

7 MIPS PROCESSOR MODULES

The MIPS processor modules provide the capability of integrating user-developed application software into an embedded system. The MIPS processors support a wide variety of architectural choices. The WM Kernel provides a Unix-like environment on the MIPS processors, with support from the host processor running the WM Kernel Server.

Each MIPS processor module includes local RAM, accessible from the PCI Local Bus, for program code and data. The shared global RAM on the V6M6 baseboard is accessible from the modules.

The MIPS processor modules provide full access to the V6M6 PCI Local Bus and to the V6M6 TDM buses. Host-based library functions provide

- Initialization and run-time control of the PCI configuration space on the modules.
- Initialization of the TDM configuration on the modules.
- Download of code and data to run on the modules.

The WM Kernel Server is a host application that provides

- Download and initialization of the WM Kernel run-time software on the modules.
- File and socket I/O services requested by the WM Kernel for applications running on the MIPS.

7.1 PM4700A Module

7.1.1 PM4700A Quick Tour

Figure 7.1-1 is a block diagram of the PM4700A showing the major components and data paths

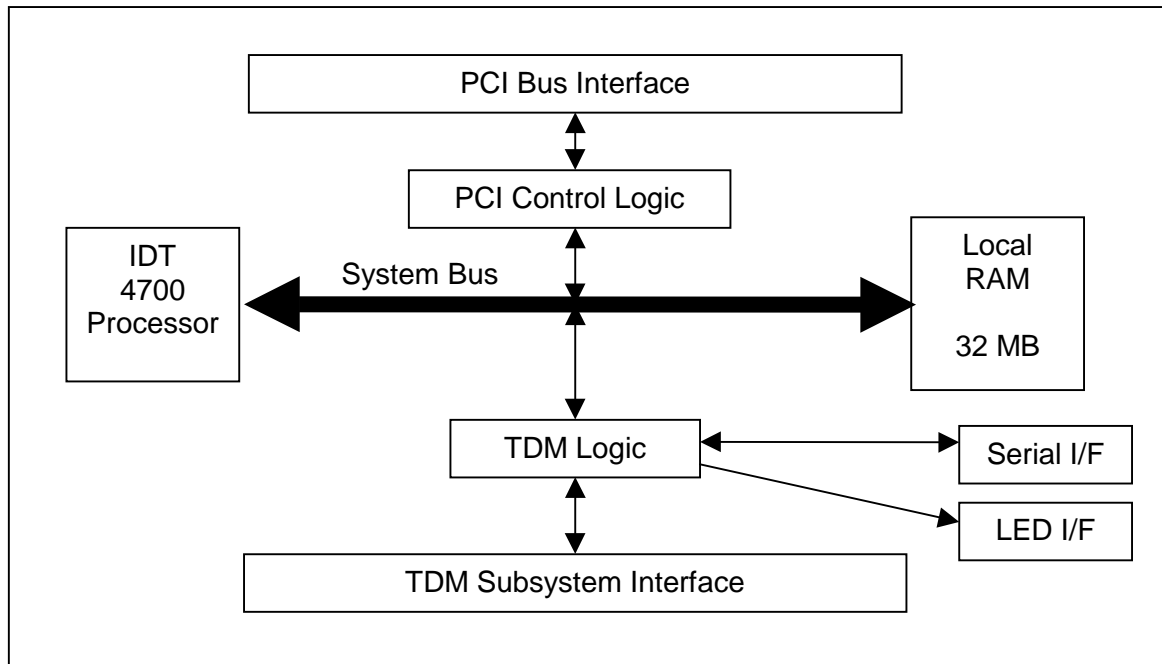


Figure 7.1-1: PM4700A Module Block Diagram

- **Major Components**

The IDT R4700 MIPS RISC Processor, running at 175 (or 200) MHz, delivers 175 (200) MIPS, or 87.5 (100) MFLOPS. The processor is a true 64-bit architecture, with 32 general-purpose registers and 32 floating point registers, separate 16 KB instruction and data caches, a flexible MMU with a large TLB, and flexible data formats, including a choice of big-endian or little-endian memory organization.

The System Bus, with 64 bits of data and 9 bits of command and data identifier, runs at a quarter of the processor clock frequency.

The Local RAM is 32 megabytes of EDO DRAM. It supports 64-bit access from the 64-bit System Bus and the 32-bit PCI Local Bus.

The PCI Bus Interface allows the module to act as a Bus Master or Target on the V6M6 PCI Local Bus. The local RAM is mapped by the module's

Configuration Space registers to an externally controllable range in the PCI memory address space.

The TDM Bus Interface allows the module to use the V6M6 Time Division Multiplex bus for low latency module-to-module data transmission.

A three-color LED interface controls one of six front-panel LEDs on the V6M6 board for module status indications.

A serial communications controller provides debugging facilities through an optional front-panel RS232 interface.

- **Module Options**

The PM4700A is available in 175 MHz and 200 MHz versions.

The front-panel RS232 interface is optional.

- **PCI Bus Interface**

The PCI Bus Interface logic maps 64-bit module data to and from the 32-bit PCI Local Bus. It supports the PCI Configuration Space, which allows V6M6 PCI modules and the host processor, via the VME bus and VME-PCI bridge, to set the operating modes and PCI memory base address for the PM4700A module at initialization, and to control the interrupts to and from the module.

- **TDM Interface**

The TDM Bus Interface drives data for selected time slots on the TDM bus, and receives data for other time slots according to the TDM configuration set by the host. The PM4700A processor writes the output data to the TDM bus interface logic, specifying the time slot and the data value, and reads the time slot and data value from the interface logic for input data. Hardware FIFOs buffer the data between the processor and the TDM buses. Flexibly configurable parameters allow the TDM bus interface logic to interrupt the processor at appropriate points to process input data and provide output data.

- **Host and Module Interrupts**

The host processor can generate a PM4700A processor interrupt through the PCI Configuration space access, and other PCI modules can generate a processor interrupt through the Module Interrupt capability supported by the

V6M6 baseboard (Version 2 and Version 3 differ in the control of this interrupt). The module can generate PCI Interrupt A for off-module interactions.

The processor can configure the module to provide interrupts for TDM synchronization signals, TDM FIFO boundary (and near-boundary) conditions, events on the serial interface, and the processor's internal timer.

7.1.2 PM4700A Hardware Details

The PM4700A PCI module consists of the mini-PCI circuit board, the processor (one chip), memory subsystem (4 64-megabit EDO DRAM chips), TDM FIFOs (one chip), serial communications level converter (one chip), two Field-Programmable Gate Array (FPGA) chips, and various transceiver, frequency generator, and voltage converter chips. The FPGAs implement the DRAM controller, module registers, bus interfaces, serial communications controller, and control logic described in the remainder of this section.

The module interfaces to the V6M6 through two connectors that implement the PCI electrical interface (excluding 64-bit expansion, cache snooping, and interrupt C and D signals), the TDM interface, a module interrupt signal, LED control signals, and initialization signals. An optional front-panel RS232 interface uses a 5-pin latching Molex connector; if equipped with the connector, the module is restricted to sites A and C, which provide access to the V6M6 front panel.

7.1.2.1 Module Resources

7.1.2.1.1 Processor

The PM4700A PCI module processor is an IDT Orion R4700. The processor is documented in the "IDT79R4600 and IDT79R4700 ORION Processor Hardware User's Manual" published by Integrated Device Technology, Inc. References to "PHUM" identify the relevant sections of the manual.

The remainder of section 7.1 assumes familiarity with the processor.

7.1.2.1.1.1 System Bus

The R4700 System Interface defines the PM4700A system address/data bus, which supports data transfers to and from the other module, baseboard, and system resources:

- local memory (64 bits)
- module registers (various widths)
- the PCI Local Bus (32 bits)
- the TDM buses (8 bits)

7.1.2.1.1.2 Processor Interrupts and Reset Conditions

The processor's external interrupts provide exception conditions for module-related events and conditions. Interrupts 0 through 4 and NMI [see PHUM 5 and 13] are defined as follows:

Interrupt	Description	Cause
0	PCI	Externally controlled through Module Interrupt Register
1	TDM	TDM FIFO threshold
2	Miscellaneous	TDM Superframe Sync Signal or Module Interrupt
3	Serial I/F	Clear-to-send received or character transfer complete
4	Spare	Currently not controllable
5	Spare/Internal timer	Not connected, used internally for timer interrupt
NMI	Non-Maskable Interrupt	PCI parity error

PCI Interrupt

The PCI Interrupt allows other PCI modules and the VME host to interrupt the PM4700A module by writing to the Module Interrupt Register (described in section 7.1.2.1.4.4, below).

TDM Interrupt

The TDM Interrupt may be caused by the TDM logic when the TDM input (receive) FIFO fills above the "almost empty" threshold and/or the TDM output (transmit) FIFO falls below the "almost full" threshold. The TDM Configuration Register bits [5:4] control which conditions may cause the interrupt. (Section 7.1.2.1.5.5, below, describes the logic more fully.)

Miscellaneous Interrupt

The Miscellaneous Interrupt is caused by:

- the PCI Module Interrupt line, if the corresponding enable bit in the SCC Interrupt Enable register (section 7.1.2.1.5.3) is 1, or
- the TDM Superframe Sync signal, controlled by the TDM Configuration Register (section 5.2.2.1.5.5).

The PCI Module Interrupt is generally cleared by clearing the source of the interrupt; how this is done depends on the inter-module communications

conventions established for a particular system. The interrupt can also be disabled in the SCC Interrupt Enable Register (section 7.1.2.1.5.3).

The TDM Superframe Sync signal is reset by writing 1 to bit 1 in the module's Superframe Sync Interrupt Clear Register (section 7.1.2.1.5.8).

Serial Interface Interrupt

The Serial Interface Interrupt is generated by any of the following events if the corresponding bit is 1 in the SCC Interrupt Enable Register, SCCIE (section 7.1.2.1.5.3):

- CTS, the RS232 Clear To Send signal becoming true, masked by SCCIE bit 3.
- Receive data full, indicating that all eight bits of the input character buffer have been received, masked by SCCIE bit 1.
- Transmit data empty, indicating that all eight bits of the output character have been sent, masked by SCCIE bit 0.

NMI and Error Conditions

Several error conditions may occur as a result of PCI Local Bus operations. Normally, the error flag (bit 5 of the System Interface Command Bus) signals the failure to transfer good data, causing the processor to take a Bus Error exception. The Non-Maskable Interrupt (NMI) is caused if timing precludes using the Bus Error, which must be caused synchronously. NMI occurs for write errors, parity errors, and errors on doublewords after the first in cache line fills. The conditions causing errors are:

- PCI errors while the module is acting as initiator:
 - Target abort received.
 - Device timeout, if a PCI transaction fails to complete without being aborted (perhaps when a target asserts DEVSEL# to claim a cycle but deasserts it before completing all data phases).
- Processor errors, reflected in the corresponding bits of the Module Status Register (section 7.1.2.1.4.3):
 - Invalid PCI access, if the processor attempts a PCI access when the Bus Master bit is not set in the PCI Command Register (section 7.1.2.1.4.1).
 - Processor timeout, if a local memory transaction fails to complete.
 - Register Burst, if the processor attempts a burst access to configuration space.
- Bad parity detected on the PCI Local Bus when the module is the bus master, if the Parity Error Response bit is set in the PCI Command Register.
- Bad parity in the data being driven on the PCI Local Bus by the module.

Reset Conditions

Several different reset conditions may occur:

- Power-on start-up causes the processor initialization string to be read in. See [PHUM 9]. It is normally followed by the PCI Reset sequence.
- The PCI Reset signal, from the PCI Local Bus, causes a general module initialization. It results in the processor taking the Reset Exception.
- The Processor Reset condition, controlled through the PCI Module Command Register, causes the processor Soft Reset Exception.

The register descriptions in succeeding sections associate initial states either with "PCI Reset" or with "Processor Reset" according to which of these conditions causes initialization.

7.1.2.1.2 Devices

7.1.2.1.2.1 Serial Communication Port

The PM4700A is optionally equipped with a front-panel RS232 port for debugging. The port is managed through the module's TDM Registers, which provide the data, status, and control interface, and the SCC Interrupt.

7.1.2.1.2.2 TDM Buses

The PM4700A implements a standard interface to the V6M6 TDM buses, which provide isochronous communications among the PCI modules on one or more V6M6 boards. The configuration of the time slots in the TDM cycle is described in section 3.5 of this manual. Data to or from all of the TDM bus slots allocated to the module pass sequentially through the TDM FIFOs with slot and control information attached. Control and status information is transferred with the data through the module's TDM Registers, supported by the TDM Interrupt.

7.1.2.1.2.3 LED

The V6M6 provides a three-color LED for each PCI module. The PM4700A module controls its LED through a field in one of the TDM Registers.

7.1.2.1.3 Memory

This section describes the physical memory spaces implemented by the PM4700A module. The physical addresses noted in this section are generated within the IDT R4700 processor by the virtual-to-physical address mapping mechanism [described in PHUM section 4]. Typically, memory mapping tables (page tables) are created and maintained by memory management software, and are used by exception-handling software to build the TLB entries used in virtual address translation.

7.1.2.1.3.1 Local RAM

The PM4700A PCI module includes 32 Megabytes of EDO DRAM as local RAM for the R4700 processor. It is addressable from the PCI Local Bus in the PCI Memory Space, starting at the address specified in the Module Base Address Register, and from the R4700 processor starting at physical address 0x10000000.

7.1.2.1.3.2 PCI I/O Space

The PM4700A Module does not define a PCI I/O space, and therefore does not recognize or respond to any I/O reads or I/O writes on the PCI Local Bus. (It is, however, able to generate I/O space accesses on the PCI Local Bus, and can thus read or write any PCI I/O space supported by other modules.)

7.1.2.1.3.3 PCI Configuration Space

The PCI Configuration Space for the PM4700A contains configuration and control registers, accessible via the PCI Local Bus, for initialization and control in the configuration space. The PM4700A module defines and recognizes the following registers:

Address	Register	Access	Description
04	PCI Status/Command	Read/ Write	Standard Status and Command Fields
10	Base Address	Write Only	Standard Format
40	Module Status/Command	Read/ Write	PM4700A Status and Command Fields
44	Module Interrupt	Read/ Write	PM4700A External Interrupt Status

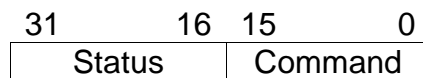
Writes to the registers are constrained as described in the succeeding sections. A write any other register in the PCI Configuration Space has no effect. Reading any other register returns a zero value. Burst access from the PCI Local Bus is not directly supported, and will be broken up into single accesses. Block access from the processor will set the Register Burst bit in the Module Status Register (see section 7.1.2.1.4.3).

7.1.2.1.4 PCI Registers

The PCI Local Bus Specification defines the layout of the PCI Configuration Space below address 0x40. The definition of registers at address 0x40 or above is device specific. This section describes a register as a source register if it can be read, and as a destination register if it can be modified by writing it.

7.1.2.1.4.1 PCI Status/Command Register (04)

The PCI Status/Command Register is a 32-bit source and destination, logically consisting of two 16-bit fields. The layout of the register is as follows:



Status Field

The most significant 16 bits of register 04 is the Status field. The following table describes the interpretation of the bits. If a bit is identified as "writable," the hardware sets it to 1 when the corresponding condition occurs; writing a 1 to the bit clears it to 0. The register is initialized as noted on PCI Reset.

Bit in Word	Bit in HW	Name	Writable	Binary Value(s)	Init
31	15	Detected Parity Error	Yes	0 (no), 1 (yes)	0
30	14	Signaled System Error	Yes	0 (no), 1 (yes)	0
29	13	Received Master Abort	Yes	0 (no), 1 (yes)	0
28	12	Received Target Abort	Yes	0 (no), 1 (yes)	0
27	11	Signaled Target Abort	No	0	
26-25	10-9	DEVSEL timing	No	10 (slow)	
24	8	Data Parity Detected	Yes	0 (no), 1 (yes)	0
23	7	Fast Back-to-Back Capable	No	1 (yes)	
22-16	6-0	<i>Reserved</i>	No	0	

Command Field

The least significant 16 bits of register 04 is the Command field. The following table describes the interpretation of the bits. Bits listed as writable are set to the value written. For a field listed as not writable, only a single value is listed, any value written is ignored, and the module does not support the options that are not selectable (e.g., memory write and invalidate). The register is initialized as noted on PCI Reset.

Bit in Word	Bit in HW	Name	Writable	Binary Value(s)	Init
15-10	15-10	<i>Reserved</i>	No	0	
9	9	Fast Back-to-Back Enable	No	0 (same agent only)	
8	8	SERR# enable	Yes	0 (disable) 1 (enable)	0
7	7	Wait cycle control	No	0 (no stepping)	
6	6	Parity error response	Yes	0 (ignore) 1 (respond)	0
5	5	VGA palette snoop	No	0 (no)	
4	4	Mem write and inv. Enable	No	0 (no)	
3	3	Special Cycles	No	0 (ignore)	
2	2	Bus Master	Yes	0 (not allowed) 1 (allowed)	0
1	1	Memory Space	Yes	0 (no access) 1 (access)	0
0	0	I/O Space	No	0 (no access)	

7.1.2.1.4.2 Base Address Register (10)

The Base Address Register defines the starting address of the region of PCI memory space that is mapped to the local memory of the PM4700A module. Only the most significant seven bits of the register (bits 31:25) are implemented, allowing the 32 MB local memory to be mapped to any 32 MB boundary in the 32-bit address space. Writing the base register sets the most significant seven bits from the data value; bits 24:0 are ignored and assumed to be zero. The low 4 bits characterize the memory, as defined in the PCI Local Bus standard. Zero values imply that the memory is in the memory space (bit 0), can be located anywhere in 32-bit address space (bits 2:1), and is marked "not prefetchable" (bit 3). The register is not readable. It is initialized to all zeroes on PCI Reset.

7.1.2.1.4.3 Module Status/Command Register (40)

The Module Status/Command Register is a 32-bit source and destination. The following table describes the significance of the bits in the register.

Bit in Word	Name	Binary Value(s)	Note	Control	Init	Reset
31-30	<i>Reserved</i>	0				
29	Invalid PCI access	0 (no), 1 (yes)	[1]	Clear	0	PCI
28	Processor Time Out	0 (no), 1 (yes)	[2]	Clear	0	PCI
27	<i>Reserved</i>	0				
26	Register Burst	0 (no), 1 (yes)	[3]	Clear	0	PCI
25-24	<i>Reserved</i>	0				
23	Processor Reset	0 (reset) 1 (not reset)	[4]	Full	0	PCI
22-16	<i>Reserved</i>	0				
15-9	<i>Reserved</i>	0				
8	Configuration Done	0 (no), 1 (yes)	[5]	None	0	PCI
7-1	<i>Reserved</i>	0				
0	Memory Lock	0 (unlocked) 1 (locked)	[6]	Full	0	Proc

The column labeled "Control" identifies how writing to the register controls the bit:

- "Full" indicates that the bit is assigned the value written to it.
- "Clear" indicates that the bit is set by the hardware and is reset when a 1 is written to it; writing a 0 has no effect.
- "None" indicates that the bit is controlled only by the hardware.

Bits in the register are initialized as noted in the "Init" column on the reset noted in the Reset column. Reserved bits are not implemented, and read as 0.

Notes:

- [1] Invalid PCI access is set if the processor issues a read or write with a PCI address and Bus Master is not set in the PCI Command Register. Causes a processor Bus Error or Non-Maskable Interrupt (NMI) exception.
- [2] Processor timeout is set if the processor waits an excessive time (about 4.1 msec) for any system bus access. Causes a processor Bus Error or Non-Maskable Interrupt (NMI) exception.
- [3] Register Burst is set if the processor attempts a burst (cacheable) read or write in PCI configuration space for any module. Causes a processor Bus Error or Non-Maskable Interrupt (NMI) exception.
- [4] The processor and module control logic is reset when the Processor Reset bit changes from 1 to 0. The hardware initializes the bit to 0, so it is necessary to write first a 1, then a 0 to it to cause the Processor Reset action. See section 7.1.2.4 "Initialization," below.

- [5] Configuration Done is set when the processing of the hardware initialization string is complete. See section 7.1.2.4 "Initialization," below. Writing this bit has no effect on the Module Status/Command Register, but defines timing constraints for the hardware initialization string; for the PM4700A, it should only be set to 0.
- [6] Memory Lock controls the processor's access to the PCI Local Bus. If Memory Lock is 1, memory accesses via the PCI Local Bus are atomic, using the PCI Lock mechanism.

7.1.2.1.4.4 Module Interrupt Register (44)

Each bit in the Module Interrupt Register (44) controls an interrupt signal. Reads and writes of the register control the state of the bit. Writing the bit sets it to zero, activating the interrupt request signal, regardless of the value written. Reading the bit sets it one, clearing the interrupt request. The byte enable bits determine which of the interrupt bits are altered by a register access, so that reading or writing the specific byte of an interrupt bit will leave the other interrupt bit unaffected. The significance of the bits is described in the following table. The bits are set as noted in the "Init" column on a Processor Reset.

Bit in Word	Name	Byte	Byte Enable	Init
31-25	<i>Reserved</i>	-	-	
24	Processor Interrupt 0	0x47	3	1
23-1	<i>Reserved</i>	-	-	
0	PCI Interrupt A	0x44	0	1

7.1.2.1.5 TDM Registers

The TDM interface is implemented with an IDT 72615 synchronous bidirectional FIFO and PM4700A logic to provide control and sequencing. The IDT 72615 provides 512 entries in each of two FIFOs (one input and one output), four registers to define "almost" full and empty offsets for each FIFO, and four signals for each FIFO (full, empty, almost-full, and almost-empty). The almost full condition is true when the number of entries not occupied by data is the offset or less. The offsets for the "almost" conditions can be set anywhere from 0 to the FIFO size. The almost-empty condition thus can be defined conceptually as "half empty" by setting the almost-empty offset to half the FIFO size, or as "not almost full" by setting the offset close to the FIFO size. The TDM Interrupt Condition described below relies on this flexibility. On Processor Reset, all four offset registers are initialized to the value 8, and any data in the FIFOs is lost.

The IDT 72615 registers and several PM4700A module registers for controlling the TDM interface and the Serial Communications Controller are addressable from the processor via System Interface Bus at physical addresses offset from 0x00000000 (high four bits '0000'). The IDT 72615 registers (the "FIFO

registers") are at offsets less than 0x80; the PM4700A registers (the "control registers") start at 0x80. Bits [7:2] of the physical address select the register during the address cycle; bit 2 is always on to ensure correct alignment on the system bus.

The behavior of the registers varies. The table below summarizes the registers defined and the effect of processor reads and writes on each. Reading or writing registers with addresses not listed, reading a register that is listed as "Write Only," or accesses of other than 32 bits will produce an undefined result. Except as noted, data and control values are unreliable after Processor Reset or PCI Reset.

Offset	Register Name	Bits	Access	Contents
0x04	TDM Data	15:0	Read/Write	Input/Output data
0x24	Output Almost-Empty Offset	8:0	Write only	Interrupt threshold
0x2C	Output Almost-Full Offset	8:0	Write only	Interrupt threshold
0x34	Input Almost-Empty Offset	8:0	Write only	Interrupt threshold
0x3C	Input Almost-Full Offset	8:0	Write only	Interrupt threshold
0x84	SCC Interrupt Enable	3:0	Write only	Condition Enables
0x8C	TDM Enable	0	Write only	TDM Local On/Off
0x94	TDM Status0/Configuration	7:0	Read/Write	Status and Control
0x9C	TDM Status1/SCC Control	7:0 1:0	Read Write	Status and Control
0xA4	SCC Data	7:0	Read/Write	RS232 Data
0xAC	RMSync Interrupt Clear	1	Write Only	Superframe ACK
0xB4	LED	1:0	Write Only	Indicator Control
0xBC	PCI ID	3:0	Write Only	Module Identifier

7.1.2.1.5.1 TDM Data Register

Writing to the TDM Data Register pushes the low 16 bits of the data written into the TDM Output queue. Reading from the register pops 16 bits from the TDM Input queue.

7.1.2.1.5.2 TDM FIFO Flag Offset Registers

The four 9-bit offset registers define the almost-empty and almost-full criteria for the FIFOs. Writing a value of N into the Output Almost-Empty Offset Register will cause the Output FIFO Almost Empty condition (in TDM Status Register 1) to be true when there are N or fewer items in the TDM Output FIFO. The four flags controlled by the offset registers allow applications to regulate the processing of TDM data to keep the TDM FIFOs fed without running into the hard full or empty conditions.

The TDM FIFO Flag Offset Registers cannot be read.

7.1.2.1.5.3 SCC Interrupt Enable Register

The SCC Interrupt Enable Register (0x84) is a 4-bit destination containing enable bits for conditions leading to the SCC and Miscellaneous processor interrupts. The listed interrupt occurs if any of the following conditions is true and the corresponding bit is on. The register must be explicitly initialized.

Bit #	Signal	Interrupt
3	CTS	Serial Interface (processor interrupt 3)
2	PCI Module Interrupt	Miscellaneous (processor interrupt 2)
1	RX Full	Serial Interface (processor interrupt 3)
0	TX Not Full	Serial Interface (processor interrupt 3)

7.1.2.1.5.4 TDM Enable Register

The TDM Enable Register (0x8C) is a 1-bit destination; bit 0 enables the TDM queues when it is 1. It is not a source register. It is initialized to 0 (queues disabled) on Processor Reset.

7.1.2.1.5.5 TDM Status0/Configuration Register

The TDM Status0/Configuration Register (0x94) is implemented as separate 8-bit source and destination registers. As the source TDM Status0 register, the bits have the following significance. The bits with values in the "Init" column are initialized as indicated on Processor Reset.

Bit	Content	Init
7	Input FIFO empty	1
6	Output FIFO full	0
5	Input FIFO almost empty	1
4	Output FIFO almost full	0
3	Superframe Sync Signal	-
2	Module Interrupt Signal	-
1	SCC Receive buffer full	-
0	SCC Transmit buff empty	-

As the destination TDM Configuration register, the bits have the following meaning. The register must be explicitly written to initialize it.

Bits	Content	Values	Note
7:6	TDM Superframe Sync	00: Normal (no Superframe) 01: Superframe Sync falling edge 10: Superframe Sync rising edge 11: Superframe Sync either edge	[1]
5:4	TDM Interrupt condition	00: Input FIFO not Almost Empty 01: Output FIFO not Almost Full 10: Input FIFO not Almost Empty OR Output FIFO not Almost Full 11: Interrupt disabled	[2]
3:2	Output Data Size	00: 8 bits 01: 32 bits 10: <i>Reserved</i> 11: 16 bits	
1:0	Input Data Size	00: 32 bits 01: 8 bits 10: 16 bits 11: <i>Reserved</i>	

Notes:

[1] The TDM Superframe Sync signal (also known as RMSync) is reflected in bit 3 of the TDM Status0 Register; when set, it causes the Miscellaneous Interrupt.

[2] The TDM Interrupt Condition definition inverts the Input FIFO Almost Empty and Output FIFO Almost Full conditions, which allows the resulting interrupt to be a "work to be done" signal for input, output, or both. The interrupt can be thought of as "Input FIFO no longer Almost Empty" or "Output FIFO no longer Almost Full." If the Input Almost Empty Offset is half the FIFO size, for example, the Input FIFO Almost Empty signal is true except when the FIFO is at least half full. When it becomes false, causing the interrupt, it can indicate that the Input FIFO contains enough data to be processed efficiently. Setting the offset closer to the FIFO size will cause the interrupt later, when it is more critical to respond; setting it closer to zero will cause the interrupt earlier, when the amount of processing to be done is less.

7.1.2.1.5.6 TDM Status1/SCC Control Register

The destination SCC Control Register (0x9C) is a 2-bit destination. The value of the two bits must be explicitly initialized by writing to the register. The bits cannot be written separately.

Bit	Content	Value = 0	Value = 1
1	Serial Interface Speed	9600 baud	19200 baud
0	RTS (request to send) control	RTS not asserted	RTS asserted

The source TDM Status1 contains several TDM and SCC status and control values. The bits with values in the "Init" column are initialized as indicated on Processor Reset.

Bits	Content	Init
7	TDM Input FIFO "almost full" status	0
6	the TDM Output "almost empty" status	1
5	the SCC RTS (request to send) status	-
4	the SCC CTS (clear to send) status	-
3:0	the Module Identifier	-

The Module Identifier contains the ID number for the module, generally assigned by the V6M6 software. The TDM logic processes time slots with a matching ID.

7.1.2.1.5.7 SCC Data Register

The SCC Data Register (0xA4) is an 8-bit source and destination. As a destination, it takes 8 bits of data to be sent out on the serial interface. As a source, it returns the 8-bit character read in. Input data is available if the SCC Receiver Buffer Full bit in the TDM Status0 Register is 1. Output data can be written if the SCC Transmit Buffer Empty bit in the TDM Status0 Register is 1.

7.1.2.1.5.8 Superframe Sync Interrupt Clear Register

The Superframe Sync Interrupt Clear Register (0xAC) is a destination register. Writing a 1 value to bit 1 clears the Superframe Sync Signal, reflected in the TDM Status Register, which causes the Miscellaneous interrupt (processor interrupt 2).

7.1.2.1.5.9 LED Register

The LED Register (0xB4) is a 2-bit destination register, controlling the three-color LED for the module:

- 00: off
- 01: green
- 10: red
- 11: amber

7.1.2.1.5.10 PCI ID Register

The PCI ID Register (0xBC) is a 4-bit destination register that accepts a 4-bit Module ID for the PM4700A module. The Module ID is reported in the SCC Control Register (#3 above). The Module ID is used in the TDM control words to identify the source and sink modules for each time slot.

7.1.2.2 Addressing

7.1.2.2.1 External addressing of Module Resources

This section summarizes resources in the PM4700a module that are externally addressable. This makes them available on the PCI Local Bus to other modules and, via the VME bridge on the baseboard, to the host.

7.1.2.2.1.1 Local Memory

The PM4700A local memory is mapped into the PCI memory space at the address specified by the Module Base Address Register (section 7.1.2.1.4.2). The initializing agent sets the Module Base Address Register from the PCI Local Bus with a Configuration Write command. Thereafter, Memory Read, Memory Read Multiple, Memory Read Line, and Memory Write commands on the PCI Local Bus will access the memory if the high seven bits of the address on the PCI Local bus match the high seven bits of the Module Base Address Register. The 25 bits of offset from the base register cover the 32 MB address range of the local memory.

Memory Write And Invalidate and Dual Address Cycle (64-bit address) commands are not supported.

7.1.2.2.1.2 R4700 Processor Registers

The PM4700A module does not support external writes to the R4700 processor registers [PHUM 12-23 ff]. Outside of initialization, no mechanism external to the processor has access to the processor registers.

7.1.2.2.1.3 Module Registers

PCI Configuration Space registers are addressable as outlined in section 7.1.2.1.3.3. They are accessible with PCI Local Bus Configuration Read and Configuration Write commands.

7.1.2.2.1.4 TDM Registers

The TDM Registers described in section 7.1.2.1.5 are not addressable from the PCI bus.

7.1.2.2.2 Internal addressing of Module Resources

This section summarizes the addressability of the module resources from the R4700 processor. All program addresses in the processor are normally virtual, and are mapped to physical addresses, which are sent out on the module system

bus, by the R4700 hardware. The virtual-to-physical address mapping depends on the operating software; this section refers only to physical addresses.

7.1.2.2.2.1 Local Memory

The 32-MB local memory is accessible starting at address 0x10000000, extending to 0x11FFFFFF.

7.1.2.2.2.2 Module Registers

There is no direct (on-module) path between the R4700 processor and the Module Registers. Instead, the processor can access the Module Registers in the module's Configuration Space with the Configuration Read and Write commands on the PCI Local Bus, using the appropriate PCI Configuration Space base address for the module plus the register offset.

7.1.2.2.2.3 TDM Registers

The R4700 processor has direct access to the TDM Registers at the physical addresses noted in section 7.1.2.1.5.

7.1.2.2.3 Internal addressing of External Resources

The R4700 processor can access PCI Configuration, I/O, and Memory Spaces for other modules. The 4 most significant bits of the 32-bit physical address identify the type of access. The PM4700a module logic generates the correct PCI Local Bus command for each address range:

Addr[31:28]	Space	PCI Commands Generated
0010	Configuration	Configuration Read/Write
0011	I/O	I/O Read/Write
0100-1111	Memory	Memory Read/Write, Read Line, Read Multiple

Access to the specific modules and to the V6M6 board resources depends on the memory map created by software.

The PCI Memory Space addresses (starting at 0x40000000) are passed to the PCI Local Bus without modification, restricting the PCI addresses that can be accessed from the PM4700A. The assignment of PCI Memory addresses must guarantee that no resources to be accessed from the PM4700A are assigned addresses below that threshold.

7.1.2.2.4 Address Map Summary

7.1.2.2.4.1 System Bus Address Map

The system bus address space includes the module resources and resources addressable on the PCI Local Bus. The table below shows the mapping of these resources into the system bus address space. Addresses within this space are generated by the processor or the PCI Local Bus interface logic, and are needed for the physical page number specifications in the EntryLo Registers when creating TLB entries. Non-kernel code on the processor normally uses only the virtual address space defined by the architecture, with the kernel managing the page tables and TLB entries to support it [see PHUM 4]. Much of the physical address space is useful only for kernel and initialization code.

Space	Address Start	Length	Note
Local TDM Registers	0x00000000	256 bytes	not contiguous
Local RAM	0x10000000	32 MB	
PCI Configuration Space	0x20000000		external modules
PCI I/O Space	0x30000000		external modules
PCI Memory space	0x40000000		external modules

7.1.2.2.4.2 PCI Local Bus Address Map

The PM4700A supports access to each of the PCI Local Bus address spaces that is supported by a PCI module or the V6M6 baseboard. The Memory Space and I/O Space base addresses for each module are determined by the base registers for the module, which are in the module's configuration space. The length of each space is implicit, defined by the hardware. System initialization software is responsible for setting the module base registers for all active PCI modules. No region of either address space should be recognized by more than one module.

Baseboard resources, particularly the global RAM, are similarly addressable.

Configuration space mappings are fixed, according to the table below. The configuration space length is no more than 256 bytes.

PCI Configuration Space Base Addresses	
PCI Module	Configuration Space Address
A	0x01000000
B	0x00800000
C	0x00400000
D	0x00200000
E	0x00100000
F	0x00080000

7.1.2.3 Interfaces

7.1.2.3.1 PCI Local Bus

7.1.2.3.1.1 Interrupts

The V6M6 architecture supports two module-generated PCI interrupts, which are named Interrupt A (INTA/) and Interrupt B (INTB/). An external interrupt named Module Interrupt (MODINT/) to each module on the baseboard allows one module to interrupt another; the V6M6 PCI interrupt logic flexibly maps the modules' INTA/ and INTB/ signals to the modules' MODINT/ signals. See section 3.4.4 for details of which V6M6 revision levels support this function and how.

Interrupt A

The PM4700A module controls PCI Interrupt A through bit 0 of the module's Interrupt Register (address 0x44 in PCI configuration space). See section 7.1.2.1.4.4.

Interrupt B

The PM4700A module does not control PCI Interrupt B.

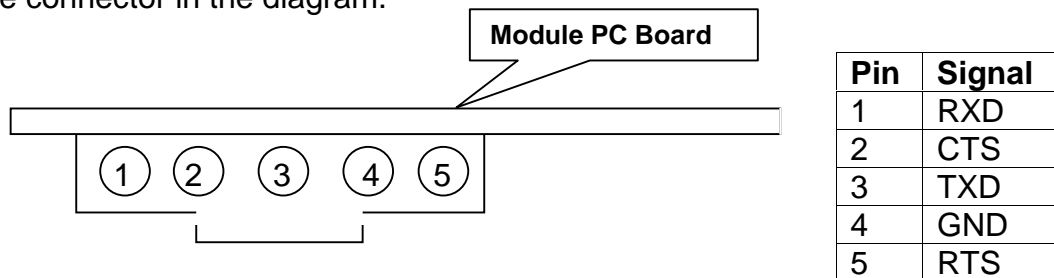
Module Interrupt

The Module Interrupt signal causes a processor Miscellaneous Interrupt when the corresponding bit in the SCC Interrupt Enable register is set. See section 7.1.2.1.5.3 for bit encodings.

7.1.2.3.2 RS-232

The optional RS-232 connector is a Molex 705-53-9405. The mating connector is a Molex 50-57-9405, with Molex 16-02-xxxx terminals (xxxx depends on the wire size and terminal plating).

When installed on the V6M6 baseboard in module site A or C, the connector emerges through the hole in the faceplate installed for the module on the front panel of the baseboard. The diagram below shows the layout of the pins in the optional RS-232 connector, looking at the front panel with the module's exposed side up (connector below the module PC board, latch below the connector). Pin 1 is next to the corner of the module. The V6M6 baseboard would appear below the connector in the diagram.



7.1.2.4 Initialization

The R4700 processor can be reset by toggling the Processor Reset bit in PCI Module Status Register (Configuration Register 0x40) to 1 and then 0. This raises and lowers the **Reset*** signal, causing a Soft Reset Exception. (The SR bit of the processor Status register is set for this exception.)

The Reset exception (caused by toggling the **ColdReset*** signal) clears the SR bit to 0.

7.1.2.5 Memory Access

7.1.2.5.1 Optimized Sequential Access

The PM4700A module supports optimized sequential memory reads and writes in several ways. The PCI Local Bus and R4700 System Interface Bus differ somewhat in structure:

- The PCI Local Bus Standard defines burst access for sequential Memory and I/O Space accesses, where a single address cycle is followed by arbitrarily many data cycles. The initiating and target agents continue 4-byte data transfers until the initiator drops the FRAME# signal or the target aborts further transfers with the STOP# signal.
- The R4700 processor issues a "block read" or "block write" when loading or storing a data cache line (32 bytes). A single address cycle on the system interface bus is followed by a sequence of 8-byte data cycles, with data identifiers on the System Command bus specifying "good/erroneous data" and "last/not last element" for each data cycle.

The local memory on the PM4700A module is designed to perform optimally with the R4700 System Interface Bus, so it also differs in structure from the PCI Local Bus. The PM4700A logic reconciles these differences as follows:

- When the processor issues a block read or block write to a PCI address, the module logic will process it as a burst on the PCI Local Bus.
- When the processor reads or writes an item of from 5 to 8 bytes on the PCI Local Bus, the PM4700a logic performs it as a PCI burst with two 32-bit data cycles, providing the appropriate PCI controls. If the size is less than 8 bytes, some byte lanes are not enabled on one or both data cycles.
- When an external agent issues a read or write to the PM4700A local memory, the module logic, as required, supports data cycles continuing until the initiator terminates the burst.

7.1.2.5.2 Memory interface Performance

The table below shows the number of system interface bus cycles for various local memory operations by the processor. The bus runs at a quarter of the processor clock speed (50 MHz for the 200 MHz processor).

Operation	Bytes	Cycles
Single Read	1-8	5
Single Write	1-8	6
Block Read (cache)	32	8
Block Write (cache)	32	8

Accesses over the PCI bus are generally slower, due to the narrower data path, and less predictable, since there may be contention for the bus.

7.1.3 PM4700A Software Support

The software support for the PM4700A has not yet been documented. This section is currently provided only in outline form.

7.1.3.1 V6M6 Host Application Library for PM4700A

7.1.3.2 PM4700A Application Development

7.1.3.2.1 PM4700A Software Development Environment

7.1.3.2.2 PM4700A Application Library

7.1.3.3 Utility Programs

7.1.3.4 Diagnostic Programs

7	MIPS PROCESSOR MODULES	1
7.1	PM4700A Module	1
7.1.1	PM4700A Quick Tour	1
	Figure 7.1-1: PM4700A Module Block Diagram	1
7.1.2	PM4700A Hardware Details	3
7.1.2.1	Module Resources	3
7.1.2.1.1	Processor	3
7.1.2.1.2	Devices	6
7.1.2.1.3	Memory	6
7.1.2.1.4	PCI Registers	8
7.1.2.1.5	TDM Registers	11
7.1.2.2	Addressing	16
7.1.2.2.1	External addressing of Module Resources	16
7.1.2.2.2	Internal addressing of Module Resources	16
7.1.2.2.3	Internal addressing of External Resources	17
7.1.2.2.4	Address Map Summary	18
7.1.2.3	Interfaces	19
7.1.2.3.1	PCI Local Bus	19
7.1.2.3.2	RS-232	19
7.1.2.4	Initialization	20
7.1.2.5	Memory Access	20
7.1.2.5.1	Optimized Sequential Access	20
7.1.2.5.2	Memory interface Performance	21
7.1.3	PM4700A Software Support	21
7.1.3.1	V6M6 Host Application Library for PM4700A	21
7.1.3.2	PM4700A Application Development	21
7.1.3.2.1	PM4700A Software Development Environment	21
7.1.3.2.2	PM4700A Application Library	21
7.1.3.3	Utility Programs	21
7.1.3.4	Diagnostic Programs	21