www.DataSheet4U.com

Raza Microelectronics, Inc.

RMI Alchemy™ Au1210™ Navigation Processor and Au1250™ Media Processor Data Book

April 2007

Revision A - Preliminary

www.DataSheet4U.com

www.DataSheet4U.com

© 2007 Raza Microelectronics, Inc. All rights reserved.

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

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

Trademarks

RMI, Au1100, and Au1500 are registered trademarks and RMI Alchemy, Au1000, Au1200, Au1210, Au1250, and Au1550 are trademarks of Raza Microelectronics, Inc.

MIPS32 is a registered trademark of MIPS Technologies, Inc.

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

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

www.DataSheet4U.com

www.DataSheet4U.com

Contents

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

List of Figures

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

List of Tables

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

1 Overview

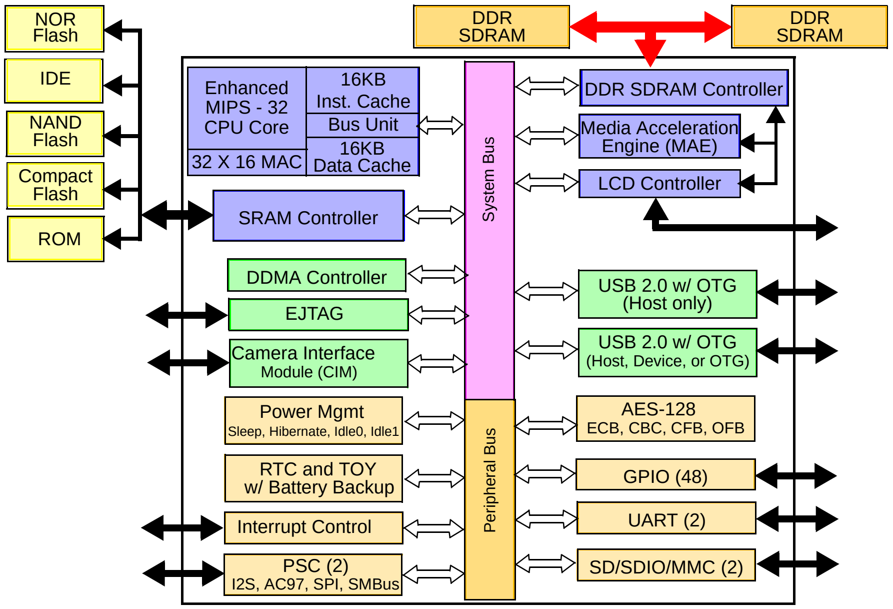
The RMI Alchemy™ Au1210™ navigation processor and Au1250™ media processor are versatile high-performance, low-power, high integration processors with System-on-a-Chip (SoC) architecture. Both the Au1210 and Au1250 processors are pin and software compatible with the existing RMI Alchemy™ Au1200™ processor's sockets and leverage the Au1200 processor's success in Portable Media Players (PMPs), automotive infotainment, navigation, and Digital Media Adapters (DMAs), by providing significantly increased performance while improving the power optimized advantages associated with the Alchemy processor family.

The Au1210 and Au1250 processors feature USB 2.0 (high speed, full speed, low speed, and On-The-Go (OTG) support), a CCIR656 camera interface module, AES-128 data encryption/decryption in hardware (Au1250 only), and programmable serial controllers. Both processors also leverage a unique approach to DDR1 and DDR2 memory to deliver better overall performance than SDRAM components.

The Alchemy Au1210 and Au1250 processors incorporate an integrated Media Acceleration Engine (MAE) optimized for video to eliminate the need for DSPs and specialized coding. This means less overall power consumption and a simplified design.

Another advantage of the Au1210 and Au1250 processors is the expansion of the memory implementation options that are available to a designer. The Au1210 and Au1250 processors enable boot start-up directly from large block NAND memories.

Both processors are available in an extended temperature range option of -40ºC to 100ºC for speeds up to 500MHz (Au1250) and 400MHz (Au1210).



AES-128 is a feature of the Au1250™ processor only.

Figure 1-1. Au1210™/Au1250™ Processor Block Diagram

www.DataSheet4U.com

www.DataSheet4U.com

1.1 Product Descriptions

1.1.1 Au1250™ Processor

The RMI Alchemy Au1250 media processor extends the strengths of the existing Alchemy processors into a broad range of digital media applications.

This elevated application processing performance, with remarkably low power, makes the Au1250 processor ideally suited to the expanding needs of multi-function Portable Media Players (mPMPs). The Au1250 processor's optimized design is tailored to efficiently support multi-format video playback, GPS navigation, mobile TV, satellite radio, along with photo viewers, digital cameras and other applications in a single, low-cost, portable design.

The Au1250 processor combines increased performance, up to 600MHz, with power reductions of more than 30% (active power) and 60% (inactive power), compared to existing Au1200 processor-based systems, the current performance per watt leader in the industry. Users can now enjoy the ability to simultaneously run multiple diverse applications with substantially longer battery life than before.

1.1.2 Au1210™ Processor

The RMI Alchemy Au1210 navigation processor is a derivative of the Au1250 processor and is optimized for Portable Navigation Devices (PNDs) and dedicated media applications. The Au1210 enjoys similar functions of the Au1250 except for reduced clock rate, video resolution, and AES encryption/decryption support.

The Au1210 processor operates at frequencies up to 400MHz and supports video decode up to Wide-CIF resolution (480x288). This targeted functionality enables the Au1210 to address the needs of cost-sensitive applications such as GPS navigation, where video playback is important, but full D1 video resolution is not required.

1.2 Features

High Speed MIPS CPU Core

- Core frequency:
  - — Au1210 processor: 333 and 400MHz
  - — Au1250 processor: 400, 500, and 600MHz
- MIPS32® instruction set 32-bit architecture
- Nominal core voltage:
  - — Au1210: 1.0V (up to 1.2V for compatibility with Au1200/Au1250)
  - — Au1250: 1.2V
- 2.5V or 1.8V DDR SDRAM I/O voltage,
  3.3V I/O voltage
- Pipeline
  - — Scalar 5-stage pipeline
  - — Load/store adder in I-stage (instr decode)
  - — Scalar branch techniques optimized: Pipelined register file access in fetch stage
  - — Zero penalty branch
- Multiply-Accumulate (MAC) and Divide Unit
  - — Max issue rate of one 32x16 MAC per every other clock
  - — Max issue rate of one 32x32 MAC per every third clock
  - — Operates in parallel to CPU pipeline
  - — Executes all integer multiply and divide instructions
  - — 32 x 16-bit MAC hardware
- Caches
  - — 16 KB non-blocking data cache
  - — 16 KB instruction cache
  - — 4-way set associative instruction/data caches
  - — 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
- MMU
  - — TLB features:
    - – 32 dual-entry fully-associative
    - – Variable page sizes: 4 KB to 16 MB
    - – 4-entry ITB
  - — Separate TLB miss interrupt exception vector

Highly-Integrated System Peripherals

- GPIO (48 total, 5 dedicated for system use)
- USB 2.0 device and host controllers with OTG support
- Two programmable serial controllers (PSC) supporting Audio Codec-97 Controller (AC97), Inter-IC Sound (I^2S), Serial Peripheral Interface (SPI), System Management Bus (SMBus)
- Two Secure Digital/SDIO/MMC controllers
- Camera interface module supporting 8- to 10-bit image sensors
- LCD controller with 32-bit alpha-RGB color resolution support
- AES-128 encryption/decryption in hardware (Au1250 only)
- Two UARTs

www.DataSheet4U.com

www.DataSheet4U.com

Media Acceleration Engine

- Accelerates video decode (MPEG1, 2, 4, H.263, WMV9/VC-1) in hardware
- Hardware color-space conversion and scaling
- Video Decode Resolution:
  - — Au1250 supports up to full D1: 720x480 NTSC or 720x576 PAL
  - — Au1210 supports up to Wide-CIF: 480x288 only

Descriptor-based DMA (DDMA)

- Linked list of DMA transfer descriptors
- Scatter/gather (SGL) and stride transfers
- 16 channels
- Memory to memory, memory to peripheral, peripheral to memory, peripheral to peripheral

Memory Buses

- High-bandwidth DDR1/DDR2 SDRAM memory controller (supports up to DDR400 and DDR2-533 (DDR2-533 SDRAM operates at reduced frequencies that match the speed rating of the Au1210 nd Au1250 as shown in Table 3-4 on page 73.))
- SRAM/Flash EPROM controller with IDE and NOR/NAND Flash support
- Compact Flash and PCMCIA support
- External 10/100 Ethernet controller support

Low SoC Power

- 1/4 Watt Typical Power at 333MHz (Au1210, QVGA video)
- 1/2 Watt Typical Power at 600MHz (Au1250, Full D1 video)
- Power Saving Modes:
  - — Idle
  - — Sleep
  - — Hibernate

Package

- 372 LBA, 19x19 mm

Operating System Support

- Microsoft Windows® CE
- Linux

Development Tool Support

- Complete MIPS32 compatible tool set
- Numerous third party compilers, assemblers, and debuggers

1.3 Processor Comparison

Table 1-1 lists the differences between the Au1200, Au1210 and Au1250 processors, with references to those areas in this data book specific to the Au1210 and Au1250 processors.

Table 1-1. Processor Comparison

Feature	Au1200™	Au1210™	Au1250™	Reference
Processor Identification	**PRId**[Company Options] = 4	**PRId**[Company Options] = 5	**PRId**[Company Options] = 4	Section 2.8.14 on page 42
Clock configurations for DDR1	500MHz	400MHz	Updated for 600MHz	Table 3-3 on page 72
Clock configurations for DDR2	500MHz	400MHz	Updated for 600MHz	Table 3-4 on page 73
Video Decode Resolution	D1: 720x480	Wide-CIF: 480x288	D1: 720x480	Section 6.1.1 on page 149 and "Media Acceleration Engine" on page 17
MAE Picture Size register		**maefe_pictsize** [HEIGHT, LINESIZ] Restricted	No change	Section 6.2.2.11 on page 155
AES Cryptography Engine	Supported	Not supported	Supported	Section 9.3 on page 325
Related and Actual CPU Frequencies Using a 12MHZ Crystal			Updated for 600MHz	Table 10-4 on page 373
Boot device selection	Updated entire table and added information for booting from Large block NAND for Au1210 and Au1250.			Table 11-1 on page 401
Device Identification for EJTAG	0x0402	0x0502	0x0402	Section 12.4.2.1 on page 413
Rated and actual CPU frequencies	500MHz	400MHz	Updated for 600MHz	Table 14-1 on page 439
Power and Voltage			Added section for 600MHz	Section 14.3.4 on page 448

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

2 Au1 Core and System Bus

The Alchemy™ Au1210™ and Au1250™ processors' core (Au1) is a unique implementation of the MIPS32® instruction set architecture (ISA), designed for high-performance and low-power. This section provides the implementation details specific to the MIPS32 compliant Au1 core.

Note: 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.

Section 2.9 "System Bus" on page 50 discusses the system bus, the main internal bus connecting the Au1 core to the system. The Au1 core communicates with peripherals and memories via the system bus.

2.1 Au1 Core Overview

The Au1 core is a high-performance, low-power implementation of the MIPS32 architecture. The core includes the following main components:

- Instruction pipeline with multiply-accumulate unit (MAC) and register file
- Coprocessor 0 registers
- Instruction and data caches
- Write buffer
- Virtual address translation unit (translation-lookaside buffer, TLB)
- EJTAG (see Section 12.0 "EJTAG Implementation" on page 403)

Figure 2-1 shows a block diagram of the Au1 core.

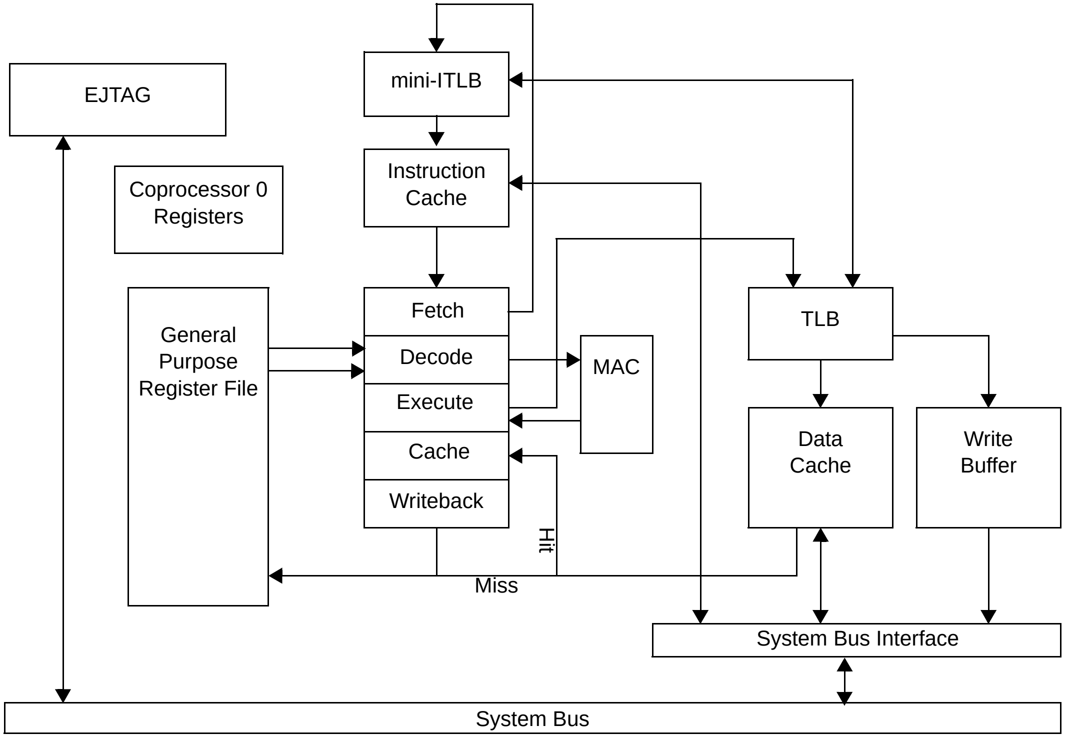


Figure 2-1. Au1 Core Block Diagram

www.DataSheet4U.com

www.DataSheet4U.com

2.2 Instruction Pipeline and MAC

The Au1 core contains a five-stage instruction pipeline. The MAC executes multiply and divide instructions in parallel with the main five-stage pipeline.

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

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

2.2.1 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, 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.

2.2.2 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.

2.2.3 Execute Stage

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

The virtual address calculation begins in the *decode* stage so that physical address calculation can complete in the *execute* stage, in time to initiate the access to the data cache in the *execute* stage. If the physical address misses in the Translation Look-aside Buffer (TLB), a TLB exception is posted.

Multiplies and divides are forwarded to the multiply-accumulate unit. These instructions require multiple cycles and execute independently 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 signaled in the *cache* stage.

2.2.4 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 from a non-cacheable location, the request is sent to the system bus to be fulfilled. Load data is forwarded to dependent instructions in the pipeline.

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

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

2.2.5 Writeback Stage

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

www.DataSheet4U.com

www.DataSheet4U.com

2.2.6 Multiply-Accumulate Unit

The multiply-accumulate unit (MAC) executes all multiply and divide instructions, except **MUL**. 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 16x16 or 32x16 multiply instruction can be issued every other cycle. The 32x16 bit multiply must have the sign-extended 16-bit register operand **rt** of the instruction.

32x32 bit multiply instructions can be issued every third cycle.

If the results are written to the HI/LO registers, four cycles are required.

Divide instructions complete in a maximum of 35 cycles.

www.DataSheet4U.com

www.DataSheet4U.com

2.3 Caches

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

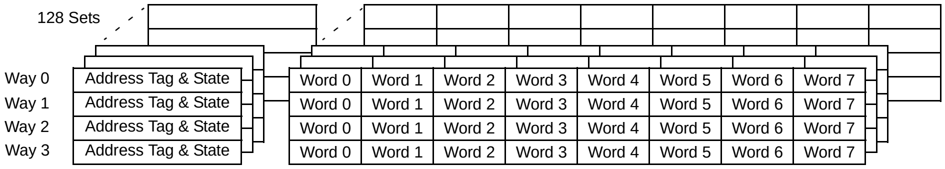

128 Sets
Way 0
Way 1
Way 2
Way 3
Address Tag & State
Address Tag & State
Address Tag & State
Address Tag & State
Word 0 Word 1 Word 2 Word 3 Word 4 Word 5 Word 6 Word 7
Word 0 Word 1 Word 2 Word 3 Word 4 Word 5 Word 6 Word 7
Word 0 Word 1 Word 2 Word 3 Word 4 Word 5 Word 6 Word 7
Word 0 Word 1 Word 2 Word 3 Word 4 Word 5 Word 6 Word 7

Figure 2-2. Cache Organization

A cache line is tagged with a 20-bit physical address, a lock bit, and a valid bit. Data cache lines also include coherency and dirty status bits. The physical address tag contains bits [31:12] of the physical address; as such, physical addresses in which bits [35:32] are non-zero are not cacheable.

Cache Line State

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

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

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
	Virtual/Physical Address																				Set Select							Byte Select				

2.3.1 Cache Line Replacement Policy

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

2.3.2 Cache Line Locking Support

The **CACHE** instruction is used to lock individual lines in the cache. A locked line is not subject to replacement. All four lines in a set cannot be locked at once; at least one line is always available for replacement. To *unlock* individual cache lines, use the **CACHE** instruction with a ‘hit invalidate’ command opcode. See Section 2.3.5 "Cache Management" on page 24 for further discussion of the **CACHE** instruction.

2.3.3 Cache Streaming Support

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

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

www.DataSheet4U.com

www.DataSheet4U.com

2.3.4 Cache Line Allocation Behavior

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

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

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

Cache Hit:
new MRU = Hit Way
```

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

1) Streaming: cache misses are forced to way 0.

2) Locking: cache misses follow policy above and set Lock bit.

3) Normal: true LRU replacement.

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

Table 2-1. Cache Line Allocation Behavior

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

www.DataSheet4U.com

www.DataSheet4U.com

2.3.5 Cache Management

The caches are managed with the **CACHE** instruction. Table 2-2 shows the cache operations, including the opcode for the **CACHE** instruction. (An “n/a” indicates that the operation is not applicable.) These cache operations permit initialization, locking/unlocking and management of the caches.

2.3.6 Cache Coherency Attributes (CCA)

The cache coherency attributes (CCA) field in Config0[K0] and in the TLB determine the cache-ability of accesses to memory. Cached accesses (except CCA = 4) are performed critical-word-first to improve performance. The Au1 core implements the CCA in Table 2-3.

Table 2-2. Cache Operations

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

Table 2-3. Cache Coherency Attributes (CCA)

CCA	CCA (3 Bits)	Description
0, 1	00x	Reserved (undefined).
2	010	Uncached, non-mergeable, non-gatherable. Required by the MIPS32 architecture. In addition, data is not merged within the write buffer to achieve a truly uncached effect. This is the setting for KSEG1 as defined by the MIPS32 architecture.
3	011	Cached, mergeable, gatherable.
4	100	Cached, mergeable, gatherable (word 0 first). Word 0 is always accessed first; that is, the cache line is accessed in word order (word 0, word 1, …, word 7).
5	101	Cached, mergeable, gatherable.
6	110	Cached, mergeable, gatherable, streaming. Instructions and/or data are placed into way 0.
7	111	Uncached, mergeable, gatherable. Even though data is not cached, data stores sent to the write buffer are subject to merging and gathering in the write buffer.

www.DataSheet4U.com

www.DataSheet4U.c

2.3.7 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

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

2.3.7.1 Instruction Cache Initialization and Invalidation

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

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

2.3.7.2 Instruction Cache Line Fills

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

The instruction cache line is selected by the replacement policy described in Section 2.3.1 "Cache Line Replacement Policy" on page 22.

2.3.7.3 Instruction Cache Coherency

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

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

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

2.3.7.4 Instruction Cache Control

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

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

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

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

www.DataSheet4U.com

www.DataSheet4U.co

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

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

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

2.3.8 Data Cache

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

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

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

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

Data Cache Line State

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

2.3.8.1 Data Cache Initialization and Invalidation

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

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

2.3.8.2 Data Cache Line Fills

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

The data cache line is selected by the replacement policy described in Section 2.3.1 "Cache Line Replacement Policy" on page 22. If the line selected contains modified data (cache line is valid and has its dirty bit set by a store hit), 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.

www.DataSheet4U.com

www.DataSheet4U.com

2.3.8.3 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 unchanged by the transaction.

Loads and stores that 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 that 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. In the Au1210 and Au1250 processors, the only caching master is the core. 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 ensures that the core can make forward progress.

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

2.3.8.4 Data Cache Control

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

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

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

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

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

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

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

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

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

The streaming load hint directs the data to 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.

www.DataSheet4U.com

www.DataSheet4U.com

2.4 Write Buffer

The Au1 write buffer is depicted in Figure 2-3. All non-cacheable processor stores and data cache store-misses (the data cache is a read-allocate policy) are routed through the write buffer. The write buffer is a 16-word deep first-in-first-out (FIFO) queue. All processor stores arrive first at the merge latch, where merging and gathering decisions are performed, and then travel through the queue. The write buffer arbitrates for the 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* (V) 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 conditions in 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 word is written to memory, it is popped from FIFO entry 0, advancing each entry in the FIFO by one.

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

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

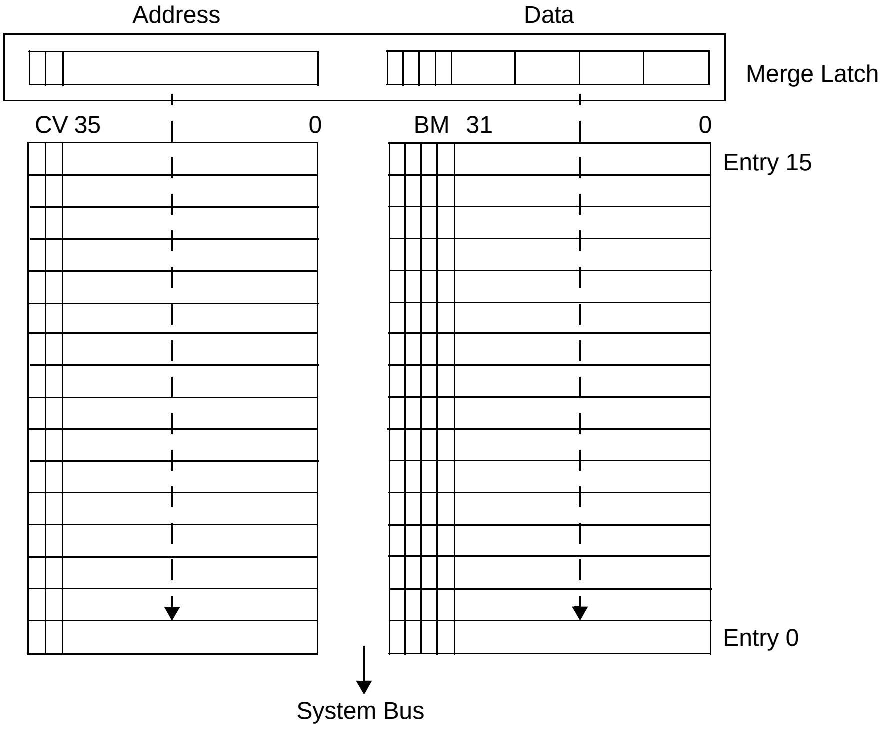


Figure 2-3. Au1 Write Buffer

www.DataSheet4U.com

www.DataSheet4U.com

2.4.1 Merge Latch

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

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 as described in Section 2.4.2, "Write Buffer Merging". The merge latch contents do not propagate to the write buffer FIFO.

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*, as described in Section 2.4.3 "Write Buffer Gathering" on page 30. The merge latch contents are propagated to the write buffer FIFO with the C bit cleared, and the incoming data is placed into the merge latch.

3) Neither 1 nor 2 is true. The merge latch contents are propagated to the FIFO with the C bit set, and the incoming data is placed into the merge latch.

When 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 size (the least significant 5 bits are 0x1C), the C bit is set for the incoming address.

2.4.2 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 merge-disable bit (Config0[NM]) and the TLB[CCA] setting:

- When Config0[NM] is 1, merging is disabled globally, regardless of CCA.
- When Config0[NM] is 0 (default value out of reset), merging is enabled globally. However, merging for individual memory regions can still be controlled by the TLB[CCA] value:
  — When CCA is 2, merging is disabled.
  — When CCA is not 2, merging is enabled.

Note: Merging takes place *only* in the merge latch. As such, writes to an address which are in the FIFO (but not in the merge latch) do not merge. In the example below, writes to 0x0001000 and 0x0001002 do not merge because the intervening write to address 0x0_0000_1005 is not in the same word address which caused 0x00001000 to leave the merge latch.
0x00001000 = 0xAB
0x00001005 = 0xCD
0x00001002 = 0xDE

www.DataSheet4U.com

www.DataSheet4U.com

2.4.3 Write Buffer Gathering

Write buffer gathering combines *sequentially adjacent* word addresses for burst transfers to the main memory. Adjacent word addresses are “gathered” following the logic rules described in Section 2.4.1 "Merge Latch" on page 29. The end of a gatherable sequence has its C bit set. The entire sequence is then written to main memory in a single burst the next time the write buffer is granted the system bus.

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 that sequentially writes words (optimized memcpy(), for example). These eight word writes occur in sequence:

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

The write buffer FIFO entries corresponding to word addresses 0x00001000 through 0x00001018 have the C bits cleared. When address 0x0 0000 101C arrives, the C bit is set to mark the end of a gatherable sequence. When the write buffer is granted the system bus, it bursts all eight entries to main memory.

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

0x00001000
0x00001004
0x0000100C
0x00001008

Based on the above write sequence, the following occurs:

1) The 0x00001000 entry is placed in the merge latch.

2) The 0x00001004 entry pushes the 0x00001000 entry with its C bit *cleared* into the write buffer FIFO.

3) Because the 0x0000100C entry is not sequential, the 0x00001004 entry is pushed into the FIFO with its C bit *set*. Entries 0x00001000 and 0x00001004 are then burst to main memory.

4) When the 0x00001008 entry arrives, the 0x0000100C entry is pushed into the FIFO with its C bit set. The 0x00001008 entry resides in the merge latch until displaced by a subsequent store.

After this example write sequence, the write buffer appears as in Figure 2-4 on page 31. The figure shows the two write requests pending because the write buffer may not be granted immediate access to the system bus.

2.4.4 Write Buffer Reads

When a read from memory is initiated, the read cache-line address (bits [35:5]) is compared against all valid 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.

www.DataSheet4U.com

www.DataSheet4U.com

2.4.5 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 (initiated by an integrated peripheral DMA engine, for example). 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, such as clearing an interrupt acknowledge bit.

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.

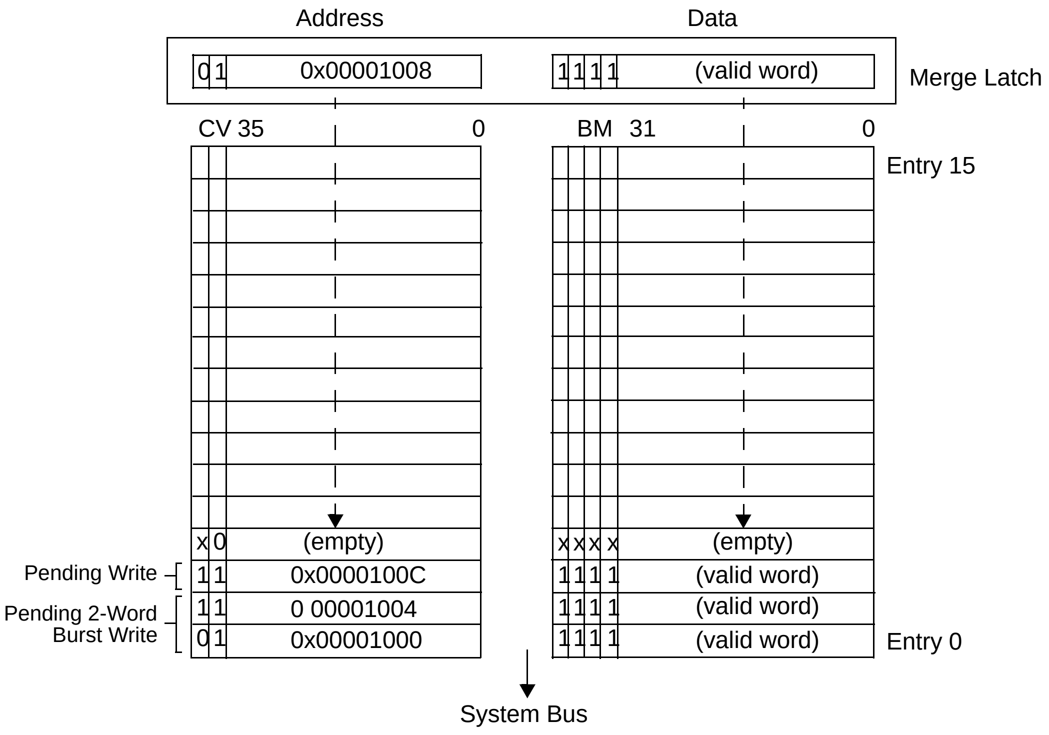


Figure 2-4. Example of Pending System Bus Requests in the Write Buffer

www.DataSheet4U.com

www.DataSheet4U.com

2.5 Virtual Memory (TLB)

The Au1 core 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.

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 2-4. A TLB entry is described as follows:

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 select bit of the program address selects either PFN0 (even) or PFN1 (odd) as the upper bits of the resulting 36-bit physical address.

The TLB mechanism permits mapping a larger, 36-bit physical address space into the smaller 32-bit program address space. The Au1 core implements an internal 36-bit physical address system bus (*sysbus_addr*) which is then decoded by integrated peripherals, and by chip-selects for external memories and peripherals. (See Section 2.9 "System Bus" on page 50.)

The cache coherency attributes (CCA) of the physical page are controlled by the TLB entry. The valid values are described in Table 2-4. In general, I/O spaces require a non-cacheable setting, whereas memory can utilize a cacheable setting.

Note: Physical addresses in which address bits [35:32] are non-zero must be mapped non-cached (CCA = 2 or 7).

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.

TLB Entry

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
Page-Mask	0			PageMask																0												
EntryHi	VPN2																			0					ASID							
EntryLo0	0	0	PFN0																								C0			D0	V0	G
EntryLo1	0	0	PFN1																								C1			D1	V1	G

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

Page Size	PageMask Register	Bits 28:13	PFN Select Bit
4 KB	0x00000000	0000000000000000	12
16 KB	0x00006000	0000000000000011	14
64 KB	0x0001E000	0000000000001111	16
256 KB	0x0007E000	0000000000111111	18
1 MB	0x001FE000	0000000011111111	20
4 MB	0x007FE000	0000001111111111	22
16 MB	0x01FFE000	0000111111111111	24

www.DataSheet4U.com

www.DataSheet4U.com

2.6 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.

2.6.1 Exception Causes

The nature of an exception is reported in the Cause[ExcCode] field; see Table 2-7 on page 37. The Au1 core can generate the exceptions shown in Table 2-5.

The Au1 core does not implement hardware floating-point. As a result, all floating-point instructions generate the *reserved instruction* (RI) exception, which permits software to emulate floating-point operations.

In addition, the Au1 core does not recognize *soft reset*, *non-maskable interrupt* (NMI), or *cache error* exception conditions.

Table 2-5. Cause[ExcCode] Encodings

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

www.DataSheet4U.com

www.DataSheet4U.com

2.6.2 Interrupt Architecture

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

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

For more information on interrupt controller 0 and 1, see Section 5.0 "Interrupt Controller" on page 143.

Table 2-6. CPU Interrupt Sources

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

www.DataSheet4U.com

www.DataSheet4U.com

2.7 Au1 Core Implementation of the MIPS32® ISA

The Au1 core implements the instruction set defined in “MIPS32® Architecture For Programmers Volume II: The MIPS32 Instruction Set”. 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). This section contains MIPS32 ISA implementation details specific to the Au1 core.

The MIPS32 floating-point instructions are not implemented in the Au1 core but may be emulated in software.

2.7.1 CACHE Instruction

The **CACHE** instruction permits management of the Au1 instruction and data caches. The valid operations are listed in Table 2-2 on page 24.

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 shown in the register below.

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 cache 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 (that is, not the Index operations) do not generate the *address error* exception or trigger *data watchpoint* exceptions.

CACHE Index Operation Address Decode

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

2.7.2 PREF Instruction

The **PREF** instruction prefetches data into the data cache. Data is prefetched to improve algorithm performance by placing the data in the cache in advance of its use, thus minimizing stalls due to data cache load misses. See also Section 2.3.8.4 "Data Cache Control" on page 27 for details on how to use **PREF**.

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

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

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

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

www.DataSheet4U.com

www.DataSheet4U.com

2.7.3 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 data coherency is no longer maintained.

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

To enter the low power mode, the **WAIT** instruction must be followed by at least four NOPs, and the entire instruction sequence must be fetched from the instruction cache. More specifically, if the core fetches the **WAIT** and **NOP** instructions from main memory, the mechanisms for accessing memory 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
    sync
    nop
    wait 0
    nop
    nop
    nop
    nop
    j ra
```

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

www.DataSheet4U.com

www.DataSheet4U.com

2.8 Coprocessor 0 Implementation

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. This section describes the CP0 registers as implemented in the Au1 core.

Table 2-7 lists the CP0 registers as defined by the MIPS32 ISA.

Table 2-7. Coprocessor 0 Registers

Register Number	Sel	Register Name	Description	Compliance[1]
0	0	Index	Pointer into TLB array	Required
1	0	Random	Pseudo-random TLB pointer	Required
2	0	EntryLo0	Low half of TLB entry for even pages	Required
3	0	EntryLo1	Low half of TLB entry of odd pages	Required
4	0	Context	Pointer to a page table entry	Required
5	0	PageMask	Variable page size select	Required
6	0	Wired	Number of locked TLB entries	Required
7	0		Reserved	Reserved
8	0	BadVAddr	Bad virtual address	Required
9	0	Count	CPU cycle count	Required
10	0	EntryHi	High half of TLB entries	Required
11	0	Compare	CPU cycle count interrupt comparator	Required
12	0	Status	Status	Required
13	0	Cause	Reason for last exception	Required
14	0	EPC	Program Counter of last exception	Required
15	0	PRId	Processor ID and Revision	Required
16	0	Config0	Configuration Register 0	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		Reserved	Reserved
26	0		Reserved	Reserved
27	0		Reserved	Reserved
28	0	DTag	Data cache tag value	Au1
28	1	DData	Data cache data value	Au1

www.DataSheet4U.com

www.DataSheet4U.com

Table 2-7. Coprocessor 0 Registers (Continued)

Register Number	Sel	Register Name	Description	Compliance[1]
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

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

2.8.1 Index Register (CP0 Register 0, Select 0)

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

Index **CP0 Register 0, Select 0**

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

Bits	Name	Description	R/W	Default
31	P	Probe Failure.	R	UNPRED
30:5	—	Reserved.	R	0
4:0	Index	TLB Index.	R/W	UNPRED

2.8.2 Random Register (CP0 Register 1, Select 0)

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

Random **CP0 Register 1, Select 0**

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

Bits	Name	Description	R/W	Default
31:5	—	Reserved.	R	0
4:0	Random	TLB Random Index.	R	31

www.DataSheet4U.com

www.DataSheet4U.com

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

These registers are required for TLB-based virtual address translation units.

EntryLo0, EntryLo1 **CP0 Registers 2 and 3, Select 0**

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

Bits	Name	Description	R/W	Default
31:30	—	Reserved.	R	0
29:6	PFN	Page Frame Number. Corresponds to physical address bits [35:12].	R/W	UNPRED
5:3	C	Cache coherency attribute of the page. See Table 2-3 on page 24.	R/W	UNPRED
2	D	Dirty bit.	R/W	UNPRED
1	V	Valid bit.	R/W	UNPRED
0	G	Global bit.	R/W	UNPRED

2.8.4 Context Register (CP0 Register 4, Select 0)

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

Context **CP0 Register 4, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PTEBase									BadVPN2																			▨	▨	▨	▨
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	0	0	0	0

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

2.8.5 PageMask Register (CP0 Register 5, Select 0)

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

PageMask **CP0 Register 5, Select 0**

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

Bits	Name	Description	R/W	Default
31:29	—	Reserved.	R	0
28:13	Mask	The Mask field is a bit mask in which a “1” bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. See Table 2-4 on page 32.	R/W	UNPRED
12:0	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.co

2.8.6 Wired Register (CP0 Register 6, Select 0)

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

Wired **CP0 Register 6, Select 0**

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

Bits	Name	Description	R/W	Default
31:5	—	Reserved.	R	0
4:0	Wired	TLB Wired Boundary.	R/W	0

2.8.7 BadVAddr Register (CP0 Register 8, Select 0)

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

BadVAddr **CP0 Register 8, Select 0**

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

Bits	Name	Description	R/W	Default
31:0	BadVAddr	Bad Virtual Address.	R	UNPRED

2.8.8 Count Register (CP0 Register 9, Select 0)

This register is a required register for a constant rate timer. This counter increments 1:1 with the Au1 core frequency.

Count **CP0 Register 9, Select 0**

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

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

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

2.8.9 EntryHi Register (CP0 Register 10, Select 0)

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

EntryHi **CP0 Register 10, Select 0**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:13	VPN2	Virtual address bits [31:13].	R/W	UNPRED
12:8	—	Reserved.	R	0
7:0	ASID	Address Space Identifier.	R/W	UNPRED

2.8.10 Compare Register (CP0 Register 11, Select 0)

This register is required for generating an interrupt from the constant rate timer.

Compare **CP0 Register 11, Select 0**

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

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

2.8.11 Status Register (CP0 Register 12, Select 0)

This register is required for general control of the processor.

Status **CP0 Register 12, Select 0**

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

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

www.DataSheet4U.com

www.DataSheet4U.c

2.8.12 Cause Register (CP0 Register 13, Select 0)

This register is required for general exception processing.

Cause **CP0 Register 13, Select 0**

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

Bits	Name	Description	R/W	Default
31	BD	Exception in branch delay slot.	R	0
30	—	Reserved.	R	0
29:28	CE	Coprocessor Error.	R	0
27:24	—	Reserved.	R	0
23	IV	Interrupt Vector.	R/W	0
22	WP	Watchpoint Exception Deferred.	R/W	0
21:16	—	Reserved.	R	0
15:10	IP[7:2]	Hardware Interrupts Pending.	R	0x20
9:8	IP[1:0]	Software Interrupts Pending.	R/W	0
7	—	Reserved.	R	0
6:2	ExcCode	Exception Code.	R	0
1:0	—	Reserved.	R	0

2.8.13 Exception Program Counter (CP0 Register 14, Select 0)

This register is required for general exception processing.

EPC **CP0 Register 14, Select 0**

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

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

2.8.14 Processor Identification (CP0 Register 15, Select 0)

This register is required for processor identification.

PRId **CP0 Register 15, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Company Options								Company ID								Processor ID								Revision							
Au1250																																
Def.	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	UNPRED							
Au1210																																
Def.	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	UNPRED							

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:24	Company Options	System-on-a-Chip (SoC) identification: 0 Au1000 1 Au1500 2 Au1100 3 Au1550 4 Au1200, Au1250 5) Au1210	R	4 (Au1250) 5 (Au1210)
23:16	Company ID	Company ID assigned by MIPS Technologies. RMI's ID is 3.	R	3
15:8	Processor Core ID	Identifies the core revision: 0 Reserved 1 Au1 revision 1 2 Au1 revision 2	R	2
7:0	Revision	Contains a manufacturing-specific revision level.	R	SoC specific

2.8.15 Configuration Register 0 (CP0 Register 16, Select 0)

This register is required for various processor configurations and capabilities.

Config0 **CP0 Register 16, Select 0**

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

Bits	Name	Description	R/W	Default
31	M	Denotes Config1 register available at select 1.	R	1
30:26	CT	Reserved.	R/W	0
25	DD	Reserved.	R/W	0
24	CD	Reserved.	R/W	0
23	UM	Reserved.	R/W	0
22	WD	Write Buffer Disable. See Section 2.4 "Write Buffer" on page 28. 0 Enable write buffer. 1 Disable write buffer.	R/W	0
21	NM	Merge Latch Disable. See Section 2.4.2 "Write Buffer Merging" on page 29. 0 Enable merging. 1 Disable merging.	R/W	0
20	SM	Serial Mode Enable. 0 Allow instruction pipelining (normal operation). 1 Serialize instructions (no pipelining).	R/W	0
19	OD	Bus Transaction Overlapping Disable. 0 Allow overlapping of system bus transactions. 1 Disable overlapping of system bus transactions.	R/W	0
16	TM	Reserved.	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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
6:3	—	Reserved.	R	0
2:0	K0	CCA of KSEG0.	R/W	3

2.8.16 Configuration Register 1 (CP0 Register 16, Select 1)

This register is required for various processor configurations and capabilities.

Config1 **CP0 Register 16, Select 1**

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

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

2.8.17 Load Linked Address Register (CP0 Register 17, Select 0)

This register provides the address of the most recent *load linked* instruction.

LLAddr **CP0 Register 17, Select 0**

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

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

2.8.18 Data WatchLo Register (CP0 Register 18, Select 0)

These registers are the interface to the data watchpoint facility.

WatchLo **CP0 Register 18, Select 0**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:3	VAddr	The virtual address to match.	R/W	UNPRED
2	—	Reserved.	R	0
1	R	Watch Exceptions for Reads. 0 Disable watch exceptions for loads. 1 Enable watch exceptions for loads that match the address.	R/W	0
0	W	Watch Exceptions for Writes. 0 Disable watch exceptions for stores. 1 Enable watch exceptions for stores that match the address.	R/W	0

2.8.19 Instruction WatchLo Register (CP0 Register 18, Select 1)

These registers are the interface to the instruction watchpoint facility.

IWatchLo **CP0 Register 18, Select 1**

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

Bits	Name	Description	R/W	Default
31:3	VAddr	The virtual address to match.	R/W	UNPRED
2	I	Watch Exceptions for Instructions. 0 Disable watch exceptions for instruction accesses. 1 Enable watch exceptions for instruction accesses that match the address.	R/W	0
1:0	—	Reserved.	R	0

2.8.20 Data WatchHi Register (CP0 Register 19, Select 0)

These registers are the interface to the data watchpoint facility.

WatchHi **CP0 Register 19, Select 0**

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

Bits	Name	Description	R/W	Default
31	M	Another pair of Watch registers is implemented at the next Select index.	R	1
30	G	If this bit is one, the ASID field is ignored and any address that matches causes a watch exception.	R/W	UNPRED
29:24	—	Reserved.	R	0
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
15:12	—	Reserved.	R	0
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
2:0	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

2.8.21 Instruction WatchHi Register (CP0 Register 19, Select 1)

These registers are the interface to the instruction watchpoint facility.

IWatchHi **CP0 Register 19, Select 1**

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

Bits	Name	Description	R/W	Default
31	—	Reserved.	R	0
30	G	If this bit is one, the ASID field is ignored and any address that matches causes a watch exception.	R/W	UNPRED
29:24	—	Reserved.	R	0
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
15:12	—	Reserved.	R	0
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
2:0	—	Reserved.	R	0

2.8.22 Scratch Register (CP0 Register 22, Select 0)

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

Scratch **CP0 Register 22, Select 0**

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

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

2.8.23 Debug Register (CP0 Register 23, Select 0)

This register is part of the interface to the EJTAG facility.

Debug **CP0 Register 23, Select 0**

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

Bits	Name	Description	R/W	Default
31	DBD	Debug exception in branch delay slot.	R	UNPRED
30	DM	If this bit is a one, then in debug mode.	R	0
29	—	Reserved.	R	0
28	LSNM	Load/stores are performed in the normal fashion when in debug mode.	R/W	0
27:18	—	Reserved.	R	0
17:15	001	EJTAG version 2.5.	R	001
14:10	DExcCode	Cause[ExcCode] for normal exceptions in debug mode.	R	UNPRED
9	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
8	SSt	Enable single step mode.	R/W	0
7:6	—	Reserved.	R	0
5	DINT	Last debug exception was asynchronous debug interrupt.	R	0
4:2	—	Reserved.	R	0
1	DBp	Last debug exception was an SDBPP instruction.	R	0
0	DSS	Last debug exception was a single step.	R	0

2.8.24 DEPC Register (CP0 Register 24, Select 0)

This register is part of the interface to the EJTAG facility.

DEPC **CP0 Register 24, Select 0**

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

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

2.8.25 Data Cache Tag Register (CP0 Register 28, Select 0)

These registers are the interface to the data cache array. This cache interface is unique to the Au1. This register corresponds to the TagLo register in the MIPS32 ISA specification.

DTag **CP0 Register 28, Select 0**

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

Bits	Name	Description	R/W	Default
31:12	TAG	TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero.	R/W	UNPRED
11:10	MRU	Most recently used way.	R/W	UNPRED
9:8	NMRU	Next most recently used way.	R/W	UNPRED
7:6	LRU	Least recently used way.	R/W	UNPRED
5:4	—	Reserved.	R	0
3	D	Cache line is dirty (modified).	R/W	UNPRED
2	S	Cache line is shared (for data cache snoops).	R/W	UNPRED
1	L	Locked. This bit is set by the user to prevent overwriting of the cache line.	R/W	UNPRED
0	V	Cache line valid.	R/W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

2.8.26 Data Cache Data Register (CP0 Register 28, Select 1)

These registers are the interface to the data cache array. This register corresponds to the DataLo register in the MIPS32 ISA specification.

DData **CP0 Register 28, Select 1**

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

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

2.8.27 Instruction Cache Tag Register (CP0 Register 29, Select 0)

These registers are the interface to the instruction cache array. This cache interface is unique to the Au1 core. This register corresponds to the TagHi register in the MIPS32 ISA specification.

ITag **CP0 Register 29, Select 0**

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

Bits	Name	Description	R/W	Default
31:12	TAG	TAG represents bits [31:12] of a physical memory address. Bits [35:32] of the physical address are always zero.	R/W	UNPRED
11:10	MRU	Most recently used way.	R/W	UNPRED
9:8	NMRU	Next most recently used way.	R/W	UNPRED
7:6	LRU	Least recently used way.	R/W	UNPRED
5:2	—	Reserved.	R	0
1	L	Locked. This bit is set by the user to prevent overwriting of the cache line.	R/W	UNPRED
0	V	Cache line valid.	R/W	UNPRED

2.8.28 Instruction Cache Data Register (CP0 Register 29, Select 1)

These registers are the interface to the instruction cache array. This register corresponds to the DataHi register in the MIPS32 ISA specification.

IData **CP0 Register 29, Select 1**

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

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

www.DataSheet4U.com

www.DataSheet4U.com

2.8.29 ErrorEPC Register (CP0 Register 30, Select 0)

This register is required for exception processing.

ErrorEPC **CP0 Register 30, Select 0**

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

Bits	Name	Description	R/W	Default
31:0	ErrorEPC	Error exception program counter.	R/W	UNPRED

2.8.30 DESAVE Register (CP0 Register 31, Select 0)

This register is part of the interface to the EJTAG facility.

DESAVE **CP0 Register 31, Select 0**

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

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

www.DataSheet4U.com

www.DataSheet4U.com

2.9 System Bus

The system bus is the high-performance, low-latency primary bus and coherency point within the Au1210 and Au1250 processors. The Au1 core communicates with peripherals and memories via the system bus. The arrangement of the system bus, the Au1 core, and the integrated peripherals is shown in Figure 2-5.

The system bus is a 36-bit physical address *sysbus_addr*[35:0] and a 32-bit data word *sysbus_data*[31:0]. All system bus transactions use *sysbus_addr*[35:0], *sysbus_data*[31:0], and four byte-mask signals.

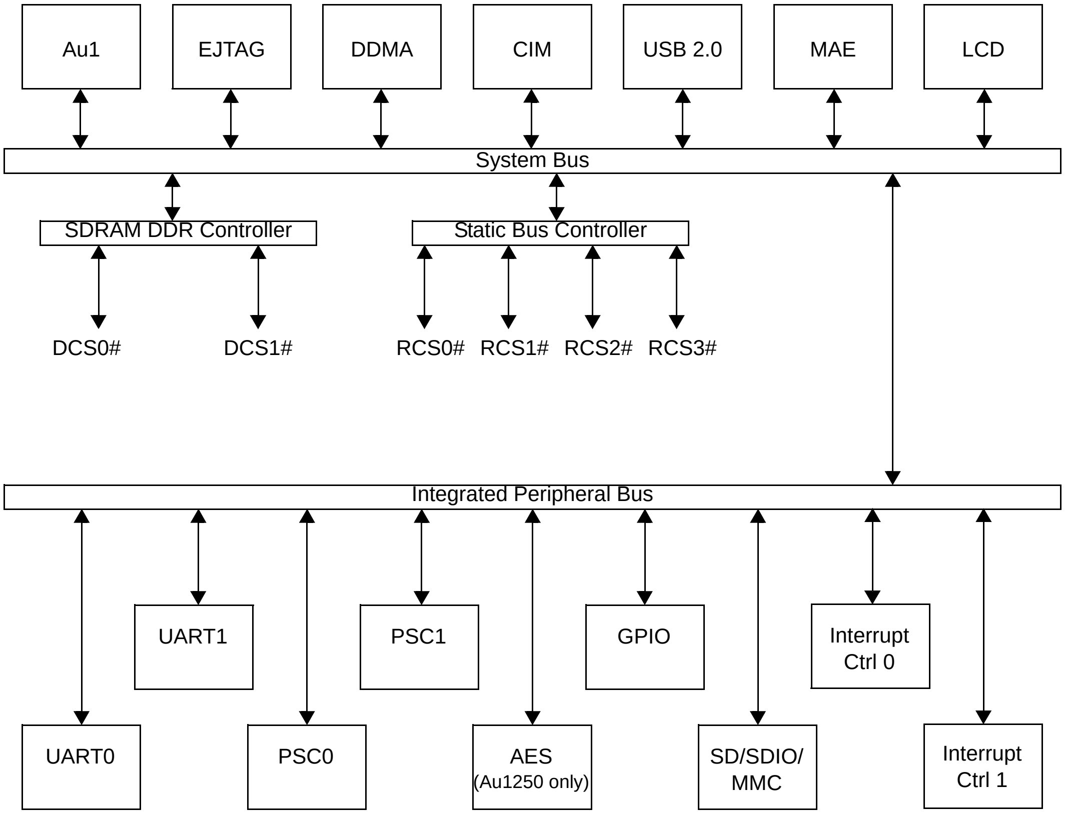


Figure 2-5. Au1210™ and Au1250™ Processors Bus Topology

www.DataSheet4U.com

www.DataSheet4U.com

2.9.1 System Bus Data Format

Data is placed and retrieved by memories and peripherals on the system bus using the data format shown in Figure 2-6.

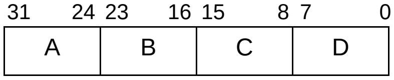


Figure 2-6. System Bus Data Format

Bit 31 is the most significant bit and bit 0 is the least significant bit. Byte masks accompany the data to indicate which bytes within the 32-bit data word are valid.

The system bus is endian-independent: the data format is the same regardless of the endian mode of the processor. Data endianness is handled by the Au1 core and peripherals that access memory (that is, peripherals with DMA). Figure 2-7 demonstrates how the system bus data word contents are interpreted depending upon endian mode of the system bus master (Au1 core or peripheral DMA engines).

For 8-bit bytes, a big endian master interprets the word from left to right (most significant byte to least significant byte), whereas a little endian master interprets the word from right to left (least significant byte to most significant byte).

For 16-bit halfwords, a big endian master interprets the word from left to right (most significant halfword to least significant halfword), whereas a little endian master interprets the word from right to left (least significant halfword to most significant halfword).

For 32-bit words, big endian and little endian masters interpret the data word in the same manner: as a single 32-bit value.

2.9.1.1 Au1 Core Endianness

The Au1 core manipulates data according to endian mode and access size to match data with the system bus data format. The Au1 core endian mode is selected by software at runtime and is controlled by the **sys_endian** register (see Section 10.4.5.3 "Endianness Register" on page 395).

Table 2-8 on page 52 shows the various load/store instructions and how data is manipulated.

Instruction fetches are 32-bit loads. An important side effect of this fact is that software can change the endian mode of the processor at runtime because instruction fetches are unaffected by endian mode.

The data manipulation of the **LWL**, **LWR**, **SWL**, and **SWR** instructions is shown in Table 2-9 on page 52.

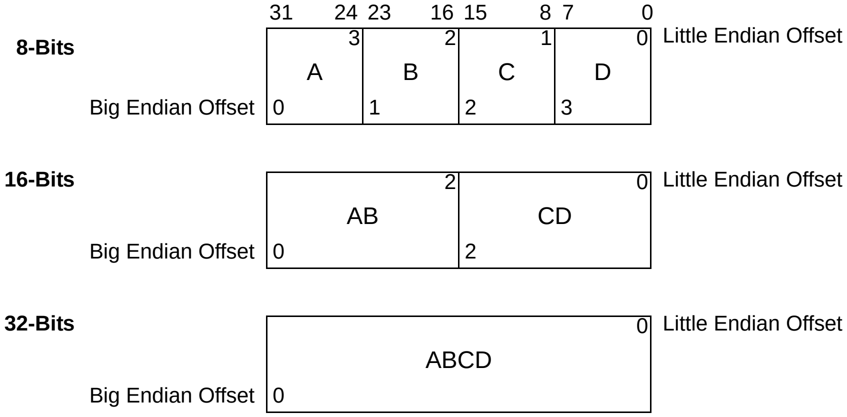


Figure 2-7. Data Endian Manipulation

www.DataSheet4U.com

www.DataSheet4U.com

Table 2-8. Au1 Manipulation of System Bus Data for Endian Mode[1]

	Au1 Little Endian	Au1 Big Endian
Instruction	Result	Result
lb rX,0(rY)	D->rX[7:0]	A->rX[7:0]
lb rX,1(rY)	C->rX[7:0]	B->rX[7:0]
lb rX,2(rY)	B->rX[7:0]	C->rX[7:0]
lb rX,3(rY)	A->rX[7:0]	D->rX[7:0]
lh rX,0(rY)	CD->rX[15:0]	AB->rX[15:0]
lh rX,2(rY)	AB->rX[15:0]	CD->rX[15:0]
lw rX,0(rY)	ABCD->rX[31:0]	ABCD->rX[31:0]
sb rX,0(rY)	rX[7:0]->D	rX[7:0]->A
sb rX,1(rY)	rX[7:0]->C	rX[7:0]->B
sb rX,2(rY)	rX[7:0]->B	rX[7:0]->C
sb rX,3(rY)	rX[7:0]->A	rX[7:0]->D
sh rX,0(rY)	rX[15:0]->CD	rX[15:0]->AB
sh rX,2(rY)	rX[15:0]->AB	rX[15:0]->CD
sw rX,0(rY)	rX[31:0]->ABCD	rX[31:0]->ABCD

[1] 'ABCD' is the system bus data word contents, see Figure 2-6 on page 51.
'rX' is any register.
'rY' is any register that contains a word-aligned address so that the offset indicates the byte address.

Table 2-9. LWL, LWR, SWL and SWR Manipulation of System Bus Data for Endian Mode[1]

	Au1 Little Endian	Au1 Big Endian
Instruction	Result	Result
lwl rX,0(rY)	D->rX[31:24]	ABCD->rX[31:0]
lwl rX,1(rY)	CD->rX[31:16]	BCD->rX[31:8]
lwl rX,2(rY)	BCD->rX[31:8]	CD->rX[31:16]
lwl rX,3(rY)	ABCD->rX[31:0]	D->rX[31:24]
lwr rX,0(rY)	ABCD->rX[31:0]	A->rX[7:0]
lwr rX,1(rY)	ABC->rX[23:0]	AB->rX[15:0]
lwr rX,2(rY)	AB->rX[15:0]	ABC->rX[23:0]
lwr rX,3(rY)	A->rX[7:0]	ABCD->rX[31:0]
swl rX,0(rY)	rX[31:24]->D	rX[31:0]->ABCD
swl rX,1(rY)	rX[31:16]->CD	rX[31:8]->BCD
swl rX,2(rY)	rX[31:8]->BCD	rX[31:16]->CD
swl rX,3(rY)	rX[31:0]->ABCD	rX[31:24]->D
swr rX,0(rY)	rX[31:0]->ABCD	rX[7:0]->A
swr rX,1(rY)	rX[23:0]->ABC	rX[15:0]->AB
swr rX,2(rY)	rX[15:0]->AB	rX[23:0]->ABC
swr rX,3(rY)	rX[7:0]->A	rX[31:0]->ABCD

[1] 'ABCD' is the system bus data word contents, see Figure 2-6 on page 51.
'rX' is any register.
'rY' is any register that contains a word-aligned address so that the offset indicates the byte address.

www.DataSheet4U.com

w w w . D

2.9.1.2 Peripheral Endianness

Peripheral DMA engines use the system bus for reading and writing data. As such, the peripheral needs to manipulate the system bus data words depending upon the endian mode of the peripheral. Some peripherals contain a control bit in the peripheral module that indicates how the peripheral is to interpret the system bus data.

Note: The endian-mode of the peripheral should match the endian-mode of the processor; otherwise, software is required to manipulate the data so that it is in the endian-mode of the peripheral. If the processor and the peripheral both use the same endian, both the processor and the peripheral have the same view of the data as it resides in memory.

For example, in 8-bit mode, the camera interface module reads data from a camera device 1-byte at a time, and must pack 4 bytes of data into a 32-bit word using the same endianess as the processor.

For peripheral DMA engines that access 8- or 16-bit FIFOs, the byte or halfword data is always placed on the system bus in the least-significant byte or halfword lane of the 32-bit system bus data word. That is, the data is right-justified to start at bit 0. For example, the 8-bit DMA channel for UART0 uses bits [7:0]. This behavior matches the register interface used by the Au1 core to read/write the FIFO directly; see Section 2.9.2 "Integrated Peripheral Bus" on page 54.

Table 2-10. Peripheral Manipulation of System Bus Data for Endian Mode

Byte Stream	Little Endian	Big Endian
Byte 0	D of Word 0	A of Word 0
Byte 1	C of Word 0	B of Word 0
Byte 2	B of Word 0	C of Word 0
Byte 3	A of Word 0	D of Word 0
Byte 4	D of Word 1	A of Word 1
Byte 5	C of Word 1	B of Word 1
Byte 6	B of Word 1	C of Word 1
Byte 7	A of Word 1	D of Word 1
Byte 8	D of Word 2	A of Word 2
…	…	…

www.DataSheet4U.com

www.DataSheet4U.com

2.9.2 Integrated Peripheral Bus

The integrated peripherals in the Au1210 and Au1250 processors expose programming registers to the Au1 core on the integrated peripheral bus. The peripheral bus is an internal bus where peripherals that do not arbitrate for the system bus (that is, do not perform DMA) are located. All integrated peripheral registers are 32-bits wide, even though many bits in the registers may not be used.

Software must access all integrated peripheral registers as 32-bits; thus, software accesses to these registers are unaffected by the endian mode of the Au1 core. (8- and 16-bit accesses to the peripheral bus are not supported.)

2.9.3 System Bus Masters

The system bus supports multiple masters: the Au1 core and peripheral DMA engines. The system bus arbitration scheme prevents any master from consuming the entire system bus bandwidth, while permitting low latency access to the system bus for masters that request the bus infrequently (such as peripherals).

The system bus requesters in the Au1210 and Au1250 processors are as follows:

- Slot 0: Au1 core (data cache, instruction cache, write buffer)
- Slot 1: Media acceleration engine (MAE)
- Slot 2: DDMA controller
- Slot 3: LCD controller
- Slot 4: USB 2.0 controller

The Au1 presents a single request to the system bus arbiter for the three possible requesters: 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.

2.9.4 System Bus Arbitration

The arbitration scheme for the system bus is round-robin; that is, each bus master slot has an equal opportunity to obtain access to the system bus. For a particular system bus master X, if no other system bus masters request the bus, bus master X immediately wins the system bus. By contrast, if multiple system bus masters request the bus, bus master X must wait for all other system bus master slots to transfer before it wins the system bus, as shown in Figure 2-8 on page 55.

When a system bus master wins arbitration of the system bus, it performs transfers to/from the integrated peripherals or the memory controllers.

2.9.5 System Bus Clock Source and Power Features

The system bus frequency is derived from the Au1 core frequency (configured in the **sys_cpupll** register). Depending upon the performance and power requirements of the application, the system bus operational frequency can be divided to achieve higher performance or lower power. Furthermore, the system bus offers dynamic power-saving features. For more information, see the **sys_powerctrl** register description in Section 10.4.5.4 "Power Control Register" on page 395.

www.DataSheet4U.com

www.DataSheet4U.com

2.9.6 System Bus Coherency Model

The system bus is the coherency point within the Au1210 and Au1250 processors. A system bus master (the Au1 core or a peripheral DMA engine) marks each transaction as either coherent or non-coherent. System bus transactions marked as coherent are then snooped by all caching masters. (The Au1 data cache is the only caching master implemented on the Au1210 and Au1250 processors.) A system bus 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 system bus 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 system bus transactions are to be marked coherent or non-coherent. If a peripheral is configured for coherent operation, 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, 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 whether to use coherent system bus transactions is left to the application. However, peripheral device drivers using coherent system bus transactions can perform better than drivers not using coherent transactions since the need to writeback the data cache is eliminated.

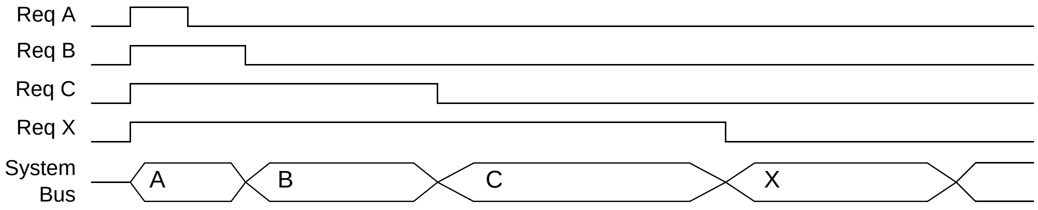


Figure 2-8. System Bus Arbitration

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

3 Memory Controllers

The Au1210™ and Au1250™ processors contain two memory controllers: an SDRAM controller for DDR1 and DDR2 SDRAM devices, and a static bus controller for static devices.

The SDRAM controller (see Section 3.1 on page 58) has the following features:

- Supports DDR1 and DDR2 SDRAM
- Separate I/O power supply supporting 2.5V or 1.8V

The static bus controller (see Section 3.2 on page 77) supports the following devices:

- ROM, SRAM, I/O peripherals
- PCMCIA/Compact Flash devices
- NOR Flash
- NAND Flash
- IDE devices

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 gap in the memory map. With the address configuration options in the Au1210 and Au1250 processors, the 4MB can be relocated to start at 16MB, and the new memory can be located at 0 to allow a 20 MB contiguous memory pool.

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

The system designer has the choice of booting from 16-bit NOR Flash, and 8- or 16-bit NAND Flash. The BOOT signals determine where the processor boots from. See Section 11.3 "Boot" on page 401 for a discussion of the boot configuration.

Table 3-1. Memory Controller Block Base Address

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

www.DataSheet4U.com

www.DataSheet4U.com

3.1 DDR SDRAM Memory Controller

The Au1210™ and Au1250™ processors' SDRAM memory controller supports double data rate DDR1 and DDR2 SDRAM devices. The following features are supported:

- 2.5V DDR1 SDRAM support: DDR200, DDR266, DDR333, and DDR400
- 1.8V DDR2 SDRAM support: DDR2-400 and DDR2-533 (DDR2-533 SDRAM operates at reduced frequencies that match the speed rating of the Au1210 and Au1250 processors, as shown in Table 3-4 on page 73. Table 3-4 shows the supported combinations of CPU frequency, system bus divider, and memory bus divider.
- Fourteen address bits
- Two differential clock output pairs
- Global setting for system bus to external bus ratios 2:1 or 1:1
- 2 full ranks (chip selects)
- 8-entry open bank address tag. Detects and manages up to 8 open device internal memory banks. Above 8, finds and closes least used bank.
- Supports up to 27 address bits, 14 row, 13 column address.
- Supports up to 4 internal banks (device)
- Fixed burst length of 4 data accesses in x16 or x32 configuration.

3.1.1 SDRAM Memory Controller Registers

The SDRAM memory controller configuration registers are shown in Table 3-2. All registers listed in Table 3-2 are 32 bits.

Table 3-2. Memory Controller Configuration Registers

Offset from 0x0 1400 0000 (Physical) 0xB400 0000 (KSEG1)	Register Name	Description
0x0800	mem_sdmode0	DCS0# Timing
0x0808	mem_sdmode1	DCS1# Timing
0x0820	mem_sdaddr0	DCS0# Address Configuration and Enable
0x0828	mem_sdaddr1	DCS1# Address Configuration and Enable
0x0840	mem_sdconfiga	Global Configuration Register A
0x0848	mem_sdconfigb	Global Configuration Register B
0x0850	mem_sdstat	Status register for error reporting
0x0880	mem_sdwrmd0	Write data to mode configuration register for SDRAM connected to DCS0#
0x0888	mem_sdwrmd1	Write data to mode configuration register for SDRAM connected to DCS1#
0x08C0	mem_sdprecmd	Issue PRECHARGE to all enabled chip selects
0x08C8	mem_sdautoref	Issue REFRESH to all enabled chip selects
0x08D0	mem_sdsref	Toggle self refresh mode
0x08D4 - 0x08FF	—	Reserved

www.DataSheet4U.com

www.DataSheet4U.com

3.1.1.1 Chip Select Mode Configuration Registers

The format of the Chip Select Mode Configuration registers (**mem_sdmode*n***) is as follows:

mem_sdmode0 **Offset = 0x0800**

mem_sdmode1 **Offset = 0x0808**

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
	▨					Twtr			▨	Twr			Tras				▨	Trp			▨	Trcdwr			▨	Trcdrd			▨	Tcas		
Def.						1	1	1		1	1	1	1	1	1	1		1	1	1		1	1	1		1	1	1		1	1	1

Bit(s)	Name	Description	R/W	Default
31:27	—	Reserved.	—	—
26:24	Twtr	Specifies the write to read turnaround time. (Twtr + 1) is the actual number of clock cycles.	R/W	0x7
23	—	Reserved.	—	—
22:20	Twr	Specifies the write recovery time. This is the last data for a write to a precharge. May also be referred to as Tdpl. (Twr + 1) is the actual number of clock cycles.	R/W	0x7
19:16	Tras	Specifies the minimum delay from a ACTIVATE to a PRECHARGE command. (Tras + 1) is the actual number of clock cycles.	R/W	0xF
15	—	Reserved.	—	—
14:12	Trp	Specifies the time from PRECHARGE to the next ACTIVATE command. (Trp + 1) is the actual number of clock cycles.	R/W	0x7
11	—	Reserved.	—	—
10:8	Trcdwr	Specifies the RAS to CAS delay for writes. (Trcdwr + 1) is the actual number of clock cycles.	R/W	0x7
7	—	Reserved.	—	—
6:4	Trcdrd	Specifies the RAS to CAS delay for reads. (Trcdrd + 1) is the actual number of clock cycles.	R/W	0x7
3	—	Reserved.	—	—
2:0	Tcas	Specifies the minimum CAS latency timing in clock units. 000 Reserved 001 2.5 (for DDR1 only) 010 Reserved 011 2. If Tcas = 2 is used for DDR2, it must be the same for both chips selects, if both are used. Otherwise, each chip select can be programmed with independent values for Tcas. 100 3 101 4 110 5 (default) 111 6	R/W	0x7

www.DataSheet4U.com

www.DataSheet4U.com

3.1.1.2 Chip Select Address Configuration Registers

The Chip Select Address Configuration registers (**mem_sdaddr*n***) assign an address range for each chip select. As shown below, each register contains a base address, an address comparison mask, and an enable bit. Each register also contains the row and column address sizes, and the address bit ordering (bank relocate).

mem_sdaddr0 **Offset = 0x0820**

mem_sdaddr1 **Offset = 0x0828**

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
		BR	RS			CS						E	CSBA										CSMASK									
Def.	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit(s)	Name	Description	R/W	Default
31	—	Reserved.	—	—
30	BR	Bank relocate. 0 Address is defined in order of [bank, row, column]. 1 Address is defined in order of [row, bank, column].	R/W	0
29:28	RS	Specifies the number of bits in the row address. 00 11 01 12 10 13 11 14	R/W	0
27	—	Reserved.	—	—
26:24	CS	Specifies the number of bits in the column address. 000 Reserved 001 8 010 9 011 10 100 11 101 12 110 13 111 Reserved	R/W	0x1
23:21	—	Reserved.	—	—
20	E	Enable. 0 Disable chip select. 1 Enable chip select. NOTE: All chip selects are disabled by default.	R/W	0
19:10	CSBA	Chip select base address *cs_base*[31:22]. *cs_base* is the physical base address for the chip select. *cs_base*[35:32] = 0. *cs_base*[31:22] = CSBA. *cs_base*[21:0] = 0.	R/W	0x3FF
9:0	CSMASK	Chip select address mask *cs_mask*[31:22]. *cs_mask* is used to decode the chip select. *cs_mask*[35:32] = 0xF. *cs_mask*[31:22] = CSMASK. *cs_mask*[21:0] = 0.	R/W	0x3FF

www.DataSheet4U.com

www.DataSheet4U.com

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

(*sysbus_addr* & *cs_mask*) == *cs_base*

where

sysbus_addr: 36-bit physical address output on the internal system bus (from the TLB for memory-mapped regions)

cs_mask: Address comparison mask taken from CSMASK

cs_base: Chip select base address taken from CSBA

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

3.1.1.3 Global Configuration Register A

This register contains the refresh timing for all chip selects. See Section 3.1.4.2 on page 73 for refresh timing programming considerations. This register also contains bits that enable the SDRAM clocks. Each clock is identical in frequency and phase. The format of the global configuration register is as follows:

mem_sdconfiga **Offset = 0x0840**

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
	E	▨	CE		RPT		Trc						▨	▨	REF																	
Def.	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit(s)	Name	Description	R/W	Default
31	E	Enable Refresh. 0 Disable refresh. 1 Enable refresh.	R/W	0
30	—	Reserved.	—	—
29:28	CE[1:0]	Clock Enables for DCK[1:0] and DCK[1:0]#. 00 Disable clocks. 01 Enable DCK0 and DCK0#. 10 Reserved 11 Enable DCK[1:0] and DCK[1:0]#.	R/W	0
27:26	RPT	Specifies number of refresh cycles performed for each refresh time interval. 00 1 refresh per interval 01 2 refreshes per interval 10 4 refreshes per interval 11 6 refreshes per interval	R/W	0x0
25:20	Trc	Specifies the minimum time from the start of an auto refresh cycle to an activate command for all chip selects. The value should reflect the larger of the specified TRC and TRFC based on the selection of RPT. (Trc + 1) is the actual number of clock cycles.	R/W	0x3F
19:18	—	Reserved.	—	—
17:0	REF	Specifies the maximum distributed refresh interval in system bus clocks for all enabled ranks. For an 64 ms refresh period with a 198MHz system bus (396MHz core) and 8192 rows with RPT = 0, the value for REF would be 0x60A: (64 ms / 8192) / (1 /198 MHz).	R/W	0x3FFFF

www.DataSheet4U.com

www.DataSheet4U.com

3.1.1.4 Global Configuration Register B

This register contains additional global configuration bits for the SDRAM memory controller. DDR1 or DDR2 memory type is selected in **mem_sdconfigb**[MT]:

- MT = 0 selects DDR1
- MT = 1 selects DDR2

System Note: Although the memory type is programmed for each chip select, systems must use either DDR1 or DDR2, not both.

Core priority 2, **mem_sdconfigb**[C2], is a priority scheme for media applications to allow the processor to maximize throughput between the MAE and the LCD controller. Data via DMA transfers from DDR memory is placed on the system bus for the MAE to process and the LCD controller to display.

The assignment of LCD and MAE to highest priority can be overwritten temporarily by **mem_sdconfigb**[BB] for the purpose of configuration or control of any external device that requires back-to-back "atomic" commands or data.

Setting this bit accordingly, the system bus can arbitrate to give this activity the highest priority in the case of media content playback, permitting other peripherals and tasks access at a lower priority.

The arbitration count, **mem_sdconfigb**[AC], can be used to ensure the lowest priority requester receives a grant. It specifies the number of grants to the 2nd highest priority requester before the lowest priority requester must get the grant.

For slower bus configurations (such as clock periods of 7.5-10ns), **mem_sdconfigb**[FS] must be set to sample data on the bus while valid. The FS bit takes into account pad delays and clock skew for slower accesses to sample data sooner, while data is valid. FS is not necessary for faster clock speeds and must be used only when necessary.

mem_sdconfigb **Offset = 0x0848**

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
	CR	BW	MT	PSEL	C2	AC		HP	PM		CKECNT		BB	▨	DS	FS	PDX		CKEMIN		CB	TXARD			BA	▨	TXSR					
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	CR	Clock Ratio. CR is a clock divider between the system bus clock and the SDRAM bus clock. 0 Divide the system bus clock frequency by 2 to derive the SDRAM bus clock. 1 Run the SDRAM bus clock at the same frequency as the system bus clock. Program CR as part of the initialization process; see Section 3.1.4.1 "SDRAM Bus Clock" on page 72. Clocking must be enabled in **mem_sdconfiga**[CE] before modifying CR.	R/W	0
30	BW	SDRAM Bus Width. 0 32-bit data bus 1 16-bit data bus	R/W	0
29	MT	Memory Type. 0 DDR1 1 DDR2	R/W	0
28	PSEL	Pin select determines which address pin to use for auto precharge. 0 Address pin 10 1 Address pin 8	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bit(s)	Name	Description	R/W	Default
27	C2	Core Priority 2. System bus arbitration configuration bit to prioritize peripherals and core usage on the system bus. Optimization for high bandwidth applications involving media decode playback and graphics rendering. 0 LCD-highest, MAE-next, Core-lowest 1 LCD-highest, Core-next, MAE-lowest	R/W	0
26:25	AC	Arbitration Count. Specifies the count value that allows the lowest priority requester to get a grant. Represents the number of grants to the 2nd highest priority requester before the lowest priority requester must get the grant. 00 1 01 2 10 4 11 8	R/W	0
24	HP	Half PLL Mode. 0 Normal operation 1 Enter half PLL mode to lower power consumption. When the bit transitions, bus transactions are disabled for a number of clocks determined by (TXSR +1) * 16.	R/W	0
23:22	PM	Power Save Modes. 00 Full power 01 Negate DCKE to the memory when the chip select is idle. This saves system power by putting the memory either into the active power-down or precharge power-down state. The controller still performs refreshes. 10 Idle, Precharge then DCKE negated 11 Idle, DCKE negated. If DCKE is negated for a period exceeding (CKECNT+1)*16, precharge issued then DCKE negated.	R/W	0
21:20	CKECNT	Idle count before precharge of DDR banks. Enabled when PM = 3 to place the memory into a precharge power-down state. Idle count = (CKE_CNT)*16	R/W	0
19	BB	Back-to-back Access. 0 Normal operation 1 MAE and LCD requests for bus access are temporarily negated. Allows for back-to-back controller-to-peripheral communications without interruption from MAE or LCD. Example: DDR precharge command followed by refresh.	R/W	0
18	—	Reserved.	—	—
17	DS	Drive strength for the SDRAM bus signals. 0 Reduced strength 1 Full strength The DRVSEL signal and the DS bit both affect the SDRAM bus signal strength: For reduced-strength signals, DRVSEL must be tied high *and* DS must be cleared; otherwise, the signals are driven at full strength.	R/W	0
16	FS	Fast Sample. 0 Default 1 Controller samples read data 1 clock earlier than default.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bit(s)	Name	Description	R/W	Default
15:14	PDX	Power-down Exit. Number of clocks from DCKE assertion to a valid command is (PDX+1).	R/W	0
13:12	CKEMIN	Minimum clocks required for DCKE to assert or negate. For DDR, value of 0 is default. Total number of clocks is (CKEMIN +1).	R/W	0
11	CB	Comparator Bypass. 0 Use the DVREF comparator for input data. 1 Bypass the DVREF comparator for input data. The CB bit takes effect immediately.	R/W	0
10:8	TXARD	Counter for exit active power-down to a read command. Value in clocks is (TXARD + 1).	R/W	0
7	BA	Block Access. When BA is set, the memory controller prevents all transactions for ((TXSR+1) * 16) memory clocks. This bit must be used when changing anything that affects clocks that go to memory: i.e., Core PLL multiplier, system bus divider, CR bit. After initiating an access block, software must clear BA.	R/W	0
6	—	Reserved.	—	—
5:0	TXSR	Number of NOP commands required on exit from self refresh mode. The counter is also used when BA is set. ((TXSR+1) * 16) is the actual number used.	R/W	0

3.1.1.5 Status Register

This register contains status information for the SDRAM memory controller and is read only. Error bits are cleared by writing a 1. Writing a 0 has no effect.

mem_sdstat **Offset = 0x0850**

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
							LPM	SLF																								
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
31:26	—	Reserved.	—	—
25	LPM	Low power mode bit reflects the state of the DCKE signal.	R	0
24	SLF	The bit is set when memories are in self refresh mode, which occurs when the self refresh register, **mem_sdsref,** is written. DCKE is negated when this bit is set.	R	0
23:0	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

3.1.1.6 External Mode Register Access

These registers allow software to directly write to the mode registers in the external SDRAM devices connected to each chip select. This can be used in initialization sequences that require certain operations be performed in a deterministic order.

The Tmrd parameter is fixed at 4 clocks.

The chip select must be enabled before accessing the corresponding **mem_sdwrmd*n***.

When the DDR SDRAM controller is configured for DDR2, the extended mode register posted CAS additive latency (AL) must be cleared for each external DDR2 device.

mem_sdwrmd0 **Offset = 0x0880**

mem_sdwrmd1 **Offset = 0x0888**

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

Bit(s)	Name	Description	R/W	Default
31:30	BA[1:0]	Bank Address. These bits are reflected on the DBA[1:0] signals. They can be used to write to the extended mode register (for battery RAM, for example). These bits must be cleared otherwise.	W	UNPRED
29:13	—	Reserved.	W	UNPRED
12:0	WM	Write Mode Data. These bits are reflected on the address signals DA[12:0]. The value written to this register is written to the external SDRAM mode register for the corresponding chip select.	W	UNPRED

3.1.1.7 Precharge All Command Register

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

mem_sdprecmd **Offset = 0x008C0**

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
	PA																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

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

www.DataSheet4U.com

www.DataSheet4U.com

3.1.1.8 Auto Refresh Command Register

Writing to this register performs an AUTO REFRESH command on all enabled chip selects. This can be used for initialization sequences requiring specific operations to be performed in a deterministic order. See also Section 3.1.4.2 "Auto Refresh" on page 73.

Reading from the **mem_sdautoref** register returns the current value of the refresh timer.

mem_sdautoref **Offset = 0x08C8**

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

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

3.1.1.9 Self Refresh Toggle Command Register

Writing to this register toggles the self refresh mode. Enter self refresh mode as follows:

1) The first write (and subsequent odd-numbered writes) to **mem_sdsref** causes the controller to issue a SELF REFRESH command to all enabled chip selects.

2) System software can then poll the self-refresh flag (**mem_sdstat**[SLF]) to confirm the command has completed.

3) When **mem_sdstat**[SLF] = 1, the memory controller negates the clock-enable signal DCKE, and software can safely disable memory clocks by clearing **mem_sdconfiga**[CE].

Exit self refresh mode as follows:

1) System software must first enable memory clock(s) in **mem_sdconfiga**[CE].

2) A second write (and subsequent even-numbered writes) to **mem_sdsref** causes the memory controller to clear **mem_sdstat**[SLF] and assert the clock-enable signal DCKE.

3) On the assertion of DCKE, the memory controller waits ((**mem_sdconfigb**[TXSR]+1) * 16) clocks before issuing a new command.

System Note: V_{DDI} powers the SDRAM controller. If V_{DDI} is disabled as part of a Sleep or Hibernate sequence and the SDRAM controller is in self-refresh mode, the SDRAM controller state machine exits self-refresh mode, even though the DCKE signal remains negated and the actual SDRAM memory continues to refresh itself. When V_{DDI} is re-enabled, the first write to **mem_sdsref** causes the SDRAM controller state machine to re-enter self-refresh mode. (The DCKE signal remains negated.) A second write to **mem_sdsref** is required for the SDRAM controller to fully exit self-refresh mode and assert DCKE.

mem_sdsref **Offset = 0x08D0**

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
	SRT																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	SRT	Self refresh toggle. Writing any value to SRT toggles the self refresh mode on all enabled chip selects.	W	UNPRED

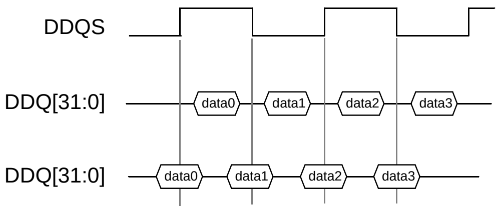
www.DataSheet4U.com

www.DataSheet4U.com

3.1.2 DDR SDRAM Timing

DDR SDRAM operates with a differential clock: DCK and DCK#. Commands, address and control signals are registered at every positive edge of DCK.

A bidirectional data strobe (DDQS) is transmitted externally along with the data for data capture at the receiver. The Au1210 and Au1250 processors drive DDQS during write cycles. For reads, the DDR memory device drives DDQS. DDQS is edge-aligned with data during read cycles and center-aligned with data during write cycles, as shown in Figure 3-1.



READ: DDQS edges straddle data.

WRITE: DDQS edges center on data.

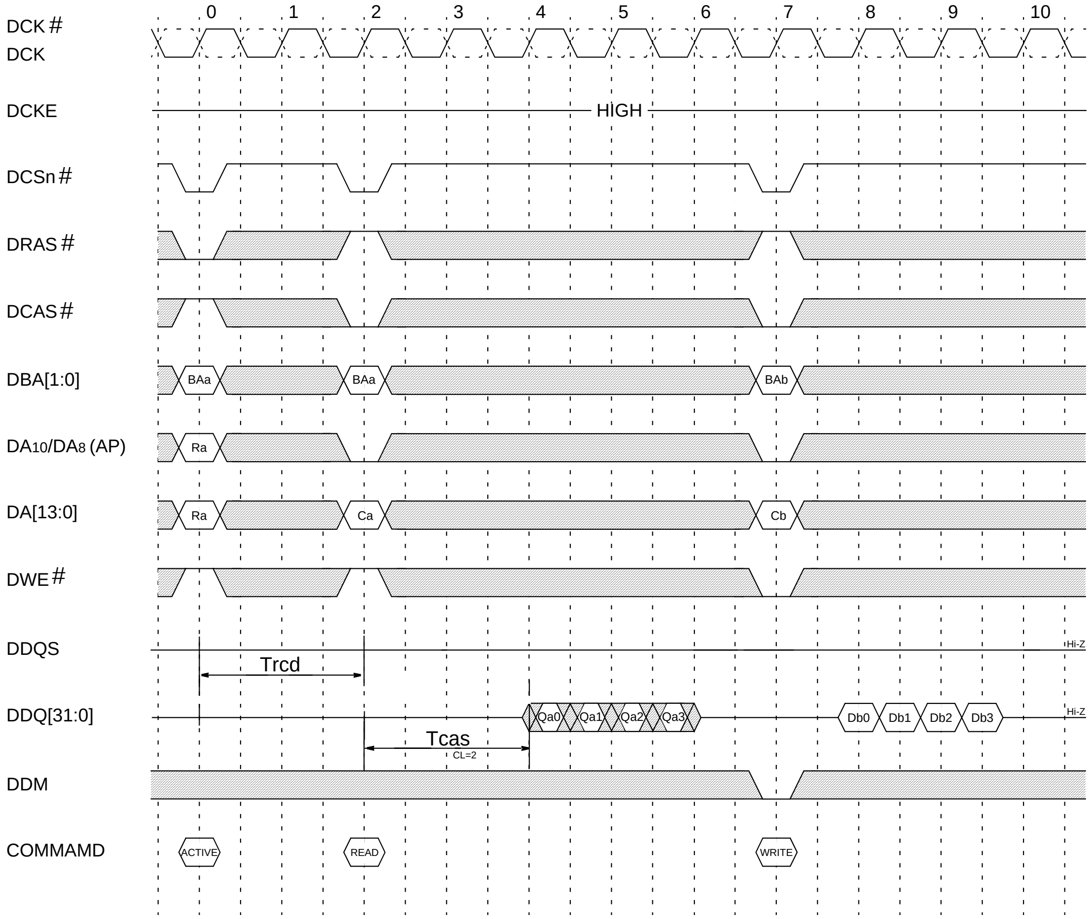
Figure 3-1. DDR Strobe Timing for Reads and Writes

Transactions begin with the registration of an ACTIVE command, which is then followed by a READ or WRITE command. The address bits registered with the READ or WRITE command select the bank and the starting column location for the burst access. Read and write transactions to the DDR SDRAM start at a selected location and continue for a programmed number of locations in a programmed sequence.

www.DataSheet4U.com

www.DataSheet4U.com

The following figures present basic access timing for DDR SDRAM: WRITE interrupted by READ timing, WRITE followed by PRECHARGE timing, and REFRESH timing. The timing diagrams are presented to concisely display the different SDRAM timing parameters. The bus behavior may differ from that displayed.

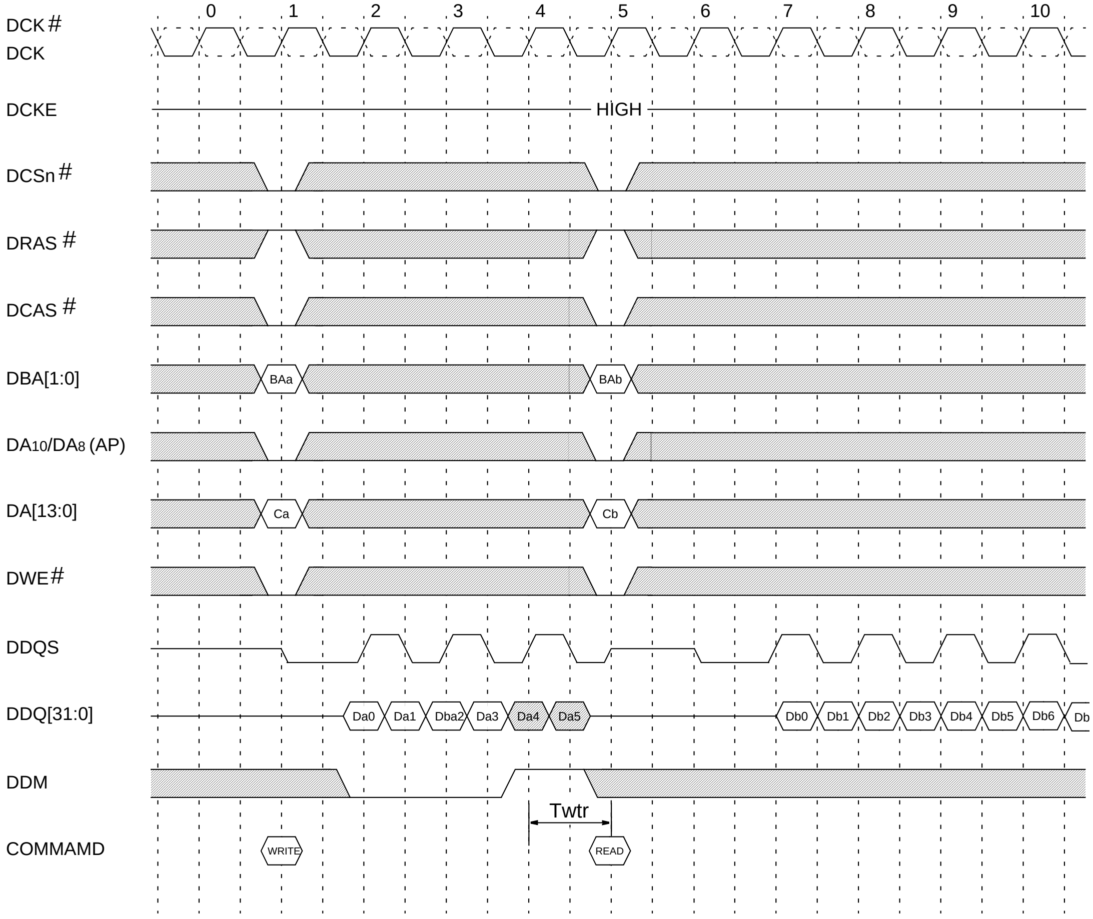


1. Trcd = 1 (2 DDR CLK cycles)
2. Tras = 3 (4 DDR CLK cycles)
3. Tcas = 3 (2 DDR CLK cycles)

Figure 3-2. DDR SDRAM Basic Access Timing

www.DataSheet4U.com

www.DataSheet4U.com

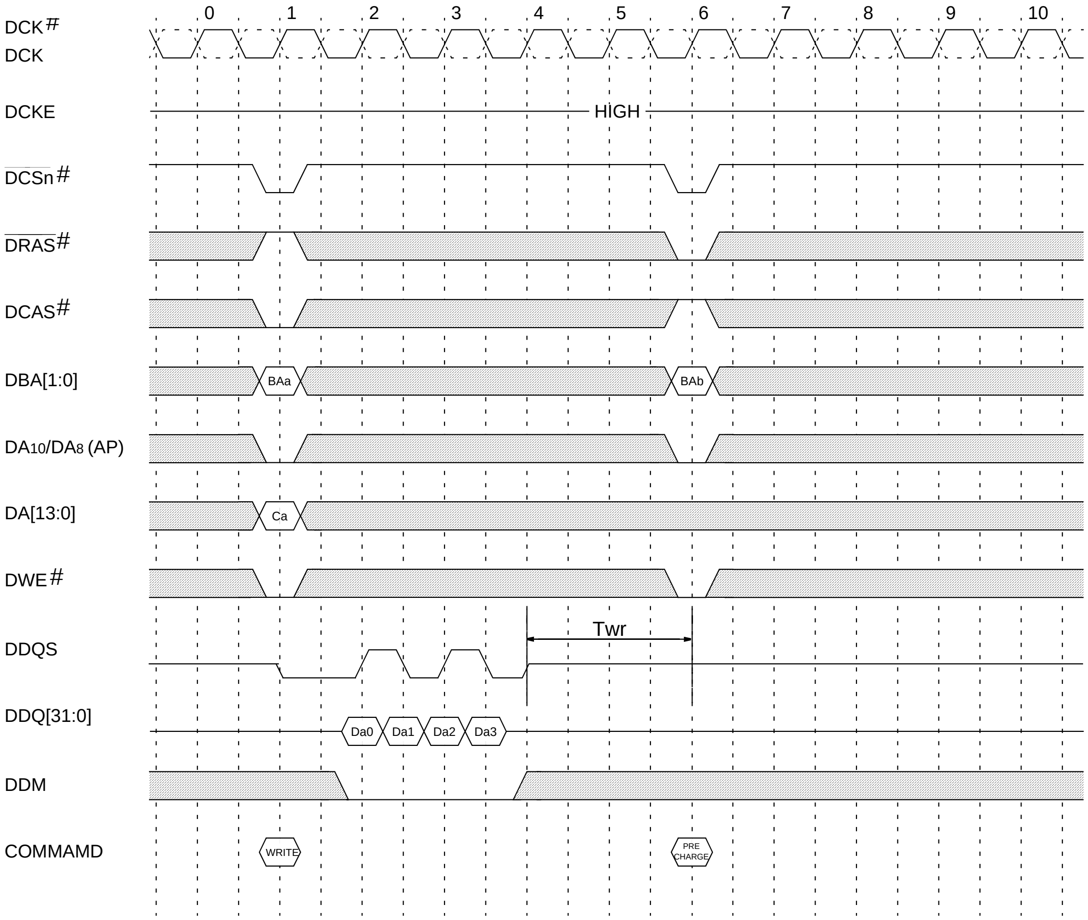


1. Twtr = 0 (1 DDR CLK cycle)

Figure 3-3. DDR SDRAM Write Interrupted by Read Timing

www.DataSheet4U.com

www.DataSheet4U.com



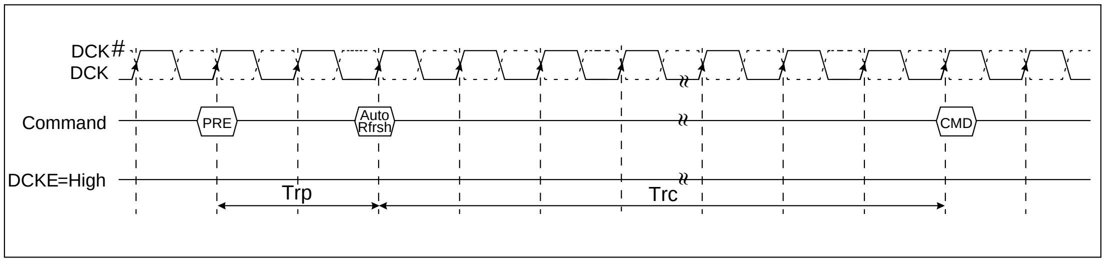
1.Twr = 1 (2 DDR CLK cycles)

Figure 3-4. DDR SDRAM Write Followed by Precharge Timing

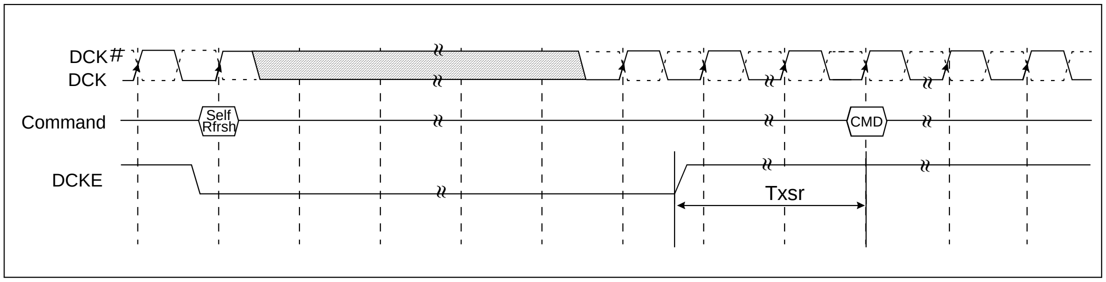
www.DataSheet4U.com

www.DataSheet4U.com

Auto Refresh



Self Refresh



1. The Txsr field determines the number of NOP commands required from Self Refresh to bank active command. A write (or Read) command can be applied as long as Trcd is satisfied after any bank active command.

Figure 3-5. DDR SDRAM Auto and Self Refresh Timing

www.DataSheet4U.com

www.DataSheet4U.com

3.1.3 Reset Operations

3.1.3.1 Reset Configuration

On a reset, all registers are reset to default values. All external output signals are driven to inactive states. The DDQS signal is driven low.

3.1.3.2 Power-on Reset

During a power-on reset, all registers are reset to default values. All external output signals are driven to deterministic states.

3.1.3.3 Runtime Reset

Memory clocks continue until a falling edge of reset is detected. When a reset is detected, all registers return to default values and all output signals are driven to inactive states.

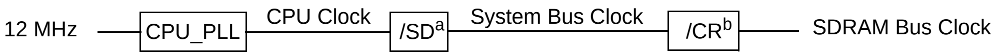
3.1.4 Programming Considerations

3.1.4.1 SDRAM Bus Clock

The Au1210 and Au1250 processors provide programmable dividers to generate different SDRAM bus clocks derived from the CPU clock, as shown in Figure 3-6.

DDR Clock Configurations

Table 3-3 shows the supported combinations of CPU frequency, system bus divider, memory bus divider, and memory data bus width for DDR1.



NOTES:

a. SD is a programmable field in the **sys_powerctrl** register as described in Section 10.4.5 "Power Management Registers" on page 393. SD can be 2, 3, or 4.

b. CR is a programmable field in the **mem_sdconfigb** register as described in Section 3.1.1.4 "Global Configuration Register B" on page 62. CR can be 1 or 2.

c. Supported combinations of SD and CR are listed in Table 3-3 on page 72 and Table 3-4 on page 73.

Figure 3-6. SDRAM Bus Clock Topology

Table 3-3. Clock Configurations for DDR1

Processor	CPU (MHz)	sys_powerctrl [SD]	System Bus (MHz)	mem_sdconfigb [CR]	DDR (MHz)	DDR Bus Width
Au1250	600	3	200	1	200	32/16
Au1250	600	3	200	2	100	32/16
Au1250	600	4	150	1	150	32/16
Au1250	492	2	246	2	123	32/16
Au1250	492	3	164	1	164	32/16
Au1250	492	4	123	1	123	32/16
Au1210/Au1250	396	2	198	2	99	32/16
Au1210/Au1250	396	2	198	1	198	32/16
Au1210/Au1250	396	3	132	1	132	32/16
Au1210/Au1250	396	4	99	1	99	32/16
Au1210	324	2	162	2	81	32/16
Au1210	324	2	162	1	162	32/16
Au1210	324	3	108	1	108	32/16
Au1210	324	4	81	1	81	32/16

www.DataSheet4U.com

www.DataSheet4U.com

Table 3-4 shows the supported combinations of CPU frequency, system bus divider, memory bus divider, and memory data bus width for DDR2.

3.1.4.2 Auto Refresh

The SDRAM memory controller supports only the Auto Refresh mode, distributing the refresh over time, with either 1, 2, 4, or 6 refresh cycles per refresh time interval (**mem_sdconfiga**[RPT]). Therefore the refresh interval **mem_sdconfiga**[REF] must be the maximum refresh time multiplied by RPT and then divided by the number of rows. If RPT is non-zero (multiple refreshes), the **mem_sdconfiga**[Trc] field must be the greater of the specified TRC or TRFC. Also, the time from precharge to refresh is either two or four clocks depending on **mem_sdmode*n***[Trp]: If Trp = 0 or 1, there are two clock cycles; otherwise, there are four.

Table 3-4. Clock Configurations for DDR2

Processor	CPU (MHz)	sys_powerctrl [SD]	System Bus (MHz)	mem_sdconfigb [CR]	DDR (MHz)	DDR Bus Width
Au1250	600	3	200	1	200	32/16
Au1250	600	3	200	2	100	32/16
Au1250	600	4	150	1	150	32/16
Au1250	492	2	246	2	123	32/16
Au1250	492	2	246	1	246	32/16
Au1250	492	3	164	1	164	32/16
Au1250	492	4	123	1	123	32/16
Au1210/Au1250	396	2	198	2	99	32/16
Au1210/Au1250	396	2	198	1	198	32/16
Au1210/Au1250	396	3	132	1	132	32/16
Au1210/Au1250	396	4	99	1	99	32/16
Au1210	324	2	162	2	81	32/16
Au1210	324	2	162	1	162	32/16
Au1210	324	3	108	1	108	32/16
Au1210	324	4	81	1	81	32/16

www.DataSheet4U.com

www.DataSheet4U.com

3.1.5 DDR SDRAM Signals

Table 3-5. DDR SDRAM Signals

Signal	In/	Description
DA[13:0]	O	Address Outputs. A0-A13 are sampled during the ACTIVE command (row-address A0-A13) 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.
DBA[1:0]	O	Bank Address Outputs. DBA1 and DBA0 define to which bank the ACTIVE, READ, WRITE, or PRECHARGE command is being applied.
DDQ[31:0]	I/O	SDRAM Data Bus. During a hardware reset the SDRAM data bus cycles from low voltage to hi-Z and then low as follows: 0 after V_{DDXOK} is asserted. Tristated when V_{DDI} is on and RESETOUT# is asserted. 0 after hardware reset sequence is complete.
DDQS[3:0]	I/O	Input/Output Strobe. The Au1210 and Au1250 processors drive DDQS during writes, and the memory device drives DDQS during reads. It is edge-aligned with read data and center-aligned with write data. DDQS0 strobes DDQ[7:0]. DDQS1 strobes DDQ[15:8]. DDQS2 strobes DDQ[23:16]. DDQS3 strobes DDQ[31:24].
DDM[3:0]	O	Output Mask. DDM is a data mask signal for writes. DDM0 masks DDQ[7:0]. DDM1 masks DDQ[15:8]. DDM2 masks DDQ[23:16]. DDM3 masks DDQ[31:24].
DRAS	O	Command Outputs. DRAS#, DCAS#, and DWE# (along with DCSn#) define the command being sent to the SDRAM rank.
DCAS	O	
DWE	O	
DCK[1:0]#, DCK[1:0]	O	Clock Outputs.
DCS[1:0]	O	Programmable Chip Selects.
DCKE	O	Clock Enable for SDRAM.
DRVSEL	I	Drive strength select for the SDRAM bus signals. 0 Full strength 1 Reduced strength The DRVSEL signal and **mem_sdconfigb**[DS] both affect the SDRAM bus signal strength: For reduced-strength signals, DRVSEL must be tied high *and* DS must be cleared; otherwise, the signals are driven at full strength.
DVREF	I	SDRAM voltage reference. Connect to V_{DDY}/2. Use a voltage divider circuit with two equal resistors between V_{DDY} and ground. Also place a 0.1 µF capacitor in parallel with each resistor.

www.DataSheet4U.com

www.DataSheet4U.com

3.1.6 Hardware Considerations

The SDRAM controller supports up to two ranks of SDRAM. See Section 14.4.1 "DDR Timing and Loading" on page 449 for loading considerations.

Complementary DQS signals are optional for DDR2. The Au1210 and Au1250 processors only have a single DQS signal for each eight DQ bits.

The Au1210 and Au1250 processors do not have on-die termination (ODT) or an ODT output signal. Both processors are designed for portable systems where trace lengths are short and power dissipation is important. It is possible to use ODT on DDR2 memories to improve signal integrity. Software can set the ODT resistor value by using **mem_sdwrmd*n*** to write to the mode register in the external DDR2 SDRAM device.

An example DDR connection is shown in Figure 3-7:

- For DDR1, V_{DDY} = 2.5 V.
- For DDR2, V_{DDY} = 1.8 V.

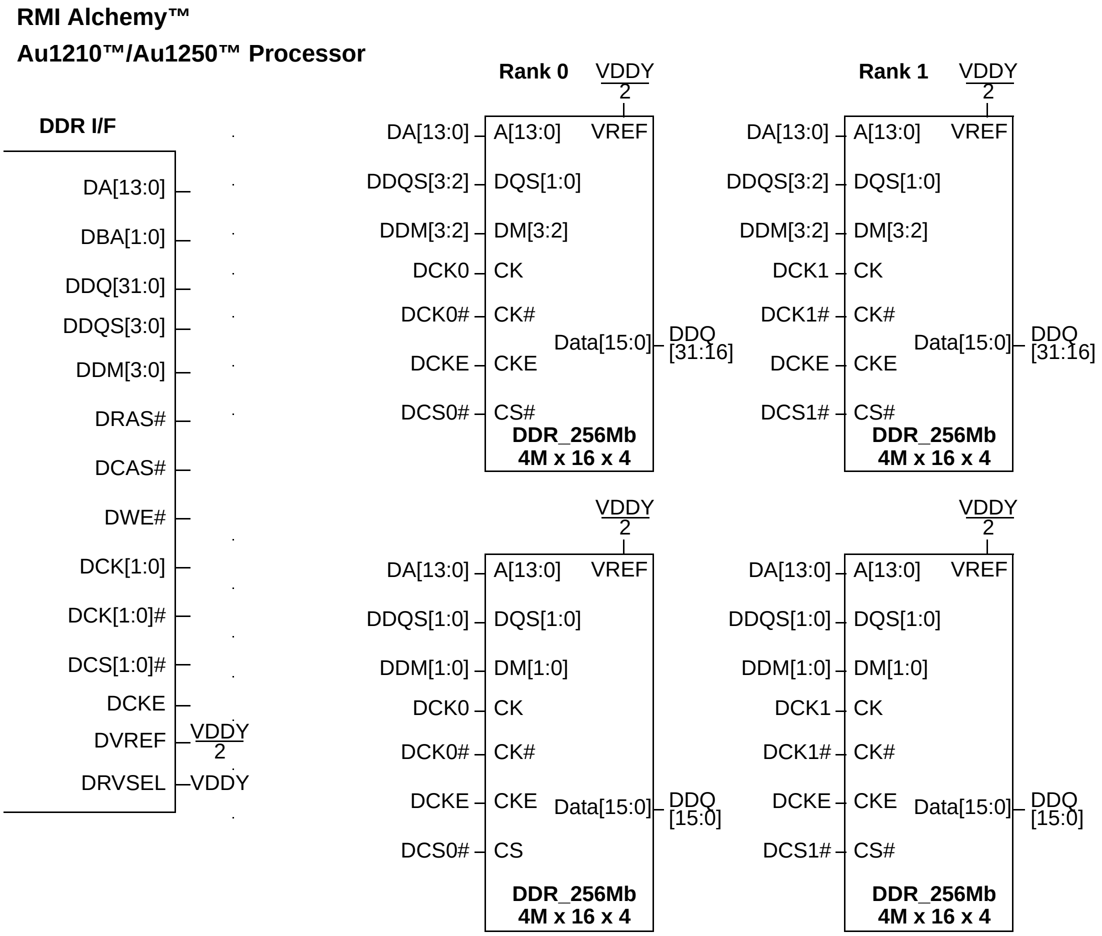


Figure 3-7. DDR Interface with Two Ranks

www.DataSheet4U.com

www.DataSheet4U.com

3.1.7 SDRAM Controller Commands

Table 3-6 provides a quick reference of available commands. More detailed descriptions can be found in the specific SDRAM device's data sheet.

Table 3-6. SDRAM Command Summary

NAME (FUNCTION)	DCS#	DRAS#	DCAS#	DWE#	DA[13:0] (Address)
DESELECT (NOP)	H	x	x	x	x
NO OPERATION (NOP)	L	H	H	H	x
ACTIVE (Select Bank and active row)	L	L	H	H	Bank/Row
READ (Select bank and column, and start read burst)	L	H	L	H	Bank/Col
WRITE (Select bank and column, and start read burst)	L	H	L	L	Bank/Col
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	Code
AUTO Refresh or Self Refresh (Enter self refresh mode)	L	L	L	H	x
LOAD MODE REGISTER	L	L	L	L	Op-Code
Power-Down Entry (DCKE high to low)	L	H	H	H	x
Power-Down Exit (DCKE low to high)	L	H	H	H	x

www.DataSheet4U.com

www.DataSheet4U.com

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 the following device types:

- ROM, SRAM, I/O peripherals with optional synchronous operation (see Section 3.2.3 on page 91)
- PCMCIA/Compact Flash devices (see Section 3.2.4 on page 94)
- NOR Flash with optional synchronous operation (see Section 3.2.5 on page 101)
- NAND Flash (see Section 3.2.6 on page 103)
- IDE PIO Mode (see Section 3.2.7 on page 109)

Because of the similarity of Compact Flash and PCMCIA, references to PCMCIA also apply to Compact Flash except where noted.

The static bus controller supports the different device types by configuring the chip select based on the device-type field in the chip select's configuration register (**mem_stcfg*n***[DTY]). However, the static bus controller does not support *multiple* chip selects programmed as a PCMCIA device (DTY = 2).

All device types use the same address and data bus signals, RAD[14:0] and RD[15:0]. The static bus has only fifteen address pins; however, an address latch mechanism is available to extend the effective range to thirty address signals (RAD[29:0]). (See Section 3.2.2 "Address Latch Mechanism" on page 90.)

3.2.1 Static Bus Controller Registers

The properties of each static bus controller chip select are determined by configuration registers. Table 3-7 shows the registers and offsets for the controller.

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

Table 3-7. Static Bus Controller Registers

Offset from 0x0 1400 0000 (Physical) 0xB400 0000 (KSEG1)	Register Name	Description
0x1000	mem_stcfg0	Configuration for RCS0#
0x1004	mem_sttime0	Timing parameters for RCS0#
0x1008	mem_staddr0	Address region control for RCS0#
0x1010	mem_stcfg1	Configuration for RCS1#
0x1014	mem_sttime1	Timing parameters for RCS1#
0x1018	mem_staddr1	Address region control for RCS1#
0x1020	mem_stcfg2	Configuration for RCS2#
0x1024	mem_sttime2	Timing parameters for RCS2#
0x1028	mem_staddr2	Address region control for RCS2#
0x1030	mem_stcfg3	Configuration for RCS3#
0x1034	mem_sttime3	Timing parameters for RCS3#
0x1038	mem_staddr3	Address region control for RCS3#
0x1040	mem_staltime	Static bus address latch timing register
0x1100	mem_stndctrl	Static bus NAND control register
0x1104	mem_ststat	Static bus status register

www.DataSheet4U.com

www.DataSheet4U.com

3.2.1.1 Static Bus Configuration Registers

These registers (**mem_stcfg*n***) configure the basic properties of each chip select. Support is included for static RAM, NOR and NAND Flash, ROM, PCMCIA, IDE, and other types of I/O devices.

When programming a chip select as an I/O or PCMCIA device, the address comparison mask expects an address with *sysbus_addr*[35:32] set as shown in Table 3-9 on page 81 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.

If mapped to the first 512 Mbytes, addresses for RAM and Flash device types do not need translation in the TLB because they map directly to KSEG0/1 (*sysbus_addr*[35:32] = 0x0).

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
	Tcsoe			Toecs			AH			NW	AS	S	DE	BEB		TA	DIV			ALD		AV	BE	TS	EW	MBSb	BS	PM	RO	DTY		
Def.	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs	Rs

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
	Tcsoe			Toecs			AH			NW	AS	S	DE	BEB									BE	TS	EW	MBSb	BS	PM	RO	DTY		
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	mem_stcfg0 Default
31:29	Tcsoe	Tcsoe represents the number of clocks needed to setup the address before asserting output enable. Valid only when **mem_stcfg**[AS] = 1. Tcsoe does not apply to NAND and PCMCIA device types. (Tcsoe + 1) is the actual number of clock cycles.	R/W	See Table 3-8 on page 81.
28:26	Toecs	Toecs represents the number of clocks needed to hold the address after negating output enable. Valid only when **mem_stcfg**[AH] = 1. Toecs does not apply to NAND and PCMCIA device types. (Toecs + 1) is the actual number of clock cycles.	R/W	
25	AH	Hold address after output enable on reads. The setup duration is programmed in **mem_stcfg**[Toecs] and is shown as Toe_cs in Figure 3-11 on page 92. 0 Do not hold address. 1 Hold address.	R/W	
24	—	Reserved.	R/W	
22	NW	NAND width. Selects the device width for NAND Flash. Applies to NAND Flash devices only (DTY = 5). 0 16-bit mode 1 8-bit mode	R/W	
21	AS	Setup address before output enable on reads. The setup duration is programmed in **mem_stcfg**[Tcsoe] and is shown as Tcs_oe in Figure 3-11 on page 92. 0 Do not setup address. 1 Setup address.	R/W	

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	mem_stcfg0 Default
20	S	Synchronous mode for this chip select. This bit does not apply to NAND operation. 0 Asynchronous mode. All static bus signals are synchronized to the internal system bus clock. Values in the **mem_sttime*n*** register are programmed in system-bus-clock resolution. 1 Synchronous mode. All static bus signals are synchronized to RCLK. Values in the **mem_sttime*n*** register are programmed in RCLK clock resolution. Note: Synchronous mode must not be selected when configured for PCMCIA. Note: If S is set in NAND mode, the NAND signal timings are still based off the system bus, and all timing parameters are still programmed in system-bus-clock resolution.	R/W	See Table 3-8 on page 81.
19	DE	Deassert chip select, output enable, write enable, and byte enables between each beat during bursts. The deassert time is programmable via **mem_sttime*n***[Tcsoff]. Note: A one-word (32-bit) transfer to a 16-bit bus is treated as a burst; see Figure 3-15 on page 94.	R/W	
18:17	BEB	Byte Enable Behavior. 00 Reserved 01 Two active beats are always generated, even for 8- and 16-bit transfers. Not recommended. 10 The static bus controller negates the chip select, the output enable and the write enable signals when RBE[1:0]# are not asserted. Recommended operation. 11 Reserved	R/W	
16	TA	Tcsh Application. 0 Apply Tcsh after reads only. 1 Apply Tcsh after both writes and reads. This bit is a global attribute and is present only in **mem_stcfg0**.	R/W	
15:13	DIV	Adjusts the divisor for the RCLK output clock. The clock frequency is set by RCLK = (System Bus Clock / 2) / (DIV + 1) Note: RCLK must not exceed 33.33MHz. This bit is a global attribute and is present only in **mem_stcfg0**.	R/W	
12	ALD	Address Latch Disable. 0 Enable address latching. 1 Disable address latching. The address latch timing parameters are programmed in **mem_staltime**. This bit is a global attribute and is present only in **mem_stcfg0**.	R/W	
11	—	Reserved.	—	

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	mem_stcfg0 Default
10	AV	Address Visible. 0 Normal operation. 1 Place the address for all internal system bus transfers on the static address bus. The AV bit is provided for debug. Using during normal operation may increase power. This bit is a global attribute and is present only in **mem_stcfg0**.	R/W	See Table 3-8 on page 81.
9	BE	Endianness. Program this bit to match the endianness of the processor core. 0 Little endian 1 Big endian. Not valid for PCMCIA.	R/W	
8	TS	Time scale for chip select timing parameters. 0 Do not scale the timing parameters. 1 Multiply the timing parameters by a factor of four. This option allows for longer access times. Table 3-10 on page 81 and Table 3-11 on page 82 show how TS affects the actual number clocks used.	R/W	
7	EW	When the EW bit is set the EWAIT# input is allowed to stretch the bus access time. The EW bit does not apply to chip selects operating in PCMCIA mode because it has a different wait mechanism (PWAIT#).	R/W	
6	MBSb	Reserved. Must be set.	R/W	
5	BS	Burst Size for Page Mode Transfers. Selects the number of beats per burst for page mode transfers. Valid only in page mode (PM = 1). 0 4 beats 1 8 beats The controller does not check for page boundaries; see Section 3.2.8.2 on page 111.	R/W	
4	PM	If the PM bit is set the chip select operates in page mode. This allows quick access to sequential locations in memory. See Section 3.2.8.2 on page 111. Page mode applies only to reads.	R/W	
3	RO	If the RO bit is set the chip select is read only. This inhibits the generation of write cycles to the chip select. An attempt to write to the address region controlled by a read only chip select is ignored.	R/W	
2:0	DTY	Device Type. Selects the type of device controlled by the static bus controller chip select. A list of device types is shown in Table 3-9 on page 81. Programming multiple chip selects as PCMCIA is not supported.	R/W	

www.DataSheet4U.com

www.DataSheet4U.com

Table 3-8. Default Register Values for mem_stcfg0

BOOT[1:0](Note1)	Default Value
0 (NOR)	0xFE2A60C3
1(8-bit NAND)	0xFCC26085
2 (16-bit NAND)	0xFC826085
3	Reserved

Note1. Determined by value of BOOT[1:0] when RESETIN# deasserts.

Table 3-9. Device Type Encoding

DTY	Chip Select Function	(*sysbus_addr*[35:32])	Reference
0	Static RAM	0x0	Section 3.2.3 on page 91
1	I/O Device	0xD	Section 3.2.3 on page 91
2	PCMCIA Device/Compact Flash	0xF	Section 3.2.4 on page 94
3	NOR Flash	0x0	Section 3.2.5 on page 101
4	Reserved	—	—
5	NAND Flash	0x0	Section 3.2.6 on page 103
6	IDE	0x0	Section 3.2.7 on page 109
7	Reserved	—	—

3.2.1.2 Static Timing Registers

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

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

- For asynchronous (**mem_stcfg*n***[S] = 0) chip selects, the clock is the system bus clock.
- For synchronous (**mem_stcfg*n***[S] = 1) chip selects, the clock is RCLK. See Section 3.2.8.1 on page 111. Synchronous mode must not be selected when configured for PCMCIA. Note that this bit does not apply to NAND operation. If S is set in NAND mode, the NAND signal timings are still based off the system bus, and all timing parameters are still programmed in system-bus-clock resolution.

The actual number of clocks for each timing parameter ($T_{parameter}$) is shown in Table 3-10. Note that the timing behavior for Tcsh is different and is shown in Table 3-11 on page 82.

Table 3-10. Actual Number of Clocks for Timing Parameters (Except Tcsh)

Device Type	TS = 0	TS = 1
Static RAM, I/O, NOR, NAND, and IDE	$T_{parameter}$ + 1	(4 * $T_{parameter}$) + 1
	$T_{parameter}$ + 2 (Note1)	(4 * $T_{parameter}$) + 2 (Note 1)
PCMCIA Device/Compact Flash	$T_{parameter}$ + 2	(4 * $T_{parameter}$) + 2

Note1. Applies to parameters Tpm and Ta only.

www.DataSheet4U.com

www.DataSheet4U.co

Table 3-11. Actual Number of Clocks for Tcsh

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

The default register values for the boot modes as seen in Table 3-12 also apply for SRAM, PCMCIA and NOR Flash. The default value for all remaining timing registers is 0xFFFFFFFF.

Table 3-12. Default Register Values for mem_sttime0

BOOT[1:0](Note1)	Default Value
0 (NOR)	0xFFFFFFDD
1(8-bit NAND)	0x00002843
2 (16-bit NAND)	0x00002843
3	Reserved

Note1. Determined by value of BOOT[1:0] when RESETIN# deasserts.

The **mem_sttime*n*** registers have different formats depending on the device type. The following shows the register format for I/O, IDE, and SRAM devices.

mem_sttime0 (I/O, SRAM)

Offset = 0x1004

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

mem_sttime1 (I/O, IDE, SRAM) **Offset = 0x1014**
mem_sttime2 (I/O, IDE, SRAM) **Offset = 0x1024**
mem_sttime3 (I/O, IDE, SRAM) **Offset = 0x1034**

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

www.DataSheet4U.com

www.DataSheet4U.co

Bits	Name	Description	R/W	mem_sttime0 Default
31	—	Reserved.	—	—
30:28	Twcs	Specifies the required chip select hold time after a write pulse. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
27:24	Tcsh	Chip Select Hold-off. Specifies the minimum number of cycles the chip select remains deasserted between transfers. The next transaction through the static bus controller is held off until the Tcsh parameter is satisfied. The system bus can arbitrarily extend the time between transfers for internal operations. This can add up to about five additional clocks to the programmed time. See Table 3-11 on page 82 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
23	—	Reserved.	—	—
22:20	Tcsoff	Specifies the required number of cycles that the chip select, output enable, write enable, and byte enables must remain deasserted between beats. Valid only when **mem_stcfg*n***[DE] is set. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
19:14	Twp	Specifies the duration of the write enable. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
13:10	Tcsw	Chip Select To Write. Applies to writes only. Defines the delay from the assertion of chip select until the controller drives data and asserts the write strobe and byte enables. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
9:6	Tpm	Specifies the number of cycles required from a burst address change until read data is valid. Applies to page mode (**mem_stcfg*n***[PM] = 1) only. Ta determines the access time for the *first* beat of each burst. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
5:0	Ta	Specifies the number of cycles required for the assertion of the chip select during a single-beat read. For page mode transfers, Ta determines the access time up to the first beat of each burst. See Section 3.2.8.2 on page 111. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.

The following shows the **mem_sttime*n*** format for PCMCIA devices.

mem_sttime0 (PCMCIA)

Offset = 0x1004

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

mem_sttime1 (PCMCIA) **Offset = 0x1014**
mem_sttime2 (PCMCIA) **Offset = 0x1024**
mem_sttime3 (PCMCIA) **Offset = 0x1034**

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

www.DataSheet4U.com

www.DataSheet4U.co

Bits	Name	Description	R/W	Default
31:24	Tmst	Specifies the strobe width during memory accesses to PCMCIA chip selects. The timing duration depends on the time-scale option **mem_stcfg*n***[TS]: When TS = 0, (Tmst + 2) is the number of cycles to the end of the strobe; however, the read occurs at (Tmst + 1). When TS = 1, [(4 * Tmst) + 2] is the number of cycles to the end of the strobe; however, the read occurs at [(4 * Tmst) + 1].	R/W	See Table 3-12 on page 82.
23:17	Tmsu	Specifies the setup time from chip select to strobe during memory accesses to PCMCIA chip selects. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	
16:11	Tmih	Specifies the hold time for address, data, and chip selects from the end of the strobe for both memory and I/O cycles to PCMCIA chip selects. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	
10:5	Tist	Specifies the strobe width for I/O transfers for a chip select configured for PCMCIA. The timing duration depends on the time-scale option **mem_stcfg*n***[TS]: When TS = 0, (Tist + 2) is the number of cycles to the end of the strobe; however, the read occurs at (Tist + 1). When TS = 1, [(4 * Tist) + 2] is the number of cycles to the end of the strobe; however, the read occurs at [(4 * Tist) + 1].	R/W	
4:0	Tisu	Specifies the setup time from chip select to strobe during I/O transfers for PCMCIA. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	

The following shows the register format for NOR Flash.

mem_sttime0 (NOR Flash) **Offset = 0x1004**

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

mem_sttime1 (NOR Flash) **Offset = 0x1014**
mem_sttime2 (NOR Flash) **Offset = 0x1024**
mem_sttime3 (NOR Flash) **Offset = 0x1034**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	mem_sttime0 Default
31	—	Reserved.	—	—
30:28	Twcs	Specifies the required chip select hold time after a write pulse. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
27:24	Tcsh	Chip Select Hold-off. Specifies the minimum number of cycles the chip select remains deasserted between transfers. The next transaction through the static bus controller is held off until the Tcsh parameter is satisfied. The system bus can arbitrarily extend the time between transfers for internal operations. This can add up to about five additional clocks to the programmed time. See Table 3-11 on page 82 for the actual number of clock cycles.	R/W	
23	—	Reserved.	—	—
22:20	Tcsoff	Specifies the required number of cycles that the chip select, output enable, write enable, and byte enables must remain deasserted between beats. Valid only when **mem_stcfg*n***[DE] is set. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	See Table 3-12 on page 82.
19:14	Twp	Specifies the duration of the write enable. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	
13:10	Tcsw	Chip Select To Write. Applies to writes only. Defines the delay from the assertion of chip select until the controller drives data and asserts the write strobe and byte enables. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	
9:6	Tpm	Specifies the number of cycles required from a burst address change until read data is valid. Applies to page mode (**mem_stcfg*n***[PM] = 1) only. Ta determines the access time for the *first* beat of each burst. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	
5:0	Ta	Specifies the number of cycles required for the assertion of the chip select during a single-beat read. For page mode transfers, Ta determines the access time up to the first beat of each burst. See Section 3.2.8.2 on page 111. See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	

www.DataSheet4U.com

www.DataSheet4U.c

The following shows the **mem_sttime*n*** format for NAND Flash devices.

mem_sttime0 (NAND Flash)	**Offset = 0x1004**
mem_sttime1 (NAND Flash)	**Offset = 0x1014**
mem_sttime2 (NAND Flash)	**Offset = 0x1024**
mem_sttime3 (NAND Flash)	**Offset = 0x1034**

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

Bits	Name	Description	R/W	NAND Boot Default (BOOT[1:0] = 1 or 2)
31:12	—	Reserved.	—	—
11:8	Tsu	Global setup timing for Tcls (command latch), Tals (address latch), Tcs (chip select), Tds (data) See Table 3-10 on page 81 for the actual number of clock cycles. **Note:** The system bus can arbitrarily extend the time between transfers for internal operations. This can add up to five additional clocks to the programmed time.	R/W	0x8
7:4	Tpul	Global timing for write-enable pulse width (Twp) and read-enable pulse width (Trp) See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	0x4
3:0	Th	Global hold timing for Talh (address latch), Tclh (command latch), Tch (chip select), Tdh (data) See Table 3-10 on page 81 for the actual number of clock cycles.	R/W	0x3

3.2.1.3 Static Chip Select Address Configuration Registers (mem_staddr*n*)

These registers assign an address range for each chip select. As shown below, each register contains a base address, an address comparison mask, and an enable bit.

mem_staddr0	**Offset = 0x1008**

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
				E	CSBA														CSMASK													
Def.	0	0	0	Rs	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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
				E	CSBA														CSMASK													
Def.	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

www.DataSheet4U.com

www.DataSheet4U.c

Bits	Name	Description	R/W	Default
31:29	—	Reserved.	R	0
28	E	Enable. 0 Disable the chip select. 1 Enable the chip select.	R/W	0, except for **mem_staddr0**(Note1)
27:14	CSBA	Chip select base address *cs_base*[31:18]. *cs_base* is the physical base address for the chip select. *cs_base*[35:32] is determined by **mem_stcfg*n***[DTY]. *cs_base*[31:18] = CSBA. *cs_base*[17:0] = 0.	R/W	0x3FFF, except for **mem_staddr0** where the default value is 0x7F0.
13:0	CSMASK	Chip select address mask *cs_mask*[31:18]. *cs_mask* is used to decode the chip select. *cs_mask*[35:32] = 0xF. *cs_mask*[31:18] = CSMASK. *cs_mask*[17:0] = 0.	R/W	0x3FFF

Note1.The enable (E) bits for chip selects RCS1#, RCS2#, and RCS3# are automatically cleared (disabled) coming out of a runtime or hardware reset. For RCS0#, however, the reset value of the E bit depends on BOOT[1:0]. If the BOOT[1:0] signals indicate that a memory device on the static bus is used for the boot vector, RCS0# is enabled (**mem_staddr0**[E] = 1); otherwise, RCS0# is disabled. See also Section 11.3 "Boot" on page 401.

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

(*sysbus_addr* & *cs_mask*) == *cs_base*

where

sysbus_addr: 36-bit physical address output on the internal system bus (from the TLB for memory-mapped regions)

cs_mask: address comparison mask taken from CSMASK

cs_base: chip select base address taken from CSBA

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

3.2.1.4 Address Latch Timing Register

This register contains the timing parameters for latching the upper portion of the address. The address latch mechanism extends the number of address signals in a system.

mem_staltime **Offset = 0x1040**

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
																								Tah			Tlw			Tasu		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0

www.DataSheet4U.com

w w w . D a t a S h e

Bits	Name	Description	R/W	Default
31:9	—	Reserved.	—	—
8:6	Tah	Address hold time after address latch enable ALE negates. Tah is the actual number of clock cycles. Tah should be zero for latches that sample on the rising edge of ALE.	R/W	0x4
5:3	Tlw	Address latch enable ALE pulse width. (Tlw + 1) is the actual number of clock cycles.	R/W	0x3
2:0	Tasu	Address setup time before address latch enable ALE asserts. Tasu is the actual number of clock cycles. Tasu should be zero for latches that sample on the falling edge of ALE.	R/W	0x4

3.2.1.5 NAND Control Register

This register enables booting and interrupts from NAND Flash. This register is write only.

mem_stndctrl **Offset = 0x1100**

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
																								IE	CS3O	CS2O	CS1O	CS0O				BOOT
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Name	Description	R/W	Default
31:9	—	Reserved.	—	—
8	IE	NAND-busy interrupt enable. 0 Disable NAND-busy interrupts. 1 Enable NAND-busy interrupts.	W	0
7	CS3O	RCS3# Override. Setting this bit asserts the chip select.	W	0
6	CS2O	RCS2# Override. Setting this bit asserts the chip select.	W	0
5	CS1O	RCS1# Override. Setting this bit asserts the chip select.	W	0
4	CS0O	RCS0# Override. Setting this bit asserts the chip select.	W	0
3:1	—	Reserved.	—	—
0	BOOT	NAND-boot Mode Enable. 0 Disable boot mode 1 Enable boot mode This bit must be cleared once the bootloader stored in NAND Flash is copied into cache or external memory.	W	1

www.DataSheet4U.com

www.DataSh

3.2.1.6 Static Bus Status Register (mem_ststat)

This register reflects the state of the flow-control signals associated with the static bus: PWAIT#, EWAIT#, and RNB. Software drivers can poll **mem_ststat** for device status information. This register is not rank specific because multiple devices can be tied to a single signal. If independent status is required, individual GPIO inputs can be used. The state of the system boot configuration signals BOOT[1:0] are also reflected in **mem_ststat**.

A boot value of 3 indicates booting from Large Block NAND. NAND size can be determined by checking **mem_stcfg0**.

mem_ststat **Offset = 0x1104**

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
																											PWT	EWT		BOOT		BSY
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:6	—	Reserved.	—	—
5	PWT	PWAIT# Status. 0 PWAIT# is not asserted. 1 PWAIT# is asserted.	R	0
4	EWT	EWAIT# Status. 0 EWAIT# is not asserted. 1 EWAIT# is asserted.	R	0
3	—	Reserved.	—	—
2:1	BOOT[1:0]	System Boot Configuration State. Reflects the state of the BOOT[1:0] signals.	R	System dependent
0	BSY	NAND Device Busy Status. The interrupt associated with the RNB signal (interrupt number 23 on interrupt controller 0) occurs when the NAND device is ready. The interrupt is enabled by setting **mem_stndctrl**[IE]. 0 RNB is low. NAND device is busy. 1 RNB is high. NAND device is ready.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

3.2.2 Address Latch Mechanism

For all static bus asynchronous and synchronous accesses that require more than fifteen address signals (ADDR[14:0]), the static bus controller provides an address latch mechanism to support up to thirty address signals (ADDR[29:0]). See Figure 3-8.

If the **mem_stcfg0**[ALD] bit is cleared, the static bus controller provides two address writes prior to a read or write from any device. The first address write contains ADDR[29:15], along with the ALE signal assertion. The second address write contains ADDR[14:0].

The **mem_staltime** register contains the programmable timing parameters for the address latch portion of the bus cycle.

The address latch option is required for systems that contain NOR Flash, RAM, PCMCIA, and memory-mapped peripherals with more than fifteen address signals. IDE and NAND Flash do not require this option.

Figure 3-9 shows an asynchronous read access. The address latch option is available for all synchronous and asynchronous read accesses except for NAND Flash. This includes non-page mode and page mode options for NOR Flash, RAM, and PCMCIA configured chip selects.

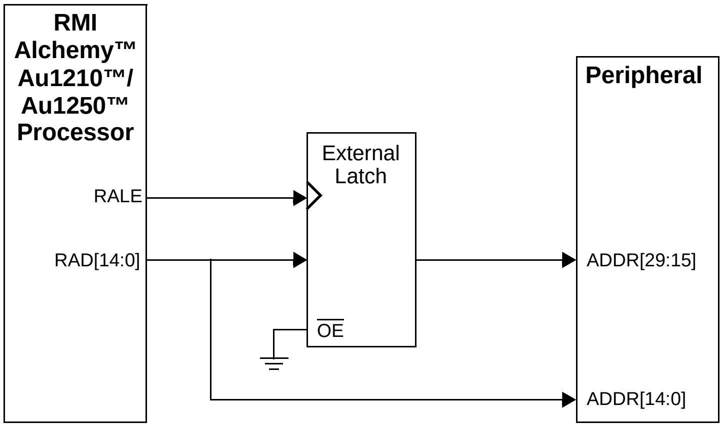
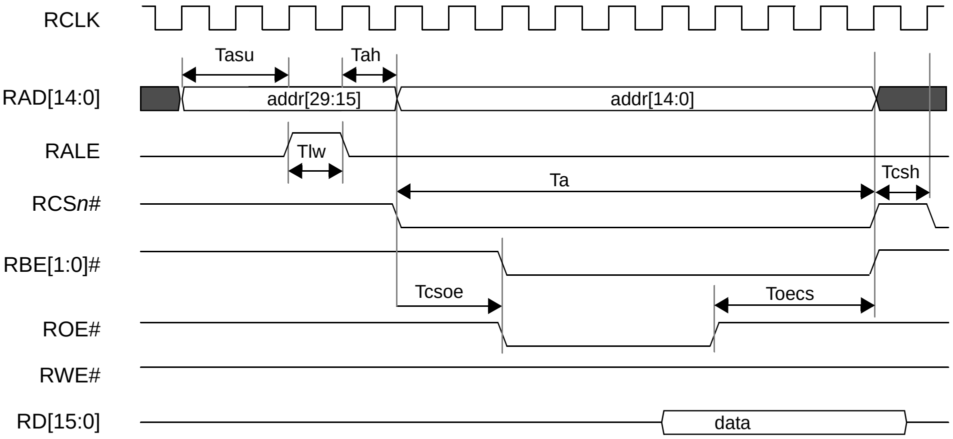


Figure 3-8. Address Latch Mechanism (mem_stcfg0[ALD] = 0)



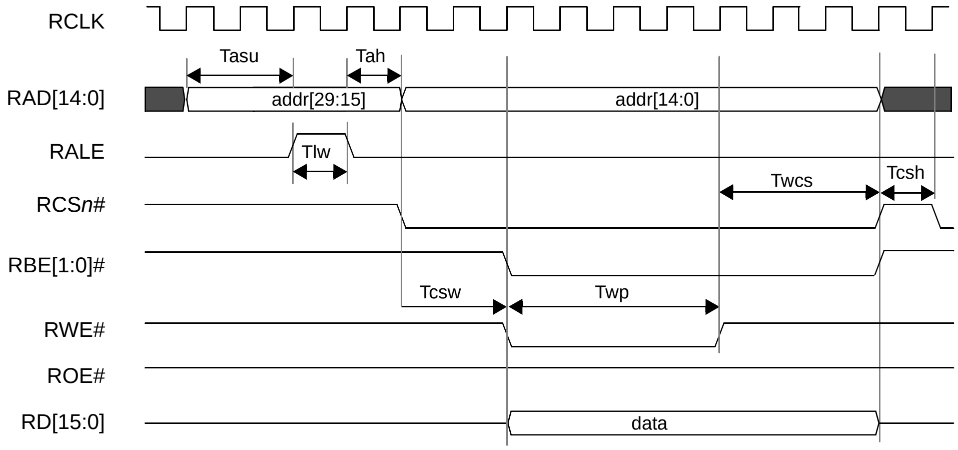
NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-9. Address Latch Timing on a Read (mem_stcfg0[ALD] = 0)

www.DataSheet4U.com

www.DataSheet4U.com

Figure 3-10 shows an asynchronous write access. The address latch option is available for all synchronous and asynchronous write accesses except for NAND Flash. This includes NOR Flash, RAM, and PCMCIA configured chip selects.



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-10. Address Latch Timing on a Write (mem_stcfg0[ALD] = 0)

3.2.3 Static RAM and I/O Device Types

This section describes the static RAM interface that is implemented when the device type (**mem_stcfg*n***[DTY]) is programmed to 0 or 1. (See “Static Bus Configuration Registers” on page 78.)

The I/O device type is identical to the static RAM type except it expects *sysbus_addr*[35:32] to be 0xD. The static RAM and I/O device types share the same timing and control signals. The SRAM and I/O signals are shown in Table 3-13.

3.2.3.1 Static Memory Timing Diagrams (I/O, SRAM)

This section presents static bus timing diagrams for reads and writes for I/O and SRAM devices. The diagrams show how the programmable timing parameters (in **mem_sttime*n***) can be used to adjust the interface timing to meet the requirements of a specific device.

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 14.0 "Electrical and Thermal Specifications" on page 439. (See also Section 14.4.2, "Asynchronous Static Bus Controller Timing" and Section 14.4.3 "Synchronous Static Bus Timing" on page 457.)

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

Table 3-13. Static RAM and I/O Device Signals

Pin Name	Input/Output	Description
RAD[14:0]	O	Address Bus.
RD[15:0]	I/O	Data Bus.
RBE[1:0]#	O	Byte Enables: RBE0# corresponds to RD[7:0]. RBE1# is for RD[15:8].
RWE#	O	Write Enable.
ROE#	O	Output Enable.
RCS[3:0]#	O	Programmable Chip Selects (4 banks). RCS*n*# is not used when configured as a PCMCIA device.
EWAIT#	I	Can be used to stretch the bus access time when enabled through **mem_stcfg*n***[EW].

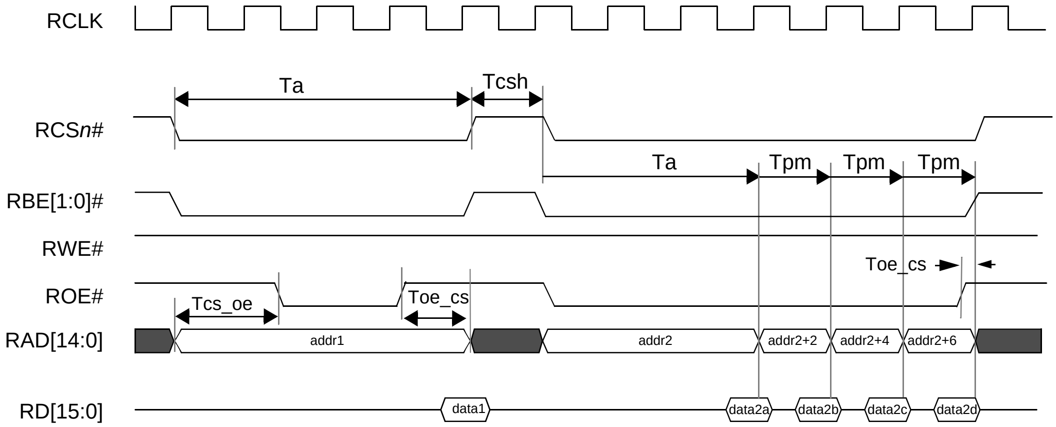
www.DataSheet4U.com

www.DataSheet4U.com

3.2.3.2 Read Timing Diagrams (I/O, SRAM)

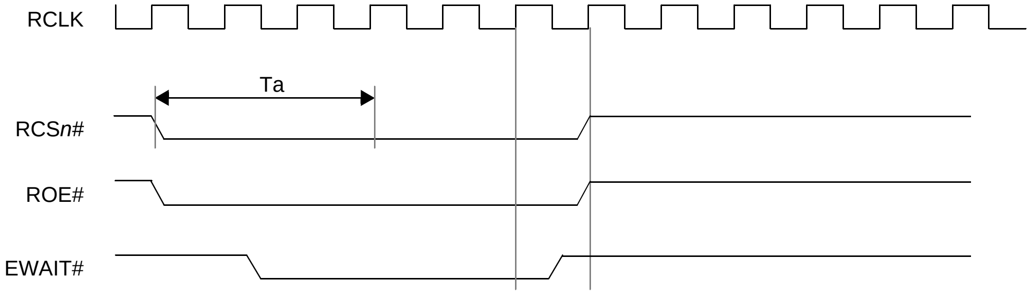
Figure 3-11 shows static bus read timing with a single 16-bit read followed by a 2-word page mode burst read.

Figure 3-12 shows how EWAIT# holds the cycle past Ta.



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-11. I/O and SRAM 16-Bit Read Timing (Single Read, Burst) (mem_stcfg0[ALD] = 1)



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

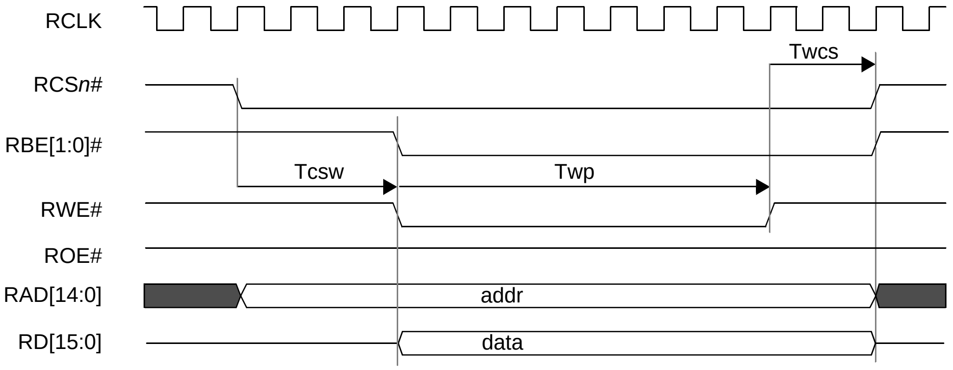
Figure 3-12. I/O and SRAM Read EWAIT# Timing

www.DataSheet4U.com

www.DataSheet4U.com

3.2.3.3 Write Timing Diagrams (I/O, SRAM)

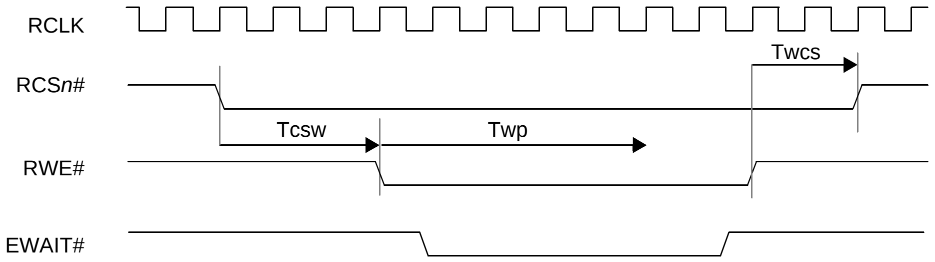
The timing diagrams below show the static bus write timing for I/O and SRAM device types. Figure 3-13 shows a single 16-bit write.



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-13. I/O and SRAM 16-Bit Write Timing (mem_stcfg0[ALD] = 1)

Figure 3-14 shows how EWAIT# holds the cycle past Twp.



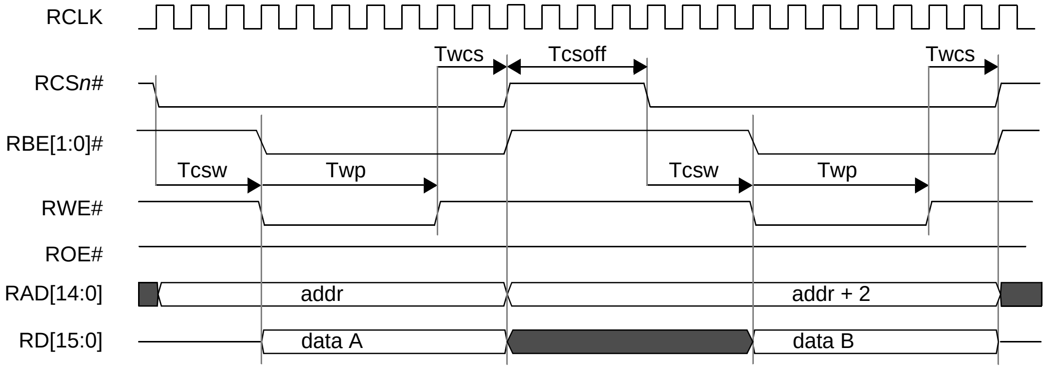
NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-14. I/O and SRAM Write EWAIT# Timing

www.DataSheet4U.com

www.DataSheet4U.com

Figure 3-15 shows a 32-bit write. The transaction is treated as a burst. The diagram shows the timing with **mem_stcfg**[DE] = 1 (deassert mode); see Section 3.2.1.1 "Static Bus Configuration Registers" on page 78. For DE = 0, the Tcsoff parameter collapses to zero and the control signals remain asserted.



NOTE: The external clock RCLK is available only in synchronous mode (mem_stcfg[S] = 1). For asynchronous chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-15. I/O and SRAM 32-Bit Write Timing (Burst) (mem_stcfg0[ALD] = 1)

3.2.4 PCMCIA/Compact Flash Device Type

Because of the similarity of Compact Flash and PCMCIA, references to PCMCIA should be taken as applicable to Compact Flash except where noted. The PCMCIA peripheral is designed to the PCMCIA 2.1 specification, but only for the bus transactions as described in this section.

The Au1210 and Au1250 processors provide a PCMCIA host adapter when the device type is programmed for PCMCIA. The static bus controller interface provides the required bus signals necessary to control a PCMCIA interface. Auxiliary signals, such as card detect and voltage sense, can be implemented with GPIOs if desired. Synchronous mode (**mem_stcfg**[S] = 1) is not available for PCMCIA.

The PCMCIA host interface adapter supports memory, attribute, and I/O transactions. External logic can be added to support DMA transfers.

The Au1210 and Au1250 processors support only 8- and 16-bit load and store instructions (byte and halfword instructions) to PCMCIA devices. 32-bit transfers are not supported. Note the following about PCMCIA transfers:

- For writes, the static bus controller uses the internal system bus byte masks to extract the data from *sysbus_data*.
- For reads, the static bus controller replicates the read-data into *sysbus_data*; that is, for an 8-bit access, the read-data is replicated into all four bytes of *sysbus_data*, and for a 16-bit access, the read-data is replicated into both halfwords of *sysbus_data*. This behavior enables Au1 core loads and stores to operate correctly in either endian mode for PCMCIA.

The PCMCIA interface occupies a 36-bit address space with *sysbus_addr*[35:32] = 0xF. The TLB is required to generate addresses that activate a PCMCIA chip select. I/O, memory and attribute spaces are differentiated by *sysbus_addr*[31:30]. Table 3-14 shows the mapping.

Table 3-14. PCMCIA Memory Mapping

Physical Address (Note1)	PCMCIA Mapping
0xF 0xxx xxxx	I/O
0xF 4xxx xxxx	Attribute
0xF 8xxx xxxx	Memory

Note1. Each of the PCMCIA physical address spaces has a maximum size of 64MB. An access beyond the 64MB space aliases back into the defined region.

www.DataSheet4U.com

www.DataSheet4U.com

The PCMCIA interface provides control signals defined for PCMCIA slots. If two cards are required, external logic must be added to allow for both cards to share the bus. When a chip select is programmed as a PCMCIA device, the associated RCS*n*# is not used. Table 3-15 shows the signals supporting the PCMCIA interface.

Table 3-15. PCMCIA Interface Signals

Pin Name	Input/ Output	Description
RAD[14:0]	O	Address Bus. The address latch mechanism (**mem_stcfg0**[ALD] = 0) is required for PCMCIA. See Section 3.2.2 "Address Latch Mechanism" on page 90. The PCMCIA bus timing structure does not change except for the additional addressing to create full address decode on the PCMCIA device.
RD[15:0]	I/O	Data Bus.
PREG#	O	When this signal is asserted card access is limited to attribute memory when a memory access occurs and to I/O ports when an I/O access occurs.
PCE[2:1]#	O	Card Enables.
POE#	O	Memory Output Enable.
PWE#	O	Memory Write Enable.
PIOR#	O	I/O Read Cycle Indication.
PIOW#	O	I/O Write Cycle Indication.
PWAIT#	I	This signal is asserted by the card to delay completion of a pending cycle. This signal must be tied high through a resistor when the PCMCIA interface is not used.
PIOS16#	I	16-bit Port Select. This signal must be tied high through a resistor when the PCMCIA interface is not used.
ROE#	O	Output Enable. This output enable is intended to be used as a data transceiver control. During a PCMCIA transaction, ROE# remains asserted (low) as configured in the timing registers (**mem_sttime*n***) for reads and negated (high) for writes.

www.DataSheet4U.com

www.DataSheet4U.com

Figure 3-16 and Figure 3-17 on page 97 show a one and two card PCMCIA implementation. For the two card implementation, RAD26 is used as a card select signal. Both figures assume that the PCMCIA card can be hot swapped at any time. Note the use of isolation buffers on the shared bus. If the card is fixed in the system, much of the interface logic can be removed. A Compact Flash implementation is similar to the PCMCIA implementation except that the number of address lines used is fewer.

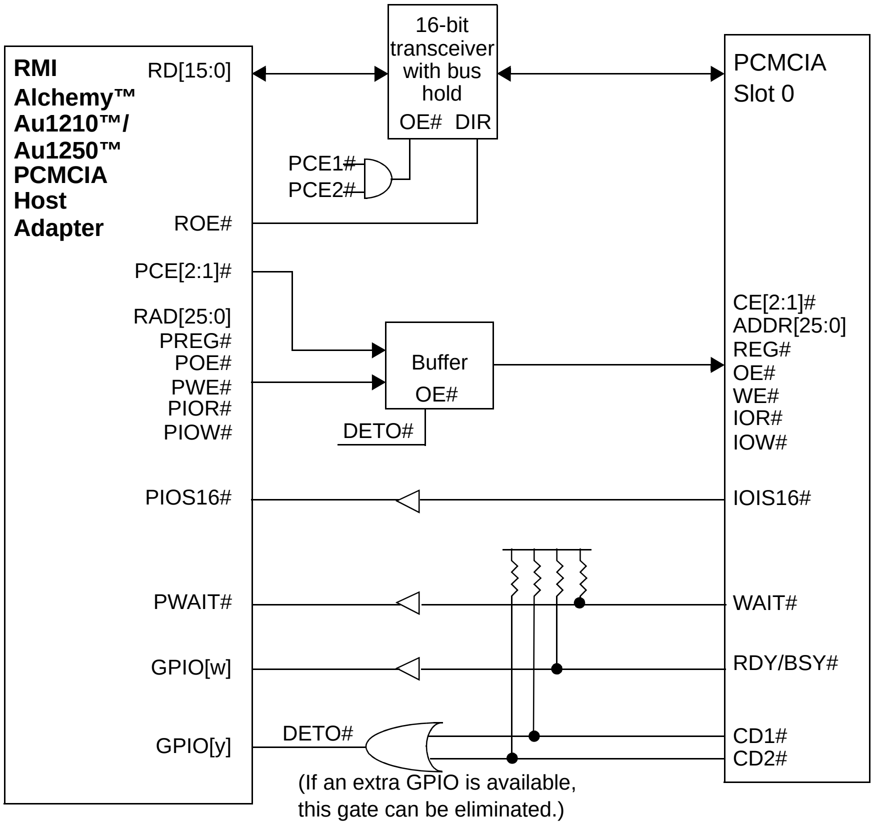


Figure 3-16. One Card PCMCIA Interface

www.DataSheet4U.com

www.DataSheet4U.com

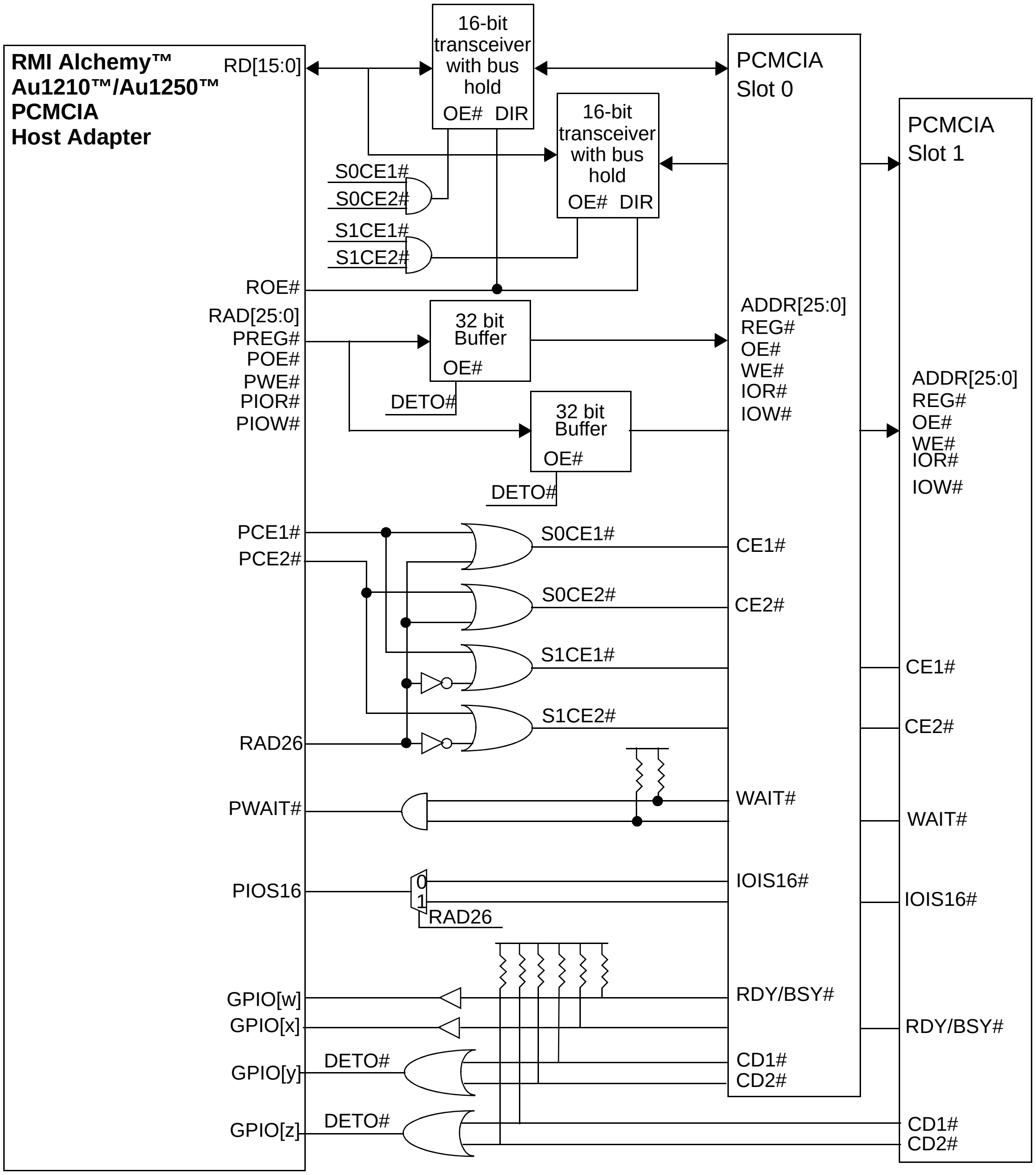


Figure 3-17. Two Card PCMCIA Interface

www.DataSheet4U.com

www.DataSheet4U.com

3.2.4.1 Static Memory Timing Diagrams (PCMCIA/Compact Flash)

This section presents static bus timing diagrams for the PCMCIA interface, including memory read timing, memory write timing, I/O read timing, and I/O write timing. The diagrams show how the programmable timing parameters (in **mem_sttime*n***) can be used to adjust the interface timing to meet the requirements of a specific device. The PWAIT# timing diagrams are also presented to show how PWAIT# holds the cycle past Tmst for *memory* transfers and past Tist for *I/O* transfers.

The address latch mechanism (**mem_stcfg0**[ALD] = 0) is required to create an extended address bus for the PCMCIA interface. The timing diagrams below do not show the double-address and RALE assertions. See Section 3.2.2 "Address Latch Mechanism" on page 90 for the latch timing of the higher-order address bits (RAD[29:15]).

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 14.0 "Electrical and Thermal Specifications" on page 439. (See also Section 14.4.2 "Asynchronous Static Bus Controller Timing" on page 453.)

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

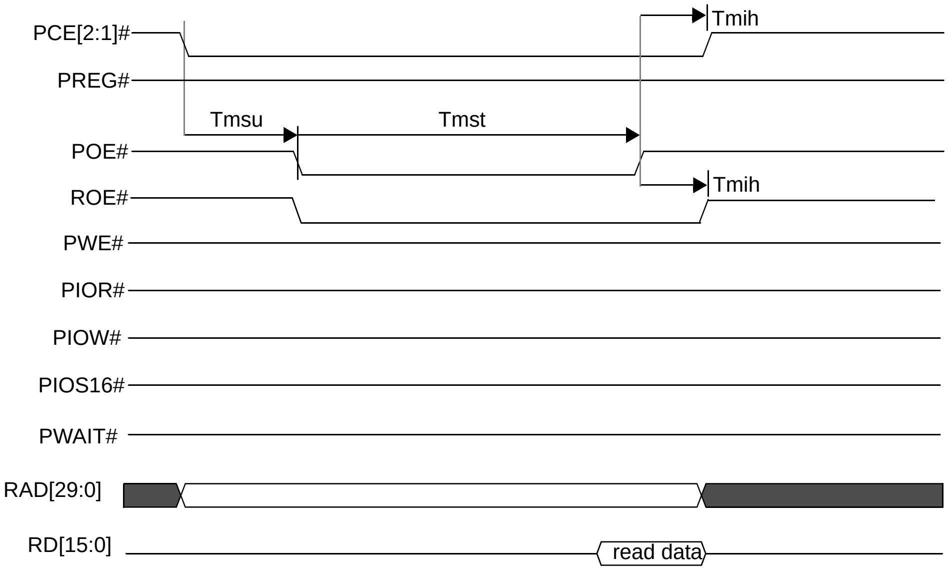


Figure 3-18. PCMCIA Memory Read Timing

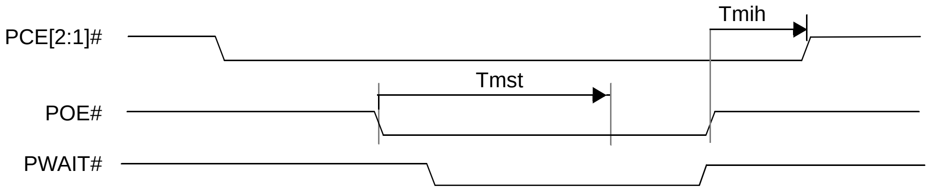


Figure 3-19. PCMCIA Memory Read PWAIT# Timing

www.DataSheet4U.com

www.DataSheet4U.com

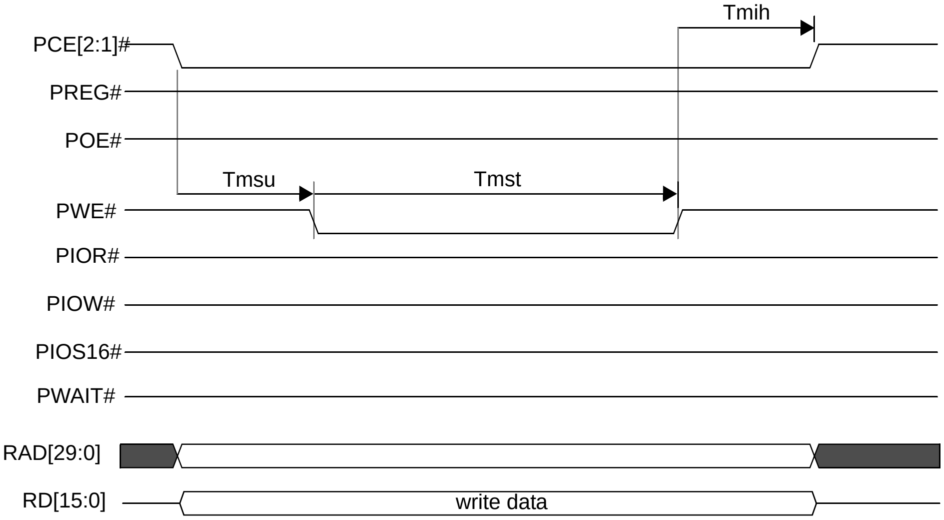


Figure 3-20. PCMCIA Memory Write Timing

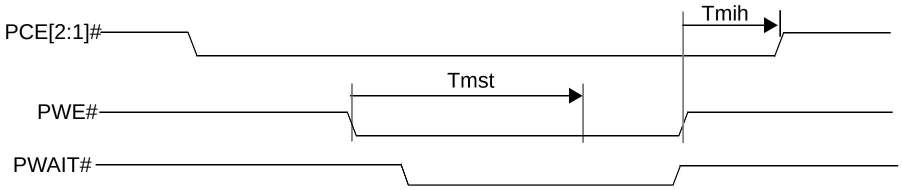


Figure 3-21. PCMCIA Memory Write PWAIT# Timing

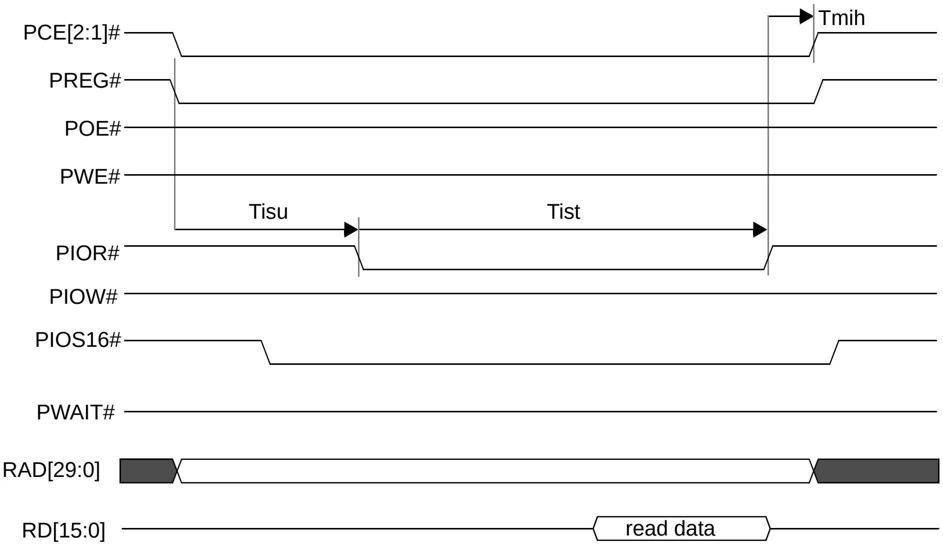


Figure 3-22. PCMCIA I/O Read Timing

www.DataSheet4U.com

www.DataSheet4U.com

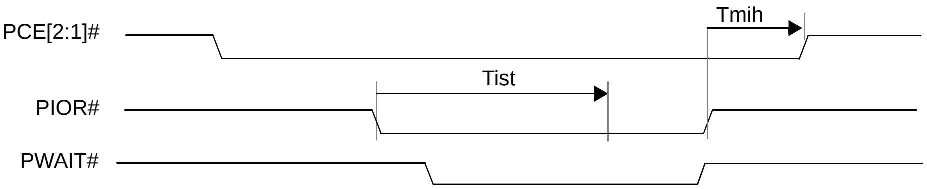


Figure 3-23. PCMCIA I/O Read PWAIT# Timing

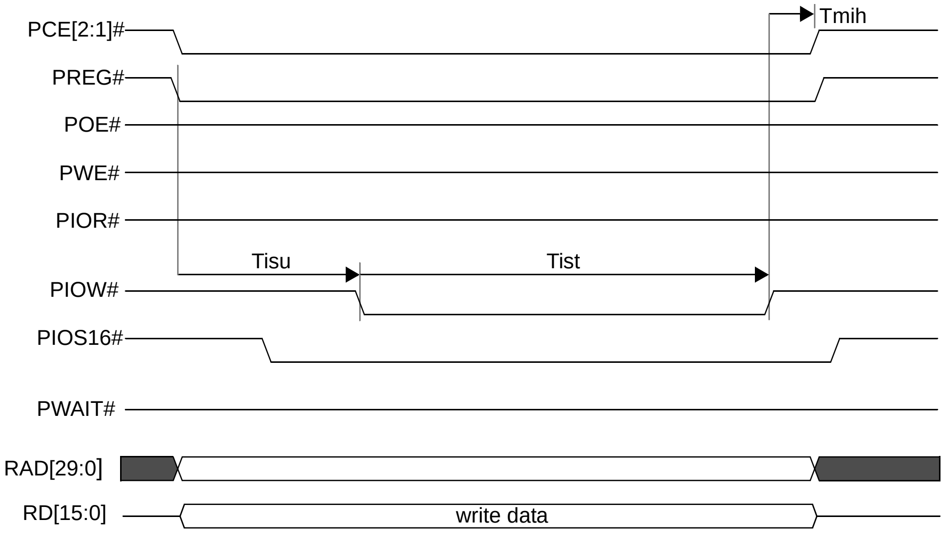


Figure 3-24. PCMCIA I/O Write Timing

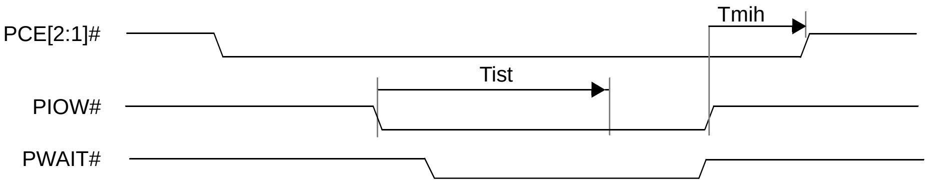


Figure 3-25. PCMCIA I/O Write PWAIT# Timing

www.DataSheet4U.com

www.DataSheet4U.com

3.2.5 NOR Flash Device Type

This section describes the static RAM interface that is implemented when the device type (**mem_stcfg*n***[DTY]) is programmed to 3. (See Section 3.2.1.1 "Static Bus Configuration Registers" on page 78.)

The NOR Flash signals are shown in Table 3-16.

3.2.5.1 Static Memory Timing Diagrams (NOR Flash)

This section presents static bus timing diagrams for reads and writes for NOR Flash. The diagrams show how the programmable timing parameters (in **mem_sttime*n***) can be used to adjust the interface timing to meet the requirements of a specific device.

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 14.0 "Electrical and Thermal Specifications" on page 439. (See also Section 14.4.2 "Asynchronous Static Bus Controller Timing" and Section 14.4.3 "Synchronous Static Bus Timing" on page 457.)

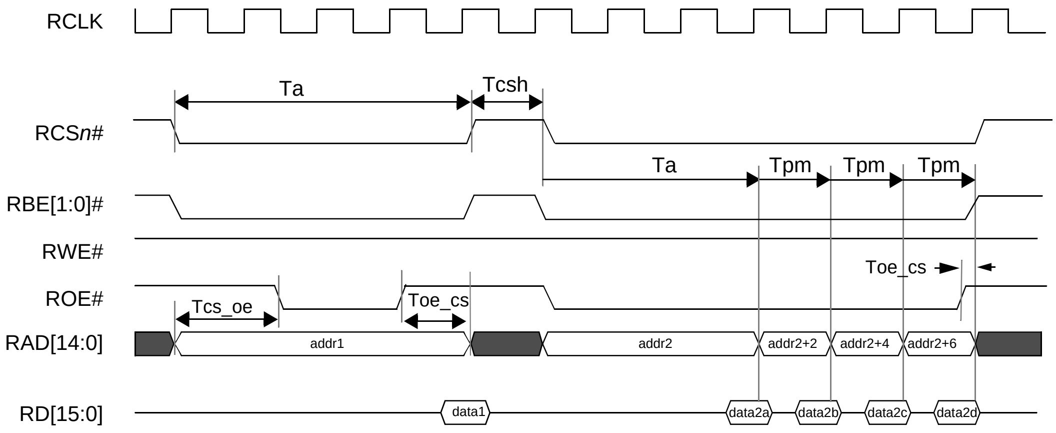
Timing parameters do not take into account system bus overhead which may add inter access delays. These delays are dependent on system design and are affected by the number of bus masters and the ability of other devices to hold the bus.

Read Timing Diagrams (NOR Flash)

Figure 3-26 shows static bus read timing with a single 16-bit read followed by a 2-word page mode burst read.

Table 3-16. NOR Flash Signals

Pin Name	Input/Output	Description
RAD[14:0]	O	Address Bus.
RD[15:0]	I/O	Data Bus.
RBE[1:0]#	O	Byte Enables: RBE0 #corresponds to RD[7:0]. RBE1 #is for RD[15:8].
RWE#	O	Write Enable.
ROE#	O	Output Enable.
RCS[3:0]#	O	Programmable Chip Selects (4 banks). RCS*n*# is not used when configured as a PCMCIA device.



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-26. NOR Flash 16-Bit Read Timing (Single Read, Burst) (mem_stcfg0[ALD] = 1)

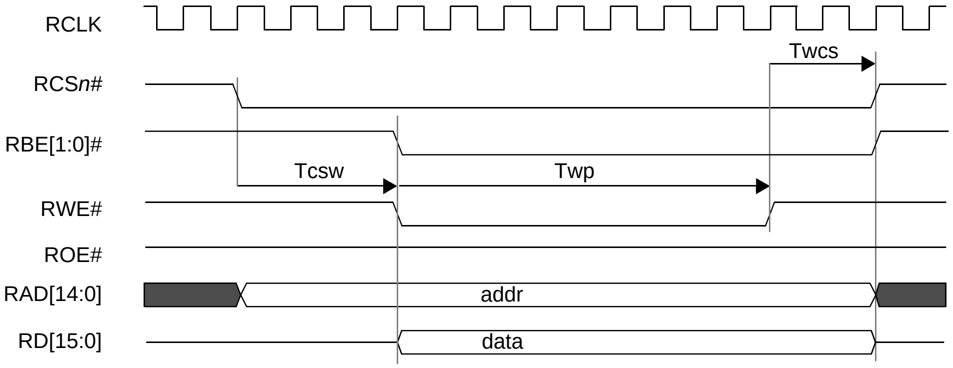
www.DataSheet4U.com

www.DataSheet4U.com

Write Timing Diagrams (NOR Flash)

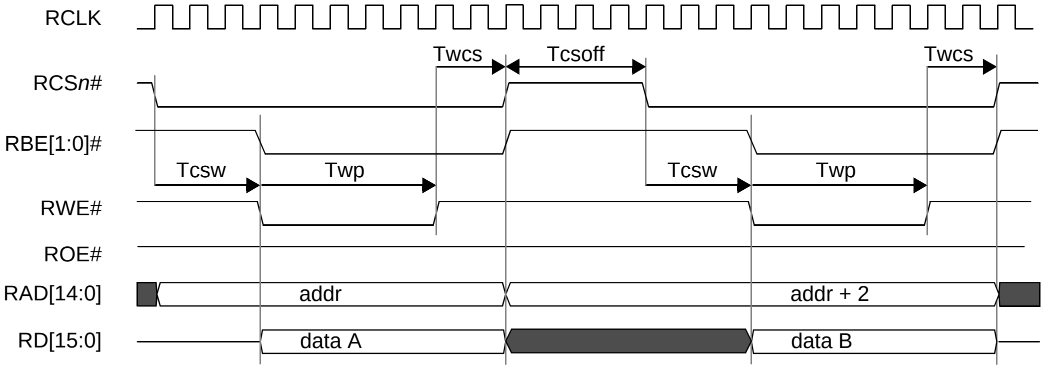
The timing diagrams below show the static bus write timing for NOR Flash device types. Figure 3-27 shows a single 16-bit write.

Figure 3-28 shows a 32-bit write. The transaction is treated as a burst. The diagram shows the timing with **mem_stcfg**[DE] = 1 (deassert mode); see Section 3.2.1.1 "Static Bus Configuration Registers" on page 78. For DE = 0, the Tcsoff parameter collapses to zero and the control signals remain asserted.



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] = 1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-27. NOR Flash 16-Bit Write Timing (mem_stcfg0[ALD] = 1)



NOTE: The external clock RCLK is available only in synchronous mode (**mem_stcfg**[S] =1). For *asynchronous* chip selects, the timing parameters are based on a separate internal-only clock. See Section 3.2.1.2 "Static Timing Registers" on page 81.

Figure 3-28. NOR Flash 32-Bit Write Timing (Burst) (mem_stcfg0[ALD] = 1)

www.DataSheet4U.com

www.DataSheet4U.com

3.2.6 NAND Flash Device Type

This section describes the static bus controller when the device type is NAND Flash (**mem_stcfg*n***[DTY] = 5). (See “Static Bus Configuration Registers” on page 78.)

NAND Flash is organized as a contiguous series of blocks. Each block contains a number of pages, and each page contains a certain number of columns. Within each block, an extended page area can be used to check the block’s viability for storage or can be used as additional storage, if needed.

NAND Flash manufacturers guarantee only a percentage of the total number of blocks as being good. Only the first block of memory is guaranteed valid; software must determine which other blocks are valid.

3.2.6.1 NAND Flash Device Registers

Unlike other memory types, NAND Flash does not present a direct, memory-mapped region to the system. Instead, access to NAND Flash memory is provided *indirectly* via three registers (command, address, and data) located on the NAND Flash device. Software addresses the NAND Flash device registers at offsets from the base address of the chip select. Table 3-17 shows the NAND Flash device registers and offsets from the chip select base address.

Table 3-17. NAND Flash Device Registers

Offset from Chip Select Base Address	Register Name	Description
0x0000	mem_stndcmd	NAND Command Register
0x0004	mem_stndaddr	NAND Address Register
0x0020	mem_stnddata	NAND Data Register

NAND Command Register

Writing to this register causes the controller to issue a command sequence with CMD as the payload. The system programmer must ensure this is a valid command.

mem_stndcmd **Offset = Chip Select Base Address + 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
																									CMD							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	R/W	0
7:0	CMD	Flash Command.	R/W	0x00

NAND Address Register

Writing to this register causes the controller to issue an address sequence. A byte write causes a single address byte to be sent out. Multiple writes cause the complete address to be written to the external device. The format of the address depends on the external NAND device.

mem_stndaddr **Offset = Chip Select Base Address + 0x4**

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

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	R/W	0
7:0	ADDR	Address Byte.	R/W	0x00

www.DataSheet4U.com

www.DataSheet4U.com

NAND Data Register

There is a single register for reading and writing data. It can be accessed in sizes of the device width, a word, or bursts of words. The following data access types are allowed:

- 8-bit device: byte, 1-8 word bursts
- 16-bit device: halfword, 1-8 word bursts

There is no address associated with this data. The controller simply passes data to or from the device as set up by previous command and address sequences.

mem_stnddata **Offset = Chip Select Base Address + 0x20**

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

Bits	Name	Description	R/W	Default
31:0	DATA	Data.	R/W	0

3.2.6.2 NAND Flash Signals

Table 3-18 shows the signals supporting the NAND Flash interface, including an 8- or 16-bit wide multiplexed I/O bus for command, address, and data.

Table 3-18. NAND Flash Signals

Au1210™/ Au1250™ Processor Signals	Type	NAND Interface	Description
RD[15:0]	I/O	I/O[15:0]	IO[7:0] (Command/Address/Data). IO[15:8] (Data). Provide the upper byte of data in 16-bit mode. Shared with other chip selects. IO pins float when RCS*n*# is negated.
RCLE, RALE	O	CLE, ALE	Command Latch Enable. When asserted, the command is latched into the command register on the rising edge of WE#. Address Latch Enable. When asserted, the address is latched into the address register on the rising edge of WE#.
RWE#	O	WE#	Write Enable. Command, address, and data are latched at the rising edge of WE# pulse. Shared with other chip selects.
ROE#	O	RE#	Read Enable. Controls the serial data-out. When asserted it drives the data onto the I/O bus. Data is valid after the falling edge of RE which also increments the internal column address counter by one. Shared with other chip selects.
RCS[3:0]#	O	CE#	Chip Enable. Chip select pin. The operation of this pin is continuously being modified with each generation of NAND Flash. For most devices, if CE is deasserted during write or erase cycles, the device is not returned to standby mode.

www.DataSheet4U.com

www.DataSheet4U.com

Table 3-18. NAND Flash Signals (Continued)

Au1210™/ Au1250™ Processor Signals	Type	NAND Interface	Description
RNB	I	R/B#	Ready/Not Busy. Indicates the status of the device operation. When low, it indicates that a write, erase, or random read operation is in progress and returns to high state upon completion. It is an open-drain output from the device and does not float to high-Z condition when the chip is deselected or when outputs are disabled. The RNB signal has an associated interrupt (interrupt number 23 on interrupt controller 0) that triggers when the NAND device becomes ready (RNB goes high).
GPIO[*x*]	O	WP#	Write Protect. Provides inadvertent write/erase protection during power transitions. This functionality can be provided by a GPIO output.

3.2.6.3 Static Memory Timing Diagrams (NAND Flash)

This section presents static bus timing diagrams for NAND Flash. The diagrams show how the programmable timing parameters (in **mem_sttime*n***) can be used to adjust the interface timing to meet specific device requirements.

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 14.0 "Electrical and Thermal Specifications" on page 439. (See also Section 14.4.2 "Asynchronous Static Bus Controller Timing" on page 453.)

The timing parameter values for setup and hold in **mem_sttime*n*** are global and are based on the system bus clock with timing parameters scaled by **mem_stcfg*n***[TS]. Note that the **mem_stcfg*n***[S] bit (synchronous mode) does not apply to NAND operation: If S is set in NAND mode, the NAND signal timings are still based off the system bus, and all timing parameters are still programmed in system-bus-clock resolution.

The NAND Flash controller cannot burst read or write an entire page of data. The maximum burst size is 32 bytes or 16 halfwords in a single transfer. Single reads and writes are burst. For example, a single read from the 32-bit data register (**mem_stnddata*n***) causes a 4-beat burst on an 8-bit device.

The chip select negates for each command and address data packet, and between bursts or for single reads and writes.

Busy timings depend on software: Software must poll **mem_ststat**[BSY] until the busy status clears or wait for interrupt (interrupt number 23 on interrupt controller 0) before beginning a read/write data sequence. When booting from NAND Flash, hardware checks **mem_ststat**[BSY] before fetching boot instructions stored in the NAND Flash device starting at address 0x0 in page0.

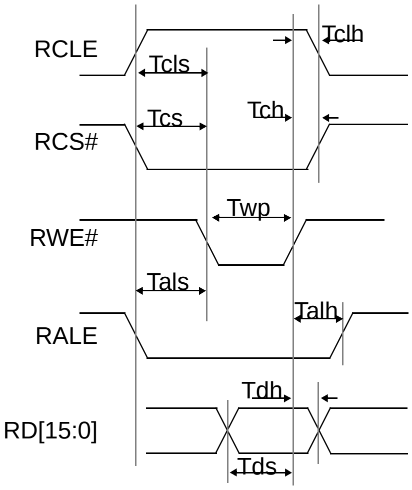


Figure 3-29. NAND Flash Command Phase Timing

www.DataSheet4U.com

www.DataSheet4U.com

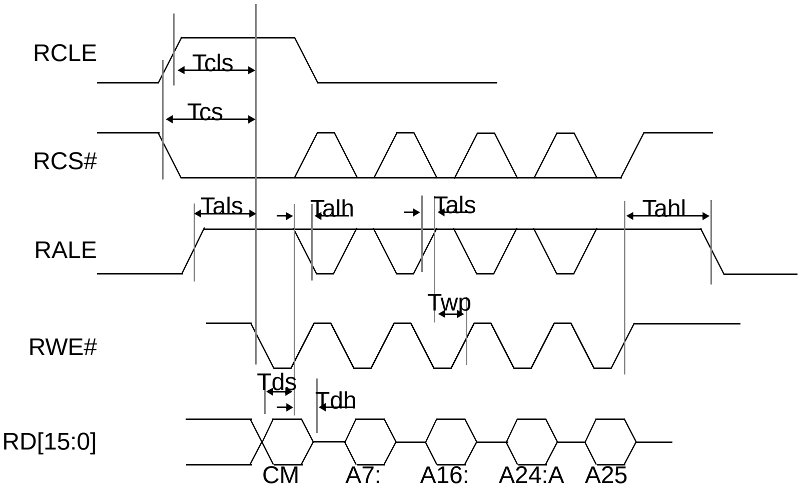


Figure 3-30. NAND Flash Address Phase Timing Diagram

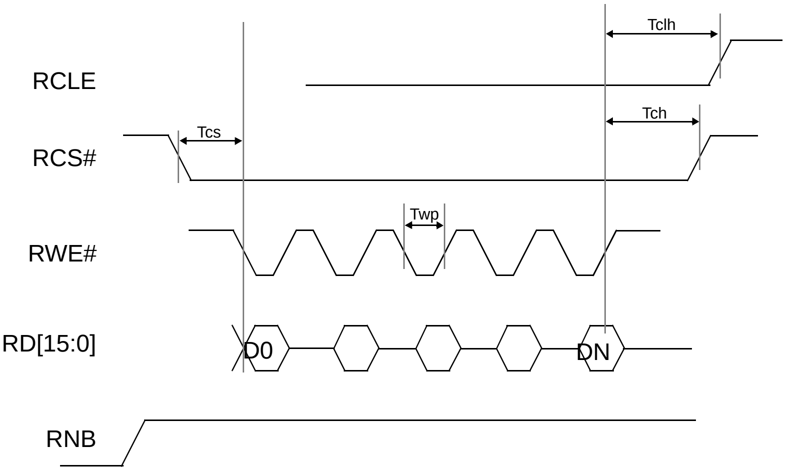


Figure 3-31. NAND Flash Data Write Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

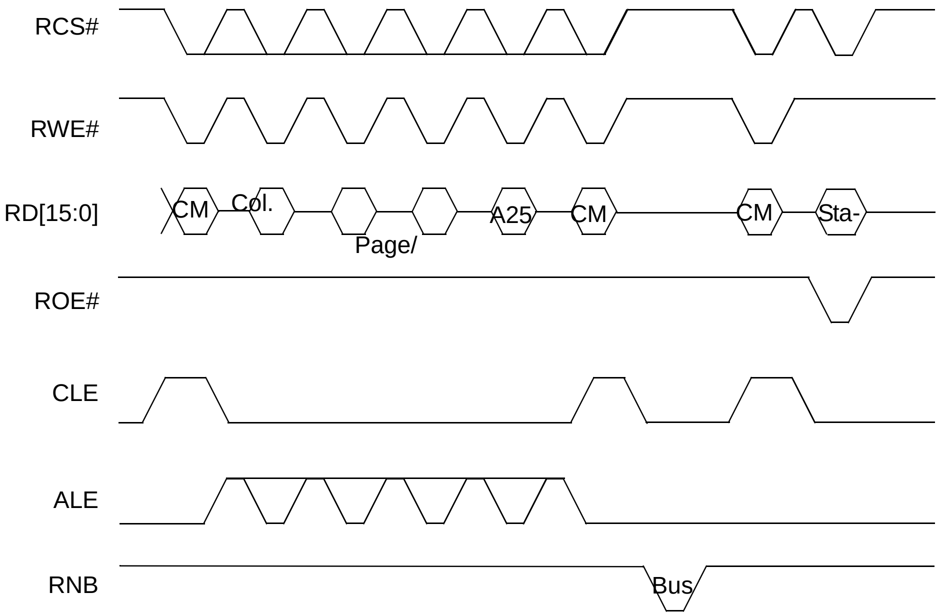


Figure 3-32. NAND Flash Block Erase Timing Diagram

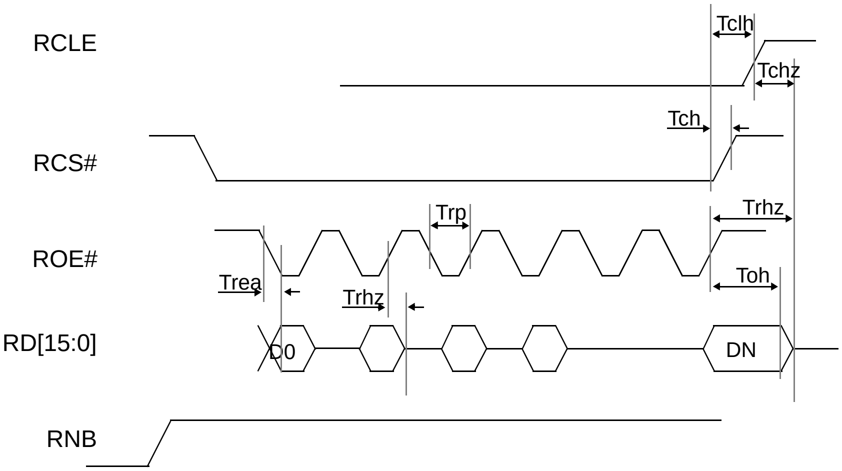


Figure 3-33. NAND Flash Data Read Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

3.2.6.4 NAND Flash Programming Considerations

Booting from NAND Flash

The device width (8-bit/16-bit) is selected via the BOOT[1:0] signals (see Table 11-1 on page 401).

NAND Flash boot code is restricted to the first page of the first block of the NAND Flash device. The boot code in this region cannot loop or branch because the memory can only be read sequentially. The static memory controller increments the address location for each column of the first page; however, during boot, the controller does not increment the page address and recycle the column address back to 0x0.

The boot procedure is as follows:

1) After reset, the NAND Flash controller's bootload hardware issues the appropriate command and address cycles to configure the NAND Flash device for reads.

2) The controller issues a read command to the 0x0 block of the NAND Flash after power-up.

3) All addresses to 0x1FC0xxxx are sent to the NAND Flash controller. Reads must be contiguous sequential words, up to the page size of the device.

4) Once the bootstrap code is fetched, system software must disable boot mode (**mem_stndctrl**[BOOT] = 0) and access the device through software.

5) Once the boot routine is transferred to the CPU core, it starts execution of the boot routine.

Example Transactions for Typical NAND Flash Devices

Software controls the sequence for each transaction type using the NAND Flash device registers, and polling **mem_ststat**[BSY] or waiting for the NAND-ready interrupt.

Reset Sequence:

1) Write 0xFF to **mem_stndcmd**

2) Poll **mem_ststat**[BSY], or wait for interrupt

Read Sequence:

1) Write 0x00 to **mem_stndcmd**

2) Write address byte to **mem_stndaddr**

3) Write address byte to **mem_stndaddr**

4) Write address byte to **mem_stndaddr**

5) Write address byte to **mem_stndaddr**

6) Poll **mem_ststat**[BSY], or wait for interrupt

7) Read data from **mem_stnddata** up to page size (loop in 1-byte, 1 word, 8-word burst increments)

Write Sequence:

1) Write 0x80 to **mem_stndcmd**

2) Write address byte to **mem_stndaddr**

3) Write address byte to **mem_stndaddr**

4) Write address byte to **mem_stndaddr**

5) Write address byte to **mem_stndaddr**

6) Write data to **mem_stnddata** up to page size (loop in 1-byte, 1 word, 8-word burst increments)

7) Write 0x10 to **mem_stndcmd**

8) Poll **mem_ststat**[BSY], or wait for interrupt

9) Read data from **mem_stnddata** for status

www.DataSheet4U.com

www.DataSheet4U.com

Block Erase Sequence:

1) Write 0x60 to **mem_stndcmd**
2) Write address byte to **mem_stndaddr**
3) Write address byte to **mem_stndaddr**
4) Write address byte to **mem_stndaddr**
5) Write 0xD0 to **mem_stndcmd**
6) Poll **mem_ststat**[BSY], or wait for interrupt
7) Write 0x70 to **mem_stndcmd**
8) Read data from **mem_stnddata** for status

3.2.7 IDE Device Type

This section describes the static bus controller when the device type is an IDE drive (**mem_stcfg*n***[DTY] = 6). (See Section 3.2.1.1 "Static Bus Configuration Registers" on page 78.)

The static bus controller supports a direct interface to an IDE drive. For 3.3V drives, the interface is glueless. For 5V drives, however, buffering with 5V tolerant devices is required because the static bus is not 5V tolerant. A system design that negates power must also include buffers to isolate all static bus controller signals connected to the drive.

Addressing for PIO mode reads and writes requires only three address signals. System software can map any three static bus address signals to control the IDE drive. The static bus controller is not required to use the address latch mechanism for IDE accesses (**mem_stcfg0**[ALD] = 1).

The timing parameters for the static bus controller also support IDE PIO mode. To fully support the mode as indicated, additional hard drive chip selects (HD_CS*n*#) are provided. This allows for full access to all registers within the IDE device.

An additional timing parameter has also been incorporated. IDE devices require a configurable holdtime from output-enable/write-enable negation to chip-select negation (**mem_stcfg*n***[Toecs]).

For IDE accesses, the page mode option is not applicable and must not be used.

3.2.7.1 IDE PIO Mode

When configured for IDE operation, the chip select controls both HD_CS0# and HD_CS1# for PIO mode accesses. HD_CS1# is memory-mapped to RAD[8], for status register read/write operations. Figure 3-34 shows the logic diagram for HD_CS*n*#.

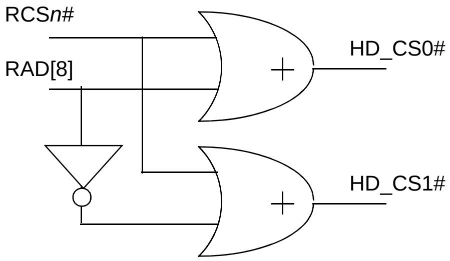


Figure 3-34. HD_CS*n*# Logic Diagram

Any chip select can be configured as IDE, but the Au1210 and Au1250 processors do not boot directly from an IDE device. During boot, RCS0# must be configured to select from one of the supported boot devices.

www.DataSheet4U.com

www.DataSheet4U.com

3.2.7.2 Static Memory Timing Diagrams (IDE)

This section presents static bus timing diagrams for reads and writes for IDE devices. The diagrams show how the programmable timing parameters can be used to adjust the interface timing to meet the requirements of a specific device.

The timing parameters for IDE PIO mode require an additional setting. The negation of ROE#/RWE# prior to HD_CS*n*# must be programmed for all PIO accesses. The timing value is dependent on the drive used and the drive transfer mode (0-4).

Setup, hold, and delay timing specifications (electrical switching characteristics) are presented in Section 14.0 "Electrical and Thermal Specifications" on page 439. (See also Section 14.4.2 "Asynchronous Static Bus Controller Timing" on page 453 and Section 14.4.3 "Synchronous Static Bus Timing" on page 457.)

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

IDE Write Cycle

Figure 3-35 shows the required output for an IDE write cycle to an IDE device.

IDE Read Cycle

For an IDE PIO mode read, the additional timing parameter unique to an IDE device is Toecs. Each read access must have a setting for this parameter. IDE PIO transfer modes (0-4) have specific minimum values for each mode. See Figure 3-36.

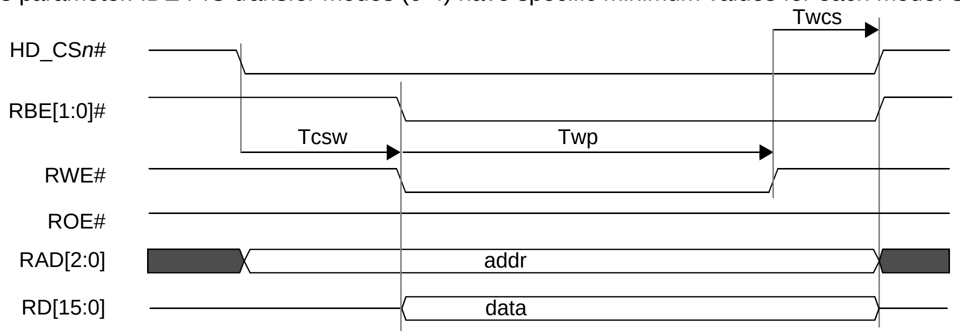


Figure 3-35. IDE Write Timing

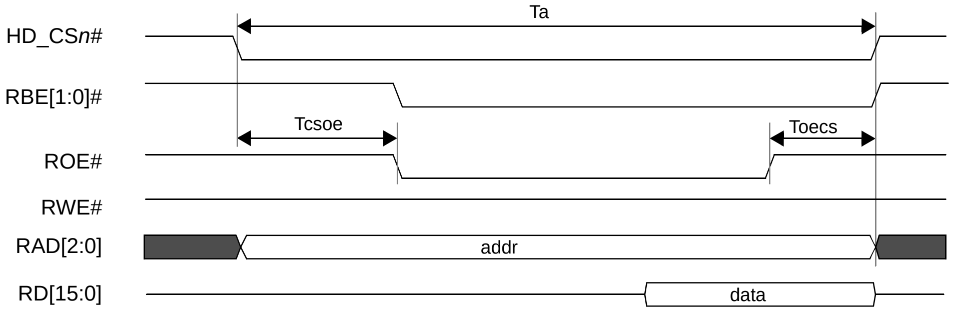


Figure 3-36. IDE Read Timing

www.DataSheet4U.com

www.DataSheet4U.com

3.2.8 Static Bus Controller Programming Considerations

3.2.8.1 Chip Select Clocking

All chip selects share a common clock derived from the system bus clock. When a chip select is configured for synchronous mode (**mem_stcfg*n***[S] = 1), the static bus controller provides the clock externally on RCLK. Once a chip select is configured for synchronous operation, the RCLK output begins running and continues to run even when the chip select is not active. The RCLK frequency is programmed in **mem_stcfg0**[DIV]. (See Section 3.2.1.1 "Static Bus Configuration Registers" on page 78.)

For interface timing, each chip select has a static timing register **mem_sttime*n*** to adjust the access timing parameters for the specific device. (See Section 3.2.1.2 "Static Timing Registers" on page 81.) In addition, a wait input signal EWAIT# (or PWAIT# for PCMCIA devices) is provided to delay each access, if needed.

3.2.8.2 Page Mode Transfers

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

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

- *Ta* is the time from chip select assertion to the first beat of valid data.
- *Tpm* is the time between beats.

Figure 3-11 on page 92 shows an example page mode read with the timing parameters Ta and Tpm.

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

3.2.8.3 Halfword Ordering

A 32-bit access (load/store/fetch or DMA) causes the static bus controller to perform two beats on RD[15:0]. The first beat is read from/written to *sysbus_data*[15:0] (even/lower halfword), and the second beat is read from/written to *sysbus_data*[31:16] (odd/upper halfword) (*sysbus_data* is the internal system bus data word.).

The ordering of the physical offsets for the two beats depends on **mem_stcfg**[BE].

Because the static bus controller is not aware of the endian mode of the Au1 core, potential halfword swapping conflicts can arise. Upon reset, chip selects default to little-endian byte ordering (**mem_stcfg**[BE] = 0). Figure 3-37 shows the data formats for the 32-bit system bus and for a little-endian chip select.

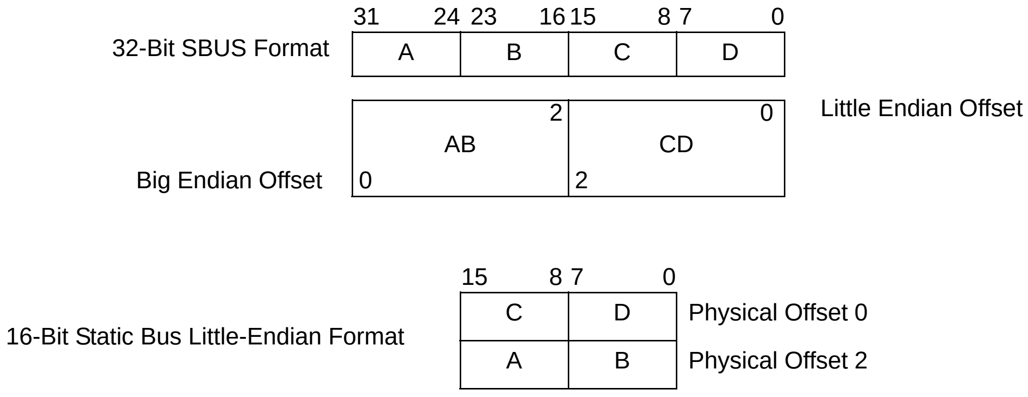
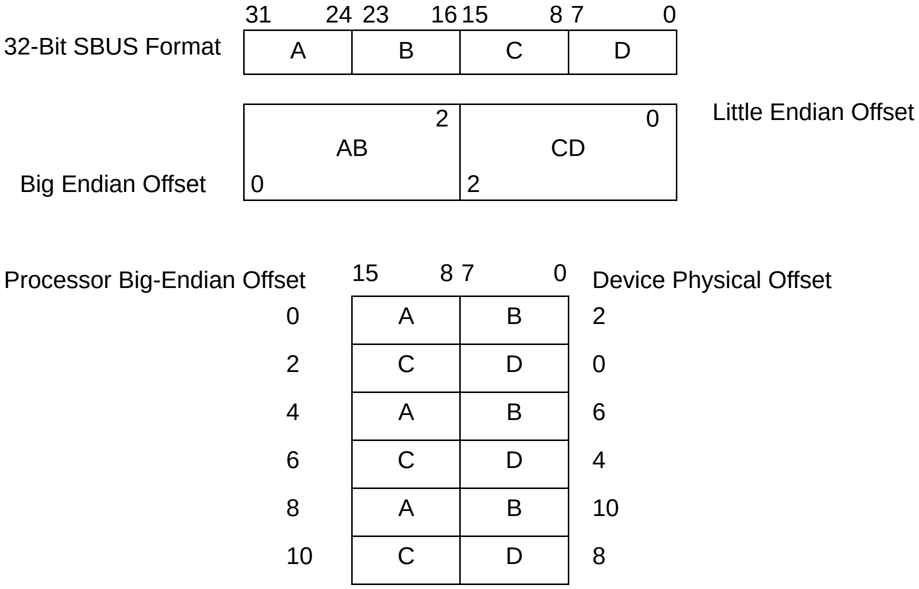


Figure 3-37. Chip Select Little-Endian Data Format (Default)

www.DataSheet4U.com

www.DataSheet4U.com

When a chip select is in little-endian mode, the static bus controller accesses the least-significant halfword CD at physical offset 0 and accesses the most-significant halfword AB at physical offset 2. When the Au1 core is also in little-endian mode, the requested Au1 core offsets match the physical offsets of the 16-bit device. That is, the static bus controller and the Au1 core have the same view of memory. However, when the processor core is in big-endian mode, the default ordering of the static bus controller effectively reverses the ordering of the halfwords from what the big-endian Au1 core expects, as shown in Figure 3-38.



Processor Big-Endian Offset	15 8	7 0	Device Physical Offset
0	A	B	2
2	C	D	0
4	A	B	6
6	C	D	4
8	A	B	10
10	C	D	8

Figure 3-38. Big-Endian Au1 Core and Little-Endian Chip Select

www.DataSheet4U.com

www.Data

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

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

To prevent halfword swapping, configure the chip select for big-endian mode (**mem_stcfg**[BE] = 1) before accessing the memory. (If booting from static memory, see Section 11.3.1 "Endianness and 16-Bit Static Bus Boot" on page 402.) The static bus controller inverts RAD1 for transfers on chip selects in big-endian mode, as shown in Figure 3-39.

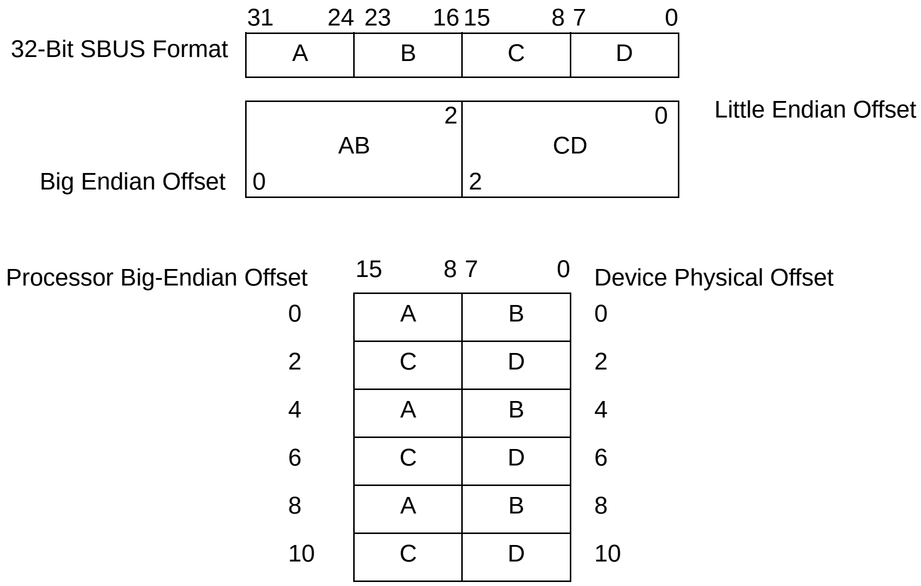


Figure 3-39. Big-Endian Au1 Core and Big-Endian Chip Select

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

4 Descriptor-Based DMA (DDMA) Controller

The descriptor-based direct memory access (DDMA) controller on the Au1210™ and Au1250™ processors use descriptors in memory to describe channel-specific attributes. The features that make the DDMA controller a flexible DMA solution include the following:

- 16 independent channels can be allocated to integrated peripherals and external DMA requests.
- Supports memory-to-FIFO, FIFO-to-memory, FIFO-to-FIFO, and memory-to-memory DMA transfers.
- Full 36-bit source/destination addresses are supported with no alignment requirement.
- Operates on chained, branching, and circular descriptor lists.
- Block and stride values for source/destination data to allow non-contiguous memory transfers (scatter/gather).
- Provides optional interrupt and updated status after descriptor completed, if needed.
- Descriptors allow source-to-destination transfers, compare and branch, subroutine calls, and literal write transfers.
- Channels can be assigned to high or low priority arbitration pools.
- Choice of two arbitration policies: round-robin or weighted priority.

The DDMA controller also accesses integrated peripherals and external memory-mapped devices through the static bus controller using a GPIO as a request. GPIO[4] and GPIO[12] can be programmed to act as external DMA request signals. See Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139 to configure these GPIO signals to act as DMA requests.

4.1 DDMA Operation

The DDMA controller supports sixteen channels with each channel containing a set of channel-specific registers. The DDMA controller operates on channel descriptors stored in system memory. Each descriptor contains transfer-specific information, such as the transfer priority (high or low) and which external DMA request signal or integrated peripheral device is involved. The channel's descriptor pointer register contains the memory address of the current channel descriptor. Figure 4-1 on page 116 shows a block diagram of the DDMA controller and also includes example descriptors and buffers in memory.

www.DataSheet4U.com

www.DataSheet4U.com

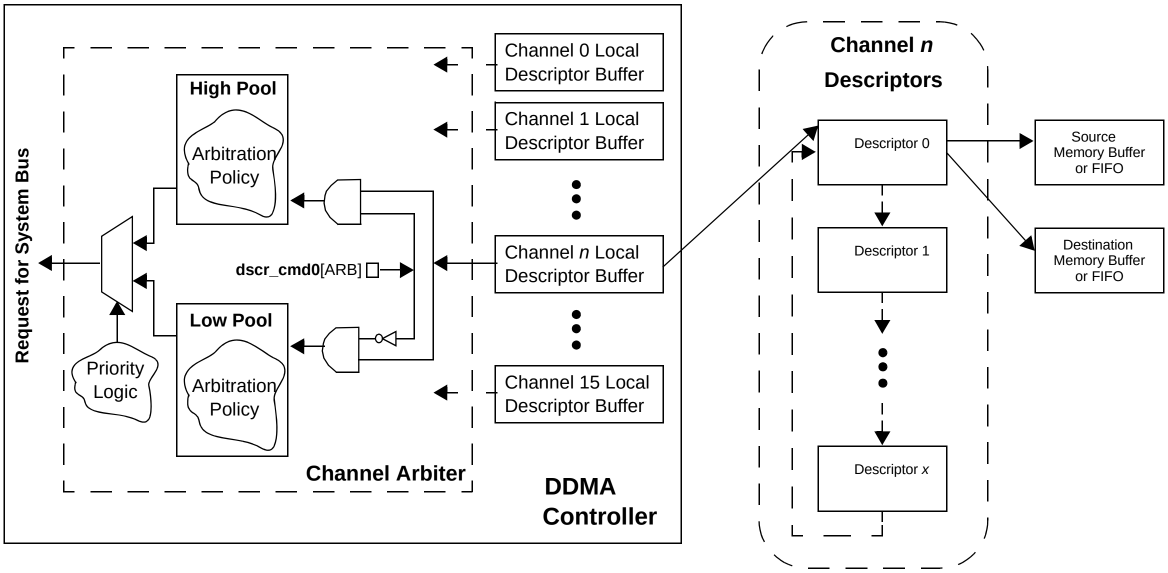

Request for System Bus
High Pool
Arbitration Policy
Low Pool
Arbitration Policy
Priority Logic
dscr_cmd0[ARB]
Channel Arbiter
Channel 0 Local Descriptor Buffer
Channel 1 Local Descriptor Buffer
Channel n Local Descriptor Buffer
Channel 15 Local Descriptor Buffer
DDMA Controller
Channel n Descriptors
Descriptor 0
Descriptor 1
Descriptor x
Source Memory Buffer or FIFO
Destination Memory Buffer or FIFO

Figure 4-1. DDMA Controller Block Diagram

Each channel contains a write only doorbell register (**ddma*n*_dbell**) that is used to activate the channel. When software writes to **ddma*n*_dbell**, the DDMA controller fetches the descriptor, loads it into the channel's local descriptor buffer, and clears the doorbell. The controller then examines the descriptor to determine the transfer type and attributes. The controller asserts a channel request to the channel arbiter to obtain the system bus for the transfer (see Section 4.1.1 "Channel Priority and Arbitration" on page 116).

The process of asserting channel requests and arbitrating for the system bus continues until all data is transferred in accordance with the descriptor. After the transfer is complete, the DDMA controller updates the status of the current descriptor in memory and marks it completed if no error occurred. The status pointer register contains the address of the status word to be written to the descriptor upon completion. The controller also generates an interrupt, if enabled.

The pointer to the next descriptor is read from the current descriptor and loaded into the local descriptor pointer register. This pointer is then used to load the next descriptor from memory. The DDMA controller continues to fetch descriptors and transfer data until it encounters a nonvalid descriptor (**dscr_cmd0**[V] = 0) or software disables the channel (**ddma*n*_cfg**[EN] = 0).

Transfers can occur using multiple descriptors.

4.1.1 Channel Priority and Arbitration

Each channel is assigned to either a high or low priority pool. The channel arbiter checks the high priority pool up to eight times for every one time it checks the low priority pool. The DDMA controller places a channel request into either the high or low priority pool based on the arbitration bit in the descriptor command0 field (**dscr_cmd0**[ARB]) (see Figure 4-1).

For each pool, the channel arbiter supports two arbitration policies, round-robin and weighted priority.

- For weighted priority, the arbiter sorts the request by channel number, with channel 0 having the highest priority and channel 15 having the lowest.
- For round-robin priority, the arbiter selects the channels with equal priority, stepping through all requesting channels before starting over.

As part of the configuration of the DDMA controller, software selects the arbitration policies using **ddma_config**[AH] for the high priority pool and **ddma_config**[AL] for the low priority pool.

The channel arbitration is summarized in Figure 4-2 on page 117.

www.DataSheet4U.com

www.DataSheet4U.com

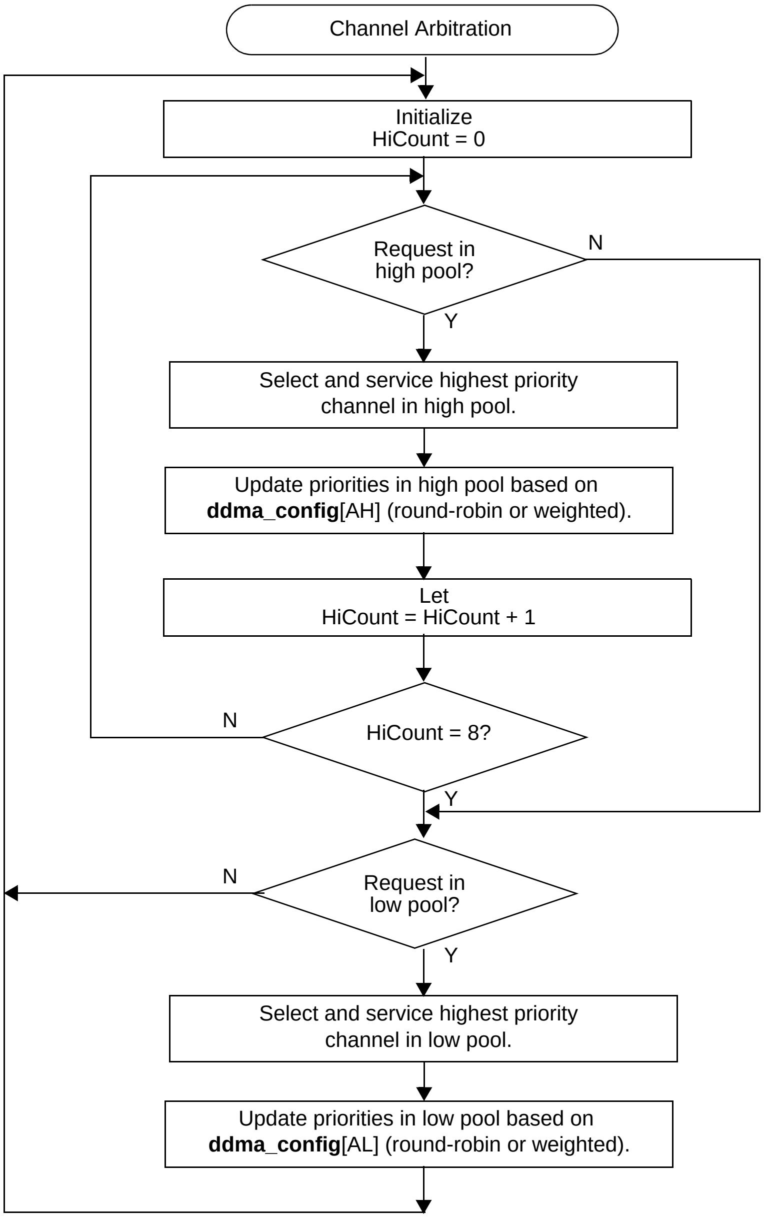

Channel Arbitration
Initialize
HiCount = 0
Request in
high pool?
N
Y
Select and service highest priority
channel in high pool.
Update priorities in high pool based on
ddma_config[AH] (round-robin or weighted).
Let
HiCount = HiCount + 1
N
HiCount = 8?
Y
N
Request in
low pool?
Y
Select and service highest priority
channel in low pool.
Update priorities in low pool based on
ddma_config[AL] (round-robin or weighted).

Figure 4-2. Channel Arbitration

4.1.2 Supported Peripherals

Table 4-1 on page 118 shows the integrated peripherals that are supported by the DDMA controller. The information provided in Table 4-1 is used to build descriptors (see Section 4.3 "DDMA Descriptors" on page 127):

- Program the source and destination device IDs, and the device width in the descriptor command 0 field (**dscr_cmd0**).
- Program the source transfer size in the descriptor source 1 field (**dscr_source1**).
- Program the destination transfer size in the descriptor destination 1 field (**dscr_dest1**).
- Program the source and destination addresses in the descriptor source and destination pointer fields (**dscr_source0** and **dscr_dest0**).
- Non-programmable device information must be written to the descriptor as defined in Table 4-1.

www.DataSheet4U.com

www.DataSheet4U.com

Table 4-1. Peripheral Addresses and Selectors

Peripheral Device	Device ID	Transfer Size (Datums)	Device Width (Bits)	Physical Address	Request Edge/Level	Request Polarity
UART0 transmit	0	programmable	8	0x0 1110 0004	level	high
UART0 receive	1	programmable	8	0x0 1110 0000	level	high
UART1 Transmit	2	programmable	8	0x0 1120 0004	level	high
UART1 Receive	3	programmable	8	0x0 1120 0000	level	high
DMA_REQ0 (GPIO[4])	4	programmable	programmable	programmable	programmable	programmable
DMA_REQ1 (GPIO[12])	5	programmable	programmable	programmable	programmable	programmable
MAE back end	6	programmable	32	0x0 1401 0000	level	high
MAE front end	7	programmable	32	0x0 1401 2000	level	high
SD0 transmit	8	programmable	8	0x0 1060 0000	level	high
SD0 receive	9	programmable	8	0x0 1060 0004	level	high
SD1 transmit	10	programmable	8	0x0 1068 0000	level	high
SD1 receive	11	programmable	8	0x0 1068 0004	level	high
AES output (Au1250 only)	12	programmable	32	0x0 0030 0008	level	high
AES input (Au1250 only)	13	programmable	32	0x0 0030 0004	level	high
PSC0 transmit	14	programmable	programmable	0x0 11A0 001C	level	high
PSC0 receive	15	programmable	programmable	0x0 11A0 001C	level	high
PSC1 transmit	16	programmable	programmable	0x0 11B0 001C	level	high
PSC1 receive	17	programmable	programmable	0x0 11B0 001C	level	high
CIM receive FIFO A	18	programmable	programmable	0x0 1400 4020	level	high
CIM receive FIFO B	19	programmable	programmable	0x0 1400 4040	level	high
CIM receive FIFO C	20	programmable	programmable	0x0 1400 4060	level	high
MAE Both Done	21	programmable	programmable	0x0 10B0 001C	level	high
LCD Retrace	22	programmable	programmable	programmable	level	high
NAND Controller	23	programmable	programmable	programmable	level	high
PSC0 Sync1	24	programmable	programmable	programmable	level	high
PSC1 Sync1	25	programmable	programmable	programmable	level	high
CIM Frame Sync	26	programmable	programmable	programmable	level	high
Interrupt Controller 0 Request 0	27	programmable	programmable	programmable	level	high
GPIO[8]	28	programmable	programmable	programmable	level	high
Interrupt Controller 0 Request 1	29	programmable	programmable	programmable	level	high
Request throttle (for memory transfers)	30	programmable	programmable	programmable	level	high
Request always high (for memory transfers)	31	programmable	programmable	programmable	level	high

www.DataSheet4U.com

www.DataSheet4U.com

4.2 DDMA Controller Registers

The DDMA controller registers configure and control the operation of the interface. The controller contains both global and channel-specific registers.

The base address of the global register block is shown in Table 4-2.

Table 4-2. DDMA Global Register Block Base Address

Register Block	Base Address (Physical)	KSEG1 Base Address
Global	0x0 1400 3000	0xB400 3000

The base address of each channel-specific register block is shown in Table 4-3.

Table 4-3. DMA Channel Base Addresses

DMA Channel	Base Address (Physical)	KSEG1 Base Address
ddma0	0x0 1400 2000	0xB400 2000
ddma1	0x0 1400 2100	0xB400 2100
ddma2	0x0 1400 2200	0xB400 2200
ddma3	0x0 1400 2300	0xB400 2300
ddma4	0x0 1400 2400	0xB400 2400
ddma5	0x0 1400 2500	0xB400 2500
ddma6	0x0 1400 2600	0xB400 2600
ddma7	0x0 1400 2700	0xB400 2700
ddma8	0x0 1400 2800	0xB400 2800
ddma9	0x0 1400 2900	0xB400 2900
ddma10	0x0 1400 2A00	0xB400 2A00
ddma11	0x0 1400 2B00	0xB400 2B00
ddma12	0x0 1400 2C00	0xB400 2C00
ddma13	0x0 1400 2D00	0xB400 2D00
ddma14	0x0 1400 2E00	0xB400 2E00
ddma15	0x0 1400 2F00	0xB400 2F00

www.DataSheet4U.com

www.DataSheet4U.com

4.2.1 Global Registers

The global registers, shown in Table 4-4, apply to the entire DDMA controller.

Table 4-4. DDMA Controller Global Registers

Offset from 0x0 1400 3000 (Physical) 0xB400 3000 (KSEG1)	Register Name	Description
0x0000	ddma_config	DDMA General Configuration Register
0x0004	ddma_intstat	DMA Interrupt Status Register
0x0008	ddma_throttle	DMA Request Throttle Register
0x000C	ddma_inten	DDMA Interrupt Enable Register
0x0010	—	Reserved

4.2.1.1 DDMA General Configuration Register

This register determines the arbitration policies (round-robin or weighted priority) for the high and low priority arbitration pools. (See Section 4.1.1 "Channel Priority and Arbitration" on page 116.)

ddma_config **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
																														AF	AH	AL
Rese	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:3	—	Reserved.	—	—
2	AF	Descriptor fetch arbitration policy. 0 Round-robin 1 Weighted priority	R/W	0
1	AH	Arbitration policy for channels in the high-priority arbitration pool. 0 Round-robin 1 Weighted priority	R/W	0
0	AL	Arbitration policy for channels in the low-priority arbitration pool. 0 Round-robin 1 Weighted priority	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

4.2.1.2 DMA Interrupt Status Register

This read only register reflects the interrupt request status for each channel.

ddma_intstat **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15:0	IRQ[15:0]	Interrupt request status for DMA channels 15:0. 0 No interrupt request is pending. 1 An interrupt request is pending for this channel. **ddma_intstat**[*n*] reflects the interrupt request status for the corresponding DMA channel (**ddma*n*_irq**[IN]) regardless of whether the interrupt is enabled in **ddma_inten**[*n*].	R	0

4.2.1.3 DMA Request Throttle Register

This register allows the user to effectively lower the bandwidth of memory transfers to prevent saturation. The request throttle is connected to ID 30. It can be used to throttle requests instead of constantly requesting via ID 31 (request always high).

ddma_throttle **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
	EN																DIV															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	EN	Enable throttle for memory transfers. 0 Disable throttle. 1 Enable throttle.	R/W	0
30:16	—	Reserved.	—	—
15:0	DIV	Request throttle counter divider. Request frequency = (system bus frequency) / (DIV+1) Increase DIV to decrease the frequency of channel requests.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

4.2.1.4 DDMA Interrupt Enable Register

This register allows individual channel interrupts to be disabled. All channels are enabled by default. The channel interrupts are OR'd together to create one interrupt source (source number 3 on interrupt controller 0), as shown in Figure 4-3. See Section 5.0 "Interrupt Controller" on page 143 for more information on how to program interrupts.

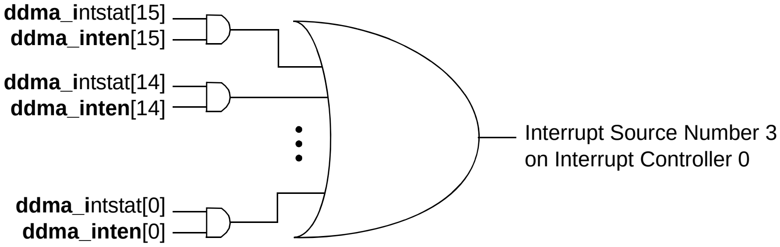


Figure 4-3. Logic for DDMA Controller Interrupts

ddma_inten **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15:0	ENABLE[15:0]	Interrupt enable for DMA channels 15:0. 0 Disable interrupts for this channel. 1 Enable interrupts for this channel.	R/W	0xFFFF

4.2.2 Channel-Specific Registers

Each DMA channel has its own block of channel-specific registers for channel configuration, status, and control. Table 4-5 shows the channel-specific registers duplicated for each of the sixteen DMA channels.

Table 4-5. DMA Channel-Specific Registers

Offset from Base of DMA Channel *n* (Note1)	Register Name	Description
0x0000	ddma*n*_cfg	Channel *n* Configuration
0x0004	ddma*n*_desptr	Channel *n* Descriptor Pointer
0x0008	ddma*n*_statptr	Channel *n* Status Pointer
0x000C	ddma*n*_dbell	Channel *n* Doorbell
0x0010	ddma*n*_irq	Channel *n* Interrupt Request
0x0014	ddma*n*_stat	Channel *n* Status
0x0018	ddma*n*_bytecnt	Channel *n* Remaining Byte Count
0x001C - 0x00FF	—	Reserved

Note1. See Table 4-3 on page 119 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

4.2.2.1 Channel Configuration Registers

The format for the Channel-specific Configuration registers is shown below.

ddma*n*_cfg **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
																							SED	SP	DED	DP	SYNC		DFN	SBE	DBE	EN
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:10	—	Reserved.	—	—
9	SED	Source DMA Request Level/edge. 0 Level trigger 1 Edge trigger	R/W	0
8	SP	Source DMA Request Polarity. 0 Positive polarity: High-level or rising-edge trigger 1 Negative polarity: Low-level or falling-edge trigger	R/W	0
7	DED	Destination DMA Request Level/edge. 0 Level trigger 1 Edge trigger	R/W	0
6	DP	Destination DMA Request Polarity. 0 Positive polarity: High-level or rising-edge trigger 1 Negative polarity: Low-level or falling-edge trigger	R/W	0
5	SYNC	Synchronization for the Static Bus Controller. 0 Normal operation. 1 Force an extra configuration read to the static bus controller after finishing a destination transfer to flush the static bus controller for external devices.	R/W	0
4	—	Reserved.	—	—
3	DFN	Descriptor Fetch Non-coherent. 0 Fetch descriptors coherently. 1 Fetch descriptors non-coherently.	R/W	0
2	SBE	Source Big Endian. Defines the byte ordering for FIFOs and buffers in memory. 0 Little endian 1 Big endian	R/W	0
1	DBE	Destination Big Endian. Defines the byte ordering for FIFOs and buffers in memory. 0 Little endian 1 Big endian	R/W	0
0	EN	Channel Enable. 0 Disable the channel regardless of the state of the doorbell register (**ddma*n*_dbell**). After disabling the channel, software must wait until the DDMA controller halts the channel (**ddma*n*_stat**[H] = 1) before modifying the channel configuration. Disabling a channel does not affect the interrupt status of the channel; ddman_irq and ddma_intstat 1 Enable the channel.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

4.2.2.2 Channel Descriptor Pointer Register

This register contains the physical address in memory of the descriptor. This register must be initialized with the 32-byte aligned physical address of the first descriptor associated with the channel. This register is automatically updated with the next descriptor address (**dscr_nxtptr**[NPTR]) at the end of the current transfer. (The next descriptor address is located in the next descriptor pointer field within the descriptor structure; see Section 4.3, "DDMA Descriptors".)

ddma*n*_desptr **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:5	DESPTR	Descriptor Pointer. Contains address[31:5] of the 32-byte aligned physical address of the current descriptor.	R/W	UNDEF
4:0	—	Reserved.	—	—

4.2.2.3 Channel Status Pointer Register

This register contains the physical address where the descriptor status information should be written. After processing a descriptor, the controller writes the descriptor status to the memory location pointed to by **ddma*n*_statptr**. The status information written is determined by each descriptor's **dscr_cmd0**[SP] and **dscr_cmd0**[ST] bits. If the programmer wishes to use the status reporting feature, this register must be programmed to a valid address.

ddma*n*_statptr **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
	STPTR																														▨	
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:2	STPTR	Status Pointer. Contains the word-aligned physical address[31:2] where descriptor status is written.	R/W	UNDEF
1:0	—	Reserved.	—	—

4.2.2.4 Channel Doorbell Register

A write to this register activates the channel and fetches a descriptor. See Section 4.5 "DDMA Controller Programming Considerations" on page 139 for information on the doorbell functionality. The format of the channel doorbell registers is as follows.

ddma*n*_dbell **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31:0	DB	A write to this register sets the channel status doorbell bit (**ddma*n*_stat[DB]** = 1) and causes the DDMA controller to fetch the descriptor at the descriptor pointer address. See Section 4.2.2.6 "Channel Status Register" on page 125.	W	UNDEF

www.DataSheet4U.com

www.DataSheet4U.com

4.2.2.5 Channel Interrupt Register

dma*n*_irq **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:1	—	Reserved.	—	—
0	IN	Interrupt Request. If interrupts are enabled in the current descriptor (**dscr_cmd0**[IE] = 1), the DDMA controller sets this bit after processing the descriptor. **ddma*n*_irq**[IN] shows the channel interrupt request status regardless of whether the interrupt is enabled in **ddma_inten**[*n*]. The value of **ddma*n*_irq**[IN] is reflected in **ddma_intstat**[*n*]. This register is set by hardware and must cleared by software. See Section 4.5 "DDMA Controller Programming Considerations" on page 139 for more information.	R/W	0

4.2.2.6 Channel Status Register

ddma*n*_stat **Offset = 0x0014**

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

Bits	Name	Description	R/W	Default
31:3	—	Reserved.	—	—
2	DB	Doorbell. The DDMA controller sets this bit when software writes to the channel doorbell register (**ddma*n*_dbell**). (See Section 4.5 "DDMA Controller Programming Considerations" on page 139 for information on the doorbell functionality.) The DDMA controller clears this bit when it fetches the descriptor. Software can clear this bit to prevent a descriptor from being fetched. The channel must first be halted before clearing this bit. (See the description of the H bit below.) After re-enabling the channel (**ddma*n*_cfg**[EN] = 1), software can re-start descriptor fetching by writing to **ddma*n*_dbell**. Software must write a ‘1’ to this bit in order to clear it; writing a zero has no effect.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
1	V	Current Descriptor Valid. 0 Current descriptor is not valid. 1 Current descriptor is valid. The DDMA controller clears this bit when (**dscr_cmd0**[V] = 0) and sets it when (**dscr_cmd0**[V] = 1). Software can abort the current descriptor (and allow a new descriptor to replace it) by clearing this valid bit. The channel must first be halted before clearing this bit. (See the description of the H bit below.) When invalidating the current descriptor, be sure also to clear the doorbell bit. (See the description of the DB bit above.) Software must write a ‘1’ to this bit in order to clear it; writing a zero has no effect.	R/W	0
0	H	Channel Halted. 0 Channel is active. 1 Channel has halted. The DDMA controller sets this bit after halting the channel. After disabling the channel (**ddma*n*_cfg**[EN] = 0), software must poll this bit to determine if the controller has halted the channel.	R	1

4.2.2.7 Channel Bytecount Register

ddma*n*_bytecnt **Offset = 0x0018**

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

Bits	Name	Description	R/W	Default
31:22	—	Reserved.	—	—
21:0	BC	Byte Count. Valid only when the channel is idle (halted). Contains the remaining byte count for a channel.	R	UNDEF

www.DataSheet4U.com

www.DataSheet4U.com

4.3 DDMA Descriptors

DDMA descriptors are memory structures that describe a data transfer between a source and destination. These descriptors contain defined fields that instruct the DDMA controller how to transfer the data. Descriptors must be aligned on a 32-byte boundary in memory.

The DDMA controller supports three types of transfers:

- Source to Destination Transfer. This is a normal DMA transfer where either or both sides can be FIFOs or memory. Both sides are set with a starting pointer, a block size / stride for non-contiguous address movement, address movement for incrementing, decrementing, or static addressing, as well as system bus burst sizing (1-8) and datum sizing (1-byte to 4-byte).
- Compare and Branch. Branching allows the user to evaluate a memory location and branch appropriately. The user can specify the source data to be a FIFO or memory. The source data is used in a comparison to select the next descriptor to be executed.
- Literal Write. Literal Write allows the user to copy 64 bits of data directly from the descriptor to memory. Instead of a source pointer and source block/stride, source data0 and source data1 can be specified. The destination side works as in a normal transfer, allowing the data to be written in any valid form.

The structure of the DDMA descriptor is different depending on the transfer type. Within the command field of each descriptor, a *descriptor-type* field determines the descriptor format.

These sections describes the descriptor structures supported:

- Section 4.3.1 "Standard DDMA Descriptor Structure" on page 127
- Section 4.3.2 "Compare and Branch DDMA Descriptor Structure" on page 134
- Section 4.3.3 "Literal Write DDMA Descriptor Structure" on page 137

4.3.1 Standard DDMA Descriptor Structure

The standard DDMA descriptor structure is shown below.

Standard DDMA Descriptor

	3 0
0x0	Command 0 (**dscr_cmd0**)
0x4	Command 1 (**dscr_cmd1**)
0x8	Source 0 (**dscr_source0**)
0xC	Source 1 (**dscr_source1**)
0x10	Destination 0 (**dscr_dest0**)
0x14	Destination 1 (**dscr_dest1**)
0x18	Status / Subroutine Pointer (**dscr_stat**)
0x1C	Next Descriptor Pointer (**dscr_nxtptr**)

The fields within the standard DDMA descriptor structure are described in the following subsections.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.1.1 Command 0

This entry in the descriptor contains command information for the transfer.

dscr_cmd0 **Offset = 0x00**

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
	V	MEM	SID					DID					SW		DW		ARB	DT		SN	DN	SM		IE				SP		CV	ST	

Bits	Name	Description
31	V	Valid. 0 Descriptor is not valid. 1 Descriptor is valid.
30	MEM	Memory-to-Memory Transfers. MEM must be set for memory-to-memory transfers. MEM must be cleared for all other transfer types.
29:25	SID	Source ID. Selects the device ID which determines the DMA request associated with the channel for source transfers. See Table 4-1 on page 118 for device ID encodings.
24:20	DID	Destination ID. Selects the device ID which determines the DMA request associated with the channel for destination transfers. See Table 4-1 on page 118 for device ID encodings.
19:18	SW	Source Width. Selects the width of the transfer on the system bus for source accesses. SW defines the source datum size. 00 Byte 01 Halfword 10 Word 11 Reserved
17:16	DW	Destination Width. Selects the width of the transfer on the system bus for destination accesses. DW defines the destination datum size. 00 Byte 01 Halfword 10 Word 11 Reserved
15	ARB	Arbitration Priority Pool. Selects the channel priority. See Section 4.1.1 "Channel Priority and Arbitration" on page 116. 0 Low priority pool 1 High priority pool
14:13	DT	Descriptor Type. Selects the descriptor type. 00 Standard descriptor (for source-to-destination transfers) 01 Literal-write descriptor 10 Compare and branch descriptor 11 Reserved
12	SN	Source Non-coherent. Transfers from an internal FIFO on the peripheral bus must be non-coherent. 0 Mark source data as coherent. 1 Mark source data as non-coherent.
11	DN	Destination Non-coherent. Transfers to an internal FIFO on the peripheral bus must be non-coherent. 0 Mark destination data as coherent. 1 Mark destination data as non-coherent.

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description
10	SM	Stride mode for both source and destination. See Section 4.5.3 "Stride Transfers" on page 140. 0 1-dimensional stride 1 2-dimensional stride. The datum size must be 32 bits (one word) for 2-dimensional stride transfers: **dscr_cmd0**[SW] and **dscr_cmd0**[DW] must be 0b10.
9	—	Reserved.
8	IE	Interrupt Enable. 0 Do not set **ddma*n*_irq**[IN] upon descriptor completion. 1 Set **ddma*n*_irq**[IN] upon descriptor completion.
7:5	—	Reserved.
4	SP	Copy Descriptor Pointer To Status Pointer. 0 The value written to **ddma*n*_statptr** is determined by the ST field**.** 1 Let **ddma*n*_statptr** = **ddma*n*_desptr**. The current descriptor status (**dscr_stat**) is written to **dscr_cmd0** upon descriptor completion**.**
3	—	Reserved.
2	CV	Clear Valid Bit. 0 Do not clear **dscr_cmd0[V]** upon descriptor completion**.** 1 Clear **dscr_cmd0[V]** upon descriptor completion**.**
1:0	ST	Status Instruction. Determines the information to report when the descriptor completes. The memory content pointed to by **ddma*n*_statptr** is changed as follows: 00 No change 01 Write the current descriptor status (**dscr_stat**). 10 Write the **dscr_cmd0** of the current descriptor with the valid bit cleared. 11 Write the remaining byte count.

4.3.1.2 Command 1

This entry in the descriptor contains the upper 4 bits of the starting physical address of the source and destination for the DMA transfer. These bits are concatenated with **dscr_source0**[SPTR] and **dscr_dest0**[DPTR] to make the 36-bit physical address. This entry also contains the byte count of the DMA transfer.

dscr_cmd1 **Offset = 0x04**

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
SUPTR				DUPTR				▨	▨	BC																					

Bits	Name	Description
31:28	SUPTR	Upper Source Pointer. Contains bits [35:32] of the 36-bit starting physical address of the source data for the DMA transfer. Bits [35:32] must not change during the transfer.
27:24	DUPTR	Upper Destination Pointer. Contains bits [35:32] of the 36-bit starting physical address of the destination location for the DMA transfer. Bits [35:32] must not change during the transfer.
23:22	—	Reserved.
21:0	BC	Byte Count. Contains the number of bytes to be transferred from source to destination for this descriptor.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.1.3 Source 0

This entry in the descriptor contains the lower 32 bits of the starting physical address of the source data for the transfer. This address is concatenated with **dscr_cmd1**[SUPTR] to make the 36-bit physical address.

dscr_source0 **Offset = 0x08**

<table>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="32">SPTR</td></tr>
</table>

Bits	Name	Description
31:0	SPTR	Source Pointer. Contains bits [31:0] of the 36-bit starting physical address of the source data for the DMA transfer.

4.3.1.4 Source 1 (1-Dimensional Stride)

This entry in the descriptor contains the block and stride values for the source transfer. For stride transfers, the format of the entry depends on the type of stride. The 1-dimensional stride format is shown below.

dscr_source1 (1-Dimensional Stride) **Offset = 0x0C**

<table>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="2">STS</td><td colspan="2">SAM</td><td colspan="14">SB</td><td colspan="14">SS</td></tr>
</table>

Bits	Name	Description
31:30	STS	Source Transfer Size. Selects the number of datums transferred per system bus grant for a source transfer. 00 1 datum 01 2 datums 10 4 datums 11 8 datums
29:28	SAM	Source Address Movement. For each source access, the address: 00 Increments 01 Decrements. The source pointer must be the fourth byte of a word-aligned address. For example, if the first source word is 0xA1000000, the source pointer should be 0xA1000003. Byte count can be any multiple of the source width. Source transfer size must be 0 (1 datum). 10 Remains static (FIFO). Not valid for stride transfers. 11 Remains static but burstable (burstable FIFO). Not valid for stride transfers. For burstable FIFOs, the static bus controller continues to increment the lower address bits during the burst transfer. The external FIFO must ignore the lower address bits that change during the burst. Each access must be aligned to the total transfer size.
27:14	SB	Source Block Size. Specifies the number of bytes transferred from the source before the stride value is added/subtracted to/from the source address.
13:0	SS	Source Stride. Specifies the stride of the transfer and is added/subtracted to/from the source address after the block size (SB) has been transferred. If SAM is programmed to increment, the stride is added to the source address. If SAM is programmed to decrement, the stride is subtracted from the source address. Clear SS to disable source stride.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.1.5 Source 1 (2-Dimensional Stride)

This entry in the descriptor contains the block and stride values for the source transfer. For stride transfers, the format of the entry depends on the type of stride. The 2-dimensional stride format is shown below.

dscr_source1 (2-Dimensional Stride) **Offset = 0x0C**

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
	STS		SAM		SB							SS										SSC										

Bits	Name	Description
31:30	STS	Source Transfer Size. Selects the number of datums transferred per system bus grant for a source transfer. 00 1 datum 01 2 datums 10 4 datums 11 8 datums
29:28	SAM	Source Address Movement. For each source access, the address: 00 Increments 01 Reserved 10 Remains static (FIFO). Not valid for stride transfers. 11 Remains static but burstable (burstable FIFO). Not valid for stride transfers. For burstable FIFOs, the static bus controller continues to increment the lower address bits during the burst transfer. The external FIFO must ignore the lower address bits that change during the burst. Each access must be aligned to the total transfer size.
27:21	SB	Source Block Size. Specifies the number of bytes transferred from the source before the stride value is added/subtracted to/from the source address.
20:11	SS	Source Stride. Specifies the transfer stride in bytes. SS is added/subtracted to/from the source address after the block size (SB) has been transferred. If SAM is programmed to increment, the stride is added to the source address. If SAM is programmed to decrement, the stride is subtracted from the source address. Clear SS to disable source stride.
10:0	SSC	Source Stride Count. The stride count is the number of strides to take before incrementing the start address with the block size (SB): start_addr = start_addr + block_size.

4.3.1.6 Destination 0

This entry in the descriptor contains the lower 32 bits of the starting physical address of the destination location for the DMA transfer. This address is concatenated with **dscr_cmd1**[DUPTR] to make the 36-bit physical address.

dscr_dest0 **Offset = 0x10**

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
	DPTR																															

Bits	Name	Description
31:0	DPTR	Destination Pointer. This descriptor entry contains the lower 32 bits of the starting physical address of the destination location for the DMA transfer.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.1.7 Destination 1 (1-Dimensional Stride)

This entry in the descriptor contains the block and stride values for the destination transfer. For stride transfers, the format of the entry depends on the type of stride. The 1-dimensional stride format is shown below.

dscr_dest1 (1-Dimensional Stride) **Offset = 0x14**

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
	DTS		DAM		DB														DS													

Bits	Name	Description
31:30	DTS	Destination Transfer Size. Selects the number of datums transferred per system bus grant for a destination transfer. 00 1 datum 01 2 datums 10 4 datums 11 8 datums
29:28	DAM	Destination Address Movement. For each destination access, the address: 00 Increments 01 Decrements. The source pointer must be the fourth byte of a word-aligned address. For example, if the first source word is 0xA1000000, the source pointer should be 0xA1000003. Byte count can be any multiple of the source width. Source transfer size must be 0 (1 datum). 10 Remains static (FIFO). Not valid for stride transfers. 11 Remains static but burstable (burstable FIFO). Not valid for stride transfers. For burstable FIFOs, the static bus controller continues to increment the lower address bits during the burst transfer. The external FIFO must ignore the lower address bits that change during the burst. Each access must be aligned to the total transfer size.
27:14	DB	Destination Block Size. Specifies the number of bytes transferred to the destination before the stride value is added/subtracted to/from the destination address.
13:0	DS	Destination Stride. Specifies the stride of the transfer and is added/subtracted to/from the destination address after the block size (DB) has been transferred. If DAM is programmed to increment, the stride is added to the destination address. If the DAM is programmed to decrement, the stride is subtracted from the destination address. Clear DS to disable destination stride.

4.3.1.8 Destination Block/Stride (2 Dimensional)

This entry in the descriptor contains the block and stride values for the destination transfer. For stride transfers, the format of the entry depends on the type of stride. The 2-dimensional stride format is shown below.

dscr_dest1 (2-Dimensional Stride) **Offset = 0x14**

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
	DTS		DAM		DB							DS										DSC										

Bits	Name	Description
31:30	DTS	Destination Transfer Size. Selects the number of datums transferred per system bus grant for a destination transfer. 00 1 datum 01 2 datums 10 4 datums 11 8 datums

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description
29:28	DAM	Destination Address Movement. For each source access, the address: 00 Increments 01 Reserved 10 Remains static (FIFO). Not valid for stride transfers. 11 Remains static but burstable (burstable FIFO). Not valid for stride transfers. For burstable FIFOs, the static bus controller continues to increment the lower address bits during the burst transfer. The external FIFO must ignore the lower address bits that change during the burst. Each access must be aligned to the total transfer size.
27:21	DB	Destination Block Size. Specifies the number of bytes transferred from the source before the stride value is added/subtracted to/from the source address.
20:11	DS	Destination Stride. Specifies the stride of the transfer and is added/subtracted to/from the destination address after the block size (DB) has been transferred. If DAM is programmed to increment, the stride is added to the destination address. If DAM is programmed to decrement, the stride is subtracted from the destination address. Clear DS to disable destination stride.
10:0	DSC	Destination Stride Count. The stride count is the number of strides to take before incrementing the start address with the block size (DB): start_addr = start_addr + block_size.

4.3.1.9 Status

This entry in the descriptor contains optional user-defined status information for the descriptor.

dscr_stat **Offset = 0x18**

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

STAT

Bits	Name	Description
31:0	STAT	User-defined Status. The DDMA controller does not update this field.

4.3.1.10 Next Descriptor Pointer

This entry in the descriptor contains the physical address of the next descriptor to be fetched.

dscr_nxtptr **Offset = 0x1C**

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

NPTR

Bits	Name	Description
31:27	—	Reserved.
26:0	NPTR	Next Descriptor Pointer. Contains the physical address[31:5] of the next descriptor to be fetched.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.2 Compare and Branch DDMA Descriptor Structure

The compare and branch DDMA descriptor is used to evaluate the first word of the source data to determine if a branch should be taken. The compare and branch descriptor structure is shown below.

Compare and Branch DDMA Descriptor

	3	0
0x0	Command (**dscr_cmd0**)	
0x4	Branch Pointer (**dscr_branchptr**)	
0x8	Source Pointer (**dscr_source0**)	
0xC	Reserved	
0x10	Compare Data (**dscr_compdata**)	
0x14	Data Mask (**dscr_mask**)	
0x18	Status (**dscr_stat**)	
0x1C	Next Descriptor Pointer (**dscr_nxtptr**)	

The DDMA controller makes the compare and branch evaluation as follows:

```
if((*source_data & dscr_mask[DMASK]) == dscr_compdata[CDATA])
then
next_descriptor = dscr_branchptr[BPTR]
else
next_descriptor = dscr_nxtptr[NPTR]
```

The fields within the compare and branch DDMA descriptor structure are described below.

4.3.2.1 Command

This entry in the descriptor contains command information for the transfer.

dscr_cmd0 **Offset = 0x00**

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
V		SID										SW		DW		ARB	DT		SN				IE				SP		CV	ST	

Bits	Name	Description
31	V	Valid. 0 Descriptor is not valid. 1 Descriptor is valid.
30	—	Reserved.
29:25	SID	Source ID. Selects the device ID that determines the DMA request associated with the channel for source transfers. Refer to Table 4-1 on page 118 for device ID encodings.
24:20	—	Reserved.
19:18	SW	Source Width. Selects the width of the transfer on the system bus for source accesses. Software defines the source datum size. 00 Byte 01 Halfword 10 Word 11 Reserved
17:16	DW	Destination Width. For compare and branch descriptors, the datum size must be programmed for word width. 10 Word All other values reserved.

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description
15	ARB	Arbitration Priority Pool. Selects the channel priority. See Section 4.1.1 "Channel Priority and Arbitration" on page 116. 0 Low priority pool 1 High priority pool
14:13	DT	Descriptor Type. Selects the descriptor type. 00 Standard descriptor (for source-to-destination transfers) 01 Literal-write descriptor 10 Compare and branch descriptor 11 Reserved
12	SN	Source Non-coherent. Transfers from an internal FIFO on the peripheral bus must be non-coherent. 0 Mark source data as coherent. 1 Mark source data as non-coherent.
11:9	—	Reserved.
8	IE	Interrupt Enable. 0 Do not set **ddma*n*_irq**[IN] upon descriptor completion. 1 Set **ddma*n*_irq**[IN] upon descriptor completion.
7:5	—	Reserved.
4	SP	Copy Descriptor Pointer To Status Pointer. 0 The value written to **ddma*n*_statptr** is determined by the ST field**.** 1 Let **ddma*n*_statptr** = **ddma*n*_desptr**. The current descriptor status (**dscr_stat**) is written to **dscr_cmd0** upon descriptor completion**.**
3	—	Reserved.
2	CV	Clear Valid Bit. 0 Do not clear **dscr_cmd0**[V] upon descriptor completion**.** 1 Clear **dscr_cmd0**[V] upon descriptor completion**.**
1:0	ST	Status Instruction. Determines the information to report when the descriptor completes. The memory content pointed to by **ddma*n*_statptr** is changed as follows: 00 No change 01 Write the current descriptor status (**dscr_stat**). 10 Write the **dscr_cmd0** of the current descriptor with the valid bit cleared. 11 Write the remaining byte count.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.2.2 Branch Pointer

This entry in the descriptor contains the upper 4 bits of the starting physical address of the source data used in the comparison. These bits are concatenated with **dscr_source0**[SPTR] to make the 36-bit physical address. This entry also contains the pointer to the next descriptor executed when the comparison evaluates true.

dscr_branchptr **Offset = 0x04**

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
	SUPTR				▨	BPTR																										

Bits	Name	Description
31:28	SUPTR	Upper Source Pointer. Contains the upper 4 bits of the 36-bit starting physical address of the source data for the DMA transfer. The upper 4 bits of the 36-bit address must not change during the DMA transfer.
27	—	Reserved.
26:0	BPTR	Branch Pointer. Address bits [31:5] of the next descriptor fetched if the compare evaluates true.

4.3.2.3 Source Pointer

See Section 4.3.1.3 "Source 0" on page 130 for register and bit descriptions.

4.3.2.4 Compare Data

This entry in the descriptor contains the data used in the comparison to determine if a branch to the descriptor pointed to by BPTR (**dscr_branchpt**r[26:0]) should be made. The data pointed to by the source pointer is masked with DMASK and then compared to CDATA. If equal, the DDMA controller begins executing the descriptor pointed to by BPTR; otherwise, execution continues with the descriptor pointed to by NPTR (**dscr_nxtptr**[26:0]).

dscr_compdata **Offset = 0x10**

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
	CDATA																															

Bits	Name	Description
31:0	CDATA	Compare Data. This descriptor entry contains the data used in the compare.

4.3.2.5 Data Mask

This entry in the descriptor contains the mask used in the comparison to determine if a branch to the descriptor pointed to by BPTR (**dscr_branchpt**r[26:0]) should be made. A logical AND of the data pointed to by the source pointer and DMASK is compared to CDATA. If equal, the DDMA controller begins executing the descriptor pointed to by BPTR; otherwise, execution continues with the descriptor pointed to by NPTR (**dscr_nxtptr**[26:0]).

dscr_mask **Offset = 0x14**

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
	DMASK																															

Bits	Name	Description
31:0	DMASK	Data Mask. This descriptor entry contains the mask used in the compare.

4.3.2.6 Status

See Section 4.3.1.9 "Status" on page 133 for Status register and bit descriptions.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.2.7 Next Descriptor Pointer

This entry in the descriptor contains the physical address of the next descriptor to be fetched, as well as the branch/compare byte count for the current descriptor.

dscr_nxtptr **Offset = 0x1C**

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
▨		BBC		▨	NPTR																										

Bits	Name	Description
31:30	—	Reserved.
29:28	BBC	Byte Count for Compare and Branch Descriptors. Determines the number of data bytes used in the comparison. Applies to current descriptor. 00 1 byte 01 2 bytes 10 3 bytes 11 4 bytes
27	—	Reserved.
26:0	NPTR	Next Descriptor Pointer. Contains the physical address[31:5] of the next descriptor to be fetched.

4.3.3 Literal Write DDMA Descriptor Structure

The literal write DDMA descriptor structure is shown below.

Literal Write DDMA Descriptor

	3	0
0x0	Command 0 (**dscr_cmd0**)	
0x4	Command 1 (**dscr_cmd1**)	
0x8	Source Data Low (**dscr_source0**)	
0xC	Source Data High (**dscr_source1**)	
0x10	Destination Pointer (**dscr_dest0**)	
0x14	Destination Stride/Block (**dscr_dest1**)	
0x18	Status (**dscr_stat**)	
0x1C	Next Descriptor Pointer (**dscr_nxtptr**)	

The fields within the literal write DDMA descriptor structure are described below.

4.3.3.1 Command 0 (See Page 128.)

See Section 4.3.1.1 "Command 0" on page 128 for details.

For literal writes, software must program the source width to be one word (**dscr_cmd0**[SW] = 0b10).

4.3.3.2 Command 1 (See Page 129.)

See Section 4.3.1.2 "Command 1" on page 129 for details.

www.DataSheet4U.com

www.DataSheet4U.com

4.3.3.3 Source 0 (Source Data 0)

This entry in the descriptor contains the first (or only) word of data used by a literal-write descriptor.

dscr_source0 **Offset = 0x08**

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
	SDATA0																															

Bits	Name	Description
31:0	SDATA0	Source Data 0. Contains the first word of data used by a literal-write descriptor.

4.3.3.4 Source 1 (Source Data 1)

This entry in the descriptor contains the second word of data used by a literal-write descriptor.

dscr_source1 **Offset = 0x0C**

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
	SDATA1																															

Bits	Name	Description
31:0	SDATA1	Source Data 1. Contains the second word of data used by a literal-write descriptor.

4.3.3.5 Destination Pointer

See Section 4.3.1.6 "Destination 0" on page 131 for details.

4.3.3.6 Destination Block/Stride (1 Dimensional)

See Section 4.3.1.7 "Destination 1 (1-Dimensional Stride)" on page 132 for details.

4.3.3.7 Destination Block/Stride (2 Dimensional)

See Section 4.3.1.8 "Destination Block/Stride (2 Dimensional)" on page 132 for details.

4.3.3.8 Status

See Section 4.3.1.9 "Status" on page 133 for details.

4.3.3.9 Next Descriptor Pointer

See Section 4.3.1.10 "Next Descriptor Pointer" on page 133 for details.

www.DataSheet4U.com

www.DataSheet4U.com

4.4 Using GPIO as External DMA Requests (DMA_REQ*n*)

GPIO[4] and GPIO[12] can be programmed to act as external DMA request signals. When configured for this system function, the GPIO signals are defined as follows:

- GPIO[4] is DMA_REQ0.
- GPIO[12] is DMA_REQ1.

To use GPIO[4] or GPIO[12] as an external DMA request (DMA_REQ*n*), follow these steps:

1) For GPIO[4] and GPIO[12]: Tristate the GPIO to make it an input through the **sys_triout** register. In addition, GPIO[12] requires configuration in **sys_pinfunc**[DMA]. (GPIO[4] does not require additional configuration in **sys_pinfunc**.) See Section 10.3 "Primary General Purpose I/O and Pin Functionality" on page 381 for more information.
2) Set **ddma*n*_cfg**[SYNC] to force all transfers to complete before the descriptor completes.
3) Program the descriptor to point to the external device data port. The static bus controller must be configured correctly to recognize this address.
4) Program the descriptor to match the direction of transfer and peripheral attributes.

The DMA_REQ*n* signal is used to signal the DDMA controller when an external FIFO or memory location can receive or transmit transfer size of data as programmed by **dscr_source1**[STS] or **dscr_dest1**[DTS]. Once a transfer begins, it continues until completion; therefore, the external device must assert DMA_REQ*n* only when it can satisfy an entire transfer size of data.

4.5 DDMA Controller Programming Considerations

4.5.1 General Configuration

As part of the general configuration of the DDMA controller, software must program **ddma_config**[AH, AL] to select the arbitration policies for the high and low priority arbitration pools. The interrupt-enable register (**ddma_inten**) and request throttle register (**ddma_throttle**) can be programmed as needed as part of the individual channel configuration.

4.5.2 Channel Configuration

Before issuing a DMA request, software must prepare the descriptor(s) for each channel to be used in the application:

- Write new descriptor(s) into memory. This includes all fields described in Section 4.3 "DDMA Descriptors" on page 127.

After building the descriptor(s), software initiates a DMA request as follows:

1) Set **dscr_cmd0**[V] for the initial descriptor. Once the valid bit is set, the descriptor cannot be modified by software until **dscr_cmd0**[V] has been cleared by the DDMA controller.
2) A write to the doorbell register (**ddma*n*_dbell**) causes the DDMA controller to begin fetching descriptors. Software can then wait for a DDMA controller interrupt or poll **dscr_cmd0**[V] before processing completed descriptors.

The DDMA controller services a channel as follows:

1) The controller waits until software writes to **ddma*n*_dbell**.
2) The controller begins processing descriptors:
   - — When a valid descriptor (**dscr_cmd0**[V] = 1) is encountered, the descriptor is performed until completion, updating the various status registers based on the control flags. The controller then updates the descriptor pointer to the next descriptor (or branch, for a branch descriptor type).
   - — When an invalid/completed descriptor (**dscr_cmd0**[V] = 0) is encountered, the controller waits for software to write to the channel's **ddma*n*_dbell**, then re-fetches the descriptor. There is no distinction between an invalid and completed descriptor.

When an interrupt is seen by software, the following sequence must be followed:

1) Clear the interrupt bit in the appropriate channel.
2) Process the descriptors for the channel.

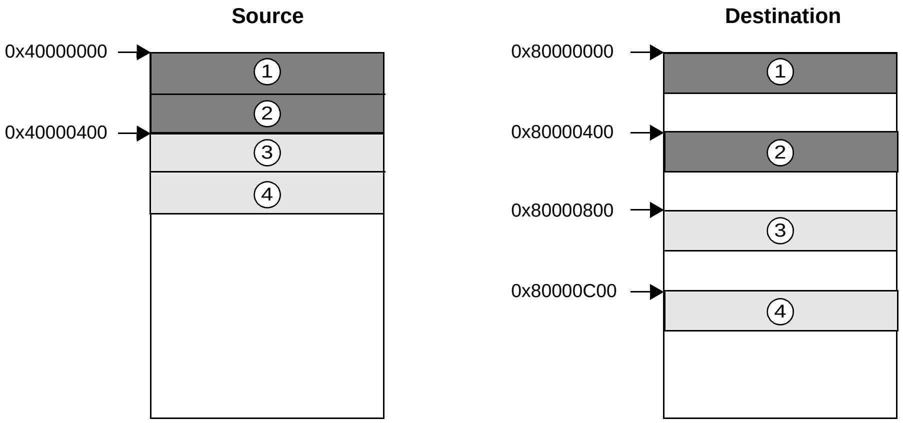
www.DataSheet4U.com

www.DataSheet4U.com

4.5.3 Stride Transfers

Figure 4-4 shows an example of a 1-dimensional stride transfer.

Command Fields	
Programming	**Description**
dscr_cmd0[MEM] = 1	Memory-to-Memory Transfer
dscr_cmd0[DT] = 00	Source-to-Destination Descriptor Type
dscr_cmd0[SM] = 0	1-Dimensional Stride
dscr_cmd1[BC] = 0x800	2 KBytes Total Transfer
dscr_cmd1[SUPTR] = 0x0	Source Address[35:32]
dscr_cmd1[DUPTR] = 0x0	Destination Address[35:32]



Source Fields		Destination Fields	
Programming	**Description**	**Programming**	**Description**
dscr_source0[SPTR] = 0x400000000	Source Address[31:0]	**dscr_dest0**[DPTR] = 0x800000000	Destination Address[31:0]
dscr_source1[SAM] = 00	Increment Source Address	**dscr_dest1**[DAM] = 00	Increment Destination Address
dscr_source1[SB] = 0	Disabled	**dscr_dest1**[DB] = 0x200	512 Byte Destination Block
dscr_source1[SS] = 0	Disable Source Stride	**dscr_dest1**[DS] = 0x400	1 KByte Destination Stride

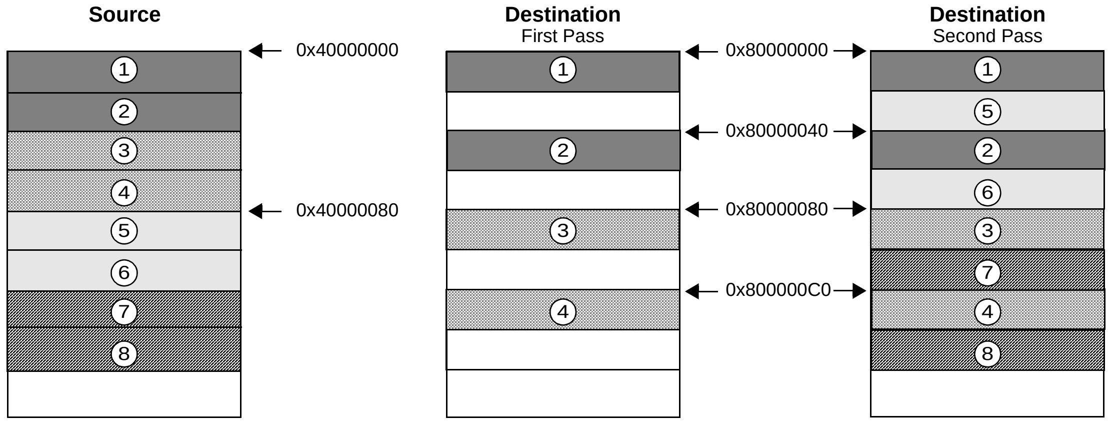
Figure 4-4. Example of 1-Dimensional Stride Transfer (Scatter)

www.DataSheet4U.com

www.DataSheet4U.com

Figure 4-5 shows an example of a 2-dimensional stride transfer.

Command Fields	
Programming	**Description**
dscr_cmd0[MEM] = 1	Memory-to-Memory Transfer
dscr_cmd0[DT] = 00	Source-to-Destination Descriptor Type
dscr_cmd0[SM] = 1	2-Dimensional Stride
dscr_cmd1[BC] = 0x100	256 Bytes Total Transfer
dscr_cmd1[SUPTR] = 0x0	Source Address[35:32]
dscr_cmd1[DUPTR] = 0x0	Destination Address[35:32]



Source Fields		Destination Fields	
Programming	Description	Programming	Description
dscr_source0[SPTR] = 0x400000000	Source Address[31:0]	**dscr_dest0**[DPTR] = 0x800000000	Destination Address[31:0]
dscr_source1[SAM] = 00	Increment Source Address	**dscr_dest1**[DAM] = 00	Increment Destination Address
dscr_source1[SB] = 0	Disabled	**dscr_dest1**[DB] = 0x20	32-Byte Destination Block
dscr_source1[SS] = 0	Disable Source Stride	**dscr_dest1**[DS] = 0x40	64-Byte Destination Stride
dscr_source1[SSC] = 0	Disabled	**dscr_dest1**[DSC] = 4	4 Destination Stride Iterations

Figure 4-5. Example of 2-Dimensional Stride Transfer

4.6 Shutdown

The DDMA controller can be shutdown by clearing all channel enable bits (**ddma*n*_cfg**[EN]).

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

5 Interrupt Controller

There are two interrupt controllers in the Au1210™ and Au1250™ processors. Each interrupt controller supports 32 interrupt sources. Interrupts can generate a signal to bring the processor out of an IDLE0 or IDLE1 state and generate a CPU interrupt.

Each interrupt controller has two outputs referred to as requests 0 and 1. Each of these outputs is connected to the CPU core; see Section 2.6.2 Interrupt Architecture" on page 34 for additional information. Table 5-1 shows the interrupt controller connections to the CPU.

Table 5-1. Interrupt Controller Connections to the CPU

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

5.1 Interrupt Controller Sources

Table 5-2 shows the mapping of interrupt sources for interrupt controller 0 and 1.

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

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

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

Table 5-2. Interrupt Sources

Controller	Interrupt Number	Source	Type
0	0	UART0	High Level
0	1	Software Counter Match	High Level
0	2	Secure Digital Controller	High Level
0	3	DDMA Controller	High Level
0	4	MAE Back End	High Level
0	5	GPIO[200]	System Dependent
0	6	GPIO[201]	System Dependent
0	7	GPIO[202]	System Dependent
0	8	UART1	High Level

www.DataSheet4U.com

www.DataSheet4U.com

Table 5-2. Interrupt Sources (Continued)

Controller	Interrupt Number	Source	Type
0	9	MAE Font End	High Level
0	10	PSC0	High Level
0	11	PSC1	High Level
0	12	AES Cryptography Engine (Au1250 only)	High Level
0	13	Camera Interface Module	High Level
0	14	TOY (tick)	Rising Edge
0	15	TOY Match 0	Rising Edge
0	16	TOY Match 1	Rising Edge
0	17	TOY Match 2	Rising Edge
0	18	RTC (tick)	Rising Edge
0	19	RTC Match 0	Rising Edge
0	20	RTC Match 1	Rising Edge
0	21	RTC Match 2	Rising Edge
0	22	GPIO[203]	System Dependent
0	23	NAND Controller	Rising Edge
0	24	GPIO[204]	System Dependent
0	25	GPIO[205]	System Dependent
0	26	GPIO[206]	System Dependent
0	27	GPIO[207]	System Dependent
0	28	Logical OR of GPIO[208:215] (See Section 9.8.1.4 Interrupt Enable Register" on page 364.)	System Dependent
0	29	USB Controller	High Level
0	30	LCD Controller	High Level
0	31	MAE Done Logical AND of Front End and Back End	High Level
1	n = 0, 1, …, 31	GPIO[n]	System Dependent

www.DataSheet4U.com

www.DataSheet4U.com

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

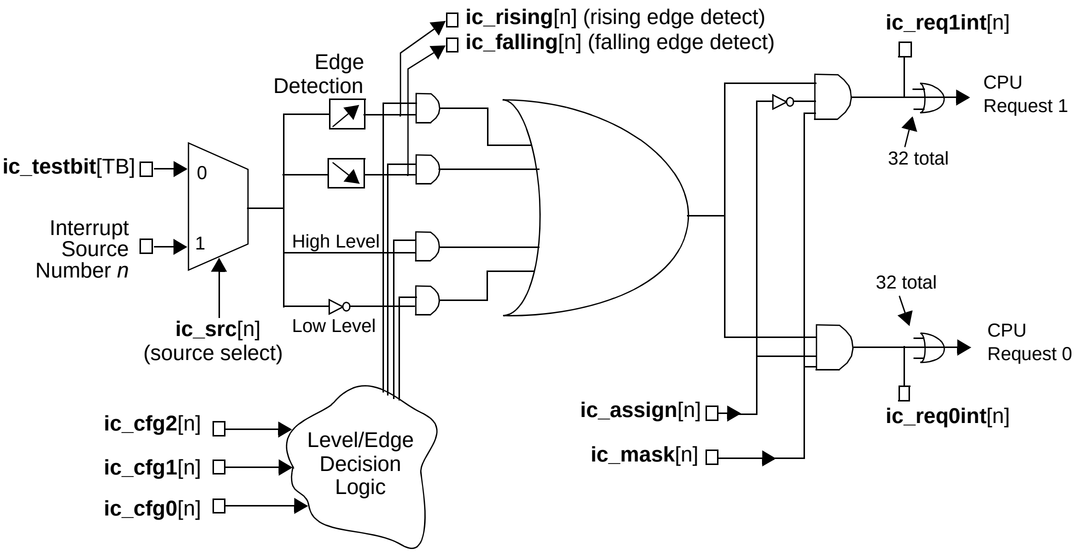


Figure 5-1. Interrupt Controller Logic

5.2 Interrupt Controller Registers

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

Table 5-3. Interrupt Controller Base Addresses

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

Each interrupt controller has an identical set of registers that controls its set of interrupts. Table 5-4 on page 146 shows the interrupt controller registers and their associated offsets. Certain offsets are shared but address different internal registers depending on whether the access is a read or a write. The register description details the functionality of the register. Bit *n* of a particular register is associated with interrupt *n* of the corresponding controller.

www.DataSheet4U.com

www.DataSheet4U.com

Table 5-4. Interrupt Controller Registers

Offset	Register	Type	Description	Default
0x0040	ic_cfg0rd	R	Configuration 0 register Configuration 1 register Configuration 2 register The combined field consisting of **ic_cfg2**[*n*], **ic_cfg1**[*n*], and **ic_cfg0**[*n*] specifies the trigger characteristics for interrupt *n* as shown in Table 5-5.	UNPRED
0x0040	ic_cfg0set	W		
0x0044	ic_cfg0clr	W		
0x0048	ic_cfg1rd	R		UNPRED
0x0048	ic_cfg1set	W		
0x004C	ic_cfg1clr	W		
0x0050	ic_cfg2rd	R		UNPRED
0x0050	ic_cfg2set	W		
0x0054	ic_cfg2clr	W		
0x0054	ic_req0int	R	Shows active interrupts on request 0. Used by software to determine the interrupt source.	0x0000 0000
0x0058	ic_srcrd	R	Selects the source of the interrupt between a test bit and the designated source. 0 Use the test bit (**ic_testbit**[TB]) as interrupt source. 1 Use the peripheral interrupt (for controller 0) or GPIO signal (for controller 1) as interrupt source.	UNPRED
0x0058	ic_srcset	W		
0x005C	ic_srcclr	W		
0x005C	ic_req1int	R	Shows active interrupts on request 1. Used by software to determine the interrupt source.	0x0000 0000
0x0060	ic_assignrd	R	Assigns the interrupt to one of the CPU requests. 0 Assign interrupt to request 1. 1 Assign interrupt to request 0.	UNPRED
0x0060	ic_assignset	W		
0x0064	ic_assignclr	W		
0x0068	ic_wakerd	R	Controls whether the interrupt can cause a wakeup from IDLE0 or IDLE1. 0 No wakeup from Idle 1 Interrupt causes wakeup from Idle. The associated interrupt must still be enabled to wake from Idle. Setting this bit without setting the associated mask bit may cause unpredictable IDLE power consumption.	0x0000 0000
0x0068	ic_wakeset	W		
0x006C	ic_wakeclr	W		
0x0070	ic_maskrd	R	Interrupt enable. 0 Disable the interrupt. 1 Enable the interrupt.	0x0000 0000
0x0070	ic_maskset	W		
0x0074	ic_maskclr	W		
0x0078	ic_risingrd	R	Designates active rising edge interrupts. If an interrupt is generated off of a rising edge, the associated rising edge detection bit must be cleared after detection.	UNPRED
0x0078	ic_risingclr	W		
0x007C	ic_fallingrd	R	Designates active falling edge interrupts. If an interrupt is generated off of a falling edge, the associated falling edge detection bit must be cleared after detection.	UNPRED
0x007C	ic_fallingclr	W		
0x0080	ic_testbit	R/W	This is a single bit register that is mapped to all the source select inputs for testing purposes.	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

5.2.1 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	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
	FUNC[31:0]																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	Read/Write	Default
31:0	FUNC[n]	The function of each register is given in Table 5-4 on page 146. FUNC[n] controls the functionality of interrupt *n* in the corresponding controller.	*rd - read only *set - write only *clr - write only See the following explanation.	See Table 5-4 on page 146.

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

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

- *rd registers (read only) return the current value of the register.
- *set registers (write only) set all bits that are written 1. Writing a value of 0 has no effect on the corresponding bit.
- *clr registers (write only) clear all bits that are written 1. Writing a value of 0 has no effect on the corresponding bit.

The three configuration registers **ic_cfg2**[*n*], **ic_cfg1**[*n*], **and ic_cfg0**[*n*] uniquely control interrupt *n*'s functionality as shown in Table 5-5.

Table 5-5. Interrupt Configuration Register Function

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

www.DataSheet4U.com

www.DataSheet4U.com

5.2.2 Test-bit Register

This register contains the test bit that can be used as a test source for each interrupt. Figure 5-1 on page 145 shows how the test bit can be selected as an optional interrupt source.

Offset = 0x0080

ic_testbit

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
																																TB
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

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

5.3 Hardware Considerations

To use a GPIO as an interrupt source, the pin functionality in the **sys_pinfunc** register must be for a GPIO (see Section 10.3.1.1 Pin Function" on page 381). In addition, the GPIO must be enabled as an input. See Section 10.3 Primary General Purpose I/O and Pin Functionality" on page 381 for more information.

5.4 Programming Considerations

The Au1210 and Au1250 processors are designed to simplify interrupt management by removing the need for a semaphore to control access to the registers. There is no need to perform a read-modify-write sequence because separate registers are provided for setting and clearing bits. In this way software can freely manipulate the interrupts associated with that application.

If using edge-triggered interrupts, it is important to clear the associated edge detection bit to allow subsequent interrupts to be detected.

Interrupts are programmed as follows: (The **set**, **clr**, and **rd** portion of the register name has been omitted.)

1) Identify the interrupt number, *n,* with the associated peripheral or GPIO.

2) Use **ic_src**[*n*] to assign the interrupt to the associated peripheral/GPIO (or the test bit can be used if testing the interrupt).

3) Set the **ic_cfg2**[*n*], **ic_cfg1**[*n*] and **ic_cfg0**[*n*] bits to the correct configuration for the corresponding interrupt (edge, level, polarity).

4) Assign the interrupt to a CPU request using **ic_assign**[*n*].

5) Use **ic_wake**[*n*] to assign the interrupt to wake the processor from Idle if necessary, or clear this register bit to keep the interrupt from waking the processor from Idle.

6) If the interrupt is an edge-triggered interrupt, clear the edge detect register (**ic_risingclr** or **ic_fallingclr**) before enabling.

7) Finally, enable the interrupt through **ic_mask**[*n*].

When servicing an interrupt, follow these steps:

1) Read **ic_req0int** and **ic_req1int** to determine the interrupt number *n*.

2) Use **ic_fallingrd** and **ic_risingrd** to determine if the interrupt is edge-triggered. If so, use **ic_fallingclr**[*n*] or **ic_risingclr**[*n*] to clear the edge detection circuitry.

3) If the interrupt is to be disabled write **ic_maskclr**[*n*].

4) Service the interrupt.

www.DataSheet4U.com

www.DataSheet4U.com

6 Media Acceleration Engine (MAE)

The media acceleration engine (MAE) supports inverse quantization (IQ), inverse discrete cosine transform (IDCT), motion compensation, image scaling, image post filtering, and color space conversion. MAE operates as two independent parts, the front end and the back end. The front end includes the IQ, IDCT, and motion compensation. The back end includes the scaler/filter and color space conversion. The interface between the front end and back end is shared memory. It is possible to use the back end in a standalone mode.

6.1 MAE Features

6.1.1 MAE Front End Features

- Supports multiple codec formats including:
  — MPEG2 Main Profile @ Main Level
  — MPEG4 Advanced Simple Profile @ Level 5
  — WMV9 Medium Profile @ Medium Level (no interlaced support)
- Designed to support full size decode (720x480-30 fps)
  — Au1250 supports up to full D1: 720x480 NTSC or 720x576 PAL
  — Au1210 supports up to Wide-CIF: 480x288 only
- Flexible inverse quantization (IQ) implementation
- Inverse discrete cosine transform (IDCT)
- Intraframe, predicted, and bidirectional motion compensation
- Support for 1, 2, and 4 motion vectors (16x16, 16x8, and 8x8 blocks)
- Support for interlaced tools (field prediction)
- Full pel, half pel, and quarter pel motion compensation
- WMV9 overlap smoothing and in-loop deblocking filters

6.1.2 MAE Back End Features

- Arbitrary scaler:
  — Independent horizontal and vertical filtering
  — 4 tap programmable filter in each direction
  — 32 programmable coefficients per tap
  — Reduced bandwidth operating mode
  — 1/32 decimation and 4x interpolation
  — Minimum size: 64x16; maximum size: 1024x1024
- Color space converter:
  — With the scaler can support 4:4:4, 4:2:2, 4:2:0, and 4:1:1 input
  — Output modes: aRGB32, RGB565, RGB555
  — Programmable coefficient data
  — Symmetrical design

6.1.3 MAE System Features

- Front end and back end can operate independently
  — Non-block based codecs can utilize the MAE back end

www.DataSheet4U.com

www.DataSheet4U.com

6.2 MAE Front End

6.2.1 Overview

Figure 6-1 shows the general flow of data through the front end of MAE. Items surrounded by dashed lines represent data that is stored in external SDRAM.

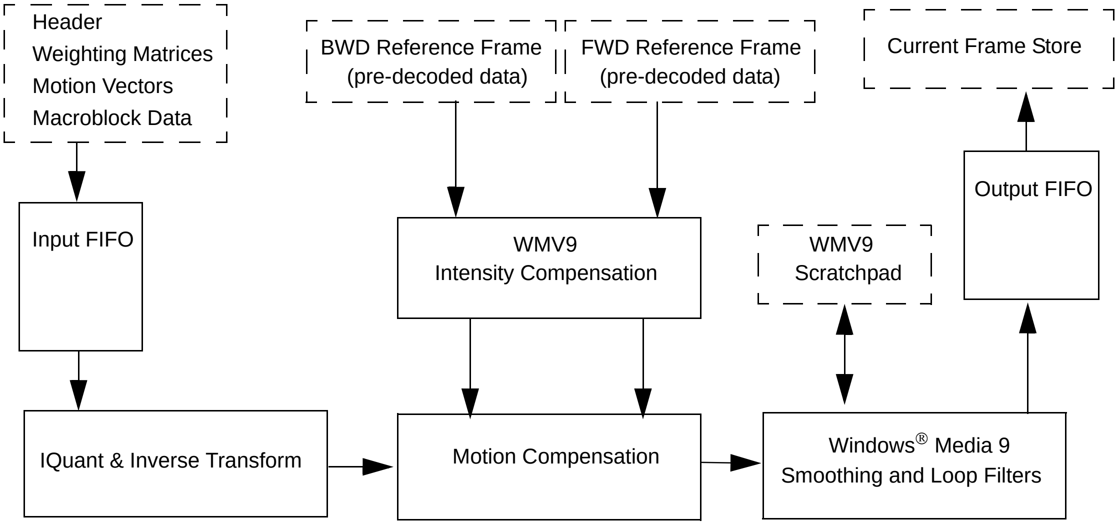

Header
Weighting Matrices
Motion Vectors
Macroblock Data
BWD Reference Frame (pre-decoded data)
FWD Reference Frame (pre-decoded data)
Current Frame Store
Input FIFO
WMV9 Intensity Compensation
WMV9 Scratchpad
Output FIFO
IQuant & Inverse Transform
Motion Compensation
Windows® Media 9 Smoothing and Loop Filters

Figure 6-1. Front End Block Diagram

Software must set registers in the MAE that describe the location of the various external frames (bwd reference, fwd reference, current). These values must be updated for each processed frame. Software must then pack the header information, weighting matrices, motion vectors, and macroblock data in the proper format for processing by the MAE. Software then sets up descriptors for this information and passes these to the MAE DMA.

The front end of the MAE fetches the data and performs IQ. IQ processing uses a formula that is a superset of that used by all the standards the MAE supports. The various parameters used by this formula are described in Section 6.2.4 "Inverse Quantization" on page 167.

The output of the IQ is then passed to the inverse transformation unit. This unit examines the codec style bit and transform size bit, and performs the appropriate transform. Following inverse transformation, the data is passed to the motion compensation block.

The motion compensation block first performs an address calculation based on the motion vectors and macroblock position, and then uses the address to fetch the part of the reference frame(s) needed for the current macroblock. The motion compensation block then performs the appropriate interpolation based on the motion vectors and motion compensation precision to calculate the reference pixels for the current macroblock. It then adds these reference pixels to the residual calculated by the inverse transform.

For Windows® Media 9 (WMV9) content, the macroblock is passed to the smoothing and loop filter block. This block performs in-loop smoothing and deblocking as defined by the WMV9 specification. Finally, the data is sent to the output FIFO where the MAE DMA writes it to the current frame.

www.DataSheet4U.com

www.DataSheet4U.com

6.2.2 MAE Front End Registers

MAE front end operation is configured and controlled by a register block whose physical base address is shown in Table 6-1.

Table 6-1. MAE Front End Base Address

Name	Physical Base Address	KSEG1 Base Address
maefe_base	0x0 1401 2000	0xB401 2000

The register block consists of the registers shown in Table 6-2.

Table 6-2. MAE Front End Registers

Offset (Note1)	Register Name	Description
0x0000	maefe_config	MAE Configuration Register
0x0004	maefe_cury	MAE Front End Current Frame Y Register
0x0008	maefe_frefy	MAE Front End Forward Reference Frame Y Register
0x000C	maefe_brefy	MAE Front End Backward Reference Frame Y Register
0x0010	maefe_curcb	MAE Front End Current Frame Cb Register
0x0014	maefe_frefcb	MAE Front End Forward Reference Frame Cb Register
0x0018	maefe_brefcb	MAE Front End Backward Reference Frame Cb Register
0x001C	maefe_curcr	MAE Front End Current Frame Cr Register
0x0020	maefe_frefcr	MAE Front End Forward Reference Frame Cr Register
0x0024	maefe_brefcr	MAE Front End Backward Reference Frame Cr Register
0x0028	maefe_pictsize	MAE Picture Size Register
0x002C	maefe_intenscomp	MAE Intensity Compensation Register
0x0030-0x0034	Reserved	Reserved, do not write
0x0038	maefe_frefboty	MAE Front End Forward Bottom Field Reference Y Register
0x003C	maefe_frefbotcb	MAE Front End Forward Bottom Field Reference Cb Register
0x0040	maefe_frefbotcr	MAE Front End Forward Bottom Field Reference Cr Register
0x0044	maefe_brefboty	MAE Front End Backward Bottom Field Reference Y Register
0x0048	maefe_brefbotcb	MAE Front End Backward Bottom Field Reference Cb Register
0x004C	maefe_brefbotcr	MAE Front End Backward Bottom Field Reference Cr Register
0x0050	maefe_intstat	MAE Front End Interrupt Status Register
0x0054	maefe_intenable	MAE Front End Interrupt Enable Register
0x0058	maefe_scratchpad	MAE Front End Scratch Pad Address Register
0x005C	maefe_wmv9pquant	MAE WMV9 PQUANT Register
0x1004	maefe_dmadscr	MAE DMA Descriptor Pointer Register
0x1008	maefe_dmadbell	MAE DMA Doorbell Register

Note1. See Table 6-1 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

6.2.2.1 MAE Configuration Register

maefe_config **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
						BC	IQMUL1						PLMB		COD	MIS			LOOP	SMOO	SATIQ											
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:27	--	Reserved.	R	0
26	BC	Block Count. Specifies the number of blocks in a macroblock. 0 6 blocks (4:2:0) per macroblock. 4Y, 1Cb, 1Cr 1 8 blocks (4:2:2) per macroblock. 4Y, 2Cb, 2Cr	R/W	0
25:24	IQMUL1	Inverse Quantization Multiplier 1. This is the iq_mul_1 parameter in the general inverse quantization formula. See Section 6.2.4 "Inverse Quantization" on page 167. The value programmed in these bits is the number of bits by which to shift left.	R/W	0
23:20	-	Reserved.	R	0
19:18	PLMB	Picture Level MBMODE. This is used when codec style is WMV9 and the loop filter is used. Valid values are: 00 INTRA 01 Forward 10 Backward 11 Bidirectional	R/W	0
17	COD	Codec Style. This bit is used to select several overall codec style features such as the type of transform that is performed and the type of filter that is used to calculate non full pel pixels during motion compensation. 0 MPEG 1 Windows Media 9	R/W	0
16	MIS	Mismatch Control. MPEG mismatch sum and correct. 0 No mismatch control processing 1 Enable mismatch control processing	R/W	0
15:14	-	Reserved.	R/W	0
13	LOOP	Windows Media 9 in Loop Deblocking Filter. 0 Do not perform in loop deblocking filter 1 Perform in loop deblocking filter	R/W	0
12	SMOO	Windows Media 9 Overlap Smoothing Filter. 0 Do not perform smoothing filter 1 Perform smoothing filter	R/W	0
11	SATIQ	Perform saturation of value after Inverse Quantization to the range [-2048, 2047]. 0 Do not perform saturation 1 Perform saturation	R/W	0
10:0	--	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

6.2.2.2 MAE Front End Current Frame Y Register

maefe_cury **Offset = 0x0004**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of current Y frame. The front end of MAE writes the current frame to this location. This address should be cache line aligned.	R/W	0

6.2.2.3 MAE Front End Forward Reference Frame Y Register

maefe_frefy **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
	FRAMEPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of forward Y reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.4 MAE Front End Backward Reference Frame Y Register

maefe_brefy **Offset = 0x000C**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of backward Y reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.c

6.2.2.5 MAE Front End Current Frame Cb Register

maefe_curcb **Offset = 0x0010**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of current C_b frame. The front end of MAE writes the current frame to this location. This address should be cache line aligned.	R/W	0

6.2.2.6 MAE Front End Forward Reference Frame C_b Register

maefe_frefcb **Offset = 0x0014**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of forward C_b reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.7 MAE Front End Backward Reference Frame C_b Register

maefe_brefcb **Offset = 0x0018**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of backward C_b reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.8 MAE Front End Current Frame C_r Register

maefe_curcr **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
	FRAMEPTR																															
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of current C_r frame. The front end of MAE writes the current frame to this location. This address should be cache line aligned.	R/W	0

6.2.2.9 MAE Front End Forward Reference Frame C_r Register

maefe_frefcr **Offset = 0x0020**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of forward C_r reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.10 MAE Front End Backward Reference Frame C_r Register

maefe_brefcr **Offset = 0x0024**

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

Bits	Name	Description	Read/Write	Default
31:0	FRAMEPTR	This is the physical address of backward C_r reference frame. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.11 MAE Picture Size Register

maefe_pictsize **Offset = 0x0028**

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

Bits	Name	Description	Read/Write	Default
31:26	—	Reserved.	R/W	0
25:16	HEIGHT	Picture Height. Picture height size in lines. This must be a multiple of 16 lines. This is the height of the luma (Y) frame. Au1210 only: The maximum value is 511. Writes to bit 25 are ignored.	R/W	0
15:10	—	Reserved.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
9:0	LINESIZ	Line Size. Picture line size in pixels. This must be a multiple of 16 pixels. This is the line size of the luma (Y) frame. Au1210 only: The maximum value is 511. Writes to bit 9 are ignored.	R/W	0

6.2.2.12 MAE Intensity Compensation Register

maefe_intenscomp **Offset = 0x002C**

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

Bits	Name	Description	Read/Write	Default
31:13	-	Reserved.	R	0
12	EN	Enable Intensity Compensation. Applies only when codec style is WMV9 (**maefe_config**[COD] = 1). 0 Disable intensity compensation. 1 Enable intensity compensation.	R/W	0
11:6	LUMSHIFT	Used for intensity compensation for remapping luminance pixels in reference frames (described in WMV9 spec, sec. 4.4.8.).	R/W	0
5:0	LUMSCALE	Used for intensity compensation for remapping luminance pixels in reference frames (described in WMV9 spec, sec. 4.4.8.).	R/W	0

6.2.2.13 MAE Front End Forward Bottom Field Reference Y Register

maefe_frefboty **Offset = 0x0038**

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

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of forward Y bottom reference field, used for interlaced frames. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.14 MAE Front End Forward Bottom Field Reference C_b Register

maefe_frefbotcb **Offset = 0x003C**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of forward C_b bottom reference field, used for interlaced frames. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.15 MAE Front End Forward Bottom Field Reference C_r Register

maefe_frefbotcr **Offset = 0x0040**

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
	FIELDPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of forward C_r bottom reference field, used for interlaced frames. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.16 MAE Front End Backward Bottom Field Reference Y Register

maefe_brefboty **Offset = 0x0044**

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
	FIELDPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of backward Y bottom reference field, used for interlaced frames. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.17 MAE Front End Backward Bottom Field Reference C_b Register

maefe_brefbotcb **Offset = 0x0048**

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

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of forward C_b bottom reference field, used for interlaced frames. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.2.2.18 MAE Front End Backward Bottom Field Reference C_r Register

maefe_brefbotcr **Offset = 0x004C**

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
	FIELDPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:0	FIELDPTR	This is the physical address of forward C_r bottom reference field, used for interlaced data. The motion compensation block of MAE reads from this buffer. This address should be cache line aligned.	R/W	0

6.2.2.19 MAE Front End Interrupt Status Register

Write a 1 to clear the status bit; writing a 0 has no effect.

maefe_intstat **Offset = 0x0050**

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
																																FEDN
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:1	-	Reserved.		0
0	FEDN	Front End Done. 0 The MAE front end is not active or is processing the current group of macroblocks. 1 The MAE front end has completed processing the current group of macroblocks.	R/W	0

6.2.2.20 MAE Front End Interrupt Enable Register

maefe_intenable **Offset = 0x0054**

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

Bits	Name	Description	Read/Write	Default
31:1	-	Reserved.		0
0	FEDNEN	Front End Done Interrupt Enable. If enabled, an interrupt will be generated when the MAE has finished processing all macroblocks in the DMA descriptor list. 0 Disable the interrupt. 1 Enable the interrupt.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.2.2.21 MAE Front End Scratch Pad Address Register

The available scratch pad area must be at least 576 bytes * no. of macroblocks per row.

maefe_scratchpad **Offset = 0x0058**

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

Bits	Name	Description	Read/Write	Default
31:0	ADDR	Physical address of the scratch pad area for WMV9 loop filtering. This address should be cache line aligned. Applies only when **maefe_config**[LOOP] = 1 or **maefe_config**[SMOO] = 1.	R/W	0

6.2.2.22 MAE WMV9 PQUANT Register

maefe_wmv9pquant **Offset = 0x005C**

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

Bits	Name	Description	Read/Write	Default
31:5	-	Reserved, must be written 0.	R/W	0
4:0	PQUANT	Parameter passed to WMV9 in-loop filters as described in the WMV9 specification. Applies only when **maefe_config**[LOOP] = 1 or **maefe_config**[SMOO] = 1.	R/W	0

6.2.2.23 MAE DMA Descriptor Pointer Register

maefe_dmadscr **Offset = 0x1004**

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

Bits	Name	Description	Read/Write	Default
31:0	PTR	This is the physical address of the first descriptor in the list. The default value for this field is undefined.	R/W	UNDEF

6.2.2.24 MAE DMA Doorbell Register

maefe_dmadbell **Offset = 0x1008**

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

Bits	Name	Description	Read/Write	Default
31:1	—	Reserved.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
0	DB	When this bit is written to 1 the MAE DMA begins fetching descriptors and the corresponding data, starting from the descriptor at the address referenced by **maefe_dmadscr**[PTR].	R/W	0

6.2.3 Data Input

Macroblocks and their associated data are passed to the front end of the MAE by way of the dedicated MAE DMA controller. The data passed to the MAE includes four header words, weighting matrices, motion vectors, and macroblock data. In addition to setting up this data, software must create descriptors that point to this information. The MAE DMA then traverses this list of descriptors, fetching the data needed and putting it into the input FIFO.

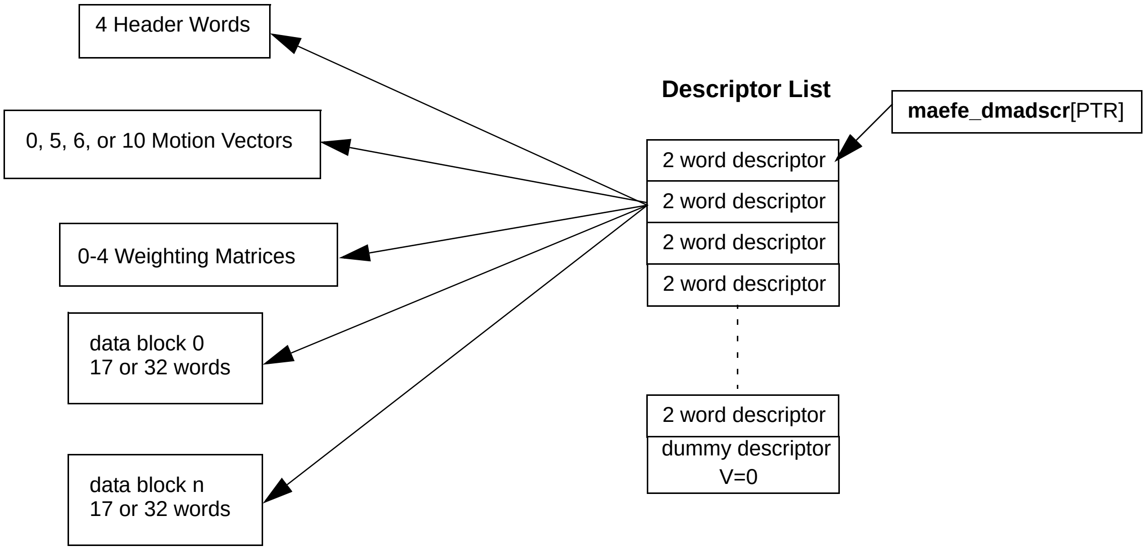


Figure 6-2. Data Input Structure

Figure 6-2 shows a conceptual drawing of the front end data and the descriptors for a series of macroblocks. As can be seen from this diagram, the size of the input data for a macroblock is variable; for example there can be between 0 and 4 weighting matrices. Each descriptor is 2 words and the list is in contiguous memory. It is up to software to decide how many descriptors to use to represent the input data for a frame. For best performance, the number of descriptors per frame should be minimal. Each frame's descriptor list must be terminated by a dummy block with the V bit set to 0.

www.DataSheet4U.com

www.DataSheet4U.com

6.2.3.1 MAE DMA and Descriptors

Data is transmitted to the front end of the MAE by use of the MAE DMA. The MAE DMA descriptor pointer **maefe_dmadscr**[PTR] should be programmed with the physical address of the first descriptor. The descriptors are stored sequentially in memory. The format of the 2-word MAE DMA descriptor is described below. The first word contains the byte count as well as the valid bit. The second word of the descriptor contains the address of the data to be fetched. After setting up the array of descriptors, software must write the doorbell register (**maefe_dmadbell**[DB] = 1) to start the MAE DMA controller.

mae_dscrword0 **Header 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
	V	▨									BC																					

Bits	Name	Description
31	V	Valid Bit. 0 This descriptor is not valid. The MAE DMA controller halts when it encounters a non-valid descriptor. 1 This descriptor is valid. The MAE DMA controller continues to the next descriptor after processing a valid descriptor.
30:22	—	Reserved.
21:0	BC	Byte Count. This is the number of bytes that are pointed to by the address pointer.

mae_dscrword1 **Header offset = 0x0004**

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

Bits	Name	Description
31:0	ADDR	This is the physical address of the data to be fetched.

6.2.3.2 Header Words

The header words precede each macroblock in memory and contain information about the macroblock, such as quant parameters, and macroblock position. All four header words must be sent in with each macroblock.

MAE Header Word 0

mae_hdr0 **Header 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
	BB	IQMUL2									WTCHGMSK				IQADD1						DCLUMA						DCCHROMA					

Bits	Name	Description
31	BB	Big Blocks. Unique to this hardware implementation. If set to 1, then the size of all the IDCT input coefficients are 16 bits. If 0, then all the AC coefficients are 8 bits, and the DC coefficients remain 16 bits. The DC coefficient is packed in its own word, leaving the AC coefficients to start in the next word. When BB = 0, this gives 1 word that holds the DC coefficient, and the next 16 words hold the 63 AC coefficients. The last byte of the 16th word is unused. In all cases of XFORMSIZE, each sub-block must have a DC coefficient that is 16 bits. This should be followed by words that contain the AC coefficients for that sub-block. The DC coefficient is in a word of its own and each sub-block should start on a word boundary.

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description
30:22	IQMUL2	Inverse Quantization Multiplier 2. This is the iq_mul_2 parameter in the general inverse quantization formula. See Section 6.2.4.1 "Quantizer Parameters" on page 167.
21:18	WTCHGMSK	Weight Change Mask. INTER_C (bit 21), INTER_Y (bit 20), INTRA_C (bit 19, and INTRA_Y (bit 18). When one of these bits is set, it indicates that the specified weighting matrix needs to be changed and the weighting matrix data follows. MAE puts no limitation on when the weighting matrices change or the number that change at any given time. MAE does not have any knowledge of differences between decoding standards. For instance, MPEG4 only supports INTRA and INTER weighting matrices. In this case, when the INTRA matrix changes, both of the MAE INTRA matrices are setup to change and are equal. The weighting matrices' data must precede the macroblock data to MAE.
17:12	IQADD1	Inverse Quantization Add 1. This is the iq_add_1 parameter in the general inverse quantization formula. See Section 6.2.4.1 "Quantizer Parameters" on page 167.
11:6	DCLUMA	DC Scaler Luma. Used with DC scaler chroma to reconstruct the DC values in intra macroblocks. In the case of h.263, luma = chroma = 8. In MPEG4, these can be different.
5:0	DCCHROMA	DC Scaler Chroma. Used with DC scaler luma to reconstruct the DC values in intra macroblocks. In the case of h.263, luma = chroma = 8. In MPEG4, these can be different.

MAE Header Word 1

mae_hdr1 **Header offset = 0x0004**

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
--												CBP								IQDIV3			IQADD2								

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>31:20</td><td>--</td><td>Reserved.</td></tr>
<tr><td>19:12</td><td>CBP</td><td>Coded Block Parameter Mask. The CBP parameter determines which blocks in the transmitted macroblock are coded. The width of this parameter is 8 bits to support macroblocks formatted up to 4:2:2. If the bit in the CBP mask is set, the corresponding block is coded. Otherwise no data should be expected for this block in the transmitted macroblock data. For INTRA blocks every thing must be coded, so the CBP parameter is 0xFF in that case. The CBP parameter is used for “P” macroblocks (as defined in the MPEG specification). The effect is the decoder treats the block as part of a “P” macroblock, where the motion vector for the block equals 0 and the coefficient data is all equal to zero. Decoding of non-coded blocks requires only a copy of the corresponding block in the previous picture. Note that the C_b1 and C_r1 blocks apply only when the <b>maefe_config</b>[BC] parameter is 1 (8 blocks, 4:2:2 mode).
<table>
<tr><th colspan="8">CBP</th></tr>
<tr><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td></tr>
<tr><td colspan="8">Corresponding Block</td></tr>
<tr><td>Y0</td><td>Y1</td><td>Y2</td><td>Y3</td><td>C_b0</td><td>C_r0</td><td>C_b1</td><td>C_r1</td></tr>
</table>
</td></tr>
<tr><td>11:9</td><td>IQDIV3</td><td>Inverse Quantization Divider 3. This is the iq_div_3 parameter in the general inverse quantization formula. See Section 6.2.4.1 "Quantizer Parameters" on page 167. The value programmed in these bits is the number of bits by which to shift right.</td></tr>
<tr><td>8:0</td><td>IQADD2</td><td>Inverse Quantization Add 2. This is the iq_add_2 parameter in the general inverse quantization formula. See Section 6.2.4.1 "Quantizer Parameters" on page 167.</td></tr>
</table>

www.DataSheet4U.com

www.DataSheet4U.com

MAE Header Word 2

mae_hdr2 **Header 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
MBMODE																XFORMSIZE															

Bits	Name	Description
31:16	MBMODE	Macroblock Mode. This field is used to determine whether motion compensation should be performed and if so what type of motion compensation to perform. There are 2 bits associated with each block, the meaning of these bits is shown below: 00 INTRA 01 Forward 10 Backward 11 Bidirectional As the table below shows the MBMODE parameter is specified for each of the blocks that make up a macroblock.
15:0	XFORMSIZE	Transform Size Parameter Mask. The transform size parameter is used to determine what size transform to perform on the coded block. The width of this parameter is 16 bits to support macroblocks formatted up to 4:2:2. If the CBP mask for a block contains a zero, no transform is performed. There are 2 bits associated with each block, these bits specify the transform size as shown below: 00 8x8 01 8x4 10 4x8 11 4x4 Software must pack each of the sub-blocks together during variable length decoding so that each macroblock resides in contiguous memory. Note that the Cb1 and Cr1 blocks apply only when the **maefe_config**[BC] = 1.

MBMODE							
31:30	29:28	27:26	25:24	23:22	21:20	19:18	17:16
Corresponding Block							
Y0	Y1	Y2	Y3	C_b0	C_r0	C_b1	C_r1

XFORMSIZE							
15:14	13:12	11:10	9:8	7:6	5:4	3:2	1:0
Corresponding Block							
Y0	Y1	Y2	Y3	C_b0	C_r0	C_b1	C_r1

MAE Header Word 3

mae_hdr3 **Header Offset = 0x000C**

Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	RND	PS	DCT	FP	FT	FB	BT	BB	MBTYPE			PRECY		PRECUV		XPOS								YPOS							

Bits	Name	Description
31	-	Reserved.
30	RND	Motion Compensation Rounding Control Value.
29	PS	Picture Structure.
28	DCT	DCT Type.

www.DataSheet4U.com

www.DataSheet4U.com

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>27</td><td>FP</td><td>Field Prediction Mode.</td></tr>
<tr><td>26</td><td>FT</td><td>Forward Top Field Reference.</td></tr>
<tr><td>25</td><td>FB</td><td>Forward Bottom Field Reference.</td></tr>
<tr><td>24</td><td>BT</td><td>Backward Top Field Reference.</td></tr>
<tr><td>23</td><td>BB</td><td>Backward Bottom Field Reference.</td></tr>
<tr><td>22:20</td><td>MBTYPE</td><td>The MBTYPE parameter is used to specify details of motion compensation:
000 16x16
001 Reserved
010 16x8 field
011 8x8
All other values are reserved.</td></tr>
<tr><td>19:18</td><td>PRECY</td><td>Motion Compensation Precision For Luminance.
00 Full pel precision. (No Filter)
01 1/4 pel precision. (2-tap filter)
10 1/2 pel precision. (Bilinear filter)
11 1/4 pel precision (Multitap filter)
The type of filtering performed is also controlled by the codec style bit <b>maefe_config</b>[COD].
Allowed values for this field depend on <b>maefe_config</b>[COD]:
<table>
<tr><th></th><th colspan="2">mae_config[COD]</th></tr>
<tr><th>PRECY</th><th>MPEG</th><th>WMV9</th></tr>
<tr><td>00</td><td>Allowed</td><td>Allowed</td></tr>
<tr><td>01</td><td>Reserved</td><td>Allowed</td></tr>
<tr><td>10</td><td>Allowed (!qpel)</td><td>Reserved</td></tr>
<tr><td>11</td><td>Allowed (qpel)</td><td>Allowed</td></tr>
</table></td></tr>
<tr><td>17:16</td><td>PRECUV</td><td>Motion Compensation Precision For Chroma.
00 Full pel precision. (No Filter)
01 1/2 pel precision. (Multitap filter)
10 1/2 pel precision. (Bilinear filter)
11 1/4 pel precision (Multitap filter)
The type of filtering performed is also controlled by the codec style bit <b>maefe_config</b>[COD].
Allowed values for this field depend on <b>maefe_config</b>[COD]:
<table>
<tr><th></th><th colspan="2">mae_config[COD]</th></tr>
<tr><th>PRECUV</th><th>MPEG</th><th>WMV9</th></tr>
<tr><td>00</td><td>Allowed</td><td>Allowed</td></tr>
<tr><td>01</td><td>Reserved</td><td>Allowed</td></tr>
<tr><td>10</td><td>Allowed</td><td>Allowed</td></tr>
<tr><td>11</td><td>Reserved</td><td>Allowed</td></tr>
</table></td></tr>
</table>

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description
15:8	XPOS	X-axis Position Of Macroblock. The units of XPOS are multiples of 8 pixels. The value set in this bit field should be the block position in pixels divided by 8. In this way the block position is constrained to 8 pixel boundaries.
7:0	YPOS	Y-axis Position Of Macroblock. The units of YPOS are multiples of 8 pixels.The value set in this bit field should be the block position in pixels divided by 8. In this way the block position is constrained to 8 pixel boundaries.

6.2.3.3 Motion Vectors

The number of motion vectors that are sent can be determined by the MBMODE and MBTYPE parameters in the macroblock header words. Based on this, the number of luma motion vectors sent is either zero, four, or eight. The number of chroma motion vectors sent is zero, one, two, or four. Table 6-3 shows how many motion vectors are transmitted to MAE. Luma vectors are sent first followed by chroma vectors.

Table 6-3. Motion Vector Rules

MBMODE	MBTYPE	Number of Luma Vectors Transmitted	Number of Chroma Vectors Transmitted	Notes for Luma Vectors
Forward or Backward	16x16	4	1	MV0=MV1=MV2=MV3
Forward or Backward	16x8	4	2	MV0=MV1, MV2=MV3
Forward or Backward	8x8	4	1	No Restrictions
Bidirectional	16x16	4	2	MV0=MV1, MV2=MV3
Bidirectional	16x8	4	4	No Restrictions
Bidirectional	8x8	8	2	No Restrictions

In some cases more motion vectors are sent than are actually needed; in these cases some of the motion vectors should be set equal to one another. For example, when MBMODE = Forward, MBTYPE = 16x8, four luma vectors are transmitted followed by one chroma vector. The first motion vector transmitted must be set equal to the second motion vector, and the third motion vector should equal the fourth motion vector. The first two motion vectors transmitted are used for the top two luma blocks (y0 & y1); the second two motion vectors are used for the second two luma blocks (y2 & y3). The chroma vector is used for the chroma blocks (cr0, cr1, cb0, cb1).

Each motion vector itself is a 32-bit word: 16 bits for the y-offset, and 16-bits for the x-offset. The format for a motion vector is shown below.

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
	MV*n*_X																MV*n*_Y															

Bits	Name	Description
31:16	MV*n*_X	X-axis component for motion vector *n* (2047 max, -2048 min). Motion vector units are defined by **mae_hdr3**[PRECY, PRECUV].
15:0	MV*n*_Y	Y-axis component for motion vector *n* (2047 max, -2048 min). Motion vector units are defined by **mae_hdr3**[PRECY, PRECUV].

Luma motion vectors are specified in terms of luma pixels, and chroma vectors are specified in terms of chroma pixels. The units for luma and chroma are determined by their respective precision fields (**mae_hdr3**[PRECY, PRECUV]). In the case of MPEG codec style, motion vectors must be clipped to keep reference fetches from going too far out of frame. These limits are shown in Table 6-4.

www.DataSheet4U.com

www.DataSheet4U.com

Table 6-4. MPEG Motion Vector Clipping

Direction	Limit (16x16)	Limit (8x8)	Limit (16x8)
Top	-16	-8	-8
Bottom	15	7	7
Left	-16	-8	-16
Right	15	7	15

6.2.3.4 Weighting Matrices

The weighting matrix used in inverse quantization can dynamically change, and there are 4 different weighting matrices: INTER_C, INTER_Y, INTRA_C, and INTRA_Y. There is a bit field in the header data (Header Word 0: WTCHGMSK), which indicates that one or more of the weighting matrices is being changed for future macroblock processing. When this occurs, the changed matrices should be sent in after the header words. Each weighting matrix is a 64 element, 8x8 block of 8-bit signed data. If a weighting matrix is not sent in for a particular macroblock, the appropriate weighting matrix passed in last is used. For WMV9 weighting matrices must have all elements set to one.

6.2.3.5 Block Data

The number blocks of data for a macroblock depends not only on the block count (**maefe_config**[BC] for 4:2:0 or 4:2:2), but also on how many blocks are coded. The coded block parameter (**mae_hdr1**[CBP]) specifies which blocks are coded and therefore sent in to MAE.

The size of block data which is transmitted to MAE is determined by the big-block bit, **mae_hdr0**[BB]. If BB = 1, the size of all the IDCT input coefficients are 16 bits; if BB = 0, all the AC coefficients are 8 bits, and the DC coefficients remain 16 bits. The DC coefficient is packed in its own word, leaving the AC coefficients to start in the next word. When BB = 0, this gives 1 word which holds the DC coefficient and the next 16 words hold the 63 AC coefficients. (The last byte of the 16th word is unused.)

www.DataSheet4U.com

www.DataSheet4U.com

6.2.4 Inverse Quantization

MAE supports a variety of compression standards by generalizing the input parameters and formula used to perform inverse quantization. The supported standards include H.263, MPEG1, MPEG2, MPEG4, and WMV9.

There are many forms that the inverse quantization can take based on the standard and block type. The general form the MAE uses is shown below:

DC coefficients INTRA blocks only:
F[0][0] = block_data * dc_scaler_luma
F[0][0] = block_data * dc_scaler_chroma

AC coefficients and INTER DC coefficients: $F = (((block\_data * 2^{iq\_mul\_1} + iq\_add\_1)) * W * iq\_mul\_2 + iq\_add\_2) / 2^{iq\_div\_3}$

6.2.4.1 Quantizer Parameters

The iq_mul_1 parameter is programmed in **maefe_config**[IQMUL1] since it does not change during the decode of a sequence. The value of this parameter specifies the number of bits to shift left, effectively multiplying by a power of two.

The iq_add_1 parameter is set in a header word (**mae_hdr0**[IQADD1]) since it changes during the decode of a sequence. This parameter specifies the value to add to the product of (block_data * iq_mul_1). The sign of iq_add_1 is determined by the sign of the input block data.

The weighting matrix is one of four 8x8 quantization matrices that can be sent in with the header data and the macroblock data. If it is not sent, then the previous sets are used. The four matrices correspond to intra luma, intra chroma, inter luma, and inter chroma. Each weighting matrix element is a signed 8 bit value.

6.3 MAE Back End

6.3.1 Scaling and Filtering (SCF)

The MAE back end supports scaling/filtering of planar or interleaved video component data.

6.3.1.1 Scaling and Filtering Definitions

Upsampling

Upsampling is the process of inserting new data samples between original data samples to increase the sampling rate.

Interpolation

Upsampling increases the sample rate and consequently the Nyquist frequency and may introduce undesirable spectral images. A low-pass filter is generally applied to the new data samples after upsampling to avoid this problem. Interpolation is the process of upsampling the original data samples followed by low-pass filtering the new data samples.

Downsampling

Downsampling is the process of reducing the sampling rate by removing or throwing away original data samples.

Decimation

Since downsampling reduces the sample rate and consequently the Nyquist frequency, aliasing may be introduced if any original data samples remain, after downsampling, that are outside the reduced Nyquist frequency. A low-pass filter is generally applied to the original data samples before downsampling to avoid this problem. Decimation is the process of filtering the original data samples with a low-pass filter followed by downsampling.

www.DataSheet4U.com

www.DataSheet4U.com

Resampling/Scaling

Resampling combines interpolation with decimation and allows the two low-pass filters to be combined into one filter as shown in Figure 6-3 on page 168. Thus, resampling is accomplished via upsampling followed by filtering followed by downsampling. This is how scaling is implemented on the MAE scaler/filter (SCF).

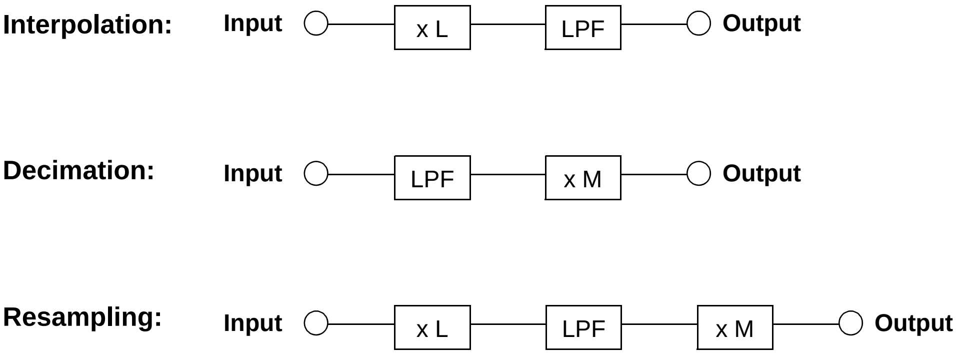


Figure 6-3. Resampling

6.3.1.2 MAE SCF Horizontal Scaling

Each block of data (32 bytes for Y; 32, 16 or 8 bytes for U and V) passes through the horizontal scaler followed by the vertical scaler. The horizontal scaler uses four taps in the filtering process. (A tap is equivalent to an original unscaled pixel of data in this context.) Refer to Figure 6-4 on page 169 for the following discussion.

MAE SCF Horizontal Upscaling

The example below shows how a block of data is scaled by 2x. The block of data is scanned from left to right (locations 0,0 to 31,0) and one new pixel is created between every two original pixels. The first pixel created is located between pixels 0,0 and 1,0. This pixel is assigned a value of 0 as a placeholder and then is filtered with the four surrounding pixels: two to the left and two to the right. Since pixel 0,0 represents the edge of the frame there is no real pixel value that can be used for the outer-most tap, T0. In this case, pixel 0,0 (the nearest pixel) is mirrored and its value is used for T0. When creating the next pixel (between pixel 1,0 and 2,0) mirroring is no longer needed because now the four nearest pixels are pixels 0,0 1,0 2,0 and 3,0. Mirroring is also applied at the right edge of the frame when encountered.

This process continues until the entire block of data has been scanned. At the end, the original data has been scaled and now contains (2 x 32) = 64 bytes.

Note: With some scale factors, such as 3x, an extra horizontal pixel may be created due to precision constraints. For example, when using a 1/3 scale ratio (inverse of 3x scale factor), an extra pixel is created in the horizontal direction exceeding the intended (3 x horizontal_frame_width). In order to avoid this, software must adjust the scale ratio by adding an offset of 0x0000 0004 to the fractional portion of the scale ratio. This provides a value slightly greater than 1/3, which prevents the extra pixel. The value needs to be 0x0000 0004, because the worst case subsampling (shift right) is >> 2.

Note: For subsampled channels due to precision issues, it is possible to get cases where the subsampled channels create a different number of pixels than the non-subsampled channels (Y compared with Cb or Cr). To avoid this, software should program the least 2 significant digits of the fractional portion of the horizontal scale ratio to 0.

www.DataSheet4U.com

www.DataSheet4U.com

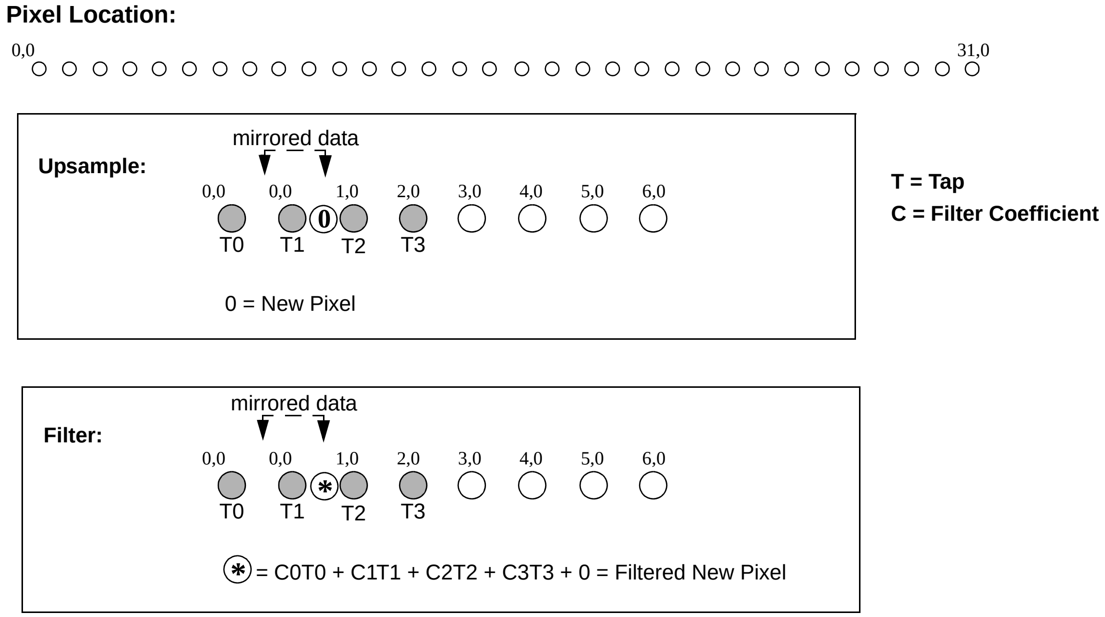


Figure 6-4. Horizontal Scaling by 2x

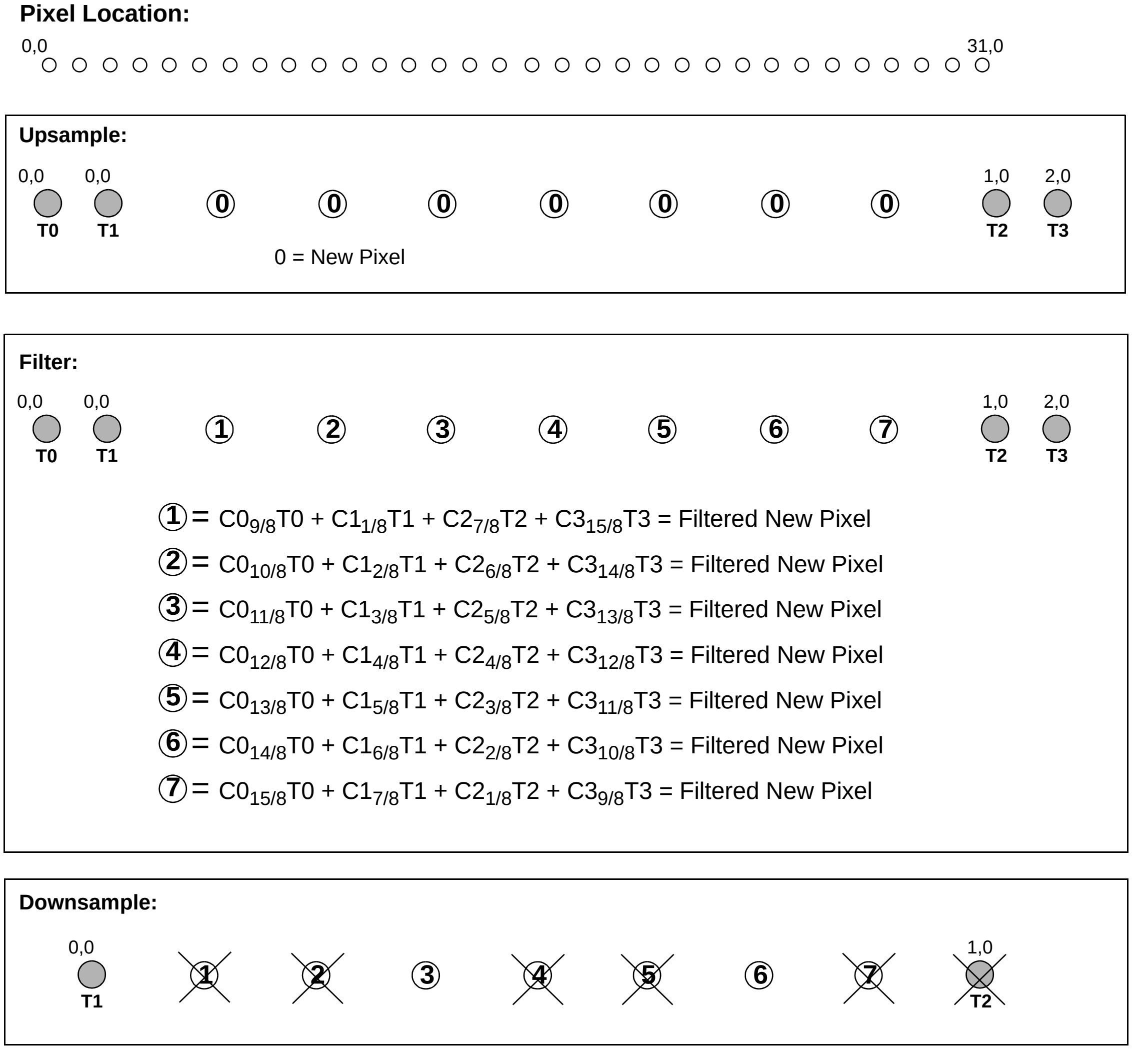
MAE SCF Horizontal Downscaling

The example in Figure 6-5 on page 170 shows how a block of data gets scaled by 8/3x (upsample by 8 and downsample by 3). The block of data is scanned from left to right (locations 0,0 to 31,0) and seven new pixels are created between every two original pixels. The first pixel created is located 1/8 of the distance between pixels 0,0 and 1,0. The next pixel is 2/8 (or 1/4) of the distance, the next 3/8, and so on. These pixels are assigned a value of 0 as a placeholder and then are filtered with the four surrounding pixels (two to the left and two to the right). As shown in Figure 6-5 on page 170, the coefficients used to filter each new pixel depend on the new pixel's distance from each tap. In total, there are 32 possible coefficients per tap that can be used for filtering corresponding to the maximum upsampling scale factor of 32. Note that the total scale factor (upsampling + downsampling) must not exceed 4x and must be greater than or equal to 1/32x.

After filtering, downsampling is applied. In the downsampling example shown in Figure 6-5 on page 170, two out of every three pixels is discarded. Between the original pixels 0,0 and 1,0 only three pixels are kept; pixel 0,0, pixel 3, and pixel 6. This process continues for the rest of the line, at which point, horizontal scaling is finished (for that line). If the original width of the line is 90, after upsampling the new width would be 720 and after downsamping 240 which corresponds to 90 x 8/3 = 240.

www.DataSheet4U.com

www.DataSheet4U.com



① = $C0_{9/8}T0 + C1_{1/8}T1 + C2_{7/8}T2 + C3_{15/8}T3$ = Filtered New Pixel

② = $C0_{10/8}T0 + C1_{2/8}T1 + C2_{6/8}T2 + C3_{14/8}T3$ = Filtered New Pixel

③ = $C0_{11/8}T0 + C1_{3/8}T1 + C2_{5/8}T2 + C3_{13/8}T3$ = Filtered New Pixel

④ = $C0_{12/8}T0 + C1_{4/8}T1 + C2_{4/8}T2 + C3_{12/8}T3$ = Filtered New Pixel

⑤ = $C0_{13/8}T0 + C1_{5/8}T1 + C2_{3/8}T2 + C3_{11/8}T3$ = Filtered New Pixel

⑥ = $C0_{14/8}T0 + C1_{6/8}T1 + C2_{2/8}T2 + C3_{10/8}T3$ = Filtered New Pixel

⑦ = $C0_{15/8}T0 + C1_{7/8}T1 + C2_{1/8}T2 + C3_{9/8}T3$ = Filtered New Pixel

Figure 6-5. Horizontal Scaling by 8/3x

www.DataSheet4U.com

www.DataSheet4U.com

6.3.1.3 MAE SCF Vertical Scaling

After horizontal scaling, the block of data is ready for vertical scaling. The vertical scaling process is exactly the same as the horizontal scaling process, only the process is applied in the y/vertical direction. For this reason, four blocks of data must be buffered in order to provide the relevant tap points. Figure 6-6 shows this process.

Note: With some scale factors, such as 3x, an extra vertical line may be created due to precision constraints. For example, when using a 1/3 scale ratio (inverse of 3x scale factor), an extra line is created in the vertical direction exceeding the intended (3 x vertical_frame_height). In order to avoid this, software must adjust the scale ratio by adding an offset of 0x0000 0004 to the fractional portion of the scale ratio. This provides a value slightly greater than 1/3, which prevents the extra line. The value needs to be 0x0000 0004 because the worst case subsampling (shift right) is >> 2.

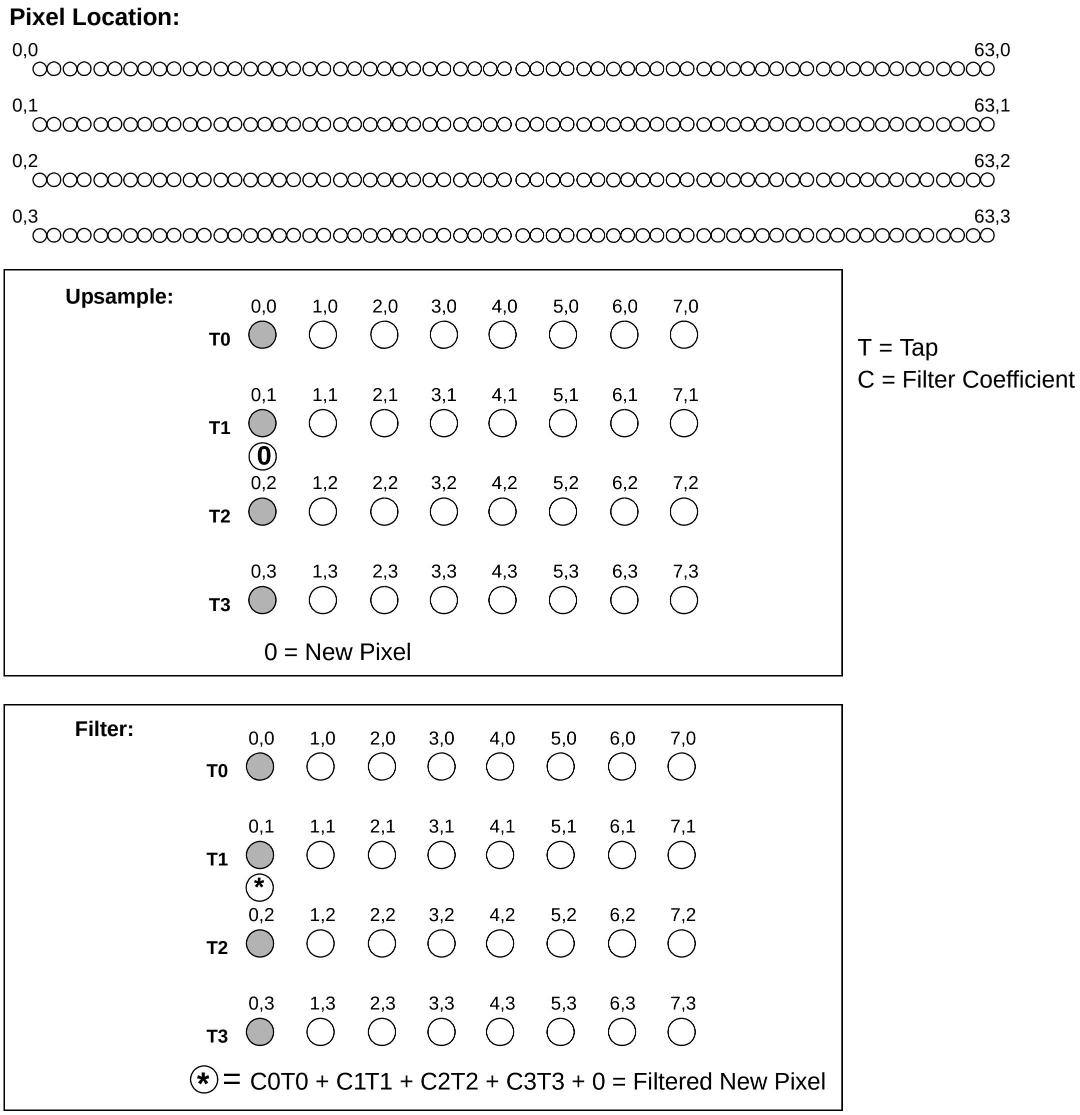


Figure 6-6. Vertical Scaling by 2x

www.DataSheet4U.com

www.DataSheet4U.com

6.3.1.4 MAE SCF Filtering

The filtering applied in the SCF engine is defined by the coefficients programmed into two (one for horizontal filtering and one for vertical filtering) 32-entry look up tables for each component (Y, U, and V), giving a total of six 32-entry tables.

For each tap, a total of 32 coefficients are to be programmed, which corresponds to the granularity of scaling available. Filtering is biased towards Tap1, meaning a new pixel's location is calculated in relation to its distance from Tap1 when upsampling. If a pixel is located exactly at Tap1, it is considered an original pixel. In this case the coefficients would normally be programmed to {0,1,0,0} (for location x =0), allowing the pixel to pass through with its value unchanged. The same principle is applied to filtering in the vertical (or y) direction.

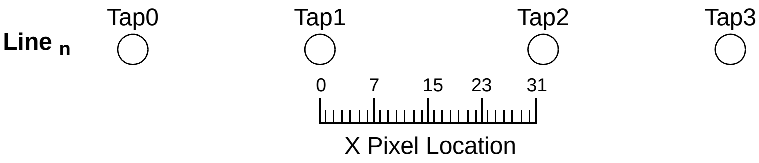


Figure 6-7. LUT Address Generation for Horizontal Filter Coefficients

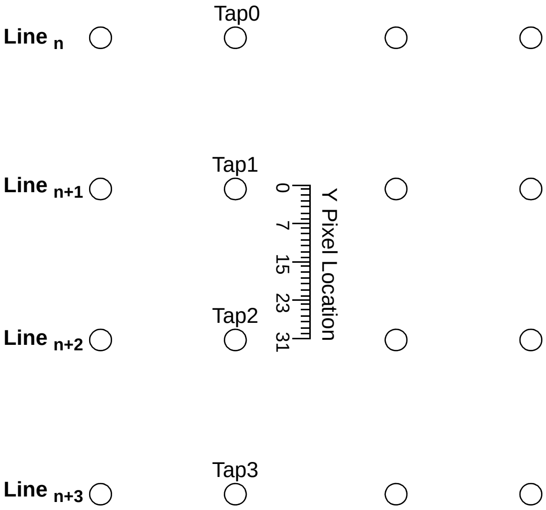


Figure 6-8. LUT Address Generation for Vertical Filter Coefficients

6.3.2 MAE Colorspace Conversion (CSC)

The MAE supports hardware frame conversion from one colorspace (such as RGB or YUV) to another. The equations used in the conversion process are generic, and the choice of colorspaces depends solely upon the coefficients programmed into the equations via the configuration registers.

6.3.2.1 MAE CSC Coefficients and Offsets

The equations below show how the coefficient and offset registers are programmed for color space conversion. In these and following expressions, A, B, and C denote the components of the original colorspace, while X, Y, and Z denote the components of the new colorspace. For example, when converting from YC_bC_r to RGB, ABC corresponds to YC_bC_r, and XYZ corresponds to RGB.

X = (maebe_cscxcffa * A) + (maebe_cscxcffb * B) + (maebe_cscxcffc * C) + maebe_cscxoff

Y = (maebe_cscycffa * A) + (maebe_cscycffb * B) + (maebe_cscycffc * C) + maebe_cscyoff

Z = (maebe_csczcffa * A) + (maebe_csczcffb * B) + (maebe_csczcffc * C) + maebe_csczoff

www.DataSheet4U.com

www.DataSheet4U.com

6.3.2.2 MAE Source Input Formats

Input Format—YC_bC_r 4:2:2

YC_bC_r 4:2:2 formatting assumes the C_b and C_r components are subsampled by two in the horizontal direction as shown in Figure 6-9.

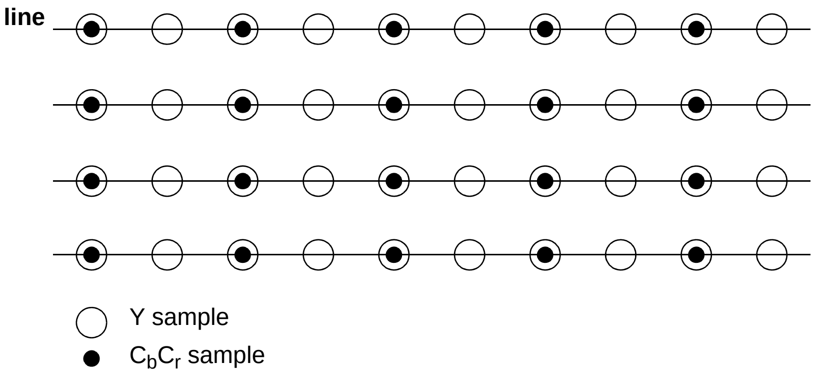


Figure 6-9. Subsampling for YC_bC_r 4:2:2 Formatting

Input Format—YC_bC_r 4:2:0

YC_bC_r 4:2:0 formatting assumes the C_b and C_r components are subsampled by two in both the horizontal and vertical direction as shown in Figure 6-10.

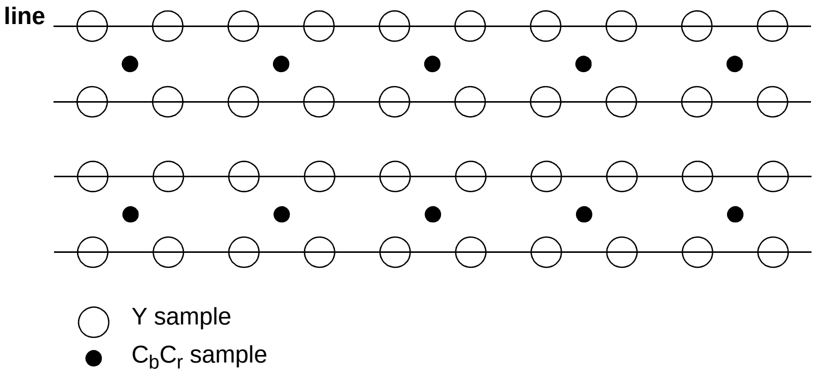


Figure 6-10. Subsampling for YC_bC_r 4:2:0 Formatting

www.DataSheet4U.com

www.DataSheet4U.com

Input Format—YC$_b$C$_r$ 4:1:1

YC$_b$C$_r$ 4:1:1 formatting assumes the C$_b$ and C$_r$ components are subsampled by four in the horizontal as shown in Figure 6-11.

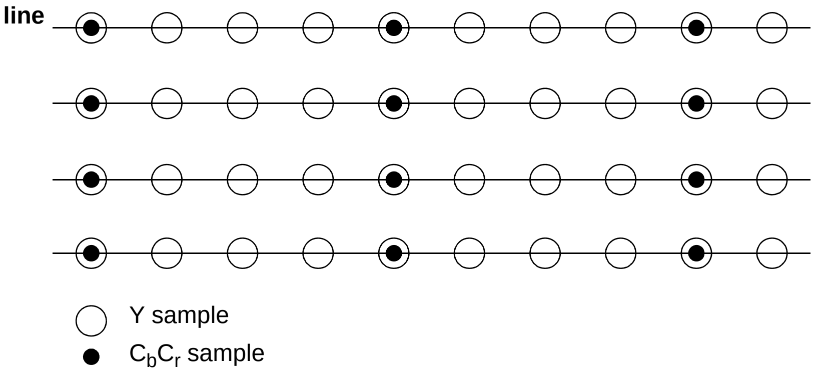


Figure 6-11. Subsampling for YC$_b$C$_r$ 4:1:1 Formatting

6.3.3 Source and Destination Memory Usage

This section explains how data should be stored in memory and how the back end sources that data.

6.3.3.1 SCF Input Source Data

The original frame of data is provided to the MAE scaler/filter (SCF) through one of two sources: software or the MAE front end. In either case the pointers to this data along with supporting information about the data are programmed into the SCF through the MAE configuration registers. The SCF uses this programmed data to set up three dedicated DMA channels A, B, and C for read memory accesses. Figure 6-12 provides an example of how these channels may be configured.

Note: A, B, C would refer to Y, U, V, or R,G,B, respectively in planar mode. In interleaved mode, A would refer to UYVY (or VYUY, YUYV, or YVYU depending on the configuration of the ILM bits, refer to Section 6.3.5.3 "Source (SRC) Registers" on page 183) and channels B and C would be considered inactive. For the purpose of readability and clarity, the following sections assume a planar YUV (YV12) configuration.

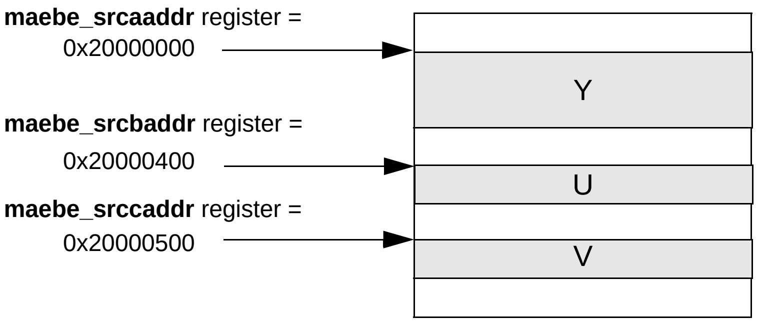

maebe_srcaaddr register =
0x20000000
maebe_srcbaddr register =
0x20000400
maebe_srccaddr register =
0x20000500
Y
U
V

Figure 6-12. SCF Input Data DMA Memory Pointers

www.DataSheet4U.com

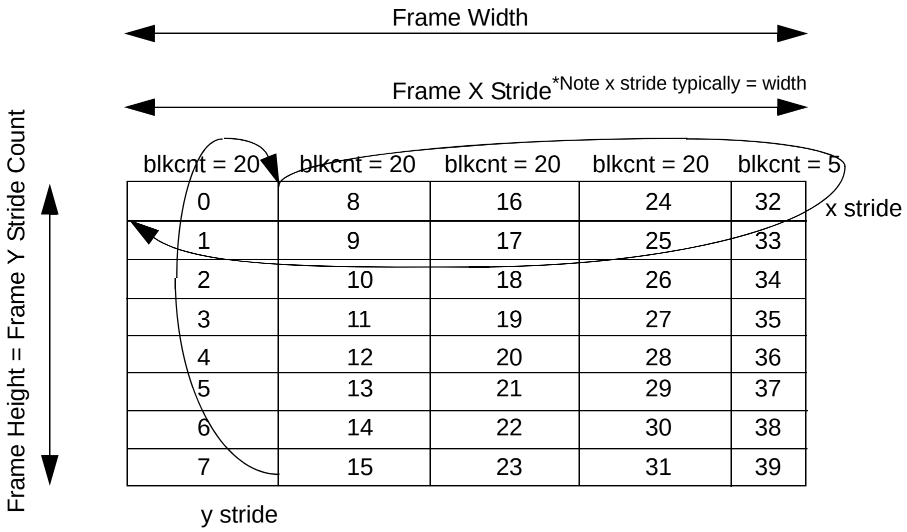
www.DataSheet4U.com

6.3.3.2 SCF Input 2D Striding

For each component, Y, U, and V, DMA blockcount and x and y striding parameters are programmed to allow the SCF to parse through the frame in a 2D fashion.

The SCF operates on a frame in 20-byte columns. The MAE internal DMA blockcount is automatically configured to equal the number of bytes the SCF operates on and tells the internal DMA engine how many bytes to fetch from memory for each read access. For luma (Y) data, this value is equal to 20, the full column width. For chroma (U and V, or C_r and C_b) data, this value may be smaller depending on the format of subsampling (i.e. 4:4:4 = 20 bytes, 4:2:0 = 10 bytes, 4:2:2 = 10 bytes, or 4:1:1 = 5bytes).

The x and y stride values are programmed in the MAE configuration registers and passed directly to the internal DMA engine. The x stride value is typically equivalent to the frame width. This value tells the DMA engine how far to stride along the x direction between memory read accesses in order to fetch the next block of data that resides directly below (visually in a frame) the previous block of data. The y stride value is actually a count and is referred to as y stride count. This value tells the DMA engine how many x strides or read accesses to perform before it strides vertically back up to the next column of data. This value is equal to the frame height. A column of data is 20 bytes wide for luma (20, 10, or 5 bytes for chroma) and y lines deep.


Frame Width
Frame X Stride
*Note x stride typically = width
Frame Height = Frame Y Stride Count
x stride
y stride

blkcnt = 20	blkcnt = 20	blkcnt = 20	blkcnt = 20	blkcnt = 5
0	8	16	24	32
1	9	17	25	33
2	10	18	26	34
3	11	19	27	35
4	12	20	28	36
5	13	21	29	37
6	14	22	30	38
7	15	23	31	39

Figure 6-13. 2D Striding Example

www.DataSheet4U.com

www.DataSheet4U.com

6.3.3.3 Source and Stride Parameters

To take advantage of cache line bursting, both the source and stride parameters for all channels should be programmed to cache line aligned addresses/values. This may necessitate padding the end of each line if the frame width is not a multiple of 32 bytes. If non-cache line aligned addresses and stride values are used, performance may be significantly impacted.

6.3.3.4 Destination Output Formatting for RGB Data

Converted RGB pixels are transferred over a 32-bit bus to the DDR interface. Output formatting is programmed through the CSC Configuration Register. Three formats (RGB or BGR) are supported as shown in Table 6-5. The alpha value is programmable, and constant over an entire frame ("CSC Alpha Value Register" on page 182)

Table 6-5. RGB 32-bit Output Formatting

Format	**32-bit Output (Big Endian)**
24-bit + α (RGB)	$\{\alpha_0[7:0], R_0[7:0], G_0[7:0], B_0[7:0]\}$
24-bit + (BGR)α	$\{B_0[7:0], G_0[7:0], R_0[7:0], \alpha_0[7:0]\}$
16-bit (RGB)	$\{R_0[7:3], G_0[7:2], B_0[7:3], R_1[7:3], G_1[7:2], B_1[7:3]\}$
16-bit (BGR)	$\{B_0[7:3], G_0[7:2], R_0[7:3], B_1[7:3], G_1[7:2], R_1[7:3]\}$
15-bit + α (RGB)	$\{\alpha_0[0], R_0[7:3], G_0[7:3], B_0[7:3], \alpha_1[0], R_1[7:3], G_1[7:3], B_1[7:3]\}$
15-bit + (BGR)α	$\{B_0[7:3], G_0[7:3], R_0[7:3], \alpha_0[0], B_1[7:3], G_1[7:3], R_1[7:3], \alpha_1[0]\}$

6.3.4 Programming Description

The MAE back end processes sequential frames and allows software to throttle the frame rate. The back end uses a multiple frame sync programming model in which the video source is initially decoded and placed into memory buffers (usually by the MAE front end). The number of buffers is determined by software. The back end should be programmed to source video data from these buffers (one at a time) in sequence.

The first step is to enable the clocks to the back end by writing to the Enable Register. When the Enable Register is cleared, all clocks are disabled and the back end is in its lowest power state. When a 1 is written to the Enable Register, the clocks are enabled and all registers in the back end (including configuration registers and storage arrays) are reset to their default state.

The configuration and control registers (except for the SCF Look-up Tables) are all double buffered, allowing software to configure the next frame before the current frame is finished processing. This reduces the possibility of system interrupt latency causing a drop in frame rate. The Frame Busy status bit indicates the current frame is being processed, while the Frame Pending status bit indicates the next frame has been configured and is pending processing. Once the first frame is configured and the Frame Start bit in the Control register has been written, hardware sets the Frame Busy status bit until that frame is finished processing and a Frame Done interrupt is generated (if enabled). During the processing of the first frame (Frame Busy is set), software can configure the second frame for processing by changing any configuration registers necessary. Once the second frame is configured, software writes the Frame Start bit again. Assuming the first frame is still processing (Frame Busy is set), hardware also sets the Frame Pending status bit indicating that a second frame is configured and ready for processing. When the first frame completes, the second frame begins processing immediately, the Frame Pending status bit clears, and the Frame Busy status bit remains set. At this point a third frame can be configured, and the process is repeated.

Frame processing can be terminated early by writing the Frame Reset bit in the Frame Control Register. A Frame Reset terminates the current frame and clears a pending frame (if one exists). A Frame Reset also resets the entire data path (not including configuration/control registers or the SCF Look-up Tables) to its default state. At this point a new frame can be configured and/or started as normal.

The software procedure for multiple frame sync is outlined in Figure 6-14 on page 177.

www.DataSheet4U.com

www.DataSheet4U.com

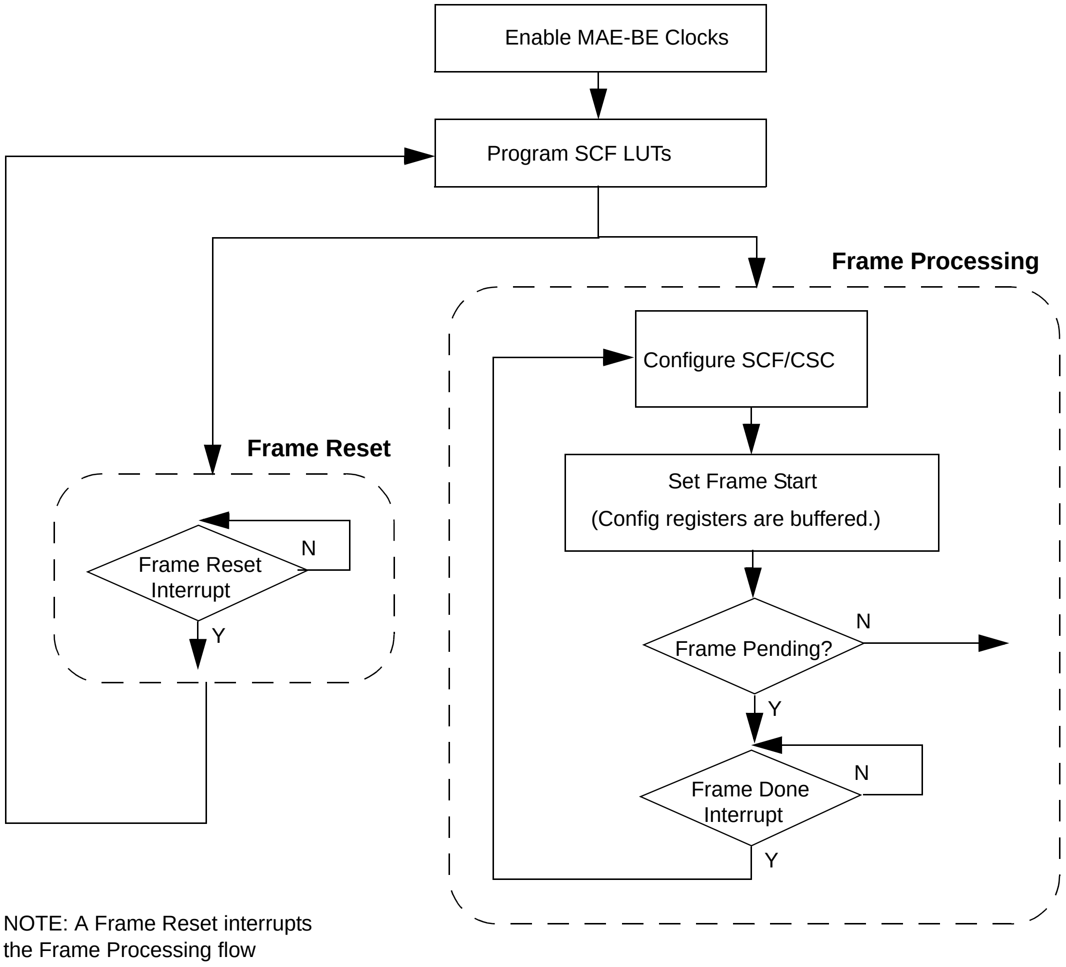


Figure 6-14. Software Programming Model

www.DataSheet4U.com

www.DataSheet4U.com

6.3.5 MAE Back End Registers

The MAE back end (BE) is controlled by a register block whose physical base address is shown in Table 6-6. The register definitions and offset values are described in the following sections.

Table 6-6. BE Base Address

Name	Physical Base Address	KSEG1 Base Address
maebe_base	0x0 1401 0000	0xB401 0000

6.3.5.1 Scaler/Filter (SCF) Registers

The scaler/filter (SCF) registers are shown in Table 6-7.

Table 6-7. SCF Registers

Offset (Note1)	Register Name	Description
0x0000	maebe_scfhsr	Horizontal Scale Ratio
0x0004	maebe_scfvsr	Vertical Scale Ratio
0x0008	maebe_scfdisable	SCF Disable
0x0100	maebe_scfhalut	Horizontal A look up table (LUT)
0x0180	maebe_scfvalut	Vertical A LUT
0x0200	maebe_scfhblut	Horizontal B LUT
0x0280	maebe_scfvblut	Vertical B LUT
0x0300	maebe_scfhclut	Horizontal C LUT
0x0380	maebe_scfvclut	Vertical C LUT

Note1. See Table 6-6 for base address.

SCF Horizontal Scale Ratio Register

This register should be programmed according to the ratio of the input width/output width. The upper halfword of this register reflects the integer portion of this ratio, while the lower halfword reflects the fractional part.

Note: The MAX supported scale factor is 4 and the MIN supported scale factor is 1/32; these translate into scale ratios of 1/4 and 32, respectively.

Note: The MAX supported frame STRIDE is 1024 pixels (or 2048 bytes in 16-bpp mode and 4096 bytes in 32-bpp mode). This implies that the MAX supported frame WIDTH is 1024 pixels (or 2048 bytes in 16-bpp mode and 4096 bytes in 32-bpp mode), since the frame width is always less than or equal to the frame stride. The MAX supported frame HEIGHT is 1024 pixels. The MAX supported FRAME_SIZE is 1,048,576 pixels. The scale factor should be programmed with these input and output frame size constraints in mind.

maebe_scfhsr **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
	SRI																SRF															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:16	SRI	The integer part of the input width/output width ratio.	R/W	0
15:0	SRF	The fractional part of the input width/output width ratio.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

SCF Vertical Scale Ratio Register

This register should be programmed according to the ratio of the input height/output height. The upper halfword of this register reflects the integer portion of this ratio, while the lower halfword reflects the fractional part.

Note: The MAX supported scale factor is 4 and the MIN supported scale factor is 1/32; these translate into scale ratios of 1/4 and 32, respectively.

The MAX supported frame STRIDE is 1024 pixels (or 2048 bytes in 16-bpp mode and 4096 bytes in 32-bpp mode). This implies that the MAX supported frame WIDTH is 1024 pixels (or 2048 bytes in 16-bpp mode and 4096 bytes in 32-bpp mode), since the frame width is always less than or equal to the frame stride. The MAX supported frame HEIGHT is 1024 pixels. The MAX supported FRAME_SIZE is 1,048,576 pixels. The scale factor should be programmed with these input and output frame size constraints in mind.

maebe_scfvsr **Offset = 0x0004**

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

Bits	Name	Description	Read/Write	Default
31:16	SRI	The integer part of the input height/output height ratio.	R/W	0
15:0	SRF	The fractional part of the input height/output height ratio.	R/W	0

SCF Disable Register

This register is used to bypass the Scaler/Filter when only Colorspace Conversion is needed. This saves power by disabling the clocks to the SCF and routing source data directly to the CSC.

maebe_scfdisable **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
																																DIS
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:1	--	Reserved.	--	0
0	DIS	SCF Disable. 0 Enable the SCF. 1 Disable (bypass) the SCF.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

SCF Filtering Coefficient LUTs

Upsampling/downsampling is accomplished through interpolation/decimation, which combines resampling with filtering in order to reduce aliasing effects. Interpolation requires upsampling followed by filtering, whereas decimation requires filtering followed by downsampling. The combination of interpolation with decimation allows the two filters to be reduced to one. Thus, resampling is accomplished via upsampling followed by filtering followed by downsampling.

The MAE back end applies filtering to planar YUV data using generic 4-tap polyphase filters. The type of filtering applied is defined by the coefficients programmed in six look up tables (LUTs), corresponding to Horizontal and Vertical A, B, and C components. Each LUT consists of 32 contiguous memory locations containing 32 sets of 4 coefficients.

For each tap, a total of 32 coefficients are to be programmed, which corresponds to the granularity of scaling available. Filtering is biased towards Tap1, meaning a new pixel's location is calculated in relation to its distance from Tap1 when upsampling (see Section 6.3.1.4 "MAE SCF Filtering" on page 172). If a pixel is located exactly at Tap1, it is considered an original pixel. In this case the coefficients would normally be programmed to {0,1,0,0} * 2^6 (for location x = 0), allowing the pixel to pass through with its value unchanged. The same principle is applied to filtering in the vertical (or y) direction.

maebe_scfhalut(N) **Offset = 0x0100+4N**

maebe_scfvalut(N) **Offset = 0x0180+4N**

maebe_scfhblut(N) **Offset = 0x0200+4N**

maebe_scfvblut(N) **Offset = 0x0280+4N**

maebe_scfhclut(N) **Offset = 0x0300+4N**

maebe_scfvclut(N) **Offset = 0x0380+4N**

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
	CF3								CF2								CF1								CF0							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:24	CF3	Coefficient 3 Entry N where: CF3 = two's complement coefficient tap3 * 2^6	R/W	0
23:16	CF2	Coefficient 2 Entry N where: CF2 = two's complement coefficient tap2 * 2^6	R/W	0
15:8	CF1	Coefficient 1 Entry N where: CF1 = two's complement coefficient tap1 * 2^6	R/W	0
7:0	CF0	Coefficient 0 Entry N where: CF0 = two's complement coefficient tap0 * 2^6	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.3.5.2 Color Space Converter (CSC) Registers

The color space converter (CSC) registers are shown in Table 6-8.

Table 6-8. CSC Registers

Offset (Note1)	Register Name	Description
0x0400	maebe_cscxcffa	X Coefficient A
0x0404	maebe_cscxcffb	X Coefficient B
0x0408	maebe_cscxcffc	X Coefficient C
0x040C	maebe_cscycffa	Y Coefficient A
0x0410	maebe_cscycffb	Y Coefficient B
0x0414	maebe_cscycffc	Y Coefficient C
0x0418	maebe_csczcffa	Z Coefficient A
0x041C	maebe_csczcffb	Z Coefficient B
0x0420	maebe_csczcffc	Z Coefficient C
0x0424	maebe_cscxoff	X Offset
0x0428	maebe_cscyoff	Y Offset
0x042C	maebe_csczoff	Z Offset
0x0430	maebe_cscalpha	Alpha Value

Note1. See Table 6-6 on page 178 for base address.

CSC Coefficient Registers

Coefficients are used in colorspace conversion calculations. The coefficient registers should be programmed according to the desired input and output video colorspaces. These coefficients correspond directly to those used in the formulae listed in Section 6.3.2.1 "MAE CSC Coefficients and Offsets" on page 172.

NOTE: CSC coefficient must fall between the range of [-2.5,2.5] (inclusive). Thus, CFF must be between [-2560,2560] (inclusive).

maebe_cscxcffa **Offset = 0x0400**

maebe_cscxcffb **Offset = 0x0404**

maebe_cscxcffc **Offset = 0x0408**

maebe_cscycffa **Offset = 0x040C**

maebe_cscycffb **Offset = 0x0410**

maebe_cscycffc **Offset = 0x0414**

maebe_csczcffa **Offset = 0x0418**

maebe_csczcffb **Offset = 0x041C**

maebe_csczcffc **Offset = 0x0420**

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
																				CFS	CFF											
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:13	—	Reserved.	R	0
12	CFS	The sign bit for the coefficient. Note this bit defines the polarity of the 12-bit unsigned CFF field; it is not combined to form a 13-bit signed integer value.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
11:0	CFF	The magnitude of the coefficient where: CFF = coefficient * 2^{10}	R/W	0

CSC Offset Registers

Offset constants are used in colorspace conversion calculations. The offset registers should be programmed according to the desired input and output video colorspaces. These offsets correspond directly to those used in the formulae listed in Section 6.3.2.1 "MAE CSC Coefficients and Offsets" on page 172.

maebe_cscxoff **Offset = 0x0424**

maebe_cscyoff **Offset = 0x0428**

maebe_csczoff **Offset = 0x042C**

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
																			OFF													
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

Bits	Name	Description	Read/Write	Default
31:14	—	Reserved.	R	0
13:0	OFF	Two’s complement offset data where: OFF = offset * 2^4	R/W	0

CSC Alpha Value Register

This register is used to program the alpha value of the output frame. This alpha value can be used by the LCD controller (or off-chip controller) to alpha-blend the video frame with the LCD frame buffer. The Alpha Value register is doubled buffered like the other configuration registers and as such, can only be changed on a per frame basis (no per-pixel alpha values).

maebe_cscalpha **Offset = 0x0430**

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
																									ALPHA							
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

Bits	Name	Description	Read/Write	Default
31:8	—	Reserved.	R	0
7:0	ALPHA	Alpha Value. The 8-bit value is used with the 24-bit + alpha output mode. Bit 7 is used for the 15-bit + alpha output mode.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.3.5.3 Source (SRC) Registers

The source (SRC) registers are shown in Table 6-9.

Table 6-9. SRC Registers

Offset (Note1)	Register Name	Description
0x0500	maebe_srccfg	SRC Configuration Register
0x0504	maebe_srcfhw	SRC Frame Height/Width Register
0x0508	maebe_srcaaddr	SRC A Starting Address Register
0x050C	maebe_srcastr	SRC A Line Stride Register
0x0510	maebe_srcbaddr	SRC B Starting Address Register
0x0514	maebe_srcbstr	SRC B Line Stride Register
0x0518	maebe_srccaddr	SRC C Starting Address Register
0x051C	maebe_srccstr	SRC C Line Stride Register

Note1. See Table 6-6 on page 178 for base address.

SRC Configuration Register

This register contains the bits necessary to configure the input data path.

maebe_srccfg Offset = 0x0500

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
																						EF	BE	BM		ILE	ILM		IF			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:11	--	Reserved.	--	0
10	EF	Endian Format. The back end processes data in Big Endian format. This bit describes the format of data in the source memory. 0 Big Endian. 1 Little Endian.	R/W	0
9	BYE	Bayer Input Enable. Set this bit to enable the selected Bayer input mode. This setting overrides the Input Format (IF field) setting. In Bayer mode channels are subsampled as follows: Channel A (R Data) is 1/2 height and 1/2 width of the input frame. Channel B (G Data) is full height and 1/2 width of the input frame. Channel C (B Data) is 1/2 height and 1/2 width of the input frame.	R/W	0
8:7	BM	Bayer Input Mode. 00 first row (RGRGRG...) second row (GBGBGB...) 01 first row (GRGRGR...) second row (BGBGBG...) 10 first row (BGBGBG...) second row (GRGRGR...) 11 first row (GBGBGB...) second row (RGRGRG...)	R/W	0
6	ILE	Interleave Enable. 0 Non-interleaved mode. 1 Interleaved mode.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
5:4	ILM	Interleaved Mode Formats. 00 UYVY 01 VYUY 10 YUYV 11 YVYU	R/W	0
3:2	IF	Input Format (ignored for Bayer mode). 00 4:2:0 01 4:2:2 10 4:1:1 11 4:4:4	R/W	0
1:0	--	Reserved.	--	0

SRC Frame Height/Width Register

This register contains the height (in lines) and width (in bytes/pixels) dimensions for a frame of data. Note: These dimensions must always reflect the full-frame pixel dimensions of the input frame size despite any subsampled channels. Some or all of the channels may be subsampled and thus have dimensions that do not correlate 1:1 with the frame's dimension. The subsampled channels are defined by the input format setting in the SRC Configuration register.

maebe_srcfhw **Offset = 0x0504**

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
	▨	▨	▨	▨	▨	H											▨	▨	▨	▨	▨	W										
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:27	--	Reserved.	--	0
26:16	H	Full-Frame image data height (in lines). Max supported height = 1024	R/W	0
15:11	--	Reserved.	--	0
10:0	W	Full-Frame image data width (in bytes/pixels). Max supported width = 1024	R/W	0

SRC A Starting Address Register

This register provides the starting address for A image data. To take advantage of cache line bursting, starting addresses should be cache line aligned (i.e., address bits[4:0] should be programmed to 0).

maebe_srcaaddr **Offset = 0x0508**

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

Bits	Name	Description	Read/Write	Default
31:0	ADDR	The starting address of A image data.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

SRC A Line Stride Register

This register provides the stride length (in bytes) for A image data. The stride length is the distance (in bytes) from the start of one line to the start of the next. Stride values must be equal to or greater than the channel A input frame width (in bytes).

maebe_srcastr **Offset = 0x050C**

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
																							STR									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:10	—	Reserved.	R	0
9:0	STR	The line stride in bytes for A image data.	R/W	0

SRC B Starting Address Register

This register provides the starting address for B image data. To take advantage of cacheline bursting, starting addresses should be cacheline aligned (i.e., address bits[4:0] should be programmed to 0).

maebe_srcbaddr **Offset = 0x0510**

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

Bits	Name	Description	Read/Write	Default
31:0	ADDR	The starting address of B image data.	R/W	0

SRC B Line Stride Register

This register provides the stride length (in bytes) for B image data. The stride length is the distance (in bytes) from the start of one line to the start of the next. Stride values must be equal to or greater than the channel B input data width (in bytes).

maebe_srcbstr **Offset = 0x0514**

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
																							STR									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:10	—	Reserved.	R	0
9:0	STR	The line stride in bytes for B image data.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

SRC C Starting Address Register

This register provides the starting address for C image data. To take advantage of cacheline bursting, starting addresses should be cacheline aligned (i.e., address bits[4:0] should be programmed to 0).

maebe_srccaddr **Offset = 0x0518**

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

Bits	Name	Description	Read/Write	Default
31:0	ADDR	The starting address of C image data.	R/W	0

SRC C Line Stride Register

This register provides the stride length (in bytes) for C image data. The stride length is the distance (in bytes) from the start of one line to the start of the next. Stride values must be equal to or greater than the channel C input data width (in bytes).

maebe_srccstr **Offset = 0x051C**

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
	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	STR									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:10	—	Reserved.	R	0
9:0	STR	The line stride in bytes for C image data.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.3.5.4 Destination (DST) Registers

The destination (DST) registers define the settings for the output frame, and are shown in Table 6-10.

Table 6-10. DST Registers

Offset (Note1)	Register Name	Description
0x0600	maebe_dstcfg	DST Configuration Register
0x0604	maebe_dstheight	DST Frame Height Register
0x0608	maebe_dstaddr	DST Starting Address Register
0x060C	maebe_dststr	DST Stride Register

Note1. See Table 6-6 on page 178 for base address.

DST Configuration Register

This register contains the bits necessary to control the output of the destination frame data.

maebe_dstcfg **Offset = 0x0600**

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
																									R	ROT		BGR	OF		EF	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:8	--	Reserved.	--	0
7	R	Pixel Value Round Setting. This setting controls the rounding parameters for the CSC output. The default setting is to constrain the output value to the range of 0.255 and to truncate the result as necessary to fit in this range. If this bit is set to 1, then the output values are constrained to a range of -128.127. This setting should be used only when the output color space has values that land in this range. 0 output value range 0..255 1 output value range -128..127	R/W	0
6:5	ROT	Video Rotate. 0 Non-rotated. 1 Rotate video 90 degrees clockwise. 2 Rotate video 180 degrees clockwise. 3 Rotate video 270 degrees clockwise.	R/W	0
4	BGR	BGR(α) Enable. 0 (α)RGB mode. 1 BGR(α) mode.	R/W	0
3:2	OF	Output Format. 00 24-bit (8, 8, 8.) + α (32–bits per pixel). 01 Reserved 10 16-bit (5, 6, 5). 11 15-bit (5, 5, 5) + α (16-bits per pixel).	R/W	0
1	EF	Endian Format. This bit designates the endian format for destination memory. 0 Big Endian. 1 Little Endian.	R/W	0
0	--	Reserved.	--	0

www.DataSheet4U.com

www.DataSheet4U.com

DST Frame Height Register

This register contains the height (number of lines) for an output frame of XYZ data. This is the frame height of the unrotated output image.

Note: The MAX supported output frame STRIDE is 1024 pixels (or 2048 bytes in 16-bpp mode and 4096 bytes in 32-bpp mode). This implies that the MAX supported output frame WIDTH is 1024 pixels, since the frame width is always less than or equal to the frame stride. The MAX supported output frame HEIGHT is 1024 pixels. The MAX supported FRAME_SIZE is 1,048,576 pixels.

NOTE: The MIN supported output frame HEIGHT is 10 lines.

maebe_dstfh **Offset = 0x0604**

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
																						FH										
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:11	—	Reserved.	R	0
10:0	FH	XYZ image data frame height (number of lines).	R/W	0

DST Starting Address Register

This register contains the starting address for an output frame of XYZ image data. The starting address depends on which of the four possible video rotate modes is being used. The starting address should always be the address of the first pixel output.

maebe_dstaddr **Offset = 0x0608**

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

Bits	Name	Description	Read/Write	Default
31:0	ADDR	The starting address of XYZ image data.	R/W	0

DST Stride Register

This register contains the stride line length (in bytes) for an output frame of XYZ image data. The DST Stride value must always be greater than or equal to the output frame width (after rotation if applicable). Setting the stride length greater than the width of the output frame in this register effectively adds padding between lines.

To take advantage of cacheline bursting, stride values should be multiples of cachelines (32 bytes) (i.e., stride bits[4:0] should be programmed to 0).

maebe_dststr **Offset = 0x060C**

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
																				STR												
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:11	—	Reserved.	R	0
12:0	STR	The line stride (in bytes) for XYZ image data.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

6.3.5.5 Control (CTL) Registers

The control (CTL) registers are shown in Table 6-11.

Table 6-11. CTL Registers

Offset (Note1)	Register Name	Description
0x0700	maebe_ctlenable	CTL Enable Register
0x0704	maebe_ctlfpc	CTL Frame Processing Control Register
0x0708	maebe_ctlstat	CTL Status Register
0x070C	maebe_ctlintenable	CTL Interrupt Registers
0x0710	maebe_ctlintstat	CTL Interrupt Registers

Note1. See Table 6-6 on page 178 for base address.

CTL Enable Register

This register is used to enable/disable the MAE back end and to reset the back end. The enable bit defaults to a 0 from power-on reset. At this point writing a 1 to the enable bit location resets the back end and then enables the back end for configuration and use. To reset the back end once it has been enabled, simply write a 0 to the enable bit location and then write a 1 to re-enable. This action resets all configuration registers and clears the entire data path.

maebe_ctlenable **Offset = 0x0700**

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

Bits	Name	Description	Read/Write	Default
31:1	--	Reserved.	--	0
0	EN	MAE Back End Enable. 0 Disable back end. 1 Enable the back end.	R/W	0

www.DataSheet4U.com

w w w . D a t a S

CTL Frame Processing Control Register

This register is used to start, stop and reset the back end. This is a self clearing register. Writing a 1 to a control bit activates the corresponding command (writing a 0 has no effect). The control bit location then clears after one clock cycle. Due to the self-clearing nature of this register, it is always read as zeros.

Note: The Frame Start control bit should only be activated when the Frame Pending status bit is zero, otherwise it is ignored. The Reset control bit can be activated at any time.

maebe_ctlfpc

Offset = 0x0704

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
																															FRST	STR
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:2	--	Reserved.	--	0
1	FRST	Frame Reset. Setting this control bit stops the current frame (if there is one), cancels a pending frame (if there is one), resets the back end status and clears the data path. The configuration registers are unchanged. The back end can then be reprogrammed and a new frame can be started. This bit is always read as 0.	W	0
0	STR	Frame Start. Setting this control bit validates the SCF/CSC/SRC/DST configurations and sets the Frame Pending status bit. When the current frame (if there is one) is done processing, the pending frame begins processing as the new current frame. If there is not a current frame, the pending frame immediately begins processing as the new current frame. In either case, once the pending frame becomes the current frame, the Frame Pending status bit is cleared. This bit is self clearing and is always read as 0.	W	0

CTL Status Register

This register reflects conditions within the back end. When a status event becomes true, the corresponding status bit is set and remains set until the event is no longer true. The Status Register is a read only register.

maebe_ctlstat

Offset = 0x0708

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
																															FP	FB
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:2	--	Reserved.	--	0
1	FP	Frame Pending. This flag indicates that the next frame has been configured and the Frame Start bit has been set. The next frame is pending.	R	0
0	FB	Frame Busy. This flag indicates that the current frame is being processed.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

CTL 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, an interrupt is issued and the corresponding bit in the Interrupt Status register is set. Interrupts can be cleared by writing a 1 to the corresponding bit in the Interrupt Status register. Writing a 0 to the Interrupt Status register has no effect. These registers may be read and written while the SCF is active.

maebe_ctlintenable **Offset = 0x070C**

maebe_ctlintstat **Offset = 0x0710**

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
																																FC
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:1	--	Reserved.	--	0
0	FC	Frame Complete. This flag indicates that the current frame has been completed.	R/W	0

6.3.5.6 Debug (DBG) Registers

These registers (shown in Table 6-12) are for debugging by giving software access to all of the storage elements in the back end. By writing to a FIFO, software can push data onto the FIFO. Care must be taken not to overflow the FIFO. By reading from a FIFO, software can pop data from the FIFO. Care must be taken not to underflow the FIFO. All FIFOs are 16 entries deep and 64 bits wide. The RAMs can be accessed by writing or reading the appropriate offset + entry (i.e., to write the SCF A Vertical RAM entry 10, with entry 0 being the first entry, software would write to address offset 0x090A). The RAMs are 80 entries deep and 32 bits wide.

Table 6-12. DBG Registers

Offset (Note1)	Register Name	Description
0x0800	maebe_dbgscfafifo	SCF A Input FIFO
0x0804	maebe_dbgscfbfifo	SCF B Input FIFO
0x0808	maebe_dbgscfcfifo	SCF C Input FIFO
0x080C	maebe_dbgcscfifo	CSC Output FIFO
0x0900	maebe_dbgscfaram	SCF A Vertical RAM
0x0B00	maebe_dbgscfbram	SCF B Vertical RAM
0x0D00	maebe_dbgscfcram	SCF C Vertical RAM

Note1. See Table 6-6 on page 178 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

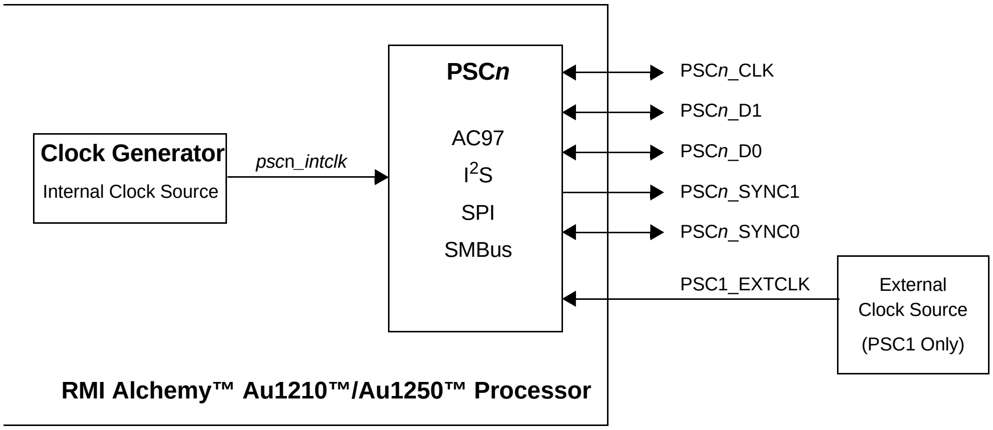
www.DataSheet4U.com

7 Programmable Serial Controllers (PSCs)

The Au1210™ and Au1250™ processors contain two programmable serial controllers (PSCs). Each PSC is a multi-protocol serial controller that can be configured for operation in one of four protocols:

- Serial Peripheral Interface (SPI) (see page 196)
- Inter-IC Sound (I^2S) (see page 207)
- Audio Codec-97 Controller (AC97) (see page 220)
- System Management Bus (SMBus) (see page 234)

Figure 7-1 shows a block diagram for a PSC.



Not all signals are used for each protocol. See Section 7.6 "PSC Signal Mapping" on page 248 for the protocol-specific signals.

Figure 7-1. Programmable Serial Controller (PSC) Block Diagram

www.DataSheet4U.com

www.DataSheet4U.com

7.1 PSC Registers

Each PSC is controlled by a separate set of registers. The base addresses for each PSC are listed in Table 7-1.

Table 7-1. PSC Base Addresses

PSC	Physical Base Address	KSEG1 Base Address
PSC0	0x0 11A0 0000	0xB1A0 0000
PSC1	0x0 11B0 0000	0xB1B0 0000

Table 7-2 shows the PSC select register (**psc_sel**) and the PSC control register (**psc_ctrl**). The register formats for **psc_sel** and **psc_ctrl** remain the same regardless of which serial protocol is selected. The protocol-specific registers are available only after enabling the PSC and are described in their respective protocol sections.

Table 7-2. PSC Registers

Offset (Note1)	Register Name	R/W	Description
0x0000	psc_sel	R/W	PSC Select Register
0x0004	psc_ctrl	R/W	PSC Control Register
0x0008 - 0x002F	—	—	Protocol-specific registers

Note1. See Table 7-1 for base address.

7.1.1 PSC Select Register

This register selects the protocol and the clock source for the PSC internal main clock (*psc*n*_mainclk*). The clock source options for *psc*n*_mainclk* depend on which protocol the PSC is running; see the individual protocol sections for more information on the clocking options. Program **psc_sel** *before* enabling the PSC in **psc_ctrl**[ENA].

psc_sel **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
																											CLK			PS		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:6	-	Reserved.	-	0
5:4	CLK	PSC Clock Source Selection. Select the PSC clock source (*psc*n*_mainclk*) for the PSC. The *psc*n*_mainclk* signal is an internal signal used to generate clocks within the PSC. 00 Select the internal *psc*n*_intclk* signal from the internal clock generator. See Section 10.1 "Clocks" on page 366. This option is available for SPI and SMBus. Also available for I^{2}S Master. 01 Select the external PSC1_EXTCLK input clock. Available for PSC1 only. This option is available for SPI, SMBus, and I^{2}S Master. 10 Select the external PSC*n*_CLK serial clock. This option is available for AC97 and I^{2}S Slave. 11 Reserved	R/W	0
3	-	Reserved.	-	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
2:0	PS	Protocol Select. Select the operating protocol for the PSC. 000 Protocol disabled 001 Reserved 010 SPI mode 011 I^{2}S mode 100 AC97 mode 101 SMBus mode 11x Reserved	R/W	0

7.1.2 PSC Control Register

This register allows software to reset the PSC, to enable normal operation, and to place the PSC into a suspended state to conserve power.

psc_ctrl **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:2	-	Reserved.	-	0
1:0	ENA	Enable. 00 Disable/reset the PSC. 01 Reserved 10 Suspend operation. To conserve power while idle, the PSC can be suspended. In this state, clocks are disabled but FIFO data and registers are preserved. 11 Enable the PSC. The PSC is not immediately ready: After enabling the PSC, sample the serial-ready bit (**psc*n*_spistat**[SR], **psc*n*_i2sstat**[SR], **psc*n*_ac97stat**[SR], or **psc*n*_smbstat**[SR]) to determine when the PSC is ready for protocol configuration and normal operation.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.cc

7.2 Serial Peripheral Interface (SPI) Controller

When the SPI protocol is selected for a PSC (**psc_sel**[PS] = 0b010), the PSC becomes an SPI controller. The SPI controller supports the following features:

- Four signal interface (SPICLK, SPISEL#, SPIMISO, SPIMOSI)
- Full duplex operation
- Supports data characters from 4 to 24 bits long
- Supports back-to-back character transfers
- Master and Slave SPI modes supported
- Multiple Master environment support
- Supports transfers using the DDMA controller
- Master SPI mode supports automatic slave select
- Supports clock rates up to 24MHz
- Independent programmable baud rate generator (in Master mode)
- Programmable clock phase and polarity
- Open-drain outputs support multiple master connections
- Local loopback capability for testing

7.2.1 SPI Registers

Table 7-3 shows the SPI registers.

Table 7-3. SPI Registers

Offset (Note1)	Register	R/W	Description
0x0000	psc_sel	R/W	PSC Select (See Section 7.1.1 "PSC Select Register" on page 194)
0x0004	psc_ctrl	R/W	PSC Control (See Section 7.1.2 "PSC Control Register" on page 195)
0x0008	psc_spicfg	R/W	SPI Configuration Register
0x000C	psc_spimsk	R/W	SPI Mask Register
0x0010	psc_spipcr	R/W	SPI Protocol Control Register
0x0014	psc_spistat	R	SPI Status Register
0x0018	psc_spievnt	R/W	SPI Event Register
0x001C	psc_spitxrx	R/W	SPI Tx/Rx Data Register

Note1. See Table 7-1 on page 194 for base address.

7.2.1.1 SPI Configuration Register

This register configures the operational parameters for the SPI protocol. While the device enable bit is set, do not modify any fields other than **psc_spicfg**[DE].

psc_spicfg **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
	RT		TT		DD	DE	▨					BRG						DIV		BI	PSE	CGE	CDE	LEN					LB	▨	MLF	MO
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:30	RT	Rx FIFO Threshold. 00 1 data entry 01 2 data entries 10 4 data entries 11 8 data entries For DMA transfers (**psc_spicfg**[DD] = 0), the threshold must match the source transfer size (STS) programmed in the descriptor. For programmed I/O (**psc_spicfg**[DD] = 1), once the number of data entries rises to the threshold, a *receive request* (RR) is triggered in the event and status registers.	R/W	0
29:28	TT	Tx FIFO Threshold. 00 1 empty slot 01 2 empty slots 10 4 empty slots 11 8 empty slots For DMA transfers (**psc_spicfg**[DD] = 0), the threshold must match the destination transfer size (DTS) programmed in the descriptor. For programmed I/O (**psc_spicfg**[DD] = 1), once the number of empty FIFO slots matches the threshold, a *transmit request* (TR) is triggered in the event and status registers.	R/W	0
27	DD	Disable DMA Transfers. 0 Enable DMA transfers. Software prepares buffers and descriptors, and the DDMA controller handles the transfers. 1 Programmed I/O. Software handles each transfer directly by using the data register and monitoring the FIFO events.	R/W	0
26	DE	Device Enable. 0 Disable the SPI controller. 1 Enable the SPI controller. Do not modify any other fields in **psc_spicfg** while the SPI controller is enabled (**psc_spistat**[DR] = 1).	R/W	0
25:21	-	Reserved.	-	0
20:15	BRG	Baud Rate Generator. The BRG controls the frequency of SPICLK for Master mode transfers. The BRG is set in terms of *pscn_tempclk*, which is derived from *pscn_mainclk*. See Section 7.2.2.1 "SPI Clocking" on page 202. Valid BRG values range from 4-63. The bit clock frequency is calculated as shown below: SPICLK = *pscn_tempclk* / (2 * (BRG + 1))	R/W	0
14:13	DIV	PSC Clock Divider. The DIV setting controls the frequency of the internal clock *pscn_tempclk*. The resulting frequency is based on the Main Clock (*pscn_mainclk*) input source frequency: See Section 7.2.2.1 "SPI Clocking" on page 202. 00 *pscn_tempclk* = *pscn_mainclk* / 2 01 *pscn_tempclk* = *pscn_mainclk* / 4 10 *pscn_tempclk* = *pscn_mainclk* / 8 11 *pscn_tempclk* = *pscn_mainclk* / 16	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
12	BI	Bit Clock Invert. 0 SPICLK begins toggling rising edge first. 1 SPICLK begins toggling falling edge first.	R/W	0
11	PSE	Port Swap Enable. Applies to slave mode only. 0 Disable port swapping. 1 Enable port swapping to support multiple master environments. Setting this bit causes the data ports to switch direction: SPIMOSI becomes an output and SPIMISO becomes an input.	R/W	0
10	CGE	Clock Gate Enable. 0 Normal operation. 1 Gate (remove) the last SPICLK cycle of a byte transmission.	R/W	0
9	CDE	Clock Phase-delay Enable. 0 Do not delay the SPICLK phase. 1 Delay SPICLK by one phase at the start of a transfer.	R/W	0
8:4	LEN	Data Length. Contains the size (in bits) of the SPI transfer data. The actual data length must range between 4 and 24 bits. (LEN + 1) is the actual data length.	R/W	0
3	LB	Loopback Mode. Applies to master mode only. 0 Normal operation 1 Loopback mode: internally connect the Tx and Rx FIFOs.	R/W	0
2	-	Reserved.	-	0
1	MLF	MSb/LSb Data First. Selects the bit ordering for data transfers. 0 MSb first 1 LSb first	R/W	0
0	MO	Master/slave Mode. 0 Enable master mode only. Disable slave mode. 1 Enable master/slave mode.	R/W	0

7.2.1.2 SPI Mask Register

This register is used to mask the interrupts triggered by the SPI event register (**psc_spievnt**) flags. Setting a mask bit disables the corresponding interrupt. All interrupts not needed by an application should be masked.

For predictable interrupt behavior, **psc_spimsk** should not be modified during an SPI transfer.

psc_spimsk **Offset = 0x000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																MM			RR	RO	RU	TR	TO	TU			SD	MD				
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

www.DataSheet4U.com

www.DataSheet4U.c

Bits	Name	Description	R/W	Default
31:17	-	Reserved.	-	0
16	MM	Mask multiple master error interrupt.	R/W	0
15:14	-	Reserved.	-	0
13	RR	Mask Rx FIFO request interrupt.	R/W	0
12	RO	Mask Rx FIFO overflow interrupt.	R/W	0
11	RU	Mask Rx FIFO underflow interrupt.	R/W	0
10	TR	Mask Tx FIFO request interrupt.	R/W	0
9	TO	Mask Tx FIFO overflow interrupt.	R/W	0
8	TU	Mask Tx FIFO underflow interrupt.	R/W	0
7:6	-	Reserved.	-	0
5	SD	Mask slave-done interrupt.	R/W	0
4	MD	Mask master-done interrupt.	R/W	0
3:0	-	Reserved.	-	0

7.2.1.3 SPI Protocol Control Register

This register is used to control transfers and to clear the Rx and Tx FIFOs. Setting a control bit issues a command to the controller. The bit is automatically cleared once the command is initiated but not necessarily completed.

psc_spipcr **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:7	-	Reserved.	-	0
6	RC	Rx Data Clear. Setting RC while the SPI is idle clears the Rx FIFO.	R/W	0
5	SP	Slave Stop. Setting SP disables slave transfers.	R/W	0
4	SS	Slave Start. Setting SS enables the SPI slave controller to respond to slave transfers.	R/W	0
3	-	Reserved.	-	0
2	TC	Tx Data Clear. Setting TC while the SPI controller is idle clears the Tx FIFO.	R/W	0
1	-	Reserved.	-	0
0	MS	Master Start. Setting MS while the SPI controller is idle causes a master transfer to begin.	R/W	0

7.2.1.4 SPI Status Register

This register contains the status bits for the SPI protocol. It is read only. A status condition is true when the status bit is set. Before the SPI controller is enabled in **psc_spicfg**[DE], only the Device Ready and PSC Ready status bits (**psc_spistat**[DR, SR]) are valid; all other status bits read as 0 until ready.

psc_spistat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			RF	RE	RR	TF	TE	TR			SB	MB		DI	DR	SR
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:14	-	Reserved.	-	0
13	RF	Receive FIFO full.	R	0
12	RE	Receive FIFO empty.	R	1
11	RR	Receive request.	R	0
10	TF	Transmit FIFO full.	R	0
9	TE	Transmit FIFO empty.	R	1
8	TR	Transmit request.	R	0
7:6	-	Reserved.	-	0
5	SB	Slave Busy. Indicates a slave SPI transfer is currently in progress.	R	0
4	MB	Master Busy. Indicates a master SPI transfer is currently in progress.	R	0
3	-	Reserved.	-	0
2	DI	Device Interrupt. Indicates at least one unmasked event has been flagged in **psc_spievnt**.	R	0
1	DR	Device Ready. Indicates the SPI controller is ready for protocol operation after setting the device-enable bit **psc_spicfg**[DE].	R	0
0	SR	PSC Ready. Indicates the PSC is ready for protocol configuration after enabling the PSC in **psc_ctrl**[ENA].	R	0

7.2.1.5 SPI Event Register

This register contains the event flags for the SPI protocol. These events generate an interrupt unless masked in the SPI Mask register (**psc_spimsk**).

Once an event flag is set, it remains set until it is cleared. A flag is cleared by writing a 1 to the appropriate bit; writing a 0 has no effect.

If multiple-master error, receive overflow, or transmit underflow occurs, the following must take place:

1) Clear the interrupt in **psc_spievnt**.

2) Clear the Rx and Tx FIFOs in **psc_spipcr**.

3) Restart transfers as shown in the SPI flowcharts; see Section 7.2.2.2 "SPI Flowcharts" on page 202.

psc_spievnt **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																MM			RR	RO	RU	TR	TO	TU			SD	MD				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:17	-	Reserved.	-	0
16	MM	Multiple Master Error. Set when the SPI is in master mode and the slave select signal is asserted, indicating another master is trying to control the SPI bus.	R/W	0
15:14	-	Reserved.	-	0
13	RR	Receive Request. Applies to programmed I/O only (**psc_spicfg**[DD] = 1). Indicates the Rx FIFO has a number of data elements equal to the FIFO threshold programmed in **psc_spicfg**[RT] available for reading.	R/W	0
12	RO	Receive Overflow. Indicates the Rx FIFO has experienced an overflow.	R/W	0
11	RU	Receive Underflow. Indicates the Rx FIFO has experienced an underflow.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
10	TR	Transmit Request. Applies to programmed I/O only (**psc_spicfg**[DD] = 1). Indicates the Tx FIFO is requesting a number of data elements equal to the FIFO threshold programmed in **psc_spicfg**[TT].	R/W	0
9	TO	Transmit Overflow. Indicates the Tx FIFO has experienced an overflow.	R/W	0
8	TU	Transmit Underflow. Indicates the Tx FIFO has experienced an underflow.	R/W	0
7:6	-	Reserved.	-	0
5	SD	Slave Done. Indicates a SPI Slave transfer has completed.	R/W	0
4	MD	Master Done. Indicates a SPI Master transfer has completed.	R/W	0
3:0	-	Reserved.	-	0

7.2.1.6 SPI Tx/Rx Data Register

This register is the interface to the transmit and receive FIFOs. Each FIFO is 16 entries deep. A write to this register causes data to be pushed onto the Tx FIFO. If the Tx FIFO is already full, the data is lost and a transmit-overflow event is triggered (**psc_spievnt**[TO] = 1). A read from this register causes data to be popped from the Rx FIFO. If the Rx FIFO is already empty, the data read is undefined and a receive-underflow event is triggered (**psc_spievnt**[RU] = 1).

If transfers are under program control (**psc_spicfg**[DD] = 1), software can monitor the transmit-request (TR) and receive-request (RR) flags to maintain proper FIFO levels either by polling **psc_spistat**[TR, RR] or by waiting for interrupts in **psc_spievnt**[TR, RR]. The DDMA controller (**psc_spicfg**[DD] = 0) automatically manages overflow/underflow conditions.

The data size of read/write transfers is always 32-bits to the register; however, depending on the data length value (**psc_spicfg**[LEN]) all unused bits must be 0s.

When DMA transfers are enabled, the **psc_spitxrx** address locations must be programmed into the Tx descriptor destination pointer and the Rx descriptor source pointer.

psc_spitxrx **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
			LC	ST					DATA																							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:30	—	Reserved.	-	-
29	LC	Last Character. Applies to master mode only. When not using DMA, write a 1 to LC with the last character of the current transfer.	W	0
28	ST	Slave Select Toggle. Writing a 1 to ST along with the character causes the PSC to toggle the slave select signal after the transfer of the character.	W	0
27:24	—	Reserved.	-	-
23:0	DATA	SPI Character Data. Bit 0 is the LSb.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

7.2.2 SPI Operation

7.2.2.1 SPI Clocking

Figure 7-2 shows the clocking options for the SPI master controller.

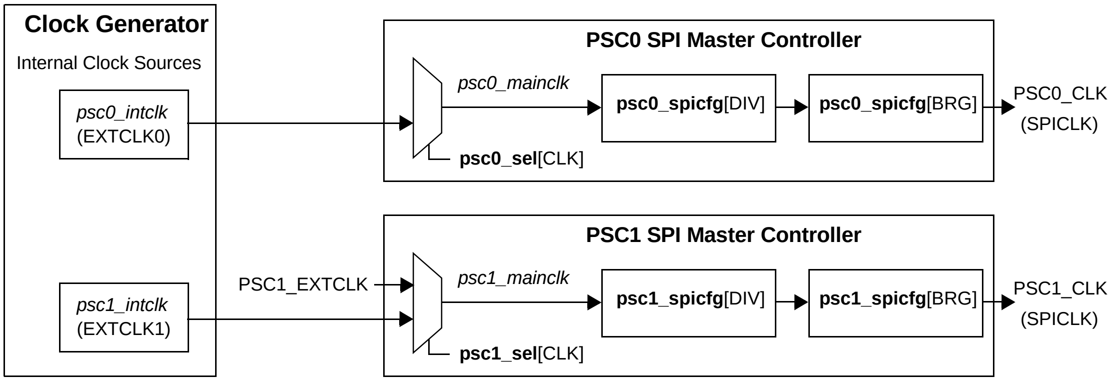


Figure 7-2. SPI Master Controller Clock Options

7.2.2.2 SPI Flowcharts

Figure 7-3 shows a configuration flowchart for the SPI controller.

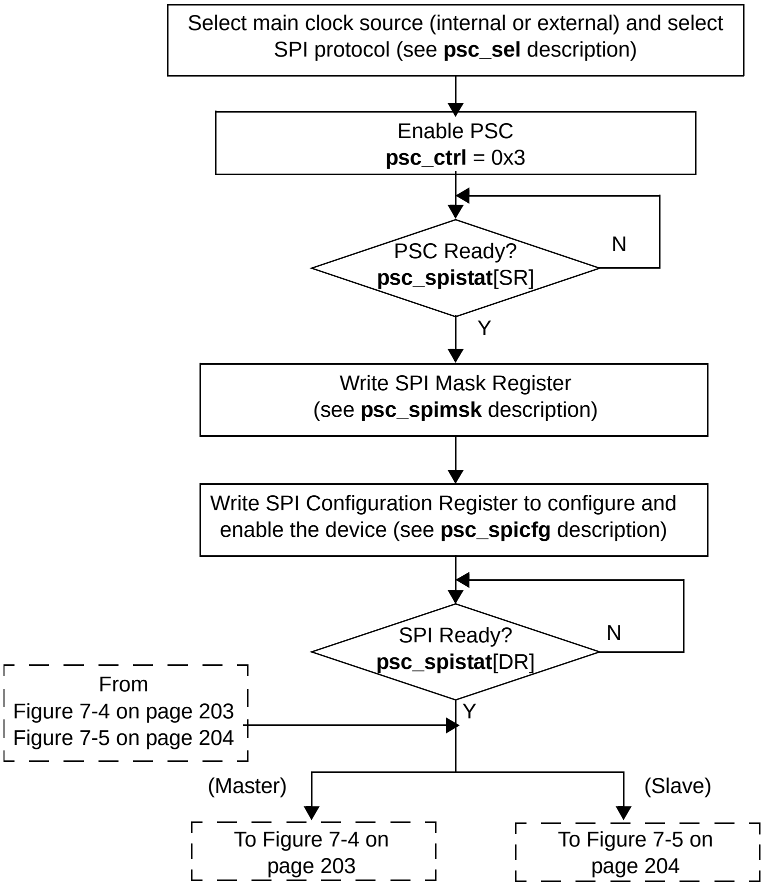
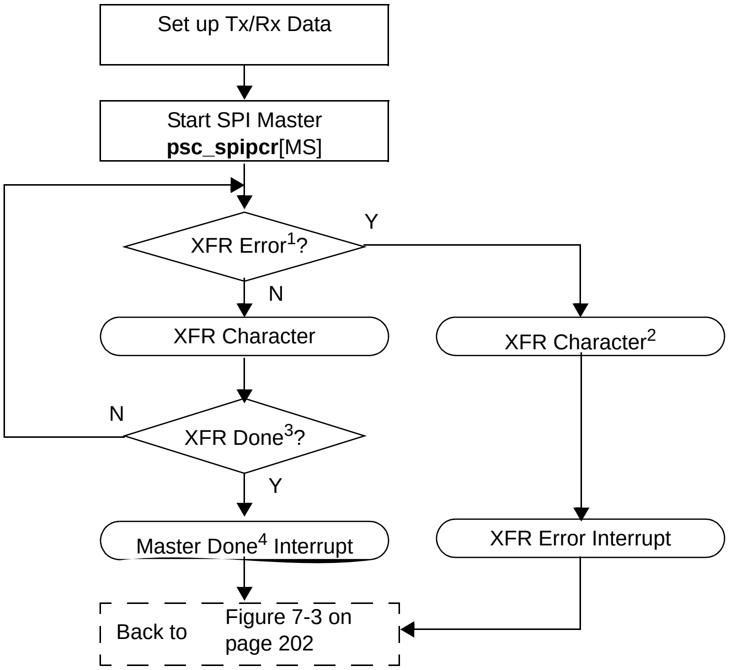

Select main clock source (internal or external) and select SPI protocol (see psc_sel description)
Enable PSC
psc_ctrl = 0x3
PSC Ready?
psc_spistat[SR]
N
Y
Write SPI Mask Register
(see psc_spimsk description)
Write SPI Configuration Register to configure and enable the device (see psc_spicfg description)
SPI Ready?
psc_spistat[DR]
N
Y
From
Figure 7-4 on page 203
Figure 7-5 on page 204
(Master)
(Slave)
To Figure 7-4 on page 203
To Figure 7-5 on page 204

Figure 7-3. SPI Configuration

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-4 shows a flowchart for SPI master operation.



[1] A XFR error interrupt occurs, unless masked, for any of the following events: Tx FIFO underflow, Rx FIFO overflow, or multiple-master error

[2] In the case of a Tx Underflow error, all zeroes are transmitted.

[3] The Master XFR is done when the last Tx character is transferred.

[4] A Master done interrupt is generated, unless masked, by the following:
DMA: When the Rx FIFO is empty
Programmed I/O: Immediately after XFR done

Figure 7-4. SPI Master Operation

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-5 shows a flowchart for SPI slave operation.

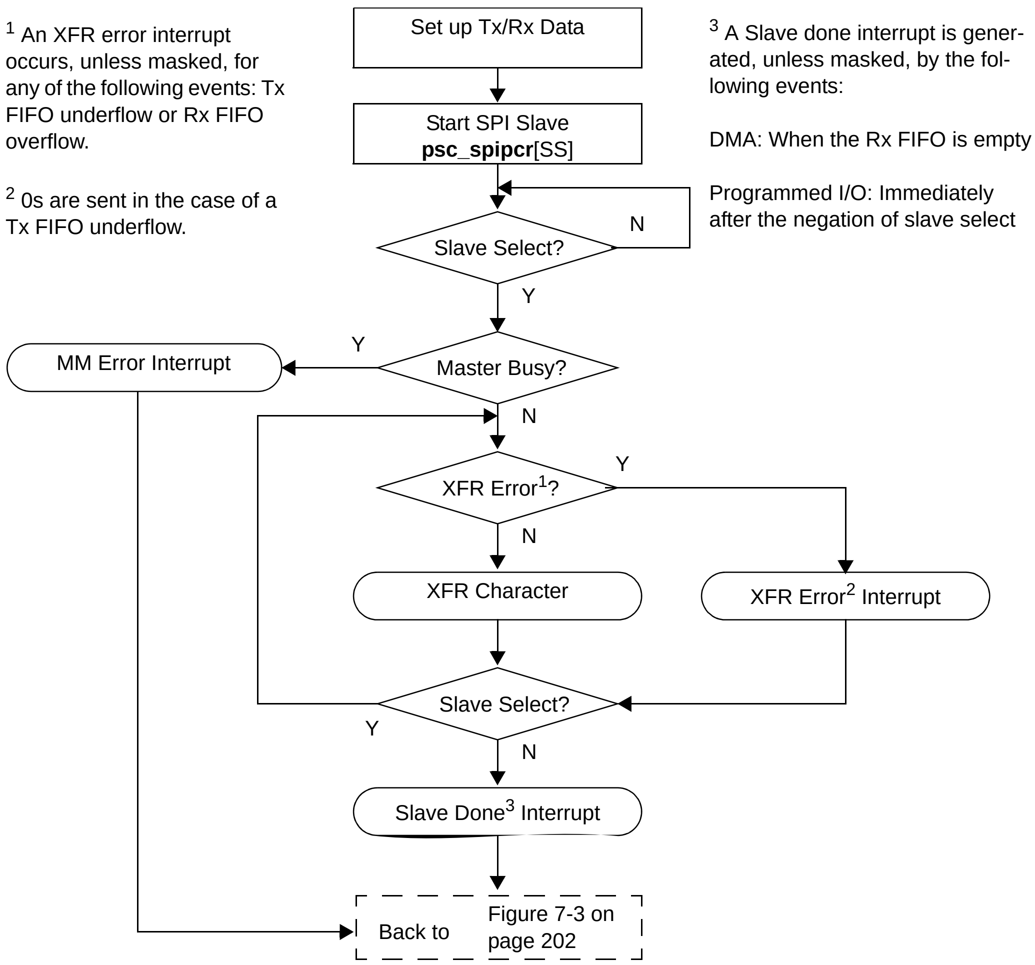

1 An XFR error interrupt occurs, unless masked, for any of the following events: Tx FIFO underflow or Rx FIFO overflow.
2 0s are sent in the case of a Tx FIFO underflow.
Set up Tx/Rx Data
Start SPI Slave
psc_spipcr[SS]
3 A Slave done interrupt is generated, unless masked, by the following events:
DMA: When the Rx FIFO is empty
Programmed I/O: Immediately after the negation of slave select
Slave Select?
N
Y
MM Error Interrupt
Y
Master Busy?
N
XFR Error1?
Y
N
XFR Character
XFR Error2 Interrupt
Slave Select?
Y
N
Slave Done3 Interrupt
Back to
Figure 7-3 on page 202

Figure 7-5. SPI Slave Operation

www.DataSheet4U.com

www.DataSheet4U.com

7.2.2.3 SPI Transfers

An SPI transfer is always full duplex even if valid data is being transferred in only one direction. The system programmer can choose to handle SPI transfers with the DDMA controller (**psc_spicfg**[DD] = 0) or with programmed I/O (**psc_spicfg**[DD] = 1).

DMA Transfers

Below is a description of how to set up the transfer descriptors and use the DDMA controller for an SPI transfer. For more information on DMA transfers, see Section 4.0 "Descriptor-Based DMA (DDMA) Controller" on page 115.

Master Mode

- For a full duplex transfer, configure a pair of Tx and Rx descriptors, both with *n* byte counts. The Tx buffer contains the valid write data payload.
- For a write only transfer, configure a Tx descriptor with *n* byte count. The Tx buffer contains the valid write data payload. Clear the Rx FIFO at the end of each transfer. Also, mask the Rx FIFO Overflow interrupt (**psc_spimsk**[RO] = 1) to prevent the SPI controller from halting.
- For a read only transfer, configure a pair of Tx and Rx descriptors, both with *n* byte counts. The Tx descriptor is for timing purposes only—the Tx byte count must match the Rx byte count. The Tx buffer is still transmitted but can overlap the Rx buffer if the device ignores the Tx data.
- With each transfer, the SPI master controller asserts and negates the slave-select signal (SPISEL#) relative to SPICLK:
  - — For **psc_spicfg**[CDE] = 0 (no clock delay), the controller asserts SPISEL# a half SPICLK cycle before the first SPICLK transition, and then negates SPISEL# one cycle after the last SPICLK transition.
  - — For **psc_spicfg**[CDE] = 1 (clock delay), the controller asserts SPISEL# one SPICLK cycle before the first SPICLK transition, and then negates SPISEL# a half cycle after the last SPICLK transition.
- The SPI controller treats each Tx descriptor as a separate transfer and toggles SPISEL# accordingly. If there are several successive valid Tx descriptors, SPISEL# is negated for a minimum of three *psc*n*_mainclk* cycles between each Tx descriptor.

Slave Mode

- For a full duplex transfer, configure pairs of Tx and Rx descriptors with sufficient total byte count for the pending transfer. The Tx buffer contains the valid write data payload. The Tx and Rx descriptors can have different byte count settings. See below for a description of how the Tx and Rx descriptors are handled differently.
- For a write only transfer, configure one (or more) Tx descriptor with sufficient total byte count for the pending transfer. The Tx buffer contains the write data payload. If the Tx descriptor remaining byte count is non-zero at the end of a slave transfer (denoted by SPISEL# negating), the Tx descriptor remains open for the next slave transfer. Also, mask the Rx FIFO Overflow interrupt (**psc_spimsk**[RO] = 1) to prevent the SPI controller from halting.
- For a read only transfer, configure one (or more) Rx descriptor with sufficient total byte count for the pending transfer. At the end of each slave transfer (denoted by SPISEL# negating), the Rx descriptor is closed (regardless of the remaining Rx descriptor byte count). If there is another slave transfer (and another valid Rx descriptor), the DDMA controller processes the new slave transfer using the next valid Rx descriptor. This allows for multiple Rx descriptors to be prepared in order to process back-to-back slave transfers. Also, mask the Tx FIFO underflow interrupt (**psc_spimsk**[TU] = 1) to prevent the SPI controller from halting.

Programmed I/O Transfers

To enable programmed I/O mode, set the DMA-disable bit (**psc_spicfg**[DD]). Below is a description of how to use the SPI controller in programmed I/O mode:

Note: If programmed I/O mode is used, care must be taken to not let the Tx FIFO underflow or the Rx FIFO overflow (or to mask the associated interrupts).

www.DataSheet4U.com

www.DataSheet4U.com

Master Mode

- Always load the Tx FIFO with *n* data elements. For a write (or full duplex) transfer the Tx FIFO contains the data payload. For a read only transfer the Tx FIFO is for timing purposes only and must be written with 0s.
- To toggle SPISEL# in the middle of a transfer, software must set **psc_spitxrx**[ST] (slave-select toggle) when writing the last character of a given transfer.
- Set **psc_spitxrx**[LC] (last character flow control) along with the last character to indicate when the PSC must terminate the Master transfer. The controller then automatically negates SPISEL# after transferring the last character.
- If a GPIO output is used instead of the SPI synchronous slave select (for running multiple slaves, for example), software must assert the GPIO slave-select before starting the SPI Master transfer (**psc_spipcr**[MS] = 1), and then negate the GPIO slave-select after the Master Done interrupt (**psc_spievnt**[MD] = 1) has occurred.

Slave Mode

- For a write (or full duplex) transfer, load the Tx FIFO with enough data elements to prevent a Tx buffer underflow. For a read only transfer, mask the Tx FIFO Underflow interrupt (**psc_spimsk**[TU] = 1) to prevent the SPI controller from halting.
- For a read (or full duplex) transfer, empty the Rx FIFO fast enough to prevent an Rx buffer overflow. For a write only transfer, mask the Rx FIFO Overflow interrupt (**psc_spimsk**[RO] = 1) to prevent the SPI controller from halting.

7.2.3 SPI Timing

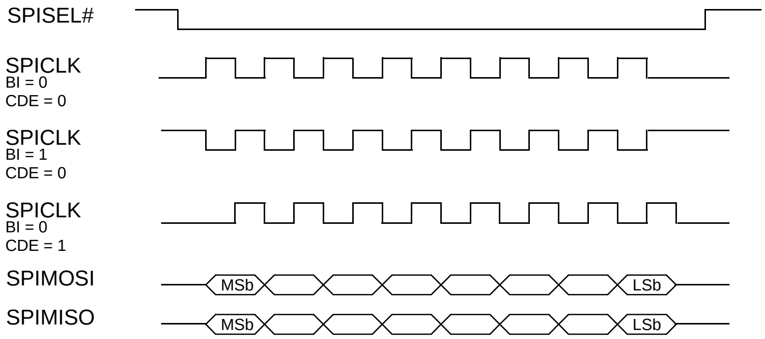


Figure 7-6. SPI Master Full Duplex Timing

7.2.4 SPI Signals

Table 7-4 shows the SPI signals.

Table 7-4. SPI Signals

Signal Name		Direction		
PSC Generic	**SPI Specific**	**Master**	**Slave**	**Termination**
PSC*n*_CLK	SPICLK	Output	Input	Pull-up or down
PSC*n*_SYNC1	SPISEL#	Output	Not used	Pull-up
PSC*n*_SYNC0	SPISEL#	Input	Input	Pull-up
PSC*n*_D1	SPIMISO	Input	Output	Pull-up
PSC*n*_D0	SPIMOSI	Output	Input	Pull-up

www.DataSheet4U.com

www.DataSheet4U.c

7.3 Inter-IC Sound Controller (I^2S)

When the I^2S protocol is selected for a PSC (**psc_sel**[PS] = 0b011), the PSC becomes an I^2S controller. The I^2S controller supports the following features:

- Five signal interface (I2SCLK, I2SMCLK, I2SWORD, I2SDO, I2SDI)
- Internally generated clock source for I2SMCLK output. PSC1 also supports an external clock source option for I2SMCLK.
- Full duplex operation
- Data sizes 4-24 bits
- Supports MSb/LSb justified
- Supports LSb/MSb first
- Master or slave I^2S supported
- Programmable Fs ratios for I2SCLK, and I2SWORD
- Local loopback capability for testing

7.3.1 I^2S Registers

Table 7-5 shows the I^2S registers.

Table 7-5. I^2S Registers

Offset (Note1)	Register	R/W	Description
0x0000	psc_sel	R/W	PSC Select (See Section 7.1.1 "PSC Select Register" on page 194)
0x0004	psc_ctrl	R/W	PSC Control (See Section 7.1.2 "PSC Control Register" on page 195)
0x0008	psc_i2scfg	R/W	I^2S Configuration Register
0x000C	psc_i2smsk	R/W	I^2S Mask Register
0x0010	psc_i2spcr	R/W	I^2S Protocol Control Register
0x0014	psc_i2sstat	R	I^2S Status Register
0x0018	psc_i2sevnt	R/W	I^2S Event Register
0x001C	psc_i2stxrx	R/W	I^2S Tx/Rx Data Register
0x0020	psc_i2sudf	R/W	I^2S Underflow Register

Note1. See Table 7-1 on Page 194 for base address.

7.3.1.1 I^2S Configuration Register

This register configures the operational parameters for the I^2S protocol. While the device enable bit is set, do not modify any fields other than **psc_i2scfg**[DE].

psc_i2scfg **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
	RT		TT		DD	DE	▨				WS						WI	DIV		BI	BUF	LSJ	XM	LEN					LB	▨	MLF	MS
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:30	RT	Rx FIFO Threshold. 00 1 data entry 01 2 data entries 10 4 data entries 11 8 data entries For DMA transfers (**psc_i2scfg**[DD] = 0), the threshold must match the source transfer size (STS) programmed in the descriptor. For programmed I/O (**psc_i2scfg**[DD] = 1), once the number of data entries rises to the threshold, a *receive request* (RR) is triggered in the event and status registers.	R/W	0
29:28	TT	Tx FIFO Threshold. 00 1 empty slot 01 2 empty slots 10 4 empty slots 11 8 empty slots For DMA transfers (**psc_i2scfg**[DD] = 0), the threshold must match the destination transfer size (DTS) programmed in the descriptor. For programmed I/O (**psc_i2scfg**[DD] = 1), once the number of empty FIFO slots matches the threshold, a *transmit request* (TR) is triggered in the event and status registers.	R/W	0
27	DD	Disable DMA Transfers. 0 Enable DMA transfers. Software prepares buffers and descriptors, and the DDMA controller handles the transfers. 1 Programmed I/O. Software handles each transfer directly by using the data register and monitoring the FIFO events.	R/W	0
26	DE	Device Enable. 0 Disable the I^2S controller. 1 Enable the I^2S controller. Do not modify any other fields in **psc_i2scfg** while the I^2S controller is enabled (**psc_i2sstat**[DR] = 1).	R/W	0
25:22	-	Reserved.	-	0
21:16	WS	Word Strobe. Applies to master mode only. Programs the number of I2SCLK cycles for left or right channel data. Word strobe = 2 * (WS + 1) * I2SCLK cycles For example, to configure 128Fs, let WS = 63.	R/W	0
15	WI	Word Strobe Invert. Selects the I2SWORD polarity. 0 I2SWORD is active-high for the first channel. 1 I2SWORD is active-low for the first channel.	R/W	0
14:13	DIV	Bit Clock Divider. Applies to master mode only. See Section 7.3.2.1 "I^2S Clocking" on page 214. 00 I2SCLK = I2SMCLK / 2 01 I2SCLK = I2SMCLK / 4 10 I2SCLK = I2SMCLK / 8 11 I2SCLK = I2SMCLK / 16	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
12	BI	Bit Clock Invert. 0 I2SCLK starts toggling with the rising edge first. 1 I2SCLK starts toggling with the falling edge first.	R/W	0
11	BUF	L/R Channel Buffer. Applies during underflow condition. 0 Disable left and right channel Tx buffering. The value in **psc_i2sudf**[DATA] is transmitted. 1 Enable left and right channel Tx buffering. The last sample is repeated.	R/W	0
10	LSJ	MSb/LSb Justified Format. Applies to justified transfer mode (XM = 0) only. See Figure 7-11 on page 219 for example transfers. 0 MSb justified 1 LSb justified	R/W	0
9	XM	Transfer Mode. See Figure 7-11 on page 219 for example transfers. 0 Justified transfer mode. When in this mode, the justification is selected by LSJ. 1 I^2S transfer mode (data delayed 1 I2SCLK)	R/W	0
8:4	LEN	Data Length. Must be between 4 and 24 bits (in even increments) and less than or equal to the word strobe programmed in **psc_i2scfg**[WS]. (LEN + 1) is the actual data length.	R/W	0
3	LB	Loopback. Applies to master mode only. 0 Normal operation 1 Loopback mode: internally connect the Tx and Rx FIFOs.	R/W	0
2	—	Reserved.	—	—
1	MLF	MSb/LSb Data First. See Figure 7-11 on page 219 for example transfers. 0 Most-significant bit first 1 Least-significant bit first	R/W	0
0	MS	Master/Slave Mode. 0 Master mode 1 Slave mode	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

7.3.1.2 I^2S Mask Register

This register is used to mask the interrupts triggered by the I^2S Event register (**psc_i2sevnt**) flags. Setting a mask bit disables the corresponding interrupt. All interrupts not needed by an application should be masked.

For predictable interrupt behavior, **psc_i2smsk** should not be modified during an I^2S transfer.

psc_i2smsk **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31:14	-	Reserved.	-	0
13	RR	Mask Rx FIFO request interrupt.	R/W	0
12	RO	Mask Rx FIFO overflow interrupt.	R/W	0
11	RU	Mask Rx FIFO underflow interrupt.	R/W	0
10	TR	Mask Tx FIFO request interrupt.	R/W	0
9	TO	Mask Tx FIFO overflow interrupt.	R/W	0
8	TU	Mask Tx FIFO underflow interrupt.	R/W	0
7:6	-	Reserved.	-	0
5	RD	Mask receive-done interrupt.	R/W	0
4	TD	Mask transmit-done interrupt.	R/W	0
3:0	-	Reserved.	-	0

7.3.1.3 I^2S Protocol Control Register

This register is used to control transfers and to clear the Rx and Tx FIFOs. Setting a bit issues a command to the controller. The bit is automatically cleared once the command is initiated but not necessarily completed.

psc_i2spcr **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:7		Reserved.	-	0
6	RC	Receive FIFO Clear. Set RC to clear the receive FIFO.	R/W	0
5	RP	Receive Stop. Set RP to stop receiving data.	R/W	0
4	RS	Receive Start. Set RS to start receiving data.	R/W	0
3	-	Reserved.	-	0
2	TC	Transmit FIFO Clear. Set TC to clear the transmit FIFO.	R/W	0
1	TP	Transmit Stop. Set TP to stop transmitting data.	R/W	0
0	TS	Transmit Start. Set TS to start transmitting data.	R/W	0

www.DataSheet4U.com

www.DataShe

7.3.1.4 I^{2}S Status Register

This register contains all the status signals for the I^{2}S protocol. It is read only. A status condition is true when the status bit is set. Before the I^{2}S controller is enabled in **psc_i2scfg**[DE], only the device-ready and PSC-ready status bits (**psc_i2sstat**[DR, SR]) are valid; all other status bits read as 0 until ready.

psc_i2sstat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			RF	RE	RR	TF	TE	TR			RB	TB		DI	DR	SR
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:14	-	Reserved.	-	0
13	RF	Receive FIFO Full.	R	0
12	RE	Receive FIFO Empty.	R	1
11	RR	Receive Request.	R	0
10	TF	Transmit FIFO full.	R	0
9	TE	Transmit FIFO Empty.	R	1
8	TR	Transmit Request.	R	0
7:6	-	Reserved.	-	0
5	RB	Receive Busy. Indicates the controller is currently receiving data.	R	0
4	TB	Transmit Busy. Indicates the controller is currently transmitting data.	R	0
3	-	Reserved.	-	0
2	DI	Device Interrupt. Indicates at least one unmasked event has been flagged in **psc_i2sevnt**.	R	0
1	DR	Device Ready. Indicates the I^{2}S controller is ready for protocol operation after setting the device-enable bit **psc_i2scfg**[DE].	R	0
0	SR	PSC Ready. Indicates the PSC is ready for protocol configuration after enabling the PSC in **psc_ctrl**[ENA].	R	0

www.DataSheet4U.com

www.DataSheet4U.com

7.3.1.5 I^{2}S Event Register

This register contains the event flags for the I^{2}S protocol, as well as the FIFO event flags. Status changes are flagged in the Event register. Each event triggers an interrupt unless masked in the I^{2}S mask register (**psc_i2smsk**).

Once an event flag is set, it remains set until software clears it. A flag is cleared by writing a 1 to the appropriate bit; writing a 0 has no effect.

psc_i2sevnt **Offset = 0x0018**

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

Bits	Name	Description	R/W	Default
31:14	-	Reserved.	-	0
13	RR	Receive Request. Applies to programmed I/O only (**psc_i2scfg**[DD] = 1). Indicates the Rx FIFO has a number of data elements equal to the FIFO threshold programmed in **psc_i2scfg**[RT] available for reading.	R/W	0
12	RO	Receive Overflow. Indicates the Rx FIFO has experienced an overflow.	R/W	0
11	RU	Receive Underflow. Indicates the Rx FIFO has experienced an underflow.	R/W	0
10	TR	Transmit Request. Applies to programmed I/O only (**psc_i2scfg**[DD] = 1). Indicates the Tx FIFO is requesting a number of data elements equal to the FIFO threshold programmed in **psc_i2scfg**[TT].	R/W	0
9	TO	Transmit Overflow. Indicates the Tx FIFO has experienced an overflow.	R/W	0
8	TU	Transmit Underflow. Indicates the Tx FIFO has experienced an underflow.	R/W	0
7:6	-	Reserved.	-	0
5	RD	Receive Done. Indicates the receive controller has been stopped and has completed its final transfer.	R/W	0
4	TD	Transmit Done. Indicates the transmit controller has been stopped and has completed its final transfer.	R/W	0
3:0	-	Reserved.	-	0

www.DataSheet4U.com

www.DataSheet4U.com

7.3.1.6 I^{2}S Tx/Rx Data Register

This register is the interface to the Transmit and Receive FIFOs. Each FIFO is 16 entries deep. A write to this register causes data to be pushed onto the Tx FIFO; if the Tx FIFO is already full, the data is lost and a transmit-overflow event is triggered (**psc_i2sevnt**[TO] = 1). A read from this register causes data to be popped from the Rx FIFO; if the Rx FIFO is already empty, the data read is undefined and a receive-underflow event is triggered (**psc_i2sevnt**[RU] = 1).

If transfers are under program control (**psc_i2scfg**[DD] = 1), software can monitor the transmit-request (TR) and receive-request (RR) flags to maintain proper FIFO levels either by polling **psc_i2sstat**[TR, RR] or by waiting for interrupts in **psc_i2sevnt**[TR, RR]. The DDMA controller (**psc_i2scfg**[DD] = 0) automatically manages overflow/underflow conditions.

The data size of read/write transfers is always 32 bits to the register, but the FIFO data size is limited to 24 bits with the valid portion depending on **psc_i2scfg**[LEN]. When DMA transfers are enabled, the **psc_i2stxrx** address locations must be programmed into the Tx descriptor destination pointer and the Rx descriptor source pointer.

psc_i2stxrx **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
									DATA																							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	0
23:0	DATA	The FIFO data is 4 to 24 bits wide. Bit 0 is the LSb.	R/W	0

7.3.1.7 I^{2}S Underflow Register

This register can be used to transmit a fixed data value when a Tx FIFO underflow occurs. If transmit buffering is disabled (**psc_i2scfg**[BUF] = 0) and a transmit underflow occurs, the controller sends **psc_i2sudf**[DATA] for both channels of the frame with underflow.

psc_i2sudf **Offset = 0x0020**

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

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	0
23:0	DATA	I^{2}S Underflow Data. Only the valid bits are transmitted regardless of the data value. The valid bits are determined by the data-length field (**psc_i2scfg**[LEN]). Bit 0 is the LSb.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

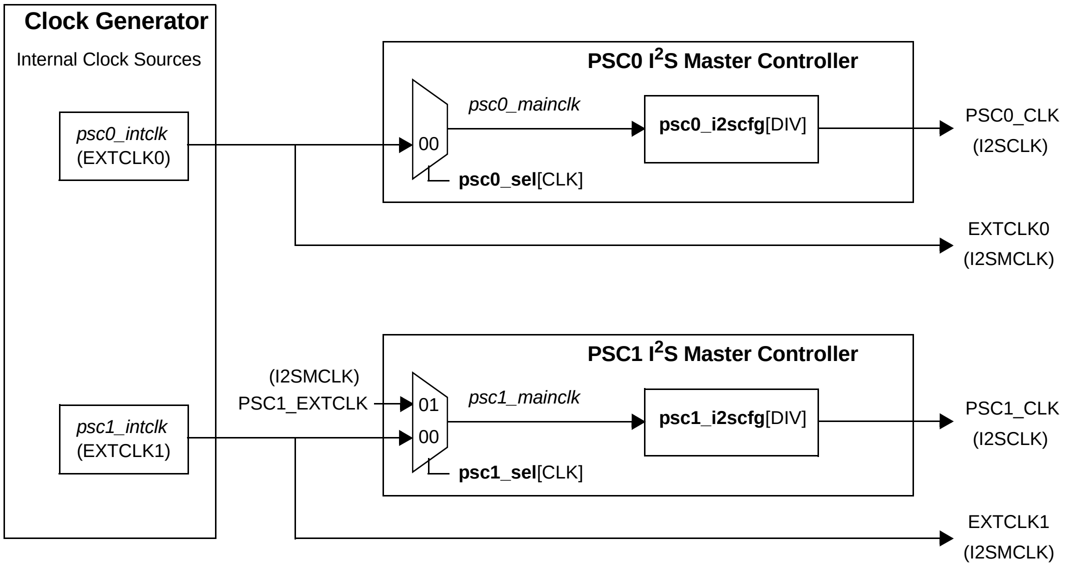
7.3.2 I^2S Operation

7.3.2.1 I^2S Clocking

For the I^2S controller, both external and internal options are available to generate or receive clocks to synchronize with incoming/outgoing data.

I^2S Master Clocking

Figure 7-7 shows the clocking options for the I^2S master controller.



For PSC1, I2SMCLK can be provided externally and input via PSC1_EXTCLK (psc1_sel[CLK] = 01), or generated internally and output via EXTCLK*n* (psc*n*_sel[CLK] = 00).

Figure 7-7. I^2S Master Controller Clock Options

Putting the I^2S controller into master mode requires that the controller generate both I2SWORD and I2SCLK for incoming and outgoing data.

I2SMCLK can be configured as an input (PSC1 only) or an output (both PSC0 and PSC1). As an input, I2SMCLK can be externally generated by an oscillator or PLL and input via PSC1_EXTCLK. (50 ppm max drift with 50% duty cycle required.) As an output, I2SMCLK can be internally generated and driven on EXTCLK*n* (muxed with GPIO[2] and GPIO[3]).

For example, to calculate an external I2SMCLK for a 44.1KHz File (compact disc audio file) => FS = 44.1KHz with Typical Codec Specification: I2SMCLK = 256FS, I2SCLK = 128FS, I2SWORD = FS:

1) I2SMCLK oscillator = 256*44.1KHz = 11.2896MHz (provided externally)

2) I2SCLK -> Clear **psc*n*_i2scfg**[DIV] for I2SMCLK/2

3) I2SWORD -> Program **psc*n*_i2scfg**[WS] to 63 for 128FS

I^2S Slave Clocking

The I^2S slave controller allows an external device to drive both I2SCLK and I2SWORD (**psc*n*_sel**[CLK] = 10). I2SMCLK is not required. The I^2S slave controller can interface with external controllers that output audio data.

www.DataSheet4U.com

www.DataSheet4U.com

7.3.2.2 I^2S Flowcharts

Figure 7-8 shows a configuration flowchart for the I^2S controller. See also Section 7.3.2.3 "I^2S Configuration" on page 218.

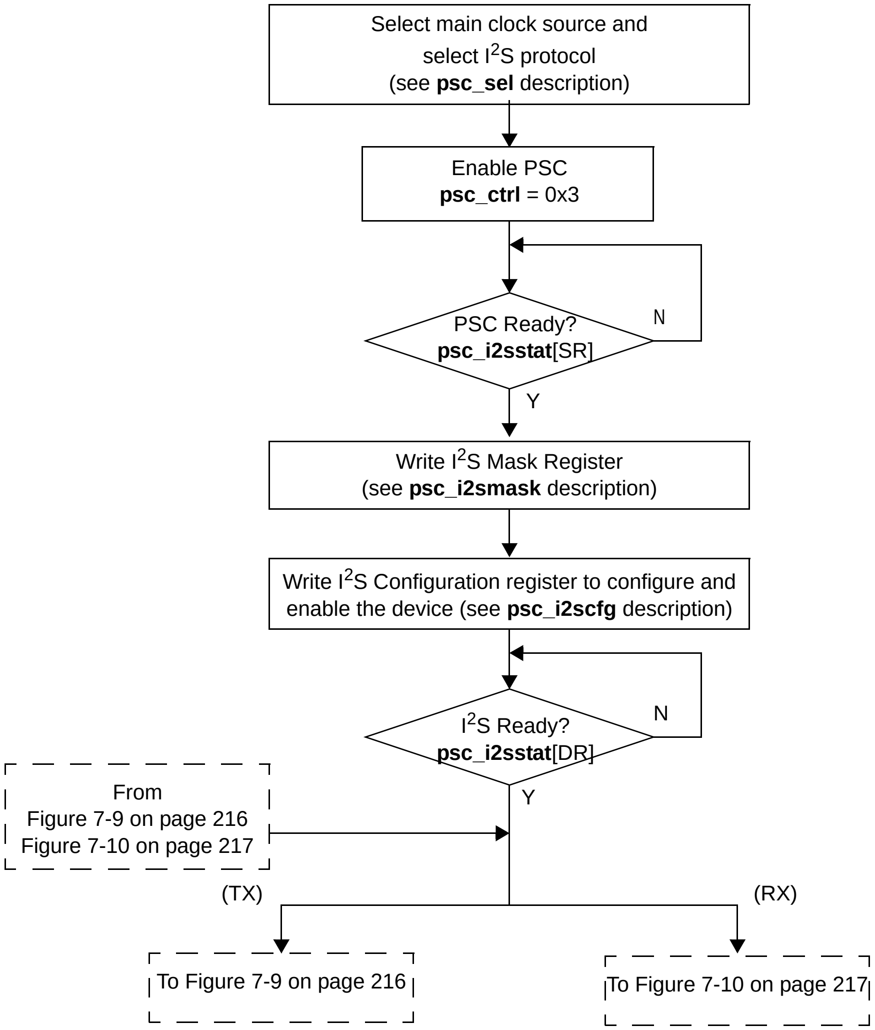
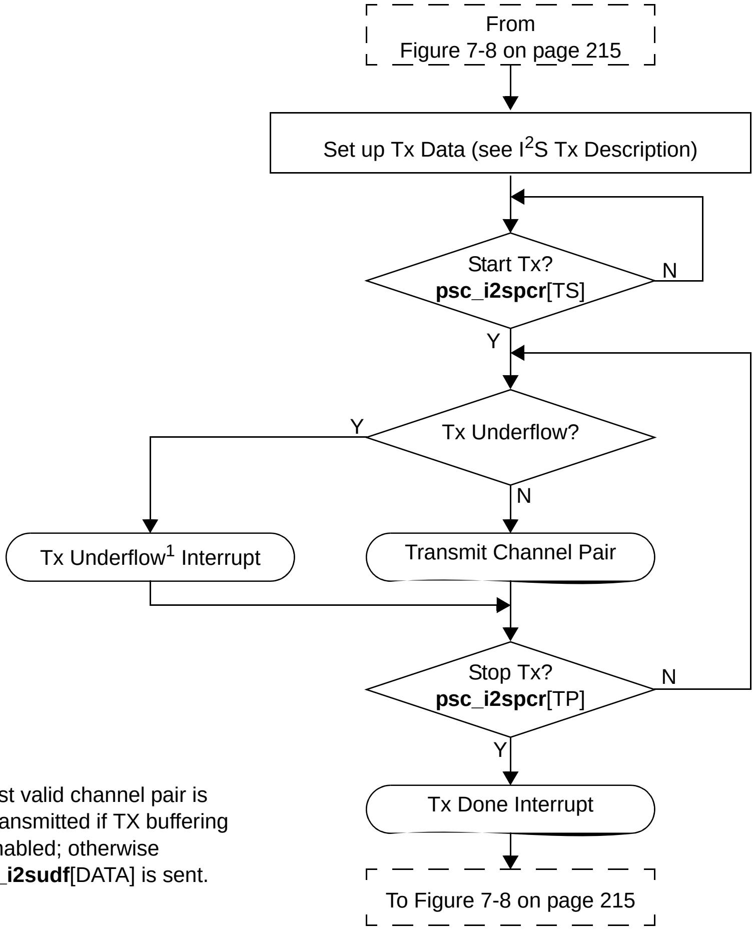

Select main clock source and select I^2S protocol (see **psc_sel** description)
Enable PSC **psc_ctrl** = 0x3
PSC Ready? **psc_i2sstat**[SR]
N
Y
Write I^2S Mask Register (see **psc_i2smask** description)
Write I^2S Configuration register to configure and enable the device (see **psc_i2scfg** description)
I^2S Ready? **psc_i2sstat**[DR]
N
Y
From Figure 7-9 on page 216 Figure 7-10 on page 217
(TX)
(RX)
To Figure 7-9 on page 216
To Figure 7-10 on page 217

Figure 7-8. I^2S Configuration

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-9 shows the flowchart for I^{2}S transmit. See also Section 7.3.2.4 "I^{2}S Transmit" on page 218.



[1] Last valid channel pair is re-transmitted if TX buffering is enabled; otherwise **psc_i2sudf**[DATA] is sent.

Figure 7-9. I^{2}S Transmit

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-10 shows the flowchart for I²S receive. See also Section 7.3.2.5 "I²S Receive" on page 218.

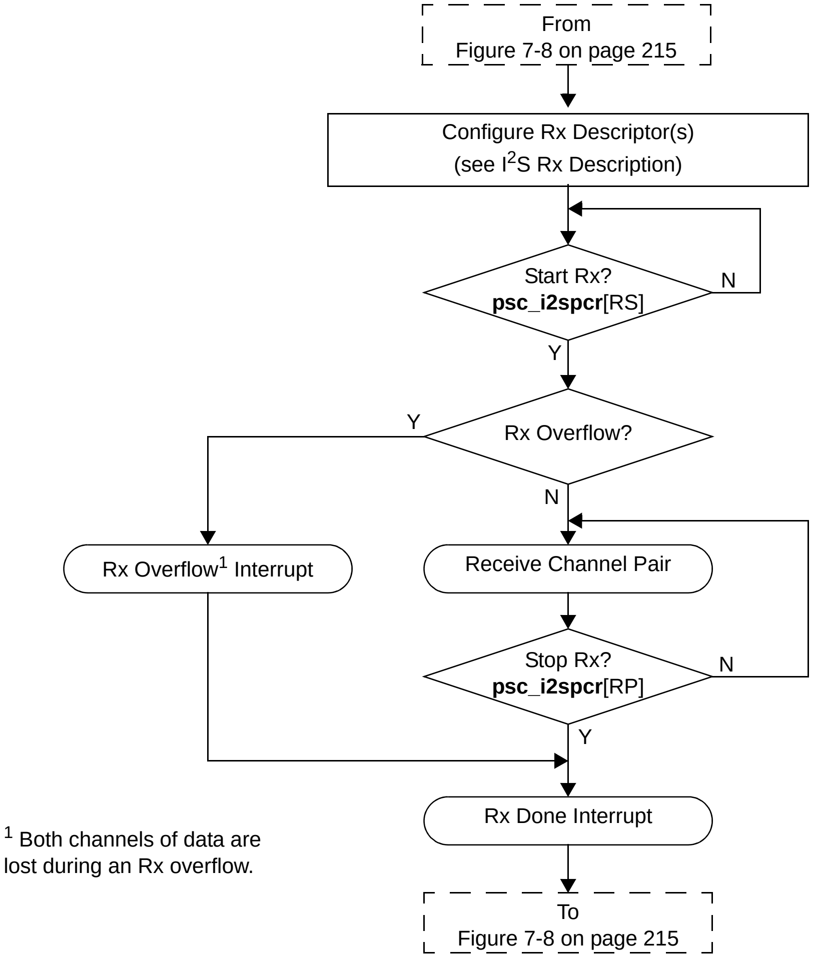


Figure 7-10. I²S Receive

www.DataSheet4U.com

www.DataSheet4U.com

7.3.2.3 I^2S Configuration

See Figure 7-8 on page 215 for reference.

1) Enable or disable DMA transfers (**psc_i2scfg**[DD]). The I^2S controller can operate in either mode.
2) Configure clock source to drive I2SCLK, I2SWORD, and I2SMCLK. See also Section 7.3.2.1 "I^2S Clocking" on page 214.
3) Enable the PSC for I^2S operation. Poll the **psc_i2sstat**[SR] bit to see if the PSC is ready.
4) Configure I2SWORD, I2SMCLK, and I2SCLK for the required sample rate. I2SMCLK is internally generated (**psc_sel**[CLK] = 00). For PSC1 only, I2SMCLK can also be configured as an input (**psc_sel**[CLK] = 01).

If the controller is configured to generate I2SWORD, I2SCLK, and I2SMCLK, the signals begin to transmit regardless of the controller status once the device is enabled. In this condition, the controller is idle: no data is being transmitted, and I2SDO is held low or receiving.

7.3.2.4 I^2S Transmit

See Figure 7-9 on page 216 for reference.

1) Enable DMA transfer or fill Tx FIFOs with data. Software must load channel data in the required order. Left channel data is loaded first.
2) Set the Tx Start bit (**psc_i2spcr**[TS] = 1) to start the data transfer. The Tx Busy status (**psc_i2sstat**[TB]) can be checked to determine if the controller has engaged.
3) I^2S controller checks for two valid Tx data elements before sending the first one to ensure it maintains synchronization with I2SWORD. A transmit underflow (**psc_i2sevnt**[TU] = 1) results if this condition is not met. Both left and right channels transmit at this time. If an underflow condition occurs, Tx FIFOs must be loaded before clearing the interrupt to avoid causing another transmit underflow interrupt from empty FIFOs. Software can monitor for an underflow condition to make sure all data has been sent.

The I^2S transmit controller sends either buffered Tx data or a preset Tx data element (depending on the configuration) for both channels of the current frame. This prevents synchronization loss on I2SWORD strobe.

To disable the transmitter, set the **psc_i2spcr**[TP] bit. Data stops transmitting at the end of the current frame. While the transmitter is disabled, it continues to write zeros on each channel.

While the transmitter is disabled, the Tx FIFO can be cleared by setting the transmit-clear bit (**psc_i2spcr**[TC] = 1). Any data in the FIFO is transmitted upon enabling the transmitter.

7.3.2.5 I^2S Receive

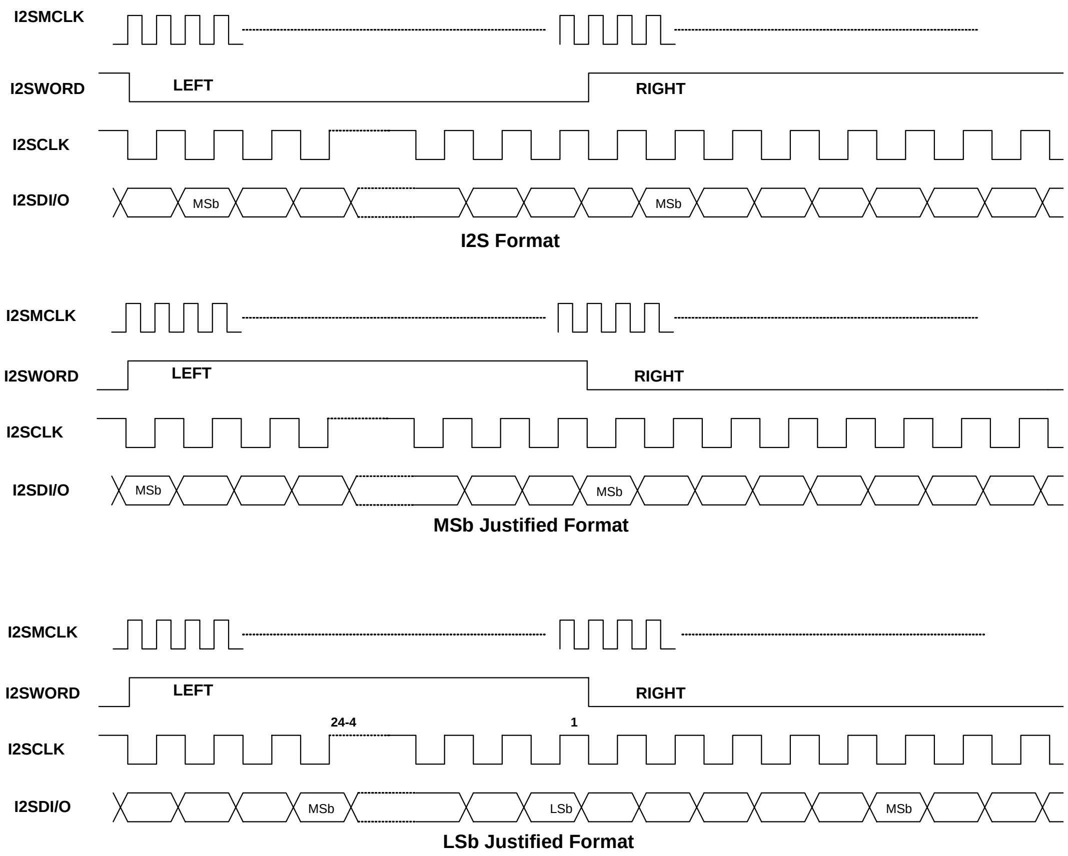
See Figure 7-10 on page 217 for reference. The Rx controller operates independently of the Tx controller. Either can be enabled or disabled without affecting the other.

1) Enable DMA (if used) for L/R channel receive. For DMA transfers into memory, the descriptors must have a byte count that is evenly divisible by the size of a channel pair (data length x 2).
2) Set the Rx Start bit (**psc_i2spcr**[RS] = 1) to start receiving. Poll status busy (**psc_i2sstat**[RB]) to see when the controller has engaged.
3) If at least two entries are not available in the Rx FIFO, an overflow occurs (**psc_i2sevnt**[RO] = 1). In this case, an overflow interrupt is triggered unless it is masked. An overflow results in the loss of data for both channels. Two entries are always required to maintain data synchronization with I2SWORD. If at least two entries are available, both channels are received and pushed into the Rx FIFO. The process continues for each frame.
4) For an Rx overflow, reads from the FIFO must occur to make available at least two empty entries. Software must then clear the overflow interrupt to avoid multiple interrupts for the same condition.
5) Set the interrupt-enable bit in the last Rx descriptor (if DMA used) when the last packet is received. The receiver can be disabled on that interrupt.
6) Similar to the transmitter, the receiver stops at the *end* of the current frame when disabled (**psc_i2spcr**[RP] = 1). The Rx busy bit indicates if the controller has been disabled. The receiver ignores incoming data while disabled.
7) While the receiver is disabled, the Rx FIFO can be cleared by setting the receive-clear bit (**psc_i2spcr**[RC] = 1).

www.DataSheet4U.com

www.DataSheet4U.com

7.3.3 I^{2}S Timing



I2S Format

MSb Justified Format

LSb Justified Format

Note: LSb/Msb justified format is configurable for 4-24 bit left and right channel data.
I2SMCLK clock output aligns to I2SCLK as seen in diagram.

Figure 7-11. I^{2}S Functional Timing Diagrams

7.3.4 I^{2}S Signals

Table 7-6 shows the I^{2}S signals.

Table 7-6. I^{2}S Signals

PSC Generic	I^{2}S Specific
PSC*n*_CLK	I2SCLK
PSC*n*_SYNC0	I2SWORD
PSC*n*_SYNC1	—
PSC*n*_D0	I2SDO
PSC*n*_D1	I2SDI
PSC1_EXTCLK	I2SMCLK (input)
pscn_intclk	I2SMCLK (output on EXTCLK*n*)

www.DataSheet4U.com

www.DataSheet4U.com

7.4 AC97 Controller

When the AC97 protocol is selected for a PSC (**psc_sel**[PS] = 0b100), the PSC becomes an AC97 controller. The AC97 controller supports the following features:

- Five signal AC-link interface (ACBCLK, ACSYNC, ACDO, ACDI, ACRST#)
- Complies with the Audio Codec '97 Specification (revision 2.3 release 0.9) except when stated otherwise.
- Supports primary codec (audio or modem) with full duplex communication
- Supports up to 3 secondary audio codecs (transmit only)
- Variable rate audio support via primary codec slot request bits
- DMA based Tx/Rx data transfers
- Support cold, warm and register codec resets
- Support AC-link power-down mode
- Supports hardware I/O control (slot 12)

7.4.1 AC97 Registers

Table 7-7 shows the AC97 registers.

Table 7-7. AC97 Registers

Offset (Note1)	Register	R/W	Description
0x0000	psc_sel	R/W	PSC Select (See Section 7.1.1 on page 194)
0x0004	psc_ctrl	R/W	PSC Control (See Section 7.1.2 on page 195)
0x0008	psc_ac97cfg	R/W	AC97 Configuration Register
0x000C	psc_ac97msk	R/W	AC97 Mask Register
0x0010	psc_ac97pcr	R/W	AC97 Protocol Control Register
0x0014	psc_ac97stat	R	AC97 Status Register
0x0018	psc_ac97evnt	R/W	AC97 Event Register
0x001C	psc_ac97txrx	R/W	AC97 Tx/Rx Data Register
0x0020	psc_ac97cdc	R/W	AC97 Codec Command Register
0x0024	psc_ac97rst	R/W	AC97 Reset Control Register
0x0028	psc_ac97gpo	R/W	AC97 GPIO Output Register
0x002C	psc_ac97gpi	R	AC97 GPIO Input Register

Note1. See Table 7-1 on Page 194 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

7.4.1.1 AC97 Configuration Register

This register configures the operational parameters for the AC97 protocol. While the device enable bit is set, do not modify any fields other than **psc_ac97cfg**[DE].

psc_ac97cfg **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
	RT		TT		DD	DE	SE	LEN				TXSLOT										RXSLOT										GE
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:30	RT	Rx FIFO Threshold. 00 1 data entry 01 2 data entries 10 4 data entries 11 8 data entries For DMA transfers (**psc_ac97cfg**[DD] = 0), the threshold must match the source transfer size (STS) programmed in the descriptor. For programmed I/O (**psc_ac97cfg**[DD] = 1), once the number of data entries rises to the threshold, a *receive request* (RR) is triggered in the event and status registers.	R/W	0
29:28	TT	Tx FIFO Threshold. 00 1 empty slot 01 2 empty slots 10 4 empty slots 11 8 empty slots For DMA transfers (**psc_ac97cfg**[DD] = 0), the threshold must match the destination transfer size (DTS) programmed in the descriptor. For programmed I/O (**psc_ac97cfg**[DD] = 1), once the number of empty FIFO slots matches the threshold, a *transmit request* (TR) is triggered in the event and status registers.	R/W	0
27	DD	Disable DMA Transfers. 0 Enable DMA transfers. Software prepares buffers and descriptors, and the DDMA controller handles the transfers. 1 Programmed I/O. Software handles each transfer directly by using the data register and monitoring the FIFO events.	R/W	0
26	DE	Device Enable. 0 Disable the AC97 controller. 1 Enable the AC97 controller. Do not modify any other fields in **psc_ac97cfg** while the AC97 controller is enabled (**psc_ac97stat**[DR] = 1).	R/W	0
25	SE	Secondary Codec Enable. 0 Disable secondary codec support. 1 Enable secondary codec support. The AC97 controller transmits all valid Tx slots (based on Tx-slot-valid configuration bits) on each requested frame. A frame request is detected each time the primary codec requests one (or more) Tx slots. The primary and secondary codecs must sample at the same variable rate.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
24:21	LEN	Data Length. Must be between 8 and 20 bits. This value is used to determine the MSb of Tx and Rx data. Data length = (LEN * 2) + 2 Be sure to program the data length between 8-20 (LEN between 3-9) for proper AC97 operation.	R/W	0
20:11	TXSLOT	Valid Transmit Slots (3-12). Designates (with a 1) which transmit slots should contain valid data. This allows software to better control the Tx data stream so that a Tx data slot can be disabled even if the primary codec is requesting. Disabled slots transmit 0s. TX SLOT VALID [12:3] = **psc_ac97cfg**[20:11]	R/W	0
10:1	RXSLOT	Valid Receive Slots (3-12). Designates (with a 1) which receive data slots should be pushed onto the Rx FIFO. This allows software to better control the Rx data stream so that an Rx data slot can be ignored even if the primary codec has the slot valid asserted. Disabled slots are not pushed onto the Rx FIFO. RX SLOT VALID [12:3] = **psc_ac97cfg**[10:1]	R/W	0
0	GE	GPIO Register Enable. 0 Disable the GPI and GPO registers. 1 Enable the GPI and GPO registers. Slot12 of every AC97 frame is sent from the GPO register and captured in the GPI register. When new GPI data is captured, the GPI Data Ready flag (**psc_ac97evnt**[GR]) is set.	R/W	0

7.4.1.2 AC97 Mask Register

This register is used to mask interrupts triggered by the AC97 Event register (**psc_ac97evnt**) flags. Setting a mask bit disables the corresponding interrupt. All interrupts not needed by an application should be masked.

For predictable interrupt behavior, **psc_ac97msk** should not be modified during an AC97 transfer.

psc_ac97msk **Offset = 0x000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							GR	CD											RR	RO	RU	TR	TO	TU			RD	TD				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:26	-	Reserved.	-	0
25	GR	GPI Data-Ready Interrupt Mask.	R/W	0
24	CD	Codec Command-Done Interrupt Mask.	R/W	0
23:14	-	Reserved.	-	0
13	RR	Rx FIFO Request Interrupt Mask.	R/W	0
12	RO	Rx FIFO Overflow Interrupt Mask.	R/W	0
11	RU	Rx FIFO Underflow Interrupt Mask.	R/W	0
10	TR	Tx FIFO Request Interrupt Mask.	R/W	0
9	TO	Tx FIFO Overflow Interrupt Mask.	R/W	0
8	TU	Tx FIFO Underflow Interrupt Mask.	R/W	0
7:6	-	Reserved.	-	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
5	RD	Receive-Done Interrupt Mask.	R/W	0
4	TD	Transmit-Done Interrupt Mask.	R/W	0
3:0	-	Reserved.	-	0

7.4.1.3 AC97 Protocol Control Register

This register is used to control transfers and to clear the Rx and Tx FIFOs. Setting a bit issues a command to the controller. The bit is automatically cleared once the command is initiated but not necessarily completed.

psc_ac97pcr **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:7	-	Reserved.	-	0
6	RC	Receive Data Clear. Set this bit to clear the receive FIFO. Do not set this bit while the receive controller is enabled.	R/W	0
5	RP	Receive Stop. Set this bit to stop the receive controller. Once stopped, **psc_ac97stat**[RB] is cleared.	R/W	0
4	RS	Receive Start. Set this bit to start the receive controller. Once started, **psc_ac97stat**[RB] is set until the Receive Stop control bit writes a 1 to stop receiving.	R/W	0
3	-	Reserved.	-	0
2	TC	Transmit Data Clear. Set this bit to clear the transmit FIFO. Do not set this bit while the transmit controller is enabled.	R/W	0
1	TP	Transmit Stop. Set this bit to stop the transmit controller. Once stopped, **psc_ac97stat**[TB] is cleared.	R/W	0
0	TS	Transmit Start. Set this bit to start the transmit controller. Once started, **psc_ac97stat**[TB] is set until the Transmit Stop control bit writes a 1 to stop sending.	R/W	0

www.DataSheet4U.com

w w w . D a t a S h e

7.4.1.4 AC97 Status Register

This register contains the status signals for the AC97 protocol. It is read only. A status condition is true when the status bit is set. Before the AC97 controller is enabled in **psc_ac97cfg**[DE], only the device-ready and PSC-ready status bits (**psc_ac97stat**[DR, SR]) are valid; all other status bits read as 0 until ready.

psc_ac97stat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CB	CP	CR											RF	RE	RR	TF	TE	TR			RB	TB		DI	DR	SR
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:27	-	Reserved.	-	0
26	CB	Codec Bit Clock Detected. Indicates the AC97 controller is receiving the primary codec bit clock to the PSC*n*_CLK input port.	R	0
25	CP	Command Pending. Indicates a codec Command is pending on the AC-link.	R	0
24	CR	Codec Ready. Indicates a codec Ready has been sampled from the previous AC-frame.	R	0
23:14	-	Reserved.	-	0
13	RF	Receive FIFO Full.	R	0
12	RE	Receive FIFO Empty.	R	1
11	RR	Receive Request.	R	0
10	TF	Transmit FIFO Full.	R	0
9	TE	Transmit FIFO Empty.	R	1
8	TR	Transmit Request.	R	0
7:6	-	Reserved.	-	0
5	RB	Receive Busy. Indicates the receive controller is enabled and is receiving data.	R	0
4	TB	Transmit Busy. Indicates the transmit controller is enabled and is transmitting data.	R	0
3	-	Reserved.	-	0
2	DI	Device Interrupt. Indicates at least one unmasked event has been flagged in **psc_ac97evnt**.	R	0
1	DR	Device Ready. Indicates the AC97 controller is ready for protocol operation after setting the device-enable bit **psc_ac97cfg**[DE].	R	0
0	SR	PSC Ready. Indicates the PSC is ready for protocol configuration after enabling the PSC in **psc_ctrl**[ENA].	R	0

www.DataSheet4U.com

www.DataSheet4U.com

7.4.1.5 AC97 Event Register

This register contains the AC97 event flags. When an event flag is set, the event generates an interrupt unless masked in the AC97 Mask register (**psc_ac97msk**).

Once software detects an interrupt, software must read **psc_ac97evnt** to determine the interrupt source. Once the event has been read, software must clear it so that the next event is not missed.

Once an event flag has been set, it remains set until cleared by software. A flag is cleared by writing a 1 to the appropriate bit; writing a 0 has no effect.

psc_ac97evnt **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							GR	CD											RR	RO	RU	TR	TO	TU			RD	TD				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:26	-	Reserved.	-	0
25	GR	GPI Data Ready. Indicates new data is ready in the GPIO Input register (Only used when the GPIO registers are enabled.)	R/W	0
24	CD	Codec Command Done. Indicates a codec command has successfully completed.	R/W	0
23:14	-	Reserved.	-	0
13	RR	Receive Request. Applies to programmed I/O only (**psc_ac97cfg**[DD] = 1). Indicates the Rx FIFO has a number of data elements equal to the FIFO threshold programmed in **psc_ac97cfg**[RT] available for reading.	R/W	0
12	RO	Receive Overflow. Indicates the Rx FIFO has experienced an overflow.	R/W	0
11	RU	Receive Underflow. Indicates the Rx FIFO has experienced an underflow.	R/W	0
10	TR	Transmit Request. Applies to programmed I/O only (**psc_ac97cfg**[DD] = 1). Indicates the Tx FIFO is requesting a number of data elements equal to the FIFO threshold programmed in **psc_ac97cfg**[TT].	R/W	0
9	TO	Transmit Overflow. Indicates the Tx FIFO has experienced an overflow.	R/W	0
8	TU	Transmit Underflow. Indicates the Tx FIFO has experienced an underflow.	R/W	0
7:6	-	Reserved.	-	0
5	RD	Receive Done. This flag indicates that the receive controller has been disabled.	R/W	0
4	TD	Transmit Done. This flag indicates that all data has been transmitted and the transmit controller has been disabled.	R/W	0
3:0	-	Reserved.	-	0

www.DataSheet4U.com

www.DataSheet4U.c

7.4.1.6 AC97 Tx/Rx Data Register

This register is the interface to the transmit and receive FIFOs. Each FIFO is 16 entries deep. A write to this register causes data to be pushed onto the Tx FIFO; if the Tx FIFO is already full, the data is lost and a transmit-overflow event is triggered (**psc_ac97evnt**[TO] = 1). A read from this register causes data to be popped from the Rx FIFO; if the Rx FIFO is already empty, the data read is undefined and a receive-underflow event is triggered (**psc_ac97evnt**[RU] = 1).

If transfers are under program control (**psc_ac97cfg**[DD] = 1), software can monitor the transmit-request (TR) and receive-request (RR) flags to maintain proper FIFO levels either by polling **psc_ac97stat**[TR, RR] or by waiting for interrupts in **psc_ac97evnt**[TR, RR]. The DDMA controller (**psc_ac97cfg**[DD] = 0) automatically manages overflow/underflow conditions.

The data size of read/write transfers is always 32-bits to the register; however, AC97 data payloads are limited to a maximum of 20-bits and a minimum of 8-bits (configured in **psc_ac97cfg**[LEN]). The upper unused data bits can be left uninitialized.

When DMA transfers are enabled, the **psc_ac97txrx** address locations must be programmed into the Tx descriptor destination pointer and the Rx descriptor source pointer.

psc_ac97txrx **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
													DATA																			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:20	-	Reserved.	-	-
19:0	DATA	AC97 data values can be between 8 and 20 bits wide. Program the **psc_ac97cfg**[LEN] register value accordingly.	R/W	0

7.4.1.7 AC97 Codec Command Register

This register controls reads and writes with the external codec(s). A write to this register contains the codec 2-bit ID, command register index and the R/W bit to indicate the direction of the transfer. If the transfer is to be a write, the write data must also be included; otherwise, write zeros to the data location (**psc_ac97cdc**[DATA]).

Upon writing the **psc_ac97cdc** register, the command-pending status bit is set (**psc_ac97stat**[CP] = 1), and on the next AC-link frame the codec ID, index, R/W bit, and data (for a write) are sent out on the AC-link.

When a read or write completes, the command-done flag is set (**psc_ac97evnt**[CD] = 1), and the command-done interrupt is generated unless masked. Writes take no more than two AC97-link frames; reads take at least three frames depending on the codec latency. If the external codec does not respond to a transfer, the command-pending flag remains set and the command-done event is never triggered.

For codec read commands, once the command-done event occurs, software must read **psc_ac97cdc**[DATA] before clearing the command-done event flag.

NOTE: Reads are supported only from the primary codec. Reads to a secondary codec are ignored.

psc_ac97cdc **Offset = 0x0020**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:26	-	Reserved.	-	0
25	RD	Codec Read/Write Command. 0 Write command 1 Read command RD reads as 0 after the command completes (**psc_ac97evnt**[CD] = 1).	R/W	0
24:23	ID	Codec ID. Contains the ID for the codec to be addressed. ID reads as 0s after the command completes (**psc_ac97evnt**[CD] = 1).	R/W	0
22:16	INDX	Codec Register Index. Program INDX with the index of the codec command register. When read, INDX reflects the codec index.	R/W	0
15:0	DATA	Read/Write Data. For a codec write command, write the data here. This field reads 0 once the command-done event (**psc_ac97evnt**[CD]) is set. For a codec read command, read the data here once the command-done event (**psc_ac97evnt**[CD]) is set.	R/W	0

7.4.1.8 AC97 Reset Control Register

This register is used to initiate a warm, or cold reset on the AC97-link:

- Cold Reset—A cold AC97 reset re-activates the AC-link and resets all codec register values to their default state.
- Warm AC97 Reset—A warm AC97 reset re-activates the AC-link without altering the current AC97 register values. A warm reset is signaled by driving ACSYNC high for a minimum of 1µs in the absence of ACBCLK.

psc_ac97rst **Offset = 0x0024**

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

Bits	Name	Description	R/W	Default
31:2	-	Reserved	-	0
1	RST	AC-link Reset Signal. 0 Drive ACRST# high. 1 Drive ACRST# low. Setting RST drives ACRST# low to initiate a cold AC97 reset. After satisfying the reset low time for the codec, clear RST to negate ACRST#. RST has no effect on the AC97 controller and must not be set during AC97 transfers.	R/W	0
0	SNC	Synchronization Signal. 0 Drive ACSYNC low. 1 Drive ACSYNC high. SNC controls the ACSYNC signal when the bit clock ACBCLK is disabled. SNC is used to initiate a warm reset on the AC97-link in the absence of ACBCLK. Do not set this bit when ACBCLK is enabled.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

7.4.1.9 AC97 GPIO Output Register

This register is used to hold GPIO data to be transmitted on the AC97-link. If the GPIO registers are enabled (**psc_ac97cfg**[GE] = 1), the **psc_ac97gpo** data is placed in Slot 12 of each outgoing frame. When new data is written to this register it can take up to two full AC frames to be transmitted.

The GPIO data length is limited by the data length setting in **psc_ac97cfg**[LEN].

psc_ac97gpo **Offset = 0x0028**

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

Bits	Name	Description	R/W	Default
31:20	-	Reserved.	-	0
19:0	DATA	GPIO Output Data.	R/W	0

7.4.1.10 AC97 GPIO Input Register

This register holds the GPIO data that is captured from the AC97-link. If the GPIO registers are enabled (**psc_ac97cfg**[GE] = 1), data from slot 12 of each incoming frame is captured and placed in the AC97 GPIO Input register. In order to prevent software from needing to read this register every frame, an event bit (GPI Data Ready) is set each time there is new data to be read from **psc_ac97gpi**. Software must read the new GPIO Input data and clear the GPI Data Ready event bit before new GPIO Input data arrives to prevent loss of GPIO data.

The GPIO data length is limited by the data length setting in **psc_ac97cfg**[LEN].

psc_ac97gpi **Offset = 0x002C**

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

Bits	Name	Description	R/W	Default
31:20	-	Reserved.	-	0
19:4	DATA	GPIO Input Data.	R/W	0
3:1	VS	Vendor specific, defined by codec.	R/W	0
0	INT	GPIO interrupt. Not supported by the PSC and must be masked in the primary codec. This bit indicates a GPIO interrupt has been generated on the codec and does not generate an interrupt on the processor.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

7.4.2 AC97 Operation

7.4.2.1 AC97 Clocking

The AC97 controller relies on the external codec to provide the bit clock ACBCLK (**psc*n*_sel**[CLK] = 10).

7.4.2.2 AC97 Flowcharts

Figure 7-12 shows a configuration flowchart for the AC97 controller. See also Section 7.4.2.3 "AC97 Configuration" on page 232.

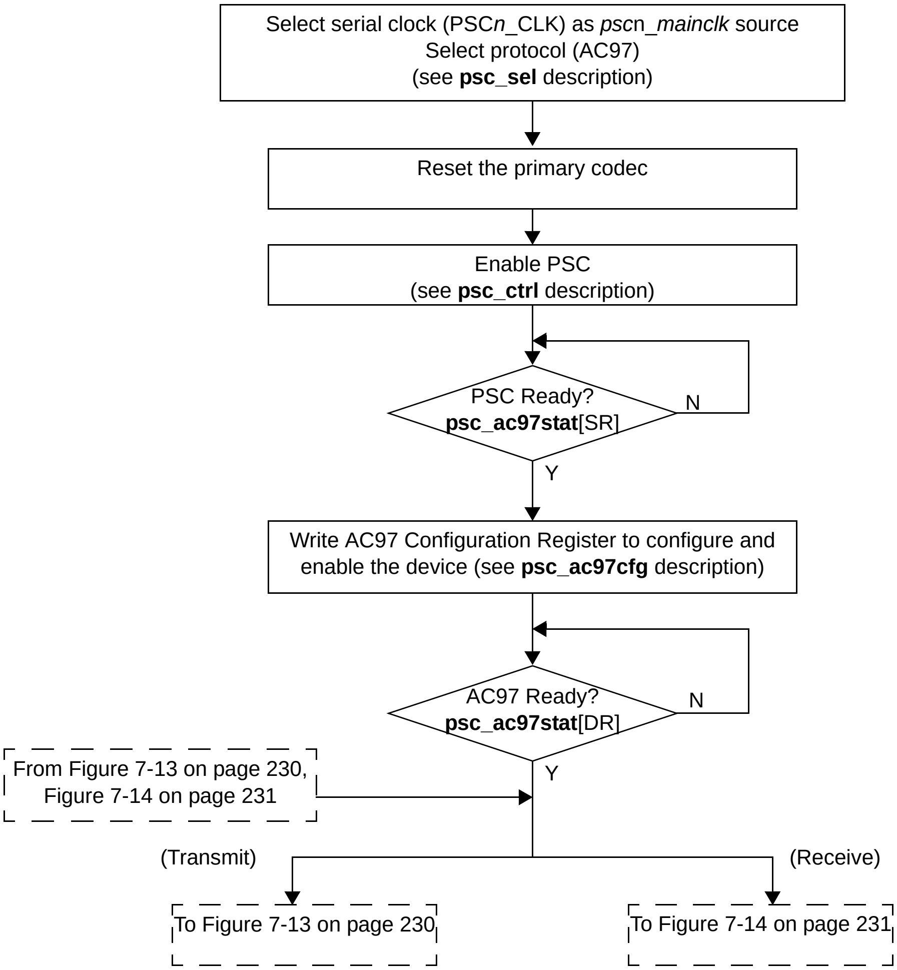

Select serial clock (PSC*n*_CLK) as pscn_*mainclk* source
Select protocol (AC97)
(see **psc_sel** description)
Reset the primary codec
Enable PSC
(see **psc_ctrl** description)
PSC Ready?
psc_ac97stat[SR]
N
Y
Write AC97 Configuration Register to configure and enable the device (see **psc_ac97cfg** description)
AC97 Ready?
psc_ac97stat[DR]
N
Y
From Figure 7-13 on page 230,
Figure 7-14 on page 231
(Transmit)
(Receive)
To Figure 7-13 on page 230
To Figure 7-14 on page 231

Figure 7-12. AC97 Configuration

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-13 shows the flowchart for AC97 transmit. See also Section 7.4.2.5 "AC97 Transmit" on page 232.

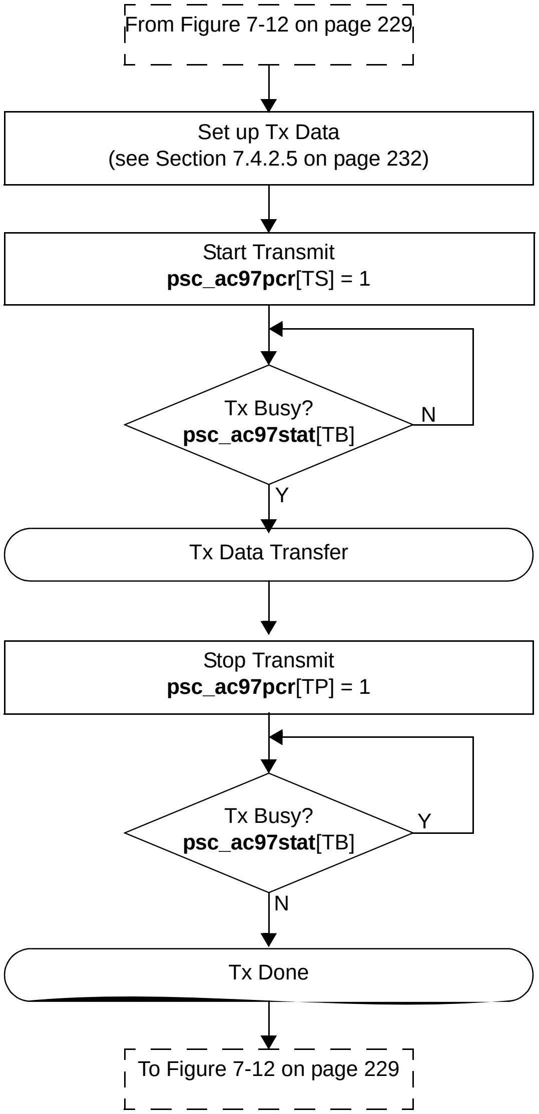


Figure 7-13. AC97 Transmit

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-14 shows the flowchart for AC97 receive. See also Section 7.4.2.6 "AC97 Receive" on page 233.

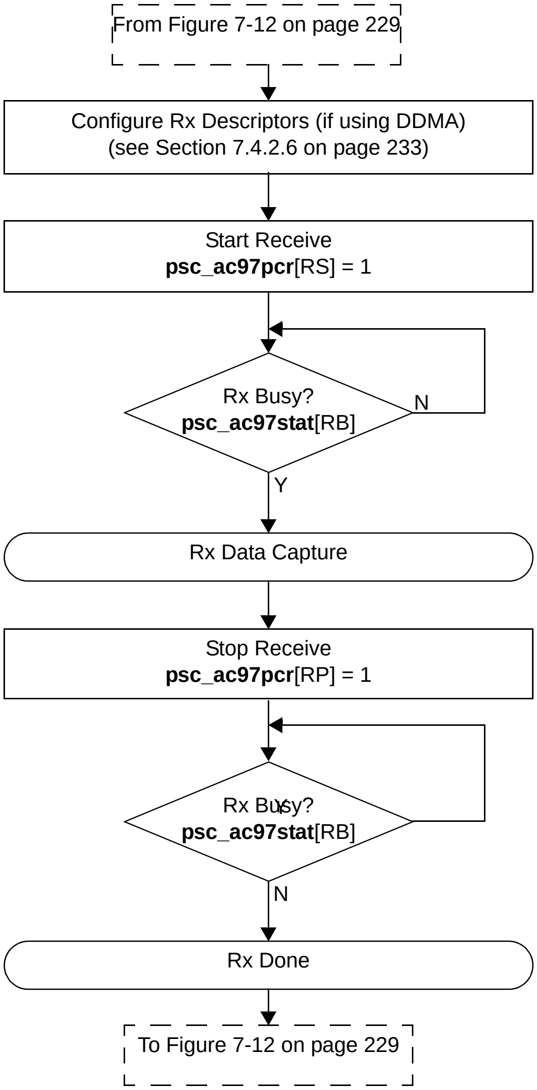

From Figure 7-12 on page 229
Configure Rx Descriptors (if using DDMA) (see Section 7.4.2.6 on page 233)
Start Receive psc_ac97pcr[RS] = 1
Rx Busy? psc_ac97stat[RB]
N
Y
Rx Data Capture
Stop Receive psc_ac97pcr[RP] = 1
Rx Busy? psc_ac97stat[RB]
Y
N
Rx Done
To Figure 7-12 on page 229

Figure 7-14. AC97 Receive

www.DataSheet4U.com

www.DataSheet4U.com

7.4.2.3 AC97 Configuration

See Figure 7-12 on page 229 for reference.

1) Select the AC97 protocol (**psc_sel**[PS] = 0b100) and the clock source (**psc_sel**[CLK] = 0b10).

2) Reset the codec. This enables the codec to drive bit clock, thus allowing the PSC configuration to continue. Reset is performed as follows:

3) Set **psc_ac97rst**[RST]. This holds ACRST# low. Clear **psc_ac97rst**[RST] after the minimum reset time required for the external codec.

4) Check for bit clock: Poll **psc_ac97stat**[CB] to see if this bit is set. Once set, configuration can continue.

5) Enable the AC97 controller by writing to **psc_ctrl**[ENA]. This brings the configuration registers out of reset. Poll **psc_ac97stat**[SR] to verify the PSC is out of reset.

6) Configure for DMA or programmed I/O (**psc_ac97cfg**[DD]), FIFO sizes, and set **psc_ac97cfg**[DE] to enable the device. Poll **psc_ac97stat**[DR] to determine if the device is ready.

The AC97 controller sends synchronization pulses during slot 0 and sets Tx slot valid bits in accordance with the AC97 protocol. All Tx and Rx data slots are cleared.

7.4.2.4 AC97 Power Management Considerations

The AC97 controller cannot be put into a low power state if the external codec is not in a power-down mode. The bit clock provided by the external codec must be off.

All AC97 data transactions must have stopped and the Tx/Rx controllers disabled before writing to the codec power-down register.

7.4.2.5 AC97 Transmit

The Tx and Rx controllers operate independently of each other. Each is enabled and disabled separately.

Before enabling the Tx controller, prepare the descriptors (if using DMA), or fill the Tx FIFO with data (if using programmed I/O).

Software controls the number of Tx FIFO slots and the order in which the slots are filled within the FIFO. **psc_ac97cfg**[TXSLOT] configures the number of enabled slots. Initialize Tx slots for programmed-I/O transfers before setting **psc_ac97pcr**[TS] to prevent an underflow condition. This ensures that FIFO data for left and right channels remain synchronized.

GPIO data is written separately using the GPIO output register **psc_ac97gpo**.

1) Set the **psc_ac97pcr**[TS] start bit to begin transmitting data for the start of the next frame. Poll **psc_ac97stat**[TB] for Tx controller ready.

2) Tx data is popped from the FIFO by the external codec.

3) Set the **psc_ac97pcr**[TP] stop bit to disable the Tx controller. Current frame is completed before controller is disabled. Poll **psc_ac97stat**[TB] to verify that the Tx controller has been disabled.

4) Set the **psc_ac97pcr**[TC] data clear bit to flush the Tx FIFO.

For underflow conditions, the Tx FIFO underflow event flag **psc_ac97evnt**[TU] is set. The corresponding interrupt can be masked by setting **psc_ac97msk**[TU]. The PSC controller records at which slot the underflow occurred and begins transmitting zeros and continues to the next frame up to the same underflow slot location.

Data is then checked in the Tx FIFO and for all frames at the same slot location. When data is available in the FIFO, the controller begins transmitting data at the next frame at the same slot location where the underflow occurred. This sequence prevents an out-of-sync condition with the Tx slot order.

www.DataSheet4U.com

www.DataSheet4U.com

7.4.2.6 AC97 Receive

Before enabling the Rx controller, prepare the descriptors if using DMA.

Software controls the number of Rx FIFO slots and the order in which the slots are filled within the FIFO. **psc_ac97cfg**[RXSLOT] configures the number of enabled slots. GPIO data is read separately using the GPIO input register **psc_ac97gpi**.

1) Set the **psc_ac97pcr**[RS] start bit to begin receiving data for the start of the next frame. Poll **psc_ac97stat**[RB] for Rx controller ready.
2) Rx data is popped from the FIFO by the DDMA controller or by software.
3) Set the **psc_ac97pcr**[RP] to disable the Rx controller. The current frame is completed before the controller is disabled. Poll **psc_ac97stat**[RB] to verify that the Rx controller has been disabled.
4) Set the **psc_ac97pcr**[RC] data clear bit to flush the Rx FIFO.

For overflow conditions, the Rx FIFO overflow event flag **psc_ac97evnt**[RO] is set. The corresponding interrupt can be masked by setting **psc_ac97msk**[RO]. The PSC controller records at which slot the overflow occurred and ignores all subsequent slots through the next frame to the slot location where the overflow occurred. This continues until data can be popped from the Rx FIFO.

Once space is available, the controller begins pushing new Rx data to the Rx FIFO at the start of the next frame. This process prevents an out-of-sync condition with the AC-link Rx slot order.

7.4.3 AC97 Signals

Table 7-8 shows the AC97 signals.

Table 7-8. AC97 Signals

PSC Generic	AC97 Specific
PSC*n*_CLK	ACBCLK
PSC*n*_SYNC0	ACSYNC
PSC*n*_SYNC1	ACRST#
PSC*n*_D0	ACDO
PSC*n*_D1	ACDI

www.DataSheet4U.com

www.DataSheet4U.cc

7.5 System Management Bus (SMBus)

When the SMBus protocol is selected for a PSC (**psc_sel**[PS] = 0b101), the PSC becomes an SMBus controller. The SMBus controller supports the following features:

- Two-signal interface consisting of clock and data signals (SCL and SDA) on a wired-AND bus (pulled high with external pull-up resistors)
- Compatible with the SMBus 2.0 and two-wire interface specifications
- Support for master and slave operation
- Multiple master environment support
- Supports a maximum clock rate of 2.5MHz (fast mode timings) using a clock source of 50 MHz
- Supports standard (100KHz) and fast (400KHz) two-wire speeds
- Independent programmable SMBus timing registers
- Supports 7-bit SMBus slave addressing
- Supports GENERAL CALL slave response
- Open-drain output signals allow for multiple master operation
- Supports back-to-back Master transfers (using RESTART)
- Supports data transfers using DMA or programmed I/O
- Local loopback capability for testing

7.5.1 SMBus Registers

Table 7-9 shows the SMBus registers.

Table 7-9. SMBus Registers

Offset (Note1)	Register	R/W	Description
0x0000	psc_sel	R/W	PSC Select (See Section 7.1.1 on page 194.)
0x0004	psc_ctrl	R/W	PSC Control (See Section 7.1.2 on page 195.)
0x0008	psc_smbcfg	R/W	SMBus Configuration Register
0x000C	psc_smbmsk	R/W	SMBus Mask Register
0x0010	psc_smbpcr	R/W	SMBus Protocol Control Register
0x0014	psc_smbstat	R/W	SMBus Status Register
0x0018	psc_smbevnt	R/W	SMBus Event Register
0x001C	psc_smbtxrx	R/W	SMBus Tx/Rx Data Register
0x0020	psc_smbtmr	R/W	SMBus Protocol Timers Register

Note1. See Table 7-1 on page 194 for base address.

7.5.1.1 SMBus Configuration Register

This register configures the operational parameters for the SMBus protocol.

psc_smbcfg **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
	RT		TT		DD	DE												DIV					GCE	SFM	SLV							
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:30	RT	Rx FIFO Threshold. 00 1 data entry 01 2 data entries 10 4 data entries 11 8 data entries For DMA transfers (**psc_smbcfg**[DD] = 0), the threshold must match the source transfer size (STS) programmed in the descriptor. For programmed I/O (**psc_smbcfg**[DD] = 1), once the number of data entries rises to the threshold, a *receive request* (RR) is triggered in the event and status registers.	R/W	0
29:28	TT	Tx FIFO Threshold. 00 1 empty slot 01 2 empty slots 10 4 empty slots 11 8 empty slots For DMA transfers (**psc_smbcfg**[DD] = 0), the threshold must match the destination transfer size (DTS) programmed in the descriptor. For programmed I/O (**psc_smbcfg**[DD] = 1), once the number of empty FIFO slots matches the threshold, a *transmit request* (TR) is triggered in the event and status registers.	R/W	0
27	DD	Disable DMA Transfers. 0 Enable DMA transfers. Software prepares buffers and descriptors, and the DDMA controller handles the transfers. 1 Programmed I/O. Software handles each transfer directly by using the data register and monitoring the FIFO events.	R/W	0
26	DE	Device Enable. 0 Disable the SMBus controller. 1 Enable the SMBus controller. Do not modify any other fields in **psc_smbcfg** while the SMBus controller is enabled (**psc_smbstat**[DR] = 1).	R/W	0
25:15	-	Reserved.	-	0
14:13	DIV	PSC Clock Divider. This setting determines the internal clock divider. The internal clock is generated by dividing the *pscn_mainclk* source. The resulting internal clock frequency determines the SMBus frequency settings. See Section 7.5.2.1 "SMBus Clocking" on page 240. 00 PSC_CLK = *pscn_mainclk* / 2 01 PSC_CLK = *pscn_mainclk* / 4 10 PSC_CLK = *pscn_mainclk* / 8 11 PSC_CLK = *pscn_mainclk* / 16	R/W	0
12:10	-	Reserved.	-	0
9	GCE	General Call Enable. Set this bit to enable the SMBus controller to respond to a general call address in slave mode. 0 Disable GENERAL CALL. 1 Enable GENERAL CALL.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
8	SFM	Standard/Fast Mode. Selects the protocol timer offsets for standard or fast mode. See Table 7-10 on page 246 for offset values. 0 Standard mode. 1 Fast mode.	R/W	0
7:1	SLV	Slave Address. The SMBus controller responds to this 7-bit address for slave transfers. To disable slave transfers for the SMBus, clear SLV and disable GENERAL CALL responses (GCE = 0).	R/W	0
0	-	Reserved.	-	0

7.5.1.2 SMBus Mask Register

This register is used to mask the interrupts triggered by the SMBus Event register (**psc_smbevnt**) flags. Setting a mask bit disables the corresponding interrupt. All interrupts not needed by an application should be masked.

For predictable interrupt behavior, **psc_smbmsk** should not be modified during an SMBus transfer.

psc_smbmsk **Offset = 0x000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		DN	AN	AL															RR	RO	RU	TR	TO	TU			SD	MD				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	-	Reserved.	-	-
30	DN	Data Not-acknowledge. Applies to master mode only.	R/W	0
29	AN	Address Not-acknowledged. Applies to master mode only.	R/W	0
28	AL	Arbitration Lost. Applies to master mode only.	R/W	0
27:14	-	Reserved.	-	0
13	RR	Mask Rx FIFO Request Interrupt.	R/W	0
12	RO	Mask Rx FIFO Overflow Interrupt.	R/W	0
11	RU	Mask Rx FIFO Underflow Interrupt.	R/W	0
10	TR	Mask Tx FIFO Request Interrupt.	R/W	0
9	TO	Mask Tx FIFO Overflow Interrupt.	R/W	0
8	TU	Mask Tx FIFO Underflow Interrupt.	R/W	0
7:6	-	Reserved.	-	0
5	SD	Mask Slave Done Interrupt.	R/W	0
4	MD	Mask Master Done Interrupt.	R/W	0
3:0	-	Reserved.	-	0

www.DataSheet4U.com

www.DataShe

7.5.1.3 SMBus Protocol Control Register

This register is used to control transfers and to clear the Rx and Tx FIFOs. Setting a bit issues a command to the controller. The bit is automatically cleared once the command is initiated but not necessarily completed.

psc_smbpcr **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:7	--	Reserved.	-	0
6	RC	Rx FIFO Clear. Set this bit to clear the receive FIFO. Do not set this bit while the SMbus is busy (in Slave or Master mode).	R/W	0
5:3	--	Reserved.	-	0
2	TC	Tx FIFO Clear. Set this bit to clear the transmit FIFO. Do not set this bit while the SMbus is busy (in Slave or Master mode).	R/W	0
1	--	Reserved.	-	
0	MS	Master Start. Set this bit to start the master controller.	R/W	0

7.5.1.4 SMBus Status Register

This register contains all the status signals for the SMBus protocol. It is read only. A status condition is true when the status bit is set. Before the SMBus controller is enabled in **psc_smbcfg**[DE], only the device-ready and PSC-ready status bits (**psc_smbstat**[DR, SR]) are valid; all other status bits read as 0 until ready.

psc_smbstat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				BB															RF	RE	RR	TF	TE	TR			SB	MB		DI	DR	SR
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:29	--	Reserved.	-	0
28	BB	SMBus Bus Busy. Indicates a transfer is in progress on the SMBus bus.	R	0
27:14	--	Reserved.	-	0
13	RF	Receive FIFO Full.	R	0
12	RE	Receive FIFO Empty.	R	1
11	RR	Receive Request.	R	0
10	TF	Transmit FIFO Full.	R	0
9	TE	Transmit FIFO Empty.	R	1
8	TR	Transmit Request.	R	0
7:6	--	Reserved.	-	0
5	SB	Slave Busy. Indicates the SMBus slave controller is not idle.	R	0
4	MB	Master Busy. Indicates the SMBus master controller is not idle.	R	0
3	--	Reserved.	-	0
2	DI	Device Interrupt. Indicates at least one unmasked event has been flagged in **psc_smbevnt**.	R	0
1	DR	Device Ready. Indicates the SMBus controller is ready for protocol operation after setting the device-enable bit **psc_smbcfg**[DE].	R	0
0	SR	PSC Ready. Indicates the PSC is ready for protocol configuration after enabling the PSC in **psc_ctrl**[ENA].	R	0

www.DataSheet4U.com

www.DataSheet4U.com

7.5.1.5 SMBus Event Register

This register contains the event flags for the SMBus protocol. When an event flag is set, a corresponding interrupt is generated unless masked in the SMBus Mask register (**psc_smbmsk**).

Once an event flag has been set, it remains set until cleared. A flag is cleared by writing a 1 to the appropriate bit; writing a 0 has no effect.

psc_smbevnt **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		DN	AN	AL															RR	RO	RU	TR	TO	TU			SD	MD				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	--	Reserved.	-	0
30	DN	Data Not-acknowledged. Applies to master mode only. Indicates Tx data sent from the SMBus in master mode was not acknowledged by the slave.	R/W	0
29	AN	Address Not-acknowledged. Applies to master mode only. Indicates the slave address sent from the SMBus in master mode was not acknowledged by any slave.	R/W	0
28	AL	Arbitration Lost. Applies to master mode only. Indicates the SMBus in master mode has lost arbitration and could not successfully retry a transfer.	R/W	0
27:14	--	Reserved.	-	0
13	RR	Receive Request. Applies to programmed I/O only (**psc_smbcfg**[DD] = 1). Indicates the Rx FIFO has a number of data elements equal to the FIFO threshold programmed in **psc_smbcfg**[RT] available for reading.	R/W	0
12	RO	Receive Overflow. Indicates the Rx FIFO has experienced an overflow.	R/W	0
11	RU	Receive Underflow. Indicates the Rx FIFO has experienced an underflow.	R/W	0
10	TR	Transmit Request. Applies to programmed I/O only (**psc_smbcfg**[DD] = 1). Indicates the Tx FIFO is requesting a number of data elements equal to the FIFO threshold programmed in **psc_smbcfg**[TT].	R/W	0
9	TO	Transmit Overflow. Indicates the Tx FIFO has experienced an overflow.	R/W	0
8	TU	Transmit Underflow. Indicates the Tx FIFO has experienced an underflow.	R/W	0
7:6	--	Reserved.	-	0
5	SD	Slave Done. This flag indicates that a slave transfer has completed.	R/W	0
4	MD	Master Done. This flag indicates that a master transfer has completed.	R/W	0
3:0	--	Reserved.	-	0

7.5.1.6 SMBus Tx/Rx Data Register

This register is the interface to the transmit and receive FIFOs. Each FIFO is 16 entries deep. A write to this register causes data to be pushed onto the Tx FIFO; if the Tx FIFO is already full, the data is lost and a transmit-overflow event is triggered (**psc_smbevnt**[TO] = 1). A read from this register causes data to be popped from the Rx FIFO; if the Rx FIFO is already empty, the data read is undefined and a receive-underflow event is triggered (**psc_smbevnt**[RU] = 1).

If transfers are under program control (**psc_smbcfg**[DD] = 1), software can monitor the transmit-request (TR) and receive-request (RR) flags to maintain proper FIFO levels either by polling **psc_smbstat**[TR, RR] or by waiting for interrupts in **psc_smbevnt**[TR, RR]. The DDMA controller (**psc_smbcfg**[DD] = 0) automatically manages overflow/underflow conditions.

The data size of read/write transfers is always 32-bits to the register; however, the SMBus data payload is fixed at 8 bits, and the upper 24 bits must be cleared except for the data flow control bits (STP and RSR). For programmed I/O, these bits can be written to control a Master transfer. The data flow control bits are taken care of automatically when using DMA.

www.DataSheet4U.com

www.DataSheet4U.com

When DMA transfers are enabled, the **psc_smbtxrx** address locations must be programmed into the Tx descriptor destination pointer and the Rx descriptor source pointer.

psc_smbtxrx **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
			STP	RSR																					ADDR/DATA							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:30	—	Reserved.	—	—
29	STP	Stop. Applies to master mode only. Write a 1 to this bit location with the last byte of data for a transfer (handled automatically by the DDMA controller). After processing this byte, the SMBus master controller generates a STOP condition on the bus, and then the controller goes idle. This bit always reads 0.	W	0
28	RSR	Restart. Applies to master mode only. Write a 1 to this bit location with the last byte of data for a transfer (handled automatically by the DDMA controller). After processing this byte, the SMBus master controller generates a RESTART condition on the bus, and then waits for the next Tx data to be processed. This bit always reads 0.	W	0
27:8	—	Reserved.	—	—
7:0	ADDR/ DATA	Contains the Tx/Rx data or slave address packet for the SMBus protocol. An address packet is needed for the first data element of each SMBus Master transfer. The address packet must contain a slave address (or GENERAL CALL address) in the upper 7 bits and a read/write bit in the lowest bit position. An address packet must precede every SMBus Master transfer, even after a RESTART condition to the same slave.	R/W	0

7.5.1.7 SMBus Protocol Timers Register

This register determines the timer values for the SMBus interface. Added to the standard- or fast-mode offsets chosen in **psc_smbcfg**[SFM], the timer values adjust the SMBus clock (high/low) period, setup/hold and buffer times for START and STOP conditions, and transmit-data hold time. See Section 7.5.3 "SMBus Timing" on page 246.

psc_smbtmr **Offset = 0x0020**

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

Bits	Name	Description	R/W	Default
31:30	TH	Transmit Data Hold Timer. Determines the amount of hold time between SMBus clock fall and data transition.	R/W	0
29:25	PS	Stop->Start Buffer Timer. Determines the amount of time between a Stop detection and new Start generation on the SMBus. (Master mode only)	R/W	0
24:20	PU	Stop Setup Timer. Determines the amount of setup time between SMBus clock rise and a Stop generation on the SMBus. (Master mode only)	R/W	0
19:15	SH	Start Hold Timer. Determines the amount of hold time between Start generation and SMBus clock fall. (Master mode only)	R/W	0
14:10	SU	Start Setup Timer. Determines the amount of setup time between SMBus clock rise and re-Start generation. (Master mode only)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
9:5	CL	Clock Low. Determines the amount of time the SMBus clock is driven low. (Master mode only)	R/W	0
4:0	CH	Clock High. Determines the amount of time the SMBus clock is left high. (Master mode only)	R/W	0

7.5.2 SMBus Operation

7.5.2.1 SMBus Clocking

SMBus Master Clocking

Figure 7-15 shows the clocking options for the SMBus master controller.

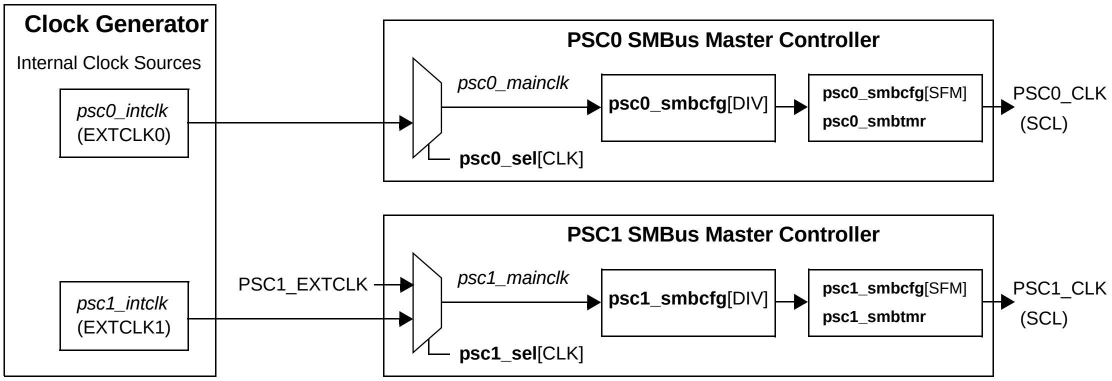


Figure 7-15. SMBus Master Controller Clock Options

www.DataSheet4U.com

www.DataSheet4U.com

7.5.2.2 SMBus Flowcharts

Figure 7-16 shows a configuration flowchart for the SMBus controller. See also Section 7.5.2.3 "SMBus Configuration" on page 244.

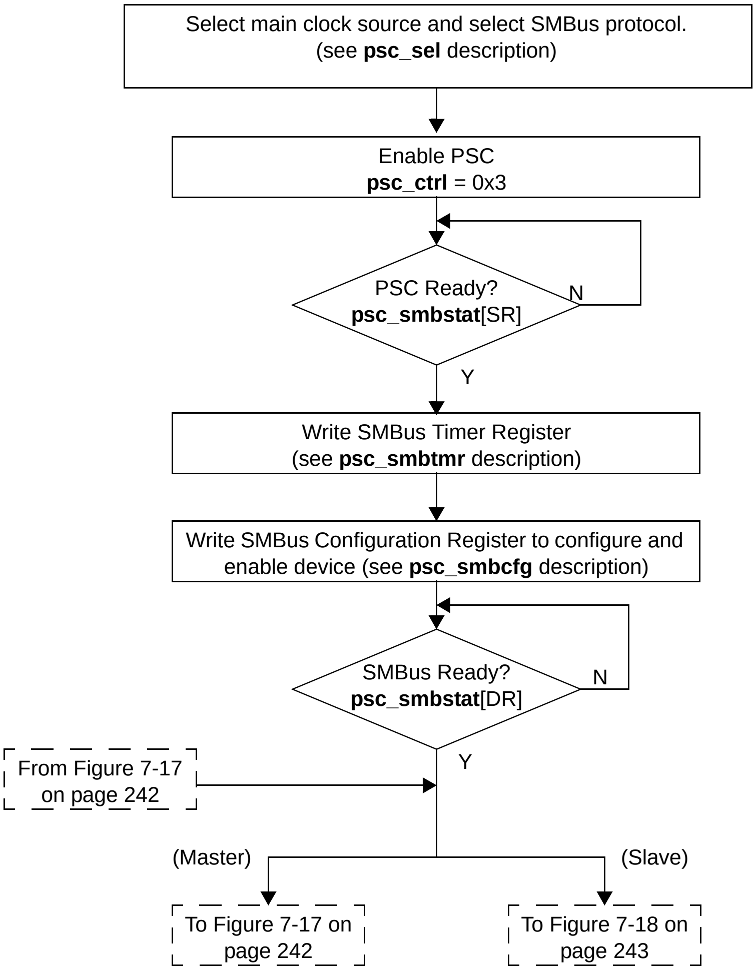
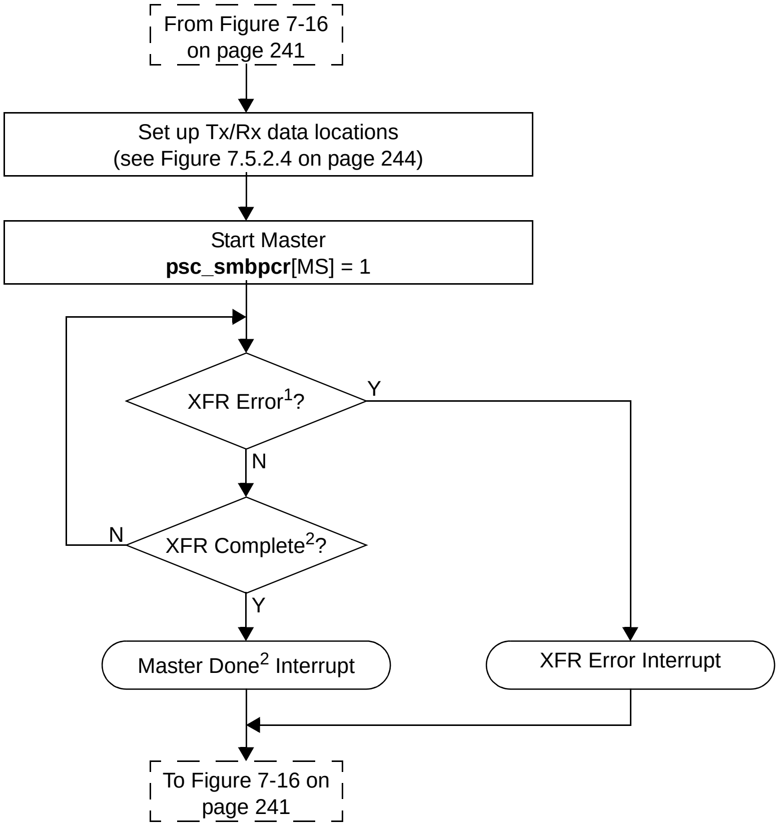

Select main clock source and select SMBus protocol.
(see **psc_sel** description)
Enable PSC
psc_ctrl = 0x3
PSC Ready?
psc_smbstat[SR]
N
Y
Write SMBus Timer Register
(see **psc_smbtmr** description)
Write SMBus Configuration Register to configure and enable device (see **psc_smbcfg** description)
SMBus Ready?
psc_smbstat[DR]
N
Y
From Figure 7-17 on page 242
(Master)
(Slave)
To Figure 7-17 on page 242
To Figure 7-18 on page 243

Figure 7-16. SMBus Configuration

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-17 shows a flowchart for the SMBus master controller. See also Section 7.5.2.4 "SMBus Master Operation" on page 244.


From Figure 7-16 on page 241
Set up Tx/Rx data locations (see Figure 7.5.2.4 on page 244)
Start Master
psc_smbpcr[MS] = 1
XFR Error1?
Y
N
XFR Complete2?
N
Y
Master Done2 Interrupt
XFR Error Interrupt
To Figure 7-16 on page 241

[3] A Master Done interrupt is generated, unless masked, as follows:

DMA: When the Rx FIFO is empty

Programmed I/O: Immediately following the generation of STOP condition

[1] An XFR error interrupt occurs, unless masked, when any of the following happen: Arbitration Lost, Address NACK, TX Data NACK.

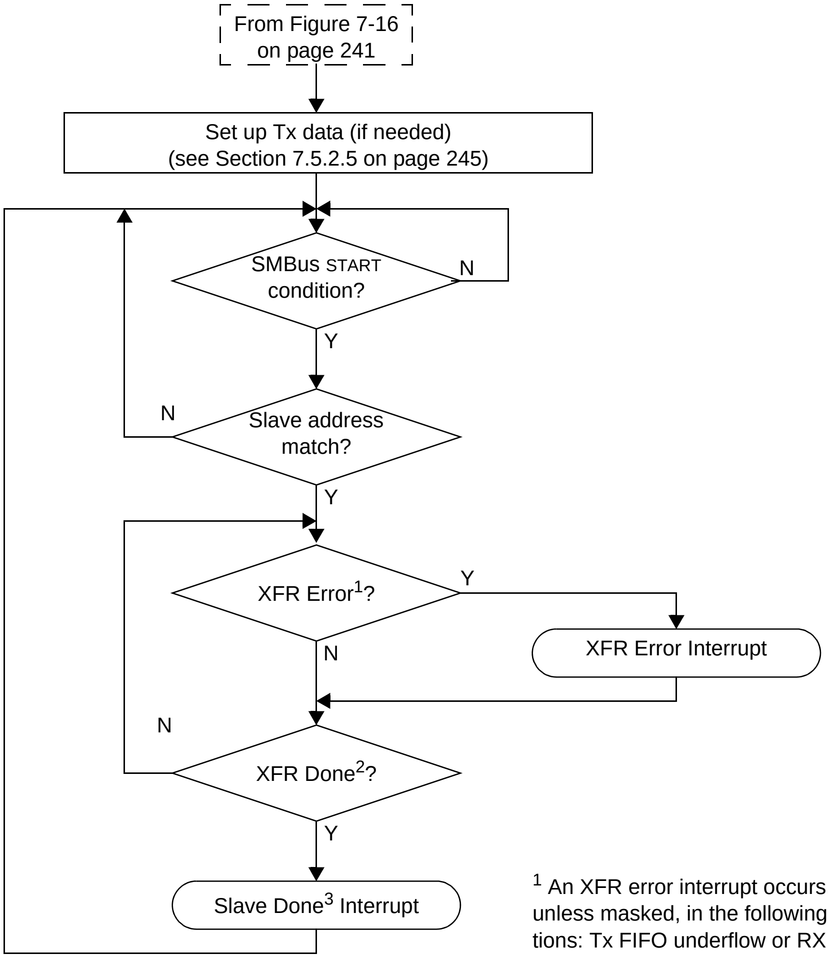
[2] A Master XFR is complete when the last TX data element has been processed and a STOP condition has been generated.

Figure 7-17. SMBus Master Operation

www.DataSheet4U.com

www.DataSheet4U.com

Figure 7-18 shows a flowchart for the SMBus slave controller. See also Section 7.5.2.5 "SMBus Slave Operation" on page 245.



[1] An XFR error interrupt occurs, unless masked, in the following situations: Tx FIFO underflow or RX FIFO overflow

[2] A Slave XFR is done when the SMBus controller detects a STOP (or RESTART) condition on the SMBus.

[3] A Slave done interrupt is generated, unless masked, in the following way:

DMA: When the RX FIFO is empty (if receiving)

Programmed I/O: Immediately after the Slave XFR is done.

Figure 7-18. SMBus Slave Operation

www.DataSheet4U.com

www.DataSheet4U.com

7.5.2.3 SMBus Configuration

SMBus transfers can be done under program control (**psc_smbcfg**[DD = 1]) or using the DDMA controller (**psc_smbcfg**[DD = 0]).

The first step to configuring the PSC as an SMBus controller is to write the select register (**psc_sel**). In this case software selects the SMBus protocol and the *psc*n*_mainclk* source (see Section 7.5.2.1 "SMBus Clocking" on page 240).

The next step is to enable the PSC by writing the Control register (**psc_ctrl**). By writing 0b11 to **psc_ctrl**[ENA] the SMBus configuration registers are automatically reset and then brought into the configuration state. Software must poll the Status register (**psc_smbstat**[SR]) to determine when the reset has completed and the SMBus controller is ready for configuration.

To configure the SMBus controller, program the SMBus Protocol Timer (**psc_smbtmr**) register and Configuration register (**psc_smbcfg**) for the desired speed. When writing the Configuration register, also set the device-enable bit (**psc_smbcfg**[DE]) to enable the SMBus controller. Poll the Status register until (**psc_smbstat**[DR] = 1) to determine when the SMBus controller is ready.

At this point the SMBus controller is ready to respond its slave address (or GENERAL CALL address if enabled) or to begin a master transfer.

7.5.2.4 SMBus Master Operation

The SMBus controller is capable of performing SMBus master transfers. These transfers are required to start with an address control byte (a 7-bit slave address with read/write bit). The SMBus also supports back-to-back Master transfers where a RESTART condition is generated on the bus. In order to perform an SMBus Master transfers follow the steps outlined below.

DMA Transfers

If using DMA, build the necessary descriptors. Configure a separate Tx descriptor for each Master transfer with the address control byte as the first Tx data element. The address control byte is needed for both transmit and receive transfers. Following the address control byte are *n* data bytes. For a master write transfer, these data bytes contain the data payload. For a master read transfer these bytes are used for timing purposes only and can be left uninitialized. Also for a read transfer, configure an Rx descriptor with *n* data bytes. The size of the Rx descriptor must match the Tx descriptor except for the address control byte: (*n*+1) bytes for Tx, *n* bytes for Rx.

- DMA Back-to-Back Transfers

In order to perform back-to-back transfers using DMA, prepare multiple Tx (and Rx for reads) descriptors. Each Tx descriptor must start with an address control byte. The SMBus automatically detects the end of a descriptor and performs a RESTART on the bus between transfers. This allows the SMBus to hold the bus for multiple transfers without risk of losing arbitration to another master.

Programmed I/O Transfers

To perform a master transfer under program control, software writes directly to the Tx FIFO and reads from the Rx FIFO. To set up for a master transfer write the address control byte to the Tx FIFO. After the transfer has begun, continue to write bytes to the Tx FIFO. For a write transfer these bytes contain the write data payload. For a read transfer these bytes are used for timing purposes only and must be written 0s.

To end a transfer, write a 1 to **psc_smbtxrx**[STP] data flow control flag to indicate the last byte. Following this byte, the SMBus controller generates a STOP condition for transmit transfers or a NOT-ACKNOWLEDGE condition for receive transfers.

For a read transfer, software must also read Rx data from the Rx FIFO. Read one Rx data byte for each Tx byte written to the Tx FIFO, excluding the address control byte: (*n*+1) bytes for Tx, *n* bytes for Rx.

- Programmed I/O Event Flags

Alternatively, software can write one byte at a time to the Tx FIFO and monitor the SMBus event flags to keep track of the transfer. In between each Tx byte, if the Tx FIFO underflows (**psc_smbevnt**[TU] = 1), the SMBus controller holds the bus (SCL low) until software prepares new Tx data and clears the event. Also, for programmed I/O read transfers, if the Rx FIFO overflows (**psc_smbevnt**[RO] = 1), the SMBus controller holds the bus (SCL low) until software empties an Rx FIFO entry and clears the event.

- Programmed I/O Back-to-Back Transfers

To perform a back-to-back Master transfer in programmed I/O mode, the last byte of the first (or subsequent) transfer is written with a 1 in the restart data flow control bit (**psc_smbtxrx**[RSR]). This causes the SMBus master controller to generate a RESTART condition on the bus after this byte has been processed. For a read transfer, the SMBus generates a NOT-ACKNOWLEDGE condition for the last Rx data byte.

www.DataSheet4U.com

www.DataSheet4U.com

Back-to-back master transfers can be chained together, as needed, using this flow control protocol. Write the stop data flow control bit (**psc_smbtxrx**[STP]) for the last Tx byte of the last transfer in the chain. This causes the SMBus controller to generate a STOP condition on the bus following the last byte for a write transfer or a NOT-ACKNOWLEDGE for a read transfer.

Initiating a Master Transfer

Once the data has been set up for a master transfer, set the master-start bit (**psc_smbpcr**[MS]) to start the SMBus master controller. The SMBus controller begins the master transfer as soon as the Tx address control byte is present in the Tx FIFO. At this point the SMBus controller arbitrates for the bus (or waits if the bus is busy) and then proceeds with the master transfer.

Upon successful completion of a master transfer, a master-done interrupt is generated unless masked. When using DMA the master-done interrupt is generated after a STOP condition has been generated on the bus and all Rx data (if receiving) has been written to memory. When using the SMBus in programmed I/O mode, a master-done interrupt is generated immediately following the STOP condition on the bus (regardless of Rx FIFO status).

If a master transfer fails (arbitration lost, address NOT-ACKNOWLEDGE, or data NOT-ACKNOWLEDGE), the appropriate interrupt is generated unless masked, and the transfer terminates without completing.

7.5.2.5 SMBus Slave Operation

The SMBus controller is ready for Slave mode accesses once it has been enabled. At this point the SMBus controller responds to its slave address (or a GENERAL CALL if enabled). Data flow must be prepared in order for the SMBus controller to respond to a slave transfer.

DMA Transfers

For Slave write transfers, prepare Tx descriptors as needed. For Slave read transfers, prepare Rx descriptors.

The Tx buffers are transmitted if the SMBus controller is addressed as a Slave for a Master read transfer (SMBus Slave -> external Master). Each Tx data byte is sent out during the Master read. If the Master read completes before a Tx buffer finishes, the buffer remains open for the next Master read transfer.

The Rx buffers are used if the SMBus controller is addressed as a Slave for a Master write transfer (external Master -> SMBus Slave). Each Rx data byte is written to the Rx buffer. If the Master write completes before the end of a Rx buffer, the Rx buffer is closed and the next Rx buffer is used for the next Master write transfer.

Programmed I/O Transfers

For programmed I/O, software writes the Tx data bytes directly to the Tx FIFO or reads Rx data bytes directly from the Rx FIFO via the **psc_smbtxrx** data register. Software can send one byte at a time by writing to **psc_smbtxrx** and then waiting for the Tx FIFO underflow event (**psc_smbevnt**[TU] = 1). The SMBus controller holds the bus (SCL low) until a new Tx data byte is forthcoming. Also the SMBus controller holds the bus (SCL low) if an Rx FIFO overflow occurs (**psc_smbevnt**[RO] = 1). Once there is room in the Rx FIFO, the SMBus controller releases SCL and continues.

Transfer Completion

The SMBus controller generates a slave-done event (**psc_smbevnt**[SD] = 1) at the completion of each slave transfer (detects a STOP or RESTART on the bus). When using DMA, the interrupt is generated once all Rx data has been written to memory (Master write to SMBus), or immediately following the STOP/RESTART detection (Master read from SMBus). If using the SMBus in programmed I/O mode, the Slave Done interrupt is generated immediately following the STOP/RESTART detection regardless of Rx FIFO status.

www.DataSheet4U.com

www.DataSheet4U.com

7.5.3 SMBus Timing

Figure 7-19 shows the SMBus format.

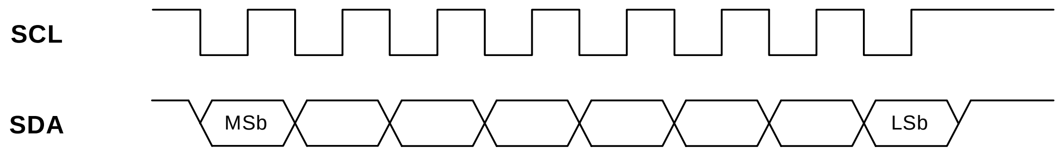


Figure 7-19. SMBus Format

Each of the timing parameters in **psc*n*_smbtmr** is programmable in time units of *psc*n_*mainclk* divided by the factor programmed in **psc*n*_smbcfg**[DIV]. Table 7-10 shows the range of the protocol timers with the standard or fast mode offsets.

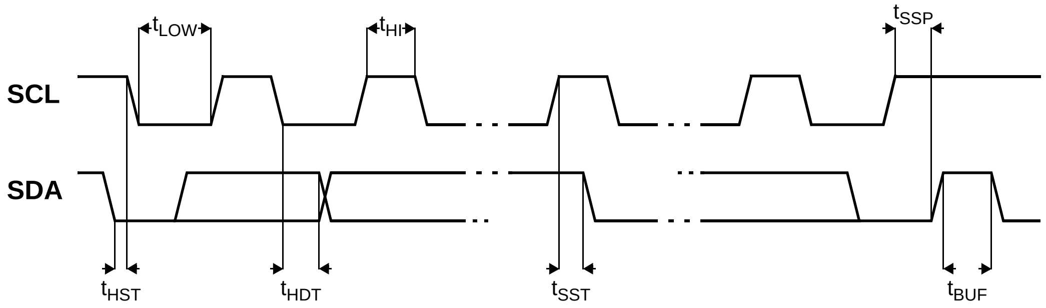


Figure 7-20. SMBus Timing

Table 7-10. SMBus Timing Parameter Ranges

		Standard Mode (100 kbs)		**Fast Mode (400 kbs)**	
Timer Name	**Symbol**	**psc_smbtmr + Offset**	**Range**	**psc_smbtmr + Offset**	**Range**
Tx Data Hold	t_{HDT}	TH + 3	3 to 6	TH + 6	6 to 9
Stop -> Start Buf	t_{BUF}	PS + 15	15 to 46	PS + 18	18 to 49
Stop Setup	t_{SSP}	PU + 10	10 to 41	PU + 4	4 to 35
Start Hold	t_{HST}	SH + 10	10 to 41	SH + 4	4 to 35
Start Setup	t_{SST}	SU + 15	15 to 46	SU + 4	4 to 35
Clock Low	t_{LOW}	CL + 15	15 to 46	CL + 18	18 to 49
Clock High	t_{HI}	CH + 10	10 to 41	CH + 4	4 to 35

www.DataSheet4U.com

www.DataSheet4U.com

Given the above offsets, there are pre-determined settings for use with standard and fast mode. The values in Table 7-11 are for use with a 50MHz *pscn_mainclk* and with the appropriate clock divider (**psc*n*_smbcfg**[DIV]) shown below. These settings are pre-determined to meet the SMBus standard mode and two-wire fast mode specifications of clock length, setup and hold times.

Table 7-11. SMBus Timing Parameter Values

		Standard Mode (*pscn_mainclk*/8) = 6.25MHz Time unit = 160ns		**Fast Mode (*pscn_mainclk*/2) = 25MHz Time unit = 40ns**	
Timer Name	**Symbol**	**psc_smbtmr**	**Bus timings**	**psc_smbtmr**	**Bus timings**
Tx Data Hold	t_{HDT}	TH = 0x0	480ns	TH = 0x2	320ns
Stop -> Start Buf	t_{BUF}	PS = 0x0F	4800ns	PS = 0x0F	1320ns
Stop Setup	t_{SSP}	PU = 0x0F	4000ns	PU = 0x0B	600ns
Start Hold	t_{HST}	SH = 0x0F	4000ns	SH = 0x0B	600ns
Start Setup	t_{SST}	SU = 0x0F	4800ns	SU = 0x0B	600ns
Clock Low	t_{LOW}	CL = 0x0F	4800ns	CL = 0x0F	1320ns
Clock High	t_{HI}	CH = 0x0F	4000ns	CH = 0x0B	600ns

7.5.4 SMBus Signals

Table 7-12 shows the SMBus signals.

Table 7-12. SMBus Signals

PSC Generic	SMBus Specific
PSC*n*_CLK	SCL
PSC*n*_D0	SDA

www.DataSheet4U.com

www.DataSheet4U.com

7.6 PSC Signal Mapping

Table 7-13 shows how the generic PSC signals map to the corresponding protocol-specific signal names.

Table 7-13. PSC Signal Mapping

PSC Instance	AC97	I^2S	SMBus	SPI
PSC*n*_CLK	ACBCLK	I2SCLK	SCL	SPICLK
PSC*n*_SYNC1	ACRST#	—	—	SPISEL# (Master)
PSC*n*_SYNC0	ACSYNC	I2SWORD	—	SPISEL# (Slave)
PSC*n*_D1	ACDI	I2SDI	—	SPIMISO
PSC*n*_D0	ACDO	I2SDO	SDA	SPIMOSI
PSC1_EXTCLK	—	I2SMCLK(Note1) (input on PSC1_EXTCLK)	—	—
pscn_intclk	—	I2SMCLK[1] (output on EXTCLK*n*)	—	—

Note1. For PSC0, the I2SMCLK signal for the I^2S controller in master mode must be generated internally on *psc0_intclk* and output via EXTCLK0 (GPIO[2]). For PSC1, the system designer has two options for the I2SMCLK clock source: It can be generated internally on *psc1_intclk* and output via EXTCLK1 (GPIO[3]), or it can be provided externally and input via PSC1_EXTCLK. See Section 7.3 "Inter-IC Sound Controller (I^2S)" on page 207.

Table 7-14 shows how the PSC signals are multiplexed with GPIO signals. The signal functionality is selected in **sys_pinfunc** (see Section 10.3.1 "Pin Functionality" on page 381).

Table 7-14. PSC Signal Muxing(Note1)

PSC Signal	PSC0	PSC1
PSC*n*_CLK	GPIO[25]	GPIO[24]
PSC*n*_SYNC1	GPIO[16]	GPIO[21]
PSC*n*_SYNC0	GPIO[215]	GPIO[20]
PSC*n*_D1	GPIO[31]	GPIO[22]
PSC*n*_D0	GPIO[18]	GPIO[11]
PSC1_EXTCLK	—	GPIO[23]

Note1. Signals shown in bold are the default signal function out of reset.

www.DataSheet4U.com

www.DataSheet4U.com

7.7 System Programming Considerations

7.7.1 PSC to DDMA Controller Connections

Each PSC has two request lines connected to the DDMA controller on the Au1210 and Au1250 processors. The connections are defined in Table 7-15 for each PSC present.

Table 7-15. PSC Base Addresses Description

PSC Instance	Data Path	DDMA Device ID
PSC0	DDMA to PSC Tx	14
PSC0	PSC Rx to DDMA	15
PSC1	DDMA to PSC Tx	16
PSC1	PSC Rx to DDMA	17

7.7.2 PSC Main Clock Sources

PSC1 has two clock sources that can be selected as the main input clock (*psc*n*_mainclk*). PSC0 has only the internal source. For the internal clock source option (*psc*n*_intclk*) each PSC is connected to a different clock output from the internal clock generator. These outputs are shared with the EXTCLK*n* to allow the internal PSC clocks to be output on external pins. For PSC1, the external clock source PSC1_EXTCLK is multiplexed with GPIO[23], with the signal function selected in **sys_pinfunc**. Table 7-16 lists the connections of each PSC and the clock sources.

Table 7-16. PSC mainclk Sources

PSC	Clock from Internal Clock Generator	PSC External Clock (Muxed GPIO)
PSC0	psc0_intclk (shared with EXTCLK0)	—
PSC1	psc1_intclk (shared with EXTCLK1)	PSC1_EXTCLK (**GPIO[23]**)

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

8 USB 2.0 Subsystem

8.1 Universal Serial Bus 2.0 Controller

The Universal Serial Bus Controller subsystem consists of a USB 2.0 Enhanced Host Controller Interface (EHCI) compliant host controller with a USB 1.1 Open Host Controller Interface (OHCI) compliant companion controller, a USB 2.0 device controller with On-The-Go (OTG) support.

There are two USB 2.0 ports. Depending on the type of device attached, the ports are either associated with the EHC or OHC host controller. Furthermore Port 2 can be configured as a device. The functional descriptions of the blocks are described in the following subsections. For this chapter the term *DWORD* means four bytes.

8.1.1 Host Controller

The host controller is responsible for:

- Detecting the attachment and removal of USB devices
- Collecting status and activity statistics
- Controlling power supply to attached USB devices
- Controlling the association to either the Open Host Controller Interface or the Enhanced Host Controller via a Port Router
- Root Hub functionality to support up to two ports

The USB system software on the host manages interactions between USB devices and host-based device software. There are five areas of interactions between the USB system software and device software:

- Device enumeration and configuration
- Isochronous data transfers
- Asynchronous data transfers
- Power management
- Device and bus management information

Whenever possible, the USB system software uses existing host system interfaces to manage the above interactions.

Attached devices are recognized by the USB PHY as either USB 1.1 compliant (full speed and low speed) or USB 2.0 compliant (high speed) devices. Low or full speed device attached USB ports are associated with the OHC and high speed devices with the EHC.

The interface combines responsibility for executing bus transactions requested by the Host Controller as well as the hub and port management specified by USB.

8.1.2 Device Controller

USB Port 2 can be configured alternatively as a device port. The device supports four bidirectional endpoints (ep1 ... ep4) plus the default endpoint ep0. Endpoint ep0 only supports control traffic. The endpoints ep1...ep4 can be programmed to support either control, bulk, isochronous, or interrupt traffic. The maximum packet size is only restricted by the sum of all IN endpoints maximum packet sizes is less than 1.5 KBytes. Independent of the OTG functionality, the USB Port 2 can be configured as a device by programming **otg_mux**[PMUX].

www.DataSheet4U.com

www.DataSheet4U.com

8.1.3 On-The-Go Controller

The basic OTG functionality is to allow software to assign the shared USB Port 2 to either the host controller or the device controller. The OTG controller contains all resources to support software for the OTG protocol, i.e., it provides status and control logic to the software for implementing the OTG state machines including an interval timer.

8.1.3.1 Port Multiplexing

The **otg_mux**[PMUX] and the alias **otg_ctl**[PMC] are used to control the assignment of the USB Port 2 to either the host controller, device controller or to inactivate that port.

8.1.3.2 Host Negotiation Protocol

The control bit **otg_ctl**[HSF] enables the device to accept the SET_FEATURE commands for the Host Negotiation Protocol, i.e., it will respond to them with ACK, when the control bit is deasserted the device will respond with STALL.

When software has enabled the SET_FEATURE commands, the receipt of the individual SET_FEATURE commands is stored in the **otg_sts**[HNPSTS] field. The bits are cleared when either the device has received a USB reset or when **otg_ctl**[HSF] has been deasserted.

8.1.3.3 Timer Functionality

The timer and its control bits assist software in detecting time-outs while waiting for other conditions to become true (for example, waiting for connect). The timer works with an accuracy of 10 µs and can cover an interval of approximately 10 ms. It is software’s responsibility to time longer intervals by means of software counters.

The interval is taken from **otg_tmr**[PRELD]. A new timing cycle with this duration is started whenever a non-zero value is written into **otg_ctl**[TMRCOND].

8.2 USB 2.0 Controller Registers

The USB 2.0 register memory map is shown in Table 8-1. Software must make only 32-bit write accesses to the USB configuration registers. Writes of sizes smaller than 32-bits may lead to undefined register contents.

Table 8-1. USB 2.0 Register Memory Map

Name	Physical Base Address	KSEG1 Base Address
usb_base	0x0 1402 0000	0xB402 0000
usb_otgbase	0x0 1402 0020	0xB402 0020
usb_ohcibase	0x0 1402 0100	0xB402 0100
usb_ehcibase	0x0 1402 0200	0xB402 0200
usb_devbase	0x0 1402 2000	0xB402 2000

8.2.1 USB 2.0 Global Registers

Table 8-2. USB 2.0 Global Registers

Offset (Note1)	Register Name	Description
0x0000	usb_cap	USB 2.0 Capabilities Register
0x0004	usb_cfg	USB 2.0 Configuration Register

Note1. See Table 8-1 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

8.2.1.1 USB 2.0 Capabilities Register

usb_cap **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
									DEVID				CLASS								ARCH											
Def.	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	1

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:20	DEVID	Device code.	R	0x5
19:12	CLASS	Class code. USB 2.0	R	0x43
11:8	ARCH	Architecture code.	R	0x5
7:0	—	Reserved.	—	—

8.2.1.2 USB 2.0 Configuration Register

usb_cfg **Offset = 0x0004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SSD				PPE	UCE	ECE	OCE			FLA							GME	DBE	DME	EBE	EME	OBE	OME
Def.	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23	SSD	Serial Short Detect Enable. When set, this bit enables the short detection circuit for the serial PHY interface.	R/W	1
22:20	—	Reserved. If modifying **usb_cfg**, software must preserve the value of this field.	R/W	0b101
19	PPE	USB PHY PLL Enable. When set, this bit enables the PHY PLL to run at 480MHz. Software must ensure that this bit is set if at least one port is used for HS host traffic (downstream) or HS and FS device traffic (upstream), i.e., when either ECE or UCE are set. If the USB PHY PLL is not needed, system software can clear this bit to conserve power.	R/W	1
18	UCE	UDC Clock Enable. When set, this bit enables UDC (USB 2.0 device) operation by enabling the respective clocks to the core (60MHz USB clock and 66MHz MBus clock). If the UDC is not needed, system software can clear this bit to conserve power.	R/W	1
17	ECE	EHC Clock Enable. When set, this bit enables EHC (USB 2.0 host) operation by enabling the respective clocks to the core (60MHz USB clock and 66MHz MBus clock). If the EHC is not needed, system software can clear this bit to conserve power.	R/W	1
16	OCE	OHC Clock Enable. When set, this bit enables OHC (USB 1.1 host) operation by enabling the respective clocks to the core (48MHz USB clock and 66MHz MBus clock). If the OHC is not needed, system software can clear this bit to conserve power.	R/W	1
15:14	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
13:8	FLA	Frame Length Adjustment. The SOF cycle time is equal to 59488 + (16*this value). The default of 20h gives a SOF cycle time of 60000. FrameLength (HS bit times)FLA value 59488 00h 59504 01h 59520 02h … … 59984 1Fh 60000 20h … … 60480 3Eh 60496 3Fh	R/W	0x20
7	—	Reserved.	—	—
6	GME	UOC Memory Enable. If set, memory space is enabled. If cleared, accesses to the memory space are blocked.	R/W	0
5	DBE	UDC Bus Master Enable. If set, the arbiter is allowed to arbitrate the UDC bus master.	R/W	0
4	DME	UDC Memory Enable. If set, memory space is enabled. If cleared, accesses to the memory space are blocked.	R/W	0
3	EBE	EHC Bus Master Enable. If set, the arbiter is allowed to arbitrate the EHC bus master.	R/W	0
2	EME	EHC Memory Enable. If set, memory space is enabled. If cleared, accesses to the memory space are blocked.	R/W	0
1	OBE	OHC Bus Master Enable. If set, the arbiter is allowed to arbitrate the OHC bus master.	R/W	0
0	OME	OHC Memory Enable. If set, memory space is enabled. If cleared, accesses to the memory space are blocked.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

8.2.2 USB 2.0 Host Controller Interface Registers

The Au1210 and Au1250 processors' USB 2.0 Host Controller implements the OpenHCI register set. Consult the *OpenHCI - Open Host Controller Interface Specification for USB* for register details and programming considerations.

Table 8-3. USB 2.0 OpenHCI Registers

Offset from usb_ohcibase (Note1)	**Register Name**
0x0000	HcRevision
0x0004	HcControl
0x0008	HcCommandStatus
0x000C	HcInterruptStatus
0x0010	HcInterruptEnable
0x0014	HcInterruptDisable
0x0018	HcHCCA
0x001C	HcPeriodCurrentED
0x0020	HCControlHeadED
0x0024	HcControlCurrentED
0x0028	HcBulkHeadED
0x002C	HcBulkCurrentED
0x0030	HcDoneHead
0x0034	HcFmInterval
0x0038	HcFmRemaining
0x003C	HcFmNumber
0x0040	HcPeriodicStart
0x0044	HcLSThreshold
0x0048	HcRhDescriptorA
0x004C	HcRhDescriptorB
0x0050	HcRhStatus
0x0054	HcRhPortStatus[1]
0x0058	HcRhPortStatus[2]

Note1. See Table 8-1 on page 252 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

8.2.3 USB 2.0 Enhanced Host Controller Interface Registers

The USB 2.0 Host Controller implements all of the necessary EHCI registers. Consult the *Enhanced Host Controller Interface Specification for USB* for register details and programming considerations. Registers that differ from the EHCI specification are described in Table 8-4.

Table 8-4. USB 2.0 EHCI Capability Registers

Offset from usb_ehcibase (Note1)	Register Name
0x0000	HCCAPBASE
0x0004	HCSPARAMS
0x0008	HCCPARAMS

Note1. See Table 8-1 on page 252 for base address.

HCCAPBASE **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
	HCIVER																								CAPLENGTH							
Def.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	HCIVER	Host Controller Interface Version Number. This is a two-byte register containing a BCD encoding of the version number of interface to which the host controller interface conforms.	R	0x100
8	—	Reserved.	—	—
7:0	CAPLENGTH	Capability Registers Length. Offset to add to usb_ehcibase to find operational registers.	R	0x10

Table 8-5. USB 2.0 EHCI Host Controller Operational Registers

Offset from usb_ehcibase (Note1)	Register Name
0x0010	USBCMD
0x0014	USBSTS
0x0018	USBINTR
0x001C	FRINDEX
0x0020	CTRLDSSEGMEN
0x0024	PERIODICLISTBASE
0x0028	ASYNCLISTADDR
0x0050	CONFIGFLAG
0x0054	PORTSC_1
0x0058	PORTSC_2

Note1. See Table 8-1 on page 252 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

8.2.3.1 CONFIGFLAG Register

CONFIGFLAG **Offset = 0x0050**

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
																																CF
Def	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:1	—	Reserved	—	—
0	CF	Configure Flag. Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 Port routing control logic default-routes all ports to this host controller.	R/W	0

8.2.3.2 Port Status and Control Registers

PORTSC_1 **Offset = 0x0054**

PORTSC_2 **Offset = 0x0058**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										WKOC	WKD	WKC	PTC							PP	LS			PR	SUS	FPR	OCC	OC	PEC	PE	CSC	CC
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:23	—	Reserved.	—	—
22	WKOC	Wake on Overcurrent Enable. Setting this bit enables the port to be sensitive to overcurrent conditions as wakeup events. This bit is zero if port power is not available (PP=0).	R/W	0
21	WKD	Wake on Disconnect Enable. Setting this bit enables the port to be sensitive to device disconnects as wakeup events. This bit is zero if port power is not available (PP=0).	R/W	0
20	WKC	Wake on Connect Enable. Setting this bit enables the port to be sensitive to device connects as wakeup events. This bit is zero if port power is not available (PP=0).	R/W	0
19:16	PTC	Port Test Control. Clear this field for normal port operation. A non-zero value configures the port to operate in a test mode as follows: 0000: Normal operation 0001: Test J_STATE 0010: Test K_STATE 0011: Test SE0_NAK 0100: Test Packet 0101: Test FORCE_ENABLE All other values are reserved.	R/W	0
15:13	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
12	PP	Port Power. This bit represents the current setting of the switch. 0 Port is off. 1 Port is on. When power is not available on a port (PP=0), the port is nonfunctional and does not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port, the PP bit in each affected port may be transitioned by the host controller from a 1 to 0 to remove power from the port.	R/W	0
11:10	LS	Line Status. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) data signals. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This status field indicates the following USB states: 00 SE0 Not Low-speed device, perform EHCI reset. 10 J-state Not Low-speed device, perform EHCI reset. 01 K-state Low-speed device, release ownership of port. 11 Undefined not low-speed device, perform EHCI reset. This field is valid only when the port enable bit is zero and the current connect status bit is set. This field is undefined if port power is not available (PP=0).	R/W	0
9	—	Reserved.	—	—
8	PR	Port Reset. When software sets this bit (from a zero), the bus reset sequence as defined in the *USB Specification Revision 2.0* is started. Software clears this bit to terminate the bus reset sequence. 0 Port is not in reset. 1 Port is in reset.	R/W	0
7	SUS	Suspend. 0 Do not suspend the port. 1 Suspend the port. When in Suspend state, downstream propagation of data is blocked on this port, except for port reset. When this bit is set, the suspension occurs at the end of the current transaction if a transaction is already in progress. In the suspend state, the port is sensitive to resume detection. The Port Enable (PE) bit and Suspend bit of this register define the port states as follows: PE SUS State 0 x Disabled 1 0 Enabled 1 1 Suspended	R/W	0
6	FPR	Force Port Resume. The functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Port Enable bits are one) and software transitions this bit to a one, then the effects on the bus are undefined. 0 No resume (K-state) detected/driven on port. 1 Resume detected/driven on port.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
5	OCC	Overcurrent Change. Hardware sets OCC when the Overcurrent Active (OC) bit changes. Software clears OCC by writing a one to it.	R/WC	0
4	OC	Overcurrent Active. This bit automatically transitions from one to zero when the over current condition is removed. 0 This port does not have an overcurrent condition. 1 This port currently has an overcurrent condition.	R/W	0
3	PEC	Port Enable/Disable Change. For the root hub, hardware sets this bit only when a port is disabled due to the appropriate conditions existing at the EOF2 point. Software clears PEC by writing a one to it. 0 No change. 1 Port enabled/disabled status has changed. This bit is zero if port power is not available (PP=0).	R/WC	0
2	PE	Port Enable. Ports can be enabled only by the host controller as a part of the reset and enable sequence. Software cannot enable a port by writing a one to this field. The host controller sets this bit only when the reset sequence determines that the attached device is a high-speed device. 0 Disable. 1 Enable. This bit is zero if port power is not available (PP=0).	R/W	0
1	CSC	Connect Status Change. Indicates a change has occurred in the Current Connect Status (CC) bit. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. Software clears this bit by writing a one to it. 0 No change. 1 Change in Current Connect Status. This bit is zero if port power is not available (PP=0).	R/WC	0
0	CC	Current Connect Status. This value reflects the current state of the port and may not correspond directly to the event that caused the Connect Status Change bit (CSC) to be set. 0 No device is present. 1 Device is present on port. This bit is zero if port power is not available (PP=0).	R/W	0

8.2.4 USB 2.0 Device Controller Registers

The USB 2.0 Device controller is configured and controlled through the registers in Table 8-6.

Table 8-6. USB 2.0 Device Controller Registers

Offset from usb_devbase (Note1)	Register Name
0x0000	dev_epinctrl0
0x0004	dev_epinsts0
0x0008	dev_epinbs0
0x000C	dev_epinmaxp0

www.DataSheet4U.com

www.DataSheet4U.com

Table 8-6. USB 2.0 Device Controller Registers

Offset from usb_devbase (Note1)	Register Name
0x0014	dev_epinddp0
0x001C	dev_epinwrc0
0x0020	dev_epinctrl1
0x0024	dev_epinsts1
0x0028	dev_epinbs1
0x002C	dev_epinmaxp1
0x0034	dev_epinddp1
0x003C	dev_epinwrc1
0x0040	dev_epinctrl2
0x0044	dev_epinsts2
0x0048	dev_epinbs2
0x004C	dev_epinmaxp2
0x0054	dev_epinddp2
0x005C	dev_epinwrc2
0x0060	dev_epinctrl3
0x0064	dev_epinsts3
0x0068	dev_epinbs3
0x006C	dev_epinmaxp3
0x0074	dev_epinddp3
0x007C	dev_epinwrc3
0x0080	dev_epinctrl4
0x0084	dev_epinsts4
0x0088	dev_epinbs4
0x008C	dev_epinmaxp4
0x0094	dev_epinddp4
0x009C	dev_epinwrc4
0x0200	dev_epoutctrl0
0x0204	dev_epoutsts0
0x0208	dev_epoutfrn0
0x020C	dev_epoutmaxp0
0x0210	dev_epoutsubp0
0x0214	dev_epoutddp0
0x021C	dev_epoutrdc0
0x0220	dev_epoutctrl1
0x0224	dev_epoutsts1
0x0228	dev_epoutfrn1
0x022C	dev_epoutmaxp1
0x0230	dev_epoutsubp1
0x0234	dev_epoutddp1

www.DataSheet4U.com

www.DataSheet4U.com

Table 8-6. USB 2.0 Device Controller Registers

Offset from usb_devbase (Note1)	Register Name
0x023C	dev_epoutrdc1
0x0240	dev_epoutctrl2
0x0244	dev_epoutsts2
0x0248	dev_epoutfrn2
0x024C	dev_epoutmaxp2
0x0250	dev_epoutsubp2
0x0254	dev_epoutddp2
0x025C	dev_epoutrdc2
0x0260	dev_epoutctrl3
0x0264	dev_epoutsts3
0x0268	dev_epoutfrn3
0x026C	dev_epoutmaxp3
0x0270	dev_epoutsubp3
0x0274	dev_epoutddp3
0x027C	dev_epoutrdc3
0x0280	dev_epoutctrl4
0x0284	dev_epoutsts4
0x0288	dev_epoutfrn4
0x028C	dev_epoutmaxp4
0x0290	dev_epoutsubp4
0x0294	dev_epoutddp4
0x029C	dev_epoutrdc4
0x0400	dev_cfg
0x0404	dev_ctrl
0x0408	dev_sts
0x040C	dev_intr
0x0410	dev_intrmsk
0x0414	dev_epintr
0x0418	dev_epintrmsk
0x0504	dev_ep0reg
0x0508	dev_ep1reg
0x050C	dev_ep2reg
0x0510	dev_ep3reg
0x0514	dev_ep4reg
0x0518	dev_ep5reg
0x051C	dev_ep6reg
0x0520	dev_ep7reg
0x0524	dev_ep8reg

Note1. See Table 8-1 on page 252 for base address.

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.1 Input Endpoint Control Registers

dev_epinctrl0 **Offset = 0x0000**

dev_epinctrl1 **Offset = 0x0020**

dev_epinctrl2 **Offset = 0x0040**

dev_epinctrl3 **Offset = 0x0060**

dev_epinctrl4 **Offset = 0x0080**

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
																								CNAK	SNAK	NAK	ET		P		F	S
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:9	—	Reserved.	—	—
8	CNAK	Clear NAK. Used by the application to clear the NAK bit (bit 6, below). After the Subsystem sets bit 6 (NAK), the Application must clear it with a write of 1 to the CNAK bit after it has decoded the Setup packet and determined it is not an invalid command. The Application must clear the NAK bit whenever the Subsystem sets it. The Subsystem sets it due to the Application setting the Stall bit.	W	0
7	SNAK	Set NAK. Used by the application to set the NAK bit (bit 6, below). If the NAK bit is already set, a Setup packet is still sent to the Application.	W	0
6	NAK	NAK Bit. If set to 1, the endpoint responds to the USB Host with a NAK handshake. If set to 0, the endpoint responds normally. On successful reception of a Setup packet (decoded by the Application), the Subsystem sets both the IN and OUT NAK bits.	R	0
5:4	ET	Endpoint Type. 00 CONTROL endpoint 01 ISO endpoint 10 BULK endpoint 11 INTERRUPT endpoint	R/W	0
3	P	Poll Demand. After updating the endpoint's system memory with a new descriptor chain, the Poll Demand bit is set by the application to indicate to the DMA engine that the new descriptor chain is available. The Subsystem clears the Poll Demand bit after it has reached the last descriptor in the chain.	R/W	0
2	—	Reserved.	—	—
1	F	Flush the TxFIFO. When set the TxFIFO will be flushed, reset by hardware after flush.	R/W	0
0	S	Stall request from the USB Host. On successful reception of a Setup packet (decoded by the application), the Subsystem clears both IN and OUT Stall bits, and sets both the IN and OUT NAK bits. The application must check for RxFIFO emptiness before setting the In and OUT Stall bits. For non-Setup packets, the subsystem clears either In or OUT Stall bits only if a STALL handshake is returned to the USB Host and then sets the corresponding NAK bit.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.2 Input Endpoint Status Registers

dev_epinsts0 **Offset = 0x0004**

dev_epinsts1 **Offset = 0x0024**

dev_epinsts2 **Offset = 0x0044**

dev_epinsts3 **Offset = 0x0064**

dev_epinsts4 **Offset = 0x0084**

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
																						TDC	HE		BNA	IN						
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:11	—	Reserved.	—	—
10	TDC	Transmit DMA Completion. Indicates the transmit DMA has completed transferring a descriptor chain’s data to the Tx FIFO. After servicing the interrupt, the application must clear this bit.	R/WC	0
9	HE	Host Error Response. When doing a data transfer, descriptor fetch, or descriptor update for this endpoint, a host error response was received. After servicing the interrupt, the application must clear this bit.	R/WC	0
8	—	Reserved.	—	—
7	BNA	Buffer Not Available. The subsystem sets this bit when the descriptor’s status is not “Host Read”. After servicing the interrupt, the application must clear this bit.	R/WC	0
6	IN	An IN token has been received by this endpoint. After servicing the interrupt, the application must clear this bit.	R/WC	0
5:0	—	Reserved.	—	—

8.2.4.3 Input Endpoint Buffer Size Registers

dev_epinbs0 **Offset = 0x0008**

dev_epinbs1 **Offset = 0x0028**

dev_epinbs2 **Offset = 0x0048**

dev_epinbs3 **Offset = 0x0068**

dev_epinbs4 **Offset = 0x0088**

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
																							BS									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:10	—	Reserved.	—	—
9:0	BS	Buffer Size. The application can program this field to make each endpoint’s buffer adaptive, providing flexibility in buffer size when the interface or configuration is changed. This value is in 32-bit words, and indicates the number of 32-bit word entries in the TxFIFO.	R/W	0

www.DataSheet4U.com

www.DataSh

8.2.4.4 Input Endpoint Maximum Packet Size Registers

dev_epinmaxp0 **Offset = 0x000C**

dev_epinmaxp1 **Offset = 0x002C**

dev_epinmaxp2 **Offset = 0x004C**

dev_epinmaxp3 **Offset = 0x006C**

dev_epinmaxp4 **Offset = 0x008C**

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
																	MAXP															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:10	—	Reserved.	—	—
15:0	MAXP	Maximum Packet Size. The maximum allowed packet size in bytes.	R/W	0

8.2.4.5 Input Endpoint Data Descriptor Pointer Registers

dev_epinddp0 **Offset = 0x0014**

dev_epinddp1 **Offset = 0x0034**

dev_epinddp2 **Offset = 0x0054**

dev_epinddp3 **Offset = 0x0074**

dev_epinddp4 **Offset = 0x0094**

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
	DESPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	DESPTR	Data Descriptor Pointer.	R/W	0

8.2.4.6 Input Endpoint Write Confirmation Registers

dev_epinwrc0 **Offset = 0x001C**

dev_epinwrc1 **Offset = 0x003C**

dev_epinwrc2 **Offset = 0x005C**

dev_epinwrc3 **Offset = 0x007C**

dev_epinwrc4 **Offset = 0x009C**

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
	WRC																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	WRC	Write Confirmation. Confirms that all data to be transferred to host has been written to the transmit FIFO (**dev_txfifomem**). Applies to Slave-Only mode only (**dev_ctrl**[MODE]=0). Software writes any value to this register to make confirmation.	W	0

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.7 Output Endpoint Control Registers

dev_epoutctrl0 **Offset = 0x0200**

dev_epoutctrl1 **Offset = 0x0220**

dev_epoutctrl2 **Offset = 0x0240**

dev_epoutctrl3 **Offset = 0x0260**

dev_epoutctrl4 **Offset = 0x0280**

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
																								CNAK	SNAK	NAK	ET			SN		S
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:9	—	Reserved.	—	—
8	CNAK	Clear NAK. Used by the Application to clear the NAK bit. After the Subsystem sets the NAK bit, the application must clear it with a write of 1 to the CNAK bit after it has decoded the Setup packet and determined it is not an invalid command. The application must clear the NAK bit whenever the subsystem sets it. The Subsystem sets it due to the Application setting the Stall bit.	W	0
7	SNAK	Set NAK. Used by the application to set the NAK bit (bit 6, below). If the NAK bit is already set, a Setup packet is still sent to the Application.	W	0
6	NAK	NAK Bit. If set to 1, the endpoint responds to the USB Host with a NAK handshake. If set to 0, the endpoint responds normally. On successful reception of a Setup packet (decoded by the application), the Subsystem sets both the IN and OUT NAK bits.	R	0
5:4	ET	Endpoint Type. 00 CONTROL endpoint 01 ISO endpoint 10 BULK endpoint 11 INTERRUPT endpoint	R/W	0
3	—	Reserved.	—	—
2	SN	Snoop mode. In this mode, the Subsystem does not check the correctness of OUT packets before transferring them to application memory.	R/W	0
1	—	Reserved.	—	—
0	S	Stall request from the USB Host. On successful reception of a Setup packet (decoded by the application), the subsystem clears both IN and OUT Stall bits, and sets both the IN and OUT NAK bits. The application must check for RxFIFO emptiness before setting the In and OUT Stall bits. For non-Setup packets, the subsystem clears either In or OUT Stall bits only if a STALL handshake is returned to the USB Host and then sets the corresponding NAK bit.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.8 Output Endpoint Status Registers

dev_epoutsts0 **Offset = 0x0204**

dev_epoutsts1 **Offset = 0x0224**

dev_epoutsts2 **Offset = 0x0244**

dev_epoutsts3 **Offset = 0x0264**

dev_epoutsts4 **Offset = 0x0284**

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
										RXPKTSIZE													HE		BNA		OUT					
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:23	—	Reserved.	—	—
22:11	RXPKT-SIZE	Receive Packet Size. Indicates the number of bytes in the current received packet to be sent to the endpoint. This field is used for Slave mode only.	R/W	0
10	—	Reserved.	—	—
9	HE	Host Error Response. When doing a data transfer, descriptor fetch, or descriptor update for this endpoint, a host error response was received. After servicing the interrupt, the application must clear this bit.	R/WC	0
8	—	Reserved.	—	—
7	BNA	Buffer Not Available. The subsystem sets this bit when the descriptor's status is not "Host Read". After servicing the interrupt, the application must clear this bit.	R/WC	0
6	—	Reserved.	—	—
5:4	OUT	Out Packet Received. An OUT packet has been received by this endpoint. The encoding of these two bits indicates the type of data received: 00 None 01 Received data 10 Received Setup data (8 bytes) 11 Reserved This field is only used in Slave mode.	R/WC	0
3:0	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.9 Output Endpoint Frame Number Registers

dev_epoutfrn0 **Offset = 0x0208**

dev_epoutfrn1 **Offset = 0x0228**

dev_epoutfrn2 **Offset = 0x0248**

dev_epoutfrn3 **Offset = 0x0268**

dev_epoutfrn4 **Offset = 0x0288**

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
																			FRN													
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:14	—	Reserved.	—	—
13:0	FRN	Frame Number. Frame number in which the packet is received. the number is given in microframe resolution for High Speed operation. For Full Speed and Low Speed operation, the frame number has the frame resolution.	R	0

8.2.4.10 Output Endpoint Maximum Packet Size Registers

dev_epoutmaxp0 **Offset = 0x020C**

dev_epoutmaxp1 **Offset = 0x022C**

dev_epoutmaxp2 **Offset = 0x024C**

dev_epoutmaxp3 **Offset = 0x026C**

dev_epoutmaxp4 **Offset = 0x028C**

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
																	MAXP															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15:0	MAXP	Maximum Packet Size. The maximum allowed packet size in bytes.	R/W	0

www.DataSheet4U.com

w w w . D a t a S h

8.2.4.11 Output Endpoint Setup Buffer Pointer Registers

dev_epoutsubp0 **Offset = 0x0210**

dev_epoutsubp1 **Offset = 0x0230**

dev_epoutsubp2 **Offset = 0x0250**

dev_epoutsubp3 **Offset = 0x0270**

dev_epoutsubp4 **Offset = 0x0290**

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
	SUBPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	SUBPTR	Setup Buffer Pointer. This register is used for Setup commands on CONTROL endpoints. For other endpoint types this register is reserved.	R/W	0

8.2.4.12 Output Endpoint Data Descriptor Pointer Registers

dev_epoutddp0 **Offset = 0x0214**

dev_epoutddp1 **Offset = 0x0234**

dev_epoutddp2 **Offset = 0x0254**

dev_epoutddp3 **Offset = 0x0274**

dev_epoutddp4 **Offset = 0x0294**

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
	DESPTR																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	DESPTR	Data Descriptor Pointer.	R/W	0

8.2.4.13 Output Endpoint Read Confirmation Registers

dev_epoutrdc0 **Offset = 0x021C**

dev_epoutrdc1 **Offset = 0x023C**

dev_epoutrdc2 **Offset = 0x025C**

dev_epoutrdc3 **Offset = 0x027C**

dev_epoutrdc4 **Offset = 0x029C**

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
	RDC																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	RDC	Read Confirmation. Read Confirmation for zero-length OUT data in Slave-Only mode only (**dev_ctrl**[MODE]=0). Software reads this register to make confirmation; the return value is a don’t-care.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

8.2.4.14 Device Configuration Register

dev_cfg **Offset = 0x0400**

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
	SRST	HSF												SD	PROG									STAT				SS	SP	RWKP	SPD	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	SRST	Software reset. 0 Normal operation 1 Reset the entire controller. The reset is immediate and unconditional.	W	0
30	HSF	HNP SET_FEATURE enable value. Reflects the value of **otg_ctl**[HSF].	R	0
29:19	—	Reserved.	—	—
18	SD	Indicates that the device supports SET_DESCRIPTOR requests. 0 The USB subsystem returns a STALL handshake to the USB host. 1 The Setup packet for the SET_DESCRIPTOR request passes to the application.	R/W	0
17	PROG	Dynamic Programming. The application is able to program the UDC registers dynamically whenever it has received an interrupt for either a SET_CONFIGURATION or a SET_INTERFACE. The subsystem returns a NAK handshake during the status-in stage of both the SET_CONFIGURATION and SET_INTERFACE requests until the application has written 1 to the **dev_ctrl**[DONE] if this bit is enabled.	R/W	0
16:9	—	Reserved.	—	—
8:7	STAT	These bits controls how the subsystem responds to data packets during the status-out stage of a control transfer. See Table 8-7 on page 270.	R/W	0
6:5	—	Reserved.	—	—
4	SS	Sync Frame Capability. 0 Device does not support Sync Frame 1 Device supports Sync Frame	R/W	0
3	SP	Self Powered Capability. 0 Device is not Self-Powered 1 Device is Self-Powered	R/W	0
2	RWKP	Remote Wakeup Capability. 0 Device does not support Remote Wakeup 1 Device supports Remote Wakeup	R/W	0
1:0	SPD	Device Speed. 00 High-Speed (HS) 01 Full-Speed (FS) 10 Low-Speed (LS) 11 Full-Speed (FS)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Table 8-7. UDC Response during the Status-Out Stage of a Control Transfer

Packet Length	SETUP Decode	dev_cfg [STAT]	Handshake to Host	Forward Packet to Application?
0	Internal	00	ACK	No
		01	ACK	No
		10	Reserved	—
		11	ACK	No
	External	00	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
		01	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
		10	Reserved	—
		11	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
greater than 0	Internal	00	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
		01	STALL	Yes
		10	Reserved	—
		11	STALL	No
	External	00	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
		01	According to **dev_epoutctrl*n***[CNAK, SNAK, S]	Yes
		10	Reserved	—
		11	STALL	No

8.2.4.15 Device Control Register

dev_ctrl **Offset = 0x0404**

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
	THLEN								BRLEN										DONE	NAK		SD	MODE		THE	BF		DU	TDE	RDE		RES
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	THLEN	Threshold Length. Indicates the number (THLEN+1) of DWORDs in the RxFIFO before the DMA can start data transfer	R/W	0
23:16	BRLEN	Burst Length. Indicates the length in DWORDs of a single burst on the system bus. The subsystem sends BRLEN+1 DWORDs.	R/W	0
15:14	—	Reserved.	—	—
13	DONE	Programming Done. The application can set this bit to tell the subsystem core when it has completed programming all the required UDC registers such that the subsystem core can send an ACK handshake to the current SET_CONFIGURATION or SET_INTERFACE command.	W	0
12	NAK	NAK all OUT EPs. If this bit is set by the application, the subsystem core returns a NAK handshake to all OUT endpoints. By writing a 1 to this bit, the application does not need to write a 1 to the SNAK bit of each endpoint control register.	R/W	0
11	—	Reserved.	—	—
10	SD	Soft Disconnect. The application software uses this bit to signal the device controller to soft-disconnect. When set, this bit causes the device to enter the disconnect state.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
9	MODE	Mode. 0 Slave-only mode (programmed I/O) 1 DMA mode	R/W	0
8	—	Reserved.	—	—
7	THE	Threshold Enable. When this bit is set a number of quadlets equivalent to the threshold value is transferred from the RxFIFO to the memory.	R/W	0
6	BF	Buffer Fill Mode. 0 Descriptor chain mode: The DMA engine transfers data by traversing the descriptor chain. 1 Buffer fill mode: The DMA engine transfers data into contiguous locations pointed to by the buffer address.	R/W	0
5	—	Reserved.	—	—
4	DU	Descriptor Update. 0 The DMA engine updates only the last descriptor in the descriptor chain with the status of the entire chain. 1 The DMA engine updates each descriptor immediately after processing.	R/W	0
3	TDE	Transmit DMA Enable.	R/W	0
2	RDE	Receive DMA Enable.	R/W	0
1	—	Reserved.	—	—
0	RES	Resume signalling on the USB.	R/W	0

8.2.4.16 Device Status Register

dev_sts **Offset = 0x0408**

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
	TS														▨	PE	RXE	ES		SUSP	ALT				INTF				CFG			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:28	TS	Number of Frames. Frame number of the received SOF.	R	0
17	—	Reserved.	—	—
16	PE	PHY Error. Either the phy_rxvalid or phy_rxactive input signal is detected to be continuously asserted for 2 ms, indicating a PHY error. The subsystem goes to the Suspend state as a result	R	0
15	RXE	FIFO Empty. 0 Receive FIFO not empty 1 Receive FIFO empty	R	1
14:13	ES	Enumerated Speed. These bits hold the speed at which the subsystem comes up after the Speed Enumeration. Possible options are 00 HS 01 FS 10 LS 11 Reserved	R	0
12	SUSP	Suspend Status. This bit is set as long as a Suspend condition is detected on the USB bus.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
11:8	ALT	This field represents the alternate setting to which the below interface (reflected in INTF) is switched.	R	0
7:4	INTF	Interface. This field reflects the interface set by the SetInterface command.	R	0
3:0	CFG	Configuration. This field reflects the configuration set by the SetConfiguration command	R	0

8.2.4.17 Device Interrupt Register

dev_intr **Offset = 0x040C**

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
																										ENUM	SOF	US	UR	ES	SI	SC
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:7	—	Reserved.	R	0
6	ENUM	Speed Enumeration Complete.	R/WC	0
5	SOF	SOF Token Detected. An SOF token has been detected on the USB bus. Once the device has received the first SOF token, this bit is set every time a micro-frame interval has elapsed, regardless of whether a new token has been received.	R/WC	0
4	US	Suspend. A suspend has been detected on the USB bus.	R/WC	0
3	UR	Reset Detected. A reset has been detected on the USB bus	R/WC	0
2	ES	Idle. An idle state has been detected on the USB bus for 3 ms.	R/WC	0
1	SI	SetInterface Interface Command. The device has received a SetInterface command	R/WC	0
0	SCR	Set Configuration Command. The device has received a SetConfiguration command	R/WC	0

8.2.4.18 Device Interrupt Mask Register

dev_intrmsk **Offset = 0x0410**

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
																										ENUMM	SOFM	USM	URM	ESM	SIM	SCM
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:7	—	Reserved.	—	—
6	ENUMM	Speed Enumeration Complete Mask. Mask the ENUMinterrupt	R/W	1
5	SOFM	Start Of Frame Mask. Mask the SOF interrupt	R/W	1
4	USM	Suspend Mask. Mask the US interrupt	R/W	1
3	URM	Reset Mask. Mask the UR interrupt	R/W	1
2	ESM	Idle Mask. Mask the ES interrupt	R/W	1
1	SIM	Set Interface Command Mask. Mask the SI interrupt	R/W	1
0	SCM	Set Configuration Command Mask. Mask the SC interrupt	R/W	1

www.DataSheet4U.com

www.DataSheet4U.cc

8.2.4.19 Endpoint Interrupt Register

dev_epintr **Offset = 0x0414**

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
												OUTEP																INEP				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:21	—	Reserved.	—	—
20:16	OUTEP	OUT Endpoint. A bit is set when there is an event on the corresponding OUT endpoint (bit 16 for EP0, bit 17 for EP1, etc.).	R/WC	0
15:5	—	Reserved.	—	—
4:0	INEP	IN Endpoint. A bit is set when there is an event on the corresponding IN endpoint (bit 0 for EP0, bit 1 for EP1, etc.)	R/WC	0

8.2.4.20 Endpoint Interrupt Mask Register

dev_epintrmsk **Offset = 0x0418**

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
												OUTEPM																INEPM				
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:21	—	Reserved.	—	—
20:16	OUTEP	OUT Endpoint Mask. Mask interrupts for events on the corresponding OUT endpoint (bit 16 for EP0, bit 17 for EP1, etc.).	R/W	1
15:5	—	Reserved.	—	—
4:0	INEP	IN Endpoint Mask. Mask interrupts for events on the corresponding IN endpoint (bit 0 for EP0, bit 1 for EP1, etc.).	R/W	1

8.2.4.21 Endpoint Configuration Registers

dev_ep0reg **Offset = 0x0504**

dev_ep1reg **Offset = 0x0508**

dev_ep2reg **Offset = 0x050C**

dev_ep3reg **Offset = 0x0510**

dev_ep4reg **Offset = 0x0514**

dev_ep5reg **Offset = 0x0518**

dev_ep6reg **Offset = 0x051C**

dev_ep7reg **Offset = 0x0520**

dev_ep8reg **Offset = 0x0524**

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
	MULT		MAXP											ALT				IF				CFG				ET		ED	EN			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

www.DataSheet4U.com

www.DataSheet4U.cc

Bits	Name	Description	R/W	Default
31:30	MULT	ISO Number. Number of ISO transfers per microframe. Reserved for non-isochronous endpoints.	R/W	0
29:19	MAXP	Maximum Packet Size.	R/W	0
18:15	ALT	Alternate Setting. Alternate setting to which this EP belongs	R/W	0
14:11	IF	Interface Number. Interface number to which this EP belongs	R/W	0
10:7	CFG	Configuration Number. Configuration number to which this EP belongs	R/W	0
6:5	ET	Endpoint Type. 00 Control 01 Isochronous 10 Bulk 11 Interrupt	R/W	0
4	ED	Endpoint Direction. 0 OUT 1 IN	R/W	0
3:0	EN	Endpoint Number.	R/W	0

8.2.4.22 Receive FIFO

dev_rxfifomem **Offset = 0x0800 - 0x0BFC**

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
	RxFIFO																															
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

8.2.4.23 Transmit FIFO

dev_txfifomem **Offset = 0x0C00 - 0x11FC**

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
	TxFIFO																															
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

www.DataSheet4U.com

www.DataSheet4U.com

8.2.5 USB 2.0 On-The-Go Controller Registers

Table 8-8. USB 2.0 On-The-Go Registers

Offset from usb_otgbase (Note1)	Register Name
0x0000	otg_cap
0x0004	otg_mux
0x0008	otg_sts
0x000C	otg_ctl
0x0010	otg_tmr
0x0014	otg_intr
0x0018	otg_inten

Note1. See Table 8-1 on page 252 for base address.

8.2.5.1 OTG Capabilities Register

otg_cap **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
																	APU															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15	APU	Automatic Pull-up Enable. This bit controls how the pull-up resistor on USBDP is activated when the port is assigned to the Device Controller. 0 Software needs to activate the pull-up. 1 The pull-up is activated as soon as **otg_sts**[VBV] is 1. Writing a 1 is to this bit also sets **otg_ctl**[OTGPADEN].	R/W	0
14:0	—	Reserved.	—	—

8.2.5.2 OTG Multiplex Register

otg_mux **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:26	—	Reserved.	—	—
8	VBV	VBus Valid. This bit is set when the voltage on USBVBUS is above 4.0 V. This bit is only valid when **otg_ctl**[OPE] is set.	R	0
7:3	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
2	PUE	Pull-up Enable. When automatic pull-up enable is configured (**otg_cap**[APU] is set) then this bit is read only and has the same value as **otg_sts**[VBV]. When configured to software control, this bit enables the pull-up resistor on USBDP. This bit is ignored when PMUX is not 0b11. 0 Pull-up disabled 1 Pull-up activated	R/W or RO	0
1:0	PMUX	Port Mux Control. 00The port is suspended and not assigned to either controller 01 The port is suspended and not assigned to either controller 10 The port is assigned to the Host Controller 11 The port is assigned to the Device Controller	R/W	0

8.2.5.3 OTG Status Register

otg_sts **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
																	OC	HNPSTS			TMH	PSUS	PCON	FSOE	PSPD		LST		SE	SV	VBV	ID
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15	OC	Over Current Status. This bit is set when an over current condition for Port 2 (dependent on **otg_cap**[OCP,OCR]) exists.	R	0
14:12	HNPSTS	Host-Negotiation-Protocol Feature Status. Bit Feature 12 b_hnp_enable 13 a_hnp_support 14 a_alt_hnp_support	R	0
11	TMH	Timer Halted. This bit is set by hardware when the timer is either currently not counting or is in the disabled state (**otg_ctl**[TMR-COND] is cleared).	R	1
10	PSUS	Port Suspended. When the port is assigned to Host, this bit is set when the respective controller has suspended the port. When the port is assigned to Device, the bit is set when the controller has detected a suspend condition on the bus. The value of this bit can be overwritten at any time by software by writing the desired value to **otg_ctl**[WPSS].	R	0
9	PCON	Port Connected. When the port is assigned to Host, this bit is set when the appropriate host controller has established a connection to a remote device. When the port is assigned to Device, the bit is set when the device controller has been contacted by the remote host. The value of this bit can be overwritten at any time by software by writing the desired value to **otg_ctl**[WPCS].	R	0
8	FSOE	FS/LS Output Enable. This bit is set when the OHCI owns the port and is driving the bus (this includes also driving SE0).	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
7:6	PSPD	Port Speed. 00 Port is in high speed 01 Port is in full speed 10 Port is in low speed 11 Reserved	R	0b11
5:4	LST	Line State. Independent of the settings in **otg_mux**[PMUX], this field reflects the status of the port. LST[1:0] = {USBDM, USBDP}	R	—
3	SE	Session End. The voltage on USBVBUS pin is below 0.6 V. This bit is only valid when **otg_ctl**[OPE] is set.	R	—
2	SV	Session Valid. The voltage on USBVBUS pin is above 1.2 V. This bit is used for both, A-Device Session Valid and B-Device Session Valid. This bit is only valid when **otg_ctl**[OPE] is set.	R	—
1	VBV	VBus Valid. This bit is set when the voltage on USBVBUS is above 4.4 V (this is a copy of **otg_mux**[VBV]). This bit is only valid when **otg_ctl**[OPE] is set.	R	—
0	ID	ID Pin. This bit reflects the value at USBOTGID. An automatic debounce of approximately 2.5 ms is applied. This bit is only valid when **otg_ctl**[IDS] is set, it is forced to zero otherwise.	R	0

8.2.5.4 OTG Control Register

otg_ctl **Offset = 0x000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	▨	▨	▨	TRP	TMRCOND				▨	▨	▨	▨	WPSS		WPCS		▨	▨	▨	▨	▨	HSF	PDE	PUE	OPE	IDS	DVB	CVB	▨	▨	PMC	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:26	—	Reserved.	—	—
28	TRP	Timer Reload Policy. 0 Reload on re-enabling 1 Just resume counting on re-enabling	R/W	0
27:24	TMRCOND	Timer Count Condition. 0x0 Timer disabled 0x1 Count unconditionally (software start) 0x2 Line state is FS-SE0 0x3 Line state is FS-J (pull-up on USBDP) 0x4 Line state is FS-K (pull-up on USBDM) 0x5 Line state is not FS-SE0 0x6 No RX activity 0x7 ID is zero 0x8-0xF Reserved	R/W	0
23:20	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
19:18	WPSS	Write Port Suspend Status. This field is used to override the value of **otg_sts**[PSUS]. However, **otg_sts**[PS] is only changed once when this field is written to, during all other times **otg_sts**[PSUS] is under hardware control. 00 Do not overwrite 01 Do not overwrite 10 Deassert otg_sts[PSUS] 11 Assert otg_sts[PSUS]	R/W	0
17:16	WPCS	Write Port Connect Status. This field is used to override the value of **otg_sts**[PCON]. However, **otg_sts**[PCON] is only changed once when this field is written to, during all other times **otg_sts**[PCON] is under hardware control. 00 Do not overwrite 01 Do not overwrite 10 Deassert **otg_sts**[PCON] 11 Assert **otg_sts**[PCON]s	R/W	0
15:11	—	Reserved.	—	—
10	HSF	Enable HNP SET_FEATURE Commands. When set, the device controller is allowed to consume the HNP SET_FEATURE commands and respond with ACK; otherwise, it rejects them and responds with STALL.	R/W	0
9	PDE	Pull-down Enable. This bit controls the pull-down resistor on USBDM. The behavior depends on **otg_mux**[PMUX]: 00 The bit is read only and fixed to 1. 01 The bit is read only and fixed to 1. 10 The bit is read only and fixed to 1. 11 The bit is R/W. When set, the pull-down resistor on USBDM is activated. When **otg_mux**[PMUX] transitions to 0b11, this bit is also reset to 0. When **otg_ctl**[PMC] transitions to 0b11 and PDE is written to concurrently then this value takes effect.	R/W or RO	0b1
8	PUE	Pull-up Enable. This bit is aliased from **otg_mux**[PUE]. When automatic pull-up enable is configured (**otg_cap**[APU] is set) then this bit is read only and has the same value as **otg_sts**[VBV]. When configured to software control, this bit enables the pull-up resistor on USBDP. This bit is ignored when PMUX is not 0b11. 0 Pull-up disabled 1 Pull-up activated	R/W or RO	0
7	OPE	OTG Pad Enable. When set this bits enables the extended OTG hardware. Writing a 1 to **otg_cap**[APU] also sets this bit to 1. Writing 0 to this bit also resets **otg_cap**[APU] to 0.	R/W	0
6	ISE	ID Pin Sense Enable. When this bit is set, a pull-up resistor is applied to the USBOTGID pin allowing to sense the state of the pin at the USB receptacle. When reset, no current is drawn on USBOTGID, but **otg_sts**[ID] is not valid. This bit is automatically reset when a 0 is written to OPE.	R/W	0
5	DVB	Discharge VBus. When this bit is set, a pull-down resistor of typically 840 Ohms to ground is applied to the USBVBUS pin. This bit is automatically reset when a 0 is written to OPE.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
4	CVB	Charge VBus. When this bit is set a pull-up resistor of typically 650 Ohms to 3.3V is applied to the USBVBUS pin. This bit is automatically reset when a 0 is written to OPE.	R/W	0
3:2	—	Reserved.	—	—
1:0	PMC	Port Mux Control. This field is aliased from **otg_mux**[PMUX] 00 The port is not assigned to any Controller 01 The port is not assigned to any Controller 10 The port is assigned to the Host Controller 11 The port is assigned to the Device Controller	R/W	0

8.2.5.5 OTG Timer Register

otg_ctmr **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:26	—	Reserved.	—	—
25:16	CURTMR	Timer Reload Policy. 0 Reload on re-enabling 1 Just resume counting on re-enabling	R	0
15:10	—	Reserved.	—	—
9:0	PRELD	Preload Value. This field contains the value used as preload for count down in multiples of 10 us.	R/W	0

8.2.5.6 OTG Current Interrupt Register

otg_intr **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			OC	HNP	TE	PS	PC	RD	HSD	FLO	PS	LS	SE	SV	VV	ID
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:14	—	Reserved.	—	—
13	OC	Over Current Detected. This bit is set when an over current condition for Port 2 (dependent on **otg_cap**[OCP,OCR]) has occured.	R/WC	0
12	HNP	HNP Features Have Changed. This bit is set when the **otg_sts**[HNPSTS] field has changed, either because the UDC received the respective SET_FEATURE command or because it received a port reset, this bit is not set when **otg_sts**[HNPSTS] is reset because software reset **otg_ctl**[HSF].	R/WC	0
11	TE	Timer Expired.	R/WC	0
10	PS	Port Suspend Has Changed. This bit is not set when **otg_sts**[PC] has changed through a software write to **otg_ctl**[WPSS] or **otg_mux**[PMUX].	R/WC	0
9	PC	Port Connect has Changed. This bit is not set when **otg_sts**[PC] has changed through a software write to **otg_ctl**[WPCS] or **otg_mux**[PMUX].	R/WC	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
8	RD	Receive Activity Detected. This bit is also cleared by hardware when **otg_mux**[PMUX] is changed.	R/WC	0
7	HSD	HS Disconnect Detected.	R/WC	0
6	FLO	FS/LS Output Enable has changed.	R/WC	0
5	PS	Port Speed Changed.	R/WC	0
4	LS	Linestate Changed.	R/WC	0
3	SE	SESSEND Changed.	R/WC	0
2	SV	SESSVLD Changed.	R/WC	0
1	VV	VBus Valid (VBV) Changed.	R/WC	0
0	ID	Value of ID Pin Changed.	R/WC	0

8.2.5.7 OTG Interrupt Enable Register

otg_inten **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GIE																		OC	HNP	TE	PS	PC	RD	HSD	FLO	PS	LS	SE	SV	VV	ID
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	GIE	Global Interrupt Enable.	R/W	0
30:14	—	Reserved.	—	—
13	OC	Over Current Interrupt Enable.	R/W	0
12	HNP	HNP Feature Change Interrupt Enable.	R/W	0
11	TE	Timer Expired Interrupt Enable.	R/W	0
10	PS	Port Suspend Change Interrupt Enable.	R/W	0
9	PC	Port connect Status Change Interrupt Enable.	R/W	0
8	RD	Receive Activity Detection Interrupt Enable.	R/W	0
7	HSD	HS Disconnect Detection Interrupt Enable.	R/W	0
6	FLO	FS/LS Output Enable Change Interrupt Enable.	R/W	0
5	PS	Port Speed Change Interrupt Enable.	R/W	0
4	LS	Linestate Change Interrupt Enable.	R/W	0
3	SE	SESSEND Change Interrupt Enable.	R/W	0
2	SV	SESSVLD Change Interrupt Enable.	R/W	0
1	VV	VBus Valid (VBV) Change Interrupt Enable.	R/W	0
0	ID	ID Pin Change Interrupt Enable.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

8.3 Hardware Considerations

Table 8-9 shows the USB signals.

Table 8-9. USB Signals

Signal	I/O	Description
USBV$_{DDX}$	P	3.3 volt nominal power for the USB I/O peripheral. Same as V$_{DDX}$ but isolate from V$_{DDX}$ using ferrite bead and capacitors.
USBV$_{DDI}$	P	1.2 volt nominal power for the USB I/O peripheral. Same as V$_{DDI}$ but isolate from V$_{DDI}$ using ferrite bead and capacitors.
USBV$_{SS}$	G	Ground for the USB I/O peripheral. Tie to V$_{SS}$.
USBDP	I/O	Positive signal of differential USB device port. Connect to an external common-mode choke.
USBDM	I/O	Negative signal of differential USB device port. Connect to an external common-mode choke.
USBXI	I	Crystal oscillator input pin. Connect to a 48MHz crystal or an external oscillator.
USBXO	O	Crystal oscillator output pin. Connect to a 48MHz crystal or tie to ground if an external oscillator is applied to USBXI.
USBHP	I/O	Positive signal of differential USB host port. Connect to an external common-mode choke.
USBHM	I/O	Negative signal of differential USB host port. Connect to an external common-mode choke.
USBREXT	G	External resistor that sets the analog bias currents. Connect with a 3.4 KΩ resistor +/- 1% to USBV$_{SS}$.
USBVBUS	I/O	Monitors the USB bus voltage on the USB connector. USBVBUS is used to sense the value of the bus voltage to generate the VBUS status signals SessEnd, Avalid, Bvalid and VbusValid. USBVBUS performs the USB OTG session request protocol. This pin does not provide power to the USB bus.
USBOTGID	I	Connect to the ID pin on the USB Mini receptacle that is used to differentiate a Mini-A plug from a Mini-B plug.
USBATEST	UN	Manufacturing test pin. Leave open.

www.DataSheet4U.com

www.DataSheet4U.com

8.4 Software Considerations

8.4.1 UDC DMA Memory Structures

In DMA mode (**dev_ctrl**[MODE] = 1), the UDC supports IN and OUT descriptor chains for each endpoint. For control endpoints, an additional SETUP buffer is implemented. These memory structures are shown in Figure 8-1.

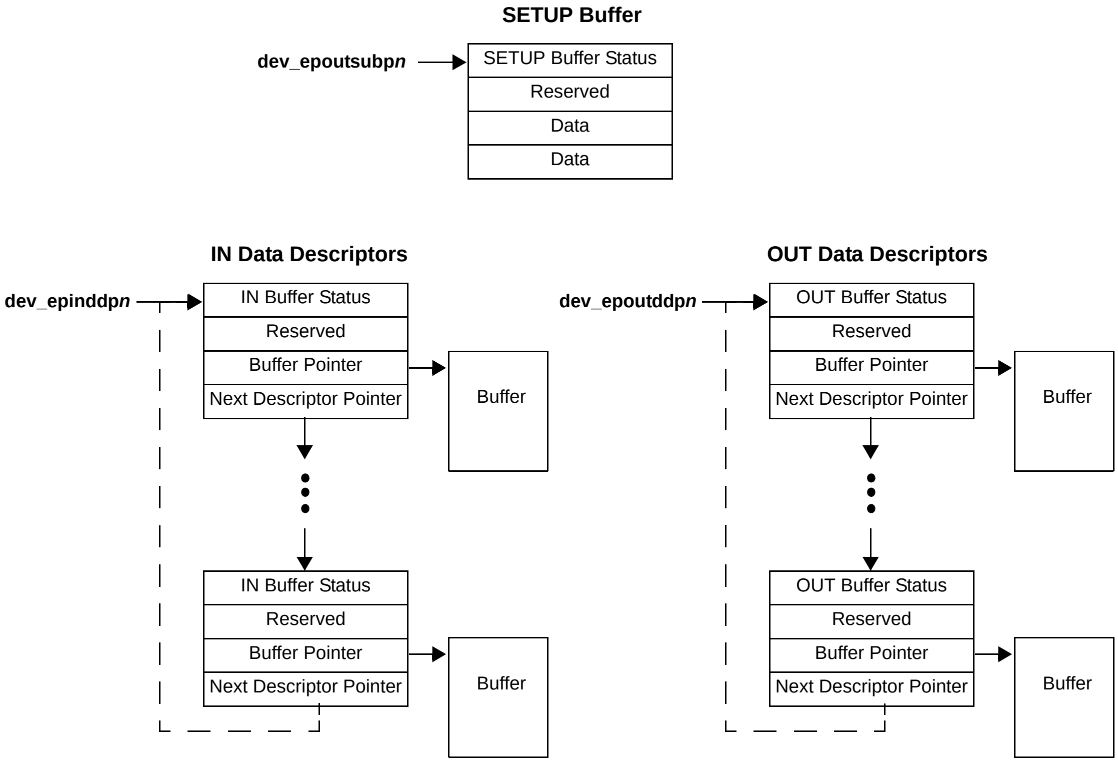


Figure 8-1. UDC Descriptor and Buffer Memory Structures

The fields within the memory structures are described below.

www.DataSheet4U.com

www.DataSheet4U.com

8.4.1.1 SETUP Buffer

The setup buffer is sixteen bytes and applies to control endpoints only. The UDC receives setup data from the USB host.

SETUP Buffer Status

setup_stat **Offset = 0x00**

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
	BS		RXSTAT		CFGSTAT																											

Bits	Name	Description
31:30	BS	Buffer Status. 00 Application ready. The application has completed processing the buffer. 01 DMA busy. The DMA engine is processing the buffer. 10 DMA done. The DMA engine has completed the data transfer. 11 Application busy. The application is processing the buffer.
29:28	RXSTAT	Receive Status. Reflects whether the setup data has been received correctly. 00 Success 01 Descriptor error. Indicates the buffer was not ready during the fetch (**setup_stat**[BS] not equal to 0b00). 1x Reserved
27:16	CFGSTAT	Configuration Status. Contains the current configuration associated with the Setup packet. [27:24] Configuration number [23:20] Interface number [19:16] Alternate setting number
15:0	—	Reserved.

SETUP Data

The eight data bytes contain the setup data as defined in the USB specification.

www.DataSheet4U.com

www.DataSheet4U.com

8.4.1.2 IN Data Descriptors

Endpoints supporting IN direction transactions (where data is sent to the USB host) must implement an IN descriptor chain with associated data buffers. Each buffer descriptor is sixteen bytes. The data buffers contain packet data for non-isochronous endpoints and frame data for isochronous endpoints.

The software application prepares the data buffers, updates the buffer status in the descriptor (**indes_stat**[BS] = 00), and sets the endpoint's poll-demand bit (**dev_epinctrl*n***[P]). The DMA engine then fetches this descriptor and transfers its data buffer. The DMA engine continues processing the descriptor chain until it reaches the end (**indes_stat**[L] = 1).

IN Buffer Status

indes_stat **Offset = 0x00**

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
	BS		TXSTAT		L	FRAME											COUNT															

Bits	Name	Description
31:30	BS	Buffer Status. 00 Application ready. The application has completed processing the descriptor. 01 DMA busy. The DMA engine is processing the descriptor. 10 DMA done. The DMA engine has completed the data transfer. 11 Application busy. The application is processing the descriptor.
29:28	TXSTAT	Transmit Status. Reflects whether the IN data has been sent correctly. 00 Success 01 Descriptor error. Indicates the application was not ready during the descriptor fetch (**indes_stat**[BS] not equal to 0b00). 10 Reserved 11 Buffer error
27	L	Last Descriptor. The DMA engine stops traversing the chain after processing the last descriptor. 0 Not the last descriptor in the chain 1 Last descriptor in the chain
26:16	FRAME	Frame Number. Applies to isochronous endpoints only and is otherwise reserved. Contains the frame number in which the current packet must be sent.
15:0	COUNT	Byte Count. Specifies the number of data bytes to be sent to the USB host.

Buffer Pointer and Next Descriptor Pointer

These pointers are 32-bit addresses to their respective data structures.

www.DataSheet4U.com

www.DataSheet4U.com

8.4.1.3 OUT Data Descriptors

Endpoints supporting OUT direction transactions (where data is received from the USB host) must implement an OUT descriptor chain with associated data buffers. Each buffer descriptor is sixteen bytes. The data buffers contain packet data for non-isochronous endpoints and frame data for isochronous endpoints.

The DMA engine fetches the first descriptor and examines the data buffer status. If the buffer is available (**outdes_stat**[BS] = 00), the DMA engine begins transferring data into the buffer. If the buffer is not available, the DMA engine skips to the next descriptor until it reaches the end of the descriptor chain (**outdes_stat**[L] = 1).

OUT Buffer Status

outdes_stat **Offset = 0x00**

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
BS	RXSTAT	L	FRAME	COUNT

Bits	Name	Description
31:30	BS	Buffer Status. 00 Application ready. The application has completed processing the descriptor. 01 DMA busy. The DMA engine is processing the descriptor. 10 DMA done. The DMA engine has completed the data transfer. 11 Application busy. The application is processing the descriptor.
29:28	RXSTAT	Transmit Status. Reflects whether the OUT data has been received correctly. 00 Success 01 Descriptor error. Indicates the application was not ready during the descriptor fetch (**outdes_stat**[BS] not equal to 0b00). 10 Reserved 11 Buffer error
27	L	Last Descriptor. The DMA engine stops traversing the chain after processing the last descriptor. 0 Not the last descriptor in the chain 1 Last descriptor in the chain
26:16	FRAME	Frame Number. Applies to isochronous endpoints only and is otherwise reserved. Contains the frame number in which the current packet is received.
15:0	COUNT	Byte Count. Specifies the number of data bytes received from the USB host.

Buffer Pointer and Next Descriptor Pointer

These pointers are 32-bit addresses to their respective data structures.

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

9 Peripheral Devices

This section provides descriptions of the peripheral devices of the Au1210™ and Au1250™ processors. The integrated peripherals include two programmable serial controllers (PSC), USB controllers, two Secure Digital controllers, the CIM, LCD controller, AES cryptography engine (Au1250 only), the software counter, and two UARTs. (Although the PSC and USB controllers are considered integrated peripherals, they are described in separate sections (see Section 7.0 "Programmable Serial Controllers (PSCs)" on page 193 and Section 8.0 "USB 2.0 Subsystem" on page 251).

Each peripheral contains an enable register. All other registers within each peripheral's register block must not be accessed until the enable register is written with the correct sequence to bring the peripheral out of reset. Accessing peripheral registers before a peripheral is enabled may result in system deadlock.

www.DataSheet4U.com

www.DataSheet4U.com

9.1 Camera Interface Module (CIM)

The camera interface module (CIM) connects to a CMOS or CCD type image sensor. The CIM sources the digital image stream through a common parallel digital protocol. The CIM can be configured to convert Bayer pattern RGB or CCIR 656 protocol data into planar format so the on-chip MAE can demosaic the data stream for the LCD controller. The CIM can also be configured in pass-through mode so that any digital stream can be written in raw format directly to memory.

The CIM has the following features:

- Start of Frame detect
- Supports 8-bit to10-bit digital pixel data bus
- Supports CCIR 656 protocol for NTSC and PAL data streams
- Three parallel data paths for image stream parsing
- Can parse Bayer data stream (all four formats) into planar R, G, and B output to memory
- Can parse interlaced YC_bC_r data stream into planar Y, C_b, and C_r output to memory
- Can pass raw data stream direct to memory (no parsing)
- Configurable CIM_FS and CIM_LS signals: active high/low and pulse/level
- Configurable CIM_CLK: active edge rising/falling and pulse/free-running
- Supports CIM_CLK frequency up to 33MHz

Figure 9-1 shows a block diagram of the CIM.

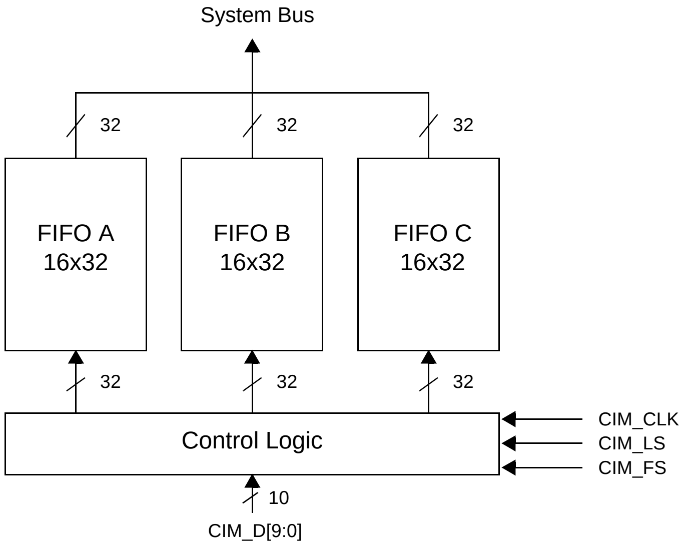


Figure 9-1. CIM Block Diagram

www.DataSheet4U.com

www.DataSheet4U.c

9.1.1 CIM Registers

The CIM is controlled by a register block whose physical base address is shown in Table 9-1.

Table 9-1. CIM Register Base Address

Name	Physical Base Address	KSEG1 Base Address
cim_base	0x1400 4000	0xB400 4000

Table 9-2 shows the CIM registers.

Table 9-2. CIM Registers

Offset (Note1)	Register Name	Description
0x0000	cim_enable	Enable Register
0x0004	cim_config	Configuration Register
0x0010	cim_capture	Capture Control Register
0x0014	cim_stat	Status Register
0x0018	cim_inten	Interrupt Registers - Enable
0x001C	cim_intstat	Interrupt Registers - Status
0x0020	cim_fifoa	Data Registers - Output FIFO A
0x0040	cim_fifob	Data Registers - Output FIFO B
0x0060	cim_fifoc	Data Registers - Output FIFO C

Note1. See Table 9-2 for base address.

9.1.1.1 Enable Register

This register is used to enable, disable and reset the CIM block.

cim_enable **Offset = 0x0000**

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

Bits	Name	Description	R/W	Default
31:1	—	Reserved.	—	0
0	EN	Enable. 0 Reset and disable the CIM. 1 Enable the CIM (before writing any CIM registers)	R/W	0

9.1.1.2 Configuration Register

This register configures the input control signal settings, the input pixel clock settings and the input data stream settings.

cim_config **Offset = 0x0004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														SI	FSEL		SF	BYT	LEN				BAY		DPS				FS	LS	CLK	PUL
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:19	—	Reserved.	—	0
18	SI	Stream Interlaced Data. Applies to a 656 interface only. Set this bit to capture interlaced input and output the interlaced fields from individual ports. Field 1 data is read from FIFO A and field 2 data is read from FIFO B.	R/W	0
17:16	FSEL	Field Select. Applies to a 656 interface only. Selects the starting field for data capture. 00 Start capturing with field 1. 01 Start capturing with field 2. 10 Start capturing with either field. 11 Reserved	R/W	0
15	SF	Single Frame Capture. Describes how the external image sensor captures images. 0 Image sensor produces a continuous stream of valid image data. 1 Image sensor produces only a single frame of valid image data	R/W	0
14	BYT	Byte Capture Mode. Byte capture mode is required when using the MAE back end to demosaic, scale or color space convert the output image data. 0 Capture full length of interface bus as defined in **cim_config**[LEN]. 1 Capture only upper 8-bits of interface bus.	R/W	0
13:10	LEN	Data Length. Selects the data length for the interface bus regardless of **cim_config**[BYT]. 000 8-bit interface 001 9-bit interface 010 10-bit interface. Must be used for 656 transfers. All other values are reserved.	R/W	0
9:8	BAY	Bayer Mode Select. 00 First line (RGRGRG...) second line (GBGBGB...) 01 First line (GRGRGR...) second line (BGBGBG...) 10 First line (BGBGBG...) second line (GRGRGR...) 11 First line (GBGBGB...) second line (RGRGRG...)	R/W	0
7:6	DPS	Data Pattern Select. 00 Raw streaming: no data parsing 01 Parse Bayer pattern: create planar R G B 10 Parse 656 YC_bC_r pattern: create planar Y C_b C_r 11 Reserved	R/W	0
5:4	—	Reserved.	—	0
3	FS	Frame Sync Input Type Select. 0 CIM_FS is active high 1 CIM_FS is active low	R/W	0
2	LS	Line Sync Input Type Select. 0 CIM_LS is active high 1 CIM_LS is active low	R/W	0
1	CLK	CIM Clock Input Type Select. 0 CIM_CLK is active falling edge. 1 CIM_CLK is active rising edge.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
0	PUL	Pulse Mode. Pulse mode expects the following protocol: CIM_CLK toggles only during valid data transfers (data-ready mode). CIM_FS pulses inactive after each frame. CIM_LS pulses inactive after each line. See Section 9.1.4 "CIM Interface Timing" on page 296 for more information. 0 Level mode: Clock is free-running. Sync signals are level-sensitive. 1 Pulse mode: Clock toggles during valid data. Sync signals are pulsed.	R/W	0

9.1.1.3 Capture Control Register

This register is used to enable/disable the capturing of image data and to clear the data path.

cim_capture **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:3	—	Reserved.	—	0
2	CLR	Clear Data Path. 0 Normal operation 1 Clear the CIM data path. Do not clear the data path while the CIM is capturing an image: **cim_stat**[SC] or **cim_stat**[VC] = 1.	R/W	0
1	SCE	Still Capture Enable. Capture a single still image frame. 0 Disable still capture. 1 Enable still capture: The CIM starts capturing image data at the start of the next frame. The CIM captures only one frame of image data. The capture-in-progress status bit **cim_stat**[SC] is set when the start of the first frame is detected; **cim_stat**[SC] is then cleared after the first frame has been captured. This bit is self clearing and always reads as a 0.	W	0
0	VCE	Video Capture Enable. Capture the video image data stream. 0 Disable video capture: If video capture is in progress, the CIM stops capturing image data at the end of the current frame, and all of the current frame data is read out through the DDMA controller. 1 Enable video capture: The CIM starts capturing image data at the start of the next frame. The capture-in-progress status bit **cim_stat**[VC] is set when the start of the first frame is detected; **cim_stat**[VC] is then cleared after the last frame has been captured.	R/W	0

www.DataSheet4U.com

w w w . D a t a S h e

9.1.1.4 Status Register

This register contains the CIM status indicators, such as capture-in-progress. A status condition is true when the status bit is set. Changes in status require one clock cycle to update after an event.

cim_stat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						CR	CE	CF	BR	BE	BF	AR	AE	AF	SC	VC
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:11	—	Reserved.	—	0
10	CR	FIFO C Request. Indicates FIFO C is requesting a read. The FIFO requests a read when it has at least 8 words of data.	R	0
9	CE	FIFO C Empty.	R	0
8	CF	FIFO C Full.	R	0
7	BR	FIFO B Request. Indicates FIFO B is requesting a read. The FIFO requests a read when it has at least 8 words of data.	R	0
6	BE	FIFO B Empty.	R	0
5	BF	FIFO B Full.	R	0
4	AR	FIFO A Request. Indicates FIFO A is requesting a read. The FIFO requests a read when it has at least 8 words of data.	R	0
3	AE	FIFO A Empty.	R	0
2	AF	FIFO A Full.	R	0
1	SC	Still Capture In Progress. Indicates the CIM is capturing still image data (single frame). SC is set at the start of the first frame after enabling still frame capture (**cim_capture**[SCE] = 1). SC clears itself after the last pixel of the first frame is captured.	R	0
0	VC	Video Capture In Progress. Indicates the CIM is capturing video image data (multiple frames). VC is set at the start of the first frame after enabling video capture (**cim_capture**[VCE] = 1). When software disables video capture (**cim_capture**[VCE] = 0), VC clears itself after the last pixel of the current frame is captured.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

9.1.1.5 Interrupt Registers

These registers (**cim_inten**) and (**cim_intstat**) are used to process CIM interrupts. If a bit is set in **cim_inten** and the corresponding condition becomes true, an interrupt request is issued and the corresponding bit in **cim_intstat** is set. Once the interrupt has been serviced, write a 1 to the corresponding bit location in **cim_intstat** to clear the interrupt; writing a 0 has no effect. These registers may be read and written while the CIM is active.

cim_inten **Offset = 0x0010**

cim_intstat **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								ERR	OFC	UFC	OFB	UFB	OFA	UFA	FD	CD
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:9	—	Reserved.	—	0
8	ERR	Protection Error. Indicates a protection error has been detected. Applies only when the 656 protocol is selected.	R/W	0
7	OFC	Overflow FIFO C. Indicates FIFO C has overflowed and data has been lost.	R/W	0
6	UFC	Underflow FIFO C. Indicates FIFO C has underflowed and 0x0 has been read.	R/W	0
5	OFB	Overflow FIFO B. Indicates FIFO B has overflowed and data has been lost.	R/W	0
4	UFB	Underflow FIFO B. Indicates FIFO B has underflowed and 0x0 has been read.	R/W	0
3	OFA	Overflow FIFO A. Indicates FIFO A has overflowed and data has been lost.	R/W	0
2	UFA	Underflow FIFO A. Indicates FIFO A has underflowed and 0x0 has been read.	R/W	0
1	FD	Frame Done. Indicates the CIM has finished capturing an image frame. Applies to video capture mode only (**cim_capture**[VCE] = 1). FD is set after each completed frame capture as long as video capture remains enabled. See also Section 9.1.1.3 "Capture Control Register" on page 291.	R/W	0
0	CD	Capture Done. Indicates the CIM has completed capturing the image data. For still capture (**cim_capture**[SCE] = 1), CD is set when one frame has been captured. For video capture (**cim_capture**[VCE] = 1), CD is set when the last frame has been captured after video capture has been disabled (**cim_capture**[VCE] = 0). See also Section 9.1.1.3 "Capture Control Register" on page 291.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.1.1.6 Data Registers

These registers (**cim_fifoa**, **cim_fifob**, and **cim_fifoc**) allow software access to the CIM FIFOs (FIFO A, FIFO B, and FIFO C). A read from a data register causes one data entry (32 bits) to be popped from the FIFO, if data is present. Each FIFO is 16 entries deep.

To prevent underflow errors during programmed I/O, software must monitor the FIFO status in **cim_stat** to ensure that data is present before reading. When performing sequential reads, software must also compensate for the additional clock cycle required for **cim_stat** to update after each read.

For DMA transfers, the FIFO status is automatically monitored by the DDMA controller.

The FIFOs are cleared during a CIM reset (**cim_enable**[EN] = 0) or data path clear (**cim_capture**[CLR] = 1).

cim_fifoa **Offset = 0x0020**
cim_fifob **Offset = 0x0040**
cim_fifoc **Offset = 0x0060**

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

Bits	Name	Description	R/W	Default
31:0	DATA	32-bit pixel data in big-endian ordering. See Section 9.1.2 "DMA Programming" on page 294 for pixel format information.	R	0

9.1.2 DMA Programming

The CIM relies on the DDMA controller to write data to memory. If the CIM is configured to parse the incoming digital image stream (Bayer or YC_bC_r modes), the CIM uses three ports on the DDMA controller (one for each FIFO). If the CIM is configured to separate 656 interlaced input (**cim_config**[SI] = 1), the CIM uses two ports on the DDMA controller (for FIFOs A and B). If the CIM is configured in pass-through mode (raw data), it uses only one port on the DDMA controller (for FIFO A). Table 9-3 lists the three FIFOs of the CIM, showing which data elements are transferred and which DMA requests are used.

Table 9-3. Data Ports

CIM FIFO	DMA Request	Data Elements by Mode			
		Bayer	YC_bC_r	Interlaced	Pass-Through
A	18	Red pixel data	Y pixel data	All field 1 pixel data	All pixel data
B	19	Green pixel data	C_b pixel data	All field 2 pixel data	—
C	20	Blue pixel data	C_r pixel data	—	—

As the CIM captures pixels from the interface, it packs them into 32-bit words before pushing the image data onto the appropriate FIFO.

- If the CIM is capturing 8-bit pixels, each pixel takes one byte. Four pixels are packed into each image data word.
- If the CIM is capturing pixels larger than 8 bits, each pixel takes two bytes with zero padding. Two pixels are packed into each image data word.

If an image data word is not completely filled when reaching the end of a pixel row, the CIM pads the remainder of the data word with zeros after the last valid pixel. Since the CIM uses big-endian byte ordering, the zeros are placed in the lower invalid portion of the data word. If needed, the bytes can be re-ordered in the DDMA controller.

Each port needs its own set of DMA descriptors. To capture multiple frames, software can create a circular linked-list of descriptors for each port to allow multiple buffers of image data in memory. For a description of how to program the DDMA controller, see Section 4.0 "Descriptor-Based DMA (DDMA) Controller" on page 115.

www.DataSheet4U.com

www.DataSheet4U.com

9.1.3 CCIR 656 Interface Programming

When the CCIR 656 mode is selected the CIM detects frame, line and field location by analyzing the header and footer blocks of the data bus; the CIM_FS and CIM_LS input signals are ignored. Table 9-4 defines the four header words that precede and follow each video section as defined by the 656 protocol.

Table 9-4. CCIR 656 Header Data Bit Definition

Data Bit	First Word (0xFF)	Second Word (0x00)	Third Word (0x00)	Fourth Word
CIM_D[9] (MSB)	1	0	0	1
CIM_D[8]	1	0	0	F
CIM_D[7]	1	0	0	V
CIM_D[6]	1	0	0	H
CIM_D[5]	1	0	0	P3
CIM_D[4]	1	0	0	P2
CIM_D[3]	1	0	0	P1
CIM_D[2]	1	0	0	P0
CIM_D[1](Note1)	x	x	x	x
CIM_D[0][a]	x	x	x	x

Note1. For compatibility with an 8-bit interface, CIM_D[1] and CIM_D[0] are not defined.

Since the CCIR 656 interface is defined as ten bits wide, always configure the CIM data interface for ten bits (**cim_config**[LEN] = 0b010) for 656 data. For 8-bit transfers, enable byte capturing (**cim_config**[BYT] = 1); in this case, the CIM_D[9:2] signals contain the 8-bit 656 data.

To ensure correct de-interlacing in the MAE back end, the CIM waits for the start of active video (SAV) for field 1 before capturing image data. The CIM also detects protection errors as defined by the 656 protection error codes and reports errors via **cim_intstat**[ERR] (if it has been enabled in **cim_inten**). The CIM ignores all blanking and non-pixel data. Table 9-5 describes the header information, how it is decoded by the CIM, and the protection codes.

Table 9-5. 656 Header Decode

Decode	F	V	H	P3	P2	P1	P0
Field 1 start of active video (SAV)	0	0	0	0	0	0	0
Field 1 end of active video (EAV)	0	0	1	1	1	0	1
Field 1 SAV (digital blanking)	0	1	0	1	0	1	1
Field 1 EAV (digital blanking)	0	1	1	0	1	1	0
Field 2 SAV	1	0	0	0	1	1	1
Field 2 EAV	1	0	1	1	0	1	0
Field 2 SAV (digital blanking)	1	1	0	1	1	0	0
Field 2 EAV (digital blanking)	1	1	1	0	0	0	1

www.DataSheet4U.com

www.DataSheet4U.com

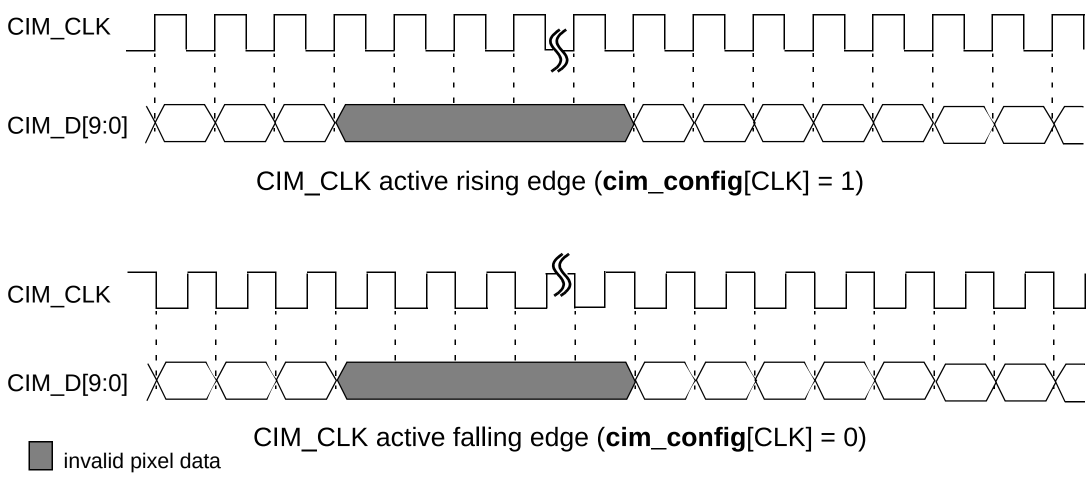
9.1.4 CIM Interface Timing

9.1.4.1 Pixel Clock (CIM_CLK)

The pixel clock input signal CIM_CLK acts as a synchronization reference for the pixel data on the digital video input bus CIM_D[9:0]. The pixel data changes on the active edge of CIM_CLK and is captured on the inactive edge.

The CIM can be configured to receive CIM_CLK in two modes: *free-running* and *data-ready*. The clock mode is selected in **cim_config**[PUL].

- In *free-running* mode (**cim_config**[PUL] = 0), CIM_CLK is always running at a fixed frequency. As shown in Figure 9-2, pixel data arriving on CIM_D[9:0] is synchronized to the specified active edge of the clock.

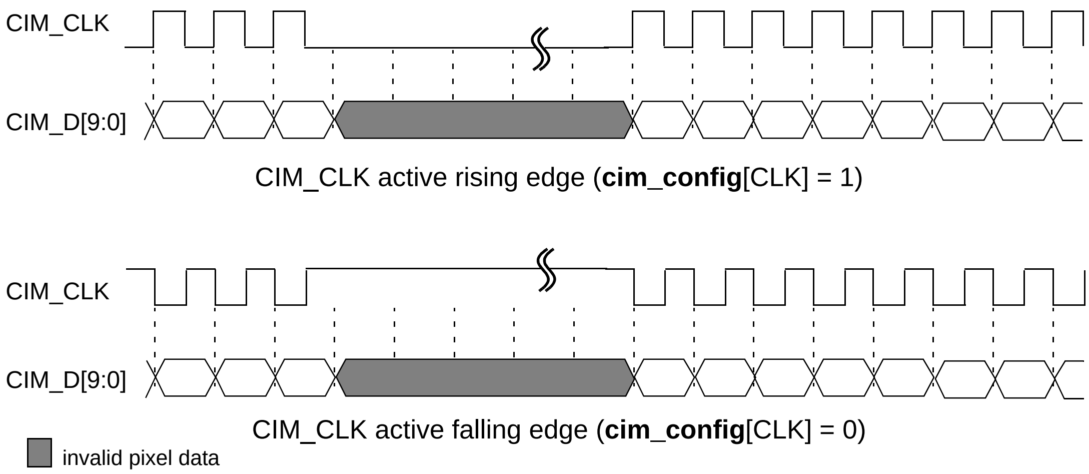


CIM_CLK active rising edge (**cim_config**[CLK] = 1)

CIM_CLK active falling edge (**cim_config**[CLK] = 0)

Figure 9-2. CIM_CLK Free-running Mode (cim_config[PUL] = 0)

- In *data-ready* mode (**cim_config**[PUL] = 1), CIM_CLK pulses with a specified level every time valid pixel data arrives on CIM_D[9:0], as shown in Figure 9-3.



CIM_CLK active rising edge (**cim_config**[CLK] = 1)

CIM_CLK active falling edge (**cim_config**[CLK] = 0)

Figure 9-3. CIM_CLK Data-ready Mode (cim_config[PUL] = 1)

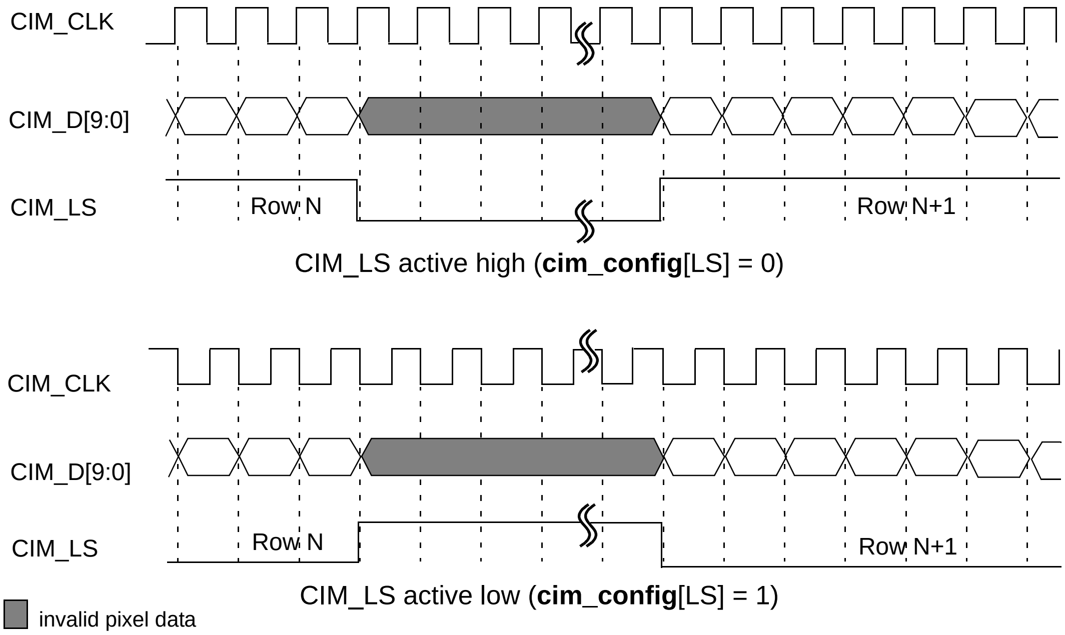
www.DataSheet4U.com

www.DataSheet4U.com

9.1.4.2 Line Synchronization (CIM_LS)

The line synchronization input signal CIM_LS indicates pixel data within a row. The CIM can be configured to expect CIM_LS to operate in two modes: *level* or *pulse*.

- In *level* mode (**cim_config**[PUL] = 0), CIM_LS asserts (active high or low) at the start of each row of valid pixel data and negates after the last pixel of that row is read, as shown in Figure 9-4. State transitions in CIM_LS must be synchronized to the active edge of CIM_CLK. Level mode corresponds to free-running mode for CIM_CLK.

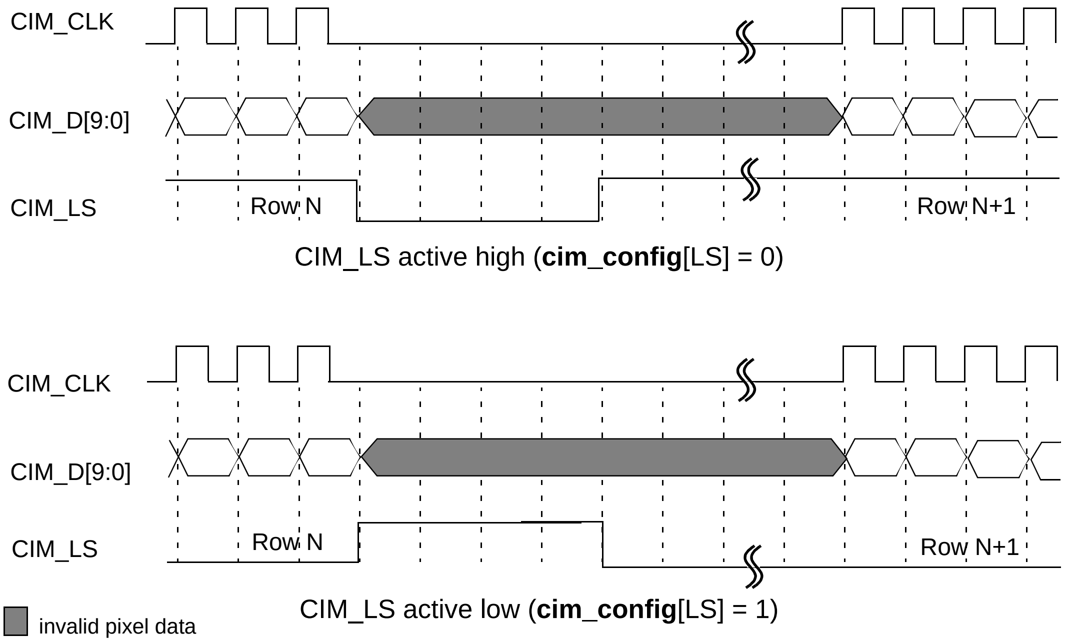


CIM_LS active high (**cim_config**[LS] = 0)

CIM_LS active low (**cim_config**[LS] = 1)

Figure 9-4. CIM_LS Level Mode (cim_config[PUL] = 0)

- In *pulse* mode (**cim_config**[PUL] = 1), CIM_LS pulses (negation followed by assertion) to indicate the end of each row as shown in Figure 9-5. The pulse width must be a minimum of four CIM_CLK cycles and must be synchronized to the active edge of CIM_CLK. Pulse mode corresponds to data-ready mode for CIM_CLK.



CIM_LS active high (**cim_config**[LS] = 0)

CIM_LS active low (**cim_config**[LS] = 1)

Figure 9-5. CIM_LS Pulse Mode (cim_config[PUL] = 1)

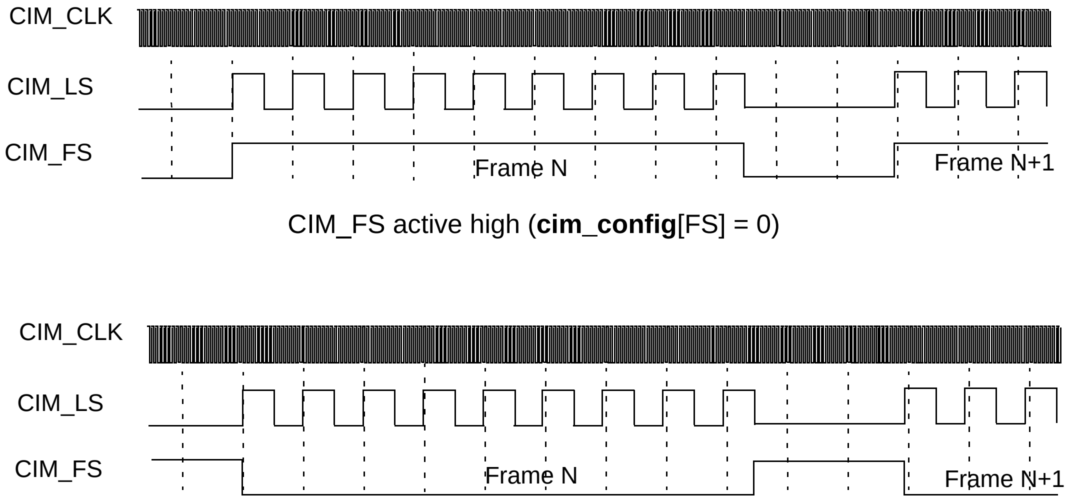
www.DataSheet4U.com

www.DataSheet4U.com

9.1.4.3 Frame Synchronization (CIM_FS)

The frame synchronization input signal CIM_FS indicates pixel data within a frame. The CIM can be configured to expect CIM_FS to operate in two modes: *level* or *pulse*.

- In *level* mode (**cim_config**[PUL] = 0), CIM_FS asserts (active high or low) at the start of each frame of valid pixel data and negates after the last pixel of that frame is read, as shown in Figure 9-6. State transitions in CIM_FS must be synchronized to the active edge of CIM_CLK. Level mode corresponds to free-running mode for CIM_CLK.

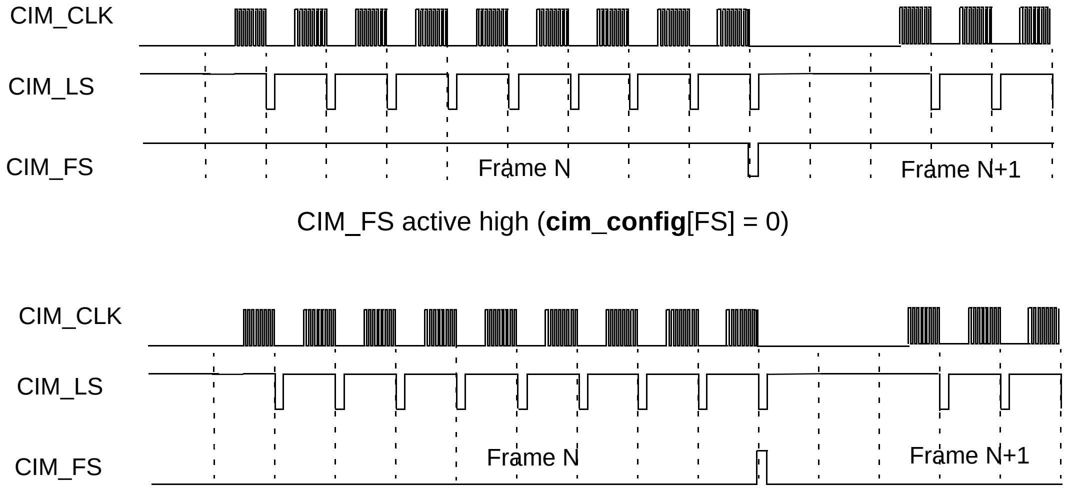


CIM_FS active high (**cim_config**[FS] = 0)

CIM_FS active low (**cim_config**[FS] = 1)

Figure 9-6. CIM_FS Level Mode (cim_config[PUL] = 0)

- In *pulse* mode (**cim_config**[PUL] = 1), CIM_FS pulses (negation followed by assertion) to indicate the end of each frame as shown in Figure 9-7. The pulse width must be a minimum of four CIM_CLK cycles and must be synchronized to the active edge of CIM_CLK. Pulse mode corresponds to data-ready mode for CIM_CLK.



CIM_FS active high (**cim_config**[FS] = 0)

CIM_FS active low (**cim_config**[FS] = 1)

Figure 9-7. CIM_FS Pulse Mode (cim_config[PUL] = 1)

www.DataSheet4U.com

www.DataSheet4U.com

9.1.5 CIM Hardware Considerations

The CIM interface consists of the signals listed in Table 9-6.

Table 9-6. CIM Signals

Signal	Input/ Output	Definition
CIM_D[9:0]	I	8- to 10-bit parallel data received from imaging device.
CIM_CLK	I	Pixel data synchronization clock received from imaging device. CIM_CLK can have a maximum frequency of 33MHz.
CIM_LS	I	Line data synchronization signal received from imaging device.
CIM_FS	I	Frame data synchronization signal received from imaging device.

The CIM signals share or multiplex their function with GPIO signals. Select the CIM signal functions as follows:

- CIM_D[0:1] *share* their function with GPIO[0:1]: Tri-state GPIO[0:1] to make them inputs by programming the **sys_triout** register. They do not require configuration in **sys_pinfunc**.
- The CIM_D[2:9], CIM_CLK, CIM_LS, and CIM_FS signals are *multiplexed* with GPIO signals: Set **sys_pinfunc**[CIM]. See Section 10.3 "Primary General Purpose I/O and Pin Functionality" on page 381 for more information.

www.DataSheet4U.com

www.DataSheet4U.com

9.2 LCD Controller

The Au1210™ and Au1250™ processors' integrated LCD controller has the capabilities necessary for driving the latest industry standard 1-8 bit gray-scale or 4-24 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. Spatio-temporal dithering (frame rate modulation) is supported 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 24-bit palette. TFT 24-bit mode allows the display of up to 1,777,721 simultaneous colors. A wide variety of LCD panels are supported through the use of user-programmable vertical and horizontal synchronization signals, bias signals and pixel clock rates.

The main features of the LCD controller are shown in Table 9-7.

Table 9-7. LCD Controller Features

Features	Descriptions
Panel Support	Supports panel sizes up to 2048x2048 TFT 12/16/18/24-bit color 1/2/4/8-bit mono STN 4/8-bit mono single-scan 8-bit color single-scan 16-bit color dual-scan
Buffer Formats	1/2/4/8-bpp passthru/gray-scale 1/2/4/8-bpp palettized 16-bpp: 5/6/5 and 1/5/5/5 24-bpp: 0/8/8/8 32-bpp: 8/8/8/8
Other Features	Four moveable overlay windows Per-Pixel alpha blending available to each window Optional alpha override or chroma key matching Per-window double buffering support Configurable on-chip memory for palette, gamma, or frame buffer array 256-entry 32-bpp palette 32-bpp gamma correction (with alpha) 8192-bit frame buffer (1 bpp / 2 bpp) Two or four 32-bpp colors plus one 24-bpp background color Sample window support, such as 90x90@1 bpp, 64x64@2 bpp 24-bpp background color 1/8-bpp alpha blending Hardware cursor (32x32x2 bpp) Two pulse width modulation clocks to support digital control of contrast and brightness voltages (requires external filter circuits)

www.DataSheet4U.com

www.DataSheet4U.com

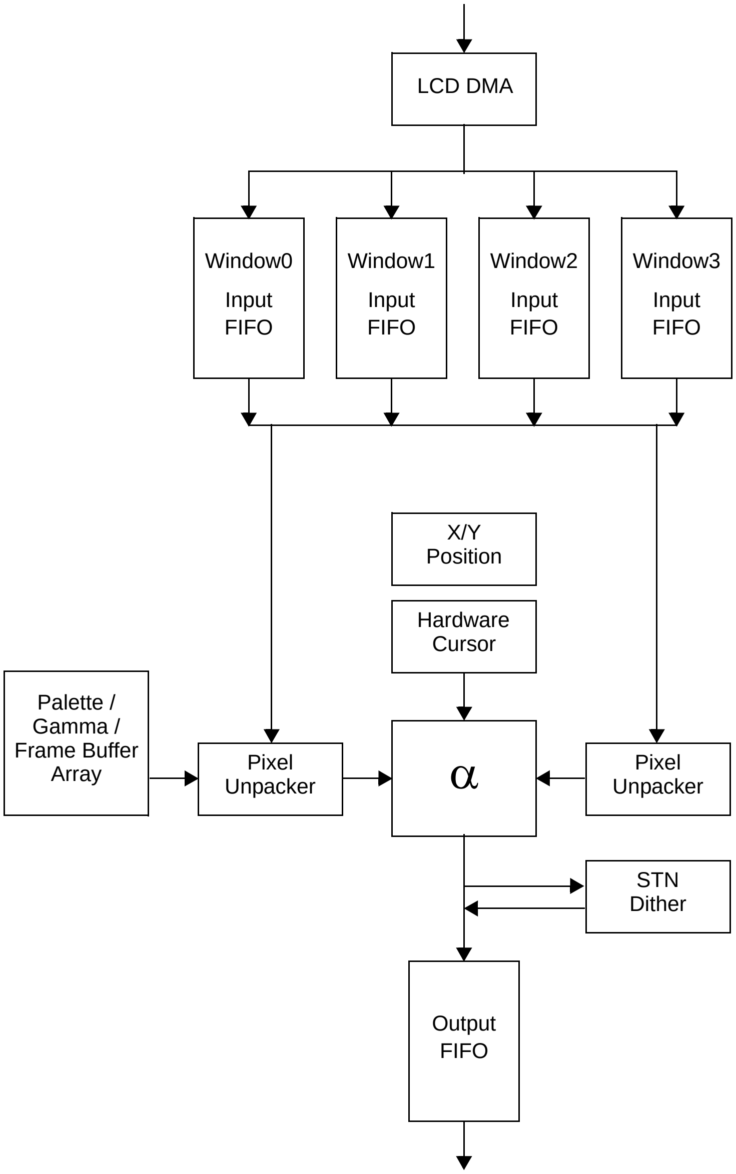


Figure 9-8. LCD Controller Block Diagram

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-8 lists the base address for the LCD controller register block.

Table 9-8. LCD Controller Base Address

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

Table 9-9 lists the LCD controller registers.

Table 9-9. LCD Controller Registers

Offset (Note1)	Register Name	Description
0x0000	—	Reserved
0x0004	lcd_screen	Screen Control Register
0x0008	lcd_backcolor	Screen Control Register
0x000C	lcd_horztiming	Horizontal Timing Register
0x0010	lcd_verttiming	Vertical Timing Register
0x0014	lcd_clockctrl	LCD Clock Control Register
0x0018	lcd_pwmdiv	Pulse Width Modulation Frequency Divider Register
0x001C	lcd_pwmhi	Pulse Width Modulation High Time Register
0x0024	lcd_winenable	Overall Window Control Registers
0x0028	lcd_colorkey	LCD Color Key Register
0x002C	lcd_colorkeymsk	LCD Color Key Mask Register
0x0030	lcd_cursorctrl	Hardware Cursor Control Register
0x0034	lcd_cursorpos	Hardware Cursor Position Register
0x0038 - 0x044	lcd_color*n*	Hardware Cursor and On-Chip Frame Buffer Colors 0-3 Registers
0x0048	lcd_intstatus	Interrupt Registers
0x004C	lcd_intenable	Interrupt Registers
0x0050	lcd_outmask	Output Enable Register
0x0054	lcd_fifoctrl	FIFO Control Register
0x0100	lcd_win0ctrl0	Window 0 Control
0x0104	lcd_win0ctrl1	Window 0 Control
0x0108	lcd_win0ctrl2	Window 0 Control
0x010C	lcd_win0buf0	Window 0 Buffer Address
0x0110	lcd_win0buf1	Window 0 Buffer Address
0x0114	lcd_win0bufctrl	Window 0 Buffer Control
0x0120	lcd_win1ctrl0	Window 1 Control
0x0124	lcd_win1ctrl1	Window 1 Control
0x0128	lcd_win1ctrl2	Window 1 Control
0x012C	lcd_win1buf0	Window 1 Buffer Address
0x0130	lcd_win1buf1	Window 1 Buffer Address
0x0134	lcd_win1bufctrl	Window 1 Buffer Control
0x0140	lcd_win2ctrl0	Window 2 Control

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-9. LCD Controller Registers

Offset (Note1)	Register Name	Description
0x0144	lcd_win2ctrl1	Window 2 Control
0x0148	lcd_win2ctrl2	Window 2 Control
0x014C	lcd_win2buf0	Window 2 Buffer Address
0x0150	lcd_win2buf1	Window 2 Buffer Address
0x0154	lcd_win2bufctrl	Window 2 Buffer Control
0x0160	lcd_win3ctrl0	Window 3 Control
0x0164	lcd_win3ctrl1	Window 3 Control
0x0168	lcd_win3ctrl2	Window 3 Control
0x016C	lcd_win3buf0	Window 3 Buffer Address
0x0170	lcd_win3buf1	Window 3 Buffer Address
0x0174	lcd_win3bufctrl	Window 3 Buffer Control
0x0400 - 0x07FC	lcd_ramarray	RAM Array for Gamma Correction / Palette Lookup / On-Chip Frame Buffer
0x0800 - 0x08FC	lcd_cursorpattern	Hardware Cursor Pattern Memory Block

Note1. See Table 9-8 on page 302 for base address.

9.2.1 Screen Control Register

This register configures the controller for the specific LCD panel. With the exception of the white-data-enable (WD) bit, fields must not be written while the controller is active.

lcd_screen **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31	EN	Enable. 0 Disable the screen. 1 Enable the screen.	R/W	0
30	—	Reserved.	—	—
29:19	X	Screen Width (pixels). Panel width is X + 1.	R/W	0
18:8	Y	Screen Height (pixels). Panel height is Y + 1.	R/W	0
7	WP	White Data Polarity. This is the value driven on the LCD_D[23:0] signals when WD is set.	R/W	0
6	WD	White Data Enable. When WD is set, the LCD_D[23:0] signals are driven with the value programmed in WP. WD is used during the startup and shutdown sequence of some LCD panels. This bit may be written at any time.	R/W	0
5:3	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
2:0	PT	Panel Type. 000 TFT 001 Color Single-Scan STN 010 Color Dual-Scan STN 011 Mono 8-bit Single-Scan STN 100 Mono 4-bit Single-Scan STN All other values are reserved.	R/W	0

lcd_backcolor **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
	▨								RED								GRN								BLU							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:16	RED	Red Background Value.	R/W	0
15:8	GRN	Green Background Value.	R/W	0
7:0	BLU	Blue Background Value.	R/W	0

9.2.2 Horizontal Timing Register

See Figure 9-12 on page 323 and Figure 9-13 on page 324 for a graphical description of the LCD timing parameters.

lcd_horztiming **Offset = 0x000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	▨					HND2									HND1									HPW								
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:27	—	Reserved.	—	—
26:18	HND2	Horizontal Non Display Period 2 in Pixels. Actual period is (HND2 + 1).	R/W	0
17:9	HND1	Horizontal Non Display Period 1 in Pixels. Actual period is (HND1 + 1).	R/W	0
8:0	HPW	Horizontal Sync Pulse Width in Pixels. Actual pulse width is (HPW + 1).	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.2.3 Vertical Timing Register

See Figure 9-12 on page 323 and Figure 9-13 on page 324 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 DMA engine to fetch the start of the next frame. The number of lines required is system dependent.

lcd_verttiming **Offset = 0x0010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						VND2									VND1									VPW								
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:27	—	Reserved	—	—
26:18	VND2	Vertical Non Display Period 2 in Lines. This parameter is not used with STN panels. Actual period is (VND2 + 1).	R/W	0
17:9	VND1	Vertical Non Display Period 1 in Lines. Actual period is (VND1 + 1).	R/W	0
8:0	VPW	Vertical Sync Pulse Width in Lines. This parameter is not used with STN panels. Actual pulse width is (VPW + 1).	R/W	0

9.2.4 LCD Clock Control Register

This register defines the parameters associated with the LCD pins.

lcd_clockctrl **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										EXT	DELAY		PCDD	IB	IC	IH	IV	BF					PCD									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:23	—	Reserved.	—	—
22	EXT	External Clock Select. 0 Use the internal clock *lcd_intclk*. See Section 10.1.1.1 "Clock Generator Registers" on page 367. 1 Use the external clock LCD_CLKIN (GPIO[3]).	R/W	0
21:20	DELAY	Pixel Clock Delay.	R/W	0
19	PCDD	Pixel Clock Divider Disable. 0 Use the pixel clock divisor (PCD). See the PCD description below. 1 Disable the pixel clock divisor. Pixel clock runs at same frequency as the LCD clock.	R/W	0
18	IB	Invert Bias. 0 Do not invert the signal. 1 Invert the signal.	R/W	0
17	IC	Invert Pixel Clock. 0 Launch data on the rising edge of the pixel clock. 1 Launch data on the falling edge of the pixel clock.	R/W	0
16	IH	Invert Line Clock. 0 Do not invert the clock. 1 Invert the clock.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
15	IV	Invert Frame Clock. 0 Do not invert the clock. 1 Invert the clock.	R/W	0
14:10	BF	Bias Signal Frequency. Used only with STN panels. The Bias signal toggles every BF line clocks. The actual number of lines is (BF + 1).	R/W	0
9:0	PCD	Pixel Clock Divisor. Determines the frequency of the pixel clock (LCD_PCLK) derived from the LCD controller clock where: F_{pclk} = LCD Clock / (2 * (PCD+1)). For STN mono 8-bit panels PCD must be greater than 2. For STN mono 4-bit panels PCD must be greater than zero. For STN color panels PCD must be greater than zero. Note: To run the pixel clock at the same frequency as the LCD clock, disable the divisor by setting PCDD (bit 19).	R/W	0

9.2.5 Pulse Width Modulation Frequency Divider Register

This register controls the frequency of the two pulse width modulation (PWM) clocks LCD_PWM[1:0]. The PWM clocks are based off the LCD controller clock.

lcd_pwmdiv **Offset = 0x0018**

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

Bits	Name	Description	R/W	Default
31	EN	Enable. 0 Disable PWM clocks. 1 Enable PWM clocks.	R/W	0
30:18	—	Reserved.	—	—
17:0	DIV	PWM Frequency Divider. Determines the frequency for the PWM clocks: F_{PWM} = LCD Clock / (DIV+1)	R/W	0

9.2.6 Pulse Width Modulation High Time Register

This register controls the duty cycle of the two PWM clocks, LCD_PWM0 and LCD_PWM1**.**

lcd_pwmhi - PWM High Time **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
	HI1																HI0															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	HI1	High Time for LCD_PWM1. Duty Cycle = (HI1 + 1) / (DIV + 1)	R/W	0
15:0	HI0	High Time for LCD_PWM0. Duty Cycle = (HI0 + 1) / (DIV + 1)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.2.7 Overall Window Control Registers

lcd_winenable **Offset = 0x0024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													WEN3	WEN2	WEN1	WEN0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3	WEN3	Window 3 Enable. 0 Disable 1 Enable	R/W	0
2	WEN2	Window 2 Enable. 0 Disable 1 Enable	R/W	0
1	WEN1	Window 1 Enable. 0 Disable 1 Enable	R/W	0
0	WEN0	Window 0 Enable. 0 Disable 1 Enable	R/W	0

9.2.8 LCD Color Key Register

lcd_colorkey **Offset = 0x0028**

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

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:16	RED	Red Color Key.	R/W	0
15:8	GRN	Green Color Key.	R/W	0
7:0	BLU	Blue Color Key.	R/W	0

9.2.9 LCD Color Key Mask Register

lcd_colorkeymsk **Offset = 0x002C**

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

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:16	RED	Red Color Key Mask.	R/W	0xFF
15:8	GRN	Green Color Key Mask.	R/W	0xFF
7:0	BLU	Blue Color Key Mask.	R/W	0xFF

www.DataSheet4U.com

www.DataSheet4U.com

9.2.10 Hardware Cursor Control Register

lcd_cursorctrl **Offset = 0x0030**

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

Bits	Name	Description	R/W	Default
31:1	—	Reserved.	—	—
0	EN	Enable. 0 Disable the hardware cursor. 1 Enable the hardware cursor.	R/W	0

9.2.11 Hardware Cursor Position Register

This register controls where the top left corner of the hardware cursor is positioned. This register is latched at the start of every frame to avoid tearing.

lcd_cursorpos **Offset = 0x0034**

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

Bits	Name	Description	R/W	Default
31:27	XOFF	X Offset. XOFF represents the X pixel offset within the 32x32 cursor pattern at which the displayed portion of the cursor is to begin. Normally, XOFF is cleared in order to display the entire cursor pattern. However, for cursors with a hot spot not at the left edge of the pattern, it may be necessary to display only the right-most pixels of the cursor when the cursor moves close to the left edge of the display.	R/W	0
26:16	XPOS	X Position. XPOS represents the X coordinate of the pixel at which the upper left corner of the cursor is to be displayed. XPOS is referenced to the screen origin (0,0), which is the pixel in the upper left corner of the screen.	R/W	0
15:11	YOFF	Y Offset. YOFF represents the Y pixel offset within the 32x32 cursor pattern at which the displayed portion of the cursor is to begin. Normally, YOFF is cleared in order to display the entire cursor pattern. However, for cursors with a hot spot not at the top edge of the pattern, it may be necessary to display only the bottom-most pixels of the cursor when the cursor moves close to the top edge of the display.	R/W	0
10:0	YPOS	Y Position. YPOS represents the Y coordinate of the pixel at which the upper left corner of the cursor is to be displayed. YPOS is referenced to the screen origin (0,0), which is the pixel in the upper left corner of the screen.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.2.12 Hardware Cursor and On-Chip Frame Buffer Colors 0-3 Registers

These four registers specify the colors displayed for a hardware cursor or on-chip frame buffer pixel.

lcd_color0 **Offset = 0x0038**

lcd_color1 **Offset = 0x003C**

lcd_color2 **Offset = 0x0040**

lcd_color3 **Offset = 0x0044**

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
	A								RED								GRN								BLU							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:24	A	Alpha value.	R/W	0
23:16	RED	Red value.	R/W	0
15:8	GRN	Green value.	R/W	0
7:0	BLU	Blue value.	R/W	0

9.2.13 Interrupt Registers

These registers (**lcd_intstatus** and **lcd_intenable**) have identical formats. If a bit is set in the interrupt enable register and the corresponding condition becomes true, an interrupt is issued with the corresponding bit in **lcd_intstatus** set. The interrupt for the LCD controller should be programmed as a high level type.

All interrupts must be cleared by writing a 1 to the corresponding bit in **lcd_intstatus**; writing a 0 has no effect. These registers may be read and written while the LCD controller is active.

lcd_intstatus **Offset = 0x0048**

lcd_intenable **Offset = 0x004C**

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
																			IU3	IU2	IU1	IU0		OFU					WAIT	SD	SA	SS
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:14	—	Reserved.	—	—
13	IU3	Window 3 Input FIFO Underflow.	R/W	0
12	IU2	Window 2 Input FIFO Underflow.	R/W	0
11	IU1	Window 1 Input FIFO Underflow.	R/W	0
10	IU0	Window 0 Input FIFO Underflow.	R/W	0
9	—	Reserved.	—	—
8	OFU	Output FIFO Underflow. 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
7:4	—	Reserved.	—	—
3	WAIT	Wait. Controller is waiting for this bit to be cleared to begin rastering at the beginning of each frame.	R/W	0
2	SD	Shutdown. The shutdown condition occurs when software disables the screen (**lcd_screen**[EN]=0) and the LCD controller has finished displaying the last frame. After SD is set all registers of the controller can be written.	R/W	0
1	SA	Start of Active Video. Occurs at the end of the vertical retrace.	R/W	0
0	SS	Start Vertical Sync Period.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.c

9.2.14 Output Enable Register

For power considerations, this register allows the 24 data pins to be selectively enabled.

lcd_outenable **Offset = 0x0050**

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
									ENABLE																							
Def.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W	Default
31:24	—	Reserved.	—	—
23:0	ENABLE	Output Enable. 0 Disable the corresponding pin. Prevent the data pin from toggling. 1 Enable the corresponding pin.	R/W	0xFFFFFF

9.2.15 FIFO Control Register

This register controls at which depths the input FIFOs request data.

lcd_fifoctrl **Offset = 0x0054**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			IF3	REQ3							IF2	REQ2							IF1	REQ1							IF0	REQ0				
Def.	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1

Bits	Name	Description	R/W	Default
31:30	—	Reserved.	—	—
29	IF3	Initial fetch max for input FIFO 3.	R/W	1
28:24	REQ3	Request depth (beats - 1) for input FIFO 3.	R/W	0xF
23:22	—	Reserved.	—	—
21	IF2	Initial fetch max for input FIFO 2.	R/W	1
20:16	REQ2	Request depth (beats - 1) for input FIFO 2.	R/W	0xF
15:14	—	Reserved.	—	—
13	IF1	Initial fetch max for input FIFO 1.	R/W	1
12:8	REQ1	Request depth (beats - 1) for input FIFO 1.	R/W	0xF
7:6	—	Reserved.	—	—
5	IF0	Initial fetch max for input FIFO 0.	R/W	1
4:0	REQ0	Request depth (beats - 1) for input FIFO 0.	R/W	0xF

9.2.16 Overlay Window Control Registers

lcd_win0ctrl0 **Offset = 0x0100**

lcd_win1ctrl0 **Offset = 0x0120**

lcd_win2ctrl0 **Offset = 0x0140**

lcd_win3ctrl0 **Offset = 0x0160**

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
	OX											OY											A								AEN	
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:21	OX	Origin X. Origin X location of window in pixels	R/W	0
20:10	OY	Origin Y. Origin Y location of window in pixels	R/W	0
9:2	A	Alpha. Alpha value used for this window when AEN is specified, or WCKEN and ChromaKey matches opposing color.	R/W	0
1	AEN	Alpha Override Enable. Use provided Alpha (A) for this window, overriding alpha in frame buffer (if available).	R/W	0
0	—	Reserved.	—	—

lcd_win0ctrl1 Offset = 0x0104

lcd_win1ctrl1 Offset = 0x0124

lcd_win2ctrl1 Offset = 0x0144

lcd_win3ctrl1 Offset = 0x0164

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
	PRI		PIPE	FORM				CCO	PO		SZX											SZY										
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:30	PRI	Priority. When windows on the same pipe side contain overlapping pixels, the pixels from the window with the highest priority value is used. When windows on opposite pipe sides contain overlapping pixels, the pixels from the window with the highest priority's alpha value is used to blend the two overlapping pixels together. No two windows can have the same priority.	R/W	0
29	PIPE	Data Pipe. 0 Left (Palette / Gamma / On-Chip Frame Buffer available) 1 Right	R/W	0
28:25	FORM	Buffer Format. See Table 9-11 on page 315.	R/W	0
24	CCO	Color Channel Orientation. 0 RGB: Follow the pixel format shown in Table 9-11 on page 315. 1 BGR: Swap blue and red in the pixel format.	R/W	0
23:22	PO	Pixel Ordering. See Table on page 314. For on-chip-frame-buffer mode, PO must be cleared.	R/W	0
21:11	SZX	Size X in Pixels. Width of window is SZX + 1.	R/W	0
10:0	SZY	Size Y in Pixels. Height of window is SZY + 1.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

lcd_win0ctrl2 **Offset = 0x0108**

lcd_win1ctrl2 **Offset = 0x0128**

lcd_win2ctrl2 **Offset = 0x0148**

lcd_win3ctrl2 **Offset = 0x0168**

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
							CKMODE		DBM	RAM		BX													SCX				SCY			
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:26	—	Reserved.	—	—
25:24	CKMODE	Colorkey Mode. 00 Disable colorkey. 01 Match full screen colorkey to other pipe's pixel. Uses next pixel from this buffer. Must be only enabled window on this pipe. 10 Windowed colorkey matches other pipe's pixel. Selects Alpha from other pipe, overriding priority (Video man mode) 11 Windowed colorkey matches other pipe's pixel. Selects Alpha from this window, overriding priority (Weatherman mode)	R/W	0
23	DBM	Double Buffer Mode. 0 Buffer selection remains static. 1 Buffer selection alternates at retrace.	R/W	0
22:21	RAM	RAM Array Usage (Available to left pipe only.). 00 None 01 Palette lookup 10 Gamma correction 11 On-chip frame buffer	R/W	0
20:8	BX	Buffer Line Width in Bytes. Most efficient mode is when BX equals (SizeX * bpp). Must be bpp aligned. Used for padded frame buffers and sub selecting a frame within a larger frame. For on-chip-frame-buffer mode, this value is in pixels and must be multiple of 8.	R/W	0
7:4	SCX	Scale X. X dimension scale factor. Valid only for on-chip frame buffers (RAM = 0b11). 000 1 (no scaling) 001 2 010 4 011 Invalid 1xx Reserved The corresponding window width (**lcd_win*n*ctrl1**[SZX]) must be programmed to the width of the image after scaling (x1, x2, x4).	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
3:0	SCY	Scale Y. Y dimension scale factor. Valid only for on-chip frame buffers (RAM=11). 000 1 (no scaling) 001 2 010 4 011 Invalid 1xx Reserved The corresponding window height (**lcd_win*n*ctrl1**[SZY]) must be programmed to the height of the image after scaling (x1, x2, x4).	R/W	0

lcd_win0buf0 **Offset = 0x010C**

lcd_win1buf0 **Offset = 0x012C**

lcd_win2buf0 **Offset = 0x014C**

lcd_win3buf0 **Offset = 0x016C**

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

Bits	Name	Description	R/W	Default
31:0	ADDR	Window *n* Buffer 0 Address. Must be bpp aligned, doubleword (8 bytes) aligned preferred. For on-chip-frame-buffer mode, the buffer address is an offset into the RAM array and must be multiple of 256.	R/W	0

lcd_win0buf1 **Offset = 0x0110**

lcd_win1buf1 **Offset = 0x0130**

lcd_win2buf1 **Offset = 0x0150**

lcd_win3buf1 **Offset = 0x0170**

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

Bits	Name	Description	R/W	Default
31:0	ADDR	Window *n* Buffer 1 Address. Must be bpp aligned, doubleword (8 bytes) aligned preferred. For on-chip-frame-buffer mode, the buffer address is an offset into the RAM array and must be multiple of 256.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

lcd_win0bufctrl **Offset = 0x0114**

lcd_win1bufctrl **Offset = 0x0134**

lcd_win2bufctrl **Offset = 0x0154**

lcd_win3bufctrl **Offset = 0x0174**

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

Bits	Name	Description	R/W	Default
31:2	—	Reserved.	—	—
1	DB	Double Buffer. Current buffer selected for window *n* 0 Buffer 0 1 Buffer 1	R	0
0	DBN	Override Buffer. Selected at Next Retrace for Window *n*. 0 Next buffer is Buffer 0. 1 Next buffer is Buffer 1. The DBN setting overrides the **lcd_win*n*ctrl2**[DBM] setting.	R/W	0

Table 9-10. Pixel Ordering

BPP = 1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO																																
00	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
01	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
10	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
11	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

BPP = 2

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
PO																																
00	p0		p1		p2		p3		p4		p5		p6		p7		p8		p9		p10		p11		p12		p13		p14		p15	
01	p12		p13		p14		p15		p8		p9		p10		p11		p4		p5		p6		p7		p0		p1		p2		p3	
10	p3		p2		p1		p0		p7		p6		p5		p4		p11		p10		p9		p8		p15		p14		p13		p12	
11	p15		p14		p13		p12		p11		p10		p9		p8		p7		p6		p5		p4		p3		p2		p1		p0	

BPP = 4

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
PO																																
00	p0				p1				p2				p3				p4				p5				p6				p7			
01	p6				p7				p4				p5				p2				p3				p0				p1			
10	p1				p0				p3				p2				p5				p4				p7				p6			
11	p7				p6				p5				p4				p3				p2				p1				p0			

BPP = 8

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
PO																																
00	p0								p1								p2								p3							
01	p3								p2								p1								p0							

www.DataSheet4U.com

www.DataSheet4U.com

<table>
<tr><th colspan="33">BPP = 12/16</th></tr>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>PO</b></td><td colspan="32"></td></tr>
<tr><td>00</td><td colspan="16">p0</td><td colspan="16">p1</td></tr>
<tr><td>01</td><td colspan="16">p1</td><td colspan="16">p0</td></tr>
</table>

<table>
<tr><th colspan="33">BPP = 24/32</th></tr>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>PO</b></td><td colspan="32"></td></tr>
<tr><td>00</td><td colspan="32">p0</td></tr>
</table>

Table 9-11. Frame Buffer Pixel Format

lcd_win*n*ctrl1 [FORM]	**Description**	**Memory Width**	**Format**
0	1 bpp	1	I:1
1	2 bpp	2	I:2
2	4 bpp	4	I:4
3	8 bpp	8	I:8
4	12 bpp (4/4/4)	16	P:4 R:4 G:4 B:4
5	16 bpp (6/5/5)	16	R:6 G:5 B:5
6	16 bpp (5/6/5)	16	R:5 G:6 B:5
7	16 bpp (5/5/6)	16	R:5 G:5 B:6
8	16 bpp w/ intensity (1/5/5/5)	16	I:1 R:5 G:5 B:5
9	16 bpp w/ intensity (5/5/5/1)	16	R:5 G:5 B:5 I:1
10	16 bpp w/ alpha (1/5/5/5)	16	α:1 R:5 G:5 B:5
11	16 bpp w/ alpha (5/5/5/1)	16	R:5 G:5 B:5 α:1
12	24 bpp (0/8/8/8)	32	P:8 R:8 G:8 B:8
13	32 bpp w/ alpha (8/8/8/8)	32	α:8 R:8 G:8 B:8
14-15	Reserved	—	—

Notes:

I - intensity, P - padding, R - red, B - blue, G - green, α - alpha

Setting **lcd_win*n*ctrl1**[CCO] swaps red and blue.

www.DataSheet4U.com

www.DataSheet4U.com

9.2.17 RAM Array—Palette Lookup

When a window’s RAM array is used for palette lookup (**lcd_win*n*ctrl2**[RAM] = 0b01), each pixel in the window’s frame buffer is treated as an index into the RAM array to select the actual color/intensity.

lcd_ramarray MONOCHROME (Palette Lookup) **Offset = 0x0400 - 0x07FC**

Bit	31								...							n	n-1							...								0
																	PMI															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:*n*	—	Reserved.	—	—
(*n*-1):0	PMI	*n*-bit Monochromatic Panel Intensity.	R/W	UNDEF

lcd_ramarray COLOR (Palette Lookup) **Offset = 0x0400 - 0x07FC**

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
	A								RED								GRN								BLU							
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:24	A	Alpha.	R/W	UNDEF
23:16	RED	Red.	R/W	UNDEF
15:8	GRN	Green.	R/W	UNDEF
7:0	BLU	Blue.	R/W	UNDEF

Figure 9-9 shows the RAM array used for palette lookup and the corresponding colors selected.

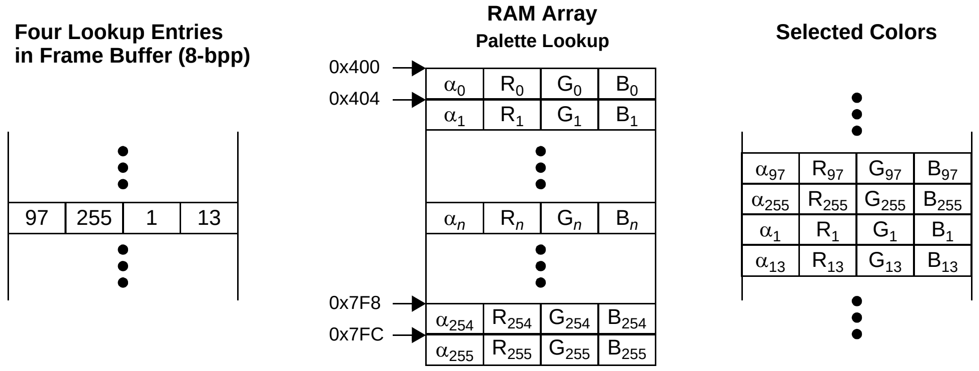

Four Lookup Entries
in Frame Buffer (8-bpp)
97
255
1
13
RAM Array
Palette Lookup
0x400
0x404
0x7F8
0x7FC
α_0 R_0 G_0 B_0
α_1 R_1 G_1 B_1
α_n R_n G_n B_n
α_{254} R_{254} G_{254} B_{254}
α_{255} R_{255} G_{255} B_{255}
Selected Colors
α_{97} R_{97} G_{97} B_{97}
α_{255} R_{255} G_{255} B_{255}
α_1 R_1 G_1 B_1
α_{13} R_{13} G_{13} B_{13}

Figure 9-9. RAM Array for Palette Lookup (lcd_win*n*ctrl2[RAM] = 01)

www.DataSheet4U.com

www.DataSheet4U.com

9.2.18 RAM Array—Gamma Correction

When a window’s RAM array is used for gamma correction (**lcd_win*n*ctrl2**[RAM] = 0b10), each pixel’s alpha, red, green, and blue color component is treated as an index into the RAM array. The corresponding alpha, red, green, or blue channel intensity value at that index is used in the actual color.

lcd_ramarray (Gamma Correction) **Offset = 0x0400 - 0x07FC**

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
	ACI								RCI								GCI								BCI							
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:24	ACI	Alpha channel intensity.	R/W	UNDEF
23:16	RCI	Red channel intensity.	R/W	UNDEF
15:8	GCI	Green channel intensity.	R/W	UNDEF
7:0	BCI	Blue channel intensity.	R/W	UNDEF

Figure 9-10 shows the RAM array used for gamma correction and the corresponding colors selected.

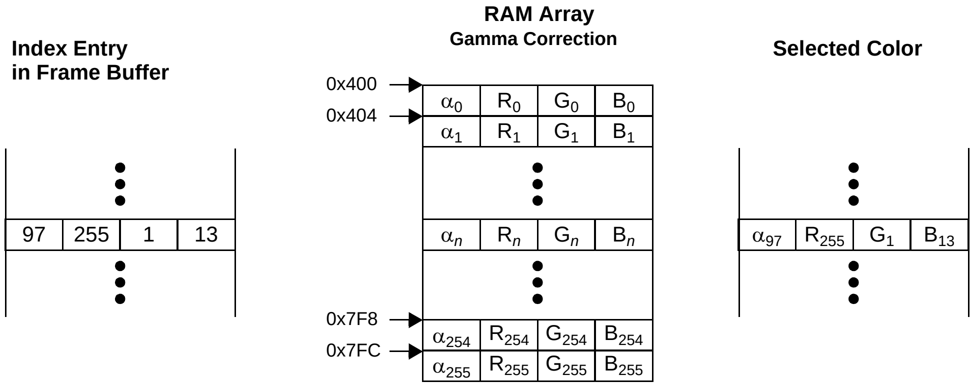


Figure 9-10. RAM Array for Gamma Correction (lcd_win*n*ctrl2[RAM] = 10)

www.DataSheet4U.com

www.DataSheet4U.com

9.2.19 RAM Array—On-Chip Frame Buffer

When a window’s RAM array is used as an on-chip frame buffer (**lcd_win*n*ctrl2**[RAM] = 0b11), each pixel in the RAM array is used to select one of the four 32-bit **lcd_color*n*** colors. The format of each 32-bit RAM array word depends on whether 1 or 2 bpp is used.

lcd_ramarray 1-bpp (On-Chip Frame Buffer) **Offset = 0x0400 - 0x07FC**

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
	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
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31, 30, 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	P0 - P31	On-chip Frame Buffer Pixel Pattern. These bits specify the color displayed for each of the on-chip frame buffer pixels when the window’s pixel format is programmed to 1 bpp (**lcd_win*n*ctrl1**[FORM] = 0). 0 Color 0 (**lcd_color0**) 1 Color 1 (**lcd_color1**)	R/W	0

lcd_ramarray 2-bpp (On-Chip Frame Buffer) **Offset = 0x0400 - 0x07FC**

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
	P0		P1		P2		P3		P4		P5		P6		P7		P8		P9		P10		P11		P12		P13		P14		P15	
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:30, 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	P0 - P15	On-chip Frame Buffer Pixel Pattern. These bits specify the color displayed for each of the on-chip frame buffer pixels when the window’s pixel format is programmed to 2 bpp (**lcd_win*n*ctrl1**[FORM] = 1). 00 Color 0 (**lcd_color0**) 01 Color 1 (**lcd_color1**) 10 Color 2 (**lcd_color2**) 11 Color 3 (**lcd_color3**)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Figure 9-11 shows the RAM array used as an on-chip frame buffer and the corresponding colors selected.

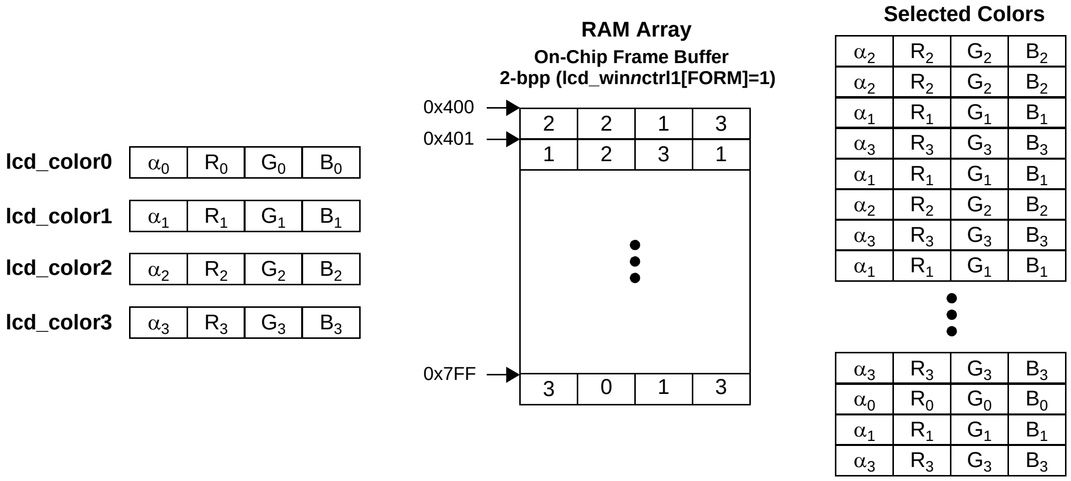

RAM Array
On-Chip Frame Buffer
2-bpp (lcd_win*n*ctrl1[FORM]=1)
Selected Colors
lcd_color0 α_0 R_0 G_0 B_0
lcd_color1 α_1 R_1 G_1 B_1
lcd_color2 α_2 R_2 G_2 B_2
lcd_color3 α_3 R_3 G_3 B_3
0x400
2 2 1 3
0x401
1 2 3 1
0x7FF
3 0 1 3
α_2 R_2 G_2 B_2
α_2 R_2 G_2 B_2
α_1 R_1 G_1 B_1
α_3 R_3 G_3 B_3
α_1 R_1 G_1 B_1
α_2 R_2 G_2 B_2
α_3 R_3 G_3 B_3
α_1 R_1 G_1 B_1
α_3 R_3 G_3 B_3
α_0 R_0 G_0 B_0
α_1 R_1 G_1 B_1
α_3 R_3 G_3 B_3

Figure 9-11. RAM Array for On-Chip Frame Buffer (lcd_win*n*ctrl2[RAM] = 11)

9.2.20 Hardware Cursor Pattern

The hardware cursor is 32 by 32 pixels. Each pixel is represented by 2 bits. The value of these bits specify the color of the displayed pixel. The pixel values start at the top left of the cursor and move to the bottom right of the cursor. There are 16 pixel values contained in each word address. To represent the entire hardware cursor 64 words (offset 0x0800 to 0x08FC) are used. To avoid visual artifacts these registers must be written only when the cursor is disabled or during the vertical sync period (**lcd_intstatus**[SS] interrupt).

lcd_cursorpattern **Offset = 0x0800 - 0x08FC**

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
	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Def.	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X

Bits	Name	Description	R/W	Default
31:30, 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	P15 - P0	Hardware Cursor Pixel Pattern. These bits specify the color displayed for each of the hardware cursor pixels. 00 Color 0 (**lcd_color0**) 01 Color 1 (**lcd_color1**) 10 Color 2 (**lcd_color2**) 11 Color 3 (**lcd_color3**)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.2.21 Hardware Considerations

Table 9-12 shows the LCD controller signals.

Table 9-12. LCD Controller Signals

Signal	Input/Output	Definition
LCD_FCLK	O	Frame Clock.
LCD_LCLK	O	Line Clock.
LCD_PCLK	O	Pixel Clock.
LCD_D[1:0]	O	LCD Data. Muxed with GPIO[201:200]. GPIO[201:200] are the default signals coming out of hardware reset, runtime reset, and Sleep.
LCD_D[7:2]	O	LCD Data.
LCD_D[8]	O	LCD Data. Muxed with GPIO[210]. GPIO[210] is the default signal coming out of hardware reset, runtime reset, and Sleep.
LCD_D[15:9]	O	LCD Data.
LCD_D[16]	O	LCD Data. Muxed with GPIO[211]. GPIO[211] is the default signal coming out of hardware reset, runtime reset, and Sleep.
LCD_D[23:17]	O	LCD Data.
LCD_BIAS	O	Bias Clock.
LCD_CLKIN (GPIO[3])	I	LCD Clock Source (Optional).
LCD_PWM0	O	Pulse Width Modulation Clock 0. Muxed with U1DCD#. U1DCD# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.
LCD_PWM1	O	Pulse Width Modulation Clock 1. Muxed with U1RI#. U1RI# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-13 shows the LCD controller data signal usage.

Table 9-13. LCD Controller Data Pin Usage

LCD Signal	Mono STN Panel		Color STN Panel		Color TFT Panel(Note1)		
	4-bit	8-bit	Single	Dual	12-bit	18-bit	24-bit
LCD_D[0]	M0	M0	D0	D0			B0
LCD_D[1]	M1	M1	D1	D1			B1
LCD_D[2]	M2	M2	D2	D2		B0	B2
LCD_D[3]	M3	M3	D3	D3		B1	B3
LCD_D[4]		M4		D4	B0	B2	B4
LCD_D[5]		M5		D5	B1	B3	B5
LCD_D[6]		M6		D6	B2	B4	B6
LCD_D[7]		M7		D7	B3	B5	B7
LCD_D[8]							G0
LCD_D[9]							G1
LCD_D[10]						G0	G2
LCD_D[11]						G1	G3
LCD_D[12]					G0	G2	G4
LCD_D[13]					G1	G3	G5
LCD_D[14]					G2	G4	G6
LCD_D[15]					G3	G5	G7
LCD_D[16]							R0
LCD_D[17]							R1
LCD_D[18]						R0	R2
LCD_D[19]						R1	R3
LCD_D[20]					R0	R2	R4
LCD_D[21]					R1	R3	R5
LCD_D[22]					R2	R4	R6
LCD_D[23]					R3	R5	R7

Note1.For TFT panels the R and B signals are reversed if **lcd_win*n*ctrl1**[CCO] is set.

www.DataSheet4U.com

www.DataSheet4U.com

9.2.22 Programming Considerations

The first step in enabling the LCD controller is to program the LCD clock generator to the desired frequency. See Section 10.1.1.1 "Clock Generator Registers" on page 367. Alternatively, the external clock LCD_CLKIN (using the GPIO[3] signal function programmed in **sys_pinfunc**[EX1]) can be used as the LCD clock source (**lcd_clockctrl**[EXT] = 1).

When starting the LCD controller, first program the configuration for the panel in use, then set the screen enable bit **lcd_screen**[EN]. The LCD controller's configuration must not be changed while the controller is enabled.

To disable the LCD controller, disable the screen by clearing **lcd_screen**[EN], then wait for hardware to set **lcd_intstatus**[SD] (shutdown) before re-configuring the controller.

9.2.22.1 Colorkey - Full Screen

In full screen colorkey mode (**lcd_win*n*ctrl2**[CKMODE] = 01), only one of the pipes is used for colorkey matching. One window is enabled and is pointed at a buffer. Origin X and origin Y (**lcd_win*n*ctrl0**[OX, OY]) do not apply in this mode. The colorkey and colorkey mask (**lcd_colorkey** and **lcd_colorkeymsk**) are programmed for colors coming from the other pipe, which may contain 0-3 of the remaining windows. If the colors match, a pixel is pulled from the colorkey buffer and overlayed over the other pipe's pixel. The buffer size must match exactly the number of matched pixels.

For example, a buffer can be configured to point to a video frame. A green (0x00FF00) box can be drawn on the graphics buffer in the other pipe. When the colorkey matches the green box, the video pixel is displayed instead of the green box. If the green box moves, so does the video.

9.2.22.2 Colorkey - Windowed

In windowed colorkey mode (**lcd_win*n*ctrl2**[CKMODE] = 10 or 11), the colorkey matching determines which window's alpha value to use, instead of the window priority when in full screen mode. All window positioning features are available.

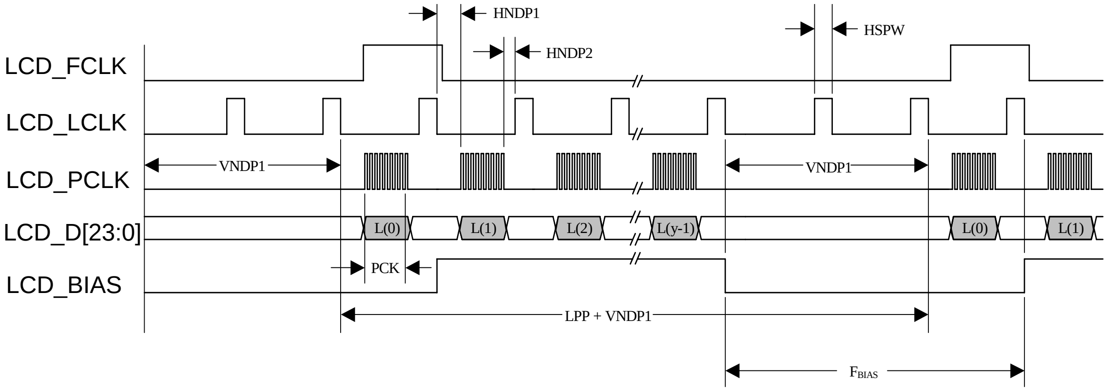
- For **lcd_win*n*ctrl2**[CKMODE] = 10, if the colorkey matches, the alpha of the colorkey window is used.
- For **lcd_win*n*ctrl2**[CKMODE] = 11, if the colorkey matches, the alpha of the other pipe is used.

www.DataSheet4U.com

www.DataSheet4U.com

9.2.22.3 Definition of Timing Parameters

The timing diagrams shown in Figure 9-12 and Figure 9-13 on page 324 show the definitions of the timing registers and an example for each mode.



Notes: PCK$_{color}$ = (PixelsPerLine*3)/DataBusWidth

PCK$_{mono}$ = (PixelsPerLine)/DataBusWidth

F$_{PCK}$ = (LCD Clock)/(2*(PCD+1))

In this diagram: F$_{BIAS}$ = 3 (**lcd_clkcontrol**[BF] = 0b00010)

HNDP1 = 4 (**lcd_horztiming**[HN1] = 0b00000011)

HNDP2 = 2 (**lcd_horztiming**[HN2] = 0b00000001)

HSPW = 3 (**lcd_horztiming**[HPW] = 0b000010)

VNDP1 = 2 (**lcd_verttiming**[VN1] = 0b000001)

LPP = y

FRAME ↑ transitions at the same time as first PCLK ↑

FRAME ↓ transitions one PCLK period after LCLK ↓

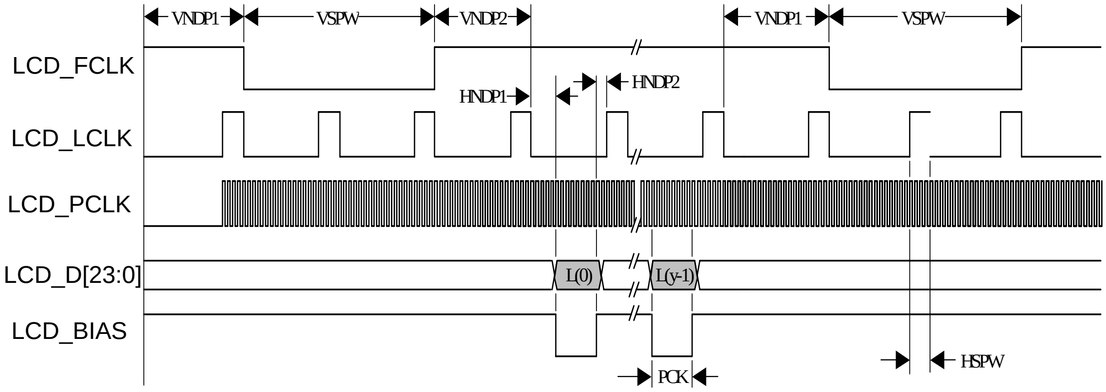
FRAME, LCLK, PCLK shown here with

lcd_clkcontrol[IC:IH:IV] = 0b000

Figure 9-12. STN (Passive Mode) Timing

www.DataSheet4U.com

www.DataSheet4U.com



Notes: PCK = PixelsPerLine
F_{PCK} = (LCD Clock)**/**(2*(PCD+1))

In this diagram:
F_{BIAS} = NA (**lcd_clkcontrol**[BF] = 0bXXXXX)
HNDP1 = 5 (**lcd_horztiming**[HN1] = 0b00000100)
HNDP2 = 2 (**lcd_horztiming**[HN2] = 0b00000001)
HSPW = 4 (**lcd_horztiming**[HPW] = 0b000011)
VNDP1 = 1 (**lcd_verttiming**[VN1] = 0b00000000)
VNDP2 = 1 (**lcd_verttiming**[VN2] = 0b00000000)
VSPW = 2 (**lcd_verttiming**[VPW] = 0b000001)
LPP = y
FRAME transitions at the same time as LCLK goes inactive
FRAME, LCLK, PCLK, and BIAS shown here with **lcd_clkcontrol**[IB:IC:IH:IV] = 0b1001

Figure 9-13. TFT (Active Mode) Timing

www.DataSheet4U.com

w w w . D a t a S h

9.3 AES Cryptography Engine (Au1250™ processor only)

The Au1250™ processor includes an integrated 128-bit AES cryptography engine that supports the following operation modes:

- Electronic codebook (ECB)
- Cipher block chaining (CBC)
- 128-bit cipher feedback (CFB)
- Output feedback (OFB)

9.3.1 AES Cryptography Engine Registers

Table 9-14 shows the base address for the registers of the AES cryptography engine.

Table 9-14. AES Cryptography Engine Base Address

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

Table 9-15 lists the registers of the AES cryptography engine.

Table 9-15. AES Cryptography Engine Registers

Offset (Note1)	Register Name	Description
0x0000	aes_status	AES Status and Control Register
0x0004	aes_indata	Input Data FIFO Register
0x0008	aes_outdata	Output Data FIFO Register
0x000C	aes_intcause	Pending Interrupt Cause Register
0x0010	aes_config	Configuration Register

Note1. See Table 9-14 for base address.

9.3.1.1 AES Status and Control Register

This register is used for starting the de-/encryption process, monitoring when payload data can be read/written, and controlling clocking and interrupts.

aes_status **Offset = 0x0000**

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

Bits	Name	Description	R/W	Default
31:6	—	Reserved.	—	—
5	IN	Input Data FIFO Ready. 0 The input data FIFO is full. 1 The input data FIFO has space for at least 4 words of data to be written.	R	0
4	OUT	Output Data FIFO Ready. 0 The output data FIFO is empty. 1 The output data FIFO contains at least 4 words of processed data ready to be read.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
3:2	CR	Clock Ratio. CR is a divider between the peripheral bus clock and the AES cryptography engine clock. 00 Do not divide the clock. Run the AES cryptography engine clock at the same frequency as peripheral bus clock. 01 Divide by 2. 10 Divide by 4. 11 Divide by 8.	R/W	0
1	IE	Interrupt Enable for the AES Cryptography Engine. 0 Disable interrupts. 1 Enable interrupts.	R/W	0
0	PS	Process Start. 0 Stop the de-/encryption process and clear interrupts pending in **aes_intcause**. 1 Start the de-/encryption process.	R/W	0

9.3.1.2 Input Data FIFO Register

This register contains the data waiting to be processed by the AES cryptography engine. The **aes_indata** register provides access to the input data FIFO. Writes to **aes_indata** are allowed only after processing has started (**aes_status**[PS] = 1) and no interrupts are pending in **aes_intcause**.

The input data FIFO is 8-words deep. If a FIFO overflow occurs, the **aes_intcause**[OVR] bit is set and an interrupt is triggered, if enabled (**aes_status**[IE] = 1).

aes_indata **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:0	INDATA	Input Data.	W	0

9.3.1.3 Output Data FIFO Register

This register contains the processed data. The **aes_outdata** register provides access to the output data FIFO. Reads from **aes_outdata** are allowed only after processing has started (**aes_status**[PS] = 1) and no interrupts are pending in **aes_intcause**.

The output data FIFO is 8-words deep. If a FIFO underflow occurs, the **aes_intcause**[UND] bit is set and an interrupt is triggered, if enabled (**aes_status**[IE] = 1).

aes_outdata **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
	OUTDATA																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:0	OUTDATA	Output Data.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

9.3.1.4 Pending Interrupt Cause Register

This register indicates the cause of an interrupt generated by the AES cryptography engine. A interrupt condition is true when the corresponding bit is set.

After processing an interrupt, the corresponding interrupt cause bit must be cleared. Writes to **aes_indata** or reads from **aes_outdata** are not allowed while an **aes_intcause** bit is set.

aes_intcause **Offset = 0x000c**

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

Bits	Name	Description	R/W	Default
31:3	—	Reserved.	—	—
2	UND	Output Data FIFO Underflow. Indicates an output data FIFO underflow has occurred.	R/W	0
1	OVR	Input Data FIFO Overflow. Indicates an input data FIFO overflow has occurred.	R/W	0
0	RDY	De-/encryption Process Complete. Applies only when the block count is defined (**aes_config**[UC] = 0). Indicates the current payload has been processed and transferred from the output data FIFO. The AES cryptography engine is ready for the next payload.	R/W	0

9.3.1.5 Configuration Register

This register defines the operation mode and the payload data format for the AES cryptography engine.

aes_config **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:7	—	Reserved.	—	—
6:5	OP	Operation Mode Select. 00 Electronic codebook (ECB) 01 Cipher block chaining (CBC) 10 128-bit cipher block feedback (128-bit CFB) 11 Output feedback (OFB)	R/W	0
4	UC	Undefined Block Count. 0 Block count is prepended to the key data or to the first payload block. 1 No block count is prepended to the key data or to the first payload block. An undefined number of data blocks is processed.	R/W	0
3	RK	Reuse Key. 0 A new key is prepended to each payload. 1 A new key is not prepended to each payload. The previously used key is applied to each new payload.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
2	RPK	Replay Key. Valid only for ECB decryption with a defined block count (UC=0). 0 Do not replay the key. 1 The AES cryptography engine sends the 10th key generated during a decryption task as a 128-bit block appended to the payload.	R/W	0
1	IKG	Internal Key Generation. 0 No forward key expansion takes place to generate the 10th key. Depending on **aes_config**[RK], the AES cryptography engine expects the 10th key to be either preexisting in the local key register or prepended to the payload. 1 The 10th key is computed using forward key expansion.	R/W	0
0	ED	Encryption/decryption Select. 0 Encrypt the data. 1 Decrypt the data.	R/W	0

9.3.2 Programming Considerations

Table 9-16 defines terms used in this section.

Table 9-16. Terminology

Term	Definition
Payload	The data to be encrypted or decrypted
Data block	A set of four consecutive words in a payload
Block count	The number of data blocks in a payload
De-/encryption process	The complete de-/encryption of a payload with a given configuration and key

9.3.2.1 Configuration

Before processing a payload with the AES cryptography engine, program the desired configuration in **aes_config**. If no configuration changes are required between payloads, the preceding payload's configuration can be reused. The content of **aes_config** is not reset when processing is halted (**aes_status**[PS] = 0).

9.3.2.2 Process Start

As long as no interrupts are pending in **aes_intcause**, a de-/encryption process starts when software sets the process-start bit (**aes_status**[PS]).

9.3.2.3 Process Stop

A de-/encryption process stops under any one of the following conditions:

- An interrupt condition occurs: underflow, overflow, or process complete.
- Software clears the process-start bit (**aes_status**[PS]).
- System hardware and runtime resets

www.DataSheet4U.com

www.DataSheet4U.com

9.3.2.4 Payload Structure

The payload consists of a variable number of 4-word data blocks defined by the block count (Section "Block Count" on page 329). Each data block is 16 bytes in length, the last of which must be padded with zeros if necessary.

Block Count

When **aes_config**[UC] is cleared, the block count is prepended to the payload. The block count is the number of 4-word data blocks that make up the payload. The block count is the first word of data sent to the input data FIFO after starting the de-/encryption process. If interrupts are enabled (**aes_status**[IE] = 1), the AES cryptography engine uses the block count to determine when to issue a completion interrupt (**aes_intcause**[RDY] = 1) after all data blocks have been processed and transferred from the output data FIFO.

When **aes_config**[UC] is set, the block count is not prepended to the payload. In this case, because the payload length is undefined, the AES cryptography engine cannot issue a completion interrupt. Instead, the AES cryptography engine continues processing the payload until stopped by software.

De-/Encryption Key

When **aes_config**[RK] is cleared, the 128-bit de-/encryption key is prepended to the payload following the optional block count. The encryption key is transferred as four words in MSB first ordering. For example, if the encryption key is 0x00112233_44556677_8899AABB_CCDDEEFF, it is transferred to the input data FIFO in the following steps:

1) Write 0x00112233 to **aes_indata**.

2) Write 0x44556677 to **aes_indata**.

3) Write 0x8899AABB to **aes_indata**.

4) Write 0xCCDDEEFF to **aes_indata**.

When **aes_config**[RK] is set, the encryption key from the previous de-/encryption process is reused, and no new de-/encryption key should be prepended to the payload.

The options for setting up the encryption key are shown in Table 9-17.

Table 9-17. Encryption Key Setup Options

aes_config [RK]	Key Included with Payload	Local Key Register	Description
0	Yes	New Key	The key is prepended to the payload. After process completion, the key is stored in the local key register.
1	No	Previous Key	The key is taken from the local key register, which contains a key from a previous encryption process.

10th-Round Decryption Key

Each data block is processed in a series of ten de-/encryption rounds. Each round produces a new key based on the key from the previous round. In the case of encryption, the rounds proceed in increasing order from one to ten. In the case of decryption, rounds proceed in decreasing order from ten to one. Decryption starts with the key produced in round ten, the 10th key. The 10th key is generated before starting decryption by processing ten forward key expansion rounds. To bypass this initial generation of the 10th key, the 10th key can be read from the current decrypted payload, and then sent with the next encrypted payload during the next decryption process.

When **aes_config**[RPK] is set, the AES cryptography engine appends the 10th key to the last data block of the processed payload. This applies to ECB decryption operation mode only. The decryption key obtained by ECB can also be applied for the CBC mode of operation because the key expansion algorithms are identical. Feedback modes of operation do not need an expanded key for starting a decryption process.

When **aes_config**[IKG] is cleared, the AES cryptography engine does not perform the ten forward key expansion rounds to generate the first decryption key. Instead, it expects the 10th key from a previous decryption process to be read by software and prepended to the payload after the block count (as described in "De-/Encryption Key" on page 329). The 10th key is stored in the local key register and can be reused during subsequent decryption processes.

www.DataSheet4U.com

www.DataSheet4U.com

The options for setting up a decryption key are shown in Table 9-18.

Table 9-18. Decryption Key Setup Options

aes_config [IKG]	aes_config [RK]	Key Included with Payload	Local Key Register	Description
0	0	10th Key (First Decryption Key)	Don't Care	The 10th key is prepended to the payload. Forward key expansion is not performed. The 10th key is stored in the local key register after process completion.
0	1	Not Included	10th Key	The 10th key is in the local key register from a previously performed decryption process. Forward key expansion is not performed.
1	0	Key	Don't Care	The key is prepended to the payload. Forward key expansion is performed before decryption to generate the 10th key. The 10th key is stored in the local key register after process completion.
1	1	Not Included	Key	The key is in the local key register from a previously performed encryption process. Forward key expansion is performed before decryption to generate the 10th key. The 10th key is stored in the local key register after process completion.

Initialization Vector Sequence

When using CBC, CFB, and OFB operation modes (see the OP bit of the Section 9.3.1.5 "Configuration Register" on page 327), the 128-bit initialization vector must be prepended to the payload as a data block in MSb first format.

Note: The initialization vector attachment does not increment the block count.

9.3.2.5 Payload Transfers

Because the input and output data FIFOs are both 8 words deep (two data blocks), it is necessary to alternate between reading a block from the output data FIFO and writing a block to the input data FIFO to achieve maximum throughput. This can be accomplished by first filling the input data FIFO with two blocks, then alternating between reading a block from the output data FIFO and writing a block to the input data FIFO.

The AES cryptography engine supports both DMA transfers and programmed I/O.

DMA Transfers

For DMA transfers, the status of the input and output data FIFOs is automatically monitored by the DDMA controller. For a description of how to program the DDMA controller, see Section 4.0 on page 115. On the AES input-side, memory is used as the *source* and the input FIFO data register as *destination* when building the descriptors. On the AES output-side, the output FIFO data register is used as source and memory as destination when building the descriptors.

Programmed I/O

To prevent overflow and underflow errors during programmed I/O, software must monitor the status of the input and output data FIFOs as follows:

- Poll **aes_status**[IN] to determine when data can be written to the input data FIFO. When **aes_status**[IN] is set, at least one data block can be written to the input data FIFO. If data is written when the input data FIFO is full, a FIFO overflow interrupt is triggered and **aes_intcause**[OVR] is set.
- Poll **aes_status**[OUT] to determine when data can be read from the output data FIFO. When **aes_status**[OUT] is set, at least one data block can be read from the output data FIFO. If software attempts to read data when the output data FIFO is empty, a FIFO underflow interrupt is triggered and **aes_intcause**[UND] is set.

www.DataSheet4U.com

www.DataSheet4U.com

9.3.2.6 Example De-/Encryption Sequence

Table 9-19 shows the programming steps involved in a typical decryption or encryption process. ‘X’ denotes a value that depends on the particular de-/encryption application.

Table 9-19. Example Programming Sequence

Step #	Description	Register Access	Required Condition
Configuration Sequence			
1	Write Configuration	**aes_config**[OP] = 0xXXXXXXXX	none
Status Sequence / Process Start			
2	Write Status	**aes_status**[CR] = 00 **aes_status**[IE] = 1 **aes_status**[PS] = 1	none
Block Count Sequence (**aes_config**[UC] = 0 only) example: block count = 4			
3	Write Block Count	**aes_indata** = 0x4	none
Key Sequence (**aes_config**[RK] = 0 only) example: key = 0x0011 2233 4455 6677 8899 AABB CCDD EEFF			
4	Write De-/Encryption Key	4a) **aes_indata** = 0x0011 2233	none
		4b) **aes_indata** = 0x4455 6677	
		4c) **aes_indata** = 0x8899 AABB	
		4d) **aes_indata** = 0xCCDD EEFF	
Initialization Vector Sequence (**aes_config**[OP] = 01 or 10 or 11 only) example: IV = 0x0011 2233 4455 6677 8899 AABB CCDD EEFF			
5	Write Initialization Vector	5a) **aes_indata** = 0x0011 2233	aes_status[IN] = 1
		5b) **aes_indata** = 0x4455 6677	
		5c) **aes_indata**= 0x8899 AABB	
		5d) **aes_indata** = 0xCCDD EEFF	
Payload Sequence, n blocks example: data block = 0x0011 2233 4455 6677 8899 AABB CCDD EEFF			
6	Write Data Block 0	6a) **aes_indata** = 0x0011 2233	aes_status[IN] = 1
		6b) **aes_indata** = 0x4455 6677	
		6c) **aes_indata** = 0x8899 AABB	
		6d) **aes_indata** = 0xCCDD EEFF	
7 ... 7+ (*n*-1)	Write Data Block	a) **aes_indata** = 0x0011 2233	**aes_status**[IN] = 1
		b) **aes_indata** = 0x4455 6677	
		c) **aes_indata** = 0x8899 AABB	
		d) **aes_indata** = 0xCCDD EEFF	
	Read Data Block	a) memory[0] = **aes_outdata**	**aes_status**[OUT] = 1
		b) memory[1] = **aes_outdata**	
		c) memory[2] = **aes_outdata**	
		d) memory[3] = **aes_outdata**	

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-19. Example Programming Sequence

Step #	Description	Register Access	Required Condition
7 + *n*	Read Data Block *n*	a) memory[0] = **aes_outdata**	**aes_status**[OUT] = 1
		b) memory[1] = **aes_outdata**	
		c) memory[2] = **aes_outdata**	
		d) memory[3] = **aes_outdata**	
Status Sequence / Process Stop (**aes_config**[UC] = 1 only)			
8 + *n*	Stop the Process	**aes_status**[PS] = 0	none
Key Replay Sequence (**aes_config**[OP] = 00 AND **aes_config**[EDS] = 1 AND **aes_config**[UC] = 0 only)			
9 + *n*	Read Replayed Key	a) memory[0] = **aes_outdata**	**aes_status**[OUT] = 1
		b) memory[1] = **aes_outdata**	
		c) memory[2] = **aes_outdata**	
		d) memory[3] = **aes_outdata**	

www.DataSheet4U.com

www.DataSheet4U.c

9.4 Secure Digital (SD) Controller

The Au1210™ and Au1250™ processors have two Secure Digital (SD) controllers that incorporate both SD and SDIO interfaces. The peripheral bus clock is used as the clock reference for the SD controllers.

The SD controllers comply with version 1.1 of the SD card specification. References in this section are to that version of the specification.

9.4.1 SD Registers

Each SD controller (SD0 and SD1) has its own block of control and configuration registers with physical base addresses shown in Table 9-20. The register block consists of the registers shown in Table 9-21.

Table 9-20. SD Base Address

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

Table 9-21. SD Registers

Offset (Note1)	Register Name	Description
0x0000	sd_txport	SD Transmit Data Port Register
0x0004	sd_rxport	SD Receive Data Port Register
0x0008	sd_config	SD Configuration Register
0x000C	sd_enable	SD Enable Register
0x0010	sd_config2	SD Configuration 2 Register
0x0014	sd_blksize	SD Block Size Register
0x0018	sd_status	SD Status Register
0x001C	sd_debug	SD Debug Register
0x0020	sd_cmd	SD Command Register
0x0024	sd_cmdarg	SD Command Argument Register
0x0028	sd_resp3	SD Response 3 Register
0x002C	sd_resp2	SD Response 2 Register
0x0030	sd_resp1	SD Response 1 Register
0x0034	sd_resp0	SD Response 0 Register
0x0038	sd_timeout	SD Timeout Register

Note1. See Table 9-20 for base address.

9.4.1.1 SD Transmit Data Port Register

This register is used to send data to the SD interface for either PIO or DMA write modes. A write to **sd*n*_txport** pushes an entry into the 8-byte transmit FIFO.

sd*n*_txport **Offset = 0x0000**

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
31:8	—	Reserved.	—	—
7:0	TXD	Transmit Data.	W	0

9.4.1.2 SD Receive Data Port Register

This register is used to read data from the SD interface from either PIO or DMA read modes. A read from **sd*n*_rxport** pops an entry from the 8-byte receive FIFO.

sd*n*_rxport **Offset = 0x0004**

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

Bits	Name	Description	Read/ Write	Default
31:8	—	Reserved.	—	—
7:0	RXD	Receive Data.	R	0

9.4.1.3 SD Configuration Register

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

sd*n*_config **Offset = 0x0008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SI	CD	RF	RA	RH	TA	TE	TH		WC	RC	SC	DT	DD	RAT	CR	I	RO	RU	TO	TU	NE	DE	DIV								
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31	SI	SDIO Device Interrupt Enable.	R/W	0
30	CD	Card Insertion/removal Detect Interrupt Enable.	R/W	0
29	RF	RX Buffer Full Interrupt Enable.	R/W	0
28	RA	RX Buffer Almost Full (all but one entry) Interrupt Enable.	R/W	0
27	RH	RX Buffer At Least Half Full Interrupt Enable.	R/W	0
26	TA	TX Buffer Almost Empty Interrupt Enable.	R/W	0
25	TE	TX Buffer Empty Interrupt Enable.	R/W	0
24	TH	TX Buffer At Most Half Empty Interrupt Enable.	R/W	0
23	—	Reserved.	—	—
22	WC	Write CRC Error Interrupt Enable.	R/W	0
21	RC	Read CRC Error Interrupt Enable.	R/W	0
20	SC	Response CRC Error Interrupt Enable.	R/W	0
19	DT	Data Access Timeout Interrupt Enable (NAC).	R/W	0
18	DD	Data Transfer Done Interrupt Enable.	R/W	0
17	RAT	Command-response Response Access Timeout Interrupt Enable (NCR).	R/W	0
16	CR	Command-response Transfer Done Interrupt Enable (or command only if the command does not require a response).	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
15	I	Master Interrupt Enable. 0 Disable all SD interrupts. 1 Enable SD interrupts. Individual interrupt enables must still be used.	R/W	0
14	RO	RX FIFO Overrun Interrupt Enable.	R/W	0
13	RU	RX FIFO Underrun Interrupt Enable.	R/W	0
12	TO	TX FIFO Overrun Interrupt Enable.	R/W	0
11	TU	TX FIFO Underrun Interrupt Enable.	R/W	0
10	NE	RX FIFO Not Empty Interrupt Enable.	R/W	0
9	DE	Divider Write Enable. 0 Do not change the clock divider, regardless of value written in DIV field. 1 Change the clock divider to the value written in the DIV field.	R/W	0
8:0	DIV	Clock Divider. The SD clock is derived from the peripheral bus clock as follows: SD clock = Peripheral clock / [2 * (DIV+1)] For example, if DIV = 0, the clock is divided by 2; if DIV = 0x1FF, the clock is divided by 1024. Note: This value must be written simultaneously with DE to take effect.	R/W	0

9.4.1.4 SD Enable Register

This register contains bits to enable clocks and reset the SD interface.

sd*n*_enable **Offset = 0x000C**

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

Bits	Name	Description	Read/ Write	Default
31:2	—	Reserved.	—	—
1	R	Peripheral Reset. Clearing this bit (with the peripheral clock running) resets the entire peripheral. Set this bit for normal operation. SD reset sequence: First write 0b01 to enable/maintain the clock while resetting. Then write 0b11 to take the peripheral out of reset.	R/W	0
0	CE	Peripheral Clock Enable. Set this bit for normal operation. Clear this bit to disable the clock and conserve power.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.4.1.5 SD Configuration 2 Register

This register is used to set up the PIO or DMA mode and state machine master enable.

sd*n*_config2 **Offset = 0x0010**

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

Bits	Name	Description	Read/ Write	Default
31:10	—	Reserved.	—	—
9	RW	Read Wait Enable. When this bit is set, serial clock-based flow control is not used. This bit is valid for SDIO mode only.	R/W	0
8	WB	Wide Bus Transfer Mode. 0 One wire data transfer 1 Four wire data transfer	R/W	0
7:5	—	Reserved.	—	—
4	DC	Disable Hardware Timeout Counter. 0 Normal hardware timeout 1 No hardware timeout (software timeout)	R/W	0
3	DF	Disable Clock Freezing for Flow Control. 0 Enable clock freezing. 1 Disable clock freezing.	R/W	0
2	—	Reserved.	R/W	0
1	FF	Force FIFO Flush and Reset. This bit is sticky and must be manually cleared to resume normal operation.	R/W	0
0	EN	Serial Interface State Machine and FIFO Master Enable. This bit must be set to enable the SD controller.	R/W	0

9.4.1.6 SD Block Size Register

This register defines the size and number of blocks to be transmitted by the SD controller.

sd*n*_blksize **Offset = 0x0014**

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

Bits	Name	Description	Read/ Write	Default
31:25	—	Reserved.	—	—
24:16	BC	Block I/O Count. This field is used for a known number of SDIO read/write blocks. The number of blocks to be transferred is (BC + 1).	R/W	0
15:11	—	Reserved.	—	—
10:0	BS	Block Size in Bytes. The value programmed is one less than the actual number of bytes in the block. (Block size = BS + 1)	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.4.1.7 SD Status Register

This 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-triggered interrupt, write a 1 to the appropriate bit. A level-triggered interrupt bit is cleared when the triggering event no longer applies. This register also contains the CRC status word resulting from a block write. The **sd*n*_status** bits always reflect the current status, regardless of the corresponding **sd*n*_config** bits.

sd*n*_status **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SI	CD	RA	RF	RH	TA	TE	TH		WC	RC	SC	DT	DD	RA	CR	I	RO	RU	TO	TU	NE			D3	CF	DB	CB		DCRCW		
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Level/ Edge	Read/ Write	Default
31	SI	SDIO Device Interrupt.	LT	R/W	0
30	CD	Card Insertion/Removal Detect Interrupt.	ET	R/W	0
29	RF	RX Buffer Full Interrupt.	LT	R/W	0
28	RA	RX Buffer Almost Full (all but one entry) Interrupt.	LT	R/W	0
27	RH	RX Buffer At Least Half Full Interrupt.	LT	R/W	0
26	TA	TX Buffer Almost Empty Interrupt.	LT	R/W	0
25	TE	TX Buffer Empty Interrupt.	LT	R/W	0
24	TH	TX Buffer At Most Half Empty Interrupt.	LT	R/W	0
23	—	Reserved.	—	—	—
22	WC	Write CRC Error Interrupt.	ET	R/W	0
21	RC	Read CRC Error Interrupt.	ET	R/W	0
20	SC	Response CRC Error Interrupt.	ET	R/W	0
19	DT	Data Access Timeout Interrupt (NAC).	ET	R/W	0
18	DD	Data Transfer Done Interrupt.	ET	R/W	0
17	RAT	Command-response Response Access Timeout Interrupt (NCR).	ET	R/W	0
16	CR	Command-response Transfer Done Interrupt (or command only if the command does not require a response).	ET	R/W	0
15	I	Master Interrupt. Reads a 1 when any unmasked (enabled) interrupt is taken. This bit is cleared automatically once all unmasked interrupts are cleared.	—	R	0
14	RO	RX FIFO Overrun Interrupt.	ET	R/W	0
13	RU	RX FIFO Underrun Interrupt.	ET	R/W	0
12	TO	TX FIFO Overrun Interrupt.	ET	R/W	0
11	TU	TX FIFO Underrun Interrupt.	ET	R/W	0
10	NE	RX FIFO Not Empty Interrupt.	LT	R/W	0
9:8	—	Reserved.	—	—	—
7	D3	Real-time direct sample of SD*n*_DAT[3] provided for software debouncing.	—	R/W	0
6	CF	Clock Freezing Status. 0 Clock frozen (potential overrun/underrun) 1 Normal clocking	—	R	0
5	DB	SD Data-response Busy Status.	—	R	0
4	CB	SD Command-response Busy Status.	—	R	0
3	—	Reserved.	—	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Level/ Edge	Read/ Write	Default
2:0	DCRCW	Device CRC Status Word. 010 No error 101 Transmission error 111 No CRC response All other bit combinations are undefined.	—	R	0

9.4.1.8 SD Debug Register

This register is used for read only access to the read and write pointers for both transmit and receive FIFOs. The pointers contain the entry number for the active FIFO entry.

sd*n*_debug **Offset = 0x001C**

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

Bits	Name	Description	Read/ Write	Default
31:15	—	Reserved.	—	—
14:12	RXR	Receive FIFO Read Pointer.	R	0
11	—	Reserved.	—	—
10:8	RXW	Receive FIFO Write Pointer.	R	0
7	—	Reserved.	—	—
6:4	TXR	Transmit FIFO Read Pointer.	R	0
3	—	Reserved.	—	—
2:0	TXW	Transmit FIFO Write Pointer.	R	0

9.4.1.9 SD Command Register

This register contains fields used to build an SD command sequence.

sd*n*_cmd **Offset = 0x0020**

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

Bits	Name	Description	Read/ Write	Default
31:24	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
23:16	RT	Response Type. 0x00 No response 0x01 R1 response (48 bits) 0x02 R2 response (136 bits) 0x03 R3 response (48 bits) 0x04 R4 response (48 bits)—SDIO only 0x05 R5 response (48 bits)—SDIO only 0x06 R6 response (48 bits) 0x81 R1b response (48 bits) All other values are reserved.	R/W	0
15:8	CI	Command Index. See SD specification for command listing.	R/W	0
7:4	CT	Command Type. Must be written with each command. See Table 9-22 for valid encoding and descriptions.	R/W	0
3:2	—	Reserved.	—	—
1	RY	Response Ready. This bit is set by the SD block once the command-response sequence is finished and automatically cleared once **sd*n*_resp0** is read.	R	0
0	GO	Command Go/Busy. This bit is set to initiate a command. The bit is automatically cleared once the last bit of the command argument is transmitted.	R/W	0

Table 9-22. Command Type Field Encodings

CT[3:0]	Action applied to SD Memory	Action applied to SDIO
0000	Non-data-write, non-data-read, non-data-stop, non-io-abort commands.	Non-data-write, non-data-read, non-data-stop, non-io-abort commands.
0001	Single block write. Use when doing a WRITE_BLOCK (CMD24) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is also programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored.	Single block I/O write. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 0 (byte mode). The block size is defined in Byte/Block Count. A 0x0 value in Byte/Block Count is considered to be 256 bytes (see p.18 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. **sd_blksize**[BC] is ignored.
0010	Single block read. Use when doing a READ_SINGLE_BLOCK (CMD17) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is also programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored.	Single block I/O read. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 0 (byte mode). the block size is defined in Byte/Block Count. A 0x0 value in Byte/Block Count is considered to be 256 bytes (see p.18 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored.

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-22. Command Type Field Encodings

CT[3:0]	Action applied to SD Memory	Action applied to SDIO
0011	Multiple block write requiring STOP command to end transfer. Use when doing a WRITE_MULTIPLE_BLOCK (CMD25) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored. The transfer has to be terminated by issuing a STOP_TRANSMISSION (CMD12) command.	Multiple block I/O write requiring writing to CCCR to end transfer. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 1 (block mode) and Byte/Block Count = 0x0 (infinite block count) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored. The transfer has to be terminated by issuing a IO_RW_DIRECT (CMD52) command to write to the abort register in CCCR (bits [2:0] of register 6) (see p.23 of SDIO spec).
0100	Multiple block read requiring STOP command to end transfer. Use when doing a READ_MULTIPLE_BLOCK (CMD18) command. Block size is defined in CSD or programmed by SET_BLOCKLEN (CMD16) command (see p.41 of SD spec) and is programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored. The transfer has to be terminated by issuing a STOP_TRANSMISSION (CMD12) command.	Multiple block I/O read requiring writing to CCCR to end transfer. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 1 (block mode) and Byte/Block Count = 0x0 (infinite block count) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. **sd*n*_blksize**[BC] is ignored. The transfer has to be terminated by issuing a IO_RW_DIRECT (CMD52) command to write to the abort register in CCCR (bits [2:0] of register 6) (see p.23 of SDIO spec).
0101	Not applicable	Multiple block I/O write with fixed number of blocks. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 1 (direction is write) and Block Mode = 1 (block mode) and Byte/Block Count set to the desired number of blocks to transfer (must be non-zero) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. Based on the Byte/Block Count, **sd*n*_blksize**[BC] is programmed with the correct number of blocks. Using either 1-bit or 4-bit wire will not affect this number because in the 4-bit wire case, 1 block of data is split into 4 sub-blocks (each 1/4 of the original block size) on each data wire. The start and stop bits still define the boundary of a block. The transfer will be terminated when the correct number of blocks have been transmitted. No abort action is required.

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-22. Command Type Field Encodings

CT[3:0]	Action applied to SD Memory	Action applied to SDIO
0110	Not applicable	Multiple block I/O read with fixed number of blocks. Use when doing an IO_RW_EXTENDED (CMD53) with fields R/W Flag = 0 (direction is read) and Block Mode = 1 (block mode) and Byte/ Block Count set to the desired number of blocks to transfer (must be non-zero) (see p.18 of SDIO spec). For function 0, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to FN0 Block Size registers (2 of them) inside CCCR (see p.26 of SDIO spec). For functions 1 to 7, block size is programmed by using the IO_RW_DIRECT (CMD52) command to write to the I/O Block Size registers (2 of them) inside FBR (see p.28 of SDIO spec). The block size is also programmed in **sd*n*_blksize**[BS]. Based on the Byte/Block Count, **sd*n*_blksize**[BC] is programmed with the correct number of blocks. Using either 1-bit or 4-bit wire will not affect this number because in the 4-bit wire case, 1 block of data is split into 4 sub-blocks (each 1/4 of the original block size) on each data wire. The start and stop bits still define the boundary of a block. The transfer will be terminated when the correct number of blocks have been transmitted. No abort action is required.
0111	Terminate transfer of a multiple block write or read. Use when doing a STOP_TRANSMISSION (CMD12) command (see p.41 of SD spec).	Not applicable.
1000	Not applicable	Terminate transfer of a multiple block I/O 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).
1001	Not applicable	Suspend current data transfer. Use when issuing an IO_RW_DIREC (CMD52) command to set BR=1 (see pp.37-39 of SDIO spec). The host needs to check the response for BS=0 to know that the data transfer is suspended. The host is free to do other commands. To resume, the host has to issue the command again. For example, if the host suspends a multiple write with undefined number of blocks, the same command has to issue again for the “resume”. If the host suspends a multiple read with defined number of blocks, the host has to find out the number of remaining blocks of data yet to be transferred and set it up in the multiple read command for the “resume”. A suspension can be done only at a block boundary when (i) in a read wait state waiting for the start bit of the next block when the device will hold off data when receiving the suspend command; (ii) during the write busy state when the device is sending logic low on data bit 0.
1010 to 1111	Reserved	Reserved

www.DataSheet4U.com

www.DataSheet4U.com

9.4.1.10 SD Command Argument Register

This register holds the argument used in an SD command-response sequence.

sd*n*_cmdarg **Offset = 0x0024**

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

Bits	Name	Description	Read/ Write	Default
31:0	CARG	Command Argument. Must write this register first, then write **sd*n*_cmd**[BY] to issue the command.	R/W	0

9.4.1.11 SD Response 3 Register

This register contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits.

sd*n*_resp3 **Offset = 0x0028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESP3																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31:0	RESP3	Bits [127:96] of Response From Device.	R	0

9.4.1.12 SD Response 2 Register

This register contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits.

sd*n*_resp2 **Offset = 0x002C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESP2																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31:0	RESP2	Bits [95:64] of Response From Device.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

9.4.1.13 SD Response 1 Register

This register contains the response from an issued command-response sequence. Valid only when the expected response length is 128 bits or (6 + 32) bits.

sd*n*_resp1 **Offset = 0x0030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESP1																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31:0	RESP1	Bits [63:32] of Response From Device.	R	0

9.4.1.14 SD Response 0 Register

This register contains the response from an issued command-response sequence.Valid for all the expected response lengths.

sd*n*_resp0 **Offset = 0x0034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESP0																															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31:0	RESP0	Bits [31:0] of Response From Device.	R	0

9.4.1.15 SD Timeout Register

This register defines the timeout value for NAC.

sd*n*_timeout **Offset = 0x0038**

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

Bits	Name	Description	Read/ Write	Default
31:21	—	Reserved.	—	—
20:0	TMAX	Maximum Timeout Value for NAC where: NAC = TAAC + NSAC (See the SD specification.) If using a 25MHz clock, the maximum timeout value is 81.02ms.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

9.4.2 Hardware Considerations

The SD interface consists of the signals listed in Table 9-23.

Table 9-23. SD Signals

Pin Name	Input/Output	Definition
SD0_CLK	O	SD Card 0 Interface Clock
SD0_CMD	I/O	SD Card 0 Half Duplex Command and Response
SD0_DAT[3:0]	I/O	SD Card 0 Data Bus
SD1_CLK	O	SD Card 1 Interface Clock
SD1_CMD	I/O	SD Card 1 Half Duplex Command and Response
SD1_DAT[3:0]	I/O	SD Card 1 Data Bus

www.DataSheet4U.com

www.DataSheet4U.com

9.5 UART Interfaces

The Au1210™ and Au1250™ processors contain two 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
- Capable of speeds up to 1.5Mbps to enable connections with Bluetooth and other peripherals through a UART interface
- Similar to personal computer industry standard 16550 UART
- Full modem flow control signals brought out on UART1
- Open-drain Tx mode allowing single-pin bidirectional operation

9.5.1 Programming Model

Each UART is controlled by a register block. Table 9-24 lists the base address for each UART register block.

Table 9-24. 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

UART1 has the full set of data and modem/flow control signals pinned out on *external* signals. UART0 has only external RXD and TXD.

The UARTs can use DMA transfers. See Section 4.0 "Descriptor-Based DMA (DDMA) Controller" on page 115 for more information.

9.5.1.1 UART Registers

Each register block contains the registers listed in Table 9-25.

Table 9-25. UART Registers

Offset (Note1)	Register Name	Description
0x0000	uart_rxdata	Receive Data FIFO Register
0x0004	uart_txdata	Transmit Data FIFO Register
0x0008	uart_inten	Interrupt Enable Register
0x000C	uart_intcause	Interrupt Cause Register
0x0010	uart_fifoctrl	FIFO Control Register
0x0014	uart_linectrl	Line Control Register
0x0018	uart_mdmctrl	Modem Control Register
0x001C	uart_linestat	Line Status Register
0x0020	uart_mdmstat	Modem Status Register
0x0024	uart_autoflow	Automatic Hardware Flow Control Register
0x0028	uart_clkdiv	Clock Divider Register

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-25. UART Registers (Continued)

Offset (Note1)	Register Name	Description
0x0100	uart_enable	UART Enable Register
0x0104	uart_mdmden	Modem Control Input Disable Register
0x0108	uart_bidir	Bidirectional UART Control Register

Note1. See Table 9-24 on page 345 for base address.

Receive Data FIFO Register

This register contains receive data. If FIFOs are enabled (**uart_fifoctrl**[FE]=1), a read from this register pops the next entry from the receive data FIFO. If enabled, the receive FIFO is 16 entries deep. This register is read only.

uart_rxdata **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
																									RXDATA							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7:0	RXDATA	Receive Data.	R	0

Transmit Data FIFO Register

This register provides access to the transmit data. If FIFOs are enabled (**uart_fifoctrl**[FE]=1), a write to this register pushes an entry into the transmit data FIFO. If enabled, the transmit FIFO is 16 entries deep. This register is write only.

uart_txdata **Offset = 0x0004**

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

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7:0	TXDATA	Transmit Data.	W	0

Interrupt Enable Register

This register contains bits that enable interrupts under certain operational conditions; see Table 9-26 on page 348.

uart_inten **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
																													MIE	LIE	TIE	RIE
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

www.DataSheet4U.com

www.DataSh

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3	MIE	Modem Status Interrupt Enable. When the MIE bit is set an interrupt is generated when changes occur in the state of the modem control signals. System Note: For systems that use the UART1 interface but do *not* use the modem control signals, the modem status interrupts must be disabled (MIE=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	R/W	0
2	LIE	Line Status Interrupt Enable. When the LIE bit is set an interrupt is generated when errors (overrun, framing, stop bits) or break conditions occur.	R/W	0
1	TIE	Transmit Buffer Available Interrupt Enable. When the TIE bit is set an interrupt is generated when the transmit data register is empty, or the transmit data FIFO reaches threshold (**uart_linestat**[TT]).	R/W	0
0	RIE	Receive Data Available Interrupt Enable. When the RIE bit is set an interrupt is generated on receive data ready (**uart_linestat**[DR]) when the receive data FIFO reaches threshold or on a character time-out.	R/W	0

Interrupt Cause Register

This register contains information about the cause of the current interrupt.

uart_intcause **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3:1	IID	Interrupt Identifier. The IID field identifies the highest priority current interrupt condition. Table 9-26 on page 348 lists the priorities and encodings of each interrupt condition.	R	0
0	IP	No Interrupt Pending. 0 An interrupt is pending. 1 No interrupts are pending.	R	1

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-26 contains information about the interrupt cause encoding.

Table 9-26. Interrupt Cause Encoding

IID	Priority	Type	Source	uart_inten Enable
0	5 (lowest)	Modem Status	Indicates a change in the external modem control signals has occurred—**uart_mdmstat**[DD, TRI, DR, DC]	MIE
1	4	Transmit Buffer Available	If FIFOs are enabled, this interrupt is generated when the number of bytes in the transmit FIFO falls to the threshold programmed in **uart_fifoctrl**[TFT]. If FIFOs are disabled, this interrupt is generated when the transmit data register becomes empty.	TIE
2	3	Receive Data Available	If FIFOs are enabled, this interrupt is generated when the number of bytes in the receive FIFO rises to the threshold programmed in (**uart_fifoctrl**[RFT]). If FIFOs are disabled, this interrupt is generated when the receive data register contains valid data.	RIE
3	1 (highest)	Receive Line Status	Indicates a receive error has occurred—**uart_linestat**[OE, PE, FE, BI]	LIE
4	—	Reserved	—	—
5	—	Reserved	—	—
6	2	Character Time-out	Character has been in receive FIFO for 0x300 UART clocks (in **uart_clkdiv**).	RIE
7	—	Reserved	—	—

FIFO Control Register

This register provides control of character buffering options.

uart_fifoctrl **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7:6	RFT	Receive FIFO Threshold. A receive threshold interrupt (IID=2) is generated when the number of bytes in the receive FIFO rises to the trigger depth listed below: 00 Trigger depth = 1 01 Trigger depth = 4 10 Trigger depth = 8 11 Trigger depth = 14 If using DMA, the mode-select bit (**uart_fifoctrl**[MS]) must be set, and the receive FIFO threshold must match the DMA transfer size.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
5:4	TFT	Transmit FIFO Threshold. A transmit threshold interrupt (IID=1) is generated when the number of bytes in the transmit FIFO falls to the trigger depth shown below: 00 Trigger depth = 0 01 Trigger depth = 4 10 Trigger depth = 8 11 Trigger depth = 12 If using DMA, the mode-select bit (**uart_fifoctrl**[MS]) must be set, and the transmit FIFO threshold must allow space for the DMA transfer size.	R/W	0
3	MS	Mode Select for DMA Transfers. 0 Byte mode. The DDMA controller uses the transmit-empty and data-ready status bits (**uart_linestat**[TE, DR]) to trigger transfers. 1 FIFO threshold mode. The DDMA controller uses the FIFO threshold values to trigger transfers.	R/W	0
2	TR	Transmitter Reset. 0 Normal operation. 1 Clear the transmit FIFO and reset the transmitter. The transmit shift register is not cleared.	R/W	0
1	RR	Receiver Reset. 0 Normal operation. 1 Clear the receive FIFO and reset the receiver. The receive shift register is not cleared.	R/W	0
0	FE	FIFO Enable. 0 Disable the transmit and receive FIFOs. Both FIFOs have an effective depth of 1 byte. 1 Enable the 16-byte transmit and receive FIFOs.	R/W	0

Line Control Register

This register provides control over the data format and parity options.

uart_linectrl **Offset = 0x0014**

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

Bits	Name	Description	R/W	Default
31:7	—	Reserved.	—	—
6	SB	Send Break. 0 Normal operation. 1 Force the transmitter output to zero.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
5:4	PAR	Parity Select. Selects the parity encoding for the transmitter and receiver. 00 Odd parity 01 Even parity 10 Mark parity. Parity always 1. 11 Zero parity. Parity always 0.	R/W	0
3	PE	Parity Enable. 0 Disable parity. Parity is not sent or expected. 1 Enable parity. Parity is selected according to the PAR field.	R/W	0
2	ST	Stop Bits Select. 0 Selects 1 stop bit for all character lengths. 1 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. Selects the number of data bits in each character. 00 5 bits 01 6 bits 10 7 bits 11 8 bits	R/W	0

Modem Control Register

This register allows the state of the output modem control signals to be set.

UART1 has the full set of data and modem/flow control signals pinned out on external signals. UART0 has only external RXD and TXD.

uart_mdmctrl **Offset = 0x0018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												LB	I1	I0	RT	DT
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:5	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>R/W</th><th>Default</th></tr>
<tr><td>4</td><td>LB</td><td>Loopback.
0 No loopback (normal operation)
1 Enable loopback for self-test. Establish the internal connections shown below:
<table>
<tr><th>Output Signal</th><th>Looped Back To</th></tr>
<tr><td>TXD</td><td>RXD</td></tr>
<tr><td>DTR#</td><td>DSR#</td></tr>
<tr><td>RTS#</td><td>CTS#</td></tr>
<tr><td>I0#</td><td>RI#</td></tr>
<tr><td>I1#</td><td>DCD#</td></tr>
</table>
When internal loopback is enabled, autoflow (<b>uart_autoflow</b>[AE]) still controls transmission of data from <b>uart_txdata</b> based on the external CTS# signal state.</td><td>R/W</td><td>0</td></tr>
<tr><td>3</td><td>I1</td><td>Internal Line 1 State.
0 Drive the internal I1# signal high.
1 Drive the internal I1 #signal low.
This is used in loopback mode.</td><td>R/W</td><td>0</td></tr>
<tr><td>2</td><td>I0</td><td>Internal Line 0 State.
0 Drive the internal I0# signal high.
1 Drive the internal I0# signal low.
This is used in loopback mode.</td><td>R/W</td><td>0</td></tr>
<tr><td>1</td><td>RT</td><td>Request to Send.
0 Drive RTS# high.
1 Drive RTS# low.
This bit has no effect if <b>uart_autoflow</b>[AE] is set.</td><td>R/W</td><td>0</td></tr>
<tr><td>0</td><td>DT</td><td>Data Terminal Ready.
0 Drive DTR# high.
1 Drive DTR# low.</td><td>R/W</td><td>0</td></tr>
</table>

www.DataSheet4U.com

w w w . D a t a S h e

Line Status Register

This register reflects the state of the interface. The status bits in this register are set when the listed condition occurs and cleared when this register is read.

UART1 has the full set of data and modem/flow control signals pinned out on external signals. UART0 has only external RXD and TXD.

uart_linestat **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	1	1	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	—	—
7	RF	Receive FIFO Contains Error. 0 Normal condition. 1 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. 0 Normal condition. 1 The transmit shift register is empty and there are no more characters in the transmit FIFO.	R	1
5	TT	Transmit Threshold. 0 Normal condition. 1 The transmitter FIFO depth is less than or equal to the value programmed in **uart_fifoctrl**[TFT]. If the FIFOs are disabled, this bit is set when the transmitter data register is empty.	R	1
4	BI	Break Indication. 0 Normal condition. 1 A break indication has been received. When a break is detected a single zero character is received. The BI bit is valid when the zero character is at the top of the receive FIFO. This bit must be cleared with a read to **uart_linestat** before more characters are received.	R	0
3	FE	Framing Error. 0 Normal condition. 1 A valid stop bit has not been detected. FE reflects the state of the character at the top of the receive FIFO.	R	0
2	PE	Parity Error. 0 Normal condition. 1 The character at the top of the receive FIFO contains a parity error.	R	0
1	OE	Overflow Error. 0 Normal condition. 1 Receive FIFO overflow condition.	R	0
0	DR	Data Ready. 0 Receive FIFO contains invalid characters. 1 Receive FIFO contains valid characters.	R	0

www.DataSheet4U.com

w w w . D a t a S h

Modem Status Register

This register reflects the state of the external modem signals. Reading this register clears any delta indications and the corresponding interrupt.

UART1 has the full set of data and modem/flow control signals pinned out on external signals. UART0 has only external RXD and TXD.

uart0_mdmstat **Offset = 0x0020**

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

uart1_mdmstat **Offset = 0x0020**

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

Bits	Name	Description	R/W	Default
31:8	—	Reserved.	R	0
7	CD	Data Carrier Detect. Reflects the status of the external DCD# signal. For UART0, this bit always reads 1.	R	1 for UART0; 0 for UART1
6	RI	Ring Indication. Reflects the status of the external RI# signal. For UART0, this bit always reads 1.	R	1 for UART0; 0 for UART1
5	DS	Data Set Ready. Reflects the status of the external DSR# signal. For UART0, this bit always reads 1.	R	1 for UART0; 0 for UART1
4	CT	Clear to Send. Reflects the status of the external CTS# signal. For UART0, this bit always reads 1.	R	1 for UART0; 0 for UART1
3	DD	Delta DCD#. Set when a change occurs in the state of the external DCD# signal.	R	0
2	TRI	Terminate Ring Indication. Set when a positive edge occurs in the state of the external RI# signal.	R	0
1	DR	Delta DSR#. Set when a change occurs in the state of the external DSR# signal.	R	0
0	DC	Delta CTS#. Set when a change occurs in the state of the external CTS# signal.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Automatic Hardware Flow Control Register

This register controls automatic hardware flow control using modem control signals CTS# and RTS#. Upon enabling this mode, internal logic controls the output signal RTS# based upon the data register state and threshold levels. The internal logic asserts RTS# (low) to request data until the internal receive FIFO reaches its preset threshold. In this mode RTS# cannot be controlled with the **uart_mdmctrl**[RT] bit. The input signal CTS controls the transmission of data by loading the transmit shift register from the data register only while CTS# is asserted (low). Once the transmit shift register is loaded with data, it sends the entire character regardless of the CTS# signal state.

UART1 has the full set of data and modem/flow control signals pinned out on external signals. UART0 has only external RXD and TXD, and autoflow cannot be used.

uart_autoflow **Offset = 0x0024**

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

Bits	Name	Description	R/W	Default
31:1	—	Reserved.	R	0
0	AE	Autoflow Enable. 0 Disable hardware flow control. The signals are controlled/ monitored in software. 1 Enable hardware flow control—overrides software control of the signals.	R/W	0

Clock Divider Register

This register contains the divider used to generate the baud rate clock. The input to the UART clock divider is the internal peripheral bus clock, which is derived from the system bus clock; see Figure 10-1 on Page 366.

uart_clkdiv **Offset = 0x0028**

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

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	—	—
15:0	CLKDIV	Clock Divider. The baud rate of the interface is computed as follows: Baud rate = CPU / (SD * 2 * CLKDIV * 16) CPU: CPU clock SD: System bus divider See Section 10.4 "Power Management" on page 387 for information on changing SD.	R/W	0x1

www.DataSheet4U.com

www.DataSheet4U.com

UART Enable Register

This register controls reset and clock enable to the UART. The method for bringing the UART 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 **Offset = 0x0100**

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

Bits	Name	Description	R/W	Default
31:2	—	Reserved.	R	0
1	E	Enable. When the E bit is clear the entire module is held in reset. After enabling clocks, set this bit to enable normal operation.	R/W	0
0	CE	Clock Enable. When the CE bit is clear the module clock source is inhibited. This can be used to place the module in a low power stand-by state. The CE bit should be set before the module is enabled for proper bringup.	R/W	0

Modem Control Input Disable Register

This register is used to manually disable the modem control signal inputs. Writing a 1 to a bit in this register causes the corresponding input signal to be internally grounded and to always read 0 despite any activity on other signals muxed with the input.

uart_mdmden **Offset = 0x0104**

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
																													DRI	DDSR	DDCD	DCTS
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3	DRI	Disable RI# Input. 0 RI# enabled. 1 RI# driven to 0.	R/W	0
2	DDSR	Disable DSR# Input. 0 DSR# enabled. 1 DSR# driven to 0.	R/W	0
1	DDCD	Disable DCD# Input. 0 DCD# enabled. 1 DCD# driven to 0.	R/W	0
0	DCTS	Disable CTS# Input. 0 CTS# enabled. 1 CTS# driven to 0.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bidirectional UART Control Register

This register enables open-drain output for UART Tx signals, allowing a single-pin bidirectional UART interface. See Section 9.5.2.1 "Bidirectional UART Mode" on page 356.

uart_bidir **Offset = 0x0108**

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
																														GT	OD	GE
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:3	—	Reserved.	—	—
2	GT	Gate Output. When **uart_bidir**[GE] = 1, gates Tx output. 0 U*n*TXD is tristated, regardless of Tx data. 1 U*n*TXD driven by Tx data.	R/W	0
1	OD	Open-drain Output Enable. When set, U*n*TXD acts as an open-drain output: An output of 0 is driven 0, and an output of 1 is tristated.	R/W	0
0	GE	Gate Enable. Enables the functionality of **uart_bidir**[GT]. 0 **uart_bidir**[GT] is ignored. 1 **uart_bidir**[GT] gates Tx output.	R/W	0

9.5.2 Hardware Considerations

9.5.2.1 Bidirectional UART Mode

When the UART is operating in open-drain output mode (**uart_bidir**[OD] = 1), an external pull-up is required to drive a value of logic 1. A single-pin bidirectional interface can be implemented by connecting U*n*TXD and U*n*RXD together as shown in Figure 9-14.

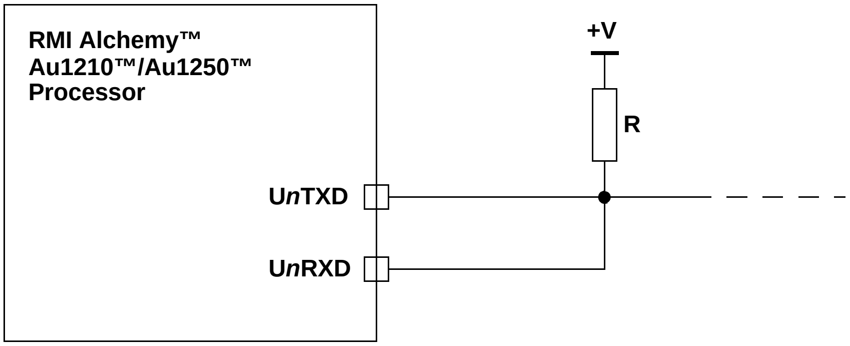


Figure 9-14. Bidirectional UART Interface

www.DataSheet4U.com

www.DataSheet4U.com

9.5.3 UART Signals

The UART signals are listed in Table 9-27. For changing pin functionality please refer to the **sys_pinfunc** register in Section 10.3 "Primary General Purpose I/O and Pin Functionality" on page 381.

Table 9-27. UART Signals

Signal	Input/Output	Definition
UART0		
U0TXD	O	UART0 transmit. Muxed with GPIO[27]. GPIO[27] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U0RXD	I	UART0 receive. Muxed with GPIO[29]. U0RXD is the default signal coming out of hardware reset, runtime reset, and Sleep.
UART1		
U1TXD	O	UART1 transmit. Muxed with GPIO[15]. GPIO[15] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U1RXD	I	UART1 receive. Muxed with GPIO[30]. U1RXD is the default signal coming out of hardware reset, runtime reset, and Sleep.
U1CTS#	I	Clear to send. Muxed with GPIO[9]. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.
U1DSR#	I	Data set ready. Muxed with GPIO[10]. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.
U1DCD#	I	Data carrier detect. Muxed with LCD_PWM0. U1DCD# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.
U1RI#	I	Ring indication. Muxed with LCD_PWM1. U1RI# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.

www.DataSheet4U.com

www.DataSheet4U.com

Table 9-27. UART Signals (Continued)

Signal	Input/Output	Definition
U1RTS#	O	Request to send. Muxed with GPIO[13]. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.
U1DTR#	O	Data terminal ready. Muxed with GPIO[14]. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.

www.DataSheet4U.com

www.DataSheet4U.com

9.6 Software Counter

In addition to the RTC and TOY clocks, the Au1210™ and Au1250™ processors contain a 26-bit software counter for clock-based measurement and counter-match interrupt functionality. The software counter is a general-purpose timer that can be used for such applications as a system time clock for audio/video synchronization in MPEG decoding.

The software counter is driven by the peripheral bus clock, and continues running and generating match interrupts during Sleep.

9.7 Software Counter Registers

Table 9-28 lists the base address for the software counter control registers. The software counter control registers are listed in Table 9-29.

Table 9-28. Software Counter Register Base Address

Name	Physical Base Address	KSEG1 Base Address
swcnt_base	0x0 1110 010C	0xB110 010C

Table 9-29. Software Counter Registers

Offset (Note1)	Register Name	Description
0x0000	swcnt_control	Software Counter Control Register
0x0004	swcnt_count	Software Counter Register
0x0008	swcnt_match	Software Counter Match Register
0x000C	swcnt_intclr	Software Counter Interrupt Clear Register

Note1. See Table 9-28 for base address.

9.7.1 Software Counter Control Register

This register enables the software counter and match interrupts.

swcnt_control **Offset = 0x0000**

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

Bits	Name	Description	Read/Write	Default
31:2	—	Reserved.	—	—
1	IE	Interrupt Enable. 0 Disable interrupt requests. 1 Enable interrupt requests.	R/W	0
0	EN	Counter Enable. 0 Disable the counter. Stop counting. 1 Enable the counter. Resume counting from last count value.	R/W	0

www.DataSheet4U.com

www.DataS

9.7.2 Software Counter Register

This register is the free-running value of the count. It can be programmed to any new value by writing that value to the register regardless of the state of **swcnt_control**[EN]. Once the counter reaches its maximum value (0x03FF FFFF), the value resets to zero on the next count.

swcnt_count **Offset = 0x0004**

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

Bits	Name	Description	Read/Write	Default
31:26	—	Reserved.	—	—
25:0	COUNT	Software Counter Value.	R/W	0

9.7.3 Software Counter Match Register

This register contains the match value for the counter. If interrupt requests are enabled (**swcnt_control**[IE] = 1), the software counter generates a request when the match value equals the value of the count register. The match value can be programmed regardless of the state of **swcnt_control**[EN].

swcnt_match **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
							MATCH																									
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:26	—	Reserved.	—	—
25:0	MATCH	Software Counter Match Value.	R/W	0

9.7.4 Software Counter Interrupt Clear Register

This register is a write only register used by software to acknowledge an interrupt. Once the interrupt has been serviced, write a 1 to **swcnt_intclr**[INT] to clear the request; writing a 0 has no effect.

swcnt_intclr **Offset = 0x000C**

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

Bits	Name	Description	Read/Write	Default
31:1	—	Reserved.	—	—
0	INT	Interrupt Clear. 0 No effect 1 Clear the interrupt.	W	—

www.DataSheet4U.com

www.DataSheet4U.com

9.7.5 Programming Considerations

The software counter increments on each peripheral bus clock cycle when enabled (**swcnt_control**[EN] = 1). When the software counter is disabled (**swcnt_control**[EN] = 0), the counter halts and retains its value. The next time the software counter is enabled, it resumes counting from the retained value unless a new count value has been programmed in **swcnt_count**.

The software counter match interrupt must be enabled in both **swcnt_control** and in the interrupt controller for interrupts to be generated; see Section 5.0 "Interrupt Controller" on page 143. The interrupt handler must service the interrupt and clear the request in **swcnt_intclr**[INT] within the time taken for the counter to reach its maximum value and loop back to the match value because there is no mechanism for counting multiple interrupt requests.

www.DataSheet4U.com

www.DataSheet4U.c

9.8 Secondary General Purpose I/O

The Au1210™ and Au1250™ processors contain two GPIO blocks (primary and secondary). This section describes the programming model of the *secondary* GPIO block that corresponds to signals labeled GPIO[215:200]. (For a description of the primary GPIO block, see Section 10.3 "Primary General Purpose I/O and Pin Functionality" on page 381 in the system control block description.)

GPIO[200] is configured at reset to output a low logic level. All other secondary GPIOs are configured as inputs on reset.

9.8.1 GPIO2 Registers

The secondary GPIO (GPIO2) logic block is controlled by a register set referenced from the base address shown in Table 9-30. The GPIO2 registers are shown in Table 9-31

Table 9-30. GPIO2 Register Base Addresses

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

Table 9-31. GPIO2 Registers

Offset (Note1)	Register Name	Description
0x0000	gpio2_dir	Direction Register
0x0004	—	Reserved
0x0008	gpio2_output	Data Output Register
0x000C	gpio2_pinstate	Pin State Register
0x0010	gpio2_inten	Interrupt Enable Register
0x0014	gpio2_enable	Enable Register

Note1. See Table 9-30 for base address.

9.8.1.1 Direction Register

This register controls the direction of each GPIO2 signal. This register controls only 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.

An open drain can be emulated on a GPIO2 pin as follows:

- Add a pull-up resistor to the GPIO2 pin.
- Clear the corresponding data bit in the **gpio2_output** register.
- Use **gpio2_dir** to configure the GPIO2 signal as an input to allow the signal to be pulled high and as an output to drive the signal low.

gpio2_dir **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
																	DIR															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	R	0
15:0	DIR	Direction Control. Each bit controls the I/O direction of one GPIO signal in the secondary block. Bits 15:0 correspond to GPIO[215:200]. 0 Pin is an input (output disabled). 1 Pin is an output. The GPIO[200] default direction is *out*.	R/W	0x0001

9.8.1.2 Data Output Register

This register controls the output data for the GPIO2 signals. Data bits [15:0] are output to the corresponding GPIO2 signal when the corresponding enable bit is set during a write to this register. For example, to output a 1 on GPIO[200] and a 0 on GPIO[201] without changing the output of any other GPIOs, write the value 0x00030001 to **gpio2_output**.

gpio2_output **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
	ENA																DATA															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:16	ENA[15:0]	Data Output Write Enable. ENA[15:0] corresponds to DATA[15:0]. ENA is write only and should be ignored on reads. 0 Disable modifications to corresponding bit in DATA[15:0]. 1 Enable modifications to corresponding bit in DATA[15:0].	W	0
15:0	DATA[15:0]	Output Data. DATA[15:0] corresponds to GPIO[215:200]. The DATA bit values are reflected in the corresponding GPIO output signal value. When modifying a bit in DATA[15:0], the corresponding bit in ENA[15:0] must be set at the same time to allow the write. This mechanism allows individual data bits to be modified without affecting DATA[15:0] as a whole.	R/W	0

9.8.1.3 Pin State Register

This register reflects the current value of the corresponding GPIO2 signal.

gpio2_pinstate **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	R	0
15:0	DATA[15:0]	Current Signal Value for GPIO[215:200].	R	0

www.DataSheet4U.com

www.DataSheet4U.com

9.8.1.4 Interrupt Enable Register

This register contains bits that enable interrupts under certain operational conditions. The **gpio2_inten** register applies only to interrupts on GPIO[215:208]. The GPIO[215:208] signals are OR'd together to create *one* interrupt source (source number 28 on interrupt controller 0), as shown in Figure 9-15. (The GPIO[207:200] signals can be used as *independent* interrupt sources.) See Section 5.0 "Interrupt Controller" on page 143 for more information on how to program interrupts.

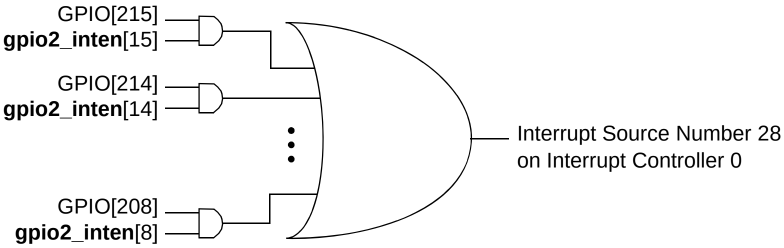


Figure 9-15. Logic for Interrupt Source Number 28 on Interrupt Controller 0

gpio2_inten **Offset = 0x0010**

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

Bits	Name	Description	R/W	Default
31:16	—	Reserved.	R	0
15:8	EN[15:8]	Interrupt Enable Bits [15:8] Correspond to GPIO[215:208]. Setting a bit enables the signal's OR'd contribution to interrupt source number 28 on interrupt controller 0.	R/W	0
7:0	—	Reserved.	R	0

9.8.1.5 Enable Register

This register is used to enable and reset the entire GPIO2 block. The following sequence must be used to bring the GPIO2 block out of reset:

- Set the CE bit to enable clocks while leaving the block in reset (MR=1).
- Clear the MR bit to enable the GPIO2 signals.

gpio2_enable **Offset = 0x0014**

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

Bits	Name	Description	R/W	Default
31:2	—	Reserved.	R	0
1	MR	Module Reset. 0 Normal operation 1 Hold the GPIO2 block in reset.	R/W	1
0	CE	Clock Enable. 0 Disable the GPIO2 block. 1 Enable the GPIO2 block.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

10 System Control

The Au1210™ and Au1250™ processors contain a robust system control strategy that includes the means to control the following:

- Clocking (See Section 10.1, "Clocks" on page 366.)
- Time of Year and Real Time Clock counters (See Section 10.2 "Time of Year Clock and Real Time Clock" on page 375.)
- GPIO control (See Section 10.3 "Primary General Purpose I/O and Pin Functionality" on page 381.)
- Power management (See Section 10.4 "Power Management" on page 387.)

All registers in the system control block are located off of the base address shown in Table 10-1.

Table 10-1. System Control Block Base Address

Name	Physical Base Address	KSEG1 Base Address
sys_base	0x0 1190 0000	0xB190 0000

The registers in the system control block are affected differently by system events such as power-on hardware reset, Sleep, Hibernate, and runtime reset. (See Figure 11 "Power-up, Reset and Boot" on page 399 for a discussion on the different reset types.) Each register is documented with how it is affected by the different system events. Care should be taken by the system designer to observe which registers do and do not revert to defaults when the different system events occur.

www.DataSheet4U.com

www.DataSheet4U.com

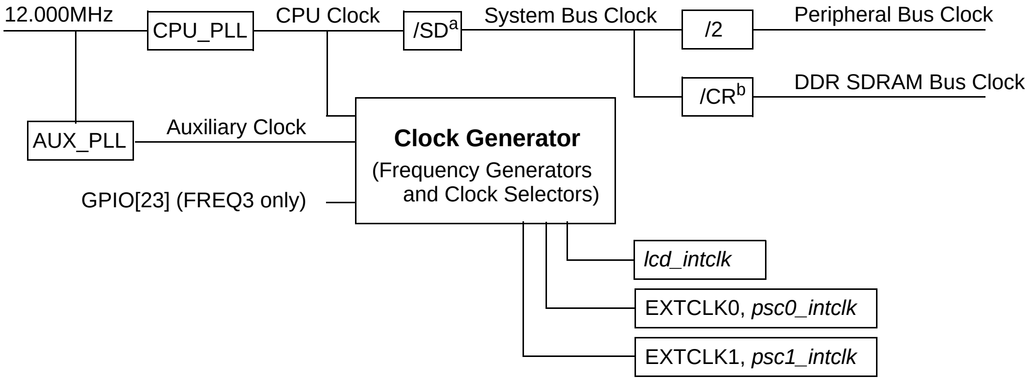
10.1 Clocks

The Au1210™ and Au1250™ processors support two oscillator inputs: 12MHz and 32.768KHz. This section documents the clock domains driven directly and indirectly by the 12MHz input. The 32.768KHz clock input drives the two programmable counters intended for use as a real time clock (RTC) and time of year clock (TOY). The programmable counters are described in Section 10.2 "Time of Year Clock and Real Time Clock" on page 375. (See also Section 14.8 "Crystal Specifications" on page 470 for the specifications of both crystals.)

The Au1210 and Au1250 processors contain two PLLs driven by the 12MHz oscillator and a clock generator from which the following are derived:

- CPU clock
- System bus clock
- Peripheral bus clock
- DDR SDRAM bus clock
- Programmable clocks needed by certain peripherals
- Programmable clocks EXTCLK[1:0] for external use (provided on pins shared with the GPIO[3:2] signals)

Figure 10-1 shows the relationship between the CPU clock, the system bus clock and the peripheral bus clock. As shown, the system bus frequency is derived by dividing the CPU clock by the value programmed in **sys_powerctrl**[SD] (described in Section 10.4.5.4 "Power Control Register" on page 395). The peripheral bus clock is the system bus frequency divided by 2. The DDR SDRAM bus clock can be taken directly from the system bus clock or divided by 2, depending on **mem_sdconfigb**[CR] (described in Section 3.1.1.4 "Global Configuration Register B" on page 62). Figure 10-1 also shows the programmable clock generator driving clocks to the peripheral blocks; see Section 10.1.1.1 "Clock Generator Registers" on page 367.



NOTES:

a. SD is a programmable field in the **sys_powerctrl** register as described in Section 10.4.5 "Power Management Registers" on page 393. SD can be 2, 3, or 4.

b. CR is a programmable field in the **mem_sdconfigb** register as described in Section 3.1.1.4 "Global Configuration Register B" on page 62. CR can be 1 or 2.

c. Supported combinations of SD and CR are listed in Table 3-3 on page 72 and Table 3-4 on page 73.

Figure 10-1. Clocking Topology

www.DataSheet4U.com

www.DataSheet4U.com

10.1.1 Clock Registers

The clock registers and their associated offsets are listed in Table 10-2.

Table 10-2. Clock Registers

Offset	Register Name	Description	Reset Type
0x0020	sys_freqctrl0	Controls frequency generators 0, 1, and 2	Hardware
0x0024	sys_freqctrl1	Controls frequency generators 3, 4, and 5	Hardware
0x0028	sys_clksrc	Controls the derived clocks	Hardware
0x0060	sys_cpupll	Controls CPU PLL frequency	Hardware
0x0064	sys_auxpll	Controls Auxiliary PLL frequency	Hardware & Runtime

10.1.1.1 Clock Generator Registers

The clock generator provides clocks to integrated peripherals and two externally available clocks. The clock generator contains six independent frequency generators which take the two PLLs as inputs. Frequency generator 3 (FREQ3) also takes GPIO[23] as an optional clock source. The frequency generator outputs and the AUX PLL provide the sources for the clock source selectors connected to the integrated peripherals and two external clocks, as shown in Figure 10-2.

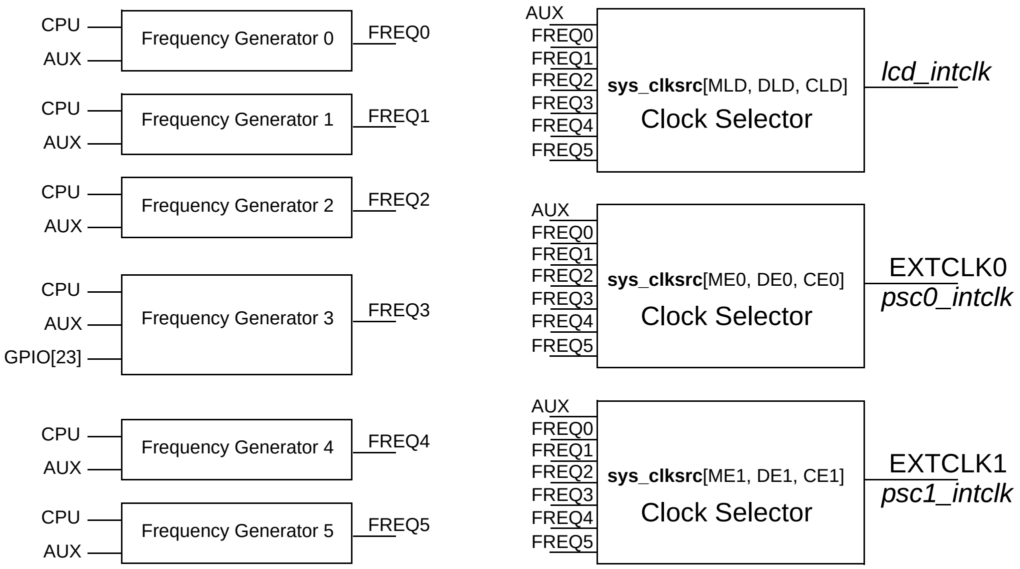
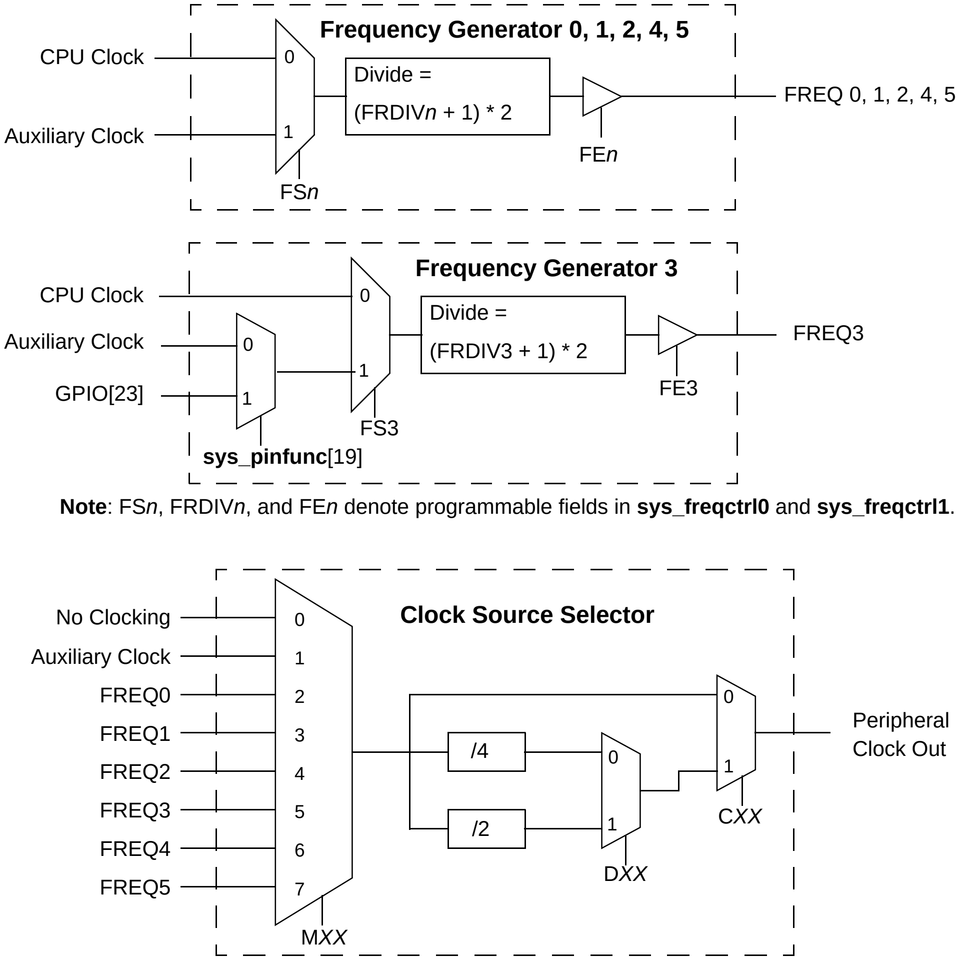


Figure 10-2. Clock Generator Block Diagram

www.DataSheet4U.com

www.DataSheet4U.com

Figure 10-3 shows a block diagram of the frequency generators and the clock source selectors. The names in the diagram correspond to the bit names in the control registers.



Note: FS*n*, FRDIV*n*, and FE*n* denote programmable fields in **sys_freqctrl0** and **sys_freqctrl1**.

Note: M*XX*, D*XX*, and C*XX* denote programmable fields in **sys_clksrc**.

Figure 10-3. Frequency Generator and Clock Source Selector Block Diagram

Each peripheral has clock restrictions. If these restrictions are not met, the peripheral does not operate correctly.

The EXTCLK[1:0] clocks can be programmed for system use. The EXTCLK[1:0] clocks have a maximum frequency rating of (F_{max} / 16), *where* F_{max} is the maximum frequency rating for the part. For example, for a 396MHz part be sure the EXTCLK[1:0] clocks are programmed to run at no more than 24.75MHz. Also, the EXTCLK[1:0] clocks are multiplexed signals and require programming of the **sys_pinfunc** register (see Section 10.3.1.1 "Pin Function" on page 381).

www.DataSheet4U.com

www.DataSheet4U.com

Frequency Control 0

This register controls frequency generators 0, 1, and 2. This register resets to defaults only on a hardware reset. During a runtime reset and during Sleep, this register retains its value.

sys_freqctrl0 **Offset = 0x0020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			FRDIV2								FE2	FS2	FRDIV1								FE1	FS1	FRDIV0								FE0	FS0
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:30	—	Reserved	—	—
29:22	FRDIV2	Divider for Frequency Generator 2. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
21	FE2	Frequency Generator 2 Output Enable. 0 Disable output 1 Enable output	R/W	0
20	FS2	Frequency Generator 2 Source. 0 CPU core clock 1 Auxiliary clock	R/W	0
19:12	FRDIV1	Divider for Frequency Generator 1. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
11	FE1	Frequency Generator 1 Output Enable. 0 Disable output 1 Enable output	R/W	0
10	FS1	Frequency Generator 1 Source. 0 CPU core clock 1 Auxiliary clock	R/W	0
9:2	FRDIV0	Divider for Frequency Generator 0. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
1	FE0	Frequency Generator 0 Output Enable. 0 Disable output 1 Enable output	R/W	0
0	FS0	Frequency Generator 0 Source. 0 CPU core clock 1 Auxiliary clock	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Frequency Control 1

This register controls frequency generators 3, 4, and 5. This register resets to defaults only on a hardware reset. During a runtime reset and during Sleep, this register retains its value.

sys_freqctrl1 **Offset = 0x0024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			FRDIV5								FE5	FS5	FRDIV4								FE4	FS4	FRDIV3								FE3	FS3
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:30	—	Reserved.	—	—
29:22	FRDIV5	Divider for Frequency Generator 5. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
21	FE5	Frequency Generator 5 Output Enable. 0 Disable output 1 Enable output	R/W	0
20	FS5	Frequency Generator 5 Source. 0 CPU core clock 1 Auxiliary clock	R/W	0
19:12	FRDIV4	Divider for Frequency Generator 4. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
11	FE4	Frequency Generator 4 Output Enable. 0 Disable output 1 Enable output	R/W	0
10	FS4	Frequency Generator 4 Source. 0 CPU core clock 1 Auxiliary clock	R/W	0
9:2	FRDIV3	Divider for Frequency Generator 3. The frequency divider is (FRDIV + 1) * 2, *where* FRDIV is the value programmed in this field.	R/W	0
1	FE3	Frequency Generator 3 Output Enable. 0 Disable output 1 Enable output	R/W	0
0	FS3	Frequency Generator 3 Source. 0 CPU core clock 1 Auxiliary clock or GPIO[23]. If **sys_pinfunc**[FS3] = 0, the auxiliary clock is selected. If **sys_pinfunc**[FS3] = 1, GPIO[23] is selected.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Clock Source Control

This register selects the clock sources for the integrated peripherals and two external clocks. This register resets to defaults only on a hardware reset. During a runtime reset and during Sleep, this register retains its value.

sys_clksrc **Offset = 0x0028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			ME1			DE1	CE1	ME0			DE0	CE0																MLD			DLD	CLD
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:30	—	Reserved.	—	—
29:27	ME1	EXTCLK1 and *psc1_intclk* Clock Mux Input Select. See Table 10-3 on page 372.	R/W	0
26	DE1	EXTCLK1 and *psc1_intclk* Clock Divider Select. 0 Divide by 4 1 Divide by 2	R/W	0
25	CE1	EXTCLK1 and *psc1_intclk* Clock Select. 0 Clock is taken directly from mux. (The divider select bit DE1 has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0
24:22	ME0	EXTCLK0 and *psc0_intclk* Clock Mux Input Select. See Table 10-3 on page 372.	R/W	0
21	DE0	EXTCLK0 and *psc0_intclk* Clock Divider Select. 0 Divide by 4 1 Divide by 2	R/W	0
20	CE0	EXTCLK0 and *psc0_intclk* Clock Select. 0 Clock is taken directly from mux. (The divider select bit DE0 has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0
19:5	—	Reserved.	—	—
4:2	MLD	*lcd_intclk* Clock Mux Input Select. See Table 10-3 on page 372.	R/W	0
1	DLD	*lcd_intclk* Clock Divider Select. 0 Divide by 4. 1 Divide by 2.	R/W	0
0	CLD	*lcd_intclk* Clock Select. 0 Clock is taken directly from mux. (The divider select bit DLD has no effect.) 1 Clock is taken from 2/4 divider.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

The specific values written to the 3-bit clock-mux-input-select fields are shown in Table 10-3. The FREQ*n* selections come from the output of the corresponding frequency generators, as shown in Figure 10-3 on page 368.

Table 10-3. Clock Mux Input Select Values

Value	Meaning
000	No clocking
001	Auxiliary Clock
010	FREQ0
011	FREQ1
100	FREQ2
101	FREQ3
110	FREQ4
111	FREQ5

10.1.1.2 PLL Control Registers

The two independent PLLs in the Au1210 and Au1250 processors drive the CPU clock and the auxiliary clock. Each PLL has its own control register. When programming the PLL control registers, the system designer must not violate the rated frequency limits of the Au1210 and Au1250 processors; configuring the PLLs outside this frequency range causes undefined behavior.

The system must provide the correct voltage for the operating frequency *before* changing the CPU clock. See Section 14.3 "DC Parameters" on page 442, for full information about the voltage/frequency requirements of the Au1210 and Au1250 processors.

CPU PLL Control

This register resets to its default value only for a hardware reset. That is, after Sleep, and during a runtime reset the CPU PLL retains its frequency. This register is read/write; however, a read from this register is valid only after it has been initialized with a write.

The CPU PLL Control register should be modified only when all integrated peripherals and the SDRAM controller are disabled.

After writing to the **sys_cpupll** register, bus clocks shut off and approximately 40µs elapse while the CPU PLL locks to the new frequency. During this period instructions are not executed and interrupts are not serviced. Interrupts are serviced once execution begins again at the new frequency.

sys_cpupll **Offset = 0x0060**

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

Bits	Name	Description	Read/Write	Default
31:6	—	Reserved.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
5:0	PLL[5:0]	CPU PLL Multiplier. Determines the integer multiplier used to multiply the CPU PLL to generate the CPU clock. For example, with the default of 16 and a 12MHz OSC frequency, the CPU clock frequency is 192MHz. PLL multiplier values that place the clock frequency outside of rated limits are invalid. 0–15: Reserved and undefined 16–(*n*-1): Valid PLL multiplier *n*–63: Reserved and undefined *n* is the smallest PLL multiplier that would cause the CPU clock frequency to exceed the rated frequency limits of the part. See Table 10-4.	R/W	0x10

Table 10-4 shows the rated CPU frequency alongside the actual CPU frequency and PLL multiplier when using a 12MHz crystal.

Table 10-4. Rated and Actual CPU Frequencies Using a 12MHz Crystal

Rated Frequency (MHz)	Actual Frequency (MHz)	CPU PLL Multiplier
600	600	50
500	492	41
400	396	33
333	324	28

www.DataSheet4U.com

www.DataSheet4U.com

Auxiliary PLL Control

This register resets to its default value on hardware reset, after Sleep, and during a runtime reset. This register is read/ write; however, a read from this register is valid only after it has been initialized with a write.

Any peripherals using the AUX PLL as a source clock should be disabled before modifying the AUX PLL register.

Unlike the **sys_cpupll** register, writing **sys_auxpll** does not cause the system to halt. As a consequence, clocks taken from the AUX PLL may be unstable for up to 40 µs. To ensure stable clocks during AUX PLL lock time, the system programmer can modify **sys_auxpll** and then write **sys_cpupll** with its current value to force the system to halt for 40 µs.

sys_auxpll **Offset = 0x0064**

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

Bits	Name	Description	Read/ Write	Default
31:6	—	Reserved.	R/W	0
5:0	PLL	Auxiliary PLL Multiplier. Determines the integer multiplier used to multiply the auxiliary PLL to generate the auxiliary clock. For example, with a value of 12 and a 12MHz OSC frequency, the auxiliary clock frequency is 144MHz. PLL multiplier values that place the clock frequency outside of rated limits are invalid. 0: Disable the auxiliary PLL. 1–7: Reserved and undefined 8–(*n*-1): Valid PLL multiplier *n*–63: Reserved and undefined *n* is the smallest PLL multiplier that would cause the auxiliary clock frequency to exceed the rated frequency limits of the part.	R/W	0

10.1.2 Hardware Considerations

The EXTCLK[1:0] clocks are multiplexed signals and require programming of the **sys_pinfunc** register (see Section 10.3.1.1 "Pin Function" on page 381).

Section 14.8 "Crystal Specifications" on page 470, defines the crystal specifications.

10.1.3 Programming Considerations

When changing the CPU PLL value through the **sys_cpupll** register, the system automatically halts for 40µs to allow clocks to stabilize. During this time no interrupts are serviced, potentially affecting real-time systems. However, modifying the **sys_auxpll** register does *not* cause the system to halt, and therefore clocks taken from the AUX PLL may be unstable for up to 40µs. To ensure stable clocks while the AUX PLL locks, the **sys_cpupll** register can be written with its current value to force the system to halt for 40µs.

www.DataSheet4U.com

www.DataSheet4U.com

10.2 Time of Year Clock and Real Time Clock

The Au1210™ and Au1250™ processors contain two programmable counters designed for use as a time of year clock (TOY) and real time clock (RTC). Because the TOY continues counting through Sleep and Hibernate, a TOY counter match can be used as a wake-up source. The RTC, however, stops counting during Sleep and Hibernate.

Both the TOY and RTC counters are driven by the 32.768KHz clock input. The clock input source can be a crystal or external clock. (See Section 14.8 "Crystal Specifications" on page 470 for crystal details.)

The TOY and RTC each contain a register to initialize the counter or load a new value, a trim divider to adjust the incoming 32.768KHz clock, and three match registers that have associated interrupts that trigger on a match. Each counter is also able to generate an interrupt on every tick. All interrupts are maintained through the interrupt controller. The TOY and RTC share a status register.

Figure 10-4 shows the functional block diagram of both the TOY and the RTC. The registers used to implement the block, including the Counter Control register (**sys_cntrctrl**), are described in the following section.

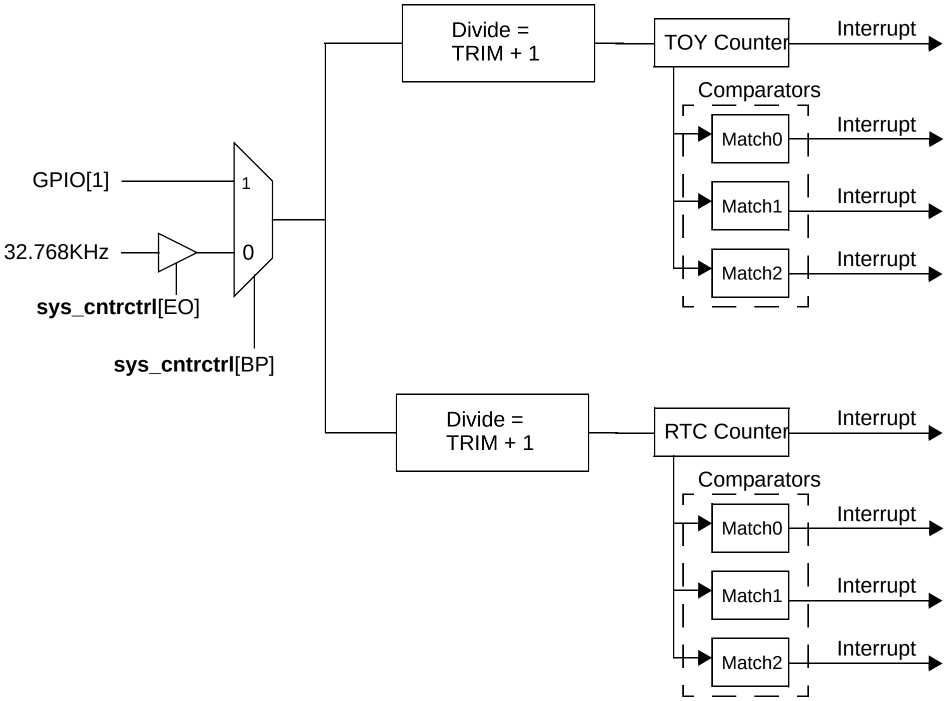


Figure 10-4. TOY and RTC Block Diagram

www.DataSheet4U.com

www.DataSheet4U.co

10.2.1 Time of Year Clock and Real Time Clock Registers

Each counter operates identically with the only difference being that the TOY continues counting through Sleep and Hibernate, and the RTC does not.

The TOY and RTC control registers are listed in Table 10-5.

Table 10-5. TOY and RTC Registers

Offset	Register Name	Description	Reset Type
0x0000	sys_toytrim	Trim value for 32.768KHz clock source for TOY	Hardware
0x0004	sys_toywrite	TOY counter value is written through this register.	Hardware
0x0008	sys_toymatch0	TOY match 0 value for interrupt generation	Hardware
0x000C	sys_toymatch1	TOY match 1 value for interrupt generation	Hardware
0x0010	sys_toymatch2	TOY match 2 value for interrupt generation	Hardware
0x0014	sys_cntrctrl	Control register for TOY and RTC	Hardware
0x0040	sys_toyread	TOY counter value is read from this register	Hardware
0x0044	sys_rtctrim	Trim value for 32.768KHz clock source for RTC	Hardware
0x0048	sys_rtcwrite	RTC counter value is written through this register.	Hardware
0x004C	sys_rtcmatch0	RTC match 0 value for interrupt generation	Hardware
0x0050	sys_rtcmatch1	RTC match 1 value for interrupt generation	Hardware
0x0054	sys_rtcmatch2	RTC match 2 value for interrupt generation	Hardware
0x0058	sys_rtcread	RTC counter value is read from this register.	Hardware

10.2.1.1 Trim Registers

This register is unpredictable at power-on. During a runtime reset and during Sleep or Hibernate this register retains its value.

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

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

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
																	TRIM[15:0]															
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	Read/Write	Default
31:16	—	Reserved.	—	—
15:0	TRIM	Divide Value for 32.768KHz Input. Divide = TRIM + 1	R/W	UNPRED

10.2.1.2 Counter Write Register

This register is unpredictable at power-on. During a runtime reset and during Sleep or Hibernate this register retains its value.

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

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

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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
31:0	COUNT	Counter Write. The respective counter is updated with the value written to this register at the next trimmed clock.	W	UNPRED

10.2.1.3 Counter Read Register

This register is unpredictable at power-on. During a runtime reset and during Sleep or Hibernate this register retains its value.

sys_toyread - TOY counter value read **Offset = 0x0040**

sys_rtcread - RTC counter value read **Offset = 0x0058**

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

Bits	Name	Description	Read/Write	Default
31:0	COUNT	Counter Read. Contains the current count of the respective counter.	R	UNPRED

10.2.1.4 Match Registers

Each match register is capable of causing an interrupt as shown in Section 5.0 "Interrupt Controller" on page 143. The **sys_toymatch2** register can be used to wake up from Sleep and Hibernate; see Section 10.4.5.2 "Wakeup Source Mask Register" on page 394. See also Section 10.2.2 "Programming Considerations" on page 380.

These registers are unpredictable at power-on. During a runtime reset and during Sleep or Hibernate these registers retain their value.

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

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

sys_toymatch2 - TOY Match 2 **Offset = 0x0010**

sys_rtcmatch0 - RTC Match 0 **Offset = 0x004C**

sys_rtcmatch1 - RTC Match 1 **Offset = 0x0050**

sys_rtcmatch2 - RTC Match 2 **Offset = 0x0054**

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

Bits	Name	Description	Read/Write	Default
31:0	MATCH	A match with the appropriate counter and the value in this register causes an interrupt.	R/W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

10.2.1.5 TOY and RTC Counter Control Register

This register contains control bits and status bits to configure and control both counters. The write status bits indicate when writes are pending for each of the corresponding trim, write, or match registers.

This register resets to default values only on a hardware reset. During a runtime reset and during Sleep or Hibernate this register retains its value.

sys_cntrctrl **Offset = 0x0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RTS	RM2	RM1	RM0	RS		BP						EO	CCS		32S	TTS	TM2	TM1	TM0	TS
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:21	—	Reserved.	—	—
20	RTS	**sys_rtctrim** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
19	RM2	**sys_rtcmatch2** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
18	RM1	**sys_rtcmatch1** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
17	RM0	**sys_rtcmatch0** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
16	RS	**sys_rtcwrite** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
15	—	Reserved.	—	—
14	BP	Bypass the 32.768KHz OSC. 0 Oscillator input (XTI32, XTO32) 1 GPIO[1] drives the counters from an external source or through software using the GPIO controller. If GPIO[1] is used to drive the counters during sleep, software must enable the GPIO[1] input by writing to **sys_pininputen** before entering Sleep mode.	R/W	Unchanged
13:9	—	Reserved.	—	—

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/Write	Default
8	EO	Enable 32.768KHz Oscillator. Enables the clock for the RTC/TOY block. 0 Disable the clock. 1 Enable the clock. The EO bit must be set to enable the RTC/TOY counters. After enabling the clock by setting EO, poll the oscillator status bit (32S) until it returns a ‘1’. Once 32S is set, wait an additional one second to allow for frequency stabilization within the block before accessing other RTC/TOY registers (not including **sys_cntrctrl**). Note: If using an external oscillator or bypassing through GPIO[1], be sure to set EO only *after* a stable clock is being driven into the part. Note: On rise of V_{DDXOK}, hardware clears EO if (FWTOY# = 1); otherwise, hardware leaves EO unchanged.	R/W	0 (except as noted in the description)
7	CCS	**sys_cntrctrl** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
6	—	Reserved.	—	—
5	32S	32.768KHz Oscillator Status. Detects two consecutive 32KHz cycles from the clock source for the RTC/TOY block. 0 Clock is not running. 1 Clock is running. Note: Be sure to wait 1 second after 32S is set to allow for frequency stabilization within the block before accessing RTC/TOY registers.	R	UNPRED
4	TTS	**sys_toytrim** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
3	TM2	**sys_toymatch2** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
2	TM1	**sys_toymatch1** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
1	TM0	**sys_toymatch0** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0
0	TS	**sys_toywrite** Write Status. 0 No write is pending. It is safe to write to the register. 1 A write is pending. Do not write to the register.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

10.2.2 Programming Considerations

Before writing to the trim, write, or match registers, the corresponding write status bit for that register must first be polled to ensure that no writes are currently pending. When software writes to one of these registers, hardware automatically sets the corresponding write status bit in **sys_cntrctrl** to indicate that a write is pending. When the write has taken effect, hardware clears the corresponding write status bit.

www.DataSheet4U.com

www.DataSheet4U.com

10.3 Primary General Purpose I/O and Pin Functionality

The Au1210™ and Au1250™ processors contain two separate GPIO blocks (primary and secondary). This section covers the programming model for the primary general purpose I/O (GPIO) signals. The processors support 43 GPIOs, 27 of which are controlled by the primary GPIO block. For a description of the programming model for the secondary GPIO block see Section 9.8 "Secondary General Purpose I/O" on page 362.

This section also documents how to change the functionality of multiplexed pins. These pins can function at the system level as a GPIO signal, or they can be assigned a signal function dedicated to an integrated peripheral device.

Each GPIO can be configured as either an input or an output. The GPIO ports also can be connected to the internal interrupt controllers to generate an interrupt from input signals. See Section 5.0 "Interrupt Controller" on page 143 for information on interrupts.

10.3.1 Pin Functionality

To maximize the functionality of the Au1210 and Au1250 processors, many of the pins have multiple uses. If a pin is programmed for a certain use, any other functionality associated with that pin cannot be utilized at the same time. In other words, a pin cannot be used as a GPIO at the same time it is assigned to a peripheral device.

For example, if **sys_pinfunc**[DMA] is set, configuring the pin as DMA_REQ1, GPIO[12] cannot be used as a GPIO nor can the GPIO be configured as an interrupt. Conversely if **sys_pinfunc**[DMA] is cleared, configuring the pin as GPIO[12], DMA_REQ1 is not usable. GPIO[12] can be used as a GPIO and to generate interrupts.

(For reference, Figure 13-1 on page 423 shows a block diagram of all external signals. Signals that are multiplexed on one pin show the shared function in parentheses.)

10.3.1.1 Pin Function

This register resets to its default state at hardware reset, runtime reset and Sleep.

sys_pinfunc **Offset = 0x002C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	1	1	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA	S0A	S1A	LP0	LP1	LD	LD8	LD1	LD0	P1A		P1B	FS3	P0A		CS	CIM	P1C		U1T	U1R	EX1	EX0	U0R	MC	S0B	S0C	P0B	U0T	S1B		
Def.	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1

Bits	Name	Description	Read/ Write	Default
31	DMA	GPIO[12]/DMA_REQ1. 0 Pin is configured as GPIO[12]. 1 Pin is configured as DMA_REQ1.	R/W	0
30	S0A	GPIO[28,19,17]/SD0. See also S0B (bit 6) and S0C (bit 5) description. 0 Pins are configured as GPIO[28], GPIO[19], GPIO[17]. 1 Pins are configured as SD0_CMD, SD0_DAT0, SD0_CLK.	R/W	0
29	S1A	PCMCIA/SD1. See also S1B (bit 2) description. 0 Pins are configured as PWAIT#, PIOR#, POE#, PIOW#, PIOS16#. 1 Pins are configured as SD1_DAT0, SD1_DAT1, SD1_DAT2, SD1_CMD, SD1_CLK.	R/W	0
28	LP0	LCD_PWM0/U1DCD#. 0 Pin is configured as U1DCD#. 1 Pin is configured as LCD_PWM0. **System Note**: For systems that use the UART1 interface but do *not* use the optional modem control signals (MC = 0), the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using LCD_PWM0.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
27	LP1	LCD_PWM1/U1RI#. 0 Pin is configured as U1RI#. 1 Pin is configured as LCD_PWM1. **System Note**: For systems that use the UART1 interface but do *not* use the optional modem control signals (MC = 0), the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using LCD_PWM1.	R/W	0
26	LD	GPIO[211]/LCD_D[16]. 0 Pin is configured as GPIO[211]. 1 Pin is configured as LCD_D[16].	R/W	0
25	LD8	GPIO[210]/LCD_D[8]. 0 Pin is configured as GPIO[210]. 1 Pin is configured as LCD_D[8].	R/W	0
24	LD1	GPIO[201]/LCD_D[1]. 0 Pin is configured as GPIO[201]. 1 Pin is configured as LCD_D[1].	R/W	0
23	LD0	GPIO[200]/LCD_D[0]. 0 Pin is configured as GPIO[200]. 1 Pin is configured as LCD_D[0].	R/W	0
22:21	P1A	GPIO[24,22,20,11]/PSC1. See also P1B (bit 20) and P1C (bit 14) description. 00 Pins are configured as PSC1_D0, PSC1_D1, PSC1_CLK, PSC1_SYNC0. Use for SPI, I^{2}S, or AC97. 01 Pins are configured as PSC1_D0, PSC1_CLK, GPIO[22], GPIO[20]. Use for SMBus. 10 Reserved 11 Pins are configured as GPIO[24], GPIO[22], GPIO[20], GPIO[11].	R/W	0x3
20	P1B	GPIO[23]/PSC1. See also P1A (bits [22:21]) and P1C (bit 14) description. 0 Pin is configured as PSC1_EXTCLK. Use for I^{2}S with external clocking. 1 Pin is configured as GPIO[23].	R/W	1
19	FS3	GPIO[23] as a clock source for Frequency Generator 3. 0 GPIO[23] is *not* available as an optional clock source for Frequency Generator 3. 1 GPIO[23] is available as an optional clock source for Frequency Generator 3. See Section 10.1.1.1 "Clock Generator Registers" on page 367.	R/W	1

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
18:17	P0A	GPIO[215,31,16]/PSC0. See also P0B (bit 4) description. 00 Pins are configured as PSC0_D1, PSC0_SYNC0, GPIO[16]. Use for I^{2}S. 01 Pins are configured as PSC0_D1, PSC0_SYNC0, PSC0_SYNC1. Use for SPI or AC97. 10 Pins are configured as GPIO[215], GPIO[31], GPIO[16]. Use for all other configurations. 11 Reserved	R/W	0x3
16	CS	Clock select. Applies only when EX0 = 1. 0 EXTCLK0 drives the pin. 1 32KHz OSC clock drives pin.	R/W	0
15	CIM	GPIO[214:212, 209:202]/CIM. 0 Pins are configured as GPIO[214:212], GPIO[209:202]. 1 Pins are configured as CIM_CLK, CIM_FS, CIM_LS, CIM_D[9:2].	R/W	0
14	P1C	GPIO[21]/PSC1. See also P1A (bits ([22:21]) and P1B (bit 20) description. 0 Pin is configured as GPIO[21]. 1 Pin is configured as PSC1_SYNC1. Use for SPI or AC97.	R/W	1
13	—	Reserved.	—	—
12	U1T	GPIO[15]/U1TXD. 0 Pin is configured as U1TXD. 1 Pin is configured as GPIO[15].	R/W	1
11	U1R	GPIO[30]/U1RXD. 0 Pin is configured as GPIO[30]. 1 Pin is configured as U1RXD.	R/W	0
10	EX1	GPIO[3]/EXTCLK1. 0 Pin is configured as GPIO[3]. GPIO[3] is used as the input clock source (LCD_CLKIN) for the LCD controller when **lcd_clockctrl**[EXT] is set. 1 Pin is configured as EXTCLK1.	R/W	0
9	EX0	GPIO[2] / (EXTCLK0 or 32KHz OSC). 0 Pin is configured as GPIO[2]. 1 Pin is configured as EXTCLK0 or 32KHz OSC. CS (bit 16) selects EXTCLK0 or the 32KHz OSC.	R/W	0
8	U0R	GPIO[29]/UART0. 0 Pin is configured as GPIO[29]. 1 Pin is configured as U0RXD.	R/W	0
7	MC	GPIO[14:13,10:9]/UART1 modem control. 0 Pins are configured as GPIO[14:13], GPIO[10:9]. 1 Pins are configured for U1RTS#, U1CTS#, U1DSR#, U1DTR#. **System Note**: For systems that use the UART1 interface but do *not* use the optional modem control signals (MC = 0), the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], and GPIO[10].	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
6	S0B	GPIO[8,6]/SD0. See also S0A (bit 30) and S0C (bit 5) description. 0 Pins are configured as GPIO[8,6]. 1 Pins are configured as SD0_DAT1, SD0_DAT2.	R/W	0
5	S0C	GPIO[26]/SD0. See also S0A (bit 30) and S0B (bit 6) description. 0 Pin is configured as SD0_DAT3. 1 Pin is configured as GPIO[26].	R/W	1
4	P0B	GPIO[25,18]/PSC0. See also P0A (bits [18:17]) description. 0 Pins are configured as PSC0_CLK, PSC0_D0. Use for all PSC configurations. 1 Pins are configured as GPIO[25], GPIO[18].	R/W	1
3	U0T	GPIO[27]/UART0. 0 Pin is configured as U0TXD. 1 Pin is configured as GPIO[27].	R/W	1
2	S1B	PCMCIA/SD1. See also S1A (bit 29) description. 0 Pin is configured as SD1_DAT3. 1 Pin is configured as PWE#.	R/W	1
1:0	—	Reserved.	—	—

10.3.2 Primary GPIO Control Registers

The primary GPIOs on the Au1210 and Au1250 processors are 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, or write, as there are separate registers for setting and clearing a bit. In this way a function can freely manipulate its associated GPIOs without interfering with other functions.

Figure 10-5 shows the logical implementation of each GPIO. The names represent bit *n* of the corresponding register that affects GPIO[*n*].

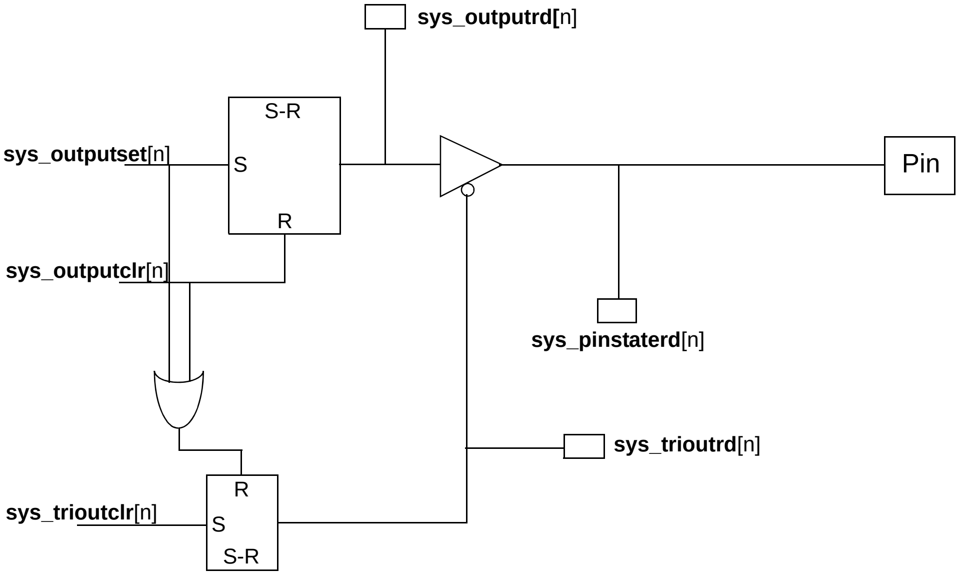


Figure 10-5. GPIO Logic Diagram

www.DataSheet4U.com

www.DataSheet4U.com

Table 10-6 shows the GPIO control registers and the associated offsets from **sys_base**. Certain registers share offsets and have different functionality depending on whether the access is a read or a write. The register descriptions detail the functionality of each register. Bit *n* of a particular register should be associated with GPIO[n] for all registers except **sys_pininputen**.

Table 10-6. GPIO Control Registers

Offset	Register Name	Register Description	Default	Reset Type
0x0100	sys_trioutrd	Displays the tristate/output state of GPIO[n].	0: All GPIOs are tristated.	Hardware
0x0100	sys_trioutclr	Set sys_trioutclr[n] to clear and tristate the corresponding bit. All input pins must be tristated.		
0x0108	sys_outputrd	Displays the output state of GPIO[n].	UNPRED	Hardware
0x0108	sys_outputset	Set sys_outputset[n] to enable a high level output.		
0x010C	sys_outputclr	Set sys_outputclr[n] to enable a low level output. Setting an output pin brings the pin out of tristate mode and enables the output.		
0x0110	sys_pinstaterd	Displays the state of the pins.	UNPRED	Hardware
0x0110	sys_pininputen	Writing to this register allows the system to use GPIO[7:0] as external inputs to wake the processor from Sleep. This register must be written before any GPIO[7:0] signal can be used as a Sleep wakeup input source.	UNPRED	

10.3.2.1 GPIO Control Registers

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

These registers reset to default values only on a hardware reset; they retain their values during a runtime reset and during Sleep.

sys_trioutrd **Offset = 0x0100**

sys_trioutclr **Offset = 0x0100**

sys_outputrd **Offset = 0x0108**

sys_outputset **Offset = 0x0108**

sys_outputclr **Offset = 0x010C**

sys_pinstaterd **Offset = 0x0110**

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
	FUNC[31:0]																															

Bits	Name	Description	Read/Write	Default
31:0	FUNC[n]	The function of each register is given in Table on page 385. FUNC[n] controls the functionality of GPIO[n].	**…rd** - read only **…set** - write only **…clr** - write only See Table 10-7 on page 386.	See Table on page 385 for the default values at hardware reset.

www.DataSheet4U.com

www.DataS

Certain GPIO control registers have the same offset but offer different functionality depending on whether a read or a write is being performed. See Table 10-7.

Table 10-7. Read/Write Accessibility of GPIO Control Registers

Register Name	Read/Write	Function
sys_trioutrd sys_outputrd sys_pinstaterd	Read only	Return the current value of the register.
sys_outputset	Write only	Set all bits that are written 1. Writing a value of 0 has no effect on the corresponding bit.
sys_trioutclr sys_outputclr	Write only	Clear all bits that are written 1. Writing a value of 0 has no effect on the corresponding bit.

10.3.2.2 GPIO Input Enable

This register is write only. Writing any value to **sys_pininputen** allows the system to use GPIO[7:0] as external inputs to wake the processor from Sleep. This register must be written before any GPIO[7:0] signal can be used as a Sleep wakeup input source.

If GPIO[1] is used to bypass the 32.768KHz oscillator input, software must write to **sys_pininputen** before entering Sleep mode to enable GPIO[1] to drive the counters during Sleep.

Writes to **sys_pininputen** are required only once per hardware reset and Hibernate event; **sys_pininputen** maintains its value for Sleep events, and runtime resets.

sys_pininputen **Offset = 0x0110**

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

Bits	Name	Description	Read/Write	Default
31:0	EN	Writing any value to this register allows a GPIO[7:0] signal to be used as a wakeup source during Sleep.	W	N/A

10.3.3 Open Drain Emulation

An open drain can be emulated on a primary GPIO pin as follows:

- Add a pull-up resistor to the GPIO pin.
- Set the corresponding bit in the **sys_outputclr** register to drive the signal low.
- Set the corresponding bit in the **sys_trioutclr** register to allow the signal to be pulled high.

10.3.4 Using GPIO for External DMA Requests

See Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139 for information.

www.DataSheet4U.com

www.DataSheet4U.com

10.4 Power Management

The Au1210™ and Au1250™ processors contain 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 Au1210 and Au1250 processors, power management can be broken into three different areas:

- CPU
- Peripherals
- Device

The low power states consist of putting the entire device into a Sleep or Hibernate state. The CPU also supports two idle states that differ as to whether bus snooping is supported. In addition each integrated peripheral can have its clocks disabled when not in use, thus significantly reducing the power drawn by those peripherals not in use.

The flow chart in Figure 10-6 shows the different stages of power management for the CPU (IDLE0,1) and the device (SLEEP) and how each state is entered and left. Any interrupt can be used to bring the CPU out of either idle state, while only a GPIO[7:0] or **sys_toymatch2** interrupt can be enabled (in **sys_wakemsk**) to bring the device out of Sleep.

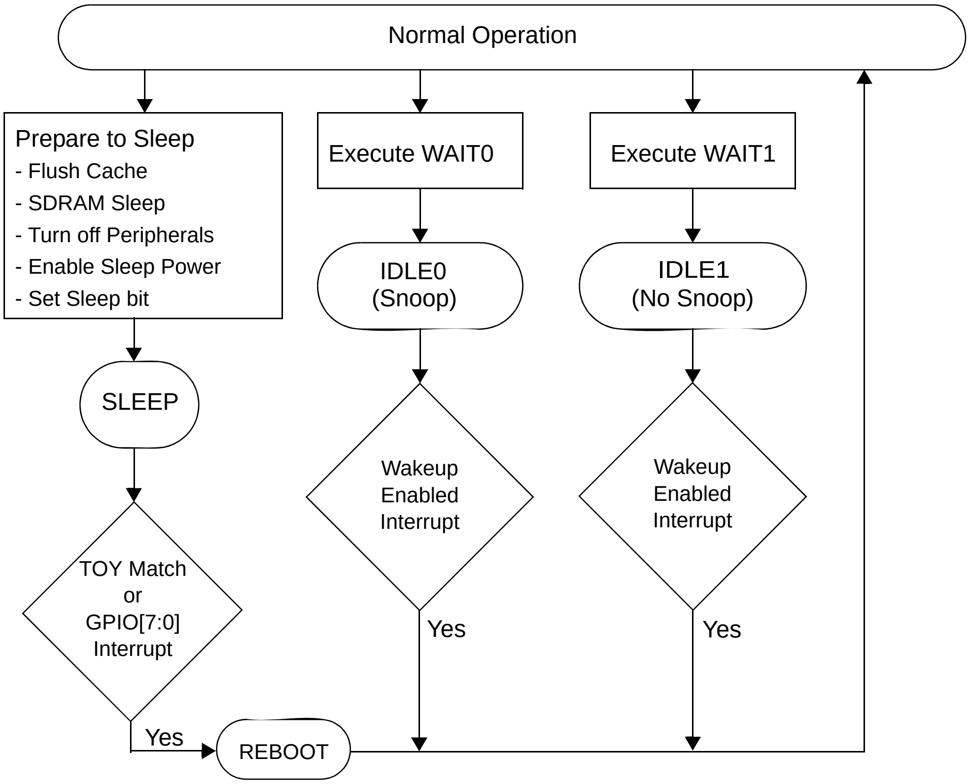


Figure 10-6. Sleep and Idle Flow Diagram

10.4.1 CPU Power Management - Idle

The CPU can be put into two different low-power idle modes (IDLE0 and IDLE1) by using the **WAIT** instruction:

- 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.

The **WAIT** instruction and at least four instructions following it must be in the instruction cache for the wait to occur. See Section 2.7.3 "WAIT Instruction" on page 36 for more information.

At all times, the MMU data cache execution and multiply-and-accumulate blocks are placed in a low power state if they are not being used.

www.DataSheet4U.com

www.DataSheet4U.com

10.4.1.1 Returning from Idle

The processor wakes from the idle state (IDLE0 or IDLE1) upon receiving an interrupt. The time required for the processor core to return to normal execution is as follows:

- Five to ten CPU clocks are needed to restart clocks to the CPU.
- It takes an additional ten CPU clocks for the core to recognize the interrupt and begin fetching the interrupt service routine.

Therefore, a maximum of 20 CPU clocks are required to resume normal instruction pipeline execution. If the interrupt service routine is in the instruction cache, the instruction returns immediately; otherwise, there is an additional delay while fetching the instruction from memory.

10.4.2 Peripheral Power Management

Peripheral power management is handled through clock management and disabling of unused peripherals. Table 10-8 lists the peripherals and their related power management registers. The actual register descriptions should be referred to for programming details.

When separate reset/peripheral enable and clock-enable bits are provided, the reset must be applied first, and then the clocks should be disabled. This simplifies programming, as the suggested bring up sequence is typically to first enable clocks and then subsequently to bring the peripheral out of reset.

Table 10-8. Peripheral Power Management

Peripheral	Power Management Register	Power Management Strategy
UART*n*	uart*n*_enable	When a UART is not being used, the E bit should be cleared to hold the part in reset, and the CE bit should be cleared to disable clocks to the block.
Primary General-Purpose I/O (GPIO) Controller	sys_trioutclr	Although there is not a specific low-power configuration for the primary GPIOs, tri-stating the unused GPIOs minimizes their power usage.
Secondary General-Purpose I/O (GPIO2) Controller	gpio2_enable	If no GPIO2 signals are being used, the GPIO2 module reset (MR) bit should be set to place the module in reset. Also, clear the CE bit to disable clocks to the block. (By default, the GPIO2 module is disabled coming out of reset.)
TOY and RTC	sys_cntrctrl	If both the TOY and RTC are not being used, then disable the oscillator.
PSC	psc*n*_ctrl	If a PSC is not in use, clear the ENA bit to disable the block.
MAE Back End	maebe_ctlenable	If the MAE back end is not in use, clear the EN bit to disable the block.
USB Controller	usb_cfg	If the USB 2.0 host controller is not in use, clear the ECE bit to disable its clock. If the USB 2.0 device controller is not in use, clear the UCE bit to disable its clock. If the USB 1.1 host controller is not in use, clear the OCE bit to disable its clock.
LCD Controller	lcd_screen, lcd_outenable, lcd_pwmdiv	If the LCD controller is not in use, clear the **lcd_screen**[EN] and **lcd_outenable**[ENABLE] bits to disable the block. If the PWM clocks are not used, clear **lcd_pwmdiv**[EN].
CIM	cim_enable	If the CIM is not in use, clear the EN bit to disable the block.
Software Counter	swcnt_control	If the software counter is not in use, clear the EN bit to disable the block.
SD*n* Controller	sd*n*_enable	If the SD block is not in use, clear the R bit to place the block in reset, and clear the CE bit to disable the clock.

www.DataSheet4U.com

www.DataSheet4U.com

10.4.3 Device Power Management - Sleep

The Sleep state of the Au1210 and Au1250 processors puts the entire device into a low-power state. Sleep requires a complete system initialization on wakeup. There are multiple steps to take when entering Sleep and when waking to ensure data integrity. During this state, all registers outside the system control block lose their software-configured values, and cache coherency is not maintained.

The TOY continues clocking and remains functional during Sleep. However, the RTC, as well as other clocks throughout the Au1210 and Au1250 processors are disabled during Sleep.

When coming out of Sleep, a programmable delay is defined by **sys_powerctrl**[VPUT]. This is the time that the system designer has to ensure V_{DDI} is stable from the rising edge of PWR_EN.

The following example shows how to enter Sleep when using SDRAM. This code must be run from non-volatile memory (such as ROM, Flash, or SRAM) because the processor cannot access SDRAM once the memory is put into self-refresh mode.

1) Enable Sleep Power by writing to the **sys_slppwr** register.

2) Turn off all peripherals. (Explicitly turning off all peripherals in use ensures a graceful transition to Sleep mode.)

3) Flush and invalidate the data cache. During Sleep, the caches do not retain their state.

4) If SDRAM contents are to be kept through Sleep, SDRAM must be put into self-refresh mode (see Section 3.1.1.9 "Self Refresh Toggle Command Register" on page 66). If SDRAM does not need to be maintained through Sleep, disable the SDRAM.

5) If using one of GPIO[7:0] as a wakeup source, **sys_pininputen** must be written to enable the GPIO as a Sleep wakeup source if this has not already been done at system startup and after each Hibernate event.

6) If using GPIO[1] to bypass the 32.768KHz oscillator (**sys_cntrctrl**[BP] = 1), software must write to **sys_pininputen** if this has not already been done at system startup and after each Hibernate event. A write to **sys_pininputen** enables GPIO[1] to drive the counters during Sleep so that a timer match can be used as a Sleep wakeup source.

7) The **sys_wakesrc** register must be written to explicitly clear any pending wake interrupts.

8) The **sys_wakemsk** register must be set with the appropriate value according to what signal(s) should wake the processor (GPIO[7:0] or **sys_toymatch2**).

9) Enable Sleep by writing to the **sys_sleep** register. This step puts the system to sleep.

10) As the system enters Sleep mode, the PWR_EN signal is negated. This can be used to disable V_{DDI} and V_{DDY} if needed. On initial power-on, PWR_EN asserts as soon as V_{DDXOK} asserts. The system should assert V_{DDXOK} only after *both* V_{DDX} and V_{DDY} have ramped up. For Sleep wakeup, however, PWR_EN can be used to ramp V_{DDY}. Thus, the logic equation for the system's V_{DDY} power-enable signal needs to be PWR_EN_for_V_{DDY} = PWR_EN *OR* (initial power-on ramp).

When the processor takes a Sleep interrupt to wakeup, the following steps must be taken:

1) PWR_EN is asserted by hardware. Within the time indicated by **sys_powerctrl**[VPUT], the system must ensure that V_{DDI} is stable. If V_{DDY} has been disabled during Sleep it must also be stable within this time.

2) The processor boots from physical address 0x1FC00000 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 must perform a complete system initialization. All registers in the system control block, except where noted, are at their default values.

www.DataSheet4U.com

www.DataSheet4U.com

10.4.3.1 Sleep Sequence and Timing

As the processor enters Sleep mode, the system designer has the option of disabling V_{DDI} and V_{DDY} to conserve power. The PWR_EN signal defines the Sleep window. Figure 10-7 shows the Sleep sequence.

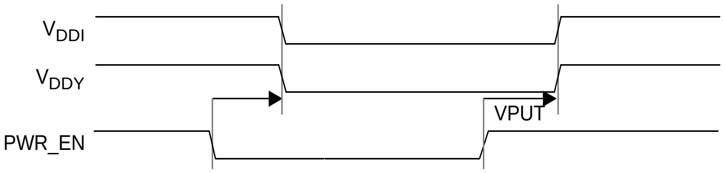


Figure 10-7. Sleep Sequence

The system designer must ensure V_{DDI} and V_{DDY} are stable from the rising edge of PWR_EN within the time period as programmed in **sys_powerctrl**[VPUT]. V_{DDXOK} (not shown) remains asserted during the Sleep sequence.

10.4.4 Device Power Management - Hibernate

In addition to the low power clock modes and Sleep features, the Au1210 and Au1250 processors have an extremely low power Hibernate mode. In Hibernate mode, the entire processor and system board can be powered down while still preserving the TOY timer. Entering and exiting Hibernate mode is handled by system hardware via the WAKE# and FWTOY# signals. Status bits in **sys_wakesrc** enable software to distinguish Hibernate wakeup from normal reset.

10.4.4.1 Hibernate Mode vs. Sleep

For Sleep, software must first take preparation steps including writing to **sys_slppwr**, flushing the data cache, and putting the SDRAMs into self-refresh mode before writing to **sys_sleep**. Only Sleep preserves the values in **sys_scratch0** and **sys_scratch1**. V_{DDX} and V_{DDXOK} must be kept high throughout Sleep until a wakeup event occurs, either through GPIO[7:0] or the TOY. After Sleep wakeup, the processor waits an amount of time dictated by **sys_powerctrl**[VPUT] for V_{DDI} to come up.

Unlike Sleep, Hibernate is initiated instantly at any time by asserting the FWTOY# signal before negating V_{DDXOK}. Flushing the cache and refreshing the SDRAM memory is not required since V_{DDX} is brought down for extreme power savings. Wakeup can be triggered only by system hardware bringing V_{DDX} back up and reasserting V_{DDXOK}. After Hibernate wakeup, **sys_powerctrl**[VPUT] is initialized to 20ms, so the processor waits 20ms for V_{DDI} to ramp. FWTOY# must be negated during this V_{DDI} ramp time.

Both Sleep and Hibernate guarantee that the TOY is kept running accurately, provided that time has been initialized prior to Sleep or Hibernate. Although the processor cannot wake from Hibernate by itself due to a timer match, it can assert WAKE#, which the system can use to bring V_{DDX} back up.

It is possible to enter Hibernate at any time that $V_{DDXOK} = 1$, including while entering Sleep, during Sleep, or while waking. However, entering Hibernate while software is still initializing the TOY timer requires the TOY timer to be re-initialized after Hibernate wakeup. Also, if WAKE# is used to wake from Hibernate, the timer match must be enabled prior to Hibernate. Attempting to enter Hibernate when $V_{DDXOK} = 0$ is not possible; Hibernate must be initiated before or concurrent with V_{DDXOK} and when V_{DDX} falls.

10.4.4.2 Hibernate Mode vs. Initial Power-up

On both initial power-up and Hibernate wakeup, **sys_wakesrc**[IP] is set. On Hibernate wakeup, a status bit denoting Hibernate wakeup (**sys_wakesrc**[DW]) is set, as well as another bit for Hibernate TOY match (**sys_wakesrc**[M2D]), if applicable. As long as the different power domains are brought up in proper order, the **sys_wakesrc** status bits are accurate; if a system is designed so that power domains are brought up out of order, software must ignore the status bits.

On initial power-on, the 32KHz oscillator is forced to a disabled state. While in Hibernate and during Hibernate wakeup, the 32KHz oscillator is not forced to a disabled state. If the 32KHz oscillator is disabled or enabled before Hibernate, it remains in the same state after Hibernate wakeup. As long as XPWR32 is brought up before V_{DDXOK} asserts, the 32KHz oscillator enables bit functions as expected. If a system is designed so that XPWR32 is brought up before V_{DDXOK} asserts, software may need to disable and reenable the 32KHz oscillator on first power-up in order for it to run properly. This is because the oscillator itself requires that it be enabled after power to it is valid.

www.DataSheet4U.com

www.DataSheet4U.com

10.4.4.3 Valid Power States

Only certain combinations of power supplies are valid as shown in Table 10-9. In particular, XPWR32 must be among the first power supplies brought up as shown in Figure 10-8; it is not valid to have XPWR32 down while another supply is up. XPWR32 must be brought up before V_{DDXOK} and V_{DDX}, in order for the Hibernate Status bits in **sys_wakesrc** to be predictable. V_{DDXOK} must remain negated until V_{DDX}, V_{DDY}, and XPWR32 are valid. V_{DDI} must come up after V_{DDXOK} asserts.

Table 10-9. Valid Power States

XPWR32	V_{DDX}	V_{DDY}	V_{DDI}	State
off	off	off	off	Power-Down
on	on	on	on	Full Power-on
on	on	x	x	Sleep, $V_{DDXOK} = 1$
on	off	x	x	Hibernate, $V_{DDXOK} = 0$

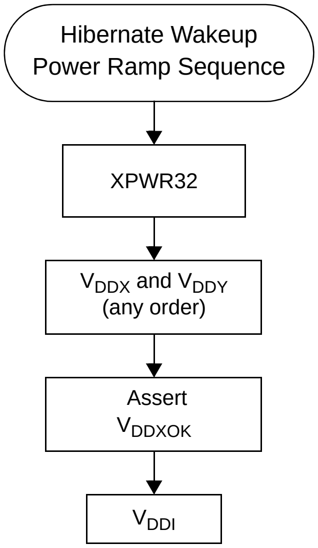


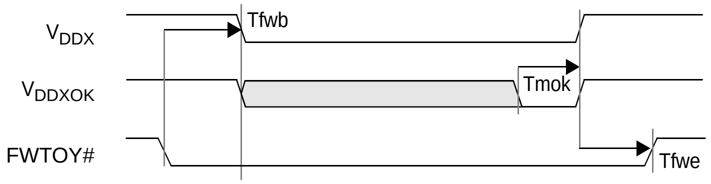
Figure 10-8. Power Ramp Sequence for Hibernate Wakeup

www.DataSheet4U.com

www.DataSheet4U.com

10.4.4.4 FWTOY# and WAKE# Signals

Two signals support Hibernate mode: FWTOY# and WAKE#. FWTOY# is an input signal used to maintain the TOY and associated timers during Hibernate mode. The FWTOY# signal must be asserted before or with V_{DDXOK}, and before V_{DDX} drops. The system can drop V_{DDX} and place the processor in Hibernate at any time, provided it asserts FWTOY# while V_{DDX} is still high. During Hibernate, FWTOY# must be held asserted; V_{DDXOK} may be unpredictable as V_{DDX} floats down. See Figure 10-9.



Tfwb: Begin Hibernate at least 100 µs before 90% of V_{DDX}.
Tmok: V_{DDXOK} negated at least 1 µs before asserting again.
Tfwe: End Hibernate [min = 100 µs, max = 20 ms] after V_{DDXOK} asserts.

Figure 10-9. FWTOY# and the Hibernate Sequence

WAKE# is an output signal that asserts when a TOY match (in **sys_toymatch2**) occurs with Hibernate Timer Match enabled (**sys_wakemsk**[D2] = 1). This is known as a Hibernate wakeup and is essentially the same as a hardware reset. WAKE# remains asserted after a Hibernate wakeup until software clears the wakeup cause (**sys_wakesrc**[M2D]). This can be used to implement an alarm clock feature: put the system into Hibernate and have it wake itself up a fixed time later. WAKE# is a pull-down only, so that the system can wire NOR Hibernate wakeup events from different voltage domains into one signal, with an external pull-up to an appropriate voltage.

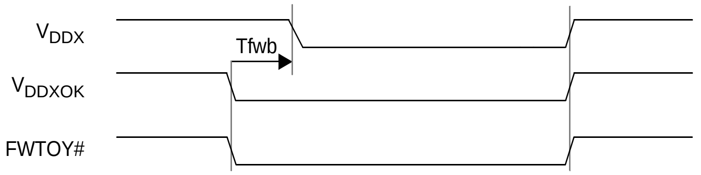
When removing and reasserting XPWR32, such as when swapping out a battery, the value of WAKE# is undefined when the power is reapplied. To avoid power-up at the reinsertion of a battery, the system designer can qualify WAKE# with an external signal.

10.4.4.5 System Configuration for Hibernate

Using Hibernate (FWTOY# = V_{DDXOK})

One design option for a Hibernate system is to use V_{DDXOK} to signal a Hibernate wakeup. In this case, the equation for Hibernate is: FWTOY# = V_{DDXOK}.

Every V_{DDXOK} rise appears as a Hibernate wakeup as shown in Figure 10-10. This ensures the TOY is never reset.



Tfwb: Begin firewall at least 0 µs before V_{DDX}.
When is FWTOY# tied to $V_{DDXOK,}$ FWTOY# is always sampled as asserted at rise of V_{DDXOK}.

Figure 10-10. Hibernate: Tie FWTOY# to V_{DDXOK}

XPWR32 is tied to a supply, such as a battery, which is always up even during Hibernate. For the first rise of V_{DDXOK} after inserting the battery, the burden is on the system to determine that the 32KHz oscillator needs to be enabled and the time needs to be set.

Note that this is not a battery backup system for V_{DDX}; lack of power to XPWR32 prevents the processor from booting even if V_{DDX} is present.

www.DataSheet4U.com

www.DataSheet4U.com

Not Using Hibernate (FWTOY# = V_{DDX})

A system that does not use Hibernate should tie FWTOY# to V_{DDX}, so that FWTOY# remains negated when V_{DDXOK} rises.

10.4.4.6 Software Visibility

Software visible differences due to Hibernate are as follows:

- The oscillator enable **sys_cntrctrl**[EO] is not reset on V_{DDXOK} rise if due to a Hibernate wakeup.
- Software can set **sys_wakemsk**[D2] (Hibernate Match Enable) to trigger the assertion of WAKE# when the TOY timer equals **sys_toymatch2**.
- Hardware sets **sys_wakesrc**[DW] (Hibernate wakeup) if FWTOY# is asserted at the time of V_{DDXOK} rise.
- Additionally, hardware sets **sys_wakesrc**[M2D] (Hibernate Time Match) if a Hibernate wakeup is accompanied by a WAKE# assertion due to a TOY timer match in **sys_toymatch2**. Software must clear **sys_wakesrc**[M2D] (any write).
- Regardless of Hibernate wakeup, the initial power-up bit **sys_wakesrc**[IP] is set on V_{DDXOK} rise.

10.4.5 Power Management Registers

The power management registers and their associated offsets are listed in Table 10-10. These registers are located off of the base shown in Table 10-1 on page 365.

Table 10-10. Power Management Registers

Offset	Register Name	Description	Reset Type
0x0018	sys_scratch0	User-defined register that retains its value through Sleep.	Hardware
0x001C	sys_scratch1	User-defined register that retains its value through Sleep.	Hardware
0x0034	sys_wakemsk	Sets which GPIO or whether TOY match can cause Sleep wakeup	Hardware
0x0038	sys_endian	Sets Big or Little Endian	Hardware & Runtime
0x003C	sys_powerctrl	Sets System Bus divider and power-up time	Mixed (see register description)
0x005C	sys_wakesrc	Gives source of Sleep wakeup	Hardware
0x0078	sys_slppwr	Initiates power state for Sleep mode	Hardware
0x007C	sys_sleep	Initiates Sleep mode	Hardware

www.DataSheet4U.com

www.DataSheet4U.com

10.4.5.1 Scratch Registers

These registers keep their values through Sleep, Hibernate, and runtime resets. These registers allow the system programmer to save user-defined state information or a pointer to a context so that the previous context can be restored when coming out of Sleep, if needed. The scratch registers have unpredictable default values after a hardware reset.

sys_scratch0 **Offset = 0x0018**

sys_scratch1 **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
	SCRATCH[31:0]																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	Read/Write	Default
31:0	SCRATCH	User-defined information.	R/W	UNPRED

10.4.5.2 Wakeup Source Mask Register

For each individual bit that is set, the corresponding signal or event (for the case of the TOY match) can be used to cause a Sleep wakeup. TOY match Hibernate wakeup is enabled with bit D2.

A high level on the enabled GPIO causes the interrupt to trigger.

This register resets to defaults only on a hardware reset. During a runtime reset and during Sleep this register retains its value.

sys_wakemsk **Offset = 0x0034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	▨																						D2	M2	GPIO							
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/Write	Default
31:10	—	Reserved	R	0
9	D2	Setting this bit enables Hibernate wakeup via the WAKE# signal. When the TOY matches the value in the programmable TOY Counter Match Register 2 (**sys_toymatch2**), the WAKE# signal asserts to initiate a Hibernate wakeup. See Section 10.2.1.4 "Match Registers" on page 377.	R/W	0
8	M2	Setting this bit enables the programmable TOY Counter Match Register 2 (**sys_toymatch2**) to cause a Sleep wakeup interrupt. See Section 10.2.1.4 "Match Registers" on page 377.	R/W	0
7:0	GPIO[7:0]	Setting bit *n* causes GPIO[*n*] to cause a Sleep wakeup.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

10.4.5.3 Endianness Register

To change the endianness of the Au1210 and Au1250 processors is a three step process as follows:

1) Program the endianness bit in the system endianness register (**sys_endian**[EN]).

2) Read the **sys_endian** register. (This is required to ensure the final write to the CP0 register updates the endian value.)

3) Read the CP0 register **Config0**. (See Section 2.8.15 "Configuration Register 0 (CP0 Register 16, Select 0)" on page 43.)

4) Write the value read back into the CP0 **Config0** register. The act of writing the CP0 register places the processor into the endian state as programmed in **sys_endian**[EN].

This register, as well as the processor endianness, defaults to big endian after a hardware reset, runtime reset, and after Sleep.

sys_endian **Offset = 0x0038**

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

Bits	Name	Description	Read/ Write	Default
31:1	—	Reserved.	R	UNPRED
0	EN	Endianness. 0 Big Endian 1 Little Endian	R/W	0

10.4.5.4 Power Control Register

Bits [6:5] of this register are reset to default values for a hardware reset, runtime reset, and after Sleep.

Bits [12, 4:0] of this register reset to default values only on a hardware reset. During a runtime reset and during Sleep these bits retain their values.

sys_powerctrl **Offset = 0x003C**

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
																				SSY						SI	SB		VPUT		SD	
Def.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	Read/ Write	Default
31:13	—	Reserved.	R	0
12	SSY	Sleep state for V_{DDY}-domain signals. Takes effect only when the part is in Sleep mode. 0 SDRAM interface signals hold their last values if driven (normal Sleep values). 1 Force the SDRAM interface signals to the values they would take during a hardware reset. The system must bring down V_{DDY} during Sleep when SSY = 1.	RW	0
11:7	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
6	SI	Idle-State System Bus Clock Divider Enable. 0 The idle-state system bus clock divider is disabled. 1 Enable the system bus clock to be divided by an additional factor of 2 when the processor is in an IDLE state (taken through the WAIT instruction). All peripheral bus clocks (such as the SDRAM and UART controllers) are internally compensated with no programmer intervention required.	R/W	0
5	SB	System Bus Clock Divider Enable. 0 The system bus clock divider is disabled. 1 Enable the system bus clock to be divided by an additional factor of 2 when there is no bus activity. All peripheral bus clocks (such as the SDRAM and UART controllers) are internally compensated with no programmer intervention required.	R/W	0
4	—	Reserved.	R	0
3:2	VPUT	V_{DDI} Power-Up Time. 00 20ms 01 5ms 10 100ms 11 2µs	R/W	Hardware Reset 00
1:0	SD	System Bus Clock Divider. 00 2 01 3 10 4 11 Reserved	R/W	Hardware Reset 00

www.DataSheet4U.com

www.DataSheet4U.com

10.4.5.5 Wakeup Cause Register

Before setting the Sleep bit, this register should be cleared. This register retains 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 because they hold their values through Sleep and a runtime reset.

All bits in this register are set by hardware and cleared by any write to this register.

sys_wakesrc **Offset = 0x005C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							M2D	M2S	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0														DW	SW	IP
Def.	0	0	0	0	0	0	Rs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Rs	Rs

Bits	Name	Description	Read/ Write	Default
31:26	—	Reserved.	R/W	0
25	M2D	Programmable TOY Match 2 caused WAKE# to assert. If a TOY match causes WAKE# to assert *and* FWTOY# is asserted on V_{DDXOK} rise, then M2D is set. Otherwise, M2D is cleared. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	Rs
24	M2S	Programmable TOY Match 2 caused wakeup from Sleep. Set by hardware on Sleep wakeup due to TOY match. Cleared by hardware on V_{DDXOK} assertion. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	0
23	GP7	GPIO[*n*] caused wakeup from Sleep. Set by hardware on Sleep wakeup due to GPIO[*n*]. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	0
22	GP6		R/W	0
21	GP5		R/W	0
20	GP4		R/W	0
19	GP3		R/W	0
18	GP2		R/W	0
17	GP1		R/W	0
16	GP0		R/W	0
15:3	—	Reserved.	R/W	0
2	DW	Hibernate Wakeup. This bit is set by hardware on a Hibernate wakeup and cleared by software by a write to this register. On V_{DDXOK} assertion, if (FWTOY# = 1), then hardware clears DW; if (FWTOY# = 0), then hardware sets DW. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	Rs
1	SW	Sleep Wakeup. Valid only when *not* waking from Hibernate; that is, SW is valid only when DW = 0. This bit is set by hardware on a Sleep wakeup and cleared by software by a write to this register. A runtime reset can be detected if both SW and IP are 0 at boot. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	Rs

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	Read/ Write	Default
0	IP	Initial Power-up. Valid only when *not* waking from Hibernate; that is, IP is valid only when DW = 0. This bit is set by hardware on a hardware reset and cleared by software by a write to this register. A runtime reset can be detected if both SW and IP are 0 at boot. This bit must be explicitly cleared by software (any write) because it holds its value through Sleep and runtime reset.	R/W	Rs

10.4.5.6 Sleep Power Register

sys_slppwr **Offset = 0x0078**

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
	SP																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	Read/ Write	Default
31:0	SP	Sleep Power. A write to this register prepares the internal power supply for going to sleep.	W	UNPRED

10.4.5.7 Sleep Register

sys_sleep **Offset = 0x007C**

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

Bits	Name	Description	Read/ Write	Default
31:0	SL	Sleep. A write to this register puts system to sleep.	W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

11 Power-up, Reset and Boot

This section presents the power-up, hardware reset, and runtime reset sequence for the Au1210™ and Au1250™ processors. In addition, the boot vector is described.

11.1 Power-up Sequence

The Au1210 and Au1250 processors' power structure is designed such that the external I/O voltage (V_{DDX} and V_{DDY}) is driven separately from the core voltage (V_{DDI}). In this way, the core voltage can be sourced at lower voltages, saving power. In addition the processors are designed to allow the system designer to remove the core voltage during Sleep to maximize power efficiency.

Two signals, V_{DDXOK} and PWR_EN, are used to facilitate this power strategy. V_{DDXOK} is used as a signal to the processor that the power level on V_{DDX} is stable. Stable is defined as having reached 90% of its nominal value. PWR_EN is an output from the Au1210 and Au1250 that is asserted after V_{DDXOK} is asserted and can be used as an enable to the regulator that is providing the core voltage, V_{DDI}.

The following describes the power-up sequence for the Au1210 and Au1250 processors:

1) Apply V_{DDX} and V_{DDY} (I/O power).

2) When V_{DDX} and V_{DDY} have reached 90% of nominal, assert V_{DDXOK}.

3) The processors then assert PWR_EN, which can be used to enable the regulator driving V_{DDI} (CPU power).

Figure 11-1 shows the power-up sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 14.5.1 "Power-up Sequence Timing" on page 467.

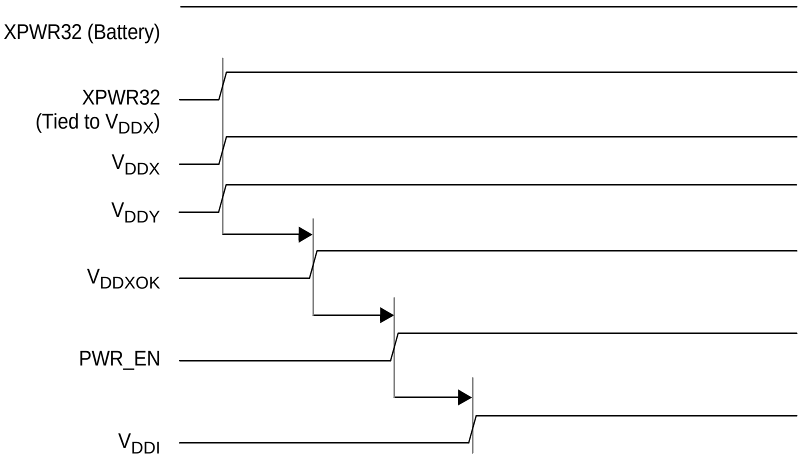


Figure 11-1. Power-up Sequence

www.DataSheet4U.com

www.DataSheet4U.com

11.2 Reset

A hardware reset is defined as a reset in which both V_{DDXOK} and RESETIN# are toggled. Typically this happens only at power-on, but a system designer can choose to tie V_{DDXOK} and RESETIN# together, in which case, all resets are hardware resets.

For a *runtime* reset, power remains applied and only the RESETIN# signal is toggled. Certain registers, specifically some registers in the system control block, are not affected by this type of reset. See the register description for the register in question for more information. If a register is not reset to defaults by both hardware reset and runtime reset, it is noted in the register description.

11.2.1 Hardware Reset

For a hardware reset, V_{DDXOK} makes a transition from low to high followed by RESETIN# negating (transitioning from low to high). The following sequence describes a hardware reset:

1) The BOOT[1:0] signals 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 V_{DDXOK} is asserted, RESETIN# can be negated. In other words, RESETIN# cannot be negated before V_{DDXOK} is asserted. This allows V_{DDXOK} and RESETIN# to be tied together.

3) RESETOUT# is negated *after* RESETIN# is negated.

Figure 11-2 shows the hardware reset sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 14.5.2 "Hardware Reset Timing" on page 468.

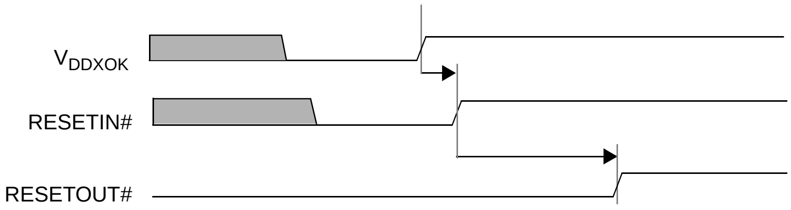


Figure 11-2. Hardware Reset Sequence

www.DataSheet4U.com

www.DataSheet4U.com

11.2.2 Runtime Reset

During runtime (after power is stable), the reset sequence can be broken down as follows:

1) During a runtime reset it is assumed that V_{DDX} and V_{DDI} remain at their nominal voltage. In addition, V_{DDXOK} must remain asserted; otherwise, a hardware reset occurs. PWR_EN remains asserted by the Au1210 and Au1250 processors.

2) RESETIN# is held asserted long enough to be recognized as a valid reset.

3) The processor acknowledges the reset by asserting RESETOUT#.

4) After RESETIN# is released, the processor signals the end of the reset by negating RESETOUT#.

Certain registers (specifically those in the system control block) are not affected by a runtime reset. Also, the BOOT[1:0] signals must already be terminated in the design so the appropriate boot type occurs—these values must not change during runtime.

Figure 11-3 shows the runtime reset sequence, including arrows representing causal dependencies. For the timing specifications of this sequence, refer to Section 14.5.3 "Runtime Reset Timing" on page 469.

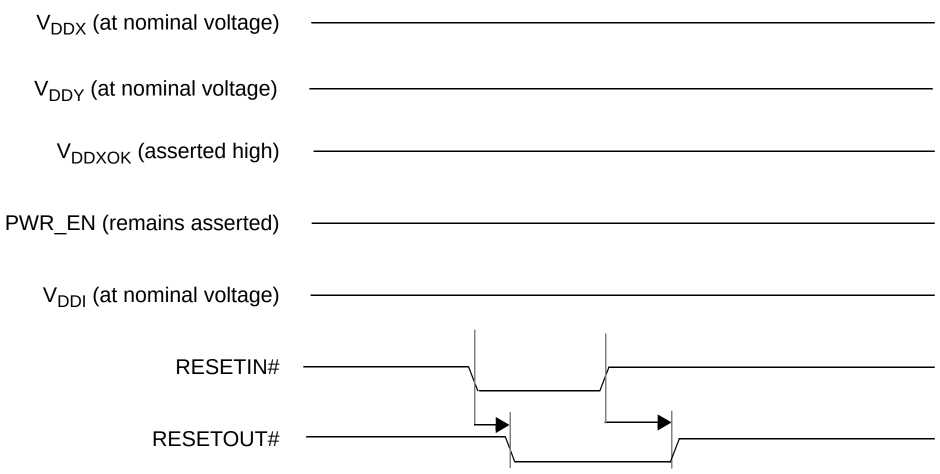

V_{DDX} (at nominal voltage)
V_{DDY} (at nominal voltage)
V_{DDXOK} (asserted high)
PWR_EN (remains asserted)
V_{DDI} (at nominal voltage)
RESETIN#
RESETOUT#

Figure 11-3. Runtime Reset Sequence

11.3 Boot

For both hardware and runtime resets, the CPU boots from KSEG1 address 0xBFC00000, which is translated to physical address 0x1FC00000; therefore, the system designer must place the start of the boot code at 0x1FC00000.

The BOOT signals determine where the processor boots from, as shown in Table 11-1. The BOOT signal states can be read by system software. See Section 3.2.1.6 "Static Bus Status Register (mem_ststat)" on page 89.

Table 11-1. Boot Device Selection (BOOT[1:0])

BOOT[1]	BOOT[0]	Boot Device	Processor
0	0	16-bit ROM on RCS0# of the static bus controller.	Au1200, Au1210, Au1250
0	1	8-bit NAND on RCS0# of the static bus controller.	Au1200, Au1210, Au1250
1	0	16-bit NAND on RCS0# of the static bus controller.	Au1200, Au1210, Au1250
1	1	Reserved	Au1200, Au1210, Au1250
RESETOUT#	1	8-bit Large block NAND on RCS0# of the static bus controller	Au1210 and Au1250 only
1	RESETOUT#	16-bit Large block NAND on RCS0# of the static bus controller	Au1210 and Au1250 only

www.DataSheet4U.com

www.DataSheet4U.com

11.3.1 Endianness and 16-Bit Static Bus Boot

When booting from the static bus, the system designer must be sure the data format (endianness) is consistent across the Au1 core, the static bus controller, and the software image itself. This section describes how to make endianness consistent for both little- and big-endian systems.

For more on how the endian mode affects the behavior of static bus chip selects, see Section 3.2.8.3 "Halfword Ordering" on page 111.

Note: When programming ROM or Flash devices with a part programmer, take care to ensure that the programmer is not swapping bytes or halfwords erroneously. The configuration of the part programmer is often a source of error when initially bringing-up a new design.

11.3.1.1 16-Bit Boot for Little-Endian System

Booting from 16-bit ROM or Flash in a system that is intended to run the Au1 core in little-endian mode is straightforward. For this example system, the boot code and/or the application is compiled for little-endian. Because the Au1 core defaults to big-endian mode, the boot code must change the Au1 core endianness to little-endian *before* any data accesses (to the 16-bit chip-select). The resulting boot code and/or application image is placed in the ROM/Flash memory in the little-endian format.

Even though the Au1 core starts in big-endian mode, the static bus controller properly retrieves instructions needed to boot the system since the application image is in little-endian format and the static bus controller defaults to little-endian ordering out of reset.

11.3.1.2 16-Bit Boot for Big-Endian System

Booting from 16-bit ROM or Flash in a system that is intended to run the Au1 core in big-endian mode requires one extra step compared to the little-endian system example described above.

For this example system, the boot code and/or the application is compiled for big-endian. The boot code must set the **mem_stcfg**[BE] bit before it can properly fetch/reference the big-endian image. The resulting boot code and/or application image is placed in the ROM/Flash memory in the big-endian format.

Coming out of reset, the static bus controller defaults to little-endian ordering, but the application image itself is in big-endian format. In this situation, place the following code at the reset exception vector (KSEG1 address 0xBFC00000, physical address 0x1FC00000):

```
.long 0xb4003c08 # lui     t0,0xb400
.long 0x10003508 # ori     t0,t0,0x1000
.long 0x00008d09 # lw      t1,0(t0)
.long 0x02003529 # ori     t1,t1,0x200
.long 0x0000ad09 # sw      t1,0(t0)
.long 0x00000000 # nop
.long 0x00000000 # nop
.long 0x00000000 # nop
.long 0x00000000 # nop
```

The code does a read-modify-write of register **mem_stcfg0** to set the BE bit. The values in the .long statements above are the halfword-swapped opcodes of the instructions in the comments to the right. With this technique, these first few instructions are actually in the little-endian format to match the static bus controller out of reset. The code sets **mem_stcfg0**[BE], which in turn allows the remainder of the big-endian memory contents to be accessed properly. The NOPs are necessary to ensure that the Au1 core pipeline does not contain incorrectly prefetched halfword-swapped instructions. Note too that the NOP opcode 0x00000000 is the same instruction regardless of endian ordering.

Note: The boot code should set **mem_stcfg0**[BE] as early as possible, preferably as the first activity. Software must ensure that no cacheable accesses to the 16-bit device take place; otherwise, the cache contains the halfword-swapped contents of the 16-bit memory.

11.3.2 System Debug

For system debug, the processor can be configured to boot from the EJTAG probe through the EJTAG port. See Section 12.0 "EJTAG Implementation" on page 403 for more information.

www.DataSheet4U.com

www.DataSheet4U.com

12 EJTAG Implementation

The Au1210™ and Au1250™ processors implement EJTAG following the MIPS EJTAG 2.5 specification. This section presents the EJTAG implementation on the Au1210 and Au1250 processors while concentrating on those features from the EJTAG 2.5 specification which are implementation specific. In addition, those features that have not been implemented or any differences in the Au1210 and Au1250 processors' implementation of EJTAG from the EJTAG 2.5 specification are noted.

Refer to the EJTAG 2.5 specification for implementation details not covered here. If a particular bit is not implemented, the functionality associated with the bit is not implemented or not applicable unless otherwise noted.

The following features comprise the EJTAG implementation on the Au1210 and Au1250 processors:

- 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 (specific to Au1210 and Au1250)
- Processor bus breakpoints (from EJTAG 2.0)
- Memory overlay (from EJTAG 2.0)
- EJTAG tap per IEEE1149.1
- Debug exceptions from CP0 watch registers

The optional data and instruction hardware breakpoint features from the EJTAG 2.5 specification are not implemented.

12.1 EJTAG Instructions

Both SDBBP and DERET are supported by the Au1210 and Au1250 processors:

- SDBBP causes a Debug Breakpoint exception.
- DERET is used to return from a Debug Exception.

12.2 Debug Exceptions

The following exceptions cause the processor to enter debug mode.

- DSS - debug single step
- DINT - debug interrupt, processor bus break
- DBp - execution of SDBBP instruction
- DWATCH - debug watch exception. The processor-specific implementation allowing a CP0 watch exception to cause a debug exception. See description of the Section 12.4.2.6 "EJWatch Register (TAP Instruction EJWATCH)" on page 418.

Other normal exceptions, when taken in debug mode, are handled by the debug exception handler.

www.DataSheet4U.com

w w w . D a t a S h

12.3 Coprocessor 0 Registers

The Coprocessor 0 Registers for EJTAG are shown in Table 12-1.

Table 12-1. Coprocessor 0 registers for EJTAG

Register Number	Select	Name	Description
23	0	debug	Debug indications and controls for the processor
24	0	depc	Program Counter at last debug exception or exception in debug mode
31	0	desave	Debug exception save register

12.3.1 Debug Register (CP0 Register 23, Select 0)

This register contains the cause of the most recent debug exception and exception in Debug Mode. It also controls single stepping. Only the DM bit and the EJTAGver field are valid when read from the Debug register in Non-Debug Mode. The value of all other bits and fields is UNPREDICTABLE.

The following bits and fields are updated only on debug exceptions and/or exceptions in Debug Mode:

- DSS, DBp, DINT are updated on both debug exceptions and on exceptions in Debug Modes.
- DExcCode is updated on normal exceptions in Debug Mode, and is undefined after a debug exception.
- DD is updated on both debug and on normal exceptions in Debug Modes.

debug **CP0 Register 23, Select 0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DD	DM	ND	LS	▨		CD	▨							VER			DEXCODE					NS	SS	▨		DI	▨			DB	DS
Def.	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	0	0	0	0	X	0	0	0	X	X

Bits	Name	Description	R/W	Default
31	DD	DBD. Indicates whether the last debug exception or exception in Debug Mode occurred in a branch or jump delay slot. 0 Not in delay slot 1 In delay slot	R	UNPRED
30	DM	Indicates that the processor is operating in Debug Mode. 0 Processor is operating in Non-debug Mode 1 Processor is operating in Debug Mode	R	0
29	ND	NoDCR. 0 drseg is present.	R	0
28	LS	LSNM. Controls access of loads/stores between drseg and remaining memory when drseg is present and while in Debug Mode. 0 Loads/stores in drseg address range go to drseg 1 Loads/stores in drseg address range go to system memory	R/W	0
27	—	Reserved. *This bit is called Doze in the EJTAG 2.5 specification and is not implemented*.	R	0
26	—	Reserved. *This bit is called Halt in the EJTAG 2.5 specification and is not implemented*.	R	0
25	CD	CountDM. 0 Count register stops in Debug Mode.	R	0
24	—	Reserved. *This bit is called IBusEP in the EJTAG 2.5 specification and is not implemented*.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
23	—	Reserved. *This bit is called MCheckP in the EJTAG 2.5 specification and is not implemented*.	R	0
22	—	Reserved. *This bit is called CacheEP in the EJTAG 2.5 specification and is not implemented*.	R	0
21	—	Reserved. *This bit is called DBusEP in the EJTAG 2.5 specification and is not implemented*.	R	0
20	—	Reserved. *This bit is called IEXI in the EJTAG 2.5 specification and is not implemented*.	R	0
19	—	Reserved. *This bit is called DDBSImpr in the EJTAG 2.5 specification and is not implemented*.	R	0
18	—	Reserved. *This bit is called DDBLImpr in the EJTAG 2.5 specification and is not implemented*.	R	0
17:15	VER	EJTAGver. 1 EJTAG Version 2.5	R	1
14:10	DEXCODE	DExcCode. Indicates the cause of the latest exception in Debug Mode. The field is encoded as the ExcCode field in the Cause register for those exceptions that can occur in Debug Mode (the encoding is shown in the MIPS32 specification), with addition of code 30 with the mnemonic CacheErr for cache errors. This value is undefined after a debug exception.	R	UNPRED
9	NS	NoSSt. 0 Single step is implemented.	R	0
8	SS	SSt. Controls whether the single-step feature is enabled: 0 Disable single-step feature. 1 Enable single-step feature.	R/W	0
7:6	—	Reserved.	R	0
5	DI	DINT. Indicates that a Debug Interrupt exception occurred. This could be either a Processor Bus Break (indicated by BS0 in the Processor Bus Break Status Register) or EJTAG break. The BS0 bit should be checked to see what caused the exception. Cleared on exception in Debug Mode. 0 No Debug Interrupt exception 1 Debug Interrupt exception	R	UNPRED
4	—	Reserved. *This bit is called DIB in the EJTAG 2.5 specification and is not implemented*.	R	0
3	—	Reserved. *This bit is called DDBS in the EJTAG 2.5 specification and is not implemented*.	R	0
2	—	Reserved. *This bit is called DDBL in the EJTAG 2.5 specification and is not implemented*.	R	0
1	DB	DBp. Indicates that a Debug Breakpoint exception occurred. Cleared on exception in Debug Mode. 0 No Debug Breakpoint exception 1 Debug Breakpoint exception	R	UNPRED
0	DS	DSS. Indicates that a Debug Single Step exception occurred. Cleared on exception in Debug Mode. 0 No debug single-step exception 1 Debug single-step exception	R	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

12.3.1.1 Debug Exception Program Counter Register

This 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 (DD) 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 **CP0 Register 24, Select 0**

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

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

12.3.1.2 Debug Exception Save Register

This 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 predetermined 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 **CP0 Register 31, Select 0**

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

Bits	Name	Description	R/W	Default
31:0	DESAVE[31:0]	Debug Exception Save contents.	R/W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

12.4 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

The physical address *sysbus_addr*[35:32] of this range is zero.

Dmseg is the memory range 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.

12.4.1 EJTAG Memory Mapped Registers

Table 12-2 shows the EJTAG memory mapped registers located in drseg.

Table 12-2. EJTAG Memory Mapped Registers at 0xFF300000

Offset	Register	Description
0x0000	ejtag_dcr	Debug Control Register
0x000C	ejtag_pbs	Processor Bus Break Status Register
0x0300	ejtag_pab	Processor Address Bus Break
0x0304	ejtag_pdb	Processor Data Bus Break
0x0308	ejtag_pdm	Processor Data Mask/Upper Overlay Address Mask
0x030C	ejtag_pbcam	Processor Bus Break Control and Address Mask
0x0310	ejtag_phab	Processor High Address Bus Break
0x0314	ejtag_pham	Processor High Address Mask

The EJTAG implementation in the Au1210 and Au1250 processors 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 a debug exception for a given physical address and data.

In addition, the implementation allows the CP0 watchpoints to cause a debug exception. This functionality is enabled through the EJTAG TAP port. See Section 12.4.2.6 "EJWatch Register (TAP Instruction EJWATCH)" on page 418 for details.

12.4.1.1 Debug Control Register

This register provides information about debug issues. The width of the register is 32 bits. The **ejtag_dcr** register is located in the drseg at offset 0x0000.

ejtag_dcr **Offset = 0x0000**

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

Bits	Name	Description	R/W	Default
31:30	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
29	EN	ENM. Displays the current endian mode of the processor. It corresponds to **sys_endian**[EN]. 0 Processor is little-endian in both debug and kernel mode. 1 Processor is big-endian in both debug and kernel mode.	R	1
28:18	—	Reserved.	R	0
17	DB	DataBrk. 0 No data hardware breakpoints implemented.	R	0
16	IB	InstBrk. 0 No instruction hardware breakpoints implemented.	R	0
15:5	—	Reserved.	R	0
4	IE	IntE. 1 Interrupt enabled in non-debug mode depending on other enabling mechanisms.	R	1
3	NE	NMIE. 1 Non-Maskable Interrupt is enabled for non-debug mode. The NMI is not implemented and does not apply to the Au1210 and Au1250 processors.	R	1
2	NP	NMIPend. 0 No NMI pending The NMI is not implemented and does not apply to the Au1210 and Au1250 processors.	R	0
1	SR	SRstE. 1 Soft reset is fully enabled. Soft reset is not implemented and does not apply to the Au1210 and Au1250 processors.	R	1
0	PE	ProbEn. Indicates value of the ProbEn value in the ECR register. 0 No access should occur to dmseg 1 Probe services accesses to dmseg	R	Same value as ProbEN in ECR

12.4.1.2 Processor Bus Break Status Register

ejtag_pbs **Offset = 0x000C**

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

Bits	Name	Description	R/W	Default
31	—	Reserved.	R	0
30	OLP	1 Memory Overlay Functionality is Implemented for Processor breaks.	R	1
29:28	—	Reserved.	R	0
27:24	BCN	Number of Processor Breaks. 1 One Channel has been implemented for the Processor Bus Break.	R	1
23:15	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
14:1	—	Reserved. These bits are the Bsn bits in the EJTAG 2.0.0 specification and are not available since only one break is implemented.	R	0
0	BS	Break Status. 0 Normal operation 1 A processor bus break has occurred. The debug handler must clear this bit before returning from debug mode. BS is cleared by activating PrRst (EJTAG CONTROL Register), hard reset, or by writing a ‘0’ to it.	R/W	0

12.4.1.3 Processor Address Bus Break

This register contains the lower 32 bits of the physical Processor Address Bus Break.

ejtag_pab **Offset = 0x0300**

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
	PAB																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	PAB[31:0]	Processor Address Bus Break 0. This index contains the lower 32 bits of the physical address. In combination with the high order address bits in ejtag_phab, these bits make up the break address.	R	UNPRED

12.4.1.4 Processor Data Bus Break

This register contains the data value for the Processor Data Bus match.

ejtag_pdb **Offset = 0x0304**

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
	PDB																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:0	PDB[31:0]	Processor Data Bus Break 0. This index contains the 32 bits of the data bus match.	R	UNPRED

12.4.1.5 Processor Data Mask/Upper Overlay Address Mask

This register is dual purpose depending on the value of the Overlay Enable bit in the Bus Break Control and Address Mask register (**ejtag_pbcam**[OE]).

This register specifies the mask value for the Processor Data Mask register. Each bit corresponds to a bit in the data register.

ejtag_pdm (ejtag_pbcam[OE] = 0) **Offset = 0x0308**

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
	PDM																															
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

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
31:0	PDM[31:0]	Processor Data Mask. 0 Data bit is not masked, data bit is compared. 1 Data bit is masked, data bit is not compared.	R/W	UNPRED

ejtag_pdm (ejtag_pbcam[OE] = 1) **Offset = 0x0308**

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
	UOAM																															
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
31:24	UOAM	Upper Overlay Address Mask. These bits represent bits [31:24] of the address mask and are combined with the LAM and HAM fields to create a complete 36 bit address mask. 0 Address bit is not masked, address bit is compared. 1 Address bit is masked, address bit is not compared.	R/W	UNPRED
23:0	—	Reserved.	—	—

12.4.1.6 Processor Bus Break Control and Address Mask

This register selects the processor bus match function to enable debug break or trace trigger. It also includes control bits to enable comparison as well as mask bits to exclude address bits from comparison. All processor break exceptions are imprecise.

ejtag_pbcam **Offset = 0x030C**

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
	LAM																								DC	DU	DIU		OE			BE
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	0	0	0	0

Bits	Name	Description	R/W	Default
31:8	LAM	Address Mask. These bits specify the mask value for the lower 24 bits of the processor address register (**ejtag_pab**[23:0]). Each bit corresponds to the same bit in **ejtag_pab**. 0 Address bit is not masked, address bit is compared. 1 Address bit is masked, address bit is not compared.	R/W	UNPRED
7	DC	Data Store to Cached Area. This bit enables the comparison on processor address and data bus for data store to the cached area. 0 Processor address and data is not compared for storing data to the cached area. 1 Processor address and data is compared for storing data to the cached area.	R/W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
6	DU	Data Store to Uncached Area. This bit enables the comparison on processor address and data bus for data store to the uncached area. 0 Processor address and data is not compared for storing data into the un-cached area. 1 Processor address and data is compared for storing data into the un-cached area.	R/W	UNPRED
5:4	DIU	Data or Instruction Fetch or Load From Uncached Area. These bits enable the comparison on processor address and data bus for data or instruction load and fetch from the un-cached area. 00 Processor address and data is not compared for loading data or fetching instructions from the un-cached area. 11 Processor address and data is compared for loading data or fetching instruction from the un-cached area. Bits 5 and 4 were named ILUC and DFUC in the EJTAG 2.0.0 specification and were implemented separately for instruction and data fetches.	R	UNPRED
3	OE	Overlay Enable. When this bit is 1 and the processor physical address, masked by the HAM, UOAM and the LAM fields (all 36 bits of the address mask), matches the PHAB and PAB registers, then the memory request is redirected to the EJTAG Probe. When OE is set, the processor bus break cannot be used for the normal break function. Therefore, BE (break-enable) must be cleared; otherwise, the behavior is undefined. The overlay feature is valid only for memory regions. It is not valid for I/O or debug space, and the behavior is unpredictable if addresses within this space are used.	R/W	0
2	—	Reserved. *This bit is called TE in the EJTAG 2.0.0 specification and is not implemented*.	R	0
1	—	Reserved. *This bit is called CBE in the EJTAG 2.0.0 specification and is not implemented*.	R	0
0	BE	Break Enable. This bit enables the processor bus break function. 0 Processor bus break function is disabled. 1 Processor bus break function is enabled. If BE 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. 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 is not taken until DM = 0. This functionality is mutually exclusive to OE so only one of OE or BE must be set at any time.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

12.4.1.7 Processor High Address Bus Break

This register specifies the high order physical address for the processor address bus break.

ejtag_phab **Offset = 0x0310**

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

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3:0	HA[3:0]	High Address Bits. The HA[3:0] bits map to the high physical address bits [35:32].	R/W	UNPRED

12.4.1.8 Processor High Address Mask

This register specifies the high order address mask for the processor address bus break.

ejtag_pham **Offset = 0x0314**

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

Bits	Name	Description	R/W	Default
31:4	—	Reserved.	—	—
3:0	HAM[3:0]	High Address Mask for Address Bits [35:32]. 0 Data bit is not masked, data bit is compared 1 Data bit is masked, data bit is not compared	R/W	UNPRED

www.DataSheet4U.com

w w w . D a t a S h e e t

12.4.2 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 12-3 shows the 5-bit instructions supported by the Au1210 and Au1250 processors.

Table 12-3. EJTAG Instruction Register Values

Hex Value	Instruction	Function
0x00	EXTEST	Boundary Scan
0x01	IDCODE	Selects ID Register
0x02	SAMPLE	Boundary Scan Sample/Preload (IEEE JTAG Instruction)
0x03	IMPCODE	Selects Implementation Register
0x04	—	Reserved
0x05	HIZ	Tristate all output pins and Select Bypass register.
0x06	—	Reserved
0x07	—	Reserved
0x08	ADDRESS	Selects Address Register.
0x09	DATA	Selects Data Register.
0x0A	CONTROL	Selects EJTAG Control Register.
0x0B	ALL	Selects the Address, Data and EJTAG Control registers.
0x0C	EJTAGBOOT	Makes the processor take a debug exception after reset.
0x0D	NORMALBOOT	Makes the processor execute the reset handler after reset.
0x0E-0x1B	—	Reserved
0x1C	EJWATCH	Selects Watch register
0x1D-0x1E	—	Reserved
0x1F	BYPASS	Bypass mode

12.4.2.1 Device Identification (ID) Register

This register is a 32-bit read only register that identifies the specific device implementing EJTAG.

IDCODE **TAP Instruction IDCODE**

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

Bits	Name	Description	R/W	Default
31:28	VER	Identifies the version of the device.	R	SoC specific
27:12	PNUM	Identifies the part number of the device.	R	Au1210: 0x0502 Au1250: 0x0402
11:1	MANID	Identifies the manufacturer ID code for the device. MANID[6:0] are derived from the last byte of the JEDEC code with the parity bit discarded. MANID[10:7] provides a binary count of the number of bytes in the JEDEC code that contain the continuation character (0x7F). When the number of continuations characters exceeds 15, these four bits contain the modulo-16 count.	R	0x147
0	—	Reserved.	R	1

www.DataSheet4U.com

www.DataSheet4U.com

12.4.2.2 Implementation Register

This register is a 32-bit read only register that identifies features implemented in this EJTAG compliant processor, mainly those accessible from the TAP.

IMPCODE **TAP Instruction IMPCODE**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VER			R3				DI		AS						M16		ND														M32
Def.	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31:29	EJTAGver	1 EJTAG version 2.5.	R	1
28	R3	0 R3k privileged environment.	R	0
27:25	—	Reserved.	R	0
24	DI	0 DINT signal from the probe is not supported.	R	0
23	—	Reserved.	R	0
22:21	AS	10 8-bit ASID.	R	0x2
20:17	—	Reserved.	R	0
16	M16	0 No MIPS16 support.	R	0
15	—	Reserved.	R	0
14	ND	1 No EJTAG DMA support.	R	1
13:1	—	Reserved.	R	0
0	M32	MIPS32/64. 0 32-bit processor	R	0

12.4.2.3 Data Register

This register is used for opcode and data transfers during processor accesses. The width of the Data register is 32 bits.

The value read in the Data register is valid only if a processor access for a write is pending, in which case the Data register holds the store value. The value written to the Data register is only used if a processor access for a pending read is finished afterwards, in which case, the data value written is the value for the fetch or load. This behavior implies that the Data register is not a memory location where a previously written value can be read afterwards.

DATA **TAP Instruction DATA or ALL**

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

Bits	Name	Description	R/W	Default
31:0	DATA[31:0]	Data used by processor access.	R/W	UNPRED

www.DataSheet4U.com

www.DataSheet4U.com

12.4.2.4 Address Register

This 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 least-significant bits 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	35	34	33	32	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
	ADDRESS																																			
Def.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Bits	Name	Description	R/W	Default
35:0	ADDRESS[35:0]	Address used by processor access.	R	UNPRED

12.4.2.5 EJTAG Control Register (ECR)

This register handles processor reset, Debug Mode indication, access start, finish, and size and read/write indication. The ECR also:

- Controls debug vector location and indication of serviced processor accesses.
- Allows debug interrupt request.
- Indicates processor low-power mode.

The EJTAG Control register is not updated/written in the Update-DR state unless the Reset occurred; that is RO (bit 31) is either already 0 or is written to 0 at the same time. This condition ensures proper handling of processor accesses after a reset.

Bits that are R/W in the register return their written value on a subsequent read, unless other behavior is defined. Internal synchronization hardware thus ensures that a written value is updated for reading immediately afterwards, even when the TAP controller takes the shortest path from the Update-DR to Capture-DR state.

Note: To ensure a write is successful to the PE, PT and EB bits when the processor is undergoing a clock change (for PLL lock/relock), the host must continue writing these bits until the write is verified by reading the change. Failure to do this could result in the write of these bits being lost.

Reset of the processor can be indicated in the TCK domain a number of TCK cycles after it is removed in the processor clock domain in order to allow for proper synchronization between the two clock domains.

ECR **TAP Instruction CONTROL or ALL**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RO	PSZ								DZ			PRW	PA		PR	PE	PT		EB									DM			
Def.	1	X	X	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
31	RO	Indicates if a processor reset has occurred since the bit was cleared. 0 No reset occurred 1 Reset occurred The RO bit stays set as long as reset is applied. This bit must be cleared to acknowledge that the reset was detected. The EJTAG Control register is not updated in the Update-DR state unless RO is 0 or written to 0 at the same time. This is in order to ensure correct handling of the processor access after reset.	R/W0	1

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
30:29	PSZ	Indicates the size of a pending processor access, in combination with the Address register. 00 Byte 01 Halfword 10 Word 11 Triple This field is valid only when a processor access is pending; otherwise, the read value is undefined.	R	UNPRED
28:23	—	Reserved.	R	0
22	DZ	Doze. Indicates if the processor is in a WAIT state. 0 Processor is not in a wait state. 1 Processor is in a wait state.	R	0
21	—	Reserved. *This bit is called Halt in the EJTAG 2.5 specification and is not implemented*.	R	0
20	—	Reserved. *This bit is called PerRst in the EJTAG 2.5 specification and is not implemented*.	R	0
19	PRW	Indicates read or write of a pending processor access. 0 Read processor access, for a fetch/load access 1 Write processor access, for a store access This value is defined only when a processor access is pending.	R	UNPRED
18	PA	Indicates a pending processor access and controls finishing of a pending processor access. When read: 0 No pending processor access 1 Pending processor access A write of 0 finishes a processor access if pending; otherwise operation of the processor is UNDEFINED if the bit is written to 0 when no processor access is pending. A write of 1 is ignored.	R/W0	0
17	—	Reserved.	R	0
16	PR	Controls the processor reset. 0 No processor reset applied 1 Processor reset applied Setting this bit applies a processor reset. When this bit is read back it always reads as 0. Startup latencies should be observed when applying reset.	R/W	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
15	PE	Controls indication to the processor of whether the probe expects to handle accesses to EJTAG memory through servicing of processor accesses. 0 Probe does not service processor accesses 1 Probe services processor accesses The PE bit is reflected as a read only bit in the Debug Control register (DCR) bit 0. When a read from this bit shows a change, the new value has taken effect in the DCR. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain. However, a change of the PE before setting the EB bit is ensured to affect execution of the debug handler due to the debug exception. The combination PE = 0 and PT = 1 is *not* allowed. See the previous note about writing this bit (in Section 12.4.2.5 on page 415).	R/W	Determined by EJTAG-BOOT
14	PT	Controls location of the debug exception vector. 0 Normal memory 0xBFC0 0480 1 EJTAG memory 0xFF20 0200 When a read from this bit shows a change, the new value has taken effect in the DCR. This handshake mechanism ensures that the setting from the TCK clock domain takes effect in the processor clock domain. However, a change of the PT before setting the EB bit is ensured to affect execution of the debug handler due to the debug exception. The combination PE = 0 and PT = 1 is *not* allowed. See the previous note about writing this bit (in Section 12.4.2.5 on page 415).	R/W	Determined by EJTAG-BOOT
13	—	Reserved.	R	0
12	EB	Requests a debug interrupt exception to the processor when this bit is written as 1. This bit is cleared by hardware when the processor enters debug mode. If software then sets EB while the processor is already in debug, the request is not ignored but is delayed. That is, once the processor returns to normal mode, the pending debug exception request immediately sends the processor back into debug. The debug request restarts the processor clock if the processor was in idle mode due to a **WAIT** instruction, which stopped the processor clock. The read value indicates a pending Debug Interrupt exception requested through this bit. 0 When reading: no pending Debug Interrupt exception requested through this bit. A write of 0 is ignored. 1 When reading: Pending Debug Interrupt exception. When writing: Request an debug Interrupt exception. See the previous note about writing this bit (in Section 12.4.2.5 on page 415).	R/W	Determined by EJTAG-BOOT
11:4	—	Reserved.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

Bits	Name	Description	R/W	Default
3	DM	Indicates if the processor is in Debug Mode. 0 Processor is in Non-Debug Mode 1 Processor is in Debug Mode	R	0
2:0	—	Reserved.	R	0

12.4.2.6 EJWatch Register (TAP Instruction EJWATCH)

This register is used to enable CP0 watchpoints to cause a debug exception. This functionality is unique to the Au1210 and Au1250 processors. When watchpoints are in Debug Watch mode:

- Writes to Watch register are blocked in Non-debug Mode.
- Watch exceptions become debug exceptions with DEXCODE = 23.
- The PC is saved in the DEPC, not in the EPC as in normal Watch Exception Mode.

The Status, Cause, and EPC are not affected by a debug watch exception when Debug Watch Mode is enabled.

EJWATCH **TAP Instruction EJWATCH**

Bit	7	6	5	4	3	2	1	0
								WE
Def.	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W	Default
7:1	—	Reserved.	R	0
0	WE	This bit controls the debug functionality of the CP0 watch register. 0 Normal Watch Exception mode 1 Debug Watch Exception mode	R/W	0

12.4.2.7 Bypass Register (TAP Instruction BYPASS)

This register is a one-bit read only register that provides a minimum shift path through the TAP. This register is also defined in IEEE 1149.1.

BYPASS **TAP Instruction BYPASS**

Bit	0
	BP
Def.	0

Bits	Name	Description	R/W	Default
0	BP	Ignored on writes; returns zeros on reads.	R	0

www.DataSheet4U.com

www.DataSheet4U.com

12.4.3 EJTAG TAP Hardware Considerations

The EJTAG interface consists of the signals listed in Table 12-4.

Table 12-4. EJTAG Signals

Signal	Input/Output	Definition
TRST#	I	Asynchronous TAP reset.
TDI	I	Test data input to the instruction or selected data registers. This signal is sampled on the rising edge of TCK.
TDO	O	Test data output from the instruction or data register. This signal transitions on the falling edge and is valid on the rising edge of TCK.
TMS	I	Control signal for TAP controller. This signal is sampled on the rising edge of TCK.
TCK	I	Control clock for updating TAP controller and shifting data through instruction or selected data register.

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 specification must be followed.

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

13 Signal Descriptions

This section describes the external signals on the Au1210™ and Au1250™ processors.

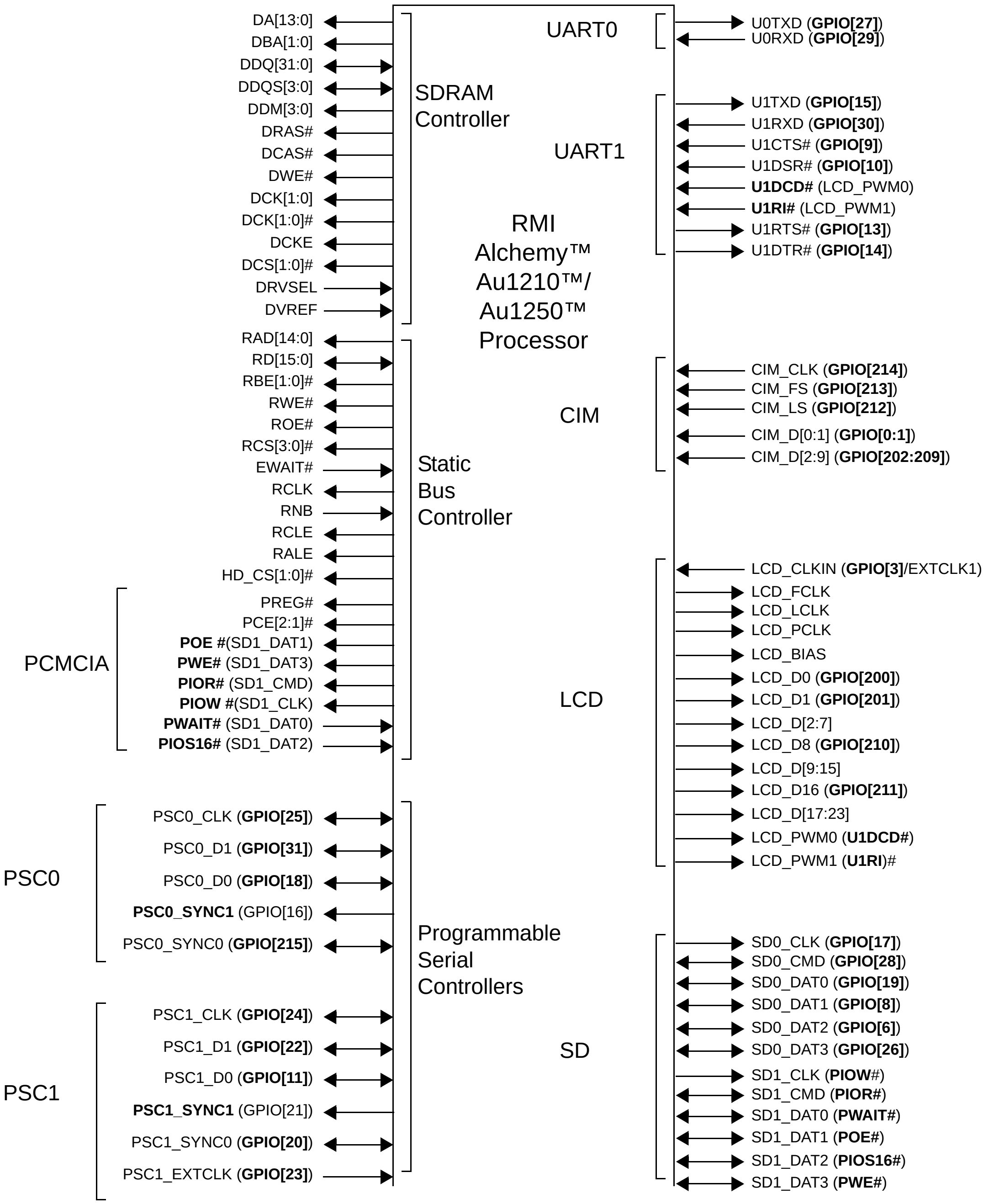
In order to maximize the functionality of the Au1210 and Au1250 processors, many of the pins have multiple uses. If a pin is configured for one use, any other functionality associated with that pin cannot be utilized at the same time. In other words a pin cannot be used as a general purpose I/O signal at the same time it is assigned to a peripheral device. (See Section 10.3.1 "Pin Functionality" on page 381.)

Figure 13-1 on page 423 shows the external signals of the Au1210 and Au1250 processors. All signals are grouped according to their functional block. Signals that share a pin are listed with the multiplexed signal name in parentheses and the signal name shown in **bold** is the default.

Note: A signal with a # is active low; that is, the signal is considered asserted (active) when low and negated when high. Active high signals (no #) are considered asserted when high and negated when low.

www.DataSheet4U.com

www.DataSheet4U.com



www.DataSheet4U.com

www.DataSheet4U.com

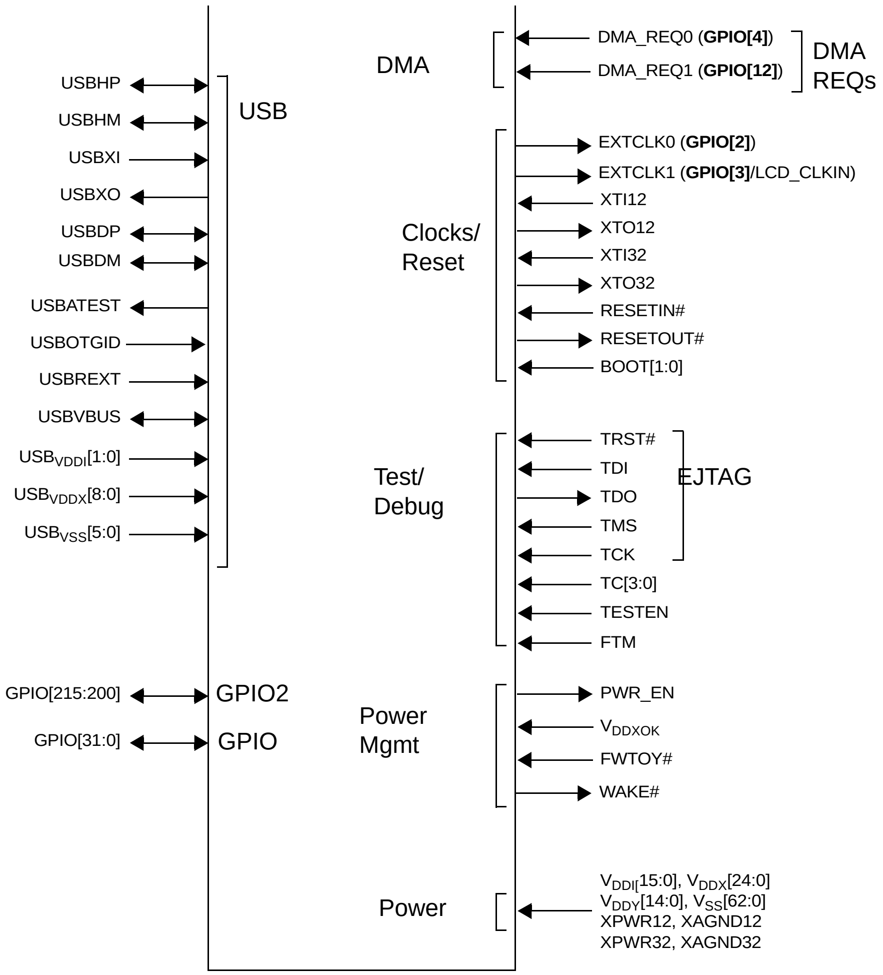


Figure 13-1. Au1210™ and Au1250™ Processors External Signals

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3 on page 425 gives a description of all external signals on the Au1210 and Au1250 processors. The signals have been grouped by functional block. Signals that require external termination are noted in the description. Table 13-3 also defines the default state of the signals coming out of a hardware reset, coming out of a runtime reset, and during Sleep. The abbreviations used for the signal types and the signal states are defined in Table 13-1 and Table 13-2.

Table 13-1. Signal Type Abbreviations for Table 13-3 on page 425

Signal Type	Definition
I	Input. All *unused* input pins must be terminated low or high via direct connection to either ground or power.
O	Output
I/O	Bidirectional
Z	Tristatable
P	Power
G	Ground

Table 13-2. Signal State Abbreviations for Table 13-3 on page 425

Signal State	Definition
0	Driven low
1	Driven high
IN	Signal is a input.
LV	If driven, an output signal continues to be driven at the last value before a reset or entering Sleep.
HIZ	Tristated
ON	Clock remains on *if already enabled*.
UN	Unpredictable
NC	Not connected
NA	Does not apply because this signal is not the default function coming out of a hardware or runtime reset.

www.DataSheet4U.com

www.DataSheet4U.com

In Table 13-3, the signal states shown in the far-right column are valid *during* Sleep. When waking from Sleep, the processor performs an internal system reset that produces the same signal behavior as a *runtime* reset with two exceptions:

- SDRAM interface behavior. During and after a runtime reset, the SDRAM Configuration Mode registers retain their values to allow a transaction in progress to complete. The remaining SDRAM configuration registers revert to their default values. Waking from Sleep, however, all SDRAM configuration registers revert to their default values, and the interface behaves the same as when coming out of a hardware reset.
- PWR_EN behavior. During a runtime reset, PWR_EN remains asserted. During Sleep, PWR_EN is negated. Waking from Sleep, PWR_EN is asserted according to the timing specified in Section 10.4.3.1 "Sleep Sequence and Timing" on page 390.

Table 13-3. External Signals

			Reset		
Signal	**Type**	**Description**	**HW**	**Run time**	**During Sleep**
DDR SDRAM Interface					
DA[13:0]	O	DA[13:0]. Address Outputs A0-A13.	UN	UN	(Note1)
DBA[1:0]	O	Bank Address Outputs: DBA1 and DBA0 define to which bank the ACTIVE, READ, WRITE, or PRECHARGE command is being applied.	UN	UN	(Note 1)
DDQ[31:0]	I/O	SDRAM Data Bus. During a hardware reset the SDRAM data bus cycles from low voltage to hi-Z and then low as follows: 0 after V_{DDXOK} is asserted. Tristated when V_{DDI} is on and RESETOUT# is asserted. 0 after hardware reset sequence is complete.	(See description at left.)	HIZ	(Note 1)
DDQS[3:0]	I/O	DDR only. Input/Output Strobe: The Au1210™ and Au1250™ processors drive DDQS during writes, and the memory device drives DDQS during reads. It is edge-aligned with read data and center-aligned with write data. DDQS0 strobes DDQ[7:0]. DDQS1 strobes DDQ[15:8]. DDQS2 strobes DDQ[23:16]. DDQS3 strobes DDQ[31:24].	1	1	(Note 1)
DDM[3:0]	O	Input/Output Mask: DDM is an input mask signal for write accesses and an output enable signal for read accesses. DDM0 masks DDQ[7:0]. DDM1 masks DDQ[15:8]. DDM2 masks DDQ[23:16]. DDM3 masks DDQ[31:24].	1	1	(Note 1)
DRAS#	O	Command Outputs. DRAS#, DCAS#, and DWE# (along with DCS*n*#) define the command being sent to the SDRAM rank.	1	1	(Note 1)
DCAS#	O		1	1	(Note 1)
DWE#	O		1	1	(Note 1)
DCK[1:0], DCK[1:0]#	O	Clock outputs.	0	ON	(Note 1)
DCS[1:0]#	O	Programmable chip selects.	1	1	(Note 1)

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
DCKE	O	Clock enable for SDRAM.	0	1	(Note 1)
DRVSEL	I	Drive Strength Select for the SDRAM bus signals. 0 Full strength 1 Reduced strength The DRVSEL signal and **mem_sdconfigb**[DS] both affect the SDRAM bus signal strength: For reduced-strength signals, DRVSEL must be tied high *and* DS must be cleared; otherwise, the signals are driven at full strength.	IN	IN	LV
DVREF	I	SDRAM Voltage Reference. Connect to $V_{DDY}/2$. Use a voltage divider circuit with two equal resistors between VDDY and ground. Also place a 0.1 µF capacitor in parallel with each resistor.	IN	IN	IN
Static Bus (SRAM/I/O/PCMCIA/Flash/ROM/NAND) Interface - Common Signals					
RAD[14:0]	O	Address Bus. (Does not apply for NAND.)	UN	UN	LV
RD[15:0]	I/O	Data Bus.	0	UN	LV
RCLK	O	Bus Clock output for synchronous mode.	0	0	LV
RBE[1:0]#	O	Byte Enable. RBE0# corresponds to RD[7:0], RBE1# corresponds to RD[15:8].	1	1	LV
RWE#	O	Write Enable.	1	1	LV
ROE#	O	Output Enable.	1	1	LV
RCS[3:0]#	O	Programmable Chip Selects. RCS*n*# is not used when configured as a PCMCIA device.	1	1	LV
EWAIT#	I	Can be used to stretch the bus access time when enabled. This input is not recognized for chip selects configured as PCMCIA because it uses PWAIT#.	IN	IN	IN
IDE					
HD_CS[1:0]#	O	Hard Drive Chip Selects.	1	1	LV
NAND					
RNB	I	Ready, Not Busy.	IN	IN	IN
RALE	O	Address Latch Enable Signal.	0	0	LV
RCLE	O	Command Latch Enable Signal.	0	0	LV
PCMCIA					
PREG#	O	Register only access signal.	1	1	LV
PCE[2:1]#	O	Card Enables.	1	1	LV
POE#	O	Output Enable. Muxed with SD1_DAT1. POE# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV
PWE#	O	Write Enable. Muxed with SD1_DAT3. PWE# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
PIOR#	O	Read Cycle Indication. Muxed with SD1_CMD. PIOR# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV
PIOW#	O	Write Cycle Indication. Muxed with SD1_CLK. PIOW# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV
PWAIT#	I	Extend Cycle. Termination Note: Tie high through a resistor when the PCMCIA interface is not used. Muxed with SD1_DAT0. PWAIT# is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	IN
PIOS16#	I	16 bit Port Select. Termination Note: Tie high through a resistor when the PCMCIA interface is not used. Muxed with SD1_DAT2. PIOS16# is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	IN
PSC0					
PSC0_CLK	I/O	PSC0 Clock. Muxed with GPIO[25]. GPIO[25] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC0_SYNC1	O	PSC0 Synchronization Signal 1. Muxed with GPIO[16]. PSC0_SYNC1 is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV
PSC0_SYNC0	I/O	PSC0 Synchronization Signal 0. Muxed with GPIO[215]. GPIO[215] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC0_D1	I/O	PSC0 Data 1. Muxed with GPIO[31]. GPIO[31] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC0_D0	I/O	PSC0 Data 0. Muxed with GPIO[18]. GPIO[18] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC1					
PSC1_CLK	I/O	PSC1 Clock. Muxed with GPIO[24]. GPIO[24] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC1_SYNC1	O	PSC1 Synchronization Signal 1. Muxed with GPIO[21]. PSC1_SYNC1 is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	1	LV
PSC1_SYNC0	I/O	PSC1 Synchronization Signal 0. Muxed with GPIO[20]. GPIO[20] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC1_D1	I/O	PSC1 Data 1. Muxed with GPIO[22]. GPIO[22] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
PSC1_D0	I/O	PSC1 Data 0. Muxed with GPIO[11]. GPIO[11] is the default signal coming out of hardware reset, runtime reset, and Sleep.	IN	IN	LV
PSC1_EXTCLK	I	PSC1 External Clock Input. PSC1_EXTCLK comes in through GPIO[23]. To use this clock, the GPIO must be configured as an input.	NA	NA	NA
USB					
USB_{VDDX}	P	3.3 volt nominal power for the USB I/O peripheral. Same as V_{DDX} but isolate from V_{DDX} using ferrite bead and capacitors.			
USB_{VDDI}	P	1.2 volt nominal power for the USB I/O peripheral. Same as V_{DDI} but isolate from V_{DDI} using ferrite bead and capacitors.			
$USBV_{SS}$	G	Ground for the USB I/O peripheral. Tie to V_{SS}.			
USBDP	I/O	Positive signal of differential USB device port. Connect to an external common-mode choke.	UN	UN	HIZ
USBDM	I/O	Negative signal of differential USB device port. Connect to an external common-mode choke.	UN	UN	HIZ
USBXI	I	Crystal oscillator input pin. Connect to a 48MHz crystal or an external oscillator.			
USBXO	O	Crystal oscillator output pin. Connect to a 48MHz crystal, or tie to ground if an external oscillator is applied to USBXI.			
USBHP	I/O	Positive signal of differential USB host port. Connect to an external common-mode choke.	UN	UN	HIZ
USBHM	I/O	Negative signal of differential USB host port. Connect to an external common-mode choke.	UN	UN	HIZ
USBREXT	G	External resistor that sets the analog bias currents. Connect with a 3.4KΩ resistor +/- 1% to $USBV_{SS}$.	NA	NA	NA
USBVBUS	I/O	Monitors the USB bus voltage on the USB connector. USBVBUS is used to sense the value of the bus voltage to generate the VBUS status signals SessEnd, Avalid, Bvalid and VbusValid. USBVBUS performs the USB OTG session request protocol. This pin does not provide power to the USB bus.	HIZ	HIZ	HIZ
USBOTGID	I	Connect to the ID pin on the USB Mini receptacle that is used to differentiate a Mini-A plug from a Mini-B plug.	HIZ	HIZ	HIZ
USBATEST	UN	Manufacturing test pin. Leave open.			
UART0					
U0TXD	O	UART0 Transmit. Muxed with GPIO[27]. GPIO[27] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
U0RXD	I	UART0 Receive. Muxed with GPIO[29]. GPIO[29] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
UART1					

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
U1TXD	O	UART1 Transmit. Muxed with GPIO[15]. GPIO[15] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
U1RXD	I	UART1 Receive. Muxed with GPIO[30]. GPIO[30] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
U1CTS#	I	Clear to Send. Muxed with GPIO[9]. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	NA	NA	NA
U1RTS#	O	Request to Send. Muxed with GPIO[13]. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
U1DSR#	I	Data Set Ready. Muxed with GPIO[10]. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	NA	NA	NA
U1DCD#	I	Data Carrier Detect. Muxed with LCD_PWM0. U1DCD# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	IN	IN	LV
U1RI#	I	Ring Indication. Muxed with LCD_PWM1. U1RI# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	IN	IN	LV
U1DTR#	O	Data Terminal Ready. Muxed with GPIO[14]. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
CIM					
CIM_FS	O	Frame Sync. Muxed with GPIO[213]. GPIO[213] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
CIM_LS	O	Line Sync. Muxed with GPIO[212]. GPIO[212] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
CIM_CLK	I	CIM Clock. Muxed with GPIO[214]. GPIO[214] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
CIM_D[1:0] (GPIO[1:0])	I	CIM Data. GPIO[1:0] can be configured as CIM_D[1:0].	HIZ	Note 2	LV
CIM_D[9:2]	I	CIM Data. Muxed with GPIO[209:202]. GPIO[209:202] are the default signals coming out of hardware reset, runtime reset, and Sleep.	NA	NA	NA
LCD Controller					
LCD_PWM1	O	Pulse Width Modulation Clock 1. Muxed with U1RI.# U1RI #is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	0	0	LV
LCD_PWM0	O	Pulse Width Modulation Clock 0. Muxed with U1DCD#. U1DCD# is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE] = 0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	0	0	LV
LCD_BIAS	O	Bias Clock.	0	0	LV
LCD_FCLK	O	Frame Clock.	0	0	LV
LCD_LCLK	O	Line Clock.	0	0	LV
LCD_PCLK	O	Pixel Clock.	0	0	LV
LCD_CLKIN (GPIO[3])	I	LCD Clock Source. Optional input clock using GPIO[3].	HIZ	(Note 2)	LV
LCD_D[1:0]	O	LCD Data. Muxed with GPIO[201:200]. GPIO[201:200] are the default signals coming out of hardware reset, runtime reset, and Sleep.	0	0	LV
LCD_D[7:2]	O	LCD Data.	0	0	LV
LCD_D[8]	O	LCD Data. Muxed with GPIO[210]. GPIO[210] is the default signal coming out of hardware reset, runtime reset, and Sleep.	0	0	LV
LCD_D[15:9]	O	LCD Data.	0	0	LV
LCD_D[16]	O	LCD Data. Muxed with GPIO[211]. GPIO[211] is the default signal coming out of hardware reset, runtime reset, and Sleep.	0	0	LV
LCD_D[23:17]	O	LCD Data.	0	0	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset HW	Reset Run time	During Sleep
Secure Digital Controller					
SD0_CLK	O	SD Card 0 Interface Clock. Muxed with GPIO[17]. GPIO[17] is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	0	LV
SD0_CMD	I/O	SD Card 0 Half Duplex Command and Response. Muxed with GPIO[28]. GPIO[28] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD0_DAT0	I/O	SD Card 0 Data Bus. Muxed with GPIO[19]. GPIO[19] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD0_DAT1	I/O	SD Card 0 Data Bus. Muxed with GPIO[8]. GPIO[8] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD0_DAT2	I/O	SD Card 0 Data Bus. Muxed with GPIO[6]. GPIO[6] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD0_DAT3	I/O	SD Card 0 Data Bus. Muxed with GPIO[26]. GPIO[26] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD1_CLK	O	SD Card 1 Interface Clock. Muxed with PIOW.# PIOW# is the default signal coming out of hardware reset, runtime reset, and Sleep.	1	0	LV
SD1_CMD	I/O	SD Card 1 Half Duplex Command and Response. Muxed with PIOR#. PIOR# is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD1_DAT0	I/O	SD Card 1 Data Bus. Muxed with PWAIT#. PWAIT# is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD1_DAT1	I/O	SD Card 1 Data Bus. Muxed with POE#. POE# is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD1_DAT2	I/O	SD Card 1 Data Bus. Muxed with PIOS16#. PIOS16# is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
SD1_DAT3	I/O	SD Card 1 Data Bus. Muxed with PWE#. PWE# is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
EJTAG					
TRST	I	Asynchronous TAP reset.	IN	IN	IN
TDI	I	Test data input to the instruction or selected data registers. Sampled on the rising edge of TCK.	IN	IN	IN
TDO	O	Test data output from the instruction or data register. Transitions occur on the falling edge (valid on rising edge) of TCK.	HIZ	UN	LV
TMS	I	Control signal for TAP controller. Sampled on the rising edge of TCK.	IN	IN	IN

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
TCK	I	Control clock for updating TAP controller and shifting data through instruction or selected data register.	IN	IN	IN
Test					
TC[3:0]	I	Test clock inputs (not used in typical applications). Tie low for normal operation.	IN	IN	IN
TESTEN	I	Test enable (not used in typical applications). Tie low for normal operation.	IN	IN	IN
FTM	I	Factory test mode (not used). Tie low through a resistor.	IN	IN	IN
GPIO					
GPIO[1:0]	I/O, Z	General Purpose I/O. Can be configured as CIM_D[1:0].	HIZ	(Note2)	LV
GPIO[3:2]	I/O, Z	General Purpose I/O. Muxed with EXTCLK[1:0]. GPIO[3:2] are the default signals coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[4]	I/O, Z	General Purpose I/O. Can be configured as DMA_REQ0; see Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139.	HIZ	(Note 2)	LV
GPIO[5]	I/O, Z	General Purpose I/O.	HIZ	(Note 2)	LV
GPIO[6]	I/O, Z	General Purpose I/O. Muxed with SD0_DAT2. GPIO[6] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[7]	I/O, Z	General Purpose I/O.	HIZ	(Note 2)	LV
GPIO[8]	I/O, Z	General Purpose I/O. Muxed with SD0_DAT1. GPIO[8] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[9]	I/O, Z	General Purpose I/O. Muxed with U1CTS#. GPIO[9] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	HIZ	(Note 2)	LV
GPIO[10]	I/O, Z	General Purpose I/O. Muxed with U1DSR#. GPIO[10] is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: For systems that use the UART1 interface without the modem control signals, the modem status interrupts must be disabled (**uart_inten**[MIE]=0) to avoid false UART1 interrupts when using GPIO[9], GPIO[10], or the LCD PWM generators.	HIZ	(Note 2)	LV
GPIO[11]	I/O, Z	General Purpose I/O. Muxed with PSC1_D0. GPIO[11] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[12]	I/O, Z	General Purpose I/O. Can be configured as DMA_REQ1; see Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139.	HIZ	(Note 2)	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
GPIO[13]	I/O, Z	General Purpose I/O. Muxed with U1RTS#. GPIO[13] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[14]	I/O, Z	General Purpose I/O. Muxed with U1DTR#. GPIO[14] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[15]	I/O, Z	General Purpose I/O. Muxed with U1TXD. GPIO[15] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[16]	I/O, Z	General Purpose I/O. Muxed with PSC0_SYNC1. PSC0_SYNC1 is the default signal coming out of hardware reset, runtime reset, and Sleep. **System Note**: The GPIO[16]/PSC0_SYNC1 pin is driven high coming out of reset.	NA	NA	LV
GPIO[17]	I/O, Z	General Purpose I/O. Muxed with SD0_CLK. GPIO[17] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[18]	I/O, Z	General Purpose I/O. Muxed with PSC0_D0. GPIO[18] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[19]	I/O, Z	General Purpose I/O. Muxed with SD0_DAT0. GPIO[19] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[20]	I/O, Z	General Purpose I/O. Muxed with PSC1_SYNC0. GPIO[20] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[21]	I/O, Z	General Purpose I/O. Muxed with PSC1_SYNC1. PSC1_SYNC1 is the default signal coming out of hardware reset, runtime reset, and Sleep. System Note: The GPIO[21]/PSC1_SYNC1 pin is driven high coming out of reset.	NA	NA	LV
GPIO[22]	I/O, Z	General Purpose I/O. Muxed with PSC1_D1. GPIO[22] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[23]	I/O, Z	General Purpose I/O. GPIO[23] can be used as a clock source for frequency generator 3 (FREQ3); see Section 10.1.1.1 "Clock Generator Registers" on page 367. Muxed with PSC1_EXTCLK. GPIO[23] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[24]	I/O, Z	General Purpose I/O. Muxed with PSC1_CLK. GPIO[24] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[25]	I/O, Z	General Purpose I/O. Muxed with PSC0_CLK. GPIO[25] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
GPIO[26]	I/O, Z	General Purpose I/O. Muxed with SD0_DAT3. GPIO[26] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[27]	I/O, Z	General Purpose I/O. Muxed with U0TXD. GPIO[27] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[28]	I/O, Z	General Purpose I/O. Muxed with SD0_CMD. GPIO[28] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[29]	I/O, Z	General Purpose I/O. Muxed with U0RXD. GPIO[29] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[30]	I/O, Z	General Purpose I/O. Muxed with U1RXD. GPIO[30] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[31]	I/O, Z	General Purpose I/O. Muxed with PSC0_D1. GPIO[31] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	(Note 2)	LV
GPIO[200]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with LCD_D0. GPIO[200] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[201]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with LCD_D1. GPIO[201] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[202]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D2. GPIO[202] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[203]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D3. GPIO[203] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[204]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D4. GPIO[204] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[205]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D5. GPIO[205] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[206]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D6. GPIO[206] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[207]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D7. GPIO[207] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[208]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D8. GPIO[208] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[209]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_D9. GPIO[209] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		During Sleep
			HW	Run time	
GPIO[210]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with LCD_D8. GPIO[210] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[211]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with LCD_D16. GPIO[211] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[212]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_LS. GPIO[212] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[213]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_FS. GPIO[213] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[214]	I/O, Z	General Purpose I/O (secondary GPIO block). Muxed with CIM_CLK. GPIO[214] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
GPIO[215]	IOZ	General Purpose I/O (secondary GPIO block). Muxed with PSC0_SYNC0. GPIO[215] is the default signal coming out of hardware reset, runtime reset, and Sleep.	HIZ	HIZ	LV
External Clocks					
EXTCLK[0]	O	General-purpose external clock. Muxed with GPIO[2]. GPIO[2] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
EXTCLK[1]	O	General-purpose external clock. Muxed with GPIO[3]. GPIO[3] is the default signal coming out of hardware reset, runtime reset, and Sleep.	NA	NA	LV
System DMA Requests					
DMA_REQ0 (GPIO[4])	I	GPIO[4] can be configured as an external, system DMA request input. See Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139.	HIZ	HIZ	HIZ
DMA_REQ1 (GPIO[12])	I	GPIO[12] can be configured as an external, system DMA request input. See Section 4.4 "Using GPIO as External DMA Requests (DMA_REQn)" on page 139.	HIZ	HIZ	HIZ
System Clocks and Reset					
XTI12	I	Internally compensated 12MHz (typical) crystal input. Termination Note: The termination depends on the application as follows: Crystal - Connect crystal between XTI12 and XTO12. Oscillator - Connect to external 12MHz clock source.			
XTO12	O	Internally compensated 12MHz (typical) crystal output. Termination Note: The termination depends on the application as follows: Crystal - Connect crystal between XTI12 and XTO12. Oscillator - Leave unconnected.			

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

Signal	Type	Description	Reset		
			HW	Run time	During Sleep
XTI32	I	Internally compensated 32.768Hz (typical) crystal input. Termination Note: The termination depends on the application as follows: Crystal - Connect crystal between XTI32 and XTO32. Oscillator - Connect to external 32.768KHz clock source. Not used - Pull-up to $_{\text{VDDX}}$ through a 10kΩ resistor.			
XTO32	O	Internally compensated 32.768KHz (typical) crystal output. Termination Note: The termination depends on the application as follows: Crystal - Connect crystal between XTI32 and XTO32. Oscillator - Leave unconnected. Not used - Pull-up to V_{DDX} through a 10kΩ resistor.			
RESETIN#	I	CPU reset input.	IN	IN	IN
RESETOUT#	O	Buffered output of CPU reset input (RESETIN#).	0	0	0
BOOT[1:0]	I	Determines boot memory. Terminate the BOOT[1:0] signals appropriately because they must not change during runtime.	IN	IN	IN
Power Management					
PWR_EN	O	Power enable output. This signal is intended to be used as the regulator enable for V_{DDI} (core power).	1	1	0
V_{DDXOK}	I	Input to signal that V_{DDX} (and V_{DDY} on power-up) is stable.	IN	IN	LV
WAKE#	O	Wake from Hibernate request signal	1	1	1
FWTOY#	I	Firewall the TOY Clock Signal. Connect FWTOY# according to the system application; see Section 10.4.4.5 "System Configuration for Hibernate" on page 392. If FWTOY# is left floating, operation is undefined.	IN	IN	IN
Power/Ground					
V_{DDI}	P	Internal Core Voltage. Follow the power supply layout guidelines in Section 14.9.2 "Decoupling Recommendations" on page 472.			
V_{DDX}	P	External I/O Voltage. Follow the power supply layout guidelines in Section 14.9.2 "Decoupling Recommendations" on page 472.			
V_{DDY}	P	External I/O Voltage for SDRAM only. Follow the power supply layout guidelines in Section 14.9.2 "Decoupling Recommendations" on page 472.			
V_{SS}	G	Ground.			
XPWR12	P	12MHz (typical) Oscillator and PLL Power. Connect to V_{DDX} through a 10Ω resistor. Also, place a 22µF capacitor in parallel with a 0.01µF capacitor from this pin to XAGND12.			
XAGND12	G	12MHz (typical) oscillator and PLL ground.			

www.DataSheet4U.com

www.DataSheet4U.com

Table 13-3. External Signals (Continued)

			Reset		
Signal	Type	Description	HW	Run time	During Sleep
XPWR32	P	32.768KHz (typical) Oscillator and PLL Power. Because XPWR32 powers other circuitry also, it must be connected even if the oscillator is not used. Connect to V_{DDX} through a 10Ω resistor. Also, place a 22µF capacitor in parallel with a 0.01µF capacitor from this pin to XAGND32. To support Hibernate mode, design the board to switch the XPWR32 power source to a battery when V_{DDX} is disabled.			
XAGND32	G	32.768KHz (typical) Oscillator and PLL Ground.			

Note1. Depends on **sys_powerctrl**[SSY]. If SSY = 0, the SDRAM interface signals hold their last values if driven (LV); if SSY = 1, the SDRAM signals are forced to their hardware reset values (as shown in the *Hardware* column).

Note2. Depends on **sys_trioutrd** and **sys_outputset**. During a runtime reset, **sys_pinfunc** returns to its default value, but the GPIO control registers **sys_trioutrd** and **sys_outputset** remain unchanged.

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

www.DataSheet4U.com

14 Electrical and Thermal Specifications

This chapter provides preliminary electrical specifications for the Au1210™ and Au1250™ processors, including the following:

- Absolute Maximum Ratings
- Thermal Characteristics
- DC Parameters
- AC Parameters
- Power-up, Reset, Sleep, Hibernate, and Idle Timing
- External Clock Specifications
- Crystal Specifications
- System Design Considerations

The electrical specifications provided in this chapter are characterized for the rated frequency of the part. The parts are tested to guarantee operation at the rated frequency. All systems require a 12MHz input to satisfy peripheral clocking requirements. Table 14-1 shows the rated frequency compared with the actual operating frequency of the part when using a 12MHz crystal.

Table 14-1. Rated and Actual CPU Frequencies Using a 12MHz Crystal

Rated Frequency (MHz)	Actual Frequency (MHz)
600	600
500	492
400	396
333	324

www.DataSheet4U.com

www.DataSheet4U.com

14.1 Absolute Maximum Ratings

Table 14-2 shows the absolute maximum ratings for the Au1210 and Au1250 processors. These ratings are stress ratings, operating at or beyond these ratings for extended periods of time may result in damage to the Au1210 and Au1250 processors.

Unless otherwise designated all voltages are relative to V_{SS}.

Table 14-2. Absolute Maximum Ratings

Parameter	Description	Minimum	Maximum	Unit
V_{DDI}	Core Voltage	V_{SS} - .5	1.3	V
V_{DDX}	I/O Voltage	V_{SS} - .5	3.6	V
V_{DDY}	I/O Voltage	V_{SS} - .5	3.6	V
XPWR12	12MHz Oscillator Voltage	V_{SS} - .5	3.6	V
XPWR32	32.768KHz Oscillator Voltage	V_{SS} - .5	3.6	V
V_{IN}	Voltage applied to any pin	V_{SS} - .5	$V_{DDX,Y}$ + .5	V
T_{CASE} Commercial	Package operating temperature	0	85	°C
T_{CASE} Extended (Note1)		-40	100	°C
T_S	Storage Temperature	-40	125	°C

Note1. Extended temperature operation at 400MHz and 500MHz is supported with DDR2 memory systems only.

14.1.1 Undershoot

The minimum DC voltage on input or I/O pins is -0.5V. However, during voltage transitions, the device can tolerate undershoot to -2.0V for up to 20ns, as shown in Figure 14-1.

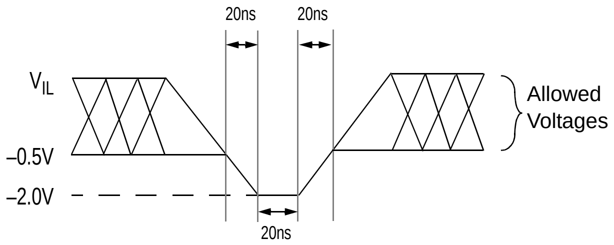


Figure 14-1. Voltage Undershoot Tolerances for Input and I/O Pins

www.DataSheet4U.com

www.DataSheet4U.com

14.1.2 Overshoot

The maximum DC voltage on input or I/O pins is ($V_{DDX,Y}$ + 0.5) V. However, during voltage transitions, the device can tolerate overshoot to ($V_{DDX,Y}$ + 2.0) V for up to 20ns, as shown in Figure 14-2.

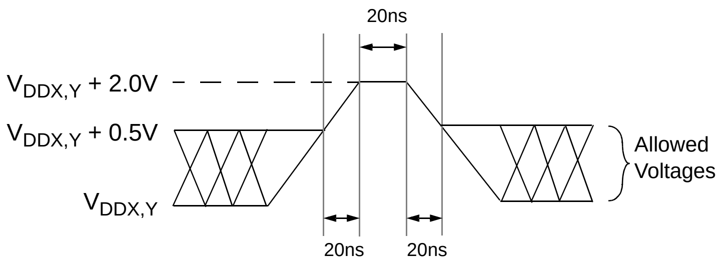


Figure 14-2. Voltage Overshoot Tolerances for Input and I/O Pins

14.1.3 Electrostatic Discharge (ESD)

The Au1210 and Au1250 processors are ESD sensitive. Handling and assembly of this device must be performed at ESD-free workstations. Table 14-3 lists the ESD ratings for the external signals of the processors.

Table 14-3. ESD Ratings

Parameter	Pin Number	Signal	Unit
Human Body Model (HBM)	A4, B4, C6, C5, A3, B3, C4, C3	XTI12, XTO12, XAGND12, XPWR12, XTI32, XTO32, XAGND32, XPWR32	900V ESD
	—	All other signals	2000V ESD
Machine Model (MM)	A4, B4, A3, B3	XTI12, XTO12, XTI32, XTO32	150V ESD
	C6, C5, C4, C3	XAGND12, XPWR12, XAGND32, XPWR32	100V ESD
	—	All other signals	200V ESD
Charged Device Model (CDM)	—	All signals	300V ESD

14.2 Thermal Characteristics

Table 14-4 shows the thermal characteristics for the Au1210 and Au1250 processors.

Table 14-4. Thermal Characteristics

Parameter	Description	Value	Unit
Θ_{JA}	Thermal resistance from device junction to ambient.	27.9(Note1)	°C/W
Ψ_{JT}	Thermal characterization parameter measured from device junction to top center of package.(Note2)	9.7	°C/W

Note1. Estimated value

Note2. See JESD51-2, Sec. 4.

www.DataSheet4U.com

www.DataSheet4U.com

14.3 DC Parameters

Tables 14-5, 14-6 and 14-7 on page 443 show the DC parameters for the Au1210 and Au1250 processors. Unless otherwise designated all voltages are relative to V_{SS}.

The operating requirements for the power supply voltages (V_{DDX}, V_{DDI}, and V_{DDY}) are given in the sections describing the DC characteristics for the different operating frequencies, beginning with Section 14.3.1 "Power and Voltage for 333MHz Rated Part" on page 444.

Table 14-5. DC Parameters for I/O (Except SDRAM)

Parameter	Description	Min	Nominal	Max	Unit
V_{IHX}	Input High Voltage	2.4			V
V_{ILX}	Input Low Voltage			0.2 * V_{DDX}	V
V_{OHX} @ 2 mA	Output High Voltage	0.8 * V_{DDX}			V
V_{OLX} @ 2 mA	Output Low Voltage			0.2 * V_{DDX}	V
I_L	Input Leakage Current (except USB)			5	µA
I_{USB}	Input Leakage Current for USB			15	µA
C_{IN}	Input Capacitance		5		pF

Table 14-6. DC/AC Parameters for DDR SDRAM(Note1)

Parameter	Description	Min	Nominal	Max	Unit
DVREF	DDR SDRAM Reference Voltage	0.49 * V_{DDY}	0.50 * V_{DDY}	0.51 * V_{DDY}	V
V_{IHDDC}	DDR SDRAM Input High Voltage (DC)	DVREF + 0.15		V_{DDY} + 0.3	V
V_{ILDDC}	DDR SDRAM Input Low Voltage (DC)	- 0.3		DVREF - 0.15	V
V_{IHDAC}	DDR SDRAM Input High Voltage (AC)	DVREF + 0.31		V_{DDY} + 0.3	V
V_{ILDAC}	DDR SDRAM Input Low Voltage (AC)	-0.3		DVREF - 0.31	V
I_L	Input Leakage Current			5	µA
C_{IN}	Input Capacitance		5		pF

Note1. The DDR interface drives and receives voltage levels and timing as specified by JEDEC for DDR memories.

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-7. DC/AC Parameters for DDR2 SDRAM(Note1)

Parameter	Description	Min	Nominal	Max	Unit
DVREF	DDR SDRAM Reference Voltage	0.49 * V_{DDY}	0.50 * V_{DDY}	0.51 * V_{DDY}	V
V_{IHDDC}	DDR SDRAM Input High Voltage (DC)	DVREF + 0.125		V_{DDY} + 0.3	V
V_{ILDDC}	DDR SDRAM Input Low Voltage (DC)	- 0.3		DVREF - 0.125	V
V_{IHDAC}	DDR SDRAM Input High Voltage (AC)	DVREF + 0.25		V_{DDY} + 0.3	V
V_{ILDAC}	DDR SDRAM Input Low Voltage (AC)	-0.3		DVREF - 0.25	V
I_I	Input Leakage Current			5	µA
C_{IN}	Input Capacitance		5		pF

Note1. The DDR interface drives and receives voltage levels and timing as specified by JEDEC for DDR memories.

The values in Tables 14-8 on page 444 through 14-12 on page 448 are determined as follows:

- All power and current measurements are made at the maximum System Bus and DRAM memory speed supported at the given CPU frequency (See Table 3-3 on page 72 and Table 3-4 on page 73).
- The Typical power and current are measured at nominal voltage under Windows CE 5.0 for a test case of video playback of 2Mbps WMV9 D1 resolution (QVGA for Au1210) video with 64kbps WMA audio from IDE hard drive driven to LCD display. Power management is enabled and USB is disabled.
- Idle power is measured at nominal voltages for Windows CE 5.0 desktop display with power management enabled and USB disabled, with no user application running.
- Maximum power and current are measured while the processor is in a tight loop writing to DRAM at maximum bandwidth, in a scenario where core, memory, and system activity are maximized. All peripherals are enabled and all power management features are disabled. In this case no operating system is running. Maximum current values are reported for typical supply voltages. Maximum power is reported for both nominal and maximum supply voltages.

Maximum current and power numbers represent the instantaneous peak values under an intentional high power scenario. While the maximum current and power numbers are appropriate when specifying regulator and thermal designs, they are not typically duplicated in an operating system with power management features enabled. Note that application software and external loading could cause the maximum power consumption to differ from what is shown.

www.DataSheet4U.com

www.DataSheet4U.com

14.3.1 Power and Voltage for 333MHz Rated Part

Table 14-8 shows the power and voltage requirements for the Au1210 processor rated for a CPU frequency of 333MHz.

Table 14-8. Voltage and Power Parameters for 333MHz Au1210™ Processor (Note1)

Parameter	Min	Typical	Max	Unit
V_{DDI}	0.95	1.0	1.3	V
V_{DDX}	3.0	3.3	3.6	V
XPWR12	3.0	3.3	3.6	V
XPWR32(Note2)	3.0	3.3	3.6	V
V_{DDY} (2.5V DDR SDRAM)	2.3	2.5	2.7	V
V_{DDY} (1.8V DDR2 SDRAM)	1.7	1.8	1.9	V
I_{VDDI}		180	400	mA
I_{VDDX}		13	67	mA
I_{VDDY} (V_{DDY} = 2.5V)		53	125	mA
I_{VDDY} (V_{DDY} = 1.8V)		34	72	mA
I_{XPWR12}		558	563	µA
I_{XPWR32}(Note3)		596	616	µA
I_{SLEEP} (V_{DDI} = $V_{SS)}$		5.0	7.5	mA
$I_{HIBERNATE}$			280	µA
P_{IDLE}(Note4) (V_{DDY} = 1.8V)		140		mW
Power Dissipation (DDR1, all voltage supplies nominal)		0.36	0.92	W
Power Dissipation (DDR2, all voltage supplies nominal)		0.28	0.75	W
Power Dissipation (DDR1, all voltage supplies at maximum)			1.3	W
Power Dissipation (DDR2, all voltage supplies at maximum)			1.1	W

Note1. Measurements are taken at 25° C. For values at 85° C, add five percent. This note applies only to power and current measurements. Voltage limits are fixed and independent of temperature.

Note2. Does not apply during Hibernate.

Note3. Does not apply during Sleep and Hibernate.

Note4. P_{IDLE} is the combined average power measured on each supply when the processor is running Windows CE with no user application running and power management enabled.

www.DataSheet4U.com

www.DataSheet4U.com

14.3.2 Power and Voltage for 400MHz Rated Part

Table 14-9 shows the power and voltage requirements for the Au1210 processor rated for a CPU frequency of 400MHz.

Table 14-9. Voltage and Power Parameters for 400MHz Au1210™ Processor (Note1)

Parameter	Min	Typical	Max	Unit
V_{DDI}	0.95	1.0	1.3	V
V_{DDX}	3.0	3.3	3.6	V
XPWR12	3.0	3.3	3.6	V
XPWR32(Note2)	3.0	3.3	3.6	V
V_{DDY} (2.5V SDRAM)	2.3	2.5	2.7	V
V_{DDY} (1.8V DDR2 SDRAM)	1.7	1.8	1.9	V
I_{VDDI}		190	440	mA
I_{VDDX}		14	65	mA
I_{VDDY} (V_{DDY} = 2.5V)		66	150	mA
I_{VDDY} (V_{DDY} = 1.8V)		35	80	mA
I_{XPWR12}		682	686	µA
I_{XPWR32}(Note3)		617	637	µA
I_{SLEEP} (V_{DDI} = V_{SS})		5.0	7.5	mA
$I_{HIBERNATE}$			280	µA
P_{IDLE}(Note4) (V_{DDY} = 1.8V)		150		mW
Power Dissipation (DDR1, all voltage supplies nominal)		0.42	1.0	W
Power Dissipation (DDR2, all voltage supplies nominal)		0.30	0.76	W
Power Dissipation (DDR1, all voltage supplies at maximum)			1.5	W
Power Dissipation (DDR2, all voltage supplies at maximum)			1.2	W

Note1. Measurements are taken at 25° C. For values at 85° C, add five percent. This note applies only to power and current measurements. Voltage limits are fixed and independent of temperature.

Note2. Does not apply during Hibernate.

Note3. Does not apply during Sleep and Hibernate.

Note4. P_{IDLE} is the combined average power measured on each supply when the processor is running Windows CE with no user application running and power management enabled.

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-9 shows the power and voltage requirements for the Au1250 processor rated for a CPU frequency of 400MHz.

Table 14-10. Voltage and Power Parameters for 400MHz Au1250™ Processor (Note1)

Parameter	Min	Typical	Max	Unit
V_{DDI}	1.1	1.2	1.3	V
V_{DDX}	3.0	3.3	3.6	V
XPWR12	3.0	3.3	3.6	V
XPWR32(Note2)	3.0	3.3	3.6	V
V_{DDY} (2.5V SDRAM)	2.3	2.5	2.7	V
V_{DDY} (1.8V DDR2 SDRAM)	1.7	1.8	1.9	V
I_{VDDI}		305	550	mA
I_{VDDX}		15	65	mA
I_{VDDY} (V_{DDY} = 2.5V)		77	150	mA
I_{VDDY} (V_{DDY} = 1.8V)		39	80	mA
I_{XPWR12}		682	686	µA
I_{XPWR32}(Note3)		617	637	µA
I_{SLEEP} (V_{DDI} = V_{SS})		5.0	7.5	mA
$I_{HIBERNATE}$			280	µA
P_{IDLE}(Note4) (V_{DDY} = 1.8V)		180		mW
Power Dissipation (DDR1, all voltage supplies nominal)		0.56	1.2	W
Power Dissipation (DDR2, all voltage supplies nominal)		0.47	1.0	W
Power Dissipation (DDR1, all voltage supplies at maximum)			1.4	W
Power Dissipation (DDR2, all voltage supplies at maximum)			1.2	W

Note1. Measurements are taken at 25° C. For values at 85° C, add five percent. This note applies only to power and current measurements. Voltage limits are fixed and independent of temperature.

Note2. Does not apply during Hibernate.

Note3. Does not apply during Sleep and Hibernate.

Note4. P_{IDLE} is the combined average power measured on each supply when the processor is running Windows CE with no user application running and power management enabled.

www.DataSheet4U.com

www.DataSheet4U.com

14.3.3 Power and Voltage for 500MHz Rated Part

Table 14-11 shows the power and voltage requirements for the Au1250 processor rated for a CPU frequency of 500MHz.

Table 14-11. Voltage and Power Parameters for 500MHz Au1250™ Processor (Note1)

Parameter	Min	Typical	Max	Unit
V_{DDI}	1.1	1.2	1.3	V
V_{DDX}	3.0	3.3	3.6	V
XPWR12	3.0	3.3	3.6	V
XPWR32(Note2)	3.0	3.3	3.6	V
V_{DDY} (2.5V SDRAM)	2.3	2.5	2.7	V
V_{DDY} (1.8V DDR2 SDRAM)	1.7	1.8	1.9	V
I_{VDDI}		320	620	mA
I_{VDDX}		16	65	mA
I_{VDDY} (V_{DDY} = 2.5V)		65	150	mA
I_{VDDY} (V_{DDY} = 1.8V)		87	120	mA
I_{XPWR12}		918	925	µA
I_{XPWR32}(Note3)		645	669	µA
I_{SLEEP} (V_{DDI} = V_{SS})		5.0	7.5	mA
$I_{HIBERNATE}$			280	µA
P_{IDLE}(Note4) (V_{DDY} = 1.8V)		235		mW
Power Dissipation (DDR1, all voltage supplies nominal) (Note5)		0.53	1.3	W
Power Dissipation (DDR2, all voltage supplies nominal) (Note5)		0.58	1.1	W
Power Dissipation (DDR1, all voltage supplies at maximum)			1.6	W
Power Dissipation (DDR2, all voltage supplies at maximum)			1.4	W

Note1. Measurements are taken at 25° C. For values at 85° C, add five percent. This note applies only to power and current measurements. Voltage limits are fixed and independent of temperature.

Note2. Does not apply during Hibernate.

Note3. Does not apply during Sleep and Hibernate.

Note4. P_{IDLE} is the combined average power measured on each supply when the processor is running Windows CE with no user application running and power management enabled.

Note5. Because of limitations on the available DDR1 speed, the maximum supported DDR1 and DDR2 memory speeds differ for a 500MHz CPU frequency. In this case a 164MHz DDR1 maximum speed uses a system bus clock divider of 3, while a 246MHz DDR2 maximum speed uses a system bus clock divider of 2.

www.DataSheet4U.com

www.DataSheet4U.com

14.3.4 Power and Voltage for 600MHz Rated Part

Table 14-12 shows the power and voltage requirements for the Au1250 processor rated for a CPU frequency of 600MHz.

Table 14-12. Voltage and Power Parameters for 600MHz Au1250™ Processor (Note1)

Parameter	Min	Typical	Max	Unit
V_{DDI}	1.1	1.2	1.3	V
V_{DDX}	3.0	3.3	3.6	V
XPWR12	3.0	3.3	3.6	V
XPWR32(Note2)	3.0	3.3	3.6	V
V_{DDY} (2.5V SDRAM)	2.3	2.5	2.7	V
V_{DDY} (1.8V DDR2 SDRAM)	1.7	1.8	1.9	V
I_{VDDI}		320	650	mA
I_{VDDX}		15	65	mA
I_{VDDY} (V_{DDY} = 2.5V)		77	170	mA
I_{VDDY} (V_{DDY} = 1.8V)		40	95	mA
I_{XPWR12}		970	975	µA
I_{XPWR32}(Note3)		690	700	µA
I_{SLEEP} (V_{DDI} = V_{SS})		5.0	7.5	mA
$I_{HIBERNATE}$			280	µA
P_{IDLE}(Note4) (V_{DDY} = 1.8V)		185		mW
Power Dissipation (DDR1, all voltage supplies nominal)		0.58	1.4	W
Power Dissipation (DDR2, all voltage supplies nominal)		0.49	1.2	W
Power Dissipation (DDR1, all voltage supplies at maximum)			1.6	W
Power Dissipation (DDR2, all voltage supplies at maximum)			1.4	W

Note1. Measurements are taken at 25° C. For values at 85° C, add five percent. This note applies only to power and current measurements. Voltage limits are fixed and independent of temperature.

Note2. Does not apply during Hibernate.

Note3. Does not apply during Sleep and Hibernate.

Note4. P_{IDLE} is the combined average power measured on each supply when the processor is running Windows CE with no user application running and power management enabled.

www.DataSheet4U.com

www.DataSheet4U.com

14.4 AC Parameters

This section describes the AC parameters for I/O devices in the Au1210 and Au1250 processors. Each class of output signal has different capacitive loads. As the capacitance on the load increases, the propagation delay increases. These specifications assume the maximum capacitive load to be 50pF for all I/O signals other than the SDRAM interfaces.

The timing of those signals that 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 is used. T_{sys} is defined in nanoseconds as:

T_{sys} = SD/CPU

The symbol, CPU, should be interpreted as the CPU clock speed in MHz as set by the CPU PLL. See Section 10.1 "Clocks" on page 366 for details. The symbol, SD, is the system bus divider. See Section 10.4 "Power Management" on page 387 for details.

14.4.1 DDR Timing and Loading

The Au1210 and Au1250 processors support three ranks of 32-bit wide DDR SDRAM. A rank is a physical grouping of DDR SDRAM devices, all tied to the same chip select. The term *rank* is used to distinguish the physical grouping of the SDRAM chips from the internal banks of an SDRAM chip.

When interfacing the Au1210 and Au1250 processors to DDR, note the following:

- Compatible with all DDR200, DDR266, DDR333, DDR400, DDR2-400, and DDR2-533 (DDR2-533 SDRAM operates at reduced frequencies that match the speed rating of the Au1210 and Au1250 as shown in Table 3-4 on page 73. Table 3-4 shows the supported combinations of CPU frequency, system bus divider, and memory bus divider) timing and levels.
- 2.5V and 1.8V DDR SDRAM is supported.
- The Au1210 and Au1250 processors are designed for point-to-point memory systems. The SDRAM controller is *not* designed to drive DIMMs commonly used in the PC motherboard environment.
- Designed for series resistor PC board termination, not parallel resistor to VTT. For DDR2, however, board termination is not needed because DDR2 SDRAM devices already have on-die termination (ODT).
- Outputs can be full-strength or reduced-strength. (See the DS bit in Section 3.1.1.4 "Global Configuration Register B" on page 62.) Strength selection affects all DDR signals.
  The reduced-strength option lowers the power consumption for system designs with lighter memory loading, typically 1 rank of DDR slower than DDR400. The full-strength option is provided if needed. The system designer is responsible for determining the drive strength requirements of the specific design.

Certain DDR SDRAM timing parameters are based off of the SDRAM bus clock. When this is the case, the symbol T_{DCK} is used. T_{DCK} is defined in nanoseconds as:

T_{DCK} = Clock-Ratio * T_{SYS}

The clock-ratio is either 1 or 2 and is programmed in **mem_sdconfigb**[CR]; see Section 3.1.1.4 "Global Configuration Register B" on page 62.

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-13. DDR SDRAM Controller Interface for DDR2-400

Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.4		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.4		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.35	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 0.45		ns

Table 14-14. DDR SDRAM Controller Interface for DDR2-533 (Note1)

Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.35		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.35		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.30	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 0.4		ns

Note1. DDR2-533 SDRAM operates at reduced frequencies that match the speed rating of the Au1210 and Au1250 as shown in Table 3-4 on page 73. Table 3-4 shows the supported combinations of CPU frequency, system bus divider, and memory bus divider.

Table 14-15. DDR SDRAM Controller Interface for DDR400

Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.4		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.4		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.4	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 0.5		ns

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-16. DDR SDRAM Controller Interface for DDR333

Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.45		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.45		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.45	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 0.55		ns

Table 14-17. DDR SDRAM Controller Interface for DDR266

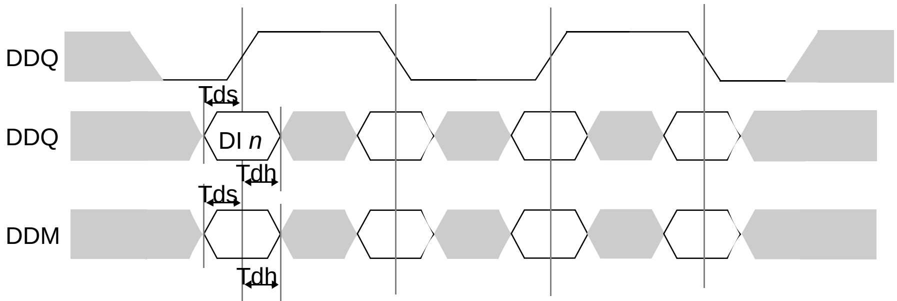
Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.5		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.5		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.5	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 0.75		ns

Table 14-18. DDR SDRAM Controller Interface for DDR200

Signals	Symbol	Parameter	Min	Max	Unit
DCKn	Tch	Clock high-level width	0.45	0.55	T_{dck}
DCKn	Tcl	Clock low-level width	0.45	0.55	T_{dck}
DDM[3:0], DDQ[31:0]	Tds	Data and mask setup to DDQS	0.6		ns
DDM[3:0], DDQ[31:0]	Tdh	Data and mask hold to DDQS	0.6		ns
DDQ[31:0], DDQS[3:0]	Tdqsq	Skew between data and strobe		0.6	ns
DDQ[31:0], DDQS[3:0]	Tqh	Data hold from strobe	Min[Tcl, Tch] - 1.0		ns

www.DataSheet4U.com

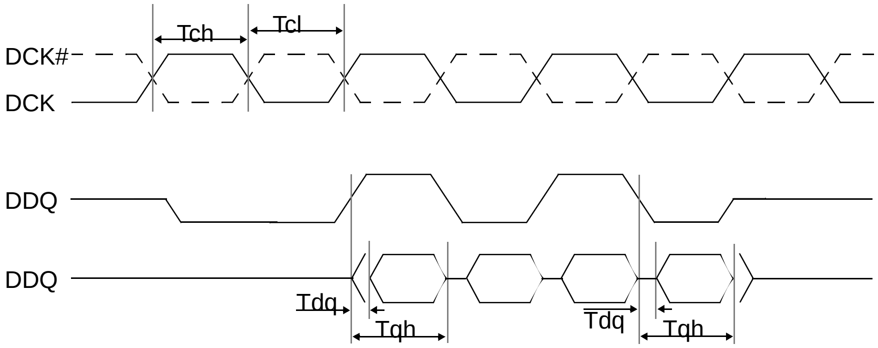
www.DataSheet4U.com



DI *n* = Data In for column *n*.

Burst Length = 4 in the case shown.

Figure 14-3. DDR Data Input (Write) Timing



Burst of four is

Figure 14-4. DDR Data Output (Read) Timing

www.DataSheet4U.com

www.DataSheet4U.com

14.4.2 Asynchronous Static Bus Controller Timing

The timing presented in registers **mem_sttime*n*** is not presented here. The parameters in these registers are presented in a certain number of clock cycles and are accurate to within ± 2ns.

Table 14-19. Static RAM and I/O Device Timing (Asynchronous Mode)

Signal	Symbol	Parameter	Min	Max	Unit
RBE[1:0]#, ROE#, RAD[14:0]	Trcd	Delay from RCS*n*#	-2	+2	ns
RD[15:0] (read)	Trsu	Data setup to RCS*n*# Trsu does not apply when EWAIT# is used to extend the cycle.	15		ns
RD[15:0] (read)	Trsue	Data setup to EWAIT# Trsue applies only when EWAIT is used to extend the cycle.	0		ns
RD[15:0] (read)	Trh	Data hold from RCS*n*	0		ns
RD[15:0] (write)	Trod	Delay from RWE# to data out	-2	2	ns
EWAIT	Trwsu	EWAIT# setup to RCS*n*# for reads, or RWE# for writes If EWAIT# does not meet this setup time, the cycle is not held.	(3 * Tsys) + 15		ns
RCS# (reads), RWE# (writes)	Trwd	Delay from EWAIT#	2 * Tsys	(3 * Tsys) + 15	

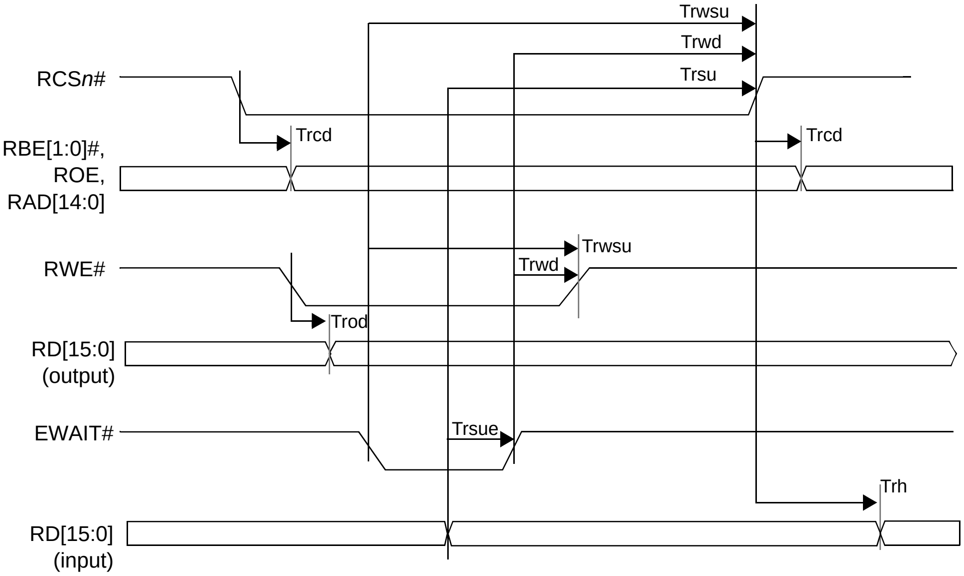


Figure 14-5. Static RAM and I/O Device Timing (Asynchronous Mode)

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-20. PCMCIA Timing

Signal	Symbol	Parameter	Min	Max	Unit
PREG#, RAD[14:0], RD[15:0] (output)	Tpcd	Delay from PCEn#	-2	+2	ns
PIOS16	Tpios	PIOS16# setup to PIOR#, PIOW	(4 * Tsys) + 15		ns
PIOS16	Tpioh	PIOS16 hold from PIOR#, PIOW#	0		ns
ROE	Tpoed	ROE# delay from POE#, PIOR#	-2	+2	ns
RD[15:0] (input)	Tpsu	Data setup to POE#, PIOR#. Tpsu does not apply when PWAIT# is used to extend the cycle.	Tsys + 15		ns
RD[15:0] (input)	Tpsup	Data setup to PWAIT#. Tpsup applies only when PWAIT# is used to extend the cycle.	0		ns
RD[15:0]	Tph	Data hold from POE#, PIOR#	0		ns
PWAIT#	Tpwsu	PWAIT# setup to POE#, PWE#, PIOR#, PIOW# If PWAIT# does not meet this setup time, the cycle is not held.	4 * Tsys + 15		ns
POE#, PWE#, PIOR#, PIOW#	Tpwd	POE#, PWE#, PIOR#, PIOW# delay from PWAIT#.	3 * Tsys	4 * Tsys + 15	ns

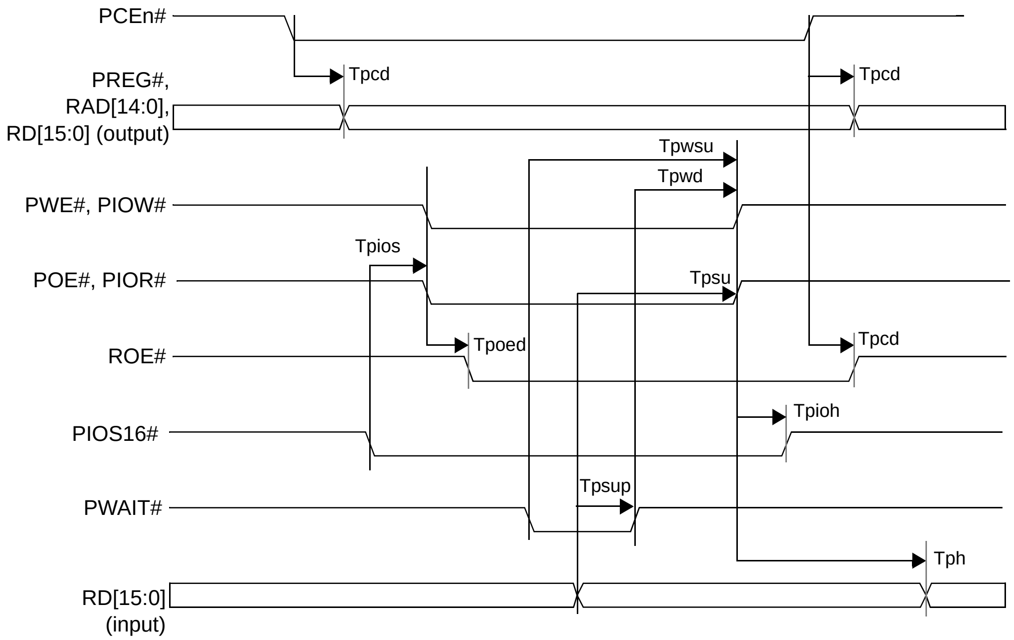


Figure 14-6. PCMCIA Host Adapter Timing

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-21. NOR Flash Timing (Asynchronous Mode)

Signal	Symbol	Parameter	Min	Max	Unit
RBE[1:0]#, ROE#, RAD[14:0]	Trcd	Delay from RCS*n*#	-2	+2	ns
RD[15:0] (read)	Trsu	Data setup to RCS*n*# Trsu does not apply when EWAIT# is used to extend the cycle.	15		ns
RD[15:0] (read)	Trsue	Data setup to EWAIT# Trsue applies only when EWAIT# is used to extend the cycle.	0		ns
RD[15:0] (read)	Trh	Data hold from RCS*n*#	0		ns
RD[15:0] (write)	Trod	Delay from RWE# to data out	-2	2	ns
EWAIT	Trwsu	EWAIT# setup to RCS*n*# for reads, or RWE# for writes. If EWAIT# does not meet this setup time, the cycle is not held.	(3 * Tsys) + 15		ns
RCS# (reads), RWE# (writes)	Trwd	Delay from EWAIT#	2 * Tsys	(3 * Tsys) + 15	

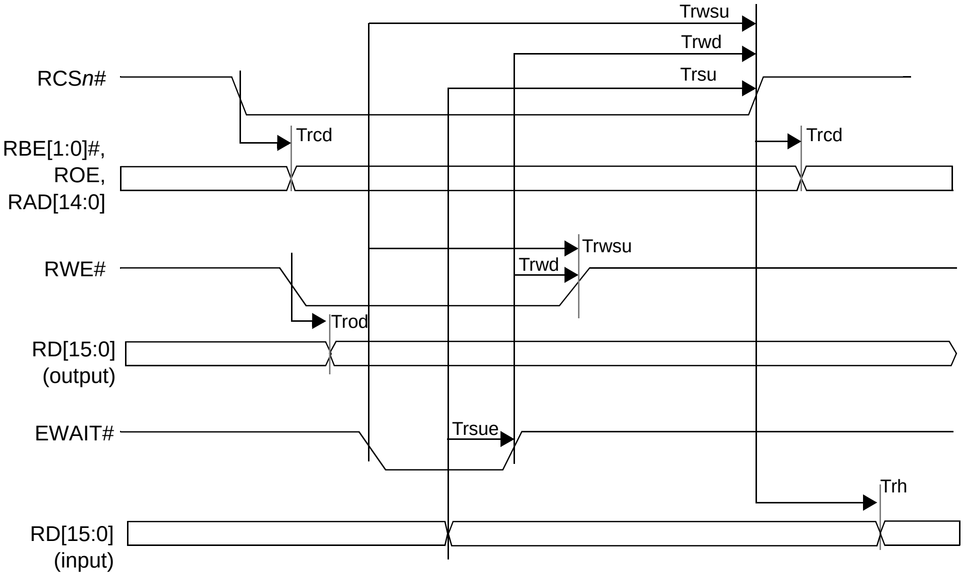


Figure 14-7. NOR Flash Timing (Asynchronous Mode)

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-22. NAND Flash Timing (Data Read)

Signal	Symbol	Parameter	Min	Max	Unit
RD[15:0], ROE#	Tih	Data input hold	0		ns

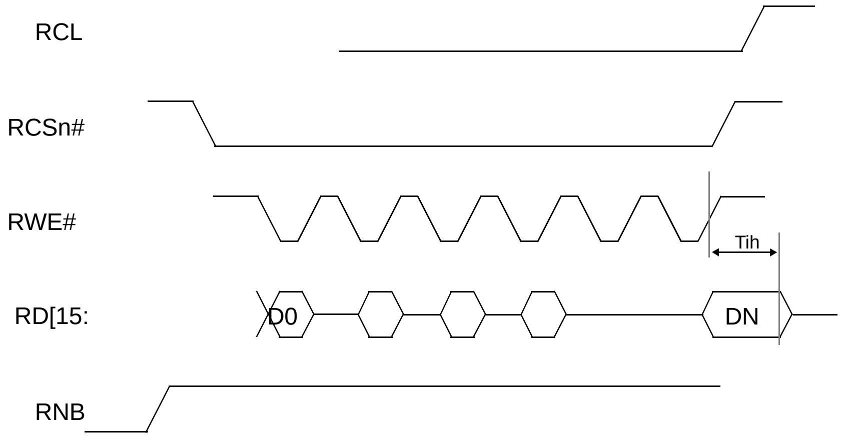


Figure 14-8. NAND Flash Data Read Timing

www.DataSheet4U.com

www.DataSheet4U.com

14.4.3 Synchronous Static Bus Timing

When the static bus is operating in synchronous mode all timing is referenced with respect to RCLK.

Table 14-23. Synchronous Static Bus

Signal	Symbol	Parameter	Min	Max	Unit
RCLK	Trclk	RCLK clock period This parameter is programmed in mem_stcfg0[DIV].	MAX[30, Tsys * 2]	Tsys * 16	ns
All outputs(Note1)	Tcd	Delay in output change from RCLK	Tsys	Tsys + 15	ns
All inputs(Note2) (except wait signals)	Tsu	Data setup to RCLK	15		ns
	Tdh	Data hold from RCLK	10		ns
EWAIT#	Twsu	Wait signal setup to RCLK for recognition of a state change in EWAIT#.	Tsys + 15		ns

Note1. Output signals: RAD[14:0], RD[15:0], RBE[1:0]#, ROE#, RWE#, RCS[3:0]#.

Note2. Input signals: RD[15:0].

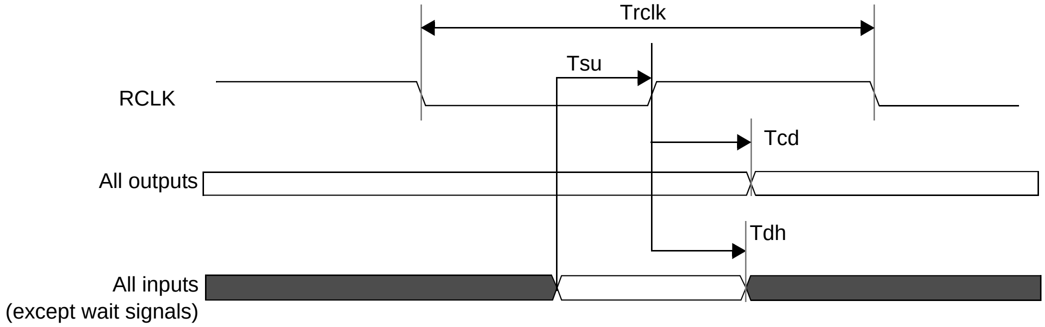


Figure 14-9. Static RAM, and I/O Device Timing (Synchronous Mode)

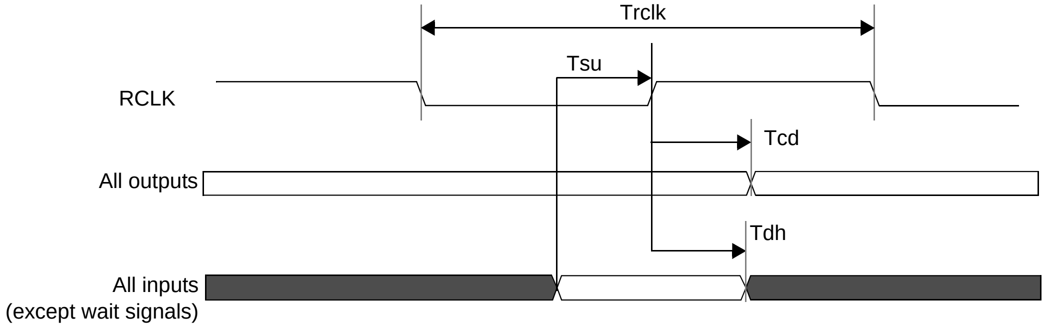


Figure 14-10. NOR Flash Timing (Synchronous Mode)

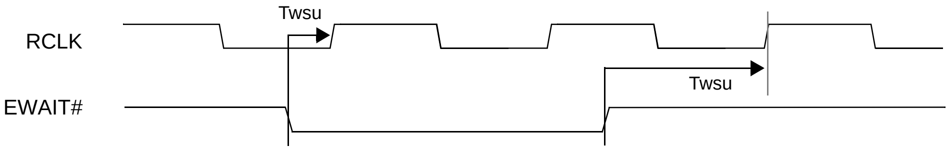


Figure 14-11. Wait Signal Recognition Timing for the Synchronous Static Bus

www.DataSheet4U.com

www.DataSheet4U.com

14.4.4 GPIO Input Timing Requirements

14.4.4.1 GPIO Input Edge Rate

For level-sensitive GPIO inputs, edge rates as slow as 5ms can be used. For edge-sensitive inputs (such as clocks and edge-triggered interrupts), a 20ns (or faster) edge rate must be used to ensure that noise does not cause false edges as the signal transitions through the threshold region.

14.4.4.2 GPIO Interrupt Timing

For system designs using GPIO signals as level-triggered interrupts, the signal level must be stable for at least 10ns in order for a signal state change to be detected. See Table 14-24 and Figure 14-12.

Table 14-24. GPIO Timing for Interrupts

Signal	Symbol	Parameter	Min	Max	Unit
GPIO[n]	Tmin	Minimum high or low time for interrupt. The level is programmable. This timing reflects the minimum active period for the level programmed.	10		ns

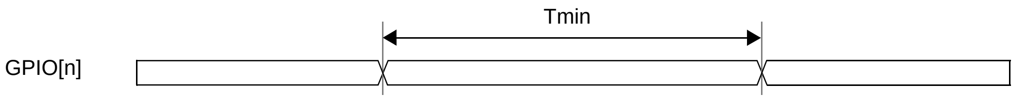


Figure 14-12. GPIO Interrupt Timing

www.DataSheet4U.com

www.DataSheet4U.com

14.4.5 I²S Timing (PSC in I²S Mode)

The I²S interface timing is shown in Table 14-25 and Figure 14-13.

Table 14-25. I²S Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
I2SCLK	Ti2s	I²S interface clock cycle time	40		ns
		I²S clock duty cycle	40	60	%
I2SDI, I2SDO, I2SWORD	Tid	Delay from I2SCLK to I2SDO and I2SWORD on output	0	10	ns
	Tisu	Setup before I2SCLK on input	20		ns
	Tih	Hold after I2SCLK on input	0 (Note1)		ns

Note1. I2SDI is shown to have a 0ns hold time relative to the falling edge. This design allows for the data source to transition data from the falling edge to the next data value.

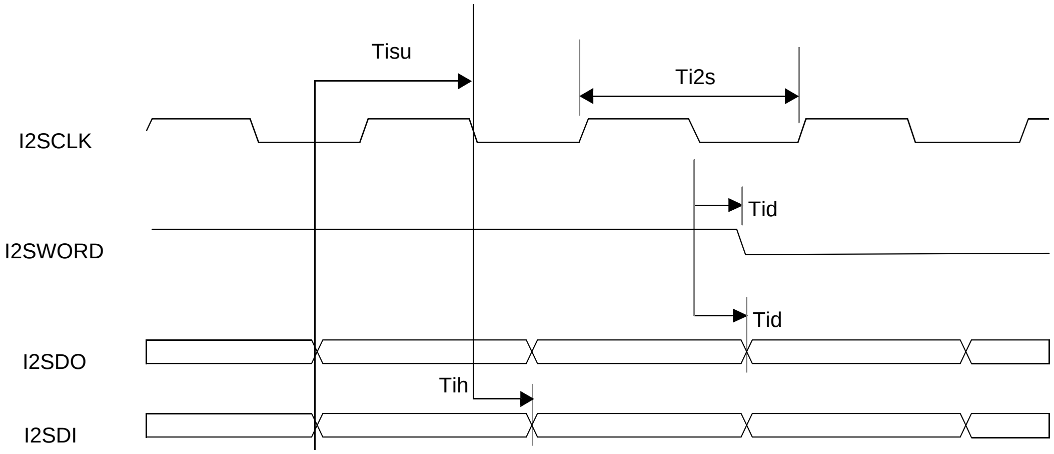


Figure 14-13. I²S Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.4.6 AC97 Timing (PSC in AC97 Mode)

The AC97 interface timing is shown in Table 14-26 and Figure 14-14.

Table 14-26. AC-Link Interface Timing (Note1)

Signal	Symbol	Parameter	Min	Nominal	Max	Unit
ACBCLK	Tabc	AC97 bit clock cycle time		12.288		MHz
	Tabh	AC97 bit clock high time	36		45	ns
	Tabl	AC97 bit clock low time	36		45	ns
ACSYNC	Tacs	AC97 sync cycle		48		KHz
	Tacsh	AC97 sync high time		1.3		µs
	Tacsl	AC97 sync low time		19.5		µs
ACSYNC ACDO ACDI	Tad	Delay from ACBCLK to ACSYNC and ACDO on output			15	ns
	Tasu	Setup before ACBCLK for ACDI	10			ns
	Tah	Hold after ACBCLK for ACDI	10			ns

Note1. ACRST# is an asynchronous signal controlled by software through **psc_ac97rst**. It has no relationship to the other AC97 signals.

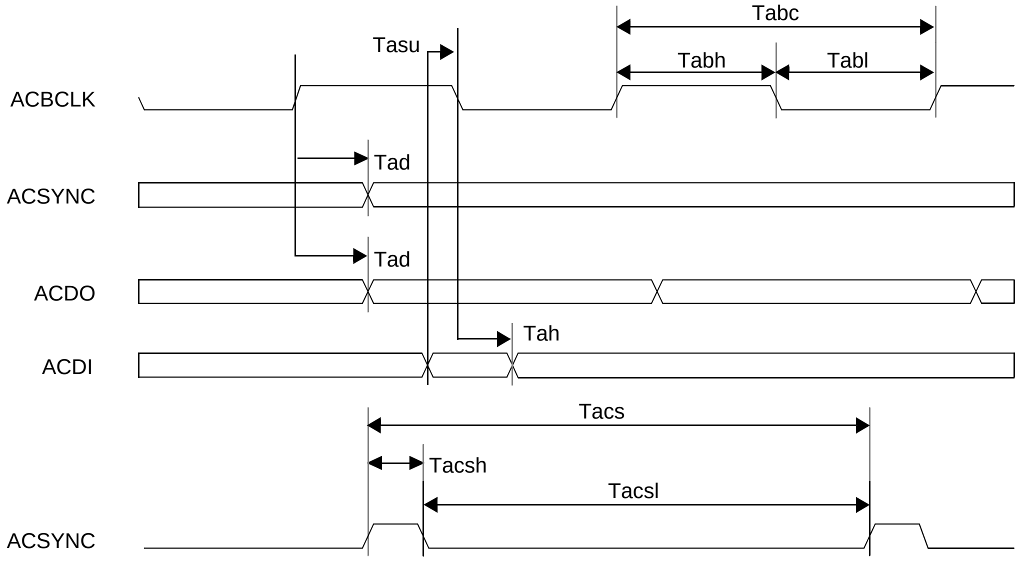


Figure 14-14. AC-Link Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.4.7 SPI Master Timing (PSC in SPI Master Mode)

The SPI master timing is shown in Table 14-27 and Figure 14-15.

Table 14-27. SPI Master Timing

Signal	Symbol	Parameter	Min	Max	Unit
SPISEL#	Ttb	Time between DMA transfers	3 * (*pcsn_mainclk* period)		ns
SPICLK	Tmcp	SPI master clock		24	MHz
SPICLK	Tchl	Clock high/low time	0.49 * SPICLK	0.51 * SPICLK	ns
SPIMISO	Tdis	Data input setup	20		ns
SPIMISO	Tdih	Data input hold	0.5 * (*pcsn_mainclk* period)		ns
SPIMOSI	Tdos	Data output setup from clock edge	-10	10	ns
SPIMOSI	Tdvs	Data valid from SPISEL# assertion	Tchl + Tdos		ns
SPIMOSI	Tdov	Data output valid	Tchl - Tdos		ns
SPIMOSI	Tdoh	Data output hold	Tchl + Tdos		ns

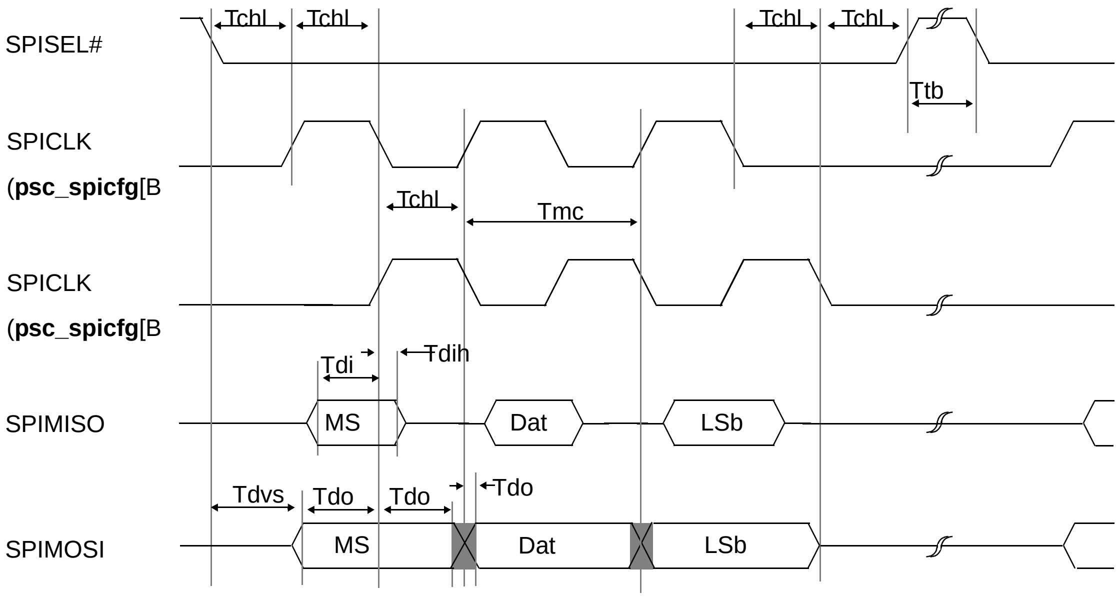


Figure 14-15. SPI Master Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.4.8 SPI Slave Timing (PSC in SPI Slave Mode)

The SPI slave timing is shown in Table 14-28, Figure 14-16 on page 463, and Figure 14-17 on page 463.

Table 14-28. SPI Slave Timing

Signal	Symbol	Parameter	Min	Max	Unit
SPICLK	Tspc	SPI clock period	6 * *pcsn_mainclk* period		ns
SPICLK	Tchl	High/low time	3 * *pcsn_mainclk* period		ns
SPISEL#, SPICLK	Tsla	Select negation lag time from clock edge	1 * *pcsn_mainclk* period		ns
SPISEL#	Tspw	Slave select pulse width	3 * *pcsn_mainclk* period		ns
SPISEL#, SPICLK	Tsle	Select assertion lead time to clock edge. Tsle applies only to a non-clock-delay configuration (**psc*n*_spicfg**[CDE]] = 0). See figure Figure 14-16 on page 463.	1 * *pcsn_mainclk* period		ns
SPISEL#, SPIMISO	Tdvs	Data valid from SPISEL# assertion. Tdvs applies only to a clock-delay configuration (**psc*n*_spicfg**[CDE]] = 1). See figure Figure 14-17 on page 463.		(2 * *pcsn_mainclk* period) + 10	ns
SPIMISO	Tdos	Data output setup from clock edge	- [(2 * *pcsn_mainclk* period) + 10]	(2 * *pcsn_mainclk* period) + 10	ns
SPIMISO	Tdov	Data output valid	Tchl - Tdos		ns
SPIMISO	Tdoh	Data output hold	Tchl + Tdos		ns
SPIMOSI	Tdis	Data input setup	2 * *pcsn_mainclk* period		ns
SPIMOSI	Tdih	Data input hold	2 * *pcsn_mainclk* period		ns

www.DataSheet4U.com

www.DataSheet4U.com

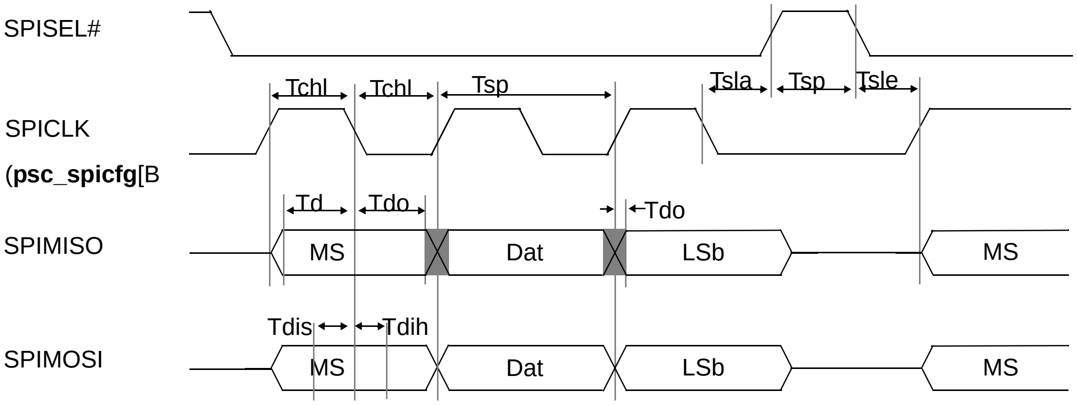


Figure 14-16. SPI Slave Timing without Clock Delay (pcs*n*_spicfg[CDE] = 0)

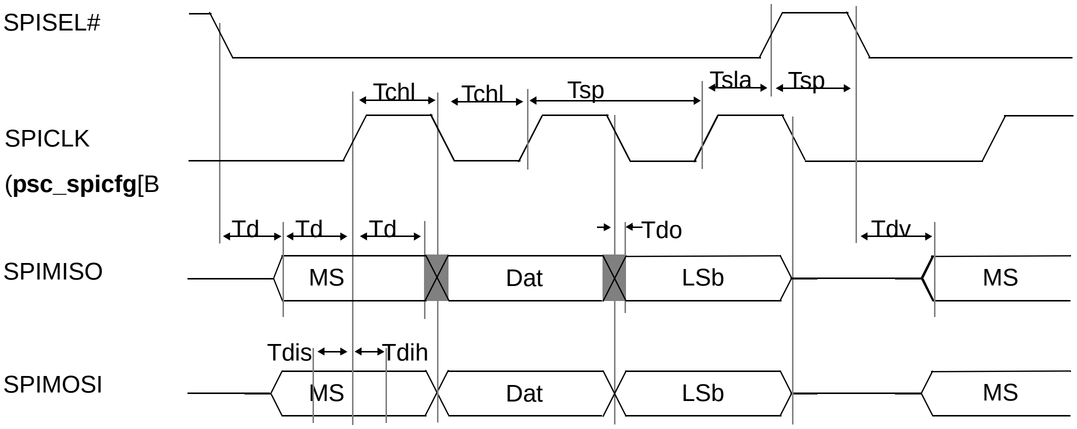


Figure 14-17. SPI Slave Timing with Clock Delay (pcs*n*_spicfg[CDE] = 1)

www.DataSheet4U.com

www.DataSheet4U.com

14.4.9 SMBus Timing (PSC in SMBus Mode)

The SMBus timing is shown in Table 14-29 and Figure 14-18.

Table 14-29. SMBus Timing

Signal	Symbol	Parameter	Min	Max	Unit
SCL		Clock frequency input (slave)	10	100	KHz
SCL		Clock frequency output (master)	10	100	KHz
SDA	Tbf	Bus free time between transfers	4.7		µs
SCL	Tlo	Low time	4.7		µs
SCL	Thi	High time	4.0	5.0	µs
SDA, SCL	Tstrs	Start condition setup	4.7		µs
SDA, SCL	Tstrh	Start condition hold	4.0		µs
SDA	Tdh	Data hold	300		ns
SDA	Tds	Data setup	250		ns
SDA, SCL	Trise	Rise time		1000	ns
SDA, SCL	Tfall	Fall time		300	ns
SDA, SCL	Tstps	Stop condition setup	4.0		µs

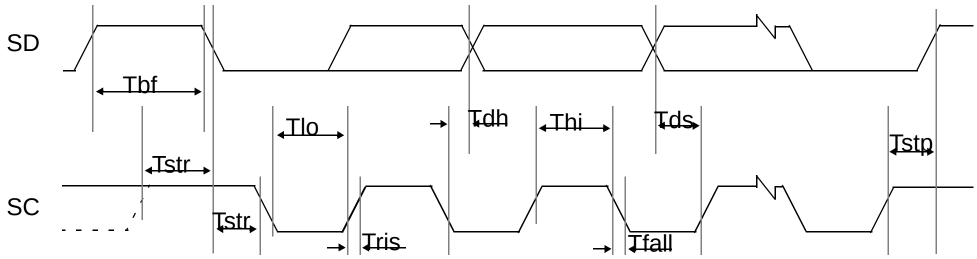


Figure 14-18. SMBus Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.4.10 LCD Controller Timing

The LCD controller timing is shown in Table 14-30 and Figure 14-19. The LCD_BIAS signal and LCD data signals change state on the inactive edge of the pixel clock (LCD_PCLK). Note that the diagram shows the timing for **lcd_clockctrl**[IC] = 0. (See the invert-pixel-clock bit description in Section 9.2.4 "LCD Clock Control Register" on page 305.)

Table 14-30. LCD Controller Timing

Signal	Symbol	Parameter	Min	Max	Unit
LCD_PCLK, LCD_D[23:0], LCD_BIAS	Tpcd	Pixel clock inactive edge to LCD_D[23:0] and LCD_BIAS valid.	-5	5	ns

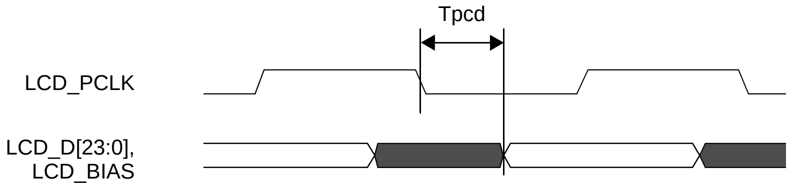


Figure 14-19. LCD Controller Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.4.11 EJTAG Interface Timing

Table 14-31. EJTAG Interface Timing

Signal	Symbol	Parameter	Min	Max	Unit
TCK	Tec	EJTAG TCK cycle time	40		ns
	Tech	TCK high time	10		ns
	Tecl	TCK low time	10		ns
TMS TDI	Tesu	Setup before TCK for TMS and TDI	5		ns
	Teh	Hold after TCK for TMS and TDI	3		ns
TDO	Teco	Delay from TCK to TDO on output		15	ns
	Tecz	Delay from TCK to TDO tristate		15	ns
TRST#	Trstl	TRST# low time	25		ns

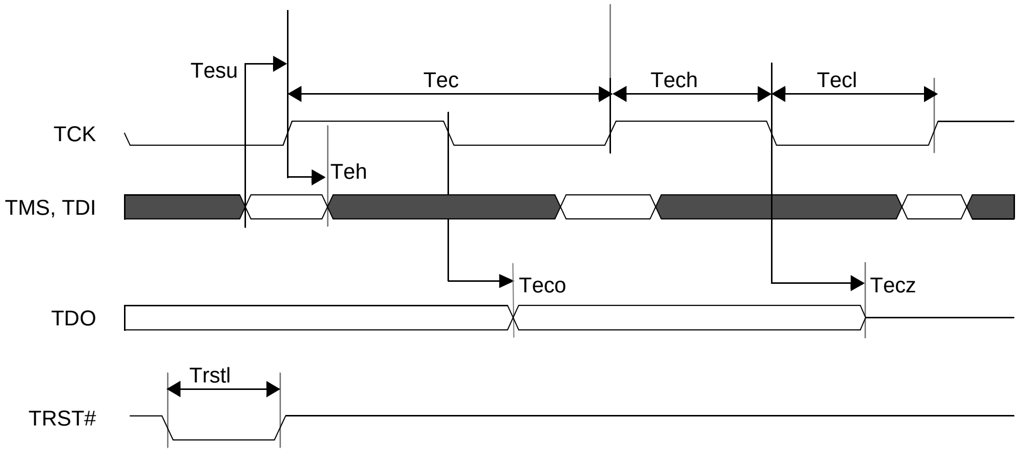


Figure 14-20. EJTAG Timing Diagram

www.DataSheet4U.com

www.DataSheet4U.com

14.5 Power-up and Reset Timing

This section provides the timing specifications for the power-up sequence, and the hardware and runtime reset sequences. (See Section 11.0 "Power-up, Reset and Boot" on page 399 for functional descriptions of the sequences.)

14.5.1 Power-up Sequence Timing

Figure 14-21 shows the power-up sequence, including timing parameters as defined in Table 14-32.

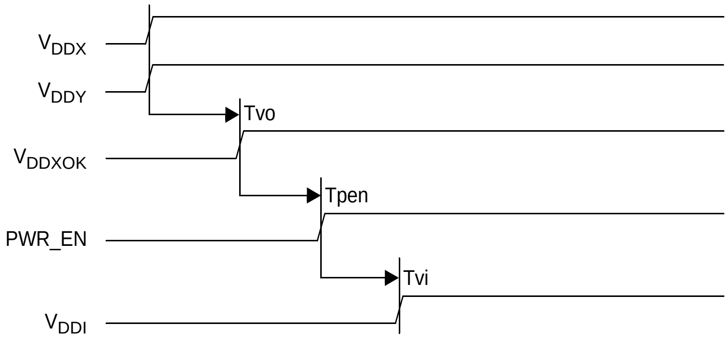


Figure 14-21. Power-up Sequence

Table 14-32. Power-up Timing Parameters

Parameter	Description	Min	Max
T_{vo}	V_{DDX} at 90% of nominal to V_{DDXOK} asserted	0ns	
T_{pen}	V_{DDXOK} asserted to PWR_EN driven high		30ns
T_{vi}	PWR_EN to V_{DDI} stable		20ms

www.DataSheet4U.com

www.DataSheet4U.com

14.5.2 Hardware Reset Timing

Figure 14-22 shows the hardware reset sequence, including timing parameters as defined in Table 14-33.

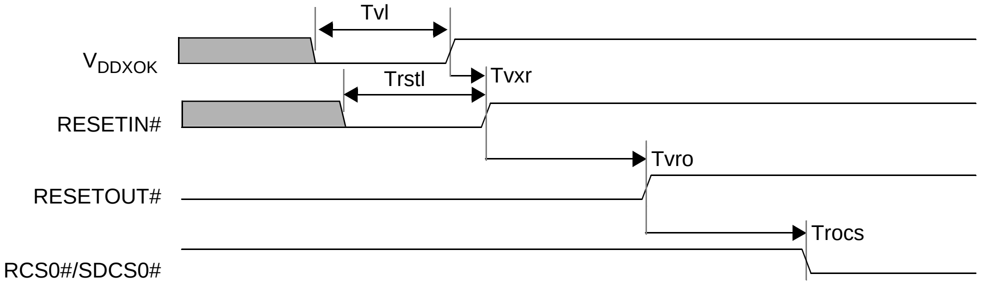


Figure 14-22. Hardware Reset Sequence

Table 14-33. Hardware Reset Timing Parameters

Parameter	Description	Min	Typical	Max
T_{vxr}	V_{DDXOK} asserted to RESETIN# deasserted	0ns		System Dependent
T_{vl}	V_{DDXOK} low time	1µs		
T_{rstl}	RESETIN# low time	1µs		
T_{vro}	RESETIN# to RESETOUT# delay MAX = max[750ns, 170ms - Tvxr]	600ns		see desc.
T_{rocs}	RESETOUT# to RCS0/SDCS0# asserted.		135ns	1µs

www.DataSheet4U.com

www.DataSheet4U.com

14.5.3 Runtime Reset Timing

Figure 14-23 shows the runtime reset sequence, including timing parameters as defined in Table 14-34.

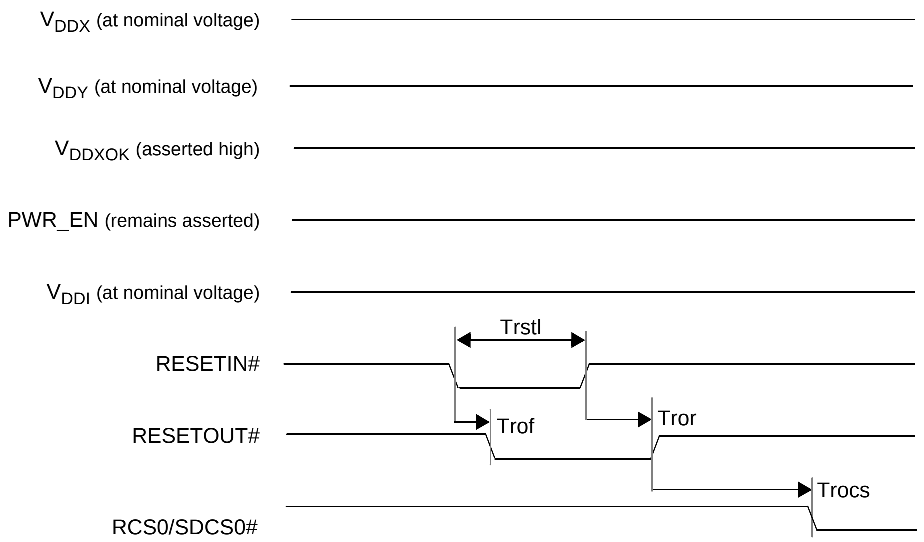

V_{DDX} (at nominal voltage)
V_{DDY} (at nominal voltage)
V_{DDXOK} (asserted high)
PWR_EN (remains asserted)
V_{DDI} (at nominal voltage)
RESETIN#
RESETOUT#
RCS0/SDCS0#
Trstl
Trof
Tror
Trocs

Figure 14-23. Runtime Reset Sequence

Table 14-34. Runtime Reset Timing Parameters

Parameter	Description	Min	Typical	Max
T_{rstl}	RESETIN# low time	1µs		
T_{rof}	RESETIN# falling to RESETOUT# falling MAX: 25ns + (0.5 * (CPU Clock/2))			see desc.
T_{ror}	RESETIN# rising to RESETOUT# rising MAX: 25ns + (0.5 * (CPU Clock/2)) + (120 * CPU Clock)	120 CPU clocks		see desc.
T_{rocs}	RESETOUT# to RCS0#/SDCS0# asserted. The timing values shown assume a 400MHz CPU clock.		65ns	500ns

www.DataSheet4U.com

www.DataSheet4U.com

14.6 Asynchronous Signals

- GPIO

The GPIO signals are driven by software. However, when GPIO signals are used as inputs, there are timing requirements to ensure signal state changes are recognized cleanly; see Section 14.4.4 "GPIO Input Timing Requirements" on page 458.

- 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.

14.7 External Clock Specifications

The EXTCLK[1:0] external clocks have a maximum frequency rating of (F_{max} / 16), where F_{max} is the maximum frequency rating for the part. Table 14-35 provides the EXTCLK[1:0] specifications.

Table 14-35. External Clock EXTCLK[1:0] Specifications

	324MHz		396MHz		492MHz		600MHz		
Characteristic	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
Frequency		20.25		24.75		31.25		37.5	MHz
Frequency jitter		4		4		4		4	%
Duty cycle	40	60	40	60	40	60	40	60	%

14.8 Crystal Specifications

Load capacitors for the external oscillators are integrated into the Au1210 and Au1250 processors so no external circuitry is required when using the specified crystal. For design layout considerations concerning the crystals, see Section 14.9.1 "Crystal Layout" on page 472.

Table 14-36 provides the specification for the parallel resonant 12MHz crystal to be placed between XTI12 and XTO12.

Table 14-36. 12MHz Crystal Specification

Specification	Min	Typical	Max	Unit
Resonant Frequency		12		MHz
Frequency Stability			±100	ppm
Motional Resistance			60	Ω
Shunt Capacitance		<5	7	pF
Load Capacitance (This capacitance is integrated on the Au1210/Au1250.)	8	12	20	pF
Drive Level			100	µW
Crystal Type	AT Cut			

www.DataSheet4U.com

www.DataSheet4U.com

Table 14-37 provides the specification for the parallel resonant 32KHz crystal to be placed between XTI32 and XTO32.

Table 14-37. 32.768KHz Crystal Specification

Specification	Min	Typical	Max	Unit
Resonant Frequency		32.768		KHz
Equivalent Series Resistance			50	kΩ
Shunt Capacitance		1.5	2.0	pF
Load Capacitance (This capacitance is integrated on the Au1210/Au1250.)	6		12	pF
Motional Capacitance		3	4	fF
Drive Level			1	µW
Quality Factor	40k			
Crystal Type	Tuning Fork			

Table 14-38 provides the specification for the USB crystal to be placed between USBXI and USBXO. The specification also requires an output differential voltage of no less than 500mV with respect to USBXI.

Table 14-38. USB Crystal Specification

Specification	Min	Typical	Max	Unit
Resonant Frequency		48		MHz
Frequency Stability			±100	ppm
Peak Jitter			±100	ps
Equivalent Series Resistance			50	Ω
Shunt Capacitance			7.0	pF
Load Capacitance	16	27		pF
Drive Level			50	µW
Crystal Type	AT Cut (Fundamental)			

www.DataSheet4U.com

www.DataSheet4U.com

14.9 System Design Considerations

This section provides information for system-level design issues.

14.9.1 Crystal Layout

The crystal layouts are critical. Without using vias, place traces directly over a ground plane on the top layer with keep-outs on all surrounding sides. Trace lengths should be less than 0.5 inches, and trace widths should be set to the minimum signal trace width for the design. Be sure not to allow other signals to come within 0.025 inches of these sensitive analog signals.

14.9.2 Decoupling Recommendations

This section provides recommendations for minimizing noise in a system. Specific decoupling requirements are system dependent.

To filter noise on the power supplies, V_{DDX}, V_{DDY} and V_{DDI}, as well as XPWR12 and XPWR32, should be bypassed to ground using 10µF capacitors: For each of the four sides of the package, place a capacitor within 0.5 inches.

To filter high-frequency noise, capacitors in the 10nF range should be placed under the package:

- For minimal high-frequency decoupling, use six to eight 10nF capacitors.
- For systems requiring a broader spectrum of high-frequency noise be filtered, use four 15nF and four 6.8nF capacitors.

www.DataSheet4U.com

www.DataSheet4U.com

15 Packaging, Pinout, and Ordering Information

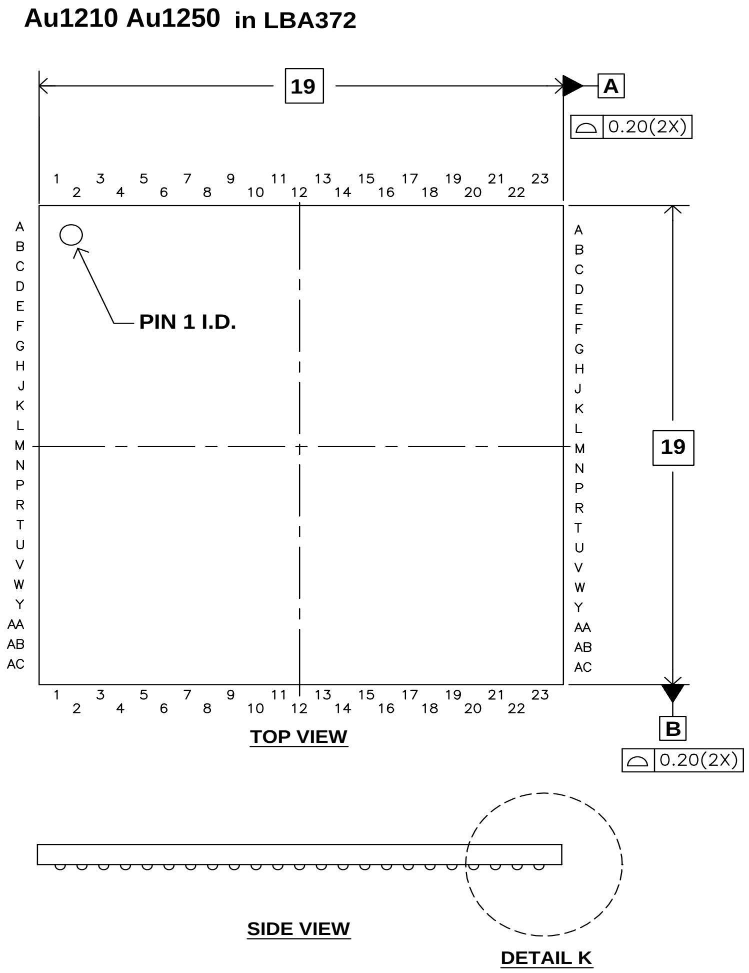
This chapter provides information about the Au1210™ and Au1250™ processors' package and pin assignments, as well as providing ordering information. The contents of the chapter are organized as follows:

- The package dimensions are shown in Figure 15-1 on page 474. Both the Au1210 and Au1250 are packaged in a 372-pin LBA device.
- Table 15-1 (starting on page 476) is the connection diagram showing the pin and signal placement on the package. For pins that provide multiple signal functions, the default signals are shown first and in bold, followed by any alternate signals. The black square in the upper-left hand corner indicates where the device is keyed.
- The pin assignment listing ordered by pin number starts on page 478.
- The pin assignment listing sorted by default signal name starts on page 482.
- The pin assignment listing sorted by alternate signal name starts on page 485.
- Ordering information is supplied on page 486.

www.DataSheet4U.com

www.DataSheet4U.com

15.1 Packaging



NOTES

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M-1994 .
2. ALL DIMENSIONS ARE IN MILLIMETERS .
3. BALL POSITION DESIGNATION PER JESD 95-1, SPP-010.

△4. BALL DIAMETER IS MEASURED AT ITS MAXIMUM DIMENSION IN A PLANE PARALLEL TO DATUM C.

5. THIS PACKAGE IS DIMENSIONED IN THE MANNER OF JEDEC OUTLINE MO-205 REV F.

Figure 15-1. Package Dimensions for the Au1210™ and Au1250™ Processors

www.DataSheet4U.com

www.DataSheet4U.com

Au1210 Au1250 LBA372

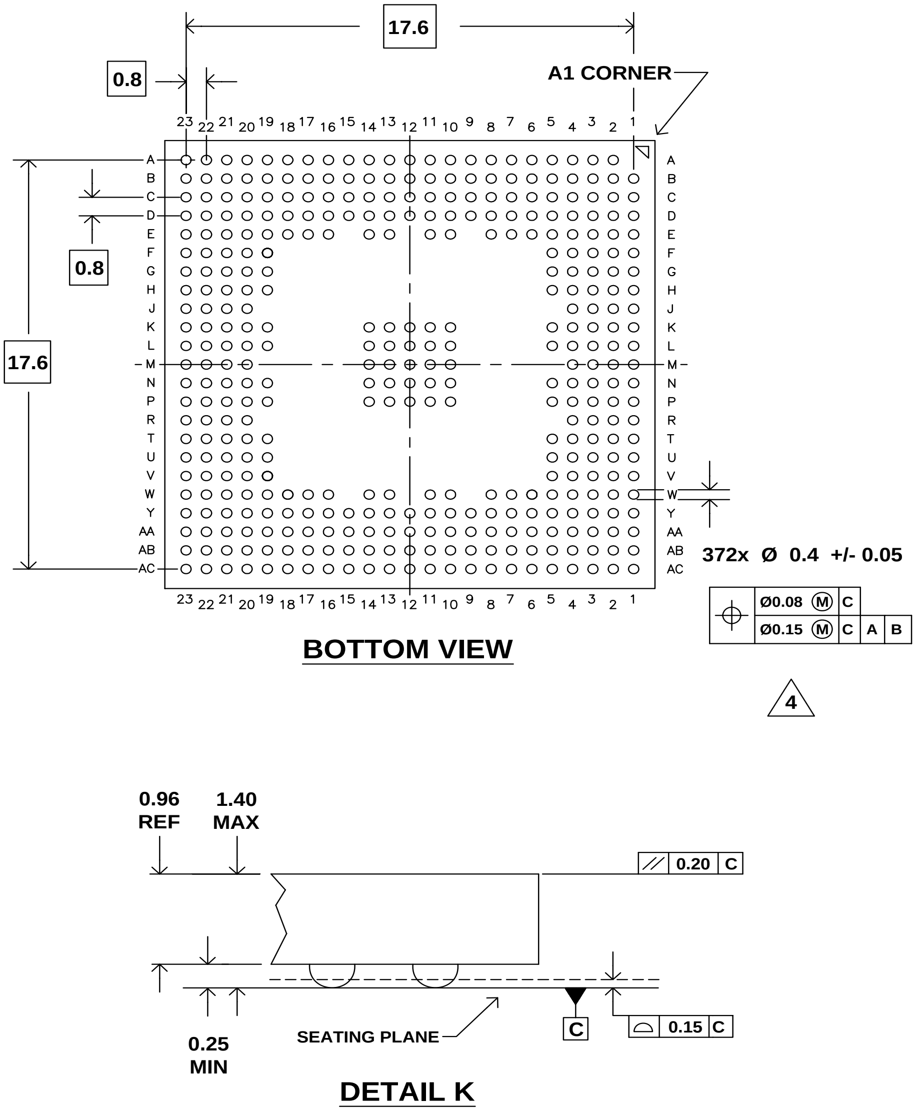


Figure 15-2. (Package Dimensions Continued)

www.DataSheet4U.com

www.DataSheet4U.com

15.2 Pin Assignments

Table 15-1. Connection Diagram for the Au1210™ and Au1250™ Processors—Top View

	1	2	3	4	5	6	7	8	9	10	11	12
A	—	FWTOY#	XTI32	XTI12	**GPIO[8]/** SD0_DAT1	PWR_EN	TC0	TC1	GPIO[7]	BOOT[0]	**GPIO[213]/** CIM_FS	**GPIO[209]/** CIM_D9
B	**GPIO[17]/** SD0_CLK	WAKE#	XTO32	XTO12	**GPIO[28]/** SD0_CMD	V_{SS}	RESETIN#	V_{SS}	**GPIO[0]/** CIM_D0	V_{SS}	**GPIO[214]/** CIM_CLK	V_{SS}
C	HD_CS1#	V_{SS}	XPWR32	XAGND32	XPWR12	XAGND12	**GPIO[1]/** CIM_D1	**GPIO[3]/** EXTCLK1	TC2	GPIO[5]	FTM	**GPIO[212]/** CIM_LS
D	PREG#	HD_CS0#	**GPIO[26]/** SD0_DAT3	V_{DDX}	RNB	**GPIO[6]/** SD0_DAT2	**GPIO[4]/** DMA_REQ0	**GPIO[2]/** EXTCLK0	TC3	V_{DDX}	TESTEN	not connected
E	RCS3#	V_{SS}	PCE1#	**GPIO[19]/** SD0_DAT0	V_{DDX}	V_{DDX}	V_{DDX}	V_{DDX}		V_{DDI}	V_{DDI}	
F	**POE#/** SD1_DAT1	**PIOW#/** SD1_CLK	**PIOS16#/** SD1_DAT2	PCE2#	V_{DDX}							
G	**PIOR#/** SD1_CMD	V_{SS}	RCS2#	RCS1#	V_{DDX}							
H	RALE	RBE1#	RCS0#	**PWE#/** SD1_DAT3	V_{DDX}							
J	RWE#	V_{SS}	**PWAIT#/** SD1_DAT0	RBE0#								
K	RAD11	ROE#	RCLE	V_{DDX}	V_{DDI}					V_{SS}	V_{SS}	V_{SS}
L	RAD12	V_{SS}	RCLK	RAD10	V_{DDI}					V_{SS}	V_{SS}	V_{SS}
M	RAD14	RAD9	RAD13	EWAIT#						V_{SS}	V_{SS}	V_{SS}
N	RAD6	V_{SS}	RAD8	RAD5	V_{DDI}					V_{SS}	V_{SS}	V_{SS}
P	RAD4	RAD3	RAD7	V_{DDX}	V_{DDI}					V_{SS}	V_{SS}	V_{SS}
R	RAD1	V_{SS}	RAD0	RD15								
T	RAD2	RD7	RD6	RD13	V_{DDX}							
U	RD14	V_{SS}	RD5	RD11	V_{DDX}							
V	RD12	RD4	RD2	RD1	V_{DDX}							
W	RD3	V_{SS}	RD9	DDQ15	V_{DDY}	V_{DDY}	V_{DDY}	V_{DDY}		V_{DDI}	V_{DDI}	
Y	RD10	RD8	DDQ14	V_{DDY}	DDQ30	DDQ29	DDQ20	DDQ21	DDQ6	V_{DDY}	DDQ10	DDQS1
AA	RD0	V_{SS}	DDQ16	DDQ1	DDQ2	V_{DDY}	DDQ4	DDQ22	DDQ23	DDQ27	DDQ9	V_{DDY}
AB	DDQ31	DDQ0	V_{SS}	DDQ18	V_{SS}	DDQ12	V_{SS}	DDQ11	V_{SS}	DDQ25	V_{SS}	DDM2
AC	V_{SS}	DDQ17	DDQ13	DDQ19	DDQ3	DDQ28	DDQ5	DDQ26	DDQ7	DDQS2	DDQ24	DDQ8
	1	2	3	4	5	6	7	8	9	10	11	12

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-2. (Connection Diagram Continued)

13	14	15	16	17	18	19	20	21	22	23	
GPIO[208]/ CIM_D8	**GPIO[205]/** CIM_D5	**GPIO[203]/** CIM_D3	**GPIO[30]/** U1RXD	TDO	**GPIO[9]/** U1CTS#	**GPIO[15]/** U1TXD	**GPIO[13]/** U1RTS#	**GPIO[22]/** PSC1_D1	**GPIO[24]/** PSC1_CLK	V_{SS}	**A**
GPIO[206]/ CIM_D6	V_{SS}	**GPIO[202]/** CIM_D2	V_{SS}	TDI	V_{SS}	**GPIO[12]/** DMA_REQ1	V_{SS}	**GPIO[23]/** PSC1_EXTCLK	V_{SS}	**GPIO[18]/** PSC0_D0	**B**
GPIO[207]/ CIM_D7	**GPIO[204]/** CIM_D4	**GPIO[29]/** U0RXD	TCK	**GPIO[10]/** U1DSR#	**PSC1_SYNC1/** GPIO[21]	**GPIO[11]/** PSC1_D0	**GPIO[20]/** PSC1_SYNC0	**GPIO[31]/** PSC0_D1	**PSC0_SYNC1/** GPIO[16]	**GPIO[25]/** PSC0_CLK	**C**
BOOT[1]	V_{DDX}	TMS	TRST#	**GPIO[27]/** U0TXD	**GPIO[14]/** U1DTR#	V_{DDX}	**GPIO[215]/** PSC0_SYNC0	USBV_{DDX}	USBATEST	USBV_{DDI}	**D**
V_{DDI}	V_{DDI}		V_{DDX}	V_{DDX}	V_{DDX}	USBV_{SS}	USBV_{SS}	USBV_{DDX}	USBDP	USBDM	**E**
						USBV_{SS}	USBV_{SS}	USBV_{DDX}	USBXI	USBXO	**F**
						USBV_{SS}	USBV_{SS}	USBV_{DDX}	USBHP	USBHM	**G**
						USBV_{DDX}	USBV_{DDX}	USBV_{DDX}	USBREXT	USBVBUS	**H**
							USBV_{DDX}	USBV_{DDX}	USBOTGID	USBV_{DDI}	**J**
V_{SS}	V_{SS}					V_{DDI}	V_{DDX}	**U1RI#/** LCD_PWM1	**U1DCD#/** LCD_PWM0	LCD_FCLK	**K**
V_{SS}	V_{SS}					V_{DDI}	LCD_LCLK	LCD_PCLK	V_{SS}	LCD_BIAS	**L**
V_{SS}	V_{SS}						not connected	**GPIO[200]/** LCD_D0	**GPIO[201]/** LCD_D1	LCD_D2	**M**
V_{SS}	V_{SS}					V_{DDI}	LCD_D5	LCD_D4	V_{SS}	LCD_D3	**N**
V_{SS}	V_{SS}					V_{DDI}	V_{DDX}	**GPIO[210]/** LCD_D8	LCD_D7	LCD_D6	**P**
							LCD_D12	LCD_D10	V_{SS}	LCD_D9	**R**
						V_{DDX}	**GPIO[211]/** LCD_D16	LCD_D14	LCD_D13	LCD_D11	**T**
						V_{DDX}	V_{DDX}	LCD_D18	V_{SS}	LCD_D15	**U**
						V_{DDX}	RESETOUT#	LCD_D21	LCD_D19	LCD_D17	**V**
V_{DDI}	V_{DDI}		V_{DDY}	V_{DDY}	V_{DDY}	V_{DDY}	DA1	LCD_D23	V_{SS}	LCD_D20	**W**
DDM1	V_{DDY}	DCAS#	DCS1#	DRAS#	DBA0	DA10	VDDY	DA3	DRVSEL	LCD_D22	**Y**
DVREF	DDM0	DA13	DCKE	DCS0#	V_{DDY}	DBA1	DA0	DA2	V_{SS}	V_{DDXOK}	**AA**
V_{SS}	DWE#	V_{SS}	DCK1	V_{SS}	DA12	V_{SS}	DA8	V_{SS}	DA5	DA4	**AB**
DDQS3	DDQS0	DDM3	DCK1#	DCK0#	DCK0	DA11	DA9	DA7	DA6	V_{SS}	**AC**
13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-3. Pin Assignments - Sorted by Pin Number

Pin Number	Default Signal	Alternate Signal
A2	FWTOY	
A3	XTI32	
A4	XTI12	
A5	GPIO[8]	SD0_DAT1
A6	PWR_EN	
A7	TC0	
A8	TC1	
A9	GPIO[7]	
A10	BOOT[0]	
A11	GPIO[213]	CIM_FS
A12	GPIO[209]	CIM_D9
A13	GPIO[208]	CIM_D8
A14	GPIO[205]	CIM_D5
A15	GPIO[203]	CIM_D3
A16	GPIO[30]	U1RXD
A17	TDO	
A18	GPIO[9]	U1CTS#
A19	GPIO[15]	U1TXD
A20	GPIO[13]	U1RTS#
A21	GPIO[22]	PSC1_D1
A22	GPIO[24]	PSC1_CLK
A23	V_{SS}	
B1	GPIO[17]	SD0_CLK
B2	WAKE#	
B3	XTO32	
B4	XTO12	
B5	GPIO[28]	SD0_CMD
B6	V_{SS}	
B7	RESETIN#	
B8	V_{SS}	
B9	GPIO[0]	CIM_D0
B10	V_{SS}	
B11	GPIO[214]	CIM_CLK
B12	V_{SS}	
B13	GPIO[206]	CIM_D6
B14	V_{SS}	
B15	GPIO[202]	CIM_D2
B16	V_{SS}	
B17	TDI	
B18	V_{SS}	
B19	GPIO[12]	DMA_REQ1
B20	V_{SS}	
B21	GPIO[23]	PSC1_EXTCLK
B22	V_{SS}	
B23	GPIO[18]	PSC0_D0
C1	HD_CS1	
C2	V_{SS}	
C3	XPWR32	
C4	XAGND32	
C5	XPWR12	
C6	XAGND12	
C7	GPIO[1]	CIM_D1
C8	GPIO[3]	EXTCLK1
C9	TC2	
C10	GPIO[5]	
C11	FTM	
C12	GPIO[212]	CIM_LS
C13	GPIO[207]	CIM_D7
C14	GPIO[204]	CIM_D4
C15	GPIO[29]	U0RXD
C16	TCK	
C17	GPIO[10]	U1DSR#
C18	PSC1_SYNC1	GPIO[21]
C19	GPIO[11]	PSC1_D0
C20	GPIO[20]	PSC1_SYNC0
C21	GPIO[31]	PSC0_D1
C22	PSC0_SYNC1	GPIO[16]
C23	GPIO[25]	PSC0_CLK
D1	PREG#	
D2	HD_CS0	
D3	GPIO[26]	SD0_DAT3
D4	V_{DDX}	
D5	RNB	
D6	GPIO[6]	SD0_DAT2
D7	GPIO[4]	DMA_REQ0
D8	GPIO[2]	EXTCLK0
D9	TC3	
D10	V_{DDX}	
D11	TESTEN	
D12	not connected	
D13	BOOT[1]	
D14	V_{DDX}	
D15	TMS	
D16	TRST#	
D17	GPIO[27]	U0TXD
D18	GPIO[14]	U1DTR#
D19	V_{DDX}	
D20	GPIO[215]	PSC0_SYNC0
D21	USBV_{DDX}	
D22	USBATEST	
D23	USBV_{DDI}	
E1	RCS3#	
E2	V_{SS}	
E3	PCE1#	
E4	GPIO[19]	SD0_DAT0
E5	V_{DDX}	
E6	V_{DDX}	
E7	V_{DDX}	
E8	V_{DDX}	
E10	V_{DDI}	
E11	V_{DDI}	
E13	V_{DDI}	
E14	V_{DDI}	
E16	V_{DDX}	
E17	V_{DDX}	
E18	V_{DDX}	
E19	USBV_{SS}	
E20	USBV_{SS}	
E21	USBV_{DDX}	
E22	USBDP	
E23	USBDM	
F1	POE#	SD1_DAT1
F2	PIOW#	SD1_CLK
F3	PIOS16#	SD1_DAT2
F4	PCE2#	
F5	V_{DDX}	
F19	USBV_{SS}	
F20	USBV_{SS}	
F21	USBV_{DDX}	
F22	USBXI	
F23	USBXO	
G1	PIOR#	SD1_CMD

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-3 "Pin Assignments - Sorted by Pin Number" (Continued)

Pin Number	Default Signal	Alternate Signal
G2	V_{SS}	
G3	RCS2#	
G4	RCS1#	
G5	V_{DDX}	
G19	USBV_{SS}	
G20	USBV_{SS}	
G21	USBV_{DDX}	
G22	USBHP	
G23	USBHM	
H1	RALE	
H2	RBE1#	
H3	RCS0#	
H4	PWE#	SD1_DAT3
H5	V_{DDX}	
H19	USBV_{DDX}	
H20	USBV_{DDX}	
H21	USBV_{DDX}	
H22	USBREXT	
H23	USBVBUS	
J1	RWE#	
J2	V_{SS}	
J3	PWAIT#	SD1_DAT0
J4	RBE0#	
J20	USBV_{DDX}	
J21	USBV_{DDX}	
J22	USBOTGID	
J23	USBV_{DDI}	
K1	RAD11	
K2	ROE#	
K3	RCLE	
K4	V_{DDX}	
K5	V_{DDI}	
K10	V_{SS}	
K11	V_{SS}	
K12	V_{SS}	
K13	V_{SS}	
K14	V_{SS}	
K19	V_{DDI}	
K20	V_{DDX}	
K21	U1RI#	LCD_PWM1
K22	U1DCD#	LCD_PWM0
K23	LCD_FCLK	
L1	RAD12	
L2	V_{SS}	
L3	RCLK	
L4	RAD10	
L5	V_{DDI}	
L10	V_{SS}	
L11	V_{SS}	
L12	V_{SS}	
L13	V_{SS}	
L14	V_{SS}	
L19	V_{DDI}	
L20	LCD_LCLK	
L21	LCD_PCLK	
L22	V_{SS}	
L23	LCD_BIAS	
M1	RAD14	
M2	RAD9	
M3	RAD13	
M4	EWAIT#	
M10	V_{SS}	
M11	V_{SS}	
M12	V_{SS}	
M13	V_{SS}	
M14	V_{SS}	
M20	not connected	
M21	GPIO[200]	LCD_D0
M22	GPIO[201]	LCD_D1
M23	LCD_D2	
N1	RAD6	
N2	V_{SS}	
N3	RAD8	
N4	RAD5	
N5	V_{DDI}	
N10	V_{SS}	
N11	V_{SS}	
N12	V_{SS}	
N13	V_{SS}	
N14	V_{SS}	
N19	V_{DDI}	
N20	LCD_D5	
N21	LCD_D4	
N22	V_{SS}	
N23	LCD_D3	
P1	RAD4	
P2	RAD3	
P3	RAD7	
P4	V_{DDX}	
P5	V_{DDI}	
P10	V_{SS}	
P11	V_{SS}	
P12	V_{SS}	
P13	V_{SS}	
P14	V_{SS}	
P19	V_{DDI}	
P20	V_{DDX}	
P21	GPIO[210]	LCD_D8
P22	LCD_D7	
P23	LCD_D6	
R1	RAD1	
R2	V_{SS}	
R3	RAD0	
R4	RD15	
R20	LCD_D12	
R21	LCD_D10	
R22	V_{SS}	
R23	LCD_D9	
T1	RAD2	
T2	RD7	
T3	RD6	
T4	RD13	
T5	V_{DDX}	
T19	V_{DDX}	
T20	GPIO[211]	LCD_D16
T21	LCD_D14	
T22	LCD_D13	
T23	LCD_D11	
U1	RD14	
U2	V_{SS}	

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-3 "Pin Assignments - Sorted by Pin Number" (Continued)

Pin Number	Default Signal	Alternate Signal
U3	RD5	
U4	RD11	
U5	V_{DDX}	
U19	V_{DDX}	
U20	V_{DDX}	
U21	LCD_D18	
U22	V_{SS}	
U23	LCD_D15	
V1	RD12	
V2	RD4	
V3	RD2	
V4	RD1	
V5	V_{DDX}	
V19	V_{DDX}	
V20	RESETOUT#	
V21	LCD_D21	
V22	LCD_D19	
V23	LCD_D17	
W1	RD3	
W2	V_{SS}	
W3	RD9	
W4	DDQ15	
W5	V_{DDY}	
W6	V_{DDY}	
W7	V_{DDY}	
W8	V_{DDY}	
W10	V_{DDI}	
W11	V_{DDI}	
W13	V_{DDI}	
W14	V_{DDI}	
W16	V_{DDY}	
W17	V_{DDY}	
W18	V_{DDY}	
W19	V_{DDY}	
W20	DA1	
W21	LCD_D23	
W22	V_{SS}	
W23	LCD_D20	
Y1	RD10	
Y2	RD8	
Y3	DDQ14	
Y4	V_{DDY}	
Y5	DDQ30	
Y6	DDQ29	
Y7	DDQ20	
Y8	DDQ21	
Y9	DDQ6	
Y10	V_{DDY}	
Y11	DDQ10	
Y12	DDQS1	
Y13	DDM1	
Y14	V_{DDY}	
Y15	DCAS#	
Y16	DCS1#	
Y17	DRAS#	
Y18	DBA0	
Y19	DA10	
Y20	V_{DDY}	
Y21	DA3	
Y22	DRVSEL	
Y23	LCD_D22	
AA1	RD0	
AA2	V_{SS}	
AA3	DDQ16	
AA4	DDQ1	
AA5	DDQ2	
AA6	V_{DDY}	
AA7	DDQ4	
AA8	DDQ22	
AA9	DDQ23	
AA10	DDQ27	
AA11	DDQ9	
AA12	V_{DDY}	
AA13	DVREF	
AA14	DDM0	
AA15	DA13	
AA16	DCKE	
AA17	DCS0#	
AA18	V_{DDY}	
AA19	DBA1	
AA20	DA0	
AA21	DA2	
AA22	V_{SS}	
AA23	V_{DDXOK}	
AB1	DDQ31	
AB2	DDQ0	
AB3	V_{SS}	
AB4	DDQ18	
AB5	V_{SS}	
AB6	DDQ12	
AB7	V_{SS}	
AB8	DDQ11	
AB9	V_{SS}	
AB10	DDQ25	
AB11	V_{SS}	
AB12	DDM2	
AB13	V_{SS}	
AB14	DWE#	
AB15	V_{SS}	
AB16	DCK1	
AB17	V_{SS}	
AB18	DA12	
AB19	V_{SS}	
AB20	DA8	
AB21	V_{SS}	
AB22	DA5	
AB23	DA4	
AC1	V_{SS}	
AC2	DDQ17	
AC3	DDQ13	
AC4	DDQ19	
AC5	DDQ3	
AC6	DDQ28	
AC7	DDQ5	
AC8	DDQ26	
AC9	DDQ7	
AC10	DDQS2	
AC11	DDQ24	
AC12	DDQ8	
AC13	DDQS3	
AC14	DDQS0	

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-3 "Pin Assignments - Sorted by Pin Number" (Continued)

Pin Number	Default Signal	Alternate Signal
AC15	DDM3	
AC16	DCK1#	
AC17	DCK0#	
AC18	DCK0	
AC19	DA11	
AC20	DA9	
AC21	DA7	
AC22	DA6	
AC23	V_{SS}	

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-4. Pin Assignments - Sorted by Default Signal Name

Default Signal	Alternate Signal	Pin Number
BOOT[0]		A10
BOOT[1]		D13
DA0		AA20
DA1		W20
DA2		AA21
DA3		Y21
DA4		AB23
DA5		AB22
DA6		AC22
DA7		AC21
DA8		AB20
DA9		AC20
DA10		Y19
DA11		AC19
DA12		AB18
DA13		AA15
DBA0		Y18
DBA1		AA19
DCAS#		Y15
DCK0		AC18
DCK0#		AC17
DCK1		AB16
DCK1#		AC16
DCKE		AA16
DCS0#		AA17
DCS1#		Y16
DDM0		AA14
DDM1		Y13
DDM2		AB12
DDM3		AC15
DDQ0		AB2
DDQ1		AA4
DDQ2		AA5
DDQ3		AC5
DDQ4		AA7
DDQ5		AC7
DDQ6		Y9
DDQ7		AC9
DDQ8		AC12
DDQ9		AA11
DDQ10		Y11
DDQ11		AB8
DDQ12		AB6
DDQ13		AC3
DDQ14		Y3
DDQ15		W4
DDQ16		AA3
DDQ17		AC2
DDQ18		AB4
DDQ19		AC4
DDQ20		Y7
DDQ21		Y8
DDQ22		AA8
DDQ23		AA9
DDQ24		AC11
DDQ25		AB10
DDQ26		AC8
DDQ27		AA10
DDQ28		AC6
DDQ29		Y6
DDQ30		Y5
DDQ31		AB1
DDQS0		AC14
DDQS1		Y12
DDQS2		AC10
DDQS3		AC13
DRAS#		Y17
DRVSEL		Y22
DVREF		AA13
DWE#		AB14
EWAIT#		M4
FTM		C11
FWTOY#		A2
GPIO[0]	CIM_D0	B9
GPIO[1]	CIM_D1	C7
GPIO[2]	EXTCLK0	D8
GPIO[3]	EXTCLK1	C8
GPIO[4]	DMA_REQ0	D7
GPIO[5]		C10
GPIO[6]	SD0_DAT2	D6
GPIO[7]		A9
GPIO[8]	SD0_DAT1	A5
GPIO[9]	U1CTS#	A18
GPIO[10]	U1DSR#	C17
GPIO[11]	PSC1_D0	C19
GPIO[12]	DMA_REQ1	B19
GPIO[13]	U1RTS#	A20
GPIO[14]	U1DTR#	D18
GPIO[15]	U1TXD	A19
GPIO[17]	SD0_CLK	B1
GPIO[18]	PSC0_D0	B23
GPIO[19]	SD0_DAT0	E4
GPIO[20]	PSC1_SYNC0	C20
GPIO[22]	PSC1_D1	A21
GPIO[23]	PSC1_EXTCLK	B21
GPIO[24]	PSC1_CLK	A22
GPIO[25]	PSC0_CLK	C23
GPIO[26]	SD0_DAT3	D3
GPIO[27]	U0TXD	D17
GPIO[28]	SD0_CMD	B5
GPIO[29]	U0RXD	C15
GPIO[30]	U1RXD	A16
GPIO[31]	PSC0_D1	C21
GPIO[200]	LCD_D0	M21
GPIO[201]	LCD_D1	M22
GPIO[202]	CIM_D2	B15
GPIO[203]	CIM_D3	A15
GPIO[204]	CIM_D4	C14
GPIO[205]	CIM_D5	A14
GPIO[206]	CIM_D6	B13
GPIO[207]	CIM_D7	C13
GPIO[208]	CIM_D8	A13
GPIO[209]	CIM_D9	A12
GPIO[210]	LCD_D8	P21
GPIO[211]	LCD_D16	T20
GPIO[212]	CIM_LS	C12
GPIO[213]	CIM_FS	A11
GPIO[214]	CIM_CLK	B11
GPIO[215]	PSC0_SYNC0	D20
HD_CS0#		D2
HD_CS1#		C1
LCD_BIAS		L23
LCD_D2		M23
LCD_D3		N23
LCD_D4		N21
LCD_D5		N20
LCD_D6		P23
LCD_D7		P22
LCD_D9		R23
LCD_D10		R21
LCD_D11		T23
LCD_D12		R20

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-4 "Pin Assignments - Sorted by Default Signal Name" (Continued)

Default Signal	Alternate Signal	Pin Number
LCD_D13		T22
LCD_D14		T21
LCD_D15		U23
LCD_D17		V23
LCD_D18		U21
LCD_D19		V22
LCD_D20		W23
LCD_D21		V21
LCD_D22		Y23
LCD_D23		W21
LCD_FCLK		K23
LCD_LCLK		L20
LCD_PCLK		L21
PCE1#		E3
PCE2#		F4
PIOR#	SD1_CMD	G1
PIOS16#	SD1_DAT2	F3
PIOW#	SD1_CLK	F2
POE#	SD1_DAT1	F1
PREG#		D1
PSC0_SYNC1	GPIO[16]	C22
PSC1_SYNC1	GPIO[21]	C18
PWAIT#	SD1_DAT0	J3
PWE#	SD1_DAT3	H4
PWR_EN		A6
RAD0		R3
RAD1		R1
RAD2		T1
RAD3		P2
RAD4		P1
RAD5		N4
RAD6		N1
RAD7		P3
RAD8		N3
RAD9		M2
RAD10		L4
RAD11		K1
RAD12		L1
RAD13		M3
RAD14		M1
RALE		H1
RBE0#		J4
RBE1#		H2
RCLE		K3
RCLK		L3
RCS0#		H3
RCS1#		G4
RCS2#		G3
RCS3#		E1
RD0		AA1
RD1		V4
RD2		V3
RD3		W1
RD4		V2
RD5		U3
RD6		T3
RD7		T2
RD8		Y2
RD9		W3
RD10		Y1
RD11		U4
RD12		V1
RD13		T4
RD14		U1
RD15		R4
RESETIN#		B7
RESETOUT#		V20
RNB		D5
ROE#		K2
RWE#		J1
TC0		A7
TC1		A8
TC2		C9
TC3		D9
TCK		C16
TDI		B17
TDO		A17
TMS		D15
TRST#		D16
TESTEN		D11
U1DCD#	LCD_PWM0	K22
U1RI#	LCD_PWM1	K21
USBATEST		D22
USBDM		E23
USBDP		E22
USBHM		G23
USBHP		G22
USBOTGID		J22
USBREXT		H22
USBVBUS		H23
$USBV_{DDX}$		D21, E21, F21, G21, H19, H20, H21, J20, J21
$USBV_{DDI}$		D23, J23
$USBV_{SS}$		E19, E20, F19, F20, G19, G20
USBXI		F22
USBXO		F23
V_{DDI}		E10, E11, E13, E14, K5, K19, L5, L19, N5, N19, P5, P19, W10, W11, W13, W14
V_{DDX}		D4, D10, D14, D19, E5, E6, E7, E8, E16, E17, E18, F5, G5, H5, K4, K20, P4, P20, T5, T19, U5, U19, U20, V5, V19
V_{DDXOK}		AA23
V_{DDY}		W5, W6, W7, W8, W16, W17, W18, W19, Y4, Y10, Y14, Y20, AA6, AA12, AA18

www.DataSheet4U.com

www.DataSheet4U.com

Table 15-4 "Pin Assignments - Sorted by Default Signal Name" (Continued)

Default Signal	Alternate Signal	Pin Number
V_{SS}		A23, B6, B8, B10, B12, B14, B16, B18, B20, B22, C2, E2, G2, J2, K10, K11, K12, K13, K14, L2, L10, L11, L12, L13, L14, L22, M10, M11, M12, M13, M14, N2, N10, N11, N12, N13, N14, N22, P10, P11, P12, P13, P14, R2, R22, U2, U22, W2, W22, AA2, AA22, AB3, AB5, AB7, AB9, AB11, AB13, AB15, AB17, AB19, AB21, AC1, AC23
WAKE#		B2
XAGND12		C6
XAGND32		C4
XPWR12		C5
XPWR32		C3
XTI12		A4
XTI32		A3
XTO12		B4
XTO32		B3

www.DataSheet4U.com

www.DataSheet4U.com

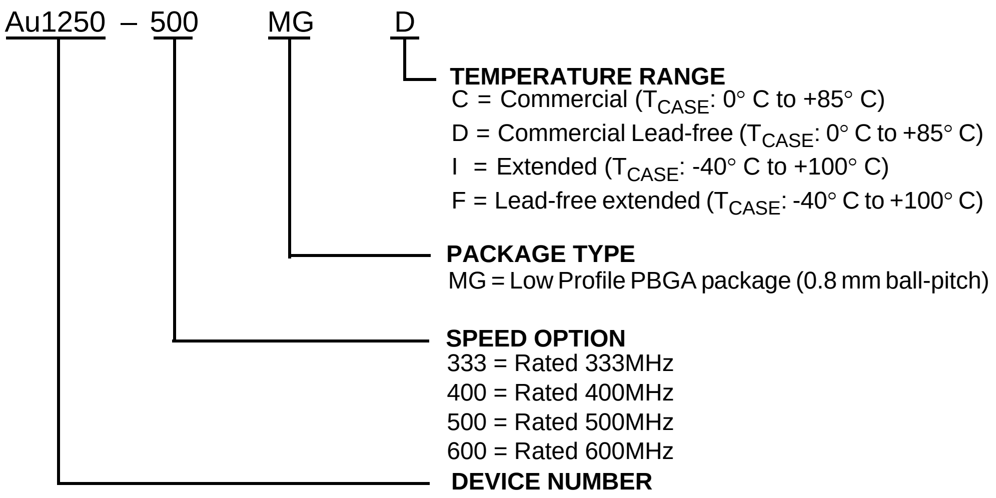
Table 15-5. Pin Assignments Sorted by Alternate Signal Name

Alternate Signal	Default Signal	Pin Number
CIM_CLK	GPIO[214]	B11
CIM_D0	GPIO[0]	B9
CIM_D1	GPIO[1]	C7
CIM_D2	GPIO[202]	B15
CIM_D3	GPIO[203]	A15
CIM_D4	GPIO[204]	C14
CIM_D5	GPIO[205]	A14
CIM_D6	GPIO[206]	B13
CIM_D7	GPIO[207]	C13
CIM_D8	GPIO[208]	A13
CIM_D9	GPIO[209]	A12
CIM_FS	GPIO[213]	A11
CIM_LS	GPIO[212]	C12
DMA_REQ0	GPIO[4]	D7
DMA_REQ1	GPIO[12]	B19
EXTCLK0	GPIO[2]	D8
EXTCLK1	GPIO[3]	C8
GPIO[16]	PSC0_SYNC1	C22
GPIO[21]	PSC1_SYNC1	C18
LCD_D0	GPIO[200]	M21
LCD_D1	GPIO[201]	M22
LCD_D8	GPIO[210]	P21
LCD_D16	GPIO[211]	T20
LCD_PWM0	U1DCD#	K22
LCD_PWM1	U1RI#	K21
PSC0_CLK	GPIO[25]	C23
PSC0_D0	GPIO[18]	B23
PSC0_D1	GPIO[31]	C21
PSC0_SYNC0	GPIO[215]	D20
PSC1_CLK	GPIO[24]	A22
PSC1_D0	GPIO[11]	C19
PSC1_D1	GPIO[22]	A21
PSC1_EXTCLK	GPIO[23]	B21
PSC1_SYNC0	GPIO[20]	C20
SD0_CLK	GPIO[17]	B1
SD0_CMD	GPIO[28]	B5
SD0_DAT0	GPIO[19]	E4
SD0_DAT1	GPIO[8]	A5
SD0_DAT2	GPIO[6]	D6
SD0_DAT3	GPIO[26]	D3
SD1_CLK	PIOW#	F2
SD1_CMD	PIOR#	G1
SD1_DAT0	PWAIT#	J3
SD1_DAT1	POE#	F1
SD1_DAT2	PIOS16#	F3
SD1_DAT3	PWE#	H4
U0RXD	GPIO[29]	C15
U0TXD	GPIO[27]	D17
U1CTS#	GPIO[9]	A18
U1DSR#	GPIO[10]	C17
U1DTR#	GPIO[14]	D18
U1RTS#	GPIO[13]	A20
U1RXD	GPIO[30]	A16
U1TXD	GPIO[15]	A19

www.DataSheet4U.com

www.DataSheet4U.com

15.3 Ordering Information



Au1250 – 500 MG D

TEMPERATURE RANGE
C = Commercial (T_{CASE}: 0° C to +85° C)
D = Commercial Lead-free (T_{CASE}: 0° C to +85° C)
I = Extended (T_{CASE}: -40° C to +100° C)
F = Lead-free extended (T_{CASE}: -40° C to +100° C)

PACKAGE TYPE
MG = Low Profile PBGA package (0.8 mm ball-pitch)

SPEED OPTION
333 = Rated 333MHz
400 = Rated 400MHz
500 = Rated 500MHz
600 = Rated 600MHz

DEVICE NUMBER

Valid Combinations

Au1210-333	MG	D, F
Au1210-400	MG	D, F(Note1)
Au1250-400	MG	D, F(Note1)
Au1250-500	MG	D, F(Note1)
Au1250-600	MG	D

Valid Combinations

Valid Combinations lists configurations planned to be supported in volume for this device. Consult the local RMI sales office to confirm availability of specific valid combinations and to check on newly released combinations.

Note1. Extended temperature operation at 400MHz and 500MHz is supported with DDR2 memory systems only.

www.DataSheet4U.com

www.DataSheet4U.com

Support Documentation

A.1 Memory Map

The peripheral devices on the Au1210™ and Au1250™ processors contain memory-mapped registers visible to software. Table A-1 contains the memory map for the peripheral devices and physical memory. The addresses are 36 bits wide.

Table A-1. Basic Au1210™ and Au1250™ Processors' 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: 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: 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 * 12	Reserved
0xD 00000000	0xD FFFFFFFF	4096	I/O Device
0xE 00000000	0xE FFFFFFFF	4096	Reserved
0xF 00000000	0xF FFFFFFFF	4096	PCMCIA Interface

The processors' system bus devices are mapped at the addresses based at 0x0 14000000. See Table A-2 for complete addresses.

Table A-2. System Bus Devices Physical Memory Map

Start Address	End Address	Size	Function
0x0 14000000	0x0 14000FFF	4 KB	DDR SDRAM Memory Controller
0x0 14001000	0x0 14001FFF	4 KB	SRAM/Flash/NAND/IDE Static Memory Controller
0x0 14002000	0x0 14002FFF	4 KB	DDMA Controller
0x0 14004000	0x0 14004FFF	4 KB	Camera Interface Module
0x0 14010000	0x0 14011FFF	8 KB	MAE Back End
0x0 14012000	0x0 14013FFF	8 KB	MAE Front End
0x0 14020000	0x0 14022FFF	12 KB	USB 2.0
0x0 15000000	0x0 150008FF	9 * 256 Bytes	LCD Controller

www.DataSheet4U.com

www.DataSheet4U.com

The processors' peripheral bus devices are based at 0x0 11000000. The individual memory spaces of the devices are defined in Table A-3.

Table A-3. Peripheral Bus Devices Physical Memory Map

Start Address	End Address	Size	Function
0x0 10000000	0x0 102FFFFF	3 MB	Reserved
0x0 10300000	0x0 103FFFFF	1 MB	AES Cryptography Engine (Au1250 only)
0x0 10400000	0x0 104FFFFF	1 MB	Interrupt Controller 0
0x0 10500000	0x0 105FFFFF	1 MB	Reserved
0x0 10600000	0x0 106FFFFF	1 MB	Secure Digital Controllers
0x0 10700000	0x0 10BFFFFF	5 MB	Reserved
0x0 11100000	0x0 1110010B	268 Bytes	UART0
0x0 1110010C	0x0 1110011B	16 Bytes	Software Counter
0x0 1110011C	0x0 111FFFFF	—	Reserved
0x0 11200000	0x0 112FFFFF	1 MB	UART1
0x0 11300000	0x0 114FFFFF	2 MB	Reserved
0x0 11700000	0x0 117FFFFF	1 MB	Secondary GPIO
0x0 11800000	0x0 118FFFFF	1 MB	Interrupt Controller 1
0x0 11900000	0x0 119FFFFF	1 MB	System Control: RTC, TOY, Timers, Primary GPIO, Power Management
0x0 11A00000	0x0 11AFFFFF	1 MB	PSC0
0x0 11B00000	0x0 11BFFFFF	1 MB	PSC1

Note: 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 2.3 "Caches" on page 22 for more information.

Table A-4 lists all of the devices that are memory mapped to the Au1210 and Au1250 processors' core. These devices are all mapped within kseg1 (non-cached, non-TLB). All 32-bit addresses are translated into 36-bit addresses by changing bits [31:29] to zero and adding bits [35:32] which are set to zero.

Table A-4. Device Memory Map

Register	KSEG1 Address	Physical Address
AES Cryptography Engine - Section 9.3 on page 325 (Au1250 only)		
aes_status	0xB0300000	0x0 10300000
aes_indata	0xB0300004	0x0 10300004
aes_outdata	0xB0300008	0x0 10300008
aes_intcause	0xB030000C	0x0 1030000C
aes_config	0xB0300010	0x0 10300010
Interrupt Controller 0 - Section 5.0 on page 143		
ic0_cfg0rd	0xB0400040	0x0 10400040
ic0_cfg0set	0xB0400040	0x0 10400040
ic0_cfg0clr	0xB0400044	0x0 10400044
ic0_cfg1rd	0xB0400048	0x0 10400048
ic0_cfg1set	0xB0400048	0x0 10400048

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
ic0_cfg1clr	0xB040004C	0x0 1040004C
ic0_cfg2rd	0xB0400050	0x0 10400050
ic0_cfg2set	0xB0400050	0x0 10400050
ic0_cfg2clr	0xB0400054	0x0 10400054
ic0_req0int	0xB0400054	0x0 10400054
ic0_srcrd	0xB0400058	0x0 10400058
ic0_srcset	0xB0400058	0x0 10400058
ic0_srcclr	0xB040005C	0x0 1040005C
ic0_req1int	0xB040005C	0x0 1040005C
ic0_assignrd	0xB0400060	0x0 10400060
ic0_assignset	0xB0400060	0x0 10400060
ic0_assignclr	0xB0400064	0x0 10400064
ic0_wakerd	0xB0400068	0x0 10400068
ic0_wakeset	0xB0400068	0x0 10400068
ic0_wakeclr	0xB040006C	0x0 1040006C
ic0_maskrd	0xB0400070	0x0 10400070
ic0_maskset	0xB0400070	0x0 10400070
ic0_maskclr	0xB0400074	0x0 10400074
ic0_risingrd	0xB0400078	0x0 10400078
ic0_risingclr	0xB0400078	0x0 10400078
ic0_fallingrd	0xB040007C	0x0 1040007C
ic0_fallingclr	0xB040007C	0x0 1040007C
ic0_testbit	0xB0400080	0x0 10400080
SD Controller 0 - Section 9.4 on page 333		
sd0_txport	0xB0600000	0x0 10600000
sd0_rxport	0xB0600004	0x0 10600004
sd0_config	0xB0600008	0x0 10600008
sd0_enable	0xB060000C	0x0 1060000C
sd0_config2	0xB0600010	0x0 10600010
sd0_blksize	0xB0600014	0x0 10600014
sd0_status	0xB0600018	0x0 10600018
sd0_debug	0xB060001C	0x0 1060001C
sd0_cmd	0xB0600020	0x0 10600020
sd0_cmdarg	0xB0600024	0x0 10600024
sd0_resp3	0xB0600028	0x0 10600028
sd0_resp2	0xB060002C	0x0 1060002C
sd0_resp1	0xB0600030	0x0 10600030
sd0_resp0	0xB0600034	0x0 10600034
sd0_timeout	0xB0600038	0x0 10600038

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
SD Controller 1 - Section 9.4 on page 333		
sd1_txport	0xB0680000	0x0 10680000
sd1_rxport	0xB0680004	0x0 10680004
sd1_config	0xB0680008	0x0 10680008
sd1_enable	0xB068000C	0x0 1068000C
sd1_config2	0xB0680010	0x0 10680010
sd1_blksize	0xB0680014	0x0 10680014
sd1_status	0xB0680018	0x0 10680018
sd1_debug	0xB068001C	0x0 1068001C
sd1_cmd	0xB0680020	0x0 10680020
sd1_cmdarg	0xB0680024	0x0 10680024
sd1_resp3	0xB0680028	0x0 10680028
sd1_resp2	0xB068002C	0x0 1068002C
sd1_resp1	0xB0680030	0x0 10680030
sd1_resp0	0xB0680034	0x0 10680034
sd1_timeout	0xB0680038	0x0 10680038
UART0 - Section 9.5 on page 345		
uart0_rxdata	0xB1100000	0x0 11100000
uart0_txdata	0xB1100004	0x0 11100004
uart0_inten	0xB1100008	0x0 11100008
uart0_intcause	0xB110000C	0x0 1110000C
uart0_fifoctrl	0xB1100010	0x0 11100010
uart0_linectrl	0xB1100014	0x0 11100014
uart0_mdmctrl	0xB1100018	0x0 11100018
uart0_linestat	0xB110001C	0x0 1110001C
uart0_mdmstat	0xB1100020	0x0 11100020
uart0_autoflow	0xB1100024	0x0 11100024
uart0_clkdiv	0xB1100028	0x0 11100028
uart0_enable	0xB1100100	0x0 11100100
uart0_mdmden	0xB1100104	0x0 11100104
uart0_bidir	0xB1100108	0x0 11100108
Software Counter - Section 9.6 on page 359		
swcnt_control	0xB110010C	0x0 1110010C
swcnt_count	0xB1100120	0x0 11100120
swcnt_match	0xB1100124	0x0 11100124
swcnt_intclr	0xB1100128	0x0 11100128
UART1 - Section 9.5 on page 345		
uart1_rxdata	0xB1200000	0x0 11200000
uart1_txdata	0xB1200004	0x0 11200004

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
uart1_inten	0xB1200008	0x0 11200008
uart1_intcause	0xB120000C	0x0 1120000C
uart1_fifoctrl	0xB1200010	0x0 11200010
uart1_linectrl	0xB1200014	0x0 11200014
uart1_mdmctrl	0xB1200018	0x0 11200018
uart1_linestat	0xB120001C	0x0 1120001C
uart1_mdmstat	0xB1200020	0x0 11200020
uart1_autoflow	0xB1200024	0x0 11200024
uart1_clkdiv	0xB1200028	0x0 11200028
uart1_enable	0xB1200100	0x0 11200100
uart1_mdmden	0xB1200104	0x0 11200104
uart1_bidir	0xB1200108	0x0 11200108
Secondary GPIO - Section 9.8 on page 362		
gpio2_dir	0xB1700000	0x0 11700000
—	0xB1700004	0x0 11700004
gpio2_output	0xB1700008	0x0 11700008
gpio2_pinstate	0xB170000C	0x0 1170000C
gpio2_inten	0xB1700010	0x0 11700010
gpio2_enable	0xB1700014	0x0 11700014
Interrupt Controller 1 - Section 5.0 on page 143		
ic1_cfg0rd	0xB1800040	0x0 11800040
ic1_cfg0set	0xB1800040	0x0 11800040
ic1_cfg0clr	0xB1800044	0x0 11800044
ic1_cfg1rd	0xB1800048	0x0 11800048
ic1_cfg1set	0xB1800048	0x0 11800048
ic1_cfg1clr	0xB180004C	0x0 1180004C
ic1_cfg2rd	0xB1800050	0x0 11800050
ic1_cfg2set	0xB1800050	0x0 11800050
ic1_cfg2clr	0xB1800054	0x0 11800054
ic1_req0int	0xB1800054	0x0 11800054
ic1_srcrd	0xB1800058	0x0 11800058
ic1_srcset	0xB1800058	0x0 11800058
ic1_srcclr	0xB180005C	0x0 1180005C
ic1_req1int	0xB180005C	0x0 1180005C
ic1_assignrd	0xB1800060	0x0 11800060
ic1_assignset	0xB1800060	0x0 11800060
ic1_assignclr	0xB1800064	0x0 11800064
ic1_wakerd	0xB1800068	0x0 11800068
ic1_wakeset	0xB1800068	0x0 11800068

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
ic1_wakeclr	0xB180006C	0x0 1180006C
ic1_maskrd	0xB1800070	0x0 11800070
ic1_maskset	0xB1800070	0x0 11800070
ic1_maskclr	0xB1800074	0x0 11800074
ic1_risingrd	0xB1800078	0x0 11800078
ic1_risingclr	0xB1800078	0x0 11800078
ic1_fallingrd	0xB180007C	0x0 1180007C
ic1_fallingclr	0xB180007C	0x0 1180007C
ic1_testbit	0xB1800080	0x0 11800080
Clock Generator and PLL Control - Section 10.1 on page 366		
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 10.2 on page 375		
sys_toytrim	0xB1900000	0x0 11900000
sys_toywrite	0xB1900004	0x0 11900004
sys_toymatch0	0xB1900008	0x0 11900008
sys_toymatch1	0xB190000C	0x0 1190000C
sys_toymatch2	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 10.3 on page 381		
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
sys_outputclr	0xB190010C	0x0 1190010C
sys_pinstaterd	0xB1900110	0x0 11900110
sys_pininputen	0xB1900110	0x0 11900110

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
Power Management - Section 10.4 on page 387		
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
PSC0 - Section 7.1 on page 194		
psc0_sel	0xB1A00000	0x0 11A00000
psc0_ctrl	0xB1A00004	0x0 11A00004
psc0_spicfg, psc0_i2scfg, psc0_ac97cfg, psc0_smbcfg	0xB1A00008	0x0 11A00008
psc0_spimsk, psc0_i2smsk, psc0_ac97msk, psc0_smbmsk	0xB1A0000C	0x0 11A0000C
psc0_spipcr, psc0_i2spcr, psc0_ac97pcr, psc0_smbpcr	0xB1A00010	0x0 11A00010
psc0_spistat, psc0_i2sstat, psc0_ac97stat, psc0_smbstat	0xB1A00014	0x0 11A00014
psc0_spievnt, psc0_i2sevnt, psc0_ac97evnt, psc0_smbevnt	0xB1A00018	0x0 11A00018
psc0_spitxrx, psc0_i2stxrx, psc0_ac97txrx, psc0_smbtxrx	0xB1A0001C	0x0 11A0001C
psc0_i2sudf, psc0_ac97cdc, psc0_smbtmr	0xB1A00020	0x0 11A00020
psc0_ac97rst	0xB1A00024	0x0 11A00024
psc0_ac97gpo	0xB1A00028	0x0 11A00028
psc0_ac97gpi	0xB1A0002C	0x0 11A0002C
PSC1 - Section 7.1 on page 194		
psc1_sel	0xB1B00000	0x0 11B00000
psc1_ctrl	0xB1B00004	0x0 11B00004
psc1_spicfg, psc1_i2scfg, psc1_ac97cfg, psc1_smbcfg	0xB1B00008	0x0 11B00008
psc1_spimsk, psc1_i2smsk, psc1_ac97msk, psc1_smbmsk	0xB1B0000C	0x0 11B0000C
psc1_spipcr, psc1_i2spcr, psc1_ac97pcr, psc1_smbpcr	0xB1B00010	0x0 11B00010
psc1_spistat, psc1_i2sstat, psc1_ac97stat, psc1_smbstat	0xB1B00014	0x0 11B00014
psc1_spievnt, psc1_i2sevnt, psc1_ac97evnt, psc1_smbevnt	0xB1B00018	0x0 11B00018

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
psc1_spitxrx, psc1_i2stxrx, psc1_ac97txrx, psc1_smbtxrx	0xB1B0001C	0x0 11B0001C
psc1_i2sudf, psc1_ac97cdc, psc1_smbtmr	0xB1B00020	0x0 11B00020
psc1_ac97rst	0xB1B00024	0x0 11B00024
psc1_ac97gpo	0xB1B00028	0x0 11B00028
psc1_ac97gpi	0xB1B0002C	0x0 11B0002C
DDR SDRAM Controller - Section 3.1 on page 58		
mem_sdmode0	0xB4000800	0x0 14000800
mem_sdmode1	0xB4000808	0x0 14000808
—		
mem_sdaddr0	0xB4000820	0x0 14000820
mem_sdaddr1	0xB4000828	0x0 14000828
—		
mem_sdconfiga	0xB4000840	0x0 14000840
mem_sdconfigb	0xB4000848	0x0 14000848
mem_sdstat	0xB4000850	0x0 14000850
—		
mem_sdwrmd0	0xB4000880	0x0 14000880
mem_sdwrmd1	0xB4000888	0x0 14000888
—		
mem_sdprecmd	0xB40008C0	0x0 140008C0
mem_sdautoref	0xB40008C8	0x0 140008C8
mem_sdsref	0xB40008D0	0x0 140008D0
Static Bus Controller - Section 3.2 on page 77		
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
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
mem_staltime	0xB4001040	0x0 14001040
mem_stndctrl	0xB4001100	0x0 14001100
mem_ststat	0xB4001104	0x0 14001104

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
DMA Channel 0 - Section 4.2.2 on page 122		
ddma0_cfg	0xB4002000	0x0 14002000
ddma0_desptr	0xB4002004	0x0 14002004
ddma0_statptr	0xB4002008	0x0 14002008
ddma0_dbell	0xB400200C	0x0 1400200C
ddma0_irq	0xB4002010	0x0 14002010
ddma0_stat	0xB4002014	0x0 14002014
ddma0_bytecnt	0xB4002018	0x0 14002018
DMA Channel 1 - Section 4.2.2 on page 122		
ddma1_cfg	0xB4002100	0x0 14002100
ddma1_desptr	0xB4002104	0x0 14002104
ddma1_statptr	0xB4002108	0x0 14002108
ddma1_dbell	0xB400210C	0x0 1400210C
ddma1_irq	0xB4002110	0x0 14002110
ddma1_stat	0xB4002114	0x0 14002114
ddma1_bytecnt	0xB4002118	0x0 14002118
DMA Channel 2 - Section 4.2.2 on page 122		
ddma2_cfg	0xB4002200	0x0 14002200
ddma2_desptr	0xB4002204	0x0 14002204
ddma2_statptr	0xB4002208	0x0 14002208
ddma2_dbell	0xB400220C	0x0 1400220C
ddma2_irq	0xB4002210	0x0 14002210
ddma2_stat	0xB4002214	0x0 14002214
ddma2_bytecnt	0xB4002218	0x0 14002218
DMA Channel 3 - Section 4.2.2 on page 122		
ddma3_cfg	0xB4002300	0x0 14002300
ddma3_desptr	0xB4002304	0x0 14002304
ddma3_statptr	0xB4002308	0x0 14002308
ddma3_dbell	0xB400230C	0x0 1400230C
ddma3_irq	0xB4002310	0x0 14002310
ddma3_stat	0xB4002314	0x0 14002314
ddma3_bytecnt	0xB4002318	0x0 14002318
DMA Channel 4 - Section 4.2.2 on page 122		
ddma4_cfg	0xB4002400	0x0 14002400
ddma4_desptr	0xB4002404	0x0 14002404
ddma4_statptr	0xB4002408	0x0 14002408
ddma4_dbell	0xB400240C	0x0 1400240C
ddma4_irq	0xB4002410	0x0 14002410
ddma4_stat	0xB4002414	0x0 14002414

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
ddma4_bytecnt	0xB4002418	0x0 14002418
DMA Channel 5 - Section 4.2.2 on page 122		
ddma5_cfg	0xB4002500	0x0 14002500
ddma5_desptr	0xB4002504	0x0 14002504
ddma5_statptr	0xB4002508	0x0 14002508
ddma5_dbell	0xB400250C	0x0 1400250C
ddma5_irq	0xB4002510	0x0 14002510
ddma5_stat	0xB4002514	0x0 14002514
ddma5_bytecnt	0xB4002518	0x0 14002518
DMA Channel 6 - Section 4.2.2 on page 122		
ddma6_cfg	0xB4002600	0x0 14002600
ddma6_desptr	0xB4002604	0x0 14002604
ddma6_statptr	0xB4002608	0x0 14002608
ddma6_dbell	0xB400260C	0x0 1400260C
ddma6_irq	0xB4002610	0x0 14002610
ddma6_stat	0xB4002614	0x0 14002614
ddma6_bytecnt	0xB4002618	0x0 14002618
DMA Channel 7 - Section 4.2.2 on page 122		
ddma7_cfg	0xB4002700	0x0 14002700
ddma7_desptr	0xB4002704	0x0 14002704
ddma7_statptr	0xB4002708	0x0 14002708
ddma7_dbell	0xB400270C	0x0 1400270C
ddma7_irq	0xB4002710	0x0 14002710
ddma7_stat	0xB4002714	0x0 14002714
ddma7_bytecnt	0xB4002718	0x0 14002718
DMA Channel 8 - Section 4.2.2 on page 122		
ddma8_cfg	0xB4002800	0x0 14002800
ddma8_desptr	0xB4002804	0x0 14002804
ddma8_statptr	0xB4002808	0x0 14002808
ddma8_dbell	0xB400280C	0x0 1400280C
ddma8_irq	0xB4002810	0x0 14002810
ddma8_stat	0xB4002814	0x0 14002814
ddma8_bytecnt	0xB4002818	0x0 14002818
DMA Channel 9 - Section 4.2.2 on page 122		
ddma9_cfg	0xB4002900	0x0 14002900
ddma9_desptr	0xB4002904	0x0 14002904
ddma9_statptr	0xB4002908	0x0 14002908
ddma9_dbell	0xB400290C	0x0 1400290C
ddma9_irq	0xB4002910	0x0 14002910

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
ddma9_stat	0xB4002914	0x0 14002914
ddma9_bytecnt	0xB4002918	0x0 14002918
DMA Channel 0xA - Section 4.2.2 on page 122		
ddmaa_cfg	0xB4002A00	0x0 14002A00
ddmaa_desptr	0xB4002A04	0x0 14002A04
ddmaa_statptr	0xB4002A08	0x0 14002A08
ddmaa_dbell	0xB4002A0C	0x0 14002A0C
ddmaa_irq	0xB4002A10	0x0 14002A10
ddmaa_stat	0xB4002A14	0x0 14002A14
ddmaa_bytecnt	0xB4002A18	0x0 14002A18
DMA Channel 0xB - Section 4.2.2 on page 122		
ddmab_cfg	0xB4002B00	0x0 14002B00
ddmab_desptr	0xB4002B04	0x0 14002B04
ddmab_statptr	0xB4002B08	0x0 14002B08
ddmab_dbell	0xB4002B0C	0x0 14002B0C
ddmab_irq	0xB4002B10	0x0 14002B10
ddmab_stat	0xB4002B14	0x0 14002B14
ddmab_bytecnt	0xB4002B18	0x0 14002B18
DMA Channel 0xC - Section 4.2.2 on page 122		
ddmac_cfg	0xB4002C00	0x0 14002C00
ddmac_desptr	0xB4002C04	0x0 14002C04
ddmac_statptr	0xB4002C08	0x0 14002C08
ddmac_dbell	0xB4002C0C	0x0 14002C0C
ddmac_irq	0xB4002C10	0x0 14002C10
ddmac_stat	0xB4002C14	0x0 14002C14
ddmac_bytecnt	0xB4002C18	0x0 14002C18
DMA Channel 0xD - Section 4.2.2 on page 122		
ddmad_cfg	0xB4002D00	0x0 14002D00
ddmad_desptr	0xB4002D04	0x0 14002D04
ddmad_statptr	0xB4002D08	0x0 14002D08
ddmad_dbell	0xB4002D0C	0x0 14002D0C
ddmad_irq	0xB4002D10	0x0 14002D10
ddmad_stat	0xB4002D14	0x0 14002D14
ddmad_bytecnt	0xB4002D18	0x0 14002D18
DMA Channel 0xE - Section 4.2.2 on page 122		
ddmae_cfg	0xB4002E00	0x0 14002E00
ddmae_desptr	0xB4002E04	0x0 14002E04
ddmae_statptr	0xB4002E08	0x0 14002E08
ddmae_dbell	0xB4002E0C	0x0 14002E0C

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
ddmae_irq	0xB4002E10	0x0 14002E10
ddmae_stat	0xB4002E14	0x0 14002E14
ddmae_bytecnt	0xB4002E18	0x0 14002E18
DMA Channel 0xF - Section 4.2.2 on page 122		
ddmaf_cfg	0xB4002F00	0x0 14002F00
ddmaf_desptr	0xB4002F04	0x0 14002F04
ddmaf_statptr	0xB4002F08	0x0 14002F08
ddmaf_dbell	0xB4002F0C	0x0 14002F0C
ddmaf_irq	0xB4002F10	0x0 14002F10
ddmaf_stat	0xB4002F14	0x0 14002F14
ddmaf_bytecnt	0xB4002F18	0x0 14002F18
DDMA Controller Global Registers - Section 4.2.1 on page 120		
ddma_config	0xB4003000	0x0 14003000
ddma_intstat	0xB4003004	0x0 14003004
ddma_throttle	0xB4003008	0x0 14003008
ddma_inten	0xB400300C	0x0 1400300C
CIM Registers		
cim_enable	0xB4004000	0x0 14004000
cim_config	0xB4004004	0x0 14004004
cim_capture	0xB4004010	0x0 14004010
cim_stat	0xB4004014	0x0 14004014
cim_inten	0xB4004018	0x0 14004018
cim_intstat	0xB400401C	0x0 1400401C
cim_fifoa	0xB4004020	0x0 14004020
cim_fifob	0xB4004040	0x0 14004040
cim_fifoc	0xB4004060	0x0 14004060
MAE Back End Registers - Section 6.3.5 on page 178		
maebe_scfhsr	0xB4010000	0x0 14010000
maebe_scfvsr	0xB4010004	0x0 14010004
maebe_scfdisable	0xB4010008	0x0 14010008
maebe_scfhalut	0xB4010100	0x0 14010100
maebe_scfvalut	0xB4010180	0x0 14010180
maebe_scfhblut	0xB4010200	0x0 14010200
maebe_scfvblut	0xB4010280	0x0 14010280
maebe_scfhclut	0xB4010300	0x0 14010300
maebe_scfvclut	0xB4010380	0x0 14010380
maebe_cscxcffa	0xB4010400	0x0 14010400
maebe_cscxcffb	0xB4010404	0x0 14010404
maebe_cscxcffc	0xB4010408	0x0 14010408

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
maebe_cscycffa	0xB401040C	0x0 1401040C
maebe_cscycffb	0xB4010410	0x0 14010410
maebe_cscycffc	0xB4010414	0x0 14010414
maebe_csczcffa	0xB4010418	0x0 14010418
maebe_csczcffb	0xB401041C	0x0 1401041C
maebe_csczcffc	0xB4010420	0x0 14010420
maebe_cscxoff	0xB4010424	0x0 14010424
maebe_cscyoff	0xB4010428	0x0 14010428
maebe_csczoff	0xB401042C	0x0 1401042C
maebe_cscalpha	0xB4010430	0x0 14010430
maebe_srccfg	0xB4010500	0x0 14010500
maebe_srcfhw	0xB4010504	0x0 14010504
maebe_srcaaddr	0xB4010508	0x0 14010508
maebe_srcastr	0xB401050C	0x0 1401050C
maebe_srcbaddr	0xB4010510	0x0 14010510
maebe_srcbstr	0xB4010514	0x0 14010514
maebe_srccaddr	0xB4010518	0x0 14010518
maebe_srccstr	0xB401051C	0x0 1401051C
maebe_dstcfg	0xB4010600	0x0 14010600
maebe_dstheight	0xB4010604	0x0 14010604
maebe_dstaddr	0xB4010608	0x0 14010608
maebe_dststr	0xB401060C	0x0 1401060C
maebe_ctlenable	0xB4010700	0x0 14010700
maebe_ctlfpc	0xB4010704	0x0 14010704
maebe_ctlstat	0xB4010708	0x0 14010708
maebe_ctlintenable	0xB401070C	0x0 1401070C
maebe_ctlintstat	0xB4010710	0x0 14010710
maebe_dbgscfafifo	0xB4010800	0x0 14010800
maebe_dbgscfbfifo	0xB4010804	0x0 14010804
maebe_dbgscfcfifo	0xB4010808	0x0 14010808
maebe_dbgcscfifo	0xB401080C	0x0 1401080C
maebe_dbgscfaram	0xB4010900	0x0 14010900
maebe_dbgscfbram	0xB4010B00	0x0 14010B00
maebe_dbgscfcram	0xB4010D00	0x0 14010D00
MAE Front End Registers - Section 6.0 on page 149		
maefe_config	0xB4012000	0x0 14012000
maefe_cury	0xB4012004	0x0 14012004
maefe_frefy	0xB4012008	0x0 14012008
maefe_brefy	0xB401200C	0x0 1401200C

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
maefe_curcb	0xB4012010	0x0 14012010
maefe_frefcb	0xB4012014	0x0 14012014
maefe_brefcb	0xB4012018	0x0 14012018
maefe_curcr	0xB401201C	0x0 1401201C
maefe_frefcr	0xB4012020	0x0 14012020
maefe_brefcr	0xB4012024	0x0 14012024
maefe_pictsize	0xB4012028	0x0 14012028
maefe_intenscomp	0xB401202C	0x0 1401202C
—		
maefe_frefboty	0xB4012038	0x0 14012038
maefe_frefbotcb	0xB401203C	0x0 1401203C
maefe_frefbotcr	0xB4012040	0x0 14012040
maefe_brefboty	0xB4012044	0x0 14012044
maefe_brefbotcb	0xB4012048	0x0 14012048
maefe_brefbotcr	0xB401204C	0x0 1401204C
maefe_intstat	0xB4012050	0x0 14012050
maefe_intenable	0xB4012054	0x0 14012054
maefe_scratchpad	0xB4012058	0x0 14012058
maefe_wmv9pquant	0xB401205C	0x0 1401205C
maefe_dmadscr	0xB4013004	0x0 14013004
maefe_dmadbell	0xB4013008	0x0 14013008
USB 2.0 Memory Map - Section 8.0 on page 251		
usb_base	0xB4020000	0x0 14020000
usb_otgbase	0xB4020020	0x0 14020020
usb_ohcibase	0xB4020100	0x0 14020100
usb_ehcibase	0xB4020200	0x0 14020200
usb_devbase	0xB4022000	0x0 14022000
LCD Controller - Section 9.2 on page 300		
—		
lcd_screen	0xB5000004	0x0 15000004
lcd_backcolor	0xB5000008	0x0 15000008
lcd_horztiming	0xB500000C	0x0 1500000C
lcd_verttiming	0xB5000010	0x0 15000010
lcd_clockctrl	0xB5000014	0x0 15000014
lcd_pwmdiv	0xB5000018	0x0 15000018
lcd_pwmhi	0xB500001C	0x0 1500001C
lcd_winenable	0xB5000024	0x0 15000024
lcd_colorkey	0xB5000028	0x0 15000028
lcd_colorkeymsk	0xB500002C	0x0 1500002C

www.DataSheet4U.com

www.DataSheet4U.com

Table A-4. Device Memory Map (Continued)

Register	KSEG1 Address	Physical Address
lcd_cursorctrl	0xB5000030	0x0 15000030
lcd_cursorpos	0xB5000034	0x0 15000034
lcd_cursorcolor0	0xB5000038	0x0 15000038
lcd_cursorcolor1	0xB500003C	0x0 1500003C
lcd_cursorcolor2	0xB5000040	0x0 15000040
lcd_cursorcolor3	0xB5000044	0x0 15000044
lcd_intstatus	0xB5000048	0x0 15000048
lcd_intenable	0xB500004C	0x0 1500004C
lcd_outmask	0xB5000050	0x0 15000050
lcd_fifoctrl	0xB5000054	0x0 15000054
lcd_win0ctrl0	0xB5000100	0x0 15000100
lcd_win0ctrl1	0xB5000104	0x0 15000104
lcd_win0ctrl2	0xB5000108	0x0 15000108
lcd_win0buf0	0xB500010C	0x0 1500010C
lcd_win0buf1	0xB5000110	0x0 15000110
lcd_win0bufctrl	0xB5000114	0x0 15000114
lcd_win1ctrl0	0xB5000120	0x0 15000120
lcd_win1ctrl1	0xB5000124	0x0 15000124
lcd_win1ctrl2	0xB5000128	0x0 15000128
lcd_win1buf0	0xB500012C	0x0 1500012C
lcd_win1buf1	0xB5000130	0x0 15000130
lcd_win1bufctrl	0xB5000134	0x0 15000134
lcd_win2ctrl0	0xB5000140	0x0 15000140
lcd_win2ctrl1	0xB5000144	0x0 15000144
lcd_win2ctrl2	0xB5000148	0x0 15000148
lcd_win2buf0	0xB500014C	0x0 1500014C
lcd_win2buf1	0xB5000150	0x0 15000150
lcd_win2bufctrl	0xB5000154	0x0 15000154
lcd_win3ctrl0	0xB5000160	0x0 15000160
lcd_win3ctrl1	0xB5000164	0x0 15000164
lcd_win3ctrl2	0xB5000168	0x0 15000168
lcd_win3buf0	0xB500016C	0x0 1500016C
lcd_win3buf1	0xB5000170	0x0 15000170
lcd_win3bufctrl	0xB5000174	0x0 15000174
lcd_palette	0xB5000400	0x0 15000400
lcd_cursorpattern	0xB5000800	0x0 15000800

www.DataSheet4U.com

www.DataSheet4U.com

A.2 Data Book Notations

This section addresses terminology used in this book.

This data book uses the conventions in Table 15-6.

Table 15-6. Notation Conventions

Convention	Description
Clear	To clear a bit, write a zero to it unless otherwise noted.
Set	To set a bit, write a one to it unless otherwise noted.
WAIT	Instruction mnemonics are shown in **UPPCASE BOLD**.
sys_pinfunc[CS]	Registers are shown in **lowercase bold**. Bit fields or ranges appear in brackets.
DDQ[31:0]	External signals are shown in uppercase. Ranges appear in brackets.
RCS[3:0]	Active-low signals are shown with overbars. Active-low signals are considered asserted (active) when low and negated when high.
sysbus_addr[35:0]	Internal signals are shown in *italics*. Ranges appear in brackets.
—	A long dash in a register definition means the field is reserved.
Reserved	Reserved bit ranges in a register definition must be cleared on writes and ignored on reads. Reserved areas of memory must not be accessed.
Rs	In register definitions, default values that depend on the system configuration at reset are designated with Rs.
MSB, LSB	Most significant byte, least significant byte
MSb, LSb	Most significant bit, least significant bit
0b	Binary numbers are denoted with a ‘0b’ prefix.
0x	Hexadecimal numbers are denoted with a ‘0x’ prefix.
MBS	Indicates a bit that must be set by the programmer.
MBC	Indicates a bit that must be cleared by the programmer.
n	Indicates a numeric variable. For example, **psc*n*_sel** represents each of the PSC*n* select registers: **psc0_sel** and **psc1_sel**.
x	x means *undefined* or *don’t-care*.
word, halfword	A *word* is defined as 32 bits; a *halfword* is 16 bits. For the USB chapter only, a *dword* is defined as 4 bytes.

www.DataSheet4U.com

www.DataSheet4U.com

A.2.1 Unpredictable and Undefined

The terms UNPREDICTABLE and UNDEFINED are used throughout this book to describe the behavior of the processor in certain cases. UNDEFINED behavior or operations can occur only as the result of executing instructions in a privileged mode (i.e., in Kernel Mode or Debug Mode, or with the CP0 usable bit set in the Status register). Unprivileged software can never cause UNDEFINED behavior or operations. Conversely, both privileged and unprivileged software can cause UNPREDICTABLE results or operations.

A.2.1.1 Unpredictable

UNPREDICTABLE results may vary from processor implementation to implementation, instruction to instruction, or as a function of time on the same implementation or instruction. Software can never depend on results that are UNPREDICTABLE. UNPREDICTABLE operations may cause a result to be generated or not. If a result is generated, it is UNPREDICTABLE. UNPREDICTABLE operations may cause arbitrary exceptions.

UNPREDICTABLE results or operations have several implementation restrictions:

- Implementations of operations generating UNPREDICTABLE results must not depend on any data source (memory or internal state) which is inaccessible in the current processor mode
- UNPREDICTABLE operations must not read, write, or modify the contents of memory or internal state which is inaccessible in the current processor mode. For example, UNPREDICTABLE operations executed in user mode must not access memory or internal state that is only accessible in Kernel Mode or Debug Mode or in another process
- UNPREDICTABLE operations must not halt or hang the processor

UNPRED used to describe the default state of registers should be taken as meaning UNPREDICATABLE.

A.2.1.2 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.

www.DataSheet4U.com

www.DataSheet4U.com

A.3 Data Book Revision History

This document is a report of the revision/creation process of the data book for the Alchemy Au1210 and Au1250 processors. Any revisions (i.e., additions, deletions, parameter corrections, etc.) are recorded in Table A-5

Table A-5. Revision History

Revision # (PDF Date)	Revisions / Comments
A (April 2007)	First release (Preliminary).

www.DataSheet4U.com

www.DataSheet4U.com

RMI®

Raza Microelectronics, Inc.

www.RazaMicro.com

www.DataSheet4U.com