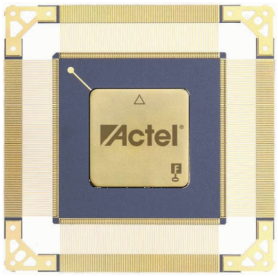
LEON3-FT SPARC V8 Processor
LEON3FT-RTAX
Data Sheet and User's Manual

Features



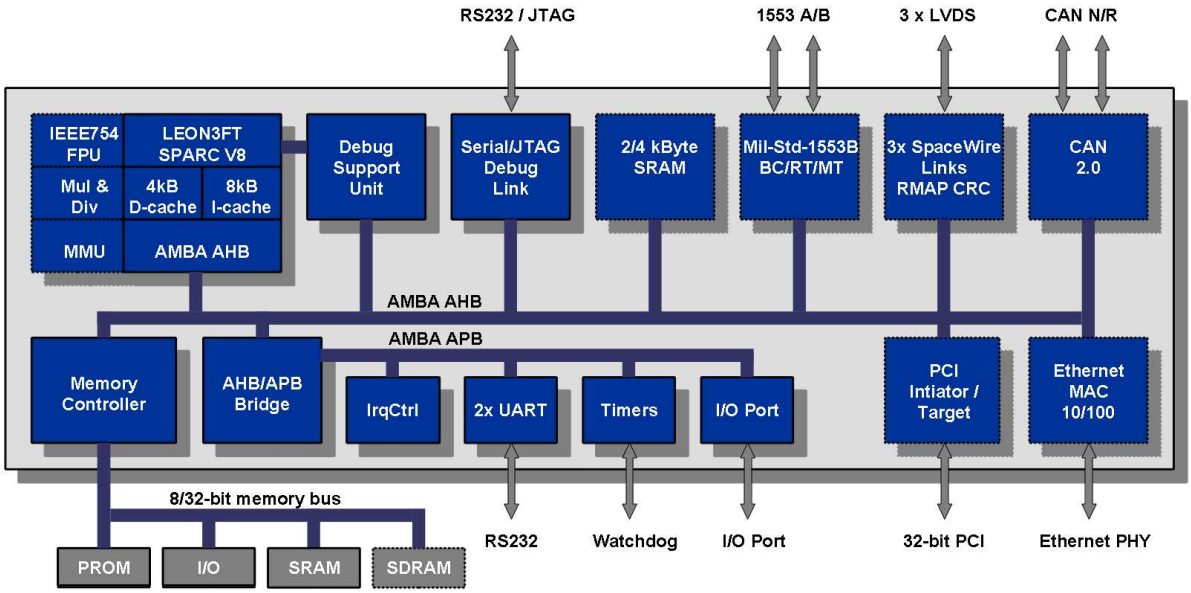
- SPARC V8 integer unit with 7-stage pipeline, 8 register windows, 8 kByte instruction and 4 kByte data caches, hardware multiplier and divider, power-down mode, hardware watchpoints, etc.
- Double precision IEEE-754 floating point unit
- Memory management unit
- EDAC protected (BCH and optional Reed-Solomon) interface to multiple 8/32-bits PROM/SRAM/SDRAM memory banks
- Advanced on-chip debug support unit
- UARTs, Timers & Watchdog, GPIO port, Interrupt controller, Status registers
- Multiple SpaceWire links with RMAP CRC
- Redundant 1553 BC/RT/MT interfaces
- Redundant CAN 2.0 interfaces
- Multiple Ethernet 10/100 Mbit MAC interfaces
- PCI Initiator/Target and Arbiter interface

Description

The LEON3FT-RTAX product is an implementation of the LEON3FT SPARC V8 processor using the Actel RTAX FPGA technology. The fault tolerant design of the processor in combination with the radiation tolerant FPGA technology provides total immunity to radiation effects.

Specification

- CQ352 and CG624 packages
- Total Ionizing Dose (TID) up to 300 krad (Si, functional)
- Single-Event Latch-Up Immunity (SEL) to LET_{TH} > 104 MeV-cm2/mg
- Immune to Single-Event Upsets (SEU) to LET_{TH} > 37 MeV-cm2/mg
- 1.5V & 3.3V supply, 500 mW consumption
- Up to 25 MHz system frequency
- Up to 20 MIPS, 4 MFLOPS



Applications

The LEON3FT-RTAX processor is provided in multiple configurations, covering instrument, payload and spacecraft control applications. Custom configurations are offered on request.

The LEON3FT-RTAX processor is ideally suited for space and other high-rel applications.

1 Introduction

1.1 Overview

The LEON3FT-RTAX processor family is based on a common architecture from which standard configurations are derived. The architecture is centered around the AMBA Advanced High-speed Bus (AHB), to which the LEON3FT processor and other high-bandwidth units are connected. Low-bandwidth units are connected to the AMBA Advanced Peripheral Bus (APB) which is accessed through an AHB to APB bridge. The architecture is shown in figure 1.

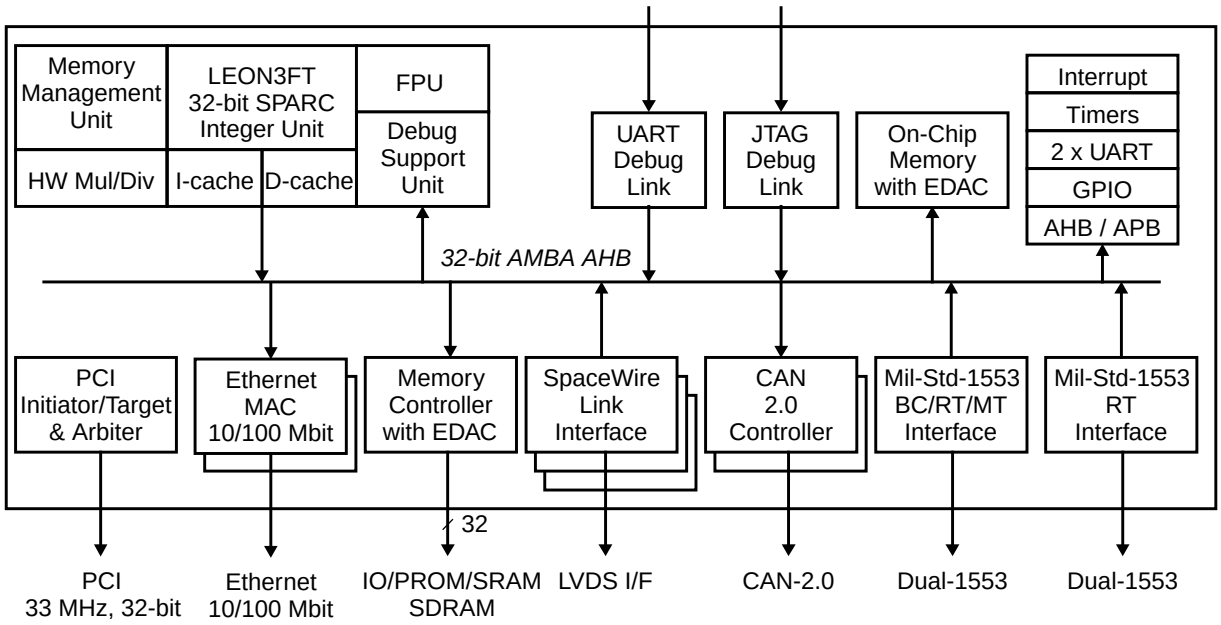


Figure 1. Architectural block diagram

The full LEON3FT-RTAX architecture includes the following modules:

- LEON3 SPARC V8 Integer Unit with 8 kByte instruction cache and 4 kByte data cache
- IEEE-754 Floating Point Unit and Memory Management Unit
- Debug Support Unit with UART and JTAG Debug Links
- On-Chip Memory with EDAC
- 8/32-bit Memory Controller with EDAC for external PROM, SRAM and I/O
- 8/32-bit SDRAM Controller with EDAC for external PC100 SDRAM, PROM, SRAM and I/O
- Timer unit with two 32-bit timers and a watchdog
- Interrupt controller for 15 interrupts in two priority levels
- Two UARTs with FIFO and separate baud rate generators
- 16-bit general purpose I/O port (GPIO). Can also generate 15 interrupts from external devices
- AMBA AHB status register
- Up to three SpaceWire links with CRC support
- Up to two CAN-2.0 controllers
- Mil-Std-1553 BC/RT/MT or Mil-Std-1553 RT based on the Actel Core1553 IP
- Up to two Ethernet Media Access Controller (MAC)
- PCI Initiator/Target and Arbiter based on Actel CorePCIF IP

1.2 Signal overview

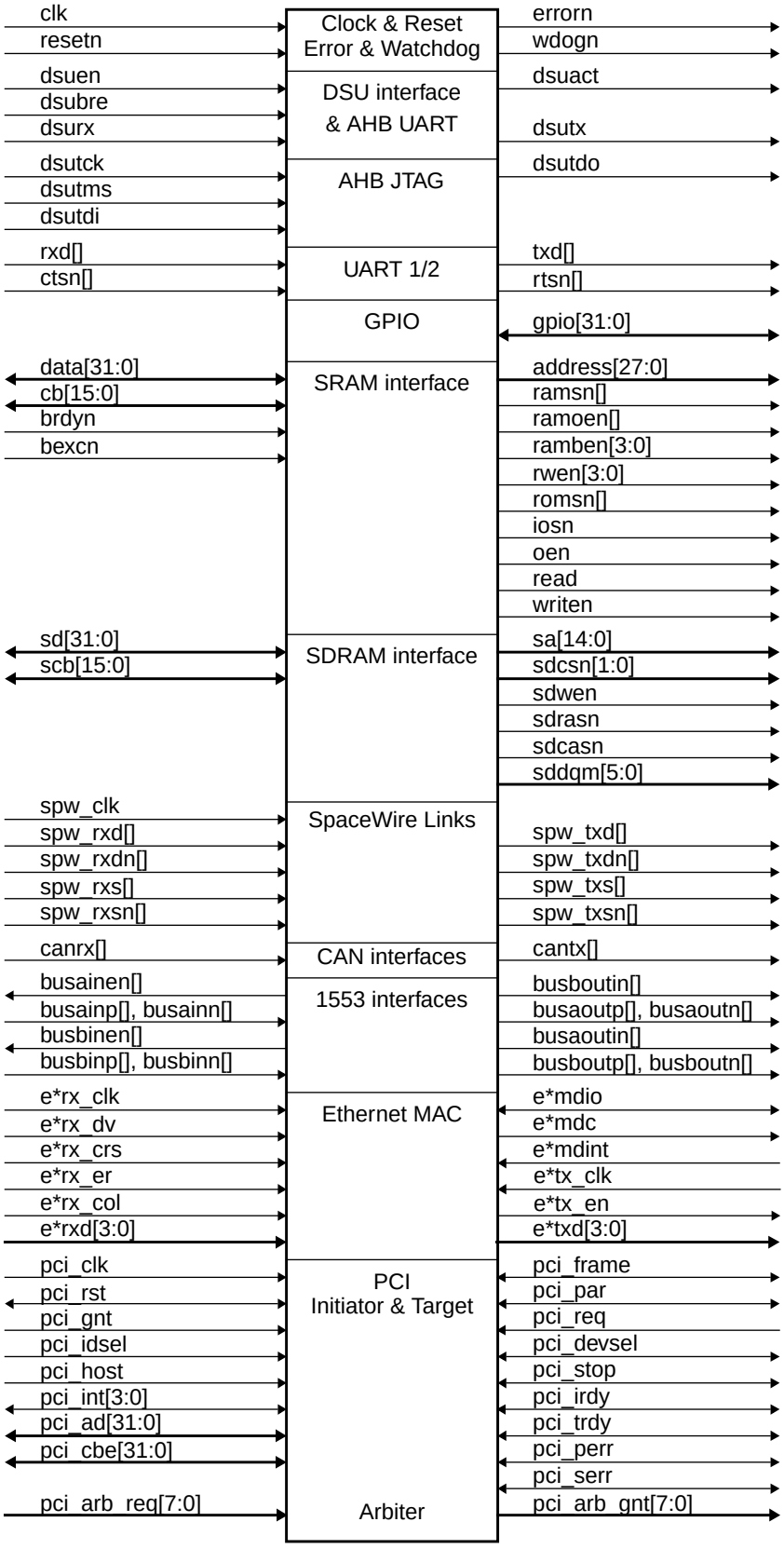


Figure 2. Signal overview

1.3 Standard configurations

Due to resource limitations, it is not possible to fit the full LEON3FT-RTAX architecture on a single RTAX2000 device. Therefore, sub-set configurations have been defined to suit applications with different computational and interfacing needs. Table 1 below shows which functions are available in each of the configurations.

Table 1. Standard configurations

Configuration name	Instrument Controller-1	Instrument Controller-2	Spacecraft Controller-1	Spacecraft Controller-2	Spacecraft Controller-3	Spacecraft Controller-4	Payload Controller-1	Payload Controller-2
Configuration ID (CID)	1	2	3	4	5	6	7	8
LEON3FT Integer Unit	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hardware multiply&divide					Yes	Yes		Yes
Multiply & accumulate								
Single-vector trapping	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Power down mode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Memory Management Unit					Yes	Yes		
Floating Point Unit	Yes	Yes	Yes				Yes	Yes
Debug Support Unit	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UART Debug Link	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
JTAG Debug Link								
On-Chip Memory	4 kByte		4 kByte					
1553 RT	1							
1553 BC/RT/MT			1					
SpaceWire		2		3	2		2	
SpaceWire LVDS		Yes [2)]		Yes [2)]	Yes [2)]		Yes [2)]	
CAN 2.0B	1				1			
PCI Initiator/Target						Yes		
PCI Arbiter						Yes		
Ethernet MAC 10/100 Mbit						1		2
Memory Controller	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SDRAM Support				Yes	Yes	Yes	Yes	Yes
Reed-Solomon for SDRAM				Yes [1)]	Yes [1)]		Yes [1)]	
Standard peripherals	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Additional 16-bit GPIO							Yes	
Package	CQFP352	CQFP352	CQFP352	CCGA624	CCGA624	CCGA624	CQFP352	CQFP352

Note 1: Optional Reed-Solomon protection for SDRAM memory can be included upon request.

Note 2: Optional LVTTL instead of LVDS drivers/receivers for SpaceWire links can be included upon request.

2 Architecture

2.1 Cores

The common architecture of the LEON3FT RTAX family is based on cores from the GRLIB IP library. The vendor and device identifiers for each core can be extracted from the plug & play information. The used IP cores are listed in table 2.

Table 2. Used IP cores

Core	Function	Vendor	Device	CID
AHBCTRL	AHB Arbiter & Decoder	0x01	-	All
APBCTRL	AHB/APB Bridge	0x01	0x006	All
LEON3FT	LEON3 SPARC V8 32-bit processor	0x01	0x053	All
DSU3	LEON3 Debug support unit	0x01	0x004	All
AHBUART	Serial/AHB debug interface	0x01	0x007	All
AHBJTAG	JTAG/AHB debug interface	0x01	0x01C	-
FTSRCTRL	PROM/SRAM/IO Memory Interface	0x01	0x051	1, 2, 3
FTMCTRL	PROM/SRAM/SDRAM/IO Memory Interface	0x01	0x054	4, 5, 6, 7, 8
FTAHBRAM	On-chip SRAM with EDAC	0x01	0x050	1, 3
AHBSTAT	AHB failing address register	0x01	0x052	All
APBUART	8-bit UART with FIFO	0x01	0x00C	All
GPTIMER	Modular timer unit with watchdog	0x01	0x011	All
IRQMP	LEON3 Interrupt controller	0x01	0x00D	All
GRGPIO	General purpose I/O port	0x01	0x01A	All
GRSPW	SpaceWire link	0x01	0x01F	2, 4, 5, 7
CAN_OC	CAN-2.0 interface	0x01	0x019	1, 5
B1553RT	MIL-STD-1553 RT	0x01	0x071	1
B1553BRM	MIL-STD-1553 BC/RT/BM	0x01	0x072	3
PCIF	PCI Initiator/Target	0x01	0x075	6
PCIARB	PCI Arbiter	0x04	0x010	6
GRETH	Ethernet Media Access Controller (MAC)	0x01	0x01D	6, 8

2.2 Interrupts

The LEON3FT RTAX family uses the same interrupt assignment for all configurations. See the description of the individual cores for how and when the interrupts are raised. All interrupts are handled by the interrupt controller and forwarded to the LEON3 processor.

Table 3. Interrupt assignment

Core	Interrupt	Comment	CID
AHBSTAT	1		All
APBUART 1	2		All
APBUART 2	3		All
CAN_OC 0, 1	4, 5		1, 5
GPTIMER	6, 7, 8, 9	Interrupt number 9 is spare	All
GRSPW 0, 1, 2	10, 11, 12		2, 4, 5, 7
B1553RT	13		1
B1553BRM	14		3
GRGPIO	1-15	Generated from external GPIO signals 1 to 15	All
PCIF	5		6
GRETH 0	12		6, 8
GRETH 1	11		8

2.3 Memory map

The LEON3FT RTAX family uses the same memory map for all standard configurations. The memory map shown in table 4 is based on the AMBA AHB address space. Access to addresses outside the ranges will return an AHB error response. The detailed register layout is defined in the description of each individual core.

Table 4. AMBA AHB address range

Core	Address range	Area	CID
FTSRCTRL	0x00000000 - 0x20000000	PROM area	1, 2, 3
	0x20000000 - 0x40000000	I/O area	
	0x40000000 - 0x60000000	SRAM area	
FTMCTRL	0x00000000 - 0x20000000	PROM area	4, 5, 6, 7, 8
	0x20000000 - 0x40000000	I/O area	
	0x40000000 - 0x80000000	SRAM / SDRAM area	
APBCTRL	0x80000000 - 0x80100000	APB bridge	All
DSU3	0x90000000 - 0xA0000000	Registers	All
FTAHBRAM	0xA0000000 - 0xA0000800	On-chip RAM	1, 3
PCIF	0xC0000000 - 0xFFF00000	PCI memory area	6
	0xFFF00000 - 0xFFF20000	PCI configuration and I/O area	
CAN_OC	0xFFFC0000 - 0xFFFC1000	Registers	1, 5
AHB plug&play	0xFFFFF000 - 0xFFFFFFFF	Registers	All

The control registers of most on-chip peripherals are accessible via the AHB/APB bridge, which is mapped at address 0x80000000. The memory map shown in table 5 is based on the AMBA AHB address space.

Table 5. APB address range

Core	Address range	Comment	CID
FTSRCTRL	0x80000000 - 0x80000100		1, 2, 3
FTMCTRL	0x80000000 - 0x80000100		4, 5, 6, 7, 8
APBUART1	0x80000100 - 0x80000200		All
IRQMP	0x80000200 - 0x80000300		All
GPTIMER	0x80000300 - 0x80000400		All
PCIF	0x80000400 - 0x80000500		6
FTAHBRAM	0x80000600 - 0x80000700		1, 3
AHBUART	0x80000700 - 0x80000800		All
GRGPIO	0x80000800 - 0x80000900		All
APBUART2	0x80000900 - 0x80000A00		All
GRSPW 0	0x80000A00 - 0x80000B00		2, 4, 5, 7
GRSPW 1	0x80000B00 - 0x80000C00		2, 4, 5, 7
GRSPW 2	0x80000C00 - 0x80000D00		4
GRETH 0	0x80000C00 - 0x80000D00		6, 8
GRETH 1	0x80000B00 - 0x80000C00		8
B1553RT	0x80000D00 - 0x80000E00		1
B1553BRM	0x80000C00 - 0x80000E00		3
PCI_ARB	0x80000E00 - 0x80000F00		6
AHBSTAT	0x80000F00 - 0x80001000		All
APB plug&play	0x800FF000 - 0x80100000		All

2.4 Plug & play information

The LEON3FT RTAX family uses the same plug & play map for all standard configurations. The plug & play memory map and bus indexes for AMBA AHB masters are shown in table 6 and is based on the AMBA AHB address space.

Table 6. Plug & play information for AHB masters

Core	Index	Function	Address range	CID
LEON3FT	0	LEON3 SPARC V8 32-bit processor	0xFFFFF000 - 0xFFFFF01F	All
AHBUART	1	Serial/AHB debug interface	0xFFFFF020 - 0xFFFFF03F	All
AHBJTAG	2	JTAG/AHB debug interface	0xFFFFF040 - 0xFFFFF05F	-
GRSPW	2	SpaceWire link 0	0xFFFFF040 - 0xFFFFF05F	2, 4, 5, 7
GRSPW	3	SpaceWire link 1	0xFFFFF060 - 0xFFFFF07F	2, 4, 5, 7
GRSPW	4	SpaceWire link 2	0xFFFFF080 - 0xFFFFF09F	4
B1553RT	2	MIL-STD-1553 RT	0xFFFFF040 - 0xFFFFF05F	1
B1553BRM	2	MIL-STD-1553 BC/RT/BM	0xFFFFF040 - 0xFFFFF05F	3
PCIF	2	PCI Initiator/Target	0xFFFFF040 - 0xFFFFF05F	6
GRETH 0	3	Ethernet Media Access Controller	0xFFFFF060 - 0xFFFFF07F	6, 8
GRETH 1	2	Ethernet Media Access Controller	0xFFFFF040 - 0xFFFFF05F	8

The plug & play memory map and bus indexes for AMBA AHB slaves are shown in table 7 and is based on the AMBA AHB address space.

Table 7. Plug & play information for AHB slaves

Core	Index	Function	Address range	CID
FTSRCTRL	0	PROM/SRAM/IO Memory Interface	0xFFFFF800 - 0xFFFFF81F	1, 2, 3
FTMCTRL	0	PROM/SRAM/SDRAM/IO Memory Interface	0xFFFFF800 - 0xFFFFF81F	4, 5, 6, 7, 8
APBCTRL	1	AHB/APB Bridge	0xFFFFF820 - 0xFFFFF83F	All
DSU3	2	LEON3 Debug support unit	0xFFFFF840 - 0xFFFFF85F	All
CAN_OC	4	CAN-2.0 interface	0xFFFFF880 - 0xFFFFF89F	1, 5
PCIF	4	PCI Initiator/Target	0xFFFFF880 - 0xFFFFF89F	6
FTAHBRAM	7	On-chip SRAM with EDAC	0xFFFFF8E0 - 0xFFFFF8FF	1, 3

The plug & play memory map and bus indexes for AMBA APB slaves are shown in table 8 and is based on the AMBA AHB address space.

Table 8. Plug & play information for APB slaves

Core	Index	Function	Address range	CID
FTSRCTRL	0	PROM/SRAM/IO Memory Interface	0x800FF000 - 0x800FF007	1, 2, 3
FTMCTRL	0	PROM/SRAM/SDRAM/IO Memory Interface	0x800FF000 - 0x800FF007	4, 5, 6, 7, 8
APBUART	1	8-bit UART with FIFO - 1	0x800FF008 - 0x800FF00F	All
IRQMP	2	LEON3 Interrupt controller	0x800FF010 - 0x800FF017	All
GPTIMER	3	Modular timer unit with watchdog	0x800FF018 - 0x800FF01F	All
PCIF	4	PCI Initiator/Target	0x800FF020 - 0x800FF027	6
FTAHBRAM	6	On-chip SRAM with EDAC	0x800FF030 - 0x800FF037	1, 3
AHBUART	7	Serial/AHB debug interface	0x800FF038 - 0x800FF03F	All
GRGPIO	8	General purpose I/O port	0x800FF040 - 0x800FF047	All
APBUART	9	8-bit UART with FIFO - 2	0x800FF048 - 0x800FF04F	All
GRSPW	10	SpaceWire link 0	0x800FF050 - 0x800FF057	2, 5, 7
GRSPW	11	SpaceWire link 1	0x800FF058 - 0x800FF05F	2, 5, 7
GRSPW	12	SpaceWire link 2	0x800FF060 - 0x800FF067	2
GRETH 0	12	Ethernet Media Access Controller	0x800FF060 - 0x800FF067	6, 8
GRETH 1	11	Ethernet Media Access Controller	0x800FF058 - 0x800FF05F	8
B1553RT	13	MIL-STD-1553 RT	0x800FF068 - 0x800FF06F	1
B1553BRM	14	MIL-STD-1553 BC/RT/BM	0x800FF070 - 0x800FF077	3
PCIARB	14	PCI Arbiter	0x800FF070 - 0x800FF077	6
AHBSTAT	15	AHB failing address register	0x800FF078 - 0x800FF07F	All

2.5 Configuration

The LEON3 SPARC V8 Integer Unit is configured as follows:

- 8 kByte instruction cache, single set, 32 bytes per line
- 4 kByte data cache, single set, 16 bytes per line
- 8 register windows
- 2 hardware watchpoints
- Optional Memory Management Unit (see table 1)
- Optional Floating Point Unit (see table 1)
- Optional hardware multiplier (see table 1)
- Power down support
- Single-vector trapping support
- No multiply and accumulate instructions

The Debug Support Unit is configured as follows:

- 2 kByte instruction trace buffer
- 2 kByte AMBA bus trace buffer

The Fault Tolerant PROM/SRAM/IO Memory Interface is configured as follows:

- Programmable wait states
- 4 SRAM banks, programmable sizes
- 4 PROM banks, programmable sizes
- Read-modify-write for sub-word writes
- EDAC with Bose, Ray-Chaudhuri, Hocquenghem (BCH) code, providing single bit error correction and double bit error detection (SEC-DED)

The Fault Tolerant PROM/SRAM/SDRAM/IO Memory Interface is configured as follows:

- Programmable wait states
- 5 SRAM banks, programmable sizes
- 4 PROM banks, programmable sizes
- 2 SDRAM banks, programmable sizes
- Support for 8-bit EDAC PROM interface
- Read-modify-write for sub-word writes
- Support for line burst
- EDAC with Bose, Ray-Chaudhuri, Hocquenghem (BCH) code, providing single bit error correction and double bit error detection (SEC-DED)
- EDAC with Reed-Solomon code for optional SDRAM protection. The 32-bit data and the 16-bit checksum, i.e. 48 bits, are used to form two separate codewords each comprising six 4-bit nibbles. Every second nibble of the 48 bits belongs to a separate codeword, which means that the two codewords are interleaved on a nibble basis. Each codeword is thus formed by four nibbles from the data and two nibbles from the checksum. The Reed-Solomon code has the capability to detect and correct a single nibble error anywhere in a single codeword. Since the two codewords are interleaved, any two adjacent nibble errors can be corrected. Also device errors in 8-bit wide memories can be protected since the 8-bits would be protected by the two separate codewords.

The SpaceWire Link Interfaces is configured as follows:

- Support for RMAP CRC calculation
- No support for RMAP protocol

2.6 Signals

The common architecture has the external signals shown in table 9.

Table 9. External signals

Name	Usage	Direction	Polarity	CID
clk	Main system clock	In	-	All
resetn	System reset	In	Low	All
errorn	Processor error mode indicator	Out-Tri	Low	All
dsuen	DSU Enable	In	High	All
dsubre	DSU Break	In	High	All
dsuact	DSU Active	Out	High	All
dsutx	DSU UART transmit data	Out	Low	All
dsurx	DSU UART receive data	In	Low	All
dsutck	JTAG Clock	In	-	-
dsutms	JTAG Mode	In	High	-
dsutdi	JTAG Input	In	High	-
dsutdo	JTAG Output	Out	High	-
address[27:0]	Memory word address	Out	High	All
data[31:0]	Memory data bus	BiDir	High	All
cb[15:0]	Memory checkbits	BiDir	High	All 1) 2)
ramsn[4:0]	SRAM chip selects	Out	Low	All 1)
ramoen[4:0]	SRAM output enable	Out	Low	All 1)
rwen[3:0]	SRAM write enable strobe	Out	Low	All 2)
ramben[3:0]	SRAM read/write byte enable	Out	Low	All 2)
oen	Output enable	Out	Low	All
writen	Write strobe	Out	Low	All
read	Read cycle indicator	Out	High	All
iosn	I/O area chip select	Out	Low	All
romsn[3:0]	PROM chip select	Out	Low	All
brdyn	Bus ready	In	Low	All
bexcn	Bus exception	In	Low	All
gpio[1:0]	Configuring PROM data width at reset. 8-bit data width when “00”, and 32-bit when “10”.	In	-	All
gpio[2]	Enabling EDAC usage for PROM at reset.	In	High	All
sa[14:0]	SDRAM address	Out	High	4, 5, 6
sd[31:0]	SDRAM data	BiDir	High	4, 5, 6
scb[15:0]	SDRAM check bits	BiDir	High	4, 5, 6 1)
sdcsn[1:0]	SDRAM chip select	Out	Low	4, 5, 6, 7, 8
sdwen	SDRAM write enable	Out	Low	4, 5, 6, 7, 8
sdrasn	SDRAM row address strobe	Out	Low	4, 5, 6, 7, 8
sdcasn	SDRAM column address strobe	Out	Low	4, 5, 6, 7, 8
sddqm[5:0]	SDRAM data mask	Out	Low	4, 5, 6, 7, 8 1) 2)

Table 9. External signals

Name	Usage	Direction	Polarity	CID
txd[2:1]	UART 2/1 transmit data	Out	Low	All
rxd[2:1]	UART 2/1 receive data	In	Low	All
rtsn[2:1]	UART 2/1 ready to send	Out	Low	All
ctsn[2:1]	UART 2/1 clear to send	In	Low	All
wdogn	Watchdog output	Out-Tri	Low	All
gpio[15:0]	General purpose I/O port (also used for PROM and SpaceWire interface configuration at reset)	BiDir	High	All
gpio[15:1]	Programmable interrupt inputs	In	-	All
gpio[31:16]	Extended general purpose I/O port	BiDir	High	7
spw_clk	Transmitter default run-state clock	In	-	2, 4, 5, 7
spw_rxd[2:0]	Data input, positive	In, LVDS	High	2, 4, 5, 7 1)
spw_rxdn[2:0]	Data input, negative	In, LVDS	Low	2, 4, 5, 7 1)
spw_rxs[2:0]	Strobe input, positive	In, LVDS	High	2, 4, 5, 7 1)
spw_rxsn[2:0]	Strobe input, negative	In, LVDS	Low	2, 4, 5, 7 1)
spw_txd[2:0]	Data output, positive	Out, LVDS	High	2, 4, 5, 7 1)
spw_txdn[2:0]	Data output, negative	Out, LVDS	Low	2, 4, 5, 7 1)
spw_txs[2:0]	Strobe output, positive	Out, LVDS	High	2, 4, 5, 7 1)
spw_txsn[2:0]	Strobe output, negative	Out, LVDS	Low	2, 4, 5, 7 1)
gpio[5:3]	Configuring SpaceWire start-up and default transmit rate at reset. “000” corresponds to *spw_clk* frequency divided by 1. “001” corresponds to the *spw_clk* frequency divided by 2, etc.	In	-	2, 4, 5, 7
cantx[1:0]	CAN transmit data	Out	Low	1, 5 1)
canrx[1:0]	CAN receive data	In	Low	1, 5 1)
busainen[1:0]	Enable for the A receiver	Out	High	1, 3 1)
busainp[1:0]	Positive data input from the A receiver	In	High	1, 3 1)
busainn[1:0]	Negative data input from the A receiver	In	Low	1, 3 1)
busbinen[1:0]	Enable for the B receiver	Out	High	1, 3 1)
busbinp[1:0]	Positive data to the B receiver	In	High	1, 3 1)
busbinn[1:0]	Negative data to the B receiver	In	Low	1, 3 1)
busaoutin[1:0]	Inhibit for the A transmitter	Out	High	1, 3 1)
busaoutp[1:0]	Positive data to the A transmitter	Out	High	1, 3 1)
busaoutn[1:0]	Negative data to the A transmitter	Out	Low	1, 3 1)
busboutin[1:0]	Inhibit for the B transmitter	Out	High	1, 3 1)
busboutp[1:0]	Positive output to the B transmitter	Out	High	1, 3 1)
busboutn[1:0]	Negative output to the B transmitter	Out	Low	1, 3 1)

Table 9. External signals

Name	Usage	Direction	Polarity	CID
pci_clk	PCI clock	In	-	6
pci_rst	PCI reset	BiDir	Low	6
pci_frame	Cycle frame	BiDir	Low	6
pci_irdy	Initiator ready	BiDir	Low	6
pci_trdy	Target ready	BiDir	Low	6
pci_stop	Stop	BiDir	Low	6
pci_idsel	Device select during configuration	In	High	6
pci_devsel	Device select	BiDir	Low	6
pci_par	Parity signal	BiDir	High	6
pci_perr	Parity error	BiDir	Low	6
pci_serr	System error	BiDir	Low	6
pci_req	Request signal	BiDir	Low	6
pci_gnt	Grant signal	In	Low	6
pci_host	System host	In	Low	6
pci_cbe[3:0]	Bus command and byte enable	BiDir	Low	6
pci_ad[31:0]	Address and Data bus	BiDir	High	6
pci_int[3:0]	Interrupt signals	BiDir	Low	6
pci_arb_req[7:0]	PCI request	In	Low	6 [1)]
pci_arb_gnt[7:0]	PCI grant	Out	Low	6 [1)]

Table 9. External signals

Name	Usage	Direction	Polarity	CID
e0mdio	Management Interface Data Input/Output	BiDir	-	6, 8
e0mdc	Management Interface Data Clock	Out	High	6, 8
e0mdint	Management Interface Data Interrupt	In	High	6, 8
e0rx_clk	Receiver clock	In	-	6, 8
e0rx_dv	Receiver data valid	In	High	6, 8
e0rx_crs	Receiver carrier sense	In	High	6, 8
e0rx_er	Receiver error	In	High	6, 8
e0rx_col	Receiver collision	In	High	6, 8
e0rxd[3:0]	Receiver input data	In	-	6, 8
e0tx_clk	Transmitter clock	In	-	6, 8
e0tx_en	Transmitter enable	Out	High	6, 8
e0txd[3:0]	Transmitter output data	Out	-	6, 8
e1mdio	Management Interface Data Input/Output	BiDir	-	8
e1mdc	Management Interface Data Clock	Out	High	8
e1mdint	Management Interface Data Interrupt	In	High	8
e1rx_clk	Receiver clock	In	-	8
e1rx_dv	Receiver data valid	In	High	8
e1rx_crs	Receiver carrier sense	In	High	8
e1rx_er	Receiver error	In	High	8
e1rx_col	Receiver collision	In	High	8
e1rxd[3:0]	Receiver input data	In	-	8
e1tx_clk	Transmitter clock	In	-	8
e1tx_en	Transmitter enable	Out	High	8
e1txd[3:0]	Transmitter output data	Out	-	8

Note 1: All bus signal elements are not used in all listed configurations. See pin assignment at end of document for details.

Note 2: Refer to signal definitions of each memory controller for detailed usage of these signals.

3 LEON3FT - Fault-Tolerant SPARC V8 Processor

3.1 Overview

LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance with low complexity and low power consumption.

The LEON3 core has the following main features: 7-stage pipeline with Harvard architecture, separate instruction and data caches, hardware multiplier and divider, on-chip debug support and multiprocessor extensions.

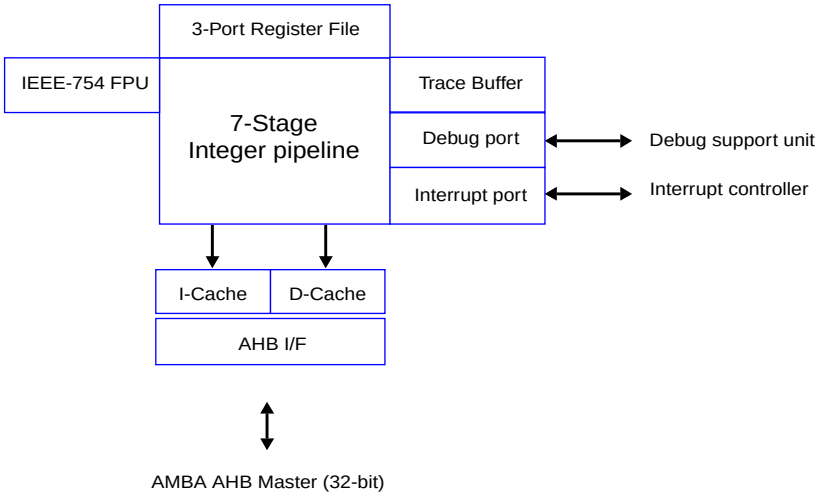


Figure 3. LEON3 processor

3.2 LEON3 integer unit

3.2.1 Overview

The LEON3 integer unit implements the integer part of the SPARC V8 instruction set. The implementation is focused on high performance and low complexity. The LEON3 integer unit has the following main features:

- 7-stage instruction pipeline
- Separate instruction and data cache interface
- Support for 8 register windows
- Radix-2 divider (non-restoring)

Figure 4 shows a block diagram of the integer unit.

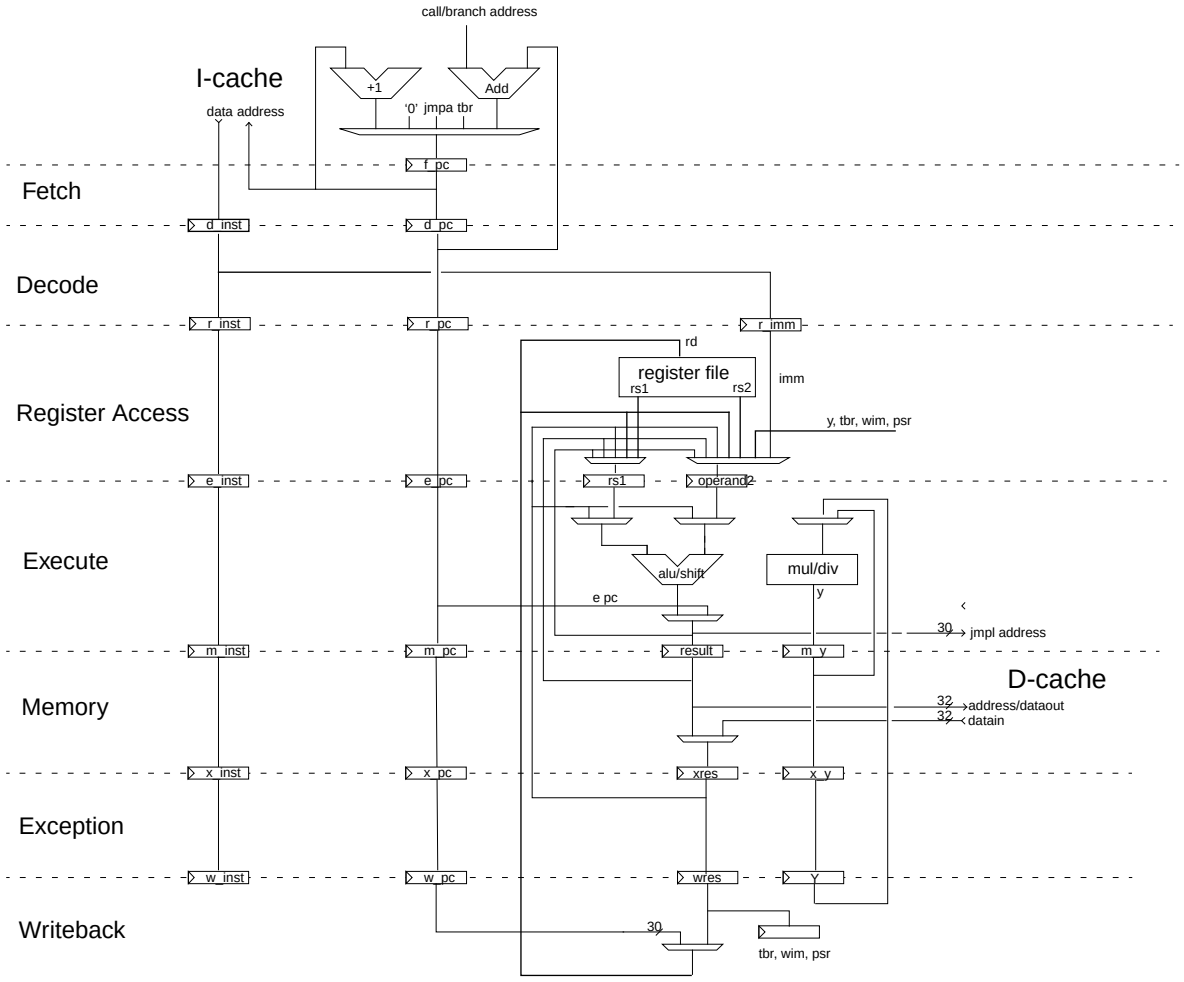


Figure 4. LEON3 integer unit datapath diagram

3.2.2 Instruction pipeline

The LEON integer unit uses a single instruction issue pipeline with 7 stages:

1. FE (Instruction Fetch): If the instruction cache is enabled, 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.
2. DE (Decode): The instruction is decoded and the CALL and Branch target addresses are generated.
3. RA (Register access): Operands are read from the register file or from internal data bypasses.
4. EX (Execute): ALU, logical, and shift operations are performed. For memory operations (e.g., LD) and for JMPL/RETT, the address is generated.
5. ME (Memory): Data cache is accessed. Store data read out in the execution stage is written to the data cache at this time.
6. XC (Exception) Traps and interrupts are resolved. For cache reads, the data is aligned as appropriate.
7. WR (Write): The result of any ALU, logical, shift, or cache operations are written back to the register file.

Table 10 lists the cycles per instruction (assuming cache hit and no icc or load interlock):

Table 10. Instruction timing

Instruction	Cycles
JMPL, RETT	3
Double load	2
Single store	2
Double store	3
SMUL/UMUL	4*
SDIV/UDIV	35
Taken Trap	5
Atomic load/store	3
All other instructions	1

3.2.3 SPARC Implementor's ID

Aeroflex Gaisler is assigned number 15 (0xF) as SPARC implementor's identification. This value is hard-coded into bits 31:28 in the %psr register. The version number for LEON3 is 3, which is hard-coded in to bits 27:24 of the %psr.

3.2.4 Divide instructions

Full support for SPARC V8 divide instructions is provided (SDIV, UDIV, SDIVCC & UDIVCC). The divide 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.

3.2.5 Multiply instructions

The LEON processor supports the SPARC integer multiply instructions UMUL, SMUL UMULCC and SMULCC. These instructions perform a 32x32-bit integer multiply, producing a 64-bit result. SMUL and SMULCC performs signed multiply while UMUL and UMULCC performs unsigned multiply. UMULCC and SMULCC also set the condition codes to reflect the result. The multiply instructions are performed using a 16x16 signed hardware multiplier, which is iterated four times.

3.2.6 Multiply and accumulate instructions

To accelerate DSP algorithms, two multiply&accumulate instructions are implemented: UMAC and SMAC. The UMAC performs an unsigned 16-bit multiply, producing a 32-bit result, and adds the result to a 40-bit accumulator made up by the 8 lsb bits from the %y register and the %asr18 register. The least significant 32 bits are also written to the destination register. SMAC works similarly but performs signed multiply and accumulate. The MAC instructions execute in one clock but have two clocks latency, meaning that one pipeline stall cycle will be inserted if the following instruction uses the destination register of the MAC as a source operand.

Assembler syntax:

```
umacrs1, reg_imm, rd
smacrs1, reg_imm, rd
```

Operation:

```
prod[31:0] = rs1[15:0] * reg_imm[15:0]
result[39:0] = (Y[7:0] & %asr18[31:0]) + prod[31:0]
(Y[7:0] & %asr18[31:0]) = result[39:0]
rd = result[31:0]
```

%asr18 can be read and written using the RDASR and WRASR instructions.

3.2.7 Hardware breakpoints

The integer unit can be configured to include up to four hardware breakpoints. Each breakpoint consists of a pair of application-specific registers (%asr24/25, %asr26/27, %asr28/30 and %asr30/31) registers; one with the break address and one with a mask:

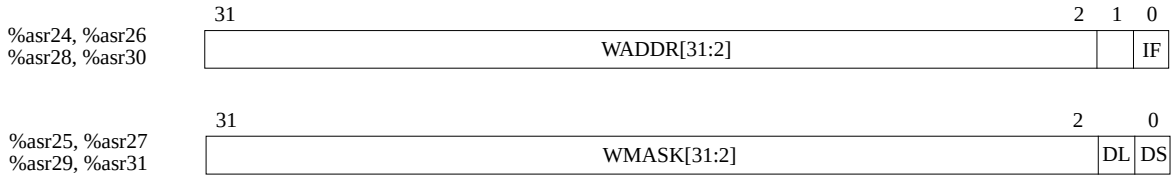


Figure 5. Watch-point registers

Any binary aligned address range can be watched - the range is defined by the WADDR field, masked by the WMASK field (WMASK[x] = 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 will effectively disable the breakpoint function.

3.2.8 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 (see the DSU description). The size of the trace buffer is 2 kByte. The trace buffer is 128 bits wide, and 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 section 6.4. Note that in multi-processor systems, each processor has its own trace buffer allowing simultaneous tracing of all instruction streams.

3.2.9 Processor configuration register

The application specific register 17 (%asr17) provides information on how various configuration options were set during synthesis. This can be used to enhance the performance of software, or to support enumeration in multi-processor systems. The register can be accessed through the RDASR instruction, and has the following layout:

Figure 6. LEON3 configuration register (%asr17)

Field Definitions:

[31:28]: Processor index. In multi-processor systems, each LEON core gets a unique index to support enumeration.
[14]: Disable write error trap (DWT). When set, a write error trap (tt = 0x2b) will be ignored. Set to zero after reset.
[13]: Single-vector trapping (SVT) enable. If set, will enable single-vector trapping. Fixed to zero if SVT is not implemented. Set to zero after reset.
[12]: Load delay. If set, the pipeline uses a 2-cycle load delay. Otherwise, a 1-cycle load delay i s used.
[11:10]: FPU option. “00” = no FPU; “01” = GRFPU; “10” = Meiko FPU, “11” = GRFPU-Lite
[9]: If set, the optional multiply-accumulate (MAC) instruction is available
[8]: If set, the SPARC V8 multiply and divide instructions are available.
[7:5]: Number of implemented watchpoints (0 - 4)
[4:0]: Number of implemented registers windows corresponds to NWIN+1.

3.2.10 Exceptions

LEON adheres to the general SPARC trap model. The table below shows the implemented traps and their individual priority. When PSR (processor status register) bit ET=0, an exception trap causes the processor to halt execution and enter error mode, and the external error signal will then be asserted.

Table 11. Trap allocation and priority

Trap	TT	Pri	Description
reset	0x00	1	Power-on reset
write error	0x2b	2	write buffer error
instruction_access_error	0x01	3	Error during instruction fetch
illegal_instruction	0x02	5	UNIMP or other un-implemented instruction
privileged_instruction	0x03	4	Execution of privileged instruction in user mode
fp_disabled	0x04	6	FP instruction while FPU disabled
cp_disabled	0x24	6	CP instruction while Co-processor disabled
watchpoint_detected	0x0B	7	Hardware breakpoint match
window_overflow	0x05	8	SAVE into invalid window
window_underflow	0x06	8	RESTORE into invalid window
register_hadrware_error	0x20	9	register file EDAC error (LEON-FT only)
mem_address_not_aligned	0x07	10	Memory access to un-aligned address
fp_exception	0x08	11	FPU exception
cp_exception	0x28	11	Co-processor exception
data_access_exception	0x09	13	Access error during load or store instruction
tag_overflow	0x0A	14	Tagged arithmetic overflow
divide_exception	0x2A	15	Divide by zero
interrupt_level_1	0x11	31	Asynchronous interrupt 1
interrupt_level_2	0x12	30	Asynchronous interrupt 2
interrupt_level_3	0x13	29	Asynchronous interrupt 3
interrupt_level_4	0x14	28	Asynchronous interrupt 4
interrupt_level_5	0x15	27	Asynchronous interrupt 5
interrupt_level_6	0x16	26	Asynchronous interrupt 6
interrupt_level_7	0x17	25	Asynchronous interrupt 7
interrupt_level_8	0x18	24	Asynchronous interrupt 8
interrupt_level_9	0x19	23	Asynchronous interrupt 9
interrupt_level_10	0x1A	22	Asynchronous interrupt 10
interrupt_level_11	0x1B	21	Asynchronous interrupt 11
interrupt_level_12	0x1C	20	Asynchronous interrupt 12
interrupt_level_13	0x1D	19	Asynchronous interrupt 13
interrupt_level_14	0x1E	18	Asynchronous interrupt 14
interrupt_level_15	0x1F	17	Asynchronous interrupt 15
trap_instruction	0x80 - 0xFF	16	Software trap instruction (TA)

3.2.11 Single vector trapping (SVT)

Single-vector trapping (SVT) is an SPARC V8e option to reduce code size for embedded applications. When enabled, any taken trap will always jump to the reset trap handler (%tbr.tba + 0). The trap type will be indicated in %tbr.tt, and must be decoded by the shared trap handler. SVT is enabled by setting bit 13 in %asr17. The model must also be configured with the SVT generic = 1.

3.2.12 Address space identifiers (ASI)

In addition to the address, a SPARC processor also generates an 8-bit address space identifier (ASI), providing up to 256 separate, 32-bit address spaces. During normal operation, the LEON3 processor accesses instructions and data using ASI 0x8 - 0xB as defined in the SPARC standard. Using the LDA/STA instructions, alternative address spaces can be accessed. The table shows the ASI usage for LEON. Only ASI[5:0] are used for the mapping, ASI[7:6] have no influence on operation.

Table 12. ASI usage

ASI	Usage
0x01	Forced cache miss
0x02	System control registers (cache control register)
0x08, 0x09, 0x0A, 0x0B	Normal cached access (replace if cacheable)
0x0C	Instruction cache tags
0x0D	Instruction cache data
0x0E	Data cache tags
0x0F	Data cache data
0x10	Flush instruction cache
0x11	Flush data cache

3.2.13 Power-down

The processor can be configured to include 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

During power-down, the pipeline is halted until the next interrupt occurs. Signals inside the processor pipeline and caches are then static, reducing power consumption from dynamic switching.

3.2.14 Processor reset operation

The processor is reset by asserting the RESET input for at least 4 clock cycles. The following table indicates the reset values of the registers which are affected by the reset. All other registers maintain their value (or are undefined).

Table 13. Processor reset values

Register	Reset value
PC (program counter)	0x0
nPC (next program counter)	0x4
PSR (processor status register)	ET=0, S=1

By default, the execution will start from address 0. This can be overridden by setting the RSTADDR generic in the model to a non-zero value. The reset address is however always aligned on a 4 kByte boundary.

3.2.15 Cache sub-system

The LEON3 processor implements a Harvard architecture with separate instruction and data buses, connected to two independent cache controllers. Both instruction and data cache controllers can be separately configured to implement a direct-mapped cache or a multi-set cache with set associativity of 2 - 4. The set size is configurable to 1 - 256 kByte, divided into cache lines with 16 or 32 bytes of data. In multi-set configurations, one of three replacement policies can be selected: least-recently-used (LRU), least-recently-replaced (LRR) or (pseudo-) random. If the LRR algorithm can only be used when the cache is 2-way associative. A cache line can be locked in the instruction or data cache preventing it from being replaced by the replacement algorithm.

NOTE: The LRR algorithm uses one extra bit in tag rams to store replacement history. The LRU algorithm needs extra flip-flops per cache line to store access history. The random replacement algorithm is implemented through modulo-N counter that selects which line to evict on cache miss.

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

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

3.3 Instruction cache

3.3.1 Operation

The instruction cache can be configured as a direct-mapped cache or as a multi-set cache with associativity of 2 - 4 implementing either LRU or random replacement policy or as 2-way associative cache implementing LRR algorithm. The set size is configurable to 1 - 64 kByte and divided into cache lines of 16- 32 bytes. Each line has a cache tag associated with it consisting of a tag field, valid field with one valid bit for each 4-byte sub-block and optional LRR and lock bits. On an instruction cache miss to a cachable location, the instruction is fetched and the corresponding tag and data line updated. In a multi-set configuration a line to be replaced is chosen according to the replacement policy.

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 (streaming). 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 burst 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=0x1) will be generated.

3.3.2 Instruction cache tag

A instruction cache tag entry consists of several fields as shown in figure 7:

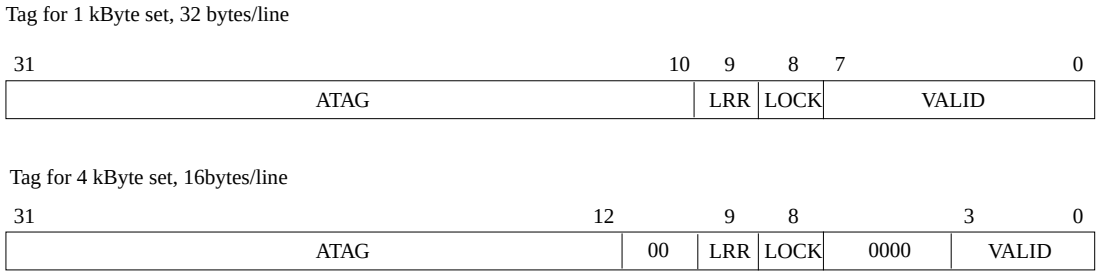


Figure 7. Instruction cache tag layout examples

Field Definitions:

[31:10]: Address Tag (ATAG) - Contains the tag address of the cache line.
[9]: LRR - Used by LRR algorithm to store replacement history, otherwise 0.
[8]: LOCK - Locks a cache line when set. 0 if cache locking not implemented.
[7:0]: Valid (V) - When set, the corresponding sub-block of the cache line contains valid data. These bits is set when a sub-block is filled due to a successful 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. V[0] corresponds to address 0 in the cache line, V[1] to address 1, V[2] to address 2 and so on.

NOTE: only the necessary bits will be implemented in the cache tag, depending on the cache configuration. As an example, a 4 kByte cache with 16 bytes per line would only have four valid bits and 20 tag bits. The cache rams are sized automatically by the ram generators in the model.

3.4 Data cache

3.4.1 Operation

The data cache can be configured as a direct-mapped cache or as a multi-set cache with associativity of 2 - 4 implementing either LRU or (pseudo-) random replacement policy or as 2-way associative cache implementing LRR algorithm. The set size is configurable to 1 - 64 kByte and divided into cache lines of 16 - 32 bytes. Each line has a cache tag associated with it consisting of a tag field, valid field with one valid bit for each 4-byte sub-block and optional lock and LRR bits. 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 write-miss. In a multi-set configuration a line to be replaced on read-miss is chosen according to the replacement policy. 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=0x9) will be generated.

3.4.2 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 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 read in to 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 instructions 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.

3.4.3 Data cache tag

A data cache tag entry consists of several fields as shown in figure 8:

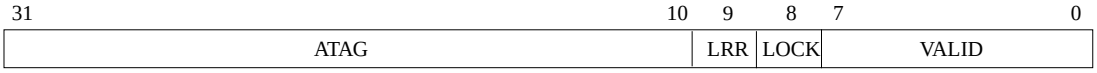


Figure 8. Data cache tag layout

Field Definitions:

[31:10]: Address Tag (ATAG) - Contains the address of the data held in the cache line.
[9]: LRR - Used by LRR algorithm to store replacement history. ‘0’ if LRR is not used.
[8]: LOCK - Locks a cache line when set. ‘0’ if instruction cache locking was not enabled in the configuration.
[3:0]: Valid (V) - When set, the corresponding sub-block of the cache line contains valid data. These bits is set when a sub-block is filled due to a successful cache miss; a cache fill which results in a memory error will leave the valid bit unset. V[0] corresponds to address 0 in the cache line, V[1] to address 1, V[2] to address 2 and V[3] to address 3.

NOTE: only the necessary bits will be implemented in the cache tag, depending on the cache configuration. As an example, a 2 kByte cache with 32 bytes per line would only have eight valid bits and 21 tag bits. The cache rams are sized automatically by the ram generators in the model.

3.5 Additional cache functionality

3.5.1 Cache flushing

Both instruction and data cache are flushed by executing the FLUSH instruction. The instruction cache is also flushed by setting the FI bit in the cache control register, or by writing to any location with ASI=0x15. The data cache is also flushed by setting the FD bit in the cache control register, or by writing to any location with ASI=0x16. Cache flushing takes one cycle per cache line, during which the IU will not be halted, but during which the caches are disabled. When the flush operation is completed, the cache will resume 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 (trap=0x09) if attempted.

3.5.2 Diagnostic cache access

Tags and data in the instruction and data cache can be accessed through ASI address space 0xC, 0xD, 0xE and 0xF by executing LDA and STA instructions. Address bits making up the cache offset will be used to index the tag to be accessed while the least significant bits of the bits making up the address tag will be used to index the cache set.

Diagnostic read of tags is possible by executing an LDA instruction with ASI=0xC for instruction cache tags and ASI=0xE 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=0xD for instruction cache data and ASI=0xF for data cache data. The sub-block to be read in the indexed cache line and set is selected by A[4:2].

The tags can be directly written by executing a STA instruction with ASI=0xC for the instruction cache tags and ASI=0xE 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 (see above) and the valid bits are written with the D[7:0] of the write data. Bit D[9] is written into the LRR bit (if enabled) and D[8] is written into the lock bit (if enabled). 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].

3.5.3 Cache Control Register

The operation of the instruction and data caches is controlled through a common Cache Control Register (CCR) (figure 9). Each cache can be in one of three modes: disabled, enabled and frozen. 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 in sync with the main memory as if it was enabled, but no new lines are allocated on read misses.

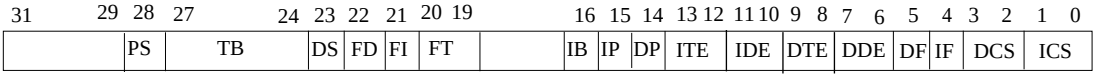


Figure 9. Cache control register

[28]: Parity Select [PS] - if set diagnostic read will return 4 check bits in the lsb bits, otherwise tag or data word is returned.

[27:24]: Test Bits [TB] - if set, check bits will be xored with test bits TB during diagnostic write

[23]: Data cache snoop enable [DS] - if set, will enable data cache snooping.

[22]: Flush data cache (FD). If set, will flush the instruction cache. Always reads as zero.

[21]: Flush Instruction cache (FI). If set, will flush the instruction cache. Always reads as zero.

[20:19]: FT scheme: “00” = no FT, “01” = 4-bit checking implemented

[16]: Instruction burst fetch (IB). This bit enables burst fill during instruction fetch.

[15]: Instruction cache flush pending (IP). This bit is set when an instruction cache flush operation is in progress.

[14]: Data cache flush pending (DP). This bit is set when an data cache flush operation is in progress.

[13:12]: Instruction Tag Errors (ITE) - Number of detected parity errors in the instruction tag cache.

[11:10]: Instruction Data Errors (IDE) - Number of detected parity errors in the instruction data cache.

[9:8]: Data Tag Errors (DTE) - Number of detected parity errors in the data tag cache.

[7:6]: Data Data Errors (IDE) - Number of detected parity errors in the data data cache.

[5]: Data Cache Freeze on Interrupt (DF) - If set, the data cache will automatically be frozen when an asynchronous interrupt is taken.

[4]: Instruction Cache Freeze on Interrupt (IF) - If set, the instruction cache will automatically be frozen when an asynchronous interrupt is taken.

[3:2]: Data Cache state (DCS) - Indicates the current data cache state according to the following: X0= disabled, 01 = frozen, 11 = enabled.

[1:0]: Instruction Cache state (ICS) - Indicates the current data cache state according to the following: 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 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 and 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 enabled again by enabling it in the CCR. This is typically done at the end of the interrupt handler before control is returned to the interrupted task.

3.5.4 Cache configuration registers

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

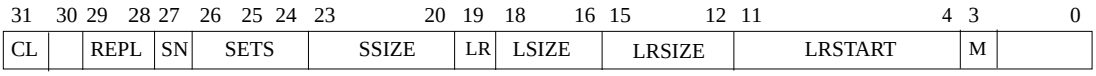


Figure 10. Cache configuration register

[31]: Cache locking (CL). Set if cache locking is implemented.

[29:28]: Cache replacement policy (REPL). 00 - no replacement policy (direct-mapped cache), 01 - least recently used (LRU), 10 - least recently replaced (LRR), 11 - random

[27]: Cache snooping (SN). Set if snooping is implemented.

[26:24]: Cache associativity (SETS). Number of sets in the cache: 000 - direct mapped, 001 - 2-way associative, 010 - 3-way associative, 011 - 4-way associative

[23:20]: Set size (SSIZE). Indicates the size (kBbyte) of each cache set. Size = 2^{SIZE}

[19]: Local ram (LR). Set if local scratch pad ram is implemented.

[18:16]: Line size (LSIZE). Indicated the size (words) of each cache line. Line size = 2^{LSZ}

[15:12]: Local ram size (LRSZ). Indicates the size (kBbyte) of the implemented local scratch pad ram. Local ram size = 2^{LRSZ}

[11:4]: Local ram start address. Indicates the 8 most significant bits of the local ram start address.

[3]: MMU present. This bit is set to ‘1’ if an MMU is present.

All cache registers are accessed through load/store operations to the alternate address space (LDA/STA), using ASI = 2. The table below shows the register addresses:

Table 14. ASI 2 (system registers) address map

Address	Register
0x00	Cache control register
0x04	Reserved
0x08	Instruction cache configuration register
0x0C	Data cache configuration register

3.5.5 Software consideration

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

```
flush
set 0x81000f, %g1
sta%g1, [%g0] 2
```

3.6 Memory management unit

The memory management unit (MMU) is compatible with the SPARC V8 reference MMU. For details on operation, see the SPARC V8 manual.

3.6.1 ASI mappings

When the MMU is used, the following ASI mappings are added:

Table 15. MMU ASI usage

ASI	Usage
0x10	Flush page
0x10	MMU flush page
0x13	MMU flush context
0x14	MMU diagnostic dcache context access
0x15	MMU diagnostic icache context access
0x19	MMU registers
0x1C	MMU bypass
0x1D	MMU diagnostic access

3.6.2 Cache operation

When the MMU is disabled, the caches operate as normal with physical address mapping. When the MMU is enabled, the caches tags store the virtual address and also include an 8-bit context field. AHB cache snooping is not available when the MMU is enabled.

3.6.3 MMU registers

The following MMU registers are implemented:

Table 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

The definition of the registers can be found in the SPARC V8 manual.

3.6.4 Translation look-aside buffer (TLB)

The MMU can be configured to use a shared TLB, or separate TLB for instructions and data. The number of TLB entries can be set to 2 - 32 in the configuration record. The organisation of the TLB and number of entries is not visible to the software and does thus not require any modification to the operating system.

3.7 Floating-point unit and custom co-processor interface

3.7.1 Floating-point unit - Lite (GRFPU-Lite)

GRFPU-Lite is a smaller version of GRFPU, suitable for FPGA implementations with limited logic resources. The GRFPU-Lite is not pipelined and executes thus only one instruction at a time. To improve performance, the FPU controller (GRLFPC) allows GRFPU-Lite to execute in parallel with the processor pipeline as long as no new FPU instructions are pending. Below is a table of worst-case throughput of the GRFPU-Lite:

Table 17. GRFPU-Lite worst-case instruction timing with GRLFPC

Instruction	Throughput	Latency
FADDS, FADDD, FSUBS, FSUBD,FMULS, FMULD, FSMULD, FITOS, FITOD, FSTOI, FDTOI, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES. FCMPED	8	8
FDIVS	31	31
FDIVD	57	57
FSQRTS	46	46
FSQRTD	65	65

When the GRFPU-Lite is enabled in the model, the version field in %fsr has the value of 3.

3.8 Fault Tolerance

The LEON3FT processor is a derivative of the standard LEON3 SPARC V8 processor, enhanced with fault-tolerance against SEU errors. The fault-tolerance is focused on the protection of on-chip RAM blocks, which are used to implement IU/FPU register files and the cache memory. The LEON3FT processor is functionally identical to the standard LEON3 processor, and this chapter only outlines the FT features.

3.8.1 IU Register file SEU protection

The SEU error detection has no impact on behaviour or timing. An uncorrectable error in the IU register file will cause trap 0x20 (*register_access_error*).

3.8.2 FPU Register file SEU protection

The FPU register file protection scheme is always 4-bit parity with pipeline restart. The SEU error detection has no impact on behaviour or timing, but a correction cycle will delay the current instruction with 6 clock cycles. An uncorrectable error in the FPU register file will cause an (deferred) FPU exception with %fsr.ftt set to 5 (*hardware_error*).

3.8.3 ASR16 register

ASR register 16 (%asr16) is used to control the IU/FPU register file SEU protection. It is possible to disable the SEU protection by setting the IDI/FDI bits, and to inject errors using the ITE/FTE bits. Corrected errors in the register file are counted, and available in ICNT and FCNT fields. The counters saturate at their maximum value (7), and should be reset by software after read-out.

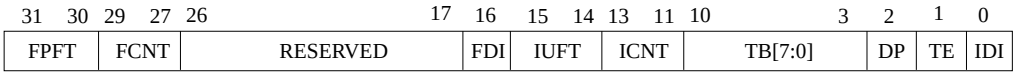


Figure 11. %asr16 - Register protection control register

[31:30]: FP FT ID - Defines which SEU protection is implemented in the FPU (0x1 = 4-bit parity with restart).
[29:27]: FP RF error counter - Number of detected parity errors in the FP register file.
[26:17]: Reserved
[16]: FP RF protection disable (FDI) - Disables FP RF parity protection when set.
[15:14]: IU FT ID - Defines which SEU protection is implemented in the IU (0x2 = 8-bit parity without restart).
[13:11]: IU RF error counter - Number of detected parity errors in the IU register file.
[10:3]: RF Test bits (FTB) - In test mode, these bits are xored with correct parity bits before written to the register file.
[2]: DP ram select (DP) - Only applicable if the IU or FPU register files consists of two dual-port rams. See text below for details.
[1]: IU RF Test Enable - Enables register file test mode. Parity bits are xored with TB before written to the register file.
[0]: IU RF protection disable (IDI) - Disables IU RF parity protection when set.

Table 18. DP ram select usage

TE	DP	Function
1	0	Write to IU register (%i, %l, %o, %g) will only write location of %rs2 Write to FPU register (%f) will only write location of %rs2
1	1	Write to IU register (%i, %l, %o, %g) will only write location of %rs1 Write to IU register (%f) will only write location of %rs1
0	X	IU and FPU registers written nominally

3.8.4 IU register file error injection

The SEU protection feature can be tested by inserting errors into the checkbits. When the RF Test Enable bit (%asr16[1]) is ‘1’, the checkbits will be XORed with the RF Test bits before written to the register file. If the IU and/or FPU register files are implemented with dual-port RAM, then the DP bit defines which of the two register file DPRAMs the checkbits will be written to according to the following table.

3.8.5 Cache memory SEU protection

Each word in the cache tag or data memories is protected by four check bits. An error during cache access will cause a cache line flush, and a re-execution of the failing instruction. This will insure that the complete cache line (tags and data) is refilled from external memory. For every detected error, a counter in the cache control register is incremented. The counters saturate at their maximum value (3), and should be reset by software after read-out. The cache memory check bits can be diagnostically read by setting the PS bit in the cache control register and then perform a normal tag or data diagnostic read.

3.8.6 Data scrubbing

There is generally no need to perform data scrubbing on either IU/FPU register files or the cache memory. During normal operation, the active part of the IU/FPU register files will be flushed to memory on each task switch. This will cause all registers to be checked and corrected if necessary. Since most real-time operating systems performs several task switches per second, the data in the register files will be frequently refreshed.

The similar situation arises for the cache memory. In most applications, the cache memory is significantly smaller than the full application image, and the cache contents is gradually replaced as part of normal operation. For very small programs, the only risk of error build-up is if a part of the application is resident in the cache but not executed for a long period of time. In such cases, executing a cache flush instruction periodically (e.g. once per minute) is sufficient to refresh the cache contents.

3.8.7 Initialisation

After power-on, the check bits in the IU and FPU register files are not initialized. This means that access to an uninitialized (un-written) register could cause a register access trap (tt = 0x20). Such behaviour is considered as a software error, as the software should not read a register before it has been written. It is recommended that the boot code for the processor writes all registers in the IU and FPU register files before launching the main application.

The check bits in the cache memories do not need to be initialized as this is done automatically during cache line filling.

3.9 Signal definitions and reset values

When the processor enters error mode, the *errorn* output is driven active low, else it is in tri-state and therefore might require an additional external pull-up.

The signals and their reset values are described in table 19.

Table 19. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
errorn	Tri-state output	Processor error mode indicator	Low	Tri-state

3.10 Timing

The timing waveforms and timing parameters are shown in figure 12 and are defined in table 20.

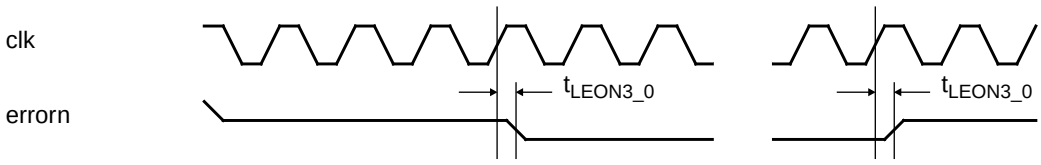


Figure 12. Timing waveforms

Table 20. Timing parameters

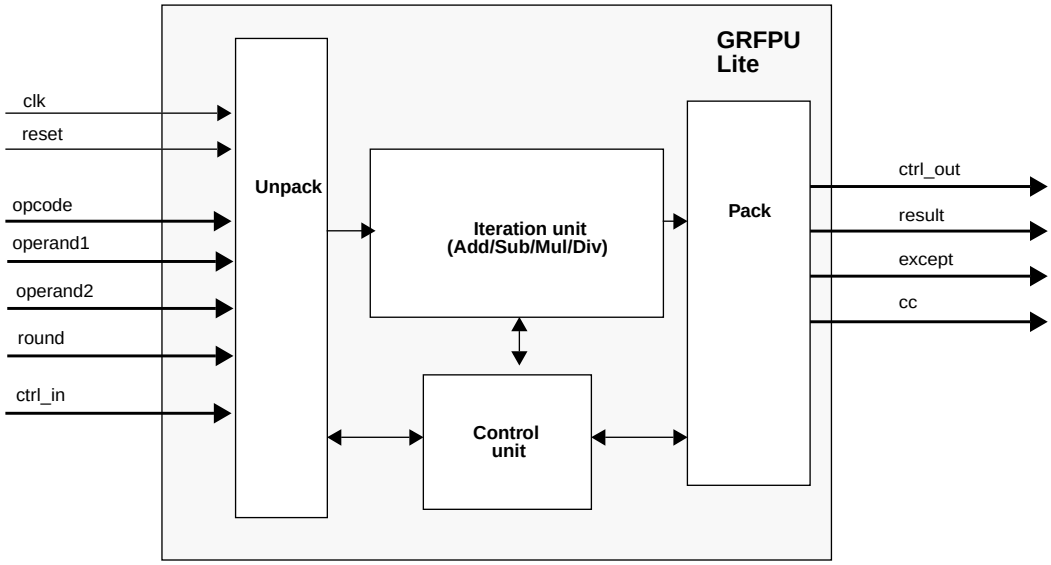
Name	Parameter	Reference edge	Min	Max	Unit
t_{LEON3_0}	clock to output delay	rising *clk* edge	0	25	ns

4 IEEE-754 Floating-Point Unit

4.1 Overview

The floating-point unit implements floating-point operations as defined in IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754) and SPARC V8 standard (IEEE-1754).

Supported formats are single and double precision floating-point numbers. The floating-point unit is not pipelined and executes one floating-point operation at a time.



4.2 Functional Description

4.2.1 Floating-point number formats

The floating-point unit handles floating-point numbers in single or double precision format as defined in IEEE-754 standard.

4.2.2 FP operations

The floating-point unit supports four types of floating-point operations: arithmetic, compare, convert and move. The operations implement all FP instructions specified by SPARC V8 instruction set. All operations are summarized in the table below.

Table 21. :Floating-point operations

Operation	Op1	Op2	Result	Exceptions	Description
Arithmetic operations					
FADDS FADDD	SP DP	SP DP	SP DP	NV, OF, UF, NX	Addition
FSUBS FSUBD	SP DP	SP DP	SP DP	NV, OF, UF, NX	Subtraction
FMULS FMULD FSMULD	SP DP SP	SP DP SP	SP DP DP	NV, OF, UF, NX NV, OF, UF, NX NV,OF, UF	Multiplication
FDIVS FDIVD	SP DP	SP DP	SP DP	NV, OF, UF, NX	Division
FSQRTS FSQRTD	- -	SP DP	SP DP	NV, NX	Square-root
Conversion operations					
FITOS FITOD	-	INT	SP DP	NX -	Integer to floating-point conversion
FSTOI FDTOI	-	SP DP	INT	NV, NX	Floating-point to integer conversion. The result is rounded in round-to-zero mode.
FSTOD FDTOS	-	SP DP	DP SP	NV NV, OF, UF, NX	Conversion between floating-point formats
Comparison operations					
FCMPS FCMPD	SP DP	SP DP	CC	NV	Floating-point compare. Invalid exception is generated if either operand is a signaling NaN.
FCMPES FCMPED	SP DP	SP DP	CC	NV	Floating point compare. Invalid exception is generated if either operand is a NaN (quiet or signaling).
Negate, Absolute value and Move					
FABSS	-	SP	SP	-	Absolute value.
FNEGS	-	SP	SP	-	Negate.
FMOVS		SP	SP	-	Move. Copies operand to result output.

SP - single precision floating-point number

DP - double precision floating-point number

INT - 32 bit integer

CC - condition codes

NV, OF, UF, NX - floating-point exceptions, see section 4.2.3

Below is a table of worst-case throughput of the floating point unit.

Table 22. Worst-case instruction timing

Instruction	Throughput	Latency
FADDS, FADDD, FSUBS, FSUBD,FMULS, FMULD, FSMULD, FITOS, FITOD, FSTOI, FDTOI, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES. FCMPED	8	8
FDIVS	31	31
FDIVD	57	57
FSQRTS	46	46
FSQRTD	65	65

4.2.3 Exceptions

The floating-point unit detects all exceptions defined by the IEEE-754 standard. This includes detection of Invalid Operation (NV), Overflow (OF), Underflow (UF), Division-by-Zero (DZ) and Inexact (NX) exception conditions. Generation of special results such as NaNs and infinity is also implemented.

4.2.4 Rounding

All four rounding modes defined in the IEEE-754 standard are supported: round-to-nearest, round-to-+inf, round-to--inf and round-to-zero.

5 Floating-point unit Controller

5.1 Overview

The Floating-Point Unit Controller is used to attach the floating-point unit (FPU) to the LEON integer unit (IU). It performs decoding and dispatching of the floating-point (FP) operations to the floating-point units as well as managing the floating-point register file, the floating-point state register (FSR) and the floating-point deferred-trap queue (FQ).

The floating-point unit is not pipelined and executes only one instruction at a time. To improve performance, the controller allows the floating-point unit to execute in parallel with the processor pipeline as long as no new floating-point instructions are pending.

5.2 Floating-Point register file

The floating-point register file contains 32 32-bit floating-point registers (%f0-%f31). The register file is accessed by floating-point load and store instructions (LDF, LDDF, STD, STDF) and floating-point operate instructions (FPop).

In the FT-version, the floating-point register file is protected using 4-bit parity per 32-bit word. The controller is capable of detecting and correcting one bit error per byte. Errors are corrected using the instruction restart function in the IU.

5.3 Floating-Point State Register (FSR)

The controller manages the floating-point state register (FSR) containing FPU mode and status information. All fields of the FSR register as defined in SPARC V8 specification are implemented and managed by the controller conform to the SPARC V8 specification and IEEE-754 standard.

The non-standard bit of the FSR register is not used, all floating-point operations are performed in standard IEEE-compliant mode.

Following floating-point trap types never occur and are therefore never set in the ftt field:
- unimplemented_FPop: all FPop operations are implemented
- unfinished_FPop: all FPop operation complete with valid result
- invalid_fp_register: no check that double-precision register is 0 mod 2 is performed

The controller implements the *qne* bit of the FSR register which reads 0 if the floating-point deferred-queue (FQ) is empty and 1 otherwise. The FSR is accessed using LDFSR and STFSR instructions.

5.4 Floating-Point Exceptions and Floating-Point Deferred-Queue

The floating-point unit implements the SPARC deferred trap model for floating-point exceptions (fp_exception). A floating-point exception is caused by a floating-point instruction performing an operation resulting in one of following conditions:

- an operation raises IEEE floating-point exception (ftt = IEEE_754_exception) e.g. executing invalid operation such as 0/0 while the NVM bit of the TEM field id set (invalid exception enabled).
- sequence error: abnormal error condition in the FPU due to the erroneous use of the floating-point instructions in the supervisor software.
- hardware_error: uncorrectable parity error is detected in the FP register file

The trap is deferred to the next floating-point instruction (FPop, FP load/store, FP branch) following the trap-inducing instruction. When the trap is taken the floating-point deferred-queue (FQ) contains the trap-inducing instruction.

After the trap is taken the *qne* bit of the FSR is set and remains set until the FQ is emptied. STDFQ instruction reads a double-word from the floating-point deferred queue, the first word is the address of the instruction and the second word is the instruction code.

6 Hardware Debug Support Unit

6.1 Overview

To simplify debugging on target hardware, the LEON3 processor implements a debug mode during which the pipeline is idle and the processor is controlled through a special debug interface. The LEON3 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 AHB master. An external debug host can therefore access the DSU through several different interfaces.

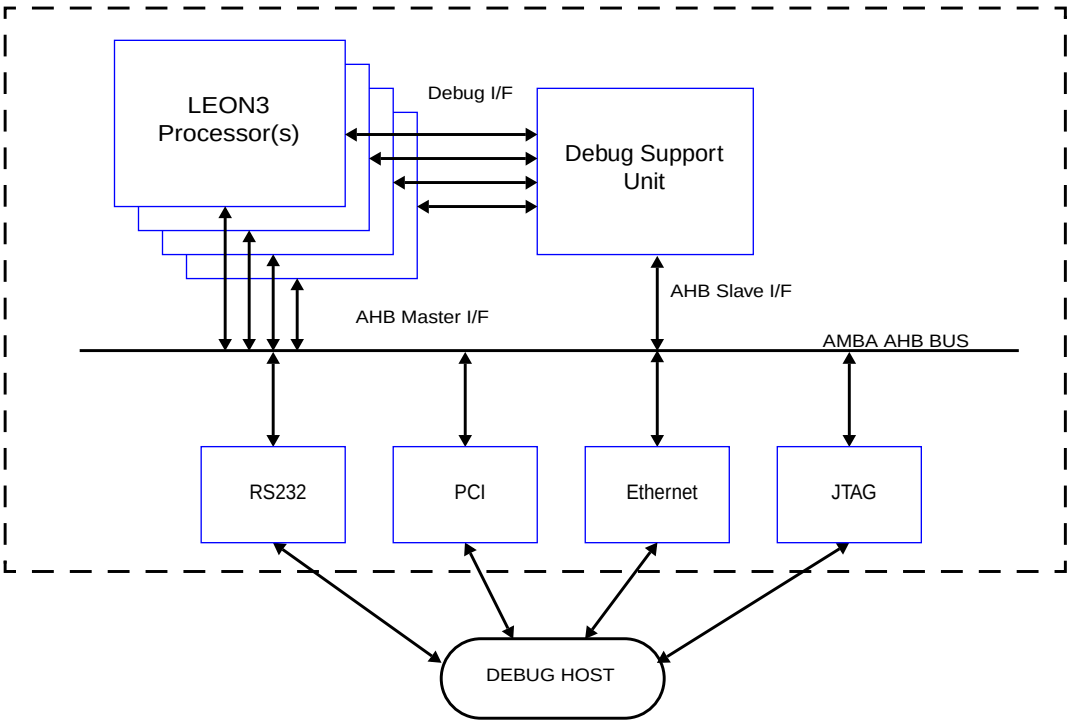


Figure 13. LEON3/DSU Connection

6.2 Operation

Through the DSU AHB slave interface, any AHB master 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 pipeline 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 instruction (ta 1)
- integer unit hardware breakpoint/watchpoint hit (trap 0xb)
- rising edge of the external break signal (DSUBRE)
- setting the break-now (BN) bit in the DSU control 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
- one of the processors in a multiprocessor system has entered the debug mode
- DSU breakpoint hit

The debug mode can only be entered when the debug support unit is enabled through an external signal (DSUEN). When the 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 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 deasserting DSUEN. The timer unit will be re-enabled and execution will continue 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. The error mode can be reset and the processor restarted at any address.

When a processor is in the debug mode, an access to ASI diagnostic area is forwarded to the IU which performs access with ASI equal to value in the DSU ASI register and address consisting of 20 LSB bits of the original address.

6.3 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, the information stored is indicated in the table below:

Table 23. AHB Trace buffer data 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 bit (EN) in the trace 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 reset, or when a AHB breakpoint is hit. Tracing is temporarily suspended when the processor enters debug mode. Note that neither the trace buffer memory nor the breakpoint registers (see below) can be read/written by software when the trace buffer is enabled.

6.4 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, the information stored is indicated in the table below:

Table 24. Instruction trace buffer data 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. Multi-cycle 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 third in case of STD). Bit 126 is set on the second and third entry to indicate this. A double load (LDD) 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 MSB 32 bits of the results in bit [63:32] while the second entry puts the LSB 32 bits in this field.

When the processor enters debug mode, tracing is suspended. The trace buffer and the trace buffer control register can be read and written while the processor is in the debug mode. During the instruction tracing (processor in normal mode) the trace buffer and the trace buffer control register can not be accessed.

6.5 DSU memory map

The DSU memory map can be seen in table 25 below. In a multiprocessor systems, the register map is duplicated and address bits 27 - 24 are used to index the processor.

Table 25. DSU memory map

Address offset	Register
0x000000	DSU control register
0x000008	Time tag counter
0x000020	Break and Single Step register
0x000024	Debug Mode Mask register
0x000040	AHB trace buffer control register
0x000044	AHB trace buffer index register
0x000050	AHB breakpoint address 1
0x000054	AHB mask register 1
0x000058	AHB breakpoint address 2
0x00005c	AHB mask register 2
0x100000 - 0x110000	Instruction trace buffer (..0: Trace bits 127 - 96, ..4: Trace bits 95 - 64, ..8: Trace bits 63 - 32, ..C : Trace bits 31 - 0)
0x110000	Instruction Trace buffer control register
0x200000 - 0x210000	AHB trace buffer (..0: Trace bits 127 - 96, ..4: Trace bits 95 - 64, ..8: Trace bits 63 - 32, ..C : Trace bits 31 - 0)
0x300000 - 0x3007FC 0x300800 - 0x300FFC	IU register file IU register file check bits
0x301000 - 0x30107C 0x301800 - 0x30187C	FPU register file FPU register file check bits
0x400000 - 0x4FFFFC	IU special purpose registers
0x400000	Y register
0x400004	PSR register
0x400008	WIM register
0x40000C	TBR register
0x400010	PC register
0x400014	NPC register
0x400018	FSR register
0x40001C	CPSR register
0x400020	DSU trap register
0x400024	DSU ASI register
0x400040 - 0x40007C	ASR16 - ASR31 (when implemented)
0x700000 - 0x7FFFFC	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 : Instruction cache data

The addresses of the IU registers depends on how many register windows has been implemented:

- %o*n* : 0x300000 + (((psr.cwp * 64) + 32 + *n**4) mod (NWINDOWS*64))
- %l*n* : 0x300000 + (((psr.cwp * 64) + 64 + *n**4) mod (NWINDOWS*64))
- %i*n* : 0x300000 + (((psr.cwp * 64) + 96 + *n**4) mod (NWINDOWS*64))
- %g*n* : 0x300000 + (NWINDOWS*64) + *n**4
- %f*n* : 0x301000 + *n**4

6.6 DSU registers

6.6.1 DSU control register

The DSU is controlled by the DSU control register:

.

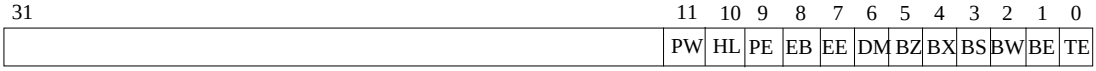


Figure 14. DSU control register

[0]: Trace enable (TE). Enables instruction tracing. If set the instructions will be stored in the trace buffer. Remains set when then processor enters debug or error mode.

[1]: Break on error (BE) - if set, will force the processor to debug mode when the processor would have entered error condition (trap in trap).

[2]: Break on IU watchpoint (BW)- if set, debug mode will be forced on a IU watchpoint (trap 0xb).

[3]: Break on S/W breakpoint (BS) - if set, debug mode will be forced when an breakpoint instruction (ta 1) is executed.

[4]: Break on trap (BX) - if set, will force the processor into debug mode when any trap occurs.

[5]: Break on error traps (BZ) - if set, will force the processor into debug mode on all *except* the following traps: priviledged_instruction, fpu_disabled, window_overflow, window_underflow, asynchronous_interrupt, ticc_trap.

[6]: Debug mode (DM). Indicates when the processor has entered debug mode (read-only).

[7]: EE - value of the external DSUEN signal (read-only)

[8]: EB - value of the external DSUBRE signal (read-only)

[9]: Processor error mode (PE) - returns ‘1’ on read when processor is in error mode, else ‘0’. If written with ‘1’, it will clear the error and halt mode.

[10]: Processor halt (HL). Returns ‘1’ on read when processor is halted. If the processor is in debug mode, setting this bit will put the processor in halt mode.

[11]: Power down (PW). Returns ‘1’ when processor in power-down mode.

6.6.2 DSU Break and Single Step register

This register is used to break or single step the processor(s). This register controls all processors in a multi-processor system, and is only accessible in the DSU memory map of processor 0.

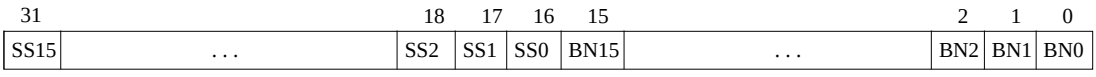


Figure 15. DSU Break and Single Step register

[15:0]: Break now (BNx) -Force processor x into debug mode if the Break on S/W breakpoint (BS) bit in the processors DSU control register is set. If cleared, the processor x will resume execution.

[31:16] : Single step (SSx) - if set, the processor x will execute one instruction and return to debug mode. The bit remains set after the processor goes into the debug mode.

6.6.3 DSU Debug Mode Mask Register

When one of the processors in a multiprocessor LEON3 system enters the debug mode the value of the DSU Debug Mode Mask register determines if the other processors are forced in the debug mode. This register controls all processors in a multi-processor system, and is only accessible in the DSU memory map of processor 0.

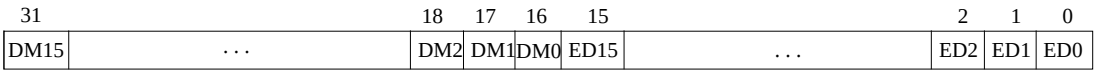


Figure 16. DSU Debug Mode Mask register

[15:0] : Enter debug mode (EDx) - Force processor x into debug mode if any of processors in a multiprocessor system enters the debug mode. If 0, the processor x will not enter the debug mode.

[31:16]: Debug mode mask. If set, the corresponding processor will not be able to force running processors into debug mode even if it enters debug mode.

6.6.4 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).

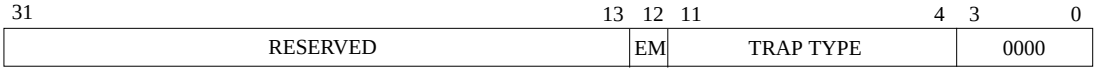


Figure 17. DSU trap register

[11:4]: 8-bit SPARC trap type

[12]: Error mode (EM). Set if the trap would have cause the processor to enter error mode.

6.6.5 Trace buffer time tag counter

The trace buffer time tag counter is incremented 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.

Figure 18. Trace buffer time tag counter

The value is used as time tag in the instruction and AHB trace buffer.

The width of the timer (up to 30 bits) is configurable through the DSU generic port.

6.6.6 DSU ASI register

The DSU can perform 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.

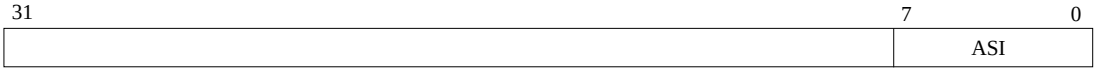


Figure 19. ASI diagnostic access register

[7:0]: ASI to be used on diagnostic ASI access

6.6.7 AHB Trace buffer control register

The AHB trace buffer is controlled by the AHB trace buffer control register:

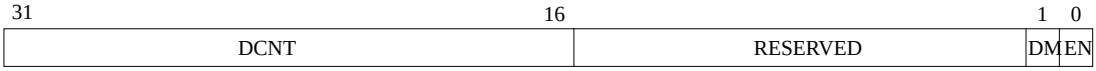


Figure 20. AHB trace buffer control register

[0]: Trace enable (EN). Enables the trace buffer.
[1]: Delay counter mode (DM). Indicates that the trace buffer is in delay counter mode.
[31:16] Trace buffer delay counter (DCNT). Note that the number of bits actually implemented depends on the size of the trace buffer.

6.6.8 AHB trace buffer index register

The AHB trace buffer index register contains the address of the next trace line to be written.

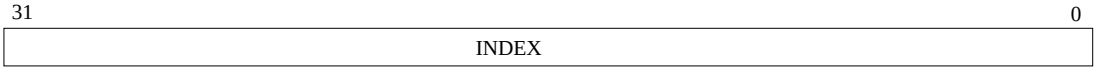


Figure 21. AHB trace buffer index register

27:4 Trace buffer index counter (INDEX). Note that the number of bits actually implemented depends on the size of the trace buffer. (During write)
31:4 Trace buffer index counter (INDEX). Note that the number of bits actually implemented depends on the size of the trace buffer. (During read)

6.6.9 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 will be decremented 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.

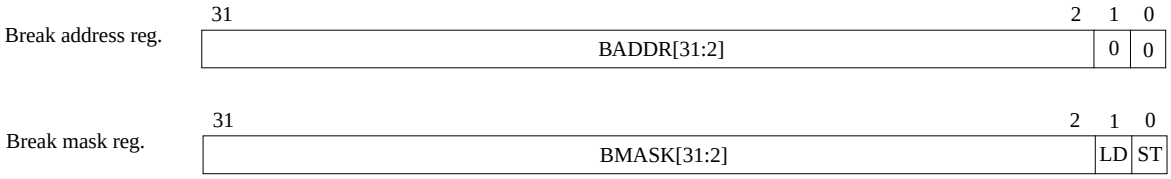


Figure 22. Trace buffer breakpoint registers

[31:2]: Breakpoint address (bits 31:2)
[31:2]: Breakpoint mask (see text)
[1]: LD - break on data load address
[0]: ST - beak on data store address

6.6.10 Instruction trace control register

The instruction trace control register contains a pointer that indicates the next line of the instruction trace buffer to be written.

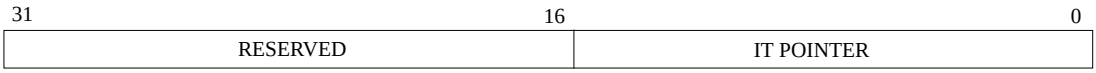


Figure 23. Instruction trace control register

[15:0] Instruction trace pointer. Note that the number of bits actually implemented depends on the size of the trace buffer.

6.7 Signal definitions and reset values

The signals and their reset values are described in table 26.

Table 26. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
dsuen	Input	DSU enable	High	-
dsubre	Input	DSU break	High	-
dsuact	Output	Debug mode	High	Logical 0

6.8 Timing

The timing waveforms and timing parameters are shown in figure 24 and are defined in table 27.

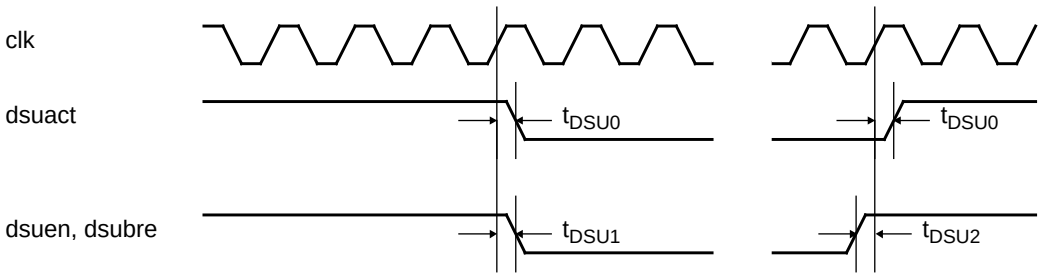


Figure 24. Timing waveforms

Table 27. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
t_{DSU0}	clock to output delay	rising *clk* edge	0	25	ns
t_{DSU1}	input to clock hold	rising *clk* edge	-	-	ns
t_{DSU2}	input to clock setup	rising *clk* edge	-	-	ns

Note: The *dsubre* and *dsuen* are re-synchronized internally. These signals do not have to meet any setup or hold requirements.

7 Fault Tolerant PROM/SRAM/IO Memory Interface

7.1 Overview

The fault tolerant 32-bit PROM/SRAM memory interface uses a common 32-bit memory bus to interface PROM, SRAM and I/O devices. Support for 8-bit PROM banks can also be separately enabled. In addition it also provides an Error Detection And Correction Unit (EDAC), correcting one and detecting two errors. Configuration of the memory controller functions is performed through the APB bus interface.

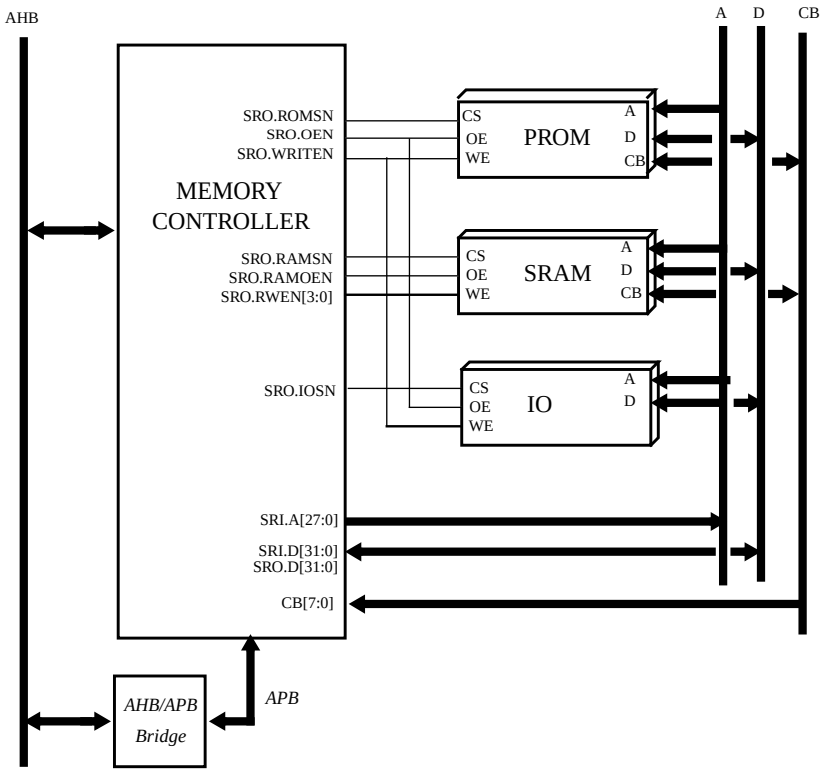


Figure 25. 32-bit FT PROM/SRAM/IO controller

7.2 Operation

The controller is configured through to decode three address ranges: PROM, SRAM and I/O area. By default the PROM area is mapped into address range 0x0 - 0x00FFFFFF, the SRAM area is mapped into address range 0x40000000 - 0x40FFFFFF, and the I/O area is mapped to 0x20000000 - 0x20FFFFFF.

One chip select is decoded for the I/O area, while SRAM has 4 and PROM 4 chip select signals. The controller generates both a common write-enable signal (WRITEN) as well as four byte-write enable signals (WREN). If the SRAM uses a common write enable signal the controller can be configured to perform read-modify-write cycles for byte and half-word write accesses. Number of waitstates is separately configurable for the three address ranges.

The configuration of the EDAC is done through a configuration register accessed from the APB bus. During nominal operation, the EDAC checksum is generated and checked automatically. Single errors are corrected without generating any indication of this condition in the bus response. If a multiple error is detected, a two cycle error response is given on the AHB bus.

When EDAC is enabled, one extra latency cycle is generated during reads and subword writes.

The EDAC function can be enabled for SRAM and PROM area accesses, but not for I/O area accesses. For the SRAM area, the EDAC functionality is only supported for accessing 32-bit wide SRAM banks. For the PROM area, the EDAC functionality is supported for accessing 32-bit wide PROM banks, as well as for read accesses to 8-bit wide PROM banks.

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
CB2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
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
```

7.2.1 8-bit PROM access

The FTSRCTRL controller can be configured to access an 8-bit wide PROM. The data bus of the external PROM should be connected to the upper byte of the 32-bit data bus, i.e. D[31:24]. When enabled, read accesses to the PROM area will be done in four-byte bursts for all 32-, 16- and 8-bit AMBA AHB accesses. The whole 32-bit word is then output on the AHB data bus, allowing the master to chose the bytes needed (big-endian).

Writes should be done one byte at a time. For correct word aligned 32-bit word write accesses, the byte should always be driven on bits 31 to 24 on the AHB data bus. For non-aligned 32-bit word write accesses, the byte should be driven on the bits of the AHB data bus that correspond to the byte address (big-endian). For correct half-word aligned 16-bit half-word write accesses, the byte should always be driven on bits 31 to 24, or 15 to 8, on the AHB data bus. For non-aligned 16-bit half-word write accesses, the byte should be driven on the bits of the AHB data bus that correspond to the byte address (big-endian). For 8-bit word write accesses the byte should always be driven on the AHB data bus bits that corresponds to the byte address (big-endian). To summarize, all legal AMBA AHB write accesses are supported according to the AMBA standard, additional illegal accesses are supported as described above, and it is always the addressed byte that is output.

It is possible to dynamically switch between 8- and 32-bit PROM mode by writing to the RBW field of the MCFG1 register. The *gpio[1:0]* input signals determines the reset value of this RBW register field. When RBW is “00” then 8-bit mode is selected. If RBW is “10” then 32-bit mode is selected. Other RBW values are reserved for future use. SRAM access is not affected by the 8-bit PROM mode.

It is also possible to use the EDAC in the 8-bit PROM mode, enabled via the MCFG3 register. Read accesses to the 8-bit PROM area will be done in five-byte bursts for all 32-, 16- and 8-bit AMBA AHB accesses. After a potential correction, the whole 32-bit word is output on the AHB data bus, allowing the master to chose the bytes needed (big-endian). EDAC support is not provided for write accesses, they are instead performed in the same way as without the EDAC enabled. The checksum byte must be written by the user into the correct byte address location.

The fifth byte corresponds to the EDAC checksum and is located in the upper part of the effective memory area, as explained in detail in the definition of the MCFG1 memory configuration register. The EDAC checksums are located in the upper quarter of what is defined as available EDAC area by

means of the EBSZ field and the ROMBSZ field. When set to 0, the size of the available EDAC area is defined as the PROM bank size. When set to 1, as twice the PROM bank size. When set to 2, as four times the PROM bank size. And when set to 3, as eight times the PROM bank size. For any other value than 0, the use of multiple PROM banks is required.

Example, if ROMBSZ=10 and EBSZ=1, the EDAC area is 8 kByte*2^ROMBSZ*2^EBSZ = 16 MByte = 0x01000000. The checksum byte for the first word located at address 0x00000000 to 0x00000003 is located at 0x00C00000. The checksum byte for the second word located at address 0x00000004 to 0x00000007 is located at 0x00C00001, and so on. Since EBSZ=1, two PROM banks are required for implementing the EDAC area, each bank with size 8 MByte = 0x00800000.

7.2.2 Access errors

The active low Bus Exception signal (BEXCN) can be used to signal access errors. It is enabled by setting the BEXCEN bit in MCFG1 and is active for all types of accesses to all areas (PROM, SRAM and I/O). The BEXCN signal is sampled on the same cycle as read data is sampled. For writes it is sampled on the last rising edge before writen/rwen is de-asserted (writen and rwen are clocked on the falling edge). When a bus exception is detected an error response will be generated for the access.

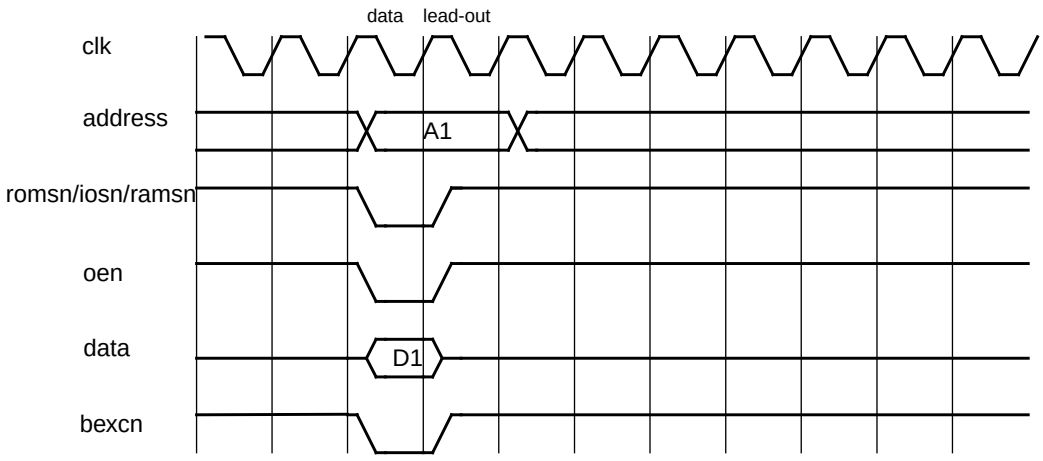


Figure 26. Read cycle with BEXCN.

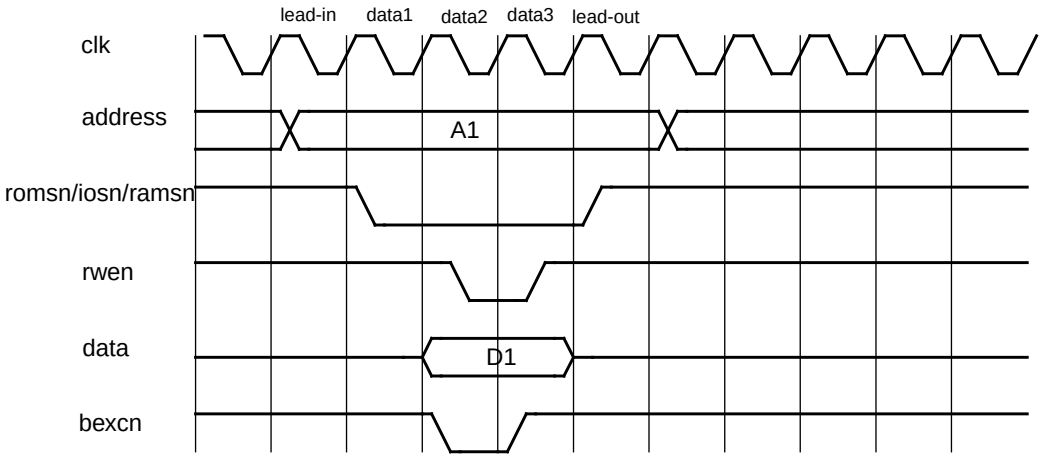


Figure 27. Write cycle with BEXCN.

7.2.3 Using bus ready signalling

The Bus Ready (BRDYN) signal can be used to add waitstates to I/O-area accesses, covering the complete memory area and both read and write accesses. It is enabled by setting the Bus Ready Enable (BRDYEN) bit in the MCFG1 register. An access will have at least the amount of waitstates set, but will be further stretched until BRDYN is asserted. Additional waitstates can thus be inserted after the pre-set number of waitstates by de-asserting the BRDYN signal. BRDYN should be asserted in the cycle preceding the last one. It is recommended that BRDYN remains asserted until the IOSN signal is de-asserted, to ensure that the access has been properly completed and avoiding the system to stall. Read accesses will have the same timing as when EDAC is enabled while write accesses will have the timing as for single accesses even if bursts are performed.

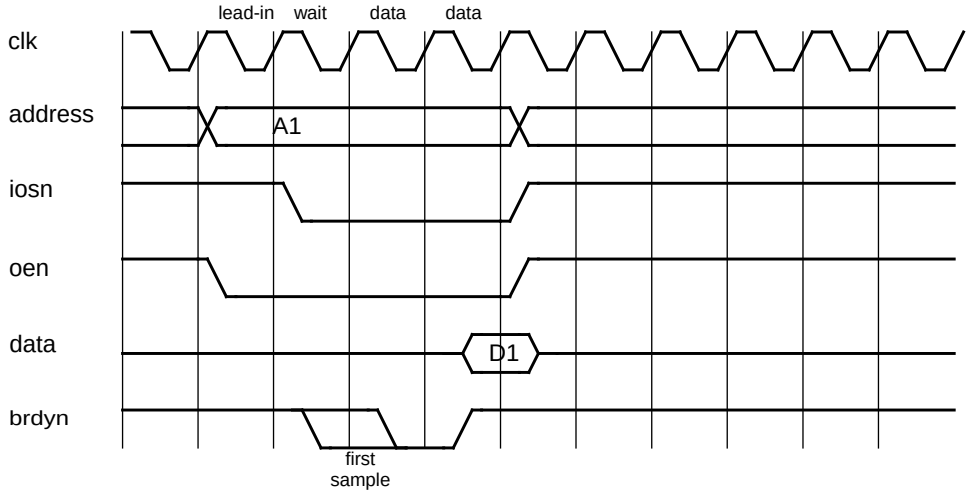


Figure 28. I/O READ cycle, programmed with 1 wait state, and with an extra data cycle added with BRDYN.

7.3 PROM/SRAM/IO waveforms

The internal and external waveforms of the interface are presented in the figures hereafter.

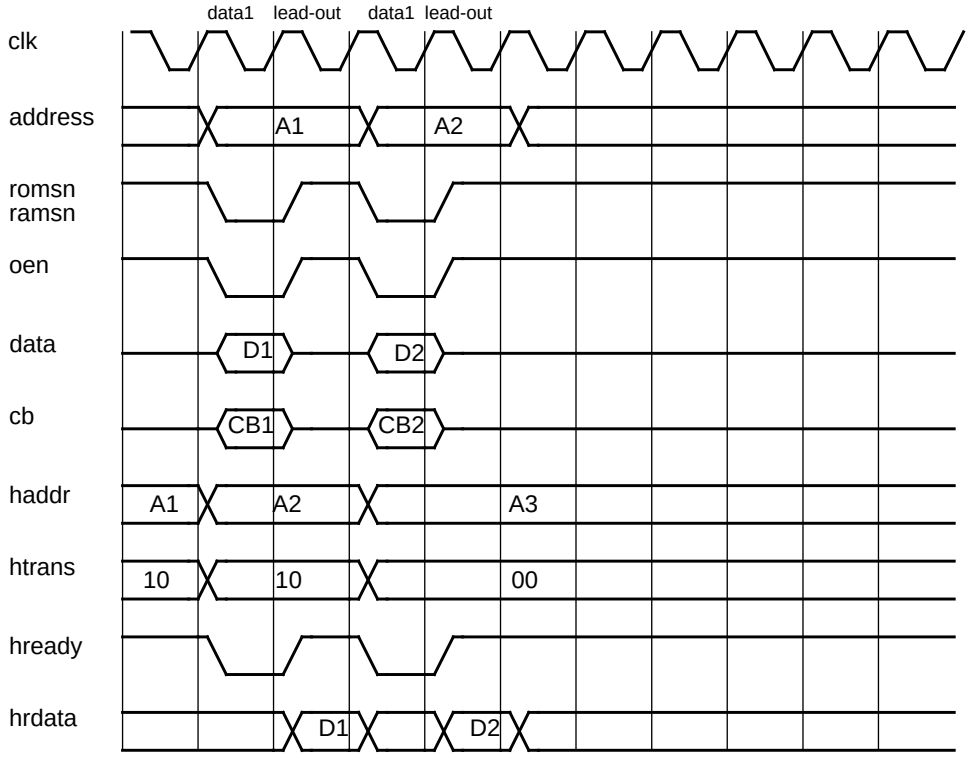


Figure 29. PROM/SRAM non-consecutive read cyclecs.

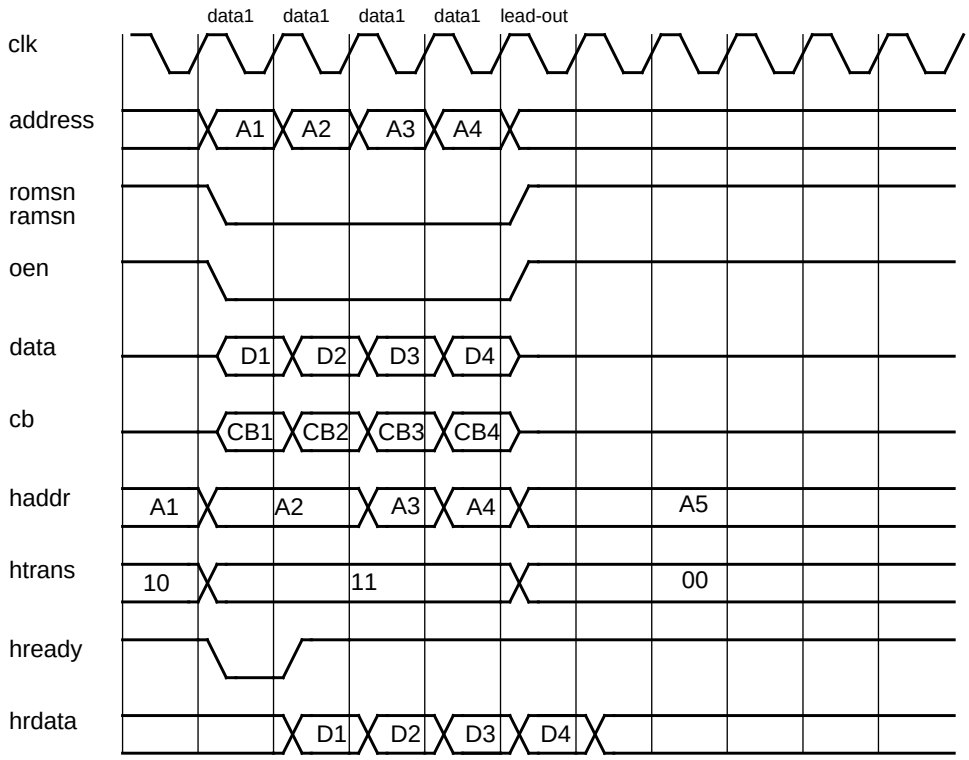


Figure 30. 32-bit PROM/SRAM sequential read access with 0 wait-states and EDAC disabled.

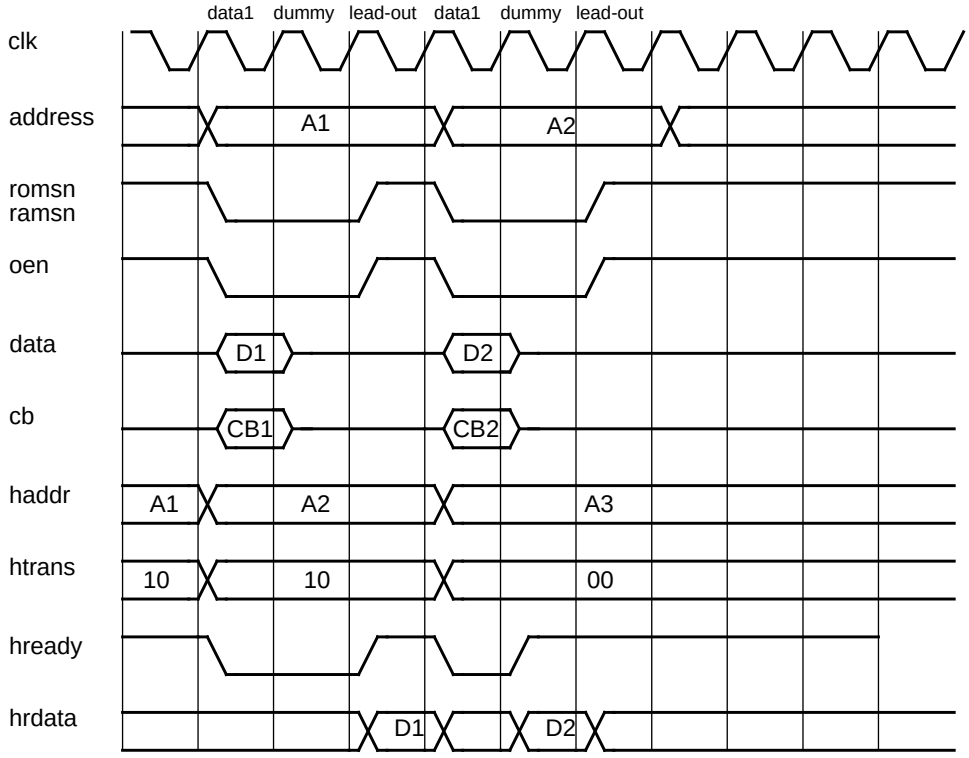


Figure 31. 32-bit PROM/SRAM non-sequential read access with 0 wait-states and EDAC enabled.

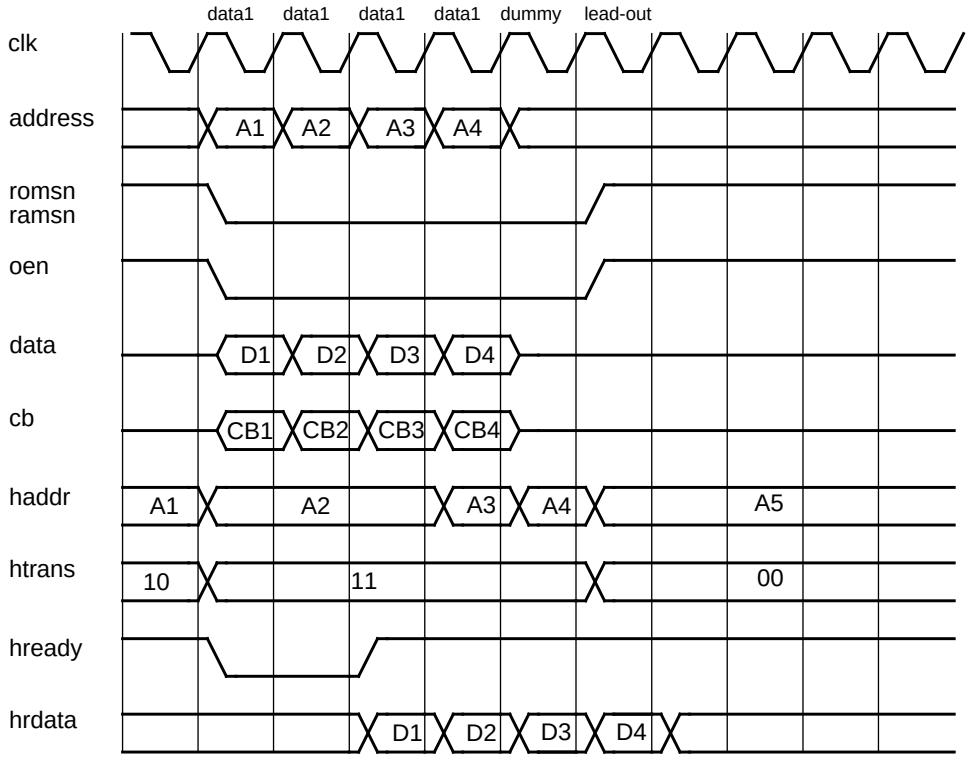


Figure 32. 32-bit PROM/SRAM sequential read access with 0 wait-states and EDAC enabled..

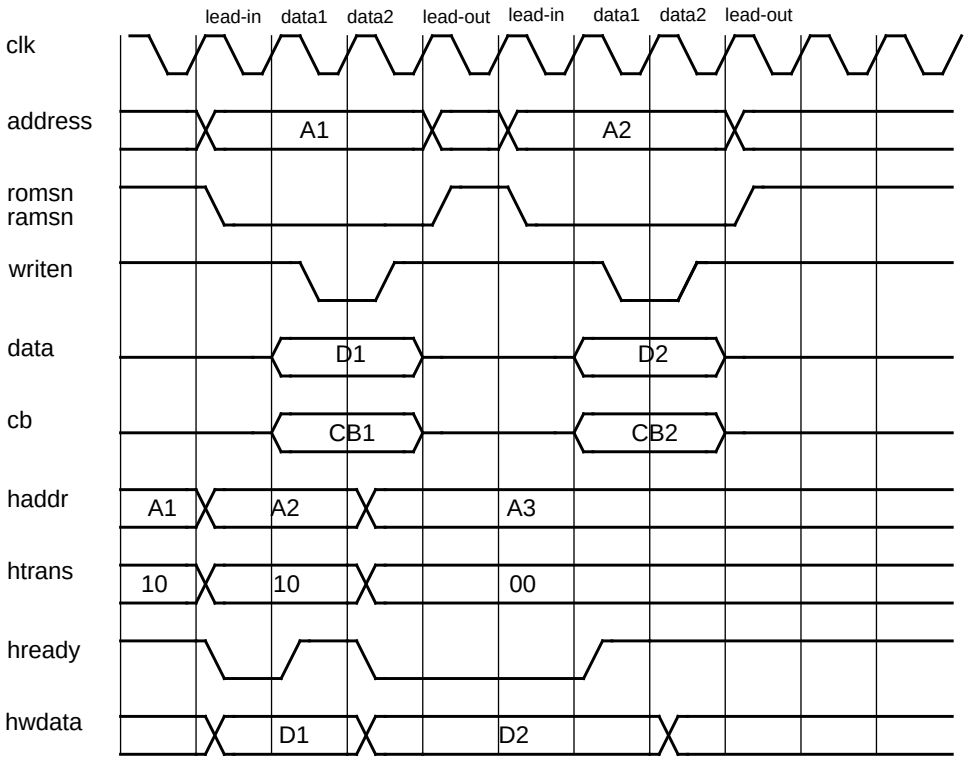


Figure 33. 32-bit PROM/SRAM non-sequential write access with 0 wait-states and EDAC disabled.

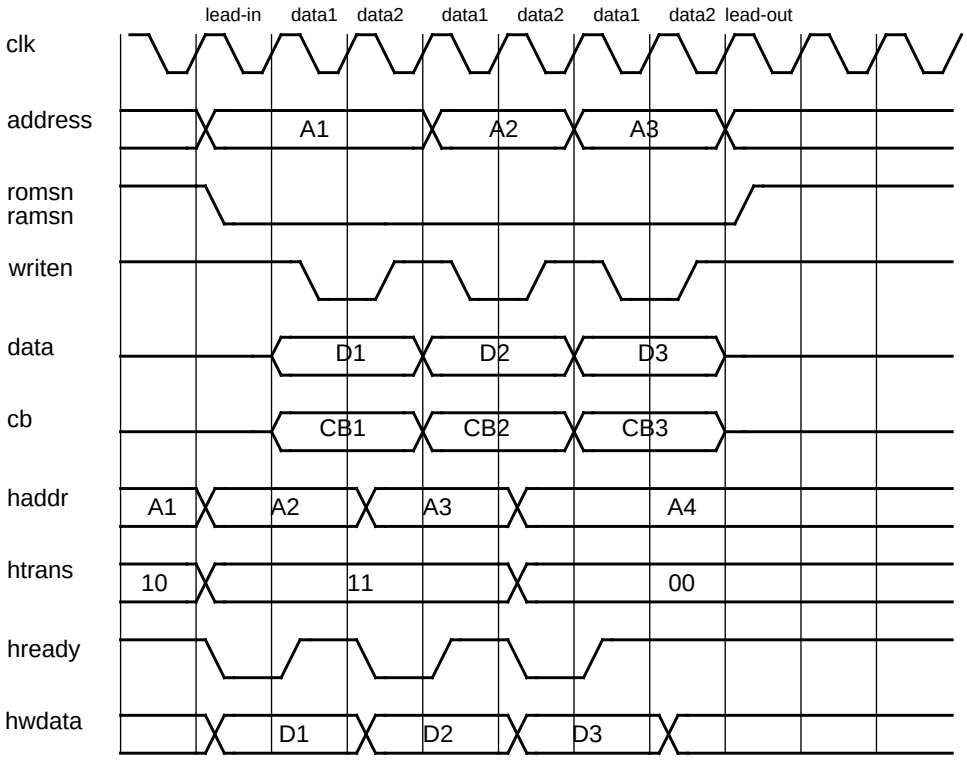


Figure 34. 32-bit PROM/SRAM sequential write access with 0 wait-states and EDAC disabled.

If waitstates are configured, one extra data cycle will be inserted for each waitstate in both read and write cycles. The timing for write accesses is not affected when EDAC is enabled while one extra latency cycle is introduced for single access reads and at the beginning of read bursts.

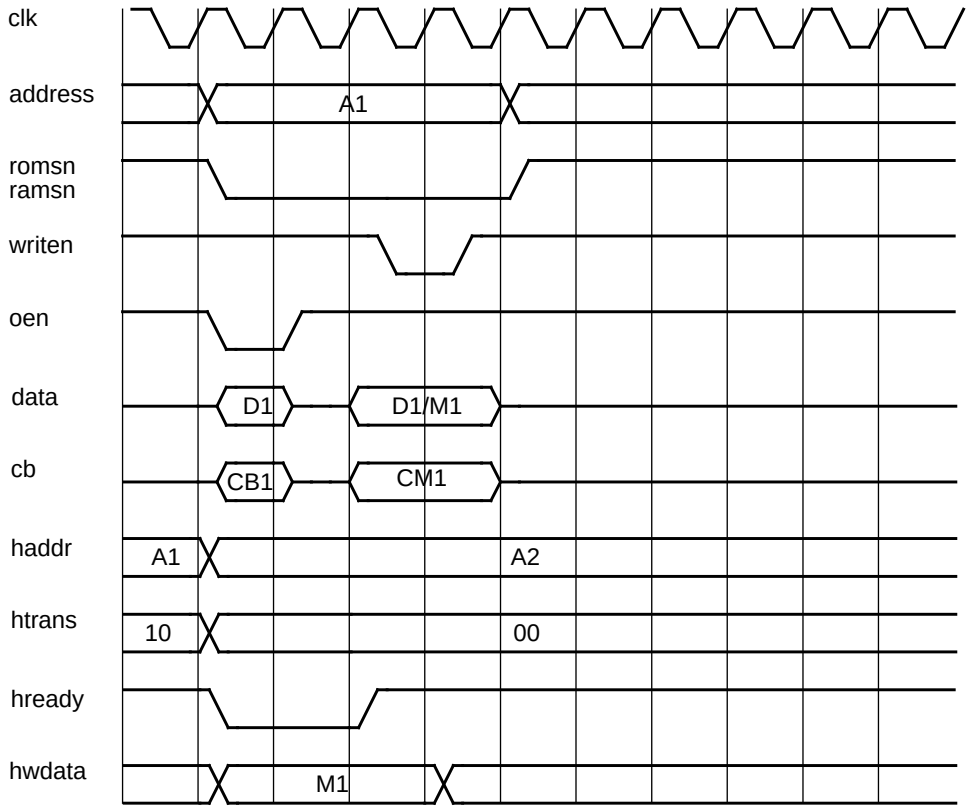


Figure 35. 32-bit PROM/SRAM rmw access with 0 wait-states and EDAC disabled.

Read-Modify-Write (RMW) accesses will have an additional waitstate inserted to accommodate decoding when EDAC is enabled.

I/O accesses are similar to PROM and SRAM accesses but a lead-in and lead-out cycle is always present.

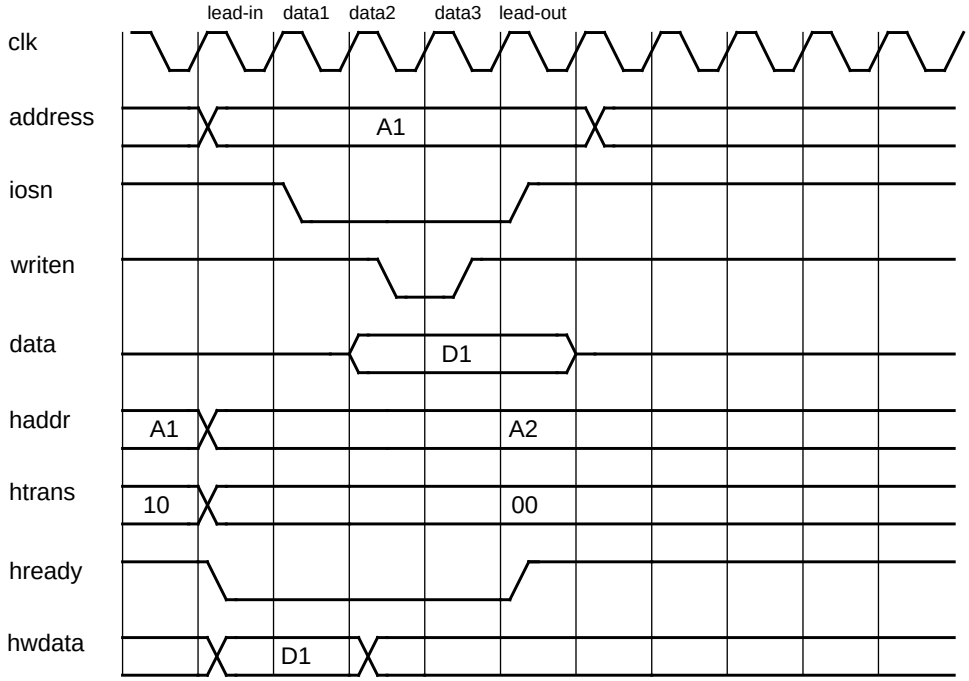


Figure 36. I/O write access with 0 wait-states.

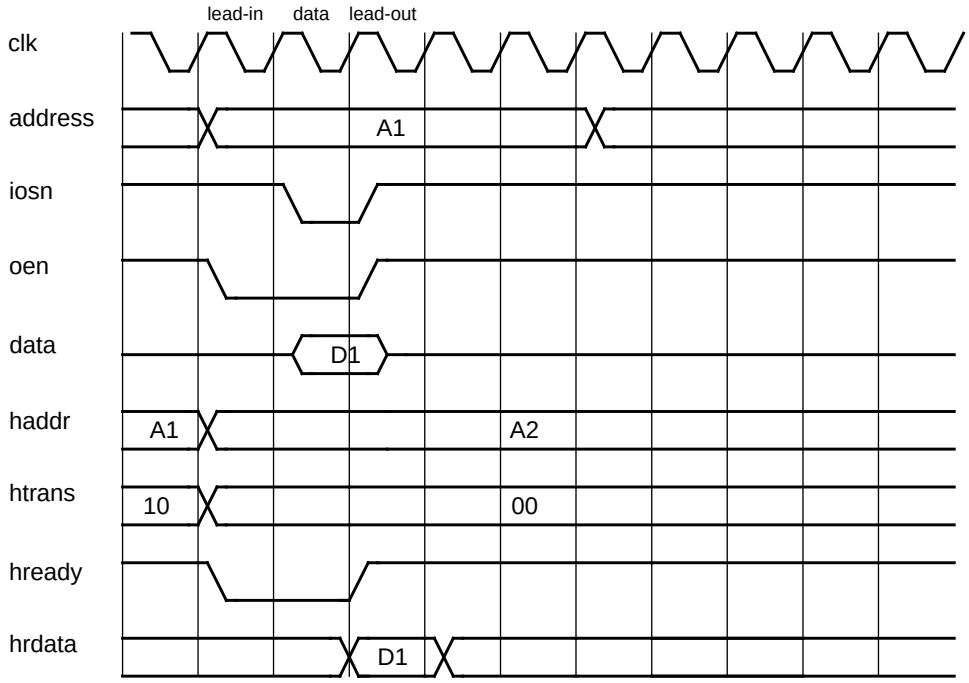


Figure 37. I/O read access with 0 wait-states

7.4 Registers

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

Table 28. FT PROM/SRAM/IO controller registers

APB Address offset	Register
0x0	Memory configuration register 1
0x4	Memory configuration register 2
0x8	Memory configuration register 3

Table 29. Memory configuration register 1.

31 27	26	25	24	23 20	19 18	17 14	13 12	11	10	9 8	7 4	3 0
RESERVED	BR	BE		IOWS		ROMBSZ	EBSZ	RW		RBW	RESERVED	ROMWS

31: 27	RESERVED
26	Bus ready enable (BR) - Enables the bus ready signal (BRDYN) for I/O area.
25	Bus exception enable (BE) - Enables the bus exception signal (BEXCEN) for PROM, SRAM and I/O areas
24	RESERVED
23: 20	I/O wait states (IOWS) - Sets the number of waitstates for accesses to the I/O-area.
19: 18	RESERVED
17: 14	ROM bank size (ROMBSZ) - Sets the PROM bank size. 0=8 kByte, 1=16 kByte, ..., 15=256 MByte
13: 12	EDAC bank size (EBSZ) - Sets the EDAC bank size for 8-bit PROM support. The resulting EDAC bank size is 2^EBSZ * 2^ROMBSZ * 8kByte. Note that only the three lower quarters of the bank can be used for user data. The EDAC checksums are placed in the upper quarter of the bank.
11	ROM write enable (RW) - Enables writes to the PROM memory area. When disabled, writes to the PROM area will generate an ERROR response on the AHB bus.
10	RESERVED
9: 8	ROM data bus width (RBW) - Sets the PROM data bus width. “00” = 8-bit, “10” = 32-bit, others reserved. At reset, these bits are initialized with the value of *gpio[1:0]*.
7: 4	RESERVED
3: 0	ROM waitstates (ROMWS) - Sets the number of waitstates for accesses to the PROM area. Reset to all-ones.

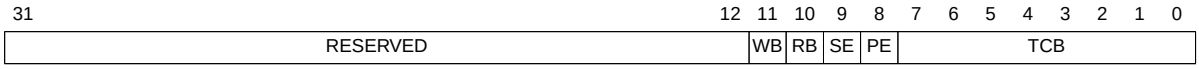
During reset, the ROM width (bits [9:8]) are set with value on *gpio[1:0]* inputs. ROM bank size is set to 8 kByte. External bus error and bus ready are disabled. All other fields are undefined.

Table 30. Memory configuration register 2.

31 13	12 9	8 7	6	5 4 3 2	1 0
RESERVED	RAMBSZ		RW	RESERVED	RAMW

31: 13	RESERVED
12: 9	RAM bank size (RAMBSZ) - Sets the number of waitstates for accesses to the RAM area.
8: 7	RESERVED
6	Read-modify-write enable (RW) - Enables read-modify-write cycles for write accesses.
5: 2	RESERVED
1: 0	RAM waitstates (RAMW) - Sets the number of waitstates for accesses to the RAM area.

Table 31. Memory configuration register 3.



31: 12	RESERVED
11	Write bypass (WB) - Enables EDAC write bypass. When enabled the TCB field will be used as checkbits in all write operations.
10	Read bypass (RB) - Enables EDAC read bypass. When enabled checkbits read from memory in all read operations will be stored in the TCB field.
9	SRAM EDAC enable (SE) - Enables EDAC for the SRAM area.
8	PROM EDAC enable (PE) - Enables EDAC for the PROM area. Reset value is taken from the input signal *gpio[2]*.
7: 0	Test checkbits (TCB) - Used as checkbits in write operations when WB is activated and checkbits from read operations are stored here when RB is activated.

During reset, the prom EDAC usage (bit [8]) is set with the value on *gpio[2]*.

7.5 Signal definitions and reset values

The signals and their reset values are described in table 32.

Table 32. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
address[27:0]	Output	Memory address	High	Undefined
data[31:0]	Input/Output	Memory data	High	Tri-state
cb[7:0]	Input/Output	Check bits	High	Tri-state
ramsn[3:0]	Output	SRAM chip select	Low	Logical 1
ramoen[3:0]	Output	SRAM output enable	Low	Logical 1
rwen[3:0]	Output,	SRAM write byte enable: rwen[0] corresponds to data[31:24], rwen[1] corresponds to data[23:16], rwen[2] corresponds to data[15:8], rwen[3] corresponds to data[7:0]. Any rwen[ ] signal can be used for cb[ ].	Low	Logical 1
ramben[3:0]	Output	SRAM read/write byte enable: ramben[0] corresponds to data[31:24], ramben[1] corresponds to data[23:16], ramben[2] corresponds to data[15:8], ramben[3] corresponds to data[7:0]. Any ramben[ ] signal can be used for cb[ ].	Low	Logical 1
oen	Output	Output enable	Low	Logical 1
writen	Output	Write strobe	Low	Logical 1
read	Output	Read strobe	High	Logical 1
iosn	Output	IO area chip select	Low	Logical 1
romsn[3:0]	Output	PROM chip select	Low	Logical 1
brdyn	Input	Bus ready. Extends accesses to the IO area.	Low	-
bexcn	Input	Bus exception.	Low	-
gpio[1:0]	Input	Configuring PROM data width at reset. 8-bit data width when “00”, and 32-bit when “10”.	-	-
gpio[2]	Input	Enabling EDAC usage for PROM at reset.	High	-

7.6 Timing

The timing waveforms and timing parameters are shown in figure 38 and are defined in table 33.

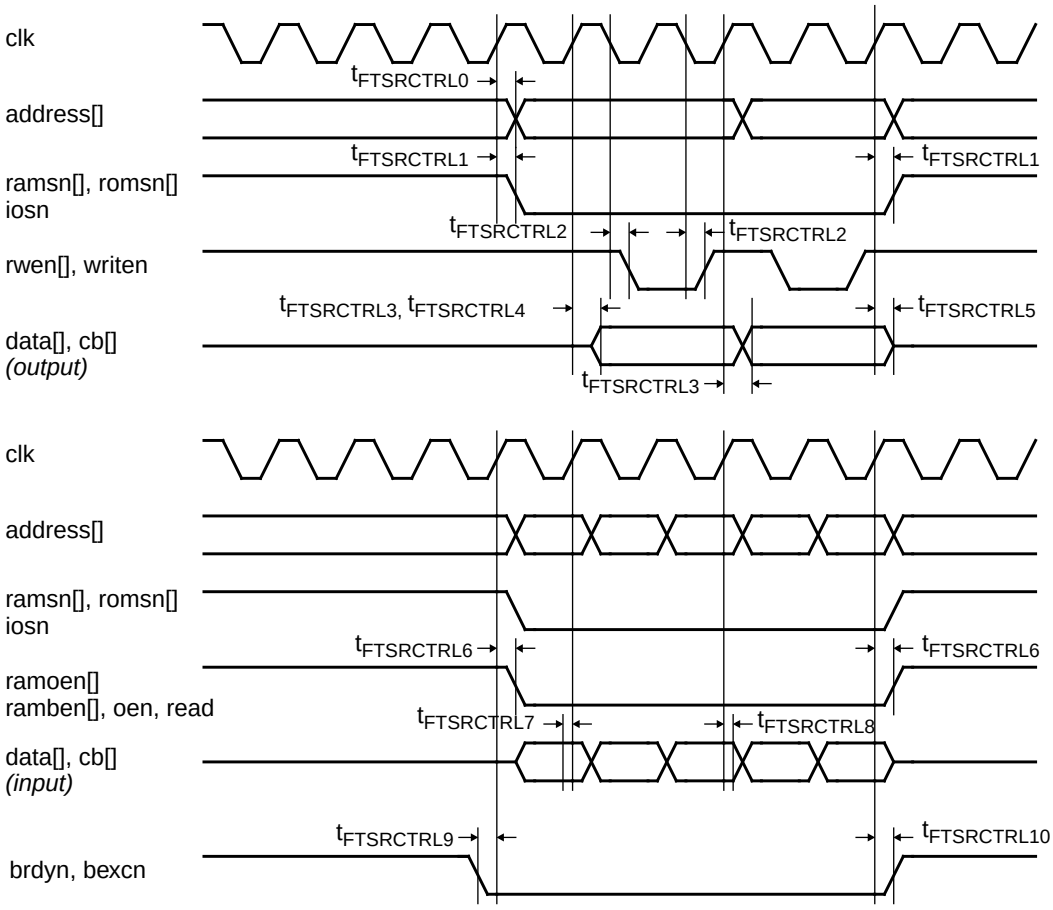


Figure 38. Timing waveforms

Table 33. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{FTSRCTRL0}$	address clock to output delay	rising *clk* edge	0	21.5	ns
$t_{FTSRCTRL1}$	clock to output delay	rising *clk* edge	0	21.5	ns
$t_{FTSRCTRL2}$	clock to output delay	falling *clk* edge	0	21.5	ns
$t_{FTSRCTRL3}$	clock to data output delay	rising *clk* edge	3	21.5	ns
$t_{FTSRCTRL4}$	clock to data non-tri-state delay	rising *clk* edge	0	21.5	ns
$t_{FTSRCTRL5}$	clock to data tri-state delay	rising *clk* edge	3	21.5	ns
$t_{FTSRCTRL6}$	clock to output delay	rising *clk* edge	0	21.5	ns
$t_{FTSRCTRL7}$	data input to clock setup	rising *clk* edge	7	-	ns
$t_{FTSRCTRL8}$	data input from clock hold	rising *clk* edge	1	-	ns
$t_{FTSRCTRL9}$	input to clock setup	rising *clk* edge	7	-	ns
$t_{FTSRCTRL10}$	input from clock hold	rising *clk* edge	1	-	ns

8 Fault Tolerant PROM/SRAM/SDRAM/IO Memory Interface

8.1 Overview

The combined 8/32-bit memory controller provides a bridge between external memory and the AHB bus. The memory controller can handle four types of devices: PROM, asynchronous static ram (SRAM), synchronous dynamic ram (SDRAM) and memory mapped I/O devices (IO). The PROM, SRAM and SDRAM areas can be EDAC-protected using a (39,7) BCH code. The BCH code 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-, or 32-bit mode, depending on application requirements. The controller decodes three address spaces on the AHB bus (PROM, IO, and SRAM/ SDRAM).

External chip-selects are provided for up to four PROM banks, one IO bank, five SRAM banks and two SDRAM banks. Figure 39 below shows how the connection to the different device types is made.

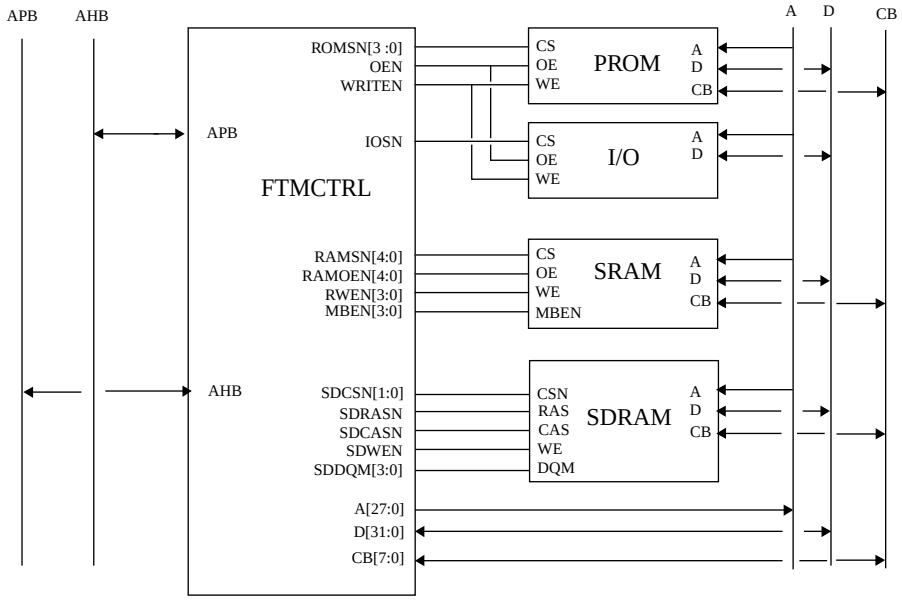


Figure 39. FTMCTRL connected to different types of memory devices

8.2 PROM access

Up to four PROM chip-select signals are provided for the PROM area, ROMSN[3:0]. There are two modes: one with two chip-select signals and one with four. The size of the banks can be set in binary steps from 16 kByte to 256 MByte.

A read access to PROM consists of two data cycles and between 0 and 30 waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. 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 PROM devices. Figure 40 shows the basic read cycle waveform (zero waitstate) for non-consecutive PROM reads. Note that the address is undefined in the lead-out cycle. Figure 41 shows the timing for consecutive cycles (zero waitstate). Waitstates are added by extending the data2 phase. This is shown in figure 42 and applies to both consecutive and non-consecutive cycles. Only an even number of waitstates can be assigned to the PROM area.

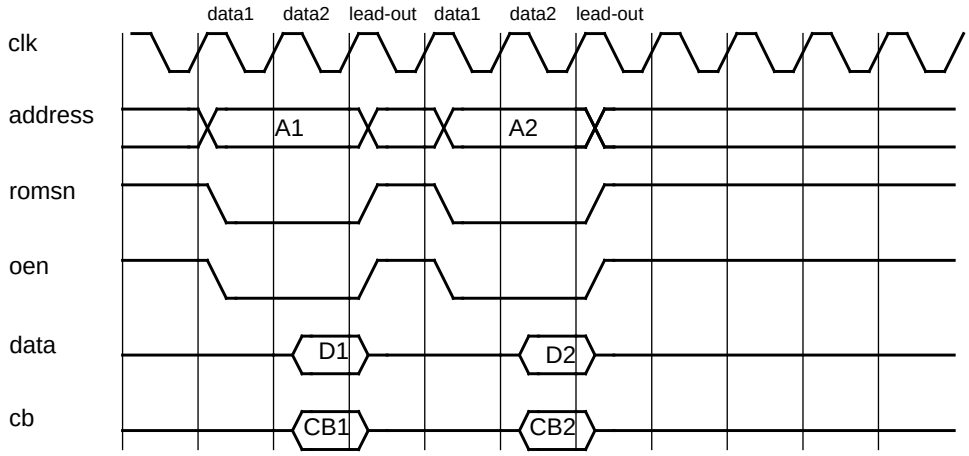


Figure 40. Prom non-consecutive read cyclecs.

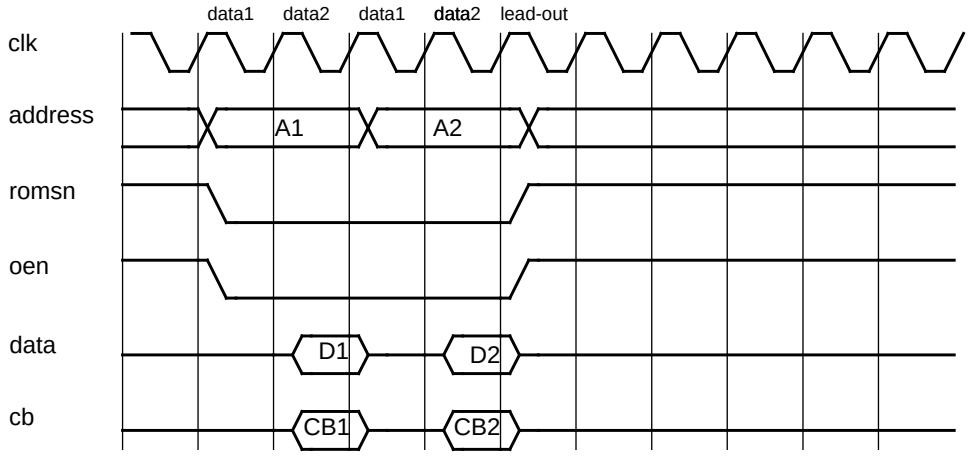


Figure 41. Prom consecutive read cyclecs.

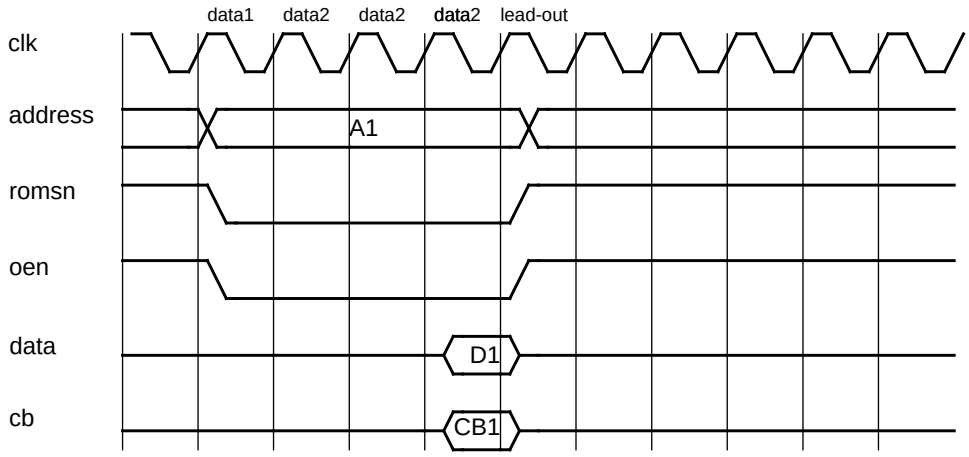


Figure 42. Prom read access with two waitstates.

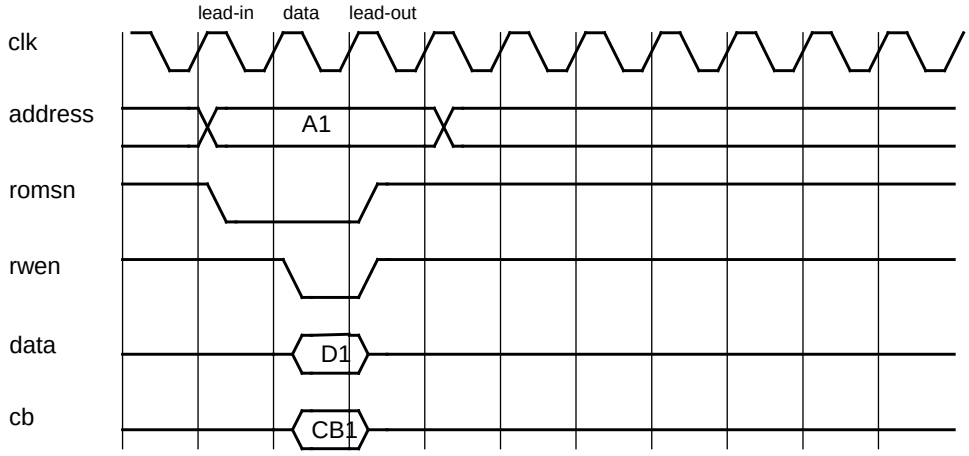


Figure 43. Prom write cycle (0-waitstates)

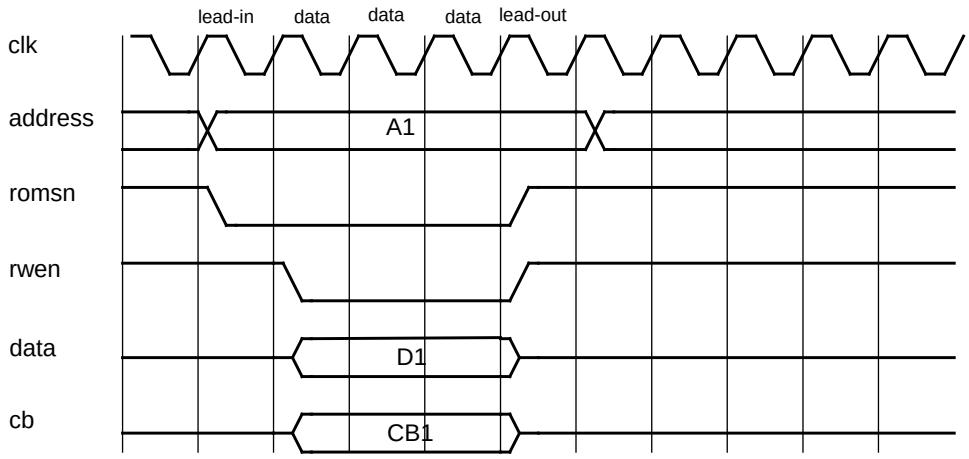


Figure 44. Prom write cycle (2-waitstates)

8.3 Memory mapped IO

Accesses to IO have similar timing as PROM accesses. The IO select (IOSN) and output enable (OEN) signals are delayed one clock to provide stable address before IOSN is asserted. All accesses are performed as non-consecutive accesses as shown in figure 45. The data2 phase is extended when waitstates are added.

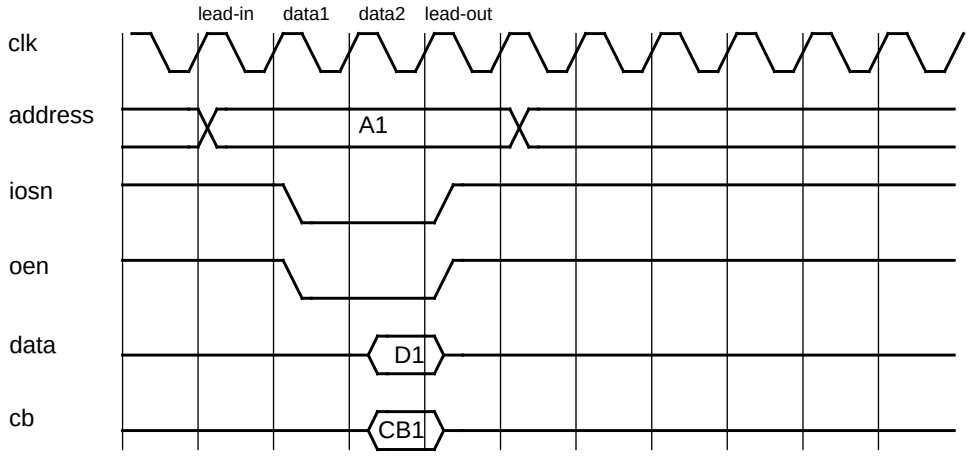


Figure 45. I/O read cycle (0-waitstates)

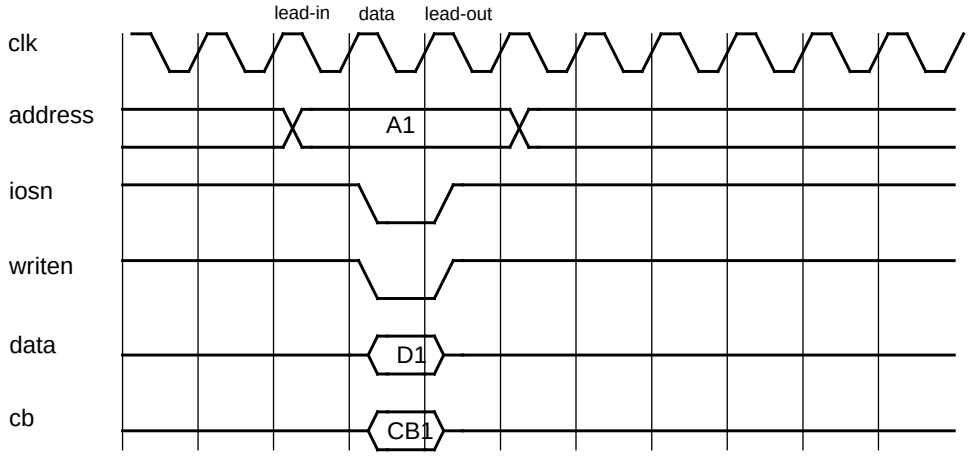


Figure 46. I/O write cycle (0-waitstates)

8.4 SRAM access

The SRAM area is divided on up to five RAM banks. The size of banks 1-4 (RAMSN[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 (RAMSN[4]) decodes the upper 512 MByte (i.e. 0x60000000 to 0x7FFFFFFF, of which the lower 256 MByte can be used with the available address pins) and cannot be used simultaneously with SDRAM memory. A read access to SRAM consists of two data cycles and between zero and three waitstates. The read data (and optional EDAC check-bits) are latched on the rising edge of the clock on the last data cycle. Accesses to RAMSN[4] can further be stretched by de-asserting BRDYN 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. Figure 47

shows the basic read cycle waveform (zero waitstate). Waitstates are added in the same way as for PROM in figure 42.

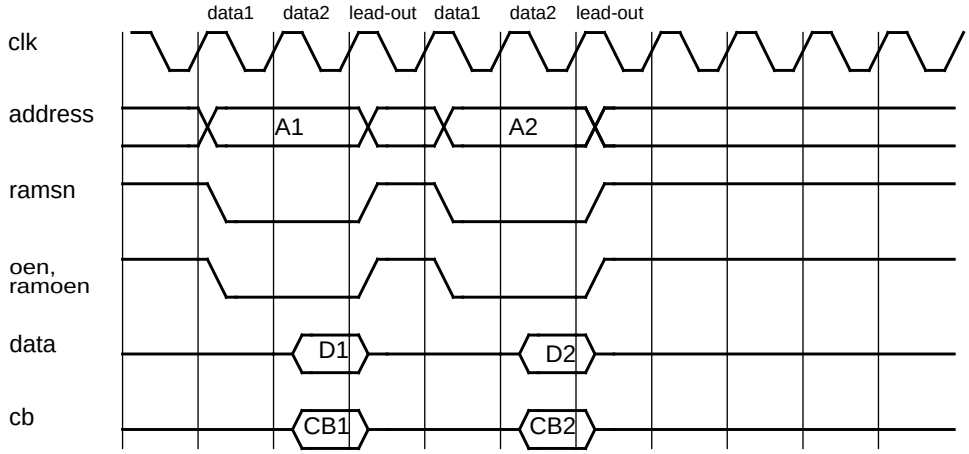


Figure 47. Sram non-consecutive read cyclecs.

For read accesses to RAMSN[4:0], a separate output enable signal (RAMOEN[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. Waitstates are added in the same way as for PROM.

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 32-bit data, the read-modify-write bit MCFG2 should be set to enable read-modify-write cycles for sub-word writes.

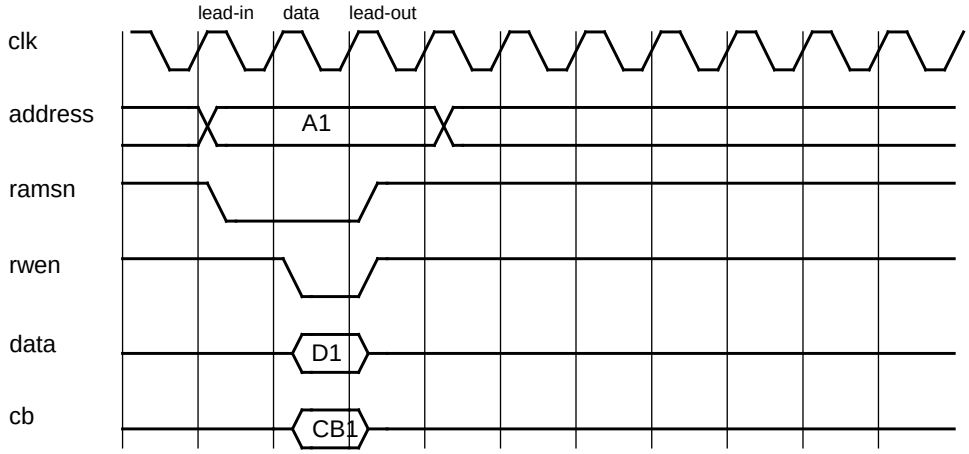


Figure 48. Sram write cycle (0-waitstates)

8.5 8-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 operation by programming the ROM and RAM width fields in the memory configuration registers. Since reads to memory are always done on 32-bit word basis, read access to 8-bit memory will be transformed in a burst of four read cycles. During writes, only the necessary bytes will be written. Figure 49 shows an interface example with 8-bit PROM and 8-bit SRAM. It is not allowed to set the ROM or RAM width fields to 16-bit width.

The RMW bit must not be set if RAM EDAC is not enabled when RAM width is set to 8-bit.

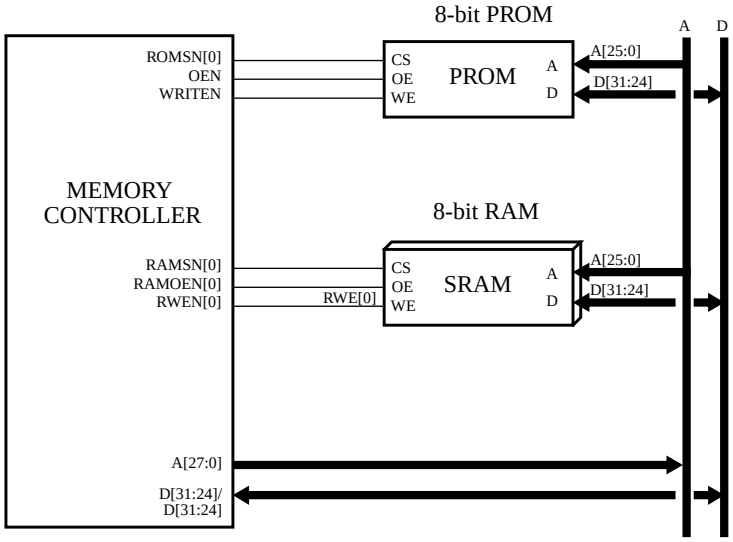


Figure 49. 8-bit memory interface example

In 8-bit mode, the PROM/SRAM devices should be connected to the MSB byte of the data bus (D[31:24]). The LSB address bus should be used for addressing (A[25:0]).

8.6 8- bit I/O access

Similar to the PROM/SRAM areas, the IO area can also be configured to 8-bits mode. However, the I/O device will NOT be accessed by multiple 8 bits accesses as the memory areas, but only with one single access just as in 32-bit mode. To access an IO device on an 8-bit bus, only byte accesses should be used (LDUB/STB instructions for the CPU).

8.7 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 includes 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 occurs after the last transfer. Burst cycles will not be generated to the IO area.

Only word (HSIZE = “010”) bursts of incremental type (HBURST=INCR, INCR4, INCR8 or INCR16) are supported.

8.8 SDRAM access

8.8.1 General

Synchronous dynamic RAM (SDRAM) access is supported to two banks of PC100/PC133 compatible devices. This is implemented by a special version of the SDCTRL SDRAM controller core, which is optionally instantiated as a sub-block. The SDRAM controller supports 64M, 256M and 512M 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 operation of the SDRAM controller is controlled through MCFG2 and MCFG3 (see below).

8.8.2 Address mapping

The two SDRAM chip-select signals are decoded. SDRAM area is mapped into the upper half of the RAM area defined by BAR2 register, and cannot be used simultaneously with fifth SRAM bank (RAMSN[4]). 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.

8.8.3 Initialisation

When the SDRAM controller is enabled, it automatically performs the SDRAM initialisation sequence of PRECHARGE, 8x AUTO-REFRESH and LOAD-MODE-REG on both banks simultaneously. The controller programs the SDRAM to use single location access on write. The controller programs the SDRAM to use line burst of length 8, selectable via the MCFG2 register.

8.8.4 Configurable SDRAM timing parameters

To provide optimum access cycles for different SDRAM devices (and at different frequencies), three SDRAM parameters can be programmed through memory configuration register 2 (MCFG2): TCAS, TRP and TRFCD. The value of these field affects the SDRAM timing as described in table 34.

Table 34. SDRAM programmable minimum timing parameters

SDRAM timing parameter	Minimum timing (clocks)
CAS latency, RAS/CAS delay (t_{CAS}, t_{RCD})	TCAS + 2
Precharge to activate (t_{RP})	TRP + 2
Auto-refresh command period (t_{RFC})	TRFC + 3
Activate to precharge (t_{RAS})	TRFC + 1
Activate to Activate (t_{RC})	TRP + TRFC + 4

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

Table 35. SDRAM example programming

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

8.9 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 15.6 µs (corresponding to 780 or 1560 clocks at 100 MHz). The generated refresh period is calculated as (reload value+1)/sysclk. The refresh function is enabled by setting bit 31 in MCFG2.

8.9.1 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. Line burst of length 8, selectable via the MCFG2 register, will be set. Remaining fields are fixed: single location write, sequential burst. The command field will be cleared 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.

8.9.2 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.

8.9.3 Write cycles

Write cycles are performed similarly to read cycles, with the difference that WRITE commands are issued after activation. A write burst on the AHB bus will generate a burst of write commands without idle cycles in-between.

8.9.4 Address bus

The memory controller is configured at design time to either share the address and data buses with the SRAM, or to use separate address and data buses. This is fixed for each Configuration Identifier (CID), see section 29.2 for details. When the buses are shared, the address bus of the SDRAMs should be connected to A[14:2], the bank address to A[16:15]. The MSB part of A[14:2] can be left unconnected if not used. When separate buses are used, the SDRAM address bus should be connected to SA[12:0] and the bank address to SA[14:13].

8.9.5 Initialisation

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

8.10 Memory EDAC

8.10.1 BCH EDAC

The FTMCTRL is provided with an BCH EDAC that can correct one error and detect two 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 adding 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
CB2 = D0 ^ D3 ^ D4 ^ D7 ^ D9 ^ D10 ^ D13 ^ D15 ^ D16 ^ D19 ^ D20 ^ D23 ^ D25 ^ D26 ^ D29 ^ D31
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 SRAM 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. Data is always accessed as words (4 bytes at a time) and the corresponding checkbits are located at the address acquired by inverting the word address (bits 2 to 27) and using it as a byte address. The same chip-select is kept active. A word written as four bytes to addresses 0, 1, 2, 3 will have its checkbits at address 0xFFFFFFF, addresses 4, 5, 6, 7 at 0xFFFFFFE and so on. All the bits up to the maximum bank size will be inverted while the same chip-select is always asserted. This way all the bank sizes can be supported and no memory will be unused (except for a maximum of 4 byte in the gap between the data and checkbit area). A read access will automatically read the four data bytes individually from the nominal addresses and the EDAC checkbit byte from the top part of the bank. A write cycle is performed the same way. Byte or half-word write accesses will result in an automatic read-modify-write access where 4 data bytes and the checkbit byte are firstly read, and then 4 data bytes and the newly calculated checkbit byte are writen back to the memory. This 8-bit mode applies to SRAM while SDRAM always uses 32-bit accesses. The size of the memory bank is determined from the settings in MCFG2.

If the ROM is configured in 8-bit mode, EDAC protection is provided in a similar way as for the SRAM memory described above. The difference is that write accesses are not being handled automatically. Instead, write accesses must only be performed as individual byte accesses by the software, writing one byte at a time, and the corresponding checkbit byte must be calculated and be written to the correct location by the software.

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 will replace 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: when the EDAC is enabled, the RMW bit in memory configuration register 2 must be set. EDAC is not supported for 64-bit wide SDRAM data busses.

NOTE: when the EDAC is enabled in 8-bit bus mode, only the first bank select (RAMSN[0], PROMSN[0]) can be used.

8.10.2 Reed-Solomon EDAC

The Reed-Solomon EDAC provides block error correction, and is capable of correcting up to two 4-bit nibble errors in a 32-bit data word or 16-bit checksum. The Reed-Solomon EDAC can be enabled for the SDRAM area only, and uses a 16-bit checksum. The Reed-Solomon EDAC is enabled by setting the RSE and RE bits in MCFG3, and the RMW bit in MCFG2. The Reed-Solomon EDAC is not supported for 64-bit wide SDRAM buses.

The Reed-Solomon data symbols are 4-bit wide, represented as GF(2^4). The basic Reed-Solomon code is a shortened RS(15, 13, 2) code, represented as RS(6, 4, 2). It has the capability to detect and correct a single symbol error anywhere in the codeword. The EDAC implements an interleaved RS(6, 4, 2) code where the overall data is represented as 32 bits and the overall checksum is represented as 16 bits. The codewords are interleaved nibble-wise. The interleaved code can correct two 4-bit errors when each error is located in a nibble and not in the same original RS(6, 4, 2) codeword.

The Reed-Solomon RS(15, 13, 2) code has the following definition:

- there are 4 bits per symbol;
- there are 15 symbols per codeword;
- the code is systematic;
- the code can correct one symbol error per codeword;
- the field polynomial is

$$f(x) = x^4 + x + 1$$

- the code generator polynomial is

$$g(x) = \prod_{i=0}^{1} (x + \alpha^i) = \sum_{j=0}^{2} g_j \cdot x^j$$

  for which the highest power of *x* is stored first;

- a codeword is defined as 15 symbols:

  c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}

  where c_0 to c_{12} represent information symbols and c_{13} to c_{14} represent check symbols.

The shortened and interleaved RS(6, 4, 2) code has the following definition:

- the codeword length is shortened to 4 information symbols and 2 check symbols and as follows:

  $c_0 = c_1 = c_2 = c_3 = c_4 = c_5 = c_6 = c_7 = c_8 = 0$

  where the above information symbols are suppressed or virtually filled with zeros;

- two codewords are interleaved (i.e. interleaved depth *I*=2) with the following mapping to the 32-bit data and 16-bit checksum, were $c_{i,j}$ is a symbol with codeword index *i* and symbol index *j*:

  $c_{0,9}$ = sd[31:28]

  $c_{1,9}$ = sd[27:24]

  $c_{0,10}$ = sd[23:20]

  $c_{1,10}$ = sd[19:16]

  $c_{0,11}$ = sd[15:12]

$c_{1,11}$ = sd[11:8]

$c_{0,12}$ = sd[7:4]

$c_{1,12}$ = sd[3:0]

$c_{0,13}$ = scb[15:12]

$c_{1,13}$ = scb[11:8]

$c_{0,14}$ = scb[7:4]

$c_{1,14}$ = scb[3:0]

where SD[ ] is interchanable with DATA[] and SCB[ ] is interchangable with CB[ ]

8.10.3 EDAC Error reporting

As mentioned above an un-correctable error results in an AHB error response which can be monitored on the bus. Correctable errors however are handled transparently and are not visible on the AHB bus. A sideband signal is provided which is asserted during one clock cycle for each access for which a correctable error is detected. This can be used for providing an external scrubbing mechanism and/or statistics.

8.11 Bus Ready signalling

The BRDYN signal can be used to stretch all types of access cycles to the PROM, I/O area and the SRAM area decoded by RAMSN[4]. This covers read and write accesses in general, and additionally read-modify-write accesses to the SRAM area. The accesses will always have at least the pre-programmed number of waitstates as defined in memory configuration registers 1 & 2, but will be further stretched until BRDYN is asserted. BRDYN should be asserted in the cycle preceding the last one. If bit 29 in MCFG1 is set, BRDYN can be asserted asynchronously with the system clock. In this case, the read data must be kept stable until the de-assertion of OEN/RAMOEN and BRDYN must be asserted for at least 1.5 clock cycle. The use of BRDYN can be enabled separately for the PROM, I/O and RAMSN[4] areas. It is recommended that BRDYN is asserted until the corresponding chip select signal is de-asserted, to ensure that the access has been properly completed and avoiding the system to stall.

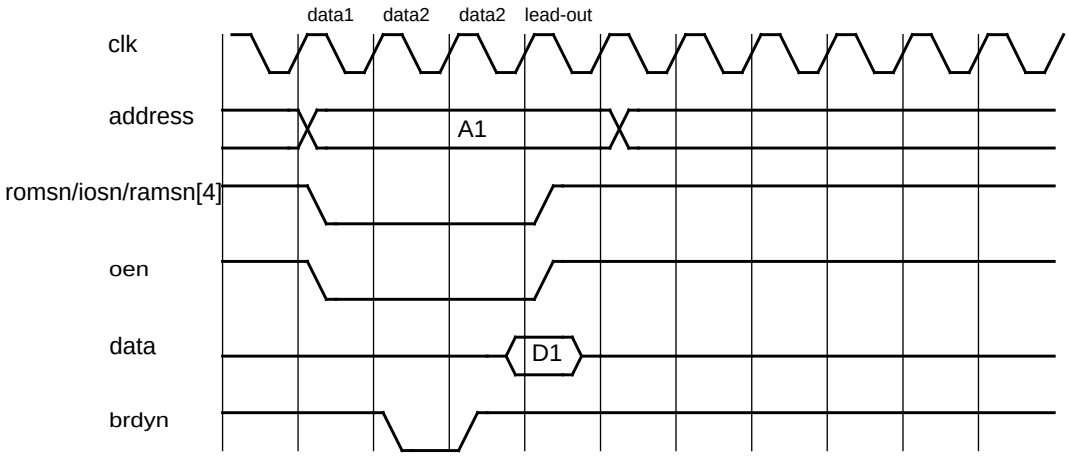


Figure 50. READ cycle with one extra data2 cycle added with BRDYN (synchronous sampling). Lead-out cycle is only applicable for I/O accesses.

Figure 51 shows the use of BRDYN with asynchronous sampling. BRDYN is kept asserted for more than 1.5 clock-cycle. Two synchronization registers are used so it will take at least one additional cycle from when BRDYN is first asserted until it is visible internally. In figure 51 one cycle is added to the data2 phase.

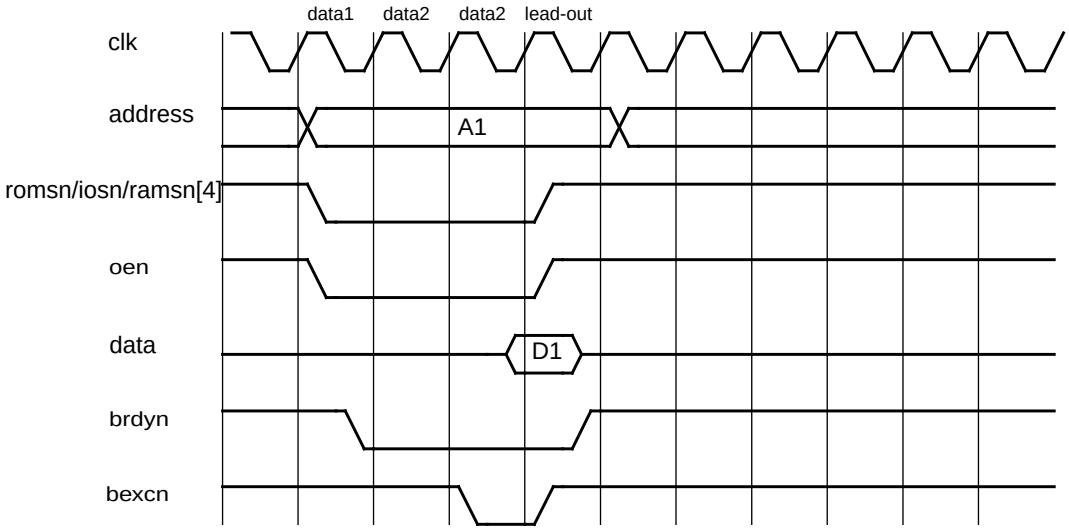


Figure 51. BRDYN (asynchronous) sampling and BEXCN timing. Lead-out cycle is only applicable for I/O-accesses.

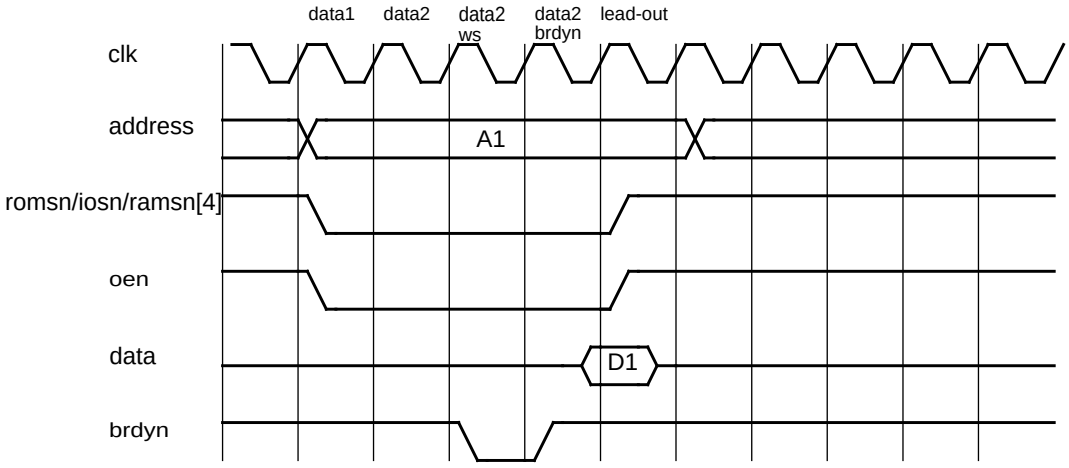


Figure 52. Read cycle with one waitstate (configured) and one BRDYN generated waitstate (synchronous sampling).

8.12 Access errors

An access error can be signalled by asserting the BEXCN signal for read and write accesses. For reads it is sampled together with the read data. 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 signalling. If the usage of BEXCN is enabled in memory configuration register 1, an error response will be generated on the internal AHB bus. BEXCN can be enabled or disabled through memory configuration register 1, and is active for all areas (PROM, IO and RAM). BEXCN is only sampled in the last access for 8-bit

mode for RAM and PROM. That is, when four bytes are written for a word access to 8-bit wide memory BEXCN is only sampled in the last access with the same timing as a single access in 32-bit mode.

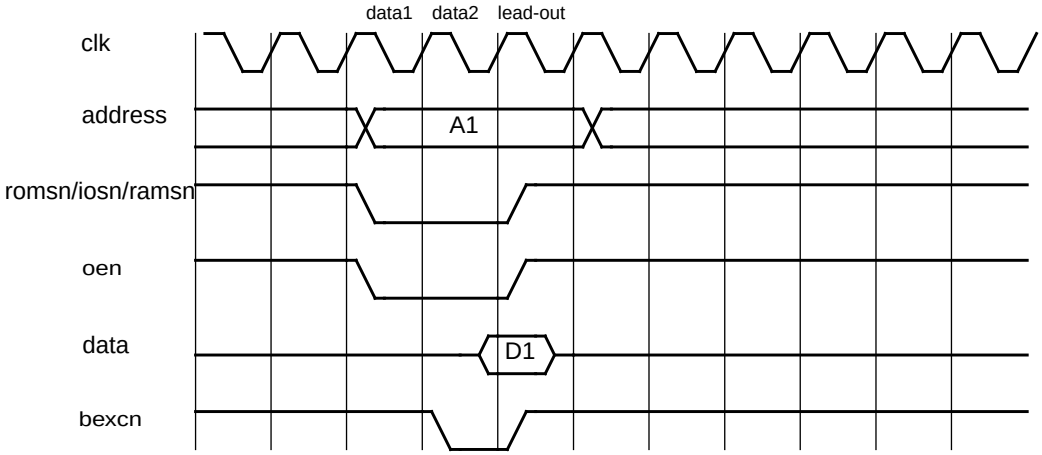


Figure 53. Read cycle with BEXCN.

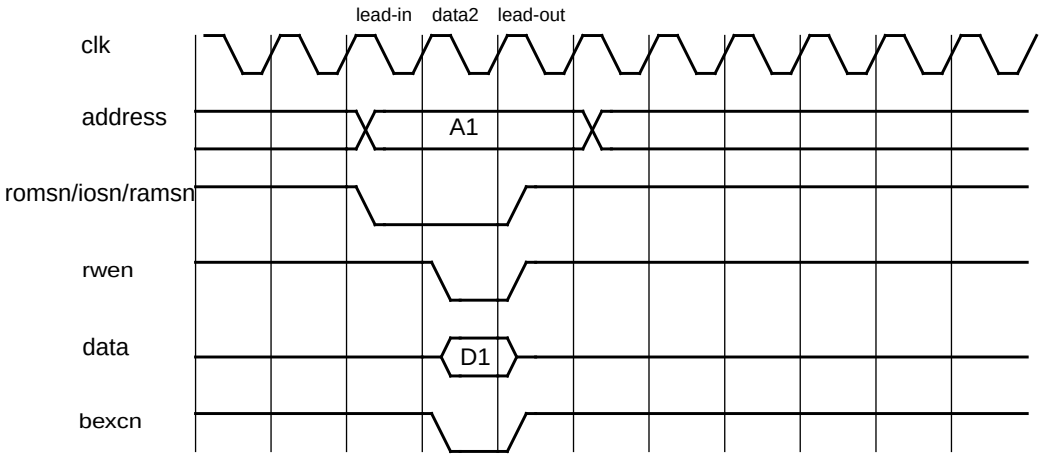


Figure 54. Write cycle with BEXCN. Chip-select (iosn) is not asserted in lead-in cycle for I/O-accesses.

8.13 Attaching an external DRAM controller

To attach an external DRAM controller, RAMSN[4] should be used since it allows the cycle time to vary through the use of BRDYN. In this way, delays can be inserted as required for opening of banks and refresh.

8.14 Output enable timing

A drive signal vector for the data I/O-pads is provided which has one drive signal for each data bit. It can be used if the synthesis tool does not generate separate registers automatically for the current technology. This can remove timing problems with output delay. An additional vector is used for the separate SDRAM bus.

8.15 Registers

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

Table 36. FTMCTRL memory controller registers

APB Address offset	Register
0x0	Memory configuration register 1 (MCFG1)
0x4	Memory configuration register 2 (MCFG2)
0x8	Memory configuration register 3 (MCFG3)
0xC	Memory configuration register 4 (MCFG4)

8.15.1 Memory configuration register 1 (MCFG1)

Memory configuration register 1 is used to program the timing of rom and IO accesses.

Table 37. Memory configuration register 1.

31	30	29	28	27	26	25	24	23 ... 20	19	18	17 ... 14	13 ... 12	11	10	9 ... 8	7 ... 4	3 ... 0
	PBRDY	ABRDY	IOBUSW		IBRDY	BEXCN		IO WAITSTATES	IOEN		ROMBANKSZ	RESERVED	PWEN		PROM WIDTH	PROM WRITE WS	PROM READ WS

31 RESERVED

30 PROM area bus ready enable (PBRDY) - Enables bus ready (BRDYN) signalling for the PROM area. Reset to ‘0’.

29 Asynchronous bus ready (ABRDY) - Enables asynchronous bus ready.

28 : 27 I/O bus width (IOBUSW) - Sets the data width of the I/O area (“00”=8, “10” =32).

26 I/O bus ready enable (IBRDY) - Enables bus ready (BRDYN) signalling for the I/O area. Reset to ‘0’.

25 Bus error enable (BEXCN) - Enables bus error signalling for all areas. Reset to ‘0’.

24 RESERVED

23 : 20 I/O waitstates (IO WAITSTATES) - Sets the number of waitstates during I/O accesses (“0000”=0, “0001”=1, “0010”=2,..., “1111”=15).

19 I/O enable (IOEN) - Enables accesses to the memory bus I/O area.

18 RESERVED

17 : 14 PROM bank size (ROMBANKSZ) - Returns current PROM bank size when read. “0000” is a special case and corresponds to a bank size of 256 MByte. All other values give the bank size in binary steps: “0001”=16 kByte, “0010”=32 kByte, ... , “1111”=256 MByte. For value “0000” or “1111” only two chip selects are available. For other values, two chip select signals are available for fixed bank sizes. For other values, four chip select signals are available for programmable bank sizes.

Programmable bank sizes can be changed by writing to this register field. The written values correspond to the bank sizes and number of chip-selects as above. Reset to “0000” when programmable.

13:12 RESERVED

11 PROM write enable (PWEN) - Enables write cycles to the PROM area.

10 RESERVED

9 : 8 PROM width (PROM WIDTH) - Sets the data width of the PROM area (“00”=8, “10”=32). At reset, these bits are initialized with the value of *gpio[1:0]*.

7 : 4 PROM write waitstates (PROM WRITE WS) - Sets the number of wait states for PROM write cycles (“0000”=0, “0001”=2, “0010”=4,..., “1111”=30).

3 : 0 PROM read waitstates (PROM READ WS) - Sets the number of wait states for PROM read cycles (“0000”=0, “0001”=2, “0010”=4,...,”1111”=30). Reset to “1111”.

During reset, the prom width (bits [9:8]) are set with value on *gpio[1:0]* inputs. The PROM waitstates fields are set to 15 (maximum). External bus error and bus ready are disabled. All other fields are undefined.

8.15.2 Memory configuration register 2 (MCFG2)

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

Table 38. Memory configuration register 2.

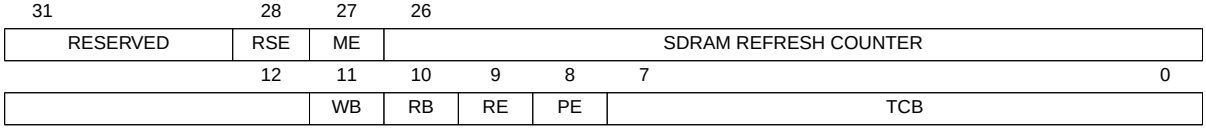
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDRF	TRP	SDRAM TRFC			TCAS	SDRAM BANKSZ			SDRAM COLSZ		SDRAM CMD		D64	SDPB	
15	14	13	12			9	8	7	6	5	4	3	2	1	0
	SE	SI	RAM BANK SIZE					RBRDY	RMW	RAM WIDTH		RAM WRITE WS		RAM READ WS	

31	SDRAM refresh (SDRF) - Enables SDRAM refresh.
30	SRAM TRP parameter (TRP) - t_{RP} will be equal to 2 or 3 system clocks (0/1).
29 : 27	SDRAM TRFC parameter (SDRAM TRFC) - t_{RFC} will be equal to 3+field-value system clocks.
26	SDRAM TCAS parameter (TCAS) - Selects 2 or 3 cycle CAS delay (0/1). When changed, a LOAD-COMMAND-REGISTER command must be issued at the same time. Also sets RAS/CAS delay ($t_{RCD)}$.
25 : 23	SDRAM bank size (SDRAM BANKSZ) - Sets the bank size for SDRAM chip selects (“000”=4 MByte, “001”=8 MByte, “010”=16 MByte.... “111”=512 MByte).
22 : 21	SDRAM column size (SDRAM COLSZ) - “00”=256, “01”=512, “10”=1024, “11”=4096 when bit 25:23=”111” 2048 otherwise.
20 : 19	SDRAM command (SDRAM CMD) - Writing a non-zero value will generate a SDRAM command. “01”=PRECHARGE, “10”=AUTO-REFRESH, “11”=LOAD-COMMAND-REGISTER. The field is reset after the command has been executed.
18	64-bit SDRAM data bus (D64) - Reads ‘1’ if the memory controller is configured for 64-bit SDRAM data bus width, ‘0’ otherwise. Read-only.
17	SDRAM Page Burst (SDPB) - SDRAM programmed for page bursts on read when set, else programmed for line burst lengths of 8 on read. Only line burst is implemented.
16 : 15	RESERVED
14	SDRAM enable (SE) - Enables the SDRAM controller and disables fifth SRAM bank (RAMSN[4]).
13	SRAM disable (SI) - Disables accesses to SRAM bank if bit 14 (SE) is set to ‘1’.
12 : 9	RAM bank size (RAM BANK SIZE) - Sets the size of each RAM bank (“0000”=8 kByte, “0001”=16 kByte, ..., “1111”=256 MByte).
8	RESERVED
7	RAM bus ready enable (RBRDY) - Enables bus ready signalling for the RAM area.
6	Read-modify-write enable (RMW) - Enables read-modify-write cycles for sub-word writes to 32-bit areas with common write strobe (no byte write strobe). Must not be set for 8-bit SRAM areas without EDAC protection.
5 : 4	RAM width (RAM WIDTH) - Sets the data width of the RAM area (“00”=8, “1X”=32).
3 : 2	RAM write waitstates (RAM WRITE WS) - Sets the number of wait states for RAM write cycles (“00”=0, “01”=1, “10”=2, “11”=3).
1 : 0	RAM read waitstates (RAM READ WS) - Sets the number of wait states for RAM read cycles (“00”=0, “01”=1, “10”=2, “11”=3).

8.15.3 Memory configuration register 3 (MCFG3)

MCFG3 contains the reload value for the SDRAM refresh counter and to control and monitor the memory EDAC.

Table 39. Memory configuration register 3.



31 : 29	RESERVED
28	Reed-Solomon EDAC enable (RSE) - if set, will enable Reed-Solomon protection of SDRAM area when implemented
27	Memory EDAC (ME) - Indicates if memory EDAC is present.
26 : 12	SDRAM refresh counter reload value (SDRAM REFRESH COUNTER)
11	EDAC diagnostic write bypass (WB) - Enables EDAC write bypass.
10	EDAC diagnostic read bypass (RB) - Enables EDAC read bypass.
9	RAM EDAC enable (RE) - Enable EDAC checking of the RAM area (including SDRAM).
8	PROM EDAC enable (PE) - Enable EDAC checking of the PROM area. Ar reset, this bit is initialized with the value of *gpio[2]*.
7 : 0	Test checkbits (TCB) - This field replaces the normal checkbits during write cycles when WB is set. It is also loaded with the memory checkbits during read cycles when RB is set.

During reset, the prom EDAC usage (bit [8]) is set with the value on *gpio[2]*.

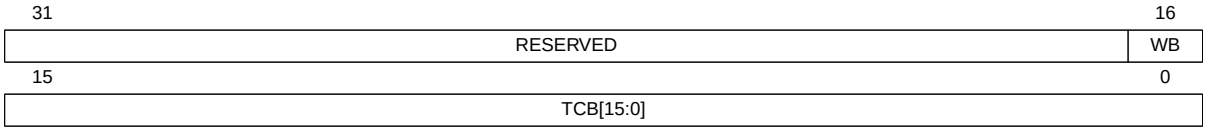
The period between each AUTO-REFRESH command is calculated as follows:

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

8.15.4 Memory configuration register 4 (MCFG4)

MCFG4 provides means to insert Reed-Solomon EDAC errors into memory for diagnostic purposes.

Table 40. Memory configuration register 4



31 : 17	RESERVED
16	EDAC diagnostic write bypass (WB) - Enables EDAC write bypass. Identical to WB in MCFG3.
15 : 0	Test checkbits (TCB) - This field replaces the normal checkbits during write cycles when WB is set. It is also loaded with the memory checkbits during read cycles when RB is set. Note that TCB[7:0] are identical to TCB[7:0] in MCFG3

8.16 Signal definitions and reset values

The signals and their reset values are described in table 41.

Table 41. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
address[27:0]	Output	Memory address	High	Undefined
data[31:0]	Input/Output	Memory data	High	Tri-state
cb[15:0]	Input/Output	Check bits	High	Tri-state
ramsn[4:0]	Output	SRAM chip select	Low	Logical 1
ramoen[4:0]	Output	SRAM output enable	Low	Logical 1
rwen[3:0]	Output	SRAM write byte enable: rwen[0] corresponds to data[31:24], rwen[1] corresponds to data[23:16], rwen[2] corresponds to data[15:8], rwen[3] corresponds to data[7:0]. Any rwen[ ] signal can be used for cb[ ].	Low	Logical 1
ramben[3:0]	Output	SRAM read/write byte enable: ramben[0] corresponds to data[31:24], ramben[1] corresponds to data[23:16], ramben[2] corresponds to data[15:8], ramben[3] corresponds to data[7:0]. Any ramben[ ] signal can be used for cb[ ].	Low	Logical 1
oen	Output	Output enable	Low	Logical 1
writen	Output	Write strobe	Low	Logical 1
read	Output	Read strobe	High	Logical 1
iosn	Output	IO area chip select	Low	Logical 1
romsn[3:0]	Output	PROM chip select	Low	Logical 1
brdyn	Input	Bus ready. Extends accesses to the IO area.	Low	-
bexcn	Input	Bus exception.	Low	-
sa[14:0]	Output	SDRAM address	High	Undefined
sd[31:0]	Input/Output	SDRAM data	High	Tri-state
scb[15:0]	Input/Output	SDRAM check bits	High	Tri-state
sdcsn[1:0]	Output	SDRAM chip select	Low	Logical 1
sdwen	Output	SDRAM write enable	Low	Logical 1
sdrasn	Output	SDRAM row address strobe	Low	Logical 1
sdcasn	Output	SDRAM column address strobe	Low	Logical 1
sddqm[5:0]	Output	SDRAM data mask: sddqm[3] corresponds to sd[31:24], sddqm[2] corresponds to sd[23:16], sddqm[1] corresponds to sd[15:8], sddqm[0] corresponds to sd[7:0]. Any sddqm[ ] signal can be used for scb[ ].	Low	Logical 1
gpio[1:0]	Input	Configuring PROM data width at reset. 8-bit data width when “00”, and 32-bit when “10”.	-	-
gpio[2]	Input	Enabling EDAC usage for PROM at reset.	High	-

8.17 Timing

The timing waveforms and timing parameters are shown in figure 55 and 56 are defined in table 42.

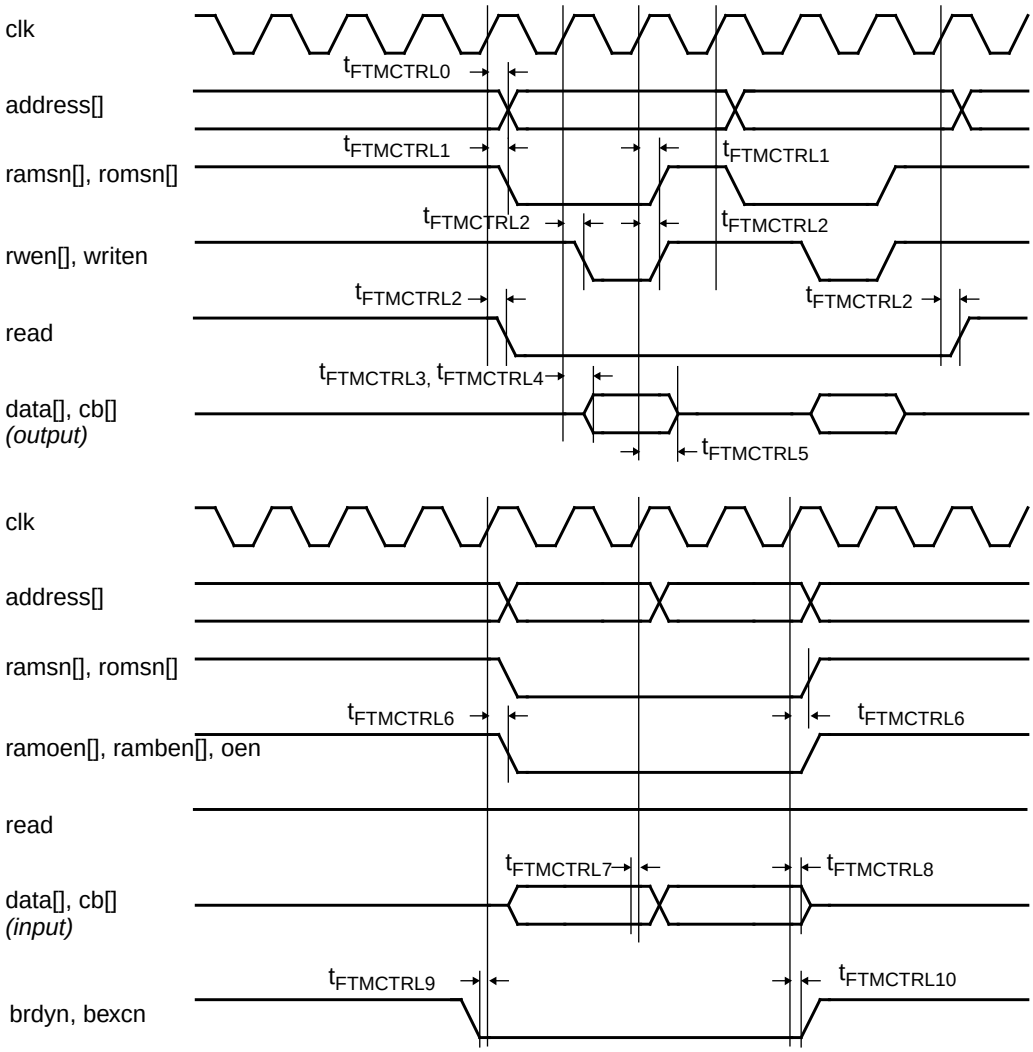


Figure 55. Timing waveforms - SRAM, PROM accesses

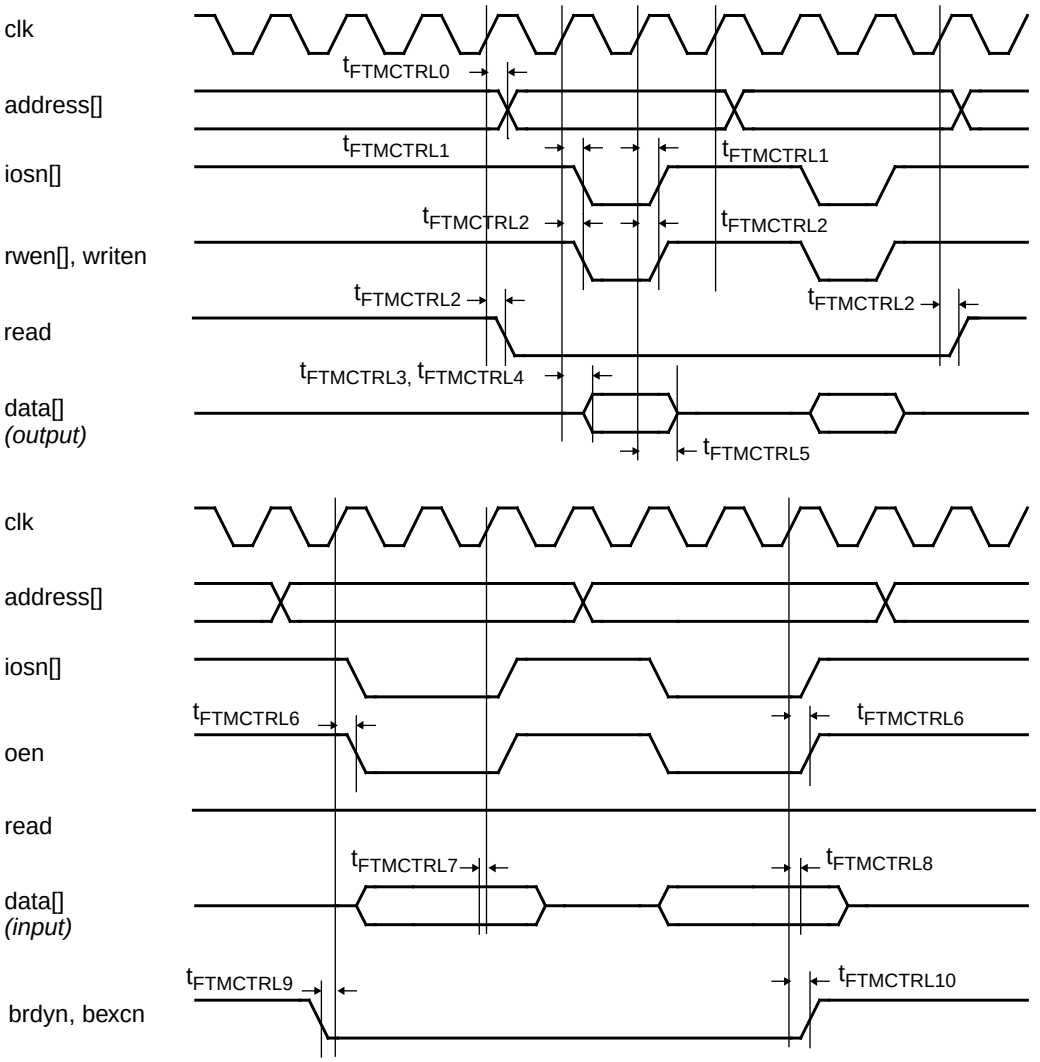


Figure 56. Timing waveforms - I/O accesses

Table 42. Timing parameters - SRAM, PROM and I/O accesses

Name	Parameter	Reference edge	Min	Max	Unit
$t_{FTMCTRL0}$	address clock to output delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL1}$	clock to output delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL2}$	clock to output delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL3}$	clock to data output delay	rising clk edge	3	21.5	ns
$t_{FTMCTRL4}$	clock to data non-tri-state delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL5}$	clock to data tri-state delay	rising clk edge	3	21.5	ns
$t_{FTMCTRL6}$	clock to output delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL7}$	data input to clock setup	rising clk edge	7	-	ns
$t_{FTMCTRL8}$	data input from clock hold	rising clk edge	1	-	ns
$t_{FTMCTRL9}$	input to clock setup	rising clk edge	7	-	ns
$t_{FTMCTRL10}$	input from clock hold	rising clk edge	1	-	ns

The timing waveforms and timing parameters are shown in figure 57 and are defined in table 43.

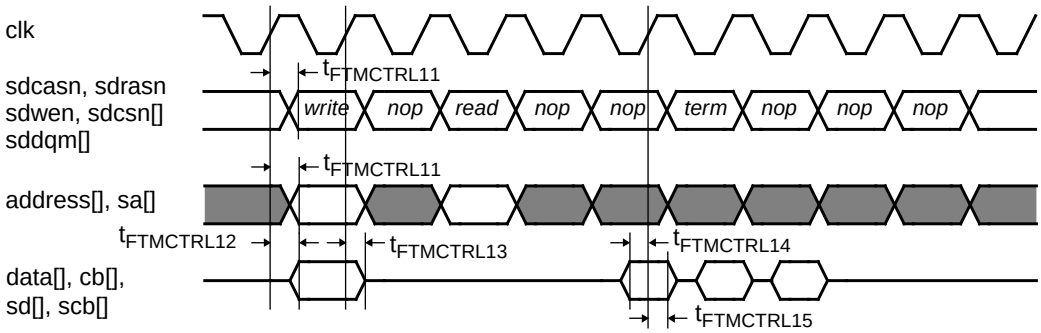


Figure 57. Timing waveforms - SDRAM accesses

Table 43. Timing parameters - SDRAM accesses

Name	Parameter	Reference edge	Min	Max	Unit
$t_{FTMCTRL11}$	clock to output delay	rising clk edge	0	21.5	ns
$t_{FTMCTRL12}$	clock to data output delay	rising clk edge	3	21.5	ns
$t_{FTMCTRL13}$	data clock to data tri-state delay	rising clk edge	3	21.5	ns
$t_{FTMCTRL14}$	data input to clock setup	rising clk edge	2	-	ns
$t_{FTMCTRL15}$	data input from clock hold	rising clk edge	2	-	ns

9 Interrupt Controller

9.1 Overview

The interrupts generated on the interrupt bus are all forwarded to the interrupt controller. The interrupt controller prioritizes, masks and propagates the interrupt with the highest priority to the processor.

9.2 Operation

9.2.1 Interrupt prioritization

The interrupt controller monitors interrupt 1 - 15 of the interrupt bus. 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 prioritised within each level, with interrupt 15 having the highest priority and interrupt 1 the lowest. The highest interrupt from level 1 will be forwarded to the processor. If no unmasked pending interrupt exists on level 1, then the highest unmasked interrupt from level 0 will be forwarded.

Interrupts are prioritised at system level, while masking and forwarding of interrupts in done for each processor separately. Each processor in an multiprocessor system has separate interrupt mask and force registers. When an interrupt is signalled on the interrupt bus, the interrupt controller will prioritize interrupts, perform interrupt masking for each processor according to the mask in the corresponding mask register and forward the interrupts to the processors.

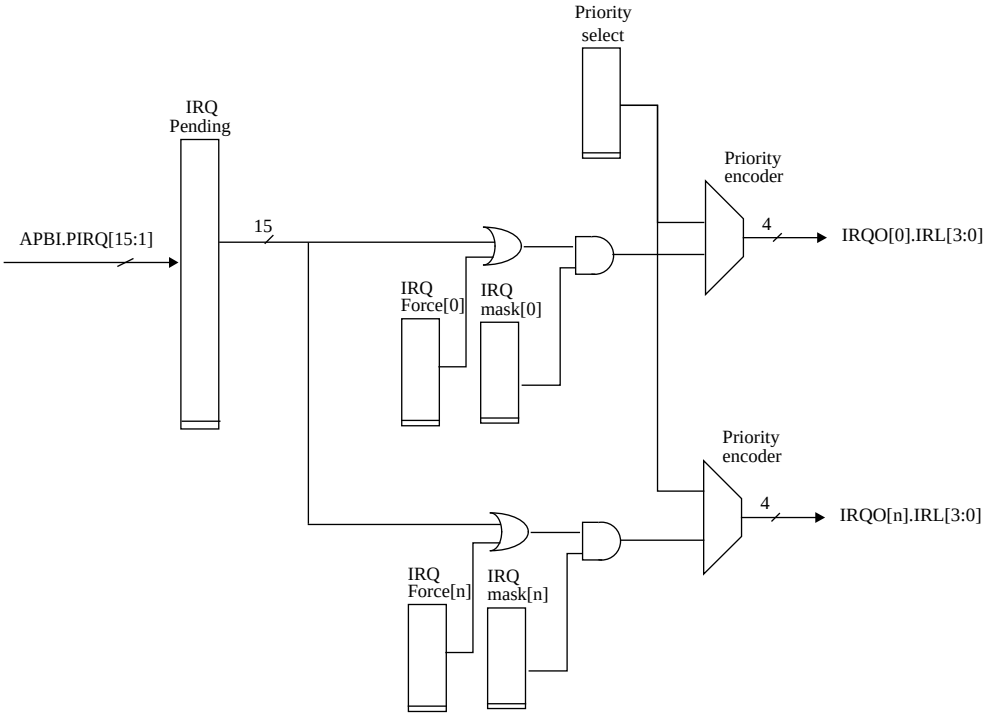


Figure 58. Interrupt controller block diagram

When a processor acknowledges the interrupt, the corresponding pending bit will automatically be cleared. Interrupt can also be forced by setting a bit in the interrupt force register. In this case, the pro-

cessor acknowledgement will clear 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 that interrupt 15 cannot be maskable by the LEON3 processor and should be used with care - most operating systems do not safely handle this interrupt.

9.2.2 Processor status monitoring

The processor status can be monitored through the Multiprocessor Status Register. The STATUS field in this register indicates if a processor is halted (‘1’) or running (‘0’). A halted processor can be reset and restarted by writing a ‘1’ to its status field. After reset, all processors except processor 0 are halted. When the system is properly initialized, processor 0 can start the remaining processors by writing to their STATUS bits.

9.3 Registers

The core is controlled through registers mapped into APB address space. The number of implemented registers depend on number of processor in the multiprocessor system.

Table 44. Interrupt Controller registers

APB address offset	Register
0x00	Interrupt level register
0x04	Interrupt pending register
0x08	Interrupt force register (NCPU = 0)
0x0C	Interrupt clear register
0x10	Multiprocessor status register
0x40	Processor interrupt mask register
0x80	Processor interrupt force register

9.3.1 Interrupt level register

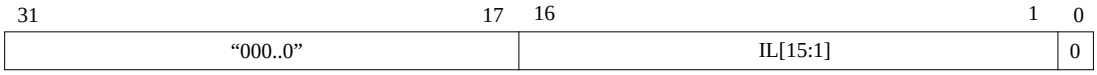


Figure 59. Interrupt level register

[31:16] Reserved.
[15:1] Interrupt Level *n* (IL[*n*]): Interrupt level for interrupt *n*.
[0] Reserved.

9.3.2 Interrupt pending register

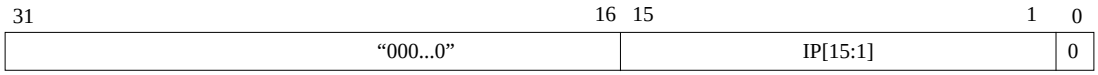


Figure 60. Interrupt pending register

[31:17] Reserved.
[16:1] Interrupt Pending n (IP[n]): Interrupt pending for interrupt n.
[0] Reserved

9.3.3 Interrupt force register (NCPU = 0)

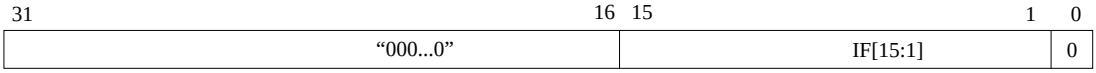


Figure 61. Interrupt force register

[31:16] Reserved.
[15:1] Interrupt Force *n* (IF[*n*]): Force interrupt no. *n*. The resulting IF[*n*] is the value of the written value to IF[*n*].
[0] Reserved.

9.3.4 Interrupt clear register

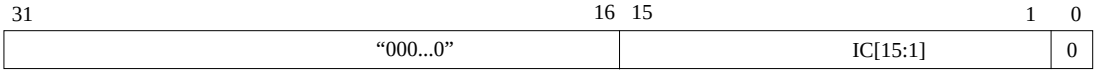


Figure 62. Interrupt clear register

[31:16] Reserved.
[15:1] Interrupt Clear *n* (IC[*n*]): Writing ‘1’ to IC*n* will clear interrupt *n*.Write only, reads zeros.
[0] Reserved.

9.3.5 Multiprocessor status register

Figure 63. Processor status register

[31:28] NCPU. Number of CPU’s in the system -1. Read only. Fixed to +, i.e. 1processor.
[27:16] Reserved.
[15:1] Power-down status of CPU [*n*]: reads ‘1’ = power-down, ‘0’ = running. Write to start processor *n*: ‘1’=to start, ‘0=has no effect.

9.3.6 Processor interrupt mask register

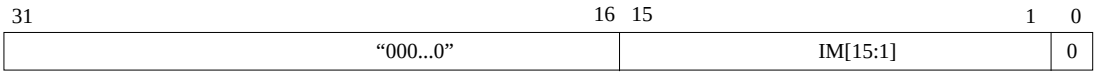


Figure 64. Processor interrupt mask register

[31:16] Reserved.
[15:1] Interrupt Mask *n* (IM[*n*]): If IMn = 0 the interrupt *n* is masked, otherwise it is enabled.
[0] Reserved.

10 UART Serial Interface

10.1 Overview

The interface is provided for serial communications. The UART supports data frames with 8 data bits, one optional parity bit and one stop bit. To generate the bit-rate, each UART has a programmable 12-bit clock divider. Two FIFOs, each 2 bytes deep, are used for data transfer between the bus and UART. Hardware flow-control is supported through the RTSN/CTSN hand-shake signals. Parity is supported.

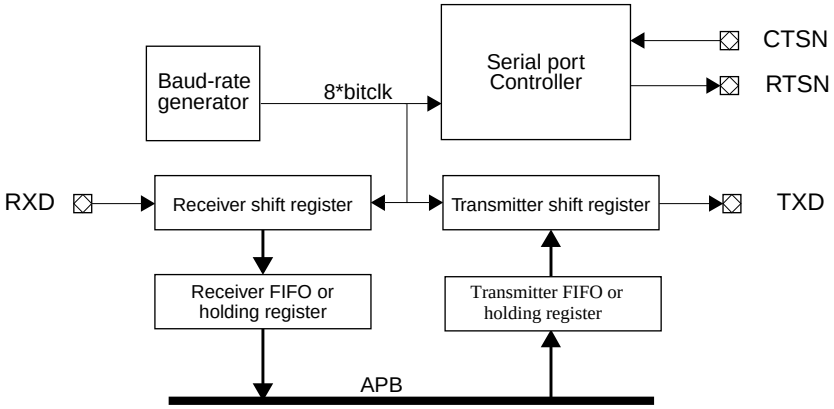


Figure 65. Block diagram

10.2 Operation

10.2.1 Transmitter operation

The transmitter is enabled through the TE bit in the UART control register. Data that is to be transferred is stored in the FIFO/holding register by writing to the data register. When ready to transmit, data is transferred from the transmitter FIFO/holding register 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 66). The least significant bit of the data is sent first.

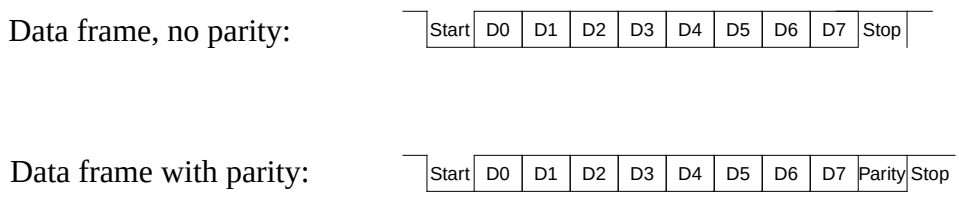


Figure 66. UART data frames

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) will be

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. If the transmitter is disabled, it will immediately stop 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 FIFO (or holding register) is full. If this is done, data might be overwritten and one or more frames are 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 (less than half of entries in the FIFO contain data). The TF control bit enables FIFO interrupts when set. The status register also contains a counter (TCNT) showing the current number of data entries in the FIFO.

When flow control is enabled, the CTSN input must be low in order for the character to be transmitted. If it is deasserted in the middle of a transmission, the character in the shift register is transmitted and the transmitter serial output then remains inactive until CTSN is asserted again. If the CTSN is connected to a receivers RTSN, overrun can effectively be prevented.

10.2.2 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 clocks 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 centre 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 have to have 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 configurable FIFO which is identical to the one in the transmitter. As mentioned in the transmitter part, both the holding register and FIFO will be referred to as FIFO.

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:ed with the old values before they are stored into the status register. Thus, they are not cleared until written to with zeros from the AMBA 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 will be lost and the overrun bit will be set in the UART status register. A break received (BR) is indicated when a BREAK has been received, which is a framing error with all data received being zero.

If flow control is enabled, then the RTSN will be negated (high) when a valid start bit is detected and the receiver FIFO is full. When the holding register is read, the RTSN will automatically be reasserted again.

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 enables receiver FIFO interrupts when set. A RCNT field is also available showing the current number of data frames in the FIFO.

10.3 Baud-rate generation

Each UART contains a 12-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. It is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be 8 times the desired baud-rate. If the EC bit is set, the tick will be generated with the same frequency as the external clock input instead of at the scaler underflow rate. In this case, the frequency of external clock must be less than half the frequency of the system clock. Note that an external clock input is not supported in this design.

10.4 Loop back mode

If the LB bit in the UART control register is set, the UART will be in loop back mode. In this mode, the transmitter output is internally connected to the receiver input and the RTSN is connected to the CTSN. 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.

10.5 FIFO debug mode

FIFO debug mode is entered by setting the debug mode bit in the control register. In this mode it is possible to read the transmitter FIFO and write the receiver FIFO through the FIFO debug register. The transmitter output is held inactive when in debug mode. A write to the receiver FIFO generates an interrupt if receiver interrupts are enabled.

10.6 Interrupt generation

For FIFOs, 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 (that is, 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).

To reduce interrupt occurrence a delayed receiver interrupt is available. It is enabled using the delayed interrupt enable (DI) bit. When enabled a timer is started each time a character is received and an interrupt is only generated if another character has not been received within 4 character + 4 bit times. If receiver FIFO interrupts are enabled a pending character interrupt will be cleared when the FIFO interrupt is active since the character causing the pending irq state is already in the FIFO and is noticed by the driver through the FIFO interrupt.

There is also a separate interrupt for break characters. When enabled an interrupt will always be generated immediately when a break character is received even when delayed receiver interrupts are enabled. When break interrupts are disabled no interrupt will be generated for break characters when delayed interrupts are enabled.

When delayed interrupts are disabled the behavior is the same for the break interrupt bit except that an interrupt will be generated for break characters if receiver interrupt enable is set even if break interrupt is disabled.

An interrupt can also be enabled for the transmitter shift register. When enabled the core will generate an interrupt each time the shift register goes from a non-empty to an empty state.

10.7 Registers

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

Table 45. UART registers

APB address offset	Register
0x0	UART Data register
0x4	UART Status register
0x8	UART Control register
0xC	UART Scaler register
0x10	UART FIFO debug register

10.7.1 UART Data Register

Table 46. UART data register

31 ... 8	7 ... 0
RESERVED	DATA

7: 0 Receiver holding register or FIFO (read access)

7: 0 Transmitter holding register or FIFO (write access)

10.7.2 UART Status Register

Table 47. UART status register

31 ... 26	25 ... 20	19 ... 11	10	9	8	7	6	5	4	3	2	1	0
RCNT	TCNT	RESERVED	RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR

31: 26 Receiver FIFO count (RCNT) - shows the number of data frames in the receiver FIFO. Reset 0.

25: 20 Transmitter FIFO count (TCNT) - shows the number of data frames in the transmitter FIFO. Reset 0.

10 Receiver FIFO full (RF) - indicates that the Receiver FIFO is full. Reset 0.

9 Transmitter FIFO full (TF) - indicates that the Transmitter FIFO is full. Reset 0.

8 Receiver FIFO half-full (RH) -indicates that at least half of the FIFO is holding data. Reset 0.

7 Transmitter FIFO half-full (TH) - indicates that the FIFO is less than half-full. Reset 0.

6 Framing error (FE) - indicates that a framing error was detected. Reset 0.

5 Parity error (PE) - indicates that a parity error was detected. Reset 0.

4 Overrun (OV) - indicates that one or more character have been lost due to overrun. Reset 0.

3 Break received (BR) - indicates that a BREAK has been received. Reset 0.

2 Transmitter FIFO empty (TE) - indicates that the transmitter FIFO is empty. Reset 1.

1 Transmitter shift register empty (TS) - indicates that the transmitter shift register is empty. Reset 1.

0 Data ready (DR) - indicates that new data is available in the receiver holding register. Reset 0.

10.7.3 UART Control Register

Table 48. UART control register

31	30	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FA	RESERVED		SI	DI	BI	DB	RF	TF	EC	LB	FL	PE	PS	TI	RI	TE	RE

31	FIFOs available (FA) - Set to 1 when receiver and transmitter FIFOs are available. When 0, only holding register are available. Read only.
30: 15	RESERVED
14	Transmitter shift register empty interrupt enable (SI) - When set, an interrupt will be generated when the transmitter shift register becomes empty. See section 10.6 for more details.
13	Delayed interrupt enable (DI) - When set, delayed receiver interrupts will be enabled and an interrupt will only be generated for received characters after a delay of 4 character times + 4 bits if no new character has been received during that interval. This is only applicable if receiver interrupt enable is set. See section 10.6 for more details. Not Reset.
12	Break interrupt enable (BI) - When set, an interrupt will be generated each time a break character is received. See section 10.6 for more details. Not Reset.
11	FIFO debug mode enable (DB) - when set, it is possible to read and write the FIFO debug register. Not Reset.
10	Receiver FIFO interrupt enable (RF) - when set, Receiver FIFO level interrupts are enabled
9	Transmitter FIFO interrupt enable (TF) - when set, Transmitter FIFO level interrupts are enabled.
8	External Clock (EC) - if set, the UART scaler will not be clocked.
7	Loop back (LB) - if set, loop back mode will be enabled. Not reset.
6	Flow control (FL) - if set, enables flow control using CTS/RTS (when implemented). Reset 0.
5	Parity enable (PE) - if set, enables parity generation and checking (when implemented). Not reset.
4	Parity select (PS) - selects parity polarity (0 = even parity, 1 = odd parity) (when implemented). Not reset.
3	Transmitter interrupt enable (TI) - if set, interrupts are generated when characters are transmitted (see section 10.6 for details). Not Reset.
2	Receiver interrupt enable (RI) - if set, interrupts are generated when characters are received (see section 10.6 for details). Not Reset.
1	Transmitter enable (TE) - if set, enables the transmitter. Reset 0.
0	Receiver enable (RE) - if set, enables the receiver. Reset 0.

10.7.4 UART Scaler Register

Table 49. UART scaler reload register

31	12	11	0
RESERVED		SCALER RELOAD VALUE	

11: 0	Scaler reload value

10.7.5 UART FIFO Debug Register

Table 50. UART FIFO debug register

31	8	7	0
RESERVED		DATA	

7: 0	Transmitter holding register or FIFO (read access)
7: 0	Receiver holding register or FIFO (write access)

10.8 Signal definitions and reset values

The signals and their reset values are described in table 51.

Table 51. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
txd[]	Output	UART transmit data line	-	Logical 1
rtsn[]	Output	Ready To Send	Low	Logical 1
rxd[]	Input	UART receive data line	-	-
ctsn[]	Input	Clear To Send	Low	-

10.9 Timing

The timing waveforms and timing parameters are shown in figure 67 and are defined in table 52.

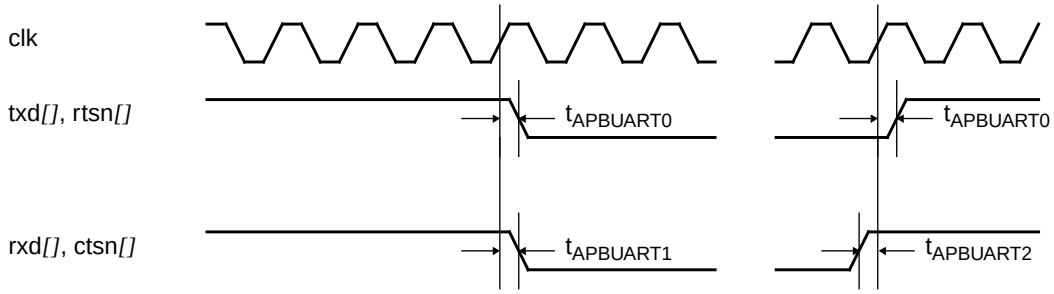


Figure 67. Timing waveforms

Table 52. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{APBUART0}$	clock to output delay	rising *clk* edge	0	25	ns
$t_{APBUART1}$	input to clock hold	rising *clk* edge	-	-	ns
$t_{APBUART2}$	input to clock setup	rising *clk* edge	-	-	ns

Note: The *ctsn[]* and *rxd[]* inputs are re-synchronized internally. These signals do not have to meet any setup or hold requirements.

11 General Purpose Timer Unit

11.1 Overview

The General Purpose Timer Unit provides a common prescaler and decrementing timer(s). The unit implements one 16 bit prescaler and 3 decrementing 32 bit timer(s). The unit is capable of asserting interrupt on timer(s) underflow.

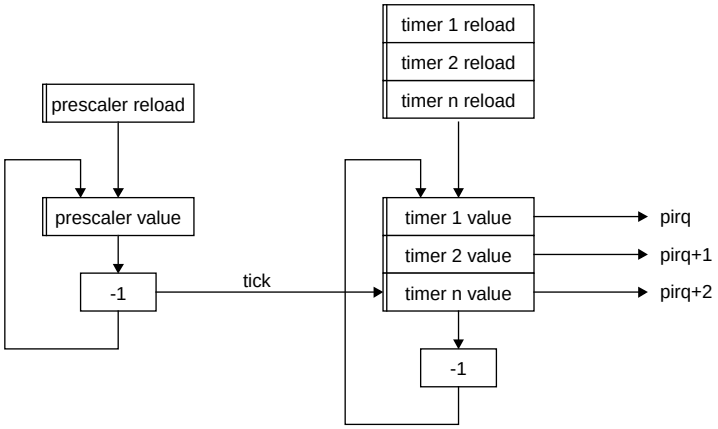


Figure 68. General Purpose Timer Unit block diagram

11.2 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.

The operation of each timers is controlled through its control register. A timer is enabled by setting the enable bit in the control register. The timer value is then decremented on each prescaler tick. When a timer underflows, it will automatically be reloaded with the value of the corresponding timer reload register if the restart bit in the control register is set, otherwise it will stop at -1 and reset the enable bit.

The timer unit will signal an interrupt on appropriate line when a timer underflows (if the interrupt enable bit for the current timer is set). The interrupt pending bit in the control register of the underflown timer will be set and remain set until cleared by writing ‘0’.

To minimize complexity, timers share the same decrementer. This means that the minimum allowed prescaler division factor is *ntimers*+1 (reload register = *ntimers*) where *ntimers* is the number of implemented timers, i.e. 3.

By setting the chain bit in the control register timer *n* can be chained with preceding timer *n*-1. Timer *n* will be decremented every time when timer *n*-1 underflows.

Each timer can be reloaded with the value in its reload register at any time by writing a ‘one’ to the load bit in the control register. The last timer acts as a watchdog, driving a watchdog output signal when expired. At reset, the scaler is set to all ones and the watchdog timer is set to 0xFFF. (This is done for new releases of the part, older parts have a timer initialization value of 0xFFFFFF.)

11.3 Registers

The core is programmed through registers mapped into APB address space. The number of implemented registers depend on number of implemented timers.

Table 53. General Purpose Timer Unit registers

APB address offset	Register
0x00	Scaler value
0x04	Scaler reload value
0x08	Configuration register
0x0C	Unused
0x10	Timer 1 counter value register
0x14	Timer 1 reload value register
0x18	Timer 1 control register
0x1C	Unused
0x*n*0	Timer *n* counter value register
0x*n*4	Timer *n* reload value register
0x*n*8	Timer *n* control register

Table 54. Scaler value

31	16	16-1	0
“000..0”		SCALER VALUE	

16-1: 0 Scaler value

Any unused most significant bits are reserved. Always reads as ‘000...0’.

Table 55. Scaler reload value

31	16	16-1	0
“000..0”		SCALER RELOAD VALUE	

16-1: 0 Scaler reload value

Any unused most significant bits are reserved. Always read as ‘000...0’.

Table 56. General Purpose Timer Unit Configuration Register

31	10	9	8	7	3	2	0
“000..0”		DF	SI	IRQ		TIMERS	

31: 10 Reserved. Always reads as ‘000...0’.

9 Disable timer freeze (DF). If set the timer unit can not be freezed, otherwise signal GPTI.DHALT freezes the timer unit.

8 Separate interrupts (SI). Reads ‘1’ if the timer unit generates separate interrupts for each timer, otherwise ‘0’. Read-only.

7: 3 APB Interrupt: If configured to use common interrupt all timers will drive APB interrupt nr. IRQ, otherwise timer *n*will drive APB Interrupt IRQ+*n* (has to be less the MAXIRQ). Read-only.

2: 0 Number of implemented timers. Read-only.

Table 57. Timer counter value register

32-1	0
TIMER COUNTER VALUE	

32-1: 0 Timer Counter value. Decremented by 1 for each prescaler tick.

Any unused most significant bits are reserved. Always reads as ‘000...0’.

Table 58. Timer reload value register

32-1	0
TIMER RELOAD VALUE	

32-1: 0 Timer Reload value. This value is loaded into the timer counter value register when ‘1’ is written to load bit in the timers control register or when the RS bit is set in the control register and the timer underflows.

Any unused most significant bits are reserved. Always reads as ‘000...0’.

Table 59. General Purpose Timer Unit Configuration Register

31	7	6	5	4	3	2	1	0
“000..0”		DH	CH	IP	IE	LD	RS	EN

31: 7 Reserved. Always reads as ‘000...0’.

6 Debug Halt (DH): Value of GPTI.DHALT signal which is used to freeze counters (e.g. when a system is in debug mode). Read-only.

5 Chain (CH): Chain with preceding timer. If set for timer *n*, decrementing timer *n* begins when timer (*n*-1) underflows.

4 Interrupt Pending (IP): Sets when an interrupt is signalled. Remains ‘1’ until cleared by writing ‘0’ to this bit.

3 Interrupt Enable (IE): If set the timer signals interrupt when it underflows.

2 Load (LD): Load value from the timer reload register to the timer counter value register.

1 Restart (RS): If set, the timer counter value register is reloaded with the value of the reload register when the timer underflows

0 Enable (EN): Enable the timer.

11.4 Signal definitions and reset values

When the watchdog times out, the *wdogn* output is driven active low, else it is in tri-state and therefore might require an additional external pull-up.

The signals and their reset values are described in table 60.

Table 60. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
wdogn	Tri-state output	Watchdog output. Equivalent to interrupt pending bit of last timer.	Low	Tri-state

11.5 Timing

The timing waveforms and timing parameters are shown in figure 69 and are defined in table 61.

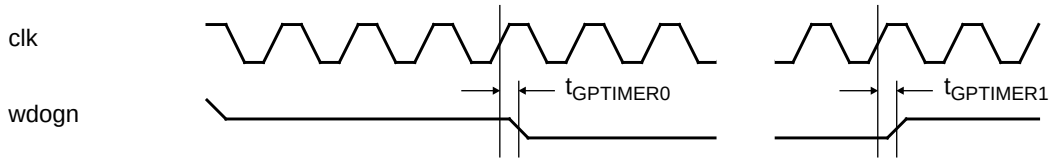


Figure 69. Timing waveforms

Table 61. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{GPTIMER0}$	clock to output delay	rising *clk* edge	0	25	ns
$t_{GPTIMER1}$	clock to output tri-state	rising *clk* edge	0	25	ns

12 General Purpose I/O Port

12.1 Overview

Each bit in the general purpose input output port can be individually set to 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 70 shows a diagram for one I/O line.

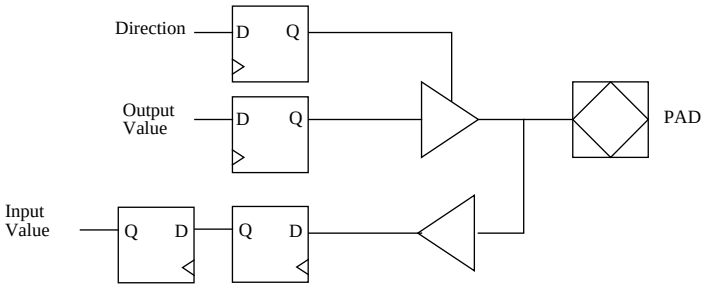


Figure 70. General Purpose I/O Port diagram

12.2 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 will enable the output buffer for the corresponding I/O line. The output value driven is taken from the I/O port output register.

Each I/O port, except gpio[0], can drive a separate interrupt line on the APB interrupt bus. The interrupt number is equal to the I/O line index (gpio[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’).

12.3 Registers

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

Table 62. General Purpose I/O Port registers

APB address offset	Register
0x00	I/O port data register
0x04	I/O port output register
0x08	I/O port direction register
0x0C	Interrupt mask register
0x10	Interrupt polarity register
0x14	Interrupt edge register

Table 63. I/O port data register

31 16	16-1 0
"000..0"	I/O port input value

16-1: 0 I/O port input value

Table 64. I/O port output register

31 16	16-1 0
"000..0"	I/O port output value

16-1: 0 I/O port output value

Table 65. I/O port direction register

31 16	16-1 0
"000..0"	I/O port direction value

16-1: 0 I/O port direction value (0=output disabled, 1=output enabled)

Table 66. Interrupt mask register

31 16	16-1 1	0
"000..0"	Interrupt mask	'0'

16-1: 1 Interrupt mask (0=interrupt masked, 1=intrrupt enabled)

Table 67. Interrupt polarity register

31 16	16-1 1	0
"000..0"	Interrupt polarity	'0'

16-1: 1 Interrupt polarity (0=low/falling, 1=high/rising)

Table 68. Interrupt edge register

31 16	16-1 1	0
"000..0"	Interrupt edge	'0'

16-1: 1 Interrupt edge (0=level, 1=edge)

12.4 Signal definitions and reset values

The signals and their reset values are described in table 69.

Table 69. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
gpio[]	Input/Output	General purpose input output	-	Tri-state

12.5 Timing

The timing waveforms and timing parameters are shown in figure 71 and are defined in table 70.

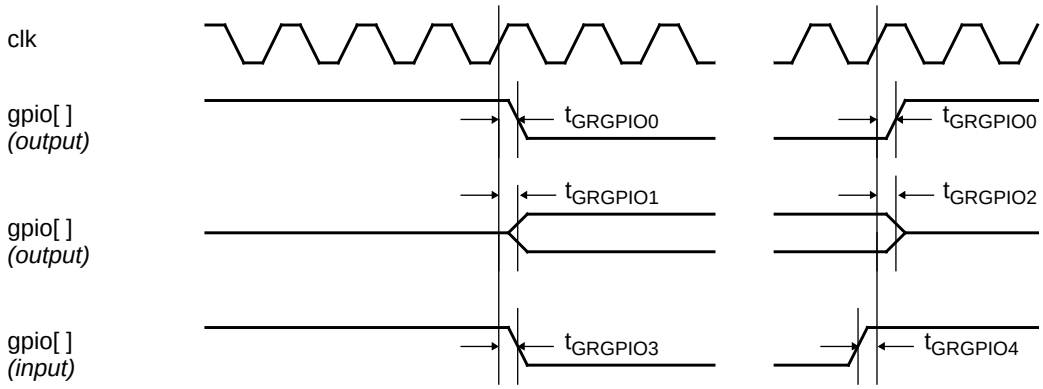


Figure 71. Timing waveforms

Table 70. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{GRGPIO0}$	clock to output delay	rising *clk* edge	0	25	ns
$t_{GRGPIO1}$	clock to non-tri-state delay	rising *clk* edge	0	25	ns
$t_{GRGPIO2}$	clock to tri-state delay	rising *clk* edge	0	25	ns
$t_{GRGPIO3}$	input to clock hold	rising *clk* edge	-	-	ns
$t_{GRGPIO4}$	input to clock setup	rising *clk* edge	-	-	ns

Note: The *gpio* inputs are re-synchronized internally. The signals do not have to meet any setup or hold requirements.

13 On-chip Memory with EDAC Protection

13.1 Overview

The on-chip memory is accessed via an AMBA AHB slave interface. The memory implements 4 or 2 kBytes of data, depending on configuration (see table 1). Registers are accessed via an AMB APB interface.

The on-chip memory implements volatile memory that is protected by means of Error Detection And Correction (EDAC). One error can be corrected and two errors can be detected, which is performed by using a (32, 7) BCH code. Some of the optional features available are single error counter, diagnostic reads and writes. Configuration is performed via a configuration register.

Figure 72 shows a block diagram of the internals of the memory.

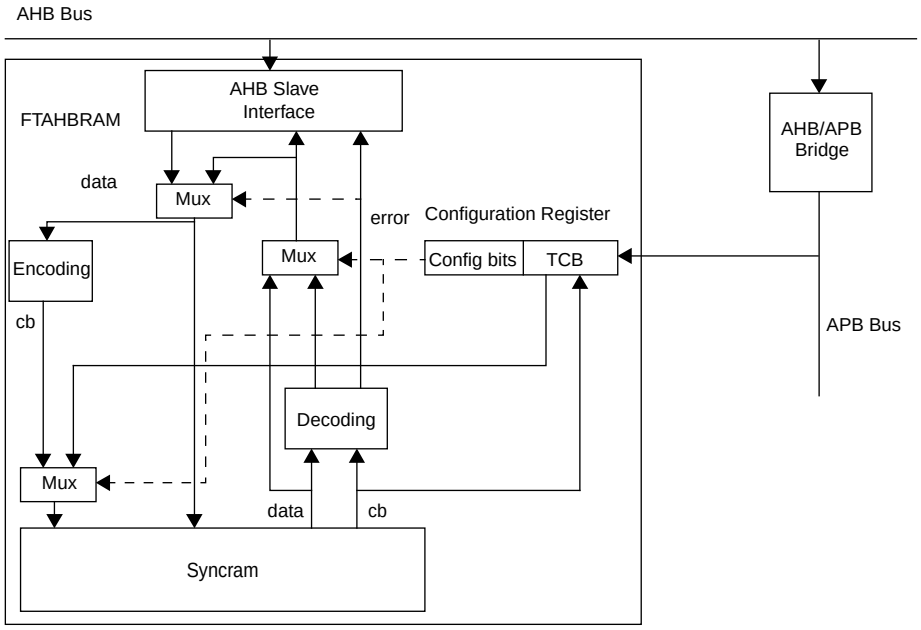


Figure 72. Block diagram

13.2 Operation

The on-chip fault tolerant memory is accessed through an AMBA AHB slave interface.

Run-time configuration is done by writing to a configuration register accessed through an AMBA APB interface.

The following can be configured during run-time: EDAC can be enabled and disabled. When it is disabled, reads and writes will behave as the standard memory. Read and write diagnostics can be controlled through separate bits. The single error counter can be reset.

If EDAC is disabled (EN bit in configuration register set to 0) write data is passed directly to the memory area and read data will appear on the AHB bus immediately after it arrives from memory. If EDAC is enabled write data is passed to an encoder which outputs a 7-bit checksum. The checksum is stored together with the data in memory and the whole operation is performed without any added waitstates. This applies to word stores (32-bit). If a byte or halfword store is performed, the whole

word to which the byte or halfword belongs must first be read from memory (read - modify - write). A new checksum is calculated when the new data is placed in the word and both data and checksum are stored in memory. This is done with 1 - 2 additional waitstates compared to the non EDAC case.

Reads with EDAC disabled are performed with 0 or 1 waitstates while there could also be 2 waitstates when EDAC is enabled. There is no difference between word and subword reads. Table 71 shows a summary of the number of waitstates for the different operations with and without EDAC.

Table 71. Summary of the number of waitstates for the different operations for the memory.

Operation	Waitstates with EDAC Disabled	Waitstates with EDAC Enabled
Read	0 - 1	0 - 2
Word write	0	0
Subword write	0	1 - 2

When EDAC is used, the data is decoded the first cycle after it arrives from the memory and appears on the bus the next cycle if no uncorrectable error is detected. The decoding is done by comparing the stored checksum with a new one which is calculated from the stored data. This decoding is also done during the read phase for a subword write. A so-called syndrome is generated from the comparison between the checksum and it determines the number of errors that occurred. One error is automatically corrected and this situation is not visible on the bus. Two or more detected errors cannot be corrected so the operation is aborted and the required two cycle error response is given on the AHB bus (see the AMBA manual for more details). If no errors are detected data is passed through the decoder unaltered.

As mentioned earlier the memory provides read and write diagnostics when EDAC is enabled. When write diagnostics are enabled, the calculated checksum is not stored in memory during the write phase. Instead, the TCB field from the configuration register is used. In the same manner, if read diagnostics are enabled, the stored checksum from memory is stored in the TCB field during a read (and also during a subword write). This way, the EDAC functionality can be tested during run-time. Note that checkbits are stored in TCB during reads and subword writes even if a multiple error is detected.

A single error counter (SEC) field is present in the configuration register, and is incremented each time a single databit error is encountered (reads or subword writes). The number of bits of this counter is 8. It is accessed through the configuration register. Each counter bit can be reset to zero by writing a one to it. The counter saturates at the value 2^8 - 1.

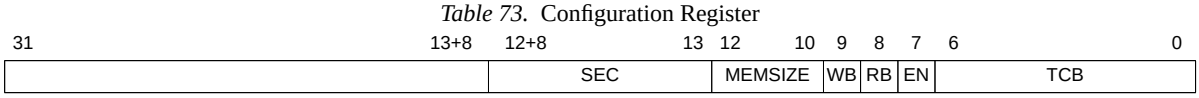
13.3 Registers

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

Table 72. FTAHBRAM registers

APB Address offset	Register
0x0	Configuration Register

Table 73. Configuration Register



12+8: 13 Single error counter (SEC): Incremented each time a single error is corrected (includes errors on checkbits). Each bit can be set to zero by writing a one to it.

12: 10 Log2 of the current memory size

Table 73. Configuration Register

9	Write Bypass (WB): When set, the TCB field is stored as check bits when a write is performed to the memory.
8	Read Bypass (RB) : When set during a read or subword write, the check bits loaded from memory are stored in the TCB field.
7	EDAC Enable (EB): When set, the EDAC is used otherwise it is bypassed during read and write operations.
6: 0	Test Check Bits (TCB) : Used as checkbits when the WB bit is set during writes and loaded with the check bits during a read operation when the RB bit is set.

Any unused most significant bits are reserved. Always read as ‘000...0’.

All fields except TCB are initialised at reset. The EDAC is initially disabled (EN = 0), which also applies to diagnostics fields (RB and WB are zero).

When available, the single error counter (SEC) field is cleared to zero.

14 Status Registers

14.1 Overview

The status registers store information about AMBA AHB accesses triggering an error response. There is a status register and a failing address register capturing the control and address signal values of a failing AMBA bus transaction, or the occurrence of a correctable error being signaled from a fault tolerant core.

14.2 Operation

14.2.1 Errors

The registers monitor AMBA AHB bus transactions and store the current HADDR, HWRITE, HMASTER and HSIZE internally. The monitoring are always active after startup and reset until an error response (HRESP = “01”) is detected. When the error is detected, the status and address register contents are frozen and the New Error (NE) bit is set to one. At the same time an interrupt is generated, as described hereunder.

Note that many of the fault tolerant units containing EDAC signal an un-correctable error as an AMBA error response, so that it can be detected by the processor as described above.

14.2.2 Correctable errors

Not only error responses on the AHB bus can be detected. Many of the fault tolerant units containing EDAC have a correctable error signal which is asserted each time a single error is detected. When such an error is detected, the effect will be the same as for an AHB error response. The only difference is that the Correctable Error (CE) bit in the status register is set to one when a single error is detected.

When the CE bit is set the interrupt routine can acquire the address containing the single error from the failing address register and correct it. When it is finished it resets the CE bit and the monitoring becomes active again. Interrupt handling is described in detail hereunder.

14.2.3 Interrupts

The interrupt is connected to the interrupt controller to inform the processor of the error condition. The normal procedure is that an interrupt routine handles the error with the aid of the information in the status registers. When it is finished it resets the NE bit and the monitoring becomes active again. Interrupts are generated for both AMBA error responses and correctable errors as described above.

14.3 Registers

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

Table 74. AHB Status registers

APB address offset	Registers
0x0	AHB Status register
0x4	AHB Failing address register

Table 75. AHB Status register

31 10	9	8	7	6 3	2 0
RESERVED	CE	NE	HWRITE	HMASTER	HSIZE

31: 10	RESERVED
9	CE: Correctable Error. Set if the detected error was caused by a single error and zero otherwise.
8	NE: New Error. Deasserted at start-up and after reset. Asserted when an error is detected. Reset by writing a zero to it.
7	The HWRITE signal of the AHB transaction that caused the error.
6: 3	The HMASTER signal of the AHB transaction that caused the error.
2: 0	The HSIZE signal of the AHB transaction that caused the error

Table 76. AHB Failing address register

31 0
AHB FAILING ADDRESS

31: 0	The HADDR signal of the AHB transaction that caused the error.

15 SpaceWire Interface

15.1 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 core is configured through a set of registers accessed through an APB interface. Data is transferred through DMA channels using an AHB master interface.

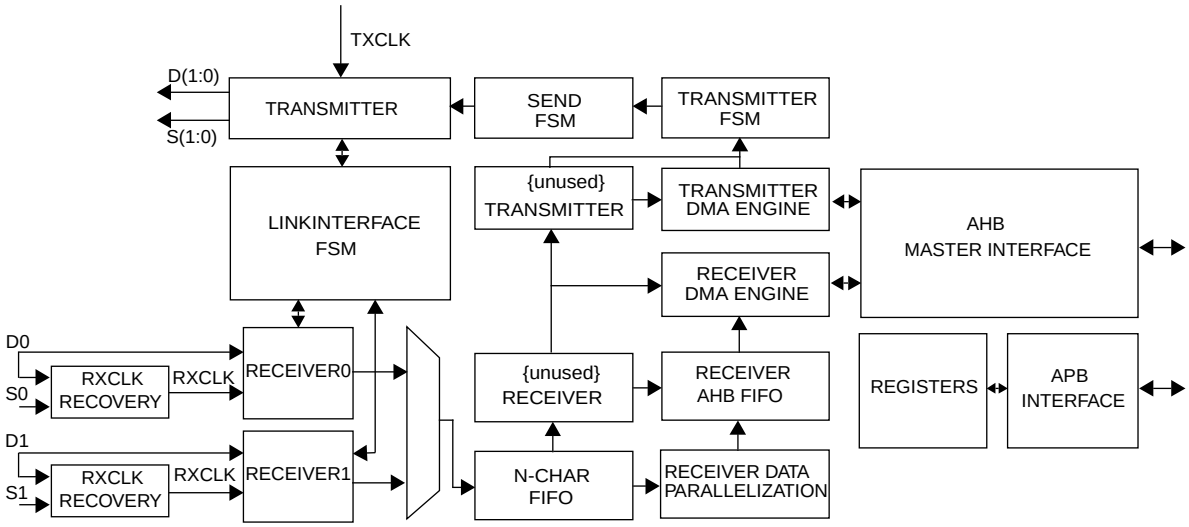


Figure 73. Block diagram

15.2 Operation

15.2.1 Overview

The main sub-blocks of the core are the link-interface and the AMBA interface. A block diagram of the internal structure can be found in figure 73.

The link interface consists of the receiver, transmitter and the link interface 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 N-Chars between the AMBA and SpaceWire domains during reception and transmission.

15.2.2 Protocol support

The core only accepts packets with a destination address corresponding to the one set in the node address register. Packets with address mismatch will be silently discarded (except in promiscuous mode which is covered in section 15.4.10). The node address register is initialized to the default address 254 during reset. It can then be changed to other values by writing to the register.

The core has support for the protocol ID specified in ECSS-E-ST-50-51C. Note that packets with the reserved extended protocol identifier (ID = 0x000000) are not ignored by the core. It is up to the client receiving the packets to ignore them.

When transmitting packets, the address and protocol-ID fields must be included in the buffers from where data is fetched. They are *not* automatically added by the core.

Figure 74 shows the packet types supported by the core. The core also allows reception and transmission with extended protocol identifiers but without support for RMAP CRC calculations.

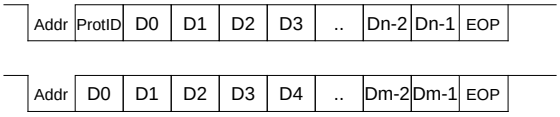


Figure 74. The SpaceWire packet types supported by the GRSPW.

15.3 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 73.

15.3.1 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 FSM in the host domain handles the exchange level.

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 autostart bit is set.

The current state of the link interface determines which type of characters are allowed to be transmitted which together with the requests made from the host interfaces determine what character will be 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 15.3.4).

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.

15.3.2 Transmitter

The state of the FSM, credit counters, requests from the time-interface and requests from the DMA-interface are used to decide 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 a separate clock-domain.

This is done because one usually wants to run the SpaceWire link on a different frequency than the host system clock. The core has a separate clock input which is used to generate the transmitter clock. Since the transmitter often runs on high frequency clocks (> 100 MHz) as much logic as possible has been placed in the system clock domain to minimize power consumption and timing issues.

The transmitter logic in the host clock domain decides what character to send next and sets the proper control signal and presents any needed character to the low-level transmitter as shown in figure 75. The transmitter sends 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 is handled in the transmitter. External LVDS drivers are needed for the data and strobe signals.

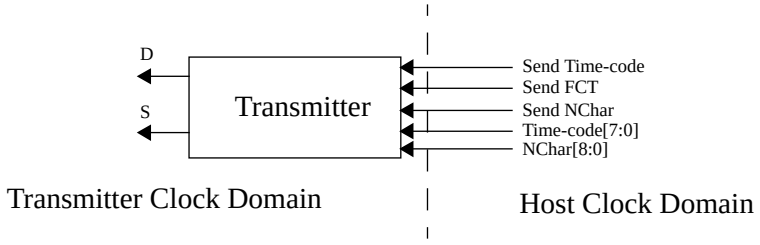


Figure 75. Schematic of the link interface transmitter.

A transmission 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 interface is notified and a new packet length value is given.

15.3.3 Receiver

The receiver detects connections from other nodes and receives characters as a bit stream on the data and strobe signals. It is also located in a separate 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 any characters. It 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 disconnected.

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 76. L-Chars are the 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.

There are no signals going directly from the transmitter clock domain to the receiver clock domain and vice versa. All the synchronization is done to the system clock.

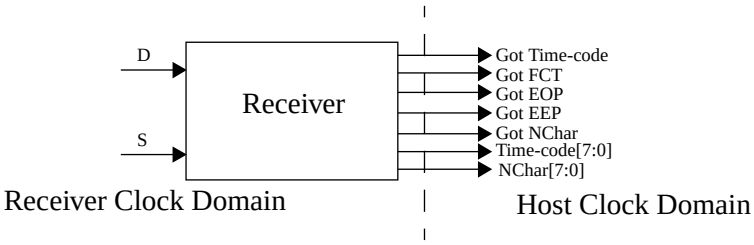


Figure 76. Schematic of the link interface receiver.

15.3.4 Time interface

The time interface is used for sending Time-codes over the SpaceWire network and consists of a time-counter register, time-ctrl register, tick-in signal, tick-out signal, tick-in register field and a tick-out register field. There are also two control register bits which enable the time receiver and transmitter respectively.

Each Time-code sent from the core is a concatenation of the time-ctrl and the time-counter register. There is a timetxen bit which 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 to the same time-ctrl and time-counter registers which are used for transmission. The timerxen bit in the control register is used for enabling time-code reception. No time-codes will be received if this bit is zero.

The two enable bits are used for ensuring that a node will not (accidentally) both transmit and receive time-codes which violates the SpaceWire standard. It 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 register is set to 0 after reset and is incremented each time the tick-in signal is asserted for one clock-period and the timetxen bit is set. This also causes the link interface to send the new value on the network. Tick-in can be generated either by writing a one to the register field or by asserting the tick-in signal. A Tick-in should not be generated too often since if the time-code after the previous Tick-in has not been sent the register will not be incremented and no new value will be sent. The tick-in field is automatically cleared when the value has been sent and thus no new ticks should be generated until this field is zero. If the tick-in signal is used there should be at least 4 system-clock and 25 transmit-clock cycles between each assertion.

A tick-out is generated each time a valid time-code is received and the timerxen bit is set. When the tick-out is generated the tick-out signal will be asserted one clock-cycle and the tick-out register field is asserted until it is cleared by writing a one to it.

The current time counter value can be read from the time register. It is updated each time a Time-code is received and the timerxen 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 register when a Time-code is received whose time-count is one more than the nodes current time-counter register. The time-ctrl register can be read through the APB interface. The same register is used during time-code transmissions.

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

15.4 Receiver DMA engine

The receiver DMA engine handles reception of data from the SpaceWire network to different DMA channels. Currently there is only one receive DMA channel available but the core has been written so that additional channels can be easily added if needed.

15.4.1 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 core 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.

15.4.2 Setting up the core for reception

A few registers need to be initialized before reception can take place. First the link interface need to be put in the run state before any data can be sent. The DMA channel has a maximum length register which sets the maximum size of packet that can be received to this channel. Larger packets are truncated and the excessive part is spilled. If this happens an indication will be given in the status field of the descriptor. The minimum value for the receiver maximum length field is 4 and the value can only be incremented in steps of four bytes. If the maximum length is set to zero the receiver will *not* function correctly.

The node address register needs to be set to hold the address of this SpaceWire node. Packets received with the incorrect address are discarded. Finally, the descriptor table and control register must be initialized. This will be described in the two following sections.

15.4.3 Setting up the descriptor table address

The core reads descriptors from an area in memory pointed to by the receiver descriptor table address register. The register consists of a base address and a descriptor selector. The base address points to the beginning of the area and must start on a 1 kbytes aligned address. It is also limited to be 1 kbytes in size which means the maximum number of descriptors is 128.

The descriptor selector points to individual descriptors and is increased by 1 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 a bit in the descriptors. The idea is that the selector should be initialized to 0 (start of the descriptor area) but it can also be written with another 8 bytes aligned value to start somewhere in the middle of the area. It will still wrap to the beginning of the area.

If one wants to use a new descriptor table the receiver enable bit has to be cleared first. When the rxactive bit 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.

15.4.4 Enabling descriptors

As mentioned earlier one or more descriptors must be enabled before reception can take place. Each descriptor is 8 byte in size and the layout can be found in the tables below. 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 after the previous one written to the area. Otherwise they will not be noticed.

A descriptor is enabled by setting the address pointer to point at a location where data can be stored and then setting the enable bit. The WR bit can be set to cause the selector to be set to zero when reception has finished to this descriptor. IE should be set if an interrupt is wanted when the reception has finished. The DMA control register interrupt enable bit 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-bit 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.

Table 77. GRSPW receive descriptor word 0 (address offset 0x0)

31	30	29	28	27	26	25	24 ... 0
TR	DC	HC	EP	IE	WR	EN	PACKETLENGTH

31	Truncated (TR) - Packet was truncated due to maximum length violation.
30	Data CRC (DC) - Unused. 1 if a CRC error was detected for the data and 0 otherwise.
29	Header CRC (HC) - Unused. 1 if a CRC error was detected for the header and 0 otherwise.
28	EEP termination (EP) - This packet ended with an Error End of Packet character.
27	Interrupt enable (IE) - If set, an interrupt will be generated when a packet has been received if the receive interrupt enable bit in the DMA channel control register is set.
26	Wrap (WR) - If set, the next descriptor used by the GRSPW will be the first one in the descriptor table (at the base address). Otherwise the descriptor pointer will be increased with 0x8 to use the descriptor at the next higher memory location. The descriptor table is limited to 1 kbytes in size and the pointer will be automatically wrap back to the base address when it reaches the 1 kbytes boundary.
25	Enable (EN) - Set to one to activate this descriptor. 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 (PACKETLENGTH) - The number of bytes received to this buffer. Only valid after EN has been set to 0 by the GRSPW.

Table 78. GRSPW receive descriptor word 1 (address offset 0x4)

31 ... 0
PACKETADDRESS

31: 0	Packet address (PACKETADDRESS) - The address pointing at the buffer which will be used to store the received packet.

15.4.5 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 rxen bit in the DMA control register (see section 15.8). This can be done anytime and before this bit is set nothing will happen. The rxdescav 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 core might not notice the new descriptors. More descriptors can be activated when reception has already started by enabling the descriptors and writing the rxdescav bit. When these bits are set reception will start immediately when data is arriving.

15.4.6 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 rxdescav 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 rxdescav bit is ‘0’ and the nospill bit is ‘0’ the packets will be discarded. If nospill is one the core waits until rxdescav is set.

When rxdescav is set the next descriptor is read and if enabled the packet is received to the buffer. If the read descriptor is not enabled, rxdescav is set to ‘0’ and the packet is spilled depending on the value of nospill.

The receiver can be disabled at any time and will cause all packets received afterwards to be discarded. If a packet is currently received when the receiver is disabled the reception will still be finished. The rxdescav 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. Rxdescav is also cleared by the core when it reads a disabled descriptor.

15.4.7 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 discarded (up to and including the next EOP/EEP). If the address matches the packet will or will not be received to the DMA channel depending on the conditions mentioned in the previous section. If received, the 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 the address matches the 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.

At least 2 non EOP/EEP N-Chars need to be received for a packet to be stored to the DMA channel. Packets smaller than the minimum size are discarded.

15.4.8 Status bits

When the reception of a packet is finished the enable bit in the current descriptor is set to zero. When enable is zero, the status bits are also valid and the number of received bytes is indicated in the length field. The DMA control register contains a status bit which is set each time a packet has been received. The core can also be made to generate an interrupt for this event as mentioned in section 15.4.4.

The RMAP CRC calculation is always active for all received packets and all bytes except the EOP/EEP are included. The packet is always assumed to be a RMAP packet and the length of the header is determined by checking byte 3 which should be the command field. The calculated CRC value is then checked when the header has been received (according to the calculated number of bytes) and if it is non-zero the HC bit is set indicating a header CRC error.

The CRC value is not set to zero after the header has been received, instead the calculation continues in the same way until the complete packet has been received. Then if the CRC value is non-zero the DC bit is set indicating a data CRC error. This means that the core can indicate a data CRC error even if the data field was correct when the header CRC was incorrect. However, the data should not be used when the header is corrupt and therefore the DC bit is unimportant in this case. When the header is not corrupted the CRC value will always be zero when the calculation continues with the data field and the behaviour will be as if the CRC calculation was restarted

If the received packet is not of RMAP type the header CRC error indication bit cannot be used. It is still possible to use the DC bit if the complete packet is covered by a CRC calculated using the RMAP CRC definition. This is because the core does not restart the calculation after the header has been received but instead calculates a complete CRC over the packet. Thus any packet format with one

CRC at the end of the packet calculated according to RMAP standard can be checked using the DC bit.

If the packet is neither of RMAP type nor of the type above with RMAP CRC at the end, then both the HC and DC bits should be ignored.

15.4.9 Error handling

If a packet reception needs to be aborted because of congestion on the network, the suggested solution is to set link disable 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 untransmitted 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 has stuck because of the transmitter not providing more 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 and then the currently active channel is disabled and the receiver enters the idle state. A bit in the channels control/status register is set to indicate this condition.

15.4.10 Promiscuous mode

The core supports a promiscuous 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 rxmaxlength register is still checked and packets exceeding this size will be truncated.

15.5 Transmitter DMA engine

The transmitter DMA engine handles transmission of data from the DMA channel to the SpaceWire network. There is one DMA channel available.

15.5.1 Basic functionality

The transmit DMA engine reads data from the AHB bus and stores them in 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 there are new descriptors enabled the core reads them and transfer the amount data indicated.

15.5.2 Setting up the core for transmission

Four steps need to be performed before transmissions can be done with the core. First the link interface must be enabled and started by writing the appropriate value to the ctrl register. Then the address to the descriptor table needs to be written to the transmitter descriptor table address register and one or more descriptors must also be enabled in the table. Finally, the txen bit in the DMA control register should be written with a one which triggers the transmission. These steps will be covered in more detail in the next sections.

15.5.3 Enabling descriptors

The descriptor table address register works in the same way as the receiver's corresponding register which was covered in section 15.4.

To transmit packets one or more descriptors have to be initialized in memory which is done in the following way: The number of bytes to be transmitted and a pointer to the data has to be set. 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 and if both are zero no packet is sent. The maximum header length is 255 bytes and the maximum data length is 16 Mbyte - 1. When the pointer and length fields have been set the enable 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 are 16 bytes in size so the maximum number in a single table is 64. The different fields of the descriptor together with the memory offsets are shown in the tables below.

The HC bit should be set if RMAP CRC should be calculated and inserted for the header field and correspondingly the DC bit should be set for the data field. The header CRC will be 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 NON-CRC bytes field is set to the number of bytes in the beginning of the header field that should not be included in the CRC calculation. The CRCs are sent even if the corresponding length is zero.

When both header and data length are zero no packet is sent not even an EOP.

15.5.4 Starting transmissions

When the descriptors have been initialized, the transmit enable bit in the DMA control register has to be set to tell the core to start transmitting. New descriptors can be activated in the table on the fly (while transmission is active). Each time a set of descriptors is added the transmit enable register bit should be set. This has to be done because each time the core encounters a disabled descriptor this register bit is set to 0.

Table 79. GRSPW transmit descriptor word 0 (address offset 0x0)

31 18	17	16	15	14	13	12	11 8	7 0
RESERVED	DC	HC	LE	IE	WR	EN	NONCRCLEN	HEADERLEN

31: 18	RESERVED
17	Append data CRC (DC) - Unused. 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	Append header CRC (HC) - Unused. 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 specified by the non-crc bytes field. The CRC will not be sent if the header length field is zero.
15	Link error (LE) - A Link error occurred during the transmission of this packet.
14	Interrupt enable (IE) - If set, an interrupt will be generated when the packet has been transmitted and the transmitter interrupt enable bit in the DMA control register is set.
13	Wrap (WR) - If set, the descriptor pointer will wrap and the next descriptor read will be the first one in the table (at the base address). Otherwise the pointer is increased with 0x10 to use the descriptor at the next higher memory location.

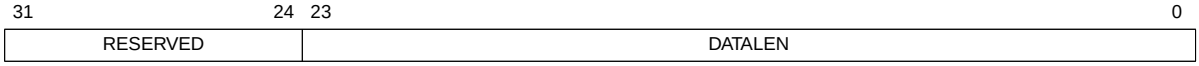
Table 79. GRSPW transmit descriptor word 0 (address offset 0x0)

12	Enable (EN) - 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 touched since this might corrupt the transmission. The GRSPW clears this bit when the transmission has finished.
11: 8	Non-CRC bytes (NONCRCLEN)- Unused. 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 (HEADERLEN) - Header Length in bytes. If set to zero, the header is skipped.

Table 80. GRSPW transmit descriptor word 1 (address offset 0x4)

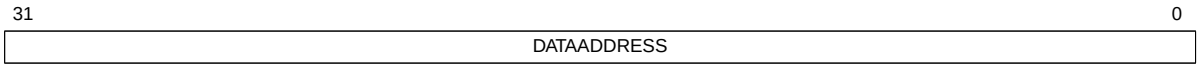
31: 0	Header address (HEADERADDRESS) - Address from where the packet header is fetched. Does not need to be word aligned.

Table 81. GRSPW transmit descriptor word 2 (address offset 0x8)



31: 24	RESERVED
23: 0	Data length (DATALEN) - Length of data part of packet. If set to zero, no data will be sent. If both data- and header-lengths are set to zero no packet will be sent.

Table 82. GRSPW transmit descriptor word 3(address offset 0xC)



31: 0	Data address (DATAADDRESS) - Address from where data is read. Does not need to be word aligned.

15.5.5 The transmission process

When the txen bit is set the core starts reading descriptors immediately. The number of bytes indicated are read and transmitted. When a transmission has finished, status will be 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 will also be generated. Then a new descriptor is read and if enabled a new transmission starts, otherwise the transmit enable bit is cleared and nothing will happen until it is enabled again.

15.5.6 The descriptor table address 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 1 kbytes 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.

15.5.7 Error handling

Abort Tx

The DMA control register contains a bit called Abort TX which if set causes the current transmission to 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. If the congestion is on the AHB bus this will not help (This should not be a problem since AHB slaves should have a maximum of 16 wait-states). The aborted packet will have its LE bit set in the descriptor. The transmit enable register bit is also cleared and no new transmissions will be done until the transmitter is enabled again.

AHB error

When an AHB error is encountered during transmission the currently active DMA channel is disabled and the transmitter goes to the idle mode. A bit in the DMA channel's control/status register is set to indicate this error condition and, if enabled, an interrupt will also be generated. Further error handling depends on what state the transmitter DMA engine was in when the AHB error occurred. If the descriptor was being read the packet transmission had not been started yet and no more actions need to be taken.

If the AHB error occurs during packet transmission the packet is truncated and an EEP is inserted. Lastly, if it occurs when status is written to the descriptor the packet has been successfully transmitted but the descriptor is not written and will continue to be enabled (this also means that no error bits are set in the descriptor for AHB errors).

The client using the channel has to correct the AHB error condition and enable the channel again. No more AHB transfers are done again from the same unit (receiver or transmitter) which was active during the AHB error until the error state is cleared and the unit is enabled again.

Link error

When a link error occurs during the transmission the remaining part of the packet is discarded up to and including the next EOP/EEP. When this is done status is immediately written (with the LE bit set) and the descriptor pointer is incremented. The link will be disconnected when the link error occurs but the core will automatically try to connect again provided that the link-start bit is asserted and the link-disabled bit is deasserted. If the LE bit in the DMA channel's control register is not set the transmitter DMA engine will wait for the link to enter run-state and start a new transmission immediately when possible if packets are pending. Otherwise the transmitter will be disabled when a link error occurs during the transmission of the current packet and no more packets will be transmitted until it is enabled again.

15.6 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. 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 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 which is half the fifo size in length. The last burst might be shorter. Byte accesses are used for non word-aligned buffers and/or packet lengths that are not a multiple of four bytes. There might be 1 to 3 single byte writes when writing the beginning and end of the received packets.

15.6.1 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.

15.6.2 AHB master interface

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

The AHB accesses are always word accesses (HSIZE = 0x010) of type incremental burst with unspecified length (HBURST = 0x001).

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 NONSEQ in this case while for incrementing accesses it is set to SEQ after the first access.

If the core does not need the bus after a burst has finished there will be one wasted cycle (HTRANS = IDLE).

BUSY transfer types are never requested and the core provides full support for ERROR, RETRY and SPLIT responses.

15.7 References

[SPW] ECSS - Space Engineering, SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C, 31 July 2008

[RMAPID] ECSS - Space Engineering, SpaceWire Protocols, ECSS-E-ST-50-51C, February 2010

[RMAP] ECSS - Space Engineering, SpaceWire Protocols, ECSS-E-ST-50-52C, February 2010

[OLD] Space Engineering, Remote Memory Access Protocol, Draft ECSS-E50-11, Draft F December 2006 *1) 2) 3)*

Note 1: The old ECSS-E50-11 Draft F document of the draft standard contains an informative VHDL description of the CRC implementation used in the RMAP protocol which is considered as incorrect with respect to the normative part of the draft standard. The VHDL descriptions in the ECSS-E50-11 Draft F document should therefore be ignored.

Note 2: The old ECSS-E50-11 Draft F document of the draft standard specifies that an all-zero CRC byte should be sent for RMAP write commands and read replies with null length data fields. This implementation does not support the transmission of an all-zero CRC byte for null length data fields. If this atypical functionality is required, the automatic CRC generation should be disabled for the packet in question.

Note 3: The old ECSS-E50-11 Draft F document of the draft standard specifies that error code 12 should be returned as a reply to RMAP commands with an unexpected destination address. This behavior is in conflict with the ECSS-E-ST-50-12C SpaceWire standard which specifies that packets with an unexpected destination address shall be discarded.

15.8 Registers

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

Table 83. GRSPW registers

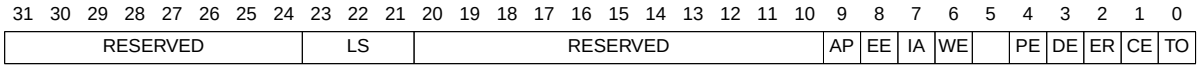
APB address offset	**Register**
0x0	Control
0x4	Status/Interrupt-source
0x8	Node address
0xC	Clock divisor
0x10	Destination key
0x14	Time
0x18	Timer and Disconnect
0x20	DMA channel 1 control/status
0x24	DMA channel 1 rx maximum length
0x28	DMA channel 1 transmit descriptor table address
0x2C	DMA channel 1 receive descriptor table address

Table 84. GRSPW control register

31	30	29	28 27 26 25 24 23 22	21	20	19 18	17	16	15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
RA	RX	RC	RESERVED	PS	NP		RD	RE	RESERVED	TR	TT	LI	TQ		RS	PM	TI	IE	AS	LS	LD

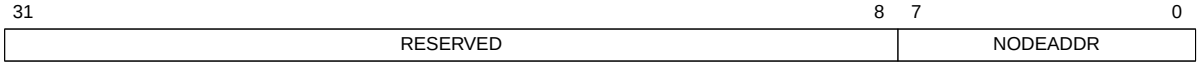
31	RMAP available (RA) - Set to one if the RMAP command handler is available. Only readable.
30	RX unaligned access (RX) - Set to one if unaligned writes are available for the receiver. Only readable.
29	RMAP CRC available (RC) - Set to one if RMAP CRC is enabled in the core. Only readable.
28: 27	RESERVED
26	Number of ports (PO) - The number of available SpaceWire ports minus one. Only readable.
25: 22	RESERVED
21	Port select (PS) - Selects the active port when the no port force bit is zero. ‘0’ selects the port connected to data and strobe on index 0 while ‘1’ selects index 1.
20	No port force (NP) - Disable port force. When disabled the port select bit cannot be used to select the active port. Instead, it is automatically selected by checking the activity on the respective receive links. Reset value: ‘0’.
19: 18	RESERVED
17	RMAP buffer disable (RD) - Unused.
16	RMAP Enable (RE) - Unused.
15: 12	RESERVED
11	Time Rx Enable (TR) - Enable time-code receptions. Reset value: ‘0’.
10	Time Tx Enable (TT) - Enable time-code transmissions. Reset value: ‘0’.
9	Link error IRQ (LI) - Generate interrupt when a link error occurs. Not reset.
8	Tick-out IRQ (TQ) - Generate interrupt when a valid time-code is received. Not reset.
7	RESERVED
6	Reset (RS) - Make complete reset of the SpaceWire node. Self clearing. Reset value: ‘0’.
5	Promiscuous Mode (PM) - Enable Promiscuous mode. Reset value: ‘0’.
4	Tick In (TI) - The host can generate a tick by writing a one to this field. This will increment the timer counter and the new value is transmitted after the current character is transferred. A tick can also be generated by asserting the tick_in signal (not supported). Reset value: ‘0’.
3	Interrupt Enable (IE) - If set, an interrupt is generated when one or both of bit 8 to 9 is set and its corresponding event occurs. Reset value: ‘0’.
2	Autostart (AS) - Automatically start the link when a NULL has been received. Not reset.
1	Link Start (LS) - Start the link, i.e. allow a transition from ready to started state. Reset value: ‘0’.
0	Link Disable (LD) - Disable the SpaceWire codec. Reset value: ‘0’.

Table 85. GRSPW status register



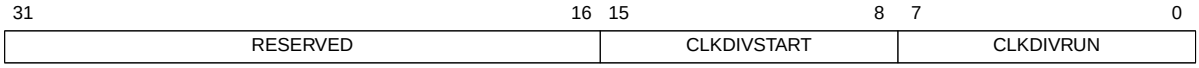
31: 24	RESERVED
23: 21	Link State (LS) - The current state of the start-up sequence. 0 = Error-reset, 1 = Error-wait, 2 = Ready, 3 = Started, 4 = Connecting, 5 = Run. Reset value: 0.
20: 10	RESERVED
9	Active port (AP) - Shows the currently active port. ‘0’ = Port 0 and ‘1’ = Port 1 where the port numbers refer to the index number of the data and strobe signals.
8	Early EOP/EEP (EE) - Set to one when a packet is received with an EOP after the first byte. Cleared when written with a one. Reset value: ‘0’.
7	Invalid Address (IA) - Set to one when a packet is received with an invalid destination address field, i.e it does not match the nodeaddr register. Cleared when written with a one. Reset value: ‘0’.
6	Write synchronization Error (WE) - A synchronization problem has occurred when receiving N-Chars. Cleared when written with a one. Reset value: ‘0’.
5	RESERVED
4	Parity Error (PE) - A parity error has occurred. Cleared when written with a one. Reset value: ‘0’.
3	Disconnect Error (DE) - A disconnection error has occurred. Cleared when written with a one. Reset value: ‘0’.
2	Escape Error (ER) - An escape error has occurred. Cleared when written with a one. Reset value: ‘0’.
1	Credit Error (CE) - A credit has occurred. Cleared when written with a one. Reset value: ‘0’.
0	Tick Out (TO) - A new time count value was received and is stored in the time counter field. Cleared when written with a one. Reset value: ‘0’.

Table 86. GRSPW node address register



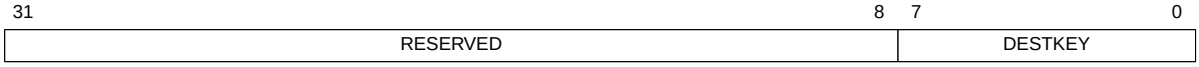
31: 8	RESERVED
7: 0	Node address (NODEADDR) - 8-bit node address used for node identification on the SpaceWire network. Reset value: 254.

Table 87. GRSPW clock divisor register



31: 16	RESERVED
15: 8	Clock divisor startup (CLKDIVSTART) - 8-bit Clock divisor value used for the clock-divider during startup (link-interface is in other states than run). The actual divisor value is Clock Divisor register + 1. Reset value taken from *gpio[5:3]* inputs, in the range 0 to 7 . “000” corresponds to *spw_clk* frequency divided by 1. “001” corresponds to the *spw_clk* frequency divided by 2, etc.
7: 0	Clock divisor run (CLKDIVRUN) - 8-bit Clock divisor value used for the clock-divider when the link-interface is in the run-state. The actual divisor value is Clock Divisor register + 1. Reset value taken from *gpio[5:3]* inputs, in the range 0 to 7 . “000” corresponds to *spw_clk* frequency divided by 1. “001” corresponds to the *spw_clk* frequency divided by 2, etc.

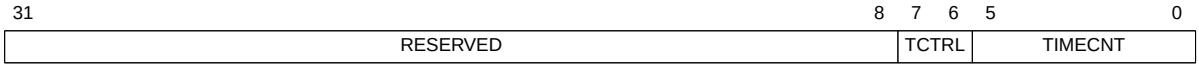
Table 88. GRSPW destination key



31: 8 RESERVED

7: 0 Destination key (DESTKEY) - RMAP destination key. Reset value: 0.

Table 89. GRSPW time register



31: 8 RESERVED

7: 6 Time control flags (TCTRL) - 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 register. Reset value: ‘0’.

5: 0 Time counter (TIMECNT) - The current value of the system time counter. It is incremented for each tick-in and the incremented value is transmitted. The register can also be written directly but the written value will not be transmitted. Received time-counter values are also stored in this register. Reset value: ‘0’.

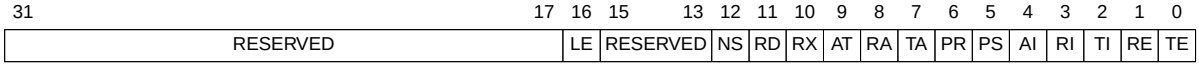
Table 90. GRSPW timer and disconnect register.

31: 22 RESERVED

21: 12 Disconnect (DISCONNECT) - Used to generate the 850 ns disconnect time period. The disconnect period is the number is the number of clock cycles in the disconnect register + 3. So to get a 850 ns period, the smallest number of clock cycles that is greater than or equal to 850 ns should be calculated and this values - 3 should be stored in the register. Reset value is set to correspond to 25 MHz system frequency.

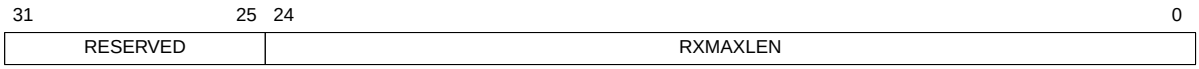
11: 0 6.4 us timer (TIMER64) - Used to generate the 6.4 and 12.8 us time periods. Should be set to the smallest number of clock cycles that is greater than or equal to 6.4 us. Reset value is set to correspond to 25 MHz system frequency.

Table 91. GRSPW DMA control register



31: 17	RESERVED
16	Link error disable (LE) - Disable transmitter when a link error occurs. No more packets will be transmitted until the transmitter is enabled again. Reset value: ‘0’.
15: 13	RESERVED
12	No spill (NS) - If cleared, packets will be discarded when a packet is arriving and there are no active descriptors. If set, the GRSPW will wait for a descriptor to be activated.
11	Rx descriptors available (RD) - Set to one, to indicate to the GRSPW that there are enabled descriptors in the descriptor table. Cleared by the GRSPW when it encounters a disabled descriptor: Reset value: ‘0’.
10	RX active (RX) - Is set to ‘1’ if a reception to the DMA channel is currently active otherwise it is ‘0’. Only readable.
9	Abort TX (AT) - Set to one to abort the currently transmitting packet and disable transmissions. If no transmission is active the only effect is to disable transmissions. Self clearing. Reset value: ‘0’.
8	RX AHB error (RA) - An error response was detected on the AHB bus while this receive DMA channel was accessing the bus. Cleared when written with a one. Reset value: ‘0’.
7	TX AHB error (TA) - An error response was detected on the AHB bus while this transmit DMA channel was accessing the bus. Cleared when written with a one. Reset value: ‘0’.
6	Packet received (PR) - This bit is set each time a packet has been received. never cleared by the SW-node. Cleared when written with a one. Reset value: ‘0’.
5	Packet sent (PS) - This bit is set each time a packet has been sent. Never cleared by the SW-node. Cleared when written with a one. Reset value: ‘0’.
4	AHB error interrupt (AI) - If set, an interrupt will be generated each time an AHB error occurs when this DMA channel is accessing the bus. Not reset.
3	Receive interrupt (RI) - If set, an interrupt will be generated each time a packet has been received. This happens both if the packet is terminated by an EEP or EOP. Not reset.
2	Transmit interrupt (TI) - If set, an interrupt will be generated each time a packet is transmitted. The interrupt is generated regardless of whether the transmission was successful or not. Not reset.
1	Receiver enable (RE) - Set to one when packets are allowed to be received to this channel. Reset value: ‘0’.
0	Transmitter enable (TE) - Write a one to this bit each time new descriptors are activated in the table. Writing a one will cause the SW-node to read a new descriptor and try to transmit the packet it points to. This bit is automatically cleared when the SW-node encounters a descriptor which is disabled. Reset value: ‘0’.

Table 92. GRSPW DMA RX maximum length register.



31: 25	RESERVED
24: 0	RX maximum length (RXMAXLEN) - Receiver packet maximum length in bytes. Only bits 24 - 2 are writable. Bits 1 - 0 are always 0. Not reset.

Table 93. GRSPW DMA transmitter descriptor table address register.

31 10	9 4	3 0
DESCBASEADDR	DESCSEL	RESERVED

31: 10 Descriptor table base address (DESCBASEADDR) - Sets the base address of the descriptor table. Not reset.

9: 4 Descriptor selector (DESCSEL) - Offset into the descriptor table. Shows which descriptor is currently used by the GRSPW. For each new descriptor read, the selector will increase with 16 and eventually wrap to zero again. Reset value: 0.

3: 0 RESERVED

Table 94. GRSPW DMA receiver descriptor table address register.

31 10	9 3	2 0
DESCBASEADDR	DESCSEL	RESERVED

31: 10 Descriptor table base address (DESCBASEADDR) - Sets the base address of the descriptor table. Not reset.

9: 3 Descriptor selector (DESCSEL) - Offset into the descriptor table. Shows which descriptor is currently used by the GRSPW. For each new descriptor read, the selector will increase with 8 and eventually wrap to zero again. Reset value: 0.

2: 0 RESERVED

15.9 Signal definitions and reset values

The signals and their reset values are described in table 95.

Table 95. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
spw_clk	Input	Transmitter default run-state clock	Rising edge	-
spw_rxd	Input, LVDS	Data input, positive	High	-
spw_rxdn	Input, LVDS	Data input, negative	Low	-
spw_rxs	Input, LVDS	Strobe input, positive	High	-
spw_rxsn	Input, LVDS	Strobe input, negative	Low	-
spw_txd	Output, LVDS	Data output, positive	High	Logical 0
spw_txdn	Output, LVDS	Data output, negative	Low	Logical 1
spw_txs	Output, LVDS	Strobe output, positive	High	Logical 0
spw_txsn	Output, LVDS	Strobe output, negative	Low	Logical 1

15.10 Timing

The timing waveforms and timing parameters are shown in figure 77 and are defined in table 96.

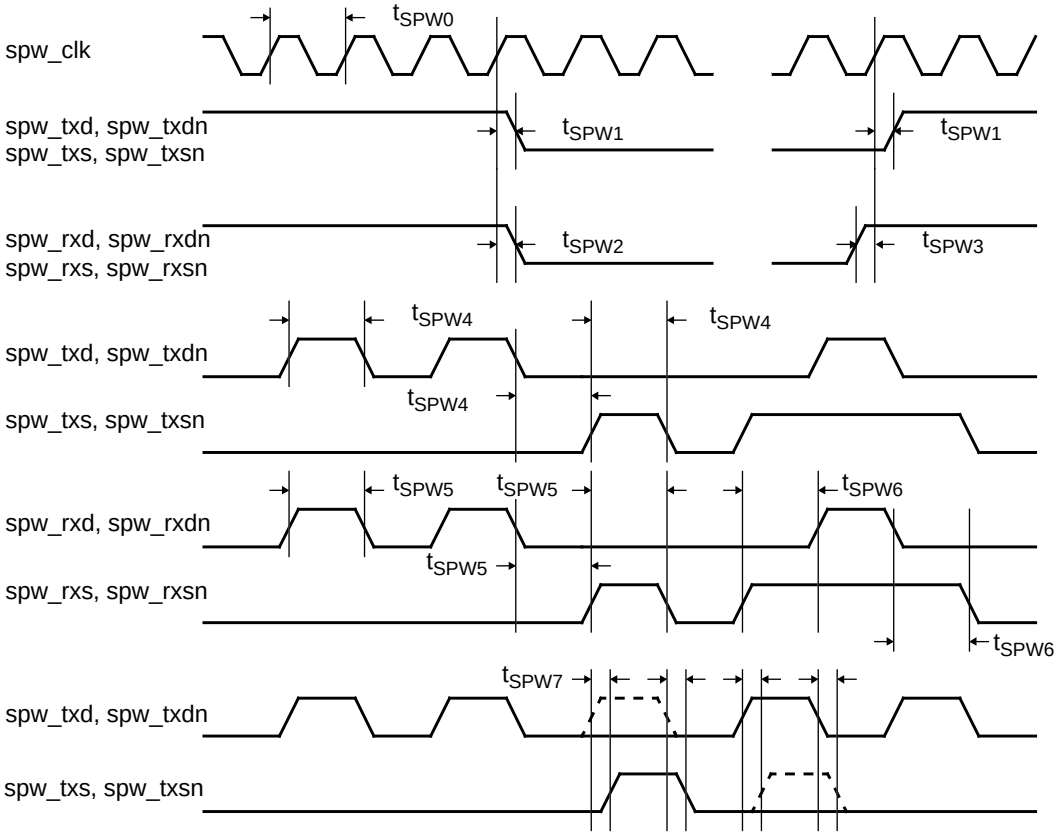


Figure 77. Timing waveforms

Table 96. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
t_{SPW0}	transmit clock period	-	20	-	ns
t_{SPW1}	clock to output delay	rising *spw_clk* edge	0	25	ns
t_{SPW2}	input to clock hold	-	-	-	not applicable
t_{SPW3}	input to clock setup	-	-	-	not applicable
t_{SPW4}	output data bit period	-	-		*clk* periods
		-	t_{SPW0} -5	t_{SPW0} +5	ns
t_{SPW5}	input data bit period	-	20	-	ns
t_{SPW6}	data & strobe edge separation	-	10	-	ns
t_{SPW7}	data & strobe output skew	-	-	5	ns

16 MIL-STD-1553B Bus Controller / Remote Terminal / Monitor Terminal

16.1 Overview

The interface provides a complete Mil-Std-1553B Bus Controller (BC), Remote Terminal (RT) or Monitor Terminal (MT). The interface connects to the MIL-STD-1553B bus through external transceivers and transformers. The interface is based on the Actel Core1553BRM core.

The interface consists of six main blocks: 1553 encoder, 1553B decoders, a protocol controller block, AMBA bus interface, command word legality interface, and a backend interface.

The interface can be configured to provide all three functions BC, RT and MT or any combination of the three. All variations use all six blocks except for the command legalization interface, which is only required on RT functions that implement RT legalization function externally. The device does not implement command legalization support.

A single 1553 encoder takes each word to be transmitted and serializes it using Manchester encoding. The encoder also includes independent logic to prevent the interface from transmitting for greater than the allowed period as well as loopback fail logic. The loopback logic monitors the received data and verifies that the interface has correctly received every word that it transmits. The output of the encoder is gated with the bus enable signals to select which buses the interface should be transmitting on. Two decoders take the serial Manchester received data from each bus and extract the received data words.

The decoder contains a digital phased lock loop (PLL) that generates a recovery clock used to sample the incoming serial data. The data is then de-serialized and the 16-bit word decoded. The decoder detects whether a command, status, or data word has been received, and checks that no Manchester encoding or parity errors occurred in the word.

The protocol controller block handles all the message sequencing and error recovery for all three operating modes, Bus Controller, Remote Terminal, and Bus Monitor. This is complex state machine that processes messages based on the message tables setup in memory, or reacts to incoming command words. The protocol controller implementation varies depending on which functions are implemented. The AMBA interface allows a system processor to access the control registers. It also allows the processor to directly access the memory connected to the backend interface, this simplifies the system design.

The interface comprises 33 16-bit registers. Of the 33 registers, 17 are used for control function and 16 for RT command legalization. Note that device does not implement command legalization support, all commands are always considered legal.

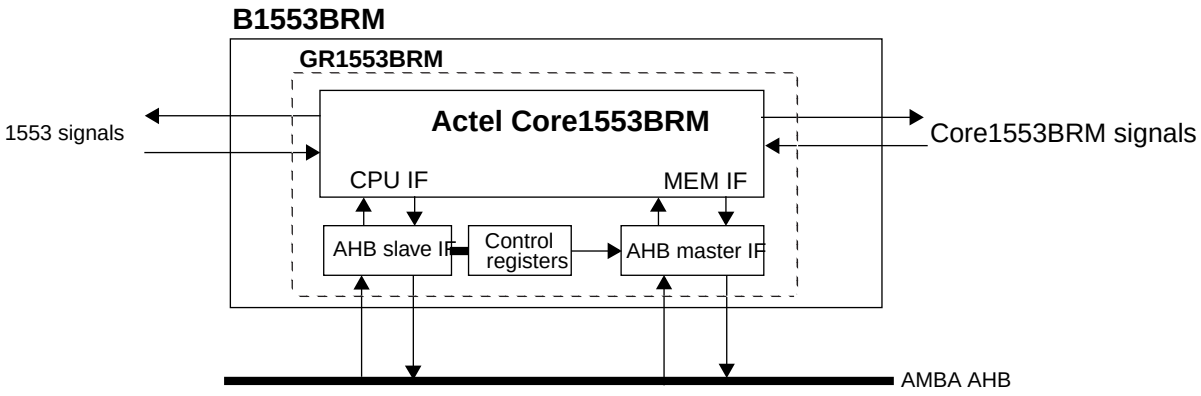


Figure 78. Block diagram

16.2 AHB interface

The amount of memory that the Mil-Std-1553B interface can address is 128 kBytes. The base address of this memory area must be aligned to a boundary of its own size and written into the AHB page address register.

The 16 bit address provided by the Core1553BRM core is shifted left one bit, and forms the AHB address together with the AHB page address register. Note that all pointers given to the Core1553BRM core needs to be right shifted one bit because of this.

When the Core1553BRM core has been granted access to the bus it expects to be able to do a series of uninterrupted accesses. To handle this requirement the AHB master locks the bus during these transfers. In the worst case, the Core1553BRM can do up to 7 writes in one such access and each write takes 2 plus the number of waitstate cycles with 4 idle cycles between each write strobe. This means care has to be taken if using two simultaneous active Core1553BRM cores on the same AHB bus.All AHB accesses are done as half word single transfers.

The AMBA AHB protection control signal is driven permanently with "0011" i.e a not cacheable, not bufferable, privileged data access. During all AHB accesses the AMBA AHB lock signal is driven with `1' and `0' otherwise.

16.3 Registers

The core is programmed through registers mapped into APB address space. The internal registers of Core1553BRM are mapped on the 33 lowest APB addresses. These addresses are 32-bit word aligned although only the lowest 16 bits are used. Refer to the Actel Core1553BRM MIL-STD-1553 BC, RT, and MT data sheet for detailed information.

Table 97. B1553BRM registers

APB address offset	Register
0x00 - 0x84	Core1553BRM registers
0x100	B1553BRM status/control
0x104	B1553BRM interrupt settings
0x108	AHB page address register

B1553BRM status/control register

Figure 79. B1553BRM status/control register

13: Bus reset indicator. If set a bus reset mode code has been received. Generates an interrupt when set.
12:5 Reserved
4: Address error. Shows the value of the rtaderr output from Core1553BRM.
3: Memory failure. Shows the value of the memfail output from Core1553BRM.
2: Busy. Shows the value of the busy output from Core1553BRM.
1: Active. Show the value of the active output from Core1553BRM.
0: Ssyfn. Connects directly to the ssyfn input of the Core1553BRM core. Resets to 1.

B1553BRM interrupt register

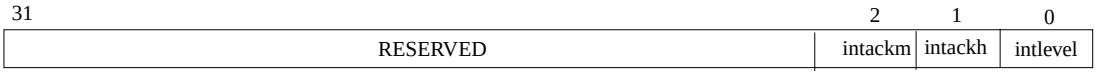


Figure 80. B1553RM interrupt register

2: Message interrupt acknowledge. Controls the intackm input signal of the Core1553BRM core.
1: Hardware interrupt acknowledge. Controls the intackh input signal of the Core1553BRM core.
0: Interrupt level. Controls the intlevel input signal of the Core1553BRM core.

AHB page address register

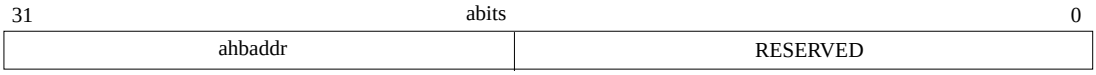


Figure 81. AHB page address register

[31:17]: Holds the top most bits of the AHB address of the allocated memory area.

16.4 Signal definitions and reset values

The signals and their reset values are described in table 98.

Table 98. Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
busainen	Output	Enable for the A receiver	High	Logical 0
busainp	Input	Positive data input from the A receiver	High	-
busainn	Input	Negative data input from the A receiver	Low	-
busbinen	Output	Enable for the B receiver	High	Logical 0
busbinp	Input	Positive data to the B receiver	High	-
busbinn	Input	Negative data to the B receiver	Low	-
busaoutin	Output	Inhibit for the A transmitter	High	Logical 0
busaoutp	Output	Positive data to the A transmitter	High	Logical 0
busaoutn	Output	Negative data to the A transmitter	Low	Logical 1
busboutin	Output	Inhibit for the B transmitter	High	Logical 0
busboutp	Output	Positive output to the B transmitter	High	Logical 0
busboutn	Output	Negative output to the B transmitter	Low	Logical 1

16.5 Timing

The timing waveforms and timing parameters are shown in figure 82 and are defined in table 99.

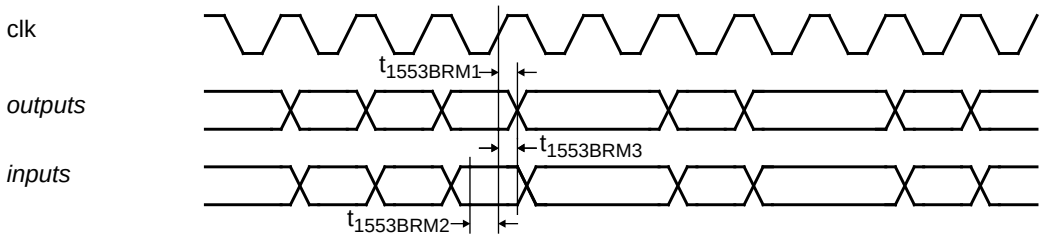


Figure 82. Timing waveforms

Table 99. Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{1553BRM0}$	clock period	-	41.66	41.66	ns
$t_{1553BRM1}$	clock to data output delay	rising *clk* edge	-	25	ns
$t_{1553BRM2}$	data input to clock setup	rising *clk* edge	7	-	ns
$t_{1553BRM3}$	data input from clock hold	rising *clk* edge	1	-	ns

17 MIL-STD-1553B Remote Terminal

17.1 Overview

The interface provides a complete Mil-Std-1553B Remote Terminal (RT). The interface connects to the MIL-STD-1553B bus through external transceivers and transformers. The interface is based on the Actel Core1553BRT core.

The interface provides a complete, dual-redundant MIL-STD-1553B remote terminal (RT) apart from the transceivers required to interface to the bus. At a high level, the interface simply provides a set of memory mapped sub-addresses that ‘receive data written to’ or ‘transmit data read from.’ The interface requires 2,048 words of memory, which can be shared with a local processor. The interface supports all 1553B mode codes and allows the user to designate as illegal any mode code or any particular sub-address for both transmit and receive operations. The command legalization can be done internally or via an external command legalization interface. Note that device does not implement command legalization support, all commands are always considered legal.

The interface consists of six main blocks: 1553B encoders, 1553B decoders, backend interface, command decoder, RT controller blocks and a command legalization block.

A single 1553B encoder is used for the interface. This takes each word to be transmitted and serializes it, after which the signal is Manchester encoded. The encoder also includes both logic to prevent the RT from transmitting for greater than the allowed period and loopback fail logic. The loopback logic monitors the received data and verifies that the interface has correctly received every word that it transmits. The output of the encoder is gated with the bus enable signals to select which buses the RT should use to transmit.

The interface includes two 1553B decoders. The decoder takes the serial Manchester data received from the bus and extracts the received data words. The decoder contains a digital phased lock loop (PLL) that generates a recovery clock used to sample the incoming serial data. The data is then deserialized and the 16-bit word decoded. The decoder detects whether a command or data word is received, and also performs Manchester encoding and parity error checking.

The command decoder and RT controller blocks decode the incoming command words, verifying the legality. Then the protocol state machine responds to the command, transmitting or receiving data or processing a mode code.

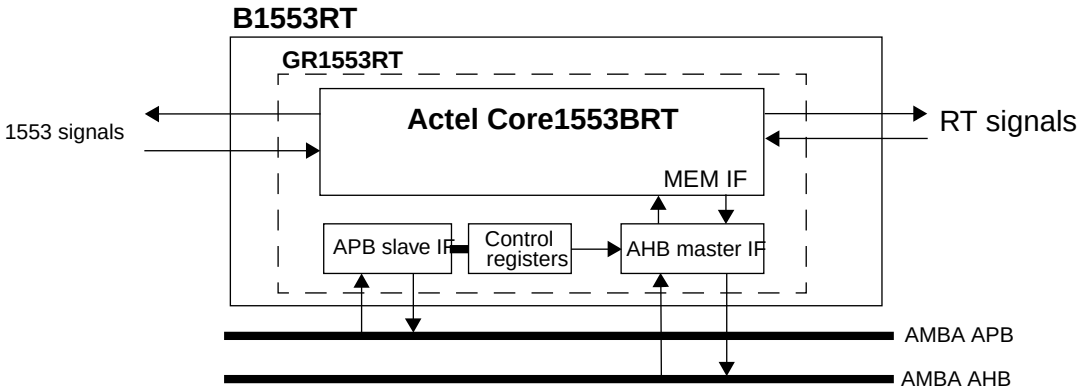


Figure 83. B1553RT block diagram

17.2 Operation

17.2.1 Reset behavior

After a external system reset, the B1553RT registers need to be initialized by software before the Remote Terminal can be accessed. For example, the RT address needs to be configured.

On an external system reset, access to the Actel Core1553BRT core is effectively disabled by lowering the receiver-enable signals going to the external transceiver. This is controlled by the *disabl* bit in the control register which is reset to '1'.

Receiving the 1553B Reset Remote Terminal mode command will cause the Actel Core1553BRT core to reset, but the B1553RT registers will keep their values and an interrupt will be generated as normally for any other transfer. The cores supports automatically setting the *disabl* bit when the Reset Remote Terminal mode command is received, preventing further commands being received before the core is re-initialized by software (by lowering the receiver-enable signals going to the external transceiver as described above). This is enabled by setting the *brdis* bit in the control register.

NOTE: the *disabl* and *brdis* bits are only implemented in newer versions of the LEON3FT-RTAX device, see the *revision* field in the status register to determine whether it is supported in a specific device.

NOTE: in older versions, not implementing the *disabl* and *brdis* bits, all the B1553RT registers will be reset by system reset and when receiving a Reset Remote Terminal mode command. The RT address will be reset to 0x1 with a valid parity and the core will be enabled for immediate transfers.

17.2.2 Memory map

The Core1553BRT core operates on a 2048*16 bit memory buffer, and therefore a 4 kilobyte memory area should be allocated. The memory is accessed via the AMBA AHB bus. The Core1553BRT uses only 11 address bits, and the top 20 address bits of the 32-bit AHB address can be programmed in the AHB page address register. The 11-bit address provided by the Core1553BRT core is left-shifted one bit, and forms the AHB address together with the AHB page address register. All AHB accesses are done as half word single transfers.

The used memory area has the following address map. Note that all 1553 data is 16 bit wide and will occupy two byte addresses. Every sub-address needs memory to hold up to 32 16 bit words.

*Table 100.*Memory map for 1553 data

Address	Content
0x000-0x03F	RX transfer status/command words
0x040-0x07F	Receive sub-address 1 ...
0x780-0x7BF	Receive sub-address 30
0x7C0-0x7FF	TX transfer status/command words
0x800-0x83F	Not used, except 0x800-0x801 for vector word if register bit *extmdata*=1
0x840-0x87F	Transfer sub-address 1 ...
0xF80-0xFBF	Transfer sub-address 30
0xFC0-0xFFF	Not used, except 0xFC0-0xFC1 for vector word if register bit *extmdata*=1

17.2.3 Data transfers

At the start of a bus transfer the core writes the 1553B command word (if the *wrtcmd* bit is set in the control register) to the address subaddress*2 for receive commands and 0x7C0 + subaddress*2 for transmit commands. After a bus transfer has completed a transfer status word is written to the same location as the command word (if *wrttsw* bit is set in the control register). The command word of the last transfer can always be read out through the interrupt vector and command value register.

The transfer status word written to memory has the following layout:

*Table 101.*Transfer Status Word layout

Bit	Name	Description
15	USED	Always set to 1 at the end of bus transfer
14	OKAY	Set to 1 if no errors were detected
13	BUSN	Set to 0 if transfer was on bus A, to 1 if bus B
12	BROADCAST	Transfer was a broadcast command
11	LPBKERRB	The loopback logic detected error on bus B
10	LPBKERRA	The loopback logic detected error on bus A
9	ILLCMD	Illegal command
8	MEMIFERR	DMA access error did not complete in time
7	MANERR	Manchester coding error detected
6	PARERR	Parity error detected
5	WCNTERR	Wrong number of words was received
4:0	COUNT	For sub address 1-30: number of words received/transmitted, 0 means 32
		For sub address 0 and 31: received/transmitted mode code

The AMBA AHB protection control signal is driven permanently with “0011”, i.e a not cacheable, not bufferable, privileged data access. The AMBA AHB lock signal is driven with ‘0’.

17.2.4 Mode commands

All mode codes defined by 1553B are legal except dynamic bus control (0), selected transmitter shutdown (20) and override selected transmitter shutdown (21).

Like data transfers, mode commands will write a command word before the transfer if the *wrtcmd* bit is set and a transmit status word after if the *wrttsw* bit is set. The transmit vector word and sync with data word will also store/fetch the data word from memory if the *extmdata* control bit is set. The locations are tabulated below. The default mapping for sync with data word with subaddress 0 places the data word and TSW at the same address 0, therefore a re-mapping has been implemented if *wrttsw* is set.

*Table 102.*Mode command memory map

Mode codes	Subaddress	CMD / TSW Location	Data word location (if extmdata=1)
1 synchronize 2 transmit status 3 initiate self-test 4 transmitter shutdown 5 override tx shutdown 6 inhibit TF 7 override inhibit TF 8 reset	0	0x7C0	(no data)
	31	0x7FE	(no data)
16 tx vector word	0	0x7C0	0x800
	31	0x7FE	0xFC0
17 synchronize with data	0	0x000	0x000 if *wrttsw*=0 0x7C0 if *wrttsw*=1
	31	0x03E	0x7C0
18 tx last command 19 tx BIT word	0	0x7C0	(internal)
	31	0x7FE	(internal)

NOTE: the remapping of the synchronized data word is only implemented in newer versions of the LEON3FT-RTAX device, see the *revision* field in the status register to determine whether the remapping is supported in a specific device.

The transfer BIT word mode code transfers a word as specified in the table below:

*Table 103.*Built In Test word

Bit	Name	Description
15	BUSINUSE	Set to 0 if transfer was on bus A, to 1 if bus B
14	LPBKERRB	The loopback logic detected error on bus B. Cleared by CLRERR.
13	LPBKERRA	The loopback logic detected error on bus A. Cleared by CLRERR.
12	SHUTDOWNB	Indicates that bus B has been shutdown
11	SHUTDOWNA	Indicates that bus A has been shutdown
10	TFLAGINH	Terminal flag inhibit setting
9	WCNTERR	Word count error has occurred. Cleared by CLRERR.
8	MANERR	Manchester coding error detected. Cleared by CLRERR.
7	PARERR	Parity error detected. Cleared by CLRERR.
6	RTRTTO	RT to RT transfer timeout. Cleared by CLRERR.
5	MEMFAIL	DMA transfer not completed in time. Cleared by CLRERR.
4:0	VERSION	Core1553RT version

17.3 Registers

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

*Table 104.*B1553RT registers

APB Address offset	Register
0x00	Status
0x04	Control
0x08	Vector word
0x0C	Interrupt vector and command value
0x10	AHB page address register
0x14	Interrupt pending/mask register

Status register (read only)

Figure 84. Status register

31:28 Core revision, read-only field:
- 0001: Added *brdis*, *disabl*, *extleg* fields.
  RT address is reset to 0x0 with an invalid parity after system reset.
  RT address is not reset on Reset Remote Terminal mode command reception.
  Remapped sync data word if *extmdata* and *wrttsw* are set.
- 0000: Initial revision, known issues:
  RT address is reset to 0x1 with a valid parity, and the core is enabled for transmission after system reset and on Reset Remote Terminal mode command reception.
  Reset Remote Terminal mode command resets all control registers.
  Sync data word remapping is not implemented.

27:4 Reserved

3: Reads ‘1’ if configured with external legalization interface, ‘0’ if all supported accesses are legal. Fixed to ‘0’ in this design.

2: RT address error. Incorrect RT address parity bit. Connected to the rtaderr output pin on Core1553BRT.

1: Memory failure. DMA transfer did not complete in time. Cleared using *clrerr* bit in control register. Shows the value of the memfail output from Core1553BRT.

0: Busy. Indicates that the RT is busy with a transfer. Shows the value of the busy output from Core1553BRT.

Control register

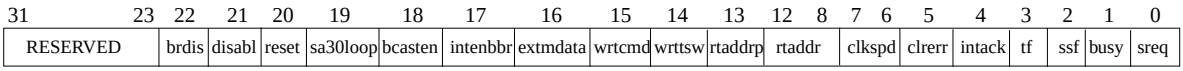


Figure 85. Control register

31:23 Reserved
22: If ‘1’ the disable bit (bit 21) will be set to ‘1’ automatically when a reset mode command is received.
21: Set to ‘1’ to disable external 1553 transceiver. Reset to ‘1’.
20: Writing ‘1’ will reset the Core1553RT and forces the B1553RT DMA to idle state. Self clearing.
19: Set to ‘1’ to enable internal loopback of subaddress 30. Transmits from sa 30 reads from the receive buffer for sa 30.
18: Set to ‘1’ to enable broadcasts messages. If ‘0’ address 31 is treated as normal RT address.
17: ‘1’ enables interrupts for bad messages. If ‘0’ only good messages generates interrupts.
16: If ‘1’ mode code data is written to / read from memory. If ‘0’ the vword register is used for transmit vector word mode code and the data for synchronize with data is discarded.
15: If ‘1’ the command word is written to memory at the start of a bus transfer.
14: If ‘1’ the transfer status word is written to memory at the end of a bus transfer.
13: RT address parity bit. Odd parity over rtaddr and rtaddrp must be achieved.
12:8 RT address.
7:6 Clock speed. Should be set to indicate the clock frequency of the core. 0 - 12, 1 - 16, 2 - 20, 3 - 24 MHz
5: Set to ‘1’ and then to ‘0’ to clear internal errors.
4: Clear the interrupt. Should be set to ‘1’ to give a interrupt pulse on each message.
3: Controls the terminal flag bit in the 1553B status word. This can be masked by the "inhibit terminal flag bit" mode code.
2: Controls the subsystem flag bit in the 1553B status word.
1: Controls the busy bit in the 1553B status word.
0: Controls the service request bit in the 1553B status word.

Vector word register

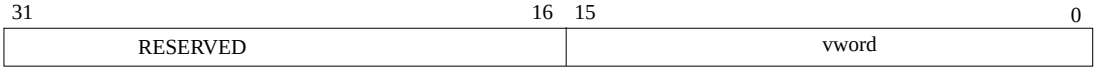


Figure 86. Vector word register

[15:0] Used for transmit vector word mode code if extmdata bit is ‘0’ in control register.

Interrupt vector and command value register

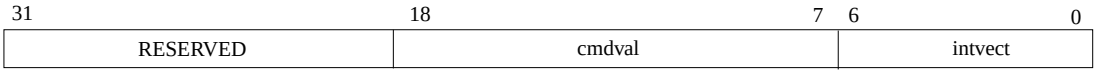


Figure 87. Interrupt vector register

[18:7] For each message the CMDVAL output of Core1553BRT is latched into this register.
18 - broadcast
17 - 1 for transmit, 0 for receive
16:12 - subaddress
11:7 - word count / mode code
[6:0] Shows the value of the interrupt vector output of the Core1553BRT.

AHB page address register

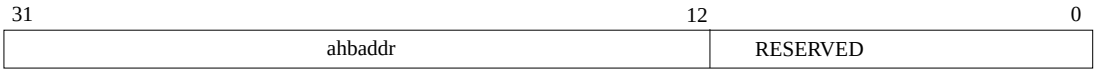


Figure 88. Address register

[31:12]: Holds the 20 top most bits of the AHB address of the allocated memory area.

Interrupt pending/mask register

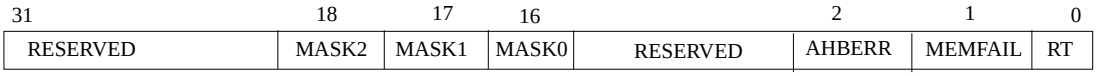


Figure 89. Address register

[31:19]: Reserved
18: MASK2 - Interrupt mask for AHBERR interrupt. Interrupt enabled if 1.
17: MASK1 - Interrupt mask for MEMFAIL interrupt. Interrupt enabled if 1.
16: MASK0 - Interrupt mask for RT interrupt. Interrupt enabled if 1.
[15:3]: Reserved
2 : AHBERR - 1 if an AHB error has occurred. Cleared on read.
1: MEMFAIL - 1 if an Core1553RT DMA has not occurred in time. Cleared on read.
0: RT - 1 if the Core1553RT has received/transmitted a message. Cleared on read.

17.4 Signal definitions and reset values

The signals and their reset values are described in table 105.

*Table 105.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
busainen	Output	Enable for the A receiver	High	Logical 0
busainp	Input	Positive data input from the A receiver	High	-
busainn	Input	Negative data input from the A receiver	Low	-
busbinen	Output	Enable for the B receiver	High	Logical 0
busbinp	Input	Positive data to the B receiver	High	-
busbinn	Input	Negative data to the B receiver	Low	-
busaoutin	Output	Inhibit for the A transmitter	High	Logical 0
busaoutp	Output	Positive data to the A transmitter	High	Logical 0
busaoutn	Output	Negative data to the A transmitter	Low	Logical 1
busboutin	Output	Inhibit for the B transmitter	High	Logical 0
busboutp	Output	Positive output to the B transmitter	High	Logical 0
busboutn	Output	Negative output to the B transmitter	Low	Logical 1

17.5 Timing

The timing waveforms and timing parameters are shown in figure 90 and are defined in table 106.

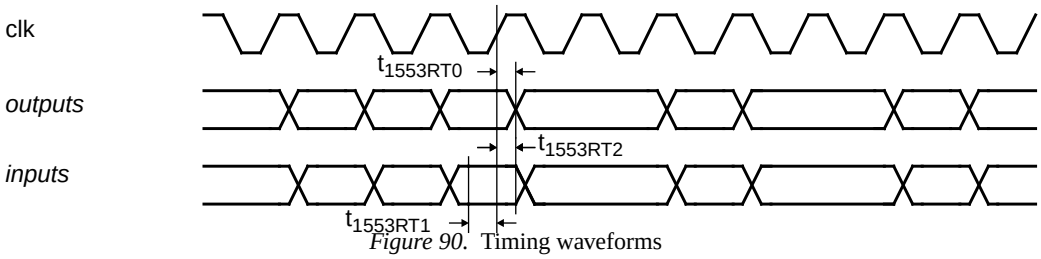


Figure 90. Timing waveforms

*Table 106.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{1553RT00}$	clk clock period	-	41.66	41.66	ns
$t_{1553RT0}$	clock to data output delay	rising *clk* edge	-	25	ns
$t_{1553RT1}$	data input to clock setup	rising *clk* edge	7	-	ns
$t_{1553RT2}$	data input from clock hold	rising *clk* edge	1	-	ns

18 CAN 2.0 Interface

18.1 Overview

This CAN interface implements the CAN 20.A and 2.0B protocols. It is based on the Philips SJA1000 and has a compatible register map with a few exceptions.

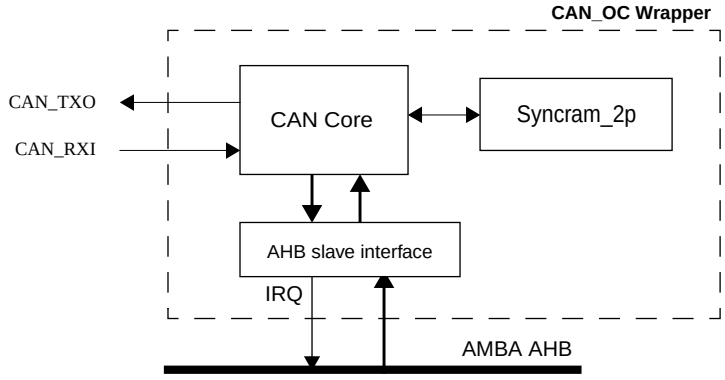


Figure 91. Block diagram

18.2 Opencores CAN controller overview

This CAN controller is based on the Philips SJA1000 and has a compatible register map with a few exceptions. It also supports both BasicCAN (PCA82C200 like) and PeliCAN mode. In PeliCAN mode the extended features of CAN 2.0B is supported. The mode of operation is chosen through the Clock Divider register.

This document will list the registers and their functionality. The Philips SJA1000 data sheet can be used as a reference if something needs clarification. See also the Design considerations chapter for differences between this core and the SJA1000.

The register map and functionality is different between the two modes of operation. First the Basic-CAN mode will be described followed by PeliCAN. Common registers (clock divisor and bus timing) are described in a separate chapter. The register map also differs depending on whether the core is in operating mode or in reset mode. When reset the core starts in reset mode awaiting configuration. Operating mode is entered by clearing the reset request bit in the command register. To re-enter reset mode set this bit high again.

18.3 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 wrapper is big endian so the core expects the MSB at the lowest address.

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

18.4 BasicCAN mode

18.4.1 BasicCAN register map

*Table 107.*BasicCAN address allocation

Address	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

18.4.2 Control register

The control register contains interrupt enable bits as well as the reset request bit.

*Table 108.*Bit interpretation of control register (CR) (address 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.

18.4.3 Command register

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

*Table 109.*Bit interpretation of command register (CMR) (address 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 not yet started transmission.
CMR.0	Transmission request	Starts the transfer of the message in the TX buffer

A transmission is started by writing 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.

Giving the Release receive buffer command should be done after reading the contents of the receive buffer in order to release this memory. If there is another message waiting in the FIFO a new receive interrupt will be generated (if enabled) and the receive buffer status bit will be set again.

To clear the Data overrun status bit CMR.3 must be written with 1.

18.4.4 Status register

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

*Table 110.*Bit interpretation of status register (SR) (address 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 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 no space in fifo.
SR.0	Receive buffer status	1 if messages available in the receive fifo.

Receive buffer status is cleared when the Release receive buffer command is given and set high if there are more messages available in the fifo.

The data overrun status signals that a message which was accepted could not be placed in the fifo because 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 is high the transmit buffer is available to be written into by the CPU. During an on-going transmission the buffer is locked and this bit is 0.

The transmission complete bit is set to 0 when a transmission request has been issued and will not be set to 1 again until a message has successfully been transmitted.

18.4.5 Interrupt register

The interrupt register signals to CPU what caused the interrupt. The interrupt bits are only set if the corresponding interrupt enable bit is set in the control register.

*Table 111.*Bit interpretation of interrupt register (IR) (address 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 SR.1 goes from 0 to 1.
IR.2	Error interrupt	Set when the error status or bus status are changed.
IR.1	Transmit interrupt	Set when the transmit buffer is released (status bit 0->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. Note that this differs from the SJA1000 behavior where all bits are reset on read in BasicCAN mode. This core resets the receive interrupt bit when the release receive buffer command is given (like in PeliCAN mode).

Also note that bit IR.5 through IR.7 reads as 1 but IR.4 is 0.

18.4.6 Transmit buffer

The table below shows the layout of the transmit buffer. In BasicCAN only standard frame messages can be transmitted and received (EFF messages on the bus are ignored).

*Table 112.*Transmit buffer layout

Addr	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. Note that it is possible to specify a DLC larger than 8 bytes but should not be done for compatibility reasons. If DLC > 8 still only 8 bytes can be sent.

18.4.7 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.

18.4.8 Acceptance filter

Messages can be filtered based on their identifiers using the acceptance code and acceptance mask registers. The top 8 bits of the 11 bit identifier are compared with the acceptance code register only comparing the bits set to zero in the acceptance mask register. If a match is detected the message is stored to the fifo.

18.5 PeliCAN mode

18.5.1 PeliCAN register map

*Table 113.*PeliCAN address allocation

#	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	reserved (0x00)		-		reserved (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	reserved (0x00)	-
25	RX data 7	RX data 5	TX data 7	TX data 5	reserved (0x00)	-
26	RX data 8	RX data 6	TX data 8	TX data 6	reserved (0x00)	-
27	FIFO	RX data 7	-	TX data 7	reserved (0x00)	-
28	FIFO	RX data 8	-	TX data 8	reserved (0x00)	-
29	RX message counter		-		RX msg counter	-
30	(0x00)		-		(0x00)	-
31	Clock divider		Clock divider		Clock divider	Clock divider

The transmit and receive buffers have different layout depending on if standard frame format (SFF) or extended frame format (EFF) is to be transmitted/received. See the specific section below.

18.5.2 Mode register

*Table 114.*Bit interpretation of mode register (MOD) (address 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	If set the controller is in self test mode
MOD.1	Listen only mode	If set 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 entered previously.

In Listen only mode the core will not send any acknowledgements. Note that unlike the SJA1000 the Opencores core does not become error passive and active error frames are still sent!

When in Self test mode the core can complete a successful transmission without getting an acknowledgement if given the Self reception request command. Note that the core must still be connected to a real bus, it does not do an internal loopback.

18.5.3 Command register

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

*Table 115.*Bit interpretation of command register (CMR) (address 1)

Bit	Name	Description
CMR.7	-	reserved
CMR.6	-	reserved
CMR.5	-	reserved
CMR.4	Self reception request	Transmits and simultaneously receives a message
CMR.3	Clear data overrun	Clears the data overrun status bit
CMR.2	Release receive buffer	Free the current receive buffer for new reception
CMR.1	Abort transmission	Aborts a not yet started transmission.
CMR.0	Transmission request	Starts the transfer of the message in the TX buffer

A transmission is started by writing 1 to CMR.0. It can only be aborted by writing 1 to 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 not successful the first time.

Giving the Release receive buffer command should be done after reading the contents of the receive buffer in order to release this memory. If there is another message waiting in the FIFO a new receive interrupt will be generated (if enabled) and the receive buffer status bit will be set again.

The Self reception request bit 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 interrupt will be generated.

18.5.4 Status register

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

*Table 116.*Bit interpretation of command register (SR) (address 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 is cleared when there are no more messages in the fifo. The data overrun status signals that a message which was accepted could not be placed in the fifo because 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 is high the transmit buffer is available to be written into by the CPU. During an on-going transmission the buffer is locked and this bit is 0.

The transmission complete bit is set to 0 when a transmission request or self reception request has been issued and will not be set to 1 again until a message has successfully been transmitted.

18.5.5 Interrupt register

The interrupt register signals to CPU what caused the interrupt. The interrupt bits are only set if the corresponding interrupt enable bit is set in the interrupt enable register.

*Table 117.*Bit interpretation of interrupt register (IR) (address 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 fifo is not empty.

This register is reset on read with the exception of IR.0 which is reset when the fifo has been emptied.

18.5.6 Interrupt enable register

In the interrupt enable register the separate interrupt sources can be enabled/disabled. If enabled the corresponding bit in the interrupt register can be set and an interrupt generated.

*Table 118.*Bit interpretation of interrupt enable register (IER) (address 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

18.5.7 Arbitration lost capture register

*Table 119.*Bit interpretation of arbitration lost capture register (ALC) (address 11)

Bit	Name	Description
ALC.7-5	-	reserved
ALC.4-0	Bit number	Bit where arbitration is 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.

18.5.8 Error code capture register

*Table 120.*Bit interpretation of error code capture register (ECC) (address 12)

Bit	Name	Description
ECC.7-6	Error code	Error code number
ECC.5	Direction	1 - Reception, 0 - transmission error
ECC.4-0	Segment	Where in the frame the error occurred

When a bus error occurs the error code capture register is set according to what kind of error occurred, if it was while transmitting or receiving and where in the frame it happened. As with the ALC register the ECC register will not change value until it has been read out. The table below shows how to interpret bit 7-6 of ECC.

*Table 121.*Error code interpretation

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

Bit 4 downto 0 of the ECC register is interpreted as below

*Table 122.*Bit interpretation of ECC.4-0

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

18.5.9 Error warning limit register

This registers allows for setting the CPU error warning limit. It defaults to 96. Note that this register is only writable in reset mode.

18.5.10 RX error counter register (address 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.

18.5.11 TX error counter register (address 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 initialized as to count down the protocol defined 128 occurrences of the bus-free signal and 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. Note that unlike the SJA1000 this core will signal bus-off

immediately and not first when entering operating mode. The bus-off recovery sequence starts when entering operating mode after writing 255 to this register in reset mode.

18.5.12 Transmit buffer

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

Table 123.

#	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 data 1	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
27	-	TX data 7
28	-	TX data 8

TX frame information (this field has the same layout for both SFF and EFF frames)

*Table 124.*TX frame information address 16

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF	RTR	-	-	DLC.3	DLC.2	DLC.1	DLC.0

Bit 7 - FF selects the frame format, i.e. whether this is to be interpreted as an extended or standard frame. 1 = EFF, 0 = SFF.
Bit 6 - RTR should be set to 1 for an remote transmission request frame.
Bit 5:4 - are don’t care.
Bit 3:0 - DLC specifies the Data Length Code and should be a value between 0 and 8. If a value greater than 8 is used 8 bytes will be transmitted.

TX identifier 1 (this field is the same for both SFF and EFF frames)

*Table 125.*TX identifier 1 address 17

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Bit 7:0 - The top eight bits of the identifier.

TX identifier 2, SFF frame

*Table 126.*TX identifier 2 address 18

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	-	-	-	-	-

Bit 7:5 - Bottom three bits of an SFF identifier.
Bit 4:0 - Don’t care.

TX identifier 2, EFF frame

*Table 127.*TX identifier 2 address 18

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Bit 7:0 - Bit 20 downto 13 of 29 bit EFF identifier.

TX identifier 3, EFF frame

*Table 128.*TX identifier 3 address 19

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Bit 7:0 - Bit 12 downto 5 of 29 bit EFF identifier.

TX identifier 4, EFF frame

*Table 129.*TX identifier 4 address 20

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.4	ID.3	ID.2	ID.1	ID.0	-	-	-

Bit 7:3 - Bit 4 downto 0 of 29 bit EFF identifier
Bit 2:0 - Don’t care

Data field

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

18.5.13 Receive buffer

Table 130.

#	Read (SFF)	Read (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
28	RX ID1 of next message in fifo	RX data 8

RX frame information (this field has the same layout for both SFF and EFF frames)

*Table 131.*RX frame information address 16

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0

Bit 7 - Frame format of received message. 1 = EFF, 0 = SFF.
Bit 6 - 1 if RTR frame.
Bit 5:4 - Always 0.
Bit 3:0 - DLC specifies the Data Length Code.

RX identifier 1(this field is the same for both SFF and EFF frames)

*Table 132.*RX identifier 1 address 17

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Bit 7:0 - The top eight bits of the identifier.

RX identifier 2, SFF frame

*Table 133.*RX identifier 2 address 18

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	RTR	0	0	0	0

Bit 7:5 - Bottom three bits of an SFF identifier.
Bit 4 - 1 if RTR frame.
Bit 3:0 - Always 0.

RX identifier 2, EFF frame

*Table 134.*RX identifier 2 address 18

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Bit 7:0 - Bit 20 downto 13 of 29 bit EFF identifier.

RX identifier 3, EFF frame

*Table 135.*RX identifier 3 address 19

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Bit 7:0 - Bit 12 downto 5 of 29 bit EFF identifier.

RX identifier 4, EFF frame

*Table 136.*RX identifier 4 address 20

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.4	ID.3	ID.2	ID.1	ID.0	RTR	0	0

Bit 7:3 - Bit 4 downto 0 of 29 bit EFF identifier
Bit 2- 1 if RTR frame
Bit 1:0 - Don’t care

Data field

For received SFF frames the data field is located at address 19 to 26 and for EFF frames at 21 to 28.

18.5.14 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 deal with it.

There are two different filtering modes, single and dual filter. Which one is used is controlled by bit 3 in the mode register. In single filter mode only one 4 byte filter is used. In dual filter two smaller filters are used and if either of these signals a match the message is accepted. Each filter consists of two parts the acceptance code and the acceptance mask. The code registers are used for specifying the pattern to match and the mask registers specify don’t care bits. In total eight registers are used for the acceptance filter as shown in the table below. Note that they are only read/writable in reset mode.

*Table 137.*Acceptance filter registers

Address	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)
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 selects if the results of the comparison doesn’t matter. 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 selects if the results of the comparison doesn’t matter. 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 selects if the results of the comparison doesn’t matter. 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 selects if the results of the comparison doesn't matter. A set bit in the mask register means don't care.

18.5.15 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.

18.6 Common registers

There are three common registers with the same addresses and the same functionality in both Basi-CAN and PeliCAN mode. These are the clock divider register and bus timing register 0 and 1.

18.6.1 Clock divider register

The only real function of this register in the GRLIB version of the Opencores CAN is to choose between PeliCAN and BasiCAN. The clkout output of the Opencore CAN core is not connected and it is its frequency that can be controlled with this register.

*Table 138.*Bit interpretation of clock divider register (CDR) (address 31)

Bit	Name	Description
CDR.7	CAN mode	1 - PeliCAN, 0 - BasiCAN
CDR.6	-	unused (cbp bit of SJA1000)
CDR.5	-	unused (rxinten bit of SJA1000)
CDR.4	-	reserved
CDR.3	Clock off	Disable the clkout output
CDR.2-0	Clock divisor	Frequency selector

18.6.2 Bus timing 0

*Table 139.*Bit interpretation of bus timing 0 register (BTR0) (address 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.

18.6.3 Bus timing 1

*Table 140.*Bit interpretation of bus timing 1 register (BTR1) (address 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.

18.7 Design considerations

This section lists known differences between this CAN controller and 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
- 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 SJA1000

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

18.8 Signal definitions and reset values

The signals and their reset values are described in table 141.

*Table 141.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
cantx[]	Output	CAN transmit data	Low	Logical 1
canen[]	Output	CAN transmit enable	-	Logical 0
canrx[]	Input	CAN receive data	Low	-

18.9 Timing

The timing waveforms and timing parameters are shown in figure 92 and are defined in table 142.

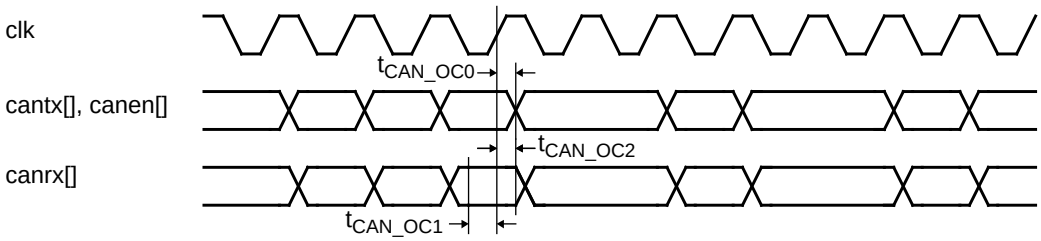


Figure 92. Timing waveforms

*Table 142.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
t_{CAN_OC0}	clock to data output delay	rising *clk* edge	0	25	ns
t_{CAN_OC1}	data input to clock setup	rising *clk* edge	-	-	ns
t_{CAN_OC2}	data input from clock hold	rising *clk* edge	-	-	ns

Note: The *canrx[]* input is re-synchronized internally. The signal does not have to meet any setup or hold requirements.

19 Ethernet Media Access Controller (MAC)

19.1 Overview

The 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. The AMBA interface consists of an APB interface for configuration and control and an AHB master interface which 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 Media Independent Interface (MII) 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. Optional hardware support for the Ethernet Debug Communication Link (EDCL) protocol is also provided. This is an UDP/IP based protocol used for remote debugging.

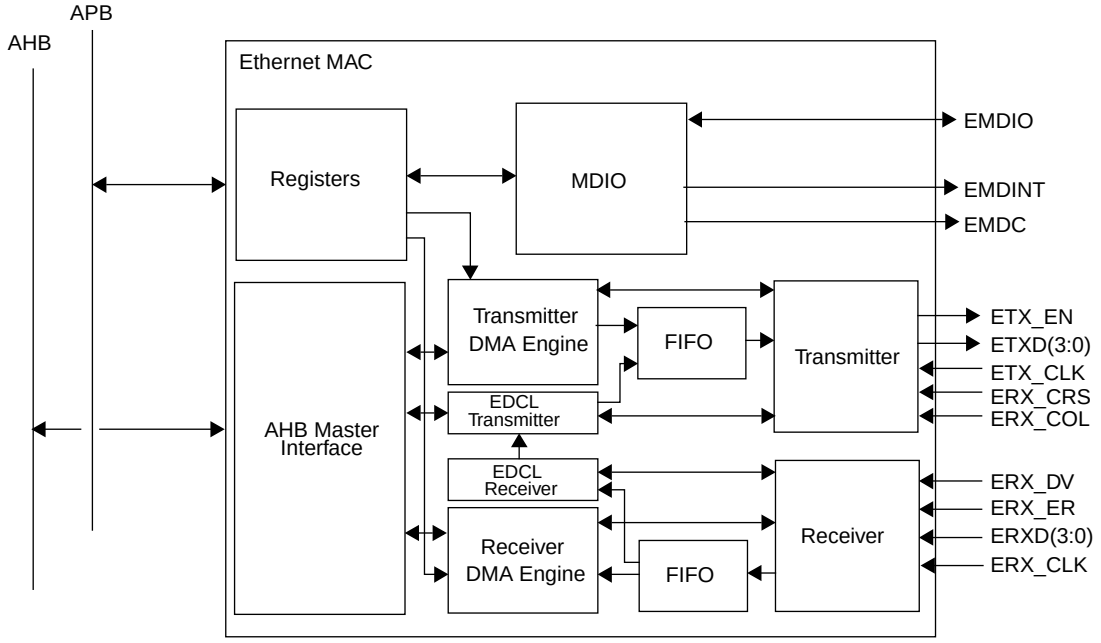


Figure 93. Block diagram of the internal structure of the GRETH.

19.2 Operation

19.2.1 System overview

The GRETH consists of 3 functional units: The DMA channels, MDIO interface and the optional Ethernet Debug Communication Link (EDCL) (not implemented).

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 controlled 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.

19.2.2 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 ctrl packets) are discarded.

19.2.3 Clocking

GRETH has three clock domains: The AHB clock, Ethernet receiver clock and the Ethernet transmitter clock. The ethernet transmitter and receiver clocks are generated by the external ethernet PHY, and are inputs to the core through the MII interface. The three clock domains are unrelated to each other and all signals crossing the clock regions are fully synchronized inside the core.

Both full-duplex and half-duplex operating modes are supported and both can be run in either 10 or 100 Mbit. The minimum AHB clock for 10 Mbit operation is 2.5 MHz, while 18 MHz is needed for 100 Mbit. Using a lower AHB clock than specified will lead to excessive packet loss.

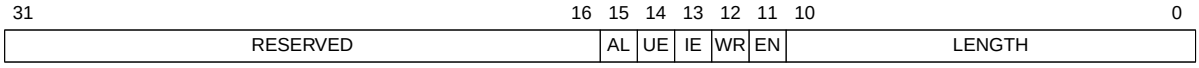
19.3 Tx DMA interface

The transmitter DMA interface is used for transmitting data on an Ethernet network. The transmission is done using descriptors located in memory.

19.3.1 Setting up a descriptor

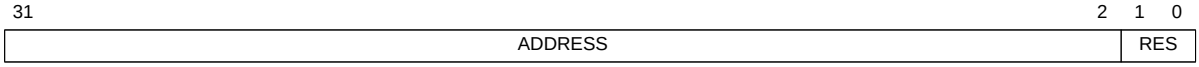
A single descriptor is shown in table 143 and 144. The number of bytes to be sent should be set in the length field and the address field should point to the data. The address must be word-aligned. If the interrupt enable (IE) bit is set, an interrupt will be generated when the packet has been sent (this requires that the transmitter interrupt bit in the control register is also set). The interrupt will be generated regardless of whether the packet was transmitted successfully 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.

Table 143. GRETH transmit descriptor word 0 (address offset 0x0)



31: 16	RESERVED
15	Attempt Limit Error (AL) - The packet was not transmitted because the maximum number of attempts was reached.
14	Underrun Error (UE) - The packet was incorrectly transmitted due to a FIFO underrun error.
13	Interrupt Enable (IE) - Enable Interrupts. An interrupt will be generated when the packet from this descriptor has been sent provided that the transmitter interrupt 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.
12	Wrap (WR) - 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 will increment by 8. The pointer automatically wraps to zero when the 1 kByte boundary of the descriptor table is reached.
11	Enable (EN) - Set to one to enable the descriptor. Should always be set last of all the descriptor fields.
10: 0	LENGTH - The number of bytes to be transmitted.

Table 144. GRETH transmit descriptor word 1 (address offset 0x4)



31: 2	Address (ADDRESS) - Pointer to the buffer area from where the packet data will be loaded.
1: 0	RESERVED

To enable a descriptor the enable (EN) bit should be set and after this is done, the descriptor should not be touched until the enable bit has been cleared by the GRETH.

19.3.2 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 kByte boundary. Bits 31 to 10 hold the base address of descriptor area while bits 9 to 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 at the next descriptor. The pointer will automatically wrap back to zero when the next 1 kByte boundary has been reached (the descriptor at address offset 0x3F8 has been used). The WR bit in the descriptors can be set to make the pointer wrap back to zero before the 1 kByte boundary.

The 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 touched when a transmission is active.

The final step to activate the transmission is to set the transmit enable bit in the control register. This tells the GRETH that there are more 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.

19.3.3 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 bit is set if the FIFO became empty before the packet was completely transmitted while the Attempt Limit Error 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 an transmission ended with an error (when at least one of the two status bits in the transmit 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 were aborted and the transmitter was disabled. The transmitter can be activated again by setting the transmit enable register.

19.3.4 Setting up the data for transmission

The data to be transmitted should be placed beginning at the address pointed by the descriptor address field. The GRETH does not add the Ethernet address and type fields so they must also be stored in the data buffer. The 4 B Ethernet CRC is automatically appended at 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 B, the packet will not be sent.

19.4 Rx DMA interface

The receiver DMA interface is used for receiving data from an Ethernet network. The reception is done using descriptors located in memory.

19.4.1 Setting up descriptors

A single descriptor is shown in table 145 and 146. The address field should point to a word-aligned buffer where the received data should be stored. The GRETH will never store more than 1514 B 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 bit in the control register is also 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.

Table 145. GRETH receive descriptor word 0 (address offset 0x0)

31 19	18	17	16	15	14	13	12	11	10 0
RESERVED	LE	OE	CE	FT	AE	IE	WR	EN	LENGTH

31: 19	RESERVED
18	Length error (LE) - The length/type field of the packet did not match the actual number of received bytes.
17	Overrun error (OE) - The frame was incorrectly received due to a FIFO overrun.
16	CRC error (CE) - A CRC error was detected in this frame.
15	Frame too long (FT) - A frame larger than the maximum size was received. The excessive part was truncated.
14	Alignment error (AE) - An odd number of nibbles were received.
13	Interrupt Enable (IE) - Enable Interrupts. An interrupt will be generated when a packet has been received to this descriptor provided that the receiver interrupt 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.
12	Wrap (WR) - 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 will increment by 8. The pointer automatically wraps to zero when the 1 kByte boundary of the descriptor table is reached.
11	Enable (EN) - Set to one to enable the descriptor. Should always be set last of all the descriptor fields.
10: 0	LENGTH - The number of bytes received to this descriptor.

Table 146. GRETH receive descriptor word 1 (address offset 0x4)

31 2	1 0
ADDRESS	RES

31: 2	Address (ADDRESS) - Pointer to the buffer area from where the packet data will be loaded.
1: 0	RESERVED

19.4.2 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 kByte boundary. Bits 31 to 10 hold the base address of descriptor area while bits 9 to 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 at the next descriptor. The pointer will automatically wrap back to zero when the next 1 kByte boundary has been reached (the descriptor at address offset 0x3F8 has been used). The WR bit in the descriptors can be set to make the pointer wrap back to zero before the 1 kByte boundary.

The 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 touched when reception is active.

The final step to activate reception is to set the receiver enable bit in the control register. This will make the GRETH read the first descriptor and wait for an incoming packet.

19.4.3 Descriptor handling after reception

The GRETH indicates a completed reception by clearing the descriptor enable bit. The other control bits (WR, 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. The status bits are described in table 149.

Packets arriving that are smaller than the minimum Ethernet size of 64 B are not considered as a reception and are discarded. The current receive descriptor will be left untouched an used for the first packet arriving with an accepted size. The 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 IA status register bit will be set.

Packets larger than maximum size cause the FT bit in the receive descriptor to be set. 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.

19.4.4 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 bit in the control register.

19.5 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 provided full support for the MDIO interface.

The MDIO interface can be used to access from 1 to 32 PHY containing 1 to 32 16-bit registers. A read transfer i set up by writing the PHY and register addresses to the MDIO Control register and setting the read bit. This caused the Busy bit to be set and the operation is finished when the Busy bit is cleared. If the operation was successful the Linkfail bit is zero and the data field contains the read data. An unsuccessful operation is indicated by the Linkfail bit being set. The data field is undefined in this case.

A write operation is started by writing the 16-bit data, PHY address and register address to the MDIO Control register and setting the write bit. The operation is finished when the busy bit is cleared and it was successful if the Linkfail bit is zero.

The core also supports status change interrupts from the PHY. A level sensitive interrupt signal can be connected on the mdint input. The PHY status change bit in the status register is set each time an event is detected in this signal. If the PHY status interrupt enable bit is set at the time of the event the core will also generate an interrupt on the AHB bus.

19.6 Media Independent Interface

There are several interfaces defined between the MAC sublayer and the Physical layer. The GRETH supports the Media Independent Interface (MII).

The MII was defined in the 802.3 standard and is most commonly supported. The ethernet interface have been implemented according to this specification. It uses 16 signals.

19.7 Registers

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

*Table 147.*GRETH registers

APB address offset	Register
0x0	Control register
0x4	Status/Interrupt-source register
0x8	MAC Address MSB
0xC	MAC Address LSB
0x10	MDIO Control/Status
0x14	Transmit descriptor pointer
0x18	Receiver descriptor pointer

Table 148. GRETH control register

31	30 28	27	26	11	10	9 8	7	6	5	4	3	2	1	0
ED	BS	-	MA	RESERVED	PI	RES	SP	RS	PM	FD	RI	TI	RE	TE

31 EDCL available (ED) - Set to one if the EDCL is available. Fixed to zero.

30: 28 EDCL buffer size (BS) - UNUSED.

27 RESERVED

26 MDIO interrupts available (MA) - Set to one when the core supports mdio interrupts. Read only.

27: 11 RESERVED

10 PHY status change interrupt enable (PI) - Enables interrupts for detected PHY status changes.

9: 8 RESERVED

7 Speed (SP) - Sets the current speed mode. 0 = 10 Mbit, 1 = 100 Mbit. Only used in RMII mode (not implemented). A default value is automatically read from the PHY after reset.

6 Reset (RS) - A one written to this bit resets the GRETH core. Self clearing.

5 Promiscuous mode (PM) - If set, the GRETH operates in promiscuous mode which means it will receive all packets regardless of the destination address. Not Reset.

4 Full duplex (FD) - If set, the GRETH operates in full-duplex mode otherwise it operates in half-duplex. Not Reset.

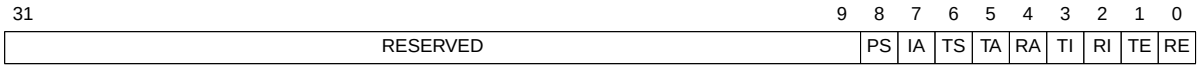
3 Receiver interrupt (RI) - Enable Receiver Interrupts. An interrupt will be generated each time a packet is received when this bit is set. The interrupt is generated regardless if the packet was received successfully or if it terminated with an error. Not Reset.

2 Transmitter interrupt (TI) - 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.

1 Receive enable (RE) - Should be written with a one each time new descriptors are enabled. As long as this bit is one the GRETH will read 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. Reset value: ‘0’.

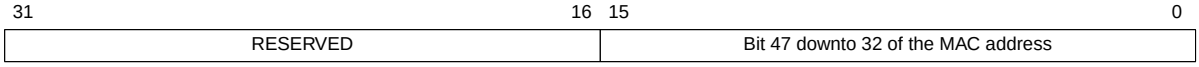
0 Transmit enable (TE) - Should be written with a one each time new descriptors are enabled. As long as this bit is one the GRETH will read new descriptors and as soon as it encounters a disabled descriptor it will stop until TE is set again. This bit should be written with a one after the new descriptors have been enabled. Reset value: ‘0’.

Table 149. GRETH status register



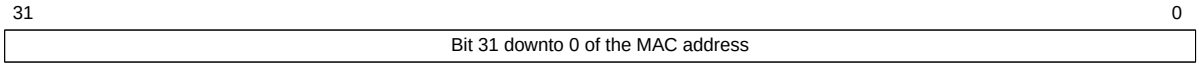
8	PHY status changes (PS) - Set each time a PHY status change is detected - UNUSED.
7	Invalid address (IA) - A packet with an address not accepted by the MAC was received. Cleared when written with a one. Reset value: ‘0’.
6	Too small (TS) - A packet smaller than the minimum size was received. Cleared when written with a one. Reset value: ‘0’.
5	Transmitter AHB error (TA) - An AHB error was encountered in transmitter DMA engine. Cleared when written with a one. Not Reset.
4	Receiver AHB error (RA) - An AHB error was encountered in receiver DMA engine. Cleared when written with a one. Not Reset.
3	Transmitter interrupt (TI) - A packet was transmitted without errors. Cleared when written with a one. Not Reset.
2	Receiver interrupt (RI) - A packet was received without errors. Cleared when written with a one. Not Reset.
1	Transmitter error (TE) - A packet was transmitted which terminated with an error. Cleared when written with a one. Not Reset.
0	Receiver error (RE) - A packet has been received which terminated with an error. Cleared when written with a one. Not Reset.

Table 150. GRETH MAC address MSB.



15: 0	The two most significant bytes of the MAC Address. Not Reset.

Table 151. GRETH MAC address LSB.



31: 0	The four least significant bytes of the MAC Address. Not Reset.

Table 152. GRETH MDIO ctrl/status register.

31 16	15 11	10 6	5	4	3	2	1	0
DATA	PHYADDR	REGADDR		NV	BU	LF	RD	WR

31: 16 Data (DATA) - Contains data read during a read operation and data that is transmitted is taken from this field. Not Reset.

15: 11 PHY address (PHYADDR) - This field contains the address of the PHY that should be accessed during a write or read operation. Not Reset.

10: 6 Register address (REGADDR) - This field contains the address of the register that should be accessed during a write or read operation. Not Reset.

5 RESERVED

4 Not valid (NV) - When an operation is finished (BUSY = 0) this bit indicates whether valid data has been received that is, the data field contains correct data. Not Reset.

3 Busy (BU) - 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. Reset value: ‘0’.

2 Linkfail (LF) - When an operation completes (BUSY = 0) this bit is set if a functional management link was not detected. Not Reset.

1 Read (RD) - Start a read operation on the management interface. Data is stored in the data field. Reset value: ‘0’.

0 Write (WR) - Start a write operation on the management interface. Data is taken from the Data field. Reset value: ‘0’.

Table 153. GRETH transmitter descriptor table base address register.

31 10	9 3	2 0
BASEADDR	DESCPNT	RES

31: 10 Transmitter descriptor table base address (BASEADDR) - Base address to the transmitter descriptor table.Not Reset.

9: 3 Descriptor pointer (DESCPNT) - Pointer to individual descriptors. Automatically incremented by the Ethernet MAC.

2: 0 RESERVED

Table 154. GRETH receiver descriptor table base address register.

31 10	9 3	2 0
BASEADDR	DESCPNT	RES

31: 10 Receiver descriptor table base address (BASEADDR) - Base address to the receiver descriptor table.Not Reset.

9: 3 Descriptor pointer (DESCPNT) - Pointer to individual descriptors. Automatically incremented by the Ethernet MAC.

2: 0 RESERVED

19.8 Signal definitions and reset values

The signals and their reset values are described in table 155.

*Table 155.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
emdio	Input/Output	Management Interface Data Input/Output	-	Tri-state
emdc	Output	Management Interface Data Clock	High	Logical 0
emdint	Input	Management Interface Data Interrupt	High	-
erx_clk	Input	Receiver clock	-	-
erx_dv	Input	Receiver data valid	High	-
erx_crs	Input	Receiver carrier sense	High	-
erx_er	Input	Receiver error	High	-
erx_col	Input	Receiver collision	High	-
erxd[3:0]	Input	Receiver input data	-	-
etx_clk	Input	Transmitter clock	-	-
etx_en	Output	Transmitter enable	High	Logical 0
etxd[3:0]	Output	Transmitter output data	-	-

19.9 Timing

The timing waveforms and timing parameters are shown in figure 94 and are defined in table 156.

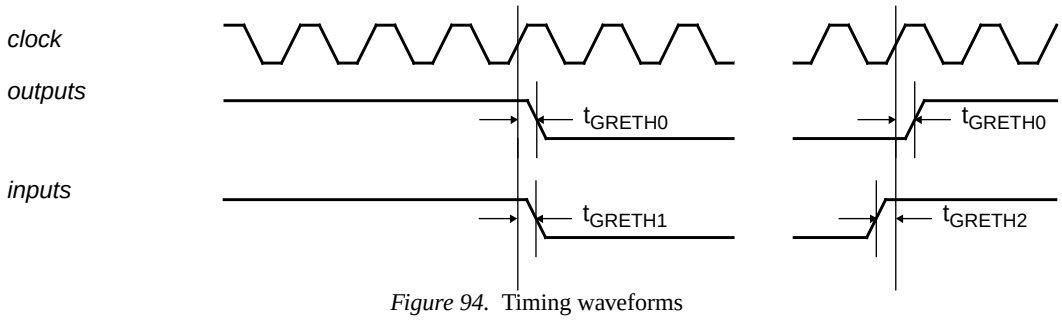


Figure 94. Timing waveforms

*Table 156.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{GRETHTXCLK0}$	Ethernet MII transmit clock period	-	40 *1)*	-	ns
$t_{GRETHRXCLK0}$	Ethernet MII receive clock period	-	40 *1)*	-	ns
t_{GRETH0}	transmitter clock to output delay	rising clock edge	0	26	ns
t_{GRETH1}	input to receiver clock hold	rising clock edge	5	-	ns
t_{GRETH2}	input to receiver clock setup	rising clock edge	10	-	ns

Note 1: The *erx_crs, erx_col, emdio* andi *emdint* inputs are re-synchronized internally. The signals do not have to meet any setup or hold requirements.

Note 2: The *emdio* and *emdc* outputs are low speed signals without any timing relationship with the *erx_clk* or *etx_clk* clocks.

Note 3 The minimum clock period, and the resulting maximum clock frequency, is dependent on the manufacturing lot for the Actel RTAX2000S parts and expected radiation levels.

The degradation criterion for Propagation Delays for the RTAX2000S parts is according to Actel Total Ionizing Dose Test Report 10% at 300 krad (Si). The above specified timing values are guaranteed up to 50 krad (Si).

If a higher total ionizing dose is expected than 50 krad (Si), the corresponding post-annealing propagation delays that can be found in the Actel Total Ionizing Dose Test Report can be used to degrade the timing more accurately (typical degradation is within 1% after 300 krad (Si)).

The functional behavior of the part is guaranteed up to 300 krad (Si).

20 PCI Initiator/Target

20.1 Overview

This core provides a complete interface to an external PCI bus, with both initiator and target functions. The PCI initiator has PCI system host capability. The interface is based on the Actel CorePCIF IP and provides an AMBA bus backend.

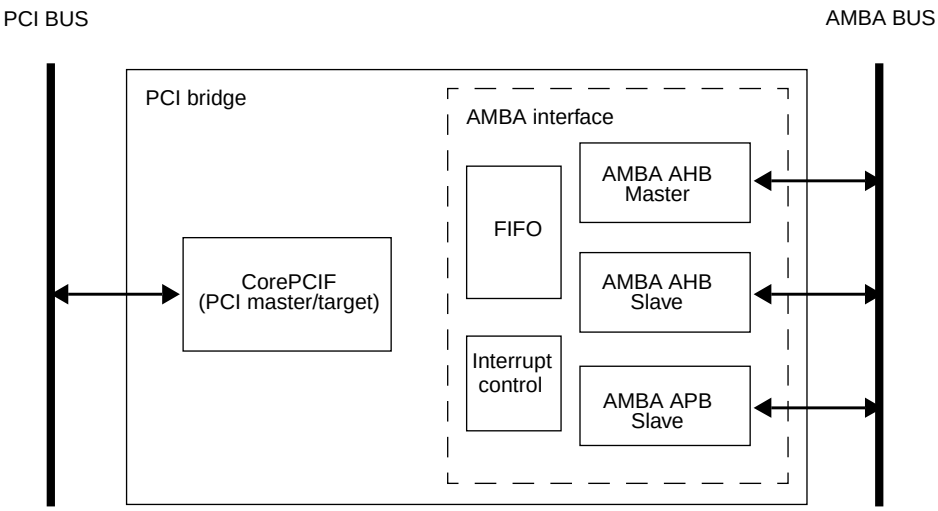


Figure 95. Block diagram

20.2 Operation

20.2.1 PCI Initiator

The PCI initiator can be enabled and disabled in the PCI configuration space. When the PCI master is disabled, all accesses to the AMBA backend except for accesses to its own PCI configuration space will generate a error response on the AMBA bus. The core occupies two areas in the AMBA address space. One AHB memory bank used for generation of PCI memory cycles and one AHB I/O bank for PCI I/O and PCI configuration cycles. The AHB I/O bank has a fixed size of 128 kBytes.

PCI memory cycles: The PCI master generates “Memory read multiple” accesses on the PCI bus for burst read accesses to the memory bank of the AMBA backend. For single read accesses to the AMBA backend, “Memory read” accesses are generated on the PCI bus. All write accesses to the AMBA backend generate “Memory write” accesses on the PCI bus. The most significant bits of the PCI address are set by mapping registers, while the least significant bits are directly transferred from the AMBA backend. A separate mapping register is implemented for each AMBA master.

To generate PCI memory accesses, the following steps must be executed. The master function must be enabled, preferably by the PCI system host during the PCI configuration and the mapping register must be programmed with the most significant bits of the PCI address that should be accessed. After this, read and write accesses to the AMBA backend will be transferred the corresponding PCI address.

PCI I/O cycles: Accesses to the lower half of the AHB I/O bank is translated to PCI I/O cycles. The upper 16 bits of the PCI address are defined by a mapping register (AMBA to PCI mapping for PCI I/O cycles) while the lower 16 bits are transferred form the AMBA bus.

To generate PCI I/O accesses, The master function must be enabled, preferably by the PCI system host during the PCI configuration and the mapping register must be programmed with the most significant bits of the PCI address that should be accessed. After this, read and write accesses to the lower half of the AHB I/O bank will be translated to “I/O read” and “I/O write” on the PCI bus.

PCI configuration cycles: Accesses to the upper half of the AHB I/O bank translates to PCI configuration cycles. The core can generate both type 0 and type 1 PCI configuration cycles. Type 0 configuration cycles is generated when bit 16 to 23 in the PCI Configuration register is zero. In this case the upper 21 bits of the PCI address is calculated from bit 15:11 of the AMBA address (PCI_address[AMBA_address[15:11] + 10] = ’1’). The lower 11 bits of the PCI address is directly transferred from AMBA address except for bit 1 and 0 which are always set to "00". When AMBA_address[15:11] is equal to zero, the core will access its own PCI configuration space (through the backend without generating a PCI access on the PCI bus). Accesses to the core\u2019s own PCI configuration space are allowed before the master function is enabled, to be able to enable the master. For type 1 configuration cycles bit 16 to 23 in the PCI Configuration register indicates which PCI bus to configure. In this case the bit 16 to 23 of the PCI address is set by bit 16 to 23 of the PCI configuration register. Bit 2 to 15 is directly transferred from the AMBA address and bit 0 and 1 is set to "01".

20.2.2 PCI Target

The PCI target function can be enabled and disabled in the PCI configuration space. When enabled, the PCI target accepts “Memory read”, “Memory read multiple”, and “Memory write” commands for data accesses. Access to the PCI configuration space is provided for “Configuration read” and “Configuration write” commands. When the PCI target is accessed (except for Configuration read/write), the core transfers this access to the AMBA backend. The most significant bits of the address used by the AMBA backend is controlled by a mapping register, while the least significant bits of the address are directly transferred from the PCI access. A separate mapping register is implemented for PCI BAR 1 - 4.

The following configuration steps are required for the PCI target to correctly respond to data accesses. The target must be setup to accept memory accesses, preferably by the PCI system host during PCI configuration. The mapping register must be programmed with the most significant bits of the AMBA address. After this configuration, accesses to the PCI target interface are transferred to the AMBA bus.

The PCI target interface provides three PCI memory bars: BAR 0 enables access to the configuration registers; BAR 1 is used for data transfers; BAR 5 is used by the master function and should never be accessed by any other master on the PCI bus.

20.2.3 Configuration

The core has configuration registers accessible via the AMBA APB interface and via the PCI BAR 0. The PCI BAR to AMBA address mapping registers and the interrupt controller registers are accessible via the PCI BAR 0. The interrupt registers must be setup to enable interrupt handling. The PCI to AMBA address mapping registers must be setup to translate the PCI access into the correct AMBA access. These mapping registers are also accessible via the AMBA APB interface. The AMBA to PCI address mapping registers are accessible via the AMBA APB interface. These registers must be setup to translate the access to the AMBA AHB slave interface into the correct PCI address.

20.2.4 Byte access

Single byte accesses are supported by the interface. The byte is directly transferred between the two buses (i.e. a write of the byte located at bit 7:0 on the AMBA bus is transferred to a write of the byte located at bit 7:0 on the PCI bus). The core does not support the PCI byte-enables to be changed during burst accesses. The PCI byte-enable is only sampled for the first word in a burst.

20.2.5 Error response

The PCI target do not generate error responses. A PCI access that transfer into an AMBA access with an invalid address is not generating an error response on the PCI bus.

The AMBA backend generate a two cycle error response in the following cases. When the AMBA backend is accessed and the PCI master function is disabled, or when Master/Target abort is detected by the PCI master (except in case a PCI configuration access is performed). In the case of a Master/Target abort, the core automatically resets the error bits in the PCI configuration space and saves the error status in its status register.

20.2.6 Input interrupts (sampling of the PCI interrupt)

The core samples the PCI interrupt signals when its mask bit, in the input interrupt mask register, is set to one. If one of the unmasked PCI interrupt signals is asserted (active low), the core generate a interrupt on the AMBA APB bus.

20.2.7 Host signal

The host signal *pci_host* is provided to be able to determine if the core should act as the system host and is inserted in the PCI host slot. The status of this signal can be read out from the status register. If the host bit in the status register is ‘0’, the core is in the host slot.

If the host signal *pci_host* is asserted during reset, then the *pci_rst* output will also be asserted active low with the same timing as the *resetn* input and then returned to tri-state. The *pci_rst* input resets the PCI interface independently of the *pci_host* host signal.

20.3 Registers

The core is programmed via registers mapped into the APB address space and into PCI BAR 0.

*Table 157.*PCIF: APB registers

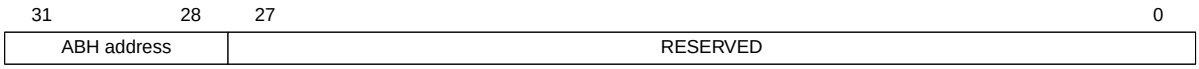
APB address offset	Register
0x00	PCI to AMBA mapping for PCI BAR 1
0x04	PCI to AMBA mapping for PCI BAR 2
0x08	PCI to AMBA mapping for PCI BAR 3 (disabled)
0x0C	PCI to AMBA mapping for PCI BAR 4 (disabled)
0x10	PCI configuration register
0x14	AMBA to PCI address mapping for PCI I/O cycles
0x18	Status register
0x1C	Input interrupt mask (for host configuration)
0x40 - 0x7C	AMBA master to PCI address mapping registers

The AMBA master to PCI mapping registers are all word aligned. Only the registers corresponding to a master included in the system are implemented (i.e. if a system includes 8 masters, with master ID 0 to 7, the 8 first mapping registers are implemented).

*Table 158.*PCIF: PCI BAR 0 registers

PCI address offset	Register
0x00	PCI to AMBA mapping for PCI BAR 1
0x04	PCI to AMBA mapping for PCI BAR 2
0x08	PCI to AMBA mapping for PCI BAR 3 (disabled)
0x0C	PCI to AMBA mapping for PCI BAR 4 (disabled)

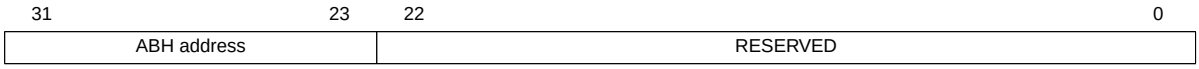
Table 159. PCIF: PCI to AMBA mapping for PCI BAR *1* register



31 : 28 MBS of the AMBA address

27 : 0 RESERVED

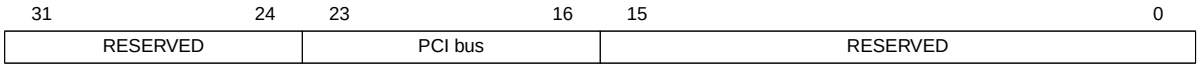
Table 160. PCIF: PCI to AMBA mapping for PCI BAR *2* register



31 : 23 MBS of the AMBA address

22 : 0 RESERVED

Table 161. PCIF: PCI configuration register

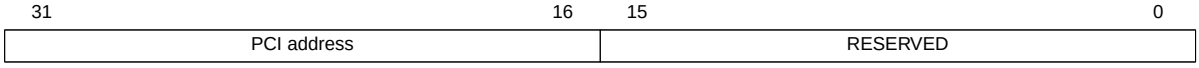


31 : 24 RESERVED

23 : 16 Indicates which PCI bus should be addressed during a type 1 PCI configuration access. When this register is zero, type 0 PCI configuration accesses will be generated. This register is zero after reset.

15 : 0 RESERVED

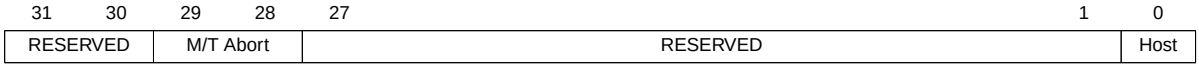
Table 162. PCIF: AMBA to PCI address mapping for PCI I/O cycles



31 : 16 MBS of the PCI address

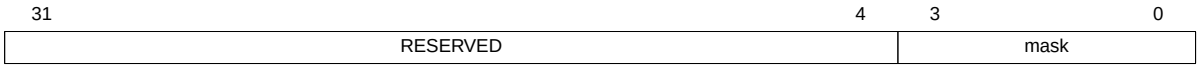
15 : 0 RESERVED

Table 163. PCIF: Status register



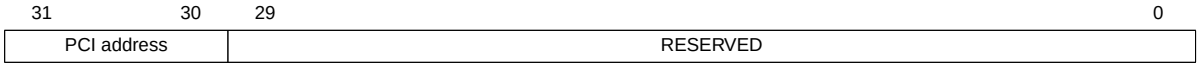
31 : 30	RESERVED
29 : 28	Master and Target abort status from PCI configuration space
27 : 1	RESERVED
0	System host (’0’ = in host slot, ’1’ = in peripheral slot)

Table 164. PCIF: Input interrupt mask



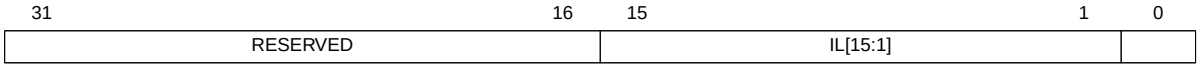
31 : 4	RESERVED
3 : 0	Interrupt mask for the four PCI interrupt signals. If bit number 0 is ‘1’ then PCI Interrupt A is unmasked, and so on. Reset to all zeros.

Table 165. PCIF: AMBA master to PCI address mapping registers



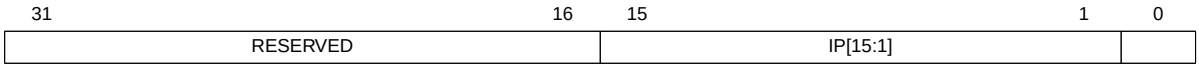
31 : 30	MBS of the PCI address
29 : 0	RESERVED

Table 166. PCIF: Interrupt level register



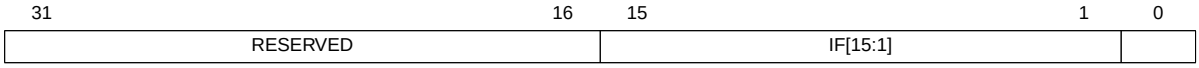
31 : 16	RESERVED
15 : 1	Interrupt level n (IL[n]): Interrupt level for interrupt n
0	RESERVED

Table 167. PCIF: Interrupt pending register



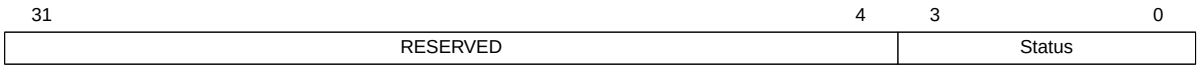
31 : 16	RESERVED
15 : 1	Interrupt pending n (IP[n]): Interrupt pending for interrupt n
0	RESERVED

Table 168. PCIF: Interrupt force register



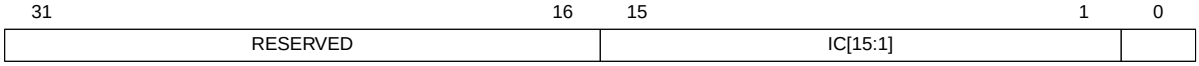
31 : 16	RESERVED
15 : 1	Interrupt force n (IF[n]): Force interrupt nr n
0	RESERVED

Table 169. PCIF: Interrupt status/ack register



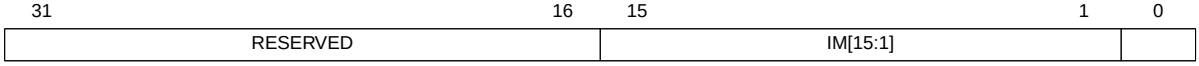
31 : 4	RESERVED
3 : 0	Interrupt status

Table 170. PCIF: Interrupt clear register



31 : 16	RESERVED
15 : 1	Interrupt clear n (IC[n]): Writing ‘1’ to IC[n] will clear interrupt n
0	RESERVED

Table 171. PCIF: Interrupt mask register



31 : 16	RESERVED
15 : 1	Interrupt mask n(IM[n]): If IM[n] = 0 the interrupt n is masked, otherwise it is enabled
0	RESERVED

20.4 Signal definitions and reset values

The signals and their reset values are described in table 172.

*Table 172.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
pci_clk	Input	PCI clock	-	-
pci_rst	Input/Output	Reset	Low	-
pci_gnt	Input	Grant signal	Low	-
pci_idsel	Input	Device select during configuration	High	-
pci_host	Input	System host	Low	-
pci_ad[31:0]	Input/Output	Address and Data bus	High	Tri.state
pci_cbe[3:0]	Input/Output	Bus command and byte enable	Low	Tri-state
pci_par	Input/Output	Parity signal	High	Tri-state
pci_frame	Input/Output	Cycle frame	Low	Tri-state
pci_devsel	Input/Output	Device select	Low	Tri-state
pci_irdy	Input/Output	Initiator ready	Low	Tri-state
pci_trdy	Input/Output	Target ready	Low	Tri-state
pci_stop	Input/Output	Stop	Low	Tri-state
pci_perr	Input/Output	Parity error	Low	Tri-state
pci_serr	Input/Output	System error	Low	Tri-state
pci_req	Input/Output	Request signal	Low	Logical 1
pci_int[3:0]	Input/Output	Interrupt signal	Low	Tri-state

20.5 Timing

The timing waveforms and timing parameters are shown in figure 96 and are defined in table 173.

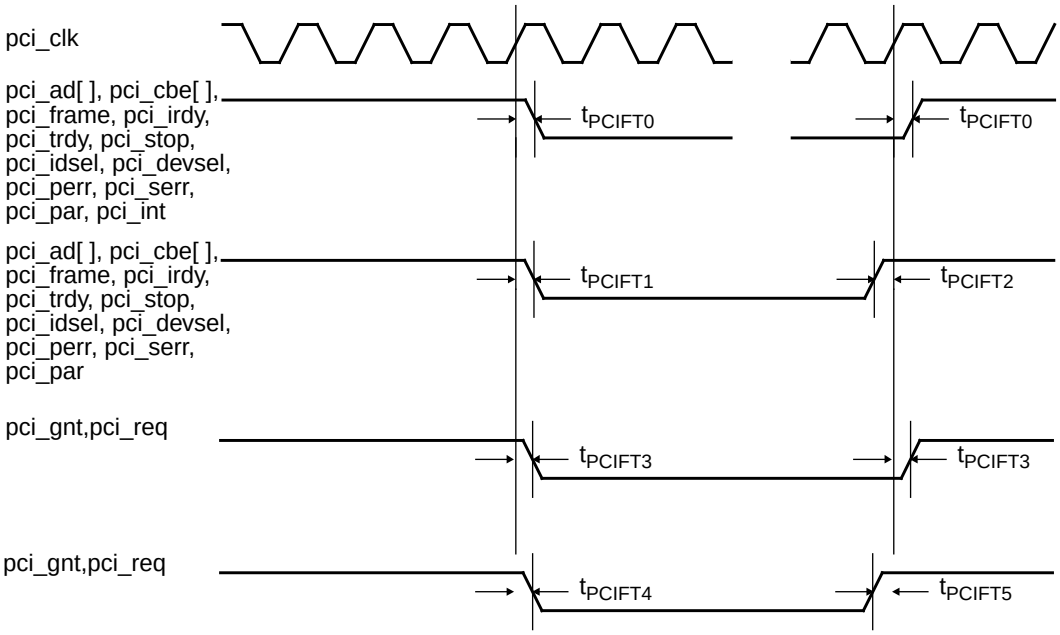


Figure 96. Timing waveforms

*Table 173.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{PCICLK0}$	PCI clock period	-	30 *1)*	-	ns
t_{PCIFT0}	clock to output delay	rising clk edge	2	11	ns
t_{PCIFT1}	input to clock hold	rising clk edge	0	-	ns
t_{PCIFT2}	input to clock setup	rising clk edge	7	-	ns
t_{PCIFT3}	clock to output delay	rising clk edge	2	12	ns
t_{PCIFT4}	input to clock hold	rising clk edge	0	-	ns
t_{PCIFT5}	input to clock setup	rising clk edge	10, 12	-	ns

Note 1: The minimum clock period, and the resulting maximum clock frequency, is dependent on the manufacturing lot for the Actel RTAX2000S parts and expected radiation levels.

The degradation criterion for Propagation Delays for the RTAX2000S parts is according to Actel Total Ionizing Dose Test Report 10% at 300 krad (Si). The above specified timing values are guaranteed up to 50 krad (Si).

If a higher total ionizing dose is expected than 50 krad (Si), the corresponding post-annealing propagation delays that can be found in the Actel Total Ionizing Dose Test Report can be used to degrade the timing more accurately (typical degradation is within 1% after 300 krad (Si)).

The functional behavior of the part is guaranteed up to 300 krad (Si).

21 PCI Arbiter

21.1 Overview

PCIARB is an arbiter for the PCI bus, according to the PCI specification version 2.1. It supports 8 agents. The arbiter uses nested round-robbing policy in two priority levels. The priority assignment is programmable through an APB interface.

21.2 Operation

21.2.1 Scheduling algorithm

The arbiter uses the algorithm described in the implementation note of section 3.4 of the PCI standard. The bus is granted by two nested round-robbing loops, where an agent number and a priority level is assigned to each agent. The agent number determines which pair of REQ/GNT lines are used. Agents are counted from 0 to 7.

All agents in one level have equal access to the bus through a round-robbing policy. All agents of level 1 as a group have access equal to each agent of level 0. Re-arbitration occurs, when FRAMEN is asserted, as soon as any other master has requested the bus, but only once per transaction. With programmable priorities, the priority level of all agents except 7 is programmable via APB.

In a 256 byte APB address range, the priority level of agent N is accessed via the address 0x80 + 4*N. The APB read returns 0 on all non-implemented addresses, and the address bits (1:0) are not decoded.

21.2.2 Time-out

The “broken master” time-out is another reason for re-arbitration (section 3.4.1 of the PCI standard). Grant is removed from an agent, which has not started a cycle within 16 cycles after request (and grant). Reporting of such a ‘broken’ master is not implemented.

21.2.3 Turn-over

A turn-over cycle is required by the standard, when re-arbitration occurs during idle state of the bus. Notwithstanding to the standard, “idle state” is assumed, when FRAMEN is high for more than 1 cycle.

21.2.4 Bus parking

The bus is parked to agent 0 after reset, it remains granted to the last owner, if no other agent requests the bus. When another request is asserted, re-arbitration occurs after one turnover cycle.

21.2.5 Lock

Lock is defined as a resource lock by the PCI standard. The optional bus lock mentioned in the standard is not considered here and there are no special conditions to handle when LOCKN is active during in arbitration.

21.2.6 Latency

Latency control in PCI is via the latency counters of each agent. The arbiter does not perform any latency check and a once granted agent continues its transaction until its grant is removed AND its

own latency counter has expired. Even though, a bus re-arbitration occurs during a transaction, the hand-over only becomes effective, when the current owner deasserts FRAMEN.

21.3 Signal definitions and reset values

The signals and their reset values are described in table 174.

*Table 174.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
pci_arb_req[7:0]	Input	PCI request	Low	-
pci_arb_gnt[7:0]	Output	PCI grant	Low	Logical 1

21.4 Timing

The timing waveforms and timing parameters are shown in figure 97 and are defined in table 175.

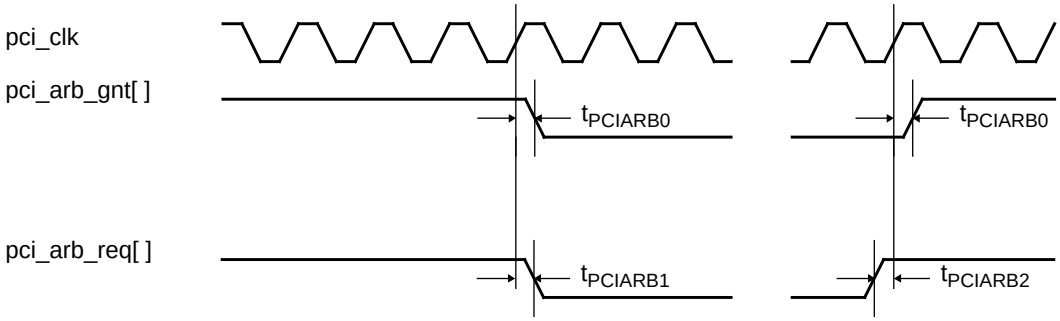


Figure 97. Timing waveforms

*Table 175.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{PCIARB0}$	clock to output delay	rising clk edge	2	12	ns
$t_{PCIARB1}$	input to clock hold	rising clk edge	0	-	ns
$t_{PCIARB2}$	input to clock setup	rising clk edge	12	-	ns

22 Serial Debug Interface

22.1 Overview

The interface consists of a UART connected to the AMBA AHB bus as a master. 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 AMBA AHB bus.

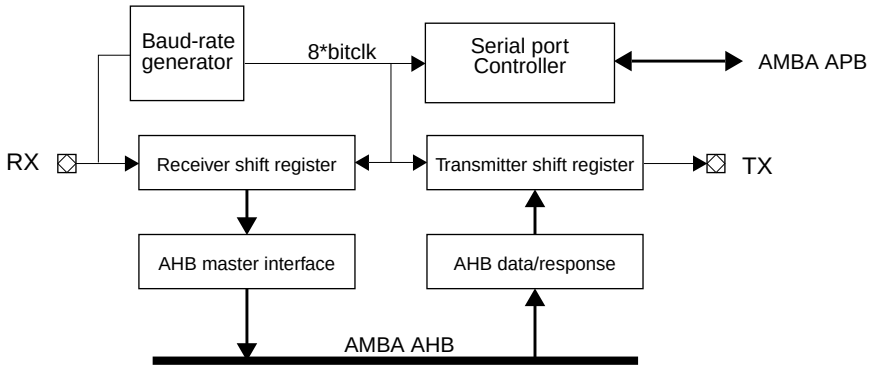


Figure 98. Block diagram

22.2 Operation

22.2.1 Transmission protocol

The interface supports a simple protocol where commands consist of a control byte, followed by a 32-bit address, followed by optional write data. Write access does not return any response, while a read access only returns the read data. Data is sent on 8-bit basis as shown below.

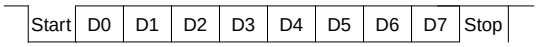


Figure 99. Data frame

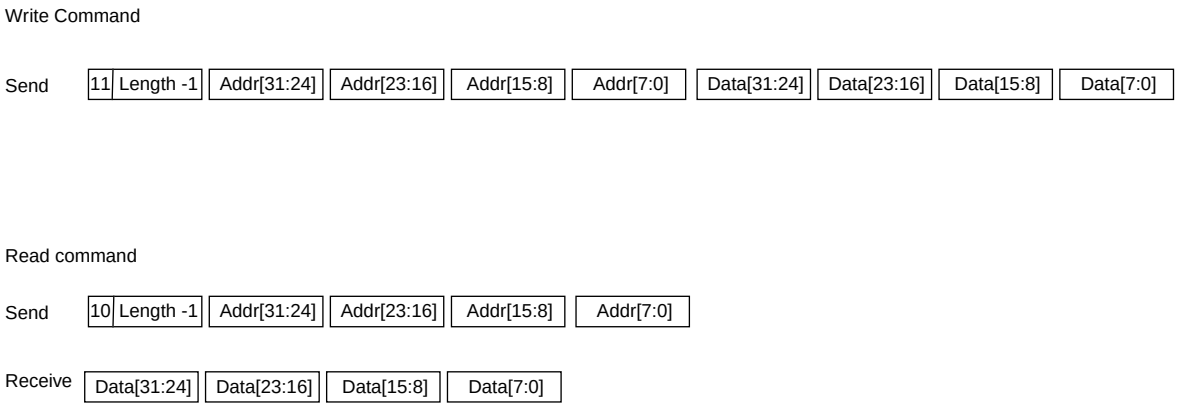


Figure 100. Commands

Block transfers can be performed be 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.

22.2.2 Baud rate generation

The 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 8 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 has been found, the corresponding scaler reload value is latched into the reload register, and the 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 to change to baudrate from the external transmitter. For proper baudrate detection, the value 0x55 should be transmitted to the receiver after reset or after sending break.

The best scaler value for manually programming the baudrate can be calculated as follows:

```
scaler = (((system_clk*10)/(baudrate*8))-5)/10
```

22.3 Registers

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

*Table 176.*AHB UART registers

APB address offset	Register
0x4	AHB UART status register
0x8	AHB UART control register
0xC	AHB UART scaler register

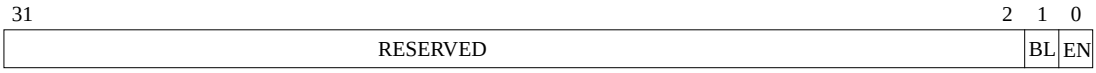


Figure 101. AHB UART control register

0: Receiver enable (EN) - if set, enables both the transmitter and receiver. Reset value: ‘0’.

1: Baud rate locked (BL) - is automatically set when the baud rate is locked. Reset value: ‘0’.

Figure 102. AHB UART status register

0: Data ready (DR) - indicates that new data has been received by the AMBA AHB master interface. Read only. Reset value: ‘0’.

1: Transmitter shift register empty (TS) - indicates that the transmitter shift register is empty. Read only. Reset value: ‘1’.

2: Transmitter hold register empty (TH) - indicates that the transmitter hold register is empty. Read only. Reset value: ‘1’.
3: Break (BR) - indicates that a BREAKE has been received. Reset value: ‘0’.
4: Overrun (OV) - indicates that one or more character have been lost due to overrun. Reset value: ‘0’.
6: Framing error (FE) - indicates that a framing error was detected. Reset value: ‘0’.

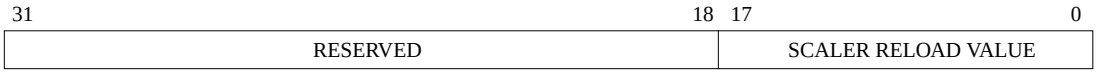


Figure 103. AHB UART scaler reload register

17:0 Baudrate scaler reload value = (((system_clk*10)/(baudrate*8))-5)/10. Reset value: “3FFFF“.

22.4 Signal definitions and reset values

The signals and their reset values are described in table 177.

*Table 177.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
dsutx	Output	UART transmit data line	-	Logical 1
dsurx	Input	UART receive data line	-	-

22.5 Timing

The timing waveforms and timing parameters are shown in figure 104 and are defined in table 178.

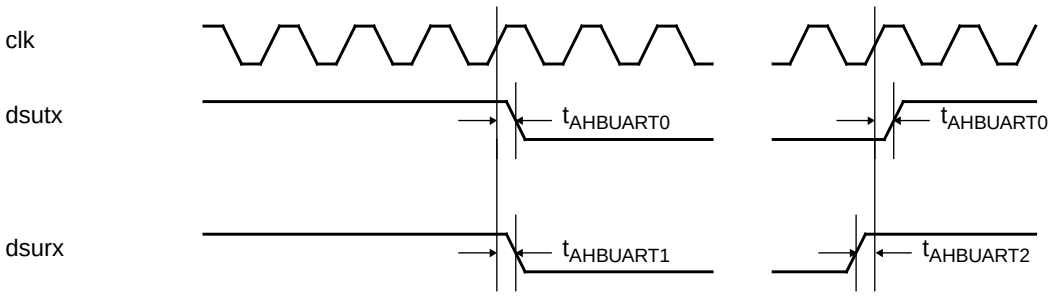


Figure 104. Timing waveforms

*Table 178.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{AHBUART0}$	clock to output delay	rising *clk* edge	0	25	ns
$t_{AHBUART1}$	input to clock hold	rising *clk* edge	-	-	ns
$t_{AHBUART2}$	input to clock setup	rising *clk* edge	-	-	ns

Note: The *dsurx* input is re-synchronized internally. The signal does not have to meet any setup or hold requirements.

23 JTAG Debug Interface

23.1 Overview

The JTAG debug interface provides access to on-chip AMBA AHB bus through JTAG. The JTAG debug interface implements a simple protocol which translates JTAG instructions to AHB transfers. Through this link, a read or write transfer can be generated to any address on the AHB bus.

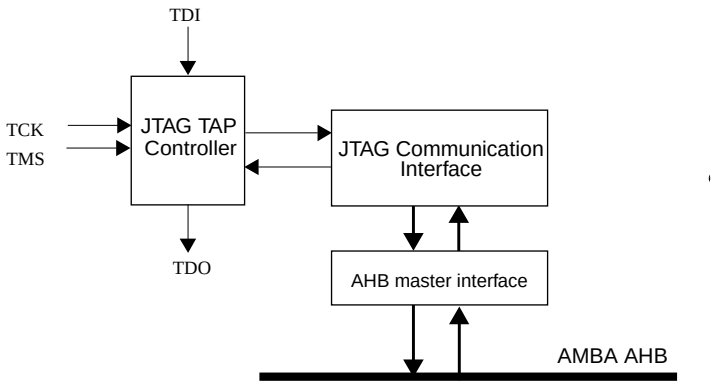


Figure 105. JTAG Debug link block diagram

23.2 Operation

23.2.1 Transmission protocol

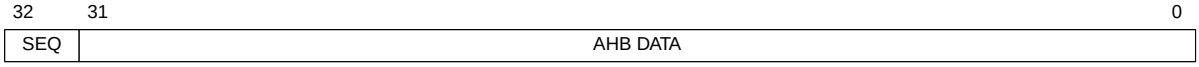
The JTAG Debug link decodes two JTAG instructions and implements two JTAG data registers: the command/address register and data register. A read access is initiated by shifting in a command consisting of read/write bit, AHB access size and AHB address into the command/address register. The AHB read access is performed and data is ready to be shifted out of the data register. Write access is performed by shifting in command, AHB size and AHB address into the command/data 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 shifting in SEQ bit in data register for following accesses. The SEQ bit will increment the AHB address for the subsequent access. Sequential transfers should not cross a 1 kByte boundary. Sequential transfers are always word based.

Table 179. JTAG debug link Command/Address register

34	33 32	31 ... 0
W	SIZE	AHB ADDRESS

34	Write (W) - ‘0’ - read transfer, ‘1’ - write transfer
33 32	AHB transfer size - “00” - byte, “01” - half-word, “10” - word, “11”- reserved
31 30	AHB address

Table 180. JTAG debug link Data register



32	Sequential transfer (SEQ) - If '1' is shifted in this bit position when read data is shifted out or write data shifted in, the subsequent transfer will be to next word address.
31 30	AHB Data - AHB write/read data. For byte and half-word transfers data is aligned according to big-endian order where data with address offset 0 data is placed in MSB bits.

23.3 Registers

The core does not implement any registers mapped in the AMBA AHB or APB address space.

23.4 Signal definitions and reset values

The signals and their reset values are described in table 181.

*Table 181.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
dsutck	Input	JTAG clock	-	-
dsutms	Input	JTAG TMS	High	-
dsutdi	Input	JTAG TDI	High	-
dsutdo	Output	JTAG TDO	High	undefined

23.5 Timing

The timing waveforms and timing parameters are shown in figure 106 and are defined in table 182.

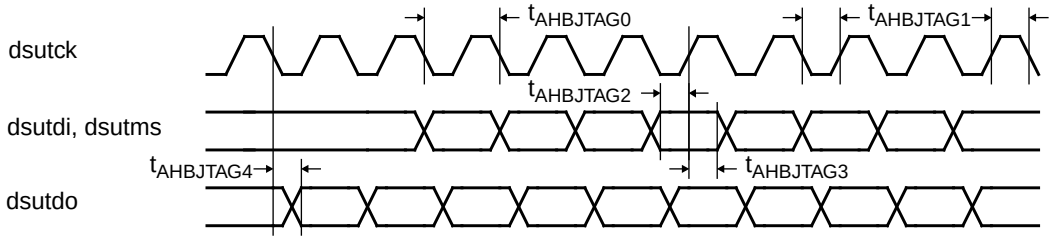


Figure 106. Timing waveforms

*Table 182.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{AHBJTAG0}$	clock period	-	100	-	ns
$t_{AHBJTAG1}$	clock low/high period	-	40	-	ns
$t_{AHBJTAG2}$	data input to clock setup	rising *dsutck* edge	15	-	ns
$t_{AHBJTAG3}$	data input from clock hold	rising *dsutck* edge	0	-	ns
$t_{AHBJTAG4}$	clock to data output delay	falling *dsutck* edge	-	25	ns

24 Clock generation

24.1 Overview

The clock generator implements internal clock generation and buffering.

24.2 Signal definitions and reset values

The signals and their reset values are described in table 183.

*Table 183.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
clk	Input	System clock	Rising edge	-

24.3 Timing

The timing waveforms and timing parameters are shown in figure 107 and are defined in table 184.

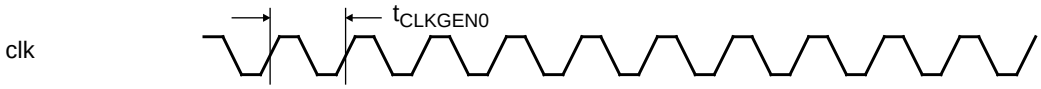


Figure 107. Timing waveforms

*Table 184.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{CLKGEN0}$	clock period, CID 1 and 3	-	41.66	41.66	ns *1)*
$t_{CLKGEN0}$	clock period, CID 2 and 4 to 8	-	40	-	ns *1)*

Note 1: The minimum clock period, and the resulting maximum clock frequency, is dependent on the manufacturing lot for the Actel RTAX2000S parts and expected radiation levels.

The degradation criterion for Propagation Delays for the RTAX2000S parts is according to Actel Total Ionizing Dose Test Report 10% at 300 krad (Si). The above specified timing values are guaranteed up to 50 krad (Si).

If a higher total ionizing dose is expected than 50 krad (Si), the corresponding post-annealing propagation delays that can be found in the Actel Total Ionizing Dose Test Report can be used to degrade the timing more accurately (typical degradation is within 1% after 300 krad (Si)).

The functional behavior of the part is guaranteed up to 300 krad (Si).

25 Reset generation

25.1 Overview

The reset generator implements input reset signal synchronization with glitch filtering and generates the internal reset signal. The input reset signal can be asynchronous.

25.2 Signal definitions and reset values

The signals and their reset values are described in table 185.

*Table 185.*Signal definitions and reset values

Signal name	Type	Function	Active	Reset value
resetn	Input	Reset	Low	

25.3 Timing

The timing waveforms and timing parameters are shown in figure 108 and are defined in table 186.

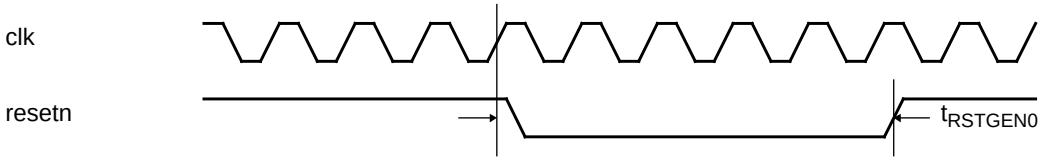


Figure 108. Timing waveforms

*Table 186.*Timing parameters

Name	Parameter	Reference edge	Min	Max	Unit
$t_{RSTGEN0}$	asserted period	-	100	-	ns

Note: The *resetn* input is re-synchronized internally. The signals does not have to meet any setup or hold requirements.

26 AMBA AHB controller with plug&play support

26.1 Overview

The AMBA AHB controller is a combined AHB arbiter, bus multiplexer and slave decoder according to the AMBA 2.0 standard.

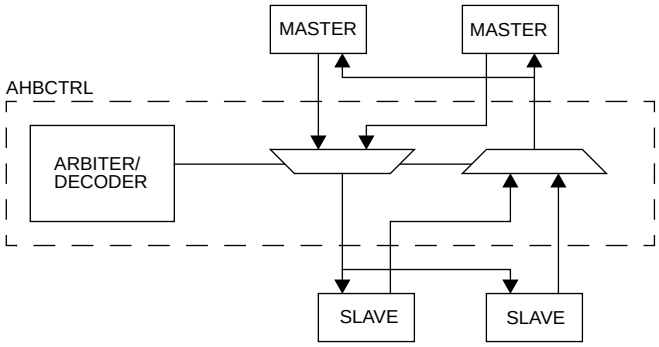


Figure 109. AHB controller block diagram

26.2 Operation

26.2.1 Arbitration

In round-robin mode, priority is rotated one step after each AHB transfer. If no master requests the bus, the last owner will be granted (bus parking).

26.2.2 Decoding

Decoding of AHB slaves is done using the plug&play method explained in the GRLIB User's Manual. A slave can occupy any binary aligned address space with a size of 1 - 4096 Mbyte. A specific I/O area is also decoded, where slaves can occupy 256 byte - 1 Mbyte. The default address of the I/O area is 0xFFF00000. Access to unused addresses will cause an AHB error response.

26.2.3 Plug&play information

The plug&play information is mapped on a read-only address area. By default, the area is mapped on address 0xFFFFF000 - 0xFFFFFFFF. The master information is placed on the first 2 kbyte of the block (0xFFFFF000 - 0xFFFFF800), while the slave information id placed on the second 2 kbyte block. Each unit occupies 32 bytes, which means that the area has place for 64 masters and 64 slaves. The address for masters is thus 0xFFFFF000 + n*32, and 0xFFFFF800 + n*32 for slaves.

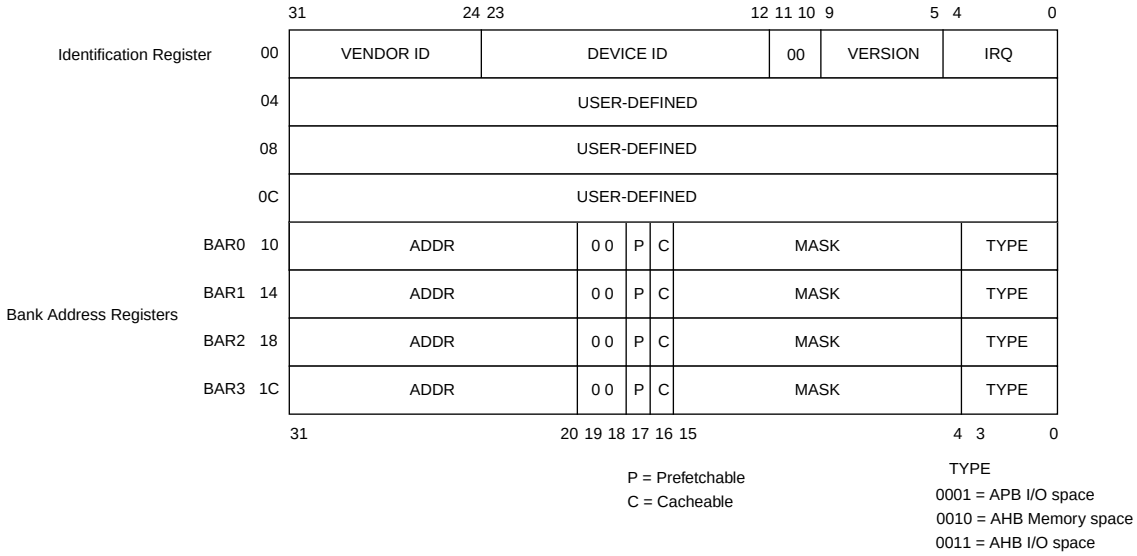


Figure 110. AHB plug&play information record

26.3 Registers

The core does not implement any registers.

27 AMBA AHB/APB bridge with plug&play support

27.1 Overview

The AMBA AHB/APB bridge is a APB bus master according the AMBA 2.0 standard.

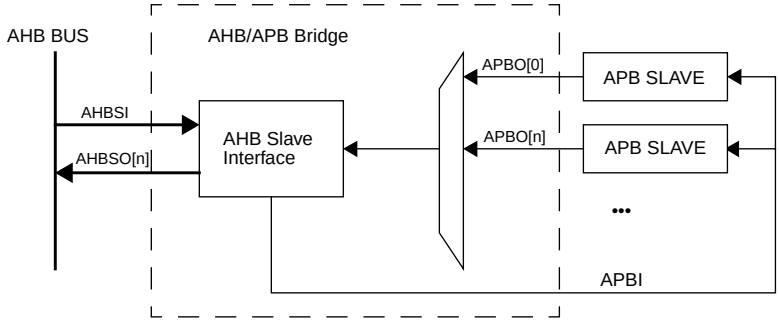


Figure 111. AHB/APB bridge block diagram

27.2 Operation

27.2.1 Decoding

Decoding of APB slaves is done using the plug&play method explained in the GRLIB IP Library User's Manual. A slave can occupy any binary aligned address space with a size of 256 bytes - 1 Mbyte.

27.2.2 Plug&play information

The plug&play information is mapped on a read-only address area at the top 4 kbytes of the bridge address space. Each plug&play block occupies 8 bytes. If the bridge is mapped on AHB address 0x80000000, the address for the plug&play records is thus 0x800FF000 + n*8.

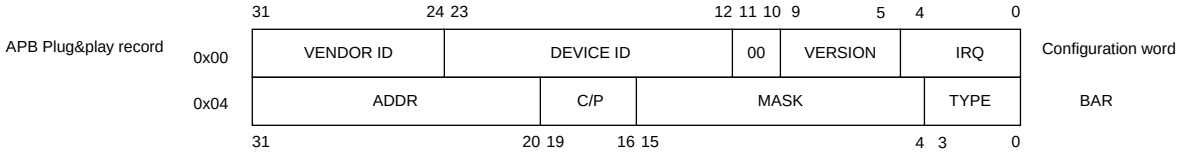


Figure 112. APB plug&play information

28 Electrical description

28.1 Absolute maximum ratings

According to Actel data sheet [RTAX].

28.2 Operating conditions

According to Actel data sheet [RTAX].

28.3 Input voltages, leakage currents and capacitances

According to Actel data sheet [RTAX].

28.4 Output voltages, leakage currents and capacitances

According to Actel data sheet [RTAX].

All output timing parameters are defined at 35 pF capacitive load, except for the LVDS and PCI electrical standards. The Ethernet interface timing parameters are defined at 15 pF capacitive load.

28.5 Clock Input voltages, leakage currents and capacitances

According to Actel data sheet [RTAX].

28.6 Power supplies

According to Actel data sheet [RTAX].

29 Mechanical description

29.1 Component and package

All LEON3F-RTAX configurations are implemented in the ACTEL RTAX2000S FPGA. Configurations 1 - 3 and 7 - 8 are provided in either CQ352 or CG624 packages, as per Actel data sheet [RTAX] and [PACK]. Configurations 4 - 5 are only available in the CG624 package.

29.2 Pin assignment

The pin assignment in table 187 shows the implementation characteristics of each signal according to the Actel data sheet [RTAX], indicating how each pin has been configured in terms of electrical levels, voltage, slew rate, drive capability and internal pull-up or pull-down in the FPGA device.

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
clk	in	314	C12	LVTTL	3.3	-	-		None	-	System clock	All
resetn	in	92	D24	LVTTL	3.3	-	-		None	Low	Reset	All
errorn	out	93	D19	LVTTL	3.3	Low	12	35	Up	Low	IU in error mode	All
dsuen	in	341	G23	LVTTL	3.3	-	-		None	High	DSU enable	All
dsubre	in	342	E24	LVTTL	3.3	-	-		None	High	DSU break	All
dsuact	out	343	H23	LVTTL	3.3	Low	12	35	None	High	DSU active	All
dsutx	out	337	F23	LVTTL	3.3	Low	12	35	None	Low	DSU data transmit	All
dsurx	in	338	E23	LVTTL	3.3	-	-		None	Low	DSU data receive	All
dsutck	in	4	U23	LVCMOS	2.5	-	-		None	-	DSU JTAG clock	-
dsutms	in	5	V23	LVCMOS	2.5	-	-		None	High	DSU JTAG mode	-
dsutdi	in	6	Y24	LVCMOS	2.5	-	-		None	High	DSU JTAG input	-
dsutdo	out	7	V22	LVCMOS	2.5	Low	12	35	None	High	DSU JTAG output	-
address[0]	out	223	P9	LVTTL	3.3	Low	12	35	None	High	Address, LSB	All
address[1]	out	224	N6	LVTTL	3.3	Low	12	35	None	High		All
address[2]	out	225	M6	LVTTL	3.3	Low	12	35	None	High		All
address[3]	out	226	N8	LVTTL	3.3	Low	12	35	None	High		All
address[4]	out	229	N7	LVTTL	3.3	Low	12	35	None	High		All
address[5]	out	230	M4	LVTTL	3.3	Low	12	35	None	High		All
address[6]	out	231	L3	LVTTL	3.3	Low	12	35	None	High		All
address[7]	out	232	M3	LVTTL	3.3	Low	12	35	None	High		All
address[8]	out	235	N10	LVTTL	3.3	Low	12	35	None	High		All
address[9]	out	236	N9	LVTTL	3.3	Low	12	35	None	High		All
address[10]	out	237	K1	LVTTL	3.3	Low	12	35	None	High		All
address[11]	out	238	L1	LVTTL	3.3	Low	12	35	None	High		All
address[12]	out	241	M5	LVTTL	3.3	Low	12	35	None	High		All
address[13]	out	242	L6	LVTTL	3.3	Low	12	35	None	High		All
address[14]	out	243	L5	LVTTL	3.3	Low	12	35	None	High		All
address[15]	out	244	K2	LVTTL	3.3	Low	12	35	None	High		All
address[16]	out	247	L2	LVTTL	3.3	Low	12	35	None	High		All
address[17]	out	248	K4	LVTTL	3.3	Low	12	35	None	High		All

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
address[18]	out	249	L4	LVTTL	3.3	Low	12	35	None	High		All
address[19]	out	250	J3	LVTTL	3.3	Low	12	35	None	High		All
address[20]	out	253	J2	LVTTL	3.3	Low	12	35	None	High		All
address[21]	out	254	J1	LVTTL	3.3	Low	12	35	None	High		All
address[22]	out	255	L7	LVTTL	3.3	Low	12	35	None	High		All
address[23]	out	256	M7	LVTTL	3.3	Low	12	35	None	High		All
address[24]	out	259	M9	LVTTL	3.3	Low	12	35	None	High		All
address[25]	out	260	M8	LVTTL	3.3	Low	12	35	None	High		All
address[26]	out	261	F1	LVTTL	3.3	Low	12	35	None	High		All
address[27]	out	262	G1	LVTTL	3.3	Low	12	35	None	High	Address, MSB	All
data[0]	inout	172	H2	LVTTL	3.3	Low	12	35	None	High	Data, LSB	All
data[1]	inout	173	E2	LVTTL	3.3	Low	12	35	None	High		All
data[2]	inout	179	F2	LVTTL	3.3	Low	12	35	None	High		All
data[3]	inout	180	H4	LVTTL	3.3	Low	12	35	None	High		All
data[4]	inout	181	J4	LVTTL	3.3	Low	12	35	None	High		All
data[5]	inout	182	H5	LVTTL	3.3	Low	12	35	None	High		All
data[6]	inout	183	H6	LVTTL	3.3	Low	12	35	None	High		All
data[7]	inout	184	D2	LVTTL	3.3	Low	12	35	None	High		All
data[8]	inout	187	J6	LVTTL	3.3	Low	12	35	None	High		All
data[9]	inout	188	J5	LVTTL	3.3	Low	12	35	None	High		All
data[10]	inout	189	F3	LVTTL	3.3	Low	12	35	None	High		All
data[11]	inout	190	E3	LVTTL	3.3	Low	12	35	None	High		All
data[12]	inout	193	G4	LVTTL	3.3	Low	12	35	None	High		All
data[13]	inout	194	G3	LVTTL	3.3	Low	12	35	None	High		All
data[14]	inout	195	K8	LVTTL	3.3	Low	12	35	None	High		All
data[15]	inout	196	L8	LVTTL	3.3	Low	12	35	None	High		All
data[16]	inout	199	W2	LVTTL	3.3	Low	12	35	None	High		All
data[17]	inout	200	Y2	LVTTL	3.3	Low	12	35	None	High		All
data[18]	inout	201	R6	LVTTL	3.3	Low	12	35	None	High		All
data[19]	inout	202	T6	LVTTL	3.3	Low	12	35	None	High		All
data[20]	inout	205	T7	LVTTL	3.3	Low	12	35	None	High		All
data[21]	inout	206	U7	LVTTL	3.3	Low	12	35	None	High		All
data[22]	inout	207	V2	LVTTL	3.3	Low	12	35	None	High		All
data[23]	inout	208	R4	LVTTL	3.3	Low	12	35	None	High		All
data[24]	inout	211	T4	LVTTL	3.3	Low	12	35	None	High		All
data[25]	inout	212	R3	LVTTL	3.3	Low	12	35	None	High		All
data[26]	inout	213	R5	LVTTL	3.3	Low	12	35	None	High		All
data[27]	inout	214	AA1	LVTTL	3.3	Low	12	35	None	High		All
data[28]	inout	217	AB1	LVTTL	3.3	Low	12	35	None	High		All
data[29]	inout	218	R8	LVTTL	3.3	Low	12	35	None	High		All
data[30]	inout	219	T8	LVTTL	3.3	Low	12	35	None	High		All
data[31]	inout	220	W1	LVTTL	3.3	Low	12	35	None	High	Data, MSB	All

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
cb[0]	inout	160	V4	LVTTL	3.3	Low	12	35	None	High	Checkbits, LSB	All
cb[1]	inout	161	Y5	LVTTL	3.3	Low	12	35	None	High		All
cb[2]	inout	164	W5	LVTTL	3.3	Low	12	35	None	High		All
cb[3]	inout	165	U6	LVTTL	3.3	Low	12	35	None	High		All
cb[4]	inout	166	U5	LVTTL	3.3	Low	12	35	None	High		All
cb[5]	inout	167	U3	LVTTL	3.3	Low	12	35	None	High		All
cb[6]	inout	170	T2	LVTTL	3.3	Low	12	35	None	High		All
cb[7]	inout	171	U2	LVTTL	3.3	Low	12	35	None	High	Checkbits, MSB	All
cb[8]	inout	52	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[9]	inout	53	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[10]	inout	54	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[11]	inout	55	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[12]	inout	58	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[13]	inout	137	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[14]	inout	313	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
cb[15]	inout	319	-	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	7
ramsn[0]	out	146	M2	LVTTL	3.3	Low	12	35	None	Low	SRAM chip select	All
ramsn[1]	out	147	P4	LVTTL	3.3	Low	12	35	None	Low		All
ramsn[2]	out	152	P1	LVTTL	3.3	Low	12	35	None	Low		All
ramsn[3]	out	153	P6	LVTTL	3.3	Low	12	35	None	Low		All
ramsn[4]	out	142	P5	LVTTL	3.3	Low	12	35	None	Low		4 - 8
ramoen[0]	out	154	P3	LVTTL	3.3	Low	12	35	None	Low	SRAM output enable	All
ramoen[1]	out	155	N4	LVTTL	3.3	Low	12	35	None	Low		All
ramoen[2]	out	158	U1	LVTTL	3.3	Low	12	35	None	Low		All
ramoen[3]	out	159	T1	LVTTL	3.3	Low	12	35	None	Low		All
ramoen[4]	out	143	R2	LVTTL	3.3	Low	12	35	None	Low		4 - 8
rwen[0]	out	272	H3	LVTTL	3.3	Low	12	35	None	Low	SRAM write strobe 1)	All
rwen[1]	out	271	G2	LVTTL	3.3	Low	12	35	None	Low		All
rwen[2]	out	270	E1	LVTTL	3.3	Low	12	35	None	Low		All
rwen[3]	out	269	D1	LVTTL	3.3	Low	12	35	None	Low		All
ramben[0]	out	282	Y1	LVTTL	3.3	Low	12	35	None	Low	SRAM read/write byte enable1)	All
ramben[1]	out	281	P2	LVTTL	3.3	Low	12	35	None	Low		All
ramben[2]	out	278	K7	LVTTL	3.3	Low	12	35	None	Low		All
ramben[3]	out	277	K6	LVTTL	3.3	Low	12	35	None	Low		All
oen	out	283	N1	LVTTL	3.3	Low	12	35	None	Low	Output enable	All
writen	out	284	P7	LVTTL	3.3	Low	12	35	None	Low	Write strobe	All
read	out	289	R7	LVTTL	3.3	Low	12	35	None	High	Read strobe	All
iosn	out	290	M1	LVTTL	3.3	Low	12	35	None	Low	IO area chip select	All
romsn[0]	out	288	N2	LVTTL	3.3	Low	12	35	None	Low	PROM chip select	All
romsn[1]	out	287	R1	LVTTL	3.3	Low	12	35	None	Low		All
romsn[2]	out	276	H24	LVTTL	3.3	Low	12	35	None	Low		All

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
romsn[3]	out	275	M19	LVTTL	3.3	Low	12	35	None	Low		All
brdyn	in	296	N3	LVTTL	3.3	-	-		Up	Low	Bus ready	All
bexcn	in	295	P8	LVTTL	3.3	-	-		Up	Low	Bus exception	All
sa[0]	out	-	D10	LVTTL	3.3	Low	12	35	None	High	SDRAM address, LSB	4, 5, 6
sa[1]	out	-	D9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[2]	out	-	A7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[3]	out	-	A6	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[4]	out	-	G9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[5]	out	-	G8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[6]	out	-	A22	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[7]	out	-	A21	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[8]	out	-	F16	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[9]	out	-	G17	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[10]	out	-	H17	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[11]	out	-	B17	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[12]	out	-	B16	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[13]	out	-	C17	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sa[14]	out	-	B18	LVTTL	3.3	Low	12	35	None	High	SDRAM address, MSB	4, 5, 6
sa[15]	out	-	H18	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[0]	inout	-	D7	LVTTL	3.3	Low	12	35	None	High	SDRAM data, LSB	4, 5, 6
sd[1]	inout	-	E7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[2]	inout	-	G7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[3]	inout	-	G6	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[4]	inout	-	B5	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[5]	inout	-	B4	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[6]	inout	-	C7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[7]	inout	-	F8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[8]	inout	-	F7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[9]	inout	-	H8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[10]	inout	-	H7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[11]	inout	-	J8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[12]	inout	-	J7	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[13]	inout	-	B6	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[14]	inout	-	E9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[15]	inout	-	D8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[16]	inout	-	B12	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[17]	inout	-	C13	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[18]	inout	-	G15	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[19]	inout	-	B14	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[20]	inout	-	B13	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[21]	inout	-	H13	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[22]	inout	-	D14	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[23]	inout	-	C14	LVTTL	3.3	Low	12	35	None	High		4, 5, 6

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
sd[24]	inout	-	A16	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[25]	inout	-	A15	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[26]	inout	-	H15	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[27]	inout	-	E15	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[28]	inout	-	F15	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[29]	inout	-	A17	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[30]	inout	-	G16	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
sd[31]	inout	-	H16	LVTTL	3.3	Low	12	35	None	High	SDRAM data, MSB	4, 5, 6
sd[32]	inout	-	B7	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[33]	inout	-	F10	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[34]	inout	-	F9	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[35]	inout	-	C11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[36]	inout	-	B8	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[37]	inout	-	H10	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[38]	inout	-	H9	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[39]	inout	-	A9	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[40]	inout	-	B9	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[41]	inout	-	B11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[42]	inout	-	B10	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[43]	inout	-	E11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[44]	inout	-	F11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[45]	inout	-	D12	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[46]	inout	-	D11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[47]	inout	-	A11	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[48]	inout	-	J18	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[49]	inout	-	J19	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[50]	inout	-	B20	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[51]	inout	-	B19	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[52]	inout	-	E17	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[53]	inout	-	F17	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[54]	inout	-	B22	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[55]	inout	-	B21	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[56]	inout	-	G18	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[57]	inout	-	G19	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[58]	inout	-	C19	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[59]	inout	-	C18	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[60]	inout	-	D18	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[61]	inout	-	D17	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[62]	inout	-	C21	LVTTL	3.3	Low	12	35	None	High	{reserved}	-
sd[63]	inout	-	C20	LVTTL	3.3	Low	12	35	None	High	{reserved}	-

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
scb[0]	inout	-	B7	LVTTL	3.3	Low	12	35	None	High	SDRAM check bits	4, 5, 6
scb[1]	inout	-	F10	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[2]	inout	-	F9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[3]	inout	-	C11	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[4]	inout	-	B8	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[5]	inout	-	H10	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[6]	inout	-	H9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[7]	inout	-	A9	LVTTL	3.3	Low	12	35	None	High		4, 5, 6
scb[8]	inout	-	B9	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[9]	inout	-	B11	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[10]	inout	-	B10	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[11]	inout	-	E11	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[12]	inout	-	F11	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[13]	inout	-	D12	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[14]	inout	-	D11	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
scb[15]	inout	-	A11	LVTTL	3.3	Low	12	35	None	High	Reed-Solomon option only 2)	4, 5
sdcsn[0]	out	300	A10	LVTTL	3.3	Low	12	35	None	Low	SDRAM chip select	4 - 8
sdcsn[1]	out	299	J13	LVTTL	3.3	Low	12	35	None	Low		4 - 8
sdwen	out	305	G11	LVTTL	3.3	Low	12	35	None	Low	SDRAM write enable	4 - 8
sdrasn	out	306	H11	LVTTL	3.3	Low	12	35	None	Low	SDRAM row strobe	4 - 8
sdcasn	out	122	K13	LVTTL	3.3	Low	12	35	None	Low	SDRAM column strobe	4 - 8
sddqm[0]	out	123	C9	LVTTL	3.3	Low	12	35	None	Low	SDRAM data mask 1)	4 - 8
sddqm[1]	out	128	C8	LVTTL	3.3	Low	12	35	None	Low		4 - 8
sddqm[2]	out	129	A20	LVTTL	3.3	Low	12	35	None	Low		4 - 8
sddqm[3]	out	136	A19	LVTTL	3.3	Low	12	35	None	Low		4 - 8
sddqm[4]	out	-	A5	LVTTL	3.3	Low	12	35	None	Low		4, 5, 6
sddqm[5]	out	-	A4	LVTTL	3.3	Low	12	35	None	Low	Reed-Solomon option only 2)	4, 5, 7
sddqm[6]	out	-	D16	LVTTL	3.3	Low	12	35	None	Low	{reserved}	-
sddqm[7]	out	-	D15	LVTTL	3.3	Low	12	35	None	Low	{reserved}	-
sdclk	out	-	G12	LVTTL	3.3	Low	12	35	None	-	{reserved}	-
sdclkfb	in	-	G13	LVTTL	3.3	-	-		None	-	{reserved}	-
txd[1]	out	335	L19	LVTTL	3.3	Low	12	35	None	Low	UART data transmit	All
rxd[1]	in	336	J23	LVTTL	3.3	-	-		None	Low	UART data receive	All
rtsn[1]	out	331	J20	LVTTL	3.3	Low	12	35	None	Low	Ready to Send	All
ctsn[1]	in	332	J21	LVTTL	3.3	-	-		None	Low	Clear to Send	All
txd[2]	out	325	K20	LVTTL	3.3	Low	12	35	None	Low	UART data transmit	All
rxd[2]	in	326	D25	LVTTL	3.3	-	-		None	Low	UART data receive	All
rtsn[2]	out	323	E25	LVTTL	3.3	Low	12	35	None	Low	Ready to Send	All
ctsn[2]	in	324	K19	LVTTL	3.3	-	-		None	Low	Clear to Send	All
wdogn	out	86	D20	LVTTL	3.3	Low	12	35	Up	Low	Watchdog timeout	All

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
gpio[0]	inout	94	F24	LVTTL	3.3	Low	12	35	None	High	General purpose I/O	All
gpio[1]	inout	95	G24	LVTTL	3.3	Low	12	35	None	High	{interrupt # 1}	All
gpio[2]	inout	98	K18	LVTTL	3.3	Low	12	35	None	High	{interrupt # 2}	All
gpio[3]	inout	99	L18	LVTTL	3.3	Low	12	35	None	High	{interrupt # 3}	All
gpio[4]	inout	100	H22	LVTTL	3.3	Low	12	35	None	High	{interrupt # 4}	All
gpio[5]	inout	101	J22	LVTTL	3.3	Low	12	35	None	High	{interrupt # 5}	All
gpio[6]	inout	104	G22	LVTTL	3.3	Low	12	35	None	High	{interrupt # 6}	All
gpio[7]	inout	105	M17	LVTTL	3.3	Low	12	35	None	High	{interrupt # 7}	All
gpio[8]	inout	106	H20	LVTTL	3.3	Low	12	35	None	High	{interrupt # 8}	All
gpio[9]	inout	107	H19	LVTTL	3.3	Low	12	35	None	High	{interrupt # 9}	All
gpio[10]	inout	110	E18	LVTTL	3.3	Low	12	35	None	High	{interrupt # 10}	All
gpio[11]	inout	111	F18	LVTTL	3.3	Low	12	35	None	High	{interrupt # 11}	All
gpio[12]	inout	112	G21	LVTTL	3.3	Low	12	35	None	High	{interrupt # 12}	All
gpio[13]	inout	113	G20	LVTTL	3.3	Low	12	35	None	High	{interrupt # 13}	All
gpio[14]	inout	118	F20	LVTTL	3.3	Low	12	35	None	High	{interrupt # 14}	All
gpio[15]	inout	119	F19	LVTTL	3.3	Low	12	35	None	High	{non-maskable interrupt # 15}	All
gpio[16]	inout	82	N19	LVTTL	3.3	Low	12	35	None	High		7
gpio[17]	inout	78	M21	LVTTL	3.3	Low	12	35	None	High		7
gpio[18]	inout	79	M20	LVTTL	3.3	Low	12	35	None	High		7
gpio[19]	inout	85	N17	LVTTL	3.3	Low	12	35	None	High		7
gpio[20]	inout	83	K24	LVTTL	3.3	Low	12	35	None	High		7
gpio[21]	inout	84	L24	LVTTL	3.3	Low	12	35	None	High		7
gpio[22]	inout	72	L20	LVTTL	3.3	Low	12	35	None	High		7
gpio[23]	inout	70	L21	LVTTL	3.3	Low	12	35	None	High		7
gpio[24]	inout	71	F25	LVTTL	3.3	Low	12	35	None	High		7
gpio[25]	inout	77	G25	LVTTL	3.3	Low	12	35	None	High		7
gpio[26]	inout	73	K22	LVTTL	3.3	Low	12	35	None	High		7
gpio[27]	inout	76	L22	LVTTL	3.3	Low	12	35	None	High		7
gpio[28]	inout	64	N16	LVTTL	3.3	Low	12	35	None	High		7
gpio[29]	inout	60	L23	LVTTL	3.3	Low	12	35	None	High		7
gpio[30]	inout	61	V3	LVTTL	3.3	Low	12	35	None	High		7
gpio[31]	inout	67	AA2	LVTTL	3.3	Low	12	35	None	High		7

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
spw_clk	in	320	G14	LVTTL	3.3	-	-		-	-	SpaceWire clock	2,4,5,7
spw_rxd[0]	in	43	P19	LVDS	2.5	-	-		-	High	SpaceWire data	2,4,5,7
spw_rxdn[0]	in	42	P20	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_rxs[0]	in	41	P23	LVDS	2.5	-	-		-	High	SpaceWire strobe	2,4,5,7
spw_rxsn[0]	in	40	R23	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_txd[0]	out	37	P25	LVDS	2.5	-	-		-	High	SpaceWire data	2,4,5,7
spw_txdn[0]	out	36	R25	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_txs[0]	out	35	P22	LVDS	2.5	-	-		-	High	SpaceWire strobe	2,4,5,7
spw_txsn[0]	out	34	R22	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_rxd[1]	in	31	R18	LVDS	2.5	-	-		-	High	SpaceWire data	2,4,5,7
spw_rxdn[1]	in	30	T18	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_rxs[1]	in	29	R24	LVDS	2.5	-	-		-	High	SpaceWire strobe	2,4,5,7
spw_rxsn[1]	in	28	T24	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_txd[1]	out	25	T25	LVDS	2.5	-	-		-	High	SpaceWire data	2,4,5,7
spw_txdn[1]	out	24	U25	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_txs[1]	out	23	R20	LVDS	2.5	-	-		-	High	SpaceWire strobe	2,4,5,7
spw_txsn[1]	out	22	T20	LVDS	2.5	-	-		-	Low	(LVDS option only)	2,4,5,7
spw_rxd[2]	in	19	T19	LVDS	2.5	-	-		-	High	SpaceWire data	4
spw_rxdn[2]	in	18	U19	LVDS	2.5	-	-		-	Low	(LVDS option only)	4
spw_rxs[2]	in	17	W25	LVDS	2.5	-	-		-	High	SpaceWire strobe	4
spw_rxsn[2]	in	16	Y25	LVDS	2.5	-	-		-	Low	(LVDS option only)	4
spw_txd[2]	out	13	AA25	LVDS	2.5	-	-		-	High	SpaceWire data	4
spw_txdn[2]	out	12	AB25	LVDS	2.5	-	-		-	Low	(LVDS option only)	4
spw_txs[2]	out	11	U20	LVDS	2.5	-	-		-	High	SpaceWire strobe	4
spw_txsn[2]	out	10	V20	LVDS	2.5	-	-		-	Low	(LVDS option only)	4
spw_rxd[3]	in	-	U23	LVDS	2.5	-	-		-	High	{reserved}	-
spw_rxdn[3]	in	-	U24	LVDS	2.5	-	-		-	Low	{reserved}	-
spw_rxs[3]	in	-	Y24	LVDS	2.5	-	-		-	High	{reserved}	-
spw_rxsn[3]	in	-	AA24	LVDS	2.5	-	-		-	Low	{reserved}	-
spw_txd[3]	out	-	V24	LVDS	2.5	-	-		-	High	{reserved}	-
spw_txdn[3]	out	-	V23	LVDS	2.5	-	-		-	Low	{reserved}	-
spw_txs[3]	out	-	U22	LVDS	2.5	-	-		-	High	{reserved}	-
spw_txsn[3]	out	-	V22	LVDS	2.5	-	-		-	Low	{reserved}	-

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
cantx[0]	out	49	M25	LVTTL	3.3	Low	12	35	None	Low	CAN 0 tx data	1, 5
canrx[0]	in	48	N25	LVTTL	3.3	-	-		None	Low	CAN 0 rx data	1, 5
canen[0]	out	-	-	LVTTL	3.3	Low	12	35	None	-	{reserved}	-
cantx[1]	out	47	M18	LVTTL	3.3	Low	12	35	None	Low	CAN 1 tx data	-
canrx[1]	in	46	J24	LVTTL	3.3	-	-		None	Low	CAN 1 rx data	-
canen[1]	out	-	-	LVTTL	3.3	Low	12	35	None	-	{reserved}	-
busainen[0]	out	82	N23	LVTTL	3.3	Low	12	35	None	High	Mil-Std-1553	1, 3
busainn[0]	in	78	N22	LVTTL	3.3	-	-		None	Low	bus 0 nominal	1, 3
busainp[0]	in	79	M22	LVTTL	3.3	-	-		None	High		1, 3
busaoutin[0]	out	85	P17	LVTTL	3.3	Low	12	35	None	High		1, 3
busaoutn[0]	out	83	M23	LVTTL	3.3	Low	12	35	None	Low		1, 3
busaoutp[0]	out	84	P18	LVTTL	3.3	Low	12	35	None	High		1, 3
busbinen[0]	out	72	N24	LVTTL	3.3	Low	12	35	None	High	Mil-Std-1553	1, 3
busbinn[0]	in	70	N18	LVTTL	3.3	-	-		None	Low	bus 0 redundant	1, 3
busbinp[0]	in	71	J25	LVTTL	3.3	-	-		None	High		1, 3
busboutin[0]	out	77	K25	LVTTL	3.3	Low	12	35	None	High		1, 3
busboutn[0]	out	73	M24	LVTTL	3.3	Low	12	35	None	Low		1, 3
busboutp[0]	out	76	L25	LVTTL	3.3	Low	12	35	None	High		1, 3
busainen[1]	out	64	N17	LVTTL	3.3	Low	12	35	None	High	Mil-Std-1553	-
busainn[1]	in	60	L24	LVTTL	3.3	-	-		None	Low	bus 1 nominal	-
busainp[1]	in	61	K24	LVTTL	3.3	-	-		None	High		-
busaoutin[1]	out	67	N19	LVTTL	3.3	Low	12	35	None	High		-
busaoutn[1]	out	65	M20	LVTTL	3.3	Low	12	35	None	Low		-
busaoutp[1]	out	66	M21	LVTTL	3.3	Low	12	35	None	High		-
busbinen[1]	out	54	G25	LVTTL	3.3	Low	12	35	None	High	Mil-Std-1553	-
busbinn[1]	in	52	L22	LVTTL	3.3	-	-		None	Low	bus 1 redundant	-
busbinp[1]	in	53	K22	LVTTL	3.3	-	-		None	High		-
busboutin[1]	out	59	L20	LVTTL	3.3	Low	12	35	None	High		-
busboutn[1]	out	55	F25	LVTTL	3.3	Low	12	35	None	Low		-
busboutp[1]	out	58	L21	LVTTL	3.3	Low	12	35	None	High		-

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
pci_clk	in	-	G14	PCI	3.3						PCI clock	6
pci_rst	inout	-	AB8	PCI	3.3						PCI reset	6
pci_frame	inout	-	AC9	PCI	3.3							6
pci_irdy	inout	-	AA9	PCI	3.3							6
pci_trdy	inout	-	AD6	PCI	3.3							6
pci_stop	inout	-	AD5	PCI	3.3							6
pci_idsel	in	-	U8	PCI	3.3							6
pci_devsel	inout	-	AB9	PCI	3.3							6
pci_par	inout	-	V10	PCI	3.3							6
pci_perr	inout	-	V9	PCI	3.3							6
pci_serr	inout	-	AD8	PCI	3.3							6
pci_req	inout	-	AE11	PCI	3.3							6
pci_gnt	in	-	AE10	PCI	3.3							6
pci_host	in	-	U13	PCI	3.3							6
pci_int[0]	inout	-	AA3	PCI	3.3						PCI interrupt A	6
pci_int[1]	inout	-	W3	PCI	3.3						PCI interrupt B	6
pci_int[2]	inout	-	AB2	PCI	3.3						PCI interrupt C	6
pci_int[3]	inout	-	W4	PCI	3.3						PCI interrupt D	6
pci_cbe[0]	inout	-	W8	PCI	3.3						PCI byte enable, LSB	6
pci_cbe[1]	inout	-	W9	PCI	3.3							6
pci_cbe[2]	inout	-	AE4	PCI	3.3							6
pci_cbe[3]	inout	-	AE5	PCI	3.3						PCI byte enable, MSB	6
pci_ad[0]	inout	-	W11	PCI	3.3						PCI address/data, LSB	6
pci_ad[1]	inout	-	W12	PCI	3.3							6
pci_ad[2]	inout	-	AA11	PCI	3.3							6
pci_ad[3]	inout	-	Y11	PCI	3.3							6
pci_ad[4]	inout	-	AE9	PCI	3.3							6
pci_ad[5]	inout	-	AE6	PCI	3.3							6
pci_ad[6]	inout	-	AE7	PCI	3.3							6
pci_ad[7]	inout	-	Y10	PCI	3.3							6
pci_ad[8]	inout	-	W10	PCI	3.3							6
pci_ad[9]	inout	-	T13	PCI	3.3							6
pci_ad[10]	inout	-	AB10	PCI	3.3							6
pci_ad[11]	inout	-	AB11	PCI	3.3							6
pci_ad[12]	inout	-	AD9	PCI	3.3							6
pci_ad[13]	inout	-	AD10	PCI	3.3							6
pci_ad[14]	inout	-	V11	PCI	3.3							6
pci_ad[15]	inout	-	AD7	PCI	3.3							6
pci_ad[16]	inout	-	AC8	PCI	3.3							6
pci_ad[17]	inout	-	AB7	PCI	3.3							6
pci_ad[18]	inout	-	AC7	PCI	3.3							6
pci_ad[19]	inout	-	AA8	PCI	3.3							6
pci_ad[20]	inout	-	Y8	PCI	3.3							6

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
pci_ad[21]	inout	-	V8	PCI	3.3							6
pci_ad[22]	inout	-	V7	PCI	3.3							6
pci_ad[23]	inout	-	Y7	PCI	3.3							6
pci_ad[24]	inout	-	W7	PCI	3.3							6
pci_ad[25]	inout	-	AC5	PCI	3.3							6
pci_ad[26]	inout	-	AC6	PCI	3.3							6
pci_ad[27]	inout	-	Y6	PCI	3.3							6
pci_ad[28]	inout	-	W6	PCI	3.3							6
pci_ad[29]	inout	-	AB6	PCI	3.3							6
pci_ad[30]	inout	-	AA6	PCI	3.3							6
pci_ad[31]	inout	-	Y3	PCI	3.3						PCI address/data, MSB	6
pci_arb_req[0]	in	-	AD14	LVTTL	3.3	-	-		Up	Low	PCI arbiter request	6
pci_arb_req[1]	in	-	AC14	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[2]	in	-	W14	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[3]	in	-	W15	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[4]	in	-	AC13	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[5]	in	-	AC21	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[6]	in	-	AC20	LVTTL	3.3	-	-		Up	Low		6
pci_arb_req[7]	in	-	AD13	LVTTL	3.3	-	-		Up	Low		6
pci_arb_gnt[0]	out	-	AE15	PCI	3.3	-	-		-	Low	PCI arbiter grant	6
pci_arb_gnt[1]	out	-	AE16	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[2]	out	-	W16	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[3]	out	-	AE14	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[4]	out	-	V15	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[5]	out	-	Y20	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[6]	out	-	W20	PCI	3.3	-	-		-	Low		6
pci_arb_gnt[7]	out	-	V16	PCI	3.3	-	-		-	Low		6

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
e0mdio	inout	60	U4	LVTTL	3.3	Low	12	15	None		10/100 Mbit Ethernet	6, 8
e0mdc	out	61	V6	LVTTL	3.3	Low	12	15	None			6, 8
e0mdcint	in	79	AA2	LVTTL	3.3	-	-		None			6, 8
e0rx_clk	in	64	N19	LVTTL	3.3	-	-		None			6, 8
e0rx_dv	in	65	M21	LVTTL	3.3	-	-		None			6, 8
e0rx_crs	in	66	M20	LVTTL	3.3	-	-		None			6, 8
e0rx_er	in	67	N17	LVTTL	3.3	-	-		None			6, 8
e0rx_col	in	70	K24	LVTTL	3.3	-	-		None			6, 8
e0rxd[0]	in	71	L24	LVTTL	3.3	-	-		None			6, 8
e0rxd[1]	in	72	L20	LVTTL	3.3	-	-		None			6, 8
e0rxd[2]	in	73	L21	LVTTL	3.3	-	-		None			6, 8
e0rxd[3]	in	76	F25	LVTTL	3.3	-	-		None			6, 8
e0tx_clk	in	77	G25	LVTTL	3.3	-	-		None			6, 8
e0tx_en	out	78	V3	LVTTL	3.3	Low	12	15	None			6, 8
e0txd[0]	out	82	K22	LVTTL	3.3	Low	12	15	None			6, 8
e0txd[1]	out	83	L22	LVTTL	3.3	Low	12	15	None			6, 8
e0txd[2]	out	84	N16	LVTTL	3.3	Low	12	15	None			6, 8
e0txd[3]	out	85	L23	LVTTL	3.3	Low	12	15	None			6, 8
e1mdio	inout	35	U22	LVTTL	3.3	Low	12	15	None		10/100 Mbit Ethernet	8
e1mdc	out	37	V21	LVTTL	3.3	Low	12	15	None			8
e1mdcint	in	16	P21	LVTTL	3.3	-	-		None			8
e1rx_clk	in	31	U21	LVTTL	3.3	-	-		None			8
e1rx_dv	in	29	AA23	LVTTL	3.3	-	-		None			8
e1rx_crs	in	30	Y23	LVTTL	3.3	-	-		None			8
e1rx_er	in	28	W22	LVTTL	3.3	-	-		None			8
e1rx_col	in	22	V24	LVTTL	3.3	-	-		None			8
e1rxd[0]	in	24	W23	LVTTL	3.3	-	-		None			8
e1rxd[1]	in	23	Y22	LVTTL	3.3	-	-		None			8
e1rxd[2]	in	25	Y21	LVTTL	3.3	-	-		None			8
e1rxd[3]	in	19	N20	LVTTL	3.3	-	-		None			8
e1tx_clk	in	17	P24	LVTTL	3.3	-	-		None			8
e1tx_en	out	18	R19	LVTTL	3.3	Low	12	15	None			8
e1txd[0]	out	10	R21	LVTTL	3.3	Low	12	15	None			8
e1txd[1]	out	12	T22	LVTTL	3.3	Low	12	15	None			8
e1txd[2]	out	11	W24	LVTTL	3.3	Low	12	15	None			8
e1txd[3]	out	13	AB24	LVTTL	3.3	Low	12	15	None			8

*Table 187.*Pin assignment

Name	I/O	Pin CQ352	Pin CG624	Level	Volt.	Slew	Drive	Load [pF]	Pull	Pol-arity	Note	CID
fifoffn	in	-	V6	LVTTL	3.3	-	-		High	low	FIFO Full Flag	-
fifohfn	in	-	-	LVTTL	3.3	-	-		High	low	FIFO Half Full Flag	-
fifoefn	in	-	U4	LVTTL	3.3	-	-		High	low	FIFO Empty Flag	-
fifowen	out	-	V3	LVTTL	3.3	Low	12	35	None	low	FIFO Write strobe	-
fiforen	out	-	AA2	LVTTL	3.3	Low	12	35	None	low	FIFO Read strobe	-
fifod[15]	inout	-	M18	LVTTL	3.3	Low	12	35	None		FIFO Data, MSB	-
fifod[14]	inout	-	J24	LVTTL	3.3	Low	12	35	None			-
fifod[13]	inout	-	L23	LVTTL	3.3	Low	12	35	None			-
fifod[12]	inout	-	N16	LVTTL	3.3	Low	12	35	None			-
fifod[11]	inout	-	L22	LVTTL	3.3	Low	12	35	None			-
fifod[10]	inout	-	K22	LVTTL	3.3	Low	12	35	None			-
fifod[9]	inout	-	G25	LVTTL	3.3	Low	12	35	None			-
fifod[8]	inout	-	F25	LVTTL	3.3	Low	12	35	None			-
fifod[7]	inout	-	L21	LVTTL	3.3	Low	12	35	None			-
fifod[6]	inout	-	L20	LVTTL	3.3	Low	12	35	None			-
fifod[5]	inout	-	L24	LVTTL	3.3	Low	12	35	None			-
fifod[4]	inout	-	K24	LVTTL	3.3	Low	12	35	None			-
fifod[3]	inout	-	N17	LVTTL	3.3	Low	12	35	None			-
fifod[2]	inout	-	M20	LVTTL	3.3	Low	12	35	None			-
fifod[1]	inout	-	M21	LVTTL	3.3	Low	12	35	None			-
fifod[0]	inout	-	N19	LVTTL	3.3	Low	12	35	None		FIFO Data, LSB	-
fifop[1]	inout	-	N25	LVTTL	3.3	Low	12	35	None		FIFO Parity, over [15.:8]	-
fifop[0]	inout	-	M25	LVTTL	3.3	Low	12	35	None		FIFO Parity, over [7.:0]	-

Pins not used in a given configuration (see CID column) should be left unconnected (or tied to ground), including the pins marked as *{reserved}*. The only exception is if the unused pin is mapped on a hardwired clock input or a routed clock input, which should then be tied to ground:

CQ352 package: *spw_clk, ramsn[4], ramoen[4], sdcsn[0], sdcsn[1], sdwen, sdrasn, sdcasn, sddqm[0], sddqm[1], sddqm[2], sddqm[3], cb[13], cb[14], cb[15]*

CG624 package: *spw_clk, sd[17], sd[18], sd[19], sd[20], sdclk, sdclkfb, pci_arb_req[2], pci_arb_req[3], pci_arb_req[4], pci_arb_req[7]*

For the usage of all other pins, please refer to the specific pins of the selected package, as described in the next sections and the Actel data sheet [RTAX].

Note 1: Refer to signal definitions of each memory controller for detailed usage of these signals.

Note 2: Signals only used in custom configurations with Reed-Solomon protected SDRAM.

29.3 RTAX2000S specific pins - CQ352 package

The Actel RTAX2000S FPGA device has special pins that need to be correctly connected on the printed circuit board, as shown in table 188. Please refer to the Actel data sheet [RTAX] for details.

*Table 188.*RTAX2000S special pins - CQ352 package

Name	Pin CQ352	Note	CID
GND	1, 9, 15, 21, 27, 33, 39, 45, 51, 57, 63, 69, 75, 81, 88, 89, 97, 103, 109, 115, 121, 133, 145, 151, 157, 163, 169, 176, 177, 186, 192, 198, 204, 210, 216, 222, 228, 234, 240, 246, 252, 258, 264, 265, 274, 280, 286, 292, 298, 310, 322, 330, 334, 340, 345, 352	Low supply voltage, ground	All
VCCA	3, 14, 32, 56, 74, 87, 102, 114, 150, 162, 175, 191, 209, 233, 251, 263, 279, 291, 329, 339	1.5 V supply voltage for array	All
VCCDA	2, 44, 90, 91, 116, 117, 130, 131, 132, 148, 149, 174, 178, 221, 266, 268, 293, 294, 307, 308, 309, 327, 328, 346	3.3 V supply voltage for I/O differential amplifier and JTAG and probe interfaces.	All
VCCIB0	321, 333, 344	3.3 V supply voltage for I/O	All
VCCIB1	273, 285, 297	3.3 V supply voltage for I/O	All
VCCIB2	227, 239, 245, 257	3.3 V supply voltage for I/O	All
VCCIB3	185, 197, 203, 215	3.3 V supply voltage for I/O	All
VCCIB4	144, 156, 168	3.3 V supply voltage for I/O	All
VCCIB5	96, 108, 120	3.3 V supply voltage for I/O	All
VCCIB6	50, 62, 68, 80	3.3 V supply voltage for I/O	All
VCCIB7	8, 20, 26, 38	3.3 V supply voltage for I/O	1,3,6,8
VCCIB7	8, 20, 26, 38	2.5 V supply voltage for I/O	2,4,5,7
VPUMP	267	Voltage External Pump. In normal operation, using the internal charge pump, should be tied to ground.	All
TRST	351	JTAG Test Clock. Actel recommends this pin be hardwired to ground for flight.	All
TCK	349	JTAG Test Clock. Actel recommends this pin be hardwired to ground. Must not be left unconnected.	All
TDI	348	JTAG Test Data Input. Actel recommends this pin be hardwired to VCCDA, or left unconnected.	All
TDO	347	JTAG Test Data Output. Must be left unconnected.	All
TMS	350	JTAG Test Mode Select. Actel recommends that this pin be hardwired to VCCDA, or left unconnected.	All
PRA	312	Test Probe. The pins' probe capabilities are disabled to protect programmed design confidentiality. These pins must be left unconnected.	All
PRB	311		All
PRC	135		All
PRD	134		All
NC	124, 125, 126, 127, 138, 139, 140, 141, 301, 302, 303, 304, 315, 316, 317, 318	No Connection. Pins are not connected to circuitry within the device. These pins can be driven to any voltage or can be left floating with no effect on the operation of the device.	All
CLK*	122, 123, 128, 129, 136, 137, 142, 143	Global clocks. When pins are unused, Actel recommends they are tied to known state, preferably ground.	All
HCLK*	299, 300, 305, 306, 313, 314, 319, 320	Hardwired clocks. When pins are unused, it is recommended that they are tied to ground.	All

29.4 RTAX2000S specific pins - CG624 package

The Actel RTAX2000S FPGA device has special pins that need to be correctly connected on the printed circuit board, as shown in table 188. Please refer to the Actel data sheet [RTAX] for details.

*Table 189.*RTAX2000S special pins - CG624 package

Name	Pin CG624	Note	CID
GND	A18, A24, A25, A2, A8, AA10, AA16, AA18, AA21, AA5, AB22, AB4, AC10, AC16, AC23, AC3, AD1, AD2, AD24, AD25, AE1, AE18, AE24, AE2, AE25, AE8, B1, B2, B25, B24, C10, C16, C23, C3, D22, D4, E10, E16, E21, E5, E8, H1, H21, H25, K21, K23, K3, K5, L11, L12, L13, L14, L15, M11, M12, M13, M14, M15, N11, N12, N13, N14, N15, P11, P12, P13, P14, P15, R11, R12, R13, R14, R15, T21, T23, T3, T5, V1, V25, V5	Low supply voltage, ground	All
VCCA	AB20, F22, F4, J17, J9, K10, K11, K15, K16, L10, L16, R10, R16, T10, T11, T15, T16, U17, U9, Y4	1.5 V supply voltage for array	All
VCCDA	A12, A14, AA13, AA15, AA20, AA7, AB13, AC11, AD11, AD4, AE12, AE17, B15, C15, C6, D13, E13, E19, F21, G10, G5, N21, N5, W21	3.3 V supply voltage for I/O differential amplifier and JTAG and probe interfaces.	All
VCCIB0	A3, B3, C4, D5, J10, J11, K12	3.3 V supply voltage for I/O	All
VCCIB1	A23, B23, C22, D21, J15, J16, K14	3.3 V supply voltage for I/O	All
VCCIB2	C24, C25, D23, E22, K17, L17, M16	3.3 V supply voltage for I/O	All
VCCIB3	AA22, AB23, AC24, AC25, P16, R17, T17	3.3 V supply voltage for I/O	1,3,6,8
VCCIB3	AA22, AB23, AC24, AC25, P16, R17, T17	2.5 V supply voltage for I/O	2,4,5,7
VCCIB4	AB21, AC22, AD23, AE23, T14, U15, U16	3.3 V supply voltage for I/O	All
VCCIB5	AB5, AC4, AD3, AE3, T12, U10, U11	3.3 V supply voltage for I/O	All
VCCIB6	AA4, AB3, AC1, AC2, P10, R9,T9	3.3 V supply voltage for I/O	All
VCCIB7	C1, C2, D3, E4, K9, L9, M10	3.3 V supply voltage for I/O	All
VPUMP	E20	Voltage External Pump. In normal operation, using the internal charge pump, should be tied to ground.	All
TRST	E6	JTAG Test Clock. Actel recommends this pin be hardwired to ground for flight.	All
TCK	F5	JTAG Test Clock. Actel recommends this pin be hardwired to ground. Must not be left unconnected.	All
TDI	C5	JTAG Test Data Input. Actel recommends that this pin be hardwired to VCCDA, or left unconnected.	All
TDO	F6	JTAG Test Data Output. Must be left unconnected.	All
TMS	D6	JTAG Test Mode Select. Actel recommends this pin be hardwired to VCCDA, or left unconnected.	All
PRA	F13	Test Probe. The pins' probe capabilities are disabled to protect programmed design confidentiality. These pins must be left unconnected.	All
PRB	A13		All
PRC	AB12		All
PRD	AE13		All
NC	AA12, AA14, E12, E14, F12, F14, H12, H14, J12, J14, U12, U14, V12, V14, Y12, Y14	No Connection. Pins are not connected to circuitry within the device. These pins can be driven to any voltage or can be left floating with no effect on the operation of the device.	All
CLK*	AC12, AC13, AD12, AD13, W14, W15, W13, Y13	Global clocks. When pins are unused, Actel recommends they are tied to known state, preferably ground.	All
CLKH*	B13, B14, C12, C13, G12, G13, G14, G15	Hardwired clocks. When pins are unused, it is recommended that they are tied to ground.	All

30 Reference documents

[AMBA] AMBA^{TM} Specification, Rev 2.0, ARM IHI 0011A, 13 May 1999, Issue A, first release, ARM Limited

[GRLIB] GRLIB IP Library User's Manual, Aeroflex Gaisler, www.aeroflex.com/gaisler

[GRIP] GRLIB IP Core User's Manual, Aeroflex Gaisler, www.aeroflex.com/gaisler

[SPARC] The SPARC Architecture Manual, Version 8, Revision SAV080SI9308, SPARC International Inc.

[SPW] ECSS - Space Engineering, SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-ST-50-12C, 31 July 2008

[RMAPID] ECSS - Space Engineering, SpaceWire Protocols, ECSS-E-ST-50-51C, February 2010

[RMAP] ECSS - Space Engineering, SpaceWire Protocols, ECSS-E-ST-50-52C, February 2010

[RTAX] RTAX-S/SL RadTolerant FPGAs, 5172169-13/8.10, Revision 13, August 2010, Actel Corporation

[PACK] Package Mechanical Drawings, 5193068-39/8.10, Revision 39, August 2008, 2010 Corporation

[PCIF] CorePCIF Handbook, 50200087-1 /2.07, v2.1, February 2007, Actel Corporation

CorePCIF PCI Interface Core, 51700057-3/02.06, V 5.0, February 2006, Actel Corporation

CorePCIF User's Guide Actel IP Solutions, 50200051-1/3.06, March 2006, Actel Corporation

CorePCIF v2.03 Release Notes, 51300025-2/2.06, February 2006, Actel Corporation

[1553BRM] Core1553BRM Product Handbook, 50200040-0/11-04, November 2004, Actel Corporation

Core1553BRM MIL-STD-1553 BC, RT, and MT, 51700052-4/12.05, v 5.0, December 2005, Actel Corporation

Core1553BRM User's Guide, 50200023-0/06.04, June 2004, Actel Corporation

Core1553BRM v2.16 Release Notes, 51300019-8/6.06, June 2006, Actel Corporation

[1553RT] Core1553BRT MIL-STD-1553B Remote Terminal, 5172165-10/9.05, v 5.0, September 2005, Actel Corporation

Core1553BRT User’s Guide, 5029140-1/9.05, September 2005, Actel Corporation

Core1553BRT v3.0 Release Notes, 5139125-7/08.05, August 2005, Actel Corporation

31 Ordering information

Ordering information is provided in table 190 and a legend is provided in table 191.

*Table 190.*Ordering information

Product	Configuration ID (CID)	Reed-Solomon	SpaceWire LVTTL	Device	Speed Grade		Package Type	Lead Count	Application		
					STD	-1			EV	E	B
LEON3FT-RTAX-IC1	1			RTAX2000S	STD		CQ	352	EV	E	B
LEON3FT-RTAX-IC2	2		No / (Yes)	RTAX2000S		-1	CQ	352	EV	E	B
LEON3FT-RTAX-SC1	3			RTAX2000S	STD		CQ	352	EV	E	B
LEON3FT-RTAX-SC2	4	No / (Yes)	No / (Yes)	RTAX2000S		-1	CG	624	EV	E	B
LEON3FT-RTAX-SC3	5	No / (Yes)	No / (Yes)	RTAX2000S		-1	CG	624	EV	E	B
LEON3FT-RTAX-SC4	6			RTAX2000S		-1	CG	624	EV	E	B
LEON3FT-RTAX-PC1	7	No / (Yes)	No / (Yes)	RTAX2000S		-1	CQ	352	EV	E	B
LEON3FT-RTAX-PC2	8			RTAX2000S		-1	CQ	352	EV	E	B

*Table 191.*Ordering legend

Designator	Option	Description
Product	LEON3FT-RTAX-IC1	Instrument Controller-1
	LEON3FT-RTAX-IC2	Instrument Controller-2
	LEON3FT-RTAX-SC1	Spacecraft Controller-1
	LEON3FT-RTAX-SC2	Spacecraft Controller-2
	LEON3FT-RTAX-SC3	Spacecraft Controller-3
	LEON3FT-RTAX-SC4	Spacecraft Controller-4
	LEON3FT-RTAX-PC1	Payload Controller-1
	LEON3FT-RTAX-PC2	Payload Controller-2
Reed-Solomon	No / (Yes)	Optional Reed-Solomon protected SDRAM memory
SpaceWire LVTTL	No / (Yes)	Optional LVTTL drivers/receivers for SpaceWire links
Device	RTAX2000S	2,000,000 Equivalent System Gates
	RTAX2000SL	2,000,000 Equivalent System Gates - Low-Power Option
Speed Grade	STD	Standard Speed
	-1	Approximately 15% Faster than Standard
Package Type	CQ	Ceramic Quad Flat Pack
	CG	Ceramic Column Grid Array
	LG	Land Grid Array
Application	B	MIL-STD-883 Class B
	E	Actel Extended Flow (Actel Space-Level Flow)
	EV	Class V Equivalent Flow Processing Consistent with MIL-PRF 38535

Commercial quality AX2000 components can be supplied for prototyping. The footprint corresponds to either a CQ352 or a CG624 package, depending on the configuration identifier (CID) as listed in table 190 above. The prototyping component is supplied in a FG896 package and should be used with an FG896-to-CQ352 or an FG896-to-CG624 adapter, respectively. This allows the use of the same footprint on prototyping and flight boards. The pinout of the prototyping component (i.e. the FG896 package) can be provided upon request. Prototyping components can also be supplied in CQ352 and CG624 packages.

32 Change record

Change record information is provided in table 192.

*Table 192.*Change record

Issue	Date	Sections	Note
1.9	2013 January	1.3, 29.2, 31	Reed-Solomon ordering option for CID6 removed
		8.17	READ signal timing corrected
		9.3	Clarified interrupt controller registers
		1.3, 31	LVTTL ordering option for SpaceWire links added
1.8	2012 April	7.4	SEC field of the MCFG3 in memory controller not implemented.
		8	Clarified that 16-bit data bus width is not supported for ROM, SRAM and IO areas in memory controller.
		8.10.1	Clarified that only the first ROM or SRAM bank can be used with EDAC protection for 8-bit wide areas.
		8.15.2	Clarified that RMW bit must not be set for 8-bit SRAM areas when EDAC is not enabled.
		17.2.1, 17.2.4, 17.3	Clarified and improved reset behavior of Mil-Std-1553B RT. Note that CID1 designs shipped before May 2012 adhere to core revision 0000, as described in status register in section 17.3.
		17.3	Re-clarified command legalization of Mil-Std-1553B RT.
		17.2.4, 17.3	Clarified and improved mode command handing of Mil-Std-1553B RT. Note that CID1 designs shipped before May 2012 adhere to core revision 0000, as described in status register in section 17.3.
		23.5	JTAG waveform and timing parameters corrected
1.7	2011 October	17.3	Clarified synchronization with word handling for Core1553BRT.
		30	Corrected reference to Core1553BRM handbook
1.6	2011 October	11.2	Clarified timer chaining
1.5	2011 June	10.2.2, 10.3, 10.5, 10.6, 10.7, 10.7.2, 10.7.3, 10.7.5	Clarified BREAK handling, clarified FIFO debug mode, clarified external clocking (not supported), clarified interrupt generation, updated UART register bit declarations.
1.4	2011 March	15.8	Destination key and time register were missing since 1.1.0.9 issue
1.3	2011 March	Figures 55 and 56	Data hold time after write access shown properly in waveform.
1.2	2011 January	1.2, 2.6, 19, 29.2	Ethernet PHY interrupt added, TX error output removed. *)
		2.3, 20.3	PCI memory area has been extended. *)
		20.2.1, 20.3	Support for generating a type 1 PCI configuration access has been added to the PCI Initiator. *)
		21, 29.2	PCI arbiter extended from 4 to 8 agents. *)
		*)	Note that CID6 and CID8 designs shipped before January 2011 should use the description provided in version 1.1.09 of this document.
		31	Clarified ordering information

*Table 192.*Change record

Issue	Date	Sections	Note
1.1.0.9	2010 November	2.5	Clarified that RMAP target is not implemented in hardware
		6.5	Updated DSU register map
		15.7	Updated SpaceWire link documentation. Note that designs shipped before May 2008 should use the description provided in version 1.1.0.8 of this document. Specifically the clock divider registers and CRC enable bits have changed since then.
		16.1, 17.1	Clarified Mil-Std-1553B RT command legalization.
		30	Updated document references
1.1.0.8	2009 June	1.1, 2.6, 12.2, 12.3	Emphasized that 16-bit general purpose I/O port (GPIO) supports 15 interrupts on gpio[15:1], as defined in section 2.2 and table 166.
		17.3	Added interrupt register to Mil-Std-1553 RT core
		3.10, 6.8, 7.6, 8.17, 10.6, 11.5, 12.5, 15.01, 16.5, 17.5, 18.9, 19.9, 22.5	Several max delays increased.
		30	References to Actel documents updated.
		31	Introduction of Class V Equivalent flow
1.1.0.7	2009 February	1.1, 1.3, 2.2, 2.3, 2.4	Mil-Std-1553 BC/RT/MT cores reduced from two to core
		1.3	Size of on-chip memory increased from 2kByte to 4kByte for CID 3
		2.5	Reed-Solomon data and checksum corrected to 48 bits
		16.3	Mil-Std-1553 BC/RT/MT registers updated
		29.2	Pins for Mil-Std-1553 BC/RT/MT (CID 3) adjusted to one core
1.1.0.6	2009 February	1.3, 2.5, 2.6	Reed-Solomon protected SDRAM introduced
		8.1, 8.10, 8.15, 8.16	Reed-Solomon protected SDRAM described
		8.5	Forbidden 16-bit memory access clarified
		8.16	Additional checkbits and SDDQM[5] signal introduced
		29.2	Pins for Reed-Solomon protected SDRAM introduced
		31	Reed-Solomon protected SDRAM option introduced

*Table 192.*Change record

Issue	Date	Sections	Note
1.1.0.5	2009 January	1, 2	Memory controller names clarified
		2.5	EDAC clarified to be a BCH code
		2.6, 7.5, 8.16, 29.2	RAMBEN and RWEN signals clarified
		2.6, 8.16, 29.2	SA[15] pin usage clarified (currently unused and undriven)
		7	Renamed to Fault Tolerant PROM/SRAM//IO Memory Interface
		7.6	Min delay for output data change specified for memory interface. Several max delays reduced since overly pessimistic. The parameters $t_{FTSRCTRL2}$ and $t_{FTSRCTRL3}$ updated table to indicate correct clock edge.
		8	Renamed to Fault Tolerant PROM/SRAM/SDRAM/IO Memory Interface
		8.10	EDAC error reporting via Status Register clarified. See also 14.2.
		8.17	Min delay for output data change specified for memory interface. Several max delays reduced since overly pessimistic.
		3.8.3	ASR16 register, DP RAM select usage table corrected
		14.2	Clarified that uncorrectable EDAC errors are signalled as an error response on the AMBA bus and how they can be distinguished from a correctable error by means of the CE bit
		19.3.3	Alignment Error changed to Attempt Limit Error in text
		19.7	RE bit description mistakenly used TE instead of RE in text. PHY status change interrupt enable (PI) and PHY status changes (PS) bits defined, but are not used in current release.
		22.2.1	Response byte for AHBAURT removed (obsolete
		22.3	Additional status bits added to AHBUART register descriptions
		29.2	Unused hardwired clock input or routed clock input pin grounding clarified.
		30	Reference to Actel RTAX-S/SL data sheet updated
1.1.0.4	2008 November	7.2.1	PROM data width configuration at reset clarified
		7.4	Reset values clarified for configuration registers
		7.5	Configuration inputs added to signals table
		8.4	SRAM addressing clarified, supporting up to 256 MByte per bank
		8.9.4	SDRAM addressing clarified
		8.16	Configuration inputs added to signals table
		13.1	Clarified on-chip memory sizes for different configurations
		29.4	VCCIB0 pins corrected
		31	Information about prototyping introduced
1.1.0.3	2008 October	7.6, 8.17	Memory controller timing increased due to reduced drive and slew
		19.9	Ethernet timing increased due to reduced drive and slew
		28.4	External load specified as a general statement regarding timing
		29.2	External load specified for all output pins
			Drive strength and slew rate reduced on most signals
		31	Std speed option removed in ordering information
		32	Change record introduced
1.1.0.2	2008 October	19.9	Ethernet signal and clock relations clarified in a note

*Table 192.*Change record

Issue	Date	Sections	Note
1.1.0.1	2008 October	1.3	Single vector trapping and power down modes added
		2.5	Configurations clarified
		2.6	SDDQM[4] signal introduced in some configurations
			Primary Ethernet included correctly in CID 6 and 8
		2.6, 20.2.7, 20.4, 29.2	PCI_RST signal made bi-directional
		8.15.2	Clarified that only line burst is implemented for SDRAM
		8.17	Timing parameter clarified
			Timing reduced for SDRAM due to reduced drive and slew
		15.10	SpaceWire transmit clock period clarified
		17.2	MIL-STD-1553B RT address map introduced
		17.3	MIL-STD-1553B RT registers introduced
		24.3	Minimum clock periods re-defined
		29.2	DSU JTAG interface pins changed (unused)
			SDRAM address, data and control signal drive and slew reduced
			Ethernet pinout modified for CG624
		30	Reference documents updated
1.1.0.0	2008 October	All	Data Sheet and User's Manual merged into one document.

Table of contents

Aeroflex Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden

tel +46 31 7758650
fax +46 31 421407
sales@gaisler.com
www.aeroflex.com/gaisler