CoreMP7 Subsystem

User's Guide

Actel®

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 50200058-1

Release: December 2005

Trademarks

Table of Contents

Introduction

This manual describes the subsystem components available within CoreConsole.

Chapter 1 – CoreMP7 and MP7Bridge describes the function of the MP7Bridge, its ports, and how to connect to it.

Chapter 2 – AHB Bus details the Advanced High-Performance Bus (AHB) fabric component, its ports, and the multiplexing of this fabric.

Chapter 3 – AMBA Bridge provides a module block diagram for the Advanced Microcontroller Bus Architecture (AMBA) Bridge and explains its function.

Chapter 4 – APB Bus describes and provides a diagram for the Advanced Peripheral Bus (APB) fabric.

Chapter 5 – Memory Controller describes the function of the memory controller.

Chapter 6 – CoreUART-APB details the communications, registers, and signals for the CoreUART-APB.

Chapter 7 – Interrupt Controller describes the function of the interrupt controller.

Chapter 8 – Watchdog describes the watchdog unit, which provides a way of recovering from software crashes.

Chapter 9 – Timers provides a functional description of the timers and their operation.

Chapter 10 – System Control Block describes the System Control block.

Chapter 11 – General Purpose I/O (GPIO) Block describes the General Purpose I/O (GPIO) block.

Chapter 12 – Flash ROM (FROM) Access Block describes the Flash Read Only Memory (FROM) Access component.

1 CoreMP7 and MP7Bridge

The CoreMP7 component is an ARM7TDMI-S processor optimized for implementation on Actel devices.

The MP7Bridge component has two functions. Firstly, it converts the native signals from the CoreMP7 processor into an AMBA AHB master interface suitable for connection to an AHB Bus. Secondly, the MP7Bridge includes circuitry which deals with clock signals, reset signals, and the signals which connect to the ARM RealView In-Circuit Emulation (ICE) JTAG port.

The MP7Bridge conditions the incoming raw system clock (SYSCLK) and generates two new clock signals, CLK and HCLK. CLK is tied to the CoreMP7 CLK input. HCLK is tied to the HCLK or PCLK inputs of all the other IP cores. This eliminates any clock skew between flip-flops internal to the CoreMP7 and flip-flops in the subsystem.

The incoming hardware reset signal (NSYSRESET) is synchronized to the system clock within the MP7Bridge and provision is made for handling a watchdog generated reset in the case where a watchdog component is included in the system.

Some circuitry to condition the RealView ICE signals is also included in the MP7Bridge.

CoreMP7 and MP7Bridge Connections

Figure 1-1 on page 8 illustrates how the CoreMP7 and MP7Bridge components are typically connected in a subsystem design.

Note: When connecting the MP7Bridge within a design in CoreConsole, the user is not required to make individual connections to every port. Three groups of signals, or interfaces, are present on the MP7Bridge and these interfaces allow all of their constituent signals to be connected in a single step.

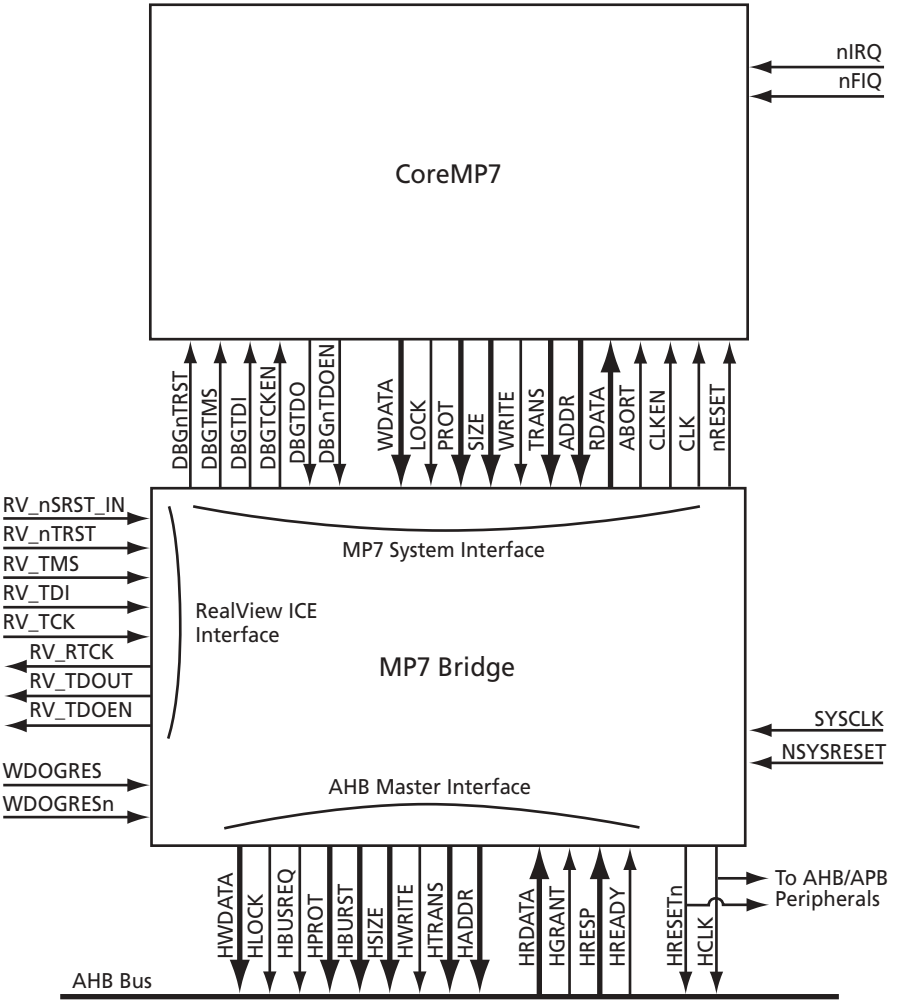

Figure 1-1. CoreMP7 and MP7Bridge Connections

Connecting CoreMP7 in CoreConsole

When the CoreMP7 processor is instantiated in a design, most of its connections to the rest of the system (both inside and outside CoreConsole) are handled through the MP7Bridge.

Table 1-1 lists the ports present on the CoreMP7 component and describes how to connect these in CoreConsole.

Table 1-1. CoreMP7 Connections

Connection	CoreConsole Label	Description
		Required Connections
CoreMP7 System Interface	MP7_SysIf	This interface groups together all of the signals which connect between the CoreMP7 and the MP7Bridge as illustrated in Figure 1-1 on page 8. Connect this interface to the MP7_SysIf interface of the MP7Bridge.
		Optional Connections
Coprocessor Interface	CoProcIf	Groups together CoreMP7 coprocessor signals which provide a means to add coprocessor functionality. This is normally left unconnected. Any inputs present in the interface will be tied to inactive levels.
Embedded Trace Macrocell (ETM) Interface	ETMIf	This interface is used to connect to an ETM which facilitates real–time tracing of code running on the processor. This is normally left unconnected. Any inputs present in the interface will be tied to inactive levels.
nIRQ	nIRQ	Active low interrupt request input. This input is tied high if left unconnected.
nFIQ	nFIQ	Active low fast interrupt request input. This input is tied high if left unconnected.
CFGBIGEND	CFGBIGEND	When asserted (high) this input configures the CoreMP7 in big-endian mode. If no connection is made to this input (normal case) it will be tied low to leave the processor in little-endian mode.
DMORE	DMORE	Output which is asserted (high) during LDM and STM instructions. Normally left unconnected.

CoreMP7 Configuration in CoreConsole

Table 1-2 describes the configurable options for the CoreMP7 component.

Table 1-2. CoreMP7 Configuration

Configurable Option	Default setting	Description
Die	M7A3P1000	Selects target die. Family (ProAsic3 or ProAsic3E) is implied by die selection.
Debug	Enabled	Determines whether or not debug functionality is included in CoreMP7 instance. Possible settings are “Enabled” or “Disabled”.
Speed grade	–2	Selects speed grade of the target Actel device

MP7Bridge Connections in CoreConsole

Table 1-3 lists the ports present on the MP7Bridge and describes how to connect these in CoreConsole.

Table 1-3. MP7Bridge Connections

Connection	CoreConsole Label	Description
Required Connections		
CoreMP7 System Interface	MP7_SysIf	This interface groups together all of the signals which connect between the CoreMP7 and the MP7Bridge as illustrated in Figure 1-1 on page 8. Connect this interface to the MP7_SysIf interface of the CoreMP7 component.
AHB master interface	AHBmaster	Groups together AHB master signals. Connect to AHB mirrored master (AHBmmaster) interface of AHB Bus.
SYSCLK	SYSCLK	Raw system clock input. Connect to susbsystem toplevel SYSCLK port.
NSYSRESET	NSYSRESET	Active low hardware reset input. Connect to subsystem toplevel NSYSRESET port.

Table 1-3. MP7Bridge Connections (Continued)

Connection	CoreConsole Label	Description
HCLK	HCLK	AMBA system clock. Connect to HCLK/PCLK ports of AHB/APB peripherals. May also be connected to subsystem toplevel if used outside the subsystem.
HRESETn	HRESETn	Active low AMBA system reset. Connect to HRESETn/PRESETn ports of AHB/APB peripherals. May also be connected to subsystem toplevel if used outside the subsystem.

Table 1-3. MP7Bridge Connections (Continued)

Connection	CoreConsole Label	Description
Optional Connections		
RealView ICE interface	RV_ICE_If	Groups together RealView ICE JTAG interface signals. If using debugging, connect this interface to the subsystem toplevel.
WDOGRES	WDOGRES	Active high input which signals that watchdog has timed-out. If your design includes a watchdog, connect this port to the WDOGRES port of the watchdog. This input is tied low if no connection is made to it.
WDOGRESn	WDOGRESn	Active low output signal used to reset watchdog component. If your design includes a watchdog, connect this port to the WDOGRESn port of the watchdog, otherwise leave unconnected.

MP7Bridge Configuration in CoreConsole

Table 1-4 describes the configurable options for the MP7Bridge.

Table 1-4. MP7Bridge Configuration

Configurable Option	Default setting	Description
Device family	ProASIC3	Selects device family of target device. Possible settings are “ProASIC3” or “ProASIC3E”.

MP7Bridge Port List

Table 1-5 on page 13 provides a detailed list and description of the ports on the MP7Bridge.

Table 1-5. MP7Bridge Port List

Port Name	Width	Direction	Description
System Connections			
SYSCLK	1	Input	Incoming raw system clock
NSYSRESET	1	Input	Raw hardware system reset input from top level. Will typically be controlled by a push button switch.
AHB Connections			
HCLK	1	Output	AHB system clock. Reset and RealView ICE JTAG signals are synchronized to this clock. This also connects to the other cores.
HRESETn	1	Output	AHB reset signal (active low). This reset may be asserted asynchronously but will always be negated synchronous to HCLK. HRESETn also connects to the other cores.
HREADY	1	Input	AHB ready signal
HRESP	2	Input	AHB response signal
HGRANT	1	Input	AHB grant signal
HRDATA	32	Input	AHB read data
HADDR	32	Output	AHB address signal
HTRANS	2	Output	AHB transfer type signal
HWRITE	1	Output	AHB read/write indication
HSIZE	3	Output	AHB size (byte, word etc.) of transfer indication
HBURST	3	Output	AHB burst signal
HPROT	4	Output	AHB protection signal
HBUSREQ_	1	Output	AHB Bus request. In a single master system this port will typically be unconnected. Where there are multiple masters it will be connected to some form of arbitration component.

Table 1-5. MP7Bridge Port List (Continued)

Port Name	Width	Direction	Description
HLOCK	1	Output	AHB lock signal
HWDATA	32	Output	AHB write data
CoreMP7 Connections			
CLK	1	Input	Clock signal. Connects to CLK port of CoreMP7.
nRESET	1	Input	Reset signal. Connects to nRESET port of CoreMP7.
ADDR	32	Input	Address bus. Connects to ADDR port of CoreMP7.
LOCK	1	Input	Lock signal. Connects to LOCK port of CoreMP7.
SIZE	2	Input	Size signal. Connects to SIZE port of CoreMP7.
WRITE	1	Input	Write signal. Connects to WRITE port of CoreMP7.
PROT	2	Input	Protection signal. Connects to PROT port of CoreMP7.
TRANS	2	Input	Transfer signal. Connects to TRANS port of CoreMP7.
WDATA	32	Input	Write data. Connects to WRITE port of CoreMP7.
ABORT	1	Output	Abort signal. Connects to ABORT port of CoreMP7.
CLKEN	1	Output	Clock enable. Connects to CLKEN port of CoreMP7.
RDATA	32	Output	Read data. Connects to RDATA port of CoreMP7.
DBGnTRST	1	Output	Debug TAP reset signal. Connects to DBGnTRST port of CoreMP7.
DBGTMS	1	Output	Debug test mode select signal. Connects to DBGTMS port of CoreMP7.
DBGTDI	1	Output	Debug test data in signal. Connects to DBGTDI port of CoreMP7.
DBGTCKEN	1	Output	Debug test clock signal. Connects to DBGTCKEN port of CoreMP7.
DBGTDO	1	Input	Debug test data out signal. Connects to DBGTDO port of CoreMP7.
DBGnTDOEN	1	Input	Debug test data out enable signal. Connects to DBGnTDOEN port of CoreMP7.

Table 1-5. MP7Bridge Port List (Continued)

Port Name	Width	Direction	Description
Watchdog Connections			
WDOGRES	1	Input	Watchdog reset input. If a watchdog is present in the system, this port should be connected to the 'bark' port of the watchdog; that is, whichever signal the watchdog asserts when it initiates a reset. This input is active high.
WDOGRESn	1	Output	Active low reset output to watchdog. Should be connected to the reset port of the watchdog (if present). This output is asserted when NSYSRESET is asserted but not when WDOGRES is asserted. This allows the watchdog to reset the system without resetting itself.
RealView ICE JTAG Connections			
RV_nSRST_IN	1	Input	RealView ICE system reset input. Connect to top level if using RealView.
RV_nTRST	1	Input	RealView ICE Test Access Port (TAP) reset. Connect to top level if using RealView.
RV_TMS	1	Input	RealView ICE test mode select. Connect to top level if using RealView.
RV_TDI	1	Input	RealView ICE test data in. Connect to top level if using RealView.
RV_TCK	1	Input	RealView ICE test clock. Connect to top level if using RealView.
RV_RTCK	1	Output	RealView ICE return test clock. Connect to top level if using RealView.
RV_TDOUT	1	Output	RealView ICE test data out. Connect to top level if using RealView.
RV_nTDOEN	1	Output	RealView ICE test data out enable. Connect to top level if using RealView.

2

AHB Bus

The AHB Bus component implements the AHB Bus fabric for a subsystem. Currently the AHB Bus essentially supports an AHB-Lite system in that it accommodates a single master. Up to 16 AHB slaves can be present on the bus. From a memory map point of view, all AHB slaves are allocated an equal amount (256 MB) of memory.

AHB-Lite Overview

The AHB Bus component available in CoreConsole essentially implements an AHB-Lite Bus fabric. AHB-Lite is a subset of the full AHB specification and is intended for use in designs where there is only a single bus master.

AHB-Lite simplifies the AHB specification by removing the protocol required for multiple bus masters. This includes arbitration based on a request/grant type mechanism and split/retry responses from AHB slaves.

The basic structure of this fabric is shown in Figure 2-1 on page 18. The elements labeled “Decoder” and “Read Data/Response Mux” in the figure are contained within the AHB Bus component.

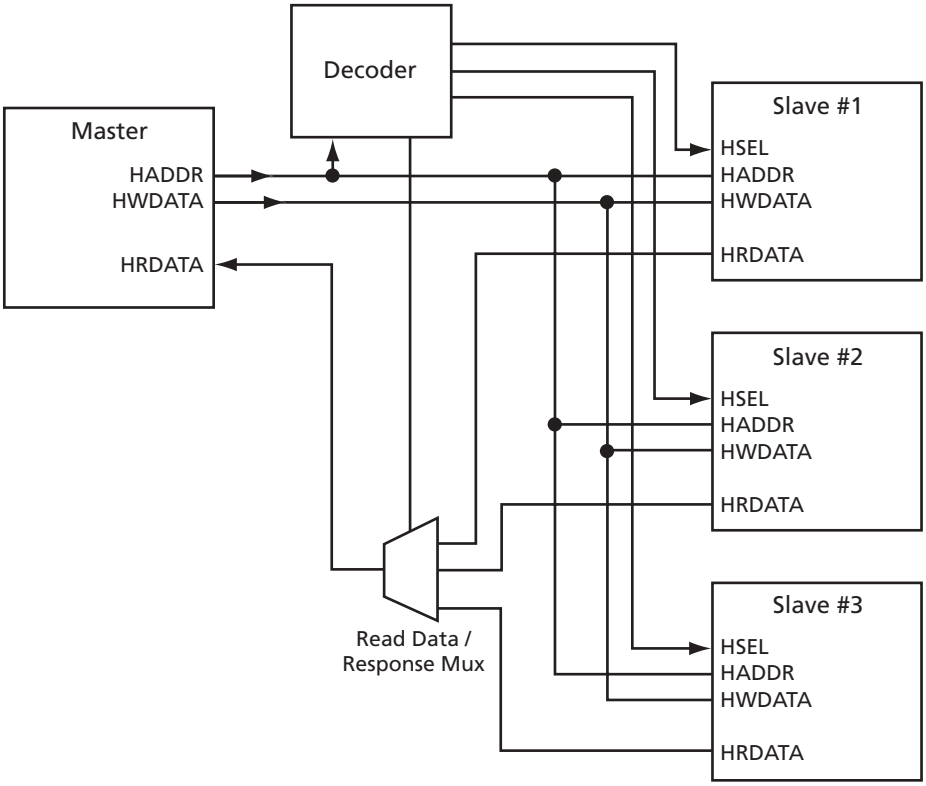

Figure 2-1. AHB-Lite Block Diagram

Connecting the AHB Bus in CoreConsole

Table 2-1 lists the ports present on the AHB Bus and describes how to connect these in CoreConsole.

Table 2-1. AHB Bus Connections

Connection	CoreConsole Label	Description
Required Connections		
AHB mirrored master interface	AHBmmaster	This interface connects to the AHB master. Normally this will be connected to the AHBmaster interface of the MP7Bridge.
HCLK	HCLK	AHB system clock input. Connect this to the HCLK output of the MP7Bridge.
HRESETn	HRESETn	Active low AHB system reset. Connect this to the HRESETn output of the MP7Bridge.
Optional Connections		
	AHBmslave0	AHB mirrored slave 0 interface. Normally connected to AHBslave_base interface of Memory Controller.
	AHBmslave1	AHB mirrored slave 1 interface
	AHBmslave2	AHB mirrored slave 2 interface
	AHBmslave3	AHB mirrored slave 3 interface
	AHBmslave4	AHB mirrored slave 4 interface
	AHBmslave5	AHB mirrored slave 5 interface
	AHBmslave6	AHB mirrored slave 6 interface
	AHBmslave7	AHB mirrored slave 7 interface
	AHBmslave8	AHB mirrored slave 8 interface
	AHBmslave9	AHB mirrored slave 9 interface
	AHBmslave10	AHB mirrored slave 10 interface
	AHBmslave11	AHB mirrored slave 11 interface

Table 2-1. AHB Bus Connections (Continued)

Connection	CoreConsole Label	Description
	AHBmslave12	AHB mirrored slave 12 interface
	AHBmslave13	AHB mirrored slave 13 interface
	AHBmslave14	AHB mirrored slave 14 interface
	AHBmslave15	AHB mirrored slave 15 interface

AHB Bus Port List

Table 2-2 lists the ports present on the AHB Bus component. Four groups of signals can be identified:

1. Common AHB system signals (clock and reset)
2. AHB mirrored master interface. This connects to the master on the AHB Bus. A mirrored master interface is made up of the same signals as a master interface but the direction of the signals is reversed.
3. Signals common to all 16 AHB mirrored slave interfaces. These are AHB master signals which connect to all slaves.
4. AHB mirrored slave (master) signals specific to each slave.

Table 2-2. Ports on the AHB Bus Component

Signal	Direction	Description
Common AHB System Signals		
HCLK	Input	Bus clock. This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn	Input	Reset. The bus reset signal is active low and is used to reset the system and the bus. This is the only active low AHB signal.

Table 2-2. Ports on the AHB Bus Component (Continued)

Signal	Direction	Description
Mirrored AHB Master Interface		
HADDR[31:0]	Input	This is the 32-bit system address bus.
HTRANS[1:0]	Input	Transfer type. Indicates the type of the current transfer: 00 – Idle 01 – Busy 10 – Non-Sequential 11 - Sequential
HWRITE	Input	Transfer direction. When high this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0]	Input	Transfer size. Indicates the size of the transfer, which can be byte (8-bit), halfword (16-bit), or word (32-bit).
HBURST[2:0]	Input	Burst type. Indicates if the transfer forms part of a burst. The CoreMP7 performs incrementing bursts of type INCR.
HPROT[3:0]	Input	Protection control. These signals indicate if the transfer is an opcode fetch or data access, and if the transfer is a Supervisor mode access or User mode access.
HWDATA[31:0]	Input	32-bit data from the master.
HRDATA[31:0]	Output	32-bit date written back to the master.
HREADY	Output	Transfer done. When high the HREADY signal indicates that a transfer has finished on the bus. This signal can be driven low to extend a transfer.
HRESP[1:0]	Input	Transfer response. Indicates an Okay Error Retry, or Split response.
Common AHB Mirrored Slave Signals		
HADDRS[31:0]	Output	This is the 32-bit system address bus.
HTRANSS[1:0]	Output	Transfer type. Indicates the type of the current transfer: 00 – Idle 01 – Busy 10 – Non-Sequential 11 - Sequential

Table 2-2. Ports on the AHB Bus Component (Continued)

Signal	Direction	Description
HWRITES	Output	Transfer direction. A write transfer is indicated when this signal is high and a read transfer is indicated this signal is low during the address phase of an AHB transfer.
HSIZES[2:0]	Output	Transfer size. Indicates the size of the transfer, which can be: 00 - byte (8-bit) 01 - halfword (16-bit) 10 - word (32-bit).
HBURSTS[2:0]	Output	Burst type. Indicates if the transfer forms part of a burst.
HPROTS[3:0]	Output	Protection control. These signals indicate if the transfer is an opcode fetch or data access, and if the transfer is a Supervisor mode access or User mode access.
HWDATAS[31:0]	Output	32-bit data to the slave
HREADYS	Output	Transfer done. Out to the slaves (alias of HREADY)
Slave-Specific Mirrored Slave Signals		
HSELx	Output	Select of slave x (where x is a integer between 0 and 15)
HRDATASx[31:0]	Input	32-bit read data from slave x.
HREADYSx	Input	Ready signal from slave x. When high indicates that slave has completed transfer and is ready for another transfer.
HRESPSx[1:0]	Input	Transfer response from slave x which can be: 00 – Okay 01 – Error 10 – Retry 11 – Split

3 AMBA Bridge

The AMBA Bridge is an AHB slave which links the AHB Bus to the APB Bus and acts as the master on the APB Bus. Address decoding for the APB Bus is carried out within the AMBA Bridge and this provides select signals for up to 16 APB slave slots.

Read and write transfers on the AHB are converted into corresponding transfers on the APB. High bandwidth peripherals such as a memory controller are typically connected to the AHB Bus whereas the APB Bus is used for less demanding peripherals such as a watchdog. Unlike the AHB Bus, transfers on the APB Bus are not pipelined.

Figure 3-1 shows the AMBA Bridge module block diagram.

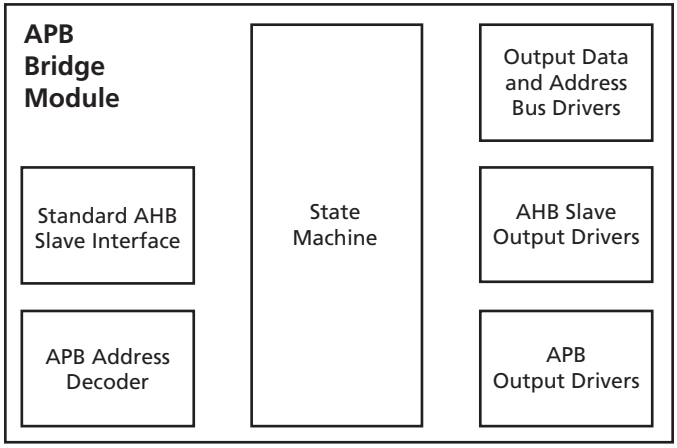

Figure 3-1. AMBA Bridge

Connecting the AMBA Bridge in CoreConsole

Table 3-1 lists the ports present on the AMBA Bridge and describes how to connect these in CoreConsole.

Table 3-1. AMBA Bridge Connections

Connection	CoreConsole Label	Description
Required Connections		
AHB slave interface	AHBslave	Connect this to the any of the 16 available slave slots on the AHB Bus.
APB master interface	APBmaster	Connect this to the APBmmaster interface of the APB Bus.
HCLK	HCLK	AHB system clock input. Connect this to the HCLK output of the MP7Bridge.
HRESETn	HRESETn	Active low AHB system reset. Connect this to the HRESETn output of the MP7Bridge.

4

APB Bus

Along with the AMBA Bridge, the APB Bus component provides an AMBA APB fabric which supports up to 16 APB slaves. The AMBA Bridge provides APB address decoding in the form of select signals. The APB Bus is concerned with multiplexing the read data busses from all the APB slaves unto one single read data bus to send to the AMBA Bridge. Figure 4-1 gives an illustration of the APB fabric.

There is one APB master interface which is typically connected to the AMBA Bridge, and 16 equal-sized (16 MB) APB slave interfaces.

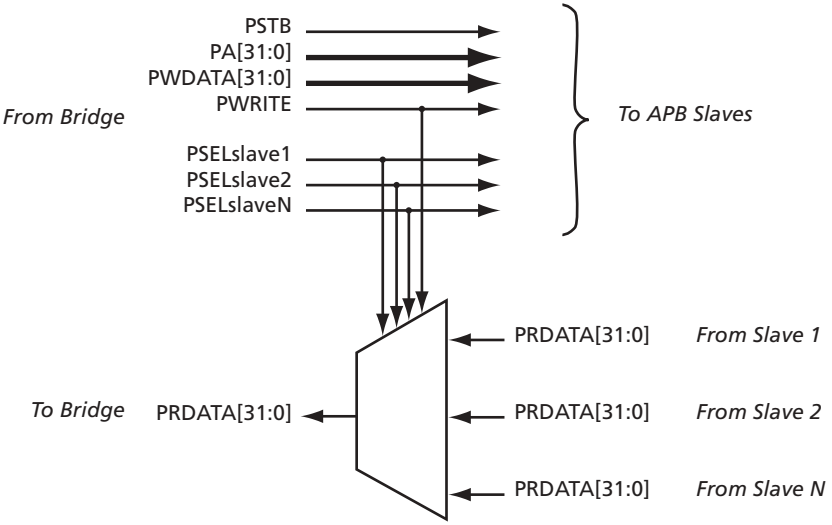

Figure 4-1. APB Fabric

Connecting the APB Bus in CoreConsole

Table 4-1 lists the ports present on the APB Bus and describes how to connect these in CoreConsole.

Table 4-1. APB Bus Connections

Connection	CoreConsole Label	Description
Required Connections		
APB mirrored master interface	APBmmaster	This interface connects to the APBmaster interface of the AMBA Bridge.
Optional Connections		
	APBmslave0	APB mirrored slave 0 interface
	APBmslave1	APB mirrored slave 1 interface
	APBmslave2	APB mirrored slave 2 interface
	APBmslave3	APB mirrored slave 3 interface
	APBmslave4	APB mirrored slave 4 interface
	APBmslave5	APB mirrored slave 5 interface
	APBmslave6	APB mirrored slave 6 interface
	APBmslave7	APB mirrored slave 7 interface
	APBmslave8	APB mirrored slave 8 interface
	APBmslave9	APB mirrored slave 9 interface
	APBmslave10	APB mirrored slave 10 interface
	APBmslave11	APB mirrored slave 11 interface
	APBmslave12	APB mirrored slave 12 interface
	APBmslave13	APB mirrored slave 13 interface
	APBmslave14	APB mirrored slave 14 interface
	APBmslave15	APB mirrored slave 15 interface

5 Memory Controller

The Memory Controller is an AHB slave component which supports access to SRAM and Flash memory resources.

The Memory Controller uses 3 slave slots on the AHB Bus. A slot is allocated to the SRAM, another to the Flash and a third slot, designated as Base, can be used to access either the SRAM or Flash via a memory aliasing mechanism.

The memory controller has a 'Remap' input which is used to select whether SRAM or Flash appears at the Base slot. Typically the Base slave interface is connected to slot 0 on the AHB Bus, the Flash slave interface to slot 1, and the SRAM slave interface to slot 2.

The Remap signal may be driven by the System Control Block (see "System Control Block" on page 67) which provides a way for the processor to change the memory aliasing by writing to a particular location in its address space.

Connecting the Memory Controller in CoreConsole

Table 5-1 lists the ports present on the Memory Controller and describes how to connect these in CoreConsole.

Table 5-1. Memory Controller Connections

Connection	CoreConsole Label	Description
Required Connections		
Base AHB slave interface	AHBslave_base	This interface groups together all of the signals used to connect the base memory region to an AHB slot. Normally connected to slave slot 0 (AHBmslave0) of the AHB Bus.
Flash AHB slave interface	AHBslave_flash	This interface groups together all of the signals used to connect the flash memory region to an AHB slot. Normally connected to slave slot 1 (AHBmslave1) of the AHB Bus.
SRAM AHB slave interface	AHBslave_sram	This interface groups together all of the signals used to connect the SRAM memory region to an AHB slot. Normally connected to slave slot 2 (AHBmslave2) of the AHB Bus.

Table 5-1. Memory Controller Connections (Continued)

Connection	CoreConsole Label	Description
External Memory Interface	ExternalMemoryInterface	This interface contains the signals used to connect to the actual memory devices and should be routed to the toplevel of your subsystem. See "External Memory Interface" on page 28 for more information on this interface.
HCLK	HCLK	AHB system clock input. Connect this to the HCLK output of the MP7Bridge.
HRESETn	HRESETn	Active low AHB system reset. Connect this to the HRESETn output of the MP7Bridge.
Optional Connections		
Remap	Remap	This input is used to control aliasing of the Flash and SRAM memory regions. When Remap is low, Flash is aliased to the Base memory region. When Remap is high, SRAM is aliased to the Base memory region. Remap is normally connected to the Remap output of the System Control block (see "System Control Block" on page 67). If no connection is made, Remap will be tied low.

External Memory Interface

The External Memory Interface of the Memory Controller should be routed to the subsystem toplevel to facilitate communication with Flash and SRAM resources.

The Memory Controller is designed to accommodate a variety of Flash and SRAM configurations as outlined in "Memory Controller Configurable Options" on page 31. For this reason, the External Memory Interface is somewhat generic in nature in order to enable connection to a range of different memory devices and memory systems.

Memory devices typically have a number of inputs that are fixed at static levels which are dependent on the particular memory architecture in place. If the memory devices in your system have such static inputs, it is intended that these are handled in the toplevel description for your FPGA device; that is, above the subsystem toplevel.

Similarly, any tri-state buffers must be instantiated above the subsystem toplevel. The data bus connecting between the FPGA and the actual memory devices is normally a bi-directional bus which is driven by tri-state buffers.

Table 5-2 lists and describes the signals which make up the External Memory Interface. Apart from "MemDataIn", all of the signals are outputs from the Memory Controller. All of the 1-bit wide control signals are active low as indicated by the "N" at the end of the signal names.

Table 5-2. Memory Controller External Memory Interface

Signal	Width	Description
Flash control signals		
FlashCSN	1	Flash chip select. In some systems the chip select pin of the Flash will be fixed at an active level in which case this signal may be left unconnected.
FlashOEnN	1	Flash output enable
FlashWEnN	1	Flash write enable
SRAM control signals		
SramCSN	1	SRAM chip select. In some systems the chip select pin of the SRAM will be fixed at an active level in which case this signal may be left unconnected.
SramOEnN	1	SRAM output enable
SramWEnN	1	SRAM write enable
SramByte0N	1	SRAM byte 0 enable
SramByte1N	1	SRAM byte 1 enable
SramByte2N	1	SRAM byte 2 enable
SramByte3N	1	SRAM byte 3 enable

Table 5-2. Memory Controller External Memory Interface (Continued)

Signal	Width	Description
Shared memory signals		
MemReadN	1	Combined Flash/SRAM read enable. This signal is asserted (low) when either FlashOEnN or SramOEnN is low and is intended for use in a memory system which does not have separate connections to the Flash and SRAM output enable pins.
MemWriteN	1	Combined Flash/SRAM write enable. This signal is asserted (low) when either FlashWEnN or SramWEnN is low and is intended for use in a memory system which does not have separate connections to the Flash and SRAM write enable pins.
MemAddr	28	Flash/SRAM address bus
MemDataOEnN	1	Flash/SRAM data out enable. Control signal for data bus tri-states. Active low; that is, low when data is driven on MemDataOut.
MemDataOut	32	Flash/SRAM data out
MemDataIn	32	Flash/SRAM data in

Memory Controller Configurable Options

There are a number of configurable options which apply to the Memory Controller; these are detailed in Table 5-3 on page 31. If a configuration different to the default is required, the user should use the configuration dialog in CoreConsole to select appropriate values for the configurable options.

Table 5-3. Memory Controller Configurable Options

Configurable Option	Default setting	Description
SRAM mode	Asynchronous	Selects either asynchronous or synchronous SRAM. Possible settings are “Asynchronous” or “Synchronous”.
Flash data bus width	32 bit	Selects the data bus width for the Flash memory interface. Possible settings are “32 bit” or “16 bit”.
Number of wait states for Flash read	1	Selects the number of wait states inserted during a Flash read access. Possible range is 0 to 3.
Number of wait states for Flash write	1	Selects the number of wait states inserted during a Flash write access. Possible range is 1 to 3.
Number of wait states for SRAM read	1	Only applicable when SRAM mode is set to “Asynchronous”. Selects the number of wait states inserted during an SRAM read access. Possible range is 0 to 3.
Number of wait states for SRAM write	1	Only applicable when SRAM mode is set to “Asynchronous”. Selects the number of wait states inserted during an SRAM write access. Possible range is 1 to 3.

6 CoreUART-APB

The CoreUART-APB component available in CoreConsole is an APB-wrapped version of the Actel DirectCores CoreUART. The serial communication interface is identical to that described in the DirectCores CoreUART datasheet which is available on the Actel website:

http://www.actel.com/ipdocs/CoreUART_DS.pdf

The CoreConsole CoreUART-APB adds an APB interface which gives access to transmit and receive data registers as well as two control registers and a status register. The CoreUART registers are described in "CoreUART-APB Programmer's Model" on page 35.

Figure 6-1 shows a block diagram of the CoreUART-APB.

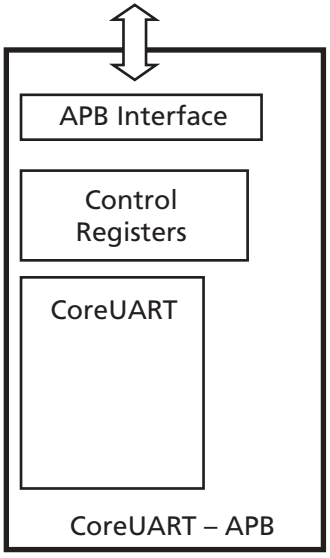

Figure 6-1. CoreUART-APB Block Diagram

Connecting CoreUART-APB in CoreConsole

Table 6-1 lists the ports present on the CoreUART-APB component and describes how to connect these in CoreConsole.

Table 6-1. CoreUART-APB Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus.
PCLK	PCLK	APB clock signal. Normally connected to the HCLK output of the MP7Bridge.
PRESETn	PRESETn	Active low APB reset input. Normally connected to the HRESETn output of the MP7Bridge.
rx	rx	Serial receive data input. Normally connected to subsystem toplevel for subsequent connection to a pin of the FPGA.
tx	tx	Serial transmit data output. Normally connected to subsystem toplevel for subsequent connection to a pin of the FPGA.
Optional Connections		
txrdy	txrdy	Status output. When low, the transmit data buffer/FIFO is not available for additional transmit data. The level of this signal is also available in the Status Register. (See "Status Register" on page 37.)
receive_full	receive_full	Status output. When high, data is available in the receive data buffer/FIFO. The level of this signal is also available in the Status Register. (See "Status Register" on page 37.)

CoreUART-APB Configurable Options

There are a number of configurable options which apply to the CoreUART-APB; these are detailed in Table 6-2. If a configuration different to the default is required, the user should use the configuration dialog in CoreConsole to select appropriate values for the configurable options.

Table 6-2. CoreUART-APB Configurable Options

Configurable Option	Default setting	Description
Transmit FIFO	Disabled	Enables or disables transmit FIFO
Receive FIFO	Disabled	Enables or disables receive FIFO
Mode of operation	Asynchronous	Selects mode of operation of CoreUART. Possible settings are “Asynchronous” or “Synchronous”.
Device family	ProASIC3	Selects target family. Possible settings are “ProASIC3” or “ProASIC3E”.

CoreUART-APB Programmer’s Model

Table 6-3 lists the registers for CoreUART-APB.

Table 6-3. CoreUART-APB Registers

Address	Type	Width	Reset value	Name	Description
base + 0x000	Write	32	0	TxData	Transmit Data Register
base + 0x004	Read	32	0	RxData	Receive Data Register
base + 0x008	Read/Write	32	0	Ctrl1	Control Register 1
base + 0x00C	Read/Write	32	0	Ctrl2	Control Register 2
base + 0x010	Read	32	0	Status	Status Register

Transmit Data Register

The 7- or 8- bit transmit data

Receive Data Register

The 7- or 8- bit receive data.

Control Register 1

Control Register 1 contains a single field, Baud value, which is used to set the baud rate for the CoreUART when in asynchronous mode. The Baud value should be set according to the following equation.

Baud value (decimal) = (clock / ((baud + 1) x 16))

where clock is the APB system clock frequency in hertz.

The result of this calculation must be rounded to the nearest integer and converted to hexadecimal to obtain the value which should be written to Control Register 1 (Table 6-4).

For example, when the APB clock frequency is 10 MHz and a baud rate of 9600 is desired, 0x41 should be written to Control Register 1.

Table 6-4. Control Register 1

Bits	Name	Type	Function
7:0	Baud value	Read/Write	8 bit value setting the baud rate

Control Register 2

Control Register 2 (Table 6-5) is used to assign values to the configuration inputs available on the CoreUART.

Table 6-5. Control Register 2

Bits	Name	Type	Function
0	bit8	Read/Write	Data width setting : Bit8 = 0 : 7 bit data Bit8 = 1 : 8 bit data
1	parity_en	Read/Write	Parity is enabled when this bit is set to 1
2	odd_n_even	Read/Write	Parity is set as follows : odd_n_even = 0 : even odd_n_even = 1 : odd
7:3			Unused

Status Register

The Status Register (Table 6-6) provides information on the status of the CoreUART.

Table 6-6. Status Register

Bits	Name	Type	Function
0	txrdy	Read Only	When low, the transmit data buffer/FIFO is not available for additional transmit data.
1	receive_full	Read Only	When high, data is available in the receive data buffer/FIFO. This bit is cleared by reading the Receive Data Register.
2	parity_err	Read Only	When high, a parity error occurred during a receive transaction. This bit is cleared by reading the Receive Data Register.
3	overflow	Read Only	When high, a receive overflow has occurred. This bit is cleared by reading the Receive Data Register.
7:4			Unused

7 Interrupt Controller

The Interrupt Controller is an AHB slave which provides a software interface to the interrupt system. In a CoreMP7 system, two levels of interrupt are available:

- Fast Interrupt Request (FIQ) for fast, low latency interrupt handling
- Interrupt Request (IRQ) for more general interrupts

Only a single FIQ source at a time is generally used in a system, to provide a true low-latency interrupt. This has the following benefits:

- You can execute the interrupt service routine directly without determining the source of the interrupt.
- Interrupt latency is reduced. You can use the banked registers available for FIQ interrupts more efficiently, because a context save is not required.

The Interrupt Controller (Figure 7-1) can accommodate up to 32 interrupt sources. The interrupt controller uses a bit position for each different interrupt source. The software can control each request line to generate software interrupts.

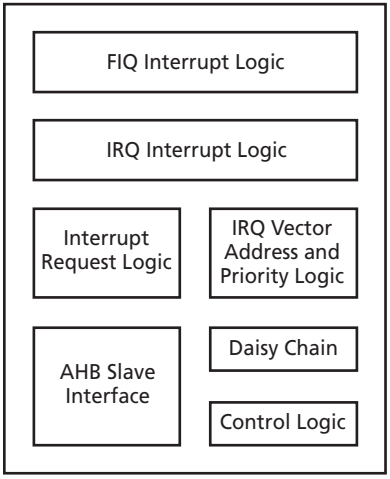

Figure 7-1. Interrupt Controller Block Diagram

The nonvectored and daisy-chained IRQ interrupts provide an address for an Interrupt Service Routine (ISR). Reading from the vector interrupt address register (ICVectAddr) provides the address of the ISR, and updates the interrupt priority hardware that masks out the current and any lower priority interrupt requests. Writing to the ICVectAddr register indicates to the interrupt priority hardware that the current interrupt is serviced, allowing lower priority interrupts to go active.

The FIQ interrupt has the highest priority, followed by nonvectored IRQ interrupts. Daisy-chained interrupts have the lowest priority. A programmed interrupt request enables you to generate an interrupt under software control. This register is typically used to downgrade an FIQ interrupt to an IRQ interrupt.

The IRQ and FIQ request logic has an asynchronous path. This enables interrupts to be asserted when the clock is disabled.

Connecting the Interrupt Controller in CoreConsole

Table 7-1 lists the ports present on the Interrupt Controller and describes how to connect these in CoreConsole.

Table 7-1. Interrupt Controller Connections

Connection	CoreConsole Label	Description
Required Connections		
AHB slave interface	AHBslave	
HCLK	HCLK	AHB system clock input. Connect this to the HCLK output of the MP7Bridge.
HRESETn	HRESETn	Active low AHB system reset. Connect this to the HRESETn output of the MP7Bridge.
Optional Connections		
Fast Interrupt Request	nICFIQ	Fast interrupt request output. Normally connected to nFIQ port of CoreMP7.
Interrupt Request	nICIRQ	Interrupt request output. Normally connected to nIRQ port of CoreMP7.
Fast Interrupt daisy-chain input	nICFIQIN	Connection for daisy-chained interrupt controllers. Tied high if left unconnected.
Interrupt daisy-chain input	nICIRQIN	Connection for daisy-chained interrupt controllers. Tied high if left unconnected.
Vector address input	ICVECTADDRIN	Vector address input. Tied to 0x00000000 if left unconnected.

Table 7-1. Interrupt Controller Connections (Continued)

Connection	CoreConsole Label	Description
Vector address output	ICVECTADDROUT	Vector address output. Normally left unconnected.
Interrupt source 0	ICINTSOURCE0	Active high interrupt input. Tied low if no connection is made to this port.
Interrupt source 1	ICINTSOURCE1	Active high interrupt input. Tied low if no connection is made to this port.
⋮		
Interrupt source 31	ICINTSOURCE31	Active high interrupt input. Tied low if no connection is made to this port.

Programmer's Model

By convention, for the IRQ interrupt, bits 1 to 5 must be used as defined in Table 7-2. Bit 0 and bit 6 upwards are available for use as required. For the FIQ interrupt, the bits can be used as required.

Table 7-2. Interrupt Standard Configuration

Bit	Interrupt source
1	Software interrupt
2	Comms Rx
3	Comms Tx
4	Timer 1
5	Timer 2

The software can control the source interrupt lines to generate software interrupts. These interrupts are generated before interrupt masking, in the same way as external source interrupts. Software interrupts are cleared by writing to the software interrupt clear register, ICSoftIntClear (see "Software Interrupt Clear Register, ICSoftIntClear" on page 46). This is normally done at the end of the interrupt service routine.

Interrupt Flow Sequence

The following procedure shows the sequence for the vectored interrupt flow:

1. An interrupt occurs.
2. The ARM processor branches to either the IRQ or FIQ interrupt vector.
3. If the interrupt is an IRQ, read the ICVectAddr register and branch to the interrupt service routine. This can be done using an LDR PC instruction.
4. Reading the ICVectorAddr register updates the interrupt controllers hardware priority register.
5. Stack the workspace so that IRQ interrupts can be re-enabled.
6. Enable the IRQ interrupts so that a higher priority can be serviced.
7. Execute the ISR.
8. Clear the requesting interrupt in the peripheral, or write to the ICSoftIntClear.
9. Register if the request was generated by a software interrupt.
10. Disable the interrupts and restore the workspace.
11. Write to the ICVectAddr register. This clears the respective interrupt in the internal interrupt priority hardware.
12. Return from the interrupt. This re-enables the interrupts.

Simple Interrupt Flow

The following procedure shows how you can use the interrupt controller without using vectored interrupts or the interrupt priority hardware. For example, you can use it for debugging.

1. An interrupt occurs.
2. Branch to IRQ or FIQ interrupt vector.
3. Branch to the interrupt handler.
4. Interrogate the ICIRQStatus register to determine which source generated the interrupt, and prioritize the interrupts if there are multiple active interrupt sources. This takes a number of instructions to compute.
5. Branch to the correct ISR.
6. Execute the ISR.
7. Clear the interrupt. If the request was generated by a software interrupt, the ICSoftIntClear register must be written to.
8. Check the ICIRQStatus register to ensure that no other interrupt is active. If there is an active request go to Step 4.
9. Return from the interrupt.

Note: If the above flow is used, you must not read or write to the ICVectorAddr register.

To ensure that the vector address register (see "Vector Address Register, ICVectAddr" on page 47) can be read in a single instruction, the IC base address must be 0xFFFFF000, the upper 4K of memory. Placing the IC anywhere else in memory increases interrupt latency as the ARM processor is unable to access the ICVectorAddr register using a single instruction. The offset of any particular register from the base address is fixed.

Table 7-3 details the memory map for the interrupt controller.

Table 7-3. Interrupt Controller Memory Map

Address	Type	Width	Reset value	Name	Description
IC base + 0x000	Read	32	0x00000000	ICIRQStatus	IRQ status register
IC base + 0x004	Read	32	0x00000000	ICFIQStatus	FIQ status register
IC base + 0x008	Read	32	–	ICRawIntr	Raw interrupt status register
IC base + 0x00C	Read/Write	32	0x00000000	ICIntSelect	Interrupt select register
IC base + 0x010	Read/Write	32	0x00000000	ICIntEnable	Interrupt enable register
IC base + 0x014	Write	32	–	ICIntEnClear	Interrupt enable clear register
IC base + 0x018	Read/Write	32	0x00000000	ICSoftInt	Software interrupt register
IC base + 0x01C	Write	32	–	ICSoftIntClear	Software interrupt clear register
IC base + 0x020	Read/Write	1	0x0	ICProtection	Protection enable register
IC base + 0x030	Read/Write	32	0x00000000	ICVectAddr	Vector address register
IC base + 0x034	Read/Write	32	0x00000000	ICDefVectAddr	Default vector address register
IC base + 0x300	Read/Write	1	–	ICITCR	Test control register
IC base + 0x304	Read	2	–	ICITIP1	Test input register (nICIRQIN/nICFIQIN)
IC base + 0x308	Read	32	–	ICITIP2	Test input register (ICVECTADDRIN)
IC base + 0x30C	Read	2	0x0	ICITOP1	Test output register (nICIRQ/nICFIQ)
IC base + 0x310	Read	32	0x00000000	ICITOP2	Test output register (ICVECTADDROUT)
IC base + 0xFE0 to + 0xFFC	Read	8	0x08		RESERVED

IRQ Status Register, ICIRQStatus

The ICIRQStatus register provides the status of interrupts [31:0] after IRQ masking. Table 7-4 shows the bit assignment of the ICIRQStatus register.

Table 7-4. Bit Assignment of the ICIRQStatus Register

Bits	Name	Type	Function
31:0	IRQStatus	Read	Shows the status of the interrupts after masking by the ICIntEnable and ICIntSelect registers. A high bit indicates that the interrupt is active, and generates an interrupt to the processor.

FIQ Status Register, ICFIQStatus

The ICFIQStatus register provides the status of the interrupts after FIQ masking. Table 7-5 below shows the bit assignment of the ICFIQStatus register.

Table 7-5. Bit Assignment of the ICFIQStatus Register

Bits	Name	Type	Function
31:0	FIQStatus	Read	Shows the status of the interrupts after masking by the ICIntEnable and ICIntSelect registers. A high bit indicates that the interrupt is active, and generates an interrupt to the processor.

Raw Interrupt Status Register, ICRawIntr

The ICRawIntr register provides the status of the source interrupts (and software interrupts) to the interrupt controller. Table 7-6 shows the bit assignment of the ICRawIntr register.

Table 7-6. Bit Assignment of the ICRawIntr Register

Bits	Name	Type	Function
31:0	RawInterrupt	Read	Shows the status of the interrupts before masking by the enable registers. A high bit indicates that the appropriate interrupt request is active before masking.

Interrupt Select Register, ICIntSelect

The ICIntSelect register selects whether the corresponding interrupt source generates an FIQ or an IRQ interrupt. Table 7-7 shows the bit assignment of the ICIntSelect register.

Table 7-7. Bit Assignment of the ICIntSelect Register

Bits	Name	Type	Function
31:0	IntSelect	Read/write	Selects type of interrupt for interrupt request: 1 = FIQ interrupt 0 = IRQ interrupt

Interrupt Enable Register, ICIntEnable

The ICIntEnable register enables the interrupt request lines, by masking the interrupt sources for the IRQ interrupt. Table 7-8 shows the bit assignment of the ICIntEnable register.

Table 7-8. Bit Assignment of the ICIntEnable Register

Bits	Name	Type	Function
31:0	IntEnable	Read/Write	Enables the interrupt request lines: 1 = Interrupt enabled. Allows interrupt request to processor. 0 = Interrupt disabled On reset, all interrupts are disabled. A high bit sets the corresponding bit in the ICIntEnable register. A LOW bit has no effect.

Interrupt Enable Clear Register, ICIntEnClear

The ICIntEnClear register clears bits in the ICIntEnable register. Table 7-9 shows the bit assignment of the ICIntEnClear register.

Table 7-9. Bit Assignment of the ICIntEnClear Register

Bits	Name	Type	Function
31:0	IntEnable Clear	Write	Clears bits in the ICIntEnable register. A high bit clears the corresponding bit in the ICIntEnable register. A LOW bit has no effect.

Software Interrupt Register, ICSoftInt

The ICSoftInt register is used to generate software interrupts. Table 7-10 shows the bit assignment of the ICSoftInt register.

Table 7-10. Bit Assignment of the ICSoftInt Register

Bits	Name	Type	Function
31:0	SoftInt	Read/Write	Setting a bit generates a software interrupt for the specific source interrupt before interrupt masking. A high bit sets the corresponding bit in the ICSoftInt register. A LOW bit has no effect.

Software Interrupt Clear Register, ICSoftIntClear

The ICSoftIntClear register clears bits in the ICSoftInt register. Table 7-11 shows the bit assignment of the ICSoftIntClear register.

Table 7-11. Bit Assignment of the ICSoftIntClear Register

Bits	Name	Type	Function
31:0	SoftIntClear	Write	A high bit clears the corresponding bit in the ICSoftInt register. A LOW bit has no effect.

Protection Enable Register, ICProtection

The ICProtection register enables or disables protected register access. Table 7-12 shows the bit assignment of the ICProtection register.

Table 7-12. Bit Assignment of the ICProtection Register

Bits	Name	Type	Function
31:1	Reserved	–	–
0	Protection	Read/Write	Enables or disables protected register access. When enabled, only privileged mode accesses (reads and writes) can access the interrupt controller registers. When disabled, both User mode and privileged mode can access the registers. This register is cleared on reset, and can only be accessed in privileged mode.

Note: If the bus master cannot generate accurate protection information, leave this register in its reset state to allow User mode access.

Vector Address Register, ICVectAddr

The ICVectAddr register contains the ISR address of the currently active interrupt. Table 7-13 shows the bit assignment of the ICVectAddr register.

Table 7-13. Bit Assignment of the ICVectAddr Register

Bits	Name	Type	Function
31:0	VectorAddr	Read/Write	Contains the address of the currently active ISR. Any writes to this register clear the interrupt

Note: Reading from this register provides the address of the ISR, and indicates to the priority hardware that the interrupt is being serviced. Writing to this register indicates to the priority hardware that the interrupt has been serviced. The register should be used as follows:

- *The ISR reads the ICVectAddr register when an IRQ interrupt is generated*
- *At the end of the ISR, the ICVectAddr register is written to, to update the priority hardware.*

Reading or writing to the register at other times can cause incorrect operation.

Default Vector Address Register, ICDefVectAddr

The ICDefVectAddr register contains the default ISR address. Table 7-14 shows the bit assignment of the ICDefVectAddr register.

Table 7-14. Bit Assignment of the ICDefVectAddr Register

Bits	Name	Type	Function
31:0	Default VectorAddr	Read/Write	Contains the address of the default ISR handle

Test Control Register, ICITCR

The ICITCR register selects test mode, and is cleared on reset. Table 7-15 shows the bit assignment of the ICITCR register.

Table 7-15. Bit Assignment of the ICITCR Register

Bits	Name	Type	Function
31:1	Reserved	–	–
0	ITEN	Read/Write	Selects test mode, to use ICITIP test registers in place of input signals.

Test Input Register (nICIRQIN/nICFIQIN), ICITIP1

The ICITIP1 register indicates the status of the nICIRQIN and nICFIQIN daisy chain input lines. Table 7-16 shows the bit assignment of the ICITIP1 register.

Table 7-16. Bit Assignment of the ICITIP1 Register

Bits	Name	Type	Function
31:8	Reserved	–	–
7	I	Read	Indicates status of nICIRQIN when ICITCR register is LOW
6	F	Read	Indicates status of nICFIQIN when ICITCR register is LOW
5:0	Reserved		–

Test input register (ICVECTADDRIN), ICITIP2

The ICITIP2 register indicates the status of the ICVECTADDRIN daisy chain input lines. Table 7-17 shows the bit assignment of the ICITIP2 register.

Table 7-17. Bit Assignment of the ICITIP2 Register

Bits	Name	Type	Function
31:0	VectorAddrIn	Read/Write	Indicates status of ICVECTADDRIN when ICITCR register is LOW

Test Output Register (nICIRQ/nICFIQ), ICITOP1

The ICITOP1 register indicates the status of the nICIRQ and nICFIQ interrupt request lines to the processor. Table 7-18 shows the bit assignment of the ICITOP1 register.

Table 7-18. Bit Assignment of the ICITOP1 Register

Bits	Name	Type	Function
31:8	Reserved	–	–
7	I	Read	Status of nICIRQ interrupt line. If set high, the interrupt request is active.
6	F	Read	Status of nICFIQ interrupt line. If set high, the interrupt request is active.
5:0	Reserved		–

Test Output Register (ICVECTADDROUT), ICITOP2

The ICITOP2 register indicates the status of the ICVECTADDROUT lines from the interrupt controller. Table 7-19 shows the bit assignment of the ICITOP2 register.

Table 7-19. Bit Assignment of the ICITOP2 Register

Bits	Name	Type	Function
31:0	VectorAddr Out	Read	Indicates status of ICVECTADDROUT from interrupt controller

8 Watchdog

The watchdog unit provides a way of recovering from software crashes. The watchdog clock is used to generate a regular interrupt (WDOGINT), depending on a programmed value. The watchdog monitors the interrupt and asserts a reset signal (WDOGRES) if the interrupt remains unserviced for the entire programmed period. You can enable or disable the watchdog unit as required.

Connecting the Watchdog in CoreConsole

Table 8-1 lists the ports present on the watchdog and describes how to connect these in CoreConsole.

Table 8-1. Watchdog Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus
PCLK	PCLK	APB clock signal. Normally connected to the HCLK output of the MP7Bridge.
PRESETn	PRESETn	Active low APB reset input. Normally connected to the HRESETn output of the MP7Bridge.
Watchdog clock	WDOGCLK	Watchdog clock input. Normally connected to the HCLK output of the MP7Bridge.
Watchdog interrupt	WDOGINT	Active high interrupt output. Connect this to one of the interrupt source inputs (ICINTSOURCEx) of the Interrupt Controller.
Watchdog timeout reset	WDOGRES	Watchdog timeout output. This signal is asserted (high) if the watchdog times out. Connect this output to the WDOGRES input of the MP7Bridge.
Watchdog reset input	WDOGRESn	This input resets the Watchdog. Connect this input to the WDOGRESn port of the MP7Bridge.
Optional Connections		
Watchdog clock enable	WDOGCLKEN	Clock enable input for watchdog. This signal is tied high (asserted) if no connection is made to it.

Programmer's Model

Table 8-2 provides a list and description of the registers for the watchdog unit.

Table 8-2. Description of Registers for the WatchDog Unit

Address	Type	Width	Reset value	Name	Description
Wdog base + 0x00	Read/Write	32	0xFFFFFFFF	WdogLoad	Watchdog load register
Wdog base + 0x04	Read Only	32	0xFFFFFFFF	WdogValue	The current value for the watchdog counter
Wdog base + 0x08	Read/Write	2	0x0	WdogControl	Watchdog control register
Wdog base + 0x0C	Write Only	–	–	WdogIntClr	Clears the watchdog interrupt
Wdog base + 0x10	Read Only	1	0x0	WdogRIS	Watchdog raw interrupt status
Wdog base + 0x14	Read Only	1	0x0	WdogMIS	Watchdog masked interrupt status
Wdog base + 0xC00	Read/Write	32	0x0	WdogLock	Watchdog lock register
Wdog base + 0xF00	Read/Write	1	0x0	WdogITCR	Integration test control register
Wdog base + 0xF04	Write Only	2	0x0	WdogITOP	Integration test output set register
Wdog base + 0xFE0	Read Only	8	0x05	WdogPeriphID0	Peripheral ID register bits 7:0
Wdog base + 0xFE4	Read Only	8	0x18	WdogPeriphID1	Peripheral ID register bits 15:8
Wdog base + 0xFE8	Read Only	8	0x04	WdogPeriphID2	Peripheral ID register bits 23:16
Wdog base + 0xFEC	Read Only	8	0x00	WdogPeriphID3	Peripheral ID register bits 31:24
Wdog base + 0xFF0	Read Only	8	0x0D	WdogPCellID0	PrimeCell ID register bits 7:0
Wdog base + 0xFF4	Read Only	8	0xF0	WdogPCellID1	PrimeCell ID register bits 15:8
Wdog base + 0xFF8	Read Only	8	0x05	WdogPCellID2	PrimeCell ID register bits 23:16
Wdog base + 0xFFC	Read Only	8	0xB1	WdogPCellID3	PrimeCell ID register bits 31:24

Watchdog Load Register, WdogLoad

This is a 32-bit register containing the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for WdogLoad is one.

Watchdog Control Register, WdogControl

This is a read/write register that enables the software to control the watchdog unit. Table 8-3 shows the bit assignment of the WdogControl register.

Table 8-3. Bit Assignment of the Wdog Control Register

Bits	Name	Type	Function
31:2	–	–	Reserved, read undefined, must read as zeros
1	RESEN	Read/Write	Enable Watchdog reset output (WDOGRES). Acts as a mask for the reset output. Set high to enable the reset, and LOW to disable the reset.
0	INTEN	Read/Write	Enable the interrupt event (WDOGINT). Set high to enable the counter and the interrupt, and set LOW to disable the counter and interrupt. Reloads the counter from the value in WdogLoad when the interrupt is enabled, and was previously disabled.

Watchdog Clear Interrupt Register, WdogIntClr

A write of any value to this location clears the watchdog interrupt, and reloads the counter from the value in WdogLoad.

Raw interrupt Status Register, WdogRIS

This register indicates the raw interrupt status from the counter. This value is ANDed with the interrupt enable bit from the control register to create the masked interrupt, which is passed to the interrupt output pin. Table 8-4 shows the bit assignment of the WdogRIS register.

Table 8-4. Bit Assignment of the WdogRIS Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Raw Watchdog Interrupt	Read	Raw interrupt status from the counter

Interrupt Status Register, WdogMIS

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the INTEN bit from the control register, and is the same value which is passed to the interrupt output pin. Table 8-5 shows the bit assignment of the WdogMIS register.

Table 8-5. Bit Assignment of the WdogMIS Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Watchdog Interrupt	Read	Enabled interrupt status from the counter

Watchdog Lock Register, WdogLock

Use of this register enables write-access to all other registers to be disabled. This is to prevent rogue software from disabling the watchdog functionality. Writing a value of 0x1ACCE551 will enable write access to all other registers; writing any other value will disable write accesses. A read from this register will return only the bottom bit:

- 0 indicates that write access is enabled (not locked).
- 1 indicates that write access is disabled (locked).

Table 8-6 shows the bit assignment of the WdogLock register.

Table 8-6. Bit Assignment of the WdogLock Register

Bits	Name	Type	Function
31:0	Enable register writes	Write	Enable write access to all other registers by writing 0x1ACCE551. Disable write access by writing any other value.
0	Register write enable status	Read	0 = write access to all other registers is enabled (default) 1 = write access to all other registers is disabled

Integration Test Control Register, WdogITCR

Single-bit register used to enable integration test mode. When in this mode, the masked interrupt output, WDOGINT, and reset output, WDOGRES, are directly controlled by the test output set register. Table 8-7 shows the bit assignment of the WdogITCR register.

Table 8-7. Bit Assignment of the WdogITCR Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Integration Test Mode Enable	Read/Write	When set high, places the Watchdog into integration test mode

Integration Test Output Set Register, WdogITOP

When in integration test mode, the enabled interrupt output and reset output are driven directly from the values in this register. Table 8-8 shows the bit assignment of the WdogITOP register.

Table 8-8. Bit Assignment of the Wdog ITOP Register

Bits	Name	Type	Function
31:2	–	–	Reserved, read undefined, must read as zeros
1	Integration Test WDOGINT value	Write	Value output on WDOGINT when in Integration Test Mode
0	Integration Test WDOGRES value	Write	Value output on WDOGRES when in Integration Test Mode

9 Timers

The Dual Input Timers module is an APB slave that provides access to two interrupt-generating, programmable 32-bit Free-Running Decrementing Counters (FRCs). The system clock (PCLK) is used to control the programmable registers, and a second clock input (TIMCLK) is used to drive the counter, enabling the counters to run from a much slower clock than the system clock. The two clocks must be synchronous while register accesses are performed. A top-level block diagram of the timers is shown in Figure 9-1 below.

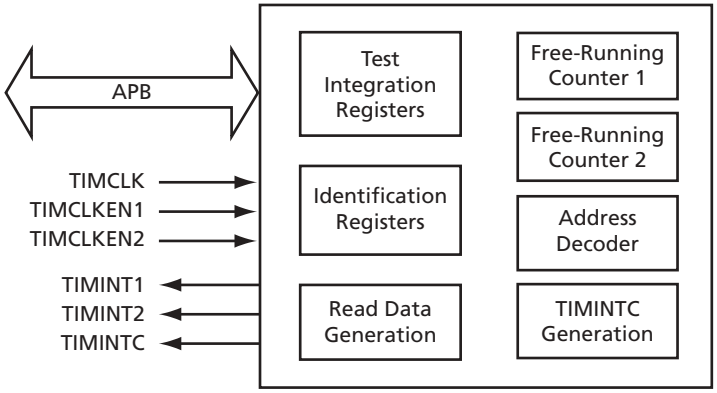

Figure 9-1. Dual Input Timer Block Diagram

Functional Description

Two timers are provided in the Timers module. For each timer, the following modes of operation are available: free-running mode, periodic mode, and one-shot timer mode.

Free-Running Mode

The counter wraps after reaching its zero value, and continues to count down from the maximum value. This is the default mode.

Periodic Timer Mode

The counter generates an interrupt at a constant interval, reloading the original value after wrapping past zero.

One-Shot Timer Mode

The counter generates an interrupt once. When the counter reaches zero, it halts until re-programmed by the user. This can be achieved by either clearing the One-Shot Count bit in the

control register (in which case the count will proceed according to the selection of free-running or periodic mode), or by writing a new value to the Load Value register.

Operation

Each timer has an identical set of registers. The operation of each timer is identical. The timer is loaded by writing to the load register and, if enabled, counts down to zero. When a counter is already running, writing to the load register will cause the counter to immediately restart at the new value. Writing to the background load value has no effect on the current count. The counter continues to decrement to zero, and then recommences from the new load value (if in periodic mode, and one-shot mode is not selected).

When zero is reached, an interrupt is generated. The interrupt can be cleared by writing to the clear register. If one-shot mode is selected, the counter halts on reaching zero until you deselect one-shot mode, or write a new load value. Otherwise, after reaching a zero count, if the timer is operating in free-running mode it continues to decrement from its maximum value. If periodic timer mode is selected, the timer reloads the count value from the load register and continues to decrement. In this mode the counter effectively generates a periodic interrupt. The mode is selected by a bit in the timer control register. At any point, the current counter value can be read from the value register. The counter is enabled by a bit in the control register. At reset, the counter is disabled, the interrupt is cleared, and the load register is set to zero. The mode and prescale values are set to free-running, and clock divide of 1 respectively. A block diagram of the free-running timer module is shown in Figure 9-2.

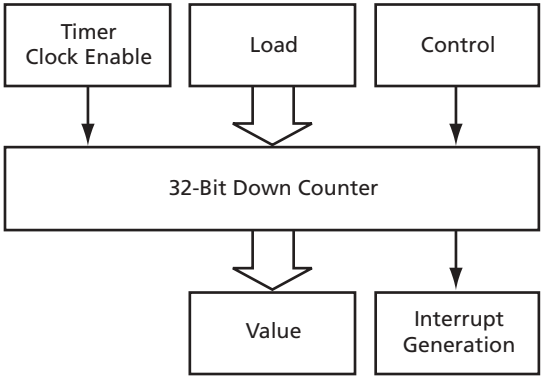

Figure 9-2. Free Running Timer Block

The timer clock enable is generated by a prescale unit. The enable is then used by the counter to create a clock with a timing of one of the following:

- The system clock
- The system clock divided by 16, generated by 4 bits of prescale
- The system clock divided by 256, generated by a total of 8 bits of prescale

Figure 9-3 shows how the timer clock frequency is selected in the prescale unit. This enables you to clock the timer at different frequencies.

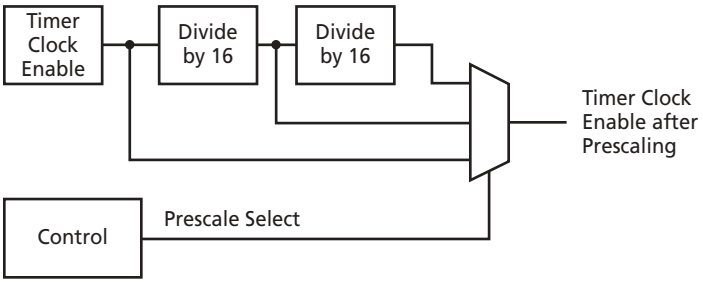

Figure 9-3. Prescale Clock Enable Generation

Interrupt Generation

An interrupt is generated when the full 32-bit counter reaches zero, and is only cleared when the TimerXClear location is written to. A register holds the value until the interrupt is cleared. The most significant carry bit of the counter detects the counter reaching zero.

Interrupts can be masked by writing 0 to the Interrupt Enable bit in the Control register. Both the raw interrupt status (prior to masking) and the final interrupt status (after masking) can be read from status registers.

The interrupts from the individual counters (after masking) are logically OR'ed into a combined interrupt, TIMINTC, which is provided as an additional output from the Timer peripheral.

Clocking

The timers have two clock inputs, PCLK and TIMCLK. PCLK is the main APB system clock, and is used by the register interface. TIMCLK is the input to the prescale units and the decrementing counters. A pulse on TIMCLK must be qualified by the appropriate TIMCLKENx being high.

The design of the timers assumes that PCLK and TIMCLK are synchronous. To enable the counter to operate from a lower effective frequency than that at which PCLK is running, either of the following can be done:

- Both PCLK and TIMCLK inputs are connected to the APB PCLK signal, and TIMCLKENx is pulsed high at the required frequency (synchronized to PCLK).
- TIMCLKENx is tied high and an enabled version of PCLK is fed into the TIMCLK input, giving sparse clock pulses synchronous to PCLK.

This provision of two clock inputs enables the counters to continue to run while the APB system is in a sleep state whereby PCLK is disabled. The changeover periods when PCLK is disabled and enabled must be handled by external system control logic, to ensure that the PCLK and TIMCLK inputs are fed with synchronous signals when any register access is to occur.

Connecting the Timers Module in CoreConsole

Table 9-1 lists the ports present on the Timers module and describes how to connect these in CoreConsole.

Table 9-1. Timers Module Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus.
PCLK	PCLK	APB clock signal. Normally connected to the HCLK output of the MP7Bridge.
PRESETn	PRESETn	Active low APB reset input. Normally connected to the HRESETn output of the MP7Bridge.
Timer clock	TIMCLK	Clock input for timers. This must be synchronous to PCLK for normal operation. This may be connected to the HCLK output of the MP7Bridge or to a separate (typically slower) clock signal.

Table 9-1. Timers Module Connections (Continued)

Connection	CoreConsole Label	Description
Optional Connections		
Timer 1 interrupt	TIMINT1	Active high interrupt output for timer 1. This signal indicates an interrupt has been generated by counter 1 having being decremented to zero. This is normally connected to one of the interrupt source (ICINTSOURCEx) inputs of the Interrupt Controller.
Timer 2 interrupt	TIMINT2	Active high interrupt output for timer 2. This signal indicates an interrupt has been generated by counter 2 having being decremented to zero. This is normally connected to one of the interrupt source (ICINTSOURCEx) inputs of the Interrupt Controller.
Combined interrupt for timers	TIMINTC	This active high interrupt output is a combination of the two timer interrupt signals. This signal indicates an interrupt has been generated by either counter having being decremented to zero, and is the logical OR of TIMINT1 and TIMINT2. This is normally connected to one of the interrupt source (ICINTSOURCEx) inputs of the Interrupt Controller.
Timer 1 clock enable	TIMCLKEN1	Clock enable input for timer 1. The counter will only decrement on a rising edge of TIMCLK when TIMCLKEN1 is high. This input will be tied high (asserted) if no connection is made to it.
Timer 2 clock enable	TIMCLKEN2	Clock enable input for timer 2. The counter will only decrement on a rising edge of TIMCLK when TIMCLKEN2 is high. This input will be tied high (asserted) if no connection is made to it.

Programmer's Model

The Timer Registers are shown in Table 9-2.

Table 9-2. Timer Registers

Address	Type	Width	Reset value	Name	Description
Timer base + 0x00	Read/Write	32	0x00000000	Timer1Load	Load value for Timer1
Timer base + 0x04	Read	32	0xFFFFFFFF	Timer1Value	The current value for Timer1
Timer base + 0x08	Read/Write	8	0x20	Timer1Control	Timer 1 control register
Timer base + 0x0C	Write	–	–	Timer1IntClr	Timer 1 interrupt clear
Timer base + 0x10	Read	1	0x0	Timer1RIS	Timer 1 raw interrupt status
Timer base + 0x14	Read	1	0x0	Timer1MIS	Timer 1 masked interrupt status
Timer base + 0x18	Read/Write	32	0x00000000	Timer1BGLoad	Background load value for Timer 1
Timer base + 0x20	Read/Write	32	0x00000000	Timer2Load	Load value for Timer 2
Timer base + 0x24	Read	32	0xFFFFFFFF	Timer2Value	The current value for Timer 2
Timer base + 0x28	Read/Write	8	0x20	Timer2Control	Timer 2 control register
Timer base + 0x2C	Write	–	–	Timer2IntClr	Timer 2 interrupt clear
Timer base + 0x30	Read	1	0x0	Timer2RIS	Timer 2 raw interrupt status
Timer base + 0x34	Read	1	0x0	Timer2MIS	Timer 2 masked interrupt status
Timer base + 0x38	Read/Write	32	0x00000000	Timer2BGLoad	Background load value for Timer 2
Timer base + 0xF00	Read/Write	1	0x0	TimerITCR	Integration test control register
Timer base + 0xF04	Write	2	0x0	TimerITOP	Integration test output set register
Timer base + 0xFE0 to 0xFFC	Read	8	0x04	RESERVED	RESERVED

Load register, TimerXLoad

This is a 32-bit register containing the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.

When this register is written to directly, the current count is immediately reset to the new value at the next rising edge of TIMCLK which is enabled by TIMCLKEN.

The value in this register is also overwritten if the TimerXBGLoad register is written to, but the current count is not immediately affected.

If values are written to both the TimerXLoad and TimerXBGLoad registers before an enabled rising edge on TIMCLK, the following occurs:

- On the next enabled TIMCLK edge the value written to the TimerXLoad value replaces the current count value.
- Following this, each time the counter reaches zero, the current count value is reset to the value written to TimerXBGLoad.

Reading from the TimerXLoad register at any time after the two writes have occurred will retrieve the value written to TimerXBGLoad. That is, the value read from TimerXLoad is always the value which will take effect for Periodic mode after the next time the counter reaches zero.

Current Value Register, TimerXValue

This register gives the current value of the decrementing counter.

Timer Control Register, TimerXControl

The bit assignments for the TimerXControl register are shown in Table 9-3.

Table 9-3. Bit Assignments for the TimerXControl Register

Bits	Name	Type	Function
31:8	–	–	Reserved, read undefined, must read as zeros
7	Timer Enable	Read/Write	Enable bit: 0 = Timer disabled (default) 1 = Timer enabled
6	Timer Mode	Read/Write	Mode bit: 0 = Timer is in free-running mode (default) 1 = Timer is in periodic mode
5	Interrupt Enable	Read/Write	Interrupt Enable bit: 0 = Timer Interrupt disabled 1 = Timer Interrupt enabled (default)
4	RESERVED		Reserved bit, do not modify, and ignore on read
3:2	TimerPre	Read/Write	Prescale bits: 00 = 0 stages of prescale, clock is divided by 1 (default) 01 = 4 stages of prescale, clock is divided by 16 10 = 8 stages of prescale, clock is divided by 256 11 = Undefined, do not use.
1	Timer Size	Read/Write	Selects 16/32 bit counter operation: 0 = 16-bit counter (default) 1 = 32-bit counter
0	One Shot Count	Read/Write	Selects one-shot or wrapping counter mode: 0 = wrapping mode (default) 1 = one-shot mode

Interrupt Clear Register, TimerXIntClr

Any write to this register will clear the interrupt output from the counter.

Raw Interrupt Status Register, TimerXRIS

This register indicates the raw interrupt status from the counter. This value is ANDed with the timer interrupt enable bit from the control register to create the masked interrupt, which is passed to the interrupt output pin. Table 9-4 shows the bit assignments for the TimerXRIS register.

Table 9-4. Bit Assignments for the TimerXRIS Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Raw Timer Interrupt	Read	Raw interrupt status from the counter

Interrupt Status Register, TimerXMIS

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the timer interrupt enable bit from the control register, and is the same value which is passed to the interrupt output pin. Table 9-5 shows the bit assignments for the TimerXMIS register.

Table 9-5. Bit Assignments for the TimerXMIS Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Timer Interrupt	Read	Enabled interrupt status from the counter

Background Load Register, TimerXBGLoad

This is a 32-bit register containing the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.

This register provides an alternative method of accessing the TimerXLoad register. The difference is that writes to TimerXBGLoad will not cause the counter immediately to restart from the new value.

Reading from this register returns the same value returned from TimerXLoad. See “Load register, TimerXLoad” on page 62 for more details.

Integration Test Control Register, TimerITCR

This is a single-bit register used to enable integration test mode. When in this mode, the masked interrupt outputs are directly controlled by the test output set register. The combined interrupt output TIMINTC then becomes the logical OR of the bits set in the test output set register. Table 9-6 shows the bit assignments for the TimerITCR register.

Table 9-6. Bit Assignments for the TimerITCR Register

Bits	Name	Type	Function
31:1	–	–	Reserved, read undefined, must read as zeros
0	Integration Test Mode Enable	Read/Write	When set high, places the Timers into integration test mode

Integration Test Output Set Register, TimerITOP

When in integration test mode, the enabled interrupt outputs are driven directly from the values in this register. Table 9-7 shows the bit assignments for the TimerITOP register.

Table 9-7. Bit Assignments for the TimerITOP Register

Bits	Name	Type	Function
31:2	–	–	Reserved, read undefined, must read as zeros
1	Integration Test TIMINT2 value	Write	Value output on TIMINT2 when in Integration Test Mode
0	Integration Test TIMINT1 value	Write	Value output on TIMINT1 when in Integration Test Mode

10 System Control Block

This APB slave is a small control block that has a single bit register which is intended to be used to control aliasing of memory resources at the bottom of the processor address space. Typically the (non-volatile) Flash is aliased to the bottom of the memory map by default but the SRAM may be made to appear at the base of the address space by setting Remap high.

The RemapDefault input determines the value of the Remap output following a reset.

The Remap output of the System Control module is normally connected to the Remap input of the Memory Controller. The RemapDefault input may be connected to the toplevel of the subsystem to provide a means of controlling the reset value of Remap.

Connecting the System Control Block in CoreConsole

Table 10-1 lists the ports present on the System Control block and describes how to connect these in CoreConsole.

Table 10-1. System Control Block Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus.
PCLK	PCLK	APB clock signal. Normally connected to the HCLK output of the MP7Bridge.
PRESETn	PRESETn	Active low APB reset input. Normally connected to the HRESETn output of the MP7Bridge.
Remap	Remap	This output is driven by an internal control register bit and is intended to be used for controlling memory aliasing. This signal should be connected to the Remap input of the Memory Controller.
Optional Connections		
Default setting for Remap	RemapDefault	This input determines the value of the Remap output following a reset. This signal may be connected to the subsystem toplevel to allow external control of memory aliasing after reset. If no connection is made to this port, it will be tied low.

Programmer's Model

The System Control block contains a single register at offset 0x00 (and aliased throughout the slot) which is described in Table 10-2.

Table 10-2. System Control Register

Bits	Name	Type	Function
31:1	–	–	Reserved
0	Remap	Read/Write	Control bit which drives Remap output

11 General Purpose I/O (GPIO) Block

The GPIO block is an APB peripheral which provides 32 inputs and 32 outputs. There is a single register at offset 0x00 (and aliased throughout the slot) which is cleared on reading. Writing to this register writes 32 bits to the outputs, reading from the register reads the state of the inputs.

It is not required that all inputs and outputs are used. The user need only connect to those inputs and outputs which are actually used. Any unused inputs should be tied either high or low and unused outputs may be left unconnected.

Connecting the GPIO Block in CoreConsole

Table 11-1 lists the ports present on the GPIO block and describes how to connect these in CoreConsole.

Table 11-1. GPIO Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus.
PCLK	PCLK	APB clock signal. Normally connected to the HCLK output of the MP7Bridge.
PRESETn	PRESETn	Active low APB reset input. Normally connected to the HRESETn output of the MP7Bridge.
Optional Connections		
Input data	dataIn	32 bit input data bus. This port provides the user with up to 32 inputs. Any unused input lines should be tied to a fixed value (high or low).
Output data	dataOut	32 bit output data bus. This port provides up to 32 outputs. Any unused outputs may be left unconnected.

Programmer's Model

The GPIO block contains a single 32 bit register which is described in Table 11-2.

Table 11-2. GPIO Control/Status Register

Bits	Name	Type	Function
31:0	I/O Control/ Status Register	Read/Write	A write to this register will set the values of the output lines. Reading this register gives the status of the input lines. The status is cleared on read.

12

Flash ROM (FROM) Access Block

The FROM Access block is an APB slave which provides a means to access the 128-byte FROM which is available in Actel ProASIC3/E devices.

If you want to access the on-chip FROM, add a FROM Access component to your design and connect it to the APB Bus. Aside from the connections to the APB Bus, there are two other ports on the FROM Access component. These are named fromAddr (7-bit FROM address bus) and fromData (8-bit FROM data bus) and should be routed to the toplevel of your subsystem design in CoreConsole for subsequent connection to a FROM instance.

The user should use the SmartGen core generator, available in the Actel Libero® Integrated Design Environment (IDE), to generate a FROM instance and to set the values to be programmed into the FROM.

More information on using the FROM on ProASIC3/E devices can be found in the following Application Note:

http://www.actel.com/documents/PA3_E_FROM_AN.pdf

Connecting the FROM Access Block in CoreConsole

Table 12-1 lists the ports present on the FROM Access block and describes how to connect these in CoreConsole.

Table 12-1. FROM Access Connections

Connection	CoreConsole Label	Description
Required Connections		
APB slave interface	APBslave	Connect this interface to any available slave slot on the APB Bus.
FROM address bus	fromAddr	7-bit FROM address bus. This port should be connected to the subsystem toplevel for subsequent connection to a FROM instance.
FROM data bus	fromData	8-bit FROM data bus. This port should be connected to the subsystem toplevel for subsequent connection to a FROM instance.

Programmer's Model

When the FROM is connected to the FROM Access component, the contents of the FROM appear at the base address of the APB slot where the FROM Access is located. The FROM contents are also aliased throughout this APB slot.

The data stored in the FROM is read only and cannot be changed by the processor.

The FROM data can only be accessed one byte at a time; it is not possible to read a word or half-word of FROM data in one step.

Table 12-2 shows how the FROM data appears to the processor.

Table 12-2. FROM data

Offset	Data			
Base address + 0x00	Byte 3	Byte 2	Byte 1	Byte 0
Base address + 0x04	Byte 7	Byte 6	Byte 5	Byte 4
Base address + 0x08	Byte 11	Byte 10	Byte 9	Byte 8
Base address + 0x0C	Byte 15	Byte 14	Byte 13	Byte 12
Base address + 0x10	Byte 19	Byte 18	Byte 17	Byte 16
Base address + 0x14	Byte 23	Byte 22	Byte 21	Byte 20
Base address + 0x18	Byte 27	Byte 26	Byte 25	Byte 24
Base address + 0x1C	Byte 31	Byte 30	Byte 29	Byte 28
Base address + 0x20	Byte 35	Byte 34	Byte 33	Byte 32
Base address + 0x24	Byte 39	Byte 38	Byte 37	Byte 36
Base address + 0x28	Byte 43	Byte 42	Byte 41	Byte 40
Base address + 0x2C	Byte 47	Byte 46	Byte 45	Byte 44
Base address + 0x30	Byte 51	Byte 50	Byte 49	Byte 48
Base address + 0x34	Byte 55	Byte 54	Byte 53	Byte 52
Base address + 0x38	Byte 59	Byte 58	Byte 57	Byte 56
Base address + 0x3C	Byte 63	Byte 62	Byte 61	Byte 60
Base address + 0x40	Byte 67	Byte 66	Byte 65	Byte 64
Base address + 0x44	Byte 71	Byte 70	Byte 69	Byte 68

Table 12-2. FROM data (Continued)

Offset	Data			
Base address + 0x48	Byte 75	Byte 74	Byte 73	Byte 72
Base address + 0x4C	Byte 79	Byte 78	Byte 77	Byte 76
Base address + 0x50	Byte 83	Byte 82	Byte 81	Byte 80
Base address + 0x54	Byte 87	Byte 86	Byte 85	Byte 84
Base address + 0x58	Byte 91	Byte 90	Byte 89	Byte 88
Base address + 0x5C	Byte 95	Byte 94	Byte 93	Byte 92
Base address + 0x60	Byte 99	Byte 98	Byte 97	Byte 96
Base address + 0x64	Byte 103	Byte 102	Byte 101	Byte 100
Base address + 0x68	Byte 107	Byte 106	Byte 105	Byte 104
Base address + 0x6C	Byte 111	Byte 110	Byte 109	Byte 108
Base address + 0x70	Byte 115	Byte 114	Byte 113	Byte 112
Base address + 0x74	Byte 119	Byte 118	Byte 117	Byte 116
Base address + 0x78	Byte 123	Byte 122	Byte 121	Byte 120
Base address + 0x7C	Byte 127	Byte 126	Byte 125	Byte 124

A

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

For more information about Actel's products, visit our website at http://www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 USA*
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL • United Kingdom*
Phone +44 (0) 1276 401 450 • Fax +44 (0) 1276 401 490

Actel Japan *• EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • www.jp.actel.com

Actel Hong Kong *• Suite 2114, Two Pacific Place • 88 Queensway, Admiralty Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

50200058-1/12.05