Features

- **pci_mt Function Implementing a 32-bit Component Interconnect (PCI) Interface Optimized for AT40K/AT40KAL/AT94K Architecture**
- **Extensive Simulation Testing, Includes Test Vectors**
- **Uses 585 Core Cells**
- **Fully-compliant Design Including:**
  - **32-bit, 33 MHz Operation**
  - **Simplified Local Side (DMA Control Engine) Interface**
- **PCI Master Features:**
  - **Memory Read/Write, One-cycle and Burst**
  - **I/O Read/Write, One-cycle and Burst**
  - **Fully Integrated DMA Engine Including Address Counter Register, Byte Counter Register, Control and Status Register, Interrupt Status Register**
  - **Zero Wait State PCI Read/Write**
- **PCI Target Features:**
  - **Type Zero Configuration Space**
  - **Parity Error Detection**
  - **Memory Read/Write (1M byte) and I/O Read/Write (512K byte), One-cycle and Burst**
  - **Configuration Registers: Device ID, Vendor ID, Status, Command, Class Code, Revision ID, Header Type, Latency Timer, One Memory Base Address, One I/O Base Address, Subsystem ID, Subsystem Vendor ID, Maximum Latency, Minimum Grant, Interrupt Pin, and Interrupt Line**

Description

The pci_mt function provides a solution for integrating 32-bit PCI peripheral devices, and is fully tested to meet the requirements of the PCI specification. It is optimized for the AT40K FPGA architecture, enabling the designers to focus efforts on the custom logic surrounding the PCI interface. The pci_mt macro is intended for use in Atmel's AT40K/AT40KAL/AT94K devices with remaining logic resources available for user-defined local side (DMA control engine) customization.

Compliance Summary

- The pci_mt function is compliant with the requirements specified in the PCI Special Interest Group's (SIG) PCI Local Bus Specification, Rev.2.1.

PCI Bus Signals

The following PCI bus signals are used by the pci_mt function:

- *Input* – Standard input-only signal
- *Output* – Standard output-only signal
- *Bidirectional* – Tri-state input/output signal
- *Sustained tri-state* – Signal that is driven by one agent at a time. An agent that drives a sustained tri-state pin low must actively drive it high for one clock cycle before tri-stating it. Another agent cannot drive a sustained tri-state signal any sooner than one clock cycle after it is released by the previous agent.
- *Open-drain* – Signal that is wire-OR with other agents. The signaling agent asserts the open-drain signal, and a weak pull-up resistor deasserts the open-drain signal. The pull-up resistor may take two or three PCI bus clock cycles to restore the open-drain signal to its inactive state.

PCI Master/Target Core Function with DMA

AT40K-PCI IP Core

Rev. 1083B–06/01

Table 1 summarizes the PCI bus signals interfacing the pci_mt to the PCI bus.

Table 1. PCI Signals Interfacing the pci_mt to the PCI bus

Type	Name	Polarity	Description
In	clk	–	Clock, provides the reference signal for all other PCI interface signals, except *reset* and *intan.*
In	reset	Low	Reset, initializes the AT40K/20 interface circuitry, and can be asserted asynchronously to the PCI bus clock edge. When active, the PCI output signals are tri-stated and the open-drain signals, such as *serrn*, high impedance.
In	gntn	Low	Grant, indicates to the master device that it has control of the PCI bus. Every master device has a pair of arbitration lines (*gntn* and *reqn*) that connect directly to the arbiter.
Out	reqn	Low	Request, indicates to the arbiter that the master wants to gain control of the PCI bus to perform a transaction.
Tri-state	ad[31:0]	–	A time-multiplexed address/data bus; each bus transaction consists of an address phase followed by one or more data phases. The data phases occur when *irdyn* and *trdyn* are both asserted.
Tri-state	cben[3:0]	Low	A time-multiplexed command/byte enable bus. During the address phase it indicates command; during the data phase it indicates byte enables.
Tri-state	par	–	Parity, shows even parity. The number of 1s on *ad[31:0]*, *cben[3:0]*, and *par* is an even number.
Sustained Tri-state Master: Out Target: In	framen	Low	Frame, as output, indicates the beginning and duration for the current bus operation. When framen is initially asserted, the address and command signals are present on the *ad[31:0]* and *cben[3:0]* buses. It remains asserted during the data operation and is deasserted to identify the end of a transaction.
Sustained Tri-state Master: Out Target: In	irdyn	Low	Initiator ready, is an output for the bus master to its target and indicates that the bus master can complete a data transaction. In a write transaction, it indicates that the valid data is on the *ad[31:0]* bus. In a read transaction, it indicates that the master is ready to accept the data on the *ad[31:0]* bus.
Sustained Tri-state Master: In Target: Out	devseln	Low	Device select. Target asserts *devseln* to indicate that it has decoded its own address.
Sustained Tri-state Master: In Target: Out	trdyn	Low	Target ready. As target output, indicates that the target can complete the current data transaction. In a read operation, it indicates that the target is providing data on the *ad[31:0]* bus. In a write operation, it indicates that the target is ready to accept data on the *ad[31:0]* bus.
Sustained Tri-state Master: In Target: Out	stopn	Low	Stop. It signals a target device request that indicates to the bus master to stop the current transaction.
In	idsel	High	Initialization device select, is a chip select for configuration read or write operations.
Sustained Tri-state	perrn	Low	Parity error, indicates a data parity error.
Open-drain	serrn	Low	System error, indicates system error and address parity error.
Open-drain	intan	Low	Interrupt A, an interrupt to the host, and must be used for any single-function device requiring an interrupt capability.

Table 2 summarizes the PCI bus signals interfacing the pci_mt to the local side peripheral device.

Table 2. PCI Signals Interfacing the pci_mt to the Local Side

Type	Name	Polarity	Description
In	irqn	Low	Local side interrupt request. The local side peripheral device asserts it to signal a PCI bus interrupt. For example, when the local side peripheral device requires a DMA transfer, it could use the *irqn* input to request servicing from the host.
In	holdn	Low	Local hold, when asserted, it suspends the current DMA transfer. As long as *holdn* is active, data transfers cannot occur between the pci_mt and the local side peripheral device.
In	req	High	Local DMA request. The local side peripheral device asserts *req*, which signals the pci_mt to request permission for a PCI DMA operation.
DMA WRITE: In DMA READ: Out	ldat[31:0]	–	A local data bus input, driven active by the local side peripheral device during pci_mt initiated DMA write transactions (i.e., local side DMA read transactions) and PCI bus target read transactions. A local data bus output, driven by the pci_mt during pci_mt initiated DMA read transactions (i.e., local side DMA write transactions) and PCI target write transactions.
Out	pciholdn	Low	Local target chip select. When active, it notifies the peripheral device of an impending target transaction. Any PCI master device can read or write to a local side peripheral device through the pci_mt. The *ackn* and the *pciholdn* are never asserted at the same time.
Out	rdn	Low	Read. The pci_mt asserts *rdn* to signal a read access to the local side peripheral device. The pci_mt uses *rdn* for reading from peripheral device target memory. For DMA read operations, the pci_mt asserts the *ackn* and *rdn* signals.
Out	wrn	Low	Write. The pci_mt asserts *wrn* to signal a write access to the local side peripheral device. The pci_mt uses *wrn* for writing to peripheral device target memory. For DMA write operations, the pci_mt asserts the *ackn* and *wrn* signals.
Out	ackn	Low	Local DMA acknowledge. When low, it notifies the local side peripheral device that it has been granted a DMA read or write transaction. The peripheral device can then transfer data to or from the PCI bus through the pci_mt.
Out	lreset	Low	Local reset. The pci_mt asserts it to reset the local side peripheral device. It follows the state of the LRST bit (bit 0 of the DMA control status register).

Parameters

The pci_mt parameters set read-only PCI bus configuration registers in the pci_mt; these registers are called device identification registers. See “Configuration Registers” on page 7 for more information on device ID registers. Table 3 describes the parameters of the pci_mt function.

Table 3. Parameters

Name	Range	Default Value (Hexadecimal)	Description
CLASS_CODE	24-bit	FF0000	Class code register
DEVICE_ID	16-bit	0001	Device ID register
DEVICE_VEND_ID	16-bit	B325	Device Vendor ID register
REVISION_ID	8-bit	01	Revision ID Register
SUBSYSTEM_ID	16-bit	0000	Subsystem ID register
SUBSYSTEM_VEND_ID	16-bit	0000	Subsystem Vendor ID register

Functional Description

The pci_mt macro consists of three main components:

- A defined 64-byte PCI bus configuration register space and master control logic.
- PCI bus target interface control logic, including target decode and local memory read/write signals.
- Embedded DMA control engine, and local side interface DMA control logic, including read/write control and PCI bus arbitration for master/target accesses.

Sustained Tri-state Signal Operation

The PCI specification defines signals that are constantly sampled by different bus agents yet driven by one agent at a time, as sustained tri-state signals. For example, *framen* is constantly sampled by different PCI bus targets (to detect the start of a transaction), and yet driven by one PCI bus master at a time.

For sustained tri-state signals, the PCI specification requires using one clock cycle to drive the signals inactive before being tri-stated. The PCI specification also requires that any sustained tri-state signal being released, such as the master device releasing *ad[31:0]* after asserting the address on a read operation, be given a full clock cycle to tri-state before another device can drive it.

The PCI specification defines a turn-around cycle as the clock cycle where a sustained tri-state signal is being tri-stated so that another bus agent can drive it. Turn-around cycles prevent contention on the bus.

Master Device Signals and Signal Assertion

Figure 1. pci_mt Master Device Signals

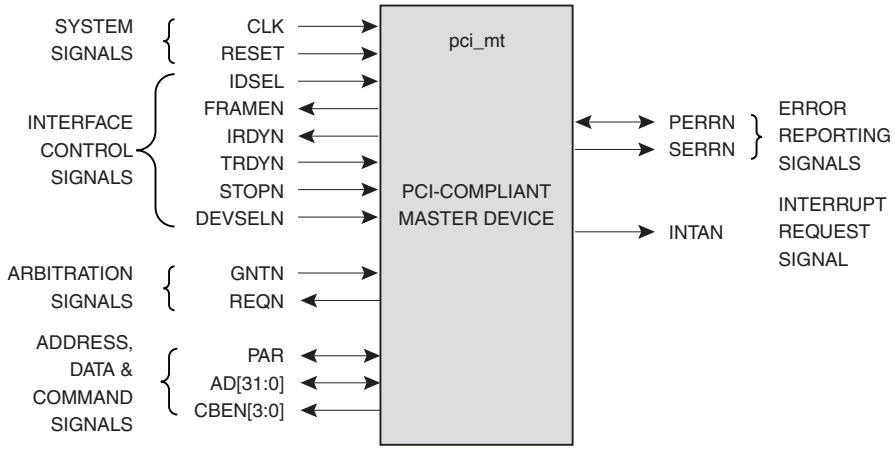


Figure 1 illustrates the PCI-compliant master device signals interfacing the pci_mt with the PCI bus. The signals are grouped by functionality, and signal directions are illustrated from the perspective of the pci_mt operating as a master on the PCI bus.

A pci_mt master sequence begins with the assertion of *reqn* to request the PCI bus. After receiving *gntn* from the arbiter (usually the PCI host bridge) and after the bus idle state is detected, the pci_mt initiates the address phase by asserting *framen* and driving both the PCI address on *ad[31:0]* and the bus command on *cben[3:0]* for one clock cycle.

When the pci_mt is ready to present data on the bus, it asserts *irdyn*. At this point, the pci_mt master logic monitors the control signals driven by the target device. (A target device is determined by the decoding of the address and command signals present on the PCI bus during the address phase of the transaction). The target device drives the control signals *devseln*, *trdyn*, and *stopn* to indicate one of the following:

- The data transaction has been decoded and accepted.
- The target device is ready for the data operation. (When both *trdyn* and *irdyn* are active, a data word is clocked from the sending to the receiving device.)
- The master device should stop the current transaction.

Target Device Signals and Signal Assertion

Figure 2. pci_mt Target Device Signals

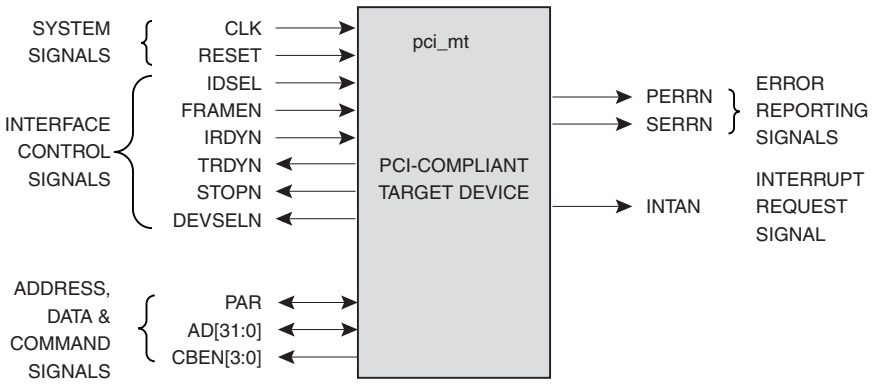


Figure 2 illustrates the PCI-compliant target device signals interfacing the pci_mt with the PCI bus. The signals are grouped by functionality, and signal directions are illustrated from the perspective of the pci_mt operating as a target on the PCI bus.

A pci_mt target sequence begins when the master device asserts *framen* and drives the address of the target and the command on the PCI bus. When the target device decodes its address on the PCI bus, it asserts *devseln* to indicate to the master that it has accepted the transaction. The master will then assert *irdyn* to indicate to the target device that:

- For a read operation, the master device can complete a data transaction.
- For a write operation, valid data is on the *ad[31:0]* bus.

When the pci_mt functions as the selected target device, it will drive the control signals *devseln*, *trdyn*, and *stopn* as discussed in “Master Signals & Signal Assertion”.

As a target device, pci_mt supports both single-cycle and burst-cycle accesses; it drives *trdyn* active in both cases, *stopn* is driven active only in the case of a single-cycle access. When qualified by an active *irdyn* signal, a data word is clocked from the sending to the receiving device.

Parity Signal Operation

All bus cycles include parity. Every device that transmits on the *ad[31:0]* bus must also drive the *par* signal, including master devices outputting the address. Because parity on the PCI bus is even, the number of logic 1s on *ad[31:0]*, *cben[3:0]*, and *par* must be even. Parity checking is not required, but can be enabled through the agent’s PCI command register. System, address and data parity errors are presented on the *serrn* output, address and data parity errors are presented on the *perrn* output. The par bit lags the *ad[31:0]* bus by one clock cycle, and parity error signal lag the *par* bit by one clock cycle; thus, parity error signals lag the address or data by two clock cycles.

Bus Master Commands

When the pci_mt acquires mastership of the PCI bus, it can initiate a memory read or memory write command. During the address phase of a transaction, the *cben[3:0]* bus is used to indicate the transaction type.

The pci_mt supports IO read/write, memory read/write, and configuration read/write commands. When operating as a master device, the pci_mt executes standard IO or memory read/write operations. When operating as a target, the pci_mt responds to standard IO or memory read/write transactions. The pci_mt also responds to configuration read/write operations.

Table 4 summarizes the PCI bus commands that are supported by the pci_mt.

Table 4. PCI Bus Command Support Summary

cben[3:0] Value	Bus Command Cycle	Target Support	Master Support
0010	IO Read	Yes	Yes
0011	IO Write	Yes	Yes
1010	Memory Read	Yes	Yes
1011	Memory Write	Yes	Yes
1010	Configuration Read	Yes	No
1011	Configuration Write	Yes	No
Others	–	No	No

Configuration Registers

Each logical PCI bus device includes a block of 64 configuration DWORDS reserved for the implementation of its configuration registers. The format of the first 16 DWORDS is defined by *the PCI SIG's PCI Local Bus Specification compliance checklist, revision 2.1*, which defines two header formats, type one and type zero. Header type one is used for PCI-to-PCI bridges; header type zero is used for all other devices, including the pci_mt.

Table 5 displays the defined 64-byte configuration space. The registers within this range are used to identify the device, control the PCI bus functions, and provide PCI bus status. The areas in italics indicate registers that are supported by the pci_mt.

Table 5. PCI Bus Configuration Registers

Address	Byte 3	Byte 2	Byte 1	Byte 0
00h	*Device ID*		*Vendor ID*	
04h	*Status Register*		*Command Register*	
08h	*Class Code*			*Revision ID*
0Ch	BIST	*Header Type*	*Latency Timer*	Cache Line Size
10h	*Base Address Register 0 (Memory)*			
14h	*Base Address Register 1 (IO)*			
18h	Base Address Register 2			
1Ch	Base Address Register 3			
20h	Base Address Register 4			
24h	Base Address Register 5			
28h	Card Bus CIS Pointer			
2Ch	*Subsystem ID*		*Subsystem Vendor ID*	
30h	Expansion ROM Base Address Register			
34h	Reserved			
38h	Reserved			
3Ch	*Maximum Latency*	*Minimum Grant*	*Interrupt Pin*	*Interrupt Line*

Table 6 summarizes the pci_mt-supported configuration registers address map. Read/write refers to the status at run time, i.e., from the perspective of other PCI bus agents. The specified default state is defined as the state of the register when the PCI bus is reset.

Table 6. pci_mt-Supported Configuration Registers Address Map

Address Offset	Range Reserved	Bytes Used/ Reserved	Read/Write	Mnemonic	Register Name
00	00 – 01	2/2	Read	ven_id	Vendor ID
02	02 – 03	2/2	Read	dev_id	Device ID
04	04 – 05	2/2	Read/Write	comd	Command
06	06 – 07	2/2	Read/Write	status	Status
08	08 – 08	1/1	Read	rev_id	Revision ID
09	09 – 0B	3/3	Read	class	Class Code
0D	0D – 0D	1/1	Read/Write	lat_tmr	Latency Timer
0E	0E – 0E	1/1	Read	header	Header Type
10	10 – 13	4/4	Read/Write	bar0	Base Address Register 0
14	14 – 17	4/4	Read/Write	bar1	Base Address Register 1
2C	2C – 2D	2/2	Read	sub_ven_id	Subsystem Vendor ID
2E	2E – 2F	2/2	Read	sub_id	Subsystem ID
3C	3C – 3C	1/1	Read/Write	int_ln	Interrupt Line
3D	3D – 3D	1/1	Read	int_pin	Interrupt Pin
3E	3E – 3E	1/1	Read	min_gnt	Minimum Grant
3F	3F – 3F	1/1	Read	max_lat	Maximum Latency

Vendor ID Register (Offset = 00 Hex)

Vendor ID is a 16-bit read-only register that identifies the manufacturer of the device (e.g., Atmel for the pci_mt). The value of this register is assigned by the PCI SIG; the default value of this register is the Atmel Vendor ID value, which is b325 hex. However, by setting the Device_Vend_ID parameter (see Table 3), designers can change the value of the Vendor ID register to their PCI SIG-assigned vendor ID value. (See Table 7.)

Table 7. Vendor ID Register Format

Data Bit	Mnemonic	Read/Write	Definition
[15:0]	ven_id	Read	PCI Vendor ID

Device ID Register (Offset = 02 Hex)

Device ID is a 16-bit read-only register that identifies the type of device. The value of this register is assigned by the manufacturer (e.g., Atmel assigned the value of the Device ID register for the pci_mt). The default value of this register is 0001 hex; however, by setting the Device_ID parameter (see Table 3), designers can change the value of the Device ID register.

Command Register (Offset = 04 Hex)

Command is a 16-bit read and write register that provides basic control over the ability of the pci_mt to respond to and/or perform PCI bus accesses. (See Table 8.)

Table 8. Command Register Format

Data Bit	Mnemonic	Read/Write	Definition
0	Unused	–	–
1	mem_ena	Read/Write	Memory access enable. When high, it enables the pci_mt to respond to the PCI bus memory accesses as a target. Because the DMA registers are set via memory target accesses, the *mem_ena* bit must be set as part of the initialization operation for the pci_mt to perform DMA transfers.
2	mstr_ena	Read/Write	Master enable. When high, it enables the pci_mt to acquire mastership of the PCI bus. For the pci_mt to perform DMA transfers, the *mstr_ena* bit must be set as part of the initialization operation.
[5:3]	Unused	–	–
6	perr_ena	Read/Write	Parity error enable. When high, it enables the pci_mt to report parity errors via the *perrn* output.
7	Unused	–	–
8	serr_ena	Read/Write	System error enable. When high, it enables the pci_mt to report system errors via the *serrn* output.
[15:9]	Unused	–	–

Status Register (Offset = 06 Hex)

Status is a 16-bit read and write register that provides the status of bus-related events. Read transactions to the Status Register behave normally. However, write transactions are different from typical write transactions in that bits in the Status Register can be cleared but not set. A bit in the Status Register is cleared by writing a logic one to that bit. For example, writing the value of 4000 hex to the Status Register clears bit number 14 and leaves the rest of the bits unchanged. The default value of the Status Register is 0000 hex. (See Table 9.)

Table 9. Status Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	Unused	–	–
8	dat_par_rep	Read/Write	Data parity error reported. When high, it indicates that during a read transaction the pci_mt asserted the *perrn* output as a master device, or that during a write transaction the *perrn* was asserted by a target device. This bit is high only when the *perr_ena* bit (bit 6 of the Command Register) is also high.
[10:9]	devsel_tim	Read/Write	Device select timing. It indicates target access timing of the pci_mt via the *devseln* output. The pci_mt is designed to be a slow target device.
11	Unused	–	–
12	tar_abrt	Read/Write	Target abort. When high, it indicates that the current target device transaction has been terminated.
13	mstr_abrt	Read/Write	Master abort. When high, it indicates that the pci_mt drove the *serrn* output active, i.e., a system error has occurred.
14	serr_set	Read/Write	System error enable. When high, it enables the pci_mt to report system errors via the *serrn* output.
15	det_par_err	Read/Write	Detected parity error. When high, it indicates that the pci_mt detected either an address or data parity error. Even if parity error reporting is disabled (via *perr_ena*), the pci_mt will set this bit.

Revision ID Register (Offset = 08 Hex)

Revision ID is an 8-bit read-only register that identifies the revision number of the device. The value of this register is assigned by the manufacturer (e.g., Atmel for the pci_mt). Therefore, the default value of this register is set as the revision number of the pci_mt, (see Table 10). However, designers can change the value of the Revision ID Register by setting the *REVISION_ID* parameter (see Table 3).

Table 10. Revision ID Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	rev_id	Read	PCI Revision ID

Class Code Register (Offset = 09 Hex)

Class Code is a 24-bit read-only register divided into three sub-registers: base class, sub-class, and programming interface. Refer to the PCI Local Bus Specification, rev.2.1 for detailed bit information. (See Table 11.) The default value of this register is FF0000 hex; however, designers can change the value by setting the *CLASS_CODE* parameter (see Table 3).

Table 11. Class Code Register Format

Data Bit	Mnemonic	Read/Write	Definition
[23:0]	class	Read	Class Code

Latency Timer Register (Offset = 0D Hex)

The Latency Timer Register is an 8-bit register with bits 2, 1, and 0 tied to GND. The register defines the maximum amount of time, in PCI bus clock cycles, that the pci_mt can retain ownership of the PCI bus. After initiating a transaction, the pci_mt decrements its latency timer by one on the rising edge of each clock. The default value of the latency timer register is 00 hex. (See Table 12.)

Table 12. Latency Timer Register Format

Data Bit	Mnemonic	Read/Write	Definition
[2:0]	lat_tmr	Read	Latency Timer Register
[7:3]	lat_tmr	Read/Write	Latency Timer Register

Header Type Register (Offset = 0E Hex)

Header Type is an 8-bit read-only register that identifies the pci_mt as a single-function device. The default value of this register is 00 hex. (See Table 13.)

Table 13. Header Type Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	header	Read	PCI Header Type

Base Address Register Zero (Offset = 10 Hex)

Base Address Register Zero (BAR0) consists of a 12-bit register (bits 31 through 20) that determines the base memory address of the pci_mt target space. Its default value is 000 hex. (See Table 14.)

Table 14. Base Address Register Zero Format

Data Bit	Mnemonic	Read/Write	Definition
0	mem_ind	Read	Memory indicator. Hardwired to a zero, it indicates a memory address decoder.
[2:1]	mem_type	Read	Memory type. It indicates the type of memory that can be implemented in the pci_mt memory address space. These bits are tied to GND, which indicates that the memory block can be located anywhere in the 32-bit address space.
3	pre_fetch	Read	Memory prefetchable. It indicates whether the block of memory defined by BAR 0 is prefetchable by the host bridge. In the pci_mt, the address space is not prefetchable, i.e., it reads as low.
[19:4]	Unused	–	–
[31:20]	bar0	Read/Write	Base Address Register Zero.

Base Address Register One (Offset = 14 Hex)

Table 15. Base Address Register One Format

Data Bit	Mnemonic	Read/Write	Definition
0	io_ind	Read	I/O space indicator. It indicates whether the register is I/O or a memory address decoder. In the pci_mt, the *io_ind* bit is tied to 1, which indicates an I/O address decoder.
1	io_res	Read	Reserved, returns zero.
[19:2]	Unused	–	–
[31:20]	bar1	Read/Write	Base Address Register One.

Base Address Register One (BAR1) consists of a 12-bit register (bits 31 through 20) that determines the base I/O address of the pci_mt target space. Its default value is 000 hex (see Table 15).

Subsystem Vendor ID Register (Offset = 2C Hex)

Subsystem Vendor ID Register is a 16-bit read-only register that identifies add-in cards designed by different vendors but with the same functional device on the card. The value of this register is assigned by PCI SIG (see Table 16). Its default value is 0000 hex; however, designers can change the value by setting the SUBSYSTEM_VEND_ID parameter, (see Table 3).

Table 16. Subsystem Vendor ID Register Format

Data Bit	Mnemonic	Read/Write	Definition
[15:0]	sub_vend_id	Read	PCI Subsystem/ Vendor ID

Subsystem ID Register (Offset = 2E Hex)

Subsystem ID Register is a 16-bit read-only register that identifies the subsystem; the value of this register is defined by the subsystem vendor, i.e., the designer (see Table 17). Its default value is 0000 hex; however, designers can change the value by setting the SUBSYSTEM _ID parameter (see Table 3).

Table 17. Subsystem ID Register Format

Data Bit	Mnemonic	Read/Write	Definition
[15:0]	sub_id	Read	PCI Subsystem ID

Interrupt Line Register (Offset = 3C Hex)

The Interrupt Line Register is an 8-bit read/write register that defines to which system interrupt request line (on the system interrupt controller) the *intan* output is routed. The Interrupt Line Register is written to by the system software on power-up; the default value is FF hex (see Table 18).

Table 18. Interrupt Line Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	int_ln	Read/Write	Interrupt Line Register

Interrupt Pin Register (Offset = 3D Hex)

The Interrupt Pin Register is an 8-bit read-only register that defines the pci_mt PCI bus interrupt request line to be *intan*. Its default value is 01 hex (see Table 19).

Table 19. Interrupt Pin Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	int_pin	Read	Interrupt Pin Register

Minimum Grant Register (Offset = 3E Hex)

Minimum Grant Register is an 8-bit read-only register that defines the length of time the pci_mt would like to retain the mastership of the PCI bus. The value set in this register indicates the required burst period length in 250-ns increments. The pci_mt request a timeslice of 4 microseconds. Its default state is 10 hex (see Table 20).

Table 20. Minimum Grant Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	min_gnt	Read	Minimum Grant Register

Maximum Latency Register (Offset = 3F Hex)

Maximum Latency Register is an 8-bit read-only register that defines the frequency in which the pci_mt would like to gain access to the PCI bus. The value set in this register is 00 hex, which indicates that the pci_mt has no major requirements for maximum latency (see Table 21).

Table 21. Maximum Latency Register Format

Data Bit	Mnemonic	Read/Write	Definition
[7:0]	max_lat	Read	Maximum Latency Register

PCI Bus Transactions

This section describes pci_mt PCI bus transactions. The pci_mt accesses the PCI bus for three types of transactions:

- Device configuration
- Target
- Master

The AT40K_PCI core kit comes with 3 simulation files. The following descriptions are extracts from the full waveform files supplied with the core.

- pci_mt.CMD: the command file, defines the setting (signals, timing,..) for the pre-layout simulation of the core.
- pci_mt.SEN: contains the test vectors, required by the pci_mt.CMD at run-time. Each group of vectors from pci_mt.SEN is associated with a command line in the pci_mt.CMD file. For an easy follow-through, each group has its own ID, at the beginning (i.e., l00_Hardware Reset, l00a_***,..). This way a test vector group from the pci_mt.SEN file can be easily paired-up with its associated command line from the pci_mt.CMD file. We will identify these groups as GROUP_00, GROUP_00a, etc.
- pci_mt.WFM: the waveform file, generated as a result of the pre-layout simulation run (running the pci_mt.CMD and pci_mt.SEN under ***viewsym***).

Configuration Transactions

A configuration transaction is generated by either a host-to-PCI bridge or PCI-to-PCI bridge access. In the address phase of a configuration transaction, the PCI bridge will drive the *idsel* signal on the PCI bus agent that it wants to access. If a PCI bus agent decodes the configuration command and detects its *idsel* to be high, the agent will claim the configuration access and assert *devsel*.

GROUP_00a (Configuration READ, DEVICE ID & VENDOR ID Registers) shows the timing of a pci_mt configuration read transaction. During the address phase, the address of the DEVICE ID & VENDOR ID Registers is present on the *adins[31:0]* bus, while the *cbeins[3:0]* bus provides the configuration read command code (i.e., A hex). Immediately after the address phase, the next clock cycle the master deasserts *framen* and asserts *irdyn*, indicating both of the following:

- The transaction contains a single data phase.
- The master device is ready to read the data that the pci_mt has presented on the *ad[31:0]* bus.

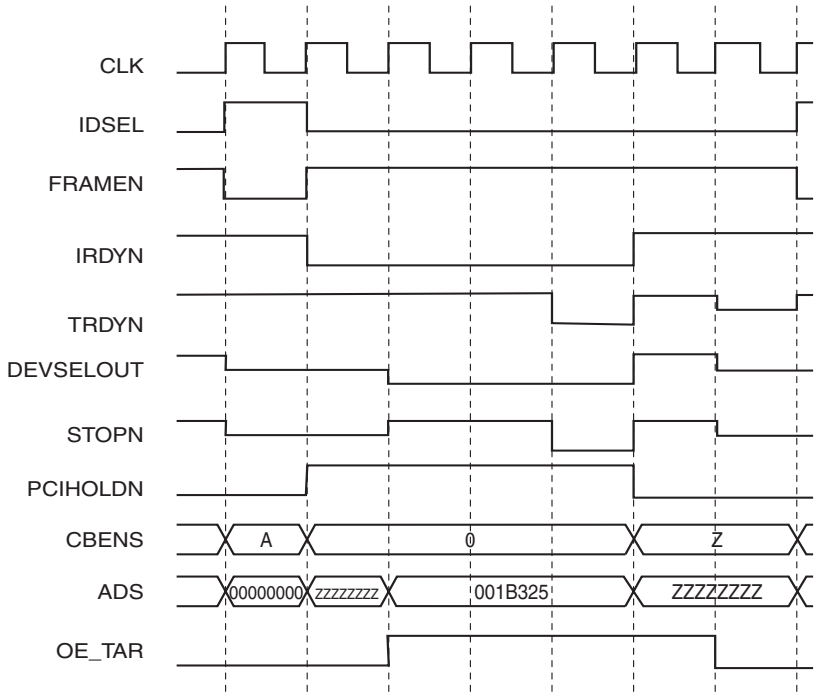
At the same time the master device tri-states the *ad[31:0]* bus (the case of a turn-around cycle).

The next clock cycle pci_mt can drive the *ad[31:0]* bus, by asserting the *aden[3:0]* lines (their state goes from 0000 to 1111, or F hex). It also asserts *devselout*, which indicates to the master device that the pci_mt has accepted the transaction; at the same time, the target starts driving the *devseln*, *trdyn*, and *stopn* signals on the PCI bus, by asserting the *OE_TAR* signal. The *devselout* is then sampled by the master device (as *devseln*) on each rising-edge of the clock. Two clock cycles later (the pci_mt is a slow decode device during target and configuration read/write transactions) pci_mt asserts *trdyout*. At the same time the *irdyin* signal is still asserted by the master, therefore a data transfer takes place. Thus, the data present on the inner *ados[31:0]* bus is transferred to the *ad[31:0]* bus as the value of the DEVICE ID & VENDOR ID Registers. Because the pci_mt does not support configuration read/write burst accesses, it will assert *stopout* to indicate a disconnect to the master. The master will sample the *stopn* signal and will subsequently end the transaction by deasserting *irdyin*.

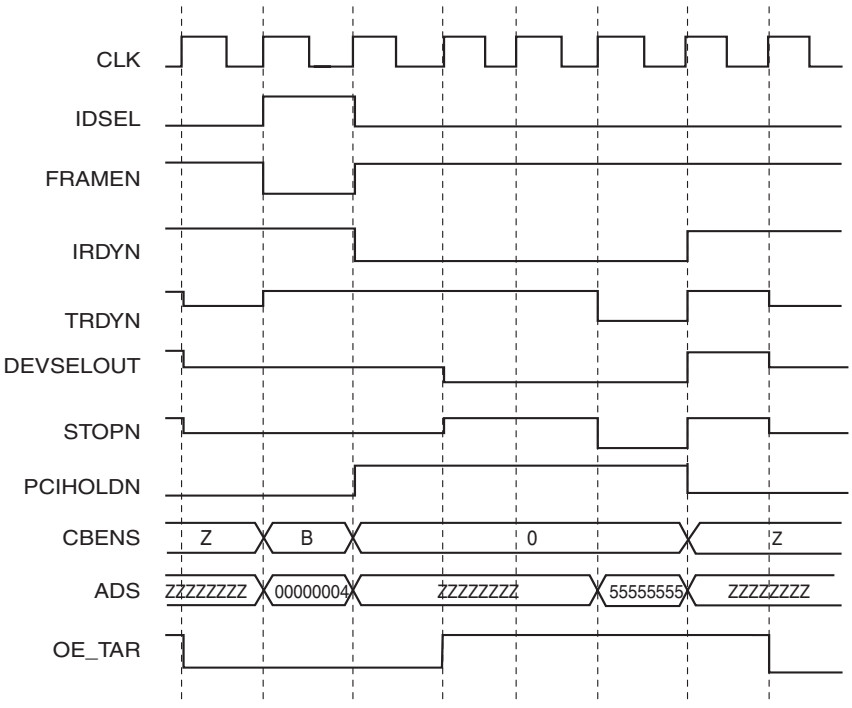
After the last data transfer cycle of the current transaction, the target device keeps the OE-TAR asserted for one more cycle, at the same time driving *devseln*, *trdyn*, and *stopn* high. Thus, the sustained tri-state signal requirement is met, i.e., driving the signal high one clock cycle before releasing it.

GROUP_01 (Configuration WRITE, STATUS & COMMAND Registers, 5555h) shows the timing of a pci_mt configuration write transaction. The protocol is identical to the protocol discussed in the PCI Configuration Read Transaction except for the command code (which is B hex for configuration write) and the *aden[3:0]* bus, which is no longer asserted after the address phase.

Configuration READ, Device ID and C Vendor ID Register (Group_00a)



Configuration WRITE, Status and Command Registers (Group_01)



Target Transactions

A target read/write transaction begins after the master acquires mastership of the PCI bus and asserts *framen* to assert the beginning of a new transaction. The pci_mt latches the address and command signals on the first clock edge when *framen* is asserted and starts the address decode phase. The pci_mt supports two types of target read/write transactions:

- *Internal target read*: Target read/write transaction from the internal DMA registers.
- *External target read*: Target read/write transaction from the local side target memory space.

Target Read Transactions

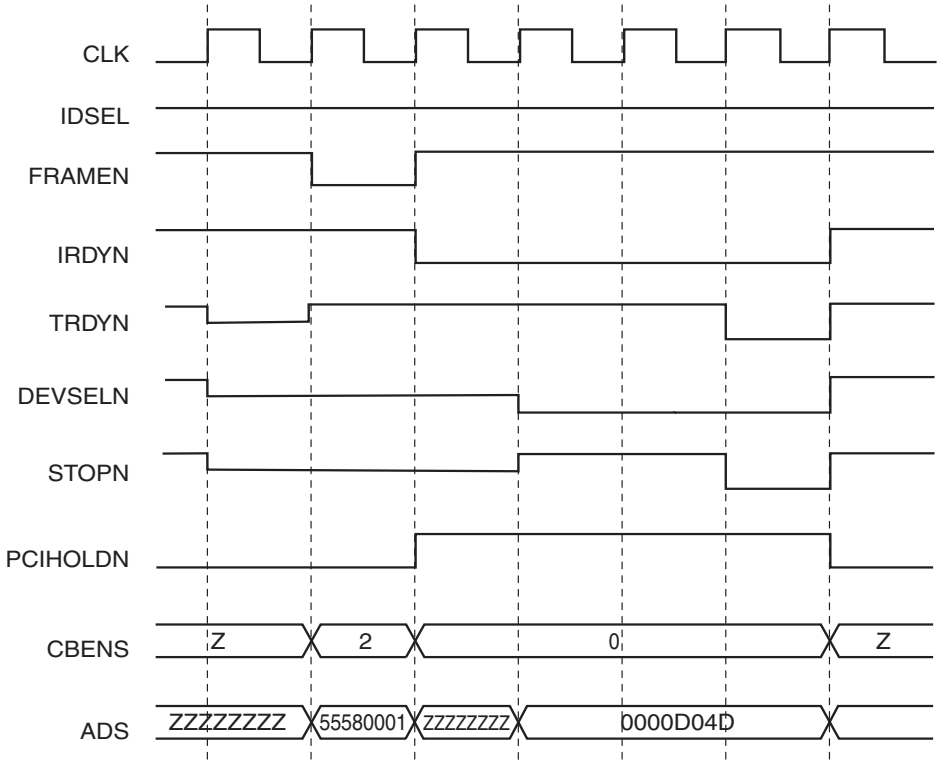
The sequence of events in both target read transactions (Internal & External) is identical; however, the timing is not. A target read transaction from the local side target memory space requires more time because the pci_mt must wait for the local side.

GROUP_0e (DMA Configuration READ, Terminal Count & Command REGISTER) shows the timing of a pci_mt Internal Target (DMA register) read transaction. The protocol is identical to the protocol discussed in the PCI Configuration Read Transaction except for the target registers, which are the DMA Terminal Count Register, the DMA Address Register & the DMA Command Register.

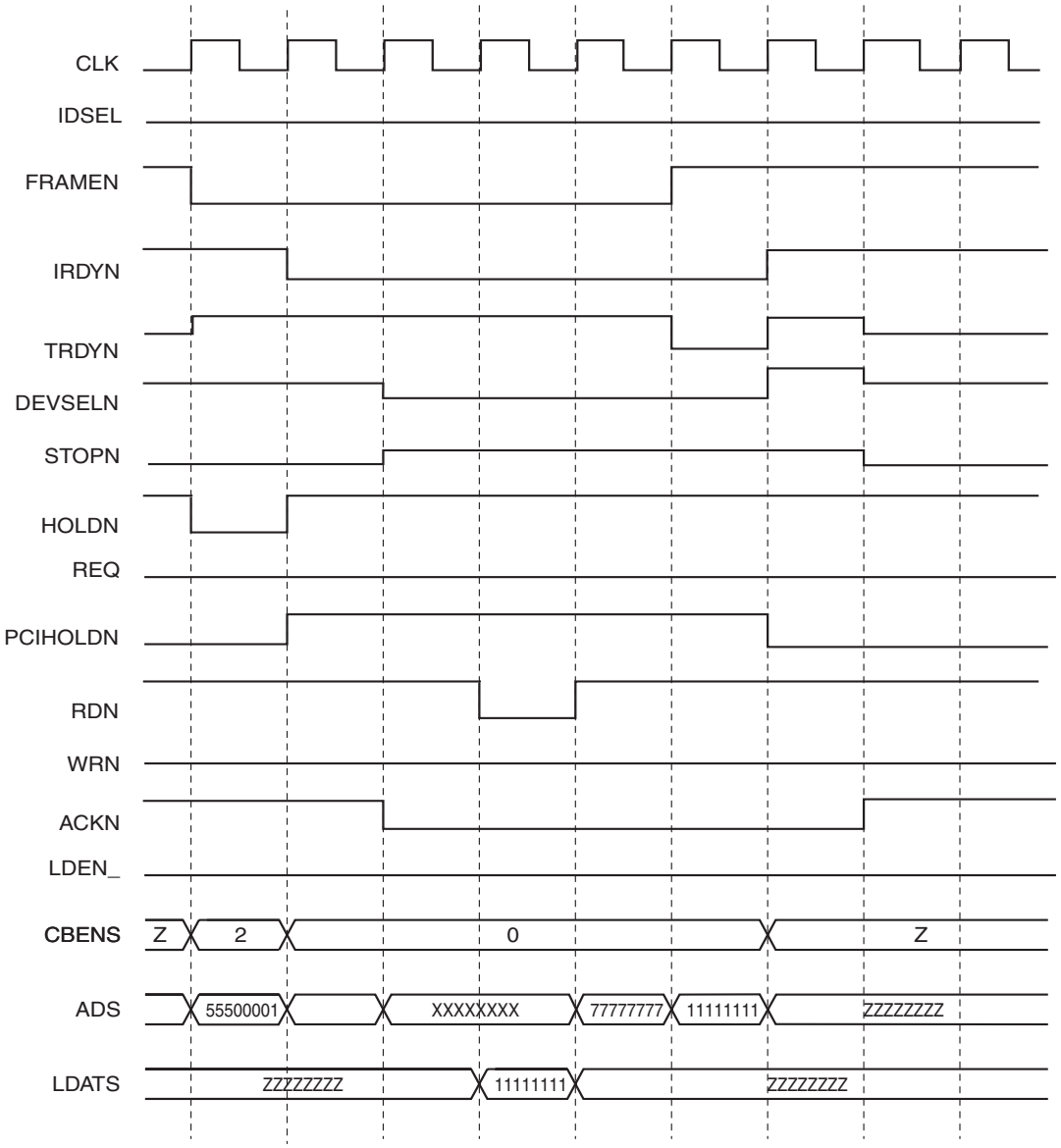
GROUP_1F (I/O READ, 1 DWord shows the timing of a pci_mt External Target (DMA register) read transaction. The protocol is identical to the protocol discussed in the PCI Configuration Read Transaction except for the target, which is the local side target memory page.

GROUP_1e, I/O WRITE, 4 DWords, WAIT before the fourth DWORD, shows the timing for a Target WRITE transaction.

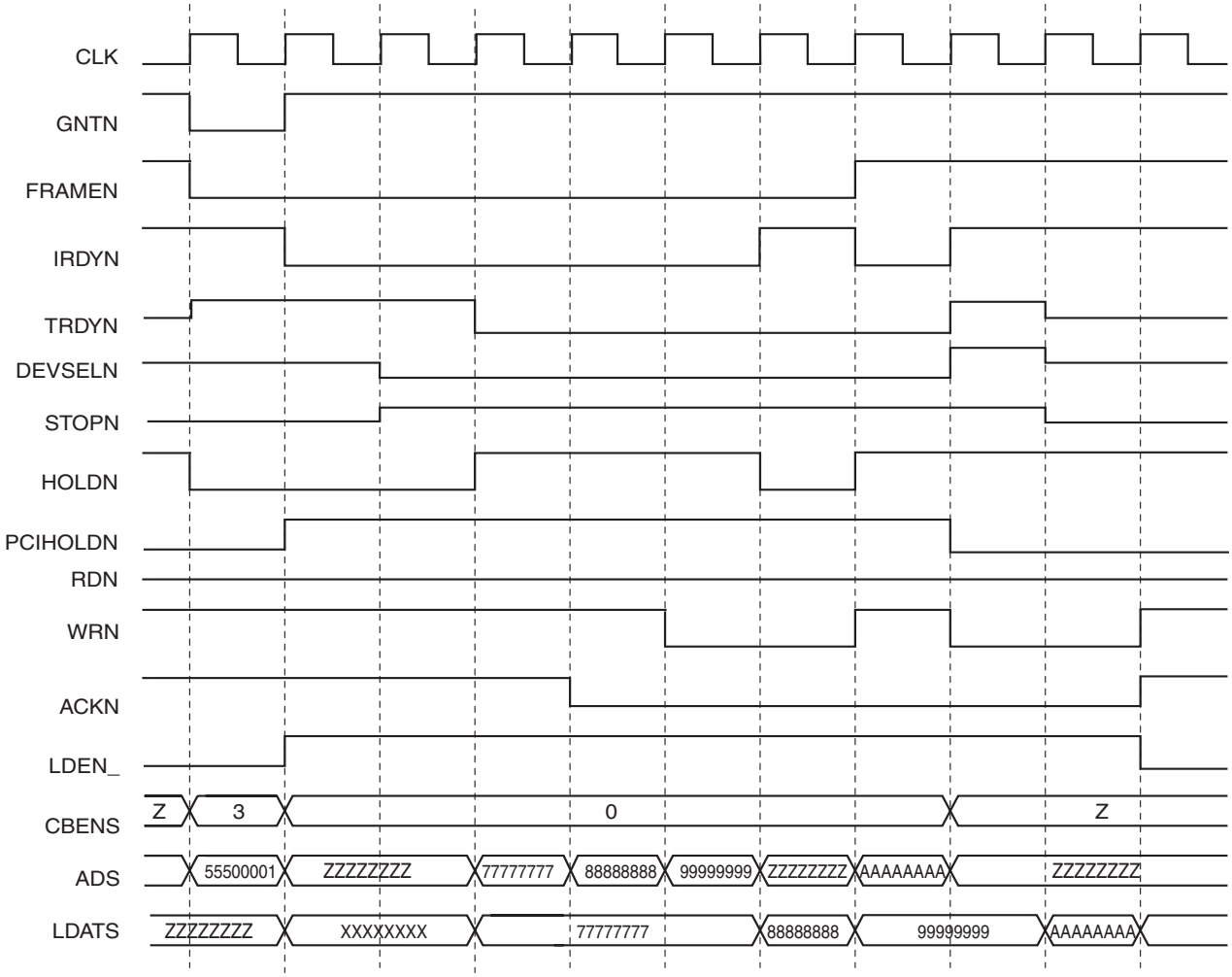
DMA Configuration READ, Terminal Count and Command Register (Group_Oe)



I/O READ, 1 DWORD (Group_1F)



I/O WRITE, 4 DWORDS, WAIT Before the Fourth DWORD (Group_1e)



DMA Operation

This section provides operating details of the DMA engine, and it covers the following:

- Target Address Space
- Internal target register memory map
- DMA Registers
- DMA transactions
- General Programming guidelines

Target Address Space

The pci_mt memory-mapped target registers (internal and external) are read and/or written over the PCI bus in Table 15 memory space. Accesses to or from BAR1 memory space occur in 32-bit transfers. Table 22 list the pci_mt memory space address map. The pci_mt BAR1 address space is 1M byte of contiguous address divided into two 512K byte spaces. The lower 512K byte region (internal target address space) contains the pci_mt DMA control registers, and the upper region (external target address space) contains user-defined I/O space.

Table 22. Memory Space Address Map

Memory Space	Block Size (DWORDS)	Address Offset	Words Used/Reserved	Read/Write	Description
BAR1	128	00000h-7FFFFh	3/128K	Read/Write	DMA Registers
BAR1	128	80000h-FFFFFh	128K/128K	Read/Write	User-defined I/O space (128K DWORDS)

Internal Target Registers Memory Map

Internal pci_mt target address space is used for the DMA registers, including the DMA Terminal_Count, Control & Status register, DMA Command Enables register, and the DMA Address Counter register. Table 23 lists the pci_mt DMA registers memory map.

Table 23. Internal Target Registers Memory Map

Range Reserved	Bytes Used	Read/Write	Mnemonic	Default State (Hex)	Register Name
00000h-00003h	00 – 03	4/4	dma_tccs	00000000	DMA Terminal_Count, Control & Status
00004h-00004h	00 – 00	0/2	dma_cesr	00000000	DMA Command Enables Select
00008h-0000Bh	00 – 03	4/4	dma_acr	00000000	DMA Address Counter

DMA Registers

This section describes the DMA registers. The specified default state is defined as the state of the storage element when the PCI bus is reset. The pci_mt contains the following DMA registers:

- Terminal_Count, Control & Status
- Byte Enables
- Address Counter

Terminal_Count, Control and Status Register (Offset = 00000 Hex)

The DMA Terminal_Count, Control & Status register (dma_tccs) configures the pci_mt DMA engine, directs the pci_mt operation, and provides status of the current memory transfer (see Table 24).

Table 24. DMA Terminal_Count, Control and Status Register Format

Data Bit	Mnemonic	Read/Write	Definition
0	lrst	Read/Write	Local reset. This bit serves as a software reset to the local side add-on logic (see Table 2). The *LRESET* output of the pci_mt is active as long as the *lrst* bit is low. The *LRESET* output is also active for PCI bus resets.
1	flush	Read/Write	Flush Buffer. When high, *flush* marks all bytes in the internal 2 x 32-bit buffer as invalid and resets *dmatc* and *adloaded* (bits 10 and 11). The *flush* bit also resets itself; therefore, it always reads as zero. The *flush* bit should never be set while *dmaon* is set, because a DMA transfer is in progress.
2	intena	Read/Write	PCI interrupt enable. It enables the *intan* output when either the *errpend* or *dmatc* bits are driven high, or when *REQ* signal is active.
3	tcidis	Read/Write	Transfer complete interrupt disable. When high, it disables *dmatc* (bit 10) from generating PCI bus interrupts.
4	dmaen	Read/Write	DMA enable. When high, it allows the pci_mt to respond to DMA requests from the local side (*REQ*) as long as the PCI bus activity is not stopped due to a pending interrupt, etc.
5	burst	Read/Write	Burst Enable. When high, the master transfers 4 DWORDS per transaction; if low, the master transfers 1‘ DWORD only.
6	intirq	Read	When high, it indicates that the local side is requesting an interrupt, i.e., the *REQ* input is asserted.
7	errpend	Read	When high, it indicates that an error occurred during a pci_mt-initiated PCI bus transfer, and the interrupt handler must read the PCI configuration status register and clear the appropriate bits. Any one of the following three PCI status register bits can assert *errpend*: *mstr_abrt*, *tar_abrt*, and *det_par_err*. See Control & Status Register.
8	intpend	Read	The pci_mt automatically asserts *intpend* to indicate that a pci_mt interrupt is pending. The three possible interrupt signals from the pci_mt are *err_pend*, *dma_tc*, and *intirq*.
9	dmaon	Read	DMA on. When high, it indicates that the pci_mt can request mastership of the PCI bus (*reqn*) if prompted by the local side (i.e., an active *REQ*). The *dmaon* bit is high when the address is loaded (*adloaded*), the DMA is enabled, and there are no pending errors. The DMA transfer sequence actually begins when the *dmaon* bit becomes set. Under normal conditions (i.e., DMA is enabled and no errors are pending) the *dmaon* bit becomes set when a write transaction to the DMA address counter register occurs.
10	dmatc	Read	When high, it indicates that the pci_mt-initiated DMA transfer is complete. When the pci_mt sets the *dma_tc* bit, an interrupt will be generated on the *intan* output as long as interrupts are enabled by the *intena* bit (bit 2) and not disabled by the *tcidis* bit (bit 3). The *dmatc* bit is reset in one of three ways: a read transaction to the *dma_tccs*; a write transaction to the *dma_tccs*, which sets the *flush* bit (bit 1); or by asserting the *rstn* input from the PCI bus.

Table 24. DMA Terminal_Count, Control and Status Register Format (Continued)

Data Bit	Mnemonic	Read/Write	Definition
11	adloaded	Read	When high, it indicates that the address has been loaded via the *dma_acr*. This bit is cleared in one of three ways: when the DMA operation is complete and the *dmatc* bit is set; when the *flush* bit is set; or when the *rstn* input is asserted from the PCI bus. The *adloaded* bit triggers the beginning of a DMA operation and is automatically set by the pci_mt when a write operation to the *dma_ccst* is performed. Therefore, the *dma_acr* should be written to last when a DMA operation is being loaded into the DMA registers.
[13-12]	Unused	–	–
[31:14]	tc	Read/Write	Terminal Counter Register.

DMA Transactions

The pci_mt DMA engine, which consists of a 4 x 32-bit FIFO and three programmable registers, is the control channel when the pci_mt acquires mastership of the PCI bus.

As a master device, the pci_mt performs DMA read and write transactions to system memory, or to another PCI bus agent capable of accepting burst target data transfers.

A DMA read transaction from memory to the local side consists of two separate transfers:

- One or more (4, if *burst*) PCI bus DWORD reads from the PCI bus to the FIFO.
- An equivalent number of DWORD transfers from the 32-bit register to the local side.

All DMA read transactions from the pci_mt use a memory read command. For example, see “DMA Configuration READ, Terminal Count and Command Register (Group_Oe)” on page 15 and “I/O READ, 1 DWORD (Group_1F)” on page 16.

Similarly, a DMA write transaction from the pci_mt to system memory consists of two separate transfers:

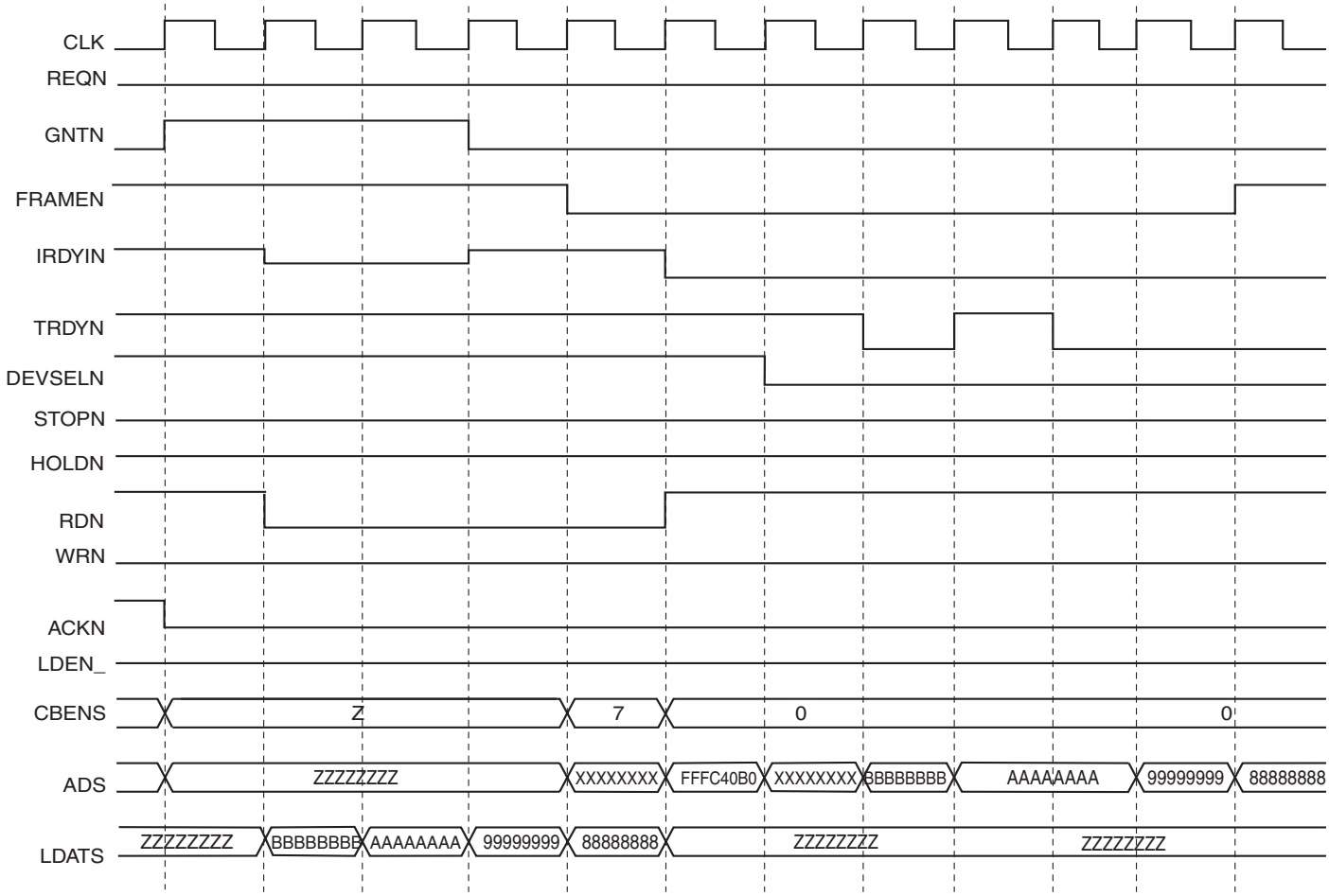
- DWORD transfers from the local side to the FIFO.
- One or more (4, if burst) PCI DWORD writes from the FIFO to a PCI agent.

All DMA (PCI bus) write transactions from the pci_mt use the memory write command.

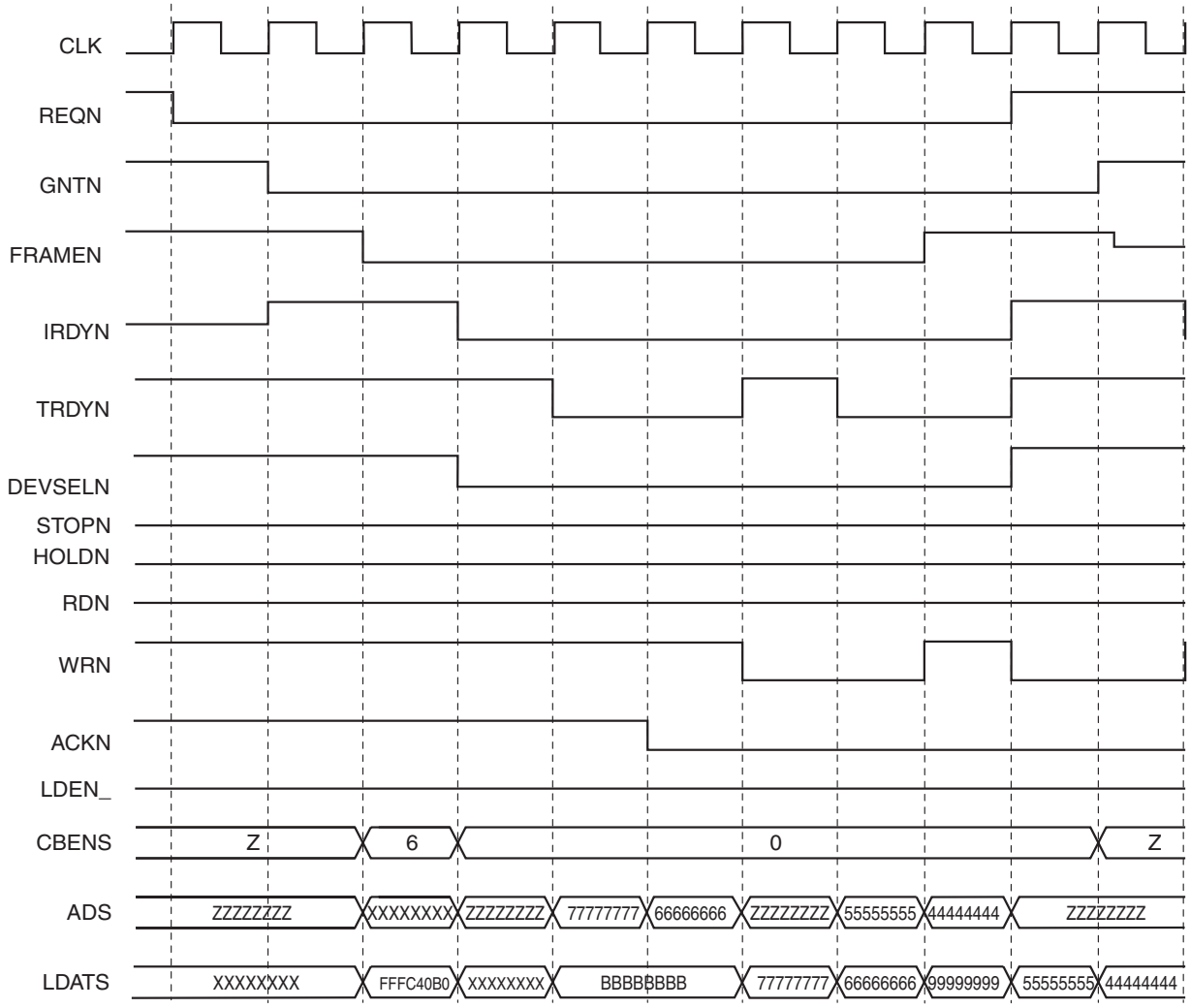
Group_26a1, MASTER WRITE, 4 DWORDS, WAIT before the second DWORD, shows the timing for a MASTER WRITE transaction.

GROUP_30a2, MASTER READ, 4 DWORDS, WAIT before the third DWORD, shows the timing for a MASTER READ transaction.

Master WRITE, 4 DWORDS, WAIT Before the Second DWORD (Group_26a1)



Master READ, 4 DWORDS, WAIT Before the Third DWORD (Group_30a2)



Implementation Description

Figure 3. Top Level Core Interface

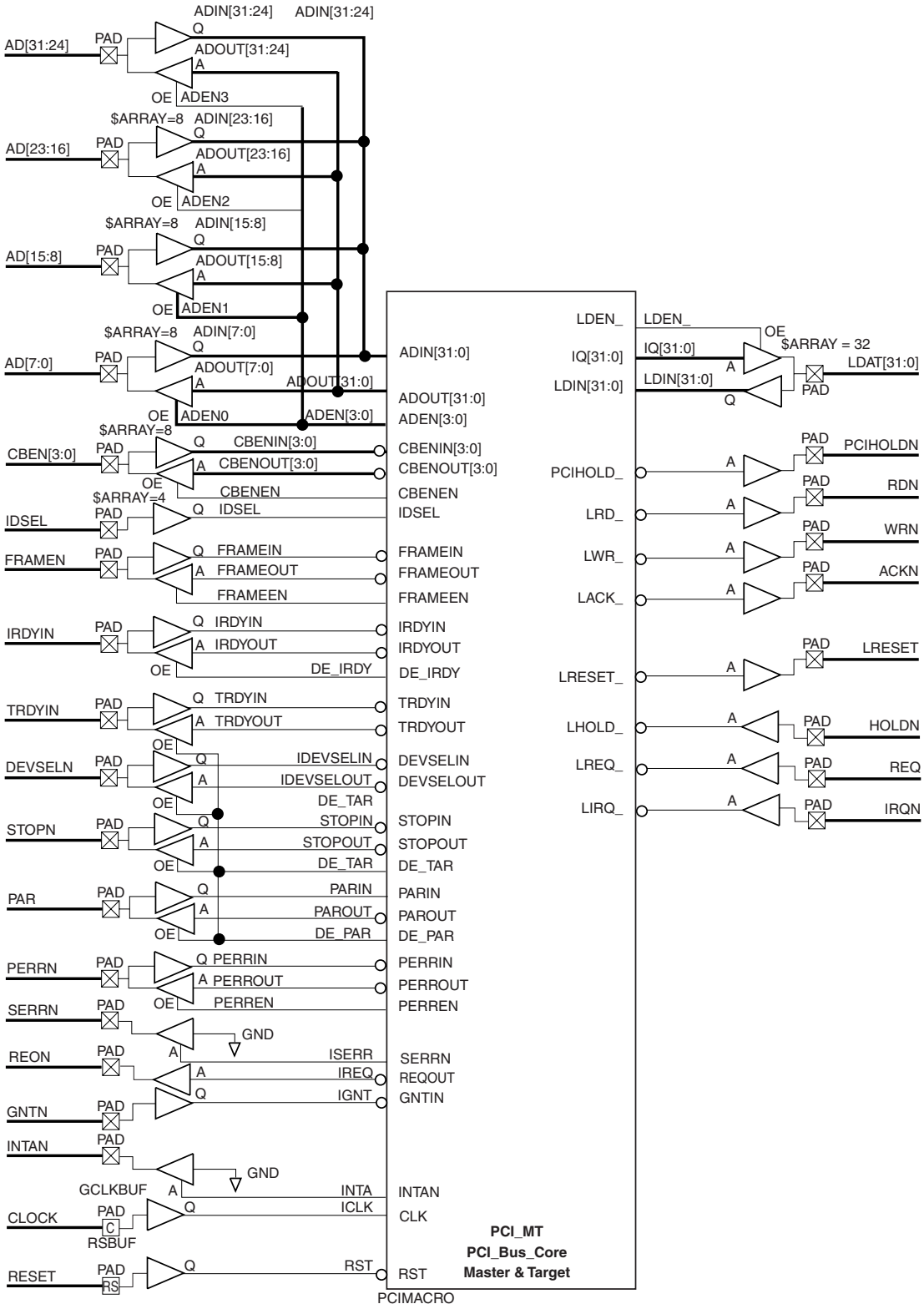
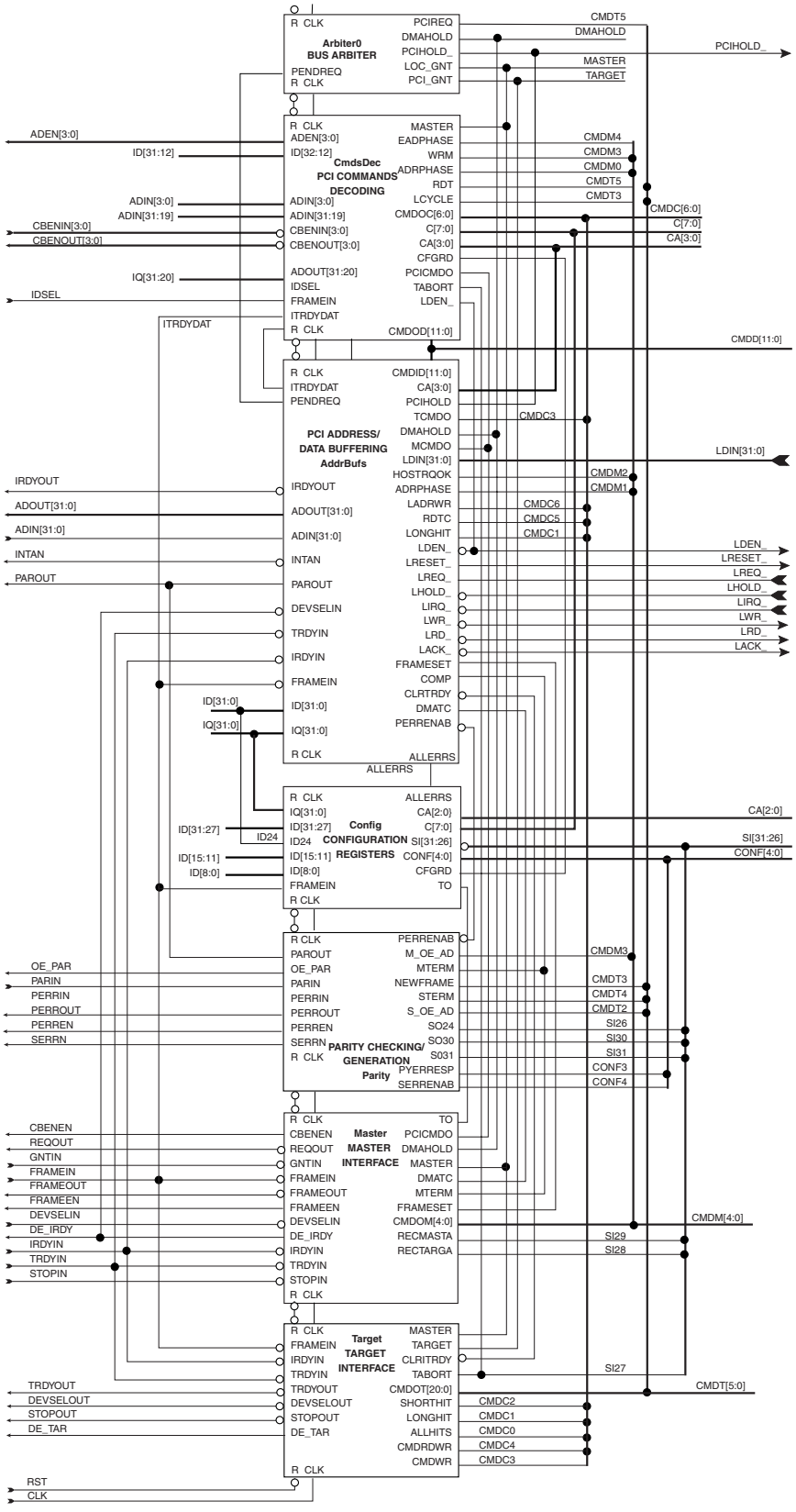


Figure 4. Internal Core Block Diagram



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Atmel FPGA Hotline
1-(408) 436-4119

Atmel FPGA e-mail
fpga@atmel.com

FAQ
Available on web site

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1083B–06/01/15M