USER MANUAL

Functional Manual

UT699 LEON 3FT /SPARC^{TM} V8 Microprocessor

Table of Contents

9/5/2018

Chapter 1: Introduction

1.1 Scope

This document describes the UT699 LEON 3FT microprocessor. The UT699 is a pipelined, monolithic, high-performance, fault-tolerant SPARCTM V8 Processor. It has been designed for reliable operation in HiRel environments; the architecture includes functionality to detect and correct (SEU) errors in all on-chip RAM memories.

The UT699 provides a 32-bit/33MHz PCI (Revision 2.1 compatible) master/target interface with DMA and host capabilities, including a 16-bit user I/O interface for off-chip peripherals. An AMBA (Rev. 2.0) bus interface integrates the on-chip LEON 3FT core, SpaceWire, Ethernet, memory controller, PCI, CAN bus, and programmable interrupt peripherals.

The UT699 is SPARC V8 compliant. Therefore, industry standard compilers and kernels for the SPARC V8 can be used for software development. A full software development suite, including a C/C++ cross-compiler system based on the Gnu C Compiler (GCC) and the Newlib embedded C-library is available from Cobham Gaisler. The Bare C Compiler (BCC), based upon GCC, includes a small run-time kernel with interrupt support and Pthreads library.

The development suite runs on either Windows or Linux operating systems. For multi-threaded applications, a SPARC compliant port of the eCos real-time kernel, RTEMS 4.6.5 and VxWorks 6.x is supported.

The UT699 LEON 3FT microprocessor is based on IP cores from Cobham Gaisler AB's GRLIB Intellectual Property (IP) library and uses a plug-and-play configuration.

1.2 Architecture

The UT699 consists of one LEON 3FT processor core, an 8/32-bit memory controller, a PCI controller, four SpaceWire links, two CAN-2.0 interfaces, one UART, four timers, one interrupt controller, a 16-bit I/O port, a serial/JTAG debug link and a 10/100 Ethernet MAC. The block diagram follows.

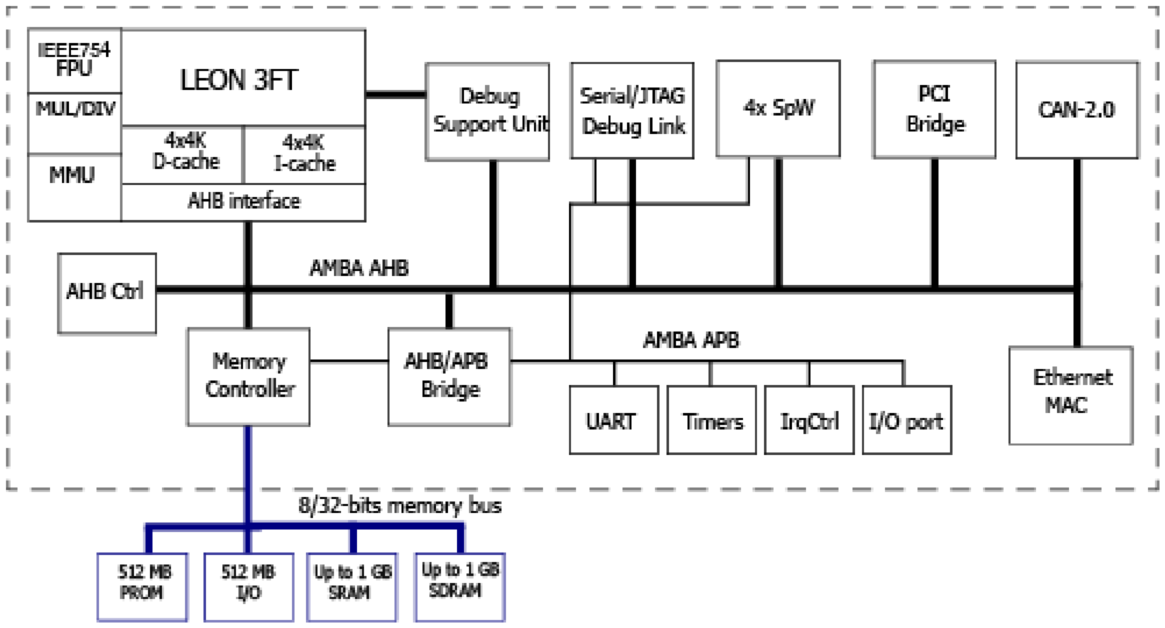


Figure 1.1: UT699 Functional Block Diagram

The design is based on the following IP cores from the GRLIB IP library:

Table 1.1: GLIB IP Cores used in UT699

CORE	FUNCTION	VENDOR ID	DEVICE ID	VER
LEON 3FT	SPARC V8 32-bit processor	0x01	0x053	0x0
DSU3	Debug support unit	0x01	0x004	0x1
IRQMP	Interrupt controller	0x01	0x00D	0x3
APBCTRL	AHB/APB bridge	0x01	0x006	0x0
FTMCTRL	8/32-bit memory controller with EDAC	0x01	0x054	0x1
AHBSTAT	AHB status register	0x01	0x052	0x0
AHBUART	Serial/AHB debug interface	0x01	0x007	0x0
AHBJTAG	JTAG/AHB debug interface	0x01	0x01C	0x1
GRSPW	SpaceWire link with DMA	0x01	0x01F	0x0
GRPCI	32-bit PCI bridge	0x01	0x014	0x0
PCIDMA	DMA controller for PCI bridge	0x01	0x016	0x0
CAN_OC	Dual CAN-2.0 interface	0x01	0x019	0x1
GRETH	10/100 Ethernet MAC	0x01	0x01D	0x0
APBUART	8-bit UART with FIFO	0x01	0x00C	0x1
GPTIMER	Modular timer unit	0x01	0x011	0x0
GPIO	General purpose I/O port	0x01	0x01A	0x1
CLKGATE	Clock gating module	0x01	0x02C	0x0
PCIARB	PCI arbiter	0x04	0x10	0x0
LEON 3FT	SPARC V8 32-bit processor	0x01	0x053	0x0

1.3 Memory Map

Table 1.2 is a memory map of the internal AHB/APB buses:

Table 1.2: Internal Memory Map

CORE	ADDRESS RANGE	BUS	Link
FTMCTRL	0x0000_0000 - 0x1FFF_FFFF : PROM area 0x2000_0000 - 0x3FFF_FFFF: I/O area 0x4000_0000 - 0x7FFF_FFFF: SRAM/SDRAM area	AHB	
APBCTRL	0x8000_0000 - 0x800F_FFFF: APB bridge	AHB	
Reserved	0x8100_0000 - 0x8FFF_FFFF: Unused	APB	
DSU3	0x9000_0000 - 0x9FFF_FFFF: Registers	APB	
Reserved	0xA000_0000 - 0xBFFF_FFFF: Unused	APB	
PCI	0xC000_0000 - 0xFFEF_FFFF: PCI Bus 0xFFF0_0000 - 0xFFF1_FFFF: PCI I/O space	APB	
CANOC1	0xFFF2_0000 - 0xFFF2_00FF: Registers	APB	
CANOC2	0xFFF2_0100 - 0xFFF2_01FF: Registers	APB	

CORE	ADDRESS RANGE	BUS	Link
Reserved	0xFFF2_2000 - 0xFFFF_EFFF: Unused	AHB	
AHB plug-and-play	0xFFF_FF000 - 0xFFFF_FFFF: Plug-and-play configuration	AHB	
FTMCTRL	0x8000_0000 - 0x8000_00FF: Registers	AHB	
APBUART	0x8000_0100 - 0x8000_01FF: Registers	AHB	
IRQMP	0x8000_0200 - 0x8000_02FF: Registers	APB	
GPTIMER	0x8000_0300 - 0x8000_03FF: Registers	AHB	
PCI	0x8000_0400 - 0x800_004FF: PCI DMA control registers	AHB	
PCI DMA CTRL	0x8000_0500 - 0x8000_05FF: Registers	AHB	
CLKGATE	0x8000_0600 - 0x8000_06FF: Registers	AHB	
AHBUART	0x8000_0700 - 0x8000_07FF: Registers	AHB	
PCIARB	0x8000_0800 - 0x8000_08FF: Registers	APB	
GPIO	0x8000_0900 - 0x8000_09AA: Registers	APB	
SPW1	0x8000_0A00 - 0x8000_0AFF: Registers	APB	
SPW2	0x8000_0B00 - 0x8000_0BFF: Registers	APB	
SPW3	0x8000_0C00 - 0x8000_0CFF: Registers	APB	
SPW4	0x8000_0D00 - 0x8000_0DFF: Registers	APB	
ETH	0x8000_0E00 - 0x8000_0EFF: Registers	APB	
AHBSTAT	0x8000_0F00 - 0x8000_0FFF: Registers	APB	
Reserved	0x8000_1000 - 0x800F_EFFF: Unused	APB	
APB plug-and-play	0x800F_F000 - 0x800F_FFFF: Plug-and-play configuration	APB	
Reserved	0x8010_0000 - 0xFFFF_EFFF: Unused	APB	

Access to addresses outside the ranges described above will return an AHB error response. Only 32-bit (word) accesses are supported for APB areas.

1.4 Interrupts

The interrupts are routed to the IRQMP interrupt controller and forwarded to the LEON 3FT processor. **Table 1.3** indicates the interrupt assignments:

Table 1.3: Interrupt Assignments

CORE	INTERRUPT #	FUNCTION
AHBSTAT	1	AHB bus error
APBUART	2	UART RX/RX interrupt
PCI	3	PCI DMA interrupt
CAN1	4	CAN1
CAN2	5	CAN2
GPTIMER	6, 7, 8, 9	Timer underflow interrupts

CORE	INTERRUPT #	FUNCTION
SPW1	10	SpaceWire1 RX/TX data interrupt
SPW2	11	SpaceWire2 RX/TX data interrupt
SPW3	12	SpaceWire3 RX/TX data interrupt
SPW4	13	SpaceWire4 RX/TX data interrupt
ETH	14	Ethernet RX/TX interrupt
GPIO	1 - 15	External I/O interrupt

1.5 Signals

The device has the following external signals, **Table 1.4**. The reset value for any signal is undefined if not otherwise indicated.

Table 1.4: Signal Assignments

PIN NAME	FUNCTION	RESET VALUE	DESCRIPTION
SYSCLK	I	--	Main system clock
$\overline{\mathrm{NODIV}}$	I*	--	Clock divider input. Set to '1' for 1x memory clock, '0' for 1/2x memory clock, relative to SYSCLK
$\overline{\mathrm{RESET}}$	IS	--	System reset
$\overline{\mathrm{ERROR1}}$	OD	--	Processor error mode indicator. This is an active low output.
$\overline{\mathrm{WDOG1}}$	OD	--	Watchdog indicator. This is an active low output.
ADDR[27:0]	O	[00...0]	Address bus
DATA[31:0]	I/O	High-z	Data bus
CB[7:0]	I/O	High-z	EDAC checkbits
$\overline{\mathrm{WRITE}}$	O	1	Write strobe for PROM and I/O
$\overline{\mathrm{OEN}}$	O	1	Output enables for PROM and I/O
$\overline{\mathrm{IOS}}$	O	1	I/O area chip select
$\overline{\mathrm{ROMS}[1:0]}$	O	1	PROM chip select
$\overline{\mathrm{RWEN}[3:0]}$	O	1	SRAM write enable strobe
$\overline{\mathrm{RAMOEN}[4:0]}$	O	1	SRAM output enable
$\overline{\mathrm{RAMS}[4:0]}$	O	1	SRAM chip select
READ	O	1	SRAM, PROM, and I/O read indicator
$\overline{\mathrm{BEXC}}$	I	--	Bus exception
$\overline{\mathrm{BRDY}}$	I	--	Bus ready
SDCLK	O	1	SDRAM clock
$\overline{\mathrm{SDRAS}}$	O	1	SDRAM row address strobe

PIN NAME	FUNCTION	RESET VALUE	DESCRIPTION
$\overline{\mathrm{SDCAS}}$	O	1	SDRAM column address strobe
$\overline{\mathrm{SDWEN}}$	O	1	SDRAM write enable
$\overline{\mathrm{SDCS[1:0]}}$	O	1	SDRAM chip select
SDDQM[3:0]	O	1	SDRAM data mask
CAN_RXD[1:0]	I	--	CAN receive data
CAN_TXD[1:0]	O	1	CAN transmit data
DSUACT	O	0	DSU mode indicator
DSUBRE	I	--	DSU break
DSUEN	I	--	DSU enable
DSURX	I	--	DSU UART receive data
DSUTX	O	1	DSU UART transmit data
$\overline{\mathrm{TRST}}$	I	--	JTAG reset
$\overline{\mathrm{TMS}}$	I	--	JTAG test mode select
TCK	I	--	JTAG clock
TDI	I	--	JTAG test data input
TDO	O	--	JTAG test data output
EMDC	O	0	Ethernet media interface clock
ERX_CLK	I	--	Ethernet RX clock
EMDIO	I/O	High-z	Ethernet media interface data
ERX_COL	I	--	Ethernet collision error
ERX_CRS	I	--	Ethernet carrier sense detect
ERX_DV	I	--	Ethernet receiver data valid
ERX_ER	I	--	Ethernet reception error
ERXD[3:0]	I	--	Ethernet receive data
ETXD[3:0]	O	[1010]	Ethernet transmit data
ETX_CLK	I	--	Ethernet TX clock
ETX_EN	O	0	Ethernet transmit enable
ETX_ER	O	0	Ethernet transmit error
$\overline{\mathrm{EMDINT}}$**	I	--	Ethernet management interface data interrupt
EDCLDIS**	I	--	Ethernet debug link disable
GPIO[15:0]	I/O	High-z	General Purpose I/O
SPW_CLK	I	--	SpaceWire clock
SPW_RXS[3:0]	I	--	SpaceWire receive strobe
SPW_RXD[3:0]	I	--	SpaceWire receive data
SPW_TXS[3:0]	O	0	SpaceWire transmit strobe
SPW_TXD[3:0]	O	0	SpaceWire transmit data
SPW_RXD[3:0]	I	--	SpaceWire receive data
RXD	I	--	UART receive data
TXD	O	1	UART transmit data
PCI_AD[31:0]	PCI-I/O	High-z	Bit 0 of PCI address and data bus

PIN NAME	FUNCTION	RESET VALUE	DESCRIPTION
$\overline{\text{PCI\_RST}}$	PCI-I	--	PCI reset input
PCI_CLK	PCI-I	--	PCI clock input
$\overline{\text{PCI\_CBE[3:0]}}$	PCI-I/O	High-z	PCI bus command and byte enable
PCI_PAR	PCI-I/O	High-z	PCI parity checkbit
$\overline{\text{PCI_FRAME}}$1	PCI-3	High-z	PCI cycle frame indicator
$\overline{\text{PCI_IRDY}}$1	PCI-3	High-z	PCI initiator ready indicator
$\overline{\text{PCI_TRDY}}$1	PCI-3	High-z	PCI target ready indicator
$\overline{\text{PCI_STOP}}$1	PCI-3	High-z	PCI target stop request
$\overline{\text{PCI_DEVSEL}}$1	PCI-3	High-z	PCI device select
PCI_IDSEL	PCI-I	--	PCI initializer device select
$\overline{\text{PCI\_REQ}}$	PCI-O	High-z	PCI request to arbiter in point-to-point configuration
$\overline{\text{PCI\_GNT}}$	PCI-I	--	PCI bus access indicator in point-to-point configuration
$\overline{\text{PCI\_HOST}}$	PCI-I	--	PCI host enable input
$\overline{\text{PCI\_ARB\_REQ[7:0]}}$	PCI-I	--	PCI arbiter bus request
$\overline{\text{PCI\_ARB\_GNT[7:0]}}$	PCI-O	High-z	PCI arbiter bus grant
$\overline{\text{PCI_PERR}}$1	PCI-3	High-z	PCI parity error indicator

Notes:

1. These pins require a pull-up resistor tied to V_{DD}. Specified resistor values are based on design requirements or as specified in the PCI Local Bus Specification Revision 2.1, Section **4.3.3**.
2. CB[15:8] is reset to a high logic level.

1.6 Clocking

1.1.1 Clock Inputs

Table 1.5 shows the clock inputs to the UT699:

Table 1.5: Clock Inputs

SIGNAL	DESCRIPTION
SYSCLK	Main system clock. The processor and AHB bus is clocked directly by CLK.
PCI_CLK	0-33 MHz PCI clock. Drives the PCI clock domain in the PCI interface.
SPW_CLK	10-200 MHz SpaceWire clock. Provides a clock to all four SpaceWire links.
ETX_CLK	Ethernet transmitter clock, 2.5 or 25 MHz generated by external PHY.
ERX_CLK	Ethernet receiver clock. 2.5 or 25 MHz generated by external PHY.

1.1.2 Clock Output

Table 1.6 shows the clock inputs to the UT699:

Table 1.6: Clock Output

SIGNAL	DESCRIPTION
SDCLK	Main clock that drives the SDRAM device. NOTE: If the AMBA frequency ("NODIV") is half the CPU frequency, the SDCLK is constant low during reset. If the AMBA frequency ("NODIV") is equal the CPU frequency, the SDCLK is toggling during reset.

1.1.3 Clock Gating

To save power, the AHB clock can be internally disabled from unused functional blocks. This is done through software by setting the appropriate bits in the clock gating unit. See Section 17.0 for more details.

1.7 Reset Operation

When $\overline{\text{RESET}}$ is asserted, the following signals are asynchronously driven to their inactive states:$\overline{\text{ROMS}}$, $\overline{\text{RAMS}}$, $\overline{\text{IOS}}$, $\overline{\text{OEN}}$ $\overline{\text{RAMOEN}}$. In addition, DATA [31:0] and CB [7:0] are driven to a high-z state.

When the PCI reset input ($\overline{\text{PCI\_RST}}$) is asserted, all PCI tri-state signals are asynchronously driven to their high-z state as required by the PCI specification.

Chapter 2: **LEON 3FT SPARC V8 32-bit Microprocessor**

1.8 Overview

The LEON 3FT is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications while combining high performance with low complexity and low power consumption. The processor core storage elements and on-chip memory are hardened against SEU errors utilizing various fault-tolerance techniques.

The LEON 3FT has the following main features: 7-stage pipeline with Harvard architecture, separate instruction and data caches, memory management unit, hardware multiplier and divider, on-chip debug support and a floating-point unit.

A block diagram of the LEON 3FT core follows:

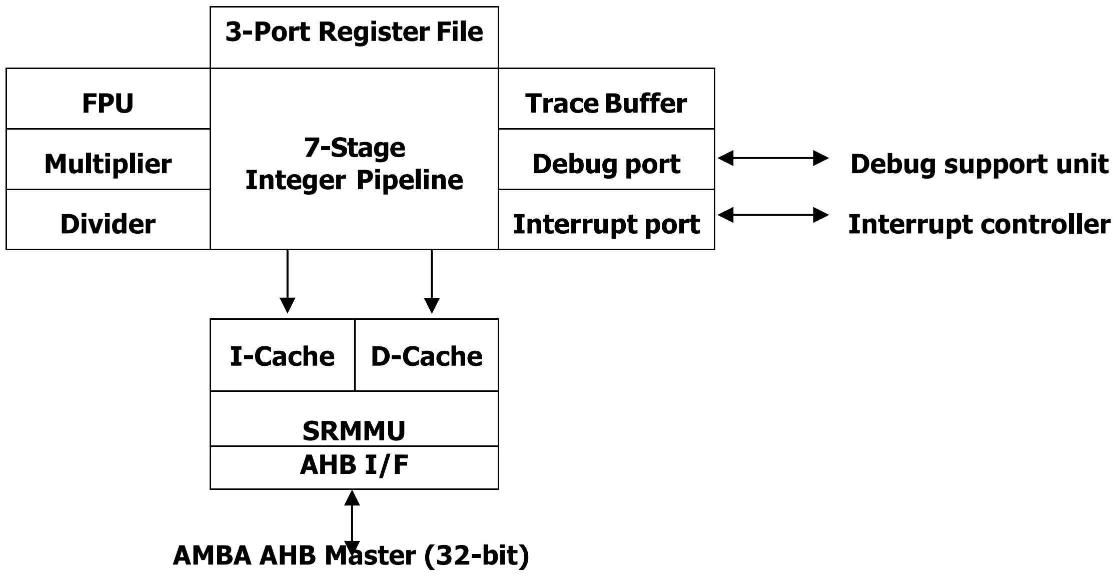


Figure 2.1: LEON 3FT Microprocessor Core Block Diagram

2.1.1 Integer Unit

The LEON 3FT integer unit supports the full SPARC V8 instruction set, including hardware multiplication and division instructions. The integer unit has eight register windows consisting of a total of one hundred and thirty-six (136) 32-bit general-purpose registers (*r* registers). These registers conform to the SPARC model for the general-purpose operand registers accessible through instructions, and implemented with RAM blocks. The instruction pipeline uses a Harvard architecture consisting of seven stages interfaced to a separate instruction and data cache.

2.1.2 Cache Sub-System

The processor is configured with 8kB instruction and 8kB data caches, both configured as two-way set-associative with 4kB per way and 32 bytes per line for instruction cache and 16 bytes per line for data cache. Sub-blocking is implemented with one valid bit per 32-bit word. The instruction cache uses streaming during line-refill to minimize refill latency. The data cache uses write-through policy and implements a double-word write buffer.

2.1.3 Floating-Point Unit

The LEON 3FT processor is configured with the Cobham's Gaisler floating-point unit (GRFPU). The GRFPU executes in parallel with the integer unit and does not block the processor operation unless a data or resource dependency exists.

2.1.4 Memory Management Unit

The LEON 3FT processor is configured with the SPARC V8 Reference Memory Management Unit (SRMMU). The SPARC V8 compliant SRMMU provides mapping between multiple 32-bit virtual address spaces and physical memory. A three-level hardware table-walk is implemented and the MMU has 16 Translation Look-Aside Buffer (TLB) entries for instructions and 16 TLB entries for data.

2.1.5 On-Chip Debug Support

The LEON 3FT pipeline provides support for non-intrusive debugging. Full access to all processor registers and cache memory are provided through the debug support unit (DSU). To aid software debugging, two hardware watchpoint registers are implemented. Each register can cause a breakpoint trap on an arbitrary instruction or data address range. When the DSU is enabled, the watchpoints can be used to enter debug mode. The DSU also allows single stepping, instruction tracing and hardware breakpoint/watchpoint control. An internal trace buffer monitors and stores executed instructions which can later be read out over the debug interface.

2.1.6 Interrupts

LEON 3FT supports the SPARC V8 trap model with a total of 15 asynchronous interrupts. The interrupt interface provides functionality to both generate and acknowledge interrupts.

2.1.7 AMBA Interface

The cache system implements an AMBA AHB master to load and store data to/from the caches. The interface is compliant with the AMBA-2.0 standard. During line refill, incremental bursts are generated to optimize the data transfer.

2.1.8 Power-down Mode

The LEON 3FT processor core implements a power-down mode which halts the pipeline and caches until the next interrupt. This is an efficient way to minimize power-consumption when the application is idle. The processor supports clock gating during the power down period that provides the means to check for wake-up conditions and maintain cache coherency.

1.9 LEON 3FT Integer Unit

2.1.9 Overview

The LEON 3FT integer unit implements the integer part of the SPARC V8 instruction set. The implementation is focused on high performance and low complexity. The LEON 3FT integer unit has the following main features:

- 7-stage instruction pipeline
- Separate instruction and data cache interfaces

- Eight register windows to access the 136 registers
- Hardware multiplier with 2 clocks latency
- Radix-2 divider (non-restoring)
- Single-vector trapping for reduced code size
- Static branch prediction

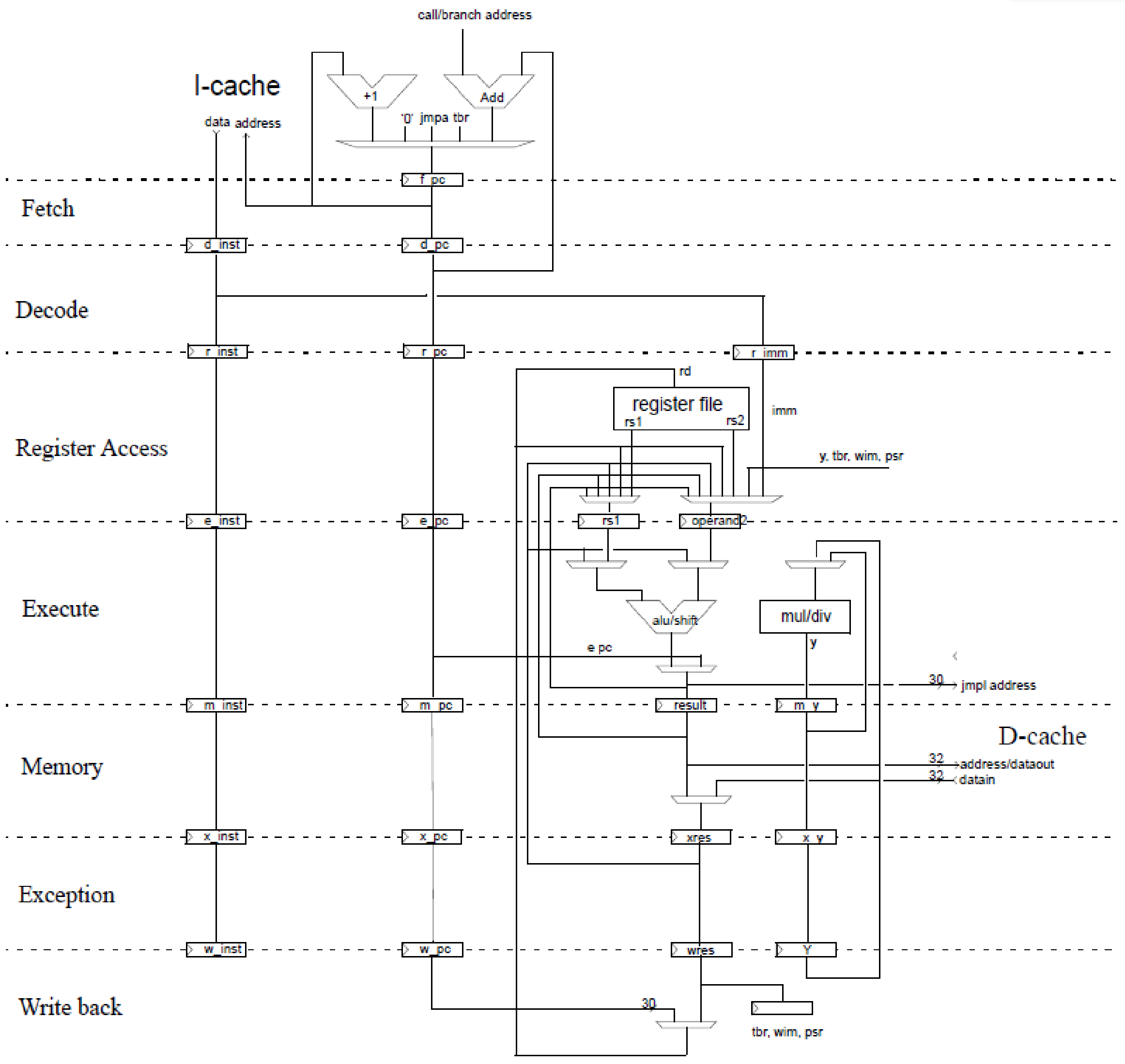


Figure 2.2: LEON 3FT Integer Unit Datapath Diagram

2.1.10 Instruction Pipeline

The LEON 3FT integer unit uses a single instruction issue pipeline with seven stages:

2. FE (Instruction Fetch): If the instruction cache is enabled and a cache hit occurs, the instruction is fetched from the instruction cache. Otherwise, the fetch is forwarded to the memory controller. The instruction is valid at the end of this stage and is latched inside the IU.
3. DE (Decode): The instruction is decoded and the CALL or branch target address is generated.

4. RA (Register Access): Operands are read from the register file or from internal data bypasses.
5. EX (Execute): ALU, logical, and shift operations are performed. For memory operations (e.g., LD/ST) and JMPL/RETT instructions, the address is generated.
6. ME (Memory): Data cache is accessed. If a data cache miss occurs, data is accessed from system memory and the cache is updated. Store data read out in the execution stage is written to the data cache at this time.
7. XC (Exception) Traps and interrupts are resolved. For cache reads, the data is aligned as appropriate.
8. WR (Write): The result of any ALU, logical, shift, or cache operations are written back to the register file.

Table 2.1 lists the cycles per instruction (assuming cache hit, and no integer condition codes or load interlock exist):

Table 2.1: Instruction Timing

INSTRUCTION	CYCLES
JMPL	3[1]
JMPL, RETT pair	4
Double load	2
Single store	2[3]
Double store	3[3]
SMUL/UMUL	1[2]
SDIV/UDIV	35
Taken Trap	5[3]
Atomic load/store	3
All other instructions	1

2.1.11 SPARC Implementer's ID

Cobham's Gaisler is assigned number 15 (0x0F) as SPARC implementer's identification. This value is hard-coded into bits 31:28 in the processor state register (%PSR:impl). The version number for LEON 3FT is 3, which is hard-coded in to bits 27:24 of the PSR (%PSR:ver).

2.1.12 Division Instructions

Full support for SPARC V8 division instructions is provided via instruction SDIV, UDIV, SDIVCC and UDIVCC. The division instructions perform a 64-by-32 bit divide and produce a 32-bit result. Rounding and overflow detection is performed as defined in the SPARC V8 standard.

2.1.13 Multiplication Instructions

The LEON 3FT processor supports the SPARC integer multiply instructions UMUL, SMUL UMULCC and SMULCC. These instructions perform 32x32-bit integer multiplication producing a 64-bit result. SMUL and SMULCC perform signed multiplication while UMUL and UMULCC perform unsigned multiplication. UMULCC and SMULCC also set the condition codes of the PSR to reflect the result of the operation. The multiply instructions are performed using a 16x16 signed hardware multiplier, which is iterated four times. To improve timing, the 16x16 multiplier is implemented with a pipeline register stage.

2.1.14 Hardware Breakpoints

The integer unit is configured with two hardware breakpoints. Each breakpoint consists of a pair of Ancillary State Registers (%asr24/25 and%asr26/27); one with the break address and one with a mask.

WPR1, WPR2 **%asr24, %asr26**

Bit#	31 … 2	1	0
R / W	WADDR[31:2]	▨	IF
Reset	[--...-]	--	0

Figure 2.3: Watchpoint Address Registers

Table 2.2: Description of Watchpoint Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	WADDR	--	Watch Address Defines the range of watch addresses used to generate a breakpoint.
1	RESERVED	--	
0	IF	0	Instruction Fetch Break Enable 0: Break on instruction fetch disabled. 1: Break on instruction fetch enabled.

WPMR1, WPMR2 **%asr25, %asr27**

Bit#	31 … 2	1	0
R / W	WMASK[31:2]	DL	DS
Reset	[--...-]	0	0

Figure 2.4: Watchpoint Mask Registers

Table 2.3: Description of Watchpoint Mask Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	WMASK[31:2]	--	Watch Mask These bits mask or unmask the corresponding bits in the WADDR. 0: Address bit not used by the WADDR.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Address bit used by the WADDR.
1	DL	0	Data Load Break Enable 0: Break on data load disabled 1: Break on data load
0	DS	0	Data Store Break Enable 0: Break on data store disabled 1: Break on data store

Any binary aligned address range can be watched. The range is defined by the WADDR field and masked by the WMASK field (WMASK[n] = 1 enables comparison). On a breakpoint hit, trap 0x0B is generated. By setting the IF, DL and DS bits, a hit can be generated on instruction fetch, data load or data store. Clearing these three bits effectively disables the breakpoint function.

2.1.15 Instruction Trace Buffer

The instruction trace buffer consists of a circular buffer that stores executed instructions. The trace buffer operation is controlled through the debug support interface and does not affect processor operation. The size of the trace buffer is 256 lines deep and 128 bits wide. The buffer stores the following information:

- Instruction address and opcode
- Instruction result
- Load/store data and address
- Trap information
- 30-bit time tag

The operation and control of the trace buffer is further described in **Chapter 14:.**

2.1.16 Processor Configuration Register

The application specific register 17 (%asr17) provides information on configuration of the LEON 3FT core. This can be used to enhance the performance of software. The register can be accessed through the RDASR instruction and has the following layout:

PCR **%asr17**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PI				RFT				▨							
W	▨															
Reset	0000				0000				[--...-]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	DW	SV	LD	FPU		M	V8	NWP			NWIN				
W				▨	▨			▨	▨			▨				
Reset	--	0	0	0	01		0	1	010			00111				

Figure 2.5: Processor Configuration Register

Table 2.4: Description of Processor Configuration Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-28	PI	[00…0]	Processor Index In multi-processor systems, each LEON 3FT core gets a unique index to support enumeration. Read=0; Write=don't care.
27-24	RFT	[00…0]	Register File RAM Timing Adjust Read=0000b; Must write 0000b.
23-15	RESERVED	[--…-]	
14	DW	0	Disable Write Error Trap 0: Write error trap (*tt*=0x2b) ignored. 1: Write error trap (*tt*=0x2b) allowed.
13	SV	0	Single-Vector Trapping Enable 0: Single-vector trapping disabled. 1: Single-vector trapping enabled.
12	LD	0	Load Delay 0: One-cycle load delay is used. 1: Two-cycle load delay is used. Read=0; Write=don't care.
11-10	FPU	01	Floating Point Implementation 00: No FPU 01: GRFPU (Hard Coded) 10: Meiko FPU 11: GRFPU-Lite Read=01; Write=don't care.
9	M	0	MAC Implementation 0: Optional multiply-accumulate (MAC) instruction not available. 1: Optional multiply-accumulate (MAC) instruction is available. Read=0; Write=don't care.
8	V8	1	Multiply and Divide Implementation 0: SPARC V8 multiplication and division instructions not available. 1: SPARC V8 multiplication and division instructions are available. Read=1; Write=don't care.
7-5	NWP	010	Watchpoint Implementation Number of implemented watchpoints. Read=010b; Write=don't care.
4-0	NWIN	00111	Register Window Implementation Number of implemented registers windows corresponds to NWIN+1. Read=00111b; Write=don't care.

2.1.17 Exceptions

LEON 3FT adheres to the general SPARC trap model. The **Table 2.5** below shows the implemented traps and their individual priority. When a trap occurs while Processor Status Register (PSR) bit ET=0, the processor halts execution and enters an error mode. The external processor error signal will be asserted (Active Low).

Table 2.5: Trap Allocation and Priority

TRAP	TT	PRI	DESCRIPTION	Class
reset	0x00	1	Power-on reset	Interrupting
write error	0x2B	2	Write buffer error	Interrupting
instruction_access_error	0x01	3	Error during instruction fetch	Precise
Illegal_instruction	0x02	5	UNIMP or other un-implemented instruction	Precise
privileged_instruction	0x03	4	Execution of privileged instruction in user mode	Precise
fp_disabled	0x04	6	FP instruction while FPU disabled	Precise
cp_disabled	0x24	6	CP instruction while co-processor disabled	Precise
watchpoint_detected	0x0B	7	Hardware breakpoint match	Precise
window_overflow	0x05	8	SAVE into invalid window	Precise
window_underflow	0x06	8	RESTORE into invalid window	Precise
register_hardware_error	0x20	9	Uncorrectable register file SEU error	Interrupting
mem_address_not_aligned	0x07	10	Memory access to un-aligned address	Precise
fp_exception	0x08	11	FPU exception	Deferred
cp_exception	0x28	11	Co-processor exception	Deferred
data_access_exception	0x09	13	Access error during load or store instruction	Precise
tag_overflow	0x0A	14	Tagged arithmetic overflow	Precise
divide_exception	0x2A	15	Divide by zero	Precise
trap_instruction	0x80 - 0xFF	16	Software trap instruction (TA)	Precise
interrupt_level_15	0x1F	17	GPIO 15	Interrupting
interrupt_level_14	0x1E	18	GPIO 14 / ETH	Interrupting
interrupt_level_13	0x1D	19	GPIO 13 / SPW4	Interrupting
interrupt_level_12	0x1C	20	GPIO 12 / SPW3	Interrupting
interrupt_level_11	0x1B	21	GPIO 11 / SPW2	Interrupting
interrupt_level_10	0x1A	22	GPIO 10 / SPW1	Interrupting
interrupt_level_9	0x19	23	GPIO 9 / GPTIMER 4 / GR1553B/ SPI	Interrupting
interrupt_level_8	0x18	24	GPIO 8 / GPTIMER 3	Interrupting
interrupt_level_7	0x17	25	GPIO 7 / GPTIMER 2	Interrupting
interrupt_level_6	0x16	26	GPIO 6 / GPTIMER 1	Interrupting
interrupt_level_5	0x15	27	GPIO 5 / CAN2	Interrupting
interrupt_level_4	0x14	28	GPIO 4 / CAN1	Interrupting
interrupt_level_3	0x13	29	GPIO 3	Interrupting
interrupt_level_2	0x12	30	GPIO 2 / APBUART	Interrupting
interrupt_level_1	0x11	31	GPIO 1 / AHBSTAT	Interrupting

2.1.18 Single Vector Interrupt (SVT)

The LEON 3FT supports Single-Vector Trapping (SVT) to reduce code size for embedded applications. When enabled, any taken trap always jumps to the reset trap handler whose address is defined by Trap Base Address Register (TBR) bits TBR.*tba*, or TBR[31:19], with the lower 12 bits don't care. The trap type will be indicated in TBR.*tt*, or TBR[11:4], and must be decoded by the shared trap handler. SVT is enabled by setting bit 13 in the PCR (%asr17).

2.1.19 Address Space Identifiers (ASI)

In addition to the address, the SPARC processor also generates an 8-bit Address Space Identifier (ASI) providing up to 256 separate, 32-bit address spaces. During normal operation, the LEON 3FT processor accesses instructions and data using ASI 0x08 - 0x0B as defined in the SPARC standard. The LDA/STA instructions are used to access the alternative address spaces. **Table 2.6** shows the ASI usage for LEON 3FT.

Table 2.6: ASI Usage

ASI	USAGE
0x01	Forced cache miss
0x02	System (cache control) registers
0x08	User instruction
0x09	Supervisor instruction
0x0A	User data
0x0B	Supervisor data
0x0C	Instruction cache tags
0x0D	Instruction cache data
0x0E	Data cache tags
0x0F	Data cache data
0x10	Flush entire instruction cache
0x11	Flush entire data cache

2.1.20 Power-Down

The processor supports a power-down feature to minimize power consumption during idle periods. The power-down mode is entered by performing a WRASR instruction to %asr19:

```
wr    %g0, %asr19    // write 0x0 to%asr19
```

During power-down, the pipeline is halted until the next interrupt occurs; therefore, this instruction should not be executed with interrupts disabled or the processor never wake up. Signals inside the processor pipeline and caches are then static, reducing power consumption from dynamic switching.

2.1.21 Processor Reset Operation

The processor is reset by asserting the $\overline{\mathrm{RESET}}$ input for at least four clock cycles. **Table 2.7** indicates the reset values of the registers that are affected by the reset. All other registers either maintain their value or are undefined.

Table 2.7: Processor Reset Value

REGISTER	RESET VALUE
PC (Program Counter)	0x00000000
nPC (Next Program Counter)	0x00000004
PSR (Processor Status Register)	ET=0, S=1

Code execution starts at address 0 following a reset.

2.1.22 Integer Unit SEU Protection

SEU protection for the integer unit register file (RF) is implemented with a Bose-Chaudhuri-Hocquenghem (BCH) algorithm utilizing 7 check bits. The protection logic can correct up to 1 error per 32-bit word in the register file and detect two errors. The correction is done transparently to the software and does not affect the instruction timing. If a detected SEU error cannot be corrected, trap 0x20 is generated.

ASR register 16 (%asr16) is used to control the IU register file SEU protection. It is possible to disable the SEU protection by setting the IDI bit and to inject errors using the TE bits. Corrected errors in the register file are counted and available in ICNT fields. The counter saturates at its maximum value (7) and should be reset by software after read-out.

RPCR **%asr17**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																IUFT
W																
Reset	[--...-]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IUFT		ICNT			TB								DP	TE	RP
W																
Reset	10		[--...-]			[--...-]								-	-	-

Figure 2.6: Register Protection Control Register

Table 2.8: Description of Register Protection Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-17	RESERVED	[--...-]	

16-14	IUFT	010	Integer Unit Fault-Tolerant Identification
			Read=010b
			Write=don't care.

13-11	ICNT	[--...-]	Integer Unit Register File Error Counter Number of detected parity errors in the IU register file. 000: 0 errors 001: 1 error 010: 2 errors ... 111: 7 errors

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
10-3	TB	[--...-]	Register File Test Bits In test mode, these bits are XORed with correct parity bits and then written back to the register file.
2	DP	-	Diagnostic Parity RAM Select Selects to insert errors in the main or redundant register file memory. See the table below for a description of operation.
1	TE	-	Integer Unit Register File Test Enable Disables or enables register file test mode. See the table below for a description of operation.
0	RP	-	Integer Unit Register File Protection Disable 0: Enable IU RF parity protection. 1: Disable IU RF parity protection.

2.1.23 Data Scrubbing

When a data word in the register file is corrected, the corrected value is used during the execution of the current instruction, but not automatically written back to the register file. There is generally no need to perform data scrubbing (read-write operation) on the IU register file. During normal operation, the active part of the IU register files will be flushed to memory on each task switch. This causes all saved registers to be checked and corrected if necessary. Since most real-time operating systems perform several task switches per second, the data in the register file will be frequently refreshed.

8.1 Floating Point Unit

The UT699 SPARC V8 architecture is configured with the GRFPU from Cobham's Gaisler.

The high-performance GRFPU operates on single- and double-precision operands and implements all SPARC V8 FPU instructions except quad precision instructions. The FPU is interfaced to the LEON 3FT pipeline using a LEON 3FT-specific FPU controller (GRFPC) that allows FPU instructions to be executed simultaneously with integer instructions. Only in case of a data or resource dependency is the integer pipeline held. The GRFPU is fully pipelined and allows the start of one instruction each clock cycle, with the exception of FDIV and FSQRT, which can only be executed one at a time. The FDIV and FSQRT instructions are executed in a separate divide unit and do not block the GRFPU from performing other operations in parallel.

All instructions except FDIV and FSQRT have a latency of four clock cycles at instruction level. **Table 2.9** below shows the GRFPU instruction timing when used together with GRFPC.

Table 2.9: GRFPU Worst-Case Instruction Timing with GRFPC

INSTRUCTION	THROUGHPUT	LATENCY
FADDS, FADDD, FSUBS, FSUBD, FMULS, FMULD, FSMULD, FITOS, FITOD, FSTOI, FDTOI, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES, FCMPED	1	4
FDIVS	14	16
FDIVD	15	17
FSQRTS	22	24

INSTRUCTION	THROUGHPUT	LATENCY
FSQRTD	23	25

The GRFPU controller uses the SPARC deferred traps and the GRFPU deferred trap queue (FQ) can contain up to eight queued instructions when a GRFPU exception is taken. The register file for the GRFPU consists of thirty-two, 32-bit registers. In the UT699, the register file has been implemented with SEU-hardened flip-flops and does not need SEU error detection or correction.

8.2 Cache Sub-System

The processor's L1 cache will cache addresses that are marked as cacheable. The cacheable address ranges are:

• PROM area	: 0x0000_0000 - 0x1FFF_FFFF
• SRAM/SDRAM area	: 0x4000_0000 - 0x7FFF_FFFF

When the processor has a memory location in cache and the same memory location is updated by another bus master (PCI controller, Ethernet controller, SpaceWire controllers) then the cache will not be automatically updated and the L1 cache will contain stale data. To maintain coherency between external memory and the L1 cache all memory areas that may be updated by other AMBA masters need to be accessed using forced cache misses (load alternate instruction to ASI 0x01). Coherency can also be attained by performing a cache flush before accessing shared memory areas.

Cache coherency is maintained using bus snooping. When the processor has a memory location in cache and the same memory location is updated by another bus master (PCI controller, Ethernet controller, SpaceWire controllers) then the corresponding cache line will be automatically invalidated by the processor. Cache coherency using snooping is only available for the data cache. If instructions that may be in the instruction cache are modified in external memory, then the L1 instruction cache needs to be flushed.

Compatibility note: The UT699 does not support bus snooping and does not automatically maintain cache coherency. Therefore, software written for the UT699 may force cache misses (using load alternate with ASI 0x01) when accessing shared memory areas. The UT700/UT699E always fetches a full cache line on a cache miss. Code that makes use of ASI 0x01 for each load operation may cause an unnecessary large number of misses and this leads to performance degradation.

2.1.24 Overview

The LEON 3FT processor implements Harvard architecture with separate internal instruction and data buses connected to two independent cache controllers. The instruction and data cache controllers can be separately configured via the configuration registers. The cache configuration is a two-way set associative with a set size of 4kB per way divided into cache lines with 16 bytes per line for data cache and 32 bytes per line for instruction cache. Both data and instruction caches use a least recently used (LRU) replacement policy.

Cacheability for both caches are controlled through the AHB plug-and-play address information. The memory mapping for each AHB slave indicates whether the area is cacheable, and this information is used to statically determine which access will be treated as cacheable. This approach means that the cacheability mapping is always coherent with the current AHB configuration.

The detailed operation of the instruction and data caches is described in the following sections.

2.1.25 Instruction Cache

The instruction cache is implemented as a 2-way set-associative cache with LRU replacement. Each way is 4kB large and divided into cache lines of 32 bytes. Each line has a cache tag associated with it, consisting of a tag field and valid bit for each 4-byte sub-block. On an instruction cache miss to a cachable location, the instruction is fetched and the corresponding tag and data line are updated.

If instruction burst fetch is enabled in the cache control register (CCR), the cache line is filled from main memory starting at the missed address and until the end of the line. At the same time, the instructions are forwarded to the IU. If the IU cannot accept the streamed instructions due to internal dependencies or multi-cycle instruction, the IU is halted until the line fill is completed. If the IU executes a control transfer instruction (branch/CALL/JMPL/RETT/TRAP) during the line fill, the line fill will be terminated on the next fetch. If instruction burst fetch is enabled, instruction streaming is enabled even when the cache is disabled. In this case, the fetched instructions are only forwarded to the IU and the cache is not updated. During cache line refill, incremental bursts are generated on the AHB bus.

If a memory access error occurs during a line fill with the IU halted, the corresponding valid bit in the cache tag will not be set. If the IU later fetches an instruction from the failed address, a cache miss will occur, triggering a new access to the failed address. If the error remains, an instruction access error trap (*tt*=0x01) will be generated.

2.1.26 Data Cache

The data cache is configured identical to the instruction cache with two ways of 4kB, 16 bytes/line and LRU replacement. On a data cache read-miss to a cachable location, 4 bytes of data are loaded into the cache from main memory. The write policy for stores is write-through with no-allocate on a write miss. If a memory access error occurs during a data load, the corresponding valid bit in the cache tag will not be set and a data access error trap (*tt*=0x09) will be generated.

2.1.27 Write Buffer

The write buffer (WRB) consists of three 32-bit registers used to temporarily hold store data until it is sent to the destination device. For half-word or byte stores, the stored data is replicated into proper byte alignment for writing to a word-addressed device before being loaded into one of the WRB registers. The WRB is emptied prior to a load-miss cache-fill sequence to avoid any stale data from being written into the data cache.

Since the processor executes in parallel with the write buffer, a write error will not cause an exception to the store instruction. Depending on memory and cache activity, the write cycle may not occur until several clock cycles after the store instruction has completed. If a write error occurs, the currently executing instruction will take trap 0x2B.

Note: The 0x2B trap handler should flush the data cache, since a write hit would update the cache while the memory would keep the old value due the write error.

2.1.28 Instruction and Data Cache Tags

The instruction and data cache tags and shown in **Figure 2.7** and **Figure 2.8**.

Bit#	31 12	11 8	7 0
R	ITAG[19:0]		IVAL[7:0]
W			
Reset	[00…0]	0000	[00…0]

Figure 2.7: Instruction Cache Tag Layout for 4KB per Way with 32 Bytes/line

Table 2.10: Instruction Cache Tag Layout for 4KB per Way with 32 Bytes/line Description

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-12	ITAG	[00...0]	Instruction Cache Address Tag Contains the tag address of the cache line.
11-8	RESERVED	[00...0]	Read=00000b; Write=don't care.
7-0	IVAL	[00...0]	Instruction Tag Valid When set, the corresponding sub-block of the cache line contains valid data. These bits are set when a sub-block is filled due to a cache miss; a cache fill which results in a memory error will leave the valid bit unset. A FLUSH instruction will clear all valid bits. IVAL[0]: corresponds to address 0 in the I-cache line IVAL[1]: corresponds to address 1 in the I-cache line IVAL[4]: corresponds to address 2 in the I-cache line IVAL[3]: corresponds to address 3 in the I-cache line IVAL[4]: corresponds to address 4 in the I-cache line IVAL[5]: corresponds to address 5 in the I-cache line IVAL[6]: corresponds to address 6 in the I-cache line IVAL[7]: corresponds to address 7 in the I-cache line

Bit#	31 12	11 4	3 0
R	DTAG[19:0]	▨	DVAL
W			
Reset	[00…0]	[00…0]	0000

Figure 2.8: Data Cache Tag Layout for 4KB per Way with 16 Bytes/line

Table 2.11: Description of Data Cache Tag Layout for 4KB per Way with 16 Bytes/line

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-12	DTAG	[00...0]	Data Cache Address Tag Contains the tag address of the cache line.
11-4	RESERVED	[00...0]	Read=00000000b; Write=don't care.
3-0	DVAL	[00...0]	Data Tag Valid When set, the corresponding sub-block of the cache line contains valid data. These bits are set when a sub-block is filled due to a cache miss; a cache fill which results in a memory error will leave the valid bit unset. A FLUSH instruction will clear all valid bits. DVAL[0]: corresponds to address 0 in the D-cache line DVAL[1]: corresponds to address 1 in the D-cache line DVAL[4]: corresponds to address 2 in the D-cache line DVAL[3]: corresponds to address 3 in the D-cache line

2.1.29 Cache Flushing

Both instruction and data caches are flushed by executing the FLUSH instruction or by writing to any location with ASI=0x10 or ASI=0x11. In addition, the entire instruction cache is flushed by setting the FI bit in the cache control register (CCR). The entire data cache is flushed by setting the FD bit in the CCR. Cache flushing takes one cycle per cache line, during which, the IU will not be halted and the caches are disabled. When the flush operation is completed, the cache resumes the state (disabled, enabled or frozen) indicated in the cache control register. Diagnostic access to the cache is not possible during a FLUSH operation and will cause a data exception (*tt*=0x09) if attempted.

2.1.30 Diagnostic Cache Access

Tags and data in the instruction and data cache can be accessed through ASI address space 0x0C, 0x0D, 0x0E and 0x0F by executing LDA and STA instructions. The ITAG and DTAG fields of the cache tag define the upper 20 bits of the address, while the twelve (12) least significant bits of the address correspond to the index of the cache set.

2.1.30.1 Diagnostic Reads of Instruction and Data Cache

Cache tags are read by executing an LDA instruction with ASI=0x0C for instruction cache tags and ASI=0x0E for data cache tags. A cache line and set are indexed by the address bits making up the cache offset and the least significant bits of the address bits making up the address tag. Similarly, the data sub-blocks may be read by executing an LDA instruction with ASI=0x0D for instruction cache data and ASI=0x0F for data cache data. The sub-block to be read in the indexed cache line and set is selected 64, the *regaddr* field of the LDA or STA instruction.

2.1.30.2 Diagnostic Writes to Instruction and Data Cache

Cache tags can be directly written to by executing a STA instruction with ASI=0xC for the instruction cache tags and ASI=0x0E for the data cache tags. The cache line and set are indexed by the address bits making up the cache offset and the least significant bits of the address bits making up the address tag. D[31:10] is written into the ATAG filed and the valid bits are written with D[7:0] of the write data for instruction cache and D[3:0] for data cache. Bit D[9] is written into the LRR bit (disabled) and D[8] is written into the lock bit (disabled). The data sub-blocks can be directly written by executing a STA instruction with ASI=0xD for the instruction cache data and ASI=0xF for the data cache data. The sub-block to be read in the indexed cache line and set is selected by A[4:2].

2.1.31 Cache Control Register

The operation of the instruction and data caches is controlled through a common Cache Control Register (CCR) is shown in **Figure 2.9**. The instruction cache can operate in the disabled, enabled, or frozen mode as configured by the Instruction Code State (ICS) field. The data cache can operate in the disabled or enabled modes, as configured in the Data Cache State (DCS) field. If disabled, no cache operation is performed and load and store requests are passed directly to the memory controller. If enabled, the cache operates as described above. In the frozen state, the cache is accessed and kept synchronized to the main memory as if it were enabled, but no new lines are allocated on read misses.

Table 2.12: ASI 0x02 (System Registers) Address Map

REGISTER	ADDRESS
Cache control register	0x00
Instruction cache configuration register	0x08

Data cache configuration register	0x0C

CCR

ASI = 0x02
Offset = 0x00

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R				PS	TB					0	0	FT				IB
W										FD	FI					
Reset	000			0	0000				-	0	0	01		0	1	1

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP	DP	ITE		IDE		DTE		DDE		DF	IF	DCS		ICS	
W																
Reset	0	0	00		00		00		00		0	0	11		11	

Figure 2.9: Cache Control Register

Table 2.13: Cache Control Register Description

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-29	RESERVED	000	
28	PS	0	Parity Select 0: Diagnostic read will return tag or data word. 1: Diagnostic read will return the check bits in the bits.
27-24	TB	0000	Test Bits 0: No effect. 1: Check bits will be XORed with test bits TB during diagnostic write.
23	RESERVED	1	Must write 0.
22	FD	0	Flush Data Cache 0: No effect. 1: Flush the data cache. Read=0.
21	FI	0	Flush Instruction Cache 0: No effect. 1: Flush the instruction cache. Read=0.
20-19	FT	01	Fault Tolerant Mode 00: No fault-tolerance 01: 4-bit parity checking 10: Unused 11: Unused
18-17	RESERVED	0	
16	IB	1	Instruction Burst Fetch 0: Disable burst fill during instruction fetch.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Enable burst fill during instruction fetch.
15	IP	0	Instruction Cache Flush Pending 0: Instruction cache flush operation not in progress. 1: Instruction cache flush operation is in progress.
14	DP	0	Data Cache Flush Pending 0: Data cache flush operation is not in progress. 1: Data cache flush operation is in progress.
13-12	ITE	00	Instruction Cache Tag Errors Number of detected parity errors in the instruction tag cache.
11-10	IDE	00	Instruction Cache Data Errors Number of parity errors in the instruction data cache.
9-8	DTE	00	Data Cache Tag Errors Number of detected parity errors in the data tag cache.
7-6	DDE	00	Data Cache Data Errors Number of detected parity errors in the data cache.
5	DF	0	Data Cache Freeze on Interrupt Data cache response to asynchronous interrupt: 0: Normal operation. Read = 0, Write = don't care
4	IF	0	Instruction Cache Freeze on Interrupt Instruction cache response to asynchronous interrupt: 0: Normal operation. 1: Instruction cache automatically frozen.
3-2	DCS	11	Data Cache State Indicates the current data cache state: 00: Disabled x1: Enabled x=don't care.
1-0	ICS	11	Instruction Cache State Indicates the current instruction cache state: x0: Disabled 01: Frozen 11: Enabled

If the DF or IF bit is set, the corresponding cache will be frozen when an asynchronous interrupt is taken. This can be beneficial in a real-time system to allow a more accurate calculation of worst-case execution time for a code segment. The execution of the interrupt handler will not evict any cache lines. When control is returned to the interrupted task, the cache state is identical to what it was before the interrupt. If a cache has been frozen by an interrupt, it can only be re-enabled by setting the DCS or ICS fields to the enabled state. This is typically done at the end of the interrupt handler before control is returned to the interrupted task.

2.1.32 Error Protection

Each word in the cache tag or cache data is protected by four parity bits. An error during a cache access causes a cache line flush and a re-execution of the failing instruction. This ensures the complete cache line (tags and data) is refilled from external memory. For every detected error, the corresponding counter in the cache control register is incremented. The counters saturate at their maximum value of three and should be reset by software after reading the fields. The cache memory check bits can be diagnostically read by setting the PS bit in the cache control register and then performing a normal tag or data diagnostic read.

2.1.33 Cache Configuration Registers

The configuration of the two caches is defined in two registers: the instruction and data configuration registers. These registers are read-only and indicate the size and configuration of the caches.

ASI = 0x02

ICCR, DCCR **Offset = 0x08, 0x0C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CL		CP		SN	WAYS			WSIZE				LR	LSIZE		
W																
Reset	0	--	01		1	011			0010				0	011(I)/010(D)		

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LRSZ				LRSA								MMU			
W																
Reset	0				[00…0]								1	000		

Figure 2.10: Cache Configuration Register

Table 2.14: Description of Cache Configuration Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	CL	0	Cache Locking 0: Cache locking is not implemented. Read=0; Write=don't care.
30	RESERVED		
29-28	CP	01	Cache Replacement Policy 01: Least recently used (LRU). Read=01; Write=don't care.
27	SN	1	Cache Snooping 0: No snooping. Read=1; Write=don't care.
26-24	WAYS	011	Cache Associativity 011: four-way set associative Read=011b; Write=don't care
23-20	WSIZE	0010	Set Way Size Indicates the size in KB of each cache way. Size = 2^{WSIZE} Read=0010b; Write=don't care.
19	LR	0	Local Ram Present Read=0; Write=don't care.
18-16	LSIZE	011 (I) 010 (D)	Line Size Indicated the size (words) of each cache line. Line size = 2^{LSZ} Read=011b for I-cache and 010b for D-cache; Write=don't care.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
15-12	LRSZ	0	Local Ram Size Read=0; Write=don't care.
11-4	LRSA	0	Local Ram Start Address Read=0; Write=don't care.
3	MMU	1	MMU Present 0: MMU not present. 1: MMU present. Read=1; Write=don't care.
2-0	RESERVED	000	

All cache registers are accessed through load/store operations to the alternate address space (LDA/STA), using ASI = 0x02. **Table 2.15** below shows the register addresses. The following assembly instruction shows how to read any of the cache system registers.

```
lda %g1, [rr] 2           // Load register%g1 with the contents of the
                          // Corresponding cache register
```

Where *rr* is 0x00, 0x08, or 0x0C. Following this instruction, the contents of the cache register whose address is *rr* will be loaded into global register%g1.

2.1.34 Software Consideration

After reset, the caches are disabled and the value of cache control register (CCR) is 0. Before the caches may be enabled, a flush operation must be performed to initialize (clear) the tags and valid bits. A suitable assembly sequence could be:

```
flush

set    0x81000F, %g1     // Load global register %g1 with 0x0081000F
sta    %g1, [%g0] 2      // Store the contents of %g1 to the CCR
```

8.3 Memory Management Unit

A memory management unit (MMU) compatible with the SPARC V8 reference MMU can optionally be configured to operate in-line with the cache subsystem. For details on operation, see the SPARC V8 manual.

2.1.35 MMU ASI Usage

When the MMU is enabled, the following ASI mappings are added.

Table 2.15: MMU ASI Usage

Address	Register
0x13	MMU flush instruction and data context
0x14	MMU diagnostic dcache context access
0x1E	Separate snoop tags
0x15	MMU diagnostic icache context access (requires that the instruction cache is disabled)
0x18	Flush TLB and instruction and data cache
0x19	MMU registers

Address	Register
0x1C	MMU bypass
0x1D	MMU diagnostic access
0x1E	MMU snoop tags diagnostic access

2.1.36 Cache Operation

When the MMU is disabled, the caches operate normally with physical address mapping. When the MMU is enabled, the cache tags contain the virtual address and include an 8-bit context field. Because the cache is virtually tagged, no extra clock cycles are needed in the case of a cache load hit. In the case of a cache miss or store hit (write-through cache) at least two extra clock cycles are used if there is a translation look-aside buffer (TLB) hit. If there is a TLB miss, the page table must be traversed; resulting in up to four AMBA read accesses and one possible write-back operation.

2.1.37 MMU Registers

The following MMU registers are implemented.

Table 2.16: MMU Registers (ASI = 0x19)

ADDRESS	REGISTER
0x000	MMU control register
0x100	Context pointer register
0x200	Context register
0x300	Fault status register
0x400	Fault address register

2.1.38 Translation Look-Aside Buffer (TLB)

The MMU is configured to use a separate TLB for instructions and data. The number of entries is sixteen for instructions and sixteen for data. The organization of the TLB and number of entries is not visible to the software and does not require any modification to the operating system.

8.4 RAM Usage

2.1.39 Integer Unit Register File

The integer unit register file has one write port and two read ports, all 39 bits wide. The data is organized as 32-data bits + 7 BCH checksum bits. The register file is implemented with two sets of three RAM blocks. Each set is implementing with 256x48 2-port RAM by concatenating three 256x16 2-port RAM blocks. The 32-bit data is stored in bits [31:0] while the parity bits are stored in [38:32]. Bits [47:39] are unused and tied to ground. The BCH bits are generated as follows:

P0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^ D29 ^ D31
P1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^ D26 ^ D28
$\overline{P2}$ = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
$\overline{P3}$ = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29
P4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31
P5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
P6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31

To form a 3-port register file, the two sets share their write ports for the same write address while the read ports have individual addresses. This way the data is always duplicated in both sets. For testing purposes, the parity bits can be individually inverted during a write, and the writing to one of the sets can be disabled. This functionality is controlled through %asr16.

2.1.40 Floating Point Unit (FPU) Register File

The FPU register file is protected with SEU hardened flip-flops.

2.1.41 Cache Memories

The following sections detail how cache information is stored in physical memory.

2.1.41.1 Instruction Cache Tags

The instruction tags are made up by 8 valid bits, 20 tag address bits, eight MMU context bits and four parity bits. A total of 40 bits are dedicated to each Instruction Cache line. There are a total of 128 instruction tags. Instruction cache tags have the following allocation.

Table 2.17: Instruction Cache Tags

BITS	USAGE
47-39	Unused and tied to ground
38-35	Parity Bit 38: XOR [35:20] Bit 37: XOR [19:12] Bit 36: XOR [11:8] Bit 35: XOR [7:0]
34-27	MMU context
26-8	Tag address
7-0	Valid 0: Word not valid 1: Word valid

2.1.41.2 Data Cache Tags

The data tags are made of one valid bit, 20 tag address bits, eight MMU context bits and four parity bits. A total of 33 bits are dedicated for each Data Cache line. There are a total of 256 data tags. Data cache tags have the following allocation.

Table 2.18: Data Cache Tags

BITS	USAGE
35-33	Unused and tied to ground
32-29	Parity Bit 39: XOR [28:13] Bit 38: XOR [12:5] Bit 30: XOR [4:1] Bit 29: XOR [0]
28-21	MMU context
20-1	Tag address
0	Valid 0: Word not valid 1: Word valid

2.1.41.3 Instruction and Data Cache Memory

The data part of the instruction and data caches consist of 32-data bits and four parity bits. They are stored in a 40-bit set made up by two 1024x20 RAM blocks. The bits are allocated as follows:

Table 2.19: Data Cache Tags

BITS	USAGE
39-36	Unused and tied to ground
35-32	Parity Bit 35: XOR [31:24] Bit 34: XOR [23:16] Bit 33: XOR [15:8] Bit 32: XOR [7:0]
31-0	Data

Chapter 3: Memory Controller with EDAC

3.1 Overview

The memory controller provides a bridge between external memory and the AHB bus. The memory controller handles four types of devices: PROM, Asynchronous Static Ram (SRAM), Synchronous Dynamic Ram (SDRAM) and memory mapped Input/Output (I/O) devices. The PROM, SRAM and SDRAM areas can be EDAC-protected using a (39, 7) BCH code. The EDAC provides single-error correction and double-error detection for each 32-bit memory word.

The SDRAM area can optionally also be protected using Reed-Solomon coding. In this case, a 16-bit checksum is used for each 32-bit word and any two adjacent 4-bit (nibble) errors can be corrected.

The memory controller is configured through three configuration registers accessible via an APB bus interface. The external data bus can be configured in 8, 16, or 32-bit mode depending on application requirements. The controller decodes three address spaces on the AHB bus (PROM, I/O, and SRAM/SDRAM).

External chip-selects are provided for two PROM banks, one I/O bank, five SRAM banks and two SDRAM banks. **Figure 3.1** below shows how the notional connection to the different device types is made.

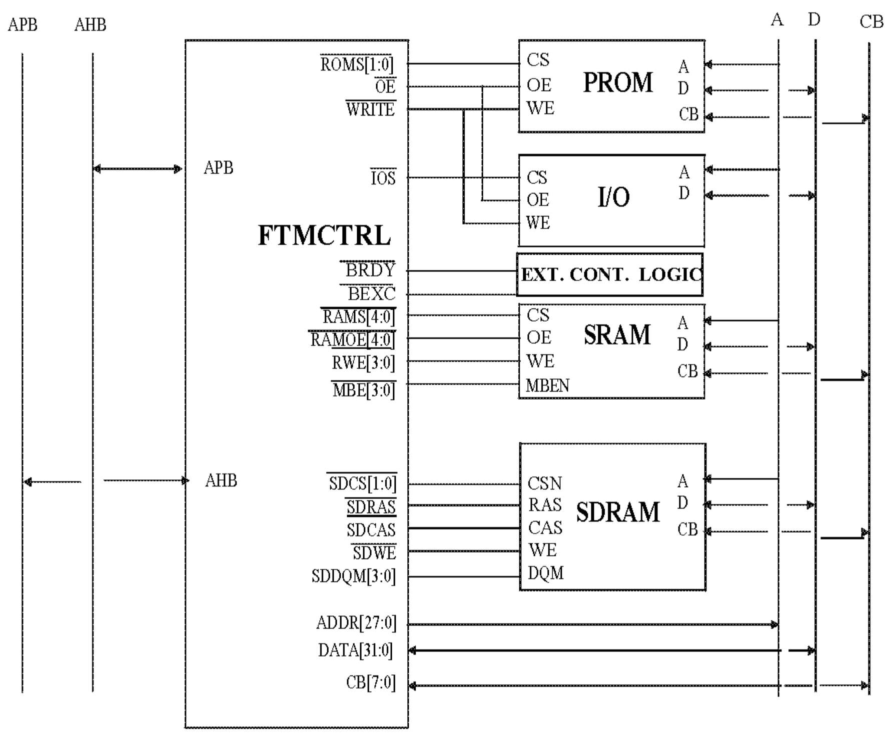


Figure 3.1: FTMCTRL Connected to Different Types of Memory Devices

3.2 Memory Mapped I/O

Accesses to I/O have similar timing to the PROM accesses. The I/O select ($\overline{\mathrm{IOS}}$) and output enable ($\overline{\mathrm{OE}}$) signals are delayed one clock to allow for a stable address before it is asserted. See Section **3.17** for timing diagram examples of I/O accesses.

3.3 SRAM Access

The SRAM area is divided up to five RAM banks. The size of banks 1-4 ($\overline{\mathrm{RAMS}[3:0]}$) is programmed in the RAM bank-size field (MCFG2[12:9]) and can be set in binary steps from 8 Kbyte to 256 Mbyte. The fifth bank ($\overline{\mathrm{RAMS}[4]}$) decodes the upper 512 Mbyte. A read access to SRAM consists of two data cycles and between zero and three waitstates. The read data (and optional EDAC check-bits CB[6:0]) are latched on the rising edge of the clock on the last data cycle. Accesses to $\overline{\mathrm{RAMS}[4]}$ can further be stretched by de-asserting $\overline{\mathrm{BRDY}}$ until the data is available. On non-consecutive accesses, a lead-out cycle is added after a read cycle to prevent bus contention due to slow turn-off time of memories. See Section **3.16** for timing diagram examples of SRAM accesses.

For read accesses to $\overline{\mathrm{RAMS}[4:0]}$, a separate output enable signal, $\overline{\mathrm{RAMOE}[\mathrm{n}]}$ is provided for each RAM bank and only asserted when that bank is selected. A write access is similar to the read access, but takes a minimum of three cycles.

Each byte lane has an individual write strobe to allow efficient byte and half-word writes. If the memory uses a common write strobe for the full 16 or 32-bit data, the read-modify-write bit MCFG2 should be set to enable read-modify-write cycles for sub-word writes. See Section **3.17** for timing diagram examples of SRAM accesses.

3.4 8-bit and 16-bit PROM and SRAM Access

To support applications with low memory and performance requirements efficiently, the SRAM and PROM areas can be individually configured for 8-bit or 16-bit operation by programming the ROM and RAM size fields in the memory configuration registers. Since read access to memory is always done on 32-bit word basis, read access to 8-bit memory is transformed in a burst of four read cycles while access to 16-bit memory generates a burst of two 16-bits reads. During writes, only the necessary bytes will be written. The following figures show interface examples with 8-bit, 16-bit, and 32-bit PROM and SRAM.

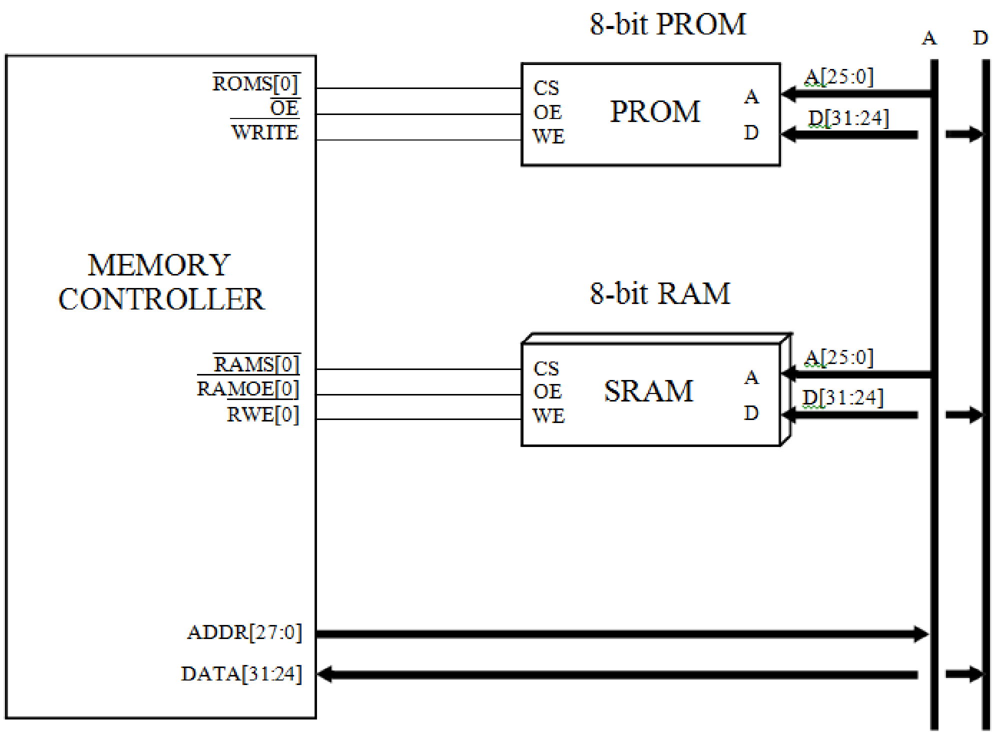


Figure 3.2: 8-bit Memory Interface Example

*When the EDAC is enabled in 8-bit bus mode, only the first bank select ($\overline{\text{RAMS[0]}}$, $\overline{\text{ROMS[0]}}$) can be used.

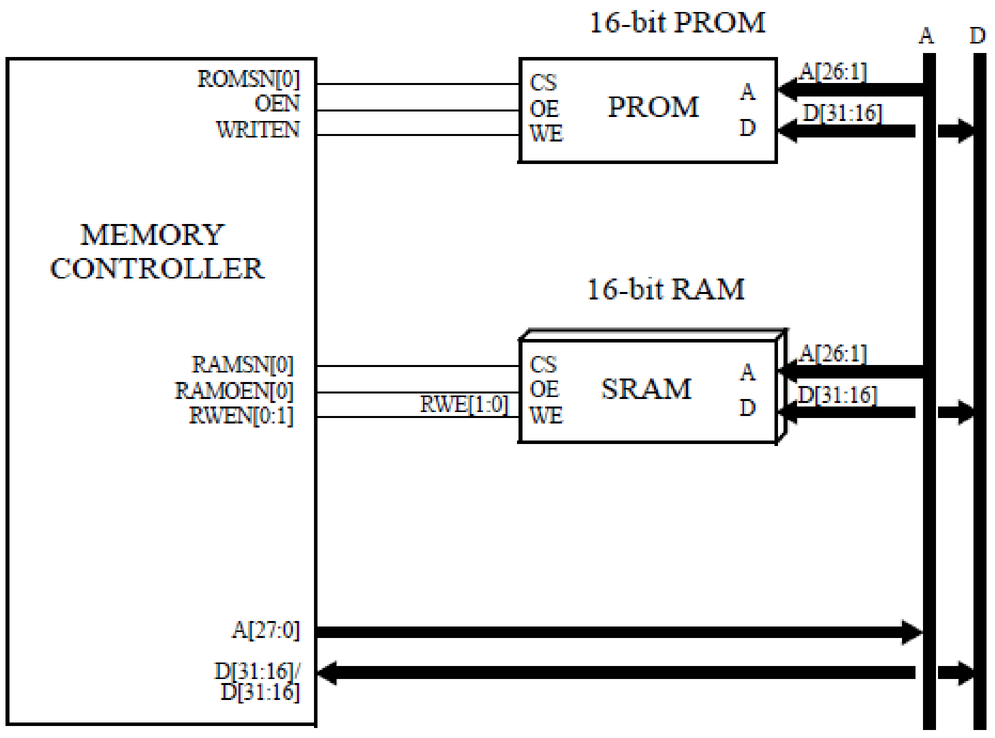


Figure 3.3: 16-bit Memory Interface Example

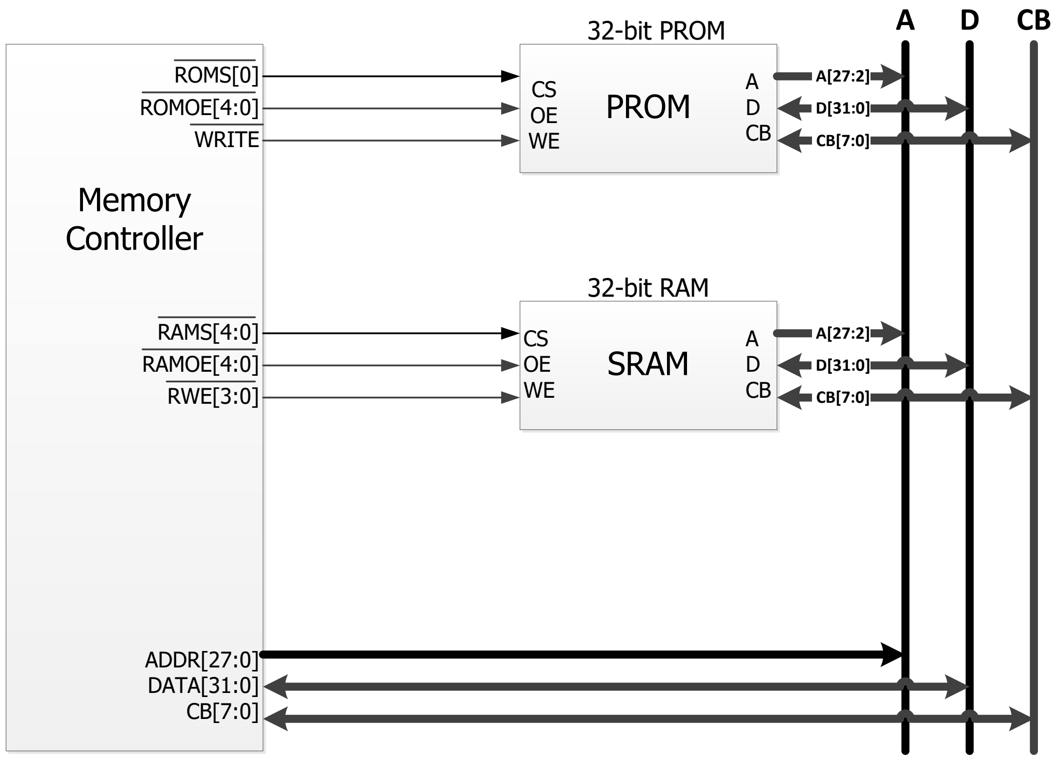


Figure 3.4: 32-bit Memory Interface Example

In 8-bit mode, the PROM/SRAM devices should be connected to the data bus DATA[31:24]. The LSB address bus should be used for addressing (ADDR[25:0]). In 16-bit mode, DATA[31:16] should be used as data bus and ADDR[26:1] as address bus. EDAC protection is not available in 16-bit mode.

3.5 8-bit and 16-bit I/O Accesses

Similar to the PROM/SRAM areas, the I/O area can also be configured to 8-bit or 16-bit mode. However, the I/O device will not be accessed by multiple 8/16 bit accesses as the memory areas, but only with one single access just as in 32-bit mode. To access an I/O device on an 8-bit bus, LDUB/STB instructions should be used. To access an I/O device on a 16-bit bus, LDUH/STH instructions should be used.

3.6 Burst Cycles

To improve the bandwidth of the memory bus, accesses to consecutive addresses can be performed in burst mode. Burst transfers will be generated when the memory controller is accessed using an AHB burst request. These include instruction cache-line fills, double loads and double stores. The timing of a burst cycle is identical to the programmed basic cycle with the exception that during read cycles, the lead-out cycle will only occur after the last transfer. Burst cycles will not be generated to the I/O area.

3.7 SDRAM Access

3.8.1 General

Synchronous Dynamic RAM (SDRAM) access is supported to two banks of PC100/PC133 compatible devices. The SDRAM controller supports 64 MB, 256 MB and 512 MB devices with 8-12 column-address bits and up to 13 row-address bits. The size of the two banks can be programmed in binary steps between 4 Mbyte and 512 Mbyte. The SDRAM controller operation is controlled through MCFG2 and MCFG3. A 32-bit data bus width is supported that can interface 64-bit DIMM modules. The memory controller can be configured to use either a shared or separate bus connecting the controller and SDRAM devices. See Section 3.17 for timing diagram examples of SDRAM accesses.

3.8.2 Address Mapping

Two SDRAM chip-select signals are used for address decoding. SDRAM area is mapped into the upper half of the RAM area. When the SDRAM enable bit is set in MCFG2, the controller is enabled and mapped into upper half of the RAM area, as long as the SRAM disable bit is not set. If the SRAM disable bit is set, all access to SRAM is disabled and the SDRAM banks are mapped into the lower half of the RAM area.

3.8.3 Initialization

When the SDRAM controller is enabled, it automatically performs the SDRAM initialization sequence of one PRECHARGE command, eight AUTO-REFRESH command and LOAD-MODE-REG command on both banks simultaneously. The controller programs the SDRAM to use page burst on read and single location access on write.

3.8.4 Configurable SDRAM Timing Parameters

To provide optimum access cycles for different SDRAM devices (and at different frequencies), three SDRAM parameters can be programmed via MCGF2: TCAS, TRP and TRFC. The value of these fields affect the SDRAM timing as described in **Error! Reference source not found.**.

If the TCAS, TRP and TRFC are programmed such that the PC100/133 specifications are fulfilled, the remaining SDRAM timing parameters will also be met. The

Table 3.1 below shows the typical settings for 100 and 133 MHz operation and the resulting SDRAM timing (in ns).

Table 3.1: SDRAM Programmable Minimum Timing Parameters

SDRAM SETTINGS	t_{CAS}	t_{RC}	t_{RP}	t_{RFC}	t_{RAS}
100 MHz, CL=2; TRP=0, TCAS=0, TRFC=4	20	80	20	70	50
100 MHz, CL=3; TRP=0, TCAS=1, TRFC=4	30	80	20	70	50
133 MHz, CL=2; TRP=1, TCAS=0, TRFC=6	15	82	22	67	52
133 MHz, CL=3; TRP=1, TCAS=1, TRFC=6	22	82	22	67	52

3.9 Refresh

The SDRAM controller contains a refresh function that periodically issues an AUTO-REFRESH command to both SDRAM banks. The period between the commands (in clock periods) is programmed in the refresh counter reload field in the MCFG3 register. Depending on SDRAM type, the required period is typically 7.8 or 15.6ms (corresponding to 780 or 1560 clocks at 100 MHz). The generated refresh period is calculated as follows:

refresh period = ((reload value) + 1) / sysclk

The refresh function is enabled by setting bit 31 in MCFG2.

3.9.1 SDRAM Commands

The controller can issue three SDRAM commands by writing to the SDRAM command field in MCFG2: PRE-CHARGE, AUTO-REFRESH and LOAD-MODE-REG (LMR). If the LMR command is issued, the CAS delay as programmed in MCFG2 will be used and remaining fields are fixed: page read burst, single location write, and sequential burst. The command field clears after a command has been executed. When changing the value of the CAS delay, a LOAD-MODE-REGISTER command should be generated at the same time.

Note: When issuing SDRAM commands, the SDRAM refresh must be disabled.

3.9.2 Read Cycles

A read cycle is started by performing an ACTIVATE command to the desired bank and row, followed by a READ command after the programmed CAS delay. A read burst is performed if a burst access has been requested on the AHB bus. The read cycle is terminated with a PRE-CHARGE command; no banks are left open between two accesses.

3.9.3 Write Cycles

Write cycles are performed similarly to read cycles, but WRITE commands are issued after activation. A write burst on the AHB bus generates a burst of write commands without idle cycles in between.

3.9.4 Address Bus

The memory controller uses a common address bus for PROM, I/O, SRAM and SDRAM. SDRAM connected to the address bus should use ADDR[14:2] for the row select and ADDR[16:15] for the bank select.

3.9.5 Data Bus

The memory controller uses a common address bus for PROM, I/O, SRAM and SDRAM. The SDRAM uses a 32-bit data bus and can access a 64-bit SDRAM at half the data capacity.

3.9.6 Clocking

The SDRAM memory is clocked by the SDCLK output. This output is in phase with the internal system clock and provides the maximum margin for setup and hold on the external signals. The SDCLK output will be available as long as the system clock is operating.

3.9.7 Initialization

Each time the SDRAM is enabled (by setting bit 14 in MCFG2), an SDRAM initialization sequence will be sent to both SDRAM banks. The sequence consists of one PRECHARGE command, eight AUTO-REFRESH commands and one LOAD-COMMAND-REGISTER command.

3.10 Memory EDAC

The FTMCTRL is provided with an error detection and correction (EDAC) controller that can correct one-bit-error and detect two-bit-errors in a 32-bit word. For each word, a 7-bit checksum is generated according to the equations below. A correctable error will be handled transparently by the memory controller, but will add one waitstate to the access. If an un-correctable error (double-error) is detected, the current AHB cycle will end with an error response. The EDAC can be used during access to PROM, SRAM, and SDRAM areas by setting the corresponding EDAC enable bits in the MCFG3 register. The equations below show how the EDAC checkbits are generated:

CB0 = D0 ^ D4 ^ D6 ^ D7 ^ D8 ^ D9 ^ D11 ^ D14 ^ D17 ^ D18 ^ D19 ^ D21 ^ D26 ^ D28 ^
D29 ^ D31
CB1 = D0 ^ D1 ^ D2 ^ D4 ^ D6 ^ D8 ^ D10 ^ D12 ^ D16 ^ D17 ^ D18 ^ D20 ^ D22 ^ D24 ^

D26 ^ D28
$\overline{\mathrm{CB2}}$ = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
$\overline{\mathrm{CB3}}$ = D0 ^ D1 ^ D5 ^ D6 ^ D7 ^ D11 ^ D12 ^ D13 ^ D16 ^ D17 ^ D21 ^ D22 ^ D23 ^ D27 ^ D28 ^ D29
CB4 = D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D14 ^ D15 ^ D18 ^ D19 ^ D20 ^ D21 ^ D22 ^ D23 ^ D30 ^ D31
CB5 = D8 ^ D9 ^ D10 ^ D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31
CB6 = D0 ^ D1 ^ D2 ^ D3 ^ D4 ^ D5 ^ D6 ^ D7 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31

If the memory is configured in 8-bit mode, the EDAC checkbit bus (CB[7:0]) is not used, but it is still possible to use EDAC protection. This is done by allocating the top 20% of the memory bank to the EDAC checksums. If the EDAC is enabled, a read access reads the data bytes from the nominal address and the EDAC checksum from the top part of the bank. A write cycle is performed the same way in SRAM memory region. In this way, 80% of the bank memory is available as program or data memory while the top 20% is used for check bits. The size of the memory bank is determined from the settings in MCFG1 and MCFG2. The EDAC cannot be used on memory areas configured in 16-bit mode.

The operation of the EDAC can be tested trough the MCFG3 register. If the WB (write bypass) bit is set, the value in the TCB field replaces the normal checkbits during memory write cycles. If the RB (read bypass) is set, the memory checkbits of the loaded data will be stored in the TCB field during memory read cycles.

Note:

1) When the EDAC is enabled, the RMW bit in memory configuration register 2 must be set.
2) In the PROM memory region, checkbit bytes are written using external tools, MKPROM2 and GRMON2.

3.11 Using $\overline{\mathrm{BRDY}}$

The $\overline{\mathrm{BRDY}}$ signal can be used to stretch access cycles to the PROM and I/O areas and the SRAM area decoded by $\overline{\mathrm{RAMS[4]}}$. The accesses will always have at least the pre-programmed number of waitstates as defined in registers MCFG1 and MCFG2, but will be further stretched until $\overline{\mathrm{BRDY}}$ is asserted. $\overline{\mathrm{BRDY}}$ should be asserted in the cycle preceding the last one. If bit 29 in MCFG1 is set, $\overline{\mathrm{BRDY}}$ can be asserted asynchronously with the system clock. In this case, the read data must be kept stable until the de-assertion of $\overline{\mathrm{OE/RAMOE}}$. $\overline{\mathrm{BRDY}}$ must be asserted for at least 1.5 clock cycles. The use of $\overline{\mathrm{BRDY}}$ can be enabled separately for the PROM, I/O and $\overline{\mathrm{RAMS[4]}}$ areas. It is recommended that $\overline{\mathrm{BRDY}}$ remain asserted until the corresponding chip select signal is de-asserted to ensure that the access has been properly completed and to avoid a datapath stall. See **Section 3.17** for timing diagram examples with $\overline{\mathrm{BRDY}}$.

3.12 Access Errors

An access error can be signaled by asserting the $\overline{\mathrm{BEXC}}$ signal which is sampled with the data. If the usage of $\overline{\mathrm{BEXC}}$ is enabled in register MCFG1, an error response generates on the internal AHB bus. $\overline{\mathrm{BEXC}}$ can be enabled or disabled through register MCFG1 and is active for all areas (PROM, I/O an RAM). For writes, it is sampled on the last rising edge before chip select is de-asserted which is controlled by means of waitstates or bus ready signaling. $\overline{\mathrm{BEXC}}$ is only sampled in the last access for 8-bit mode for RAM and PROM. When four bytes are written for a word access to 8-bit wide memory $\overline{\mathrm{BEXC}}$ is only sampled in the last access with the same timing as a single access in 32-bit mode. See **Section 3.17** for timing diagram examples using $\overline{\text{BEXC}}$.

3.13 Attaching an External DRAM Controller

To attach an external DRAM controller, $\overline{\text{RAMS[4]}}$ should be used since it allows the cycle time to vary through the use of $\overline{\text{BRDY}}$. In this way, delays can be inserted as required for opening of banks and refresh.

3.14 Registers

The core is programmed through registers mapped into APB address space.

Table 3.2: FTMCTRL Memory Controller Registers

REGISTER	APB ADDRESS
Memory configuration register 1 (MCFG1)	0x80000000
Memory configuration register 2 (MCFG2)	0x80000004
Memory configuration register 3 (MCFG3)	0x80000008

3.14.1 Memory Configuration Register 1 (MCFG1)

Memory configuration register 1 is used to program the timing of ROM and I/O accesses.

MCFG 1 **Address = 0x8000_0000**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	PB	AB	ID[1:0]		IB	BE	▨	IW[3:0]				IE	▨	PZ[3:2]	
W															▨	▨
Reset	0	0	0	--		0	0	0	0000				0	0	11	

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PZ[1:0]		▨	▨	PE	▨	PD[1:0]		PW[3:0]				PR[3:0]			
W	▨	▨														
Reset	11		00		0	--	GPIO[1:0]		1111				1111			

Figure 3.5: Fault Status Register

Table 3.3: Description of Memory Configuration Register 1

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	RESERVED	0	
30	PB	0	PROM area bus enable 0: BRDY disabled for PROM area. 1: BRDY enabled for PROM area.
29	AB	0	Asynchronous bus ready 0: BRDY is synchronous relative to system clock. 1: BRDY input can be asserted without relation to the system clock.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
28-27	ID	_	I/O data bus width 00: 8 bits 01: 16 bits 10: 32 bits 11: Not used
26	IB	0	I/O area bus ready enable 0: BRDY disabled for I/O area. 1: BRDY enabled for I/O area.
25	BE	0	Bus error enable 0: BEXC disabled 1: BEXC enabled
24	RESERVED	0	
23-20	IW	0000	Number of waitstates during I/O accesses 0000: 0 waitstates 0001: 1 waitstates 0010: 2 waitstates ... 1111: 15 waitstates
19	IE	0	I/O enable 0: Access to memory mapped I/O space is disabled 1: Access to memory mapped I/O space is enabled
18	RESERVED	0	
17-14	PZ	1111	PROM size is fixed 1111: 256MB
13-12	RESERVED	00	
11	PE	0	PROM write enable 0: PROM write cycles disabled 1: PROM write cycles enabled
10	RESERVED	0	
9-8	PD	GPIO [1:0]	Data width of the PROM area PWIDTH is set to the value GPIO[1:0] following a reset. 00: 8 bits 01: 16 bits 10: 32 bits 11: Not used
7-4	PW	1111	Number of waitstates during PROM write cycles PWS is set to the maximum value of 15 which equals 30 wait-states following a reset. 0000: 0 0001: 2 0010: 4 ... 1111: 30
3-0	PR	1111	Number of waitstates during PROM read cycles PRS is set to the maximum value of 15 which equals 30 waitstates following a reset.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			0000: 0 0001: 2 0010: 4 ... 1111: 30

3.14.2 Memory Configuration Register 2 (MCFG2)

Memory configuration register 2 is used to control the timing of the SRAM and SDRAM.

MCFG2 **Address = 0x8000_0004**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DR	DP	DF[2:0]			DC	DZ[2:0]			DS[1:0]		DD[1:0]		BW	▨	▨
W														▨	▨	▨
Reset	0	1	111			1	000			10		00		0	0	0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	DE	SI	SZ[3:0]				▨	SB	RM	SD[1:0]		SW[1:0]		SR[1:0]	
W	▨							▨								
Reset	0	0	0	--				0	0	-	--		00		00	

Figure 3.6: Memory Configuration Register 2

Table 3.4: Description of Memory Configuration Register 2

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	DR	0	SDRAM refresh 0: SDRAM refresh disable 1: SDRAM refresh enabled
30	DP	1	SDRAM TRP parameter 0: t_{RP} = 2 system clocks 1: t_{RP} = 3 system clocks
29-27	DF	111	SDRAM TRFC parameter t_{RFC} = 3 + DF system clocks
26	DC	1	SDRAM CAS parameter When changed, a LOAD-COMMAND-REGISTER command must be issued at the same time and also sets RAS/CAS delay (tRCD). 0: CAS = 2 cycle delay 1: CAS = 3 cycle delay
25-23	DZ	000	Bank size for SDRAM chip selects defined as 4MB*2^{DZ} 000: 4MB 001: 8MB 010: 16MB ... 111: 512MB

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
22-21	DS	10	SDRAM column size is 4096 when bits [25:23] = 111 Otherwise, the column size is defined as: 00: 256 01: 512 10: 1024 11: 2048
20-19	DD	00	SDRAM command Writing a non-zero value generates an SDRAM command. The field is reset to 00b after command has been executed. 01: PRECHARGE 10: AUTO-REFRESH 11: LOAD-COMMAND-REGISTER
18	BW	0	Memory controller data bus width. 0: 32-bit 1: 64-bit Read=0; Write=don't care.
17-15	RESERVED	00	
14	DE	0	SDRAM enable 0: SDRAM controller disabled 1: SDRAM controller enabled
13	SI	0	SRAM disable If set together with bit 14 the SRAM can be disabled. DE=0 x: SRAM enabled DE=1 0: SRAM enabled 1: SRAM disabled x=don't care
12-9	SZ	--	Size of each SRAM bank as $8*2^{SZ}$ KB 0000: 8KB 0001: 16KB 0010: 32KB ... 1111: 256MB
8	RESERVED	0	
7	SB	0	SRAM area bus ready enable 0: $\overline{\mathrm{BRDY}}$ disabled for SRAM area 1: $\overline{\mathrm{BRDY}}$ enabled for SRAM area
6	RM	--	Enable read-modify-write cycles on sub-word writes to 16- and 32-bit areas with common write strobe (no byte write strobe). 0: Disabled 1: Enabled
5-4	SD	--	Data width of the SRAM area 00: 8 01: 16 1x: 32 x=don't care

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
3-2	SW	00	Number of waitstates during SRAM write cycles 00: 0 01: 1 10: 2 11: 3
1-0	SR	00	Number of waitstates during SRAM read cycles 00: 0 01: 1 10: 2 11: 3

3.14.3 Memory Configuration Register 3 (MCFG3)

MCFG3 contains the reload value for the SDRAM refresh counter and to control and monitor the memory EDAC. It also contains the configuration of the register file EDAC.

MCFG3 **Address = 0x8000_0008**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R				RSE	ME	RLVAL[14:4]										
W																
Reset	000			0	1	[--...-]										

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RLVAL[3:0]				WB	RB	SE	PE	TCB[7:0]							
W																
Reset	--				0	0	--	GPIO[2]	[--...-]							

Figure 3.7: Memory Configuration Register 3

Table 3.5: Description of Memory Configuration Register 3

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-30	RFC	000	Number of checkbits are used for the register file 00: 0 01: 1 10: 2 11: 7 (EDAC)
29-28	RESERVED	--	

27	ME	1	Memory EDAC
			Indicates if memory EDAC is present Read = 1;
			Write = Don't care

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
26-12	RLDVAL	[--...-]	SDRAM refresh counter reload value The period between each AUTO-REFRESH command is calculated as follows: $t_{REFRESH}$ = (RLDVAL + 1) / SYS_CLK
11	WB	0	EDAC diagnostic write bypass 0: Normal operation 1: EDAC write bypassed
10	RB	0	EDAC diagnostic read bypass 0: Normal operation 1: EDAC read bypassed
9	R	-	Enable EDAC checking of the SDRAM or SRAM area 0: EDAC checking of the RAM area disabled 1: EDAC checking of the RAM area enabled
8	PE	GPIO [2]	Enable EDAC checking of the PROM area At reset, this bit is initialized with the value GPIO[2]. 0: EDAC checking of the PROM area disabled 1: EDAC checking of the PROM area enabled
7-0	TCB[7:0]	[--...-]	Test checkbits This field replaces the normal checkbits during store cycles when WB (bit 11) is set. TCB is also loaded with the memory checkbits during load cycles when RB (bit 10) is set.

The period between each AUTO-REFRESH command is calculated as follows:

$$t_{REFRESH} = ((\text{reload value}) + 1) / SYSCLK$$

3.15 Vendor and Device Identifiers

The core has vendor identifier 0x01 (GAISLER) and device identifier 0x054. For description of vendor and device identifiers, see GRLIB IP Library User's Manual at www.gaisler.com/products/grlib/grlib.pdf.

3.16 Boot Strap Configuration

Upon power up or external $\overline{\text{RESET}}$, GPIO[2] is configured as an input and a high logic level seen on this pin when $\overline{\text{RESET}}$ goes high configures the PROM to operate with EDAC enable. Upon power up or external $\overline{\text{RESET}}$, GPIO[1:0] is configured as an input and the following logic level combinations on each pin when $\overline{\text{RESET}}$ goes high to configure the PROM data width:

Table 3.6: Logic Level Combinations

GPIO[1:0]	PROM Data width
00	8-bit
01	16-bit
10	32-bit

3.17 PROM, SRAM, and Memory Mapped I/O Timing Diagrams

This section shows typical timing diagrams for PROM, SRAM and I/O accesses. These timing diagrams are functional, and are intended to show the relationship between control signals and SDCLK. The actual value of the timing parameters can be found in Chapter 4 of the UT699 datasheet.

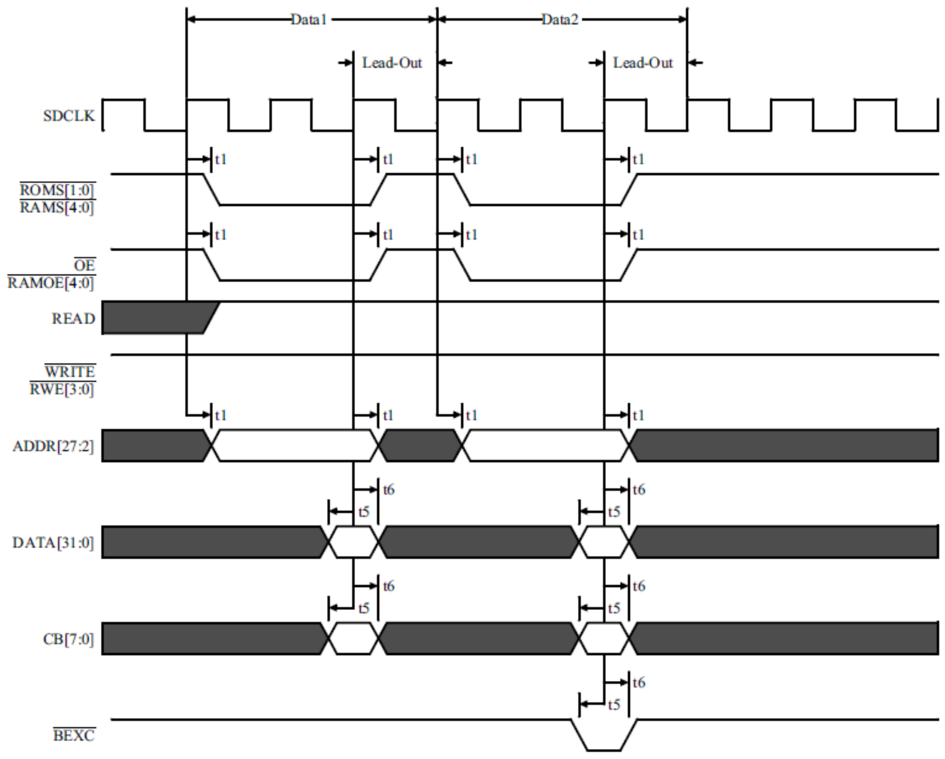

Data1
Data2
Lead-Out
Lead-Out
SDCLK
$\overline{\text{ROMS[1:0]}}$
$\overline{\text{RAMS[4:0]}}$
$\overline{\text{OE}}$
$\overline{\text{RAMOE[4:0]}}$
READ
$\overline{\text{WRITE}}$
$\overline{\text{RWE[3:0]}}$
ADDR[27:2]
DATA[31:0]
CB[7:0]
$\overline{\text{BEXC}}$
t1
t5
t6

Figure 3.8: PROM and SRAM 32-bit Read

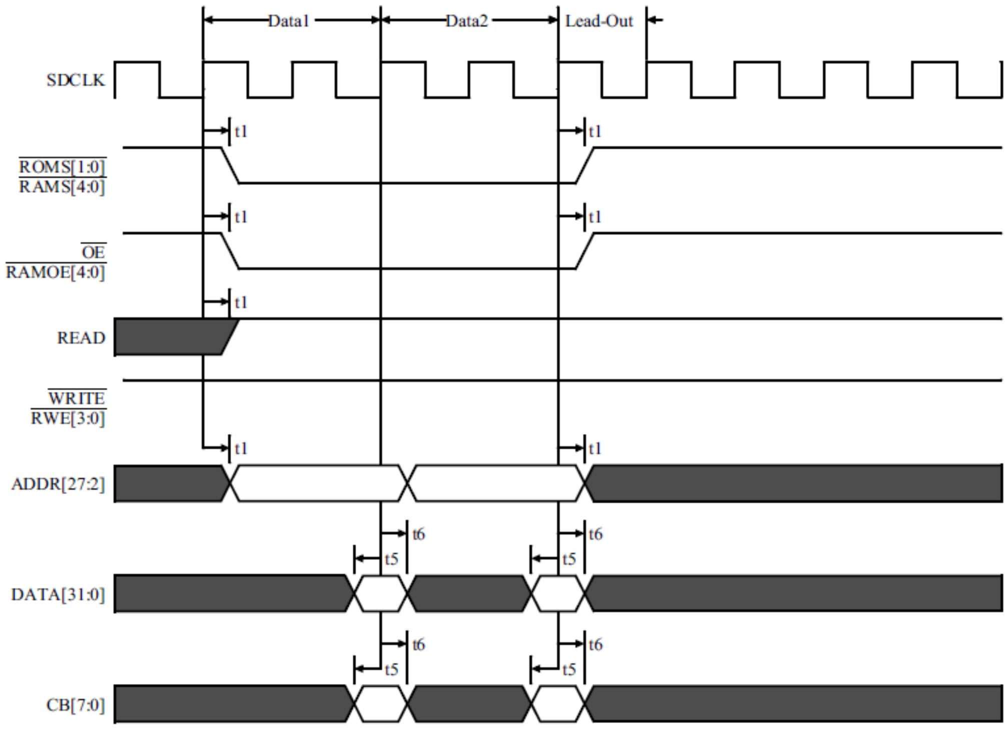


Figure 3.9: PROM and SRAM 32-bit Read Cycle Consecutive

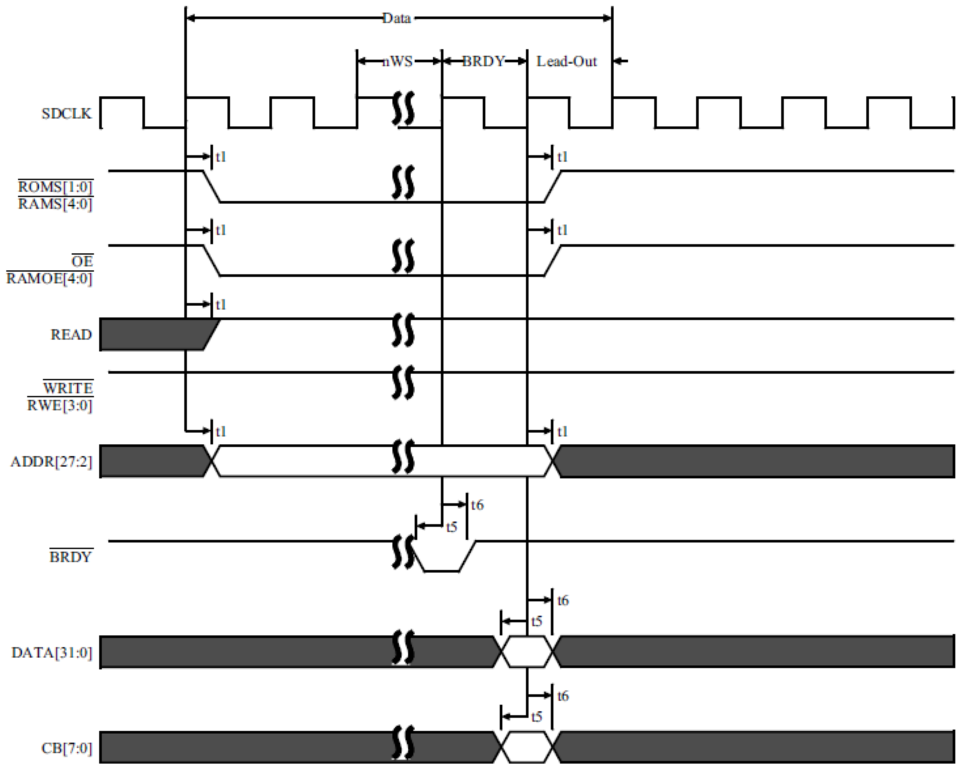


Figure 3.10: PROM and SRAM 32-bit Read Cycle with Waitstates and $\overline{\mathrm{BRDY}}$

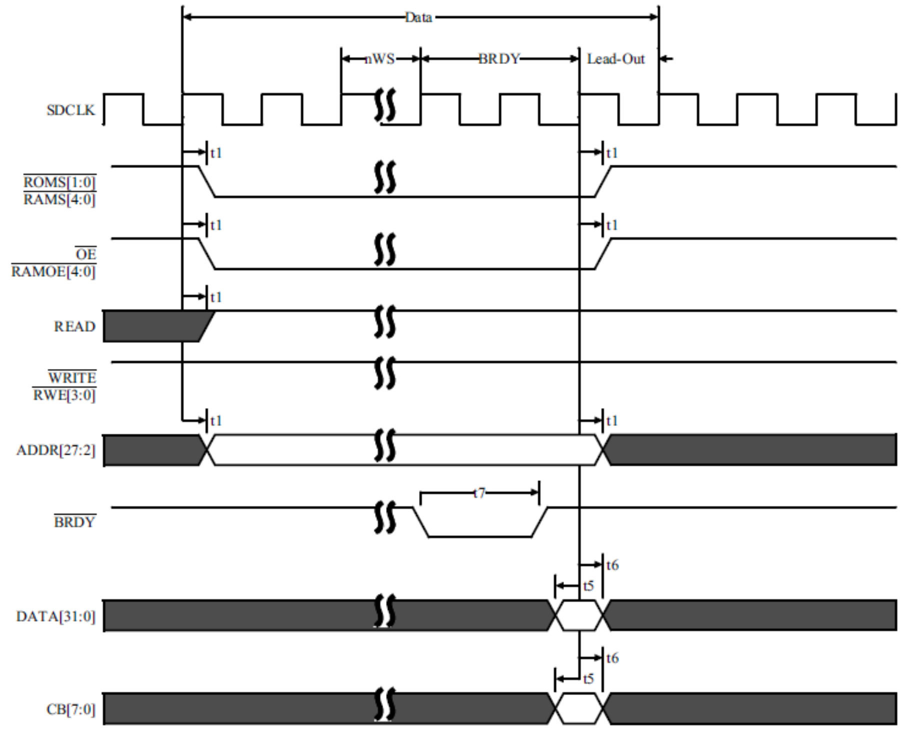


Figure 3.11: PROM and SRAM 32-bit Read Cycle with Waitstates and Asynchronous $\overline{\text{BRDY}}$

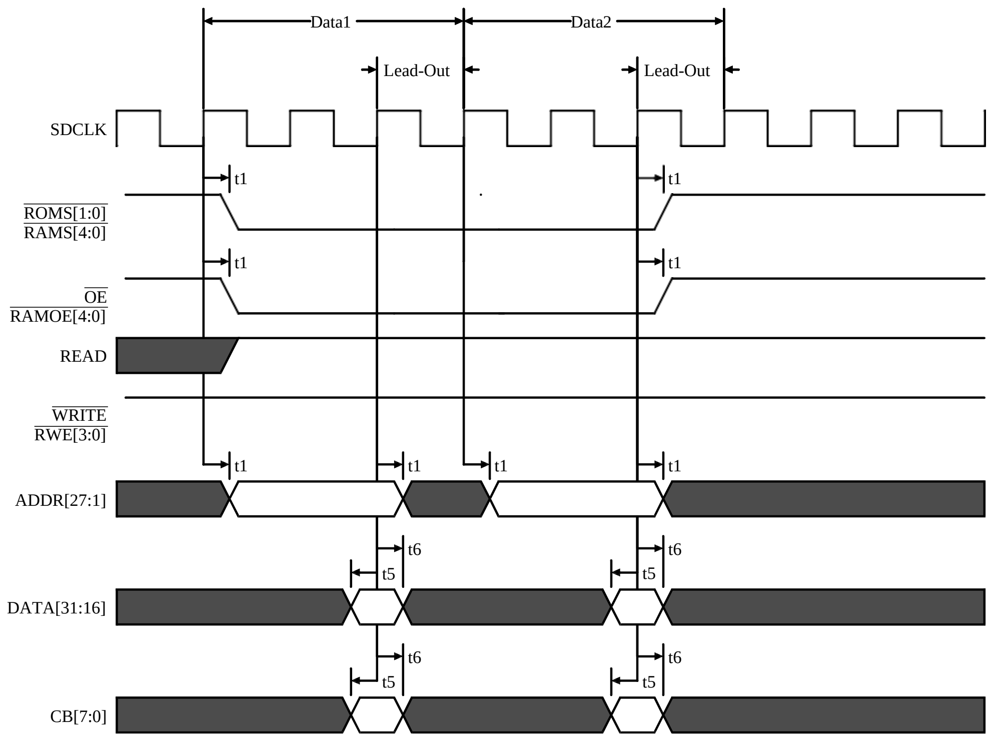


Figure 3.12: PROM and SRAM 16-bit Read Cycle

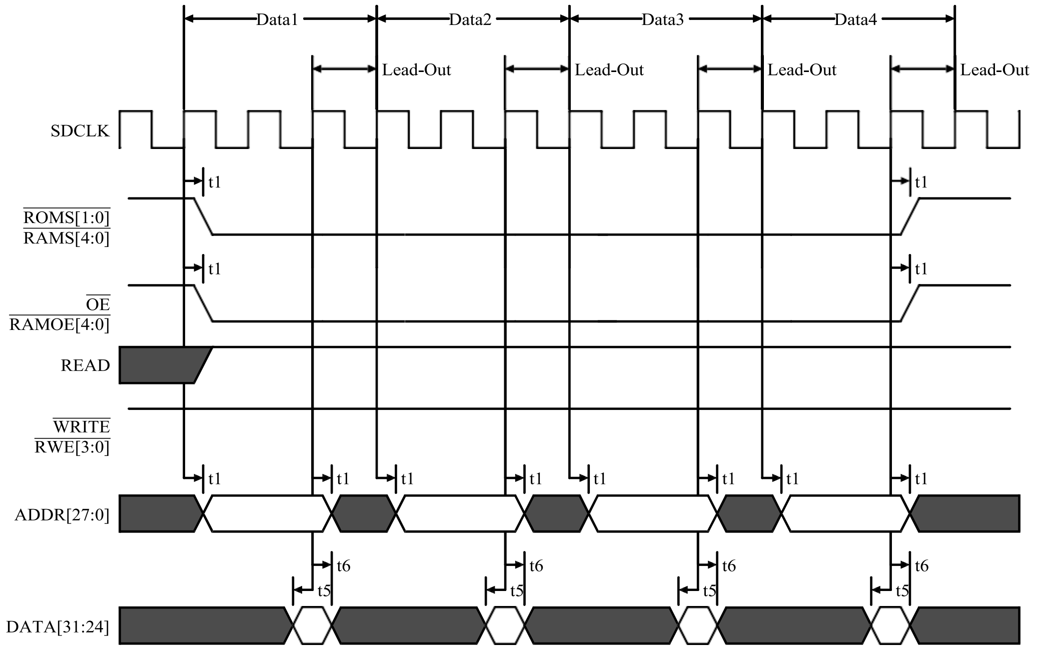


Figure 3.13: PROM and SRAM 8-bit Read Cycle

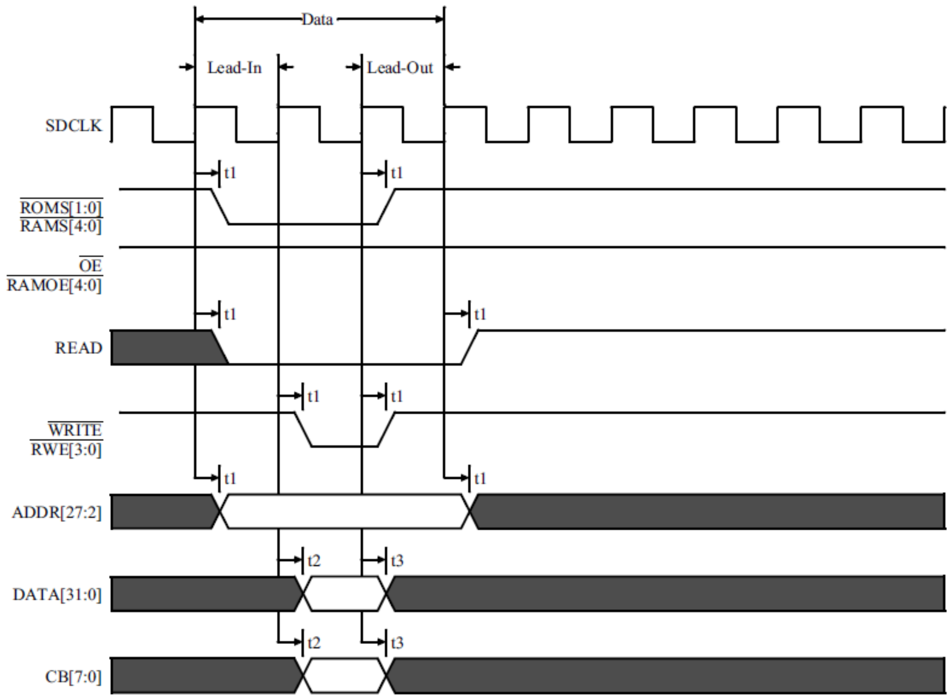


Figure 3.14: PROM and SRAM 32-bit Write Cycle on a 32-bit Bus

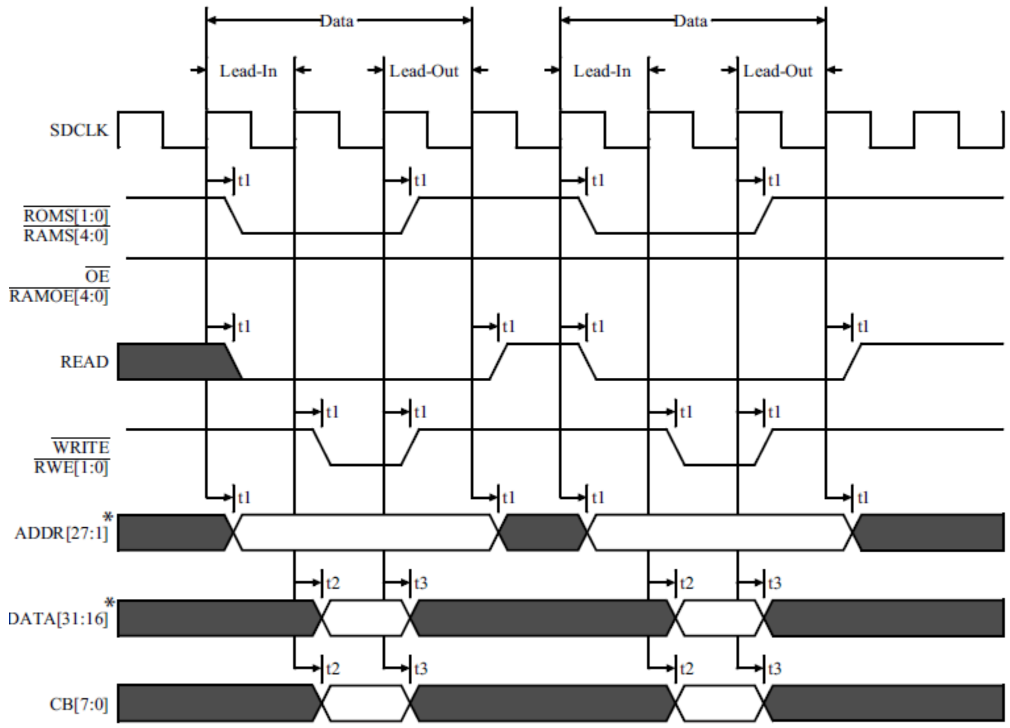


Figure 3.15: PROM and SRAM 16-bit Write Cycle on a 16-bit Bus

***** ADDR[27:0] and DATA[31:24] and RWE[0] are used for 8-bit bus architecture.

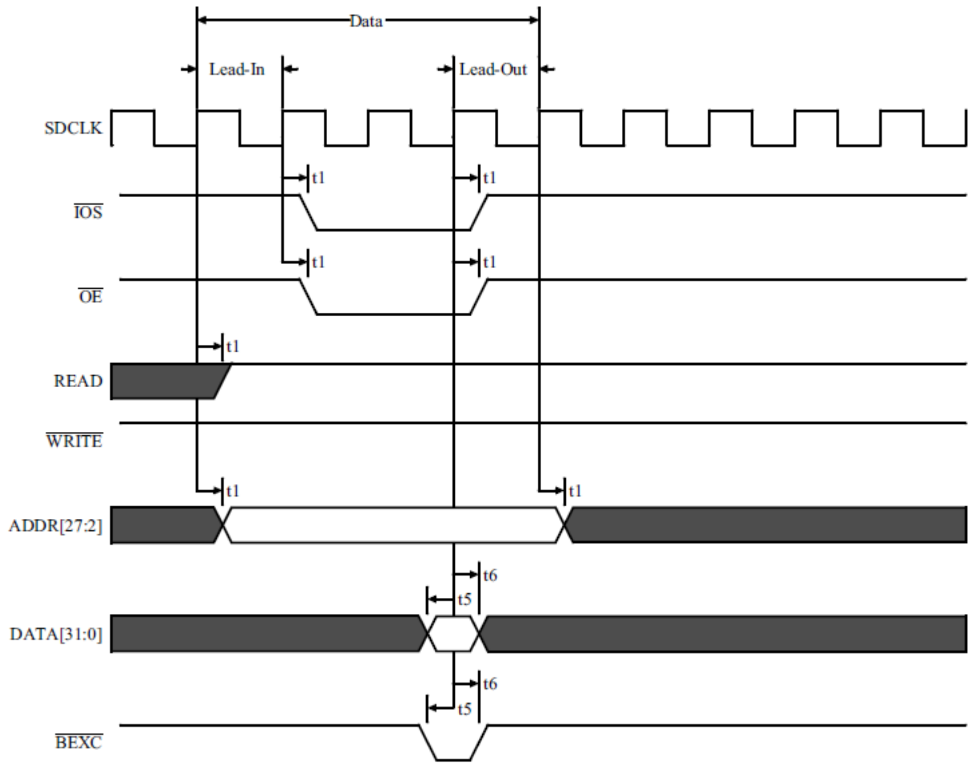


Figure 3.16: Memory Mapped I/O 32-bit Read Cycle

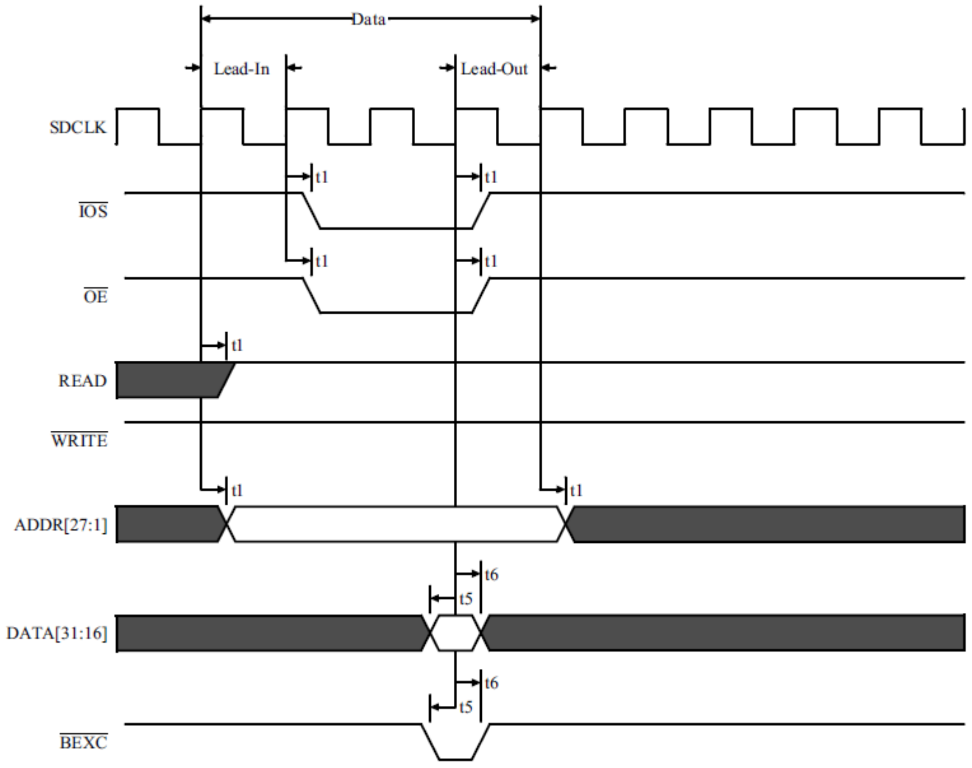


Figure 3.17: Memory Mapped I/O 16-bit Read Cycle

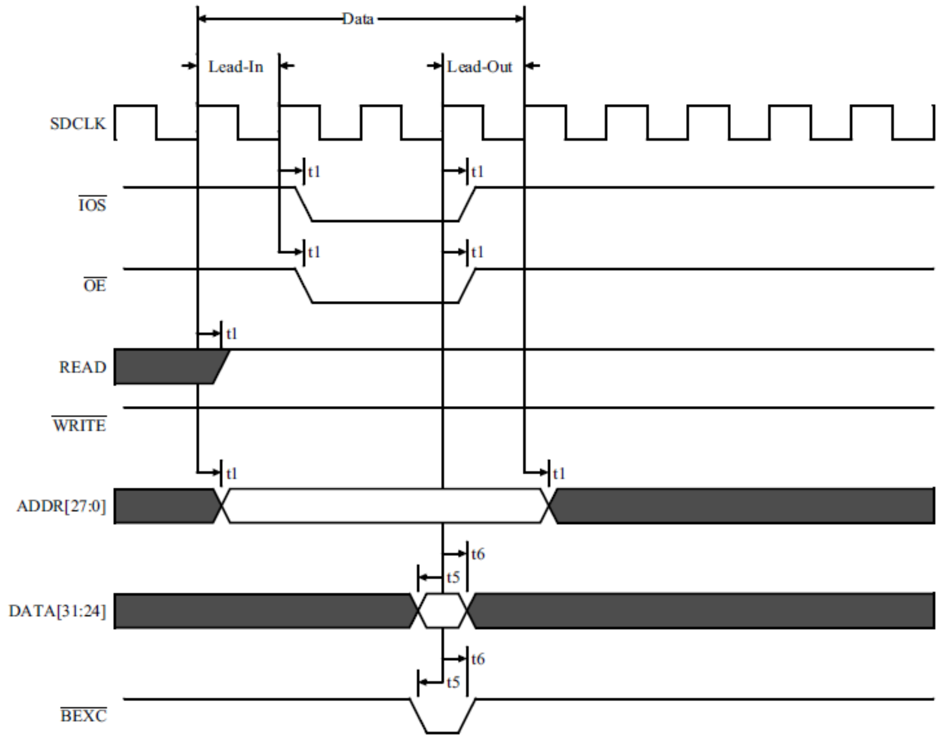


Figure 3.18: Memory Mapped I/O 8-bit Read Cycle

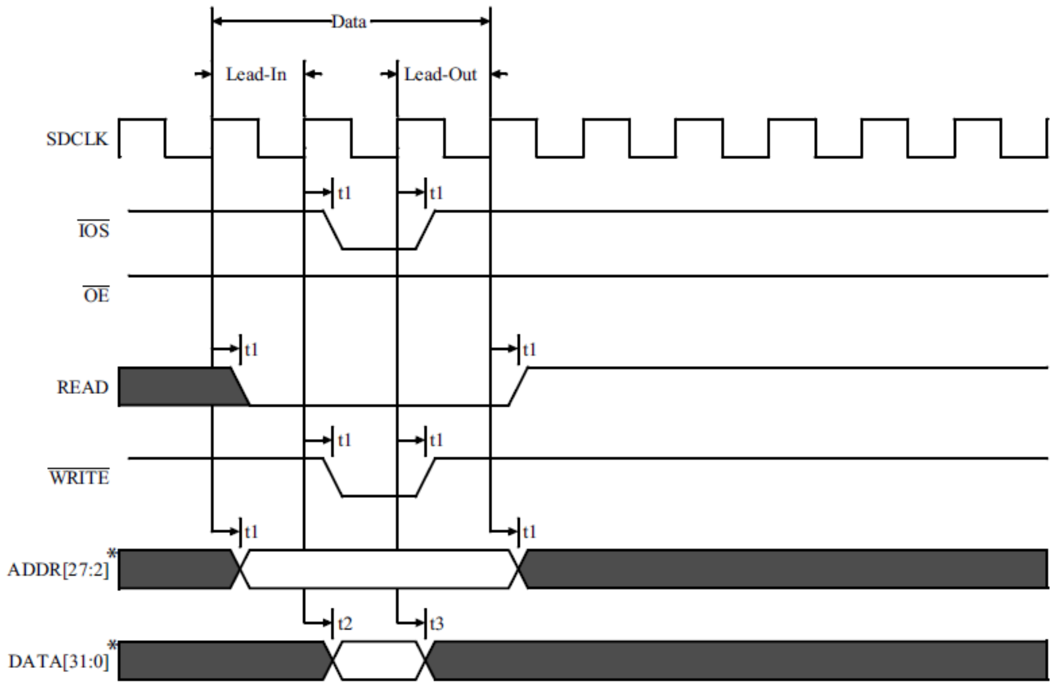


Figure 3.19: Memory Mapped I/O 32-bit Write Cycle

*For 16-Bit I/O bus configurations ADDR[27:1] and Data [31:16] are used. For 8-bit I/O bus configurations ADDR[27:0] and Data [31:24] are used.

3.18 SDRAM Timing Diagram

This section shows typical timing diagrams for PROM, SRAM and I/O accesses. These timing diagrams are functional and are intended to show the relationship between control signals and SDCLK. The actual values of the timing parameters can be found in Chapter 4 of the UT699 datasheet.

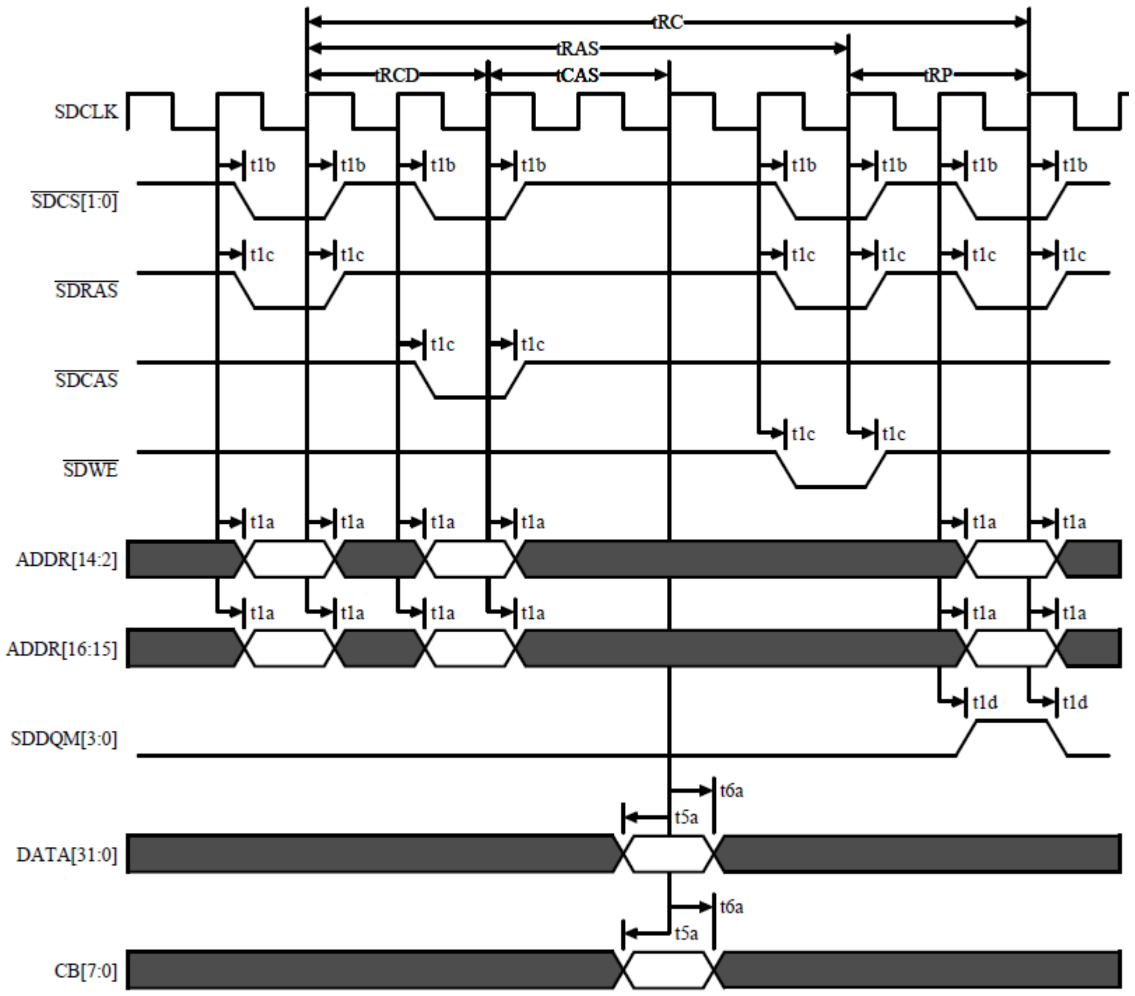


Figure 3.20: SDRAM Read Cycle

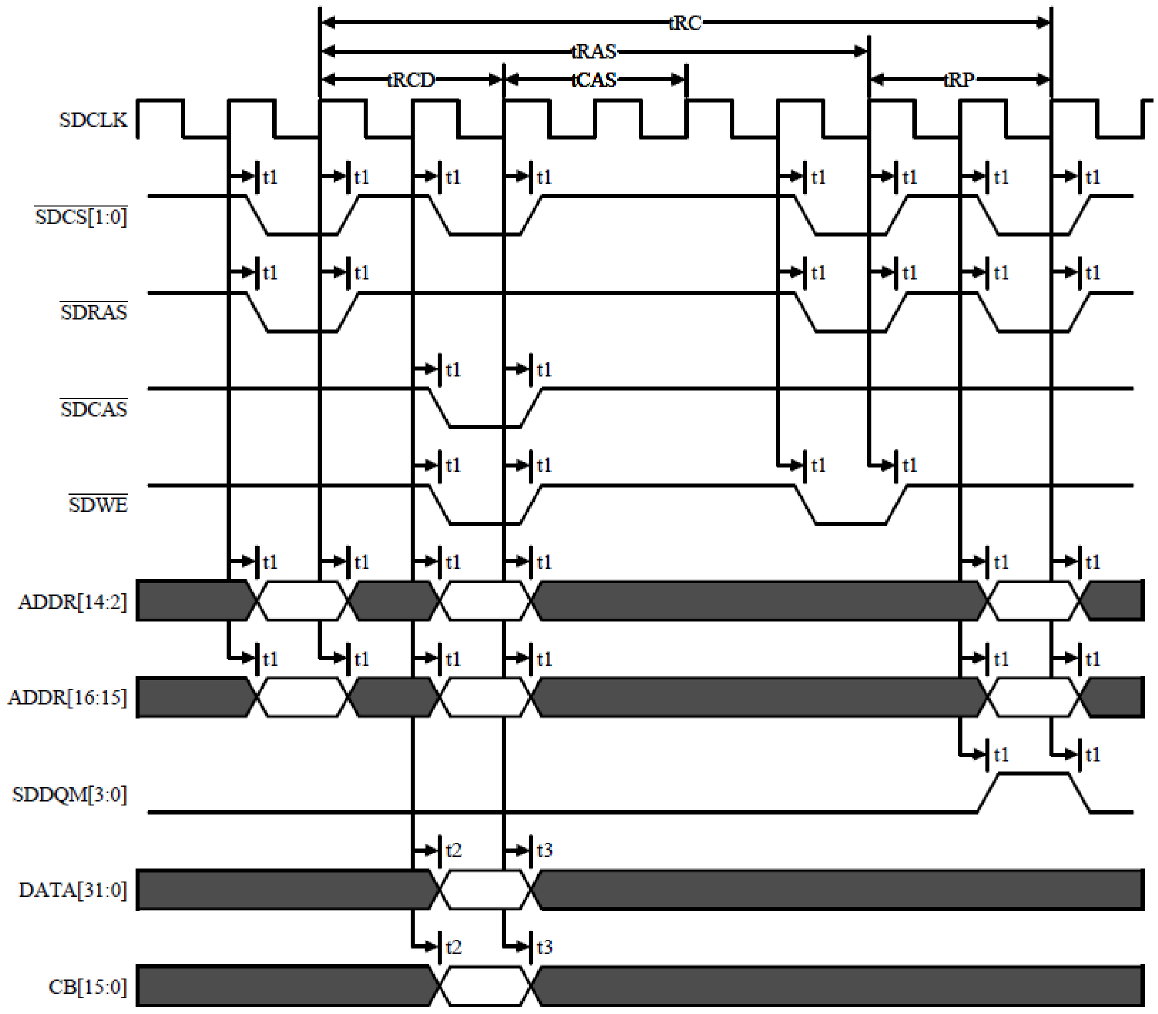


Figure 3.21: SDRAM Write Cycle

Chapter 4: AHB Status Registers

4.1 Overview

The AHB status registers store information about AHB accesses triggering an error response. There is a status register (AHB- STAT) and a failing address register (AHBADDR). Both are contained in a module accessed from the APB bus.

4.2 Operation

The AHB status module monitors AHB bus transactions and stores the current HADDR, HWRITE, HMASTER and HSIZE internally. It is automatically enabled after power-on reset and monitors the AHB bus until an error response (HRESP = "01") is detected. When the error is detected, the status and address register content is frozen and the New Error (NE) bit is set to one. At the same time interrupt 1 is generated. To start monitoring the bus again, the NE bit must be cleared by software.

The status registers are also frozen when the memory controller signals a correctable error even though HRESP is "00" in this case. The software can then scrub the corrected address in order to prevent error build-up and un-correctable multiple errors.

4.3 Registers

Figure 4.1 and **Figure 4.2** show the status register and failing address register. The registers are accessed from the APB bus.

AHBSTAT **Address = 0x8000_0F00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							CE	NE	HW	HM[3:0]				HS[2:0]		
W																
Reset	[--...-]						0	0	0	0000				000		

Figure 4.1: AHB Status Register

Table 4.1: AHB Status Register Description

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	RESERVED	--	
9	CE	0	Correctable error. Set if the memory controller signaled a correctable error.
8	NE	0	New error. Deasserted at start-up and after reset. Asserted when an error is detected. Reset by writing a zero to it.
7	HW	0	The HWRITE signal of the AHB transaction that caused the error.
6-3	HM	0000	The HMASTER signal of the AHB transaction that caused the error.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
2-0	HS	000	The HSIZE signal of the AHB transaction that caused the error.

AHBADDR **Address = 0x8000_0F04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HADDR[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HADDR[15:0]															
W																
Reset	[00…0]															

Figure 4.2: AHB Failing Address Register

Table 4.2: Description of AHB Failing Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	HADDR	[00...0]	The failing address of the AHB transaction that caused the error.

Chapter 5: Interrupt Controller

5.1 Overview

The interrupts generated by on-chip units are all forwarded to the interrupt controller. The controller core then propagates the interrupt with highest priority to the LEON 3FT microprocessor.

5.2 Operation

5.2.1 Interrupt Prioritization

The interrupt controller receives all on-chip interrupts. Each interrupt can be assigned to one of two levels (0 or 1) as programmed in the interrupt level register. Level 1 has higher priority than level 0. The interrupts are prioritized within each level with interrupt 15 having the highest priority and interrupt 1 the lowest. The highest interrupt from level 1 is forwarded to the processor. If no unmasked pending interrupt exists on level 1, then the highest unmasked interrupt from level 0 forwards.

Interrupts are prioritized at system level while masking and forwarding of interrupts is done for each processor separately. Each processor in a multi-processor system has separate interrupt mask and force registers. When an interrupt is signaled on the AMBA bus, the interrupt controller prioritizes interrupts, perform interrupt masking for each processor according to the mask in the corresponding mask register, and forward the interrupts to the processors.

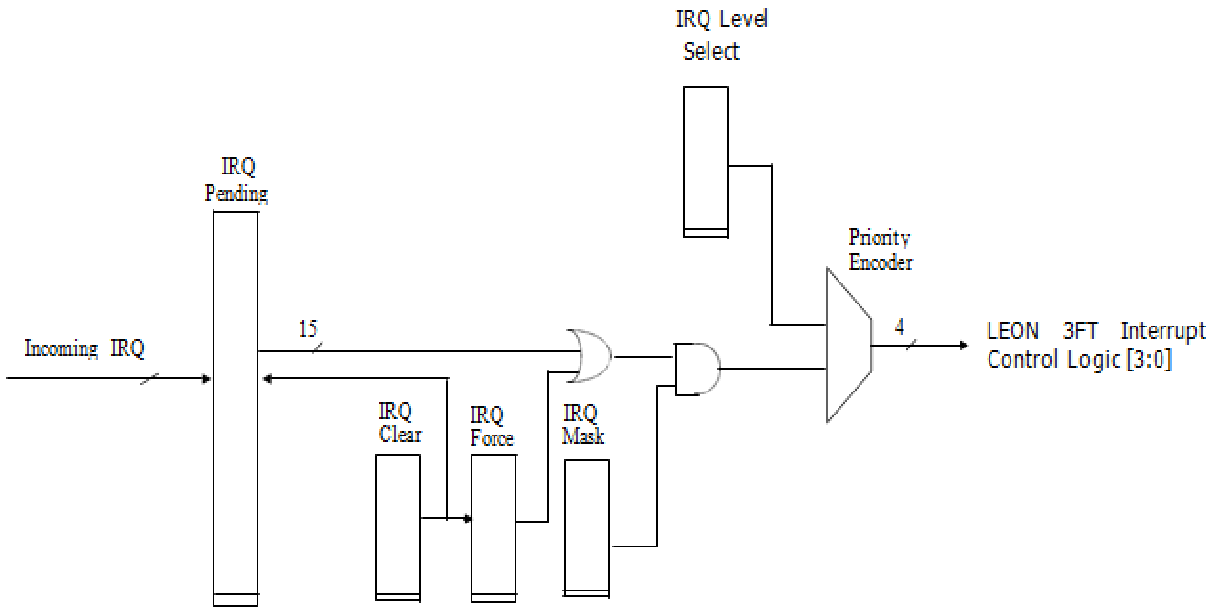


Figure 5.1: Interrupt Controller Block Diagram

When the processor acknowledges the interrupt, the corresponding pending bit automatically clears. Interrupt can also be forced by setting a bit in the interrupt force register. In this case, the processor acknowledgement clears the force bit rather than the pending bit. After reset, the interrupt mask register is set to all zeros while the remaining control registers are undefined.

Note: Interrupt 15 cannot be masked by the LEON 3FT microprocessor and should be used with care as most operating systems do not safely handle this interrupt.

5.2.2 Interrupt Allocation

Table 5.1 indicates the interrupt assignment in the UT699.

Table 5.1: Interrupt Assignment

CORE	INTERRUPT #	FUNCTION
AHBSTAT	1	AHB bus error
APBUART	2	UART1 RX/TX interrupt
GRPCI	3	PCI Error and Abort Interrupts
CAN1	4	CAN1 interrupt
CAN2	5	CAN2 interrupt
GPTIMER	6, 7, 8, 9	Timer underflow interrupts
IRQMP	9	Extended interrupt vector
SPW1	10	SpaceWire1data interrupt
SPW2	11	SpaceWire2 data interrupt
SPW3	12	SpaceWire3 data interrupt
SPW4	13	SpaceWire4data interrupt
ETH	14	Ethernet interrupt
1553B	17	MIL-STD-1553 BC, RT, BM interrupt
SPICTRL	18	SPI controller interrupt
GPIO	1 - 15	External I/O interrupt

5.3 Registers

Table 5.2 shows the Interrupt Controller registers. The base address of the registers is 0x8000_0200.

Table 5.2: IRQ Controller Register

REGISTER	APB Address
Interrupt level register (ILR)	0x80000200
Interrupt pending register (IPR)	0x80000204
Interrupt force register (IFR)	0x80000208
Interrupt clear register (ICR)	0x8000020C
Interrupt status register (ISR)	0x80000210
Interrupt mask register (IMR)	0x80000240

5.3.1 Interrupt Level Register

ILR **Address = 0x8000_0200**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IL[14:0]															▨
W																▨
Reset	[00…0]															0

Figure 5.2: Interrupt Level Register

Table 5.3: Description of Interrupt Level Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	IL	[00...0]	Interrupt level for interrupt IL[n] 0: Interrupt level 0 1: Interrupt level 1
0	RESERVED	0	

5.3.2 Interrupt Pending Register

IPR **Address = 0x8000_0204**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP[14:0]															▨
W																▨
Reset	[00…0]															0

Figure 5.3: Interrupt Pending Register

Table 5.4: Description of Interrupt Pending Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	IP	[00...0]	Interrupt pending for interrupt IP[n] 0: Interrupt not pending 1: Interrupt pending

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
0	RESERVED	0	

5.3.3 Interrupt Force Register

IFR **Address = 0x8000_0208**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IF[14:0]															▨
W																▨
Reset	[00…0]															0

Figure 5.4: Interrupt Force Register

Table 5.5: Description of Interrupt Force Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	IF	[00...0]	Force interrupt IF[n] 0: Normal operation 1: Force interrupt
0	RESERVED	0	

5.3.4 Interrupt Clear Register

ICR **Address = 0x8000_020C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IC[14:0]															▨
W																▨
Reset	[00…0]															0

Figure 5.5: Interrupt Clear Register

Table 5.6: Description of Interrupt Clear Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	IC	[00...0]	Clear interrupt IC[n] 0: Normal operation 1: Clear interrupt
0	RESERVED	0	

5.3.5 Interrupt Status Register

ISR **Address = 0x8000_0210**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NCPU[3:0]				BA	▨							EIRQ[3:0]			
W																
Reset	0000				0	[00…0]							1001			

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STATUS[15:0]															
W																
Reset	[00…0]															

Figure 5.6: Interrupt Status Register

Table 5.7: Description of Interrupt Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-28	NCPU	[00...0]	Number of CPUs in the system -1
27	BA	0	Broadcast Available - set to 1 if NCPU > 0
26-20	RESERVED	[00...0]	
19-16	EIRQ	1001	Extended Interrupts 1-15
15-0	STATUS	[00...0]	Power-down status of CPU[n] (STATUS[n]) - 1 - power-down, 0 - running Write STATUS[n] with a '1' to start processor n

5.3.6 Interrupt Mask Register

IMR **Address = 0x8000_0240**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W																
Reset	0000_0000_0000_0000															
Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R	IM[14:0]	▨
W		
Reset	[00…0]	0

Figure 5.7: Interrupt Mask Register

Table 5.8: Description of Interrupt Mask Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	IM	[00...0]	Interrupt mask for IM[n] 0: Interrupt is masked 1: Interrupt is enabled
0	RESERVED	0	

Chapter 6: **UART with APB Interface**

6.1 Overview

A UART is provided for serial communications. The UART supports data frames with eight data bits, one optional parity bit, and one stop bit. To generate the bit-rate, the UART has a programmable 12-bit clock divider. Two 8-byte FIFOs are used for the data transfers between the bus and UART. **Figure 6.1** shows a block diagram of the UART.

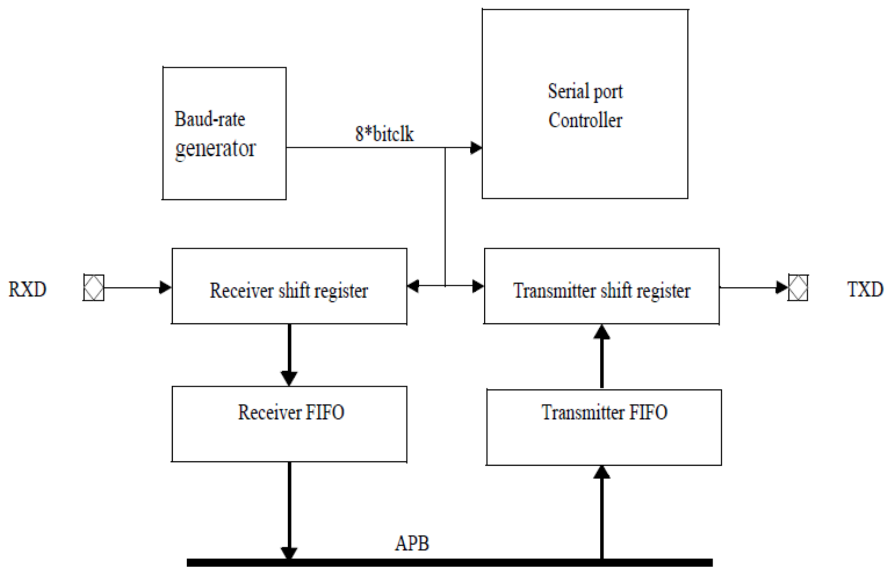
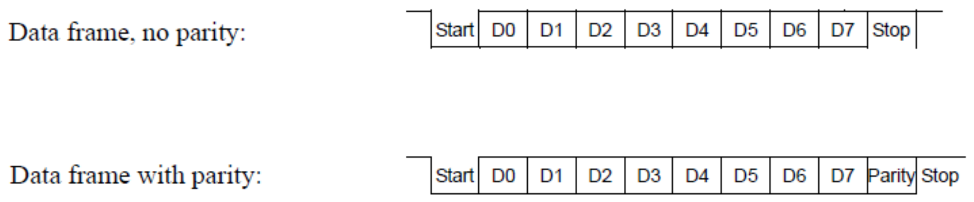


Figure 6.1: APB UART Block Diagram

6.2 Operation

6.2.1 Transmitter Operation

The transmitter is enabled through the TE bit in the UART control register UARTCTR. Data that is to be transferred is stored in the transmitter FIFO by writing to the data register UARTDTR [7:0]. When ready to transmit, data is transferred from the transmitter FIFO to the transmitter shift register and converted to a serial stream on the transmitter serial output pin (TXD). It automatically sends a start bit followed by eight data bits, an optional parity bit, and one stop bit (**Figure 6.2**). The least significant bit of the data is sent first.



Data frame, no parity:
Start D0 D1 D2 D3 D4 D5 D6 D7 Stop
Data frame with parity:
Start D0 D1 D2 D3 D4 D5 D6 D7 Parity Stop

Figure 6.2: UART Data Frames

Following the transmission of the stop bit, if a new character is not available in the transmitter FIFO, the transmitter serial data output remains high and the transmitter shift register empty bit (TS) is set in the UART status register. Transmission resumes and the TS is cleared when a new character is loaded into the transmitter FIFO. When the FIFO is empty, the TE bit is set in the status register UARTSTR. If the transmitter is disabled, it immediately stops any active transmissions including the character currently being shifted out from the transmitter shift register. The transmitter holding register may not be loaded when the transmitter is disabled or when the transmitter FIFO is full. If this is done, data might be overwritten and one or more frames lost.

The TF status bit (not to be confused with the TF control bit) is set if the transmitter FIFO is currently full and the TH bit is set as long as the FIFO is *less* than half-full, i.e. less than half of entries in the FIFO contain data. The TF control bit in the control register UARTCTR enables FIFO interrupts when set. The status register also contains a counter (TCNT) showing the current number of data entries in the transmitter FIFO.

6.2.2 Receiver Operation

The receiver is enabled for data reception through the receiver enable (RE) bit in the UART control register. The receiver looks for a high to low transition of a start bit on the receiver serial data input pin. If a transition is detected, the state of the serial input is sampled a half bit clock later. If the serial input is sampled high, the start bit is invalid and the search for a valid start bit continues. If the serial input is still low, a valid start bit is assumed and the receiver continues to sample the serial input at one bit time intervals (at the theoretical center of the bit) until the proper number of data bits and the parity bit have been assembled and one stop bit has been detected. The serial input is shifted through an 8-bit shift register where all bits need the same value before the new value is taken into account, effectively forming a low-pass filter with a cut-off frequency of 1/8 system clock.

The receiver also has a 8-byte receiver FIFO identical to the transmitter. FIFO data from the receiver FIFO is removed by reading the data register UARTDTR [7:0].

During reception, the least significant bit is received first. The data is then transferred to the receiver FIFO and the data ready (DR) bit is set in the UART status register as soon as the FIFO contains at least one data frame. The parity, framing and overrun error bits are set at the received byte boundary at the same time as the receiver ready bit is set. The data frame is not stored in the FIFO if an error is detected. Also, the new error status bits are OR'd with the old values before they are stored into the status register. Thus, they are not cleared until written to with zeros from the APB bus. If both the receiver FIFO and shift registers are full when a new start bit is detected, then the character held in the receiver shift register is lost and the overrun bit is set in the UART status register.

The RF status bit (not to be confused with the RF control bit) is set when the receiver FIFO is full. The RH status bit is set when the receiver FIFO is half-full (at least half of the entries in the FIFO contain data frames). The RF control bit in the control register UARTCTR enables receiver FIFO interrupts when set. A RCNT field is also available showing the current number of data frames in the FIFO.

6.3 Baud Rate Generation

Each UART contains a 12-bit down-counting scalar to generate the desired baud-rate. The scalar is clocked by the system clock and generates a UART tick each time it underflows. It is reloaded with the value of the UART scalar reload register after each underflow. The resulting UART tick frequency should be eight times the desired baud-rate.

SCALER_RELOAD_VALUE = SYS_CLK / (BAUD_RATE*8)

6.3.1 Loop Back Mode

If the LB bit in the UART control register is set, the UART is in loop back mode. In this mode, the transmitter output is internally connected to the receiver input. It is then possible to perform loop back tests to verify operation of receiver, transmitter and associated software routines. In this mode, the outputs remain in the inactive state, in order to avoid sending out data.

6.3.2 Interrupt Generation

Two different kinds of interrupts are available: normal interrupts and FIFO interrupts. For the transmitter, normal interrupts are generated when transmitter interrupts are enabled (TI), the transmitter is enabled and the transmitter FIFO goes from containing data to being empty. FIFO interrupts are generated when the FIFO interrupts are enabled (TF), transmissions are enabled (TE) and the UART is less than half-full, i.e. whenever the TH status bit is set. This is a level interrupt and the interrupt signal is continuously driven high as long as the condition prevails. The receiver interrupts work in the same way. Normal interrupts are generated in the same manner as for the holding register. FIFO interrupts are generated when receiver FIFO interrupts are enabled, the receiver is enabled and the FIFO is half-full. The interrupt signal is continuously driven high as long as the receiver FIFO is half-full (at least half of the entries contain data frames).

6.4 UART Registers

The APB UART is controlled through four registers mapped into APB address space. UART registers are mapped as follows:

Table 6.1: APB UART Register

REGISTER	APB ADDRESS
UART Data Register (UARTDTR)	0x80000100
UART Status Register (UARTSTR)	0x80000104
UART Control Register (UARTCTR)	0x80000108
UART Scaler Register (UARTSCR)	0x8000010C

6.4.1 UART Data Register

The UART data register provides access to the receiver and transmit FIFO register. The transmitter FIFO is accessed by writing to the data register. The receiver FIFO is accessed by reading the data register.

UARTDTR **Address = 0x8000_0100**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0000_0000_0000_0000															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									DATA[7:0]							
W																
Reset	[00…0]								[00…0]							

Figure 6.3: UART Data Register

Table 6.2: Description of UART Data Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[00...0]	
7-0	DATA	[00...0]	Reading the data register accesses the receiver FIFO. Writing to the data register access the transmitter FIFO.

6.4.2 UART Status Register

The UART Status Register provides information about the state of the FIFO's and error conditions.

UARTSTR **Address = 0x8000_0104**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RCNT[5:0]						TCNT[5:0]									
W																
Reset	[00…0]						[00…0]						[00…0]			

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR
W																
Reset	[00…0]					0	0	0	0	0	0	0	0	1	1	0

Figure 6.4: UART Status Register

Table 6.3: Description of UART Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-26	RCNT	[00...0]	Receiver FIFO Count Shows the number of data frames in the Receiver FIFO.
25-20	TCNT	[00...0]	Transmitter FIFO Count Shows the number of data frames in the Transmitter FIFO.
19-11	RESERVED	[00...0]	
10	RF	0	Receiver FIFO Full Indicates that the Receiver FIFO is full.
9	TF	0	Transmitter FIFO Full Indicates that the Transmitter FIFO is full.
8	RH	0	Receiver FIFO Half Full Indicates that at least half of the FIFO is holding data.
7	TH	0	Transmitter FIFO Half Full Indicates that the FIFO is less than half-full.
6	FE	0	Framing Error Indicates that a framing error was detected.
5	PE	0	Parity Error Indicates that a parity error was detected.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
4	OV	0	Overrun Indicates that one or more characters have been lost due to overrun.
3	BR	0	Break Received Indicates that a BREAK has been received.
2	TE	1	Transmitter FIFO Empty Indicates that the transmitter FIFO is empty.
1	TS	1	Transmitter Shift Register Empty Indicates that the transmitter shift register is empty.
0	DR	0	Data Ready Indicates that new data is available in the receiver holding register.

6.4.3 UART Control Register

The UART Control Register is used to enable the transmitter and receiver and control how interrupts are generated.

UARTCTR **Address = 0x8000_0108**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨					RF	TF	EC	LB	FL	PE	PS	TI	RI	TE	RE
W	▨															
Reset	[00…0]					0	0	0	0	0	0	0	0	0	0	0

Figure 6.5: UART Control Register

Table 6.4: Description of UART Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-11	RESERVED	[00...0]	
10	RF	0	Receiver FIFO Interrupt Enable 0: Receiver FIFO level interrupts disabled 1: Receiver FIFO level interrupts enabled
9	TF	0	Transmitter FIFO Interrupt Enable 0: Transmitter FIFO level interrupts disabled 1: Transmitter FIFO level interrupts enabled
8	EC	0	External Clock (EC) 0: the UART scaler will be clocked by SYSCLK 1: the UART scaler will be clocked by UARTI.EXTCLK
7	LB	0	Loopback Mode 0: Disabled

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Enabled
6	FL	0	Flow control (FL) control using CTS/RTS 0: Disabled 1: Enabled
5	PE	0	Parity Enable 0: Parity generation and checking disabled 1: Parity generation and checking enabled
4	PS	0	Parity Select 0: Even parity 1: Odd parity
3	TI	0	Transmitter Interrupt Enable 0: No frame interrupts 1: Interrupts generated when a frame is transmitted
2	RI	0	Receiver Interrupt Enable 0: No frame interrupts 1: Interrupts generated when a frame is received
1	TE	0	Transmitter Enable 0: Transmitter disabled 1: Transmitter enabled
0	RE	0	Receiver Enable 0: Receiver disabled 1: Receiver enabled

6.4.4 UART Scaler Register

UARTSCR **Address = 0x8000_010C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	SRV[11:0]											
W	▨	▨	▨	▨												
Reset	0000				[00…0]											

Figure 6.6: UART Scalar Register

Table 6.5: Description of UART Scalar Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-12	RESERVED	[00...0]	
11-0	SRV	[00…0]	Scaler Reload Value SRV=SYS_CLK / (BAUD_RATE*8)

Chapter 7: Timer Unit

7.1 Overview

The Timer Unit implements a 12-bit prescaler and four 32-bit decrementing timers. The timer unit registers are accessed through the APB bus. The unit is capable of asserting an interrupt when a timer underflow. A separate interrupt is available for each timer.

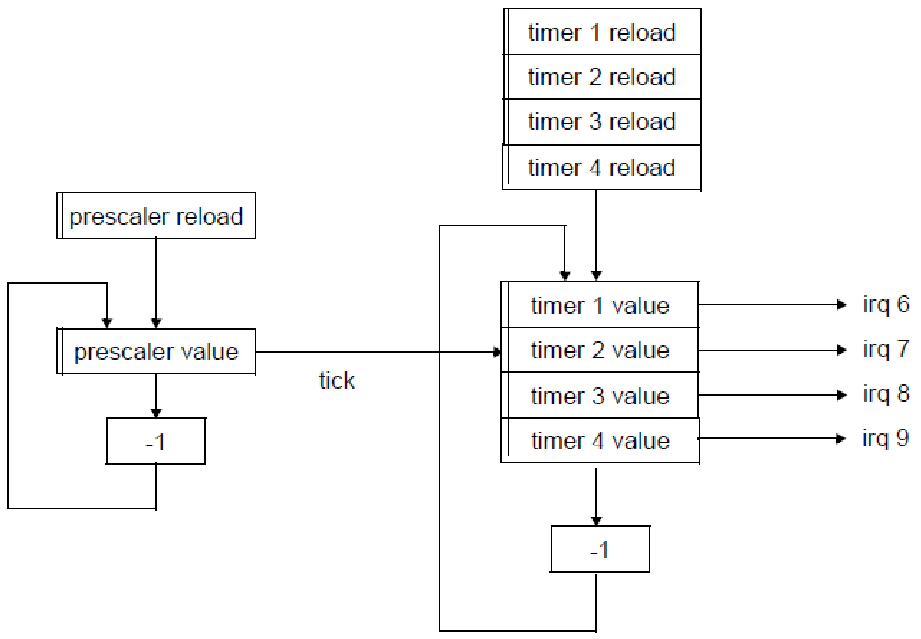


Figure 7.1: Timer Unit Block Diagram

7.2 Operation

The prescaler is clocked by the system clock and decremented on each clock cycle. When the prescaler underflows, it is reloaded from the prescaler reload register and a timer tick is generated. Timers share the decrementer to save area. On the next timer tick, timer n+1 next timer is decremented giving an effective division rate equal to SCALER_RELOAD_VALUE+1.

The operation of each timer is controlled through its control register. A timer is enabled by setting the enable bit (EN) in the control register. The timer value is then decremented on each prescaler tick. When a timer underflow, it will automatically reload with the value from the corresponding timer reload register; if the restart bit (RS) in the control register is set, otherwise it stops at -1 and reset the enable bit.

Each timer signals its interrupt when the timer underflows (if the interrupt enable bit (IE) for the current timer is set). The interrupt pending bit (IP) in the control register of the underflow timer sets and remains set until cleared by writing '1'. Timer 1 generates interrupt 6, timer 2 generates interrupt 7, timer 3 generates interrupt 8, and timer 4 generates interrupt 9.

To minimize complexity, timers share the same decrementer. This means that the minimum allowed prescaler division factor is 5 (SCALER_RELOAD_VALUE=4).

By setting the chain bit (CH) in the control register timer *n* can be chained with preceding timer *n*-1. Decrementing timer *n* will start when timer *n*-1 underflows.

Each timer can be reloaded with the value in it reload register at any time by writing a '1' to the load bit (LD) in the control register. Timer 4 also operates as a watchdog, driving the watchdog output signal ($\overline{\text{WDOG}}$) low when Timer 4 interrupt pending bit is set. The interrupt pending bit is only set when interrupt is enabled for the timer.

7.3 Registers

Table 7.1 shows the timer unit registers.

Table 7.1: General Purpose Timer Unit Register

REGISTER	APB ADDRESS
Scaler Value	0x80000300
Scaler Reload Value	0x80000304
Configuration Register	0x80000308
Timer 1 Counter Value Register	0x80000310
Timer 1 Reload Value Register	0x80000314
Timer 1 Control Register	0x80000318
Timer 2 Counter Value Register	0x80000320
Timer 2 Reload Value Register	0x80000324
Timer 2 Control Register	0x80000328
Timer 3 Counter Value Register	0x80000330
Timer 3 Reload Value Register	0x80000334
Timer 3 Control Register	0x80000338
Timer 4 Counter Value Register	0x80000340
Timer 4 Reload Value Register	0x80000344
Timer 4 Control Register	0x80000348

Figure 7.2 and **Figure 7.3** show the layout of the timer unit registers

TIMSVR **Address = 0x8000_0300**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[11…1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					SCALER_VALUE[11:0]											
W																
Reset	--				[11…1]											

Figure 7.2: Scalar Value Register

TIMSRVR **Address = 0x8000_0304**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[11…1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					SCALER_RELOAD_VALUE[11:0]											
W																
Reset	--				[11…1]											

Figure 7.3: Scaler Reload Value Register

TIMTCR **Address = 0x8000_0308**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							DF	SI						TIMERS		
W																
Reset	[00…0]						0	1	--					100		

Figure 7.4: Timer Configuration Register

Table 7.2: Description of Timer Configuration Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	RESERVED	[00...0]	
9	DF	0	Disable Timer Freeze 0: Timer unit can be frozen during debug mode 1: Timer unit cannot be frozen during debug mode
8	SI	1	Separate Interrupts 0: Single interrupt for timers 1: Each timer generates a separate interrupt Read=1; Write=Don’t care
7-3	RESERVED	--	
2-0	TIMERS	100	Number of Implemented Timers Read=100b; Write=Don’t care

TIMCVR1	**Address = 0x8000_0310**
TIMCVR2	**Address = 0x8000_0320**
TIMCVR3	**Address = 0x8000_0330**
TIMCVR4	**Address = 0x8000_0340**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMER_COUNTER_VALUE[31:16]															
W																
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER_COUNTER_VALUE[15:0]															
W																
Reset	[--...-]															

Figure 7.5: Timer Counter Value Register

Table 7.3: Description of Timer Counter Value Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	Timer Counter Value	[--...-]	Decremented by 1 for each tick, or when timer *n- 1* underflows if in chain mode.

TIMRVR1	**Address = 0x8000_0314**
TIMRVR2	**Address = 0x8000_0324**
TIMRVR3	**Address = 0x8000_0334**
TIMRVR4	**Address = 0x8000_0344**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMER_RELOAD_VALUE[31:16]															
W																
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER_RELOAD_VALUE[15:0]															
W																
Reset	[--...-]															

Figure 7.6: Timer Reload Value Register

Table 7.4: Description of Timer Reload Value Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	Timer Reload Value	[--...-]	This value is loaded into the timer counter value register when '1' is written to bit LD in the timers control register, or when the RS bit is set in the control register and the timer underflows.

Note: Timer resets to value 0x000_FFFF

TIMCTR1	**Address = 0x8000_0318**
TIMCTR2	**Address = 0x8000_0328**
TIMCTR3	**Address = 0x8000_0338**
TIMCTR4	**Address = 0x8000_0348**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										DH	CH	IP	IE	LD	RS	EN
W																
Reset	[00…0]									0	0	0	1*	0	0	1*

Figure 7.7: Timer Control Register

Table 7.5: Description of Timer Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-7	RESERVED	[00...0]	
6	DH	0	Debug Halt State of timer when DF=0. Read only 0: Active 1: Frozen
5	CH	0	Chain with preceding timer 0: Timer functions independently 1: Decrementing timer *n* begins when timer (*n*-1) underflows.
4	IP	0	Interrupt Pending 0: Interrupt not pending 1: Interrupt pending. Remains '1' until cleared by writing '1' to this bit
3	IE	1*	Interrupt Enable (*see EN bit) 0: Interrupts disabled 1: Timer underflow signals interrupt
2	LD	0	Load Timer Writing a '1' to this bit loads the value from the timer reload register to the timer counter value register.
1	RS	0	Restart Writing a '1' to this bit reloads the timer counter value register with the value of the reload register when the timer underflows.
0	EN	1*	Timer Enable *Timer 4 has reset value 0x9 (Watchdog timer is enable on reset) *Timers 1 to 3 have reset values 0. 0: Disable

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Enable

Chapter 8: General Purpose I/O Port

8.1 Overview

A general purpose I/O port is provided using the GRGPIO core from GRLIB. The unit implements a 16-bit I/O port with interrupt support. Each bit in the port is individually set as an input or output and can optionally generate an interrupt. For interrupt generation, the input can be filtered for polarity and level/edge detection. The **Figure 8.1** below shows a diagram for one I/O line.

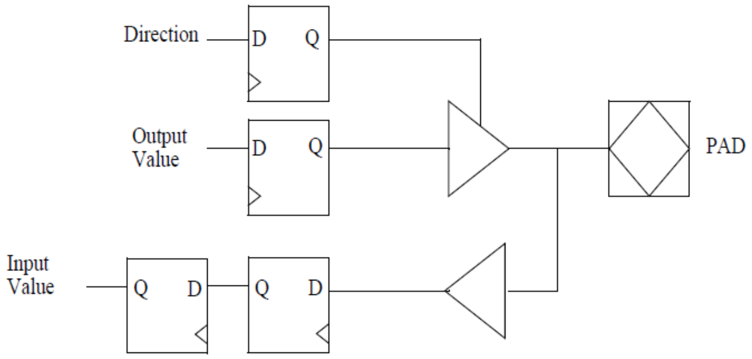


Figure 8.1: General Purpose I/O Port Diagram

8.2 Operation

The I/O ports are implemented as bi-directional buffers with programmable output enable. The input from each buffer is synchronized by two flip-flops in series to remove potential meta-stability. The synchronized values can be read out from the I/O port data register. The output enable is controlled by the I/O port direction register. A '1' in a bit position enables the output buffer for the corresponding I/O line. The output value driven is taken from the I/O port output register.

I/O ports 1-15 drive a separate interrupt line on the APB interrupt bus. The interrupt number is equal to the I/O line index (PIO[1] = interrupt 1, etc.). The interrupt generation is controlled by three registers: interrupt mask, polarity and edge registers. To enable an interrupt, the corresponding bit in the interrupt mask register must be set. If the edge register is '0', the interrupt is treated as level sensitive. If the polarity register is '0', the interrupt is active low. If the polarity register is '1', the interrupt is active high. If the edge register is '1', the interrupt is edge-triggered. The polarity register then selects between rising edge ('1') or falling edge ('0').

8.3 Registers

Table 8.1 shows the I/O port register addresses.

Table 8.1: I/O Port Registers

REGISTER	APB ADDRESS
GPIO Port Data Register (GPIODVR)	0x80000900
GPIO Port Output Register (GPIODOR)	0x80000904
GPIO Port Direction Register (GPIODDR)	0x80000908

REGISTER	APB ADDRESS
Interrupt Mask Register (GPIOIMR)	0x8000090C
Interrupt Polarity Register (GPIOIPR)	0x80000910
Interrupt Edge Register (GPIOIER)	0x80000914

8.3.1 GPIO Port Input Value Register

GPIODVR **Address = 0x8000_9000**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORTVAL[15:0]															
W																
Reset	[00…0]															

Figure 8.2: GPIO Port Value Register

Table 8.2: Description of GPIO Port Value Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-0	PORTVAL	[00...0]	GPIO Port Value This read-only register indicates the state of port pin *n*. 0: Data='0' 1: Data='1'

8.3.2 GPIO Port Data Output Register

GPIODOR **Address = 0x8000_9004**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	PORTOUT[15:0]															
Reset	[00…0]															

Figure 8.3: GPIO Port Data Register

Table 8.3: Description of GPIO Port Data Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-0	PORTOUT	[00...0]	GPIO Port Data The port output register sets the state of port pin *n* when configured as an output. 0: Data='0' 1: Data='1'

8.3.3 GPIO Port Data Direction Register

GPIODDR **Address = 0x8000_9008**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORTDDR[15:0]															
W																
Reset	[00...0]															

Figure 8.4: GPIO Port Data Direction Register

Table 8.4: Description of GPIO Port Data Direction Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-0	PORTDDR	[00...0]	GPIO Data Direction 0: Port pin *n* configured as input 1: Port pin *n* configured as output

8.3.4 GPIO Interrupt Mask Register

GPIOIMR **Address = 0x8000_900C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPIOIMR[15:0]															▨
W																▨
Reset	[00...0]															--

Figure 8.5: GPIO Interrupt Mask Register

Table 8.5: Description of GPIO Interrupt Mask Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	PORTDDR	[00...0]	GPIO Mask Register 0: Interrupt n disabled 1: Interrupt n enabled
0	RESERVED	--	

8.3.5 GPIO Interrupt Priority Register

GPIOIPR **Address = 0x8000_9010**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPIOIPR[15:0]															▨
W																▨
Reset	[00…0]															--

Figure 8.6: GPIO Interrupt Mask Register

Table 8.6: Description of GPIO Interrupt Mask Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-1	GPIOIPR	[00...0]	GPIO Polarity Register This register configures the polarity of interrupt *n*. GPIOIER[*n*]=0 0: Active low 1: Active high GPIOIER[*n*]=1 0: Falling edge 1: Rising edge
0	RESERVED	--	

8.3.6 GPIO Interrupt Edge Register

GPIOIER **Address = 0x8000_9014**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPIOIER[15:0]															▨
W																▨
Reset	[00...0]															--

Figure 8.7: GPIO Interrupt Edge Register

Table 8.7: Description of GPIO Interrupt Edge Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-0	GPIOIER	[00...0]	GPIO Interrupt Edge Register This register configures how an interrupt on port pin *n* is triggered. 0: Level triggered 1: Edge triggered
0	RESERVED	--	

Chapter 9: **PCI Master/Target Unit**

9.1 Overview

The PCI Target/Master Unit is a bridge between the PCI bus and the AMBA AHB bus. The unit is connected to the PCI bus through the PCI Target interface and PCI Master Interface. The AHB Slave and AHB Master interfaces connect the PCI core to the AHB bus. The PCI Configuration and Status registers are accessed via the APB bus.

The PCI and AMBA interfaces belong to two different clock domains. Synchronization is performed inside the core through FIFOs.

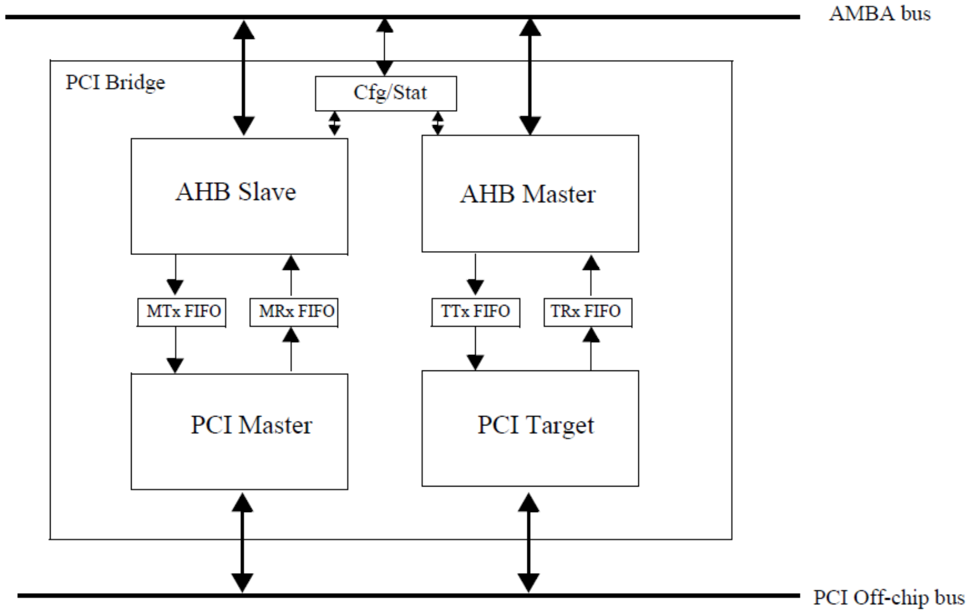


Figure 9.1: PCI Master/Target Unit

9.2 Operation

9.2.1 PCI Target Unit

The PCI target interface and AHB master provide a connection between the PCI bus and the AHB bus. The PCI target is capable of handling configuration and single or burst memory cycles on the PCI bus. Configuration cycles are used to access the Configuration Space Header of the target, while the memory cycles are translated to AHB accesses. The PCI target interface can be programmed to occupy two areas in the PCI address space via registers BAR0 and BAR1 (Section **9.4**). Mapping to the AHB address space is defined by map registers PAGE0 and PAGE1, which are accessible from PCI and AHB address space, respectively.

9.2.2 PCI Master Unit

The PCI master interface occupies one 1GB AHB memory bank and one 128 KB AHB I/O bank. Accesses to the memory area are translated to PCI memory cycles and accesses to the I/O area generate I/O or configuration cycles. Generation of PCI cycles and mapping to the PCI address spaces is controlled through the Configuration/Status Register and the I/O Map Register (Section **9.9**).

9.2.3 Burst Transactions

Both target and master interfaces are capable of burst transactions. Data is buffered internally in FIFO.

9.2.4 Byte Twisting

As PCI is little endian and the AHB controller is big endian, byte twisting is performed on all accesses to preserve the byte ordering as shown in **Figure 9.2**. Byte twisting can be enabled or disabled in the PAGE0 register (Section **9.5**).

Because of byte twisting, byte accesses work correctly. However, 16- and 32-bit PCI accesses need to be byte twisted before being sent to the PCI core.

Note: Accesses between the AHB bus and PCI bus are twisted. Accesses to the configuration space are not byte twisted.

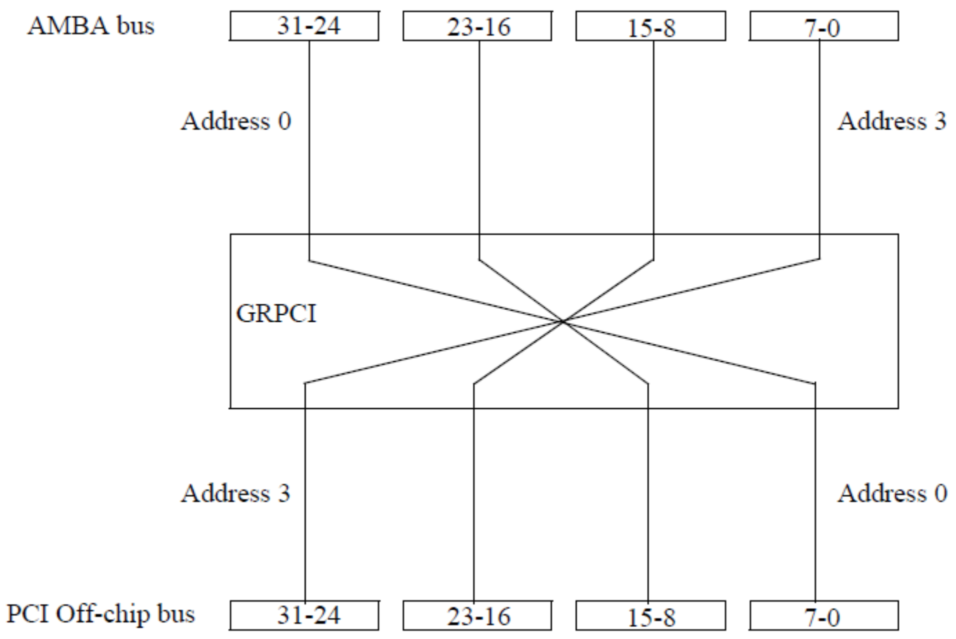


Figure 9.2: GRPCI Byte Twisting

9.3 PCI Target Interface

The PCI target interface occupies two memory areas in the PCI address space as defined by the BAR0 and BAR1 registers in the Configuration Space Header. Register BAR0 maps to a PCI space of 1MB; register BAR1 maps to a PCI space of 64MB.

The PCI Target interface handles the following PCI commands:

- Configuration Read/Write: Single access to the Configuration Space Header. No AHB access is performed
- Memory Read: If prefetching is enabled, the AHB master interface fetches a cache line. Otherwise, a single AHB access is performed
- Memory Read Line: The unit prefetches data according to the value of the Cache Line Size register
- Memory Read Multiple: The unit performs maximum prefetching
- Memory Write
- Memory Write and Invalidate

The target interface supports incremental bursts for PCI memory cycles. The target interface can finish a PCI transaction with one of the following abnormal responses:

- **Retry:** This response indicates that the master should perform the same request later, as the target is temporarily busy. This response is always given at least one time for read accesses, but can also occur for write accesses.
- **Disconnect with data**: Indicates that the target can accept one more data transaction, but no more. This occurs if the master tries to read more data than the target has prefetched.
- **Disconnect without data:** Indicates that the target is unable to accept more data. This occurs if the master tries to write more data than the target can buffer.
- **Target-Abort:** Indicates that the current access caused an internal error and that the target will not be able to complete the access.

The AHB master interface of the target is capable of burst transactions. Burst transactions are performed on the AHB when supported by the destination unit (AHB slave); otherwise, multiple single access is performed. A PCI burst crossing a 1 KB address boundary will be performed as multiple AHB bursts by the AHB master interface. The AHB master interface inserts an idle-cycle before requesting a new AHB burst to allow for re-arbitration of the AHB. AHB transactions with a 'retry' response are repeated by the AHB master until an 'okay' or 'error' response is received. The 'error' response on AHB bus results in a Target-Abort response for the PCI memory read cycle. In the case of a PCI memory write cycle, the AHB access will not finish with an error response since write data is posted to the destination unit. Instead, the WE bit will be set in the Configuration/Status register (APB address 0x80000400).

9.4 PCI Target Configuration Space Header Registers

The registers implemented in the PCI Configuration Space Header are listed in **Table 9.1** and described in this section.

Table 9.1: Configuration Space Header Registers

REGISTER	CONFIGURATION SPACE HEADER ADDRESS OFFSET
Device ID & Vendor ID	0x00
Status & Command	0x04
Class Code & Revision ID	0x08
BIST, Header Type, Latency Timer, Cache Line Size	0x0C
BAR0	0x10
BAR1	0x14

Device/Vendor **Offset = 0x00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEVICE_ID[15:0]															
W																
Reset	0x699															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VENDOR_ID[15:0]															
W																
Reset	0x1AD0															

Figure 9.3: Device ID and Vendor ID Register

Table 9.2: Description of Device ID and Vendor ID Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	DEVICE_ID	0x699	Returns the value of *Device ID*. Read=0x0699; Write=don't care
15-0	VENDOR_ID	0x1AD0	Returns the value of *Vendor ID*. Read=0x1AD0; Write=don't care

PCISTAT **Offset = 0x04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DPE		RMA	RTA	STA	DST[1:0]		DPD								
W																
Reset	0	0	0	0	0	10		0	[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R												MIE		BM	MS	
W																
Reset	[00…0]											0	0	0	0	0

Figure 9.4: Status and Command Register

Table 9.3: Description of Status and Command Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	DPE	0	Detected Parity Error 0: No parity error detected 1: Parity error detected
30	RESERVED	0	
29	RMA	0	Received Master Abort Set by the PCI master interface when its transaction is terminated with Master-Abort. 0: PCI master transaction terminated with no Master-Abort 1: PCI master transaction terminated with Master-Abort
28	RTA	0	Received Target Abort Set by the PCI master interface when its transaction is terminated with Target-Abort. 0: PCI master transaction terminated with no Target-Abort 1: PCI master transaction terminated with Target-Abort
27	STA	0	Signaled Target Abort Set by the PCI target interface when the target terminates transaction with Target-Abort. 0: PCI target terminated with no Target-Abort 1: PCI target terminated with Target-Abort
26-25	DST	10	Device Select Timing Read=10b; Write=don't care.
24	DPD	0	Data Parity Error Detected 0: No data parity error detected

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Data parity error detected
23-5	RESERVED	0	
4	MIE	0	Memory Write and Invalidate Enable Enables the PCI Master interface to generate Memory Write and Invalidate command. 0: Disabled 1: Enabled
3	RESERVED	0	
2	BM	0	Bus Master Enables the Master Interface to generate PCI cycles. 0: Disabled 1: Enabled
1	MS	0	Memory Space Allows the unit to respond to memory space accesses. 0: Disabled 1: Enabled
0	RESERVED	0	

PCICLASS **Offset = 0x08**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLASS_CODE[23:8]															
W	▨															
Reset	0x0B40															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLASS_CODE[7:0]								REVISION_ID[7:0]							
W	▨								▨							
Reset	00								[00…0]							

Figure 9.5: Class Code and Revision Register

Table 9.4: Description of Class Code and Revision Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	CLASS_CODE	0x0B400	Returns the value of *Class Code*. Read=0x0B4000; Write=don't care
7-0	REVISION_ID	0x00	Returns the value of *Revision ID*. Read=0x00; Write=don't care

PCIBHLC **Offset = 0x0C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BIST[7:0]								HEADER[7:0]							
W	▨								▨							

Reset	[00…0]								[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTIM[7:0]								CLS[7:0]							
W																
Reset	[00…0]								[00…0]							

Figure 9.6: BIST, Header Type, Latency Timer and Cache Line Size Register

Table 9.5: Description of BIST, Header Type, Latency Timer and Cache Line Size Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-24	BIST	[00...0]	Built-in Self-Test Not supported. Read=0x00; Write=don't care
23-16	HEADER	[00...0]	Header Type Read=0x00; Write=don't care
15-8	LTIM	[00...0]	Latency Timer Maximum number of PCI clock cycles that the PCI bus master can own the bus.
7-0	CLS	[00...0]	System Cache Line Size Defines the prefetch length for Memory Read and Memory Read Line commands.

Offset = 0x10

PCIBAR0

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASE_ADDRESS_0[10:0]															
W																
Reset	[00…0]											0_0000				

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													PR	TP[1:0]		MS
W																
Reset	[00…0]												0	00		0

Figure 9.7: BAR0 Register

Table 9.6: Description of BAR0 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-21	BASE_ADDRESS _0	[00...0]	PCI Target Interface Base Address 0 A memory area of 2MB is defined at memory location BASE_ADDRESS_0. PCI memory accesses to the lower half of this area are translated to AHB accesses using PAGE0 map register (Section **9.5**).
20-4	RESERVED	[00...0]	This field can be used to determine the memory requirement of the target by writing all '1s' to the BAR0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			register and reading back the value. The device returns ‘0s’ in unimplemented bits positions effectively defining the requested memory area. Read=00...0b; Write=don’t care
3	PR	0	Prefetchable Read=0; Write=don’t care
2-1	TP	00	Type Read=0; Write=don’t care
0	MS	0	Memory Space Indicator Read=0; Write=don’t care

PAGE0 register is mapped into upper half of the PCI address space defined by BAR0 register.

PCIBAR1 **Offset = 0x14**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASE_ADDRESS_1															
W																
Reset	[00…0]						[00…0]									

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													PR	TP[1:0]		MS
W																
Reset	[00…0]												0	00		0

Figure 9.8: BAR1 Register

Table 9.7: Description of BAR1 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-26	BASE_ADDRESS_1	[00...0]	PCI Target Interface Base Address 1. A memory area of 64MB is defined at memory location BASE_ADDRESS_1. PCI memory accesses to this memory space are translated to AHB accesses using PAGE1 map register (Section **9.5**).
25-4	RESERVED	[00...0]	This field can be used to determine the memory requirement of the target by writing all ‘1s’ to the BAR1 register and reading back the value. The device returns ‘0s’ in unimplemented bits positions effectively defining the requested memory area. Read=00...0b; write=don’t care.
3	PR	0	Prefetchable Read=0; Write=don’t care.
2-1	TP	00	Type Read=0; Write=don’t care.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
0	MS	0	Memory Space Indicator Read=0; Write=don't care.

9.5 PCI Target Map Registers

PAGE0 and PAGE1 registers map PCI accesses to AHB address space. PAGE0 is accessed from PCI accesses. PAGE1 can be accessed from the APB.

Table 9.8: PCI Target Map Registers

REGISTER	ADDRESS
PAGE0	First address in the upper half of PCI address space defined by BAR0 register (BAR0 + 0x100000). Accessible only from the PCI address space.
PAGE1	APB address 0x80000410

9.5.1 PAGE0 Register

Register PAGE0 provides a memory mapping between the PCI address space defined by register BAR0 and the AHB address space. PAGE0 is only accessible from a PCI memory access and its location in PCI address space depends upon the value of BAR0. BAR0 provides a mapping of 2 MB between the PCI address space and the PCI target. Only the lower half of the BAR0 space is used. The upper half is unused except for the lowest word, which is the location of the PAGE0 register. This means that the effective address space of BAR0 is 1 MB. Any PCI access to the lower half of the BAR0 address space will map to the AHB address space as defined by PAGE0. The following code example shows how to determine the location of PAGE0 using a PCI memory access, and how to configure PAGE0 to provide a mapping between the PCI address space and the APB address space.

```
/* Read the address of BAR0 */
pci_read_config_dword (bus, slot, function, 0x10, &tmp)

/* Determine the PCI address of PAGE0 */
addr_page0 = tmp + 0x100000;

/* Set PAGE0 to point to start of the APB memory space */
*addr_page0 = (unsigned int *) 0x80000000;
```

In this example, if the address of BAR0 is 0xC0000000, then the address of PAGE0 will be 0xC0100000. A write to PCI address 0xC0000000 translates to an AHB memory access at address 0x80000000.

PCIPAGE0 **Address = 0x8000_0408**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PAGE0_MAP[11:0]												▨			
W																
Reset	[00…0]												0000			

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨															BTEN
W	▨															
Reset	[00…0]															1

Figure 9.9: PAGE0 Register

Table 9.9: Description of PAGE0 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-20	PAGE0_MAP	[00...0]	Maps PCI accesses to the PCI BAR0 address space to the AHB address space. The AHB address is formed by concatenating PAGE0_MAP with PCI address AD[19:0].
19-1	RESERVED	[00...0]	Read=0x0; write=don't care
0	BTEN	1	Byte Twisting Enable May be altered only when bus mastering is disabled. 0: Disabled 1: Enabled

9.5.2 PAGE1 Register

Register PAGE1 provides a memory mapping between the PCI address space defined by register BAR1 and the AHB address space. PAGE1 is accessible directly from the APB bus (APB address 0x80000410), or indirectly from a PCI memory access if PAGE0 is used to map PCI accesses to the APB memory space. BAR1 provides a mapping of 64MB between the PCI address space and the PCI target. PAGE1 can therefore be used to map 64MB of the PCI address space to an equivalent size in the AHB memory space.

PCIPAGE1 **Address = 0x8000_0410**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PAGE1_MAP[5:0]						▨									
W							▨									
Reset	[00…0]						[00…0]									

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨															
W	▨															
Reset	[00…0]															

Figure 9.10: PAGE1 Register

Table 9.10: Description of PAGE1 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-26	PAGE1_MAP	[00...0]	Maps PCI accesses to the PCI BAR1 address space to the

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			AHB address space. The AHB address is formed by concatenating PAGE1_MAP with PCI address AD[25:0].
25-0	RESERVED	[00...0]	Read=0x0; write=don't care

9.6 PCI Master Interface

The PCI master interface occupies 1GB of AHB memory address space and 128 KB of AHB I/O address space. The PCI master interface handles AHB accesses to its back-end AHB Slave interface and translates them to one of the following PCI cycles: PCI configuration cycle, PCI memory cycle, or PCI I/O cycle.

Mapping of the PCI master's AHB address space is configurable through the Configuration/Status Register and I/O Map Register (**Section 9.9**).

9.6.1 PCI Configuration Cycles

Single PCI Configuration cycles are performed by accessing the upper 64 kB of AHB I/O address space allocated by the PCI master's AHB slave starting at address 0xFFF0FFFF. Type 0 configuration cycles are generated by setting the bus number (BUSNO) field to 0 in the Configuration/Status Register. Type 1 configuration cycles are generated by setting the bus number to a value between 1 and 15. Mapping of the configuration cycles is shown in the figure below.

PCICC **Address = 0xFFF0_FFFF**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨												BUSNO[3:0]			
W	▨															
Reset	[00…0]												[00…0]			

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDSEL[4:0]					FUNC[2:0]			REGISTER[5:0]						TP[1:0]	
W																
Reset	[00…0]					000			[00…0]						00	

Figure 9.11: Mapping of AHB I/O Addresses to PCI Address for PCI Configuration Cycles

Table 9.11: Description of Mapping of AHB I/O Addresses to PCI Address for PCI Configuration Cycles

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-20	RESEVERED	[00...0]	AHB Configuration Register Mapping These bits must always be set to 0xFFF1
19-16	BUSNO	[00...0]	Number of the bus that is accessed for Type 1 configuration cycles. This field is ignored for Type 0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			configuration cycles.
15-11	IDSEL	[00...0]	This field is decoded to drive the PCI IDSEL lines during configuration cycles.
10-8	FUNC	000	This field selects the function on a multi-function device.
7-2	REGISTER	[00...0]	This field selects the DWORD register in the con-figuration space.
1-0	TP	00	Must be driven to '00' to generate a Type 0 configuration cycle.

9.6.2 PCI I/O Cycles

PCI I/O cycles are performed by accessing the lower 64 KB of the AHB I/O address space occupied by the master's AHB slave interface are translated into PCI I/O cycles starting at address 0xFFF00000. Mapping is determined by the IOMAP field of I/O Map Register. The IOMAP field of the I/O Map Register maps memory accesses between the PCI master (AHB memory space) and the PCI memory space when performing PCI I/O cycles. The PCI address is formed by concatenating IOMAP with AHB address 15:0. IOMAP provides the 16 most-significant bits of the PCI I/O cycle address.

9.6.3 PCI Memory Cycles

PCI memory cycles are performed by accessing the 1GB AHB address space occupied by the master's AHB slave starting at address 0xC0000000. Mapping and PCI command generation are configured by programming the Configuration/Status Register (APB address 0x80000400). Burst operation is supported for PCI memory cycles. The MMAP field of the Configuration/Status Register maps memory accesses between the PCI master (AHB memory space) and the PCI address space when performing PCI memory cycles. The PCI address is formed by concatenating MMAP with AHB address 29:0. MMAP provides the two most-significant bits of the PCI memory cycle address. PCI commands generated by the master are directly dependent upon the AMBA transfer type and the value of Configuration/Status Register. The Configuration/Status Register can be programmed to issue the following PCI commands: Memory Read, Memory Read Line, Memory Read Multiple, Memory Write, and Memory Write and Invalidate. If an AHB burst access is made to the PCI master's AHB memory space, it is translated to burst PCI memory cycle. When the PCI master interface is busy performing the transaction on the PCI bus, its AHB slave interface will not be able to accept new requests. A 'Retry' response will be given to all accesses to its AHB slave interface. The requesting AHB master repeats its request until an 'OK' or 'Error' response is given by the PCI master's AHB slave interface.

Note: : 'RETRY' responses on the PCI bus are not transparent and will automatically be retried by the master PCI interface until the transfer is either finished or aborted. For burst accesses, only linear-incremental mode is supported and is directly translated from AMBA commands. Byte enables on the PCI bus are translated from the HSIZE control AHB signal and the AHB address according to the table below. Only WORD, HALF-WORD and BYTE values of HSIZE are valid (**Table 9.12**). Byte enables generation

Table 9.12: Byte Enable Generation

HSIZE	PCI_AD[1:0]	PCI_CBE[3:0]
00 (8 bit)	00	1110
00 (8 bit)	01	1101

HSIZE	PCI_AD[1:0]	PCI_CBE[3:0]
00 (8 bit)	10	1011
00 (8 bit)	11	0111
01 (16 bit)	00	1100
01 (16 bit)	10	0011
10 (32 bit)	00	0000

9.7 PCI Host Operation

The PCI core provides a host input signal that must be asserted (active low) for PCI host operation. If this signal is asserted, the bus master interface is automatically enabled and the Bus Master (BM) bit is set in the Status and Command Register. An asserted PCI host signal also enables the PCI target to respond to configuration cycles when the IDSEL signals AD[31:11] are not asserted. This is done in order for the master to be able to configure its own target. For designs intended to operate only as a host or peripheral, this signal can be tied low or high in the design. For multi-purpose designs it should be connected to the appropriate PCI connector pin. The PCI Industrial Computers Manufacturers Group (PICMG) cPCI specification specifies pin C2 on connector P2 for this purpose. The pin should have pull-up resistors as peripheral slots leave it unconnected. PCI interrupts are supported as inputs for PCI hosts.

Note: PCI arbiter is NOT affected by the PCI_HOST input.

9.8 Interrupt Support

Since there is no support for monitoring or driving the 4 PCI interrupt lines, INTA#, INTB#, INTC#, INTD# in the PCI core, these lines should be monitored or driven using the GPIO lines.

9.9 Registers

The core is programmed through registers mapped into APB address space.

Table 9.13: AMBA Register

REGISTER	APB ADDRESS	NOTE
Configuration/Status Register	0x80000400	Read/write access from the APB bus
BAR0 Register	0x80000404	Read-only access from the APB bus Read/write access from the PCI bus
PAGE0 Register	0x80000408	Read-only access from the APB bus Read/write access from the PCI bus
BAR1 Register	0x8000040C	Read-only access from the APB bus Read/write access from the PCI bus
PAGE1 Register	0x80000410	Read/write access from the APB bus
IO Map Register	0x80000414	Read/write access from the APB bus
Status & Command Register	0x80000418	Read-only access from the APB bus Read/write access from the PCI bus

APBCONF **Address = 0x8000_0400**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MMAP[1:0]					BUSNO[3:0]				LTIMER[7:0]						
W																
Reset	00		000			0000				[00…0]						

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		WE	SH	BM	MS	WB	RB	CTO	CLS[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	[00…0]							

Figure 9.12: Configuration and Status Register

Table 9.14: Description of Configuration and Status Register (Read Only)

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-30	MMAP	00	Memory Space Map Register Maps memory accesses between the PCI master (AHB memory space) and PCI address space when performing PCI memory cycles. The PCI address is formed by concatenating MMAP with AHB address 29:0
29-27	RESERVED	[00...0]	
26-23	BUSNO	[00...0]	Number of the bus to access.
22-15	LTIMER	[00...0]	Latency Timer (read only) Value of *Latency Timer* in the Configuration Space Header.
14	WE	0	Target Write Error (read only) 0: No error 1: Write access to target interface resulted in error
13	SH	0	System Host (read only) 0: Unit is not system host 1: Unit is system host
12	BM	0	Bus Master (read only) Value of the *Bus Master* field in the Command register of the Configuration Space Header
11	MS	0	Memory Space (read only) Value of *Memory Space* field in the Command register of the Configuration Space Header
10	WB	0	Write Burst Command Defines the PCI command used for PCI write bursts. 0: Memory Write 1: Memory Write and Invalidate
9	RB	0	Read Burst Command Defines the PCI command used for PCI read bursts. 0: Memory Read Multiple 1: Memory Read Line
8	CTO	0	Configuration Timeout (read only) 0: No timeout occurred during configuration cycle 1: Timeout occurred during configuration cycle

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	CLS	[00...0]	Cache Line Size (read only) Value of Cache Line Size register in Configuration Space Header.

APBBAR0 **Address = 0x8000_0404**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASE_ADDRESS_0[10:0]											▨				
W												▨				
Reset	[00…0]											0_0000				

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨												PR	TP[1:0]		MS
W	▨												▨	▨		▨
Reset	[00…0]												0	00		0

Figure 9.13: Configuration and Status Register

Table 9.15: Description of Configuration and Status Register (Read Only)

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-21	BASE_ADDRESS_0	[00...0]	PCI Target Interface Base Address 0 A memory area of 2MB is defined at memory location BASE_AD- DRESS_0. PCI memory accesses to the lower half of this area are translated to AHB accesses using PAGE0 map register (Section **9.5**).
20-4	RESERVED	[00...0]	This field can be used to determine the memory requirement of the target by writing all '1s' to the BAR0 register and reading back the value. The device returns '0s' in unimplemented bits positions effectively defining the requested memory area. Read=00...0b; write=don't care
3	PR	0	Prefetchable Read=0; Write=don't care
2-1	TP	0	Type Read=0; Write=don't care

0	MS	00	Memory Space Indicator Read=0;
			Write=don't care

APBPAGE0 **Address = 0x8000_0408**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PAGE0_MAP[11:0]												▨			
W													▨			

Reset	[00…0]	0000

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																BTEN
W																
Reset	[00…0]															1

Figure 9.14: PAGE0 Register

Table 9.16: Description of PAGE0 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-20	PAGE0_MAP	[00...0]	Maps PCI accesses to the PCI BAR0 address space to the AHB address space. The AHB address is formed by concatenating PAGE0_MAP with PCI address AD[19:0].
19-1	RESERVED	[00...0]	Read=00...0b; write=don't care
0	BTEN	1	Byte Twisting Enable May be altered only when bus mastering is disabled. 0: Disabled 1: Enabled

APBBAR1 **Address = 0x8000_040C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASE_ADDRESS_1[5:0]															
W																
Reset	[00…0]						[00…0]									

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													PR	TP[1:0]		MS
W																
Reset	[00…0]												0	00		0

Figure 9.15: BAR1 Register

Table 9.17: Description of BAR1 Register (Read Only)

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-26	BASE_ADDRESS_1	[00...0]	PCI Target Interface Base Address 1. A memory area of 64MB is defined at memory location BASE_ADDRESS_1. PCI memory accesses to this memory space are translated to AHB accesses using PAGE1 map register (Section **9.5**).
25-4	RESERVED	[00...0]	This field can be used to determine the memory requirement of the target by writing all '1s' to the BAR1 register and reading back the value. The device returns '0s' in unimplemented bits positions effectively defining the requested memory area.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			Read=00...0b; write=don't care.
3	PR	0	Prefetchable Read=0; Write=don't care.
2-1	TP	00	Type Read=0; Write=don't care.
0	MS	0	Memory Space Indicator Read=0; Write=don't care.

APBPAGE1 **Address = 0x8000_0410**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PAGE1_MAP[5:0]						▨									
W																
Reset	[00…0]						[00…0]									

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨															
W																
Reset	[00…0]															

Figure 9.16: PAGE1 Register

Table 9.18: Description of PAGE1 Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-26	PAGE1_MAP	[00...0]	Maps PCI accesses to the PCI BAR1 address space to the AHB address space. The AHB address is formed by concatenating PAGE1_MAP with PCI address AD[25:0].
25-0	RESERVED	[00...0]	Read=00...0b; write=don't care

APBIOM **Address = 0x8000_0414**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IOMAP[15:0]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨															
W																
Reset	[00…0]															

Figure 9.17: I/O Map Register

Table 9.19: Description of I/O Map Register (Read Only)

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	IOMAP	[00...0]	Maps memory accesses between the PCI master (AHB memory space) and the PCI memory space when performing PCI I/O cycles. The PCI address is formed by concatenating MMAP with AHB address 15:0.
15-0	RESERVED	[00...0]	

APBSTAT **Address = 0x8000_0418**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DPE		RMA	RTA	STA	DST[1:0]		DPD								
W																
Reset	0	0	0	0	0	10		0	[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R												MIE		BM	MS	
W																
Reset	[00…0]											0	0	0	0	0

Figure 9.18: Status and Command Register

Table 9.20: Description of Status and Command Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	DPE	0	Detected Parity Error 0: No parity error detected 1: Parity error detected
30	RESERVED	0	
29	RMA	0	Received Master Abort Set by the PCI master interface when its transaction is terminated with Master-Abort. 0: PCI master transaction terminated with no Master-Abort 1: PCI master transaction terminated with Master-Abort
28	RTA	0	Received Target Abort Set by the PCI master interface when its transaction is terminated with Target-Abort. 0: PCI master transaction terminated with no Target-Abort 1: PCI master transaction terminated with Target-Abort Clear by writing a "1" to this bit, otherwise this bit is read only.
27	STA	0	Signaled Target Abort Set by the PCI target interface when the target terminates transaction with Target-Abort.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			0: PCI target terminated with no Target-Abort 1: PCI target terminated with Target-Abort Clear by writing a "1" to this bit, otherwise this bit is read only.
26-25	DST	10	Device Select Timing Read=10b; Write=don't care.
24	DPD	0	Data Parity Error Detected 0: No data parity error detected 1: Data parity error detected Clear by writing a "1" to this bit, otherwise this bit is read only.
23-5	RESERVED	[[00…0]	
4	MIE	0	Memory Write and Invalidate Enable Enables the PCI Master interface to generate Memory Write and Invalidate command. 0: Disabled 1: Enabled
3	RESERVED	0	
2	BM	0	Bus Master Enables the Master Interface to generate PCI cycles. 0: Disabled 1: Enabled
1	MS	0	Memory Space Allows the unit to respond to memory space accesses. 0: Disabled 1: Enabled
0	RESERVED	0	

9.10 Vendor and Device Identifiers

The core has vendor identifier 0x01 (Cobham's Gaisler) and device identifier 0x014.

Chapter 10: **DMA Controller for the GRPCI Interface**

10.1 Introduction

The DMA controller is an add-on interface to the GRPCI interface. This controller performs bursts to or from the PCI bus using the master interface of the PCI Target/Master unit.

Figure 10.1 illustrates how the DMA controller is attached between the AHB bus and the PCI master interface.

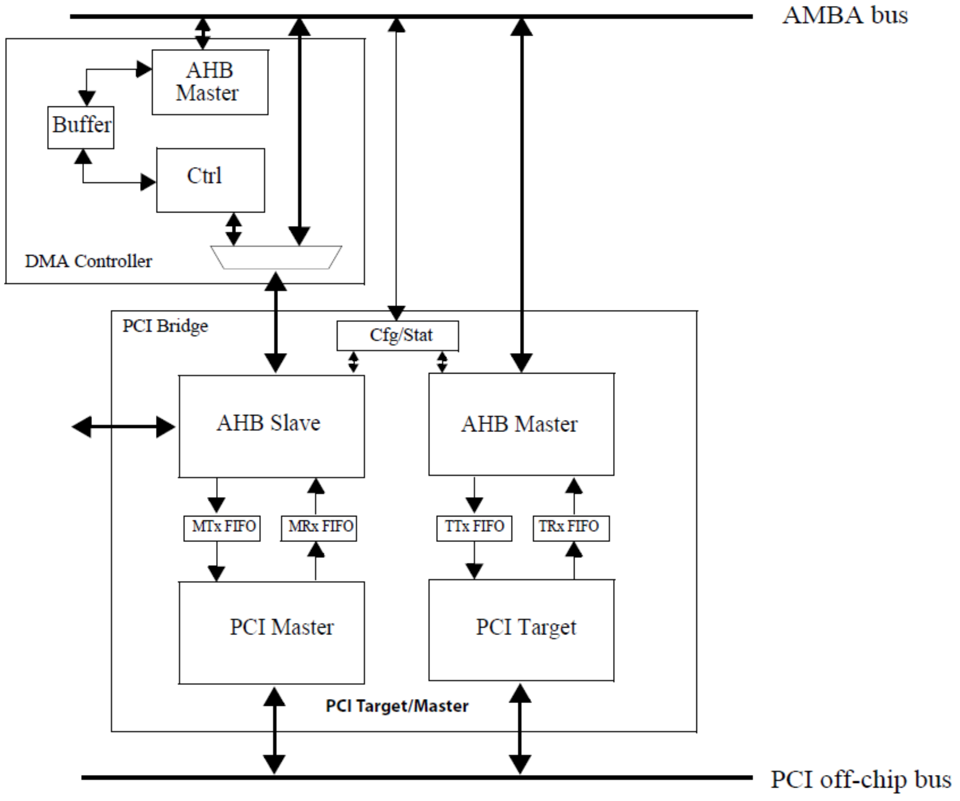


Figure 10.1: DMA Controller Unit

10.2 Operation

The DMA controller is set up by defining the location of memory areas between which the DMA interfaces to PCI and AHB address spaces, as well as the direction, length, and type of transfer. Only 32-bit word transfers are supported.

The DMA transfer is automatically aborted when any kind of error is detected during a transfer. In the event of an error, the ERR bit of the Status and Command Register is set. The DMA controller does not detect deadlocks in its communication channels. If the system concludes that a deadlock has occurred, it manually aborts the DMA transfer. The DMA controller may perform bursts over a 1 KB boundary of the AHB bus, which is the maximum data burst that may occur over the bus per AMBA specification. When the size of the data burst exceeds the 1 KB boundary, AHB idle cycles are automatically inserted to break up the burst over the boundary.

When the DMA is not active, the AHB slave interface of PCI Target/Master unit directly connects to AMBA AHB bus.

10.3 Registers

The core is programmed through registers mapped into APB address space.

Table 10.1: APB Address Register

Register	APB Address
Command and Status Register (DMASCR)	0x80000500
AMBA Target Address Register (DMAATA)	0x80000504
PCI Target Address Register (DMAPTA)	0x80000508
Burst Length Register (DMALNR)	0x8000050C

DMASCR **Address = 0x8000_0500**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									TTYPE[3:0]				ERR	RDY	TD	ST
W																
Reset	[00…0]								0000				0	0	0	0

Figure 10.2: Status and Command Register

Table 10.2: Description of Status and Command Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[00...0]	
7-4	TTYPE	0000	Transfer Type Perform either PCI memory or I/O cycles 0100b: Perform I/O cycles 1000b: Perform memory cycles
3	ERR	0	Error Last transfer was abnormally terminated. If set by the DMA controller, this bit remains one until cleared by writing ‘1’ to it.
2	RDY	0	Ready Current transfer is completed. When set by the DMA Controller this bit remains one until cleared by writing ‘1’ to it.
1	TD	0	Transfer Direction 0: Read from PCI 1: Write to PCI
0	ST	0	Start DMA Transfer Writing ‘1’ starts the DMA transfer. All other registers must be configured before setting this bit. Set by the PCI Master interface when its transaction is terminated with Target-Abort.

DMAATA **Address = 0x8000_0504**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ATA[31:16]															
W																
Reset	0x77FF															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ATA[15:0]															
W																
Reset	0x9324															

Figure 10.3: AMBA Target Address Register

Table 10.3: Description of AMBA Target Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	ATA	0x77FF9324	AMBA Target Address AHB start address for the data on the AMBA bus. In case of error, it indicates the failing address.

DMAPTA **Address = 0x8000_0508**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTA[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PTA[15:0]															
W																
Reset	[00…0]															

Figure 10.4: PCI Target Address Register

Table 10.4: Description of PCI Target Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	PTA	[00...0]	PCI Target Address PCI start address on the PCI bus. This is a complete 32-bit PCI address and is not further mapped by the PCI Target/ Master unit. In case of error, it indicates the failing address.

DMALNR **Address = 0x8000_050C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W																

Reset	[00…0]

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨				LEN[11:0]											
W	▨															
Reset	0000				0x933											

Figure 10.5: Burst Length Register

Table 10.5: Description of Burst Length Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-12	RESERVED	[00...0]	
11-0	LEN	0x933	Burst Length Number of 32-bit words to be transferred.

10.4 Vendor and Device Identifiers

The core has vendor identifier 0x01 (Cobham's Gaisler) and device identifier 0x016.

Chapter 11: SpaceWire Interface

11.1 Overview

The SpaceWire core provides an interface between the AHB bus and a SpaceWire network. It implements the SpaceWire standard (ECSS-E-ST-50-12C) with the protocol identification extension (ECSS-E-ST-50-51C). The Remote Memory Access Protocol (RMAP) target implements the ECSS standard (ECSS-E-ST-50-52C).

The SpaceWire interface is configured through 11 hardware registers accessed through the APB interface. Data is transferred through DMA channels using an AHB master interface.

There are two clock domains for the four SpaceWire ports: (1) the system clock is utilized for the AHB interface, (2) the transmitter clock (TxClk) and the receive sample clock comes from the external SPW_CLK pin. For proper operation, the receiver data rate must be no more than eight times as fast as the system clock and the transmitter clock frequency must be no more than eight times the system clock. The link frequency must be 10+1 MHz.

Figure 11.1 shows a block diagram of the GRSPW module.

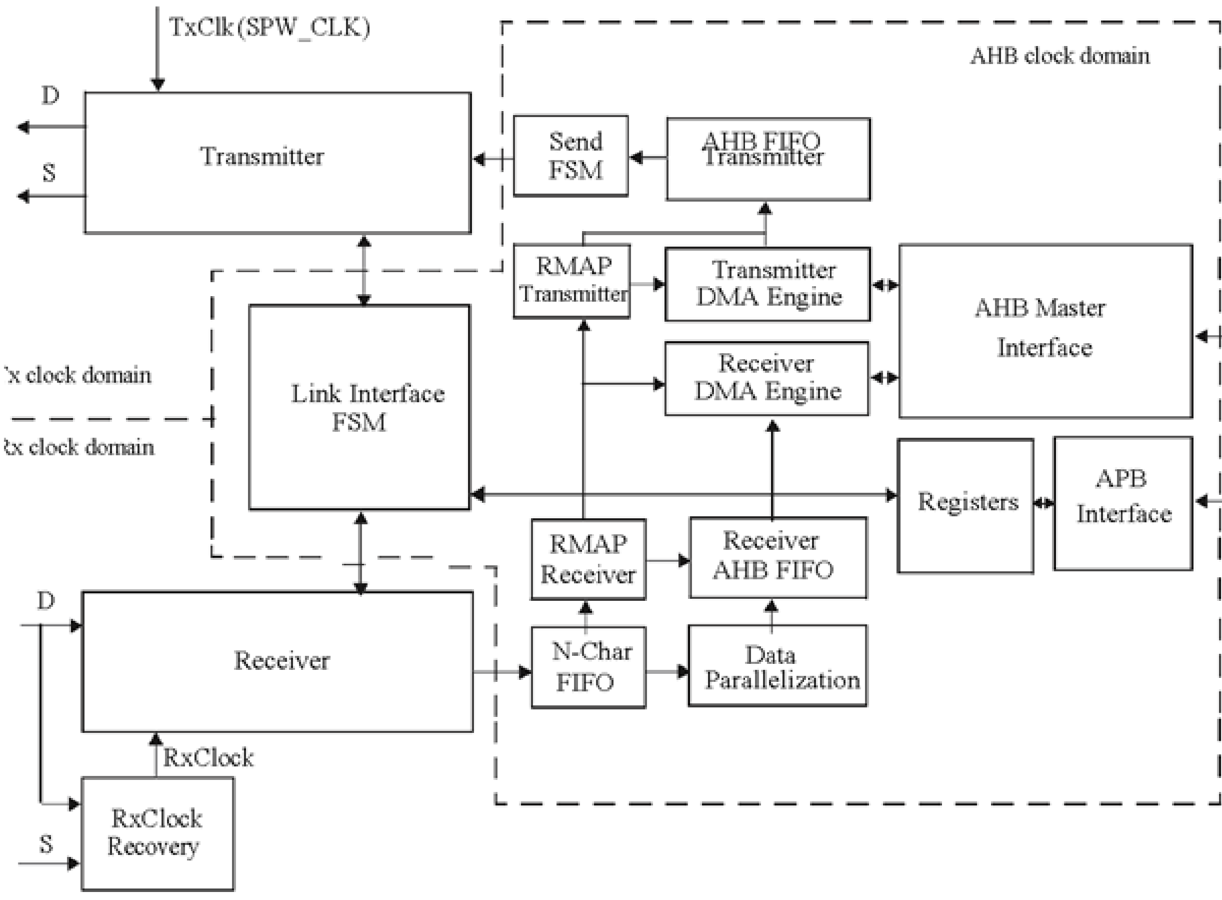


Figure 11.1: GRSPW Block Diagram

11.2 Operation

11.2.1 Overview

The GRSPW is comprised of the link interface, the AMBA interface, and the RMAP handler. A block diagram of the internal structure is shown in **Figure 11.1**.

The link interface consists of the receiver, transmitter and the link interface finite state machine (FSM). They handle communication on the SpaceWire network. The AMBA interface consists of the DMA engines, the AHB master interface and the APB interface. The link interface provides FIFO interfaces to the DMA engines. These FIFOs are used to transfer nor- mal characters (N-Characters) between the AMBA and SpaceWire domains during reception and transmission. The AHB FIFOs are 64-bytes (32-bits wide by 16 words deep).

The RMAP handler is a part of the GRSPW that handles incoming packets which are determined to be RMAP commands. The RMAP command is decoded, and if it is valid, the operation is performed on the AHB bus. If a reply is requested, it is automatically transmitted back to the source by the RMAP transmitter.

Each GRSPW core is controlled through eleven user registers accessible from the APB interface. The registers control clock-generation, the DMA engines, the RMAP handler, and the link interface.

11.2.2 Protocol Support

The GRSPW only accepts packets with a destination address corresponding to the NODE_ADDRESS field of the SPW Node Address Register. Packets with address mismatch will be silently discarded, except when operating in open packet mode, which is covered Section **12.4.10**). The Node Address Register is initialized during reset to the default address of 254. Its value can then be changed to another value by writing to the register.

The GRSPW also requires that the byte following the destination address is a protocol identifier as specified in Part 2 of the SpaceWire standard (does not apply for Open Packet Mode). It is used to determine the destination DMA-channel for a packet. **Figure 11.2** shows the packet type expected by the GRSPW.

RMAP (Protocol ID = 0x01) commands are handled separately from other packets if the hardware RMAP handler is enabled. RMAP is enabled by setting the RE bit in the SPW Control Register. When enabled, all RMAP commands are processed, executed, and replied to in hardware. RMAP replies are still written to the DMA channel. If the RMAP handler is disabled, all packets are written to the DMA channel. More information on the RMAP protocol support is found in Section **11.5**.

All packets arriving with the extended protocol ID (0x00) are sent to the DMA channel. This means that the hardware RMAP command handler will not process incoming RMAP packets that use the extended protocol ID. Note: the reserved extended protocol identifier (ID = 0x000000) is not ignored by the GRSPW. It is treated as ID(0x00) with 2 zero value data bytes. It is up to the client receiving the packets to choose to ignore them.

When transmitting packets, the address and protocol ID fields must be included in the transmit data buffers. They are *not* automatically added by the GRSPW.

Figure 11.2 shows a packet with a normal protocol identifier. The GRSPW also allows reception and transmission of packets with extended protocol identifiers. However, the hardware RMAP handler and RMAP CRC calculator will not process packets with extended protocol identifiers.

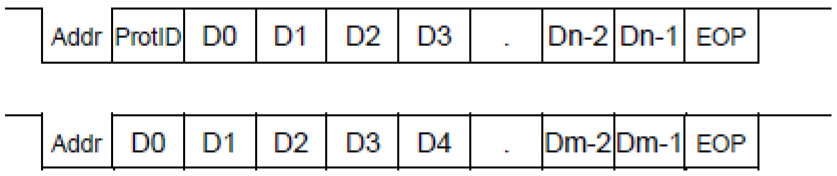


Figure 11.2: SpaceWire Packet Types Supported by the Core

11.3 Link Interface

The link interface handles the communication on the SpaceWire network and consists of a transmitter, receiver, a FSM and FIFO interfaces. An overview of the architecture is found in **Figure 11.1**.

11.3.1 Link Interface FSM

The FSM controls the link interface (a more detailed description is found in the SpaceWire standard). The low-level protocol handling (the signal and character level of the SpaceWire standard) is handled by the transmitter and receiver while the exchange level is handled by the FSM.

The link interface FSM is controlled through the control register. The link can be disabled through the link disable bit, which depending on the current state, either prevents the link interface from reaching the started state or forces it to the error-reset state. When the link is not disabled, the link interface FSM is allowed to enter the started state when either the link start bit is set or when a NULL character has been received and the auto-start (AS) bit is set.

The current state of the link interface determines which type of characters is allowed to be transmitted together with the requests made from the host interface determines the character sent. Time-codes are sent when the FSM is in the run-state and a request is made through the time-interface (described in **Section 11.3.4**).

When the link interface is in the connecting- or run-state, it is allowed to send FCTs. FCTs are sent automatically by the link interface when possible. This is done based on the maximum value of 56 for the outstanding credit counter and the currently free space in the receiver N-Char FIFO. FCTs are sent as long as the outstanding counter is less than or equal to 48 and there are at least 8 more empty FIFO entries than the counter value.

N-Chars are sent in the run-state when they are available from the transmitter FIFO and there are credits available. NULLs are sent when no other character transmission is requested or the FSM is in a state where no other transmissions are allowed.

The credit counter (incoming credits) is automatically increased when FCTs are received and decreased when N-Chars are transmitted. Received N-Chars are stored to the receiver N-Char FIFO for further handling by the DMA interface. Received time-codes are handled by the time-interface.

11.3.2 Transmitter

The state of the FSM, credit counters, requests from the time-interface and requests from the DMA-interface are used to determine the next character to be transmitted. The type of character and the character itself (for N-Chars and time-codes) to be transmitted are presented to the low-level transmitter which is located in the TxClk clock domain.

The separate clock domains allow the SpaceWire link to run on a different frequency than the host system clock. The UT699 has a separate clock input which is used to generate the transmitter clock. Since the transmitter often runs at frequencies greater than 100MHz, as much logic as possible has been placed in the slower system clock domain to minimize power consumption and timing issues.

The transmitter logic in the host clock domain decides what character to send next, sets the proper control signal, and presents any required characters to the low-level transmitter as shown in Figure 80. The transmitter handles send the requested characters and generates parity and control bits as needed. If no requests are made from the host domain, NULLs are sent as long as the transmitter is enabled. Most of the signal and character levels of the SpaceWire standard are handled in the transmitter.

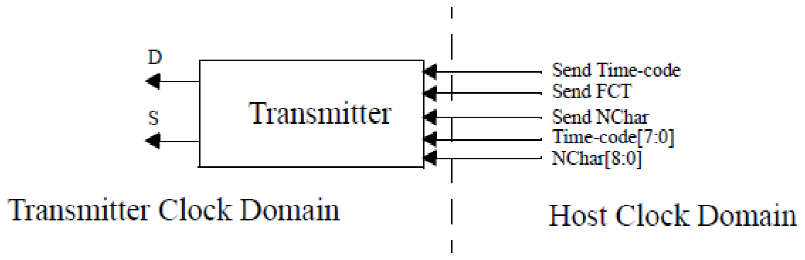


Figure 11.3: Schematic of the Link Interface Transmitter

The transmitter FSM reads N-Chars for transmission from the transmitter FIFO. It is given packet lengths from the DMA interface and appends EOPs/EEPs and RMAP CRC values if requested. When it is finished with a packet, the DMA inter- face is notified and a new packet length value is given.

11.3.3 Receiver

The receiver detects connections from other nodes and receives characters as a bit stream on the data and strobe signals. The receiver is located in the RxClk clock domain, which runs on a clock generated from the received data and strobe signals.

The receiver is activated as soon as the link interface leaves the error reset state. Then after a NULL is received it can start receiving characters. The receiver detects parity, escape, and credit errors, which causes the link interface to enter the error reset state. Disconnections are handled in the link interface part in the system clock domain because no receiver clock is available when the receiver is disconnected from the SpaceWire network.

Received characters are flagged to the host domain and the data is presented in parallel form. The interface to the host domain is shown in Figure 81. L-Chars are handled automatically by the host domain link interface part, while all N-Chars are stored in the receiver FIFO for further handling. If two or more consecutive EOPs/EEPs are received, all but the first are discarded.

No signals go directly from the transmitter clock domain to the receiver clock domain, or vice versa. All signals are

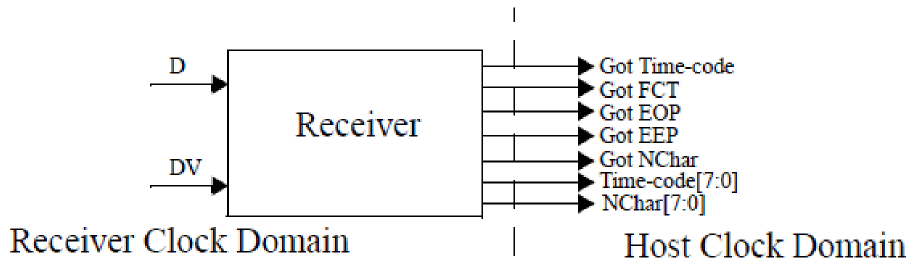


Figure 11.4: Schematic of the Link Interface Receiver

11.3.4 Time Interface

The time interface is used for sending time-codes over the SpaceWire network. Control of the time interface consists of the TICK_IN field of the SPW Control Register, the TICK_OUT field of the SPW Channel Control and Status Register, and the SPW Time Register. The TT and TR bits of the SPW Control Register enable the time transmitter and time receiver, respectively.

Each time-code sent from the SpaceWire port is a concatenation of the TIME_CTRL and TIME_COUNTER fields of the SPW Time Register. The TT bit is used to enable time-code transmissions. It is not possible to send time-codes if this bit is zero.

Received time-codes are stored in the same TIME_CTRL and TIME_COUNTER registers that are used for transmission. The TR bit in the SPW Control Register is used for enabling time-code reception. No time-codes will be received if this bit is zero.

The two enable bits TR and TT are used to ensure that a node will not accidentally both transmit and receive time-codes, which is in violation of the SpaceWire standard. This also ensures that a the master sending time-codes on a network will not have its time-counter overwritten if another faulty node starts sending time-codes.

The TIME_COUNTER field is set to 0 after reset and is incremented each time the TICK_IN field is written to when the TT bit is set. This action causes the link interface to send the new value on the network. A tick-in should not be generated too often. If the time-code after the previous tick-in has not been sent, the register will not be incremented and the new value will not be sent. The TICK_IN field is automatically cleared when the value has been sent. Therefore, no new ticks should be generated until this field is zero.

A tick-out is generated each time a valid time-code is received and the TR bit is set. When the tick-out is generated, the TICK_OUT register field is asserted until it is cleared by writing a one to it.

The TIME_COUNTER field of the SPW Time Register indicates the current time counter value. It is updated each time a time-code is received and the TR bit is set. The same register is used for transmissions and can also be written directly from the APB interface.

The control bits of the time-code are always stored to the TIME_CTRL field of the SPW Time Register when a time-code is received whose time-count is one more than the current time-counter register of the node. The TIME_CTRL field is accessed by reading the SPW Time Register from the APB interface.

It is possible to have both the time-transmission and reception functions enabled at the same time.

11.3.5 Clock Divider

The transmitter frequencies for the link-state and run-state are set in the SPW Clock Divisor Register. During link initialization, the divisor for the SpaceWire input clock is (CLOCK_DIVISOR_LINK+1). For example, if the SpaceWire input clock is 50MHz, CLOCK_DIVISOR_LINK should be set to four in order to set the transmitter frequency to 10Mbit/s during link initialization.

Once the link interface has entered run-state, the transmitter frequency is the SpaceWire input clock divided by (CLOCK_DIVISOR_RUN+1). For example, if the SpaceWire input clock is 50MHz, the transmitter operates sat 50Mbit/s if CLOCK_DIVISOR_RUN is set to zero.

11.4 Receiver DMA Channels

11.4.1 Basic Functionality

The receiver DMA engine reads N-Chars from the N-Char FIFO and stores them to a DMA channel. Reception is based on descriptors located in a consecutive area in memory that hold pointers to buffers where packets should be stored. When a packet arrives at the GRSPW it reads a descriptor from memory and stores the packet to the memory area pointed to by the descriptor. Then it stores status to the same descriptor and increments the descriptor pointer to the next one.

11.4.2 Setting up the GRSPW for reception

A few registers need to be initialized before reception can take place. First the link interface needs to be put in the run state before any data can be sent. This is accomplished by enabling the link interface which is accomplished by setting the Link Start (LS) bit in the SWP Control Register, and by enabling the receiver, and setting the SPW Clock Divisor register, described below. The DMA Receiver Max Length Register sets the maximum packet size that the channel can receive.

Larger packets are truncated with the excessive part spilled. When truncation occurs, the Truncated (TR) bit will be given in the status field of the descriptor. The minimum value for the RX_MAX_LENGTH field in the SPW DMA Channel Receiver Max Length Register is four, and the value can only be incremented in steps of four bytes. If the RX_MAX_LENGTH field is set to zero, the receiver will ***not*** **function correctly**.

The SPW Node Address Register needs to be set to hold the address of the SpaceWire node. Packets received with an incorrect address are discarded. Finally, the descriptor table and SPW Control Register must be initialized. This will be described in the two following sections.

11.4.3 Setting up the Descriptor Table Address

The GRSPW core reads descriptors from an area in memory pointed to by the SPW Receiver Descriptor Table Address Register. The register consists of a base address and a descriptor selector. The RX_BASE ADDRESS field points to the beginning of the memory area and must start on a 1 kB aligned address. It is also limited to be 1 kB in size, which means the maximum number of descriptors is 128.

The RX_DESC_SELECT field points to individual descriptors and is incremented by one when a descriptor has been used. When the selector reaches the upper limit of the area, it wraps to the beginning automatically. It can also be set to wrap automatically by setting the Wrap (WR) bit in the descriptors. The idea is that the selector should be initialized to 0 (the start of the descriptor area). But it can also be written with another 8-byte aligned value to start somewhere in the middle of the area. It still wraps to the beginning of the area.

If one wants to use a new descriptor table the Receiver Enable (RE) bit in the SPW Channel Control and Status Register has to be cleared first. When the RX Active (RX) bit in the SPW Channel Control and Status Register for the channel is cleared, it is safe to update the descriptor table register. When this is finished and descriptors are enabled the receiver enable bit can be set again.

11.4.4 Enable Descriptors

As mentioned earlier one or more descriptors must be enabled before reception can take place. Each descriptor is 8 bytes in size and the layout is shown in Figure 82a and 82b. The descriptors should be written to the memory area pointed to by the receiver descriptor table address register. When new descriptors are added, they must always be placed immediately following the previous one written to the area. Otherwise, they will not be recognized.

A descriptor is enabled by setting the PACKET_ADDRESS pointer to point at a location where data can be stored, and then set- ting the Enable (EN) bit. The Wrap (WR) bit can be set to cause the DESCRIPTOR_SELECTOR field in the SPW Receiver Descriptor Table Address Register to be set to zero when reception has finished to this descriptor. The Interrupt Enable (IE) bit should be set if an interrupt is wanted when the reception has finished. The Receive Interrupt (RI) bit of the DMA Channel Control and Status Register must also be set for this to happen.

The descriptor packet address should be word aligned. All accesses on the bus are word accesses so complete words will always be overwritten regardless of whether all 32-bits contain received data. Also, if the packet does not end on a word boundary, the complete word containing the last data byte will be overwritten.

SPWRDW0 **Offset = 0x00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TR	DC	HC	EP	IE	WR	EN	PACKET_LENGTH[24:16]								
Reset	0	0	0	0	0	0	0	[--…-]								

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PACKET_LENGTH[15:0]															
Reset	[--…-]															

Figure 11.5: SpaceWire Receive Descriptor Word 0

Table 11.1: Description of SpaceWire Receive Descriptor Word 0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	TR	0	Truncated Packet was truncated due to maximum length violation.
30	DC	0	Data CRC 0: No data CRC error detected 1: Data CRC error detected
29	HC	0	Header CRC 0: No header CRC error detected 1: Header CRC error detected
28	EP	0	EEP Termination 0: Normal packet termination 1: Packet ended with an Error End of Packet character
27	IE	0	Interrupt Enable 0: No interrupt generated upon packet reception 1: An interrupt generates when a packet has been received if the Receive Enable interrupt bit in the DMA Channel Control and Status Register is set.
26	WR	0	Wrap 0: DESCRIPTOR_SELECTOR increases by 0x8 to use the descriptor at the next memory location. The descriptor

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			table is limited to 1 KB in size and the pointer automatically wraps back to the base address when it reaches the 1 KB boundary. 1: The next descriptor used by the GRSPW2 will be the first one in the descriptor table at the base address.
25	EN	0	Enable Descriptor 0: Descriptor disabled 1: Descriptor enabled. This means that the descriptor contains valid control values and the memory area pointed to by the PACKET_ADDRESS field can be used to store a packet.
24-0	Packet Length	[--...-]	The number of bytes received by the buffer. Only valid after EN has been set to 0 by the GRSPW2.

SPWRDW1 **Offset = 0x04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PACKET_ADDRESS[31:16]															
W																
Reset	[--…-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PACKET_ADDRESS[15:2]														00	
W																
Reset	[--…-]														00	

Figure 11.6: SpaceWire Receive Descriptor Word 1

Table 11.2: Description of SpaceWire Receive Descriptor Word 1

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	Packet Address	[--…-]	The address pointing to the buffer which will be used to store the received packet.
1-0	Packet Address	00	VHDL generics are both set to zero

11.4.5 Setting up the DMA Control Register

The final step to receive packets is to set the control register in the following steps: The receiver must be enabled by setting the Receiver Enable (RE) bit in the SPW DMA Channel Control and Status Register. This can be done anytime; before this bit is set, no receiver operation will occur. The Receiver Descriptors Available (RD) bit in the DMA Control Register is then set to indicate that there are new active descriptors. This must always be done after the descriptors have been enabled or the GRSPW might not notice the new descriptors. More descriptors can be activated when reception has already started by enabling the descriptors and setting the RD bit. When these bits are set, reception starts immediately as soon as data arrives.

11.4.6 The Effect to the Control Bits during Reception

When the receiver is disabled, all packets going to the DMA-channel are discarded. If the receiver is enabled the next state is entered where the RD bit is checked. This bit indicates whether there are active descriptors or not and should be set by the external application using the DMA channel each time descriptors are enabled as mentioned above. If the RD is '0' and the No Spill (NS) bit is 0, the packets will be discarded. If NS is '1', the GRSPW core waits until RD is set and then stores the received data.

When RD is set the next descriptor is read and if enabled, the packet is received to the buffer. If the read descriptor is not enabled, i.e., the EN bit in the descriptor is 0, RD is set to '0' and the packet is spilled depending on the value of NS.

The receiver can be disabled at any time and causes all packets received afterwards to be discarded. If a packet is currently being received when the receiver is disabled, the reception will finish normally. The RD bit can also be cleared at any time. It will not affect any ongoing receptions, but no more descriptors will be read until it is set again. RD is also cleared by the GRSPW when it reads a disabled descriptor as discussed above.

11.4.7 Address Recognition and Packet Handling

When the receiver N-Char FIFO is not empty, N-Chars are read by the receiver DMA engine. The first character is interpreted as the logical address, which is compared to the node address register. If it does not match, the complete packet is dis- carded (up to and including the next EOP/EEP). Otherwise, the next action taken depends on whether the node is configured with RMAP or not. If RMAP is disabled all packets are stored to the DMA channel, and depending on the conditions mentioned in the previous section, the packet will be received or not. If the packet is received, complete packet including address and protocol ID, but excluding EOP/EEP, is stored to the address indicated in the descriptor. Otherwise, the complete packet is discarded.

If RMAP is enabled, the Protocol Identifier and Packet Type / Command / Source Path Address Length bytes in the received packet are first checked before any decisions are made. If the incoming packet is a RMAP packet (ID = 0x01) and the Command field is 01b, the packet is processed by the RMAP command handler which is described in Section 11.6. Otherwise, the packet is processed by the DMA engine as when RMAP is disabled.

At least 2 non-EOP/EEP N-Chars need to be received for a packet to be stored to the DMA channel. If it is an RMAP packet with hardware RMAP enabled, 3 N-Chars are needed since the Command field of the Packet Type / Command / Source Path Address Length byte determines where the packet is processed. Packets are smaller than these sizes are discarded.

11.4.8 Status Bits

When the reception of a packet is finished, the enable (EN) bit in the current descriptor is set to '0'. When EN is '0', the status bits are also valid as are the number of received bytes indicated in the PACKET_LENGTH field. The Packet Received (PR) bit in the SPW DMA Channel Control and Status Register is set each time a packet has been received. The GRSPW can also be made to generate an interrupt for this event if the Receive Interrupt (RI) bit is set.

CRC is always checked for all RMAP packets. If the received packet is not of RMAP type, the CRC error indication bits in the descriptor should be ignored. If the received packet is of RMAP type, the bits are valid and the Header CRC (HC) bit is set if a header CRC error was detected. In this case, the data CRC will not be calculated and the Data CRC (DC) bit is undefined. If the header CRC was correct, the DC bit will also contain a valid value and is set to '1' if a data CRC error was detected.

11.4.9 Error Handling

If a packet reception needs to be aborted because of congestion on the network, the suggested solution is to set the Link Dis- able (LD) bit in the SPW Control Register to '1'. Unfortunately, this will also cause the packet currently being transmitted to be truncated, but this is the only safe solution since packet reception is a passive operation depending on the transmitter at the other end. A channel reset bit could be provided, but is not a satisfactory solution since the not transmitted characters would still be in the transmitter node. The next character (somewhere in the middle of the packet) would be interpreted as the node address which would probably cause the packet to be discarded, but not with 100% certainty. Usually, this action is performed when a reception is stuck because of the transmitter not providing additional data. The channel reset would not resolve this congestion.

If an AHB error occurs during reception, the current packet is spilled up to and including the next EEP/EOP. The currently active channel is disabled and the receiver enters the idle state. The Link Disable (LD) bit in the SPW Control Register and the Link State (LS) bit in the SPW Status Register indicate this condition.

11.4.10 Open Packet Mode

The GRSPW supports an open packet mode where all the data received is stored to the DMA channel regardless of the node address and possible early EOPs/EEPs. This means that all non-EOP/EEP N-Chars received will be stored to the DMA channel. The RX_MAX_LENGTH field of the SPW DMA Channel Receiver Max Length Register is still checked and packets exceeding this size will be truncated.

RMAP commands will be handled by the RMAP handler when open packet mode is enabled by setting the OPM bit in the SPW Control Register, provided that the RMAP Enable (RE) bit is also set. If RE is cleared, RMAP commands will be stored to the DMA channel.

11.5 Transmitter DMA Engine

11.5.1 Basic Functionality

The transmitter DMA engine reads data from the AHB bus and stores it to the transmitter FIFO for transmission on the SpaceWire network. Transmission is based on the same type of descriptors as for the receiver and the descriptor table has the same alignment and size restrictions. When new descriptors are enabled, the GRSPW reads them and transfers the amount of data indicated by the descriptor.

11.5.2 Setting up the GRSPW2 for Transmission

Four steps need to be performed before transmissions can be done with the GRSPW. First, the link interface must be enabled and started by setting the Link Start (LS) bit in the SPW Control Register. Then the address of the descriptor table needs to be written to the TX_BASE_ADDRESS field of the SPW Transmitter Descriptor Table Address Register and one or more descriptors must be enabled in the table. Finally, the Transmit Enable (TE) bit in the SPW DMA Channel Control and Status Register is written with a ‘1’ to initiate transmission. These steps will be covered in more detail in the next sections.

11.5.3 Enable Descriptors

The transmitter descriptor table address register works in the same way as the corresponding address register for the receiver descriptor table, which was covered in **Section 11.4.3**.

To transmit packets, one or more descriptors have to be initialized in memory as follows: First, the number of bytes to be transmitted must be written to the HEADER_LENGTH and DATA_LENGTH fields. Next, a pointer to the header and data has to be set by writing to the HEADER_ADDRESS and DATA_ADDRESS fields. There are two different length and address fields in the transmit descriptors because there are separate pointers for header and data. If a length field is zero, the corresponding part of a packet is skipped. If both are zero, no packet is sent. The maximum header length is 255 bytes and the maximum data length is 16MB - 1. When the pointer and length fields have been set, the Enable (EN) bit should be set to enable the descriptor. This must always be done last. The other control bits must also be set before enabling the descriptor.

The transmit descriptors occupy 16-bytes in memory and the maximum number of descriptors in a single table is 64. The different fields of a descriptor are shown in the following figure along with their relative memory offsets.

The Calculate CRC (CC) field should be set if RMAP CRC is to be calculated and inserted into the current packet. The header CRC is calculated from the data fetched from the header pointer and the data CRC is generated from data fetched from the data pointer. The CRCs are appended after the corresponding fields. The first NON_CRC_BYTES of the header are not included in the CRC calculation.

The CRC is skipped if the corresponding length field is zero. If both fields are zero, nothing will be sent including the EOP.

11.5.4 Starting Transmission

When the descriptors have been initialized, the Transmit Enable (TE) bit in the SPW DMA Control Register has to be set to tell the GRSPW to start transmitting. New descriptors can be activated in the table even while a transmission is active or in progress. Each time a set of descriptors is added, the TE bit should be set. This has to be done as the GRSPW clears the TE bit whenever it encounters a disabled descriptor.

SPWTDW0 **Offset = 0x00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R															DC	HC
W																
Reset	[--...-]														--	--

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LE	IE	WR	EN	NON_CRC_BYTES[3:0]				HEADER_LENGTH[7:0]							
W																
Reset	--	--	--	--	--				[--...-]							

Figure 11.7: SpaceWire Transmitter Descriptor Word 0

Table 11.3: Description of SpaceWire Transmitter Descriptor Word 0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-18	RESERVED	[--...-]	
17	DC	-	Append data CRC Append CRC calculated according to the RMAP specification after the data sent from the data pointer. The CRC covers all the bytes from this pointer. A null CRC will be sent if the length of the data field is zero.
16	HC	-	Append header CRC (HC) - Append CRC calculated according to the RMAP specification after the data sent from the header pointer. The CRC covers all bytes from this pointer except a number of bytes in the beginning

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			specified by the non-crc bytes field. The CRC will not be sent if the header length field is zero.
15	LE	-	Link Error 0: No link error occurred 1: A link error occurred during the transmission of this packet
14	IE	-	Interrupt Enable 0: Disable interrupts 1: An interrupt generates when the packet has been transmitted and the Transmitter Interrupt TE enable bit in the DMA control register is set.
13	WR	-	Wrap 0: The descriptor pointer is increased with 0x10 to use the descriptor at the next higher memory location. 1: The descriptor pointer wraps and the next descriptor read will be the first one in the table (at the base address).
12	EN	-	Enable 0: Disable transmitter descriptor 1: Enable transmitter descriptor. When all control fields (address, length, wrap and crc) are set, this bit should be set. While the bit is set the descriptor should not be modified since this might corrupt the transmission in progress. The GRSPW2 clears this bit when the transmission has finished.
11-8	NON_CRC_B YTES	[--...-]	Sets the number of bytes in the beginning of the header which should not be included in the CRC calculation. This is necessary when using path addressing since one or more bytes in the beginning of the packet might be discarded before the packet reaches its destination.
7-0	HEADER_LENGTH	[--...-]	Header length in bytes. If set to zero, the header is skipped.

SPWTDW1 **Offset = 0x04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HEADER_ADDRESS[31:16]															
W																
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HEADER_ADDRESS[15:0]															
W																
Reset	[--...-]															

Figure 11.8: SpaceWire Transmitter Descriptor Word 1

Table 11.4: Description of SpaceWire Transmitter Descriptor Word 1

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	HEADER_ADDRESS	[--...-]	Address from where the packet header is fetched. Does not need to be word aligned.

SPWTDW2 **Offset = 0x08**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨								DATA_LENGTH[23:16]							
W																
Reset	[00…0]								[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA_LENGTH[15:0]															
W																
Reset	[00…0]															

Figure 11.9: SpaceWire Transmitter Descriptor Word 2

Table 11.5: Description of SpaceWire Transmitter Descriptor Word 2

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-24	RESERVED	[00...0]	
23-0	DATA_LENGTH	[00...0]	Length of data part of the packet in bytes. If set to zero, no data will be sent. If both data- and header-lengths are set to zero, no packet will be sent.

SPWTDW3 **Offset = 0x0C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA_ADDRESS[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA_ADDRESS[15:0]															
W																
Reset	[00…0]															

Figure 11.10: SpaceWire Transmitter Descriptor Word 3

Table 11.6: Description of SpaceWire Transmitter Descriptor Word 3

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	DATA_ADDRESS	[00...0]	Address from where data is read. Does not need to be word aligned.

11.5.5 The Transmission Process

When the txen bit is set, the GRSPW2 starts reading descriptors immediately. The number of bytes indicated is read and transmitted. When a transmission has finished, status is written to the first field of the descriptor and a packet sent bit is set in the DMA control register. If an interrupt was requested, it also generates. Then a new descriptor is read and, if enabled, a new transmission starts; otherwise, the transmit enable bit clears and nothing happens until it is enabled again.

11.5.6 The Descriptor Register

The internal pointer which is used to keep the current position in the descriptor table can be read and written through the APB interface. This pointer is set to zero during reset and is incremented each time a descriptor is used. It wraps automatically when the 1024 bytes limit for the descriptor table is reached or it can be set to wrap earlier by setting a bit in the current descriptor.

The descriptor table register can be updated with a new table anytime when no transmission is active. No transmission is active if the transmit enable bit is zero and the complete table has been sent or if the table is aborted (explained below). If the table is aborted, one has to wait until the transmit enable bit is zero before updating the table pointer.

11.5.7 Error Handling

If the Abort TX (AT) bit in the SPW DMA Channel Control and Status Register is set, the current transmission will be aborted; the packet is truncated and an EEP is inserted. This is only useful if the packet needs to be aborted because of congestion on the SpaceWire network. Aborting a transmission will not help with congestion on the AHB since AHB slaves have a maximum of 16 waitstates. The aborted packet will have its Link Error (LE) bit set in the descriptor. The TE bit in the DMA control register bit is also cleared and no new transmissions will occur until the transmitter is enabled again.

When an AHB error is encountered during transmission, the currently active DMA channel is disabled, the packet is truncated, an EEP is inserted if the transmission has started, and the transmitter goes to idle mode. The TX AHB Error (TA) bit in the DMA control register is set to indicate this error condition. The client using the channel has to correct the error and enable the channel again.

11.6 Remote Memory Access Protocol (RMAP)

The Remote Memory Access Protocol (RMAP) is used to implement access to resources in the node via the SpaceWire Link. Some common operations are reading and writing to memory, registers and FIFOs. This section describes the basics of the RMAP protocol and the target implementation.

11.6.1 Fundamentals of the Protocol

RMAP is a protocol which is designed to provide remote access via a SpaceWire network to memory mapped resources on a SpaceWire node. It has been assigned protocol ID 0x01. It provides three operations: write, read and read-modify-write. These operations are posted operations, which means that a source does not wait for an acknowledgement or reply. It also implies that any number of operations can be outstanding at any time and that no timeout mechanism is implemented in the protocol. Time-outs must be implemented in the user application which sends the commands. Data payloads of up to 16 MB - 1 are supported in the protocol. A destination can be requested to send replies and to verify data before executing an operation. A complete description of the protocol is found in the RMAP standard.

11.6.2 Implementation

The GRSPW includes a handler for RMAP commands that processes all incoming packets with protocol ID = 0x01 and Packet Type field in the Command byte set to 0x01. When such a packet is detected, it is not stored to the DMA channel. Instead, it is passed to the RMAP receiver, which is seen in **Figure 11.1**.

The GRSPW implements all three commands defined in the standard with some restrictions. The implementation is based on Draft C of the RMAP standard. Support is only provided for 32-bit, big-endian systems. This means that the first byte received is the MSB in a word. The command handler will not receive RMAP packets using the extended protocol ID. These are always sent to the DMA channel.

The RMAP receiver processes commands. If they are correct and accepted, the operation is performed on the AHB bus and a reply is formatted. If an acknowledge is requested, the RMAP transmitter automatically sends the reply. RMAP transmissions have priority over DMA channel transmissions.

Packets with a mismatching destination logical address are never passed to the RMAP handler. The DESTINATION_KEY field of the SPW Destination Key Register must match the Destination Key byte of the incoming packet. If there is a mismatch and a reply has been requested, the error code of the Status byte is set to '3'. Replies are sent if and only if the ACK field in the Command byte is set to '1'.

Detection of all error codes is supported. When a failure occurs during a bus access, the error code is set to 1 (General error). There is predetermined order in which error-codes are set in the case of multiple errors in the GRSPW as shown in **Table 11.7**.

Table 11.7: SpaceWire RMAP Packet Error Codes and Detection Order (Highest Priority is 1)

DETECTION ORDER	ERROR CODE	ERROR	ERROR DESCRIPTION	APPLICABILITY		
				WRITE	READ	RMW
1	2	Unused RMAP Packet Type or Command Code	The Header CRC was decoded correctly but the packet type is reserved or the command is not used by the RMAP protocol.	X	X	X
2	3	Invalid key	The Header CRC was decoded correctly but the device key did not match that expected by the target user application.	X	X	X
11	4	Invalid Data CRC	Error in the CRC of the data field	X		X
7	5	Early EOP	EOP marker detected before the end of	X	X	X

DETECTION ORDER	ERROR CODE	ERROR	ERROR DESCRIPTION	APPLICABILITY		
				WRITE	READ	RMW
			the data.			
11	6	Too much data	More than the expected amount of data in a command has been received.	X	X	X
7	7	EEP	EEP marker detected immediately after the header CRC or during the transfer of data and Data CRC or immediately thereafter. Indicates that there was a communication failure of some sort on the network.	X	X	X
NA	8	Reserved	Reserved			
3	9	Verify buffer overrun	The verify before write bit of the command was set so that the data field was buffered in order to verify the Data CRC before transferring the data to target memory. The data field was longer than able to fit inside the verify buffer resulting in a buffer overrun. Note that the command is not executed in this case.	X		X
5	10	RMAP Command not implemented or not authorized	The target user application did not authorize the requested operation. This may be because the command requested has not been implemented.	X	X	X
4	11	RMW Data Length error	The amount of data in a RMW command is invalid (0x01, 0x03, 0x05, 0x07 or greater than 0x08).			X
12	12	Invalid Target Logical Address	The Header CRC was decoded correctly but the Target Logical Address was not the value expected by the target.	X	X	X

Note: *The AHB error is not guaranteed to be detected before Early EOP/EEP or Invalid Data CRC. For very long accesses the AHB error detection might be delayed causing the other two errors to appear first.

Read accesses are performed on the fly; that is, they are not stored in a temporary buffer before transmitting. This means that the error code 1 (General error) will never be seen in a read reply since the header has already been sent when the data is read. If the AHB error occurs, the packet will be truncated and ended with an EEP.

The details of the support for the different commands are now presented. All defined commands that are received with a non-supported option set will not be executed. This might result in a reply being sent with error code 10 (RMAP Command not implemented or authorized)

11.6.3 Write Commands

Write commands are divided into two subcategories when examining their capabilities: verified writes and non-verified writes. Verified writes have a length restriction of four bytes and the address must be aligned to the size. For example, a 1 byte write can be written to any address, 2-byte writes must be halfword aligned, 3-byte writes are not allowed, and 4-byte writes must be word aligned. Since there will always be only one AHB operation performed for each RMAP verified write command, the Increment Address bit in the Command byte can be set to either '0' or '1'.

Non-verified writes have no restrictions when the incrementing bit is set to '1', indicating sequential memory access. If it is set to '0' the number of bytes must be a multiple of 4 and the address must be word aligned. There is no guarantee how many words will be written when early EOP/EEP is detected for non-verified writes.

11.6.4 Read Commands

Read commands are performed on the fly when the reply is sent. Thus, if an AHB error occurs, the packet will be truncated and ended with an EEP. There are no restrictions for incrementing reads. But non-incrementing reads have the same alignment restrictions as non-verified writes. Note: error code 10 will be sent in the reply if a violation was detected, even if the length field was zero.

11.6.5 Read-Modify-Write Commands

All read-modify-write sizes are supported except six, which would result in three bytes being read from and written to the AMBA bus. The RMW bus accesses have the same restrictions as the verified writes. As in the verified write case, the incrementing bit can be set to any value since only one AHB bus operation will be performed for each RMW command.

11.6.6 Controls

The RMAP command handler runs in the background without any external intervention, but there are a few control configuration options. The RMAP Enable (RE) bit in the SPW Control Register can be used to completely disable the RMAP command handler. When it is set to '0', no RMAP packets will be stored to the DMA channel instead of being handled in hardware.

There is a possibility that RMAP commands will not be performed in the order they arrive. This can happen if a read command arrives before one or more write commands. Since the command handler stores replies in a buffer with more than one entry, several commands may be processed even if no replies are sent. Data for read replies is read when the reply is sent and writes coming after the read might have been performed before the read command if there was congestion in the transmitter. To prevent this situation, the RMAP Buffer Disable (RD) bit can be set to force the command handler to only use one buffer. The last control option for the command handler is to set the destination key, which is discussed in **Section 11.6.2**.

Table 11.8: GRSPW2 Hardware RMAP Handling of Different Packet Type and Command Fields

PACKET TYPE		RMAP COMMAND CODES				COMMAND	ACTION
BIT 7	BIT6	BIT 5	BIT4	BIT 3	BIT 2		
Reserved	Command / Response	Write /Read	Verify data before write	Acknow -ledge	Increment Address		
0	0	0	0	0	0	Response	Stored to DMA-channel.
0	0	0	0	0	0	Reply to acknowledge	
0	1	0	0	0	0	Not used	Does nothing. No reply is sent.
0	1	0	0	0	1	Not used	Does nothing. No reply is sent.
0	1	0	0	1	0	Read single address	Executed normally. Address has to be word aligned and data size a multiple of four. Reply is sent. If alignment restrictions are violated error code is set to 10.
0	1	0	0	1	1	Read incrementing address.	Executed normally. No restrictions. Reply is sent.
0	1	0	1	0	0	Not used	Does nothing. No reply is sent.
0	1	0	1	0	1	Not used	Does nothing. No reply is sent.
0	1	0	1	1	0	Not used	Does nothing. Reply is sent with error code 2.
0	1	0	1	1	1	Read-Modify-Write incrementing address	Executed normally. If length is not one of the allowed rmw values nothing is done and error code is set to 11. If the length was correct, alignment restrictions are checked next. 1 byte can be rmw to any address. 2 bytes must be halfword aligned. 3 bytes are not allowed. 4 bytes must be word aligned. If these restrictions are violated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
0	1	1	0	0	0	Write, single-	Executed normally.

PACKET TYPE		RMAP COMMAND CODES				COMMAND	ACTION
BIT 7	BIT6	BIT 5	BIT4	BIT 3	BIT 2		
Reserved	**Command / Response**	**Write /Read**	**Verify data before write**	**Acknow -ledge**	**Increment Address**		
						address, do not verify before writing, no acknowledge	Address has to be word aligned and data size a multiple of four. If alignment is violated nothing is done. No reply is sent.
0	1	1	0	0	1	Write, incrementing address, do not verify before writing, no acknowledge	Executed normally. No restrictions. No reply is sent.
0	1	1	0	1	0	Write, single-address, do not verify before writing, send acknowledge	Executed normally. Address has to be word aligned and data size a multiple of four. If align- ment is violated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
0	1	1	0	1	1	Write, incrementing address, do not verify before writing, send acknowledge	Executed normally. No restrictions. If AHB error occurs error code is set to 1. Reply is sent.
0	1	1	1	0	0	Write, single address, verify before writing, no acknowledge	Executed normally. Length must be 4 or less. Other- wise nothing is done. Same alignment restrictions apply as for rmw. No reply is sent.
0	1	1	1	0	1	Write, incrementing address, verify before writing, no acknowledge	Executed normally. Length must be 4 or less. Other- wise nothing is done. Same alignment restrictions apply as for rmw. If they are violated nothing is done. No reply is sent.
0	1	1	1	1	0	Write, single address,	Executed normally. Length must be 4 or

PACKET TYPE		RMAP COMMAND CODES				COMMAND	ACTION
BIT 7	BIT6	BIT 5	BIT4	BIT 3	BIT 2		
Reserved	Command / Response	Write /Read	Verify data before write	Acknow -ledge	Increment Address		
						verify before writing, send acknowledge	less. Other- wise nothing is done and error code is set to 9. Same alignment restrictions apply as for rmw. If they are violated nothing is done and error code is set to 10. If an AHB error occurs error code is set to 1. Reply is sent.
0	1	1	1	1	1	Write, single acknowledge	

11.7 AMBA Interface

The AMBA interface consists of an APB interface, an AHB master interface, and DMA FIFOs. The APB interface provides access to the user registers, which are described in Section 11.8. The DMA engines have 32-bit wide FIFOs to the AHB master interface, which are used when reading and writing to the bus.

The transmitter DMA engine reads data from the bus in bursts which are half the FIFO size in length. A burst is always started when the FIFO is half-empty or if it can hold the last data for the packet. The burst containing the last data might have a shorter length if the packet is not an even number of bursts in size.

The receiver DMA works in the same way except that it checks if the FIFO is half-full and then performs a burst write to the bus that is half the FIFO size in length. The last burst might be shorter.

11.7.1 APB Slave Interface

As mentioned above, the APB interface provides access to the user registers which are 32-bits in width. The accesses to this interface are required to be aligned word accesses. The result is undefined if this restriction is violated.

11.7.2 AHB Master Interface

The GRSPW contains a single master interface used by both the transmitter and receiver DMA engines. The arbitration algorithm between the channels is done so that the current owner always acquires the interface if requested. This will not lead to starvation problems since the DMA engines always deassert their requests between accesses.

AHB accesses are always word accesses (HSIZE = 0x010) of type incremental burst with unspecified length (HBURST = 0x001) if RMAP is disabled. Byte and halfword accesses are always non-sequential. The burst length will be half the AHB FIFO size except for the last transfer for a packet which might be smaller. Shorter accesses are also done during descriptor reads and status writes.

The AHB master also supports non-incrementing accesses where the address will be constant for several consecutive accesses. HTRANS will always be non-sequential in this case, while for incrementing accesses it is set to sequential after the first access. This feature is included to support non-incrementing reads and writes for RMAP.

If the GRSPW does not need the bus after a burst has finished there will be one wasted cycle (HTRANS = IDLE). The core provides full support for ERROR, RETRY and SPLIT responses, while BUSY transfer types are never requested.

11.8 Registers

The UT699 has four SpaceWire nodes, each comprised of a GRSPW cores. Each core has its own set of registers mapped into APB memory space. The APB mapping for each GRSPW core is listed in Figure 2. The relative offset of each register is shown in Table 84 below. GRSPW cores 3 and 4 have RMAP functionality, so any RMAP registers apply to these cores. Cores 1 and 2 do not have RMAP functionality.

Table 11.9: GRSPW2 Registers

REGISTER	APB ADDRESS OFFSET
SPW Control Register (SPWCTR)	0x0
SPW Status Register (SPWSTR)	0x4
SPW Node Address Register (SPWNDR)	0x8
SPW Clock Divisor Register (SPW_CLK)	0xC
SPW Destination Key Register (SPWKEY)	0x10
SPW Time Register (SPWTIM)	0x14
Reserved	0x18
SPW DMA Channel Control and Status Register (SPWCHN)	0x20
SPW DMA Channel Receiver Maximum Length Register (SPWRXL)	0x24
SPW DMA Transmit Descriptor Register (SPWTXD)	0x28
SPW DMA Receive Descriptor Register (SPWRXD)	0x2C

Address = 0x8000_0A00
Address = 0x8000_0B00
Address = 0x8000_0C00
Address = 0x8000_0D00

SPWCTR

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RA	RX	RC												RD	RE
W																
Reset	1	0	1	--											0	1

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					TR	TT	LI	TQ		RS	PM	TI	IE	AS	LS	LD
W																
Reset	--				0	0	--	--	--	0	0	0	0	--	1	0

Figure 11.11: SpaceWire Control Register

Table 11.10: Description of SpaceWire Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	RA	1	RMAP Available 0: RMAP unavailable 1: Set to one if the RMAP command handler is available Read=0; Write=don't care
30	RX	0	RX Unaligned Access 0: Unaligned writes not available for the receiver 1: Unaligned writes available for the receiver Read=0; Write=don't care
29	RC	1	RMAP CRC Available 0: RMAP CRC not available 1: RMAP CRC available Read=0; Write=don't care.
28-18	RESERVED	[--…-]	
17	RD	0	RMAP Buffer Disable 0: All RMAP buffers are used 1: Only one RMAP buffer is used. This ensures that all RMAP commands are executed consecutively.
16	RE	1	RMAP Enable 0: Disable RMAP command handler 1: Enable RMAP command handler
15-12	RESERVED	--	
11	TR	0	Time RX Enable 0: Disable time-code receptions 1: Enable time-code receptions
10	TT	0	Time TX Enable 0: Disable time-code transmissions 1: Enable time-code transmissions
9	LI	--	Link Error IRQ 0: No interrupt 1: Generate interrupt when a link error occurs. Not reset.
8	TQ	--	Tick-Out IRQ 0: No interrupt 1: Generate interrupt when a valid time-code is received. Not reset.
7	RESERVED	--	
6	RS	--	Reset 0: No action 1: Make complete reset of the SpaceWire node. Self-clearing.
5	PM	0	Promiscuous Mode 0: Disable 1: Enable
4	TI	0	Tick In The host can generate a tick by writing a one to this field. This increment the timer counter and the new value is transmitted after the current character is transferred.
3	IE	0	Interrupt Enable 0: No interrupt

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: Interrupt is generated when bits 8 or 9 is set and its corresponding event occurs.
2	AS	--	Autostart 0: No effect 1: Automatically start the link when a NULL has been received. Not reset.
1	LS	1	Link Start 0: No effect 1: Start the link, i.e., allow a transition from ready to started state.
0	LD	0	Link Disable 0: No effect 1: Disable the SpaceWire codec.

Address = 0x8000_0A04
Address = 0x8000_0B04
Address = 0x8000_0C04
Address = 0x8000_0D04

SPWSTR

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									LS[2:0]							
W																
Reset	--								000			--				

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							AP	EE	IA			PE	DE	ER	CE	T0
W																
Reset	--						0	0	0	--		0	0	0	0	0

Figure 11.12: SpaceWire Status Register

Table 11.11: Description of SpaceWire Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-24	RESERVED	--	
23-21	LS	000	Link State This field indicates the current state of the start-up sequence. 0: Error-reset 1: Error-wait 2: Ready 3: Started 4: Connecting 5: Run
20-10	RESERVED	--	

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
9	AP	0	Active port Shows the currently active port. 0: Port 0 1: Port 1 where the port numbers refer to the index number of the data and strobe signals.
8	EE	0	Early EOP/EEP Read: 0: Packet received normally 1: Packet was received with an EOP after the first byte for a non- RMAP packet or after the second byte for a RMAP packet. Write: 0: No effect 1: Clear bit
7	IA	0	Invalid Address Read: 0: Packet received normally 1: Packet was received with an invalid destination address field Write: 0: No effect 1: Clear bit
6-5	RESERVED	--	
4	PE	0	Parity Error Read: 0: Packet received normally 1: A parity error has occurred Write: 0: No effect 1: Clear bit
3	DE	0	Disconnect Error Read: 0: No disconnection error 1: A disconnection error has occurred Write: 0: No effect 1: Clear bit
2	ER	0	Escape Error Read: 0: No escape error 1: An escape error has occurred Write: 0: No effect 1: Clear bit
1	CE	0	Credit Error Read: 0: No credit error 1: A credit has occurred Write: 0: No effect 1: Clear bit
0	TO	0	Tick Out Read:

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			0: No new time count 1: A new time count value was received and is stored in the time counter field. Write: 0: No effect 1: Clear bit

Address = 0x8000_0A08
Address = 0x8000_0B08
Address = 0x8000_0C08
SPWNAR **Address = 0x8000_0D08**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	--															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨								NODE_ADDRESS[7:0]							
W	▨															
Reset	--								254							

Figure 11.13: SpaceWire Node Address Register

Table 11.12: Description of SpaceWire Node Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[--...-]	
7-0	NODE_ADDRESS	254	8-Bit Node Address Used for node identification on the SpaceWire net- work.

Address = 0x8000_0A0C
Address = 0x8000_0B0C
Address = 0x8000_0C0C
SPWCLK **Address = 0x8000_0D0C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	--															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLOCK_DIVISOR_LINK[7:0]								CLOCK_DIVISOR_RUN[7:0]							
W																

Reset	{0000,GPIO[7:4]}	{0000,GPIO[7:4]}

Figure 11.14: SpaceWire Clock Divisor Register

Table 11.13: Description of SpaceWire Clock Divisor Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	--	
15-8	CLOCK_DIVISOR_ LINK	4]	8-Bit Clock Divisor Link State Value Used for the clock-divider when the link interface is in the startup-state. Actual divisor value = (CLOCK_DIVISOR_LINK + 1).
7-0	CLOCK_DIVISOR_ RUN	4	8-Bit Clock Divisor Run State Value Used for the clock-divider when the link-interface is in the run-state. Actual divisor value = (CLOCK_DIVISOR_RUN + 1).

Address = 0x8000_0A10
Address = 0x8000_0B10
Address = 0x8000_0C10
SPWKEY **Address = 0x8000_0D10**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[--…-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									DESTINATION_KEY[7:0]							
W																
Reset	[--…-]								[00…0]							

Figure 11.15: SpaceWire Destination Key Register

Table 11.14: Description of SpaceWire Destination Key Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[--...-]	
7-0	DESTINATION_KEY	[00...0]	RMAP Destination Key

Address = 0x8000_0A14
Address = 0x8000_0B14
Address = 0x8000_0C14
SPWTIM **Address = 0x8000_0D14**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[--…-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨								TIME_CTRL[1:0]		TIME_COUNTER[5:0]					
W	▨															
Reset	[--…-]								00		[00…0]					

Figure 11.16: SpaceWire Destination Key Register

Table 11.15: Description of SpaceWire Destination Key Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[--...-]	
7-6	TIME_CTRL	00	Time Control Flags The current value of the time control flags. Sent with time-code resulting from a tick-in. Received control flags are also stored in this field.
5-0	TIME_COUNTER	[00…0]	Time Counter The counter represents the system time count. The counter is incremented each time a time code is received and decoded into a tick-out or when the host writes to the tick-in bit.

Address = 0x8000_0A18
Address = 0x8000_0B18
Address = 0x8000_0C18
SPWTDR **Address = 0x8000_0D18**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨										DISCONNECT[9:4]					
W	▨															
Reset	[--…-]										0x27					

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISCONNECT[3:0]				TIMER64[10:0]											
W																
Reset					0x140											

Figure 11.17: SpaceWire DMA Channel and Status Register

Table 11.16: Description of SpaceWire DMA Channel and Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-22	RESERVED	[--...--]	
21-12	DISCONNECT	0x27	Disconnect Time Period Used to generate the 850ns disconnect time period. The disconnect period is (DISCONNECT + 3), e.g., to get an 850ns period, the smallest number of clock cycles that is greater than or equal to 850ns should be calculated, and this values - 3 should be stored in the field.
11-0	TIMER64	0x140	Timer 6.4us and 12.8us Used to generate the 6.4us and 12.8us time periods. Should be set to the smallest number of clock cycles that is greater than or equal to 6.4us.

Address = 0x8000_0A20
Address = 0x8000_0B20
Address = 0x8000_0C20
SPWCHN **Address = 0x8000_0D20**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																LE
W																
Reset	[--…-]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				NS	RD	RX	AT	RA	TA	PR	PS	AI	RI	TI	RE	TE
W																
Reset	--			0	0	0	0	0	0	0	0	--	--	--	0	0

Figure 11.18: SpaceWire DMA Channel and Status Register

Table 11.17: Description of SpaceWire DMA Channel and Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-17	RESERVED	[00...0]	
16	LE	0	Link error disable Disable transmitter when a link error occurs. No more packets are transmitted until the transmitter is enabled again.
15-13	RESERVED	[00...0]	
12	NS	0	No Spill 0: If cleared, packets are discarded when a packet is arriving and there are no active descriptors. 1: If set, the GRSPW2 waits for a descriptor to be activated
11	RD	0	RX Descriptors Available Read:

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			0: Cleared by GRSPW2 when it encounters a disabled descriptor. 1: Indicates enabled descriptors in the descriptor table. Write: 0: No active descriptors in the descriptor table. 1: Set to one to indicate to the GRSPW2 that there are enabled descriptors in the descriptor table.
10	RX	0	RX Active 0: No DMA channel activity 1: Set if a reception to the DMA channel is currently active. (Read Only)
9	AT	0	Abort TX 0: No effect 1: Setting the bit aborts the currently transmitting packet and disables transmissions. If no transmission is active, the only effect is to disable transmissions. Self-clearing.
8	RA	0	RX AHB Error 0: No error response 1: An error response was detected on the AHB bus while this receive DMA channel was accessing the bus. Cleared when written with a one.
7	TA	0	TX AHB Error 0: No error response 1: An error response was detected on the AHB bus while this transmit DMA channel was accessing the bus. Cleared when written with a one.
6	PR	0	Packet Received 0: No new packet 1: This bit is set each time a packet has been received. Not self-clearing. Cleared when written with a one.
5	PS	0	Packet Sent 0: No packet sent 1: This bit is set each time a packet has been sent. Not self-clearing. Cleared when written with a one.
4	AI	-	AHB Error Interrupt 0: No interrupt will be generated 1: If set, an interrupt generates each time an AHB error occurs when this DMA channel is accessing the bus. Not reset.
3	RI	-	Receive Interrupt 0: No interrupt will be generated 1: If set, an interrupt generates each time a packet has been received. This happens if either the packet is terminated by an EEP or EOP. Not reset.
2	TI	-	Transmit Interrupt 0: No interrupt generates 1: If set, an interrupt generates each time a packet is

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			transmitted. The interrupt is generated regardless of whether the transmission was successful or not. Not reset.
1	RE	0	Receiver Enable 0: Channel is not allowed to receive packets 1: Channel is allowed to receive packets
0	TE	0	Transmitter Enable Read: 0: Cleared by GRSPW2 when it encounters a disabled descriptor. 1: Indicates enabled descriptors in the descriptor table. Write: 0: No active descriptors in the descriptor table. 1: Set to one to indicate to the GRSPW2 that there are enabled descriptors in the descriptor table. Writing a one will cause the GRSPW2 to read a new descriptor and try to transmit the packet to which it points.

Address = 0x8000_0A24
Address = 0x8000_0B24
Address = 0x8000_0C24
SPWRXL **Address = 0x8000_0D24**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨							RX_MAX_LENGTH[24:16]								
W	▨							RX_MAX_LENGTH[24:16]								
Reset	[--...-]							[--...-]								

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RX_MAX_LENGTH[15:0]															
W	RX_MAX_LENGTH[15:0]															
Reset	[--...-]															

Figure 11.19: SpaceWire DMA Channel Receiver Max Length Register

Table 11.18: Description of SpaceWire DMA Channel Receiver Max Length Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-25	RESERVED	[--...-]	
24-0	RX_MAX_LENGTH	[--...-]	Receiver Packet Maximum Length The maximum number of bytes in a packet that may be received. Only bits 24-2 are writable. Bits 1-0 always read 0. Not reset.

Address = 0x8000_0A28
Address = 0x8000_0B28
Address = 0x8000_0C28

SPWTXD **Address = 0x8000_0D28**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	TX_BASE_ADDRESS[22:7]															
Reset	[--...-]															
Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	TX_BASE_ADDRESS[6:0]						TX_DESC_SELECT[5:0]						▨			
Reset	[--...-]						[00...0]						--			

Figure 11.20: SpaceWire Transmitter Descriptor Register

Table 11.19: Description of SpaceWire Transmitter Descriptor Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	TX_BASE_ADDRESS	[--...-]	Transmitter Descriptor Table Base Address Sets the base address of the descriptor table. Not reset.
9-4	TX_DESC_SELECT	[00..0]	Transmitter Descriptor Selector This is the relative offset into the descriptor table and indicates which descriptor is currently used by the GRSPW2. For each new descriptor read, TX_DESC_SELECT increases by one and wrap to zero when the field increments to 64.
3-0	RESERVED	--	

Address = 0x8000_0A2C
Address = 0x8000_0B2C
Address = 0x8000_0C2C
SPWRXD **Address = 0x8000_0D2C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	RX_BASE_ADDRESS[21:6]															
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	RX_BASE_ADDRESS[5:0]						RX_DESC_SELECT[6:0]							▨		
Reset	[--...-]						[00...0]							--		

Figure 11.21: SpaceWire Receiver Descriptor Register

Table 11.20: Description of SpaceWire Receiver Descriptor Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	RX_BASE_ADDRES S	[--...-]	Receiver Descriptor Table Base Address Sets the base address of the descriptor table. Not reset.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
9-3	RX_DESC_SELECT	[00…0]	Receiver Descriptor Selector This is the relative offset into the descriptor table and indicates which descriptor is currently used by the GRSPW2. For each new descriptor read, RX_DESC_SELECT increases by one and wraps to zero when the field increments to 64.
2-0	RESERVED	--	

Chapter 12: **CAN 2.0 Interface**

12.1 Overview

The CAN-2.0 interfaces in UT699 are based on the CAN core from OpenCores with an AHB slave interface for accessing all CAN core registers. The CAN core is a derivative of the Philips SJA1000 and has a compatible register map with a few exceptions. Each CAN core is capable of up to 1Mb/s band rate. These exceptions are indicated in the register description tables and in **Section 12.6**. The CAN core supports both BasicCAN and PeliCAN modes. In PeliCAN mode the extended features of CAN 2.0B are supported. The mode of operation is chosen through the clock divider register.

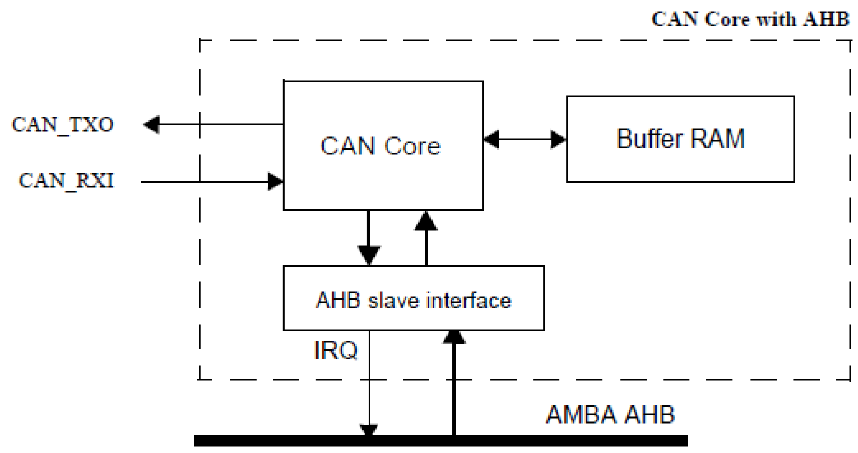


Figure 12.1: CAN Core Block Diagram

This chapter lists the CAN core registers and their functionality. The Philips SJA1000 data sheet can be used as an additional reference, except as noted in **Section 12.6**.

The register map and functionality is different depending upon which mode of operation is selected. BasicCAN mode will be described in **Section 12.3**, followed by PeliCAN in **Section 12.4**. The common registers (Clock Divisor and Bus Timing) are described in **Section 12.5**. The register map also differs depending on whether the core is in operating mode or in reset mode. After reset, the CAN core starts up in reset mode awaiting configuration. Operating mode is entered by clearing the Reset Request (CR.0) bit in the Control Register. Set the bit to re-enter reset mode.

The UT699 implements two identical instances of the OpenCores CAN core. Both operate completely independent of each other. The AHB register mapping for each core is indicated in **Table 1.2** of **Section 1.3** of this manual.

12.2 AHB Interface

All registers are one byte wide and the addresses specified in this document are byte addresses. Byte reads and writes should be used when interfacing with this core. The read byte is duplicated on all byte lanes of the AHB bus. The interface is big endian so the core expects the MSB at the lowest address.

The bit numbering in this document uses bit 7 as the MSB and bit 0 as the LSB.

12.3 BasicCAN Mode

12.3.1 BasicCAN Register Map (Address 0xFFF20000 and 0xFFF20100)

Table 12.1: BasicCAN Address Allocation

OFFSET	OPERATING MODE		RESET MODE	
	Read	Write	Read	Write
0	Control	Control	Control	Control
1	(0xFF)	Command	(0xFF)	Command
2	Status	-	Status	-
3	Interrupt	-	Interrupt	-
4	(0xFF)	-	Acceptance Code	Acceptance Code
5	(0xFF)	-	Acceptance Mask	Acceptance Mask
6	(0xFF)	-	Bus Timing 0	Bus Timing 0
7	(0xFF)	-	Bus Timing 1	Bus Timing 1
8	(0x00)	-	(0x00)	-
9	(0x00)	-	(0x00)	-
10	TX ID1	TX ID1	(0xFF)	-
11	TX ID2, rtr, dlc	TX ID2, rtr, dlc	(0xFF)	-
12	TX Data Byte 1	TX Data Byte 1	(0xFF)	-
13	TX Data Byte 2	TX Data Byte 2	(0xFF)	-
14	TX Data Byte 3	TX Data Byte 3	(0xFF)	-
15	TX Data Byte 4	TX Data Byte 4	(0xFF)	-
16	TX Data Byte 5	TX Data Byte 5	(0xFF)	-
17	TX Data Byte 6	TX Data Byte 6	(0xFF)	-
18	TX Data Byte 7	TX Data Byte 7	(0xFF)	-
19	TX Data Byte 8	TX Data Byte 8	(0xFF)	-
20	RX ID1	-	RX ID1	-
21	RX ID2, rtr, dlc	-	RX ID2, rtr, dlc	-
22	RX Data Byte 1	-	RX Data Byte 1	-
23	RX Data Byte 2	-	RX Data Byte 2	-
24	RX Data Byte 3	-	RX Data Byte 3	-
25	RX Data Byte 4	-	RX Data Byte 4	-
26	RX Data Byte 5	-	RX Data Byte 5	-
27	RX Data Byte 6	-	RX Data Byte 6	-
28	RX Data Byte 7	-	RX Data Byte 7	-
29	RX Data Byte 8	-	RX Data Byte 8	-
30	(0x00)	-	(0x00)	-
31	Clock Divider	Clock Divider	Clock Divider	Clock Divider

12.3.2 Control Register

The Control Register contains interrupt enable bits, as well as the reset request bit.

Table 12.2: Bit Interpretation of Control Register (CR), Offset 0

BIT	NAME	DESCRIPTION
CR.7	-	Reserved
CR.6	-	Reserved
CR.5	-	Reserved
CR.4	Overrun Interrupt Enable	1 - enabled, 0 - disabled
CR.3	Error Interrupt Enable	1 - enabled, 0 - disabled
CR.2	Transmit Interrupt Enable	1 - enabled, 0 - disabled
CR.1	Receive Interrupt Enable	1 - enabled, 0 - disabled
CR.0	Reset request	Writing 1 to this bit aborts any ongoing transfer and enters reset mode. Writing 0 returns to operating mode.

12.3.3 Command Register

Writing a one to the corresponding bit in this register initiates an action supported by the core.

Table 12.3: Bit Interpretation of Command Register (CMR), Offset 1

BIT	NAME	DESCRIPTION
CMR.7	-	Reserved
CMR.6	-	Reserved
CMR.5	-	Reserved
CMR.4	-	Not used (go to sleep in SJA1000 core)
CMR.3	Clear Data Overrun	Clear the Data Overrun status bit
CMR.2	Release Receive Buffer	Free the current receive buffer for new reception
CMR.1	Abort Transmission	Aborts a transmission that has not yet started
CMR.0	Transmission Request	Starts the transfer of the message in the TX buffer

A transmission is started by writing a '1' to CMR.0. It can only be aborted by writing '1' to CMR.1 and only if the transfer has not yet started. If the transmission has started it will not be aborted when setting CMR.1, but it will not be retransmitted if an error occurs.

Release the receive buffer by setting the Release Receive Buffer bit (CMR.2) after reading the contents of the receive buffer. If there is another message waiting in the FIFO, a new receive interrupt will be generated if enabled by setting the Receive Interrupt bit (IR.0), and the Receive Buffer Status (SR.0) bit will be set again. Set the Clear Data Overrun bit (CMR.3) to clear the Data overrun status bit.

12.3.4 Status Register

The status register is read only and reflects the current status of the core.

Table 12.4: Bit Interpretation of Status Register (SR), Offset 2

BIT	NAME	DESCRIPTION
SR.7	Bus Status	1 when the core is in the bus-off state and is not allowed to have any influence on the bus
SR.6	Error Status	At least one of the error counters have reached or exceeded the CPU warning limit (96)
SR.5	Transmit Status	1 when transmitting a message
SR.4	Receive Status	1 when receiving a message
SR.3	Transmission Complete	1 indicates the last message was successfully transferred.
SR.2	Transmit Buffer Status	1 means CPU can write into the transmit buffer
SR.1	Data Overrun Status	1 if a message was lost because of no space in the FIFO
SR.0	Receive Buffer Status	1 if messages are available in the receive FIFO

Receive buffer status is cleared when the Release Receive Buffer command (CMR.2) is given and is set if there are more messages available in the FIFO. The Data Overrun Status (SR.1) signals that a message that was accepted could not be placed in the receive FIFO because there was not enough space left. **Note:** This bit differs from the SJA1000 behavior and is set when the FIFO has been read out. When the Transmit Buffer Status is high, the transmit buffer can be written to by the CPU. During an on-going transmission the buffer is locked and this bit is 0. The Transmission Complete bit is cleared when a transmission request has been issued and will not be set again until a message has successfully been transmitted.

12.3.5 Interrupt Register

The interrupt register signals to the CPU what caused the interrupt. The interrupt bits are only set if the corresponding interrupt enable bit is set in the control register. The interrupt assignment for both CAN cores is shown in **Table 1.3** of Section **1.5**.

Table 12.5: Bit Interpretation of Interrupt Register (IR), Offset 3

BIT	NAME	DESCRIPTION
IR.7	-	Reserved
IR.6	-	Reserved
IR.5	-	Reserved
IR.4	-	Not used (wake-up interrupt of SJA1000)
IR.3	Data Overrun Interrupt	Set when Data Overrun Status (SR.1) transitions from 0 to 1
IR.2	Error Interrupt	Set when Error Status (SR.6) or Bus Status (SR.7) change
IR.1	Transmit Interrupt	Set when Transmit Buffer is released (SR.2 transitions from 0 to 1)
IR.0	Receive Interrupt	This bit is set while there are more messages in the FIFO

This register is reset on read with the exception of IR.0. This core resets the Receive Interrupt bit when the Release Receive Buffer command is given (as in PeliCAN mode).

NOTE: Bit IR.5 through IR.7 read '1'. Bit IR.4 reads '0'.

12.3.6 Transmit Buffer

The **Table 12.6** below shows the layout of the transmit buffer. In BasicCAN only standard frame messages can be transmitted and received. Extended Frame Format (EFF) messages on the bus are ignored.

Table 12.6: Transmit Buffer Layout

OFFSET	NAME	BITS							
		7	6	5	4	3	2	1	0
10	ID byte 1	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
11	ID byte 2	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
12	TX data 1	TX byte 1							
13	TX data 2	TX byte 2							
14	TX data 3	TX byte 3							
15	TX data 4	TX byte 4							
16	TX data 5	TX byte 5							
17	TX data 6	TX byte 6							
18	TX data 7	TX byte 7							
19	TX data 8	TX byte 8							

If the RTR bit is set no data bytes will be sent, but DLC is still part of the frame and must be specified according to the requested frame. It is possible to specify a DLC larger than eight bytes, but should not be done for compatibility reasons. If DLC is greater than 8, only 8 bytes can be sent.

12.3.7 Receive Buffer

The receive buffer on address 20 through 29 is the visible part of the 64-byte RX FIFO. Its layout is identical to that of the transmit buffer.

12.3.8 Acceptance Filter

Messages can be filtered based on their identifiers using the Acceptance Code and Acceptance Mask registers. Bits ID.10 through ID.3 of the 11-bit identifier are compared with the Acceptance Code Register. Only the bits set to '0' in the Acceptance Mask Register are used for comparison. If a match is detected, the message is stored to the FIFO.

12.4 PeliCAN Mode

12.4.1 PeliCAN Register Map (Address 0xFFF20000 and 0xFFF20100)

Table 12.7: PeliCAN Address Allocation

OFFSET	OPERATING MODE				RESET MODE	
	Read		Write		Read	Write
0	Mode		Mode		Mode	Mode
1	(0x00)		Command		(0x00)	Command
2	Status		-		Status	-
3	Interrupt		-		Interrupt	-
4	Interrupt enable		Interrupt enable		Interrupt Enable	Interrupt Enable
5	reserved (0x00)		-		reserved (0x00)	-
6	Bus Timing 0		-		Bus Timing 0	Bus Timing 0
7	Bus Timing 1		-		Bus Timing 1	Bus Timing 1
8	(0x00)		-		(0x00)	-
9	(0x00)		-		(0x00)	-
10	(0x00)		-		(0x00)	-
11	Arbitration Lost Capture		-		Arbitration Lost Capture	-
12	Error Code Capture		-		Error Code Capture	-
13	Error Warning Limit		-		Error Warning Limit	Error Warning Limit
14	RX Error Counter		-		RX Error Counter	RX Error Counter
15	TX Error Counter		-		TX Error Counter	TX Error Counter
16	RX FI SFF	RX FI EFF	TX FI SFF	TX FI EFF	Acceptance Code 0	Acceptance Code 0
17	RX ID 1	RX ID 1	TX ID 1	TX ID 1	Acceptance Code 1	Acceptance Code 1
18	RX ID 2	RX ID 2	TX ID 2	TX ID 2	Acceptance Code 2	Acceptance Code 2
19	RX Data 1	RX ID 3	TX Data 1	TX ID 3	Acceptance Code 3	Acceptance Code 3
20	RX Data 2	RX ID 4	TX Data 2	TX ID 4	Acceptance Mask 0	Acceptance Mask 0
21	RX Data 3	RX Data 1	TX Data 3	TX Data 1	Acceptance Mask 1	Acceptance Mask 1
22	RX Data 4	RX Data 2	TX Data 4	TX Data 2	Acceptance Mask 2	Acceptance Mask 2
23	RX Data 5	RX Data 3	TX Data 5	TX Data 3	Acceptance Mask 3	Acceptance Mask 3
24	RX Data 6	RX Data 4	TX Data 6	TX Data 4	(0x00)	-
25	RX Data 7	RX Data 5	TX Data 7	TX Data 5	(0x00)	-
26	RX Data 8	RX Data 6	TX Data 8	TX Data 6	(0x00)	-
27	FIFO	RX Data 7	-	TX Data 7	(0x00)	-
28	FIFO	RX Data 8	-	TX Data 8	(0x00)	-
29	RX Message Counter		-		RX Message Counter	-
30	(0x00)		-		(0x00)	-

OFFSET	OPERATING MODE		RESET MODE	
	Read	Write	Read	Write
31	Clock Divider	Clock Divider	Clock Divider	Clock Divider

The transmit and receive buffers have a different layout depending on if standard frame format (SFF) or extended frame format (EFF) is to be transmitted or received.

12.4.2 Mode Register

Table 12.8: Bit Interpretation of Mode Register (MOD), Offset 0

BIT	NAME	DESCRIPTION
MOD.7	-	Reserved
MOD.6	-	Reserved
MOD.5	-	Reserved
MOD.4	-	Not used (sleep mode in SJA1000)
MOD.3	Acceptance Filter Mode	1 - single filter mode, 0 - dual filter mode
MOD.2	Self-Test Mode	Set if the controller is in self-test mode
MOD.1	Listen-Only Mode	Set if the controller is in listen-only mode
MOD.0	Reset Mode	Writing 1 to this bit aborts any ongoing transfer and enters reset mode. Writing 0 returns to operating mode

Writing to MOD.1-3 can only be done when reset mode has been previously entered. In listen-only mode, the core will not send any acknowledgements.

When in self-test mode, the core can complete a successful transmission without getting an acknowledgement if given the Self Reception Request command (CMR.4). The core must still be connected to a real bus as it does not do an internal roll-back.

12.4.3 Command Register

Writing a '1' to the corresponding bit in this register initiates an action supported by the core.

Table 12.9: Bit Interpretation of Command Register (CMR), Offset 1

BIT	NAME	DESCRIPTION
MOD.7	-	Reserved
MOD.6	-	Reserved
MOD.5	-	Reserved
MOD.4	-	Not used (sleep mode in SJA1000)
MOD.3	Acceptance Filter Mode	1 - single filter mode, 0 - dual filter mode
MOD.2	Self-Test Mode	Set if the controller is in self-test mode
MOD.1	Listen-Only Mode	Set if the controller is in listen-only mode
MOD.0	Reset Mode	Writing 1 to this bit aborts any ongoing transfer and enters reset mode. Writing 0 returns to operating mode

A transmission is started by setting CMR.0. It can only be aborted by setting CMR.1 and only if the transfer has not yet started. Setting CMR.0 and CMR.1 simultaneously will result in a so-called single shot transfer, i.e., the core will not try to retransmit the message if unsuccessful the first time.

Giving the Release Receive Buffer command (CMR.2) should be done after reading the contents of the receive buffer in order to release the memory. If there is another message waiting in the FIFO, a new Receive Interrupt (IR.0) will be generated (if enabled) and the Receive Buffer Status bit (SR.0) will be set again.

The Self Reception Request bit (CMR.4) together with the self-test mode makes it possible to do a self-test of the core without any other cores on the bus. A message will simultaneously be transmitted and received and both receive and transmit interrupts will be generated

12.4.4 Status Register

The status register is read only and reflects the current status of the core.

Table 12.10: Bit Interpretation of Status Register (SR), Offset 2

BIT	NAME	DESCRIPTION
SR.7	Bus Status	1 when the core is in bus-off and not involved in bus activities
SR.6	Error Status	At least one of the error counters have reached or exceeded the error warning limit.
SR.5	Transmit Status	1 when transmitting a message
SR.4	Receive Status	1 when receiving a message
SR.3	Transmission Complete	1 indicates the last message was successfully transferred
SR.2	Transmit Buffer Status	1 means CPU can write into the transmit buffer
SR.1	Data Overrun Status	1 if a message was lost because no space in FIFO
SR.0	Receive Buffer Status	1 if messages available in the receive FIFO

Receive Buffer Status (SR.0) is cleared when there are no more messages in the receive FIFO. The Data Overrun Status (SR.1) signals that a message that was accepted could not be placed in the FIFO because there was not enough space left.

NOTE: This bit differs from the SJA1000 behavior and is set first when the FIFO has been read out.

When the Transmit Buffer Status (SR.2) is high the transmit buffer is available to be written to by the CPU. During an on- going transmission the buffer is locked and this bit is '0'.

The Transmission Complete bit (SR.3) is cleared when a Transmission Request (CMR0) or Self Reception Request (CMR.4) has been issued and will not be set again until a message has successfully been transmitted.

12.4.5 Interrupt Register

The Interrupt Register signals to CPU what caused an interrupt. The interrupt bits are only set if the corresponding interrupt enable bit is set in the Interrupt Enable Register. The interrupt assignment for both CAN cores is shown in **Table 1.3** of **Section 1.4**. This register is reset on read with the exception of IR.0 which is reset when the FIFO has been emptied.

Table 12.11: Bit Interpretation of Interrupt Register (IR), Offset 3

BIT	NAME	DESCRIPTION
IR.7	Bus Error Interrupt	Set if an error on the bus has been detected
IR.6	Arbitration Lost Interrupt	Set when the core has lost arbitration
IR.5	Error Passive Interrupt	Set when the core goes between error active and error passive
IR.4	-	Not used (wake-up interrupt of SJA1000)
IR.3	Data Overrun Interrupt	Set when Data Overrun Status bit is set
IR.2	Error Warning Interrupt	Set on every change of the error status or bus status
IR.1	Transmit Interrupt	Set when the transmit buffer is released
IR.0	Receive Interrupt	Set while the receive FIFO is not empty.

12.4.6 Interrupt Enable Register

Interrupts sources can be enabled or disabled in the Interrupt Enable Register. If a bit is enabled, the corresponding interrupt can be generated.

Table 12.12: Bit Interpretation of Interrupt Enable Register (IER), Offset 4

BIT	NAME	DESCRIPTION
IR.7	Bus Error Interrupt	1 - enabled, 0 - disabled
IR.6	Arbitration Lost Interrupt	1 - enabled, 0 - disabled
IR.5	Error Passive Interrupt	1 - enabled, 0 - disabled
IR.4	-	Not used (wake-up interrupt of SJA1000)
IR.3	Data Overrun Interrupt	1 - enabled, 0 - disabled
IR.2	Error Warning Interrupt	1 - enabled, 0 - disabled.
IR.1	Transmit Interrupt	1 - enabled, 0 - disabled
IR.0	Receive Interrupt	1 - enabled, 0 - disabled

12.4.7 Arbitration Lost Capture Register

Interrupts sources can be enabled or disabled in the Interrupt Enable Register. If a bit is enabled, the corresponding interrupt can be generated.

Table 12.13: Bit Interpretation of Arbitration Lost Capture Register (ALC), Offset 11

BIT	NAME	DESCRIPTION
ALC.7-5	-	Reserved
ALC.4-0	Bit number	Bit where arbitration was lost

When the core loses arbitration the bit position of the bit stream processor is captured into arbitration lost capture register. The register will not change content again until read out.

12.4.8 Error Code Capture Register

Table 12.14: Bit Interpretation of Error Code Capture Register (ECC), Offset 12

BIT	NAME	DESCRIPTION
ECC.7-6	Error code	Error code number
ECC.5	Direction	1 - Reception, 0 - transmission error
ECC.4-0	Segment	Location in frame where error occurred

When a bus error occurs, the Error Code Capture Register is set according to the type of error that occurred, i.e., if it occurred while transmitting or receiving, and where in the frame it occurred. As with the ALC register, the ECC register will not change value until it has been read out. **Table 12.15** shows how to interpret bit ECC.7-6.

Table 12.15: Error Code Interpretation

ECC.7-6	DESCRIPTION
0	Bit error
1	Form error
2	Stuff error
3	Other

Table 12.16 below indicates how to interpret ECC.4-0.

Table 12.16: Bit Interpretation of ECC[4:0] Code Interpretation

ECC.4-0	DESCRIPTION
0x03	Start of frame
0x02	ID.28 - ID.21
0x06	ID.20 - ID.18
0x04	Bit SRTR
0x05	Bit IDE
0x07	ID.17 - ID.13
0x0F	ID.12 - ID.5
0x0E	ID.4 - ID.0
0x0C	Bit RTR
0x0D	Reserved bit 1
0x09	Reserved bit 0
0x0B	Data length code
0x0A	Data field
0x08	CRC sequence
0x18	CRC delimiter
0x19	Acknowledge slot
0x1B	Acknowledge delimiter

ECC.4-0	DESCRIPTION
0x1A	End of frame
0x12	Intermission
0x11	Active error flag
0x16	Passive error flag
0x13	Tolerate dominant bits
0x17	Error delimiter
0x1C	Overload flag

12.4.9 Error Warning Limit Register

This register allows for setting the CPU error warning limit. The default is 96.

Note: This register is only writable in reset mode.

12.4.10 RX Error Counter Register, Offset 14

This register shows the value of the RX error counter. It is writable in reset mode. A bus-off event resets this counter to 0.

12.4.11 TX Error Counter Register, Offset 15

This register shows the value of the TX error counter. It is writable in reset mode. If a bus-off event occurs, this register is configured to count down the protocol-defined 128 occurrences of the bus-free signal. The status of the bus-off recovery can be read out from this register. The CPU can force a bus-off by writing 255 to this register. Unlike the SJA1000, this core signals bus-off immediately, not initially when entering operating mode. The bus-off recovery sequence starts when entering operating mode after writing 255 to this register in reset mode.

12.4.12 Transmit Buffer

The transmit buffer is write-only and is mapped to address 16 to 28. Reading of this area is mapped to the same address offset as the receive buffer described in the Section **12.4.13**. The layout of the transmit buffer depends on whether a standard frame (SFF) or an extended frame (EFF) is to be sent as seen in **Table 12.17**.

Table 12.17: Transmit Buffer Layout

OFFSET	WRITE (SFF)	WRITE (EFF)
16	TX Frame Information	TX Frame Information
17	TX ID 1	TX ID 1
18	TX ID 2	TX ID 2
19	TX Data1	TX ID 3
20	TX Data 2	TX ID 4
21	TX Data 3	TX Data 1
22	TX Data 4	TX Data 2
23	TX Data 5	TX Data 3
24	TX Data 6	TX Data 4
25	TX Data 7	TX Data 5
26	TX Data 8	TX Data 6

OFFSET	WRITE (SFF)	WRITE (EFF)
27	-	TX Data 7
28	-	TX Data 8

- **TX Frame Information**

This field has the same layout for both SFF and EFF frames

Table 12.18: Description of TX Frame Information, Offset 16

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7	FF	-	FF selects the frame format, i.e., whether this is to be interpreted as an extended or standard frame. 1 = EFF, 0 = SFF.
6	RTR	0	RTR should be set to 1 for a Remote Transmission Request frame.
5-4	--	00	Don't care
3-0	DLC[3:0]	0000	DLC specifies the Data Length Code and should have a value between 0 and 8. If the value is greater than 8, only 8 bytes will be transmitted.

- **TX Identifier 1**

This field has the same layout for both SFF and EFF frames

Table 12.19: Description of TX Identifier 1, Offset 17

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[28:21]	[00…0]	The top eight bits of the identifier.

- **TX Identifier 2, SFF Frame**

Table 12.20: Description of TX Identifier 2, Offset 18

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-5	ID[20:18]	000	Bottom three bits of an SFF identifier.
4-0	--	[00…0]	Don't care

- **TX Identifier 2, EFF Frame**

Table 12.21: Description of TX Identifier 2, Offset 18

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[20:13]	[00...0]	Bit 20:13 of 29 bit EFF identifier.

- **TX Identifier 3, EFF Frame**

Table 12.22: Description of TX Identifier 3, Offset 19

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[12:5]	[00...0]	Bit 12:5 of 29 bit EFF identifier.

- **TX Identifier 4, EFF Frame**

Table 12.23: Description of TX Identifier 4, Offset 20

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-3	ID[4:0]	00000	Bit 4:0 of 29 bit EFF identifier.
2-0	--	000	Don't care

Data field

The data field is located at offset 19 to 26 for SFF frames and at offset 21 to 28 for EFF frames. The data is transmitted starting from the MSB at the lowest address.

12.4.13 Receiver Buffer

Table 12.24: Receive Buffer Layout

OFFSET	WRITE (SFF)	WRITE (EFF)
16	RX Frame Information	RX Frame Information
17	RX ID 1	RX ID 1
18	RX ID 2	RX ID 2
19	RX Data 1	RX ID 3
20	RX Data 2	RX ID 4
21	RX Data 3	RX Data 1
22	RX Data 4	RX Data 2
23	RX Data 5	RX Data 3
24	RX Data 6	RX Data 4
25	RX Data 7	RX Data 5
26	RX Data 8	RX Data 6
27	RX FI of next message in FIFO	RX Data 7

OFFSET	WRITE (SFF)	WRITE (EFF)
28	RX ID1 of next message in FIFO	RX Data 8

- **RX Frame Information**

Table 12.25: Description of RX Information, Offset 16

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7	FF	0	Frame format of received message. 1 = EFF, 0 = SFF.
6	RTR	0	1 if RTR frame.
5-4	RESERVED	00	
3-0	DLC[3:0]	0000	DLC specifies the Data Length Code.

- **RX Identifier 1**

This field has the same layout for both SFF and EFF frames

Table 12.26: Description of RX Identifier 1, Offset 17

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[28:21]	[00…0]	The top eight bits of the identifier.

- **RX Identifier 2, SFF Frame**

Table 12.27: Description of RX Identifier 2, Offset 18

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-5	ID[20:18]	000	Bottom three bits of an SFF identifier.
4	RTR	0	1 if RTR frame.
3-0	RESERVED	0000	

- **RX Identifier 2, EFF Frame**

Table 12.28: Description of RX Identifier 2, Offset 18

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[20:13]	[00…0]	Bit 20:13 of 29 bit EFF identifier.

- **RX Identifier 3, EFF Frame**

Table 12.29: Description of RX Identifier 3, Offset 19

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-0	ID[12:5]	[00…0]	Bit 12:5 of 29 bit EFF identifier.

- **RX Identifier 4, EFF Frame**

Table 12.30: Description of RX Identifier 4, Offset 20

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
7-3	ID[4:0]	[00…0]	Bit 4:0 of 29 bit EFF identifier
2	RTR	0	1 if RTR frame
1-0	RESERVED	00	

Data field

The data field is located at offset 19 to 26 for SFF frames and at offset 21 to 28 for EFF frames.

12.4.14 Acceptance Filter

The acceptance filter can be used to filter out messages not meeting certain demands. If a message is filtered out, it will not be put into the receive FIFO and the CPU will not have to process it.

There are two different filtering modes: Single filter mode and dual filter mode. The mode is selected by the Acceptance Filter Mode bit (MOD.3) in the Mode Register. In single filter mode, a single filter is used. In dual filter, two smaller filters are used. If there is a match with either filter, the message is accepted. Each filter consists of an acceptance code and an acceptance mask. The Acceptance Code registers are used for specifying the pattern to match, and the Acceptance Mask registers specify which bits to use for comparison. In total, eight registers are used for the acceptance filters as shown in the following **Table 12.31**.

NOTE: The registers are only read/writable in reset mode.

Table 12.31: Acceptance Filter Register

OFFSET	DESCRIPTION
16	Acceptance Code 0 (ACR0)
17	Acceptance Code 1 (ACR1)
18	Acceptance Code 2 (ACR2)
19	Acceptance Code 3 (ACR3)
20	Acceptance Mask 0 (AMR0)
21	Acceptance Mask 1 (AMR1)

OFFSET	DESCRIPTION
22	Acceptance Mask 2 (AMR2)
23	Acceptance Mask 3 (AMR3)

- **Single filter mode, standard frame**

When receiving a standard frame in single filter mode the registers ACR0-3 are compared against the incoming message in the following way:

- ACR0.7-0 & ACR1.7-5 are compared to ID.28-18 ACR1.4 is compared to the RTR bit.
- ACR1.3-0 are unused.
- ACR2 & ACR3 are compared to data byte 1 & 2.

The corresponding bits in the AMR registers select which bits are used for comparison. A set bit in the mask register means don't care.

- **Single filter mode, extended frame**

When receiving an extended frame in single filter mode the registers ACR0-3 are compared against the incoming message in the following way:

- ACR0.7-0 & ACR1.7-0 are compared to ID.28-13 ACR2.7-0 & ACR3.7-3 are compared to ID.12-0 ACR3.2 are compared to the RTR bit
- ACR3.1-0 are unused.

The corresponding bits in the AMR registers select which bits are used for comparison. A set bit in the mask register means don't care.

- **Dual filter mode, standard frame**

When receiving a standard frame in dual filter mode the registers ACR0-3 are compared against the incoming message in the following way:

Filter 1

ACR0.7-0 & ACR1.7-5 are compared to ID.28-18 ACR1.4 is compared to the RTR bit.

ACR1.3-0 are compared against upper nibble of data byte 1 ACR3.3-0 are compared against lower nibble of data byte 1.

Filter 2

ACR2.7-0 & ACR3.7-5 are compared to ID.28-18 ACR3.4 is compared to the RTR bit.

The corresponding bits in the AMR registers select which bits are used for comparison. A set bit in the mask register means don't care.

- **Dual filter mode, extended frame**

When receiving a standard frame in dual filter mode the registers ACR0-3 are compared against the incoming message in the following way:

Filter 1

ACR0.7-0 & ACR1.7-0 are compared to ID.28-13

Filter 2

ACR2.7-0 & ACR3.7-0 are compared to ID.28-13

The corresponding bits in the AMR registers select which bits are used for comparison. A set bit in the mask register means don't care.

12.4.15 RX Message Counter

The RX message counter register at address 29 holds the number of messages currently stored in the receive FIFO. The top three bits are always 0.

12.5 Common Registers

There are three common registers that are at the same addresses and have the same functionality in both BasicCAN and PeliCAN mode. These are the Clock Divider Register and Bit Timing Registers 0 and 1.

12.5.1 Clock Divider Register

The only function of this register in the GRLIB version of the OpenCores CAN is to choose between BasicCAN and PeliCAN.

Table 12.32: Bit Interpretation of Clock Divider Register (CDR), Offset 31

OFFSET		DESCRIPTION
CDR.7	CAN mode	1 - PeliCAN, 0 - BasicCAN
CDR.6	-	Unused
CDR.5	-	Unused
CDR.4	-	Reserved
CDR.3	Clock Off	Disable the clkout output
CDR.2-0	Clock Divisor	Frequency selector

12.5.2 Bus Timing 0

Table 12.33: Bit Interpretation of Bus Timing 0 Register (BTR0), Offset 6

BIT	NAME	DESCRIPTION
BTR0.7-6	SJW	Synchronization jump width
BTR0.5-0	BRP	Baud rate prescaler

The CAN core system clock is calculated as:

$t_{scl} = 2*t_{clk}*(BRP+1)$

where t_{clk} is the system clock.

The sync jump width defines how many clock cycles (t_{scl}) a bit period may be adjusted with by one re-synchronization.

12.5.3 Bus Timing 1

Table 12.34: Bit Interpretation of Bus Timing 1 Register (BTR1), Offset 7

BIT	NAME	DESCRIPTION
BTR1.7	SAM	1 - The bus is sampled three times, 0 - single sample point
BTR1.6-4	TSEG2	Time segment 2
BTR1.3-0	TSEG1	Time segment 1

The CAN bus bit period is determined by the CAN system clock and time segment 1 and 2 as shown in the equations below:

$t_{tseg1} = t_{scl} * (TSEG1+1)$
$t_{tseg2} = t_{scl} * (TSEG2+1)$
$t_{bit} = t_{tseg1} + t_{tseg2} + t_{scl}$

The additional t_{scl} term comes from the initial sync segment. Sampling is done between TSEG1 and TSEG2 in the bit period.

12.6 CAN-OC vs SJA1000

There are three common registers that are at the same addresses and have the same functionality in both BasicCAN and PeliCAN mode. These are the Clock Divider Register and Bit

This section lists the known differences between this CAN controller and the SJA1000 on which is it based.

- All bits related to sleep mode are unavailable
- Output control and test registers do not exist (reads 0x00)
- Clock divisor register bit 6 (CBP) and 5 (RXINTEN) are not implemented
- Data overrun IRQ and status not set until FIFO is read out

BasicCAN specific differences:

- The receive IRQ bit is not reset on read (works like in PeliCAN mode)
- Bit CR.6 always reads 0 and is not a flip-flop with no effect as in the SJA1000
- Bit IRQ is not reset on read as in SJA1000
- Does not become error passive and active error frames are still sent.

PeliCAN specific differences:

- Writing 256 to TX error counter gives immediate bus-off when still in reset mode
- Read Buffer Start Address register does not exist
- Addresses above 31 are not implemented (i.e., the internal RAM/FIFO access)
- The core transmits active error frames in Listen only mode

Chapter 13: Ethernet Media Access Controller (MAC) with EDCL Support

13.1 Overview

Cobham’s Gaisler Ethernet Media Access Controller (GRETH) provides an interface between an AMBA AHB bus and an Ethernet network. It supports 10/100 Mbit speed in both full- and half-duplex modes. The AMBA interface consists of an APB interface for configuration and control and an AHB master interface that handles the dataflow. The dataflow is handled through DMA channels. There is one DMA engine for the transmitter and one for the receiver. Both share the same AHB master interface. The Ethernet interface supports the MII interface, which should be connected to an external PHY. The GRETH also provides access to the MII management interface which is used to configure the PHY.

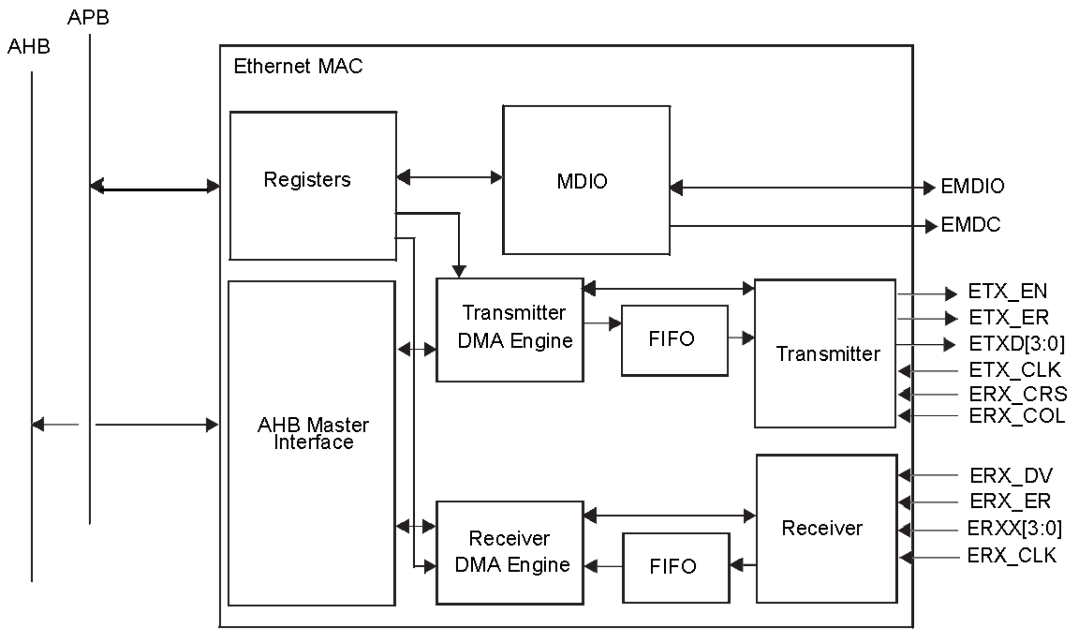


Figure 13.1: Block Diagram of the Internal Structure of the GRETH

NOTE: Ethernet Interface operation is intended for terrestrial use only, and not guaranteed in radiation environments.

13.2 Operation

13.2.1 System Overview

The GRETH consists of two functional units: The DMA channels and the MDIO interface.

The main functionality consists of the DMA channels, which are used to transfer data between an AHB bus and an Ethernet network. There is one transmitter DMA channel and one Receiver DMA channel. The operation of the DMA channels is con- trolled through registers accessible through the APB interface.

The MDIO interface is used for accessing configuration and status registers in one or more PHYs connected to the MAC. The operation of this interface is also controlled through the APB interface.

The Media Independent Interface (MII) is used for communicating with the PHY. There is an Ethernet transmitter which sends all data from the AHB domain on the Ethernet using the MII interface. Correspondingly, there is an Ethernet receiver which stores all data from the Ethernet on the AHB bus. Both of these interfaces use FIFOs when transferring the data streams.

13.2.2 Protocol Support

The GRETH is implemented according to IEEE standard 802.3-2002. There is no support for the optional control sublayer and no multicast addresses can be assigned to the MAC. This means that packets with type 0x8808 (the only currently defined control packets) are discarded.

13.2.3 Hardware Requirements

There are three clock domains: The AHB clock, the Ethernet receiver clock and the Ethernet transmitter clock. Both full-duplex and half-duplex operating modes are supported. The system frequency requirement (SYSCLK) is 2.5 MHz for up to 10 Mbit/s and 25 MHz for up to 100 Mbit/s operations.

13.2.4 Transmitter DMA Interface

The transmitter DMA interface is used for transmitting data on an Ethernet network. The transmission is done using descriptors located in memory.

13.2.5 Setting up a Descriptor

A single descriptor is shown in **Figure 13.2** and **Figure 13.3**. The number of bytes to be sent should be set in the Length field and the Address field should point to the data. The address must be word-aligned. If the Interrupt Enable (IE) bit is set, an interrupt will be generated when the packet has been sent (this requires that the Transmitter Interrupt (TI) bit in the Control Register also be set). The interrupt will be generated regardless of whether the packet was transmitted success- fully or not. The Wrap (WR) bit is also a control bit that should be set before transmission and it will be explained later in this section.

ETHTDW0 **Offset = 0x00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨													LE	▨	
W																
Reset	[00…0]													0	00	

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AL	UE	IE	WR	EN	LENGTH[10:0]										
W																
Reset	0	0	0	0	0	[00…0]										

Figure 13.2: Ethernet Transmitter Descriptor Word 0

Table 13.1: Description of Ethernet Transmitter Descriptor Word 0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-19	RESERVED	[00...0]	
18	LE	0	Length Error The length/type field of the packet did not match the actual number of received bytes. Set if the packet was not transmitted because the maximum number of attempts was reached.
16-15	RESERVED	00	Attempt Limit Error Set if the packet was not transmitted because the maximum number of attempts was reached.
14	UE	0	Underrun Error Set if the packet was incorrectly transmitted due to a FIFO under- run error.
13	IE	0	Interrupt Enable An interrupt will be generated when the packet from this descriptor has been sent provided that the Transmitter Interrupt (TI) enable bit in the Control Register is set. The interrupt is generated regardless if the packet was transmitted successfully or if it terminated with an error. 0: Disable interrupts 1: Enable interrupts
12	WR	0	Wrap Set to one to make the descriptor pointer wrap to zero after this descriptor has been used. If this bit is not set the pointer increments by 8. The pointer automatically wraps to zero when the 1 KB boundary of the descriptor table is reached. 0: Wrap disabled 1: Wrap enabled
11	EN	0	Enable Set to one to enable the descriptor. Should always be the last bit set in the descriptor fields. 0: Descriptor disabled 1: Descriptor enabled
10-0	LENGTH	[00...0]	The number of bytes to be transmitted.

ETHTDW1 **Offset = 0x04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDRESS[29:14]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R	ADDRESS[13:2]	
W		
Reset	[00…0]	00

Figure 13.3: Ethernet Transmitter Descriptor Word 1

Table 13.2: Description of Ethernet Transmitter Descriptor Word 1

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	ADDRESS	[00...0]	Pointer to the buffer area from where the packet data will be loaded.
1-0	RESERVED	00	Read=00b; Write=don't care.

To enable a descriptor the Enable (EN) bit must be set. After a descriptor is enabled, it should not be modified until the Enable bit has been cleared.

13.2.6 Starting Transmissions

Enabling a descriptor is not enough to start a transmission. A pointer to the memory area holding the descriptors must first be set in the GRETH. This is done in the Transmitter Descriptor Pointer Register. The address must be aligned to a 1 KB boundary. Bits 31:10 hold the base address of descriptor area, while bits 9:3 form a pointer to an individual descriptor. The first descriptor should be located at the base address and when it has been used by the GRETH, the Descriptor Pointer field is incremented by eight to point to the next descriptor. The pointer automatically wraps back to zero after the last 1 KB boundary has been reached at address offset 0x3F8. The Wrap (WR) bit in the descriptors can be set to make the pointer wrap back to zero before the 1 KB boundary.

The Descriptor Pointer field has also been made writable for maximum flexibility. However, care should be taken when writing to the Descriptor Pointer Register. It should never be modified when a transmission is active. The final step to activate the transmission is to set the Transmit Enable (TE) bit in the control register. This tells the GRETH there are active descriptors in the descriptor table. This bit should always be set when new descriptors are enabled, even if transmissions are already active. The descriptors must always be enabled before the Transmit Enable bit is set.

13.2.7 Descriptor Handling After Transmission

When a transmission of a packet has finished, status is written to the first word in the corresponding descriptor. The Underrun Error (UE) bit is set if the FIFO became empty before the packet was completely transmitted. The Attempt Limit Error (AL) bit is set if more collisions occurred than allowed. The packet was successfully transmitted only if both of these bits are zero. The other bits in the first descriptor word are set to zero after transmission while the second word is left untouched.

The Enable bit should be used as the indicator when a descriptor can be used again, which is when it has been cleared by the GRETH. There are three bits in the GRETH status register that hold transmission status. The Transmitter Error (TE) bit is set each time a transmission ended with an error (when at least one of the two status bits in the transmitter descriptor has been set). The Transmitter Interrupt (TI) is set each time a transmission ended successfully. The Transmitter AHB Error (TA) bit is set when an AHB error was encountered either when reading a descriptor or when reading packet data. Any active transmissions are aborted and the transmitter is disabled. The transmitter can be activated again by setting the Transmit Enable bit in the Control Register.

13.2.8 Setting up the Data for Transmission

The data to be transmitted should be placed beginning at the address pointed by the descriptor Address field of the transmitter descriptor. The GRETH does not add the Ethernet address and type fields, so they must also be stored in the data buffer. The 4-bytes Ethernet CRC is automatically appended to the end of each packet. Each descriptor will be sent as a single Ethernet packet. If the size field in a descriptor is greater than 1514 byte, the packet will not be sent.

13.2.9 Receiver DMA Interface

The receiver DMA interface is used for receiving data from an Ethernet network. The reception is done using descriptors located in memory.

13.2.10 Setting up Descriptors

A single descriptor is shown in **Figure 13.4** and **Figure 13.5**. The address field should point to a word-aligned buffer where the received data is to be stored. The GRETH never stores more than 1514 byte to the buffer. If the Interrupt Enable (IE) bit is set, an interrupt will be generated when a packet has been received to this buffer (this requires that the Receiver Interrupt (RI) bit in the Control Register be set). The interrupt will be generated regardless of whether the packet was received successfully or not. The Wrap (WR) bit is also a control bit that should be set before the descriptor is enabled and it will be explained later in this section.

ETHRDW0 **Offset = 0x00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R															OE	CE
W																
Reset	[00…0]														0	0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FT	AE	IE	WR	EN	LENGTH[10:0]										
W																
Reset	0	0	0	0	0	[00…0]										

Figure 13.4: Ethernet Receiver Descriptor Word 0

Table 13.3: Description of Ethernet Receiver Descriptor Word 0

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-19	RESERVED	[00...0]	
17	OE	0	Overrun Error Set if the frame was incorrectly received due to a FIFO over- run.
16	CE	0	CRC Error Set if a CRC error was detected in this frame.
15	FT	0	Frame Too Long Set if a frame larger than the maximum size was received. The excessive part was truncated.
14	AE	0	Alignment Error Set if an odd number of nibbles were received.
13	IE	0	Interrupt Enable An interrupt will be generated when a packet has been

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			received to this descriptor provided the Receiver Interrupt (RI) enable bit in the Control Register is set. The interrupt is generated regardless if the packet was received successfully or if it terminated with an error. 0: Interrupts disabled 1: Interrupts enabled
12	WR	0	Set to one to make the descriptor pointer wrap to zero after this descriptor has been used. If this bit is not set, the pointer increments by 8. The pointer automatically wraps to zero when the 1 KB boundary of the descriptor table is reached. 0: Wrap disabled 1: Wrap enabled
11	EN	0	Enable Set to one to enable the descriptor. Should always be set last of all the descriptor fields. 0: Descriptor disabled 1: Descriptor enabled
10-0	LENGTH	[00...0]	The number of bytes received by this descriptor.

ETHRDW1 **Offset = 0x04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDRESS[29:14]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS[13:0]														▨	
W																
Reset	[00…0]														00	

Figure 13.5: Ethernet Receiver Descriptor Word 1

Table 13.4: Description of Ethernet Receiver Descriptor Word 1

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	ADDRESS	[00...0]	Pointer to the buffer area from where the packet data will be loaded.
1-0	RESERVED	00	Read=00b; Write=don't care.

13.2.11 Starting Reception

Enabling a descriptor is not enough to start reception. A pointer to the memory area holding the descriptors must first be set in the GRETH. This is done in the Receiver Descriptor Pointer Register. The address must be aligned to a 1 KB boundary. Bits 31:10 hold the base address of the descriptor area while bits 9:3 form a pointer to an individual descriptor. The first descriptor should be located at the base address and when it has been used by the GRETH, the pointer field is incremented by

8 to point to the next descriptor. The pointer automatically wraps back to zero when the last 1 KB boundary has been reached at address offset 0x3F8. The Wrap (WR) bit in the descriptors can be set to make the pointer wrap back to zero before the 1 KB boundary.

The Descriptor Pointer field has also been made writable for maximum flexibility, but care should be taken when writing to the descriptor pointer register. It should never be modified when reception is active.

The final step to activate reception is to set the Receiver Enable (RE) bit in the Control Register. This makes the GRETH read the first descriptor and wait for an incoming packet.

13.2.12 Descriptor Handling After Reception

The GRETH indicates a completed reception by clearing the descriptor's Enable bit. Control bits WR and IE are also cleared. The number of received bytes is shown in the Length field. The parts of the Ethernet frame stored are the destination address, source address, type, and data fields. Bits 17:14 in the first descriptor word are status bits indicating different receive errors. All four bits are zero after a reception without errors.

Packets arriving that are smaller than the minimum Ethernet size of 64 bytes are not considered valid and are discarded. The current receive descriptor will be left untouched and used for the first packet arriving with an accepted size. The Too Small (TS) bit in the Status Register is set each time this event occurs.

If a packet is received with an address not accepted by the MAC, the Invalid Address (IA) bit in the Status Register will be set.

Packets larger than maximum size cause the Frame Too Long (FT) bit in the receiver descriptor to be set. In this case, the Length field is not guaranteed to hold the correct value of received bytes. The counting stops after the word containing the last byte up to the maximum size limit has been written to memory.

The address word of the descriptor is never touched by the GRETH.

13.2.13 Reception with AHB Errors

If an AHB error occurs during a descriptor read or data store, the Receiver AHB Error (RA) bit in the Status Register will be set and the receiver is disabled. The current reception is aborted. The receiver can be enabled again by setting the Receive Enable (RE) bit in the Control Register.

13.2.14 MDIO Interface

The MDIO interface provides access to PHY configuration and status registers through a two-wire interface which is included in the MII interface. The GRETH provides full support for the MDIO interface.

The MDIO interface can be used to access from 1 to 32 PHYs containing 1 to 32 16-bit registers. A read transfer is set up by writing to the PHY Address and Register Address fields of the MDIO Control and Status Register and setting the Read (RD) bit. This causes the Busy (BU) bit to be set and the operation is finished when the Busy bit is cleared. If the operation was successful, the Link Fail (LF) bit is cleared and the data field contains the read data. An unsuccessful operation is indicated by the Link Fail bit being set. The data field is undefined in this case.

A write operation is started by writing to the 16-bit Data field and to the PHY Address and Register field of the MDIO Control Register and setting the Write (WR) bit. The operation is finished when the Busy (BU) bit is cleared and it was successful if the Link Fail bit is zero.

13.2.15 Media Independent Interfaces

There are several interfaces defined between the MAC sublayer and the physical layer. The GRETH supports the Media Independent Interface. The MII was defined in the 802.3 standard and is most commonly supported. The Ethernet interface has been implemented according to this specification and uses 16 signals.

13.2.16 Software Drivers

Drivers for the GRETH MAC are provided for the following operating systems: RTEMS, eCos, uClinux and Linux. The drivers are freely available in full source code under the GPL license from Cobham's Gaisler.

13.3 Registers

The core is programmed through registers mapped into APB address space.

Table 13.5: GRETH Registers

REGISTER	APB ADDRESS
Ethernet Control Register (ETHCTR)	0x80000E00
Ethernet Status and Interrupt Source Register (ETHSIS)	0x80000E04
Ethernet MAC Address MSB (MACMSB)	0x80000E08
Ethernet MAC Address LSB (MACLSB)	0x80000E0C
Ethernet MDIO Control and Status Register (ETHMDC)	0x80000E10
Ethernet Transmitter Descriptor Pointer Register (ETHTDP)	0x80000E14
Ethernet Receiver Descriptor Pointer Register (ETHRDP)	0x80000E18

ETHCTR **Address = 0x8000_0E00**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ED															
W																
Reset	1	[--...-]														

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										RS	PM	FD	RI	TI	RE	TE
W																
Reset	[--...-]									0	--	--	0	0	0	0

Figure 13.6: Ethernet Control Register

Table 13.6: Description of Ethernet Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31	ED	1	EDCL Available 0: EDCL unavailable

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			1: EDCL available Read=0; Write=don't care.
30-7	RESERVED	[--...-]	
6	RS	0	Reset Setting this bit resets the GRETH core. Self-clearing.
5	PM	-	Open Packet Mode If set, the GRETH operates in open packet mode, which means it will receive all packets regardless of the destination address. 0: Not open-packet mode 1: Operates in open-packet mode Not reset
4	FD	-	Full Duplex 0: GRETH operates in half-duplex mode 1: GRETH operates in full-duplex mode Not reset
3	RI	0	Enable Receiver Interrupts An interrupt will be generated each time a packet is received when this bit is set. The interrupt is generated regardless if the packet was received successfully or if it terminated with an error. Not reset. 0: Receiver interrupts disabled 1: Receiver interrupts enabled
2	TI	0	Enable Transmitter Interrupts An interrupt will be generated each time a packet is transmitted when this bit is set. The interrupt is generated regardless if the packet was transmitted successfully or if it terminated with an error. Not reset. 0: Transmitter interrupts disabled 1: Transmitter interrupts enabled
1	RE	0	Receive Enable Should be written with a one each time new descriptors are enabled. As long as this bit is one, the GRETH reads new descriptors and as soon as it encounters a disabled descriptor it will stop until RE is set again. This bit should be written with a one after the new descriptors have been enabled.
0	TE	0	Transmit Enable Should be written with a one each time new descriptors are enabled. As long as this bit is one the GRETH reads new descriptors and as soon as it encounters a disabled descriptor it stops until TE is set again. This bit should be written with a one after the new descriptors have been enabled.

ETHSIS **Address = 0x8000_0E04**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

R	▨
W	▨
Reset	[--...-]

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	▨	▨	▨	▨	IA	TS	TA	RA	TI	RI	TE	RE
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	[--...-]								0	0	--	--	--	--	--	--

Figure 13.7: GRETH Status and Interrupt Source Register

Table 13.7: Description of GRETH Status and Interrupt Source Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[--...-]	
7	IA	0	Invalid Address A set bit indicates a packet with an address not accepted by the MAC was received. Cleared when written with a one.
6	TS	0	Too Small A set bit indicates a packet smaller than the minimum size was received. Cleared when written with a one.
5	TA	_	Transmitter AHB Error A set bit indicates an AHB error was encountered in transmitter DMA engine. Cleared when written with a one. Not reset.
4	RA	_	Receiver AHB Error A set bit indicates an AHB error was encountered in receiver DMA engine. Cleared when written with a one. Not reset.
3	TI	_	Transmitter Interrupt A set bit indicates a packet was transmitted without errors. Cleared when written with a one. Not reset.
2	RI	_	Receiver Interrupt A set bit indicates a packet was received without errors. Cleared when written with a one. Not reset.
1	TE	_	Transmitter Error A set bit indicates a packet was transmitted which terminated with an error. Cleared when written with a one. Not reset.
0	RE	_	Receiver Error A set bit indicates a packet has been received which terminated with an error. Cleared when written with a one. Not reset.

MACMSB **Address = 0x8000_0E08**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

W	▨
Reset	[00…0]

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS_MSB[15:0]															
W																
Reset	[--…-]															

Figure 13.8: Ethernet MAC Address MSB

Table 13.8: Description of Ethernet MAC Address MSB

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	RESERVED	[00...0]	
15-0	Address MSB	[--...-]	The two most significant bytes of the MAC address. Not reset.

MACLSB **Address = 0x8000_0E0C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDRESS_LSB[31:16]															
W																
Reset	[--…-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS_LSB[15:0]															
W																
Reset	[--…-]															

Figure 13.9: Ethernet MAC Address LSB

Table 13.9: Description of Ethernet MAC Address LSB

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-0	Address LSB	[--...-]	The 4 least significant bytes of the MAC address. Not reset.

ETHMDC **Address = 0x8000_0E10**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA[15:0]															
W																
Reset	[--…-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_ADDRESS[4:0]					REG_ADDRESS[4:0]					▨	NV	BU	LF	RD	▨

W			▨				░	WR
Reset	[--…-]	[--…-]	--	--	0	--	0	0

Figure 13.10: Ethernet MDIO Control and Status Register

Table 13.10: Description of Ethernet MDIO Control and Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	Data	[--…-]	Contains data read during a read operation and data that is trans- mitted is taken from this field. Not Reset.
15-11	PHY_ADDRESS	[--…-]	This field contains the address of the PHY that should be accessed during a write or read operation. Not Reset.
10-6	REG_ADDRESS	[--…-]	This field contains the address of the register that should be accessed during a write or read operation. Not Reset.
5	RESERVED	--	
4	NV	--	Not Valid When an operation is finished (BU = 0) this bit indicates whether valid data has been received that is, the data field contains correct data. Not Reset. 0: Data field contains valid data 1: Data field contains invalid data
3	BU	0	Busy When an operation is performed, this bit is set to one. As soon as the operation is finished and the management link is idle, this bit is cleared. 0: Management link idle 1: Management link active
2	LF	_	Link Fail When an operation completes (BU = 0), this bit is set if a functional management link was not detected. Not Reset. 0: Functional management link detected 1: Functional management link not detected
1	RD	0	Read Set to start a read operation from the management interface. Data is stored in the Data field.
0	WR	0	Write Set to start a write operation to the management interface. Data is taken from the Data field.

ETHTDP **Address = 0x8000_0E14**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

R	TXDTRA[21:6]
W	
Reset	[--...-]

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXDTRA[5:0]						TX_DESCRIPTOR_PTR[6:0]							000		
W																
Reset	[--...-]						[--...-]							000		

Figure 13.11: Ethernet Transmitter Descriptor Pointer Register

Table 13.11: Description of Ethernet Transmitter Descriptor Pointer Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	TXDTRA	[--...-]	Base address of the Transmitter Descriptor Table
9-3	TX_DESCRIPTOR_PTR	[--...-]	This field is incremented by one each time a descriptor has been used. It is automatically incremented by the Ether- net MAC.
2-0	RESERVED	000	Read: 000b; Write=don't care.

ETHRDP **Address = 0x8000_0E18**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXDTRA[21:6]															
W																
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXDTRA[5:0]						RX_DESCRIPTOR_PTR[6:0]							000		
W																
Reset	[--...-]						[--...-]							000		

Figure 13.12: Ethernet Receiver Descriptor Pointer Register

Table 13.12: Description of Ethernet Receiver Descriptor Pointer Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-10	RXDTBA	[--...-]	Base address of the Receiver Descriptor Table
9-3	RX_DESCRIPTOR_PTR	[--...-]	This field is incremented by one each time a descriptor has been used. It is automatically incremented by the Ethernet MAC.
2-0	RESERVED	000	Read: 000b; Write=don't care.

Chapter 14: **Hardware Debug Support**

14.1 Overview

To simplify debugging on target hardware, the LEON 3FT processor implements a debug mode during which the pipeline is idle and the processor is controlled through a special debug interface. The LEON 3FT Debug Support Unit (DSU is used to control the processor during debug mode. The DSU acts as an AHB slave and can be accessed by any of the following AHB masters: the debug UART, the JTAG port, the PCI port, or a SpaceWire link using RMAP.

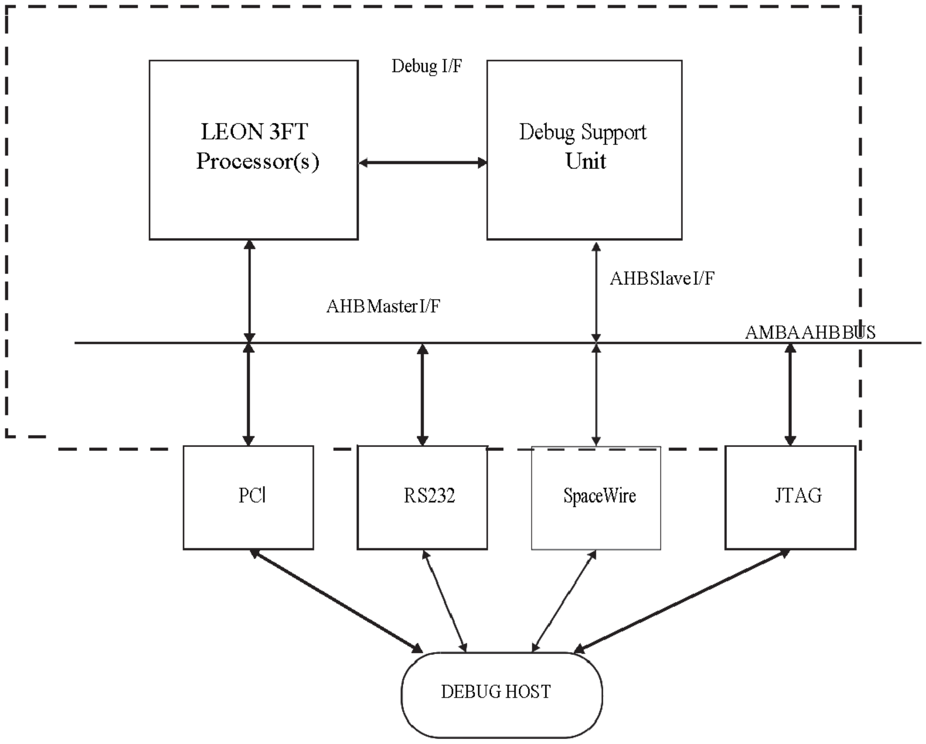


Figure 14.1: LEON 3FT DSU Connection

14.2 Operation

Through the DSU AHB slave interface, the AHB masters listed above can access the processor registers and the contents of the instruction trace buffer. The DSU control registers can be accessed at any time, while the processor registers, caches and trace buffer can only be accessed when the processor has entered debug mode. In debug mode, the processor pipe- line is held and the processor state can be accessed by the DSU. Entering the debug mode can occur on the following events:

- Executing a breakpoint, i.e., Trap Always instruction (ta 1)
- Integer unit hardware breakpoint/watchpoint hit (trap 0x0B)
- Rising edge of the external break signal (DSUBRE)
- Setting the Break-Now (BN) bit in the DSU Break and Single-Step Register
- A trap that would cause the processor to enter error mode
- Occurrence of any, or a selection of traps, as defined in the DSU Control Register
- After a single-step operation
- DSU breakpoint hit

Debug mode can only be entered when the debug support unit is enabled by setting the DSUEN pin high. When debug mode is entered, the following actions are taken:

- PC and nPC are saved in temporary registers (accessible by the debug unit)
- An output signal (DSUACT) is asserted to indicate the debug state
- The timer unit is (optionally) stopped to freeze the LEON 3FT timers and watchdog

The instruction that caused the processor to enter debug mode is not executed and the processor state is kept unmodified. Execution is resumed by clearing the BN bit in the DSU Control Register or by de-asserting DSUEN. The timer unit will be re-enabled and execution continues from the saved PC and nPC. Debug mode can also be entered after the processor has entered error mode. For instance, when an application has terminated and halted the processor error mode can be reset and the processor restarted at any address.

When a processor is in debug mode, accesses to the ASI diagnostic area are forwarded to the IU, which performs accesses with the ASI equal to value in the DSU ASI Diagnostic Register and address consisting of 20 least-significant bits of the original address.

14.3 AHB Trace Buffer

The AHB trace buffer consists of a circular buffer that stores AHB data transfers. The address, data, and various control signals of the AHB bus are stored and can be read out for later analysis. The trace buffer is 128 bits wide and 256 lines deep. The information stored is indicated in the **Table 14.1** below.

Table 14.1: AHB Trace Buffer Allocation

BITS	NAME	DEFINITION
127	AHB breakpoint hit	Set to '1' if a DSU AHB breakpoint hit occurred.
126	-	Not used
125-96	Time tag	DSU time tag counter
95	-	Not used
94-80	Hirq	AHB HIRQ[15:1]
79	Hwrite	AHB HWRITE
78-77	Htrans	AHB HTRANS
76-74	Hsize	AHB HSIZE
73-71	Hburst	AHB HBURST
70-67	Hmaster	AHB HMASTER
66	Hmastlock	AHB HMASTLOCK
65-64	Hresp	AHB HRESP
63-32	Load/Store data	AHB HRDATA or HWDATA
31-0	Load/Store address	AHB HADDR

In addition to the AHB signals, the DSU time tag counter is also stored in the trace. The trace buffer is enabled by setting the Enable (EN) bit in the AHB Trace Buffer Control Register. Each AHB transfer is then stored in the buffer in a circular manner. The address to which the next transfer is written is held in the Trace Buffer Index Register and is automatically incremented after each transfer. Tracing is stopped when the EN bit is cleared, or when an AHB breakpoint is hit. Tracing is temporarily suspended when the processor enters debug mode. Neither the trace buffer memory nor the breakpoint registers (see below) can be read/written by software when the trace buffer is enabled.

14.4 Instruction Trace Buffer

The instruction trace buffer consists of a circular buffer that stores executed instructions. The instruction trace buffer is located in the processor and read out via the DSU. The trace buffer is 128 bits wide and 256 lines deep. The information stored is indicated in the **Table 14.2** below.

Table 14.2: Instruction Trace Buffer Allocation

BITS	NAME	DEFINITION
127	-	Unused
126	Multi-cycle instruction	Set to '1' on the second and third instance of a multi- cycle instruction (LDD, ST or FPOP)
125-96	Time tag	The value of the DSU time tag counter
95-64	Load/Store parameters	Instruction result, Store address or Store data
63-34	Program counter	Program counter (2 LSB bits removed since they are always zero)
33	Instruction trap	Set to '1' if traced instruction trapped
32	Processor error mode	Set to '1' if the traced instruction caused processor error mode
31-0	Opcode	Instruction opcode

During tracing, one instruction is stored per line in the trace buffer with the exception of multi-cycle instructions. Multicycle instructions are entered two or three times in the trace buffer. For store instructions, bits 63:32 correspond to the store address on the first entry and to the stored data on the second entry (and the third in case of an STD instruction). Bit 126 is set on the second and third entry to indicate this. A double load (LDD) instruction is entered twice in the trace buffer with bits 63:32 containing the loaded data. Multiply and divide instructions are entered twice, but only the last entry contains the result. Bit 126 is set for the second entry. For FPU operation producing a double-precision result, the first entry puts the most-significant 32 bits of the results in bits 63:32, while the second entry puts the least-significant 32 bits in this field.

When the processor enters debug mode, tracing is suspended. The trace buffer and the AHB Trace Buffer Control Register can be read and written to, while the processor is in debug mode. During instruction tracing (processor in normal mode), the trace buffer and the AHB Trace Buffer Control Register cannot be accessed

14.5 DSU Memory Map

The DSU memory map can be seen in **Table 14.3** below. The base address is 0x9000_0000.

Table 14.3: DSU Memory Map

REGISTER	ADDRESS
DSU Control Register	0x90000000
DSU Trace Buffer Time Tag Counter Register	0x90000008
DSU Break and Single-Step Register	0x90000020
AHB Trace Buffer Control Register	0x90000040
AHB Trace Buffer Index Register	0x90000044
AHB Trace Buffer Breakpoint Address Register 1	0x90000050
AHB Trace Buffer Breakpoint Mask Register 1	0x90000054
AHB Trace Buffer Breakpoint Address Register 2	0x90000058
AHB Trace Buffer Breakpoint Mask Register 2	0x9000005c
Instruction Trace Buffer	0x90100000 - 0x9010FFFF
Instruction Trace Buffer Control Register	0x90110000
AHB Trace Buffer	0x90200000 - 0x9021FFFF
IU Register File	0x90300000 - 0x903007FC
IU Register File Port 1(ASR16) and Port 2 (ASR16)	0x90300800 - 0X90300FFC
FPU Register File	0x90301000 - 0x9030107C
IU Special Purpose Registers	0x90400000 - 0x904FFFFC
Y Register	0x90400000
PSR Register	0x90400004
WIM Register	0x90400008
TBR Register	0x9040000C
PC Register	0x90400010
NPC Register	0x90400014
FSR Register	0x90400018
CPSR Register	0x9040001C
DSU Trap Register	0x90400020
DSU ASI Diagnostic Access Register	0x90400024
ASR16 - ASR31 (when implemented)	0x90400040 - 0x9040007C

ASI Diagnostic Access (ASI = value in DSU ASI register,
address = address[19:0])
ASI = 0x9: Local instruction RAM
ASI = 0xB: Local data RAM
ASI = 0xC: Instruction cache tags
ASI = 0xD: Instruction cache data
ASI = 0xE: Data cache tags
ASI = 0xF: Data cache data
ASI = 0x1E: Separate snoop tags
	0x90700000 - 0x907FFFFC

The addresses of the IU registers are calculated as follows:

- %o*n*: 0x90300000 + (((psr.cwp * 64) + 32 + *n**4) mod 128)
- %l*n*: 0x90300000 + (((psr.cwp * 64) + 64 + *n**4) mod 128)

- %i*n*: 0x90300000 + (((psr.cwp * 64) + 96 + *n**4) mod 128)
- %g*n*: 0x90300000 + 128 + *n* *4
- %f*n*: 0x90301000 + *n*4*

14.6 DSU Registers

14.6.1 DSU Control Register

The DSU is controlled by the DSU control register:

DCR **Address = 9000_0000**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	PW	HL	PE	EB	EE	DM	BZ	BX	BS	BW	BE	TE
W	▨	▨	▨	▨				▨	▨	▨						
Reset	0000				0	0	0	--	--	0	0	0	0	0	0	0

Figure 14.2: DSU Control Register

Table 14.4: Description of DSU Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-12	RESERVED	[00...0]	
11	PW	0	Power Down Returns '1' when processor in power-down mode.
10	HL	0	Processor Halt Returns '1' on read when processor is halted. If the processor is in debug mode, setting this bit puts the processor in halt mode.
9	PE	0	Processor Error Mode Returns '1' on read when processor is in error mode, else '0'. If written with '1', it clears the error and halt mode.
8	EB	-	Value of the external DSUBRE signal (read-only).
7	EE	-	Value of the external DSUEN signal (read-only).
6	DM	0	Debug Mode Indicates when the processor has entered debug mode (read- only).
5	BZ	0	Break on Error Traps If set, forces the processor into debug mode on all *except* the following traps: privileged instruction, fpu_disabled, window_overflow, window_underflow, asynchronous_interrupt, ticc_trap.
4	BX	0	Break on Trap If set, forces the processor into debug mode when any

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			trap occurs.
3	BS	0	Break on Software Breakpoint If set, debug mode will be forced when a breakpoint instruction (ta 1) is executed.
2	BW	0	Break on IU Watchpoint If set, debug mode will be forced on an IU watchpoint (trap 0xb).
1	BE	0	Break on Error If set, forces the processor to debug mode when the processor would have entered error condition (trap in trap).
0	TE	0	Trap Enable Enables instruction tracing. If set the instructions will be stored in the trace buffer. Remains set when then processor enters debug or error mode.

14.6.2 DSU Break and Single-Step Register

This register is used to break or single-step the processor:

DBSSR **Address = 0x9000_0020**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																SS
W																
Reset	[00…0]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																BN
W																
Reset	[00…0]															0

Figure 14.3: DSU Break and Single-Step Register

Table 14.5: Description of DSU Break and Single-Step Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-17	RESERVED	[00...0]	
16	SS	0	Single-Step If set, the processor executes one instruction and return to debug mode. The bit remains set after the processor goes into the debug mode.
15-1	RESERVED	[00...0]	
0	BN	0	Break Now

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			Force the processor into debug mode if the Break on S/W break- point (BS) bit in the processors DSU control register is set. If cleared, the processor x resumes execution.

14.6.3 DSU Trap Register

The DSU trap register is a read-only register that indicates which SPARC trap type that caused the processor to enter debug mode. When debug mode is force by setting the BN bit in the DSU control register, the trap type will be 0xb (hardware watchpoint trap).

DTR **Address = 0x9040_0020**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				EM	TRAP_TYPE[7:0]								0000			
W																
Reset	000			0	[00…0]								0000			

Figure 14.4: DSU Trap Register

Table 14.6: Description of DSU Trap Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-13	RESERVED	[00...0]	
12	EM	0	Error Mode Set if the trap would have caused the processor to enter error mode.
11-4	TRAP_TYPE	[00...0]	8-bit SPARC trap type.
3-0		[00...0]	Read=0000b; Write=don't care.

14.6.4 DSU Trace Buffer Time Tag Counter Register

The trace buffer time tag counter increments each clock as long as the processor is running. The counter is stopped when the processor enters debug mode and restarted when execution is resumed. The value is used as time tag in the instruction and AHB trace buffer.

DTBTCR **Address = 0x9000_0008**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			DSU_TIME_TAG_VALUE[29:16]													

W	▨	
Reset	00	[--...-]

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DSU_TIME_TAG_VALUE[15:0]															
W																
Reset	[--...-]															

Figure 14.5: DSU Trace Buffer Time Tag Counter Register

Table 14.7: Description of DSU Trace Buffer Time Tag Counter Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-30	RESERVED	00	Read = 00b; Write = Don't Care
29-0	DSU_TIME_TAG_VALUE	[--...-]	

14.6.5 DSU ASI Diagnostic Access Register

The DSU performs diagnostic accesses to different ASI areas. The value in the ASI diagnostic access register is used as ASI while the address is supplied from the DSU.

DADAR **Address = 0x9040_0024**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨															
W	▨															
Reset	[--...-]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨								ASI[7:0]							
W	▨															
Reset	[--...-]								[--...-]							

Figure 14.6: DSU ASI Diagnostic Access Register

Table 14.8: Description of DSU ASI Diagnostic Access Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[--...-]	
7-0	ASI	[--...-]	ASI to be used on diagnostic ASI access.

14.6.6 AHB Trace Buffer Control Register

The AHB trace buffer is controlled by the AHB Trace Buffer Control Register:

ATBCR **Address = 0x9000_0040**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DCNT[15:0]															
W																
Reset	0x00F2															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RAM_TIM[3:0]				00		DM	EN
W																
Reset	[00…0]								0000				01		0	0

Figure 14.7: AHB Trace Buffer Control Register

Table 14.9: Description of AHB Trace Buffer Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-16	DCNT	0x00F2	Trace Buffer Delay Counter The number of bits actually implemented depends on the size of the trace buffer.
15-8	RESERVED	[00...0]	
7-4	RAM TIM	0000	Trace Buffer RAM Timing Registers Used for test only, must always be written with "0000".
3-2	RESERVED	01	Read=00b; Write=don't care.
1	DM	0	Delay Counter Mode Indicates that the trace buffer is in delay counter mode.
0	EN	0	Trace Buffer Enable 0: Disabled 1: Enabled

14.6.7 AHB Trace Buffer Index Register

The AHB trace buffer index register contains the address of the next trace line to be written.

ATBIR **Address = 0x9000_0044**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INDEX[27:12]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INDEX[11:0]												0000			
W																
Reset	[00…0]												0000			

Figure 14.8: AHB Trace Buffer Index Register

Table 14.10: Description of AHB Trace Buffer Index Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-4	INDEX	[00...0]	Trace Buffer Index Counter **Note:** The number of bits used depends on the size of the trace buffer.
3-0		0000	Read=0000b; Write=don't care.

14.6.8 AHB Trace Buffer Breakpoint Registers

The DSU contains two breakpoint registers for matching AHB addresses. A breakpoint hit is used to freeze the trace buffer by automatically clearing the enable bit. Freezing can be delayed by programming the DCNT field in the trace buffer control register to a non-zero value. In this case, the DCNT value decrements for each additional trace until it reaches zero, after which the trace buffer is frozen. A mask register is associated with each breakpoint, allowing breaking on a block of addresses. Only address bits with the corresponding mask bit set to '1' are compared during breakpoint detection. To break on AHB load or store accesses, the LD and/or ST bits should be set.

ATBAR1, ATBAR2 **Address = 0x9000_0050, 0x9000_0058**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BADDR[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BADDR[15:2]														00	
W																
Reset	[00…0]														00	

Figure 14.9: AHB Trace Buffer Breakpoint Address Register 1 and 2

Table 14.11: Description of AHB Trace Buffer Breakpoint Address Register 1 and 2

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	BADDR	[00...0]	Breakpoint Address [31:2]
1-0	RESERVED	0000	Read = 00b; write = don't care

ATBBMR1, ATBBMR2 **Address = 0x9000_0054, 0x9000_005C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BMASK[29:14]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R / W	BMASK[13:2]	LD	ST
Reset	[00…0]	0	0

Figure 14.10: AHB Trace Buffer Breakpoint Address Mask 1 and 2

Table 14.12: Description of AHB Trace Buffer Breakpoint Mask Register 1 and 2

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	BMASK	[00...0]	Breakpoint mask
1	LD	0	Break on data load address
0	ST	0	Break on data store address

14.6.9 Instruction Trace Control Registers

The instruction trace control register contains a pointer that indicates the next line of the instruction trace buffer to be written.

ITCR **Address = 0x9011_0000**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W									INSTR_TRACE_PTR[7:0]							
Reset	[00…0]								[00…0]							

Figure 14.11: Instruction Trace Control Register

Table 14.13: Description of Instruction Trace Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-8	RESERVED	[00...0]	
7-0	INSTR_TRACE_PTR	[00...0]	Pointer to the next line of the instruction trace buffer.

Chapter 15: **Serial Debug Link**

15.1 Overview

The serial debug link consists of a UART connected to the AHB bus as a master as shown in the **Figure 15.1** below. A simple communication protocol is supported to transmit access parameters and data. Through the communication link, a read or write transfer can be generated to any address on the AHB bus.

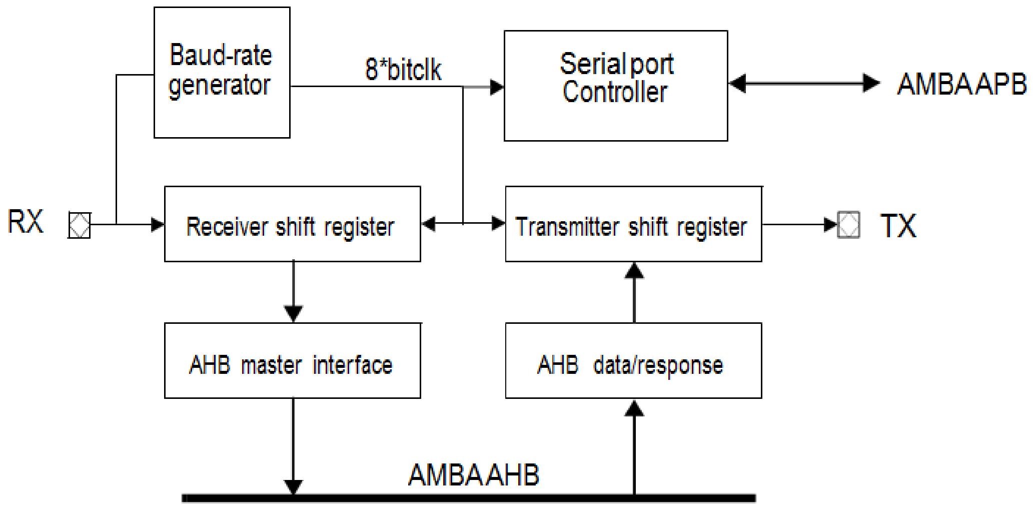


Figure 15.1: Debug UART Block Diagram

15.2 Operation

15.2.1 Transmission Protocol

The debug UART supports simple protocol where commands consist of a control byte, followed by a 32-bit address, followed by optional write data. A Write access does not return any response, while a read access returns only the read data. Data is sent on 8-bit basis as shown below.

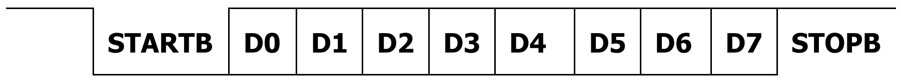


Figure 15.2: Debug UART Data Frame

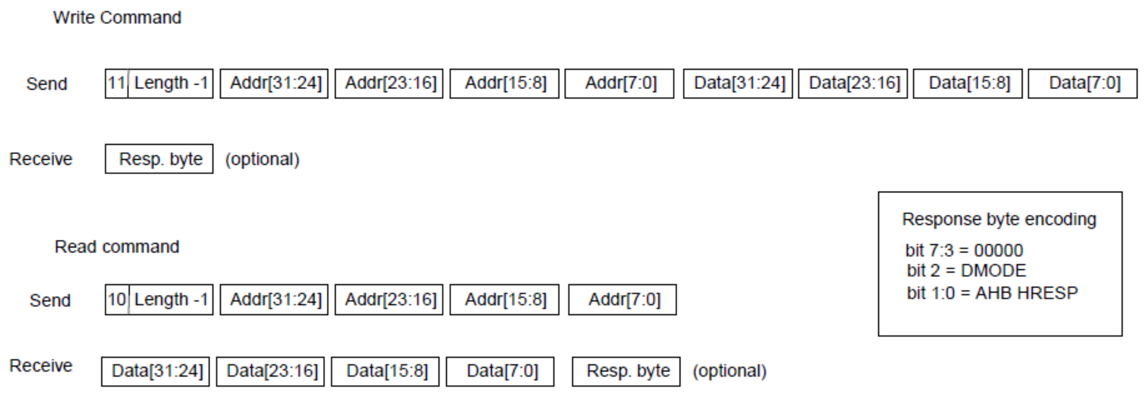


Figure 15.3: Debug UART Commands

Block transfers can be performed by setting the length field to *n*-1, where *n* denotes the number of transferred words. For write accesses, the control byte and address is sent once, followed by the number of data words to be written. The address is automatically incremented after each data word. For read accesses, the control byte and address is sent once and the corresponding number of data words is returned

15.2.2 Baud Rate Generator

The debug UART contains a 18-bit down-counting scaler to generate the desired baud-rate. The scaler is clocked by the system clock and generates a UART tick each time it underflows. The scaler is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be eight times the desired baud-rate.

If not programmed by software, the baud rate will be automatically discovered. This is done by searching for the shortest period between two falling edges of the received data (corresponding to two-bit periods). When three identical two-bit periods have been found, the corresponding scaler reload value is latched into the reload register, and the Baud Rate Lock (BL) bit is set in the UART Control Register. If the BL bit is reset by software, the baud rate discovery process is restarted. The baud rate discovery is also restarted when a 'break' or framing error is detected by the receiver, allowing the system to change the baud rate from the external transmitter. For proper baud rate detection, the value 0x55 should be transmitted to the receiver after reset or after sending a break.

The best scaler value for manually programming the baudrate can be calculated as follows:

scaler = (((system_clk*10)/ (baudrate*8))-5)/10

15.3 Registers

The debug UART can be programmed through the registers mapped into APB address space.

Table 15.1: Description of Debug UART Register Address

REGISTER	APB ADDRESS
AHB UART Status Register (SDLSTR)	0x80000704
AHB UART Control Register (SDLCTR)	0x80000708
AHB UART Scaler Register (SDLSCL)	0x8000070C

SDLCTR **Address = 0x8000_0704**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset									--							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	BL	EN
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Reset								[--...-]							0	0

Figure 15.4: Debug UART Control Register

Table 15.2: Description of Debug UART Control Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-2	RESERVED	[--...-]	
1	BL	0	Baud Rate Locked This bit is automatically set when the baud rate is locked.
0	RE	0	Receiver Enable If set, both the transmitter and receiver are enabled.

SDLSTR **Address = 0x8000_0708**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset									[--...-]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	▨	▨	▨	▨	▨	FE	▨	OV	▨	TH	TS	DR
W	▨	▨	▨	▨	▨	▨	▨	▨	▨		▨		▨			
Reset					[--...-]					0	0	0	0	1	1	0

Figure 15.5: Debug UART Status Register

Table 15.3: Description of Debug UART Status Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-7	RESERVED	[--...-]	
6	FE	0	Frame Error Indicates that a frame error was detected.
5	RESERVED	0	
4	OV	0	Overrun Indicates that one or more characters have been lost due to overrun.

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
3	RESERVED	0	
2	TH	1	Transmitter Hold Register Empty Indicates that the transmitter hold register is empty. Read only.
1	TS	1	Transmitter Shift Register Empty Indicates that the transmitter shift register is empty.
0	DR	0	Data Ready Indicates that new data has been received by the AHB master interface.

SDLSCL **Address = 0x8000_070C**

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			SRV[13:0]													
W																
Reset	00		[00…0]													

Figure 15.6: Debug UART Scaler Reload Register

Table 15.4: Description of Debug UART Scaler Reload Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
31-14	RESERVED	[00...0]	
13-0	SRV	[00...0]	Scaler Reload Value See Section **15.2.2**. The optimum value is: SCALER = ((SYSCLK*10)/baudrate*8))-5/10.

Chapter 16: **JTAG Debug Link**

16.1 Overview

The JTAG debug interface provides access to the AMBA AHB bus through JTAG. The JTAG debug interface implements a simple protocol that translates JTAG instructions to AHB transfers. Through this link, a read or write transfer can be generated to any address on the AHB bus.

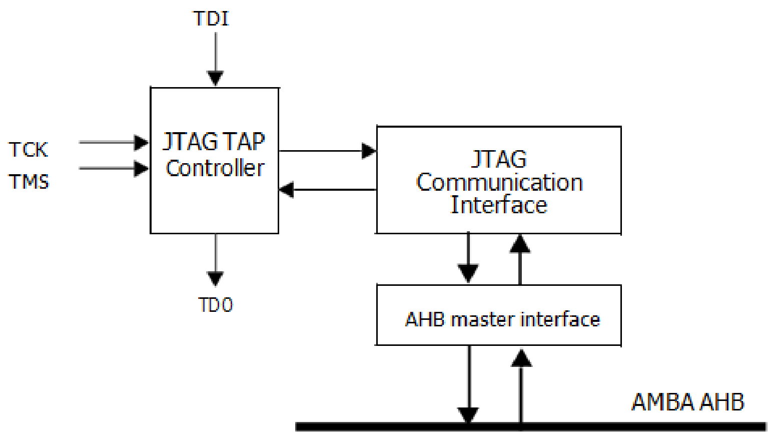


Figure 16.1: JTAG Debug Link Block Diagram

16.2 Operation

16.2.1 Transmission Protocol

The JTAG Debug link decodes two JTAG instructions and implements two JTAG data registers: the JTAG Debug Link Command and Address Register and Data Register. A read access is initiated by setting the Write bit, and the SIZE and AHB_ADDRESS fields of the Command and Address Register and shifting out the data over the JTAG port. The AHB read access is performed and data is ready to be shifted out of the Data Register. Write accesses are performed by setting the Write bit, and the SIZE and AHB_ADDRESS fields of the Command and Address Register, followed by shifting in write data into the Data Register. Sequential transfers can be performed by shifting in command and address for the transfer start address and setting the SEQ bit in Data Register for subsequent accesses. The SEQ bit increments the AHB address for the subsequent access. Sequential transfers should not cross a 1 KB boundary. Sequential transfers are always word based.

JDLCAR

Bit#	34	33	32
R	W	SIZE[1:0]	
W			
Reset	0	00	

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AHB_ADDRESS[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AHB_ADDRESS[15:0]															
W																
Reset	[00…0]															

Figure 16.2: JTAG Debug Link Command and Address Register

Table 16.1: Description of JTAG Debug Link Command and Address Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
34	W	0	Write 0: Read transfer 1: Write transfer
33-32	SIZE	00	AHB Transfer Size 00: Byte 01: Half word 10: Word 11: Reserved
31-0	AHB_ADDRESS	[00...0]	AHB Address

JDLDR

Bit#	32
R	SQ
W	
Reset	0

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AHB_DATA[31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AHB_DATA[15:0]															
W																
Reset	[00…0]															

Figure 16.3: JTAG Debug Link Data Register

Table 16.2: Description of JTAG Debug Link Data Register

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
32	SQ	0	Sequential Transfer 0: Non-sequential transfer 1: When read data is shifted out or write data shifted in, the subsequent transfer will be to the word address.
31-0	AHB_DATA	[00...0]	For byte and half-word transfers, data is aligned according to big-endian order where data with address

BIT NUMBER(S)	BIT NAME	RESET STATE	DESCRIPTION
			offset 0 data is placed in MSB bits.

16.3 Register

The core does not implement any registers mapped in the AMBA AHB or APB address space.

16.4 Vendor and Devices Identifiers

The core has vendor identifier 0x01 (Gaisler Research) and device identifier 0x01C.

Chapter 17: **CLKGATE Clock Gating Unit**

17.1 Overview

The CLKGATE clock gating unit provides a means to save power by disabling the clock to unutilized core blocks. The unit can enable and disable up to eight individual clock signals or reset the core state and registers to default.

17.2 Operation

The operation of the clock gating unit is controlled through three registers: the unlock, the clock enables, and the core reset registers. The clock enable register defines if a clock is enabled or disabled. A '1' in a bit location enables the corresponding clock, while a '0' disables the clock. The core reset register resets each core to a default state. A reset will be generated as long as the corresponding bit is set to '1'. The bits in clock enable and core reset registers can only be written when the corresponding bit in the unlock register is 1. If a bit in the unlock register is 0, the corresponding bits in the clock enable and core reset registers cannot be written.

To gate the clock for a core, the following procedure should be applied:

1. Disable the core through software to make sure it does not initialize any AHB accesses.
2. Write 1 to the corresponding bit in the unlock register.
3. Write 1 to the corresponding bit in the core reset register.
4. Write 0 to the corresponding bit in the clock enable register.
5. Write 0 to the corresponding bit in the unlock register.

To enable the clock for a core, the following procedure should be applied

1. Write 1 to the corresponding bit in the unlock register.
2. Write 1 to the corresponding bit in the core reset register.
3. Write 1 to the corresponding bit in the clock enable register.
4. Write 0 to the corresponding bit in the core reset register.
5. Write 0 to the corresponding bit in the unlock register.

The clock gating unit directly controls the clock and reset lines for peripheral units without any guards in place that check if the peripheral unit is idle. If a peripheral is disabled in the middle of a bus transaction this can lead to a system freeze that requires a full system reset to resolve. Because of this it is not recommended to disable units via the clock gating unit unless the peripheral being clock gated off is guaranteed to be in idle state.

The cores connected to the clock gating unit are defined in the **Table 17.1** below:

Table 17.1: Clocks Controller by CLKGATE Unit

BIT	FUNCTIONAL MODE
0	GRSPW SpaceWire link 0
1	GRSPW SpaceWire link 1
2	GRSPW SpaceWire link 2
3	GRSPW SpaceWire link 3
4	CAN core 1 & 2
5	GRETH 10/100 Mbit Ethernet MAC (AHB Clock)
6	GRPCI 32-bit PCI Bridge (AHB Clock)

17.3 Registers

Table 17.2 shows the clock gating unit registers. The base address for the registers is 0x80000600.

Table 17.2: Clocks Unit Control Registers

APB ADDRESS	FUNCTIONAL MODULE	RESET VALUE
0x80000600	Unlock Register	00000000
0x80000604	Clock Enable Register	01111111
0x80000608	Core Reset Register	00000000

NOTE: The clock for the GR1553B is disabled after reset and must be enabled before the unit can be used.

Appendix A: Register Format

MCFG2 Address = 0x8000_0004

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DR	DP	DF[2:0]			DC	DZ[2:0]			DS[1:0]		DD[1:0]		BW	PB	▨
W														░		
Reset	0	1	111			1	000			10		00		0	0	0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	▨	DE	SI	SZ[3:0]				▨	SB	RM	SD[1:0]		SW[1:0]		SR[1:0]	
W																
Reset	0	0	0	--				0	0	-	--		00		00	

Legend:

Bit#	Bit Number
R	Read Operation
W	Write Operation
░	Don't Care
▨	Reserved
--	Reset State, Not Initialized
0	Reset State is 0
1	Reset State is 1

REVISION HISTORY

REV	Revision Date	Description of Change	Page(s)	Author
1.0.0	08/21/2016	Initial Release		Simms
1.1.0	01/06/2017	New format	All	Sim
1.1.1	07/31/2017	Table 1.2	12	MTS
1.1.2	01/29/2018	2.4.5	38	MTS
1.1.3	09/05/2018	Table 1.2	12.13	MTS

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.