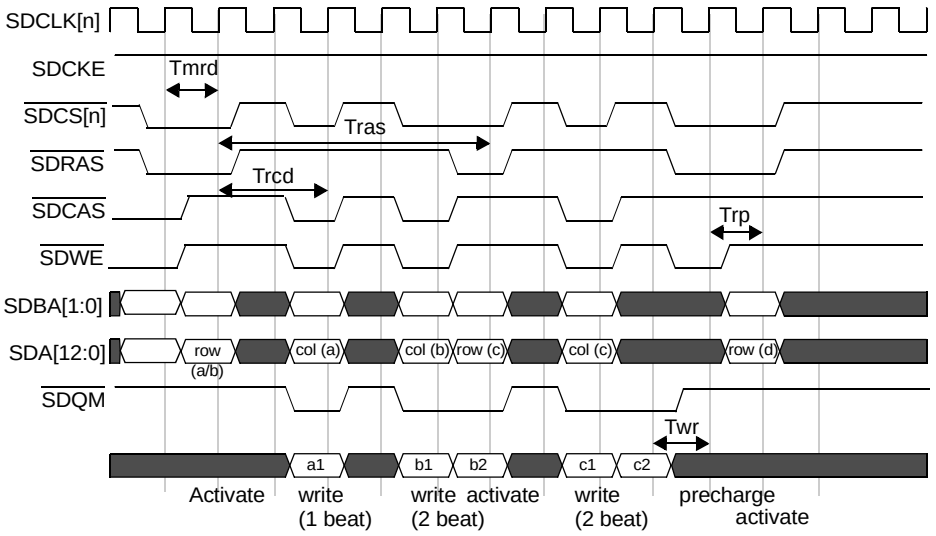
The following figures show examples of typical read, typical write and refresh timing..




The above timing represents the following:
1. Tras = 4 (5 SDRAM clock cycles)
2. Trp = 0 (1 SDRAM clock cycles)
3. Trcd = 1 (2 SDRAM clock cycles)
4. Tcl = 1(2 SDRAM clock cycles)
5. Tmrd = 0(2 SDRAM clock cycles)

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

**FIGURE 4. SDRAM Typical Read Timing**



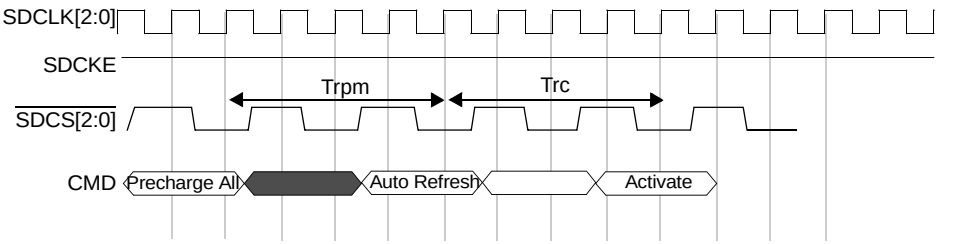

Mode Register Set

The above timing represents the following:

1. Tras = 4 (5 SDRAM clock cycles)
2. Trp = 0 (1 SDRAM clock cycles)
3. Trcd = 1 (2 SDRAM clock cycles)
4. Tmrd = 0(1 SDRAM clock cycles)

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

**FIGURE 5. SDRAM Typical Write Timing**




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

The above timing represents the following:
1. Trpm = 3 (4 SDRAM clock cycles)
2. Trc = 3(4 SDRAM clock cycles)

**FIGURE 6. SDRAM Refresh Timing**

## 3.1.2 SDRAM Hardware Considerations

Table 9 shows the signals associated with the SDRAM interface.

**TABLE 9. SDRAM Signals**

| Pin Name | 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 Inputs: BA0 and BA1 define to which bank the ACTIVE, READ, WRITE or PRECHARGE command is being applied. |
| SDD[31:0] | IO | SDRAM data bus |
| $\overline{\text{SDQM[3:0]}}$ | O | Active Low Input/Output Mask: SDQM is a mask signal for write accesses and an output enable signal for read accesses. SDQM0 masks SDD7:0, SDQM1 masks SDD15:8, SDQM2 masks SDD23:16, SDQM3 masks SDD31:24. |

**TABLE 9. SDRAM Signals (Continued)**

| Pin Name | Input/ Output | Description |
|---|---|---|
| $\overline{\text{SDRAS}}$ | O | Active Low Command Output. $\overline{\text{SDRAS}}$, $\overline{\text{SDCAS}}$ and $\overline{\text{SDWE}}$ (along with $\overline{\text{SDCSn}}$) define the command being sent to the SDRAM rank. |
| $\overline{\text{SDCAS}}$ | O | Active Low Command Output. $\overline{\text{SDRAS}}$, $\overline{\text{SDCAS}}$ and $\overline{\text{SDWE}}$ (along with $\overline{\text{SDCSn}}$) define the command being sent to the SDRAM rank. |
| $\overline{\text{SDWE}}$ | O | Active Low Command Output. $\overline{\text{SDRAS}}$, $\overline{\text{SDCAS}}$ and $\overline{\text{SDWE}}$ (along with $\overline{\text{SDCSn}}$) define the command being sent to the SDRAM rank. |
| SDCLK[2:0] | O | Clock output corresponding to each of the three chip selects. Clock speed is 1/2 system bus frequency when corresponding $\overline{\text{SDCSn}}$ is set to SDRAM or SyncFlash, 1/4 system bus frequency when corresponding $\overline{\text{SDCSn}}$ is set to SMROM. |
| $\overline{\text{SDCS[2:0]}}$ | O | Active Low 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.<br>*Muxed with GPIO6. If ROMSEL and ROMSIZE are configured to boot from Synchronous Mask ROM, SMROMCKE will control the pin out of reset, else GPIO6 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 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_stcfg*n*** register. All device types utilize the address and data bus, RAD[31:0] and RD[31:0] respectively.

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

Any read to the static bus will cause a 32 bit access. This can cause a potential problem with volatile devices as a single 16 bit read will result in two 16 bit reads on the external bus.

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

While the static bus controller is a synchronous device internally, there is no external clock available to reference the control signals.

### 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 7. Table 10 shows the registers and offsets for the static bus controller.

**TABLE 10. Static Bus Controller Configuration Registers**

| Offset | Register Name | Description |
|---|---|---|
| 0x1000 | mem_stcfg0 | Configuration for $\overline{\text{RCS0}}$ |
| 0x1004 | mem_sttime0 | Timing parameters for $\overline{\text{RCS0}}$ |
| 0x1008 | mem_staddr0 | Address region control for $\overline{\text{RCS0}}$ |
| 0x1010 | mem_stcfg1 | Configuration for $\overline{\text{RCS1}}$ |
| 0x1014 | mem_sttime1 | Timing parameters for $\overline{\text{RCS1}}$ |
| 0x1018 | mem_staddr1 | Address region control for $\overline{\text{RCS1}}$ |
| 0x1020 | mem_stcfg2 | Configuration for $\overline{\text{RCS2}}$ |

**TABLE 10. Static Bus Controller Configuration Registers**

| Offset | Register Name | Description |
|---|---|---|
| 0x1024 | mem_sttime2 | Timing parameters for $\overline{RCS2}$ |
| 0x1028 | mem_staddr2 | Address region control for $\overline{RCS2}$ |
| 0x1030 | mem_stcfg3 | Configuration for $\overline{RCS3}$ |
| 0x1034 | mem_sttime3 | Timing parameters for $\overline{RCS3}$ |
| 0x1038 | mem_staddr3 | Address region control for $\overline{RCS3}$ |

### Static Bus Configuration Registers

The static bus configuration registers set 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 11 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 program the TLB for use with an LCD controller, bits 29:26 of CoProcessor register Entry Lo must be set to 1110b (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 | DV | 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 |

**mem_stcfg1** Offset = 0x1010
**mem_stcfg2** Offset = 0x1020
**mem_stcfg3** Offset = 0x1030

| 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 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:22 | RES | This field is reserved and should be written as zero. | R | 0 |
| 21 | AS | Setup address before chip select on reads. | R/W | 0 |
| 20 | S | Synchronous mode: when this bit is set all static bus signals are synchronized to LCLK; values in the **mem_sttime** register are programmed in LCLK clock resolution.<br>Note: Synchronous Mode cannot be used when set as 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 Tcsh.<br>Note: a one-word transfer to a 16-bit bus is treated as a burst. | R/W | 0 |
| 18 | PH | Block Phantom mode: Deassert chip select, output enable, and write enable when RBEN[1:0] are not asserted.<br>Note: this bit only affects 16-bit mode. | R/W | 0 |
| 17 | V | Volatile: for single word reads, RBEN[3:0] is asserted only for the bytes explicitly addressed. This is valid for both 16 and 32 bit mode. | R/W | 0 |
| 16 | TA | Apply Tcsh after both writes and reads.<br>This bit is a global attribute and is only present in mem_stcfg0. | R/W | 0 |
| 15:13 | DIV | Setting these bits will adjust the divsor for the LCLK output clock. The clock frequency is set by<br>LCLK = (System Bus Clock / 2) / (DIV + 1)<br>Note: LCLK should not exceed 50 MHz.<br>This bit is a global attribute and is only present in mem_stcfg0. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 12 | BV | When this bit is set the burst size for static transfers will be output on the LCD controller pins 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. All beats are guaranteed to occur. The mapping of the burst size to pins is shown in Table 12.<br>This bit is a global attribute and is only present in mem_stcfg0. | R/W | 0 |
| 11 | DV | Debug Visibility: Setting this bit will place information for internal accesses to the system bus 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.<br>This bit is a global attribute and is only present in mem_stcfg0. | R/W | 0 |
| 10 | AV | Setting this bit will place the address for all internal accesses to the system bus 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.<br>This bit is a global attribute and is only present in mem_stcfg0. | R/W | 0 |
| 9 | BE | Endianness<br>0 - Little Endian<br>1 - Big Endian<br>This bit should be set to match the endianess of the processor. This bit should not be set for PCMCIA. | R | 0 |
| 8 | TS | The TS bit selects the timing scale for the chip select. When TS is cleared the timing is derived from the system bus clock. Setting TS causes timing to be based on the system bus clock divided by 4 resulting in larger timing granularity and potentially longer access times. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 7 | EW | When the EW bit is set the $\overline{\text{EWAIT}}$ input is allow to stretch the bus access time. Chip selects operating in LCD or PCMCIA mode have different wait mechanisms. The EW bit does not apply to chip selects in these modes. | R/W | 0 |
| 6 | H | If the H bit is set this chip select will function as a 16 bit wide bus. When the H bit is clear the chip select will function as a 32 bit wide bus. In 16 bit mode data is presented and accepted on bits 15:0 of the data bus.<br>This bit should not be set when the device type is set for PCMCIA. This bit should be set for LCD device type. | R/W | 0 |
| 5 | BS | If BS is zero then the burst size for page mode accesses is 4 beats. If BS is nonzero then 8 beat accesses will be used. | R/W | 0 |
| 4 | PM | If the PM bit is set the chip select will operate in page mode. This allows quicker access to sequential locations in memory. Page mode is only applicable to reads. | 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 | The DTY field describes the type of device controlled by the static controller chip select. A list of device types and encodings is shown in Table 11.<br>Programming multiple chip selects as LCD or PCMCIA is illegal. Only one of each is supported. | R/W | 3 |

### TABLE 11. Device Type Encoding

| DTY | Chip Select Function | PFN[35:32] (upper nibble of physical address) | Reference |
|---|---|---|---|
| 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 ($\overline{CE2}$ only) | 0xE | Section 3.2.4 |
| 5 | Reserved | | |
| 6 | Reserved | | |
| 7 | Reserved | | |

### TABLE 12. Burst Size Mapping

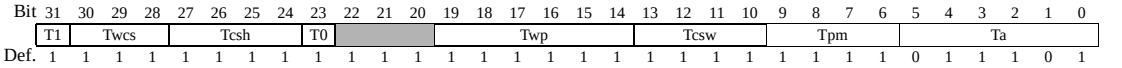
| Signal | Pin |
|---|---|
| burst_size[2] | $\overline{LWR0}$ |
| burst_size[1] | $\overline{LRD1}$ |
| burst_size[0] | $\overline{LRD0}$ |

#### Static Timing Registers

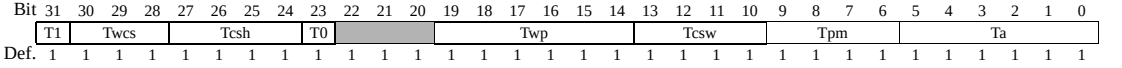
The static timing registers allow software to control the timing of each phase of a static bus access. The names of the timing parameters correspond directly to timing parameters shown on the timing diagrams.

For asynchronous chip selects, all timing parameters are expressed as a number of clock cycles. The clock is either system bus clock or the system bus clock divided by 4, depending on the TS parameter in the **mem_stcfg*n*** register. However, for synchronous chip selects the clock is LCLK and timing parameters are expressed as a number of LCLK cycles.

**mem_sttime0 (I/O, Flash, SRAM config)** Offset = 0x1004



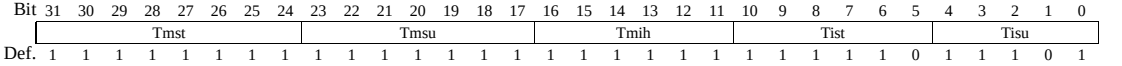

**mem_sttime1 (I/O, Flash, SRAM config)** Offset = 0x1014
**mem_sttime2 (I/O, Flash, SRAM and LCD config)** Offset = 0x1024
**mem_sttime3 (I/O, Flash, SRAM config)** Offset = 0x1034



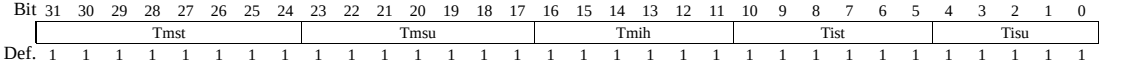

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31 | T1 | Most significant bit of Tcs_oe[1:0]. Tcs_oe represents the number of clocks to setupthe address before asserting chip select and output enable. | R | 0x0 |
| 30:28 | Twcs | This field specifies the required chip select hold time after a write pulse.<br>(Twcs + 1) is the actual number of cycles. | R/W | 0x03 |
| 27:24 | Tcsh | This field specifies the required number of cycles that the chip select must remain deasserted between accesses.<br>(Tcsh + 1) is the actual number of cycles. | R/W | 0x0f |
| 23 | T0 | Least significant bit of Tcs_oe[1:0]. Tcs_oe represents the number of clocks to setupthe address before asserting chip select and output enable. | R/W | 0x0 |
| 22:20 | RES | These bits are reserved and should be written a 0. | R | 0x00 |
| 19:14 | Twp | This field specifies the duration of the write enable.<br>(Twp + 1) is the actual number of cycles. | R/W | 0x3f |
| 13:10 | Tcsw | This field sets the delay from the assertion of chip select until the write strobe is asserted.<br>The timing is as follows:<br>(Twcsw + 1) is the actual number of clock cycles | R/W | 0x0f |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 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 **static_config*n*** register.<br>The actual read happens at (Tpm + 1) clock cycles.<br>Ta will determine the access time for the first beat of each burst. | R/W | 0x0f |
| 5:0 | Ta | The Ta parameter determines the number of cycles required from the assertion of chip select until read data is valid.<br>For page mode accesses Ta determines the access time for only the first beat of each burst, or the first beat after a page mode wrap.<br>(Ta + 1) is the actual number of clock cycles | R/W | 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




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:24 | Tmst | This field specifies the strobe width during memory accesses to PCMCIA chip selects.<br>(Tmst + 2) is the actual number of cycles to the end of the strobe, however the actual read occurs at (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.<br>(Tmsu + 2) is the actual number of 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 PCM-CIA chip selects.<br>(Tmih + 2) is the actual number of cycles. | R/W | 0x3f |
| 10:5 | Tist | This field specifies the strobe width for I/O accesses for a chip select configured for PCM-CIA.<br>(Tist + 2) is the actual number of cycles to the end of the strobe, however the actual read occurs at (Tist + 1). | R/W | 0x3f |
| 4:0 | Tisu | This field specifies the setup time from chip select to strobe during I/O accesses for PCM-CIA.<br>(Tisu+ 2) is the actual number of cycles. | R/W | 0x1f |

### Static Chip Select Address Registers

The chip select address registers determine the address range that each chip select will respond to. A chip select will become active when E is set and

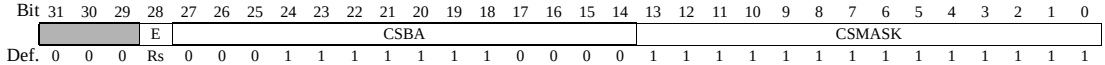
*(physical_addr & CSMASK) == CSBA*

the given chip select is activated. *Physical_addr* is the actual 36 bit physical address as output on the internal system bus (from the TLB for memory mapped regions), *CSMASK* is the mask set below and *CSBA* is the chip select base address as programmed below. The upper nibble, bits 35:32, of *CSBA* are determined by the DTY encoding from the **mem_stcfgn** register.

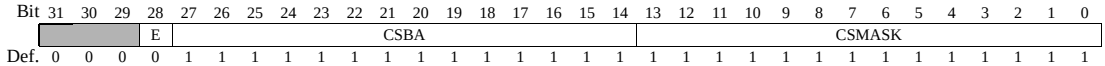
Chip select regions must be programmed so that each chip select occupies a unique area of the physical address space. Programming overlapping chip select regions will result in undefined operation.

The E bits for chip selects CS1, CS2, and CS3 are initialized to zero. The E bit to CS0 is set nonzero when the ROM_SEL pin is zero indicating that ROM should be used for the boot vector.

**mem_staddr0** Offset = 0x1008




**mem_staddr1** Offset = 0x1018
**mem_staddr2** Offset = 0x1028
**mem_staddr3** Offset = 0x1038




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:29 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 28 | E | Enable<br>0: Chip select is not enabled<br>1: Chip select is enabled | R/W | 0 (see note) |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 27:14 | CSBA | This field specifies bits 31:18 of the physical base address for this chip select. Thee upper nibble of the chip select address is determined by the DTY encoding in the **mem_stcfg** register. | R/W | 0x3fff |
| 13:0 | CSMASK | This field specifies which bits of CSBA are used to decode this chip select (see text). | R/W | 0x3fff |

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

This section describes the Static Ram interface which is implemeneted when the DTY field is set to 0, 1 or 3.

The Static RAM, I/O device and Flash device types are all very similar. The I/O Device type is identical to the Static RAM type except that is expecting the upper nibble of the system address (bits 35:32) to be a 0xD (see Static Bus Configuration Registers description for more information). 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 (Twcs in the **mem_sttime*n*** register).

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 13.

**TABLE 13. Static RAM, I/O Device and Flash Control Signals**

| Pin Name | Input/Output | Description |
|---|---|---|
| RAD[31:0] | O | Address bus |
| RD[31:0] | IO | Data bus |
| $\overline{\text{RBEN}}$[3:0] | O | Active Low Byte Enable. $\overline{\text{RBEN0}}$ corresponds to RD7:0, $\overline{\text{RBEN1}}$ corresponds to RD15:8, $\overline{\text{RBEN2}}$ corresponds to RD23:16, $\overline{\text{RBEN3}}$ corresponds to RD31:24. |
| $\overline{\text{RWE}}$ | O | Write enable |

**TABLE 13. Static RAM, I/O Device and Flash Control Signals**

| Pin Name | Input/Output | Description |
|---|---|---|
| $\overline{\text{ROE}}$ | O | Output enable |
| $\overline{\text{RCS[3:0]}}$ | O | Programmable Chip Selects (4 banks). $\overline{\text{RCSn}}$ is not used when configured as a PCMCIA device. |
| $\overline{\text{EWAIT}}$ | I | This active low input can be used to stretch the bus access time when enabled through bit EW in the **mem_stcfg*n*** register. This input is not recognized for chip selects configured as LCD or PCMCIA as these buses have their own wait mechanisms. |

### 3.2.2.1 Static Memory Timing

The following figures show static memory timing. Figure 7 illustrates static memory read timing, and Figure 9 illustrates static memory write timing. The $\overline{\text{EWAIT}}$ timing diagrams are presented to show how $\overline{\text{EWAIT}}$ will hold the cycle past Ta for reads and Twp for writes.

Setup, hold and delay are presented in Chapter 11, Electrical Specifications.

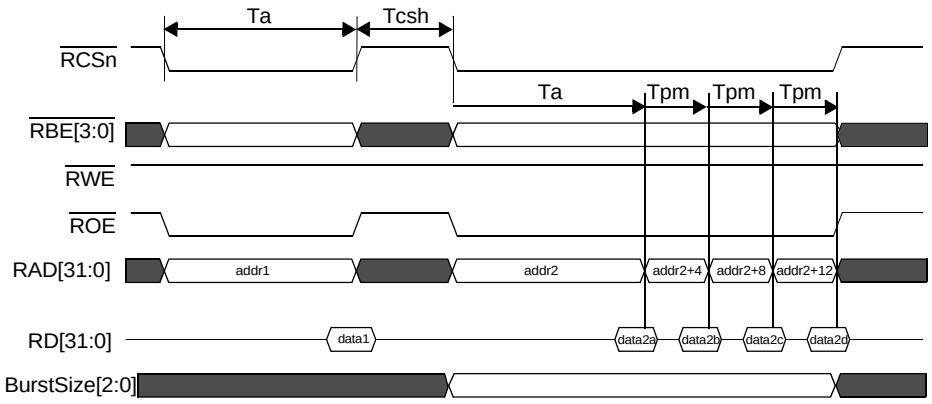



**FIGURE 7. Static Memory Read Timing**

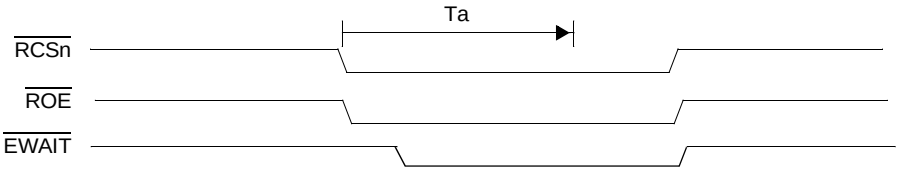



**FIGURE 8. Static Memory Read EWAIT Timing**

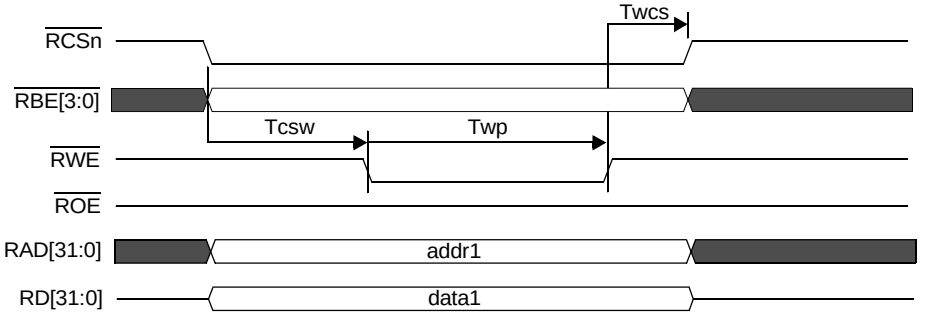



**FIGURE 9. Static Memory Write Timing**


**FIGURE 10.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 Au1100 provides a PCMCIA host adapter when the device type is programmed for PCMCIA. The static controller interface provides all the basic bus signals necessary to control a PCMCIA interface. There are a few auxiliary signals for card detect, voltage sense, etc. that can be implemented with the Au1100 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 only supports 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. It should be noted that when a chip select is programmed as a PCMCIA device that the associated $\overline{\text{RCSn}}$ is not used.

The PCMCIA interface occupies 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 14 shows the mapping.

### TABLE 14. 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 MByte. Any access beyond the 64 MByte space will alias back into the defined region.

Table 15 enumerates the signals to support the PCMCIA interface.

### TABLE 15. PCMCIA Interface Signals

| Pin Name | Input/ Output | Description |
|---|---|---|
| RAD[31:0] | O | Address bus |
| RD[31:0] | IO | Data bus |
| $\overline{\text{PREG}}$ | O | When this signal is asserted card access will be limited to attribute memory when a memory access occurs and to I/O ports when an I/O access occurs.<br>*Muxed with GPIO204 which controls the pin out of hardware reset, runtime reset and sleep.* |

**TABLE 15. PCMCIA Interface Signals (Continued)**

| Pin Name | Input/ Output | Description |
|---|---|---|
| $\overline{\text{PCE[1:0]}}$ | O | Card Enables (Active Low)<br>*Muxed with GP[206:205] which controls the pins out of hardware reset, runtime reset and sleep.* |
| $\overline{\text{POE}}$ | O | Memory Output enable |
| $\overline{\text{PWE}}$ | O | Memory Write Enable<br>*Muxed with GP207 which controls the pin out of hardware reset, runtime reset and sleep.* |
| $\overline{\text{PIOR}}$ | O | I/O Read Cycle Indication |
| $\overline{\text{PIOW}}$ | O | I/O Write Cycle Indication |
| $\overline{\text{PWAIT}}$ | I | This signal is asserted by the card to delay completion of a pending cycle. |
| $\overline{\text{PIOS16}}$ | I | 16 bit port select |
| $\overline{\text{OE}}$ | O | Output Enable - This output enable is intended to be used as a data transceiver control as it will remain asserted for read or deasserted for write for the entire PCMCIA transaction |

Figure 11 and Figure 12 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. 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.

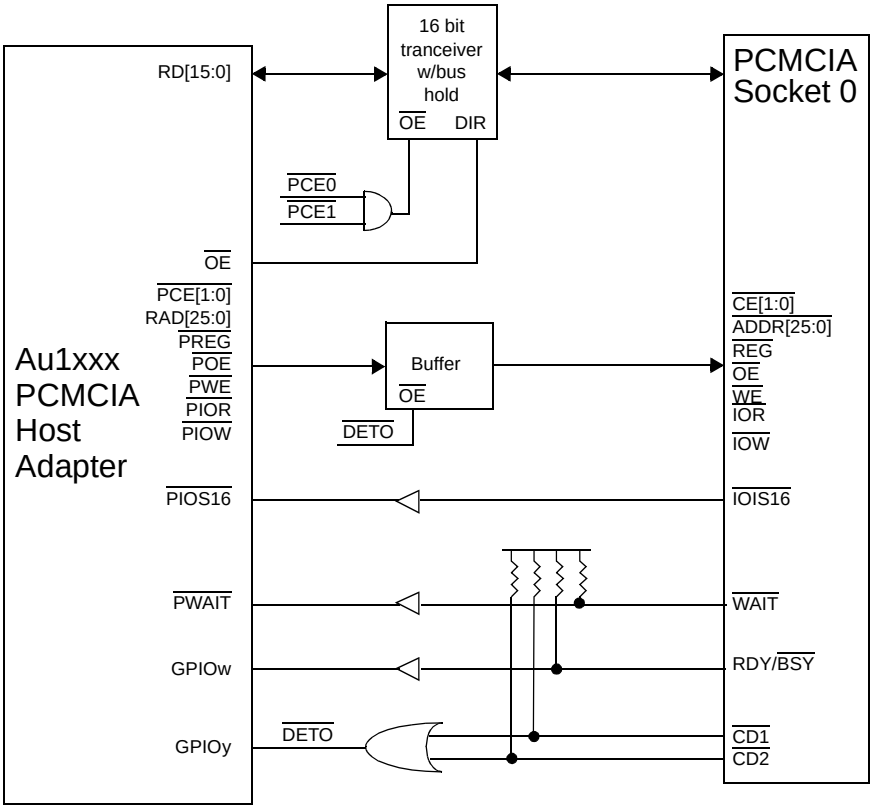



**FIGURE 11.One Card PCMCIA Interface**

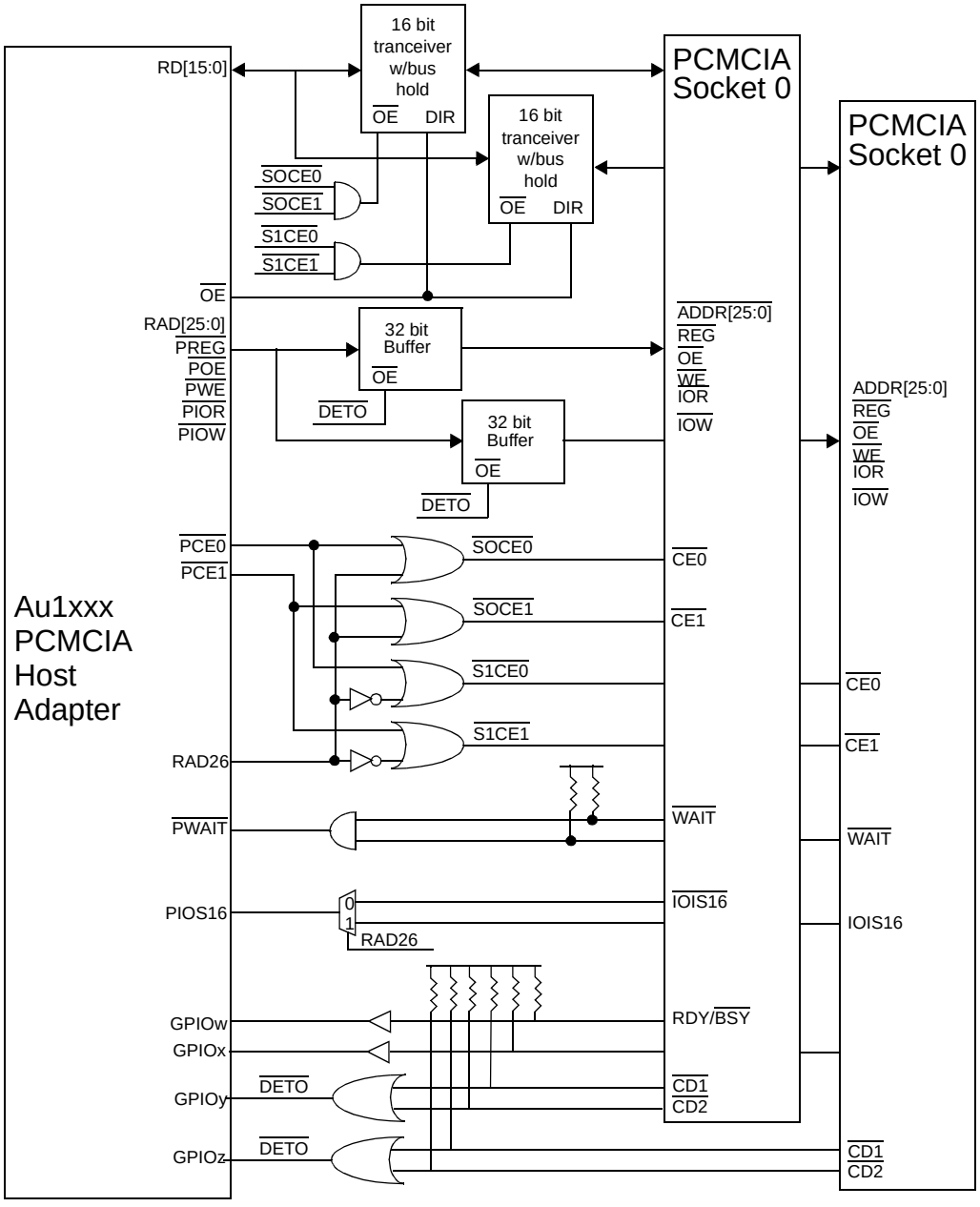



**FIGURE 12.Two Card PCMCIA Interface**

### 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 $\overline{\text{PWAIT}}$ timing diagrams are presented to show how $\overline{\text{PWAIT}}$ 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 Chapter 11, Electrical Specifications.

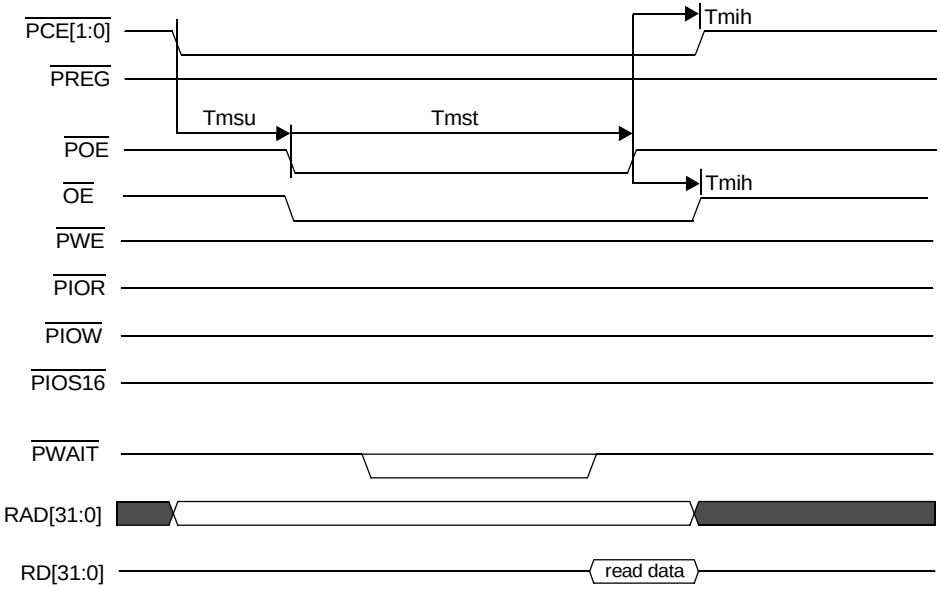



**FIGURE 13.PCMCIA Memory Read Timing**

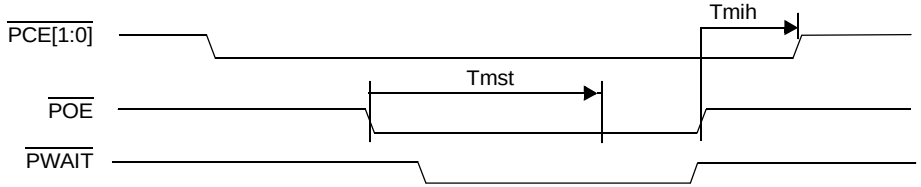



**FIGURE 14.PCMCIA Memory Read $\overline{\text{PWAIT}}$ Timing**

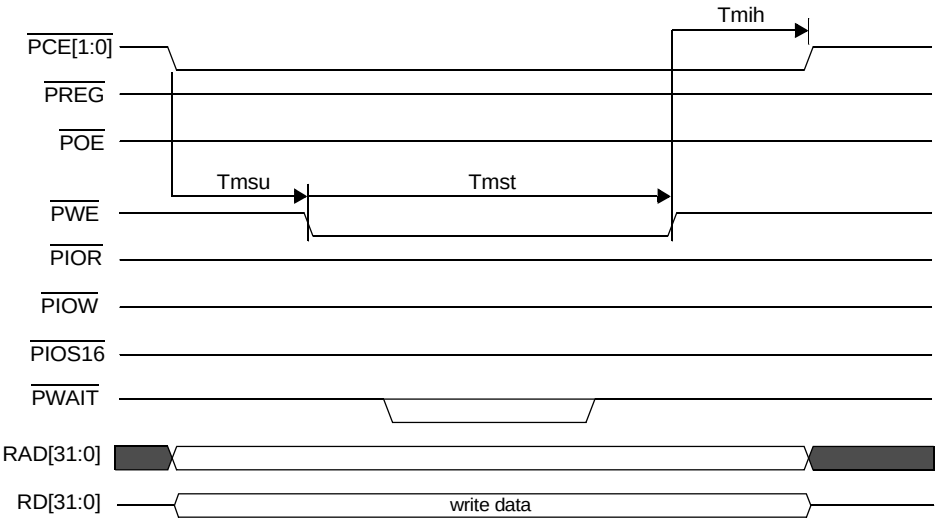



**FIGURE 15.PCMCIA Memory Write Timing**

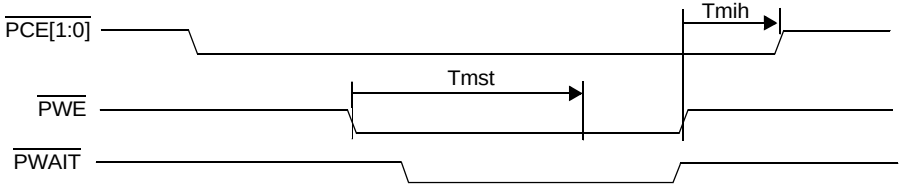



**FIGURE 16.PCMCIA Memory Write $\overline{\text{PWAIT}}$ Timing**

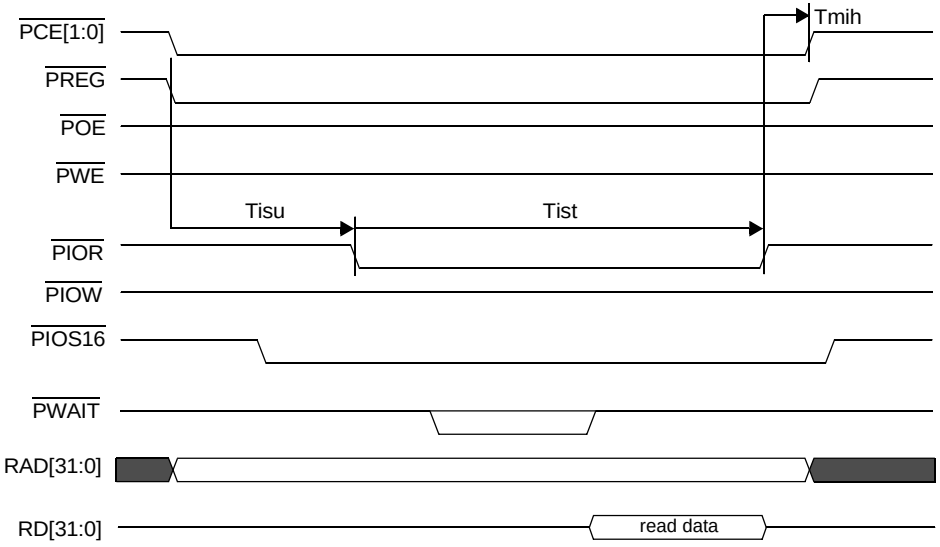



**FIGURE 17.PCMCIA I/O Read Timing**

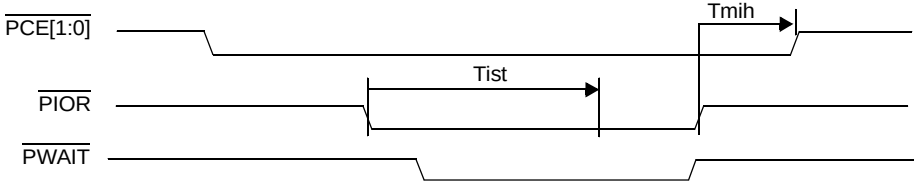



**FIGURE 18.PCMCIA I/O Read $\overline{\text{PWAIT}}$ Timing**

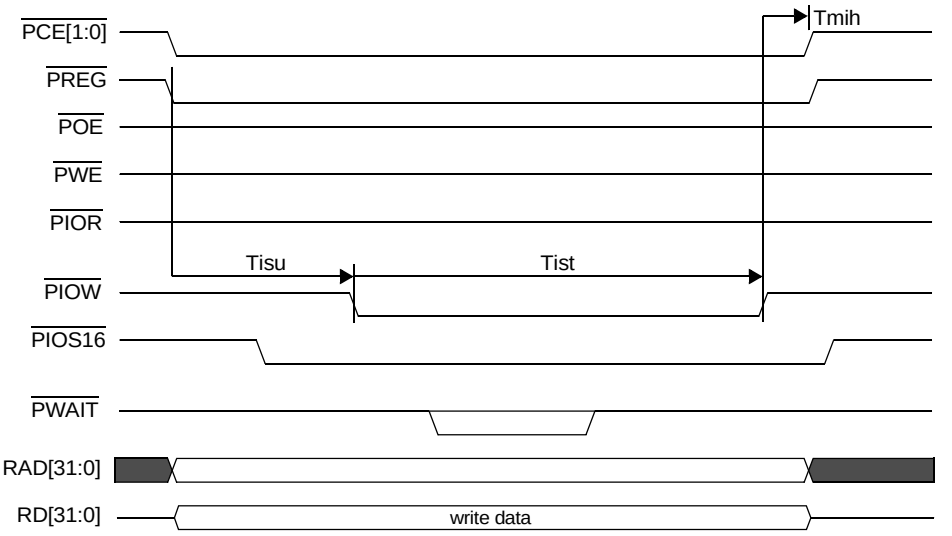



**FIGURE 19.PCMCIA I/O Write Timing**

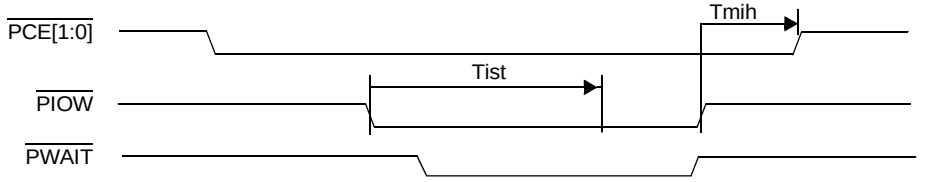



**FIGURE 20.PCMCIA I/O Write $\overline{\text{PWAIT}}$ Timing**

## 3.2.4 LCD Controller Device Type

The Au1100 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.

There is a dedicated clock, LCLK, intended for use with the LCD interface. The LCLK rate is determined by the D5 bit in **mem_stcfg0**. The rate will be the system bus block divided by 4 (D5 = 0) or 5 (D5 = 1).

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 16 lists the control signals to support the LCD controller.

**TABLE 16. LCD Controller Interface Signals**

| Signal | | Function |
|---|---|---|
| RAD[31:0] | O | Address bus |
| RD[31:0] | IO | Data bus |
| $\overline{\text{RCS[3:0]}}$ | O | Chip Selects |
| LCLK | O | Interface Clock |
| $\overline{\text{LWAIT}}$ | I | Extend Cycle |
| $\overline{\text{LRD[1:0]}}$ | O | Read Indicators<br>*Muxed with GP[201:200] which controls the pins out of hardware reset, runtime reset and sleep.* |
| $\overline{\text{LWR[1:0]}}$ | O | Write Indicators<br>*Muxed with GP[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 $\overline{\text{LWAIT}}$ timing diagrams are presented to show how $\overline{\text{LWAIT}}$ will hold the cycle past Ta for memory reads and Twp for memory writes.

LWAIT timing requirements as well as setup and hold times are presented in Chapter 11, Electrical Specifications.

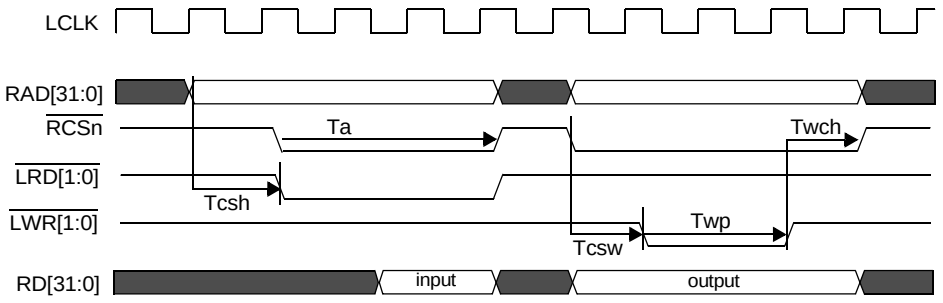



**FIGURE 21.LCD Controller Timing**

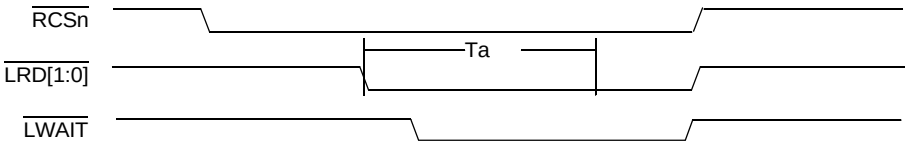



**FIGURE 22.LCD Read $\overline{\text{LWAIT}}$ Timing**

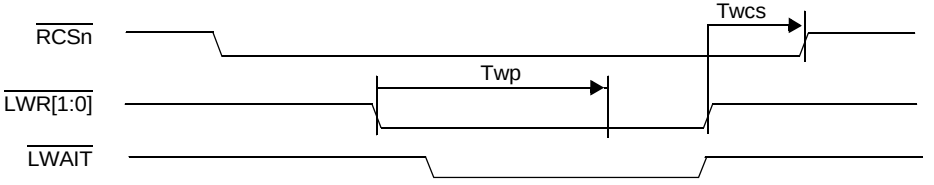



**FIGURE 23.LCD Write $\overline{\text{LWAIT}}$ Timing**

# 4 DMA Controller

The Au1100 contains an eight-channel DMA controller. Each channel is capable of transferring data between memory and any of 20 peripherals or between memory and a memory mapped FIFO through the Static Controller using a GPIO as a request.

GPIO4 and GPIO5 can be programmed to act as external DMA request lines. See Section 4.2 for details.

## 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 17.

**TABLE 17. DMA Channel Base Addresses**

| DMA Channel | Base Address | KSEG0 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) |

Each register block contains the registers shown in Table 18.

### TABLE 18. DMA Channel Configuration Registers

| Offset | 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 |

Table 19 shows the different peripherals that are capable of DMA. The Device ID, Transfer Size and Transfer 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 if the **dma_mode** register.

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

### TABLE 19. Peripheral Addresses and Selectors

| Peripheral Device | Device ID Select | Device ID | Transfer Size | Transfer Width | FIFO Physical Address |
|---|---|---|---|---|---|
| UART 0 transmit | 0 | 0 | program-mable | 8 | 0x0 1110 0004 |
| UART 0 receive | 0 | 1 | program-mable | 8 | 0x0 1110 0000 |
| GP04 | 0 | 2 | program-mable | program-mable | programmable |
| GP05 | 0 | 3 | program-mable | program-mable | programmable |
| AC97 Transmit | 0 | 4 | 4 | 16 | 0x0 1000 0008 |
| AC97 Receive | 0 | 5 | 4 | 16 | 0x0 1000 0008 |
| UART3 transmit | 0 | 6 | program-mable | 8 | 0x0 1140 0004 |

**TABLE 19. Peripheral Addresses and Selectors (Continued)**

| Peripheral Device | Device ID Select | Device ID | Transfer Size | Transfer Width | FIFO Physical Address |
|---|---|---|---|---|---|
| UART3 receive | 0 | 7 | program-mable | 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 2 transmit | 0 | 10 | 4 | 8 | 0x0 1020 0008 |
| USB Device Endpoint 3 transmit | 0 | 11 | 4 | 8 | 0x0 1020 000c |
| USB Device Endpoint 4 receive | 0 | 12 | 4 | 8 | 0x0 1020 0010 |
| USB Device Endpoint 5 receive | 0 | 13 | 4 | 8 | 0x0 1020 0014 |
| $I^2S$ transmit | 0 | 14 | 4 | program-mable | 0x0 1100 0000 |
| $I^2S$ receive | 0 | 15 | 4 | program-mable | 0x0 1100 0000 |
| SD 0 transmit | 1 | 0 | program-mable | 8 | 0x0 1060 0000 |
| SD 0 receive | 1 | 1 | program-mable | 8 | 0x0 1060 0004 |
| SD 1 transmit | 1 | 2 | program-mable | 8 | 0x0 1068 0000 |
| SD 1 receive | 1 | 3 | program-mable | 8 | 0x0 1068 0004 |
| Reserved | 1 | 4-15 | n/a | n/a | n/a |

### 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.

The **mode_set** register sets bits in the channel control register when the corresponding bit is written as a one.

The **mode_clear** register clears bits in the channel control register when the corresponding bit is written as a one. Bits written as zero to either register do not affect the corresponding bit in the channel mode register.

The Au1100 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:

```
mode_set = new_value & field_mask;
mode_clear = ~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 19.

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" for more information.

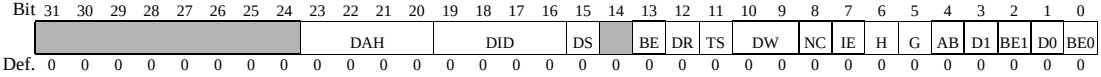
For the $I^2S$ fifos the transfer width is programmable. It is the programmers responsibility to insure that the Transfer Width field matches the word size in the $I^2S$ configuration register and that memory is packed accordingly. See Section 6.6, "I2S Controller" for more information.

For external DMA using GPIO signals as requests, it is the system designers responsibility to insure that the Transfer Size and Device Width match the external FIFO and that memory is packed accordingly.

**dma_moderead - Read DMA Mode Register** Offset = 0x0000

**dma_modeset - Set DMA Mode Register** Offset = 0x0000

**dma_modeclr - Clear DMA Mode Register** Offset = 0x0004




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:24 | RES | These bits are reserved and should be written a 0. | R | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 23:20 | DAH | Device Address High<br>Provides the most significant 4 bits of physical device address. | R/W | 0 |
| 19:16 | DID | Device ID<br>Identifies the peripheral device to act as source or destination. This ID is used in combination with the Device ID Select bit (see Table 19). | R/W | 0 |
| 15 | DS | Device ID Select<br>This bit selects between two banks of Device IDs. It is used in combination with the Device ID bits (see Table 19). | R/W | 0 |
| 14 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 13 | BE | Big Endian<br>0 - Little Endian byte order<br>1 - Big Endian byte order | R/W | 0 |
| 12 | DR | Device Read<br>0 - Data is transferred from memory to device.<br>1 - Data is transferred from device to memory. | R/W | 0 |
| 11 | TS | Transfer Size<br>0 - Transfer size will be 4 datums. The datum width for the transfer is set in bit DW.<br>1 - Transfer size will be 8 datums. The datum width for the transfer is set in bit DW. | R/W | 0 |
| 10:9 | DW | Device Width<br>0 - Device FIFO is 8 bits wide<br>1 - Device FIFO is 16 bits wide<br>2 - Device FIFO is 32 bits wide<br>3 - Reserved<br>Typically the device width should match the FIFO width. | RW | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 8 | NC | Not Coherent<br>0 - Memory reads and writes are marked coherent on the system bus.<br>1 - Memory reads and writes are marked non coherent on the system bus.<br>For more information on coherency see Section 2.8.2, "SBUS Coherency Model" for more information on coherency. | R/W | 0 |
| 7 | IE | Interrupt Enable<br>0 - No interrupts will be generated.<br>1 - Interrupts are generated when either D1 or D0 is set. | R/W | 0 |
| 6 | H | Channel Halted<br>0 - Channel is active.<br>1 - Channel is halted.<br>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<br>Setting the channel go bit enables the channel. When this bit is cleared the DMA controller will not arbitrate for this channel regardless of the state of the buffer enable bits. When the go bit is cleared by the processor the channel configuration should not be modified until the DMA controller sets the halt bit to indicate that the channel is inactive. | | |
| 4 | AB | Active Buffer<br>0 - Buffer 0 is currently in use by the DMA.<br>1 - Buffer 1 is currently in use by the DMA.<br>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.<br>It should be noted that the DMA will ping pong between the two buffers. In other words, it is not possible to only use one buffer, DMA transactions must be alternated between each buffer. | R | 0 |

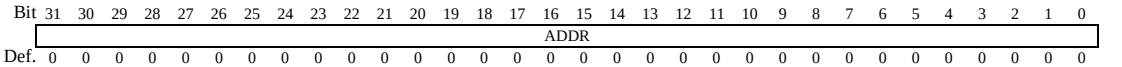
| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 3 | D1 | Done 1<br>The D1 bit is set by the DMA controller to indicate that a transfer to or from buffer 1 is complete. This bit must be cleared by the processor. | R/W | 0 |
| 2 | BE1 | The BE1 bit enables buffer 1. This bit is 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<br>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 |

### 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 19. 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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | ADDR | Peripheral FIFO address | R/W | 0 |

### DMA Buffer Starting Address Registers

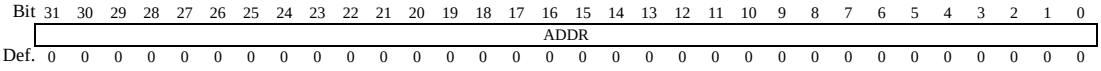
Each DMA channel has two buffers, labeled buffer0 and buffer1. The starting address of each buffer should be written to the **dma_buf0addr** and **dma_buf0addr** registers respectively. The starting address must be 32-bit word aligned.

The 4 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.

It should be noted that the DMA will ping pong between the two buffers. In other words, it is not possible to only use one buffer, DMA transactions must be alternated 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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | ADDR | Lower 32 bits of the physical starting address of the DMA memory buffer. | R/W | 0 |

### DMA Channel Buffer Size Registers

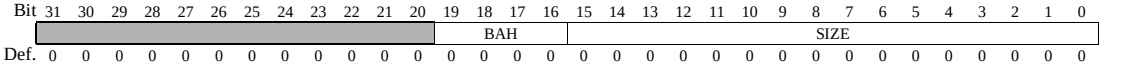
The size of each DMA buffer is given by the **dma_buff0size** and **dma_buff1size** registers. The buffer size registers also contributes the most significant four bits of the buffer physical address.

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

It should be noted that the DMA will ping pong between the two buffers. In other words, it is not possible to only use one buffer, DMA transactions must be alternated between each buffer. The AB bit in the **dma_mode** register can be used to determine the active buffer.

**dma_buff0size - Buffer 0 Size** Offset = 0x0010

**dma_buff1size - Buffer 1 Size** Offset = 0x0018




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:20 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 19:16 | BAH | Buffer Address High<br>This field provides the 4 most significant bits of the buffer address. | R/W | 0 |
| 15:0 | SIZE | Buffer Size and Count Remaining<br>This field indicates the number of datums remaining in the current transfer. | R/W | 0 |

## 4.2 Using GPIO Lines as External DMA Requests

To use GPI04 or GPI05 as an external DMA request the following steps must be done:

1. Write the **sys_gpinputen** to enable the GPIO to be used as an input. See Section 7.3, "Primary General Purpose I/O" for more information.
2. Tristate the GPIO to make it an input through the **sys_triout** register. See Section 7.3, "Primary General Purpose I/O" for more information.
3. Set the **dma_peraddr** register to point to the external device data port. The Static Controller must be configured correctly to recognize this address.
4. Program the mode register to match the direction of transfer and peripheral attributes.

The external request must be driven high to request a DMA transfer. It must be held high until the DMA transaction is started. Once the DMA transaction has started it will continue

until finished regardless of the DMA request. A DMA transaction refers to a DMA transfer of one transfer size as defined by the TS bit in the **dma_mode** register.

The external DMA request should be tied to the external FIFO threshold indicator. In this way it will assert when the FIFO threshold is reached and remain asserted until the FIFO fills or empties past the threshold (after the DMA transaction starts). It should then deassert after 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.

# 5 Interrupt Controller

There are two interrupt controllers in the Au1100. Each interrupt controller supports 32 interrupt sources. Interrupts can generate a signal to bring the Au1100 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" for a complete Au1100 Interrupt architecture discussion. Table 20 shows the Interrupt controller connections to the CPU.

**TABLE 20. 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 21 shows the mapping of interrupt sources for Interrupt Controller 0 and 1.

Care should be taken to follow the correct interrupt type or there is potential to miss an interrupt. 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 21. 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 |

**TABLE 21. Interrupt Sources (Continued)**

| Controller | Interrupt Number | Source | Type |
|---|---|---|---|
| 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 | GPIO 215:208 | System Dep. |
| 0 | 30 | I2S | High Level |
| 0 | 31 | AC97 Command Done | Rising Edge |
| 1 | n = 0..15 | GPIO[n] | System Dep. |

Figure 24 shows the Interrupt Controller Logic Diagram. Where applicable, the names in the diagram correspond to bit *n* in the relative control register.

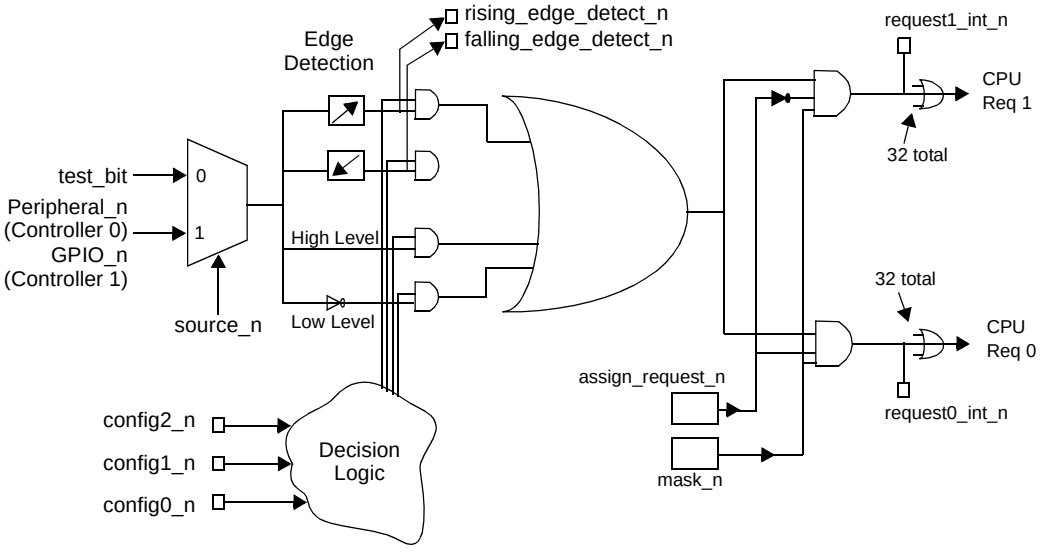



**FIGURE 24. Interrupt Controller Logic**

## 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 22 shows each interrupt controller's base address.

**TABLE 22. 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 23 shows the interrupt controller registers and their associated offsets. Certain offsets are shared but address different 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 should be associated with interrupt *n* of the corresponding controller.

### TABLE 23. Interrupt Controller Registers

| Offset | Register Name | Type | Register Description | Default |
|---|---|---|---|---|
| 0x0040 | ic_cfg0rd | R | Configuration 0 register<br>Combined, Config2[n], Config1[n], and Config0[n] specifies the interrupt n characteristics as shown in Table 24. | UNPRED |
| 0x0040 | ic_cfg0set | W | | |
| 0x0044 | ic_cfg0clr | W | | |
| 0x0048 | ic_cfg1rd | R | Configuration 1 register<br>Combined, Config2[n], Config1[n], and Config0[n] specifies the interrupt n characteristics as shown in Table 24. | UNPRED |
| 0x0048 | ic_cfg1set | W | | |
| 0x004C | ic_cfg1clr | W | | |
| 0x0050 | ic_cfg2rd | R | Configuration 2 register<br>Combined, Config2[n], Config1[n], and Config0[n] specifies the interrupt n characteristics as shown in Table 24. | UNPRED |
| 0x0050 | ic_cfg2set | W | | |
| 0x0054 | ic_cfg2clr | W | | |
| 0x0054 | ic_req0int | R | Shows active interrupts on request 0.<br>This register is used by host to determine source of interrupt. | 0x0000 0000 |
| 0x0058 | ic_srcrd | R | Controls the source of the interrupt between a test bit and the designated source.<br>0 - test bit is used as interrupt source.<br>1 - peripheral signal (controller 0) or GPIO (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.<br>This register is used by host to determine source of interrupt. | 0x0000 0000 |
| 0x0060 | ic_assignrd | R | Assigns the interrupt to one of the CPU requests (the assignment is inverse to the value programmed).<br>0 - interrupt assigned to request 1<br>1 - interrupt assigned to request 0 | UNPRED |
| 0x0060 | ic_assignset | W | | |
| 0x0064 | ic_assignclr | W | | |

**TABLE 23. Interrupt Controller Registers (Continued)**

| Offset | Register Name | Type | Register Description | Default |
|---|---|---|---|---|
| 0x0068 | ic_wakerd | R | Controls whether the interrupt can cause a wakeup from IDLE0 or IDLE1.<br>0 - no wakeup from idle<br>1 - interrupt will cause wakeup from idle.<br>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 | Enables/Disables the interrupt.<br>0 - interrupt is disabled.<br>1 - interrupt is enabled. | 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 |

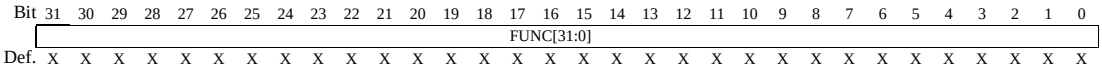
**Interrupt Controller Registers**

Each register is 32 bits wide with bit *n* in each register affecting interrupt *n* in the corresponding controller.

***rd**

***set**

***clr**



| Bit(s) | Name | Description | Read/Write | Default |
|---|---|---|---|---|
| 31:0 | FUNC[n] | The function of each register is given in Table 23. FUNC[n] controls the functionality of interrupt *n* in the corresponding controller. | *rd - read only<br>*set - write only<br>*clr - write only<br>See the following explanation. | See Table 23 |

Certain registers in the list have the same offset but offer different functionality. This is by design. Care should be taken when programming the registers as 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[bit n], ic_cfg1[bit n], ic_cfg0[bit n] uniquely control interrupt *n*'s functionality as shown in Table 24. In general ic_cfg2[n], ic_cfg1[n] and ic_cfg0[n] can be described as follows (Table 24 should be referred for exact functionality):

ic_cfg2[n] - Edge/Level select. When ic_cfg2[n] is low, ic_cfg1[n] and ic_cfg0[n] will enable edge triggered interrupts. When ic_cfg2[n] is high, ic_cfg1[n] and ic_cfg0[n] will enable level triggered interrupts. If ic_cfg2[n], ic_cfg1[n] and ic_cfg0[n] are all high then both level and edge interrupts will be enabled

ic_cfg1[n] - Falling Edge/Low Level enable. Depending on how ic_cfg2[n] is set, ic_cfg1[n] will enable falling edge or low level interrupts when set high.

ic_cfg0[n] - Rising Edge/High Level enable. Depending on how ic_cfg2[n] is set, ic_cfg0[n] will enable rising edge or high level interrupts when set high.

**TABLE 24. 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" 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 the interrupt controller can be broken into the following steps (the set, clr, rd portion of the register name has been ommitted):

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 (or a 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 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 determining 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 Interrupt

# 6 Peripheral Devices

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

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

## 6.1 AC97 Controller

The Au1100 contains an AC'97 Controller which incorporates an AC-link capable of bridging to an AC'97 compliant CODEC.

All data being sent and received through the AC97 controller must be 48kHz.

### 6.1.1 AC97 Registers

The AC'97 controller is controlled by a register block whose physical base address is shown in Table 25.

**TABLE 25. AC97 Base Address**

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

The register block consists of 5 registers as shown in Table 26.

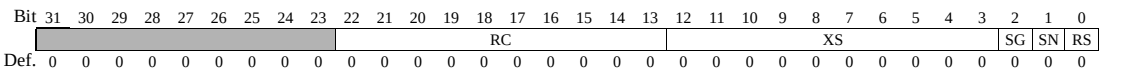
**TABLE 26. AC97 Registers**

| Offset | 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 |

#### AC-Link Configuration Register

The configuration register contains bits necessary to configure and reset the AC-link and CODEC.

**ac97_config - AC-link Configuration** Offset = 0x0000



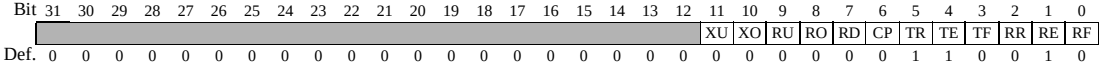
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:23 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 22:13 | RC | Receive Slots<br>The bits set in RC will control what data from valid slots are put into the input buffer.<br>The corresponding valid bits in the AC'97 tag (slot 0 of SDATA_IN) must be marked valid for the incoming PCM data to be put in the input buffer.<br>Slot 3 is mapped to bit 13, slot 4 to 14 and so on.<br>It is the programmers responsibility to ensure that the CODEC is configured such that there will be valid data in the slots corresponding to what receive slots are enabled. | R/W | 0 |
| 12:3 | XS | Transmit Slots<br>The bits making up XMIT_SLOTS map to the valid bits in the AC'97 tag (slot 0 on SDATA_OUT) and indicate which outgoing slots have valid PCM data. Bit 3 maps to slot 3, bit 4 to slot 4 and so on. Setting the corresponding bit indicates to the CODEC that valid data will be in the respective slot. The number of valid bits will designate how many words will be pulled out of the FIFO per audio frame | R/W | 0 |
| 2 | SG | Sync Gate<br>Setting this bit to 1 will gate the clock from being driven on SYNC. This allows the SN bit to control the value on SYNC. In combination with SN, the SG bit can be used to initiate a warm reset. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | SN | Sync<br>This bit controls the value of the SYNC signal when SG is set to 1. In combination with SG, the SN bit can be used to initiate a warm reset. | R/W | 0 |
| 0 | RS | AC-link Reset<br>When the RST bit is set high this will drive the ACRST signal of the AC-link low to initiate a cold AC'97 reset. After satisfying the ACRST low time for the CODEC this bit should be set low to deassert ACRST. | R/W | 0 |

### AC97 Controller Status

The AC97 Controller Status register contains status bits for the transmit and receive FIFOs, command status and the CODEC.

**ac97_status - Controller Status** Offset = 0x0004




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:12 | RES | These bits are reserved. | R | UNPRED |
| 11 | XU | Transmit Underflow<br>When set to 1, this bit indicates that the transmit FIFO has experienced an underflow. This sticky bit will be cleared when read. | R | 0 |
| 10 | XO | Transmit Overflow<br>When set to 1, this bit indicates that the transmit FIFO has experienced an overflow. This sticky bit will be cleared when read. | R | 0 |
| 9 | RU | Receive Underflow<br>When set to 1, this bit indicates that the receive FIFO has experienced an underflow. This sticky bit will be cleared when read. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 8 | RO | Receive Overflow<br>When set to 1, this bit indicates that the receive FIFO has experienced an overflow. This sticky bit will be cleared when read. | R | 0 |
| 7 | RD | Ready<br>This bit is mapped from the CODEC_READY bit in the SDATA_IN tag word. It indicates that the CODEC is properly booted and ready for normal operation. | R | 0 |
| 6 | CP | Command Pending<br>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.)<br>The command register should not be written until the CP bit is clear.<br>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 | RES | Reserved | R | UNPRED |
| 4 | TE | Transmit Empty<br>When set this bit indicates that the transmit FIFO is empty. | R | 0 |
| 3 | TF | Transmit Full<br>When set this bit indicates the transmit FIFO is full. | R | 0 |
| 2 | RES | Reserved | R | UNPRED |

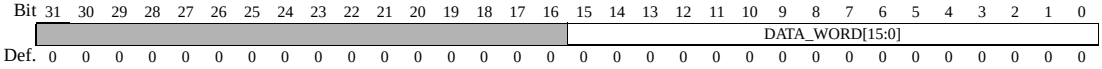
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | RE | Receive Empty<br>When set this bit indicates that the receive FIFO is empty. | R | 0 |
| 0 | RF | Receive Full<br>When set this bit indicates that the receive FIFO is full. | R | 0 |

### TX/RX Data

The TX/RX 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 in the FIFO for a read or write transaction. This will be taken care of automatically by using DMA.

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




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written 0. | R | 0 |
| 15:0 | DATA_ WORD | Data Word<br>This is where data will be written to or read from the FIFO. Each data word is 16 bits. | R/W | 0 |

### CODEC Command

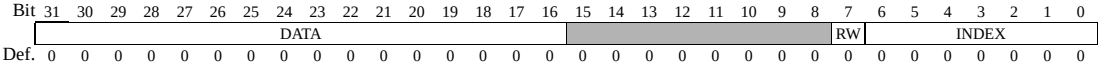
The CODEC Command and Command Response registers share the same physical address.

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 command pending returns to 0.

The CODEC Command register should only be written if the Command Pending bit in the status register is 0.

**ac97_cmmd - CODEC Command** Offset = 0x000C



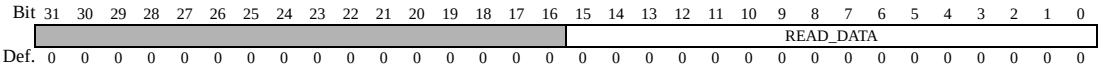

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | DATA | DATA<br>These bits will be the actual 16 bit word written to the register indicated by INDEX if RW is a 0. If RW is a 1 indicating a read, these bits should be written 0. | W | 0 |
| 15:8 | RES | These bits are reserved and should be written 0. | W | 0 |
| 7 | R/W | Read/$\overline{\text{Write}}$ Bit (1=read, 0=write)<br>This bit maps to the Read/$\overline{\text{Write}}$ 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<br>These bits will address the specific register to be read or written to inside the CODEC. | W | 0 |

**CODEC Command Response**

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

When a read command was sent through the CODEC Command register, the response can be read from the CODEC Response register. The response is only valid if the Command Pending bit in the status register is 0.

**ac97_cmmdresp - CODEC Command Response** Offset = 0x000C



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written 0. | R | 0 |
| 15:0 | READ_DATA | READ_DATA<br>These bits will be the response to the last read command sent to the CODEC. The value read is only valid if Command Pending = 0. | R | 0 |

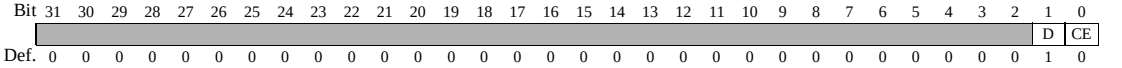
### AC97 Enable

The AC97 Enable register is used to enable and reset the entire AC97 Controller block.The suggested power on reset would be to enable clocks with the block disabled. Then clear D for run-time operation.

The correct routine for bringing the AC97 controller out of reset is as follows:

1. Set the CE bit to enable clocks.
2. Clear the D bit to enable the peripheral.

**ac97_enable - AC97 Block Control** Offset = 0x0010




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written 0. | W | 0 |
| 1 | D | AC97 Controller Disable<br>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<br>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 27.

**TABLE 27. AC-Link Signals**

| Pin Name | Input/ Output | Definition |
|---|---|---|
| ACSYNC | O | Fixed rate sample sync<br>*Muxed with S1DOUT* |
| ACBCLK | I | Serial data clock. |
| ACDO | O | TDM output stream<br>*Muxed with S1CLK* |
| ACDI | I | TDM input stream |
| $\overline{\text{ACRST}}$ | O | CODEC reset<br>*Muxed with S1DEN* |

For changing pin functionality please refer to the **sys_pinfunc** register in Section 7.3, "Primary General Purpose I/O".

### 6.1.3 Programming Considerations

To use the AC97 controller the AC97 bit (bit 0) in the **pin_function** register (in GPIO Controller) 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 will be 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 **peripheral_addr** register in the DMA configuration registers will be set to point to the AC97 **ac97_data** register. The DMA mode register will need to be set up with the correct Device ID (DID). The Device Read bit (DR) will depend on the 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 (TS8) should be set since the fifo threshold indicators correspond to 8 word 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 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 interupt.

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, 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 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 28.

### TABLE 28. 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 will not work). The interrupt controller should be programmed to reflect this by setting the USB Host interrupt to low level. See Chapter 5, Interrupt Controller for details.

#### 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.

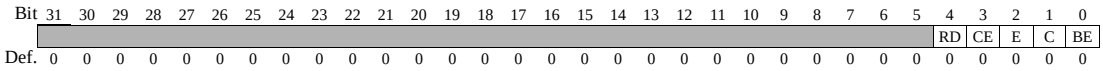
The correct routine for bringing the USB Host Controller out of reset is as follows:

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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:5 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 4 | RD | Reset Done<br>Wait for this bit to be set before issuing any commands to the OpenHCI controller. | R | 0 |
| 3 | CE | Clock Enable<br>When this bit is set, clocks are enabled to the USB Host controller. | R/W | 0 |
| 2 | E | Enable<br>This bit enables the USB Host controller. When this bit is clear the controller is held in reset. | R/W | 0 |
| 1 | C | Coherent<br>If this bit is set memory accesses by the controller will be marked coherent on the system bus. When this bit is clear memory accesses by the USB Host controller are non coherent.<br>For more information on coherency see Section 2.8.2, "SBUS Coherency Model" for more information on coherency. | R/W | 0 |
| 0 | BE | Big Endian<br>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.<br>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 |

### 6.2.1 Hardware considerations

Table 29 shows the pins associated with the two USB host root hub ports. The USB root hub port pins have USB 1.1 compliant drivers. No external driver circuitry is required.

### TABLE 29. USB Pins

| Pin Name | Input/ Output | Description |
|---|---|---|
| USB (Host) | | |
| USBH1P | IO | Positive signal of differential USB host port 1 driver.<br>Requires 15k pulldown to be USB 1.1 compliant. |
| USBH1M | IO | Negative signal of differential USB host port 1 driver.<br>Requires 15k pulldown to be USB 1.1 compliant. |
| USBH0P | IO | Positive signal of differential USB host port 0 driver<br>Requires 15k pulldown to be USB 1.1 compliant.<br>*Muxed with USBDP which controls the pin out of reset.* |
| USBH0M | IO | Negative signal of differential USB host port 0 driver<br>Requires 15k pulldown to be USB 1.1 compliant.<br>*Muxed with USBDM which controls the pin out of reset.* |

For changing pin functionality please refer to the **sys_pinfunc** register in Section 7.3, "Primary General Purpose I/O".

## 6.3 USB Device Controller

The Au1500 USB Device controller supports endpoints 0, 2, 3, 4, and 5. Endpoint 0 is always configured as a bidirectional control endpoint. Endpoints 2 and 3 are always IN endpoints and endpoints 4 and 5 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 30.

### TABLE 30. USB Device Base Address

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

Table 31 shows the offsets of each register from the register base.

### TABLE 31. USB Device Register Block

| Name | Offset | Function |
|---|---|---|
| usbd_ep0rd | 0x0000 | Read from endpoint 0 |
| usbd_ep0wr | 0x0004 | Write to endpoint 0 |
| usbd_ep2wr | 0x0008 | Write to endpoint 2 |
| usbd_ep3wr | 0x000c | Write to endpoint 3 |
| usbd_ep4rd | 0x0010 | Read from endpoint 4 |
| usbd_ep5rd | 0x0014 | Read from endpoint 5 |
| usbd_inten | 0x0018 | Interrupt Enable Register |
| usbd_intstat | 0x001c | Interrupt Status Register |
| usbd_config | 0x0020 | Write Configuration Data |
| usbd_ep0cs | 0x0024 | Endpoint 0 control and status |
| usbd_ep2cs | 0x0028 | Endpoint 2 control and status |
| usbd_ep3cs | 0x002c | Endpoint 3 control and status |
| usbd_ep4cs | 0x0030 | Endpoint 4 control and status |
| usbd_ep5cs | 0x0034 | Endpoint 5 control and status |

**TABLE 31. USB Device Register Block**

| Name | Offset | Function |
|---|---|---|
| usbd_framenum | 0x0038 | Current frame number |
| usbd_ep0rdstat | 0x0040 | EP0 Read FIFO Status |
| usbd_ep0wrstat | 0x0044 | EP0 Write FIFO Status |
| usbd_ep2wrstat | 0x0048 | EP2 Write FIFO Status |
| usbd_ep3wrstat | 0x004c | EP3 Write FIFO Status |
| usbd_ep4rdstat | 0x0050 | EP4 Read FIFO Status |
| usbd_ep5rdstat | 0x0054 | EP5 Read FIFO 5 Status |
| usbd_enable | 0x0058 | USB Device Controller Enable |

### 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 will return unpredictable results if read.

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

**usbd_ep0rd** Offset = 0x0000

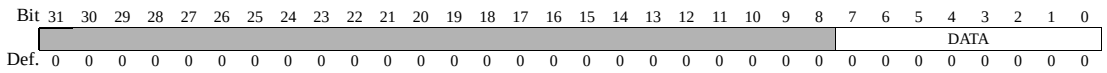
**usbd_ep0wr** Offset = 0x0004

**usbd_ep2wr** Offset = 0x0008

**usbd_ep3wr** Offset = 0x000c

**usbd_ep4rd** Offset = 0x0010

**usbd_ep5rd** Offset = 0x0014




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 7:0 | DATA | Data<br>Byte of data to be written to, or read from the endpoint FIFO. | R/W | 0 |

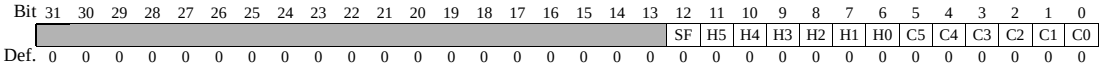
### Interrupt Registers

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 will be 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




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 12 | SF | Start of Frame<br>This interrupt issues when an SOF token is received. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 11:6 | H5:H0 | Fifo Half Full<br>These interrupts issue when the corresponding FIFO reaches the half full/half empty mark.<br>The bits correspond as follows:<br>H0 - ep0rd<br>H1 - ep0wr<br>H2 - ep2wr<br>H3 - ep3wr<br>H4 - ep4rd<br>H5 - ep5rd | R/W | 0 |
| 5:0 | C5:C0 | Complete<br>These interrupts issue when a transmission or reception completes on the corresponding FIFO. For FIFOs 0, 4, and 5 these interrupts indicate the reception of a DATA0 or DATA1 packet or a SETUP packet (FIFO 0 only). For FIFOs 1, 2, and 3 these interrupts indicate the transmission of a DATA0 or DATA1 packet.<br>The bits correspond as follows:<br>C0 - ep0rd<br>C1 - ep0wr<br>C2 - ep2wr<br>C3 - ep3wr<br>C4 - ep4rd<br>C5 - ep5rd | R/W | 0 |

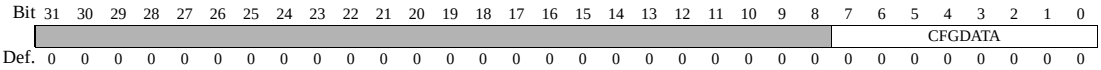
### Device Configuration Register

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

The data must be written to the usbd_config register in order beginning with byte 0. Bytes are written individually as unsigned 32-bit words. For example:

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

**usbd_config** Offset = 0x0020



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 7:0 | CFG-DATA | Configuration Data<br>Configuration data consists of a block of 25 bytes which must be written to this register after the controller is removed from reset. Each write must contain only one byte of data. The data is enumerated in Table 32. Data should be written in order, starting with byte 0. | R/W | 0 |

**TABLE 32. Configuration Data**

| Byte | Format | Description |
|---|---|---|
| 0 | 0000 0100 | fixed |
| 1 | 0000 0sss | sss = Endpoint 0 Max Packet Size bits[9:7] |
| 2 | ssss sss0 | ssss sss = Endpoint 0 Max Packet Size bits[6:0] |
| 3 | 0000 0000 | fixed |
| 4 | 0000 0001 | fixed |
| 5 | 0010 0100 | fixed |
| 6 | 00tt 1sss | tt = Endpoint 2 type: 00 = control, 01 = isochronous, 10 = bulk, 11 = interrupt<br>sss = Endpoint 2 Max Packet Size bits[9:7] |
| 7 | ssss sss0 | ssss sss = Endpoint 2 Max Packet Size bits[6:0] |
| 8 | 0000 0000 | fixed |
| 9 | 0000 0010 | fixed |
| 10 | 0011 0100 | fixed |
| 11 | 00tt 1sss | tt = Endpoint 3 type: 00 = control, 01 = isochronous, 10 = bulk, 11 = interrupt<br>sss = Endpoint 3 Max Packet Size bits[9:7] |
| 12 | ssss sss0 | ssss sss = Endpoint 3 Max Packet Size bits[6:0] |
| 13 | 0000 0000 | fixed |
| 14 | 0000 0011 | fixed |

**TABLE 32. Configuration Data**

| Byte | Format | Description |
|---|---|---|
| 15 | 0100 0100 | fixed |
| 16 | 00tt 0sss | tt = Endpoint 4 type: 00 = control, 01 = isochronous, 10 = bulk, 11 = interrupt<br>sss = Endpoint 4 Max Packet Size bits[9:7] |
| 17 | ssss sss0 | ssss sss = Endpoint 4 Max Packet Size bits[6:0] |
| 18 | 0000 0000 | fixed |
| 19 | 0000 0100 | fixed |
| 20 | 0101 0100 | fixed |
| 21 | 00tt 0sss | tt = Endpoint 5 type: 00 = control, 01 = isochronous, 10 = bulk, 11 = interrupt<br>sss = Endpoint 5 Max Packet Size bits[9:7] |
| 22 | ssss sss0 | ssss sss = Endpoint 5 Max Packet Size bits[6:0] |
| 23 | 0000 0000 | fixed |
| 24 | 0000 0101 | fixed |

### Endpoint Control Registers

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

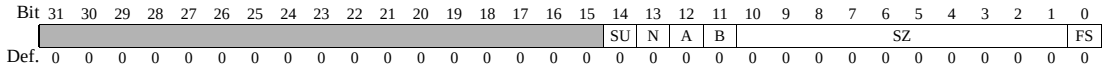
**usbd_ep0cs** Offset = 0x0024

**usbd_ep2cs** Offset = 0x0028

**usbd_ep3cs** Offset = 0x002c

**usbd_ep4cs** Offset = 0x0030

**usbd_ep5cs** Offset = 0x0034

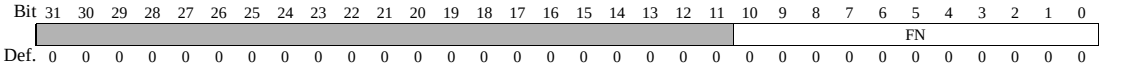


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:15 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 14 | SU | Setup Received - This bit is set when a SETUP packet is received from the host. It is only valid for EP 0. | R | 0 |
| 13 | N | NAK - This bit will be 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.<br>SETUP packets that are automatically handled in hardware are also indicated by the N bit. For these transactions the data in the receive FIFO should be discarded. | 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 packet or that a DATA packet was received correctly and an ACK returned to the Host.<br>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 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 10:1 | SZ | The TSIZE field specifies the data size of an IN transfer. The TSIZE field is only relevant on endpoints 0, 2, and 3. | R/W | 0 |
| 0 | FS | Force Stall - Setting this bit will place the endpoint in a stalled condition. Any transaction directed to the endpoint will be 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 will not clear a stall condition forced by this bit. | R/W | 0 |

### Current Frame Number

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

**usbd_framenum - Current Frame Number** Offset = 0x0038



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:11 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 10:0 | FN | Frame Number<br>This field contains the frame number from the start of frame packet. | R | 0 |

### FIFO Status Registers

Each FIFO has a status register that indicates the current state and any error conditions.

The USB FIFOs are 8 words deep.

**usbd_ep0rdstat** Offset = 0x0040

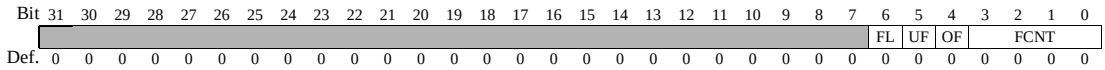
**usbd_ep0wrstat** Offset = 0x0044

**usbd_ep2wrstat** Offset = 0x0048

**usbd_ep3wrstat** Offset = 0x004c

**usbd_ep4rdstat** Offset = 0x0050

**usbd_ep5rdstat** Offset = 0x0054




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:7 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 6 | FL | Flush FIFO - Writing a 1 to this bit will clear the corresponding FIFO and discard any data contained in it. | W | 0 |
| 5 | UF | Underflow Flag - This bit will be set if a read is done to an empty FIFO. This bit can be cleared by writing a 1 to it. | R/W | 0 |
| 4 | OF | Overflow Flag - This bit will be set if a byte is written to a full FIFO. This bit can be cleared by writing a 1 to it. | R/W | 0 |
| 3:0 | FCNT | FIFO Count - This field reflects the current number of bytes in the corresponding FIFO. | R | |

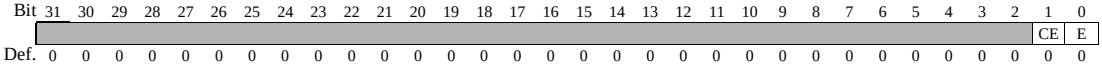
**Device Controller Enable Register**

The Device Controller Enable register controls the clocks and reset to the device controller. The programmer should first enable clocks and then enable the device controller to bring out of reset.

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

**usbd_enable** Offset = 0x0058




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 1 | CE | Clock Enable - Clearing this bit disables all clocks to the USB Device core. Setting this bit allows normal operation. | R/W | 0 |
| 0 | E | Enable - When this bit is cleared the Device Controller will be held in reset. Setting this bit enables normal operation. | R/W | 0 |

## 6.3.1 Programming Considerations

### 6.3.1.1 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 1us.
3. Write a 0x0003 to the **usbd_enable** register to remove the controller from reset.
4. 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.

Accessing the endpoint control registers (**usbd_ep*n*cs)**, frame number register (**usbd_framenum**) or config data register (**usbd_config**) while the USB Device is in sus-

pend mode will result in a system bus deadlock. This will inhibit any further operation of the CPU, including EJTAG debugger operation.

#### 6.3.1.2 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 will not be passed to the buffer, only DATA frames are transferred. When a DATA frame is received the difference between an OUT and a SETUP can be determined by examining the SU bit in the **usbd_epNcs** register.

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.

#### 6.3.1.3 Using DMA

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

When setting up a buffer to service a series of IN transactions the DMA size should be set to either less than MAXPACKET or a multiple of MAXPACKET. The size in the **usbd_epNcs** reister should be set to MAXPACKET for all but the last buffer and to the actual remaining transfer size for the last packet. The size must be set correctly 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_epNcs** 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. This will happen when DATA packets are not multiples of 4 bytes long because the DMA controller will not receive a request when less than 4 bytes are in the FIFO.

#### 6.3.1.4 Special Handling of SETUP Transactions

Since SETUP transactions do not follow a simple IN/OUT model some special handling is required. If a SETUP transaction has an OUT phase, the acknowledgment is implied by a zero length DATAx packet returned in response to a following IN. For this data packet to be sent it is necessary to write a byte to the **usbd_ep0wr** FIFO after the SETUP is received and before the IN is received. Failure to do so will result in a NAK in response to the IN which

will imply a failure to execute the command. Software must flush this data from the **usbd_ep0wr** FIFO when the IN is completed.

#### 6.3.1.5 Automatic Execution of Commands

Some standard setup commands directed at endpoint zero will be automatically serviced by the USB device hardware. These commands are still passed to the memory buffer but are indicated by the N bit in **usbd_ep0cs**. No further action is required to service these commands although they may be used to signal state changes within the software.

The following commands will be automatically serviced:

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

#### 6.3.1.6 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.

#### 6.3.1.7 Automatic Suspension

If the USB is idle for more than 3ms the device controller will enter a suspend state. In this state the device controller will 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.

### 6.3.2 Hardware considerations

The USB root hub port pins have USB 1.1 compliant drivers. No external driver circuitry is required.

The USB Device implementation is full speed and requires the termination mentioned below. Low speed is not supported.

Table 33 shows the pins associated with the USB Device.

**TABLE 33. USB Pins**

| Pin Name | Input/ Output | Description |
|---|---|---|
| **USB (Device)** | | |
| USBDP | IO | Positive signal of differential USB device driver<br>Requires a 1,5k pullup to denote a full speed device.<br>*Muxed with USBH0P* |
| USBDM | IO | Negative signal of differential USB device driver<br>*Muxed with USBH0M* |

For changing pin functionality please refer to the **sys_pinfunc** register in Section 7.3, "Primary General Purpose I/O".

## 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 34.

**TABLE 34. 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 35. The register set index is described in Table 36.

**TABLE 35. IrDA Base Address**

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

## TABLE 36. IrDA Registers

| Offset | 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 |

### 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.

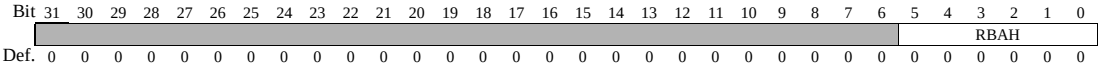
**ir_rngptrstat - Infrared Ring Pointer Status** Offset = 0x0000

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:15 | RES | These bits are reserved and are read as 0. | R | 0 |
| 14 | RES | This bit is always read as 1. | R | 1 |
| 13:8 | TRPI | Transmit Ring Pointer Index<br>Gives the current pointer location in the transmit ring buffer. | R | 0 |
| 7:6 | RES | This bits are reserved and are always read as 0. | R | 0 |
| 5:0 | RRPI | Receive Ring Pointer Index<br>Gives the current pointer location in the receive ring buffer. | R | 0 |

### 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

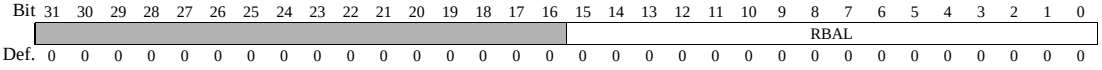



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:6 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 5:0 | RBAH | Ring buffer base address bits [31:26] | R/W | 0 |

### 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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15:0 | RBAL | Ring buffer base address bits [25:10] | R/W | 0 |

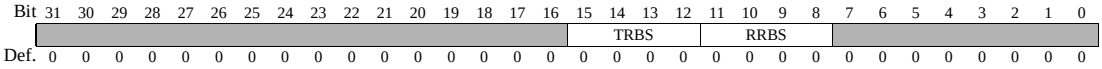
### Infrared Ring Size Register

This register defines the size for both the transmit and receive ring buffers. Each ring buffer size is programmed using an encoded value from the following table:

**TABLE 37. Ring Buffer Sizes**

| Encoding | Ring Buffer Size |
|---|---|
| 0000 | 4 |
| 0001 | 8 |
| 0011 | 16 |
| 0111 | 32 |
| 1111 | 64 |

**ir_ringsize - Infrared Ring Size Register** Offset = 0x000C

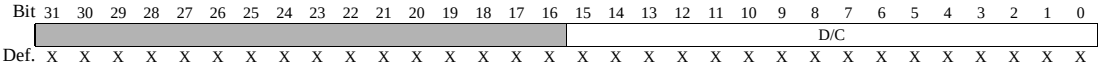



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15:12 | TRBS | Transmit ring buffer size encoded using Table 37. | R/W | 0 |
| 11:8 | RRBS | Receive ring buffer size encoded using Table 37. | R/W | 0 |
| 7:0 | RES | This bit is always read/written as 0. | R/W | 0 |

### 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



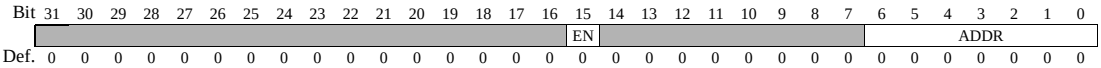

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written as 0. | W | UNPRED |
| 15:0 | D/C | Don't care. | W | UNPRED |

### 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

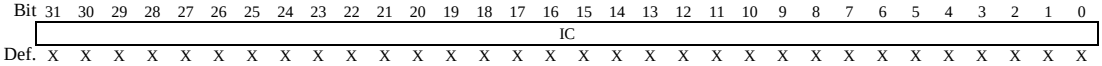



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R | 0 |
| 15 | EN | Address comparison enable<br>1 - address comparison enabled.<br>0 - address comparison disabled. | R/W | 0 |
| 14:8 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 7:0 | ADDR | IrDA packet address to compare | R/W | 0 |

### IrDA Interrupt Clear

Writing to this register will clear all pending IrDA interrupts.

**ir_intclear - IrDA Interrupt Clear** Offset = 0x0018

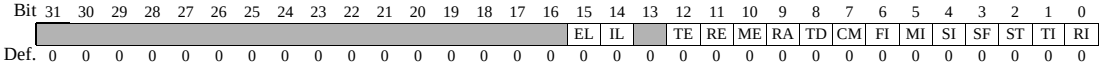



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:0 | IC | Interrupt Clear<br>Any write to this register will clear all pending IrDA interrupts. | W | UNPRED |

### Infrared Configuration Register 1

This register defines general setup parameters for the IrDA controller.

**ir_config1 - Infrared Configuration Register 1** Offset = 0x0020



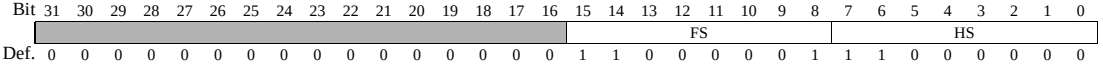

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15 | EL | Enable external transmit while in loopback. | R/W | 0 |
| 14 | IL | Enable internal loopback. | R/W | 0 |
| 13 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 12 | TE | Transmit enable. | R/W | 0 |
| 11 | RE | Receive enable. | 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.<br>This bit should always be set for normal operation. | R/W | 0 |
| 9 | RA | Receive all small/runt packets of size less than 4 bytes (SIR mode only). | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 8 | TD | Transparency destuffing disable for SIR receive filter<br>1 - transparency destuffing disabled<br>0 - transparency destuffing enabled | R/W | 0 |
| 7 | CM | Cyclical Redundancy Check (CRC) mode<br>1 - 16-bit CRC<br>0 - 32-bit CRC | R/W | 0 |
| 6 | FI | Fast infrared mode enable (FIR)<br>When this bit is set the IRFIRSEL output will be a 1 | R/W | 0 |
| 5 | MI | Medium infrared mode enable (MIR)<br>When this bit is set the IRFIRSEL output will be a 1 | R/W | 0 |
| 4 | SI | Slow infrared mode enable (SIR)<br>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 |

### 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

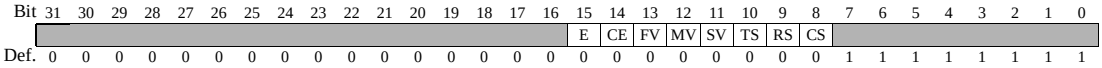



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read as 0. | R | 0 |
| 15:8 | FS | Footer bit sequence for end-of-frame | R | 0xC1 |
| 7:0 | HS | Header bit sequence for start-of-frame | R | 0xC0 |

### 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



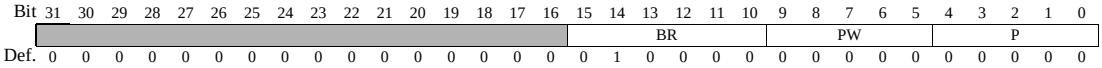

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15 | E | Enable PHY layer. | R/W | 0 |
| 14 | CE | Configuration Error<br>This bit is set when more than one operating mode (SIR, MIR, or FIR) is enabled 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 | RES | These bits are always read as 1. | R | 0xFF |

### 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



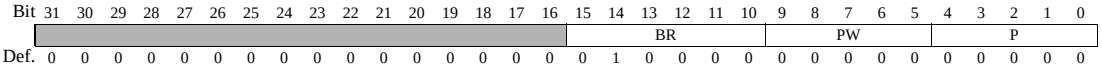
| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read as 0. | R | 0 |
| 15:10 | BR | Baud rate (see “Programming Considerations” on page 146) | R | 0 |
| 9:5 | PW | Pulse width (see “Programming Considerations” on page 146) | R | 0 |
| 4:0 | P | This register will determine the number of preamble bytes to send for FIR, or start flags for MIR.<br>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).<br>This field does not apply to SIR. | R | 0 |

### 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



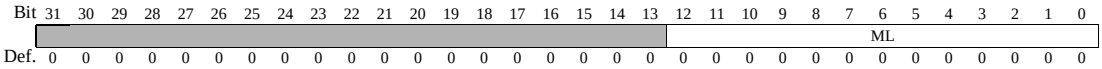

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15:10 | BR | Baud rate (see “Programming Considerations” on page 146) | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 9:5 | PW | Pulse width (see “Programming Considerations” on page 146) | R/W | 0 |
| 4:0 | P | This register will determine the number of preamble bytes to send for FIR, or start flags for MIR.<br>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).<br>This field does not apply to SIR. | R/W | 0 |

### 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

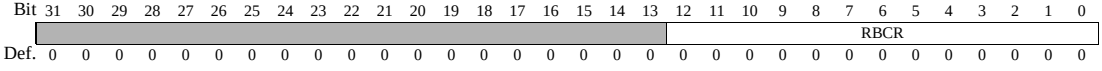



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 12:0 | ML | Maximum received packet length. | R/W | 0 |

### Infrared Receive Byte Count Register

This register returns the current number of received bytes.

**ir_rxbytecnt - Infrared Receive Byte Count Register** Offset = 0x0038

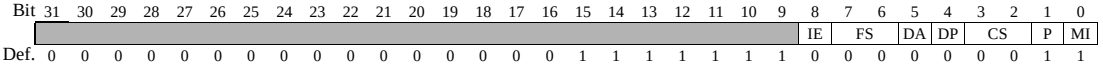



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:13 | RES | These bits are reserved and are read as 0. | R | 0 |
| 12:0 | RBCR | Received byte count. | R | 0 |

### Infrared Configuration Register 2

This register defines general setup parameters for the IrDA controller.

**ir_config2 - Infrared Configuration Register 2** Offset = 0x003C




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 15:9 | RES | These bits are reserved and are read as 1. | R | 0x7F |
| 8 | IE | Interrupt Enable<br>Setting this bit will allow interrupts to be generated when a ring buffer has been transmitted or received.<br>Writing to the ir_intclear register will clear all pending interrupts. | R/w | 0 |
| 7:6 | FS | Filter selection for finite impulse response DPLL<br>11 - lowest filter<br>10 - medium low filter<br>01 - medium high filter<br>00 - highest filter | R/W | 0x0 |
| 5 | DA | Disable adjacent pulse width packet circuit in the FIR DPLL.<br>1 - circuit disabled<br>0 - circuit enabled | R/W | 0 |
| 4 | DP | Disable pulse width adjustment circuit in the FIR DPLL. | R/W | 0 |
| 3:2 | CS | PHY layer clock speed<br>11 - 64 MHz<br>10 - 56 MHz<br>01 - 48 MHz<br>00 - 40 MHz<br>The IrDA Clock must be set to match value set in CS. The IrDA clock is programmed from the clock generator which is documented in Section 7.1, Clocks. | R/W | 0x0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 1 | P | One receive pin mode<br>1 - one pin each for receive and speed select (slow or fast)<br>0 - two pins for receive | R/W | 0 |
| 0 | MI | Mode inversion (when P=1)<br>1 - Fast speed is chosen by asserting speed select high.<br>0 - Fast speed is chosen by asserting speed select low. | R/W | 0 |

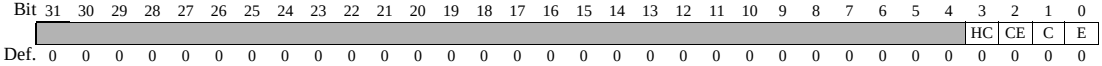
### Infrared Enable Register

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, CA, and E bit set appropriately.

**ir_enable - Infrared Enable Register** Offset = 0x0040




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| 3 | HC | Half clock speed mode<br>1 - IrDA clock will run at one-half system bus frequency.<br>0 - IrDA clock will run at full system bus frequency.<br>HC is not just for power savings. HC must be 1 if the system bus is greater than 100MHz. | R/W | 0 |
| 2 | CE | Clock enable - Setting this bit will enable clocks. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 1 | C | Coherent<br>1 - Memory accesses are marked coherent<br>0 - Memory accesses are marked non coherent<br>For more information on coherency see Section 2.8.2, "SBUS Coherency Model" for more information on coherency. | R/W | 1 |
| 0 | E | Endian mode<br>1 - Big Endian<br>0 - Little Endian | R/W | 1 |

## 6.4.2 Hardware Considerations

Table 38 describes the connection between the IrDA peripheral and the external transceiver.

**TABLE 38. IrDA Hardware Connections**

| Pin | Input/Output | Description |
|---|---|---|
| IRDATX | O | Serial IrDA output<br>*Muxed with GP211 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.<br>*Muxed with GPIO15 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".

## 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" 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 is greater than 100MHz, 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 system bus if it has the data, and E should be set for the appropriate endianess.

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 system bus. 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 39 shows the proper parameters to configure communications speed and IrDA mode.

**TABLE 39. IrDA PHY Configuration Table**

| Mode | Speed (bps) | Baud Rate | Pulse Width min - nom - max | | | Preamble/ Start Flags |
|---|---|---|---|---|---|---|
| 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 should be set to 1) |
| FIR | 4000000 | 0 | N/A | N/A | N/A | 16 (P field should be set to 15) |

Table 40, Table 41 and Table 42 show the ordered steps for programming the IrDA peripheral for each mode.

### TABLE 40. Fast Infrared Mode (FIR)

| Step | Register | Value | Notes |
|---|---|---|---|
| 1 | ir_intcfg | 0x000C | Enable Coherency (C) and Half Clock (HC). |
| 2 | ir_enable | 0x0000 | Clear bit E to allow peripheral programming (disable IrDA). |
| 3 | ir_mxpktlen | 0x0020 | 32 bytes maximum per packet |
| 4 | ir_wrphycfg | 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).<br>NOTE: set pin inversion bits (TI and/or RI) accordingly for proper transceiver operation. |
| 6 | ir_rngbsadrl | user defined | Write the logical address of ring buffer memory.<br>NOTE: The final address must have zeroes for address bits 9:0 (i.e. the address must reside on a 1 kByte boundary). |
| 7 | ir_rngbsadrh | user defined | Write the logical address of ring buffer memory.<br>NOTE: The final address must have zeroes 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_enable | 0x8000 | Set bit E to enable the peripheral, then read register again for correct status (should equal 0xC7FF). |
| 11 | ir_rngprompt | 0x0000 | Write a zero to this register to start the IrDA transfers. |

### TABLE 41. Medium Infrared Mode (MIR)

| Step | Register | Value | Notes |
|---|---|---|---|
| 1 | ir_intcfg | 0x000C | Enable Coherency (C) and Half Clock (HC). |
| 2 | ir_enable | 0x0000 | Clear bit E to allow peripheral programming (disable IrDA). |
| 3 | ir_mxpktlen | 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).<br>NOTE: Set pin inversion bits (TI and/or RI) accordingly for proper transceiver operation. |
| 6 | ir_rngbsadrl | user defined | Write the logical address of ring buffer memory.<br>NOTE: the final address must have zeroes for address bits 9:0 (i.e. the address must reside on a 1 kByte boundary). |
| 7 | ir_rngbsadrh | user defined | Write the logical address of ring buffer memory.<br>NOTE: The final address must have zeroes 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_enable | 0x8000 | Set bit E to enable the peripheral, then read register again for correct status (should equal 0xA7FF). |
| 11 | ir_rngprompt | 0x0000 | Write a zero to this register to start the IrDA transfers. |

**TABLE 42. Slow Infrared Mode (SIR)**

| Step | Register | Value | Notes |
|---|---|---|---|
| 1 | ir_intcfg | 0x000C | Enable Coherency (C) and Half Clock (HC). |
| 2 | ir_enable | 0x0000 | Clear bit E to allow peripheral programming (disable IrDA). |
| 3 | ir_mxpktlen | 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 logical address of ring buffer memory.<br>NOTE: The final address must have zeroes for address bits 9:0 (i.e. the address must reside on a 1 kByte boundary). |
| 7 | ir_rngbsadrh | user defined | Write the logical address of ring buffer memory.<br>NOTE: The final address must have zeroes 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_enable | 0x8000 | Set bit E to enable the peripheral, then read register again for correct status (should equal 0xA7FF). |
| 11 | ir_rngprompt | 0x0000 | Write a zero to this register to start the IrDA transfers. |

### 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 used is programmed via the **ir_ring_size** register.

The format for each transmit ring entry is shows in Figure 25.

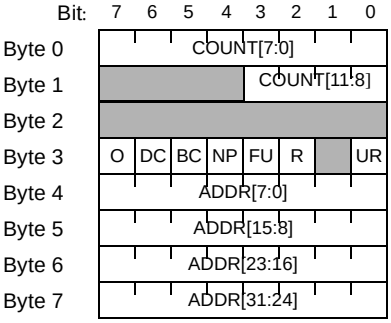



**FIGURE 25. Transmit Ring Buffer Entry Format**

| Bit(s) | 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 | COUNT | Number of bytes to transmit (upper 4 bits) | R/W | user assigned |
| Byte 1: bits 3:0 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 2: bits 7:0 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 3: bit 7 | O | Ownership flag<br>1 - Hardware has ownership of the packet and is sending the packet data to the transmitter.<br>0 - User has ownership of the packet.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 6 | DC | Disable the transmit CRC.<br>1 - used by IrDA SIR mode.<br>0 - used for synchronous packet operation.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| Byte 3: bit 5 | BC | Force a bad CRC.<br>1 - Send an 'invalid' CRC flag in the packet. Used to test receiver CRC checking.<br>0 - Normal CRC operation.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 4 | NP | Need an indication pulse.<br>1 - Transmit an indication pulse after the packet has been transmitted.<br>0 - Normal operation.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 3 | FU | Force an underrun condition.<br>1 - Force an underrun on this packet. Packet size must be greater than 18 bytes. Used for testing only.<br>0 - Normal operation. | R/W | user assigned |
| Byte 3: bit 2 | R | Request to disable transmitter.<br>1 - Hardware will clear **ir_config_1** transmit enable (TE) bit after this packet has been transmitted. Used to shut down the transmitter immediately after the last packet.<br>0 - Normal operation. | R/W | user assigned |
| Byte 3: bit 1 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 3: bit 0 | UR | Hardware Underrun error<br>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 26.

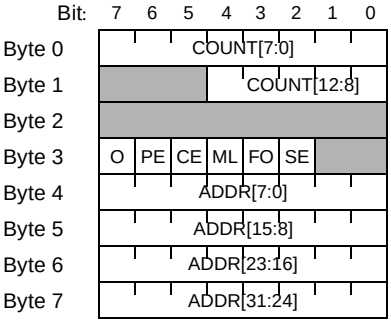



**FIGURE 26. Receive Ring Buffer Entry Format**

| Bit(s) | 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 | COUNT | Number of bytes to transmit (upper 4 bits) | R/W | user assigned |
| Byte 1: bits 3:0 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 2: bits 7:0 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 3: bit 7 | O | Ownership flag<br>1 - Hardware has ownership of the packet and is sending the packet data to the transmitter.<br>0 - User has ownership of the packet.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 6 | DC | Disable the transmit CRC.<br>1 - Used by IrDA SIR mode.<br>0 - Used for synchronous packet operation.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| Byte 3: bit 5 | BC | Force a bad CRC.<br>1 - Send an 'invalid' CRC flag in the packet. Used to test receiver CRC checking.<br>0 - Normal CRC operation.<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 4 | NP | Need an indication pulse.<br>1 - Transmit an indication pulse after the packet has been transmitted.<br>0 - Normal operation<br>NOTE: This bit will be cleared by the hardware when packet has been transmitted. | R/W | user assigned |
| Byte 3: bit 3 | FU | Force an underrun condition.<br>1 - Force an underrun on this packet. Packet size must be greater than 18 bytes. Used for testing only.<br>0 - Normal operation. | R/W | user assigned |
| Byte 3: bit 2 | R | Request to disable transmitter.<br>1 - Hardware will clear **ir_config_1** transmit enable (TE) bit after this packet has been transmitted. Used to shut down the transmitter immediately after the last packet.<br>0 - Normal operation. | R/W | user assigned |
| Byte 3: bit 1 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 3: bit 0 | UR | Hardware Underrun error<br>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 |

| Bit(s) | 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 | RES | These bits are reserved and are 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 | RES | These bits are reserved and are read/written as 0. | R/W | 0 |
| Byte 3: bit 7 | O | Ownership flag<br>1 - Hardware has ownership of the packet and is writing packet data from the receiver to memory.<br>0 - User has ownership of the packet.<br>NOTE: This bit will be cleared by the hardware when 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.<br>Valid for FIR and MIR modes only. | R/W | user assigned |
| Byte 3: bit 4 | ML | Maximum packet length reached.<br>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.<br>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 | RES | These bits are reserved and are read/written as 0. | R/W | 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 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 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 |

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 Chapter 5, Interrupt Controller.

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 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 Au1100 and selectable by the customer as long as support for the MII is provided.

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 core 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 us for 10Mbps and 0.96us for 100Mbps.

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
- 10/100Mbs 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 4bit 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 Register Descriptions

The Ethernet MAC contained in the Au1100 is located at the base addresses shown in Table 43. In addition, both the enable register and the MAC DMA register base address is shown.

**TABLE 43. 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 |

#### 6.5.1.1 MAC Registers

The Ethernet MAC registers and their associated offsets are listed in Table 44. Each Ethernet MAC has an identical register set with identical offsets. The associated base addresses for each ethernet MAC are shown in Table 43. The registers are only presented once.

**TABLE 44. MAC Register Index List**

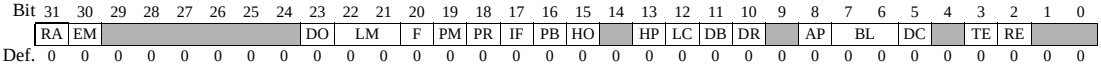
| Offset | Register Name | Description |
|---|---|---|
| 0x0000 | mac_control | Operation Mode and address filter |
| 0x0004 | mac_addrhigh | High 16 bits of the MAC physical address |
| 0x0008 | mac_addrlow | Lower 32 bits of the MAC physical address |
| 0x000C | mac_hashhigh | High 32 bits of the Multicast 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 |

### MAC Control Register

The MAC Control Register establishes the receive and transmit operating modes and controls for address filtering and packet filtering.

It should be noted 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




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | RA | Receive All<br>0 - Normal Operation<br>1 - All incoming packets will be received regardless of the destination address.<br>The address filter status is reported in Receive Status bit Filtering Fail.<br>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:<br>0 - Little Endian<br>1 - Big Endian<br>The Endian mode is only for data buffers. | R/W | 0 |
| 29:24 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 23 | DO | Disable Receive Own<br>0 - The MAC receives all packets that are given by the PHY.<br>1 - The MAC disables reception of frames when the TXEN is asserted. The MAC will ignore any loop backed receive packets.<br>This bit should be reset when the Full Duplex Mode bit is set or the Operating Mode is set to other than Normal Mode. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 22:21 | LM | Loopback Operating Mode<br>$00_b$ Normal Mode<br>$01_b$ Internal Loopback<br>$10_b$ External Loopback<br>$11_b$ Reserved | R/W | $00_b$ |
| 20 | F | Full Duplex Mode<br>0 - half duplex mode<br>1 - full duplex mode<br>Changing of this bit is permitted only if the transmitter and receiver are disabled. | R/W | 0 |
| 19 | PM | Pass All Multicast:<br>0 - Normal<br>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.<br>Incoming frames with physical address destinations are filtered according to bit 13, HP and bit 15, HO. | R/W | 0 |
| 18 | PR | Promiscuous Mode<br>0 - Normal<br>1 - Any incoming valid frame is received regardless of its destination address.<br>The Filter Fail bit is always reset in Promiscuous Mode. | R/W | 1 |
| 17 | IF | Inverse Filtering<br>0 - Normal<br>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.<br>This is valid only during perfect filtering mode. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 16 | PB | Pass Bad Frames<br>0 - Normal<br>1 - All incoming frames that passed the address filtering are received including runt frames, collided frames, or truncated frames caused by buffer overflow.<br>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 to '1'. | R/W | 0 |
| 15 | HO | Hash Only Filtering Mode<br>0 - perfect address filtering mode for physical addresses<br>1 - imperfect address filtering mode both for physical and multicast addresses<br>Setting this bit to 1 is only valid if bit HP is set to 1. | R/W | 0 |
| 14 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 13 | HP | Hash/Perfect Filtering Mode<br>0 - Address Check block does a perfect address filter of incoming frames according the address specified in the MAC Address register.<br>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. | R/W | 0 |
| 12 | LC | Late Collision Control:<br>0 - When reset, the MAC aborts the frame transmission on a late collision.<br>1 - Enables the retransmission of the collided frame even after the collision period (late collision).<br>In either case the Late Collision Status is appropriately updated in the Transmit Packet Status. This bit is valid only when the MAC is operating in Half Duplex mode. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 11 | DB | Disable Broadcast Frames<br>0 - Forwards all the broadcast frames to the application (Packet Filter bit is set).<br>1 - Disables the reception of broadcast frames (Packet Filter bit is reset). | R/W | 0 |
| 10 | DR | Disable Retry<br>0 - The MAC will attempt 16 transmissions before signaling a retry error.<br>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.<br>This bit is valid only when the MAC is operating in Half Duplex mode. | R/W | 0 |
| 9 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 8 | AP | Automatic Pad Stripping<br>0 - the MAC will pass all the incoming frames to the host unmodified.<br>1 - the MAC will strip the pad field on all the incoming frames if the length field is less than 46 bytes. The FCS field will also be stripped, since it is computed at the transmitting station based on the data and pad field characters, and will 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).<br>Pad stripping is done only on the IEEE 802.3 formatted frames (frames with Length field). | R/W | 0 |
| 7:6 | BL | Backoff Limit<br>The Backoff limit determines the integer number of slot times the MAC waits before rescheduling a transmission attempt (during retries after a collision). | R/W | $00_b$ |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 5 | DC | Deferral Check<br>0 - The deferral check is disabled in the MAC and the MAC defers indefinitely.<br>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.<br>This bit is valid only when the MAC Core is operating in Half Duplex Mode. | R/W | 0 |
| 4 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 3 | TE | Transmitter Enable<br>0 - The MAC transmitter is disabled and will not transmit any frames on the MII interface.<br>1 - The MAC transmitter is enabled and it will transmit frames from the buffer on to the MII interface. | R/W | 0 |
| 2 | RE | Receiver Enable<br>0 - The MAC receiver is disabled and will not receive any frames from the MII interface.<br>1 - The MAC receiver is enabled and will receive frames from the MII interface. | R/W | 0 |
| 1:0 | RES | These bits are reserved and should be written a 0. | R | 0 |

### MAC Address High Register

### MAC Address Low Register

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.

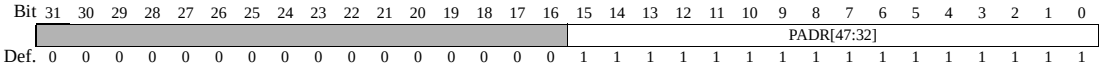
The MAC address will be compared with the destination address from the incoming frame with PADR[0] (bit 0 of the Mac Address Low register) being compared with the first bit of the destination address and PADR[47] (bit 15 of the MAC Address High register) compared with the 48th bit of the destination 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

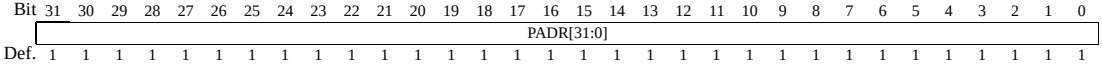
mac_addrlow = 0x0cc25000

**mac_addrhigh** Offset = 0x0004




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | PADR[47:32] | Physical Address [47:32]<br>This field contains the upper 16 bits (47 to 32) of the Physical Address of the MAC. | R/W | 0xFFFF |

**mac_addrlow** Offset = 0x0008




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | PADR[31:0] | Physical Address [31:0]<br>This field contains the lower 32 bits (31 to 0) of the Physical Address of the MAC. | R/W | 0xFFFFFFFF |

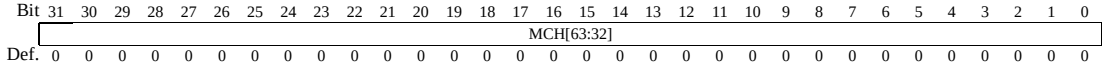
**Multicast Address High Hash Table Register**

**Multicast Address 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.

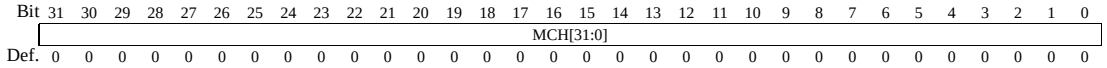
If the corresponding bit in the hash table is '1', then the multicast frame is accepted, otherwise it is rejected. If the Pass All Multicast is set, then all multi-cast frames are accepted regardless of the multi-cast hash values. The Multi Cast Hash Table High Register contains the higher 32 bits of the hash table and the Multi Cast Hash Table Low Register contains the lower 32 bits of the hash table.

**mac_hashhigh** Offset = 0x000C




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | MCH[63 :32] | Multicast Address Hash Table High<br>These bits map to the upper 32 bits of the 64 bit hash table. | R/W | 0x00000 000 |

**mac_hashlow** Offset = 0x0010

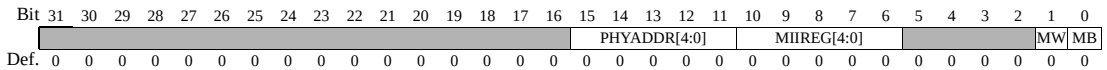



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | MCH[31: 0] | Multicast Address Hash Table Low<br>These bits map to the lower 32 bits of the 64 bit hash table. | R/W | 0x00000 000 |

### 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

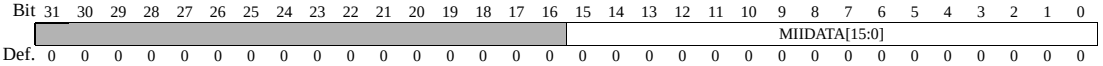


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:11 | PHYADDR | Phy Address<br>These bits tell which of the 32 possible PHY devices are being accessed. | R/W | $00000_b$ |
| 10:6 | MIIREG | MII register<br>These bits select the desired MII register in the selected PHY device. | R/W | $00000_b$ |
| 5:2 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 1 | MW | MII Write<br>0 - operation will be a read (data read is placed in MII Data Register)<br>1 - operation will be a write (data to be written is taken from MII Data Register) | R/W | 0 |
| 0 | MB | MII Busy<br>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.<br>This bit will be set by the MAC Core to signify that a read or write access to 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.<br>The MII address register should not be written to until this bit is cleared. The MAC Core will clear this bit after the PHY access is done. | R/W | 0 |

### 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

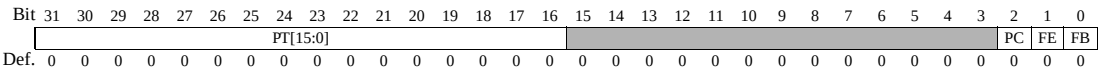



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | MII-DATA | MII Data<br>16-bit value read from the PHY after a MII read operation or the 16-bit data value to be written to the PHY before a MII write operation. | R/W | 0x0000 |

### Flow Control Register

This register is used to control the generation and reception of the Control (PAUSE Command) frames by the MAC Core'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 transmittred. 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.

**mac_flowctrl** Offset = 0x001C

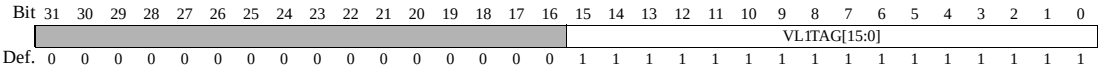


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | PT | Pause Time<br>This field will be used in the PAUSE TIME field in the generation of the PAUSE control frame. | R/W | 0x0000 |
| 15:3 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 2 | PC | Pass Control Frame<br>0 - The MAC Core will decode the control frames but will not pass the frames to the Host. The Control Frame bit in the Receive Status (bit 25) will be 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.<br><br>1 - When set, the control frames are passed to the Host. The MAC will decode 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 Transmitter Pause Mode signal indicates the current state of the MAC Transmitter. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | FE | Flow Control Enable<br>0 - The flow control operation in the MAC Core is disabled and the Core does not decode the frames for control frames. This bit is valid only when the MAC Core is operating in Full Duplex mode.<br>1 - The MAC Core is enabled for flow control operation and it will decode all the incoming frames for control frames. If the MAC Core receives a valid control frame (PAUSE command), it will disable the transmitter for the specified time. | R/W | 0 |
| 0 | FB | Flow Control Busy<br>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 to '1'. During a transfer of Control Frame, this bit will continue to be set to signify that a frame transmission is in progress. After the completion of PAUSE control frame transmission, the MAC Core will reset to '0'.<br>The Flow Control register should not be written to until this bit is cleared. | R/W | 0 |

### VLAN1 Tag Register

**mac_vlan1** Offset = 0x0020




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | VL1TAG | VLAN 1 Tag Identifier<br>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 |

**VLAN2 Tag Register**

**mac_vlan2** Offset = 0x0024


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | VL2TAG | VLAN 2 Tag Identifier<br>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 |

### 6.5.1.2 Enable Registers

Each ethernet MAC has an identical enable register. Both enable registers are located off of the **macen_base** shown in Table 43.

**MAC0 Enable**

**MAC1 Enable**

The enable register for each MAC contains a bit that enables the entire block. The block should be disabled if not in use to minimize power consumption. In addition, each enable register contains a TOSS bit which will keep frames that do not pass the address filter from being put into memory.

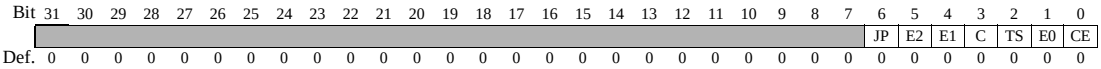
The process for bringing the MAC out of reset is the following:

1. Enable Clocks.
2. Bring E[2:0] high together with the other bits configured as desired (keeping clocks enabled).

It should be noted that the MAC clocks must be running before the internal MAC registers are accessed.

**macen_mac0** Offset = 0x0000

**macen_mac1** Offset = 0x0004




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:7 | RES | These bits are reserved and should be written a 0. | W | 0 |
| 6 | JP | Jumbo Packet Enable<br>0 - Normal (Max packet length = 0x800 bytes)<br>1 - Enable Jumbo Packet (Max packet length = 0x2800 bytes) | W | 0 |
| 5 | E2 | Enable2<br>0 - Reset<br>1 - Enable | W | 0 |
| 4 | E1 | Enable1<br>0 - Reset<br>1 - Enable | W | 0 |
| 3 | C | Coherent<br>0 - Memory accesses are marked coherent on system bus<br>1 - Memory accesses are marked non coherent on system bus<br>For more information on coherency see Section 2.8.2, "SBUS Coherency Model" for more information on coherency. | W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 2 | TS | Disable Toss<br>0 - Only frames passing address filter will be passed to memory. Frames which fail length error, crc error or other non address filter failures will still be passed to memory.<br>Frames will not be passed to memory if the filter fail bit is set or the frame is a broadcast frame and broadcast frames have been disabled.<br>In promiscuous mode all frames will be passed to memory unless the disable broadcast bit is set which will prevent broadcast frames from being passed to memory.<br>Frames that are not passed to memory will be transparent to software (no status or indication will inform software).<br>1 - All frames will be passed to memory, regardless of address filter result. | W | UNPRED |
| 1 | E0 | Enable 0<br>0 - Reset<br>1 - Enable | W | 0 |
| 0 | CE | Clock Enable<br>0 - Clocks Disabled to MAC<br>1 - Clocks Enabled to MAC | W | 0 |

### 6.5.2 DMA Registers

The MAC has four DMA buffers for both receive and transmit (4 for RX, 4 for TX). The DMA buffers will be serviced in a round robin fashion. Each MAC has a 32 word FIFO for both transmit and receive. The transfer size for the MAC DMA is 8 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 DMA registers can be described as a set of transmit and receive entries off of the MAC DMA base addresses shown in Table 43.

Each MAC DMA base address contains 8 entries which correspond to 4 transmit buffer entries and 4 receive buffer entries as shown in Table 45.

### TABLE 45. MAC DMA Entry List

| Offset | 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 |

Within each receive entry there are 2 registers implemented as shown in Table 46 (The third and fourth reserved entries are shown for completeness but are not used).

### TABLE 46. MAC DMA Receive Entry Registers

| Offset | Receive Entry Register Name | 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 |

Within each transmit entry, there are 3 registers implemented as shown in Table 47 (The fourth reserved entry is shown for completeness but is not used).

### TABLE 47. MAC DMA Transmit Entry Registers

| Offset | Transmit Entry Register Name | Description |
|---|---|---|
| 0x0 | stat | Status register |

### TABLE 47. MAC DMA Transmit Entry Registers

| Offset | Transmit Entry Register Name | Description |
|---|---|---|
| 0x4 | addr | Address/enable register |
| 0x8 | len | Length register |
| 0xc | Reserved | Nothing is implemented at this offset |

To calculate the address of a specific MAC DMA buffer all offsets should be combined. For example the physical address of the MAC1 receive buffer 3 address register is

macdma1_rx3addr = mac1dma_base + rx3 + addr = 0x0 1400 4200 + 0x130 + 0x4

macdma1_rx3addr = 0x0 1400 4334

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 48.

### TABLE 48. MAC DMA Block Indexed Address Bit Definitions

| AddrBit(s) | Description |
|---|---|
| 8 | TX/RX<br>0 - Transmit Block<br>1 - Receive Block |
| 7:6 | These bits should be set to 0. |

**TABLE 48. MAC DMA Block Indexed Address Bit Definitions**

| AddrBit(s) | Description |
|---|---|
| 5:4 | MAC DMA Buffer<br>00 - Buffer 0<br>01 - Buffer 1<br>10 - Buffer 2<br>11 - Buffer 3 |
| 3:2 | Register Select<br>00 - Status Register<br>01 - Address/Enable Register<br>10 - Length Register (valid for transmit only)<br>11 - Reserved |
| 1:0 | The registers are aligned on a word boundary, so these bits should be set to 0. |

The enumerated DMA registers are shown in Appendix A .

### 6.5.2.1 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.

In the naming of the receive registers dummy variables m and n have been inserted into the name to designate MAC number (m) and buffer number (n).

#### 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.

**macdma*m*_rx*n*stat** 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 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | MI | Missed Frame<br>0 - The frame is received normally by the Application without any latency or error violations<br>1 - Indicates that a frame was missed due to an internal FIFO overrun. | R | UNPRED |
| 30 | PF | Packet Filter<br>0 - Indicates that the current frame failed the packet filter.<br>1 - Indicates that the current frame passed the packet filter that is implemented in the MAC.<br>Packet Filter will indicate failed frame when any of the following conditions happens.<br>a. FF = 0 and frame is not a Broadcast or RA is 1<br>b. Frame is Broadcast and DB is 0<br>c. Frame is not Control Frame or PC is 1<br>d. No Error Status or PB Frames is 1<br>e. Unsupported Control Frame is 0<br>The Application can use this bit to decide whether to keep the packet in the memory or flush the packet from the memory.<br>It should be noted 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<br>0 - current frame passed address filtering<br>1 - Destination Address field in the current frame failed the Address filtering. | R | UNPRED |
| 28 | BF | Broadcast Frame<br>0 - Destination address is not Broadcast.<br>1 - Destination address is all 1’s indicating broadcast address. | R | UNPRED |
| 27 | MF | Multicast Frame<br>0 - Destination address is not multicast.<br>1 - Destination address is multicast (the first bit is 1). | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 26 | UC | Unsupported Control Frame<br>0 - If the Control Frame bit is set, this bit indicates a supported control frame has been received (Pause Frame).<br>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. | R | UNPRED |
| 25 | CF | Control Frame<br>0 - Current frame is not a control frame.<br>1 - Current frame is a control frame. This bit is only set when operating in Full Duplex mode. | R | UNPRED |
| 24 | LE | Length Error<br>0 - No length error occurred.<br>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).<br>This bit is not applicable for frame lengths greater than max ethernet size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2). | R | UNPRED |
| 23 | V2 | VLAN2 ID<br>0 - no match with VLAN2 tag<br>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. | R | UNPRED |
| 22 | V1 | VLAN1 ID<br>0 - no match with VLAN1 tag<br>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. | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 21 | CR | CRC Error<br>0 - no CRC error in current frame<br>1 - CRC error occurred in received frame.<br>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. | R | UNPRED |
| 20 | DB | Dribbling Bit<br>0 - An integer multiple of eight bits was received.<br>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. | R | UNPRED |
| 19 | ME | MII Error<br>0 - no MII error<br>1 - MII error during frame reception | R | UNPRED |
| 18 | FT | Frame Type<br>0 - IEEE 802.3 Frame<br>1 - Ethernet-type frame (frame length field is greater than max ethernet size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2).<br>This bit is still applicable when Jumbo packets are enabled.<br>This bit is not valid for runt frames of less than 14 bytes. | R | UNPRED |
| 17 | CS | Collision Seen<br>0 - No collision seen during frame reception.<br>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. | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 16 | FL | Frame Too Long<br>0 - Frame size is less than or equal to max ethernet frame size (1500 bytes- normal, 1518 bytes - VLAN1, 1538 bytes - VLAN2).<br>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.<br>Frame too long is only a length indication and does not cause frame truncation. | R | UNPRED |
| 15 | RF | Runt Frame<br>0 - Frame was not damaged in collision window.<br>1 - Frame was damaged by a collision or premature termination before the collision window passed. | R | UNPRED |
| 14 | WT | Watchdog Timeout<br>0 - Frame was received before timeout occurred.<br>1 - The receive watchdog timer expired while receiving the frame. The watchdog timer inside the MAC is programmed to be twice the MaxFrameLength. When set, the Frame Length field is invalid.<br>Any time the max frame length is exceeded (0x800 bytes for normal mode, 0x2800 with Jumbo packets enabled) the WT bit will be set. | R | UNPRED |
| 13 | L[13:0] | Frame Length<br>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. | R | UNPRED |

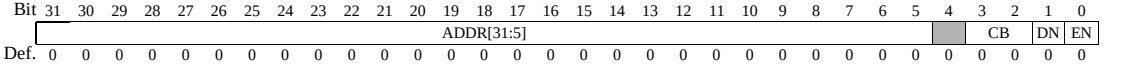
**Receive Buffer Address/Enable Register**

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 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_mac*n*** 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.

**macdma*m*_rx*n*addr** offset = 0x4




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:5 | ADDR | Buffer Address<br>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 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 3:2 | CB | Current Buffer<br>Current DMA Receive Buffer | R | 0 |
| 1 | DN | Transaction Done<br>This bit will be set by hardware to indicate that the receive transaction has been completed and that the receive packet status is valid.<br>If the respective MAC DMA interrupt is enabled (see Chapter 5, Interrupt Controller), 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.<br>This bit must be cleared explicitly by software after checking for done. This will also clear the interrupt. | R/W | 0 |
| 0 | EN | MAC DMA Enable<br>When set, this bit enables a DMA receive transaction to the memory location designated in ADDR. | R/W | 0 |

## 6.5.2.2 Transmit Registers

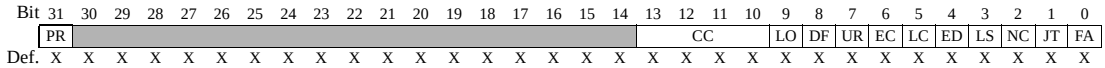
There are three transmit registers, including the status register, the address/enable register, and the length register. In the naming of the transmit registers dummy variables m and n have been inserted into the name to designate MAC number (m) and buffer number (n).

### 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.

**macdma*m*_tx*n*stat** offset = 0x0




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | PR | Packet Retry<br>0 - Transmission of current packet is complete.<br>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 | RES | These bits are reserved. | R | UNPRED |
| 13:10 | CC | Collision Count<br>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.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 9 | LO | Late Collision Observed<br>0 - No late collision observed during transmission.<br>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.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |
| 8 | DF | Deferred<br>0 - Transmitter did not defer when transmitting.<br>1 - The transmitter had to defer while ready to transmit a frame. This bit is valid only when the MAC core is operating in half-duplex mode. | R | UNPRED |
| 7 | UR | Under Run<br>0 - no data under run<br>1 - The transmitter aborted the message because of data under run during the frame's transmission. | R | UNPRED |
| 6 | EC | Excessive Collisions<br>0 - Transmission did not abort due to excessive collisions.<br>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.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |
| 5 | LC | Late Collision<br>0 - no late collision<br>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.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 4 | ED | Excessive Deferral<br>0 - no excessive deferral<br>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.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |
| 3 | LS | Loss of Carrier<br>0 - no loss of carrier<br>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).<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |
| 2 | NC | No Carrier<br>0 - carrier present<br>1 - The carrier signal from the transceiver was not present during transmission.<br><br>This bit is valid only when the MAC is operating in half-duplex mode. | R | UNPRED |

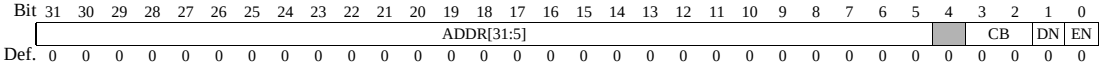
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | JT | Jabber Timeout<br>0 - no jabber timeout<br>1 - The MAC transmitter has been active for an abnormally long time (twice the Ethernet maxFrameLength size). | R | UNPRED |
| 0 | FA | Frame Aborted<br>0 - Current frame was successfully transmitted.<br>1 - The transmission of the current frame has been aborted by the MAC because of one or more of the following conditions:<br>Jabber Timeout (bit 1)<br>No Carrier (bit 2)<br>Loss of Carrier (bit 3)<br>Excessive Deferral (bit 4)<br>Late Collision (bit 5)<br>Retry Count exceeds the attempt limit (bit 6).<br>Data under run (bit 7) | R | UNPRED |

### Transmit Buffer Address/Enable Register

This register contains the starting address for the transmit memory buffer. The MAC DMA will transfer the number of bytes designated by length 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.

**macdma*m*_tx*n*addr** offset = 0x4

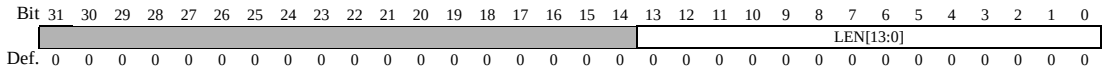


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:5 | ADDR | Buffer Address<br>Upper 27 bits of the starting physical address for the DMA buffer. This address must be cache line (32 bytes) aligned so only 27 bits are used.<br>note: This address must be written for each DMA transaction (the address will not remain after the transaction is enabled). | R/W | 0 |
| 4 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 3:2 | CB | Current Buffer<br>Current DMA Transmit Buffer | R | 0 |
| 1 | DN | Transaction Done<br>This bit will be set by hardware to indicate that the transmit transaction has been completed and that the transmit packet status is valid.<br>If the respective MAC DMA interrupt is enabled (see Chapter 5, Interrupt Controller), 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.<br>This bit must be cleared explicitly by software after checking for done. This will also clear the interrupt. | R/W | 0 |
| 0 | EN | MAC DMA Enable<br>When set, this bit enables a DMA transmit transaction from the memory location designated in ADDR. | R/W | |

### Transmit Buffer Length Register

This register contains the length of the memory buffer in bytes to be transmitted.

**macdma*m*_tx*n*len** offset = 0x8



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:14 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 13:0 | LEN | Buffer Length<br>This field sets the length of the memory buffer (in bytes).<br>When the normal bit is set the length can only be up to 0x800 bytes.<br>When the Jumbo packets are enabled in the enable register the length can be set up to 0x2800 bytes. | R/W | 0 |

### 6.5.3 Hardware Connections

Table 49 shows the pins associated with the two ethernet MAC MII interfaces.

**TABLE 49. Ethernet Pins**

| Pin Name | Input/ Output | Description |
|---|---|---|
| **Ethernet Controller 0** | | |
| N0TXCLK | I | Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100-Mb/s and 2.5 MHz when operating at 10-Mb/s. |
| N0TXEN | O | Active high. Indicates that the data nibble on N0TXD[3:0] is valid.<br>*Muxed with GP24 which controls the pin 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 TX_EN is asserted, TXD[3:0] will have the data to be accepted by the PHY. While TX_EN is de-asserted the data presented on TXD[3:0] should be ignored.<br>*Muxed with GP[28:25] which control the pins 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. |

**TABLE 49. Ethernet Pins**

| Pin Name | Input/ Output | Description |
|---|---|---|
| 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 RX_DV 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 system bus clock divided by 160.<br>*Muxed with GP215 which controls the pin 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. |

MAC1 shares its pins with GPIO[28:24]; those pins must be assigned to MAC1 in able to use MAC1. Please see Section 7.3, Primary General Purpose I/O for more information.

### 6.5.3.1 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.

## 6.5.3.2 Initialization

This section demonstrates the functional requirements for getting the MAC running. This is assuming 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 ORing 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 very basic form. Typically this would be split between an interrupt handler and another higher layer.

1. Construct Frame
2. Set length in **macdma*i*_tx*n*len** 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 **macdma*i*_tx*n*addr** register.
5. Read status (it's validity is signaled by the reception of the done signal).

The following is a very 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 I²S Controller

The Au1100 contains an I²S controller capable of interfacing with a CODEC or a discreet DAC and ADC. The I²S 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 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 I²S Register Descriptions

The I²S Interface is controlled by a register block whose physical base address is shown in Table 50.

**TABLE 50. I2S Base Address**

| Name | Physical Base Address | KSEG1 Base Address |
|---|---|---|
| i2s_base | 0x0 1100 0000 | 0xB100 0000 |

The I²S register block consists of 3 registers as shown in Table 51.

**TABLE 51. I²S Interface Register Block**

| Offset | Register Name | Type | Description |
|---|---|---|---|
| 0x0000 | i2s_data | R/W | Input and output from data FIFOs |
| 0x0004 | i2s_config | R/W | Configuration and status register |
| 0x0008 | i2s_enable | R/W | Allows port to be enabled and disabled |

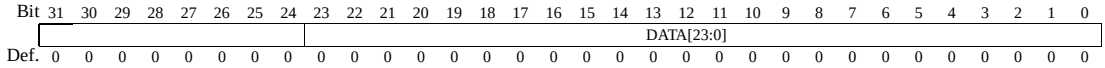
#### I²S Data

The I²S 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

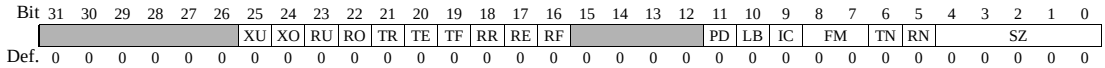



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 23:0 | DATA | Data Word<br>Up to 24 bit transmit data when written read data when read from. | R/W | |

**Configuration and Status Register**

The I2S 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:26 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 25 | XU | Transmit Underflow<br>When set to 1, this bit indicates that the transmit FIFO has experienced an underflow. This sticky bit will be cleared when read.<br>This bit can be written so care should be taken when writing the configuration register that this bit is masked if necessary. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 24 | XO | Transmit Overflow<br>When set to 1, this bit indicates that the transmit FIFO has experienced an overflow. This sticky bit will be cleared when read.<br>This bit can be written so care should be taken when writing the configuration register that this bit is masked if necessary. | R/W | 0 |
| 23 | RU | Receive Underflow<br>When set, this bit indicates that the receive FIFO has experienced an underflow. This sticky bit will be cleared when read.<br>This bit can be written so care should be taken when writing the configuration register that this bit is masked if necessary. | R/W | 0 |
| 22 | RO | Receive Overflow<br>When set, this bit indicates that the receive FIFO has experienced an overflow. This sticky bit will be cleared when read.<br>This bit can be written so care should be taken when writing the configuration register that this bit is masked if necessary. | R/W | 0 |
| 21 | TR | Transmit Request<br>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<br>This bit indicates that the transmit FIFO is empty. | R | 0 |
| 19 | TF | Transmit Full<br>This bit indicates the transmit FIFO is full. | R | 0 |
| 18 | RR | Receive Request<br>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<br>This bit indicates that the receive FIFO is empty. | R | 0 |
| 16 | RF | Receive Full<br>This bit indicates that the receive FIFO is full. | R | 0 |
| 15:12 | RES | These bits are reserved and should be written a 0. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 11 | PD | Pin Direction<br>This bit sets the direction of the I2SDIO pin.<br>0: This pin is an output.<br>1: This pin is an input (I2SDIN should not be used). | R/W | 0 |
| 10 | LB | Loopback<br>When set this bit will enable a loop back mode where data coming on the input will be presented on the output. | R/W | 0 |
| 9 | IC | Initial Channel<br>0: The left sub channel will be 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.<br>1: The right sub channel will be 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). | R/W | 0 |
| 8:7 | FM | Format<br>The following formats are supported:<br>$00_b$ - I2S mode. In this mode the first bit of a sample word will be presented after one I2SCLK delay from the transition of I2SWRD. The left sample data will be presented when the word clock is low. The data is presented MSB first.<br>$01_b$ - Left Justified mode. In this mode the first bit of a sample word will be presented on the first I2SCLK after an I2SWRD transition. The left sample data will be presented when the word clock is high. The data is presented MSB first.<br>$10_b$ - Right Justified mode. In this mode the first bit of a sample word will be presented on the first I2SCLK after an I2SWRD transition. The left sample data will be presented when the word clock is high. The data is presented LSB first. | R/W | 00 |
| 6 | TN | Transmit Enable<br>This will enable the transmit FIFO and must be enabled if the output is being used.<br>0: Disable transmit FIFO<br>1: Enable transmit FIFO | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 5 | RN | Receive Enable<br>This will enable the receive FIFO and must be enabled if either of the inputs are being used.<br>0: Disable Receive FIFO<br>1: Enable receive FIFO | R/W | 1 |
| 4:0 | SZ | Size<br>These bits will set the size of the sample word. The following combinations are valid:<br>01000 - 8 bit words<br>10000 - 16 bit words<br>10010 - 18 bit words<br>10100 - 20 bit words<br>11000 - 24 bit words<br>If using DMA it is important that memory is packed consistently with the transfer width programmed for the DMA channel and the Size field. | R/W | $10010_b$ |

### $I^2S$ Enable

The $I^2S$ Block Control register is used to enable clocks to and reset the entire $I^2S$ block.

The suggested power on reset is as follows:

1. Set both CE and D
2. Clear D for to enable the peripheral.

**i2s_enable - I2S Block Control** Offset = 0x0008

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written 0. | W | 0 |
| 1 | D | DISABLE<br>Setting this bit will disable the I2S block. After enabling the clock with CE, this bit should be cleared for normal operation. | W | 1 |
| 0 | CE | Clock Enable<br>This bit should be set to enable the clock driving the I2S block. It can cleared to disable the clock for power considerations. | W | 0 |

### 6.6.2 Hardware Considerations

Table 52 shows the signals associated with this port.

**TABLE 52. I2S Signals**

| Pin Name | Input/ Output | Description |
|---|---|---|
| I2SCLK | O | Serial bit clock.<br>Muxed with GPIO30 |
| I2SWORD | O | Word clock typically configured to the sampling frequency (Fs).<br>Muxed with GPIO31 |

**TABLE 52. I2S Signals**

| Pin Name | Input/ Output | Description |
|---|---|---|
| I2SDI | O | Serial data input which should be valid on the rising edge of I2SCLK.<br>Muxed with GPIO8 which controls the pin out of hardware reset, runtime reset and sleep. |
| 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.<br>Muxed with GPIO29 |
| EXTCLK0(1) | O | This is the System audio clock and typically is programmed to 256Fs (Fs is the sampling frequency for the system). The system audio clock will be taken from EXTCLK0 or EXTCLK1 as the signals are synchronous to I2SCLK and I2SWRD. These clocks will be programmed individually. See Section 7.1, Clocks for information. |

For changing pin functionality please refer to the **sys_pinfunc** register in Section 7.3, "Primary General Purpose I/O".

## 6.6.3 Programming Considerations

It is the programmer's responsibility to set up DMA channels and the I2SCLK and the EXTCLK0(1) to be used with the system.

The I2SWRD clock, which is typically equal to the sampling frequency, will be a function of the word width and the I2SCLK frequency. The I2SCLK and the EXTCLK0(1) are programmable as shown in Section 7.1, Clocks. The EXTCLK0(1) will be taken from the external clocks available on the pins shared with GPIO2 or GPIO3.

## 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 1.5Mbs 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 53 lists the base address for each UART register block.

**TABLE 53. 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 |

UART0 and UART3 are capable of being used with DMA. See Chapter 4, DMA Controller for more information.

## 6.7.2 UART Registers

Each register block contains the registers listed in Table 54.

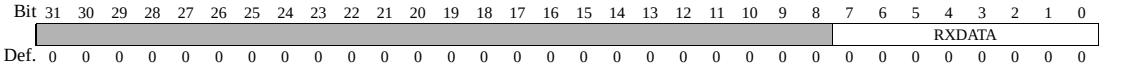
**TABLE 54. UART Registers**

| Offset | Register Name | Description |
|---|---|---|
| 0x0000 | uart_rxdata | Received Data FIFO |
| 0x0004 | uart_txdata | 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 |

### Received Data FIFO

The **uart_rxdata** register contains the next entry in the received data fifo. This register is read only.

**uart_rxdata - Received Data FIFO** Offset = 0x0000




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7:0 | RXDATA | Receive Data | R | 0 |

### Transmit Data FIFO

The **uart_txdata** register provides access to the transmit data FIFO. This register is write only.

**uart_txdata - Transmit Data FIFO** Offset = 0x0004


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7:0 | TXDATA | Transmit Data | R | 0 |

### Interrupt Enable Register

The **uart_inten** register contains bits which enable interrupts under certain operational conditions.

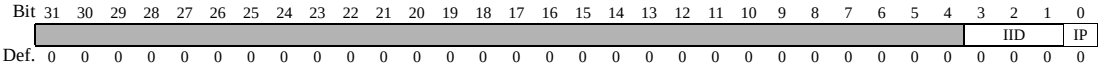
**uart_inten - Interrupt Enable Register** Offset = 0x0008

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 3 | MIE | Modem Status Interrupt Enable<br>When the MIE bit is set an interrupt will be generated when changes occur in the state of the modem control signals (UART3 only). | R/W | 0 |
| 2 | LIE | Line Status Interrupt Enable<br>When the LIE bit is set an interrupt will be generated when errors (overrun, framing, stop bits) or break conditions occur. | R/W | 0 |
| 1 | TIE | Transmit Interrupt Enable<br>When the TIE bit is set an interrupt will be generated when the transmit FIFO is not full. | R/W | 0 |
| 0 | RIE | Receive Interrupt Enable<br>When the RIE bit is set the UART will generate an interrupt on received data ready (*DR* bit in the **uart_linestat** register) or a character time out. | R/W | 0 |

### Interrupt Cause Register

The **uart_intcause** register contains information about the cause of the current interrupt.

**uart_intcause - Interrupt Cause Register** Offset = 0x000c




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 3:1 | IID | Interrupt Identifier<br>The IID field identifies the highest priority current interrupt condition. Table 55 lists the priorities and encodings of each interrupt condition. | R | 0 |
| 0 | IP | Interrupt Pending<br>The IP bit is set when an interrupt is pending. | R | 0 |

Table 55 contains information about the interrupt cause encoding.

**TABLE 55. Interrupt Cause Encoding**

| IID | Priority | Type | Source |
|---|---|---|---|
| 0 | 5 (lowest) | Modem Status | DD, TRI, DR or DC of **uart_mdmstat** |
| 1 | 4 | Transmit Buffer Available | TT of **uart_linestat** |
| 2 | 3 | Receive Data Available | The receive FIFO having greater than RFT (of **uart_fifoctrl)** bytes in it if FIFOs are enabled.<br>DR of **uart_linestat** if FIFOs are disabled. |
| 3 | 1 (highest) | Receive Line Status | OE, PE, FE, BI in **uart_linestat** register |
| 4 | | Reserved | |
| 5 | | Reserved | |
| 6 | 2 | Character Time Out | Character has been in receive FIFO for 0x300 UART clocks (set by **uart_clkdiv**) |
| 7 | | Reserved | |

### FIFO Control Register

The **uart_fifoctrl** register provides control of character buffering options.

**uart_fifoctrl - FIFO Control Register** Offset = 0x0010

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7:6 | RFT | Receiver FIFO Threshold<br>A receiver threshold interrupt will be generated when the number of characters in the receiver FIFO is greater than or equal to the trigger level listed in Table 57.<br>If using DMA it is important that the Receiver FIFO threshold and Transmit FIFO threshold are the same and programmed consistently with the Transfer Size for the DMA channel being used. See Chapter 4, DMA Controller for more information. | R/W | 0 |
| 5:4 | TFT | Transmit FIFO Threshold<br>A threshold interrupt will be 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 in Table 56.<br>If using DMA it is important that the Receiver FIFO threshold and Transmit FIFO threshold are the same and programmed consistently with the Transfer Size for the DMA channel being used. See Chapter 4, DMA Controller for more information. | R/W | 0 |
| 3 | MS | Mode Select<br>If the MS bit is clear interrupts will be 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<br>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 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 1 | RR | Receiver Reset<br>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<br>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 |

Table 56 provides trigger depth encoding information for the transmit FIFO.

### TABLE 56. Transmit FIFO Trigger Depth Encoding

| TFT | Trigger Depth |
|---|---|
| 0 | 0 |
| 1 | 4 |
| 2 | 8 |
| 3 | 12 |

Table 57 provides trigger depth encoding information for the receiver FIFO.

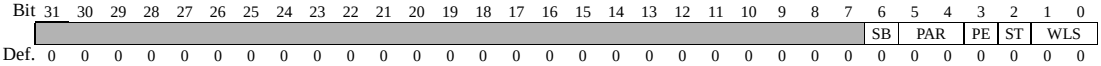
### TABLE 57. Receiver FIFO Trigger Depth Encoding

| RFT | Trigger Depth |
|---|---|
| 0 | 1 |
| 1 | 4 |
| 2 | 8 |
| 3 | 14 |

### Line Control Register

The **uart_linectrl** register provides control over the data format and parity options.

**uart_linectrl - Line Control Register** Offset = 0x0014




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:7 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 6 | SB | Send Break<br>Setting the SB bit will force the transmitter output to zero. | R/W | 0 |
| 5:4 | PAR | Parity Select<br>The PAR field selects parity encoding for the transmitter and receiver. Valid encodings are listed in Table 58. | R/W | 0 |
| 3 | PE | Parity Enable<br>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<br>If the ST bit is clear one stop bit will be 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<br>The WLS field selects the number of data bits in each character. The number of bits is WLS+5. | R/W | 0 |

Table 58 describes parity encoding for the **uart_line_control** register.

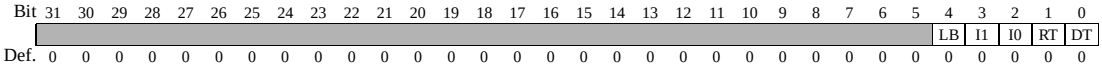
### TABLE 58. Parity Encoding

| PAR | Parity |
|---|---|
| 0 | Odd Parity |
| 1 | Even Parity |
| 2 | Mark Parity |
| 3 | Zero Parity |

### Modem Control Register

The **uart_mdmctrl** register allows the state of the output modem control signals to be set. The external modem signals are only available on UART3.

**uart_mdmctrl - Modem Control Register** Offset = 0x0018




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:5 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 4 | LB | Loop Back<br>When the LB bit is low the UART operates in normal mode. When this bit is set a loopback mode is established. This mode can be used for self-test. Table 59 lists the internal connections made in loop back mode. | R/W | 0 |
| 3 | I1 | Internal Line 1 State<br>When the I1 bit is set the internal $\overline{I1}$ line for this port is driven low. This can be used in loopback mode. | R/W | 0 |
| 2 | I0 | Internal Line 0 State<br>When the I0 bit is set the external $\overline{I0}$ line for this port is driven low.This can be used in loopback mode. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 1 | RT | Request To Send<br>When the RTS bit is set the external $\overline{\text{RTS}}$ line for this port is driven low.<br>NOTE: This bit has no effect if **uart_autoflow**[AE] is set. | R/W | 0 |
| 0 | DT | Data Terminal Ready<br>When the DTR bit is set the external $\overline{\text{DTR}}$ line for this port is driven low. | R/W | 0 |

Table 59 lists the internal connections made in loop back mode.

### TABLE 59. Loop Back Mode Connections

| Output Signal | Wrapped Back To |
|---|---|
| TXD | RXD |
| DTR | DSR |
| RTS | CTS |
| I0 | RIN |
| I1 | DCD |

#### 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

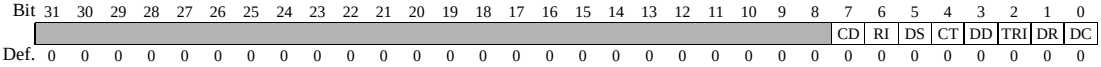
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | RF | TE | TT | BI | FE | PE | OE | DR |
| 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 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7 | RF | Receiver FIFO Contains Error<br>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<br>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<br>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.<br>When FIFOs are not enabled this bit will be set when the transmitter data register is empty | R | 0 |
| 4 | BI | Break Indication<br>This bit is set if a break is received. When a break is detected a single zero character will be received. The BI bit is valid when the zero character is at the top of the receive FIFO. This bit is cleared by a read to the uart_line_status register. | R | 0 |
| 3 | FE | Framing Error<br>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 the uart_line_status register. | R | 0 |
| 2 | PE | Parity Error<br>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 the uart_line_status register. | R | 0 |
| 1 | OE | Overrun Error<br>The OE bit is set when a receiver overrun occurs. This bit is cleared when the uart_line_status register is read. | R | 0 |
| 0 | DR | Data Ready<br>The DR bit is set when the receive FIFO contains valid characters. | R | 0 |

### 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. External modem signals are only present on UART3.

**uart_mdmstat - Modem Status Register** Offset = 0x0020



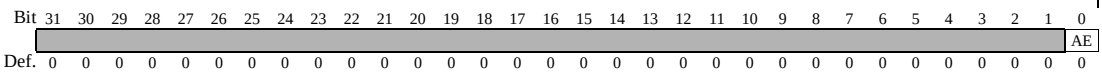

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7 | CD | Data Carrier Detect<br>The CD bit reflects the status of the external DCD pin. | R | 0 |
| 6 | RI | Ring Indication<br>The RI bit reflects the status of the external RI pin. | R | 0 |
| 5 | DS | Data Set Ready<br>The DS bit reflects the status of the external DSR pin. | R | 0 |
| 4 | CT | Clear To Send<br>The CT bit reflects the status of the external CTS pin. | R | 0 |
| 3 | DD | Delta DCD<br>The DD bit is set when a change occurs in the state of the external DCD pin. | R | 0 |
| 2 | TRI | TRI - Terminate Ring Indication<br>The TRI bit is set when a positive edge occurs in the state of the external RI pin. | R | 0 |
| 1 | DR | Delta DSR<br>The DR bit is set when a change occurs in the state of the external DSR pin. | R | 0 |
| 0 | DC | Delta CTS<br>The DC bit is set when a change occurs in the state of the external CTS pin. | R | 0 |

### Automatic Hardware Flow Control Register

The **uart_autoflow** register controls automatic hardware flow control using modem control signals CTS and RTS on UART 3. Upon enabling this mode RTS becomes and output signal

and CTS becomes an input signal. RTS is asserted low to request data until the internal FIFO reaches its preset fullness threshold. The UART will transmit and data in its internal FIFO while the CTS signal is low.

**uart_autoflow - Automatic Hardware Flow ControlRegister** Offset = 0x0024




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:1 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 0 | AE | Autoflow Enable<br>Setting this bit will enable automatic hardware flow control on UART3. Enabling this mode will override software software control of the pins. | R/W | 0 |

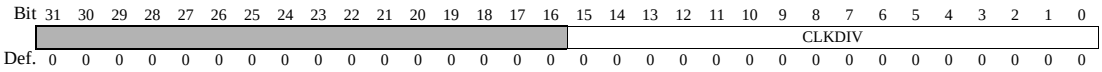
**Clock Divider Register**

The **uart_clkdiv** contains the divider used to generate the baud rate clock. The input to the clock divider is the internal pbus clock. The actual baud rate of the interface will be:

Baudrate = *CPU / (SD * 2 * CLKDIV * 16)*

CPU = CPU clock
SD = System bus divider (see Section 7.4, "Power Management" information on changing SD).

**uart_clkdiv - Clock Divider Register** Offset = 0x0028




**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


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 1 | E | Enable<br>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<br>When the CE bit is clear the module clock source is inhibited. This can be used to place the module in a low power stand-by state. The CE bit should be set before the module is enabled for proper bringup. | R/W | 0 |

### 6.7.2.1 Hardware Considerations

The UARTs consists of the signals listed in Table 60.

**TABLE 60. UART Signals**

| Pin Name | Input/ Output | Definition |
|---|---|---|
| **UART0** | | |
| U0TXD | O | UART0 transmit<br>*Muxed with GP212 which controls the pin out of hardware reset, runtime reset and sleep.* |
| U0RXD | I | UART0 receive |
| **UART1** | | |
| U1TXD | O | UART1 transmit<br>*Muxed with GP213 which controls the pin out of hardware reset, runtime reset and sleep.* |
| U1RXD | I | UART1 receive |

**TABLE 60. UART Signals**

| Pin Name | Input/ Output | Definition |
|---|---|---|
| **UART3** | | |
| U3TXD | O | UART3 transmit<br>*Muxed with GP214 which controls the pin out of hardware reset, runtime reset and sleep.* |
| U3RXD | I | UART3 receive |
| U3CTS | I | Clear to Send<br>Muxed with GPIO9 which controls the pin out of hardware reset, runtime reset and sleep. |
| U3DSR | I | Data Set Ready<br>Muxed with GPIO10 which controls the pin out of hardware reset, runtime reset and sleep. |
| U3DCD | I | Data Carrier Detect<br>Muxed with GPIO11 which controls the pin out of hardware reset, runtime reset and sleep. |
| U3RI | I | Ring Indication<br>Muxed with GPIO12 which controls the pin out of hardware reset, runtime reset and sleep. |
| U3RTS | O | Request to Send<br>Muxed with GPIO13 which controls the pin out of hardware reset, runtime reset and sleep. |
| U3DTR | O | Data Terminal Ready<br>Muxed with GPIO14 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".

# 6.8 SSI Interfaces

The Au1100 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 will always drive 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_config** 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 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 27 shows a typical write transaction. For this transaction the address field is 3 bits long, data is 8 bits, and direction precedes address.

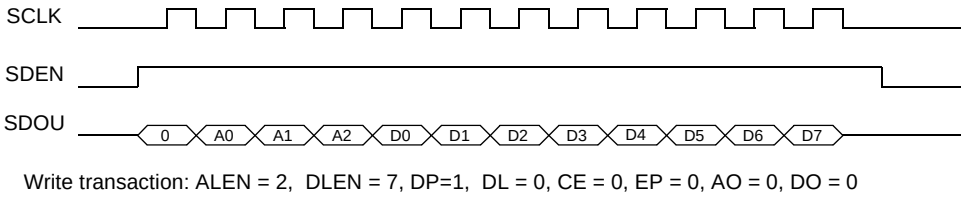



**FIGURE 27. Typical Write Transaction Timing**

### 6.8.1.2 Read Transactions

A read transaction is initiated by writing the address and direction to the ssi_adat 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 28 shows a typical read transaction where the bus mode is set to hold SCLK high during the bus turnaround.

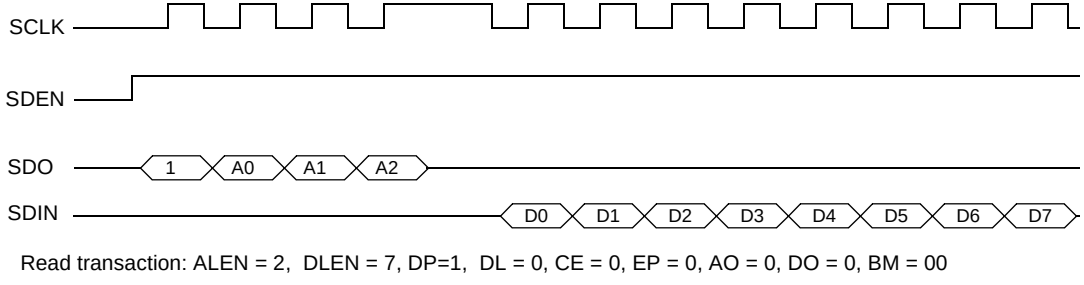



**FIGURE 28. Typical Read Transaction Timing**

## 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 61. 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 62 shows the offset and function of each register.

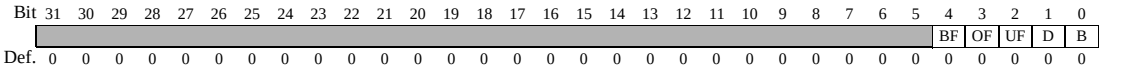
### TABLE 62. SSI Registers

| Offset | 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 |

#### SSI Interface Status Register

The **ssi_status** register reflects the current status of the interface.

**ssi_status - SSI Interface Status** Offset = 0x0000

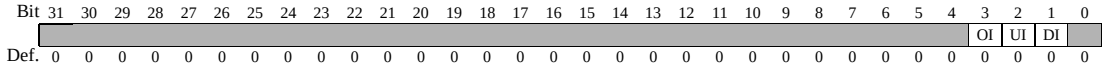


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:5 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 4 | BF | Buffer Full - This bit indicates that the data buffer is currently full. It is set by either receiving a buffer from the serial interface or a write 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.<br>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.<br>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.<br>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 |

### Interrupt Pending Register

The **ssi_int** register shows which interrupt indications are currently active.

**ssi_int - SSI Interrupt Pending Register** Offset = 0x0004



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 3 | OI | The OI bit indicates that the current interrupt is being generated by an overflow condition.<br>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.<br>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.<br>This bit is sticky. Once set high it must be written a '1' to clear the bit. | R | 0 |
| 0 | RES | This bit is reserved and should be written a 0. | R | 0 |

### 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**.

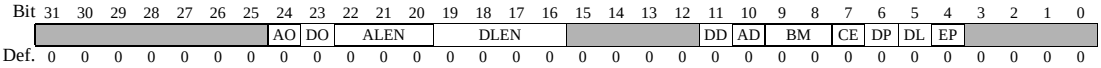
**ssi_inten - SSI Interrupt Enable Register** Offset = 0x0008


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | 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 | RES | This bit is reserved and should be written a 0. | R | 0 |

### 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




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:25 | RES | These bits are reserved and should be written as 0. | R | 0 |
| 24 | AO | Address Field Order<br>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<br>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<br>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<br>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 | RES | These bits are reserved and should be written as 0. | R/W | 0 |
| 11 | DD | Direction Bit Disable<br>If the DD bit is set the direction bit will not be sent. | R/W | 0 |
| 10 | AD | Address Field Disable<br>If the AD bit is set the address field will not be sent. | R/W | 0 |

| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 9:8 | BM | Bus Mode<br>The BM field determines the turnaround behavior for read cycles. Table 63 shows the selection matrix. | 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<br>The DP bit determines whether a write is indicated by an active high or active low direction bit. If DP is cleared a write will be indicated by an active high direction bit. If DP is set a write is indicated by a low direction bit. | R/W | 0 |
| 5 | DL | Direction Bit Location<br>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<br>The EP bit selects the polarity of the enable signal on the interface. A zero value indicates that the enable is active high, a nonzero value indicates an active low enable. | R/W | 0 |
| 3:0 | RES | This bit is reserved and should be written a 0. | R/W | 0 |

Table 63 shows the selection matrix for the **ssi_config** register.

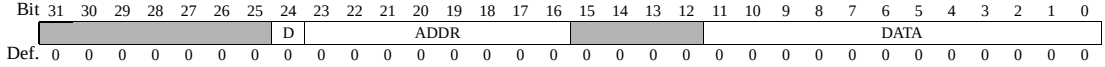
### TABLE 63. Bus Turnaround Selection

| BM | Turnaround Behavior |
|---|---|
| 0b00 | SCLK held high during turnaround |
| 0b01 | SCLK held low during turnaround |
| 0b10 | SCLK cycle during turnaround |
| 0b11 | Reserved |

### SSI Address/Data Register

The **ssi_adata** register contains the address, data, and direction fields. Writing to **ssi_adata** will initiate a transfer, the direction will be determined by the D bit. If D is clear the transaction will be a read and the data field will contain the value of the serial input at the end of the transaction. If the D bit is set the transaction will be a write.

**ssi_adata - SSI address/data Register** Offset = 0x0024




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:25 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 24 | D | Direction Bit | R/W | 0 |
| 23:16 | ADDR | Address Field | R/W | 0 |
| 15:12 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 11:0 | DATA | Data Field | R/W | 0 |

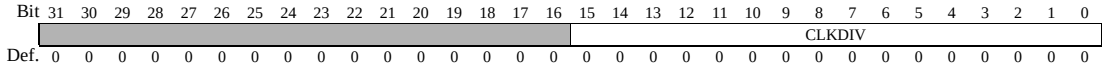
### SSI Clock Divider Register

The **ssi_clkdiv** register determines the baud rate of the serial port. The baud rate is defined as follows:

Baudrate = *CPU / (SD * 2 * (**CLKDIV** + 1))*

CPU = CPU clock
SD = System bus divider (see Section 7.4, "Power Management" for information on SD)

**ssi_clkdiv - SSI clock divider** Offset = 0x0028



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | CLKDIV | The CLKDIV field determines the baud rate of the interface. | R/W | 0 |

### 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 USB Device 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(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 1 | CD | Clock Disable<br>This bit inhibits the clock to the SSI module when set. This bit should be cleared for normal operation. | R/W | 1 |
| 0 | E | Enable<br>When this bit is clear the SSI module is held in reset. | R/W | 0 |

### 6.8.2.1 Hardware Considerations

The SSI ports consist of the signals listed in Table 64.

**TABLE 64. SSI Signals**

| Pin Name | Input/ Output | Definition |
|---|---|---|
| **SSI0** | | |
| S0CLK | O | Master only clock output. The speed and polarity of clock edge is programmable.<br>*Muxed with GP209 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.<br>*Muxed with GP208 which controls the pin out of hardware reset, runtime reset and sleep.* |
| S0DEN | O | Enable signal which frames transaction. The polarity is programmable.<br>*Muxed with GP210 which controls the pin out of hardware reset, runtime reset and sleep.* |

**TABLE 64. SSI Signals (Continued)**

| Pin Name | Input/ Output | Definition |
|---|---|---|
| **SSI1** | | |
| S1CLK | O | Master only clock output. The speed and polarity of clock edge is programmable.<br>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 S0DOUT to create a single bidirectional data signal.<br>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 S0DIN to create a single bidirectional data signal.<br>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.<br>Muxed with $\overline{\text{ACRST}}$ 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".

## 6.9 LCD Controller

The Au1100 integrated LCD controller 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 Au1100 LCD Controller are:

**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 palletized TFT
- 12/16 bpp non-palletized TFT
- 1/2/4 bpp mono STN
- 1/2/4/8 bpp palletized color STN
- 12 bpp non-palletized color STN

**Miscellaneous Features**

- Double buffering support
- Hardware Swivel (90, 180, and 270 degrees) for up to 320x240 pixel displays
- Two pulse width modulation 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 65.

**TABLE 65. LCD Base Address**

| Name | Physical Base Address | KSEG1 Base Address |
|---|---|---|
| lcd_base | 0x0 1500 0000 | 0xB500 0000 |

The register block consists of 5 registers as shown in Table 66.

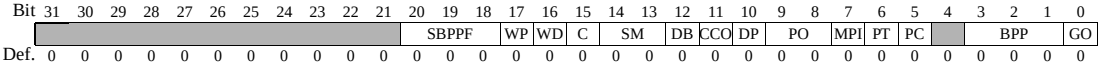
**TABLE 66. LCD Controller Registers**

| Offset | 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_pallettebase | Pallette Interface Registers |

### 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | Reserved | R | 0 |
| 20:18 | SBPPF | Sixteen Bits Per Pixel Data Format<br>000 - 6 bits Red, 5 bits Green, 5 bits Blue<br>001 - 5 bits Red, 6 bits Green, 5 bits Blue<br>010 - 5 bits Red, 5 bits Green, 6 bits Blue<br>011 - 1 bit Intensity, 5 bits Red, 5 bits Green, 5 bits Blue<br>100 - 5 bits Red, 5 bits Green, 5 bits Blue, 1 bit Intensity<br>101, 110, 111 - Reseved | R/W | 0 |
| 17 | WP | White Data Polarity<br>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<br>When this bit is high LCD_D[15:0] pins are set to the value set 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<br>1- LCD transactions are marked as coherent on the system bus.<br>0 - LCD transactions are marked as non-coherent on the system bus | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 14:13 | SM | Swivel Mode<br>00 - normal portrait<br>01 - 90 degree rotate (only supported for panels up to 320x240 pixels)<br>10 - 180 degree rotate<br>11 - 270 degree rotate (only supported for panels up to 320x240 pixels) | R/W | 0 |
| 12 | DB | TFT Data bits<br>This bit is used in paletized TFT modes to indicate how many LCD_DATA pins are to be used.<br>0 - 16 data pins<br>1 - 12 data pins | R/W | 0 |
| 11 | CCO | Color Channel Orientation<br>0 - RGB Channel Format<br>1 - BGR Channel Format | R/W | 0 |
| 10 | DP | STN Panel Type<br>0 - Single Panel<br>1 - Dual Panel | R/W | 0 |
| 9:8 | PO | Pixel Order<br>These bits show the order that pixels are packed in to words in the frame buffer. See Table 67. | R/W | 0 |
| 7 | MPI | Monochrome Panel Interface<br>0 - 4 bit monochrome panel<br>1 - 8 bit monochrome panel | R/W | 0 |
| 6 | PT | Panel Type<br>0 - STN Passive: frame rate modulation algorithm used<br>1 - TFT Active | R/W | 0 |
| 5 | PC | Panel Color<br>0 - monochrome<br>1 - color | R/W | 0 |
| 4 | RES | Reserved | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 3:1 | BPP | Bits Per Pixel<br>000 - 1 bit per pixel<br>001 - 2 bits per pixel<br>010 - 4 bits per pixel<br>011 - 8 bits per pixel<br>100 - 12 bits per pixel<br>101 - 16 bits per pixel<br>110,111 - Reserved | R/W | 0 |
| 0 | GO | LCD Go<br>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 67. 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 | | | | | | | | | | | | | | | |

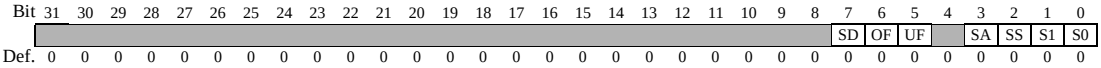
### Interrupt Registers

The Interrupt Status and Interrupt Enable registers have identical formats. If a bit is set in the interrupt enable register and the corresponding condition becomes true then an interrupt will be issued and the corresponding bit in the Interrupt Status register will be set. The interrupt for the LCD Controller should be programmed as a high level type.

All Interrupts except for shutdown must be cleared by writing a 1 to the corresponding bit in the Interrrupt Status register. 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



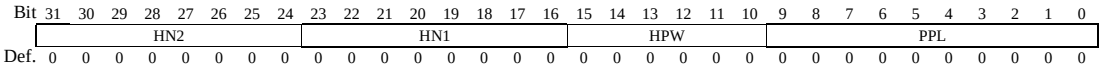

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:8 | RES | Reserved | R | 0 |
| 7 | SD | Shutdown<br>This condition occurs when the controller's GO bit is written low and the controller has finished displaying the last frame. After this bit goes high all registers of the controller can be written. | R/W | 0 |
| 6 | OF | Output Fifo Overflow | R/W | 0 |
| 5 | UF | Output Fifo Underflow<br>This can occur when there is too much traffic on the system bus, causing the LCD Controller to be unable to fetch data fast enough to refresh the LCD panel. | R/W | 0 |
| 4 | RES | Reserved | R | 0 |
| 3 | SA | Start Of Active Video<br>Occurs at the end of the vertical retrace | R/W | 0 |
| 2 | SS | Start Vertical Sync Period | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | S1 | Start Address 1 Latched<br>This interrupt is used for "double buffering". When this interrupt occurs it means that the LCD controller has latched DMA Start Address 1and 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<br>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 |

### Horizontal Timing Register

See Figure 29 and Figure 30 for a graphical description of the LCD timing parameters.

**lcd_horztiming - LCD Horizontal Timing** Offset = 0x000C



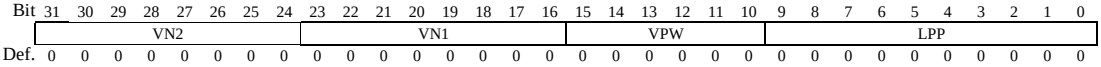

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | HN2 | Horizontal Non Display Period 2 (in pixels)<br>Value programmed is one pixel less than actual value. | R/W | 0 |
| 23:16 | HN1 | Horizontal Non Display Period 1 (in pixels)<br>Value programmed is one pixel less than actual value. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 15:10 | HPW | Horizontal Sync Pulse Width (in pixels)<br>Value programmed is one pixel less than actual value. | R/W | 0 |
| 9:0 | PPL | Pixels Per Line (in pixels)<br>Value programmed is one pixel less than actual value. | R/W | 0 |

**Vertical Timing Register**

See Figure 29 and Figure 30 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_verttiming- LCD Vertical Timing** Offset = 0x0010

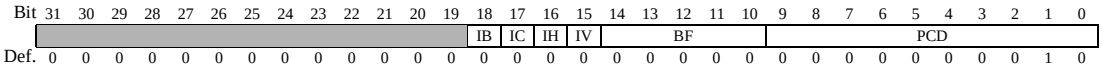



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | VN2 | Vertical Non Display Period 2 (in lines)<br>Value programmed is one line less than actual value. This parameter is not used with STN panels. | R/W | 0 |
| 23:16 | VN1 | Vertical Non Display Period 1 (in lines)<br>Value programmed is one line less than actual value. | R/W | 0 |
| 15:10 | VPW | Vertical Sync Pulse Width (in lines)<br>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)<br>Value programmed is one line less than actual value. | R/W | 0 |

### LCD Clock Control Register

The LCD Clock Control Register defines the parameters associated with the LCD pins.

**lcd_clkcontrol - LCD Clock Control** Offset = 0x0014

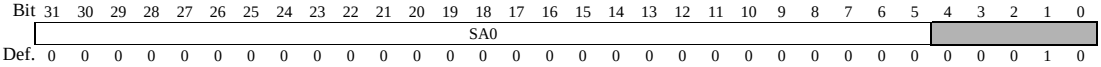



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:19 | RES | Reserved | R | 0 |
| 18 | IB | Invert Bias | R/W | 0 |
| 17 | IC | Invert Pixel Clock<br>0: Data is launched on the posedge of pixel clock<br>1: Data ia launched on the negedge of pixel clock | R/W | 0 |
| 16 | IH | Invert Line Clock | R/W | 0 |
| 15 | IV | Invert Frame Clock | R/W | 0 |
| 14:10 | BF | BIAS Signal Frequency<br>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 |
| 9:0 | PCD | Pixel Clock Divisor<br>Determines the pixel clock frequency from on chip programmable pll clock where:<br>Fpcd = LCD Clock Generator/ (2 * (PCD+1)).<br>For STN mono 8-bit panels this value must be greater than 2. For STN mono 4-bit panels this value must be greater than zero.<br>For STN color panels this value must be greater than zero. | R/W | 0 |

### 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

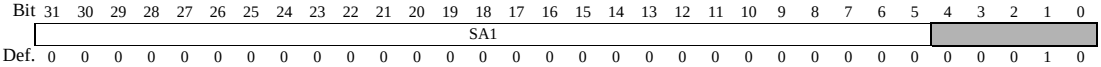



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:5 | SA0 | Frame Buffer Start Address 0<br>This is a physical address and must be cache line aligned. | R/W | 0 |
| 4:0 | RES | These bits are reserved and must be set to 0. | R/W | 0 |

### LCD DMA Start Address 1Register

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

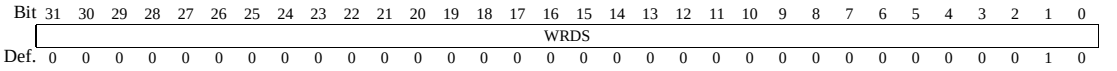



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | SA1 | Frame Buffer Start Address 1<br>This is a physical address and must be cache line aligned. | R/W | 0 |
| 4:0 | RES | These bits are reserved and must be set to 0. | R/W | 0 |

### Frame Buffer Words Register

This register represents the number of words in the frame buffer.

**lcd_words - Frame Buffer Words** Offset = 0x0020

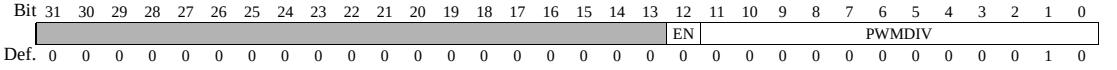



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | WRDS | Frame Buffer Words<br>90 & 270 degree swivel: Words per frame buffer line. The value programmed is 1 less than the actual value.<br>0 & 180 degreee swivel: Words in entire frame buffer. In 180 degree swivel this value must be evenly divisable by 8. The value programmed is 1 less than the actual value. | R/W | 0 |

### Pulse Width Modulation Frequency Divider

This register controls the frequency of the 2 PWM clocks.

**lcd_pwmdiv - PWM Frequency Divider** Offset = 0x0024

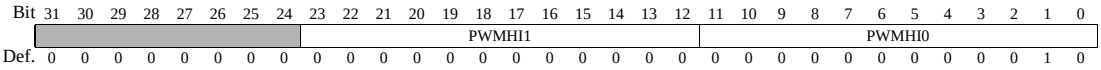



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:11 | RES | Reseved | R | 0 |
| 12 | EN | Enable<br>1 : Enable PWM clocks<br>0 : Disable PWM clocks | R/W | 0 |
| 11:0 | PWMDIV | PWM Frequency Divider | R/W | 0 |

### Pulse Width Modulation High Time

This register controls the duty cycle of the 2 PWM clocks.

**lcd_pwmhi - PWM High Time** Offset = 0x0028

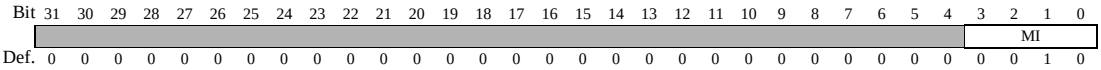



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | RES | Reseved | R | 0 |
| 23:12 | PWMHI1 | PWM High time for clock 1 | R/W | 0 |
| 11:0 | PWMHI0 | PWM High time for clock 0 | R/W | 0 |

### LCD Pallette Interface Registers

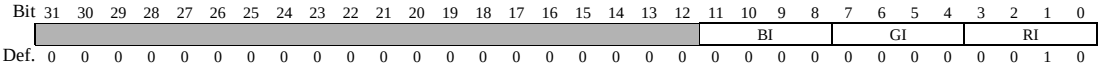
The 256 color pallette entries in the controller are read and written through the following 16 bit pallette interface registers mapped to offset range 0x0400 - 0x4FF. All register must be accessed as words.

**lcd_pallettebase MONOCHROME MODE** Offset Mapped = 0x0400 - 0x04FF




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:4 | RES | Reserved | R | 0 |
| 3:0 | MI | Monochromatic Panel Intensity | R/W | 0 |

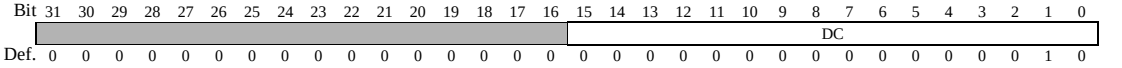
**lcd_pallettebase COLOR STN MODE** Offset Mapped = 0x0400 - 0x04FF



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:12 | RES | Reserved | R | 0 |
| 11:8 | RI | Red Channel Intensity* | R/W | 0 |
| 7:4 | GI | Green Channel Intensity | R/W | 0 |
| 3:0 | BI | Blue Channel Intensity* | R/W | 0 |

*Note: these values are swapped when lcd_control[CCO] bit is set.

**lcd_pallettebase COLOR TFT PALLETIZED** Offset Mapped = 0x0400 - 0x04FF




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | 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 68.

### TABLE 68. LCD Controller Signals

| Pin Name | 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 |

### TABLE 68. LCD Controller Signals

| Pin Name | Input/ Output | Definition |
|---|---|---|
| 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 69.

### TABLE 69. LCD Controller Data Pin Usage

| LCD Pin Name | Mono STN Panel | | Color STN Panel | | Color TFT Panel | |
|---|---|---|---|---|---|---|
| | 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: For TFT panels the R and B pin will be reversed if CCO bit is set.

## 6.9.3 Programming Considerations

### 6.9.3.1 Enabling the LCD Controller

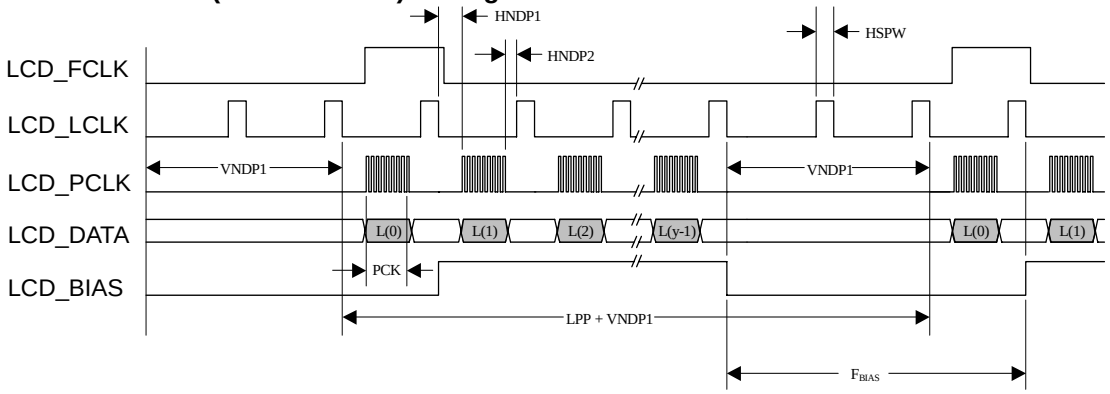
The first step in enabling the LCD controller is to program the LCD clock generator to the desired frequency.

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 the controller.

### 6.9.3.2 Definition Of Timing Parameters

The timing diagrams shown in figure Table 29 and figure Table 30 show the definitions of the timing registers plus an example for each mode.

**FIGURE 29.STN (Passive Mode) Timing**



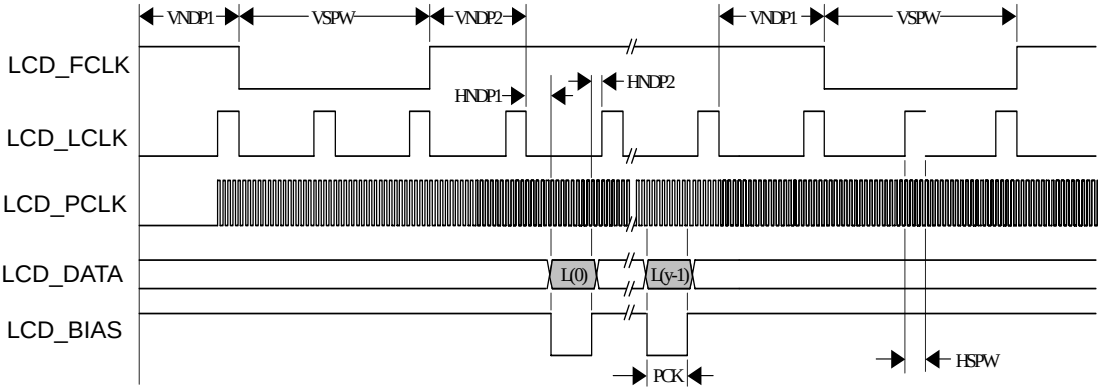

**Notes:**

$PCK_{color}$ = (PixelsPerLine*3)/DataBusWidth
$PCK_{mono}$ = (PixelsPerLine)/DataBusWidth
$F_{PCLK}$ = (LCD Clock Generator*2)/(PCD+1)

**In this diagram**:

$F_{BIAS}$ = 3 (**lcd_clkcontrol**[BF] = ‘b00010)
HNDP1 = 4 (**lcd_horztiming**[HN1] = ‘b00000011)
HNDP2 = 2 (**lcd_horztiming**[HN2] = ‘b00000001)
HSPW = 3 (**lcd_horztiming**[HPW] = ‘b000010)
VNDP1 = 2 (**lcd_verttiming**[VN1] = ‘b000001)
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**[IV:IH:IC] = ‘b000

**FIGURE 30.TFT (Active Mode) Timing**




**Notes:** PCK = PixelsPerLine

$F_{PCLK}$ = (LCD Clock Generator*2)/(PCD+1)

**In this diagram**: $F_{BIAS}$ = NA (**lcd_clkcontrol**[BF] = ‘bXXXXX)

HNDP1 = 5 (**lcd_horztiming**[HN1] = ‘b00000100)

HNDP2 = 2 (**lcd_horztiming**[HN2] = ‘b00000001)

HSPW = 4 (**lcd_horztiming**[HPW] = ‘b000011)

VNDP1 = 1 (**lcd_verttiming**[VN1] = ‘b00000000)

VNDP2 = 1 (**lcd_verttiming**[VN2] = ‘b00000000)

VSPW = 2 (**lcd_verttiming**[VPW] = ‘b000001)

LPP = y

FRAME transitions at the same time as LCLK goes inactive

FRAME, LCLK, PCLK, and BIAS shown here with **lcd_clkcontrol**[IV:IH:IC:IO] = ‘b1001

## 6.10 Secure Digital Controller

The Au1100 has two Secure Digital Controllers which incorporate both SD and SDIO interfaces.

### 6.10.1 SD Registers

The SD controller has two block IDs (ID=0 and ID=1). Each are controlled by a register block whose physical base address is shown in Table 70.

**TABLE 70. SD Base Address**

| Name | Physical Base Address | KSEG1 Base Address |
|---|---|---|
| sd0_base | 0x0 1060 0000 | 0xB060 0000 |
| sd1_base | 0x0 1068 0000 | 0xB068 0000 |

The register block consists of 5 registers as shown in Table 71.

**TABLE 71. SD Registers**

| Offset | 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 |

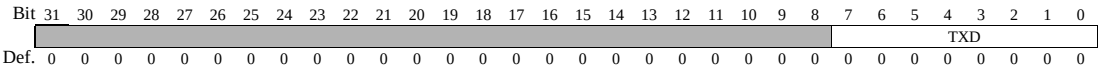
**TABLE 71. SD Registers**

| Offset | Register Name | Description |
|---|---|---|
| 0x0030 | sd_resp1 | SD Report Response 1 |
| 0x0034 | sd_resp0 | SD Report Response 0 |
| 0x0038 | sd_timeout | SD NAC Timeout Value |

### SD Transmit Data Port Register

The Transmit Data Port Register is used to send data to the SD interface for either PIO or DMA write modes.

**sd_txport - SD Transmit Data Port** Offset = 0x0000

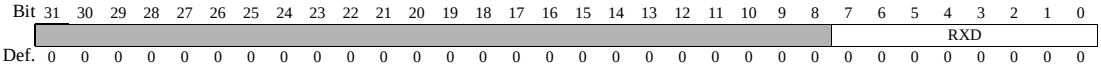



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:8 | RES | Reserved | R | 0 |
| 7:0 | TXD | Transmit Data | W | 0 |

### SD Receive Data Port Register

The Receive Data Port Register is used to read data from the SD interface from either PIO or DMA read modes.

**sd_rxport - SD Receive Data Port** Offset = 0x0004

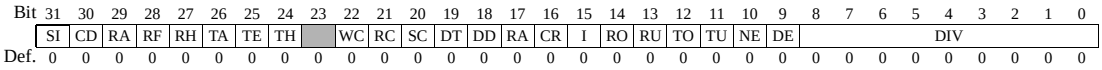



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:8 | RES | Reserved | R | 0 |
| 7:0 | RXD | Received Data | R | 0 |

### SD Configuration Register

This register is used to program interrupts and configure SD clocks.

**sd_config - SD Configuration** Offset = 0x0008



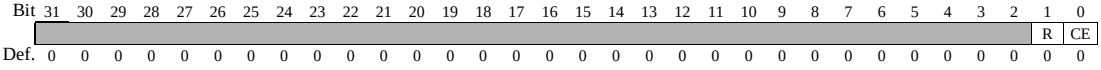

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | SI | Slot 0 device insertion interrupt enable | R/W | 0 |
| 30 | CD | Slot 0 card detect interrupt enable | R/W | 0 |
| 29 | RA | RX buffer almost full interrupt enable | R/W | 0 |
| 28 | RF | RX buffer full interrupt 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 | RES | Reserved. Must be written with zero. | R/W | 0 |
| 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 | RA | Command-response response access time-out interrupt enable (NCR) | R/W | 0 |
| 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 | 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 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 9 | DE | Divider enable:<br>0 - divider maximum count not changed<br>1 - divider maximum count changed to value written in bits [8:0]. | R/W | 0 |
| 8:0 | DIV | Number of source clock cycles in one phase of derived clock is DIV[8:0] + 1. For example: if DIV[8:0] = 0x000, clock is divided by 2; if DIV[8:0] = 0x1FF, clock is divided by 1024.<br>Note: this value must be written simultaneously with DE to take effect. | R/W | 0 |

### SD Enable Register

The SD Enable register contains bits necessary to enable clocks to and reset the SD interface.

**sd_enable - SD Enable** Offset = 0x000C

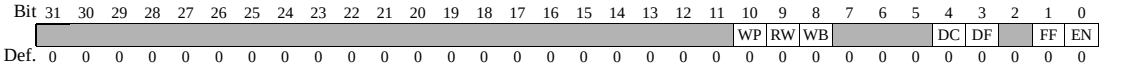



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:2 | RES | Reserved. | R | 0 |
| 1 | R | Peripheral reset. Clearing this bit will reset the entire peripheral. After enabling the clock using bit CE, this bit should be set for normal operation. | R/W | 0 |
| 0 | CE | Peripheral clock enable. This bit is set for normal operation and my be cleared to disable the clock for low power mode. | R/W | 0 |

### SD Configuration 2 Register

The SD Configuration 2 register is used to set up the PIO or DMA mode and state machine master enable.

**sd_config2 - SD Configuration 2** Offset = 0x0010

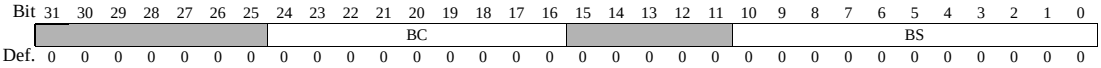



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:11 | RES | Reserved | R | 0 |
| 10 | WP | Write protect enable. This bit is set by software to prevent writing to the card. | R/W | 0 |
| 9 | RW | Read wait enable. When this bit is set serial clock-based flow control is not used. This bit is valid for SDIO mode only. | R/W | 0 |
| 8 | WB | Wide bus transfer mode:<br>0 - one wire data transfer<br>1 - four wire data transfer | R/W | 0 |
| 7:5 | RES | Reserved | R | 0 |
| 4 | DC | Disable hardware timeout down counter.<br>0 - normal hardware timeout<br>1 - no hardware timeout (software timeout) | R/W | 0 |
| 3 | DF | Disable clock freezing for flow control<br>0 - enable clock freezing<br>1 - disable clock freezing | R/W | 0 |
| 2 | RES | Reserved. This bit must be written with 0. | 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 | R/W | 0 |

### SD Block Size Register

The SD Block Size register defines the size and number of blocks to be transmitted by the SD controller..

**sd_blksize - SD Block Size** Offset = 0x0014

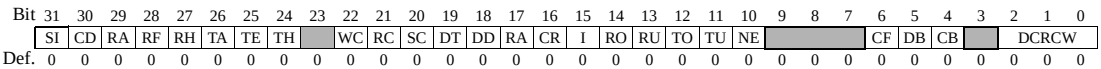



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:25 | RES | Reserved | R | 0 |
| 24:16 | BC | Block I/O count. This field is used for a known number of SDIO read/write blocks. | R/W | 0 |
| 15:11 | RES | Reserved | R | 0 |
| 10:0 | BS | Block size in bytes. The value programmed is one less than the actual number of bytes in the block. | R/W | 0 |

### SD Status Register

The SD Status register 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 interrupt bit a '1' must be written to the desired bit. This register also contains the CRC status word resulting from a block write. Note: these bits are not masked by the corresponding **sd_config** bits.

**sd_status - SD Status** Offset = 0x0018




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | SI | Slot 0 device insertion interrupt (LT) | R/W | 0 |
| 30 | CD | Slot 0 card detect interrupt (LT) | R/W | 0 |
| 29 | RA | RX buffer almost full interrupt (LT) | R/W | 0 |
| 28 | RF | RX buffer full interrupt (LT) | R/W | 0 |
| 27 | RH | RX buffer at least half full interrupt (LT) | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 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 | RES | Reserved | R/W | 0 |
| 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 enable | R/W | 0 |
| 17 | RA | Command-response response access time-out interrupt enable (NCR) | R/W | 0 |
| 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 pending | R | 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 |
| 9:7 | RES | These bits are reserved and should be written with zero. | R/W | 0x0 |
| 6 | CF | Clock freezing status:<br>1 - normal clocking<br>0 - clock frozen (potential overrun/underrun) | R | 0 |
| 5 | DB | SD data-response busy status | R | 0 |
| 4 | CB | SD command-response busy status | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 3 | RES | This bit is reserved and should be written with zero. | R/W | 0 |
| 2:0 | DCRCW | Device CRC status word:<br>010 - no error<br>101 - transmission error<br>111 - no CRC response | R | 0x0 |

### SD Debug Register

The SD Debug register is used primarily for debugging the read and write pointers for both transmit and receive FIFOs..

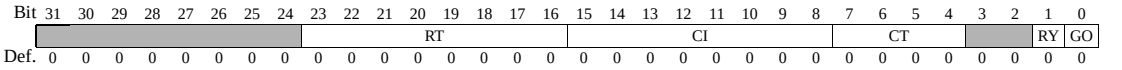
**sd_debug - SD Debug** Offset = 0x001C


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:15 | RES | Reserved. | R | 0 |
| 14:12 | RXR | Receive FIFO read pointer. | R | 0 |
| 11 | RES | Reserved. | R | 0 |
| 10:8 | RXW | Receive FIFO write pointer. | R | 0 |
| 7 | RES | Reserved. | R | 0 |
| 6:4 | TXR | Transmit FIFO read pointer. | R | 0 |
| 3 | RES | Reserved. | R | 0 |
| 2:0 | TXW | Transmit FIFO write pointer. | R | 0 |

### SD Command Register

The SD Command register contains fields used to build an SD command sequence.

**sd_cmd - SD Command** Offset = 0x0020




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 23:16 | RT | Response Type<br>0000 - no response<br>0001 - R1 response (48 bits)<br>0010 - R2 response (136 bits)<br>0011 - R3 response (48 bits)<br>0110 - R6 resposne (48 bits)<br>1001 - R1b response (48 bits)<br>All other values are reserved. | R/W | 0 |
| 15:8 | CI | Command Index | R/W | 0 |
| 7:4 | CT | Command Type - see Table 33 for valid encoding and descriptions. | R/W | 0 |
| 3:2 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 1 | RY | Response Ready.<br>This bit is set by the SD block once the command-response sequence is finished and reset once **sd_resp0** is read. | R | 0 |
| 0 | GO | Command Go/Busy<br>This bit is set to initiate a command. The bit is reset once the last bit of the command argument is transmitted. | R/W | 0 |

**TABLE 72.** Command Type Field Encodings

| CT[3:0] | Action applied to SD Memory | Action applied to SDIO |
|---|---|---|
| 0000b | 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. |
| 0001b | 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field is ignored. |
| 0010b | 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field is ignored. |
| 0011b | 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 **sd_blksize**[BS] field. **sd_blksize**[BC] field 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field 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). |
| 0100b | 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field 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 into **sd_blksize**[BS] field. **sd_blksize**[BC] field 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). |

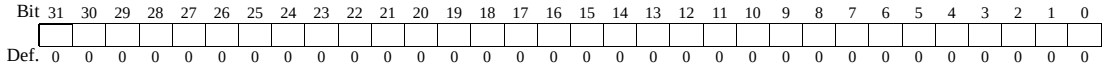
**TABLE 72.** Command Type Field Encodings

| CT[3:0] | Action applied to SD Memory | Action applied to SDIO |
|---|---|---|
| 0101b | 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 into the **sd_blksize**[BS] field. Based on the Byte/Block Count, the **sd_blksize**[BC] field 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. |
| 0110b | 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 into the **sd_blksize**[BS] field. Based on the Byte/Block Count, the **sd_blksize**[BC] field 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. |
| 0111b | 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. |
| 1000b | 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). |
| 1001b to 1111b | Reserved. | Reserved. |

### SD Command Argument Register

The SD Command Argument register holds the argument used in an SD command-response sequence..

**sd_cmdarg - SD Command Argument** Offset = 0x0024

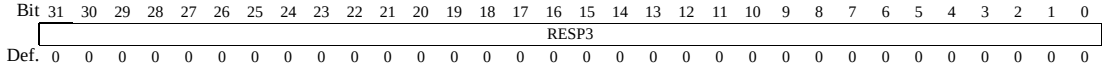



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | CARG | Command argument. Must write this register first, then write **sd_cmd**[BY] to issue the command. | R/W | 0 |

### SD Response 3 Register

The SD Response 3 register contains the response from an issued command-response sequence. Valid only when the response is of type 128 bit..

**sd_resp3 - SD Response 3** Offset = 0x0028

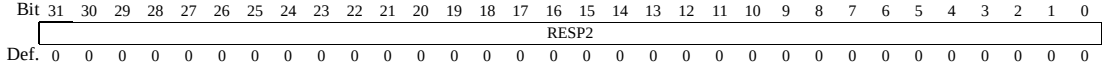



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | RESP3 | Response from device. | R | 0 |

### SD Response 2 Register

The SD Response 2 register contains the response from an issued command-response sequence. Valid only when the response is of type 128 bit..

**sd_resp2 - SD Response 2** Offset = 0x002C

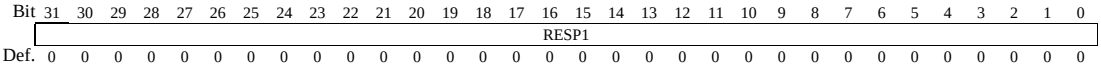


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | RESP2 | Response from device. | R | 0 |

### SD Response 1 Register

The SD Response 1 register contains the response from an issued command-response sequence. Valid only when response size is of type 128 bits or (6 + 32) bits

**sd_resp1 - SD Response 1** Offset = 0x0030

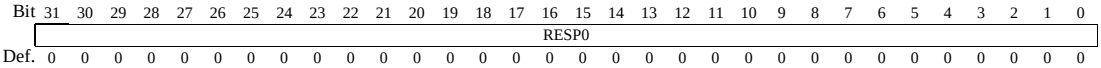



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | RESP1 | Response from device. | R | 0 |

### SD Response 0 Register

The SD Response 0 register contains the response from an issued command-response sequence.Valid for all modes where a response is expected.

**sd_resp0 - SD Response 0** Offset = 0x0034

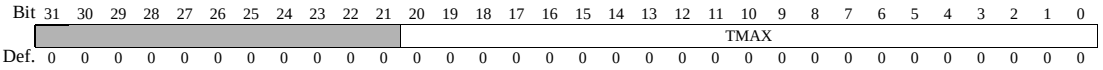



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | RESP0 | Response from device. | R | 0 |

**SD Timeout Register**

The SD Timeout register defines the timeout value for NAC.

**sd_timeout - SD Timeout** Offset = 0x0038




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:21 | RES | Reserved. | R | 0 |
| 20:0 | TMAX | Maximum timeout value for NAC where:<br>NAC = TAAC + NSAC (see SD specification)<br>Maximum timeout value is 81.02 ms. Counter is based upon 25 MHz clock. | R/W | 0 |

## 6.10.2 Hardware Considerations

The SD interface consists of the signals listed in Table 73.

**TABLE 73. SD Signals**

| Pin Name | 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 |

## 6.10.3 Programming Considerations

TBD.

## 6.11 Secondary General Purpose I/O

The Au1100 contains two seperate GPIO sections. This section describes the secondary GPIO block which corresponds to pins labeled GP200 through GP215. For a description of the primary GPIO block refer to Section 7.3, "Primary General Purpose I/O" in the system control block description.

## 6.12 Programming Model

The secondary GPIO block is controlled by a register block referenced from the base address described in Table 74.

### TABLE 74. GPIO2 Register Base Addresses

| Name | Physical Base Address | KSEG1 Base Address |
|---|---|---|
| gpio2_base | 0x0 1170 0000 | 0xB170 0000 |

## 6.12.1 GPIO2 Registers

Each register block contains the registers listed in Table 75.

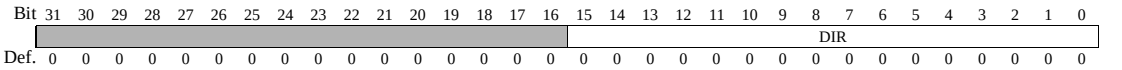
### TABLE 75. GPIO2 Registers

| Offset | 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 |
| 0x0014 | gpio2_enable | GPIO2 Enable |

#### 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

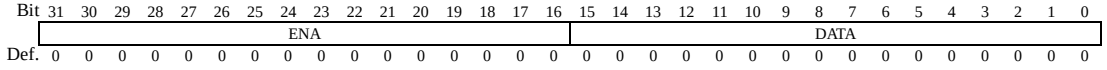


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | DIR | Direction Control. Each bit controls the I/O direction of one GPIO in the secondary block. Bits 15:0 correspond to GP[215..200].<br>0 = pin is an input (output disabled)<br>1 = pin is an output | R/W | 0 |

### Data Output Register

The **gpio2_output** register controls the output data for the secondary GPIOs. Data bits 15:0 will be output to the corresponding GPIO when the enable bit is set for that bit during a write to this register

**gpio2_output - Data Output Register** Offset = 0x0008

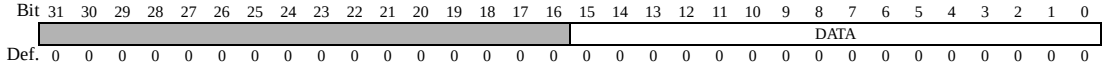



| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | ENA | Data Output Enable. Bits [31:16] correspond to GP[215:200].<br>1 - Data output is enabled<br>0 - Data output is disabled | R/W | 0 |
| 15:0 | DATA | Output Data. Bits 15:0 correspond to GP[215:200]. | R/W | 0 |

### Pin State Register

The **gpio2_pinstate** register reflects the current state of the corresponding secondary GPIO pin.

**gpio2_pinstate - Pin State** Offset = 0x000c

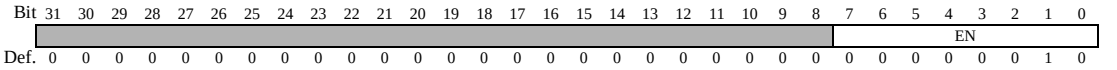


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and will be read 0. | R | 0 |
| 15:0 | DATA | Current Pin State for GP[215:200] | R | 0 |

### Interrupt Enable Register

The **gpio2_inten** register contains bits which enable interrupts under certain operational conditions.

**gpio2_inten - Interrupt Enable Register** Offset = 0x0010




| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:8 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 7:0 | EN | Interrupt Enable Bits [7:0] correspond to GP[215:208] | R/W | 0 |

### Enable Register

The **gpio2_enable** register controls the clocks and reset to the secondary GPIO block.

**gpio2_enable - Enable Register** Offset = 0x0010


| Bit(s) | Name | Description | R/W | Default |
|---|---|---|---|---|
| 31:2 | RES | These bits are reserved and should be written a 0. | 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 |

# 7 System Control

The Au1100 contains a robust system control strategy that includes the means to control the following:

- Clocking
- Time of Year and Real Time Clock counters
- GPIO control
- Power management

All registers in the system control block are located off of the base address shown in Table 76.

### TABLE 76. 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 Chapter 8, Powerup, Reset and Boot 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 supports two oscillator inputs, 12MHz and 32.768kHz. This section documents the clock domains driven directly and indirectly by the 12MHz input. Please see Section 11.5, Crystal Specifications for the Crystal specification.

The Au1100 contains 2 PLLs driven by the 12MHz input and a clocking block from which the following are derived:

- CPU Clock
- Core Cycle Counter register clocked by the CPU clock
- System Bus clock
- Peripheral Bus Clock
- SDRAM Bus Clock
- Programmable frequency clocks needed by certain peripherals
- Programmable frequency clocks for external use (provided on pins shared with GPIO2 and GPIO3)

The 32.768kHz clock input drives the 2 programmable counters intended for use as a Real Time Clock and Time of Year Clock. The programmable counters are documented in Section 7.2.

Figure 31 shows the basic clocking topology and the relationship between the CPU Clock, the System bus clock and the Peripheral Clock. As shown the System bus frequency is derived by dividing the CPU Clock by the value SD. SD is a bit field in the **sys_powerctrl** register. This register is documented in Section 7.4. The Peripheral Bus clock and the SDRAM bus are fixed at the System Bus frequency divided by 2. The blocks driven by the Peripheral Bus clock and the System Bus clock are shown in Figure 32.

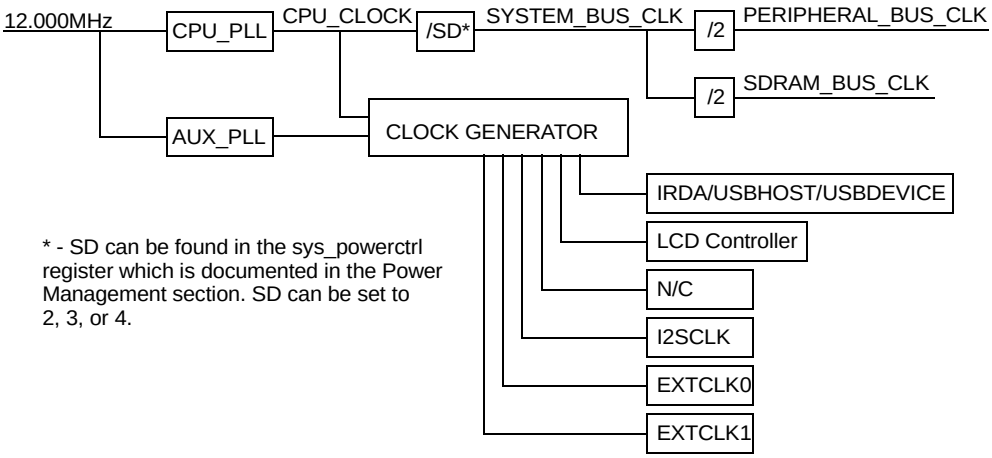



**FIGURE 31. Clocking Topology**

## 7.1.1 Clock Register Descriptions

The clock manager registers and their associated offsets are listed in Table 77.

**TABLE 77. Clock Generation Registers**

| Offset | Register Name | Description | Reset Type |
|---|---|---|---|
| 0x0020 | sys_freqctrl0 | Controls frequency generator 0 through 2 divider and enable | Hardware |
| 0x0024 | sys_freqctrl1 | Controls frequency generator 3 through 5 divider and enable | Hardware |
| 0x0028 | sys_clksrc | Controls source of the 6 derived clocks | Hardware |
| 0x0060 | sys_cpupll | Changes CPU PLL frequency | Hardware |
| 0x0064 | sys_auxpll | Changes Auxiliary PLL frequency | Hardware & Runtime |

## 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 32 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 33 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 then the peripheral will not operate correctly).

The USB Host Clocks, USB Device Clock and IrDA Clock must be programmed to 48MHz. 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" 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 I2S 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).

EXTCLK0 shares a pin with GPIO2. If EXTCLK0 is to be used the ***EX0*** bit in the **sys_pinfunc** register must be set to allow the clock to drive this pin. In addition the ***CS*** bit in the **sys_pinfunc** register must be cleared.

EXTCLK1 shares a pin with GPIO3. If EXTCLK1 is to be used the ***EX1*** bit in the **sys_pinfunc** register must be set to allow the clock to drive this pin.

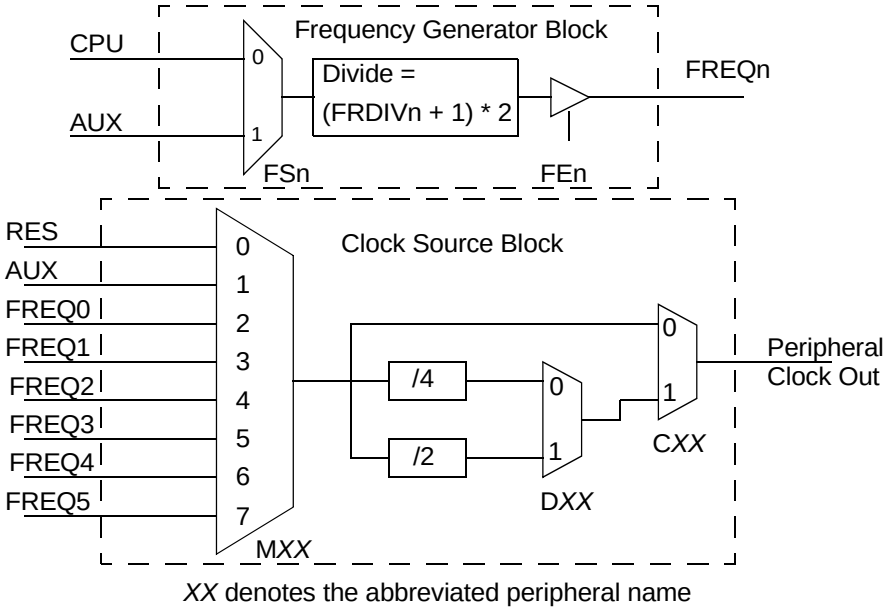



**FIGURE 32. Frequency Generator and Clock Source Block Diagram**

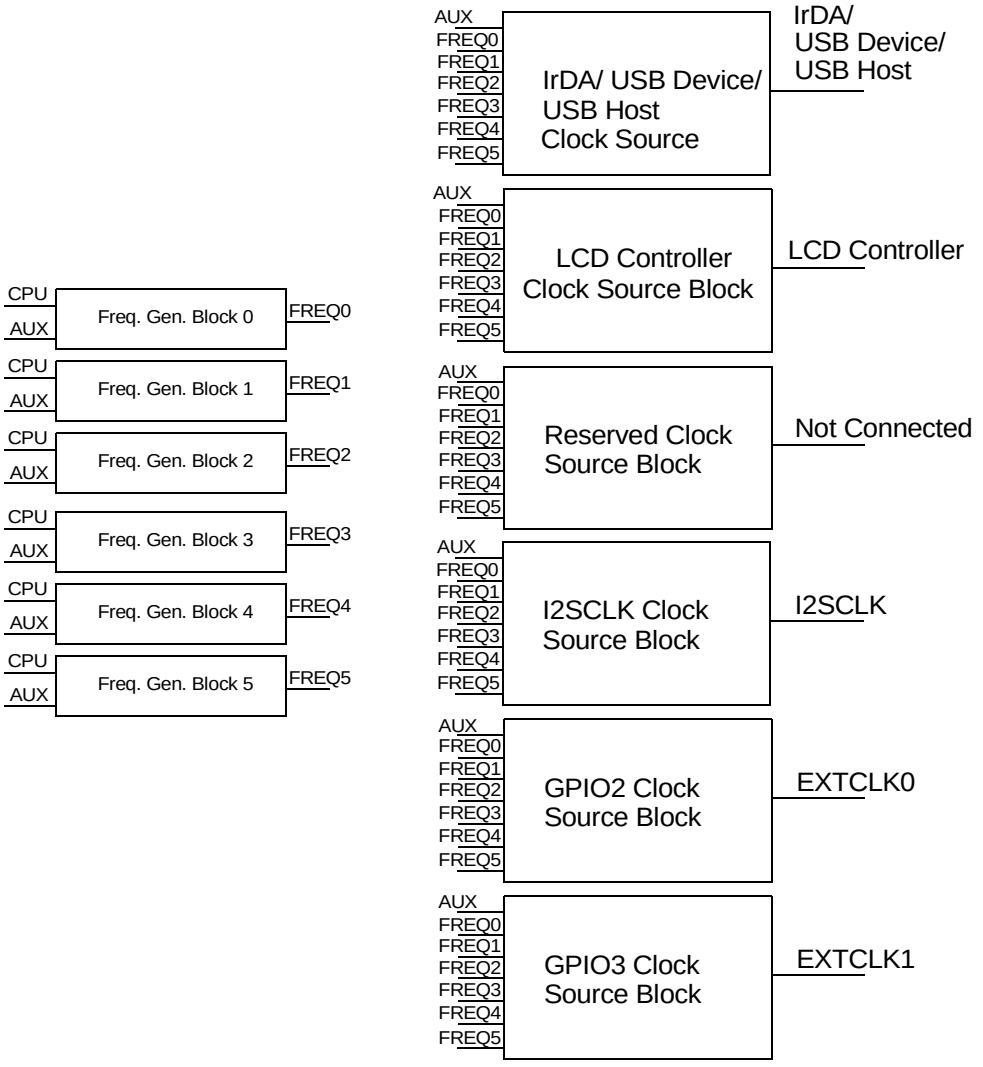

CPU
AUX
Freq. Gen. Block 0
FREQ0
CPU
AUX
Freq. Gen. Block 1
FREQ1
CPU
AUX
Freq. Gen. Block 2
FREQ2
CPU
AUX
Freq. Gen. Block 3
FREQ3
CPU
AUX
Freq. Gen. Block 4
FREQ4
CPU
AUX
Freq. Gen. Block 5
FREQ5
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
IrDA/ USB Device/ USB Host Clock Source
IrDA/ USB Device/ USB Host
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
LCD Controller Clock Source Block
LCD Controller
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
Reserved Clock Source Block
Not Connected
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
I2SCLK Clock Source Block
I2SCLK
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
GPIO2 Clock Source Block
EXTCLK0
AUX
FREQ0
FREQ1
FREQ2
FREQ3
FREQ4
FREQ5
GPIO3 Clock Source Block
EXTCLK1

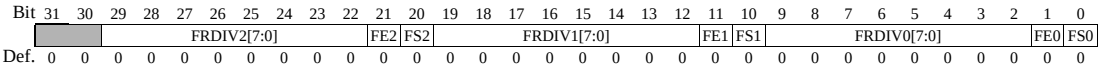
**FIGURE 33. Frequency Generator and Clock Source Mapping**

### 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




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:30 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 29:22 | FRDIV2 | Frequency Divider 2<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 21 | FE2 | Frequency Generator Output Enable 2<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 20 | FS2 | Frequency Generator Source 2<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |
| 19:12 | FRDIV1 | Frequency Divider 1<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 11 | FE1 | Frequency Generator Output Enable 1<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 10 | FS1 | Frequency Generator Source 1<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |

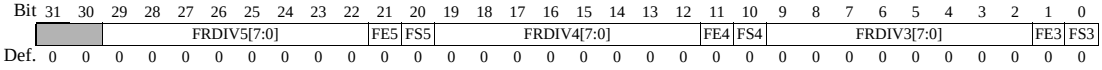
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 9:2 | FRDIV0 | Frequency Divider 0<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 1 | FE0 | Frequency Generator Output Enable 0<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 0 | FS0 | Frequency Generator Source 0<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |

**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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:30 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 29:22 | FRDIV5 | Frequency Divider<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 21 | FE5 | Frequency Generator Output Enable<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 20 | FS5 | Frequency Generator Source<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |

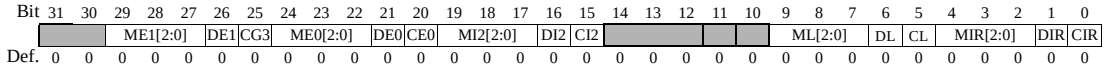
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 19:12 | FRDIV4 | Frequency Divider<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 11 | FE4 | Frequency Generator Output Enable<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 10 | FS4 | Frequency Generator Source<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |
| 9:2 | FRDIV3 | Frequency Divider<br>These bits set the frequency divider.<br>The actual divide value is (FRDIV + 1) * 2. | R/W | 0 |
| 1 | FE3 | Frequency Generator Output Enable<br>0 - Disable output<br>1 - Enable output | R/W | 0 |
| 0 | FS3 | Frequency Generator Source<br>0 - CPU Core clock<br>1 - Auxiliary Clock Input | R/W | 0 |

### Clock Source Control

This register controls the clock source for all 6 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:30 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 29:27 | ME1 | EXTCLK1 Clock Mux input select<br>See Table 78. | R/W | $000_b$ |
| 26 | DE1 | EXTCLK1 Clock Divider Select<br>0 - divide by 4<br>1 - divide by 2 | R/W | 0 |
| 25 | CE1 | EXTCLK1 Clock Select<br>0 - Clock is taken directly from mux (DIVSEL has no effect for this choice).<br>1 - Clock is taken from 2/4 divider. | R/W | 0 |
| 24:22 | ME0 | EXTCLK0 Clock Mux input select<br>See Table 78. | R/W | $000_b$ |
| 21 | DE0 | EXTCLK0 Clock Divider Select<br>0 - divide by 4<br>1 - divide by 2 | R/W | 0 |
| 20 | CE0 | EXTCLK0 Clock Select<br>0 - Clock is taken directly from mux (DIVSEL has no effect for this choice).<br>1 - Clock is taken from 2/4 divider. | R/W | 0 |
| 19:17 | MI2 | I2S Clock Mux input select<br>See Table 78. | R/W | $000_b$ |
| 16 | DI2 | I2S Clock Divider Select<br>0 - divide by 4<br>1 - divide by 2 | R/W | 0 |
| 15 | CI2 | I2S Clock Select<br>0 - Clock is taken directly from mux (DIVSEL has no effect for this choice).<br>1 - clock is taken from 2/4 divider | R/W | 0 |
| 14:10 | RES | Reserved | | |
| 9:7 | ML | LCD Controller Clock Mux input select<br>See Table 78. | R/W | $000_b$ |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 6 | DL | LCD Controller Clock Divider Select<br>0 - Divide by 4.<br>1 - Divide by 2. | R/W | 0 |
| 5 | CL | LCD Controller Clock Select<br>0 - Clock is taken directly from mux (DIVSEL has no effect for this choice).<br>1 - Clock is taken from 2/4 divider. | R/W | 0 |
| 4:2 | MIR | IrDA/ USB Host/ USB Device Clock Mux input select<br>See Table 78. | R/W | $000_b$ |
| 1 | DIR | IrDA/ USB Host/ USB Device Clock Divider Select<br>0 - Divide by 4.<br>1 - Divide by 2. | R/W | 0 |
| 0 | CIR | IrDA/ USB Host/ USB Device Clock Clock Select<br>0 - Clock is taken directly from mux (DIVSEL has no effect for this choice).<br>1 - Clock is taken from 2/4 divider. | R/W | 0 |

The specific values written to the Clock Mux Input Select field are shown in Table 78. The inputs to all CMn are shown in Figure 32. The FREQn selections come from the output of the corresponding frequency generators.

### TABLE 78. Clock Mux Input Select Values

| Value | Meaning |
|---|---|
| 000b | Reserved |
| 001b | Auxiliary Clock |
| 010b | FREQ0 |
| 011b | 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. Each PLL is independently programmable. Care should be taken when programming the registers that the limits of the Au1100 being used are observed. Although it is possible to program the PLL outside this frequency range, care should be taken so as not to violate the limits or undefined results could occur.

For higher frequencies the Au1100 core requires a higher core voltage, VDDI. Care should be taken that the system is providing the correct voltage for the operating frequency before changing the CPU. See Chapter 11, Electrical Specifications, for full information about the voltage /frequency requirements of the Au1100.

There is a Core Cycle Counter register located at CP0 register 9 that can be used to count core cycles. Please see Section 2.7, "Coprocessor 0", for more information.

The two PLLs in the Au1100 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 12MHz crystal on XTI and XTO:

- CPU Clock = 192MHz
- AUX Clock = Disabled
- System Bus Clock = 96MHz (SD  - System bus divider - defaults to 2)
- Peripheral Bus = 48MHz
- SDRAM bus = 48MHz.

When changing core frequency approximately 20 microseconds will pass while the clocks shut off and the PLL is re-synced with the new frequency. During this period no CPU execution will occur. Interrupts will not be serviced in this time. They will be serviced once execution begins at the new frequency.

#### CPU PLL Control

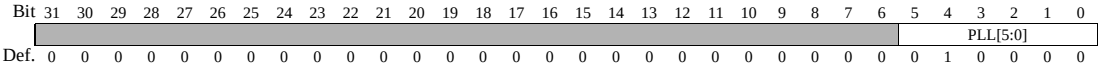
The CPU PLL will reset to its default value only at power up. After sleep, and during a run-time reset the **sys_cpupll** will retain its previous value.

It should be noted that certain values may produce a clock speed that is higher than allowable for the CPU and System Clock. Care should be taken to avoid this or undefined results may occur.

This register is read/write, however the value read is only valid 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 the **sys_cpupll** register is written any value, the system will automatically halt for 20uS to allow for the PLL to relock and clocks to become stable.

**sys_cpupll** Offset = 0x0060




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:6 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 5:0 | PLL | The PLL value determines the integer multiplier that is used to multiply the clock coming from the oscillator.<br>For example, with the default of 16 and a 12MHz OSC frequency, the CPU frequency will be 192MHz.<br>Only values from 16-46 are supported. All other values are reserved.<br>These values represent the limits of the CPU PLL and may place the actual CPU frequency outside the limits of the Au1100. | R/W | 0x10 |

**Auxiliary PLL Control**

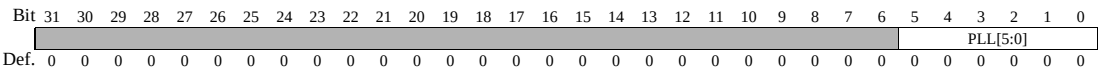
This AUX PLL will reset to its default value on hardware reset, after sleep, and during a runtime reset.

This register is read/write, however the value read is only valid after initialization.

For this reason it is suggested that this registers be initialized at hardware reset, runtime reset and sleep, even if with its default value.

Writing the sys_auxpll will not cause the system to halt and and clocks taken from the AUX PLL may be unstable for up to 20uS. To guarantee 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 20uS.

**sys_auxpll** Offset = 0x0064



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:6 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 5:0 | PLL | The PLL value determines the integer multiplier that is used to multiply the clock coming from the oscillator.<br>For example, with a value of 12 and a 12MHz OSC frequency, the CPU frequency will be 144MHz.<br>Only values from 8-46 and 0 are supported. All other values are reserved.<br>A value of 0 (default) disables the AUX PLL. | R/W | 0x00 |

### 7.1.4 Hardware Considerations

When using the external clocks from the clock generation block, the **pin_function** register must be programmed such that GPIO2 and/or GPIO3 are configured to be driven by EXTCLK0 and/or EXTCLK1.

Section 11, "Electrical Specifications", should be referenced for the crystal specifications.

### 7.1.5 Programming Considerations

When changing the CPU PLL value through the sys_cpupll register the system will automaticaly halt for 20uS to allow clocks to stabilize. During this time no interrupts will be serviced potentially affecting real time systems.

Writing the sys_auxpll will not cause the system to halt and and clocks taken from the AUX PLL may be unstable for up to 20uS. To guarantee 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 20uS.

## 7.2 Time of Year Clock and Real Time Clock

The Au1100 contains two programmable counters designed for use as a Time of Year Clock (TOY) and Real Time Clock (RTC). One counter will continue counting through sleep thus making it ideal for use as a TOY. The other counter will power down in sleep mode and can be used as an RTC.

Both counters are driven by a 32.768kHz clock input. Please see Section 11.5, "Crystal Specifications" for the crystal specifications.

Each programmable counter employs a register to initialize the counter or load a new value, a trim divider to adjust the incoming 32.768kHz clock and 3 match registers which have associated interrupts that will 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 34 shows a functional block diagram of both the TOY and the RTC (only one block diagram is shown that is applicable to each). The naming of the bits correspond to control bits presented in the next section.

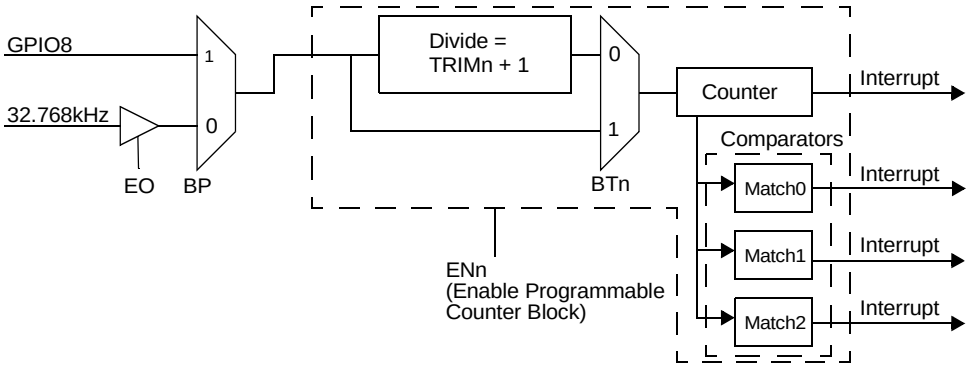



**FIGURE 34. TOY and RTC Block Diagram**

### 7.2.1 Time of Year Clock and Real Time Clock Registers

Each counter operates identically with the only difference being that the TOY will continue counting through sleep and the RTC will not.

The programmable counter control registers and their associated offsets are listed in Table 79. When functionality is identical for registers in the different programmable counters, only

one register description is presented with both offsets of the respective registers presented above the register.

### TABLE 79. Programmable Counter Registers

| Offset | Register Name | Description | Reset Type |
|---|---|---|---|
| 0x0000 | sys_toytrim | Trim value for 32.768kHz input to TOY | Hardware |
| 0x0004 | sys_toywrite | the 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 32.768kHz input to RTC | Hardware |
| 0x0048 | sys_rtcwrite | the 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 |

**Trim Register**

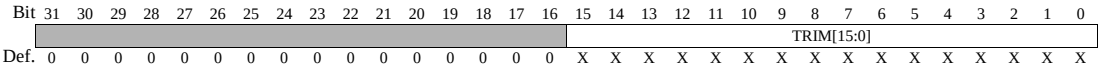
The TOY Trim Write Status (bit 4 in the sys_cntrctrl) must be clear before writing **sys_toytrim**. It will be set upon writing this register and cleared by hardware when the write takes effect.

The RTC Trim Write Status (bit 20 in the sys_cntrctrl) must be clear before writing **sys_rtctrim**. It will be set upon writing this register and cleared by hardware when the write takes effect.

This register is unpredictable at power on. During a runtime reset and during sleep this register will retain its value.

**sys_toytrim - TOY Trim** Offset = 0x0000

**sys_rtctrim - RTC Trim** Offset = 0x0044




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:16 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 15:0 | TRIM | Divide value for 32.768kHz input.<br>Divide = TRIM + 1 | R/W | UNPRED |

**Counter Write**

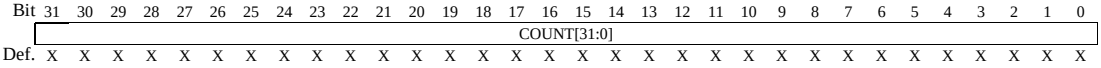
The TOY Value Write Status (bit 0 in the sys_cntrctrl) must be clear before writing **sys_toytrim**. It will be set upon writing this register and cleared by hardware when the write takes effect.

The RTC Value Write Status (bit 16 in the sys_cntrctrl) must be clear before writing **sys_rtcwrite**. It will be set upon writing this register and cleared by hardware when the write takes effect.

This register is unpredictable at power on. During a runtime reset and during sleep this register will retain its value.

**sys_toywrite - TOY counter value write** Offset = 0x0004

**sys_rtcwrite - RTC counter value write** Offset = 0x0048




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | COUNT | Counter Write<br>The respective counter will be updated with the value written to this register at the next trimmed clock. | W | UNPRED |

### Match Registers

The corresponding write status bit in the sys_cntrctrl register must be clear before writing the below registers. It will be set upon writing the register and cleared by hardware when the write takes effect.

Each match register is capable of causing an interrupt as shown in Chapter 5, Interrupt Controller. The **sys_toymatch2** can be used to wake up from sleep, see Section 7.2.2.

These registers are unpredictable at power on. During a runtime reset and during sleep these registers will retain their value.

**sys_toymatch0 - TOY Match 0** Offset = 0x0008

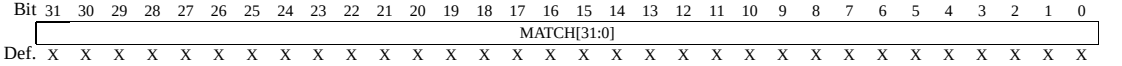
**sys_toymatch1 - TOY Match 1** Offset = 0x000C

**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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | MATCH | A match with the counter and the value in this register will cause an interrupt. | R/W | UNPRED |

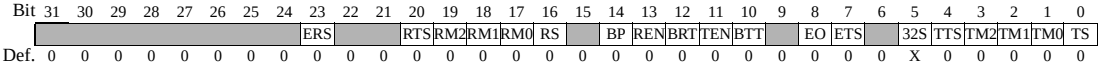
### TOY and RTC Counter Control

The TOY and RTC counter control register 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 to 1 indicating that there is a write pending. When this bit is reset to 0 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 will reset to defaults only on a hardware reset. During a runtime reset and during sleep this register will retain its value.

**sys_cntrctrl** Offset = 0x0014




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:24 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 23 | ERS | REN (bit 13) write status. | R | 0 |
| 22:21 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 20 | RTS | **sys_rtctrim** Write Status | R | 0 |
| 19 | RM2 | **sys_rtcmatch2** Write Status | R | 0 |
| 18 | RM1 | **sys_rtcmatch1** Write Status | R | 0 |
| 17 | RM0 | **sys_rtcmatch0** Write Status | R | 0 |
| 16 | RS | **sys_rtcwrite** Write Status | R | R |
| 15 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 14 | BP | Bypass the 32.768k OSC<br>0 - Select Oscillator Input<br>1 - GPIO8 will drive counters. This is a test mode where GPIO8 can drive the counters from an external source or through software using the GPIO controller. | R/W | 0 |
| 13 | REN | Enable RTC<br>0 - RTC is disabled<br>1 - RTC is enabled | R/W | 0 |
| 12 | BRT | Bypass RTC Trim<br>0 - normal operation<br>1 - The RTC is driven directly by the 32.768k clock, bypassing the trim. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 11 | TEN | Enable TOY<br>0 - TOY is disabled<br>1 - TOY is enabled | R/W | 0 |
| 10 | BTT | Bypass TOY Trim<br>0 - normal operation<br>1 - The TOY is driven directly by the 32.768k clock, bypassing the trim. | R/W | 0 |
| 9 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 8 | EO | Enable 32.768kHz Oscillator<br>0 - 32.768kHz osc. is disabled<br>1 - 32.768kHz osc. is enabled<br><br>The enable controls the OSC going to both the RTC and TOY.<br>The 32S (OSC status) should be polled until set after the OSC is enabled. | R/W | 0 |
| 7 | ETS | TEN (bit 11) write status | R | 0 |
| 6 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 5 | 32S | 32.768kHz Oscillator Status<br>0 - Oscillator is not running<br>1 - Oscillator is running | R | 0 |
| 4 | TTS | **sys_toytrim** Write Status | R | 0 |
| 3 | TM2 | **sys_toymatch2** Write Status | R | 0 |
| 2 | TM1 | **sys_toymatch1** Write Status | R | 0 |
| 1 | TM0 | **sys_toymatch0** Write Status | R | 0 |
| 0 | TS | **sys_toywrite** Write Status | 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 state bit in the Status register. When the Write Status bit is 0 it is okay to write

a new value. Once the new value is written to the register the State bit will change to a 1. When the State bit is 1 the new value is being updated in supporting hardware. When the State value changes to a 0 then the new value is active in the device.

# 7.3 Primary General Purpose I/O

This section covers the Primary General Purpose I/O (GPIO) pins of the Au1100 and also documents how to change the functionality of certain pins between being a GPIO or being part of a peripheral device.

The Au1100 supports 48 GPIOs, 32 of which are controlled by the Primary GPIO block. For a description of the Secondary GPIO block see Section 6.11, "Secondary General Purpose I/O". Each Primary 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. Please see Chapter 5, Interrupt Controller for information on interrupts.

## 7.3.1 Pin Functionality and Drive Strength

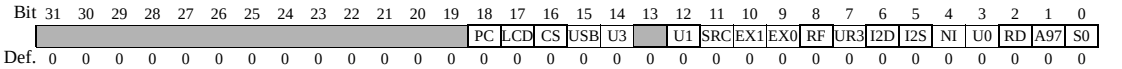
To maximize the functionality of the Au1100, many of the pins have multiple uses. It should be noted that if a pin is programmed for a certain use, any other functionality associated with that pin can not be utilized at the same time. In other words, a pin can not be used as a GPIO at the same time it is assigned to a peripheral device.

Chapter 10, Signal Description shows a block diagram and description of multiplexed signals.

### Pin Function

This register will reset to its default state at hardware reset, runtime reset and sleep.

**sys_pinfunc** Offset = 0x002C




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:19 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 18 | PC | PCMCIA/GP[207:204]<br>1 - PREG, PCE1, PCE2 and PWEN will drive pins.<br>0 - GP[207:204] will drive pins. | R/W | 1 |
| 17 | LCD | External LCD Controller/GP[203:200]<br>0 - LRD, LRD1, LWR0 and LWR1 will drive pins.<br>1 - GP[203:200] wil drive pins. | R/W | 1 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 16 | CS | Clock Select<br>0 - EXTCLK0 will be routed to the GP02 output<br>1 - 32kHz OSC clock will be routed to the GP02 output<br>This bit's functionality applies only when EX0 = 1 | R/W | 0 |
| 15 | USB | USB Functionality<br>0 - USBDP and USBDM will drive pins (pins are connected to USB device module).<br>1 - USBHP and USBHM will drive pins (pins are connected to USB root hub port 0). | R/W | 0 |
| 14 | U3 | UART3/GP214<br>0 - U3TXD drives pin.<br>1 - Pin is configured for GP214. | R/W | 1 |
| 13 | RES | This bit is reserved and should be written as 1. | R/W | 1 |
| 12 | U1 | UART1/GP213<br>0 - U1TXD drives pin.<br>1 - Pin is configured for GP213. | R/W | 1 |
| 11 | SRC | GP06/SMROMCKE<br>0 - Pin is configured for GP06.<br>1 - SMROMCKE drives pin. | R/W | 0 |
| 10 | EX1 | GP03/EXTCLK1<br>0 - Pin is configured for GP03.<br>1 - EXTCLK1 will drive pin. | R/W | 0 |
| 9 | EX0 | GP02 / (EXTCLK0 or 32kHz OSC)<br>0 - Pin is configured for GP02.<br>1 - EXTCLK0 or 32kHz OSC will drive pin.<br>Bit 16, CS, will select whether EXTCLK0 or the 32kHz OSC will drive pin | R/W | 0 |
| 8 | IRF | GP15/IRFIRSEL<br>0 - Pin is configured for GP15.<br>1 - IRFIRSEL will drive pin. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 7 | UR3 | GP[14:9]/UART3<br>0 - Pins are configured as GP[14:9].<br>1 - Pins are configured for UART3 flow control. U3DTR, U3RTS, U3RI, U3DCD, U3DSR, U3CTS will drive pins. | R/W | 0 |
| 6 | I2D | GP8/I2SDI<br>0 - Pin is configured for GP8.<br>1 - Pin is configured as I2SDI.<br>NOTE: The I2S port can operate without I2SDI if this input is not needed. | R/W | 0 |
| 5 | I2S | I2S/GP[29:31]<br>0 - Pins are configured for I2S mode. I2SWORD, I2SCLK, I2SDIO will drive pins.<br>1 - Pins are configured as GP[31:29]. | R/W | 1 |
| 4 | NI | MAC0/GPIO<br>0 - Pins are configured as Ethernet port 0. N0TXD[3:0], N0TXEN and N0MDC will drive port.<br>1 - Pins are configured as GP[28:24] and GP215. | R/W | 1 |
| 3 | U0 | UART0/GP212<br>0 - Pin is configured for U0TXD (necessary for UART0 operability).<br>1 - Pin is configured as GP212. | R/W | 1 |
| 2 | IRD | IRDA/GP211<br>0 - Pin is configured for IRTXD (necessary for IRDA operability).<br>1 - Pin is configured as GP211. | R/W | 1 |
| 1 | A97 | AC97/SSI_1<br>0 - Pins are configured for AC97 mode. ACSYNC, ACBCLK, ACDO, $\overline{\text{ACRST}}$ will drive pins.<br>1 - Pins are configured for SSI_1 mode. S1DOUT, S1DIN, S1CLK and S1DEN will drive pins. | R/W | 0 |
| 0 | S0 | SSI_0/GP[210:208]<br>0 - Pins are configured for SSI_0 mode. S0CLK, S0DOUT and S0DEN will drive pins.<br>1 - Pins are configured as GP[210:208]. | R/W | 1 |

## 7.3.2 GPIO Control Registers

The Primary GPIOs on the Au1100 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 the GPIOs associated with that function.

Figure 35 shows the logical implementation of each GPIO. The names represent bit *n* of the corresponding register which affect GPIO[n].

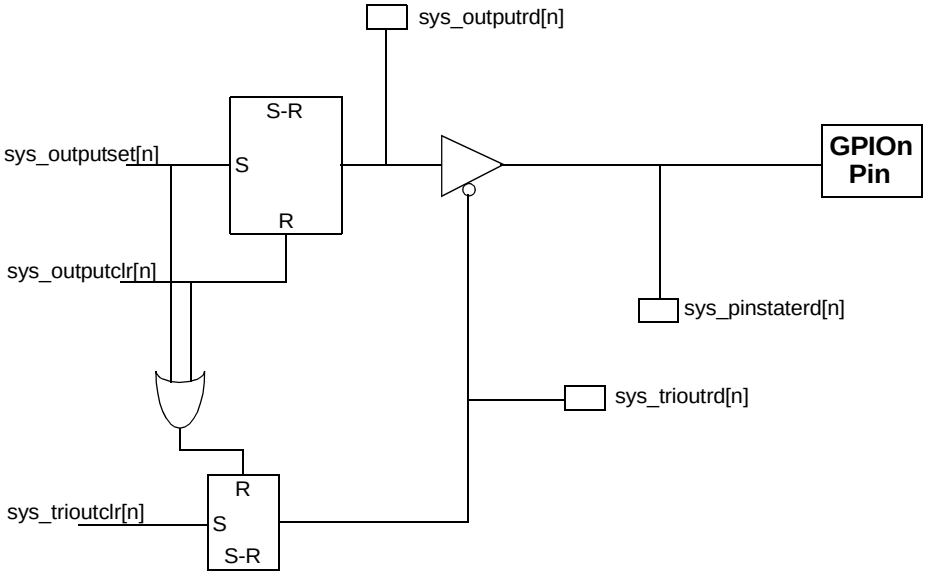



**FIGURE 35. GPIO Logic Diagram**

The following table shows the GPIO control registers and the associated offsets from sys_base. Certain offsets are shared 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 80. GPIO Control Registers**

| Offset | Register Name | Register Description | Default |
|---|---|---|---|
| 0x0100 | sys_trioutrd | The Tristate/Output state register shows the current state of the GPIO<br>0 - GPIO[n] is tristated. Tristating GPIO[n] is accomplished by setting the corresponding bit in the **sys_tristateclr**<br>1 - Output is enabled. Enabling GPIO[n] as an output is accomplished by programming GPIO[n] as a 0 or 1 using the respective **sys_outputset** or **sys_outputclr** registers.<br>If the pin is not an output it should be tristated. | 0x00000000<br>(all GPIOs are tristated) |
| 0x0100 | sys_trioutclr | | |
| 0x0108 | sys_outputrd | Controls the state of the GPIO as an output.<br>0 - output low<br>1 - output high<br>Setting or clearing bits in the output register will bring the pin out of tristate mode and enable the output. | UNPRED |
| 0x0108 | sys_outputset | | |
| 0x010C | sys_outputclr | | |
| 0x0110 | sys_pinstaterd | Allows the pin state to be read when an input. This register will also give the output state. | UNPRED |
| 0x0110 | sys_pininputen | Writing a zero to this register will allow GPIO[31:0] to be used as inputs. This register must be written a 0 before any GPIO can be used as an input an interrupt source or for use as a wake up source. Please see description below for more information. | UNPRED |

### GPIO Control Registers

Each register is 32 bits wide with bit *n* in each register affecting GPIO[n].

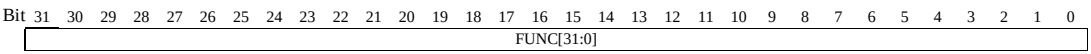
These registers will reset to defaults only on a hardware reset. During a runtime reset and during sleep this register will retain its value.

Please see Table 80 for the default values at hardware reset.

***rd**

***set**

***clr**




| Bit(s) | Name | Description | Read/Write | Default |
|---|---|---|---|---|
| 31:0 | FUNC[n] | The function of each register is given in the previous table. FUNC[n] controls the functionality of GPIO[n]. | *_read - read only<br>*_set - write only<br>*_clear - write only<br>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:

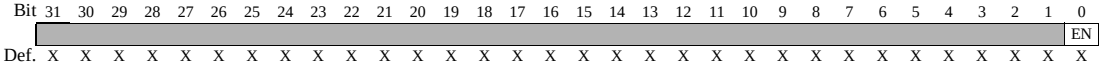
- *rd registers are read only registers will read back the current value of the register.
- *set registers are write only registers and will set to 1 all bits that are written 1. Writing a value of 0 will have no impact on the corresponding bit.
- *clr registers are write only registers and will clear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

### GPIO Input Enable

The **sys_pinputen** is a 32 bit write only register. When this register is written a 0 all GPIO's input functionality is enabled. This register will enable GPIOs for use as an input but will not explicitly make all GPIOs inputs. The value of the GPIO control registers and the pin function register will define the state of each GPIO. GPIOs can not be used as inputs until this register is written a 0.

This write is only required once per hardware reset (i.e. sleep and a runtime reset will not require another write to this register).

**sys_gpinputen** Offset = 0x0110




| Bit(s) | Name | Description | Read/Write | Default |
|---|---|---|---|---|
| 31:1 | RES | These bits are reserved and should be written a 0. | W | N/A |
| 0 | EN | A write of 0 will enable all GPIOs to be used as inputs. | W | N/A |

## 7.3.3 Hardware Considerations

The Pin Function register controls the functionality of many GPIO/peripheral pins. If a pin is programmed for a certain functionality, all other functionality associated with that pin can not be used at the same time.

For example, if bit 14 in the **sys_pinfunc** register is cleared configuring the pin as U3TXD, GPIO23 can not be used as a GPIO nor can the GPIO be configured as an interrupt.

Conversely if bit 14 is set configuring the pin as GPIO23, U3TXD and thus UART3 is not usable. GPIO23 can be used as a GPIO and to generate interrupts.

## 7.3.4 Using GPIO Lines as External DMA Requests

See Section 4.2, "Using GPIO Lines as External DMA Requests" for information.

## 7.4 Power Management

The Au1100 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, 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 36 shows the different stages of power management for the CPU (IDLE0,1) and the device (SLEEP) and how each state is entered and left. It should be noted 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_wakesrc** register) to bring the device out of sleep.

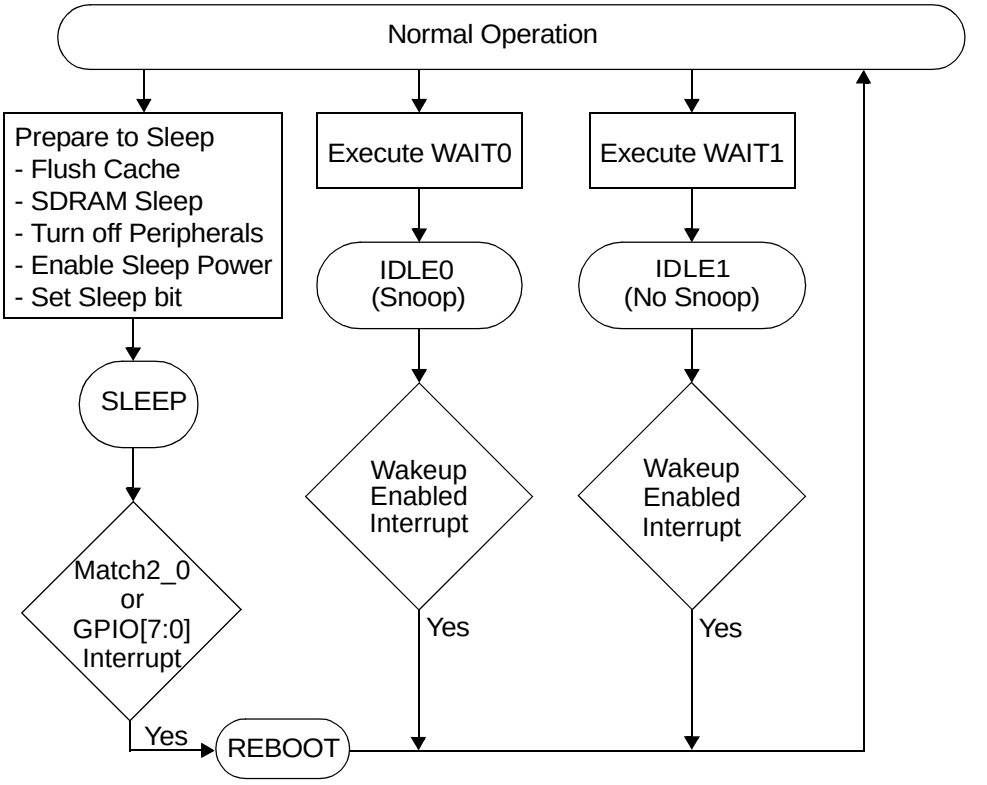



**FIGURE 36. Sleep and Idle Flow Diagram**

## 7.4.1 CPU Power Management

The CPU can be put into 2 different low power modes. This is accomplished through the instruction. The wait instruction and at least 4 instructions following it must be in the cache for the wait to occur. See Section 2.6.3, "WAIT Instruction" for more information.

In the IDLE0 state the CPU snoops the bus and cache coherency is maintained.

In the IDLE1 state the CPU does not snoop the bus and cache coherency is lost.

At all times the MMU, Data Cache, execution and multiply and accumulate block are placed in a low power state if they are not being used.

### 7.4.2 Peripherals

Peripheral power management is handled through clock management and disabling of unused peripherals. Table 81 shows a peripheral list and the relative register(s) containing any Power Management function(s).

The actual register descriptions should be referred to for programming details. When both a reset/block enable bit and clock enable bit are provided it is suggested that the reset be applied and the clocks disabled when trying to conserve power. This will simplify programming, as the suggested bring up sequence is typically to enable clocks and then subsequently to bring the peripheral out of reset.

**TABLE 81. 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_mac[0] | When this block is not being used then the bit should be cleared to disable the MAC and the CE bit should be cleared to gate clocks to the MAC. |
| UART[3:0] | uart_enable[3:0] | 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 gate clocks from the block. |
| SSI | ssi_enable | When the SSI is not being used the E bit should be cleared to hold the part in reset and the CD bit should be set to gate clocks from the block. |
| IRDA | ir_enable | The HC bit can be used to run the IRDA at half the system bus. The CE should be disabled when not using the IRDA to gate clocks from this peripheral. |
| GPIO Controller | tristate_state_set | Although there is not a specific low power register for the GPIOs, tristating all GPIOs not in use will minimize the power used by the GPIOs |

**TABLE 81. Peripheral Power Management (Continued)**

| Peripheral | Power Management Register | Power Management Strategy |
|---|---|---|
| Programmable Counters (TOY and RTC) | sys_cntrctrl | If either counter is not being used then its respective enable bit (EN0 or EN1) should be left disabled. If both counters are not being used then both bits should be disabled as well as the oscillator |
| AC97 | ac97_enable | If the AC97 block is not in use then the D bit should be used to disable the module and the CE bit should be disabled to gate clocks from the block |
| I2S | i2s_enable | If the I2S block is not in use then the E bit should be used to place the part in reset and the CE bit should be disabled to gate clocks from the block |
| LCD Controller | | #### then the E bit should be used to place the part in reset and the CE bit should be disabled to gate clocks from the block |
| SD Controller | [1:0] | ### then the E[1:0] bits should be used to place the part in reset and the CE bit should be disabled to gate clocks from the block |

### 7.4.3 Device Power Management - Sleep

The sleep state of the Au1100 puts the entire device into a low power sleep state. This is the lowest power state of the part and requires a complete 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) will continue clocking and will remain functional during sleep. The programmable counter 1 as well as other clocks throughout the Au1100 will be disabled during sleep.

When coming out of sleep there is a delay set by bit $V_{put}$ in the **sys_powerctrl** register. This is the time that the system designer has to guarantee VDDI is stable from the rising edge of PWREN.

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 as after SDRAM is put into autorefresh, 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 peripheral in use will insure a graceful transition to sleep mode.
3. Push dirty data out of the cache. During sleep all data in the caches will be lost.
4. If SDRAM contents are to be kept through sleep, SDRAM should be put into a self refresh mode. See Section 3.1, "SDRAM Memory Controller" for more information.
   If SDRAM is not needed through to be kept through sleep, disable the SDRAM.
5. If using one of GPIO[7:0] as a wakeup source, **sys_gpinputen** must be written a 0 to enable the GPIO as an input if this has not already been done by the system at startup.
6. The **sys_wakemsk** register should be set with the appropriate value according to what signal(s) should wake the processor up.
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 will put system to sleep.
9. When the system is going to sleep, the PWR_EN pin will go low. This can be used to disable VDDI and VDDY if desired.

When the processor takes a sleep interrupt to wake up, the following steps should be followed:

1. After the sleep interrupt is taken, the PWR_EN pin will be asserted by hardware. Within the time indicated by Vput (bits [3:2] **sys_powerctrl** register), the system must ensure that VDDI is stable. If VDDY 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.

The power management registers and their associated offsets are listed in Table 82. These registers are located off of the base shown in Table 76.

### TABLE 82. Power Management Registers

| Offset | Register Name | Description | Reset Type |
|---|---|---|---|
| 0x0018 | sys_scratch0 | User definable register will retain value through sleep | Hardware |
| 0x001C | sys_scratch1 | User definable register will retain 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 System Bus divider and powerup 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 |

#### Scratch Registers

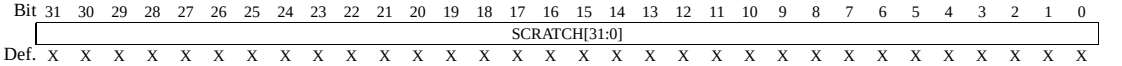
The Scratch registers are kept through sleep and are user definable.

The intention of these registers is to allow the system programmer to save state information or a pointer to a context so that the previous context can be resumed when coming out of sleep if desired.

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_scratch0** Offset = 0x0018

**sys_scratch1** Offset = 0x001C



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | SCRATCH | The scratch registers are user definable and retain their value through sleep. | R/W | UNPRED |

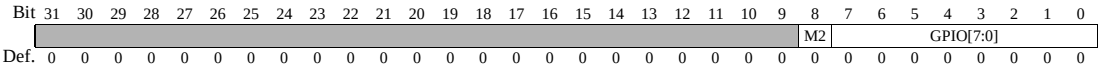
### Wakeup Source Mask Register

For each individual bit that is set, the corresponding signal or event (for the case of the 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 will retain its value.

**sys_wakemsk** Offset = 0x0034




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:9 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 8 | M20 | Setting this bit will enable Programmable Counter 0 Match Register 2 to cause a wakeup interrupt. | R/W | 0 |
| 7:0 | GPIO | Setting bit n will cause GPIO[n] to cause a sleep wakeup. | R/W | 0 |

### Endianess Register

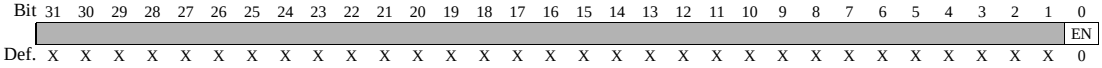
To change the endianess of the Au1100 is a three step process as follows:

1. Set Endianess bit in the **sys_endian** register
2. Read 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**

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 set by the Endianess bit in the **sys_endian** register.

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

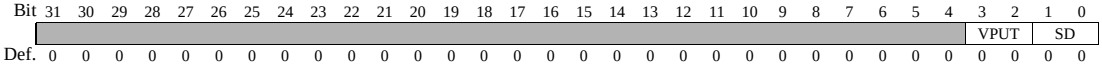



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:1 | RES | These bits are reserved and should be written a 0. | R | UNPRED |
| 0 | EN | Endianess<br>0 - Big Endian<br>1 - Little Endian | R/W | 0 |

**Power Control Register**

This register will reset to defaults only on a hardware reset. During a runtime reset and during sleep these bits will retain their value.

**sys_powerctrl** Offset = 0x003C




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 3:2 | VPUT | VDDI Power Up Time<br>00 - 20ms<br>01 - 5ms<br>10 - 100ms<br>11 - 2us | R/W | Hard-ware Reset<br>$00_b$ |
| 1:0 | SD | System Bus Clock Divider<br>00 - 2<br>01 - 3<br>10 - 4<br>11 - reserved | R/W | Hard-ware Reset<br>$00_b$ |

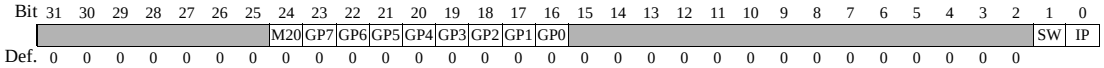
### Wakeup Cause Register

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

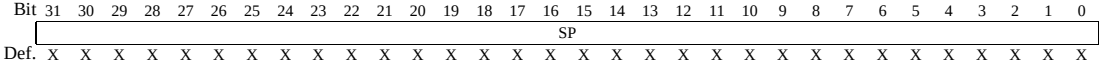



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:25 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 24 | M20 | Programmable Counter 0 Match 2 caused wakeup from sleep | R/W | 0 |
| 23 | GP7 | GPIO7 caused wakeup from sleep | R/W | 0 |
| 22 | GP6 | GPIO6 caused wakeup from sleep | R/W | 0 |
| 21 | GP5 | GPIO5 caused wakeup from sleep | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 20 | GP4 | GPIO4 caused wakeup from sleep | R/W | 0 |
| 19 | GP3 | GPIO3 caused wakeup from sleep | R/W | 0 |
| 18 | GP2 | GPIO2 caused wakeup from sleep | R/W | 0 |
| 17 | GP1 | GPIO1 caused wakeup from sleep | R/W | 0 |
| 16 | GP0 | GPIO0 caused wakeup from sleep | R/W | 0 |
| 15:2 | RES | These bits are reserved and should be written a 0. | R/W | 0 |
| 1 | SW | Sleep Wakeup<br>This bit is set by hardware on a sleep wakeup and cleared by software by a write to this register.<br>A runtime reset can be detected if both SW and IP are 0 at boot.<br>This bit must be explicitly cleared by software as it will hold its value through sleep and runtime reset. | R/W | 0 |
| 0 | IP | Initial Powerup<br>This bit is set by hardware on a hardware reset and cleared by software by a write to this register.<br>A runtime reset can be detected if both SW and IP are 0 at boot.<br>This bit must be explicitly cleared by software as it will hold its value through sleep and runtime reset. | R/W | 1 |

**Sleep Power Register**

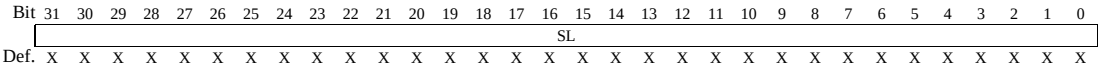
**sys_slppwr** Offset = 0x0078




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | SP | A write to this register will prepare the internal power supply for going to sleep. | W | UNPRED |

**Sleep Register**

**sys_sleep** Offset = 0x007C




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | SL | A write to this register will put system to sleep. | W | UNPRED |

# 8 Powerup, Reset and Boot

This section presents the powerup, hardware reset and runtime reset sequence for the Au1100. In addition the boot vector is described.

## 8.1 Powerup Sequence

The Au1100 power structure is designed such that the external I/O voltage, VDDX, is driven separately from the core voltage, VDDI. In this way the core voltage can be sourced at lower voltages saving power. In addition the Au1100 was designed to allow the system designer to remove the core voltage during sleep to maximize power efficiency.

Two signals VDDXOK and PWREN are used to facilitate this power strategy. VDDXOK is used as a signal to the processor that power on VDDX is stable. Stable is defined as having reached 90% of it's nominal value. PWREN 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, VDDI.

The following describes the powerup sequence for the Au1100:

1. Apply VDDX and VDDY (3.3V I/O power).
2. When VDDX and VDDY have reached 90% of nominal, assert VDDXOK.
3. The Au1100 will assert PWREN which can be used to enable the regulator driving VDDI (CPU Power).
4. VPUT defines the amount of time to ensure that VDDI has reached its nominal value. VPUT is defined in the **sys_powerctrl** register in Section 7.4, Power Management.

Figure 37 shows the power up sequence pictorially. Actual timing numbers are given in Section 8.3

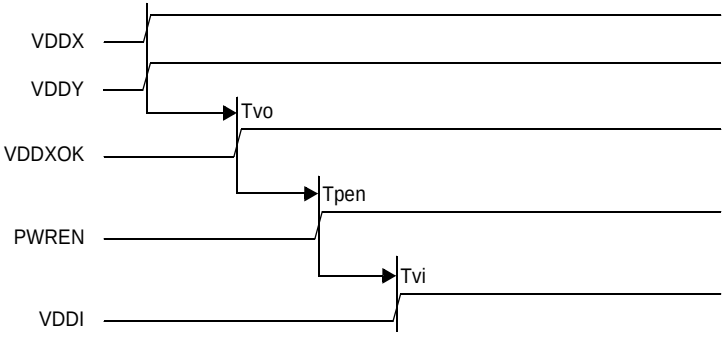



**FIGURE 37. Powerup Sequence**

## 8.2 Reset

A hardware reset is is defined as a reset in which both VDDXOK and $\overline{\text{RESETIN}}$ and are toggled. Typically this happens only at power on but a system designer can choose to tie VDDXOK and $\overline{\text{RESETIN}}$ together in which case all resets will be a hardware reset.

Runtime reset is that in which power remains applied and only the $\overline{\text{RESETIN}}$ signal is toggled. It should be noted that certain registers, specifically some of those in the System Control Block, will not be affected by this type of reset. Please 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 software reset then it will be noted in the register description.

The state of ROMSEL and ROMSIZE will determine where the CPU boots from and the width of the boot ROM according to Table 83. This applies to both runtime reset and hardware reset. ROMSEL and ROMSIZE should be terminated appropriately as these signals should not change during runtime.

**TABLE 83. Boot Type**

| ROMSEL | ROMSIZE | Boot Type |
|---|---|---|
| 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 |

### 8.2.1 Hardware Reset

As previously mentioned a hardware reset is defined as one in which VDDXOK makes a transition from low to high followed by $\overline{\text{RESETIN}}$ deassserting (transitioning from low to high).

The following sequence describes a hardware reset. Figure 38 shows this sequence pictorially. Actual timing numbers are given in Section 8.3.

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, $\overline{\text{RESETIN}}$ can be deasserted. In other words, $\overline{\text{RESETIN}}$ can not be deasserted before VDDXOK is asserted. This allows VDDXOK and $\overline{\text{RESETIN}}$ to be tied together.
3. $\overline{\text{RESETOUT}}$ will be deasserted $T_{vro}$ after VDDI is applied.

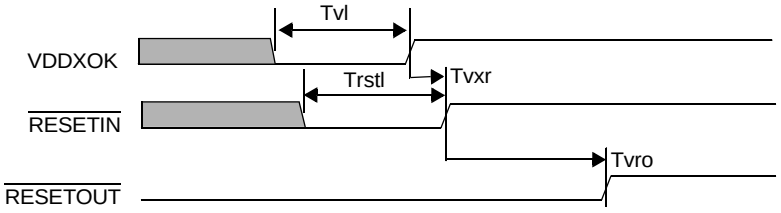



**FIGURE 38. Hardware Reset Sequence**

## 8.2.2 Runtime Reset

During runtime (after power is stable) the reset sequence can be broken down as follows. It should be noted that certain registers (specifically those in the System Control Block will not be affected by this type of reset). Figure 38 shows this sequence pictorially. Actual timing numbers are given in Section 8.3.

1. ROMSEL and ROMSIZE should be terminated in the design so the appropriate boot type occurs. These values should not change during runtime.
2. During a runtime reset it is assumed that VDDX and VDDI remain at their nominal voltage. In addition, VDDXOK must remain asserted or a hardware reset will occur. PWREN will remain asserted by the Au1100.
3. $\overline{\text{RESETIN}}$ must be held low long enough to satisfy $T_{rstl}$.
4. $T_{ror}$ and $T_{rof}$ will apply when resetting during runtime

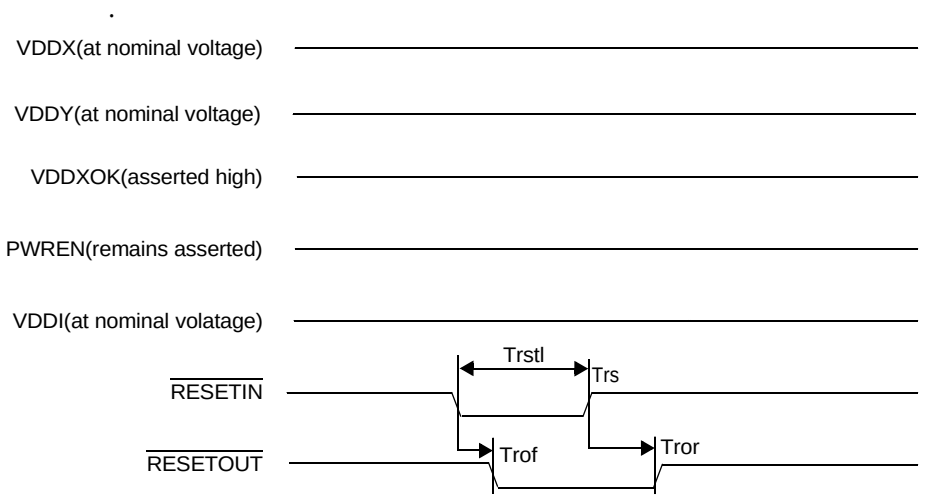



**FIGURE 39. Run time Reset Sequence**

## 8.3 Powerup and Reset Timing

Table 84 describes the timing signals defining the reset sequence as shown in the three reset different scenarios.

**TABLE 84. Reset Timing Parameters**

| Parameter | Description | Min | Max |
|---|---|---|---|
| $T_{vo}$ | VDDX at 90% of nominal to VDDXOK asserted | 0ns | |
| $T_{pen}$ | VDDXOK asserted to PWREN driven high | | TBD |
| $T_{vi}$ | PWREN to VDDI stable<br>VPUT is bits [3:2] in the sys_powerctrl register. See Section 7.4, Power Management, for more information. | | VPUT |
| $T_{vxr}$ | VDDXOK asserted to RESETIN deasserted | 0ns | |
| $T_{vl}$ | VDDXOK low time | 1us | |
| $T_{rstl}$ | $\overline{\text{RESETIN}}$ low time | 1us | |

**TABLE 84. Reset Timing Parameters**

| Parameter | Description | Min | Max |
|---|---|---|---|
| $T_{vro}$ | $\overline{\text{RESETIN}}$ to $\overline{\text{RESETOUT}}$ delay<br>MAX: {min[750nS,150mS + Vput - Tvxr(actual)]} | 600ns | see desc. |
| $T_{rof}$ | $\overline{\text{RESETIN}}$ falling to $\overline{\text{RESETOUT}}$ falling<br>MAX: 25nS + (0.5 * (CPU Clock/2)) | | see desc. |
| $T_{ror}$ | $\overline{\text{RESETIN}}$ rising to $\overline{\text{RESETOUT}}$ rising<br>MAX: 25nS + (0.5 * (CPU Clock/2)) + (120 * CPU Clock) | | see desc. |

## 8.4 Boot

The CPU will boot from KSEG1 address 0xBFC0 0000 which is translated to physical address 0x1FC0 0000. The processor can be set to boot from the EJTAG probe through the EJTAG port. Please see Chapter 9 for more information.

The system designer should set the ROMSEL and ROMSIZE pins appropriately so boot width and ROM type will match that designated in Table 84 and have the start of the boot code located at 0x1FC0 0000.

$\overline{\text{RCE0}}$ is configured to be enabled for 0x1FC0 0000 at default when booting from a ROM device. Please see Section 3.2, Static Bus Controller, for more information about the default timing and size of the address enabled at reset.

$\overline{\text{SDCS0}}$ is configured to be enabled for 0x1FC0 0000 at default when booting from a SMROM device. Please see Section 3.1, SDRAM Memory Controller, for more information about the default timing and size of the address enabled at reset.

# 9 EJTAG

The Au1100 implements EJTAG following the MIPS' EJTAG 2.5 Specification. This section presents the Au1100 EJTAG implementation while concentrating on those features implemented 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 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:

- Extended instructions SDBBP and DERET
- Debug Exceptions
- Extended CP0 registers DEBUG, DEPC and DESAVE
- EJTAG Memory Range 0xFF200000 - 0xFF3FFFFF
- Instruction/Data Breakpoints through the watch exception (Au1100 specific)
- Processor Bus Breakpoints (from EJTAG 2.0)
- Memory Overlay (from EJTAG 2.0)
- EJTAG tap per IEEE1149.1

## 9.1 EJTAG Instructions

Both SDBBP and DERET are supported by the Au1100.

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 specific implementation allowing CPU watch exception to cause debug exception. See description of the "EJWatch Register (TAP Instruction EJWATCH)" on page 334 register.

It should be noted 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 85.

**TABLE 85. 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 |

**Debug Register (CP0 Register 23, Select 0)**

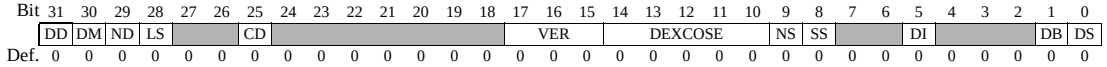
The Debug register contains the cause of the most recent debug exception and exception in Debug Mode. It also controls single stepping.

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 only updated 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | DD | DBD<br>Indicates whether the last debug exception or exception in Debug Mode occurred in a branch or jump delay slot.<br>0: Not in delay slot<br>1: In delay slot | R | UNPRED |
| 30 | DM | Indicates that the processor is operating in Debug Mode.<br>0: Processor is operating in Non-Debug Mode<br>1: Processor is operating in Debug Mode | R | 0 |
| 29 | ND | NoDCR<br>0: DSEG is present. | R | 0 |
| 28 | LS | LSNM<br>Controls access of loads/stores between dseg and remaining memory when dseg is present and while in debug mode.<br>0: Loads/stores in dseg address range go to dseg<br>1: Loads/stores in dseg address range go to system memory | R/W | 0 |
| 27 | RES | This bit is reserved and should be written a 0. *This bit is called Doze in the EJTAG 2.5 specification and was not implemented*. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 26 | RES | This bit is reserved and should be written a 0. *This bit is called Halt in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 25 | CD | CountDM<br>This bit is 0, indicating that the counter will be stopped in debug mode. | R | 0 |
| 24 | RES | This bit is reserved and should be written a 0. *This bit is called IBusEP in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 23 | RES | This bit is reserved and should be written a 0. *This bit is called MCheckP in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 22 | RES | This bit is reserved and should be written a 0. *This bit is called CacheEP in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 21 | RES | This bit is reserved and should be written a 0. *This bit is called DBusEP in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 20 | RES | This bit is reserved and should be written a 0. *This bit is called IEXI in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 19 | RES | This bit is reserved and should be written a 0. *This bit is called DDBSImpr in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 18 | RES | This bit is reserved and should be written a 0. *This bit is called DDBLImpr in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 17:15 | VER | EJTAGver<br>1: EJTAG Version 2.5 | R | 1 |
| 14:10 | DEX-CODE | DExcCode<br>Indicates the cause of the latest exception in Debug Mode.<br>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.<br>This value is undefined after a debug exception. | R | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 9 | NS | NoSSt<br>0: Single step is implemented. | R | 0 |
| 8 | SS | SSt<br>Controls whether single-step feature is enabled:<br>0: No enable of single-step feature<br>1: Single-step feature enabled | R/W | 0 |
| 7:6 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 5 | DI | DINT<br>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.<br>Cleared on exception in Debug Mode.<br>0: No Debug Interrupt exception<br>1: Debug Interrupt exception | R | UNPRED |
| 4 | RES | This bit is reserved and should be written a 0. *This bit is called DIB in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 3 | RES | This bit is reserved and should be written a 0. *This bit is called DDBS in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 2 | RES | This bit is reserved and should be written a 0. *This bit is called DDBL in the EJTAG 2.5 specification and was not implemented*. | R | 0 |
| 1 | DB | DBp<br>Indicates that a Debug Breakpoint exception occurred. Cleared on exception in Debug Mode.<br>0: No Debug Breakpoint exception<br>1: Debug Breakpoint exception | R | UNPRED |
| 0 | DS | DSS<br>Indicates that a Debug Single Step exception occurred. Cleared on exception in Debug Mode.<br>0: No debug single-step exception<br>1: Debug single-step exception | R | UNPRED |

### 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.

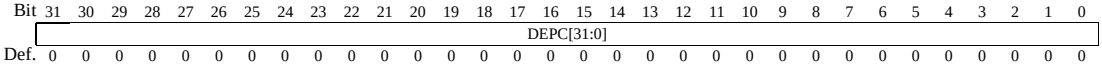
Hardware updates this register on debug exceptions and exceptions in Debug Mode.

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



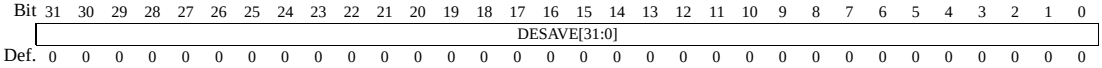

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | DEPC | Debug Exception Program Counter | R/W | UNPRED |

### 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | DESAVE | Debug Exception Save contents | R/W | UNPRED |

## 9.4 EJTAG Memory Range

In debug mode accesses to virtual 0xFF200000-0xFF3FFFFF bypass translation.

The debug memory is split into two logical divisions:

- dmseg: 0xFF20 0000 - 0xFF2F FFFF
- drseg: 0xFF30 0000 - 0xFF3F FFFF

It should be noted that 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 86 shows the EJTAG memory mapped registers located in drseg.

**TABLE 86. 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 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.

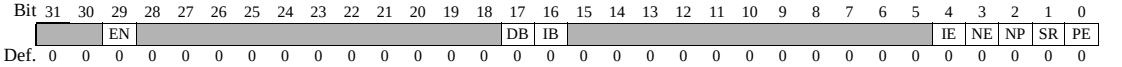
The Processor Bus Match registers monitor the bus interface of the MIPS CPU and provide debug exception or trace trigger for a given physical address and data.

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 334 for details.

### 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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:30 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 29 | EN | ENM<br>1: Processor is big Endian in both debug and kernel mode. | R | 1 |
| 28:18 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 17 | DB | DataBrk<br>0: No data hardware breakpoints implemented. | R | 0 |
| 16 | IB | InstBrk<br>0: No instruction hardware breakpoints implemented. | R | 0 |
| 15:5 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 4 | IE | IntE<br>1: Interrupt enabled in debug mode depending on other enabling mechanisms. | R | 1 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 3 | NE | NMIE<br>1: Non-Maskable Interrupt is enable for non-debug mode.<br>The NMI is not implemented in the Au1100 so this bit has no applicability. | R | 1 |
| 2 | NP | NMIPend<br>0: no NMI pending<br>The NMI is not implemented in the Au1100 so this bit has no applicability. | R | 0 |
| 1 | SR | SRstE<br>1: Soft reset is fully enabled.<br>Soft Reset is not implemented in the Au1100 so this bit has no applicability. | R | 1 |
| 0 | PE | ProbEn<br>Indicates value of the ProbEn value in the ECR register.<br>0: No access should occur to dmseg<br>1: Probe services accesses to dmseg | R | Same value as ProbEN in ECR |

**Processor Bus Break Status Register**

**pbs - Processor Bus Break Status** Offset = 0x000C


.

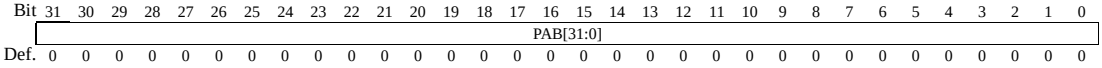
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 30 | OLP | 1: Memory overlay functionality is implemented for processor breaks. | R | 1 |
| 29:28 | RES | These bits are reserved and should be written a 0 | R | 0 |
| 27:24 | BCN | Number of Processor Breaks<br>1: One Channel has been implemented for the Processor Bus Break. | R | 1 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 23:15 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 14:1 | RES | These bits are reserved and should be written a 0. *These bits are the Bsn bits in the EJTAG 2.0.0 specification and are not needed since only one break is implemented.* | R | 0 |
| 0 | BS | Break Status<br>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.<br>The Debug handler must clear this bit before returning from debug mode. | R/W | 0 |

**Processor Address Bus Break**

This register contains the bits of the physical Processor Address Bus Break.

**pab - Processor Address Bus Break** Offset = 0x0300

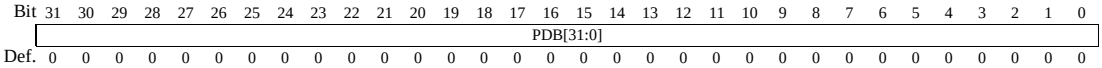



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | PAB | Processor Address Bus Break 0. This index contains the lower 32 bits of the physical address. In combination with the high order address bits, these bits make up the break address. | R | UNPRED |

**Processor Data Bus Break**

This register specifies the data value for the Processor Data Bus match.

**pdb - Processor Data Bus Break** Offset = 0x0304



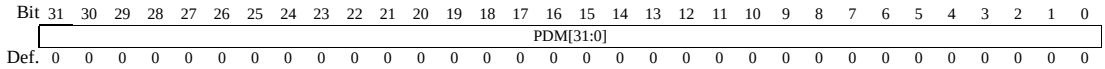
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | PDB | Processor Data Bus Break 0<br>This index contains the 32 bits of the data bus match. | R | UNPRED |

### Processor Data Mask/Upper Overlay Address Mask

This register is dual purpose depending on the value of the Overlay Enable bit in the 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



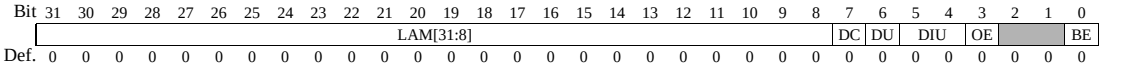

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | PDM | When OE not enabled<br><br>Processor Data Mask 0<br>When OE in the pbcam register is not enabled.<br>0: Data bit is not masked, data bit is compared.<br>1: Data bit is masked, data bit is not compared. | R | UNPRED |
| 31:24 | UOAM | When OE is enabled<br><br>Upper Overlay Address Mask<br>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.<br><br>0: Address bit is not masked, address bit is compared.<br>1: Address bit is masked, address bit is not compared.<br><br>It should be noted that bits 23:0 are not used when OE is set and should be written 0. | R/W | UNPRED |

### 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.

It should be noted that all processor break exceptions are imprecise.

**pbcam - Bus Break Control and Address Mask** Offset = 0x030C




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:8 | LAM | Address Mask<br>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.<br>0: Address bit is not masked, address bit is compared.<br>1: Address bit is masked, address bit is not compared. | R/W | UNPRED |
| 7 | DC | Data Store to Cached Area<br>This bit enables the comparison on Processor Address and Data Bus for Data Store to the Cached area.<br>0: Processor Address and Data is not compared for storing data to the Cached area.<br>1: Processor Address and Data is compared for storing data to the Cached area. | R/W | UNPRED |
| 6 | DU | Data Store To Uncached Area<br>This bit enables the comparison on Processor Address and Data Bus for Data Store to the uncached area.<br>0: Processor Address and Data is not compared for storing data into the un-cached area.<br>1: Processor Address and Data is compared for storing data into the un-cached area. | R/W | UNPRED |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 5:4 | DIU | Data or Instruction fetch or load from Uncached Area<br>These bits enable the comparison on Processor Address and Data Bus for Data or Instruction load and fetch from the un-cached area.<br><br>00: Processor Address and Data is not compared for loading data or fetching instruction from the un-cached area.<br>11: Processor Address and Data is compared for loading data or fetching instruction from the un-cached area.<br><br>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<br>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.<br>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.<br>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 | RES | This bit is reserved and should be written a 0. *This bit is called TE in the EJTAG 2.0.0 specification and was not implemented*. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 1 | RES | This bit is reserved and should be written a 0. *This bit is called CBE in the EJTAG 2.0.0 specification and was not implemented*. | R | 0 |
| 0 | BE | Break Enable<br>This bit enables the Processor Bus break function.<br>0: Processor Bus break function is disabled<br>1: Processor Bus break function is enabled<br><br>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.<br>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.<br><br>This functionality is mutually exclusive to OLE so only one of OLE or BE should be set at any time. | R/W | 0 |

### Processor High Address Bus Break

This register specifies the high order address for the processor address bus break.

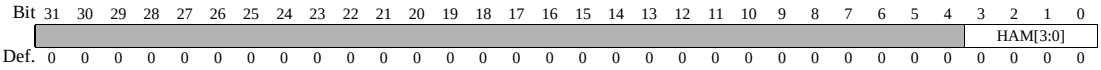
**pha - Processor High Address Bus Break** Offset = 0x0310


| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 3:0 | HA | These bits map to the high physical address bits 35:31. | R/W | UNPRED |

**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(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:4 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 3:0 | HAM | High Address Mask for address bits 35:31<br>0: Data bit is not masked, data bit is compared<br>1: Data bit is masked, data bit is not compared | R/W | UNPRED |

## 9.4.2 EJTAG Test Access Port (TAP)

The EJTAG TAP contains the 5 TAP pins and a 16 state controller with a 5 bit instruction register.

Table 87 shows the 5-bit instructions supported by the Au1100.

### TABLE 87. 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 | RES | Reserved |
| 0x05 | RES | This reserved register is for test mode HIZ - Tristate all output pins and Select Bypass register. |
| 0x06 | RES | This reserved register is for test mode CLAMP - IEEE Clamp pins and select bypass register. |

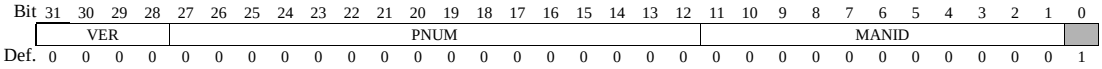
**TABLE 87. EJTAG Instruction Register Values (Continued)**

| Hex Value | Instruction | Function |
|---|---|---|
| 0x07 | RES | 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 | RES | Reserved |
| 0x1C | EJWATCH | Selects Watch register |
| 0x1D-0x1E | RES | Reserved |
| 0x1F | BYPASS | Bypass mode |

### 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



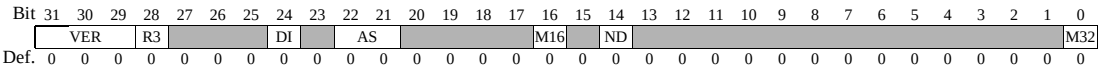

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:28 | VER | Identifies the version of the device. | R | 0 |
| 27:12 | PNUM | Identifies the part number of the device. | R | 0x03E8 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 11:1 | MANID | Identifies the manufacturer ID code for the device.<br>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 of the number of continuation characters. | R | 0x147 |
| 0 | RES | This bit is reserved and should be written a 1. | R | 1 |

### 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** TAP Instruction IMPCODE




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:29 | EJTAGver | 1: EJTAG version 2.5 | R | 1 |
| 28 | R3 | 0: R3k privileged environment | R | 0 |
| 27:25 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 24 | DI | 0: DINT signal from the probe is not supported. | R | 0 |
| 23 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 22:21 | AS | $10_b$: 8 bit ASID | R | $10_b$ |
| 20:17 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 16 | M16 | 0: no MIPS16 support | R | 0 |
| 15 | RES | This bit is reserved and should be written a 0. | R | 0 |

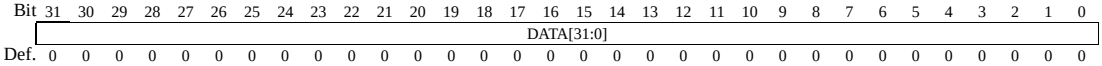
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 14 | ND | 1: No EJTAG DMA support | R | 1 |
| 13:1 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 0 | MIPS32/64 | 0: 32-bit processor | R | 0 |

### 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 Date 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** TAP Instruction DATA or ALL




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31:0 | DATA | Data used by processor access | R/W | UNPRED |

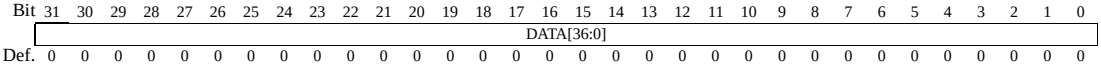
### Address Register

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).

**ADDRESS** TAP Instruction ADDRESS or ALL




| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 36:0 | Address | Address used by processor access | R | UNPRED |

### 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.

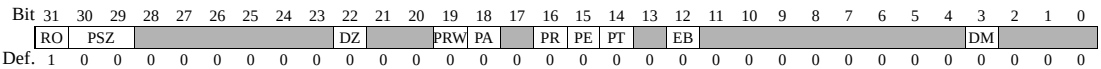
The EJTAG Control register is not updated/written in the Update-DR state unless the Reset occurred; that is Rocc (bit 31) is either already 0 or is written to 0 at the same time. This condition ensures proper handling of processor accesses after a reset.

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 (i.e. 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.

**ECR - EJTAG Control Register** TAP Instruction CONTROL or ALL



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 31 | RO | Indicates if a processor reset has occurred since the bit was cleared:<br>0: No reset occurred<br>1: Reset occurred<br>The Rocc bit stays set as long as reset is applied.<br><br>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 Rocc 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 |
| 30:29 | PSZ | Indicates the size of a pending processor access, in combination with the Address register.<br><br>0: Byte<br>1: Halfword<br>2: Word<br>3: Triple<br><br>This field is valid only when a processor access is pending; otherwise, the read value is undefined. | R | UNPRED |
| 28:23 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 22 | DZ | Doze<br>Indicates if the processor is in a WAIT state:<br>0: Processor is not in a wait state.<br>1: Processor is in a wait state. | R | 0 |
| 21 | RES | This bit is reserved and should be written a 0. *This bit is called Halt in the EJTAG 2.0.0 specification and was not implemented*. | R | 0 |
| 20 | RES | This bit is reserved and should be written a 0. *This bit is called PerRst in the EJTAG 2.0.0 specification and was not implemented*. | R | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 19 | PRW | Indicates read or write of a pending processor access.<br>0: Read processor access, for a fetch/load access<br>1: Write processor access, for a store access<br><br>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:<br>0: No pending processor access<br>1: Pending processor access<br><br>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 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 16 | PR | Controls the processor reset.<br>0: No processor reset applied<br>1: Processor reset applied<br><br>Setting this bit to 1 will apply a processor reset. When this bit is read back it will always read a 0. It should be noted that startup latencies should be observed when applying reset. | R/W | 0 |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 15 | PE | Controls indication to the processor of whether the probe expects to handle accesses to EJTAG memory through servicing of processors accesses.<br><br>0: Probe does not service processors accesses<br>1: Probe will service processor accesses<br><br>The ProbEn bit is reflected as a read-only bit in the Debug Control Register (DCR) bit 0.<br><br>When this bit is changed, then it is guaranteed that the new value has taken effect in the DCR when it can be read back here. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain.<br><br>However, a change of the ProbEn prior to setting the EjtagBrk bit is ensured to affect execution of the debug handler due to the debug exception.<br><br>Not all combinations of ProbEn and ProbTrap are allowed.<br>Please see the previous note about writing this bit (in **“EJTAG Control Register (ECR)” on page 329**). | R/W | Determined by EJTAG-BOOT |

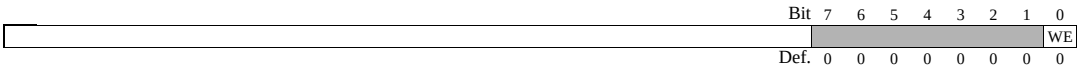
| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 14 | PT | Controls location of the debug exception vector:<br>0: Normal memory 0xBFC0 0480<br>1: EJTAG memory 0xFF20 0200<br>When this bit is changed, then it is guaranteed that the new value is indicated to the processor when it can be read back here. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain.<br><br>However, a change of the ProbTrap prior to setting the EjtagBrk bit is ensured to affect execution of the debug handler due to the debug exception.<br><br>Not all combinations of ProbEn and ProbTrap are allowed.<br>Please see the previous note about writing this bit (in **"EJTAG Control Register (ECR)" on page 329**). | R/W | Determined by EJTAG-BOOT |
| 13 | RES | This bit is reserved and should be written a 0. | R | 0 |
| 12 | EB | Requests a Debug Interrupt exception to the processor when this bit is written as 1. The debug exception request is ignored if the pro-cessor is already in debug at the time of the request. 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:<br><br>0: No pending Debug Interrupt exception requested through this bit<br>1: Pending Debug Interrupt exception<br><br>The read value can, but is not required to, indicate other pending DINT debug requests (for example, through the DINT signal). This bit is cleared by hardware when the pro-cessor enters Debug Mode.<br><br>Please see the previous note about writing this bit (in **"EJTAG Control Register (ECR)" on page 329**). | R/W1 | Determined by EJTAG-BOOT |

| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 11:4 | RES | These bits are reserved and should be written a 0 | R | 0 |
| 3 | DM | Indicates if the processor is in Debug Mode:<br>0: Processor is in Non-Debug Mode<br>1: Processor is in Debug Mode | R | 0 |
| 2:0 | RES | These bits are reserved and should be written a 0. | R | 0 |

### 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

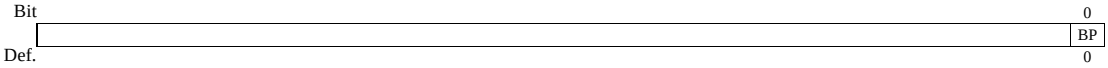



| Bit(s) | Name | Description | Read/ Write | Default |
|---|---|---|---|---|
| 7:3 | RES | These bits are reserved and should be written a 0. | R | 0 |
| 2 | RES | These bits are reserved and should be written a 0.<br>This bit is the Global Scan test bit. | R | 0 |
| 1 | RES | These bits are reserved and should be written a 0.<br>This bit is a Test Mode bit. | R | 0 |
| 0 | WATCH | This bit controls the debug functionality of the CPU watch register.<br>0: normal Watch Exception Mode<br>1: Debug Watch Exception Mode<br>- Blocks writes to Watch register in non-debug mode<br>- Watch Exception will become debug exceptions with DEXCODE=23<br>- The PC will be saved in the DEPC (not in the EPC as with a normal watch exception).<br>It should be noted that the Status, Cause, and EPC will not be affected by a debug watch exception when this bit is enabled. | R/W | 0 |

**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(s) | Name | Description | Read/ Write | 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 88.

### TABLE 88. EJTAG Signals

| Pin Name | 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 wising edge of TCK. |
| TCK | I | Control clock for updating TAP controller and shifting data through instruction or selected data register. |

It should be noted that the EJTAG TAP signal TCK must always be less than 1/4 the system bus clock speed for proper operation. In addition, termination as shown in EJTAG 2.5 spec must be followed.

# 10 Signal Description

This section describes the external I/O signals on the Au1100.

In order to maximize the functionality of the Au1100, many of the pins have multiple uses. It should be noted 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 GPIO at the same time it is assigned to a peripheral device. The configuration of the multiplexed signals is discussed in Table 89.

Figure 40 shows a block diagram of the Au1100 from an external signal perspective. All signals are grouped functionally according to the block in which they are a part. Signals that share a pin with multiple functionality are designated with a * with the alternative signal following the name in parentheses.

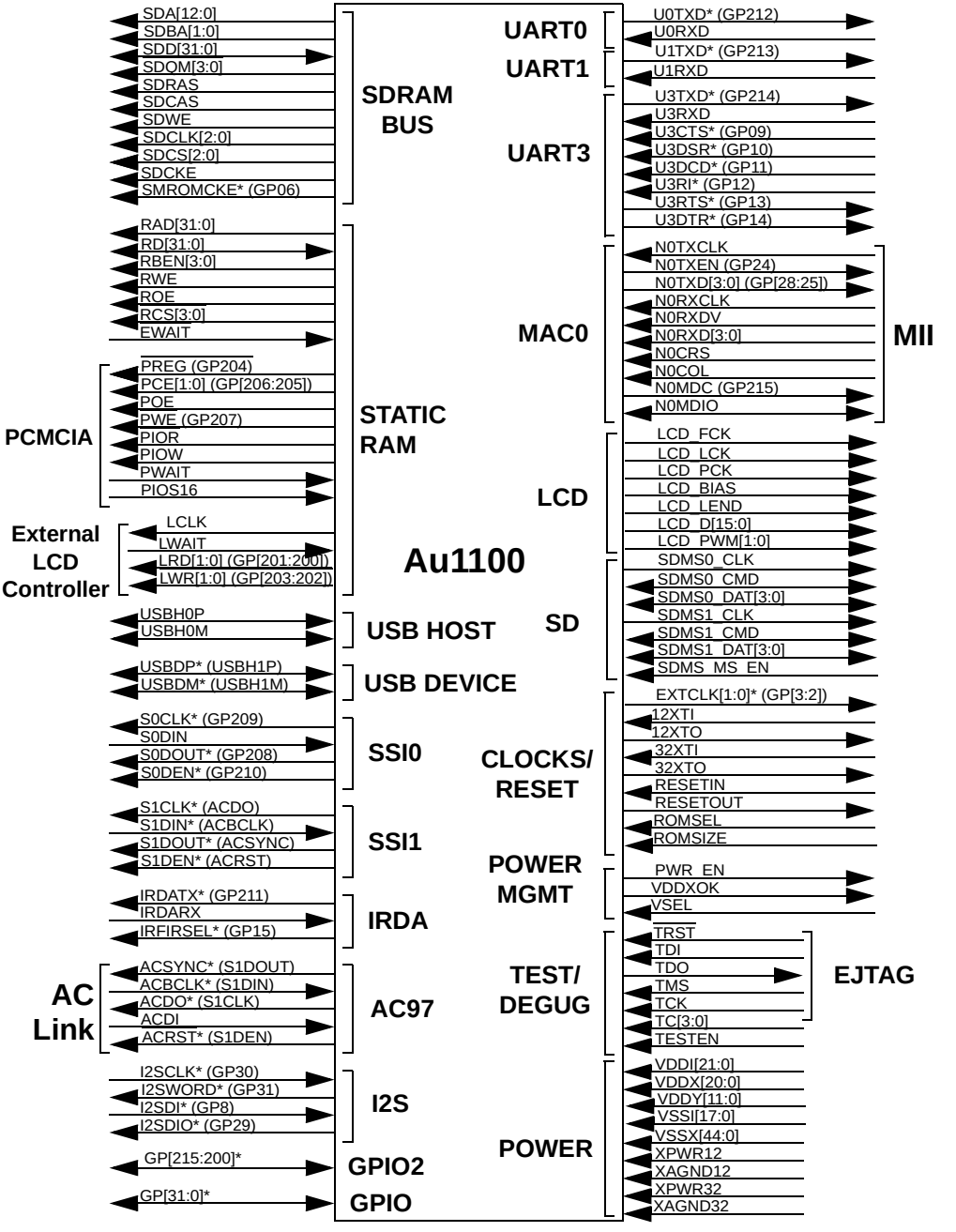



**FIGURE 40. Au1100 Block Diagram**

Table 89 gives a description of all external signals on the Au1100. The signals have been grouped by functionality. The signals that share a pin with multiple functionality are designated by a * appended to their name.

The type description can be decoded as follows:

I - Input
O - Output
IO - Bidirectional
Z - Tristatable
P - Power
G - Ground

RES - Reserved

The last columns represent default values during and coming out of Hardware Reset (HR), Runtime Reset (RR), and Sleep (S). The values can be decoded as follows:

0 - driven low
1 - driven high
IN - Signal is an input
LV - driven at the last value before the reset
HIZ - Tristated
ON - Clock remains on
NC - Not Connected
NA - Defaults are not applicable because the alternate function of the pin controls the pin coming out of reset.

### TABLE 89. Signal Description

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **SDRAM Interface:** | | | | | |
| 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. | 0 | LV | LV |
| SDBA[1:0] | O | Bank Address Inputs: BA0 and BA1 define to which bank the ACTIVE, READ, WRITE or PRECHARGE command is being applied. | 0 | LV | LV |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| SDD[31:0] | IO | SDRAM data bus<br>Hardware Reset Description:<br>0 - after VDDXOK is asserted<br>tristated when VDDI is on and RESETIN is asserted<br>0 after hardware reset sequence is complete | See desc. at left | HIZ | HIZ |
| SDQM[3:0] | O | Active Low Input/Output Mask: SDQM is an input mask signal for write accesses and an output enable signal for read accesses. SDQM0 masks SDD7:0, SDQM1 masks SDD15:8, SDQM2 masks SDD23:16, SDQM3 masks SDD31:24. | 1 | 1 | 1 |
| SDRAS | O | Active Low Command Output. SDRAS, SDCAS and SDWE (along with SDCSn) define the command being sent to the SDRAM rank. | 1 | 1 | 1 |
| SDCAS | O | Active Low Command Output. SDRAS, SDCAS and SDWE (along with SDCSn) define the command being sent to the SDRAM rank. | 1 | 1 | 1 |
| SDWE | O | Active Low Command Output. SDRAS, SDCAS and SDWE (along with SDCSn) define the command being sent to the SDRAM rank. | 1 | 1 | 1 |
| SDCLK[2:0] | O | Clock output corresponding to each of the three chip selects. Clock speed is 1/2 system bus frequency when corresponding SDCSn is set to SDRAM, 1/4 system bus frequency when corresponding SDCSn is set to SMROM. | 0 | ON | 0 |
| SDCS[2:0] | O | Active Low Programmable Chip selects | 1 | 1 | 1 |
| SDCKE | O | Clock enable for SDRAM | 1 | 1 | 0 |
| SMROMCKE | O | Synchronous Mask ROM Clock Enable.<br>This signal must be pulled high if the system is booting from SMROM.<br>*Muxed with GPIO6. If ROMSEL and ROMSIZE are configured to boot from Synchronous Mask ROM, SMROMCKE will control the pin out of reset, else GPIO6 will control the pin out of reset.* | NA | NA | NA |
| **Static Bus (SRAM/IO/PCMCIA/Flash/ROM/LCD) Interface - Common Signals** | | | | | |
| RAD[31:0] | O | Address bus | 0 | LV | LV |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| RD[31:0] | IO | Data bus | 0 | LV | LV |
| RBEN[3:0] | O | Active Low Byte Enable. $\overline{\text{RBEN0}}$ corresponds to RD7:0, $\overline{\text{RBEN1}}$ corresponds to RD15:8, $\overline{\text{RBEN2}}$ corresponds to RD23:16, $\overline{\text{RBEN3}}$ corresponds to RD31:24. | 1 | 1 | 1 |
| RWE | O | Write enable | 1 | 1 | 1 |
| ROE | O | Output enable | 1 | 1 | 1 |
| RCS[3:0] | O | Programmable Chip Selects. $\overline{\text{RCSn}}$ is not used when configured as a PCMCIA device. | 1 | 1 | 1 |
| EWAIT | I | This active low input can be used to stretch the bus access time when enabled. This input is not recognized for chip selects configured as LCD or PCMCIA as these buses have their own wait mechanisms. | IN | IN | IN |
| **PCMCIA** | | | | | |
| PREG | O | Active low register only access signal.<br>*Muxed with GPIO204 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| PCE[1:0] | O | Card Enables (Active Low)<br>*Muxed with GP[206:205] which controls the pins out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| POE | O | Output enable | 1 | 1 | 1 |
| PWE | O | Write Enable<br>*Muxed with GP207 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| PIOR | O | Read Cycle Indication | 1 | 1 | 1 |
| PIOW | O | Write Cycle Indication | 1 | 1 | 1 |
| PWAIT | I | Extend Cycle | IN | IN | IN |
| PIOS16 | I | 16 bit port select | IN | IN | IN |
| **LCD Controller Chip Interface** | | | | | |
| LCLK | O | Interface Clock | 0 | 0 | 0 |
| LWAIT | I | Extend Cycle | IN | IN | IN |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| LRD[1:0] | O | Read Indicators<br>*Muxed with GP[201:200] which controls the pins out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| LWR[1:0] | O | Write Indicators<br>*Muxed with GP[203:202] which controls the pins out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| **USB (Host)** | | | | | |
| USBH1P | IO | Positive signal of differential USB host port 1 driver.<br>Requires 15k pulldown to be USB 1.1 compliant. | IN | IN | IN |
| USBH1M | IO | Negative signal of differential USB host port 1 driver.<br>Requires 15k pulldown to be USB 1.1 compliant. | IN | IN | IN |
| USBH0P | IO | Positive signal of differential USB host port 0 driver<br>Requires 15k pulldown to be USB 1.1 compliant.<br>*Muxed with USBDP which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| USBH0M | IO | Negative signal of differential USB host port 0 driver<br>Requires 15k pulldown to be USB 1.1 compliant.<br>*Muxed with USBDP which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| **USB (Device)** | | | | | |
| USBDP | IO | Positive signal of differential USB device driver<br>Muxed with *USBH1P*. | IN | IN | IN |
| USBDM | IO | Negative signal of differential USB device driver<br>Muxed with *USBH1M*. | IN | IN | IN |
| **SSI0** | | | | | |
| S0CLK | O | Master only clock output. The speed and polarity of clock edge is programmable.<br>*Muxed with GP209 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| S0DIN | I | Serial Data Input. This signal may be tied with S0DOUT to create a single bidirectional data signal. | IN | IN | IN |

**TABLE 89. Signal Description (Continued)**

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| 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.<br>*Muxed with GP208 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| S0DEN | O | Enable signal which frames transaction. The polarity is programmable.<br>*Muxed with GP210 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **SSI1** | | | | | |
| S1CLK | O | Master only clock output. The speed and polarity of clock edge is programmable.<br>Muxed with ACDO which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| S1DIN | I | Serial Data Input. This signal may be tied with S0DOUT to create a single bidirectional data signal.<br>Muxed with ACBCLK which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| S1DOUT | 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.<br>Muxed with ACSYNC which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| S1DEN | O | Enable signal which frames transaction. The polarity is programmable.<br>Muxed with $\overline{\text{ACRST}}$ which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| **IrDA** | | | | | |
| IRDATX | O | Serial IrDA output<br>*Muxed with GP211 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| IRDARX | I | Serial IrDA input | IN | IN | IN |
| 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.<br>*Muxed with GPIO15 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| **UART0** | | | | | |

**TABLE 89. Signal Description (Continued)**

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| U0TXD | O | UART0 transmit<br>*Muxed with GP212 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| U0RXD | I | UART0 receive | IN | IN | IN |
| **UART1** | | | | | |
| U1TXD | O | UART1 transmit<br>*Muxed with GP213 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| U1RXD | I | UART1 receive | IN | IN | IN |
| **UART3** | | | | | |
| U3TXD | O | UART3 transmit<br>*Muxed with GP214 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| U3RXD | I | UART3 receive | IN | IN | IN |
| U3CTS | I | Clear to Send<br>Muxed with GPIO9 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| U3DSR | I | Data Set Ready<br>Muxed with GPIO10 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| U3DCD | I | Data Carrier Detect<br>Muxed with GPIO11 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| U3RI | I | Ring Indication<br>Muxed with GPIO12 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| U3RTS | O | Request to Send<br>Muxed with GPIO13 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| U3DTR | O | Data Terminal Ready<br>Muxed with GPIO14 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **Ethernet Controller 0** | | | | | |
| N0TXCLK | I | Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100-Mb/s and 2.5 MHz when operating at 10-Mb/s. | IN | IN | IN |
| N0TXEN | O | Active high. Indicates that the data nibble on N0TXD[3:0] is valid.<br>*Muxed with GP24 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| N0TXD[3:0] | O | Nibble wide data bus synchronous to N0TXCLK. For each N0TXCLK period in which TX_EN is asserted, TXD[3:0] will have the data to be accepted by the PHY. While TX_EN is de-asserted the data presented on TXD[3:0] should be ignored.<br>*Muxed with GP[28:25] which control the pins out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| 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). | IN | IN | IN |
| N0RXDV | I | Active high. Indicates that a receive frame is in process and that the data on N0RXD[3:0] is valid. | IN | IN | IN |
| 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 RX_DV is de-asserted, RXD[3:0] will have no effect on the MAC. | IN | IN | IN |
| 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 | IN |

**TABLE 89. Signal Description (Continued)**

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| 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 | IN |
| N0MDC | O | N0MDC is sourced by the MAC to the PHYas 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 GP215 which controls the pin out of hardware reset, runtime reset and sleep.* | NA | NA | NA |
| N0MDIO | IO | N0MDIO is the bidirectional data signal between the MAC and the PHY that is clocked by N0MDC. | 0 | LV | LV |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **External Clocks** | | | | | |
| EXTCLK[1:0] | O | General purpose clock outputs mapped from internal clock generator clocks 5 and 4.<br>Muxed with GPIO[3:2] which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |

**TABLE 89. Signal Description (Continued)**

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **LCD Controller** | | | | | |
| LCD_PWM1 | O | Pulse Width Modulation Clock 1 | 0 | 0 | 0 |
| LCD_PWM0 | O | Pulse Width Modulation Clock 0 | 0 | 0 | 0 |
| LCD_BIAS | O | Bias Clock | 0 | 0 | 0 |
| LCD_FCK | O | Frame Clock | 0 | 0 | 0 |
| LCD_LCK | O | Line Clock | 0 | 0 | 0 |
| LCD_LEND | O | Line End | 0 | 0 | 0 |
| LCD_PCK | O | Pixel Clock | 0 | 0 | 0 |
| LCD_D[15:0] | O | LCD Data | 0 | 0 | 0 |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| **Secure Digital Controller** | | | | | |
| SDMS_MS_EN | I | Reserved for future use. Must be 0. | HIZ | HIZ | HIZ |
| SDMS0_CLK | O | SD Card 0 Interface Clock | 0 | 0 | 0 |
| SDMS0_CMD | I/O | SD Card 0 Half Duplex Command and Response | HIZ | HIZ | HIZ |
| SDMS0_DAT[3:0] | I/O | SD Card 0 Data Bus | HIZ | HIZ | HIZ |
| SDMS1_CLK | O | SD Card 1 Interface Clock | 0 | 0 | 0 |
| SDMS1_CMD | I/O | SD Card 1 Half Duplex Command and Response | HIZ | HIZ | HIZ |
| SDMS1_DAT[3:0] | I/O | SD Card 1 Data Bus | HIZ | HIZ | HIZ |
| **$I^2S$** | | | | | |
| I2SCLK | O | Serial bit clock.<br>Muxed with GPIO30 | 0 | LV | LV |
| I2SWORD | O | Word clock typically configured to the sampling frequency (Fs).<br>Muxed with GPIO31 | 0 | LV | LV |
| I2SDI | O | Serial data input which should be valid on the rising edge of I2SCLK.<br>Muxed with GPIO8 which controls the pin out of hardware reset, runtime reset and sleep. | NA | NA | NA |
| 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.<br>Muxed with GPIO29 | 0 | LV | LV |
| **AC-Link** | | | | | |
| ACSYNC | O | Fixed rate sample sync<br>Muxed with S1DOUT | 0 | 0 | 0 |
| ACBCLK | I | Serial data clock. | IN | IN | IN |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| ACDO | O | TDM output stream<br>Muxed with S1CLK | 0 | 0 | 0 |
| ACDI | I | TDM input stream | IN | IN | IN |
| $\overline{\text{ACRST}}$ | O | CODEC reset<br>Muxed with S1DEN | 0 | 0 | 0 |
| **EJTAG** | | | | | |
| TRST | I | Asynchronous TAP reset | IN | IN | IN |
| TDI | I | Test data input to the instruction or selected data registers. This signal will be sampled on the rising edge of TCK | IN | IN | IN |
| 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 | HIZ | LV | LV |
| TMS | I | Control signal for TAP controller. This signal is sampled on the wising edge of TCK. | IN | IN | IN |
| TCK | I | Control clock for updating TAP controller and shifting data through instruction or selected data register. | IN | IN | IN |
| **Test** | | | | | |
| TC[3:0] | I | Test clock inputs (not used in typical application)<br>These pins should be pulled low for normal operation. | IN | IN | IN |
| TESTEN | I | Test Enable (not used in typical applications)<br>This pin should be pulled low for normal operation. | IN | IN | IN |
| **RESERVED** | | | | | |
| RESVD[5:4] | RES | Reserved. | IOZ | IOZ | IOZ |
| RESVD[3] | RES | Reserved. | I | I | I |
| RESVD[2:0] | RES | Reserved. | IOZ | IOZ | IOZ |
| **GPIO** | | | | | |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| GPIO[2:0] | IOZ | General Purpose IO | HIZ | HIZ | HIZ |
| GPIO[3:2] | IOZ | General Purpose IO<br>Muxed with EXTCLK[1:0] | HIZ | HIZ | HIZ |
| GPIO6 | IOZ | General Purpose IO<br>Muxed with SROMCKE. If ROMSEL and ROMSIZE are configured to boot from Synchronous Mask ROM, SMROMCKE will control the pin out of reset, else GPIO6 will control the pin out of reset. | HIZ | HIZ | HIZ |
| GPIO7 | IOZ | General Purpose IO | HIZ | HIZ | HIZ |
| GPIO8 | IOZ | General Purpose IO<br>Muxed with I2SDI | HIZ | HIZ | HIZ |
| GPIO9 | IOZ | General Purpose IO<br>Muxed with U3CTS | HIZ | HIZ | HIZ |
| GPIO10 | IOZ | General Purpose IO<br>Muxed with U3DSR | HIZ | HIZ | HIZ |
| GPIO11 | IOZ | General Purpose IO<br>Muxed with U3DCD | HIZ | HIZ | HIZ |
| GPIO12 | IOZ | General Purpose IO<br>Muxed with U3RI | HIZ | HIZ | HIZ |
| GPIO13 | IOZ | General Purpose IO<br>Muxed with U3RTS | HIZ | HIZ | HIZ |
| GPIO14 | IOZ | General Purpose IO<br>Muxed with U3DTR | HIZ | HIZ | HIZ |
| GPIO15 | IOZ | General Purpose IO<br>Muxed with IRFIRSEL | HIZ | HIZ | HIZ |
| GPIO[23:16] | IOZ | General Purpose IO | HIZ | HIZ | HIZ |
| GPIO24 | IOZ | General Purpose IO<br>Muxed with N1TXEN | HIZ | HIZ | HIZ |
| GPIO[28:25] | IOZ | General Purpose IO<br>Muxed with N1TXD[3:0] | HIZ | HIZ | HIZ |

**TABLE 89. Signal Description (Continued)**

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| GPIO29 | IOZ | General Purpose IO<br>Muxed with I2SDIO | HIZ | HIZ | HIZ |
| GPIO30 | IOZ | General Purpose IO<br>Muxed with I2SCLK | HIZ | HIZ | HIZ |
| GPIO31 | IOZ | General Purpose IO<br>Muxed with I2SWORD | HIZ | HIZ | HIZ |
| GPIO200 | IOZ | General Purpose IO<br>Muxed with LRD0 | HIZ | HIZ | HIZ |
| GPIO201 | IOZ | General Purpose IO<br>Muxed with LRD1 | HIZ | HIZ | HIZ |
| GPIO202 | IOZ | General Purpose IO<br>Muxed with LWR0 | HIZ | HIZ | HIZ |
| GPIO203 | IOZ | General Purpose IO<br>Muxed with LWR1 | HIZ | HIZ | HIZ |
| GPIO204 | IOZ | General Purpose IO<br>Muxed with PREG | HIZ | HIZ | HIZ |
| GPIO205 | IOZ | General Purpose IO<br>Muxed with PCE0 | HIZ | HIZ | HIZ |
| GPIO206 | IOZ | General Purpose IO<br>Muxed with PCE1 | HIZ | HIZ | HIZ |
| GPIO207 | IOZ | General Purpose IO<br>Muxed with PWE | HIZ | HIZ | HIZ |
| GPIO208 | IOZ | General Purpose IO<br>Muxed with S0DOUT | HIZ | HIZ | HIZ |
| GPIO209 | IOZ | General Purpose IO<br>Muxed with S0CLK | HIZ | HIZ | HIZ |
| GPIO210 | IOZ | General Purpose IO<br>Muxed with S0DE | HIZ | HIZ | HIZ |
| GPIO211 | IOZ | General Purpose IO<br>Muxed with IRDATX | HIZ | HIZ | HIZ |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| GPIO212 | IOZ | General Purpose IO<br>Muxed with U0TXD | HIZ | HIZ | HIZ |
| GPIO213 | IOZ | General Purpose IO<br>Muxed with U1TXD | HIZ | HIZ | HIZ |
| GPIO214 | IOZ | General Purpose IO<br>Muxed with U3TXD | HIZ | HIZ | HIZ |
| GPIO215 | IOZ | General Purpose IO<br>Muxed with N0MDC | HIZ | HIZ | HIZ |
| **Clocks and Reset** | | | | | |
| 12XTI | I | Internally compensated 12MHz (typical) crystal input | | | |
| 12XTO | O | Internally compensated 12MHz (typical) crystal output | | | |
| 32XTI | I | Internally compensated 32.768kHz (typical) crystal input | | | |
| 32XTO | O | Internally compensated 32.768kHz (typical) crystal output | | | |
| RSTIN | I | CPU reset input | IN | IN | IN |
| RSTOUT | O | Buffered output of CPU reset input | 0 | 0 | 0 |
| ROMSEL | I | Determines if boot is from ROM or SMROM.<br>ROMSEL should be terminated appropriately as these signals should not change during runtime. | IN | IN | IN |
| ROMSIZE | I | Latched at the rising edge of reset to determine if ROM width is 16 bit or 32 bit.<br>ROMSIZE should be terminated appropriately as these signals should not change during runtime. | IN | IN | IN |
| **Power Management** | | | | | |
| PWR_EN | O | Active high power enable output. This signal is intended to be used as the regulator enable for the VDDI (core power). | 0 | 1 | 0 |
| VDDXOK | I | Active high input to signal that VDDX is stable. | IN | IN | IN |
| **Power/Ground** | | | | | |

## TABLE 89. Signal Description (Continued)

| Interface Type | Type | Description | HR | RR | S |
|---|---|---|---|---|---|
| VDDI | P | Internal core voltage. | | | |
| VDDX | P | External I/O voltage. | | | |
| VDDY | P | Individual External I/O voltage for SDRAM only. | | | |
| VSEL | I | External SDRAM voltage type:<br>1 - 3.3V<br>0 - 2.5V | IN | IN | IN |
| VSS | G | Ground | | | |
| XPWR12 | P | 12MHz (typical) oscillator and PLL power.<br>This pin should be connected to VDDX through a 10 Ohm resistor. In addition a 22uF CAP in parallel with a .01uF CAP should be placed from this pin to XAGND12. | | | |
| XAGND12 | G | 12MHz (typical) oscillator and PLL ground. | | | |
| XPWR32 | P | 32.768kHz (typical) oscillator power.<br>This pin should be connected to VDDX through a 10 Ohm resistor. In addition a 22uF CAP in parallel with a .01uF CAP should be placed from this pin to XAGND32. | | | |
| XAGND32 | G | 32.768kHz (typical) oscillator ground | | | |

# 11 Electrical Specifications

This chapter provides the following electrical specifications for the Au1100:

- Absolute Maximum Ratings
- DC Parameters
- AC Parameters
- Crystal Specifications

This chapter contains preliminary information that is subject to full Au1100 characterization.

## 11.1 Absolute Maximum Ratings

Table 90 shows the absolute maximum ratings for the Au1100. These ratings are stress ratings, operating at or beyond these ratings for extended periods of time may result in damage to the Au1100.

Unless otherwise designated all voltages are relative to VSS.

**TABLE 90. Absolute Maximum Ratings**

| Parameter | Description | Minimum | Maximum | Units |
|---|---|---|---|---|
| VDDI | Core Voltage | VSS - .5 | 1.2 | Volts |
| VDDX | I/O Voltage | VSS - .5 | 3.6 | Volts |
| VDDY | I/O Voltage | VSS - .5 | 3.6 | Volts |
| XPWR12, XPWR32 | Oscillator Voltage | VSS - .5 | 3.6 | Volts |
| Vin | Voltage applied to any pin | VSS - .5 | VDDX + .5 | Volts |
| Ts | Storage Temperature | -40 | 125 | Deg. C |

## 11.2 DC Parameters

Table 91 shows the DC parameters for the Au1100.

Unless otherwise designated all voltages are relative to VSS. The operating requirements for VDDX and VDDI are given in the sections describing the DC characteristics for the different operating frequencies. For the SDRAM interface any parameters specified as functions of VDDX are assumed to be functions of VDDY unless noted otherwise.

**TABLE 91. DC Parameters**

| Parameter | Description | Min | Nominal | Max | Units |
|---|---|---|---|---|---|
| Vihx | Input High Voltage | 0.8 * VDDX | | | Volts |
| Vilx | Input Low Voltage | | | 0.2 * VDDX | Volts |
| Vohx @ 2mA | Output High Voltage | 0.8 * VDDX | | | Volts |
| Volx @ 2mA | Output Low Voltage | | | 0.2 * VDDX | Volts |
| Vihy | SDRAM Input High Voltage | 0.8 * VDDY | | | Volts |
| Vily | SDRAM Input Low Voltage | | | 0.2 * VDDY | Volts |
| Vohy @ 2mA | SDRAM Output High Voltage | 0.8 * VDDY | | | Volts |
| Voly @ 2mA | SDRAM Output Low Voltage | | | 0.2 * VDDY | Volts |
| Ii | Input Leakage Current | | | 25 | uA |
| Cin | Input Capacitance (This parameter is by design and not tested) | | 5 | | pF |
| Ta | Operating Temperature | 0 | | 70 | Deg. C |
| Ixpwr12 | XPWR12 Current | | 1 | 3 | mA |

**TABLE 91. DC Parameters (Continued)**

| Parameter | Description | Min | Nominal | Max | Units |
|---|---|---|---|---|---|
| Ixpwr32 | XPWR32 Current | | 1 | 3 | mA |
| VDDX - VDDI | Difference between I/O voltage and core voltage during operation. | 1 | | | V |
| VDDY - VDDI | Difference between I/O voltage and core voltage during operation. | 0 | | | V |

## 11.2.1 CPU Frequency <= 333 MHz

Table 90 shows the power and voltage requirements for the Au1100 with a CPU frequency less than or equal to 333 MHz.

**TABLE 92. DC Parameters for CPU frequency <= 333 MHz**

| Parameter | Min | Typical | Max | Units |
|---|---|---|---|---|
| VDDI | TBD | 1.0 | TBD | Volts |
| VDDX, XPWR12, XPWR32<br>XPWR12 and XPWR32 should be connected to VDDX through the circuit descibed in Chapter 10 | 2.7 | 3.3 | 3.6 | Volts |
| VDDY (VSEL=1) | 2.7 | 3.3 | 3.6 | Volts |
| VDDY (VSEL=0) | TBD | 2.5 | TBD | Volts |
| Power: VDDI | TBD | TBD | TBD | mW |
| Power: VDDX+VDDY where VDDX=VDDY | TBD | 75 | TBD | mW |
| IDLE0 Power | | TBD | | mW |
| IDLE1 Power | | TBD | | mW |
| Sleep Current (VDDI = VSS) | | | 50 | uA |

### 11.2.2 CPU Frequency <= 400 MHz

Table 93 shows the power and voltage requirements for the Au1100 with a CPU frequency less than or equal to 400 MHz.

**TABLE 93. DC Parameters for CPU frequency <= 400 MHz**

| Parameter | Min | Typical | Max | Units |
|---|---|---|---|---|
| VDDI | TBD | 1.0 | TBD | Volts |
| VDDX, XPWR12, XPWR32<br>XPWR12 and XPWR32 should be connected to VDDX through the circuit descibed in Chapter 10 | 3.0 | 3.3 | 3.6 | Volts |
| VDDY (VSEL=1) | TBD | 3.3 | 3.6 | Volts |
| VDDY (VSEL=0) | TBD | 2.5 | TBD | Volts |
| Power: VDDI | TBD | TBD | TBD | mW |
| Power: VDDX+VDDY where VDDX=VDDY | TBD | 75 | TBD | mW |
| IDLE0 Power | | TBD | | mW |
| IDLE1 Power | | TBD | | mW |
| Sleep Current (VDDI = VSS) | | | 50 | uA |

### 11.2.3 CPU Frequency <= 500 MHz

Table 94 shows the power and voltage requirements for the Au1100 with a CPU frequency less than or equal to 500 MHz.

**TABLE 94. DC Parameters for CPU frequency <= 500 MHz**

| Parameter | Min | Typical | Max | Units |
|---|---|---|---|---|
| VDDI | TBD | 1.2 | TBD | Volts |
| VDDX, XPWR12, XPWR32<br>XPWR12 and XPWR32 should be connected to VDDX through the circuit descibed in Chapter 10 | 3.0 | 3.3 | 3.6 | Volts |
| VDDY (VSEL=1) | TBD | 3.3 | 3.6 | Volts |
| VDDY (VSEL=0) | TBD | 2.5 | TBD | Volts |
| Power: VDDI | TBD | TBD | TBD | mW |
| Power: VDDX+VDDY where VDDX=VDDY | TBD | 200 | TBD | mW |
| IDLE0 Power | | TBD | | mW |
| IDLE1 Power | | TBD | | mW |
| Sleep Current (VDDI = VSS) | | | 50 | uA |

## 11.3 AC Parameters

This chapter describes the AC parameters for I/O devices in the Au1100. Each class of output has different capacitive loads. As the capacitance on the load increases the propagation delay will increase. The capacitive load of all I/O other than the SDRAM interface is 50pF.

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 clock. When this is the case the symbol Tsys will be used which is defined in nanoseconds as:

Tsys = 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 for details. The symbol SD should be interpreted as the system bus divider. See section Section 7.4, Power Management for details.

### 11.3.1 SDRAM Timing

All SDRAM outputs are assumed to have a load of 35pF capacitive load, except for the clocks which are 15pF.

Each chip select can support a maximum of two loads.

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” or less.

**TABLE 95. SDRAM Controller Interface**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| SDCLK[n] | Tsdclk | SDCLK[n] Clock Cycle | $\frac{T_{sys}}{2}$ | | ns |
| $\overline{\text{SDCS[n]}}$, $\overline{\text{SDRAS}}$, $\overline{\text{SDCAS}}$, $\overline{\text{SDWE}}$, 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) | Tsdsu | Data setup to SDCLK[n] | 3 | | ns |
| SDD[31:0] (input) | Tsdh | Data hold from SDCLK[n] | 2 | | ns |

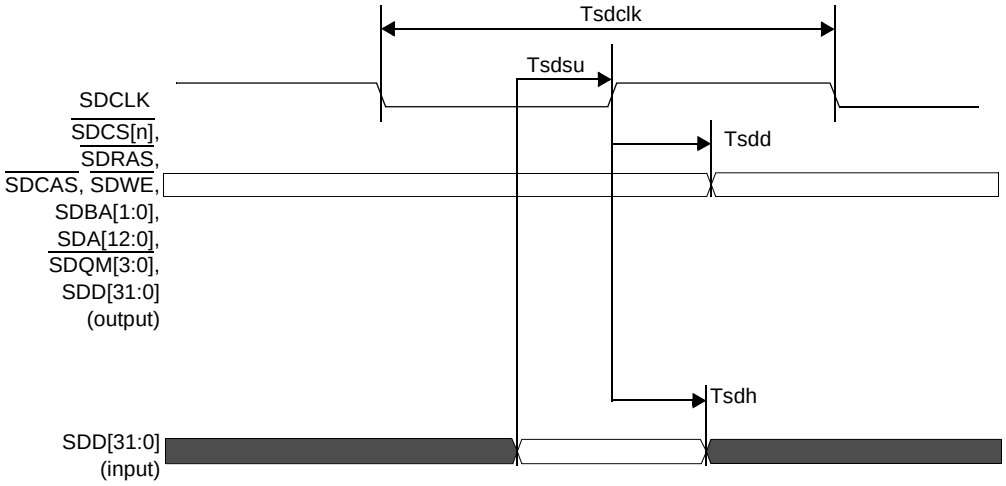



**FIGURE 41. SDRAM Timing**

## 11.3.2 Static Bus Controller Timing

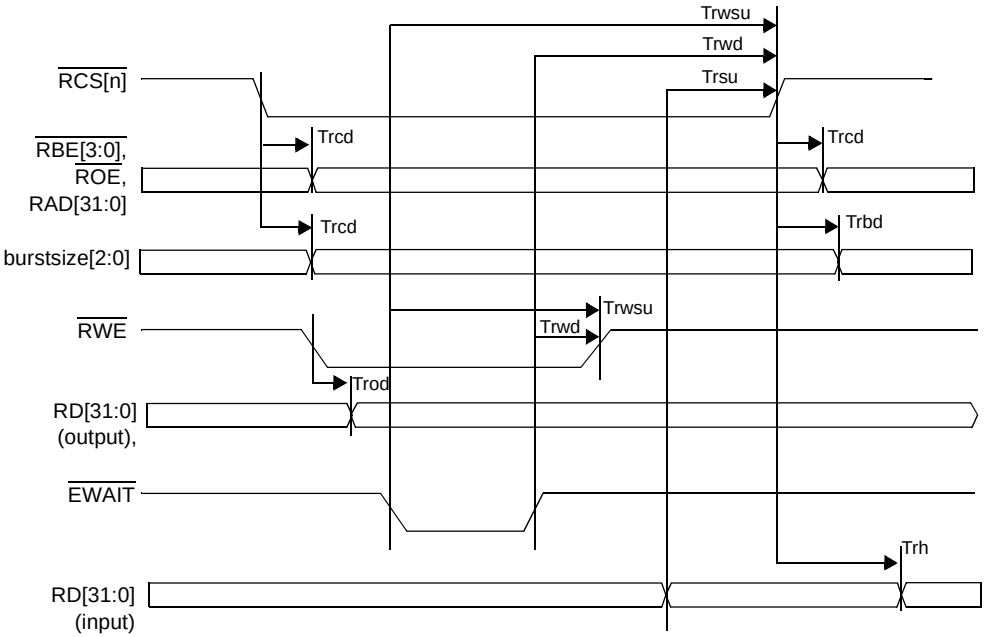
The timing presented in registers **mem_sttime*n*** are not presented here. The parameters in these registers are presented in a certain number of clock cycles and and will be accurate within +/- 2ns.

**TABLE 96. Static Ram, I/O Device and Flash timing**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| RBE[3:0], ROE, RAD[31:0], burstsize[2:0] | Trcd | Delay from RCSn | -2 | +2 | ns |
| RD[31:0] (read) | Trsu | Data setup to RCSn | 5 | | ns |
| RD[31:0] (read) | Trh | Data hold from RCSn | 0 | | ns |
| RD[31:0] (write) | Trod | Delay from RWE to data out | -2 | 2 | ns |

## TABLE 96. Static Ram, I/O Device and Flash timing

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| $\overline{\text{EWAIT}}$ | Trwsu | $\overline{\text{EWAIT}}$ setup to $\overline{\text{RCS[n]}}$, $\overline{\text{RWE}}$<br>If EWAIT does not meet this setup time the cycle will not be held | 3 * Tsys + 15 | | ns |
| $\overline{\text{ROE}}$, $\overline{\text{RWE}}$ | Trwd | $\overline{\text{ROE}}$ and $\overline{\text{RCS[n]}}$ delay from $\overline{\text{EWAIT}}$ | | 3 * Tsys + 15 | |
| burstsize[2:0] | Trbd | Delay from $\overline{\text{RCSn}}$ | | Tsys + 2 | |



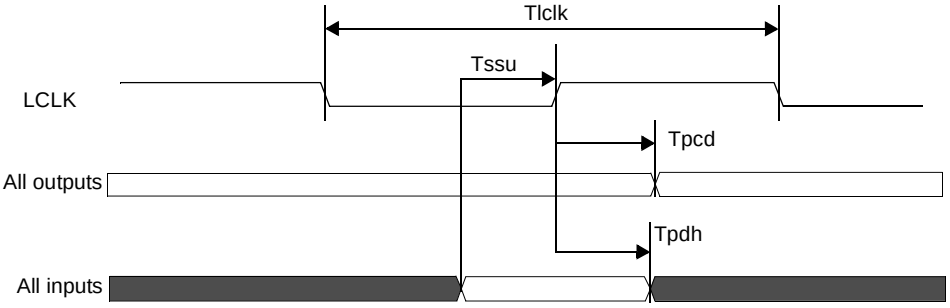

## FIGURE 42. Static Ram, I/O Device and Flash timing

### 11.3.3 Synchronous Static Bus Timing

When the static bus is operating in synchronous mode all timing is reference with respect to LCLK. .

**TABLE 97. PCI Controller Interface**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| LCLK | Tlclk | LCLK Clock Cycle | 20 | | ns |
| All outputs | Tcd | Delay from LCLK | 2 | 10 | ns |
| All inputs | Tsu | Data setup to LCLK | 10 | | ns |
| | Tdh | Data hold from LCLK | 0 | | ns |




### 11.3.4 PCI Timing

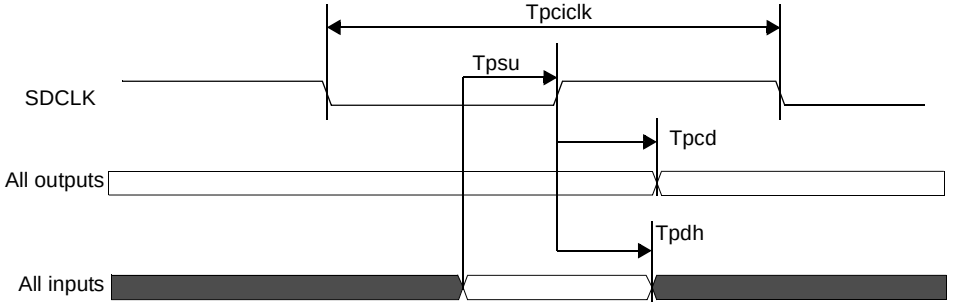
The PCI controller conforms to the PCI 2.2 Local Bus Specification. The PCI timing shown assumes a maximum of 3 PCI loads with no more than one slot. Trace lengths of PCI signals should be kept to a maximum of 9 inches.

**TABLE 98. PCI Controller Interface**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| PCICLK | Tpclk | PCICLK Clock Cycle | 15 | | ns |

### TABLE 98. PCI Controller Interface

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| All outputs | Tpcd | Delay from PCICLK | 2 | 6 | ns |
| All inputs | Tpsu | Data setup to PCICLK | 3 | | ns |
| | Tpdh | Data hold from PCICLK | 0 | | ns |




### TABLE 99. PCMCIA Timing

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| $\overline{\text{PREG}}$, RAD[31:0], RD[31:0] (output) | Tpcd | Delay from $\overline{\text{PCE[n]}}$ | -2 | +2 | ns |
| $\overline{\text{PIOS16}}$ | Tpios | $\overline{\text{PIOS16}}$ setup to $\overline{\text{PIOR}}$, $\overline{\text{PIOW}}$ | TBD | | ns |
| $\overline{\text{PIOS16}}$ | Tpioh | $\overline{\text{PIOS16}}$ hold from $\overline{\text{PIOR}}$, $\overline{\text{PIOW}}$ | TBD | | ns |
| $\overline{\text{OE}}$ | Tpoed | $\overline{\text{OE}}$ delay from $\overline{\text{POE}}$, $\overline{\text{PIOR}}$ | -2 | +2 | ns |
| RD[31:0] (input) | Tpsu | Data setup to $\overline{\text{POE}}$, $\overline{\text{PIOR}}$ | Tsys + 15 | | ns |

**TABLE 99. PCMCIA Timing**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| RD[31:0] | Tph | Data hold from $\overline{\text{POE}}$, $\overline{\text{PIOR}}$ | 0 | | ns |
| $\overline{\text{PWAIT}}$ | Tpwsu | $\overline{\text{PWAIT}}$ setup to $\overline{\text{POE}}$, $\overline{\text{PWE}}$, $\overline{\text{PIOR}}$, $\overline{\text{PIOW}}$<br>If $\overline{\text{PWAIT}}$ does not meet this setup time the cycle will not be held | 4 * Tsys + 15 | | ns |
| $\overline{\text{POE}}$, $\overline{\text{PWE}}$, $\overline{\text{PIOR}}$, $\overline{\text{PIOW}}$ | Tpwd | $\overline{\text{POE}}$, $\overline{\text{PWE}}$, $\overline{\text{PIOR}}$, $\overline{\text{PIOW}}$ delay from $\overline{\text{PWAIT}}$. | | 4 * Tsys + 2 | ns |

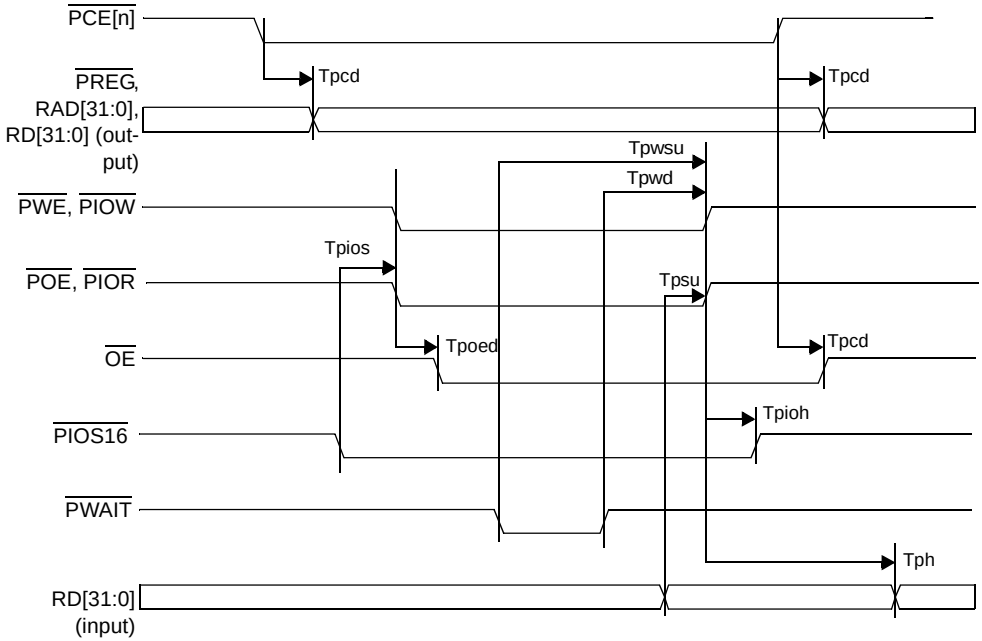



**FIGURE 43. PCMCIA Host Adapter Timing**

### TABLE 100.LCD Timing

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| LCLK | Tlclk | LCLK Period<br>This parameter is set with the D5 bit in the mem_stcfg0. Please see Section 3.2, Static Bus Controller for information | Tsys/5 | Tsys/4 | ns |
| $\overline{\text{RCS[n]}}$, RAD[31:0], RD[31:0] (output) | Tlcd | Delay from LCLK | -2 | 2 | ns |
| RD[31:0] (input) | Tlsu | Data setup to $\overline{\text{LRD[n]}}$ | Tsys + 15 | | ns |
| RD[31:0] (input) | Tlh | Data hold from $\overline{\text{LRD[n]}}$ | 0 | | ns |
| $\overline{\text{LWAIT}}$ | Tlwsu | $\overline{\text{LWAIT}}$ setup to $\overline{\text{LRD[n]}}$, $\overline{\text{LWR[n]}}$.<br>If $\overline{\text{LWAIT}}$ does not meet this setup time the cycle will not be held | 4 * Tsys + Tlclk + 15 | | |
| $\overline{\text{LRD[n]}}$, $\overline{\text{LWR[n]}}$ | Tlwd | $\overline{\text{LRD[n]}}$, $\overline{\text{LWR[n]}}$ delay from $\overline{\text{LWAIT}}$ | | 4 * Tsys + Tlclk + 15 | |

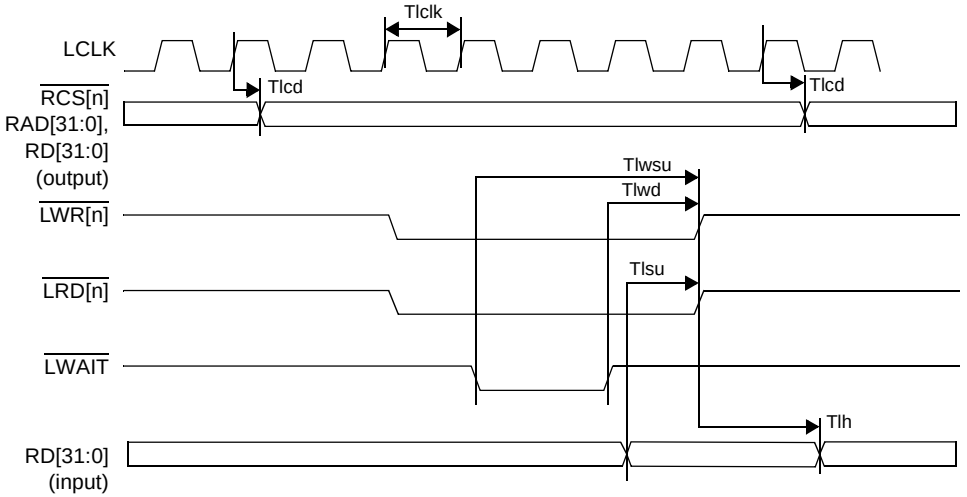



**FIGURE 44. LCD Interface Timing**

## 11.3.5 Peripheral Timing

### TABLE 101.Ethernet MII Interface

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| N0TXCLK<br>N0RXCLK<br>N1TXCLK<br>N1RXCLK | $T_{eth}$ | Ethernet Transmit/ Receive Clock Cycle Time (25% of data rate) | 40 +/- 100ppm (for 100Mbs)<br>400 +/- 100ppm (for 10Mbs) | | ns |
| | | Ethernet Transmit/ Receive Clock Duty Cycle | 35 | 65 | % |
| N0TXEN,<br>N0TXD[3:0]<br>N1TXEN<br>N1TXD[3:0] | Ted | Delay from TXCLK to TXEN, TXD[3:0] | 0 | 25 | ns |
| N0RXD[3:0]<br>N0RXDV<br>N1RXD[3:0]<br>N1RXDV | Tesu | Setup Time before RXC-CLK for RXD, and RXDV | 10 | | ns |
| | Teh | Hold time from RXCLK for RXD, and RXDV | 10 | | ns |
| N0MDC,<br>N1MDC | Tmdc | MDC cycle time | system bus clock / 160 | | |
| | | MDC Duty Cycle | 40 | 60 | % |
| N0MDIO,<br>N1MDIO | Tmdd | Delay from MDC to MDIO | 0 | 300 | ns |
| | Tmsu | Setup time before MDC for MDIO | 10 | | ns |
| | Tmh | Hold time from MDC for MDIO | 10 | | ns |
| | Tmz | Delay from MDC to MDIO tristate | 0 | 300 | ns |
| N0CRS,<br>N0COL,<br>N1CRS,<br>N1COL | Ta | Minimum active time | | | |

*Note: Both Ethernet MACs offer identical timing. The Nn prefix on each of the signals should be taken as referring to MAC N0 or N1 with all timing relative to signals within the same MAC.*

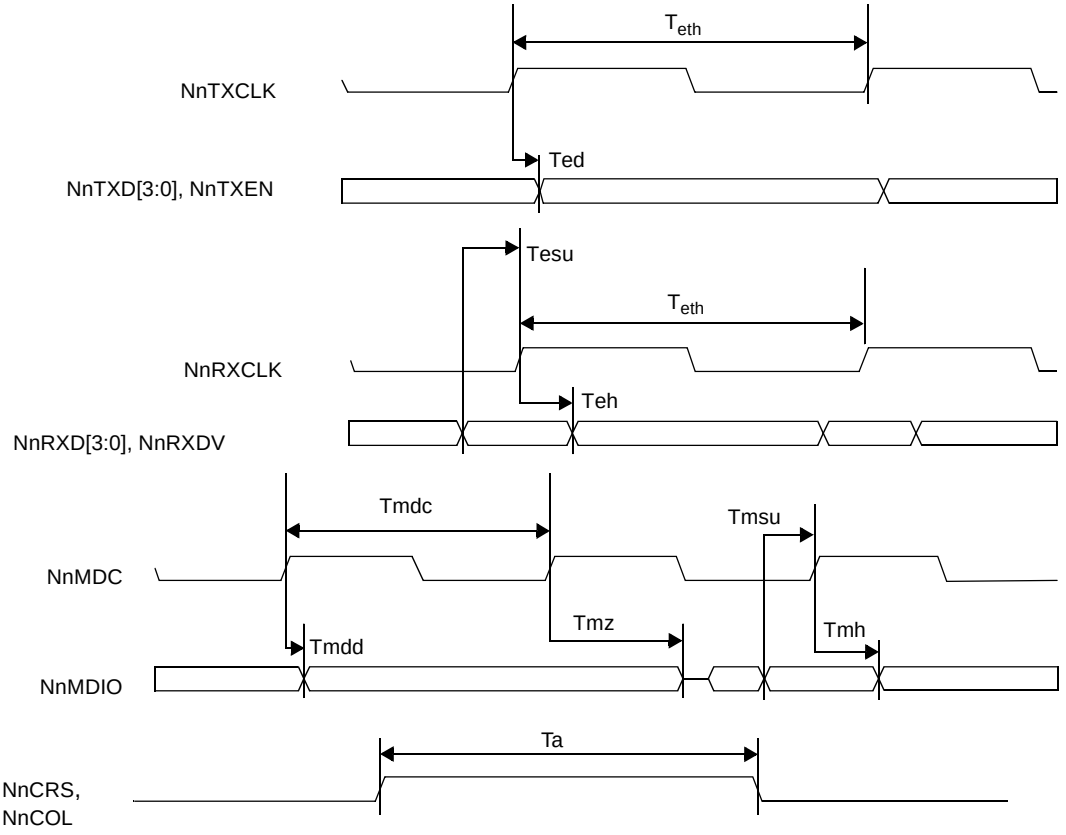



**FIGURE 45. MII Interface Timing**

**TABLE 102.AC-Link Interface**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| 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 |

**TABLE 102.AC-Link Interface**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| ACSYNC | Tacs | AC97 sync cycle | 48 (typical) | | kHz |
| | Tacsh | AC97 sync high time | 1.3 (typical) | | us |
| | Tacsl | AC97 sync low time | 19.5 (typical) | | us |
| 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* *$\overline{ACRST}$ is an Asynchronous signal controlled by software through the register ac97_config. It has no relationship to other AC97 signals.*

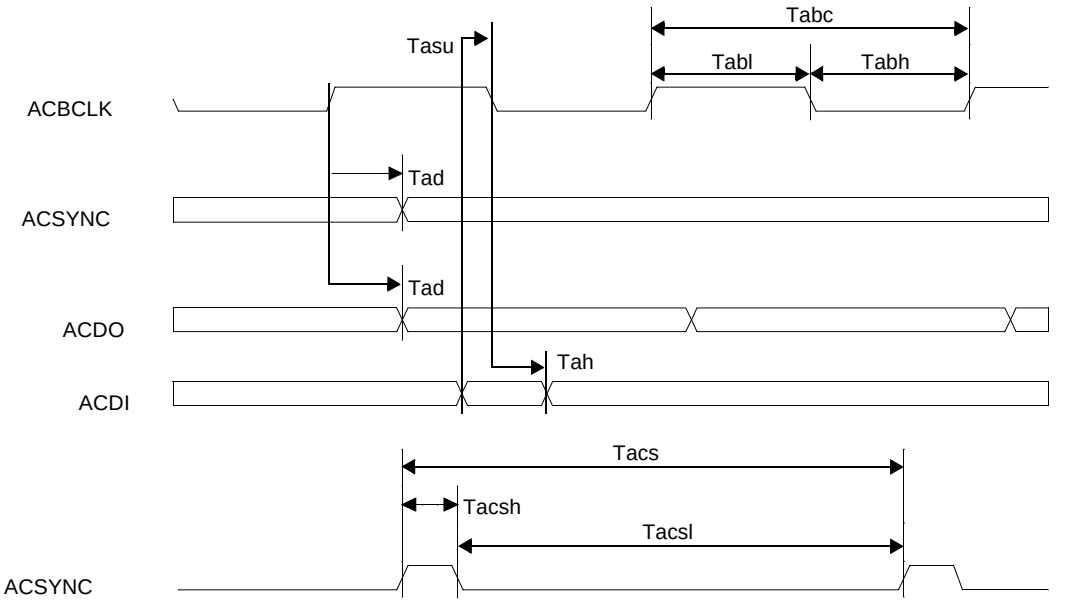



**FIGURE 46. AC-Link Timing**

**TABLE 103.GPIO Timing for interrupt**

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| GPIO[n] | Tmin | Minimum high or low time for interrupt.<br>The level is programmable, this timing reflects the minimum active period for the level programmed | 10 | | ns |

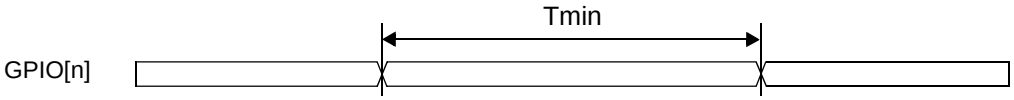



**FIGURE 47. GPIO Interrupt Timing**

### TABLE 104.EJTAG Interface

| Signal | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| TCK | Tec | EJTAG TCK cycle time<br>*see note below | 35 | | ns |
| | Tech | TCK high time | 10 | | ns |
| | Tecl | TCK low time | 10 | | ns |
| TMS<br>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 |

*Note: $T_{ec}$ minimum is the greater of 25ns or 1/4 the period of the system bus clock.*

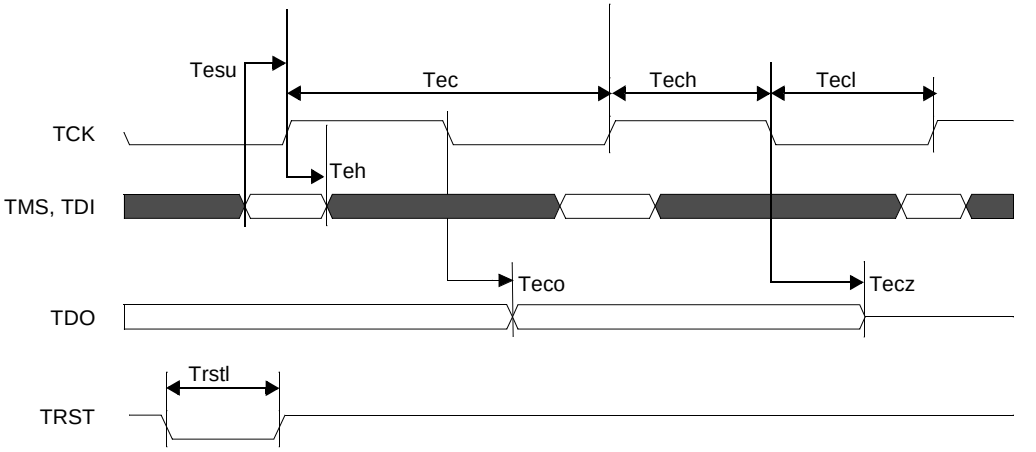



**FIGURE 48. EJTAG Timing**

## 11.4 Asynchronous Signals

**GPIO**

The GPIO signals are driven by software.

**Reset**

Reset timing and all signals relative to reset are shown in the Reset section.

**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.

## 11.5 Crystal Specifications

Table 105 provides the specification for the parallel resonant 12MHz crystal to be placed between XTI12 and XTO12. Load capacitors for this oscillator are integrated into the Au1100.

### TABLE 105.12MHz Crystal Specification

| Specification | Min | Typ | Max | Unit |
|---|---|---|---|---|
| Resonant Frequency | 11 | 12 | 15 | MHz |
| Motional Resistance | | | 60 | Ohms |
| Shunt Capacitance | | <5 | 7 | pF |
| Load Capacitance (this capacitance is integrated on the Au1100) | 8 | 12 | 20 | pF |
| Drive Level | | | 100 | uW |
| Crystal Type | AT Cut | | | |

Table 106 provides the specification for the parallel resonant 32MHz crystal to be placed between XTI32 and XTO32. Load capacitors for this oscillator are integrated into the Au1100 so no external circuitry is required when using the specified crystal.

### TABLE 106.32.768kHz 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 (this capacitance is integrated on the Au1100) | 6 | | 12 | pF |
| Motional Capacitance | | 3 | 4 | fF |
| Drive Level | | | 1 | uW |
| Quality Factor | 40k | | | |
| Crystal Type | Tuning Fork | | | |

# 12 Packaging and Pinout

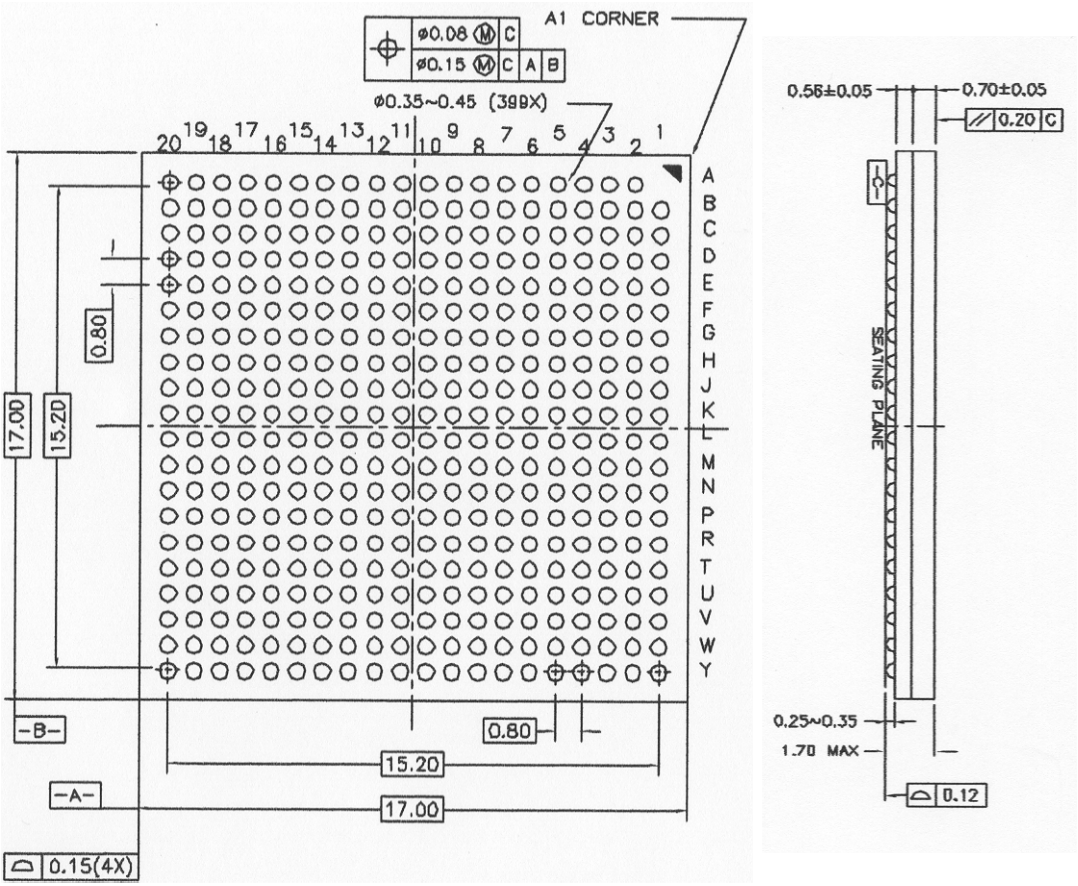



**FIGURE 49.Package Dimensions: Bottom View (left) and Side View (right)**

## TABLE 107.Pin Placement Expanded View (table 1 of 3)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| A | N/C | XTI32 | XPWR12 | XTO12 | XTI12 | TC3 | PWE (GP207) |
| B | XPWR32 | XAGND32 | XTO32 | ROMSEL | XAGND12 | TC0 | PIOWN |
| C | VSEL | TMS | TSTEN | TRSTN | CVDDXOK | GP00 | TC1 |
| D | SDA11 | CRSTON | SDA12 | ROMSIZ | CRSTN | GP01 | GP04 |
| E | SDA5 | SDA6 | SDA9 | SDA8 | SDA10 | CPWREN | GP02 (EXTCLK0) |
| F | SDA3 | SDA1 | SDA2 | SDA4 | SDA7 | VSSI | VDDI |
| G | SDQM3 | SDQM2 | SDA0 | SDBA0 | SDBA1 | VSSI | VDDY |
| H | SDWEN | SDQM1 | SDQM0 | SDCASN | SDRASN | VDDI | VDDY |
| J | SDCSN2 | SDCKE | VDDY | SDCLK2 | VSSX | VDDI | VDDY |
| K | SDCSN0 | SDCSN1 | VSSX | SDCLK1 | VDDY | VSSI | VDDY |
| L | SDD31 | SDD28 | VDDY | SDCLK0 | VSSX | VSSI | VDDY |
| M | SDD30 | SDD27 | SDD24 | SDD25 | SDD26 | VDDI | VDDY |
| N | SDD29 | SDD21 | SDD20 | SDD22 | SDD23 | VDDI | VDDY |
| P | SDD17 | SDD15 | SDD18 | SDD19 | SDD14 | VSSI | VDDY |
| R | SDD16 | SDD11 | SDD12 | SDD13 | SDD10 | VSSI | VDDI |
| T | SDD9 | SDD7 | SDD8 | SDMS1_DAT1 | VSSX | SDMS0_DAT0 | SDMS0_CLK |
| U | SDD6 | SDD1 | SDD2 | SDD3 | SDMS_MS_EN | LCD_PWM1 | GP22 |
| V | SDD5 | SDD0 | SDMS1_DAT3 | SDMS0_DAT1 | SDMS0_CMD | LCD_PWM0 | GP23 |
| W | SDD4 | SDMS1_DAT2 | SDMS1_CMD | SDMS1_CLK | RESVD_4 | GP17 | RESVD_1 |
| Y | SDMS1_DAT0 | SDMS0_DAT3 | SDMS0_DAT2 | RESVD_5 | RESVD_3 | GP18 | GP16 |

## TABLE 108.Pin Placement Expanded View (table 2 of 3)

| | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|
| A | PREG (GP204) | PWAITN | EWAIT | LWR1 (GP203) | RD4 | RD9 | RD11 |
| B | POEN | PIOS16N | PCE1 (GP206) | LRD0 (GP200) | RD0 | RD6 | RD10 |
| C | GP05 | PIORN | LRD1 (GP201) | LWR0 (GP202) | RD5 | RD8 | RD12 |
| D | TC2 | GP07_F | PCE0 (GP205) | VSSX | LWAITN | RD2 | RD3 |
| E | GP03 (EXTCLK1) | GP06_F (SROMCKE) | VSSX | LCLK | RD1 | RD14 | RD7 |
| F | VDDI | VSSI | VSSI | VDDI | VDDI | VSSI | VSSI |
| G | VDDY | VDDX | VDDX | VDDX | VDDX | VDDX | VDDX |
| H | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| J | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| K | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| L | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| M | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| N | VSSX | VSSX | VSSX | VSSX | VSSX | VSSX | VDDX |
| P | VDDX | VDDX | VDDX | VDDX | VDDX | VDDX | VDDX |
| R | VDDI | VSSI | VSSI | VDDI | VDDI | VSSI | VSSI |
| T | RESVD_2 | GP19 | N0RXD3 | U3DTR (GP14) | N0RXD2 | ACDI | U1RXD |
| U | N0TXCLK | N0TXD2 (GP27) | N0TXEN (GP24) | N0RXD1 | I2SCLK (GP30) | ACBCLK (S1DIN) | ACRST (S1DEN) |
| V | GP21 | N0TXD3 (GP28) | N0TXD1 (GP26) | IRFIRSEL (GP15) | U1TXD (GP213) | I2SWRD (GP31) | ACSYNC (S1DOUT) |
| W | RESVD_0 | N0MDC (GP215) | N0COL | N0RXCLK | U3RTS (GP13) | N0RXD0 | ACDO (S1CLK) |
| Y | GP20 | N0MDIO | N0RXDV | N0TXD0 (GP25) | U3TXD (GP214) | U0TXD (GP212) | N0CRS |

## TABLE 109.Pin Placement Expanded View (table 3 of 3)

| | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|
| A | RD16 | RD17 | RD24 | RD25 | RD26 | RD28 |
| B | RD15 | RD18 | RD22 | RD27 | RD31 | RAD2 |
| C | RD19 | RD21 | RD23 | RD29 | RAD1 | RAD10 |
| D | RD13 | RD20 | RD30 | RAD0 | RAD5 | RAD11 |
| E | VDDI | VSSX | RAD3 | RAD4 | RAD9 | RAD15 |
| F | VDDI | VDDI | RAD6 | RAD8 | RAD16 | RAD19 |
| G | VDDI | RAD12 | RAD7 | RAD14 | RAD20 | RAD21 |
| H | VSSI | RAD18 | RAD13 | RAD17 | RAD22 | RAD27 |
| J | VSSI | RAD25 | RAD23 | RAD24 | RAD26 | RAD28 |
| K | VDDI | RBEN0 | RAD31 | RAD30 | RBEN2 | RAD29 |
| L | VDDI | VDDX | RBEN3 | RWEN | ROEN | RBEN1 |
| M | VSSI | LCD_PCK | LCD_LCK | LCD_D5 | RCSN1 | RCSN0 |
| N | VSSI | VSSX | LCD_D4 | LCD_D3 | LCD_D2 | RCSN2 |
| P | VDDI | LCD_D6 | LCD_FCK | LCD_D11 | LCD_D10 | RCSN3 |
| R | VDDI | VDDI | LCD_D7 | LCD_D8 | LCD_BIAS | LCD_D0 |
| T | VDDI | VSSX | LCD_LEND | LCD_D9 | LCD_D12 | LCD_D1 |
| U | S0DE (GP210) | I2SDI (GP08) | U3CTS (GP09) | TDO | TDI | LCD_D13 |
| V | IRDATX (GP211) | U3RI (GP12) | U3DCD (GP11) | TCK | USBH1M | LCD_D14 |
| W | I2SDIO (GP29) | S0CLK (GP209) | USBDP (USBH0P) | S0DIN | U0RXD | LCD_D15 |
| Y | U3RXD | IRDARX | S0DOUT (GP208) | USBDM (USBH0M) | U3DSR (GP10) | USBH1P |

# A Memory Map

The Au1100 is a collection of several devices. The devices contain software visible registers that are memory mapped. Table 110 contains the memory map for the Au1100 peripheral devices and physical memory. The addresses are 36 bits wide.

**TABLE 110.Basic Au1100 Physical Memory Map**

| Start Address | End Address | Size (MB) | Function |
|---|---|---|---|
| 0x0 00000000 | 0x0 0FFFFFFF | 256 | Memory KSEG 0/1 |
| 0x0 10000000 | 0x0 11FFFFFF | 32 | I/O Devices on Peripheral Bus |
| 0x0 12000000 | 0x0 13FFFFFF | 32 | Reserved |
| 0x0 14000000 | 0x0 17FFFFFF | 64 | I/O Devices on System Bus |
| 0x0 18000000 | 0x0 1FFFFFFF | 128 | Memory Mapped<br>0x0 1FC00000 must contain the boot vector so this is typically where flash or ROM is located. |
| 0x0 20000000 | 0x0 7FFFFFFF | 1536 | Memory Mapped |
| 0x0 80000000 | 0x0 EFFFFFFF | 1792 | Memory Mapped<br>Currently this space is memory mapped but it should be considered reserved for future use. |
| 0x0 F0000000 | 0x0 FFFFFFFF | 256 | Debug Probe |
| 0x1 00000000 | 0xC FFFFFFFF | 4096 | Reserved |
| 0xD 00000000 | 0xD FFFFFFFF | 4096 | I/O Device |
| 0xE 00000000 | 0xE FFFFFFFF | 4096 | External LCD Controller Interface |
| 0xF 00000000 | 0xF FFFFFFFF | 4096 | PCMCIA Interface |

The Au1100 System bus devices are mapped at the addresses based at 0x0 14000000. See Table 111 for complete addresses.

**TABLE 111.System Bus Devices Physical Memory Map**

| Start Address | End Address | Size | Function |
|---|---|---|---|
| 0x0 14000000 | 0x0 14000FFF | 512KB | SDRAM Memory Controller |
| 0x0 14001000 | 0x0 14001FFF | 512KB | SRAM/FLASH Memory Controller |
| 0x0 14002000 | 0x0 14002FFF | 512KB | DMA |
| 0x0 14004000 | 0x0 14004FFF | 512KB | Ethernet DMA |
| 0x0 15000000 | 0x0 14007FFF | 512KB | LCD Controller |

The Au1100 peripheral bus devices are based at 0x0 11000000. The individual memory spaces of those devices are defined in Table 112.

**TABLE 112.Peripheral Bus Devices Physical Memory Map**

| Start Address | End Address | Size | Function |
|---|---|---|---|
| 0x0 10000000 | 0x0 100FFFFF | 1MB | AC97 Controller |
| 0x0 10100000 | 0x0 101FFFFF | 1MB | USB Host |
| 0x0 10200000 | 0x0 102FFFFF | 1MB | USB Device |
| 0x0 10300000 | 0x0 103FFFFF | 1MB | IrDA |
| 0x0 10400000 | 0x0 104FFFFF | 1MB | Interrupt Controller 0 |
| 0x0 10500000 | 0x0 105FFFFF | 1MB | Ethernet MAC |
| 0x0 10600000 | 0x0 106FFFFF | 1MB | SD Controller |
| 0x0 10700000 | 0x0 10FFFFFF | 9MB | |
| 0x0 11000000 | 0x0 110FFFFF | 1MB | $I^2S$ |
| 0x0 11100000 | 0x0 111FFFFF | 1MB | UART0 |
| 0x0 11200000 | 0x0 112FFFFF | 1MB | UART1 |
| 0x0 11300000 | 0x0 113FFFFF | 1MB | |
| 0x0 11400000 | 0x0 114FFFFF | 1MB | UART3 |
| 0x0 11500000 | 0x0 115FFFFF | 1MB | |
| 0x0 11600000 | 0x0 116FFFFF | 1MB | SSI |
| 0x0 11700000 | 0x0 117FFFFF | 1MB | Secondary GPIO |
| 0x0 11800000 | 0x0 118FFFFF | 1MB | Interrupt Controller 1 |
| 0x0 11900000 | 0x0 119FFFFF | 1MB | System Control - RTC, TOY, Timers, Primary GPIO, Power Management |

## Programming Tips

## 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 for more information.

## Device Memory Map

Table 113 lists all of the devices which are memory mapped to the Au1100 core CPU. 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 113. Device Memory Map

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| **SDRAM Controller** | | | Section 3.1 |
| mem_sdmode0 | 0xB4000000 | 0x0 14000000 | |
| mem_sdmode1 | 0xB4000004 | 0x0 14000004 | |
| mem_sdmode2 | 0xB4000008 | 0x0 14000008 | |
| mem_sdaddr0 | 0xB400000c | 0x0 1400000c | |
| mem_sdaddr1 | 0xB4000010 | 0x0 14000010 | |
| mem_sdaddr2 | 0xB4000014 | 0x0 14000014 | |
| mem_sdrefcfg | 0xB4000018 | 0x0 14000018 | |
| mem_sdprecmd | 0xB400001c | 0x0 1400001c | |
| mem_sdautoref | 0xB4000020 | 0x0 14000020 | |
| mem_sdwrmd0 | 0xB4000024 | 0x0 14000024 | |
| mem_sdwrmd1 | 0xB4000028 | 0x0 14000028 | |
| mem_sdwrmd2 | 0xB400002C | 0x0 1400002C | |
| mem_sdsleep | 0xB4000030 | 0x0 14000030 | |
| mem_sdsmcke | 0xB4000034 | 0x0 14000034 | |
| **Static Bus Controller** | | | Section 3.2 |
| mem_stcfg0 | 0xB4001000 | 0x0 14001000 | |
| mem_sttime0 | 0xB4001004 | 0x0 14001004 | |
| mem_staddr0 | 0xB4001008 | 0x0 14001008 | |
| mem_stcfg1 | 0xB4001010 | 0x0 14001010 | |
| mem_sttime1 | 0xB4001014 | 0x0 14001014 | |
| mem_staddr1 | 0xB4001018 | 0x0 14001018 | |
| mem_stcfg2 | 0xB4001020 | 0x0 14001020 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| mem_sttime2 | 0xB4001024 | 0x0 14001024 | |
| mem_staddr2 | 0xB4001028 | 0x0 14001028 | |
| mem_stcfg3 | 0xB4001030 | 0x0 14001030 | |
| mem_sttime3 | 0xB4001034 | 0x0 14001034 | |
| mem_staddr3 | 0xB4001038 | 0x0 14001038 | |
| **DMA Controller 0** | | | Chapter 4 |
| dma_moderead | 0xB4002000 | 0x0 14002000 | |
| dma_modeset | 0xB4002000 | 0x0 14002000 | |
| dma_modeclr | 0xB4002004 | 0x0 14002004 | |
| dma_peraddr | 0xB4002008 | 0x0 14002008 | |
| dma_buf0addr | 0xB400200c | 0x0 1400200c | |
| dma_buf0size | 0xB4002010 | 0x0 14002010 | |
| dma_buf1addr | 0xB4002014 | 0x0 14002014 | |
| dma_buf1size | 0xB4002018 | 0x0 14002018 | |
| **DMA Controller 1** | | | Chapter 4 |
| dma_moderead | 0xB4002100 | 0x0 14002100 | |
| dma_modeset | 0xB4002100 | 0x0 14002100 | |
| dma_modeclr | 0xB4002104 | 0x0 14002104 | |
| dma_peraddr | 0xB4002108 | 0x0 14002108 | |
| dma_buf0addr | 0xB400210c | 0x0 1400210c | |
| dma_buf0size | 0xB4002110 | 0x0 14002110 | |
| dma_buf1addr | 0xB4002114 | 0x0 14002114 | |
| dma_buf1size | 0xB4002118 | 0x0 14002118 | |
| **DMA Controller 2** | | | Chapter 4 |
| dma_moderead | 0xB4002200 | 0x0 14002200 | |
| dma_modeset | 0xB4002200 | 0x0 14002200 | |
| dma_modeclr | 0xB4002204 | 0x0 14002204 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| dma_peraddr | 0xB4002208 | 0x0 14002208 | |
| dma_buf0addr | 0xB400220c | 0x0 1400220c | |
| dma_buf0size | 0xB4002210 | 0x0 14002210 | |
| dma_buf1addr | 0xB4002214 | 0x0 14002214 | |
| dma_buf1size | 0xB4002218 | 0x0 14002218 | |
| **DMA Controller 3** | | | Chapter 4 |
| dma_moderead | 0xB4002300 | 0x0 14002300 | |
| dma_modeset | 0xB4002300 | 0x0 14002300 | |
| dma_modeclr | 0xB4002304 | 0x0 14002304 | |
| dma_peraddr | 0xB4002308 | 0x0 14002308 | |
| dma_buf0addr | 0xB400230c | 0x0 1400230c | |
| dma_buf0size | 0xB4002310 | 0x0 14002310 | |
| dma_buf1addr | 0xB4002314 | 0x0 14002314 | |
| dma_buf1size | 0xB4002318 | 0x0 14002318 | |
| **DMA Controller 4** | | | Chapter 4 |
| dma_moderead | 0xB4002400 | 0x0 14002400 | |
| dma_modeset | 0xB4002400 | 0x0 14002400 | |
| dma_modeclr | 0xB4002404 | 0x0 14002404 | |
| dma_peraddr | 0xB4002408 | 0x0 14002408 | |
| dma_buf0addr | 0xB400240c | 0x0 1400240c | |
| dma_buf0size | 0xB4002410 | 0x0 14002410 | |
| dma_buf1addr | 0xB4002414 | 0x0 14002414 | |
| dma_buf1size | 0xB4002418 | 0x0 14002418 | |
| **DMA Controller 5** | | | Chapter 4 |
| dma_moderead | 0xB4002500 | 0x0 14002500 | |
| dma_modeset | 0xB4002500 | 0x0 14002500 | |
| dma_modeclr | 0xB4002504 | 0x0 14002504 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| dma_peraddr | 0xB4002508 | 0x0 14002508 | |
| dma_buf0addr | 0xB400250c | 0x0 1400250c | |
| dma_buf0size | 0xB4002510 | 0x0 14002510 | |
| dma_buf1addr | 0xB4002514 | 0x0 14002514 | |
| dma_buf1size | 0xB4002518 | 0x0 14002518 | |
| **DMA Controller 6** | | | Chapter 4 |
| dma_moderead | 0xB4002600 | 0x0 14002600 | |
| dma_modeset | 0xB4002600 | 0x0 14002600 | |
| dma_modeclr | 0xB4002604 | 0x0 14002604 | |
| dma_peraddr | 0xB4002608 | 0x0 14002608 | |
| dma_buf0addr | 0xB400260c | 0x0 1400260c | |
| dma_buf0size | 0xB4002610 | 0x0 14002610 | |
| dma_buf1addr | 0xB4002614 | 0x0 14002614 | |
| dma_buf1size | 0xB4002618 | 0x0 14002618 | |
| **DMA Controller 7** | | | Chapter 4 |
| dma_moderead | 0xB4002700 | 0x0 14002700 | |
| dma_modeset | 0xB4002700 | 0x0 14002700 | |
| dma_modeclr | 0xB4002704 | 0x0 14002704 | |
| dma_peraddr | 0xB4002708 | 0x0 14002708 | |
| dma_buf0addr | 0xB400270c | 0x0 1400270c | |
| dma_buf0size | 0xB4002710 | 0x0 14002710 | |
| dma_buf1addr | 0xB4002714 | 0x0 14002714 | |
| dma_buf1size | 0xB4002718 | 0x0 14002718 | |
| **Interrupt Controller 0** | | | Chapter 5 |
| ic_cfg0rd | 0xB0400040 | 0x0 10400040 | |
| ic_cfg0set | 0xB0400040 | 0x0 10400040 | |
| ic_cfg0clr | 0xB0400044 | 0x0 10400044 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| ic_cfg1rd | 0xB0400048 | 0x0 10400048 | |
| ic_cfg1set | 0xB0400048 | 0x0 10400048 | |
| ic_cfg1clr | 0xB040004C | 0x0 1040004C | |
| ic_cfg2rd | 0xB0400050 | 0x0 10400050 | |
| ic_cfg2set | 0xB0400050 | 0x0 10400050 | |
| ic_cfg2clr | 0xB0400054 | 0x0 10400054 | |
| ic_req0int | 0xB0400054 | 0x0 10400054 | |
| ic_srcrd | 0xB0400058 | 0x0 10400058 | |
| ic_srcset | 0xB0400058 | 0x0 10400058 | |
| ic_srcclr | 0xB040005C | 0x0 1040005C | |
| ic_req1int | 0xB040005C | 0x0 1040005C | |
| ic_assignrd | 0xB0400060 | 0x0 10400060 | |
| ic_assignset | 0xB0400060 | 0x0 10400060 | |
| ic_assignclr | 0xB0400064 | 0x0 10400064 | |
| ic_wakerd | 0xB0400068 | 0x0 10400068 | |
| ic_wakeset | 0xB040006C | 0x0 1040006C | |
| ic_wakeclr | 0xB0400070 | 0x0 10400070 | |
| ic_maskrd | 0xB0400070 | 0x0 10400070 | |
| ic_maskset | 0xB0400074 | 0x0 10400074 | |
| ic_maskclr | 0xB0400078 | 0x0 10400078 | |
| ic_risingrd | 0xB0400078 | 0x0 10400078 | |
| ic_risingclr | 0xB040007C | 0x0 1040007C | |
| ic_fallingrd | 0xB040007C | 0x0 1040007C | |
| ic_fallingclr | 0xB0400080 | 0x0 10400080 | |
| **Interrupt Controller 1** | | | Chapter 5 |
| ic_cfg0rd | 0xB1800040 | 0x0 11800040 | |
| ic_cfg0set | 0xB1800040 | 0x0 11800040 | |
| ic_cfg0clr | 0xB1800044 | 0x0 11800044 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| ic_cfg1rd | 0xB1800048 | 0x0 11800048 | |
| ic_cfg1set | 0xB1800048 | 0x0 11800048 | |
| ic_cfg1clr | 0xB180004C | 0x0 1180004C | |
| ic_cfg2rd | 0xB1800050 | 0x0 11800050 | |
| ic_cfg2set | 0xB1800050 | 0x0 11800050 | |
| ic_cfg2clr | 0xB1800054 | 0x0 11800054 | |
| ic_req0int | 0xB1800054 | 0x0 11800054 | |
| ic_srcrd | 0xB1800058 | 0x0 11800058 | |
| ic_srcset | 0xB1800058 | 0x0 11800058 | |
| ic_srcclr | 0xB180005C | 0x0 1180005C | |
| ic_req1int | 0xB180005C | 0x0 1180005C | |
| ic_assignrd | 0xB1800060 | 0x0 11800060 | |
| ic_assignset | 0xB1800060 | 0x0 11800060 | |
| ic_assignclr | 0xB1800064 | 0x0 11800064 | |
| ic_wakerd | 0xB1800068 | 0x0 11800068 | |
| ic_wakeset | 0xB180006C | 0x0 1180006C | |
| ic_wakeclr | 0xB1800070 | 0x0 11800070 | |
| ic_maskrd | 0xB1800070 | 0x0 11800070 | |
| ic_maskset | 0xB1800074 | 0x0 11800074 | |
| ic_maskclr | 0xB1800078 | 0x0 11800078 | |
| ic_risingrd | 0xB1800078 | 0x0 11800078 | |
| ic_risingclr | 0xB180007C | 0x0 1180007C | |
| ic_fallingrd | 0xB180007C | 0x0 1180007C | |
| ic_fallingclr | 0xB1800080 | 0x0 11800080 | |
| **AC97 Controller** | | | Section 6.1 |
| ac97_config | 0xB0000000 | 0x0 10000000 | |
| ac97_status | 0xB0000004 | 0x0 10000004 | |
| ac97_data | 0xB0000008 | 0x0 10000008 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| ac97_cmmd | 0xB000000C | 0x0 1000000C | |
| ac97_cmmdresp | 0xB000000C | 0x0 1000000C | |
| ac97_control | 0xB0000010 | 0x0 10000010 | |
| **USB Host Controller** | | | Section 6.2 |
| Open HCI Register Set Base | 0xB0100000 | 0x0 10100000 | |
| usbh_enable | 0xB017FFFC | 0x0 1017FFFC | |
| **USB Device Controller** | | | Section 6.3 |
| usbd_ep0rd | 0xB0200000 | 0x0 10200000 | |
| usbd_ep0wr | 0xB0200004 | 0x0 10200004 | |
| usbd_ep2wr | 0xB0200008 | 0x0 10200008 | |
| usbd_ep3wr | 0xB020000c | 0x0 1020000c | |
| usbd_ep4rd | 0xB0200010 | 0x0 10200010 | |
| usbd_ep5rd | 0xB0200014 | 0x0 10200014 | |
| usbd_inten | 0xB0200018 | 0x0 10200018 | |
| usbd_intstat | 0xB020001c | 0x0 1020001c | |
| usbd_config | 0xB0200020 | 0x0 10200020 | |
| usbd_ep0cs | 0xB0200024 | 0x0 10200024 | |
| usbd_ep2cs | 0xB0200028 | 0x0 10200028 | |
| usbd_ep3cs | 0xB020002c | 0x0 1020002c | |
| usbd_ep4cs | 0xB0200030 | 0x0 10200030 | |
| usbd_ep5cs | 0xB0200034 | 0x0 10200034 | |
| usbd_ep0rdstat | 0xB0200040 | 0x0 10200040 | |
| usbd_ep0wrstat | 0xB0200044 | 0x0 10200044 | |
| usbd_ep2wrstat | 0xB0200048 | 0x0 10200048 | |
| usbd_ep3wrstat | 0xB020004c | 0x0 1020004c | |
| usbd_ep4rdstat | 0xB0200050 | 0x0 10200050 | |
| usbd_ep5rdstat | 0xB0200054 | 0x0 10200054 | |
| usbd_enable | 0xB0200058 | 0x0 10200058 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| **IrDA Controller** | | | Section 6.4 |
| ir_rngptrstat | 0xB0300000 | 0x0 10300000 | |
| ir_rngbsadrh | 0xB0300004 | 0x0 10300004 | |
| ir_rngbsadrl | 0xB0300008 | 0x0 10300008 | |
| ir_ringsize | 0xB030000C | 0x0 1030000C | |
| ir_rngprompt | 0xB0300010 | 0x0 10300010 | |
| ir_rngadrcmp | 0xB0300014 | 0x0 10300014 | |
| ir_intclear | 0xB0300018 | 0x0 10300018 | |
| ir_config1 | 0xB0300020 | 0x0 10300020 | |
| ir_sirflags | 0xB0300024 | 0x0 10300024 | |
| ir_statusen | 0xB0300028 | 0x0 10300028 | |
| ir_rdphycfg | 0xB030002C | 0x0 1030002C | |
| ir_wrphycfg | 0xB0300030 | 0x0 10300030 | |
| ir_maxpktlen | 0xB0300034 | 0x0 10300034 | |
| ir_rxbytecnt | 0xB0300038 | 0x0 10300038 | |
| ir_config2 | 0xB030003C | 0x0 1030003C | |
| ir_enable | 0xB0300040 | 0x0 10300040 | |
| **Ethernet Controller MAC0** | | | Section 6.5 |
| mac_control | 0xB0500000 | 0x0 10500000 | |
| mac_addrhigh | 0xB0500004 | 0x0 10500004 | |
| mac_addrlow | 0xB0500008 | 0x0 10500008 | |
| mac_hashhigh | 0xB050000C | 0x0 1050000C | |
| mac_hashlow | 0xB0500010 | 0x0 10500010 | |
| mac_miictrl | 0xB0500014 | 0x0 10500014 | |
| mac_miidata | 0xB0500018 | 0x0 10500018 | |
| mac_flowctrl | 0xB050001C | 0x0 1050001C | |
| mac_vlan1 | 0xB0500020 | 0x0 10500020 | |
| mac_vlan2 | 0xB0500024 | 0x0 10500024 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| **Ethernet Controller Enable** | | | Section 6.5 |
| macen_mac0 | 0xB0520000 | 0x0 10520000 | |
| **Ethernet Controller DMA Channels** | | | Section 6.5 |
| macdma0_tx0stat | 0xB4004000 | 0x0 14004000 | |
| macdma0_tx0addr | 0xB4004004 | 0x0 14004004 | |
| macdma0_tx0len | 0xB4004008 | 0x0 14004008 | |
| macdma0_tx1stat | 0xB4004010 | 0x0 14004010 | |
| macdma0_tx1addr | 0xB4004014 | 0x0 14004014 | |
| macdma0_tx1len | 0xB4004018 | 0x0 14004018 | |
| macdma0_tx2stat | 0xB4004020 | 0x0 14004020 | |
| macdma0_tx2addr | 0xB4004024 | 0x0 14004024 | |
| macdma0_tx2len | 0xB4004028 | 0x0 14004028 | |
| macdma0_tx3stat | 0xB4004030 | 0x0 14004030 | |
| macdma0_tx3addr | 0xB4004034 | 0x0 14004034 | |
| macdma0_tx3len | 0xB4004038 | 0x0 14004038 | |
| macdma0_rx0stat | 0xB4004100 | 0x0 14004100 | |
| macdma0_rx0addr | 0xB4004104 | 0x0 14004104 | |
| macdma0_rx1stat | 0xB4004110 | 0x0 14004110 | |
| macdma0_rx1addr | 0xB4004114 | 0x0 14004114 | |
| macdma0_rx2stat | 0xB4004120 | 0x0 14004120 | |
| macdma0_rx2addr | 0xB4004124 | 0x0 14004124 | |
| macdma0_rx3stat | 0xB4004130 | 0x0 14004130 | |
| macdma0_rx3addr | 0xB4004134 | 0x0 14004134 | |
| macdma1_tx0stat | 0xB4004200 | 0x0 14004200 | |
| macdma1_tx0addr | 0xB4004204 | 0x0 14004204 | |
| macdma1_tx0len | 0xB4004208 | 0x0 14004208 | |
| macdma1_tx1stat | 0xB4004210 | 0x0 14004210 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| macdma1_tx1addr | 0xB4004214 | 0x0 14004214 | |
| macdma1_tx1len | 0xB4004218 | 0x0 14004218 | |
| macdma1_tx2stat | 0xB4004220 | 0x0 14004220 | |
| macdma1_tx2addr | 0xB4004224 | 0x0 14004224 | |
| macdma1_tx2len | 0xB4004228 | 0x0 14004228 | |
| macdma1_tx3stat | 0xB4004230 | 0x0 14004230 | |
| macdma1_tx3addr | 0xB4004234 | 0x0 14004234 | |
| macdma1_tx3len | 0xB4004238 | 0x0 14004238 | |
| macdma1_rx0stat | 0xB4004300 | 0x0 14004300 | |
| macdma1_rx0addr | 0xB4004304 | 0x0 14004304 | |
| macdma1_rx1stat | 0xB4004310 | 0x0 14004310 | |
| macdma1_rx1addr | 0xB4004314 | 0x0 14004314 | |
| macdma1_rx2stat | 0xB4004320 | 0x0 14004320 | |
| macdma1_rx2addr | 0xB4004324 | 0x0 14004324 | |
| macdma1_rx3stat | 0xB4004330 | 0x0 14004330 | |
| macdma1_rx3addr | 0xB4004334 | 0x0 14004334 | |
| **SD Controller 0** | | | |
| sd_txport | 0xB0600000 | 0x0 10600000 | |
| sd_rxport | 0xB0600004 | 0x0 10600004 | |
| sd_config | 0xB0600008 | 0x0 10600008 | |
| sd_enable | 0xB060000C | 0x0 1060000C | |
| sd_config2 | 0xB0600010 | 0x0 10600010 | |
| sd_blksize | 0xB0600014 | 0x0 10600014 | |
| sd_status | 0xB0600018 | 0x0 10600018 | |
| sd_debug | 0xB060001C | 0x0 1060001C | |
| sd_cmd | 0xB0600020 | 0x0 10600020 | |
| sd_cmdarg | 0xB0600024 | 0x0 10600024 | |
| sd_resp3 | 0xB0600028 | 0x0 10600028 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| sd_resp2 | 0xB060002C | 0x0 1060002C | |
| sd_resp1 | 0xB0600030 | 0x0 10600030 | |
| sd_resp0 | 0xB0600034 | 0x0 10600034 | |
| sd_timeout | 0xB0600038 | 0x0 10600038 | |
| **SD Controller 1** | | | |
| sd_txport | 0xB0680000 | 0x0 10680000 | |
| sd_rxport | 0xB0680004 | 0x0 10680004 | |
| sd_config | 0xB0680008 | 0x0 10680008 | |
| sd_enable | 0xB068000C | 0x0 1068000C | |
| sd_config2 | 0xB0680010 | 0x0 10680010 | |
| sd_blksize | 0xB0680014 | 0x0 10680014 | |
| sd_status | 0xB0680018 | 0x0 10680018 | |
| sd_debug | 0xB068001C | 0x0 1068001C | |
| sd_cmd | 0xB0680020 | 0x0 10680020 | |
| sd_cmdarg | 0xB0680024 | 0x0 10680024 | |
| sd_resp3 | 0xB0680028 | 0x0 10680028 | |
| sd_resp2 | 0xB068002C | 0x0 1068002C | |
| sd_resp1 | 0xB0680030 | 0x0 10680030 | |
| sd_resp0 | 0xB0680034 | 0x0 10680034 | |
| sd_timeout | 0xB0680038 | 0x0 10680038 | |
| **$I^2S$ Controller** | | | Section 6.6 |
| i2s_data | 0xB1000000 | 0x0 11000000 | |
| i2s_config | 0xB1000004 | 0x0 11000004 | |
| i2s_enable | 0xB1000008 | 0x0 11000008 | |
| **UART0** | | | Section 6.7 |
| uart_rxdata | 0xB1100000 | 0x0 11100000 | |
| uart_txdata | 0xB1100004 | 0x0 11100004 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| uart_inten | 0xB1100008 | 0x0 11100008 | |
| uart_intcause | 0xB110000C | 0x0 1110000C | |
| uart_fifoctrl | 0xB1100010 | 0x0 11100010 | |
| uart_linectrl | 0xB1100014 | 0x0 11100014 | |
| uart_mdmctrl | 0xB1100018 | 0x0 11100018 | |
| uart_linestat | 0xB110001C | 0x0 1110001C | |
| uart_mdmstat | 0xB1100020 | 0x0 11100020 | |
| uart_clkdiv | 0xB1100028 | 0x0 11100028 | |
| uart_modctrl | 0xB1100100 | 0x0 11100100 | |
| **UART1** | | | Section 6.7 |
| uart_rxdata | 0xB1200000 | 0x0 11200000 | |
| uart_txdata | 0xB1200004 | 0x0 11200004 | |
| uart_inten | 0xB1200008 | 0x0 11200008 | |
| uart_intcause | 0xB120000C | 0x0 1120000C | |
| uart_fifoctrl | 0xB1200010 | 0x0 11200010 | |
| uart_linectrl | 0xB1200014 | 0x0 11200014 | |
| uart_mdmctrl | 0xB1200018 | 0x0 11200018 | |
| uart_linestat | 0xB120001C | 0x0 1120001C | |
| uart_mdmstat | 0xB1200020 | 0x0 11200020 | |
| uart_clkdiv | 0xB1200028 | 0x0 11200028 | |
| uart_modctrl | 0xB1200100 | 0x0 11200100 | |
| **UART3** | | | Section 6.7 |
| uart_rxdata | 0xB1400000 | 0x0 11400000 | |
| uart_txdata | 0xB1400004 | 0x0 11400004 | |
| uart_inten | 0xB1400008 | 0x0 11400008 | |
| uart_intcause | 0xB140000C | 0x0 1140000C | |
| uart_fifoctrl | 0xB1400010 | 0x0 11400010 | |
| uart_linectrl | 0xB1400014 | 0x0 11400014 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| uart_mdmctrl | 0xB1400018 | 0x0 11400018 | |
| uart_linestat | 0xB140001C | 0x0 1140001C | |
| uart_mdmstat | 0xB1400020 | 0x0 11400020 | |
| uart_clkdiv | 0xB1400028 | 0x0 11400028 | |
| uart_modctrl | 0xB1400100 | 0x0 11400100 | |
| **SSI0** | | | Section 6.8 |
| ssi_status | 0xB1600000 | 0x0 11600000 | |
| ssi_int | 0xB1600004 | 0x0 11600004 | |
| ssi_inten | 0xB1600008 | 0x0 11600008 | |
| ssi_config | 0xB1600020 | 0x0 11600020 | |
| ssi_adata | 0xB1600024 | 0x0 11600024 | |
| ssi_clkdiv | 0xB1600028 | 0x0 11600028 | |
| ssi_enable | 0xB1600100 | 0x0 11600100 | |
| **SSI1** | | | Section 6.8 |
| ssi_status | 0xB1680000 | 0x0 11680000 | |
| ssi_int | 0xB1680004 | 0x0 11680004 | |
| ssi_inten | 0xB1680008 | 0x0 11680008 | |
| ssi_config | 0xB1680020 | 0x0 11680020 | |
| ssi_adata | 0xB1680024 | 0x0 11680024 | |
| ssi_clkdiv | 0xB1680028 | 0x0 11680028 | |
| ssi_enable | 0xB1680100 | 0x0 11680100 | |
| **Secondary GPIO** | | | |
| gpio2_dir | 0xB1700000 | 0x0 11700000 | |
| reserved | 0xB1700004 | 0x0 11700004 | |
| gpio2_output | 0xB1700008 | 0x0 11700008 | |
| gpio2_pinstate | 0xB170000C | 0x0 1170000C | |
| gpio2_inten | 0xB1700010 | 0x0 11700010 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| gpio2_enable | 0xB1700014 | 0x0 11700014 | |
| **Clock Controller** | | | Section 7.1 |
| sys_freqctrl0 | 0xB1900020 | 0x0 11900020 | |
| sys_freqctrl1 | 0xB1900024 | 0x0 11900024 | |
| sys_clksrc | 0xB1900028 | 0x0 11900028 | |
| sys_cpupll | 0xB1900060 | 0x0 11900060 | |
| sys_auxpll | 0xB1900064 | 0x0 11900064 | |
| **TOY & RTC** | | | Section 7.2 |
| sys_toytrim | 0xB1900000 | 0x0 11900000 | |
| sys_toywrite | 0xB1900004 | 0x0 11900004 | |
| sys_matchtoy0 | 0xB1900008 | 0x0 11900008 | |
| sys_matchtoy1 | 0xB190000C | 0x0 1190000C | |
| sys_matchtoy2 | 0xB1900010 | 0x0 11900010 | |
| sys_cntrctrl | 0xB1900014 | 0x0 11900014 | |
| sys_toyread | 0xB1900040 | 0x0 11900040 | |
| sys_rtctrim | 0xB1900044 | 0x0 11900044 | |
| sys_rtcwrite | 0xB1900048 | 0x0 11900048 | |
| sys_rtcmatch0 | 0xB190004C | 0x0 1190004C | |
| sys_rtcmatch1 | 0xB1900050 | 0x0 11900050 | |
| sys_rtcmatch2 | 0xB1900054 | 0x0 11900054 | |
| sys_rtcread | 0xB1900058 | 0x0 11900058 | |
| **Primary GPIO** | | | Section 7.3 |
| sys_pinfunc | 0xB190002C | 0x0 1190002C | |
| sys_trioutrd | 0xB1900100 | 0x0 11900100 | |
| sys_trioutclr | 0xB1900100 | 0x0 11900100 | |
| sys_outputrd | 0xB1900108 | 0x0 11900108 | |
| sys_outputset | 0xB1900108 | 0x0 11900108 | |

**TABLE 113. Device Memory Map (Continued)**

| Register | KSEG1 Address | Physical Address | Reference |
|---|---|---|---|
| sys_outputclr | 0xB190010C | 0x0 1190010C | |
| sys_pinstaterd | 0xB1900110 | 0x0 11900110 | |
| sys_pininputen | 0xB1900110 | 0x0 11900110 | |
| **Power Management** | | | Section 7.4 |
| sys_scratch0 | 0xB1900018 | 0x0 11900018 | |
| sys_scratch1 | 0xB190001C | 0x0 1190001C | |
| sys_wakemsk | 0xB1900034 | 0x0 11900034 | |
| sys_endian | 0xB1900038 | 0x0 11900038 | |
| sys_powerctrl | 0xB190003C | 0x0 1190003C | |
| sys_wakesrc | 0xB190005C | 0x0 1190005C | |
| sys_slppwr | 0xB1900078 | 0x0 11900078 | |
| sys_sleep | 0xB190007C | 0x0 1190007C | |
| **LCD Controller** | | | |
| lcd_control | 0xB5000000 | 0x0 15000000 | |
| lcd_intstatus | 0xB5000004 | 0x0 15000004 | |
| lcd_intenable | 0xB5000008 | 0x0 15000008 | |
| lcd_horztiming | 0xB500000C | 0x0 1500000C | |
| lcd_verttiming | 0xB5000010 | 0x0 15000010 | |
| lcd_clkcontrol | 0xB5000014 | 0x0 15000014 | |
| lcd_dmaaddr0 | 0xB5000018 | 0x0 15000018 | |
| lcd_dmaaddr1 | 0xB500001C | 0x0 1500001C | |
| lcd_words | 0xB5000020 | 0x0 15000020 | |
| lcd_pwmdiv | 0xB5000024 | 0x0 15000024 | |
| lcd_pwmhi | 0xB5000028 | 0x0 15000028 | |
| lcd_pallettebase | 0xB5000400 | 0x0 15000400 | |

# Index

## E



## G



## H



## I

## W