TURBOchannel Interface ASIC Functional Specification

Revision 1.0

Jim Crapuchettes
TRI/ADD Program
Workstation Systems Engineering
Digital Equipment Corporation
Palo Alto, CA 94301

June 21, 1993

Blank Page

Table of Contents

List of Figures

List of Tables

Revision History

Date	Version	Content/Change
28 Dec 90	0.0	- Initial Rough Draft
10 Jan 91	0.1	- Changed "user" to "option". - Added definitions of transmit and receive. - Added op.den. - Changed Register Addresses to make them longword aligned.
18 Feb 91	0.2	- Changed register address. - Changed the op.~trValid spec. - Removed 2-cycle I/O write. - Changed CSR bit locations for easier hardware implementation.
22 Mar 91	0.3	- Many Major changes.
15 Oct 91	0.4	- Added Scatter/Gather Table Function. - Changed Option Side Programmed I/O protocol to resemble TURBOchannel, regardless of option clock speed. - Shifted OCR address locations. - Added Timing Diagrams. - Added Application Examples.
18 Nov 91	0.5	- Added threshold to receive DMA burst. - Added counter-based flush of receive FIFO. - Added transmit burst size/word count access to register file for intelligent controllers. - Added pin assignments. - Programmable page sizes enhanced.
08 Mar 92	0.5C	- Implemented additional parity support features. - Added programmable slot size. - Changed name to TcIA. - Added detailed Transmit DMA programming steps. - Changed name of OCR register to OCR1. - Changed name of MISC register to OCR2. - Added twc=0 & tfe=1 interrupt. - Changed locations of OCR1, OCR2, TSGP, RSGP. - Removed address align feature. - Continuous bit removed; redundant.
31 Aug 92	0.6	-Renamed RQFULLWORDS and TQFREEWORDS fields in OCR2 to RFFULLWORDS and TFFREEWORDS. -Revised, corrected and expanded DMA programming sections. -Made many minor formatting changes. -Package pin assignment table and diagram moved to end. -Company Confidential removed from all figures. -Removed remaining references to superseded functions.

21 Jun 93	1.0	-Corrected and improved Figure 1 -Added FAR as name of FIFO Access Register -Added to signal descriptions in Chapter 2

1 Introduction

This document defines the functional specification for the TURBOchannel Interface ASIC (TcIA).

The TcIA is a general purpose ASIC designed for use on TURBOchannel option modules. The design is meant to be applicable to a majority of TURBOchannel option applications.

Most TURBOchannel options are one of these three types:

- Options requiring only programmed I/O; these comprise a major portion of TURBOchannel option applications.
- Simple DMA (Direct Memory Access) devices that do not generate their own address but simply echo data to and from memory; these represent a majority of TURBOchannel option DMA applications.
- Intelligent controller options that supply their own DMA address and execute their own instructions.

TcIA features, listed in following sections, support all of these option types with a single device.

1.0.1 Transmit and Receive and Other Definitions

The terms transmit and receive are very important to the understanding of this specification Since there are several devices that "transmit" and "receive", direction can be confusing. Throughout this document these terms have a specific and consistent meaning.

<u>Transmit</u> DMA is a movement of data <u>from</u> system memory; thus, a transmit uses DMA read transactions.

<u>Receive</u> DMA is a movement of data <u>to</u> system memory; a receive uses DMA write transactions.

The following definitions are used throughout this document:

"DMA" (Direct Memory Access) is the mechanism that allows an I/O option to directly access system memory through the I/O interconnect (the TURBOchannel, in this case) without intervention by the CPU.

"byte" denotes an 8 bit value.

"half-word" denotes a 16 bit value.

"word" denotes a 32 bit value.

A "physical word address" is the high 32 bits of the TURBOchannel 34 bit physical memory address. It is generated from the physical memory address by an unsigned right shift of 2 bits. This is the form of system memory address used in all TcIA pointer registers and in scatter/gather tables in system memory.

An "I/O transaction" takes place when the system module writes to or reads from an option module. It consists of an address cycle and a data cycle.

A "DMA operation" is the software view of the TcIA's support for DMA. The TcIA will perform one or more DMA transactions to carry out a DMA operation.

A "DMA transaction" takes place when a TURBOchannel option module reads from or writes to the system module. It consists of an address cycle and a burst of data cycles.

A DMA "burst" consists of the flow of one or more data words during a DMA transaction. **NOTE:** this is NOT the same definition of burst used in the *TURBOchannel Hardware Specification*.

1.0.2 Related documents

The *TURBOchannel Specifications* is a set of documents describing the TURBOchannel. The *TURBOchannel Hardware Specification* defines the electrical, mechanical, protocol and timing characteristics for the TURBOchannel; it should be used as the reference for all TURBOchannel signals. The *TURBOchannel Firmware Specification* defines the address space available to a TURBOchannel option and the contents (header and objects) of the Read-Only Memory (ROM) which must be present on any option. The *TURBOchannel System Parameters* defines the characteristics and limitations for each of the systems implementing TURBOchannel.

1.1 Programmed I/O and Other Standard Features

- An address latch is provided on chip.
- The programmed I/O subsystem is simple, complete and independent of the DMA subsystem.
- The option has direct control of situations requiring the use of ~conflict.
- The TcIA provides synchronization so that the option side can operate at a different frequency than the TURBOchannel.
- The option may also be placed in a synchronous mode so that no extra delay is introduced when the option is run from the TURBOchannel clock.
- 8 read-write general purpose output bits; 4 readable general purpose input bits that can be used to generate TURBOchannel interrupts.
- TURBOchannel parity generation and checking.

1.2 Simple DMA Features

Simple DMA is defined as the sequential transfer of data between system memory and the option through the TURBOchannel with memory addresses supplied by the TcIA.

- Variable option-data size (byte, half-word or word).
- Word to option-data unpacking on transmit DMA operations.

- Option-data to word packing on receive DMA operations.
- Selectable byte ordering convention (endian-ness).
- Word addressable transmit and receive DMA buffer pointers.
- Transmit and receive buffer pointers can be reloaded from scatter/gather tables when the data crosses page boundaries to support DMA operations using virtually contiguous buffers that are physically scattered in memory.
- Transmit and receive DMA word counters (24 bits each).
- Independent 15 word transmit and receive FIFOs.
- A DMA operation stops when a word counter is zero and can be stopped by an I/O write to reset a DMA enable bit.
- Various features to support DMA operations that do not begin and end on word boundaries.
- Receive DMA transactions are processed at maximum TURBOchannel bandwidth with a programmable minimum burst size. A programmable flush counter (timer driven by the TURBOchannel clock) prevents data from becoming stale in the FIFO when less than the minimum burst size.
- Transmit DMA transactions are processed with programmable fixed bursts from 1 to 127 words.
- TURBOchannel DMA word address scrambling.

1.3 Additional Features for Intelligent DMA controllers

Intelligent DMA is defined as the transfer of data between system memory and the option through the TURBOchannel with memory addresses supplied by the option-side hardware. It is assumed that these options will contain on-board processors.

- Physical word addresses are supplied externally for DMA (memory) reads and writes.
- External addresses may be any width up to 32 bits.
- The word count and burst size for transmit DMA are programmed from the option interface of the TcIA.

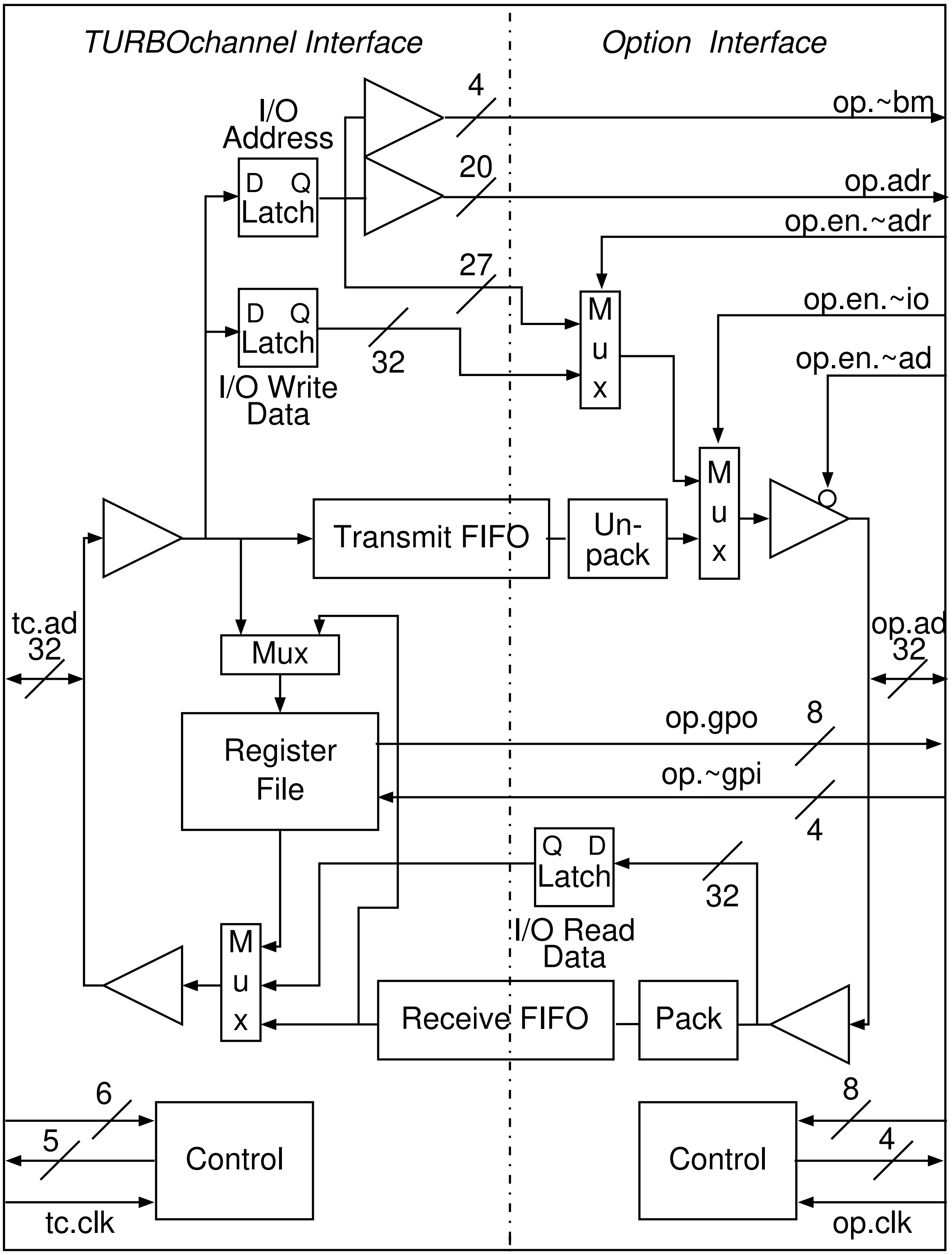
1.4 Block Diagram description

Figure 1, **Block Diagram of the TcIA**, on page 5, shows the major functional units of the TcIA. The left side of the diagram, labeled *TURBOchannel Interface*, contains the major elements of the TcIA that operate in the TURBOchannel time domain. The right side of the diagram, labeled *Option Interface*, contains the major elements of the TcIA that operate in the option time domain. Blocks in the diagram are functions within the TcIA, lines show signal or data connections, arrows on lines show the direction of signal or data flow.

The functions of the major blocks are

- Register File block. This contains all of the registers described in Section 3. The connections between the option side of the TcIA and GPO and GPI fields, accessible through Option Control Register 1 (OCR1) and the Option Interrupt Register (OIR), are not shown in the diagram.
- Latch blocks. The I/O Address, I/O Write Data and I/O Read Data latches hold data that are traveling between the TURBOchannel time domain and the option time domain.
- Transmit FIFO and Unpack blocks. Transmit DMA data is transferred from the TURBOchannel to the transmit FIFO, then to the unpacker (which unpacks only when option data is less than 32 bits) and out to the option. The flow of data from the TURBOchannel is controlled by the burst size fields in Option Control Register 1 (OCR1) and the Transmit Word Count register (TWC).
- Pack and Receive FIFO blocks. Receive DMA data is transferred from the option through the packer (which packs only when option data is less than 32 bits), then to the receive FIFO and out to the TURBOchannel. The flow of data to the TURBOchannel is controlled by the receive burst and receive flush count fields in Option Control Register 2 (OCR2) and the Receive Word Count register (RWC).
- Mux blocks. The four multiplexers select one of several data sources. The two in the path from the TURBOchannel to the option select their data under the control of option hardware. The one in the path from the option to the TURBOchannel and the one in the path from the TURBOchannel to the register file select their data under control of the TcIA.

Figure 1   Block Diagram of the TcIA



Blank Page

2 Signal Description

2.1 Option Interface Signals

Notes: The tilde character "~" in a signal name denotes a signal with a low assertion level. The character(s) in parentheses describe the pin signal direction(s).

- op.ad[31:0] (I/O) 3-state, multiplexed option interface address and data.
- op.adr[19:0] (O) Latched I/O address bits <21:2>, taken from tc.ad[26:8] during an address cycle. These bits can be used to specify an option address when 20 bits are sufficient. When more than 20 address bits are needed on the option, the full 27 address bits supplied on the TURBOchannel are available by asserting op.en.~adr, op.en.~io and op.en.~ad during an address cycle and latching the data present on the op.ad lines.
- op.~bm[3:0] (O) Latched byte mask for I/O read and write transactions, taken from tc.ad[4:1] during the address cycle. These bits are not used for register address decoding within the TcIA.
- op.clk (I) Option interface clock.
- op.en.~ad (I) Assert to enable op.ad drivers.
- op.en.~io (I) Deasserted selects transmit FIFO data onto op.ad lines; assert to select I/O address/data onto op.ad lines.
- op.en.~adr (I) Deasserted selects data onto op.ad lines; assert to select latched TURBOchannel I/O address bits <31:5> onto op.ad[28:2] (op.ad[31:29] and op.ad[0:1] are set to 0).
- op.~gpi[3:0] (I) Status bits visible in OIR; they can be enabled to request interrupts.
- op.gpo[7:0] (O) General purpose outputs that reflect bits in OCR1.
- op.synchronous (I) Deasserted configures the TcIA in asynchronous mode; assert to configure the TcIA in synchronous mode.

  NOTE

  When operating the TcIA in synchronous mode with an option data size of 8 or 16 bits, the transmit DMA burst sizes MUST NOT be greater than 15.
- op.~conflict (I) Assert to drive tc.~conflict, aborting the ongoing I/O transaction.

  NOTE

  This signal must be asserted with op.~rdy.
- op.~rcvAdr (I) When op.~rcvAdr is asserted without op.~trAdr, the address on op.ad is loaded into the receive FIFO on the next option clock (see

op.synchronous). The receive DMA pointer will be loaded with this address when it reaches the head of the FIFO. Subsequent data will be written to memory at this new address.
When op.~rcvAdr and op.~trAdr are asserted simultaneously, the burst size/word count word on op.ad is loaded into the receive FIFO on the next option clock (see op.synchronous); see description of op.~trAdr.

- op.~trAdr (I) When op.~trAdr is asserted without op.~rcvAdr, the address on op.ad is loaded into the receive FIFO on the next option clock (see op.synchronous). The transmit DMA pointer will be loaded with this address when it is at the head of the FIFO and the transmit DMA word count is zero.
When op.~trAdr and op.~rcvAdr are asserted simultaneously, the burst size/word count word on op.ad (BURSTSIZE on op.ad[31:24] and TWC on op.ad[23:0]) is loaded into the receive FIFO on the next option clock (see op.synchronous). The transmit word count and burst size will be loaded with this word when it is at the head of the FIFO and the transmit DMA word count is zero. The TcIA will then use the current transmit address to read words into the transmit FIFO as specified by the just-loaded burst size/word count word.
- op.~rcvDat (I) Assert to load the data on the op.ad lines into the receive FIFO on the next option clock (see op.synchronous).
- op.~rcvValid (O) Asserted if there are at least RCVTHRESHOLD free option-words in the receive FIFO.
- op.~rdy (I) Asserted to acknowledge PIO read or write data.
- op.~sel (O) Asserted when option selected for PIO read or write.
- op.~trDack (I) When asserted prior to an option clock cycle (see op.synchronous), the next option-word in the transmit FIFO will be available in the following option clock cycle.
- op.~trValid (O) Indicates there is at least one option-word in the transmit FIFO ready for the option.
- op.~write (O) When op.~test is deasserted, op.~write is asserted to indicate that the current PIO transaction is a write and deasserted to indicate that the current PIO transaction is a read.
When op.~test is asserted, op.~write is the output of the parametric nand-tree (see op.~test description).
- op.~test (I) When asserted, disables all output bi-directional pin drivers except tc.~rReq, tc.~wReq and op.~write. op.~write then becomes the output of the parametric nand-tree, a tree of nand gates which is connected to all of the input pins; this allows for simple testing of all input pins (changing the inputs, one at a time, should cause op.~write to change).

2.2 TURBOchannel Interface Signals

Notes: Refer to the *TURBOchannel Hardware Specification* for additional information on these signals. The tilde character "~" in a signal name denotes a signal with a low assertion level. The character(s) in parentheses describe the pin signal direction(s).

- tc.~ack (I) DMA read/write acknowledge.
- tc.ad[31:0] (I/O) TURBOchannel address and data.
- tc.p (I/O) TURBOchannel parity.
- tc.clk (I) TURBOchannel clock.
- tc.~conflict (O) I/O read/write conflict.
- tc.~err (I) TURBOchannel DMA error.
- tc.~int (O) TURBOchannel interrupt.
- tc.~reset (I) Resets the option when asserted.
- tc.~rdy (O) Asserted to complete an I/O read or write.
- tc.~rReq (O) TURBOchannel DMA read request.
- tc.~sel (I) I/O read or write select.
- tc.~write (I) I/O transaction read/write specifier.
- tc.~wReq (O) TURBOchannel DMA write request.

2.3 TcIA Pin Assignment

Table 1, **Package Pin Assignments**, on page 68, shows the package pin assignments for the TcIA. The pin number, direction (i=input, b=bi-directional, o=output), and signal name are shown. Figure 16, **TcIA Package Pin Diagram**, on page 69, shows the package pin diagram for the TcIA. The breakdown of pin types is as follows:

84 Option Interface Signal Pins
44 TURBOchannel Interface Signal Pins
32 VDD/VSS Pins

160 Total Pins

Blank Page

3 Control and Status Registers

Each option in a system responds to a different I/O address range. The parameters SLOTSIZE and SLOTBASE are system dependent variables residing in the system ROM. In general, the address of each slot is defined by:

I/OBASE = (Slot #)*(SLOTSIZE) + SLOTBASE

Any specific address within an option is given by:

Slot Address = I/OBASE + Offset

Offset	ASIC Register or Address Space
0x300	Transmit DMA Pointer (TDP)
0x304	Receive DMA Pointer (RDP)
0x308	Transmit DMA Scatter/Gather Pointer (TSGP)
0x30C	Receive DMA Scatter/Gather Pointer (RSGP)
0x310	Option Control Register 1 (OCR1)
0x314	Option Control Register 2 (OCR2)
0x318	DMA Pointer Mask (DPM)
0x31C	Returns 0x00000000 when read.
0x320	Transmit DMA Word Counter (TWC)
0x324	Receive DMA Word Counter (RWC)
0x328	Option Interrupt Register (OIR)
0x32C	Option Interrupt Mask Register (OIMR)
0x330	FIFO access. (FAR)
0x334 to 0x33C	Reserved Space
0x340 to end of Slot	ROM and Option I/O Space

The CSRs are placed beginning at slot offset 0x300. Writes and reads to these registers will not be visible at the option interface, (op.~sel will not assert). The register file is word based, and programmed I/O access to the register file must be word based. Byte masks are not valid for register file reads or writes. Note that the option ROM, which starts at an offset of 0x3E0, does not conflict with these register addresses.

3.0.1 Reset State

When tc.~reset is asserted during power up/down, all the CSRs are reset to 0's. The FIFOs (which are implemented with RAM words and pointers) are emptied by setting the head and tail pointers to 0, and all outputs are placed in a deasserted state. The tc.ad lines are tristated, but op.ad is under option tristate control. If op.en.~ad is deasserted, op.ad will be tristated by the TcIA. The TcIA requires at least 8 clock cycles of tc.~reset to initialize completely.

3.1 Transmit DMA Pointer (TDP)

<31:0> (R/W) The transmit DMA pointer contains the physical word address that is used to move data from memory into the transmit FIFO. It is incremented as each word is received from the TURBOchannel. If an error occurs then the DMA operation is halted and the error is reported in the OIR. The pointer will contain the address of the data word at the time of the error.

3.2 Transmit DMA Scatter/Gather Table Pointer (TSGP)

<31:0> (R/W) The TSGP contains the physical word address for the transmit DMA scatter/gather table pointer. For DMA operations that cross page boundaries, the controlling software builds a scatter/gather table in main memory. This table should contain the physical word addresses for the first word of any subsequent pages used in the DMA operation. This table must be in a physically contiguous portion of system memory. The address entries in this table are 32 bit word addresses. When a transmit DMA operation reaches a page boundary, as defined by the PAGESIZE field in OCR2, the TcIA reads the next page address pointed to by the transmit DMA scatter/gather table pointer and loads this address into the transmit DMA pointer. The transmit DMA scatter/gather table pointer is then incremented. This feature allows the TcIA and controlling software to transfer virtually contiguous DMA buffer pages that are located in non-contiguous physical memory pages. This scheme allows DMA to continue without interrupting the processor to load a new buffer pointer for the next page.

3.3 Receive DMA Pointer (RDP)

<31:0> (R/W) The receive DMA pointer contains the physical word address that is used to move data from the receive FIFO into memory. It is incremented as each word is written into memory. If an error occurs then the DMA operation is halted and the error is reported in the OIR. The pointer will contain the address of the data in error.

3.4 Receive DMA Scatter/Gather Table Pointer (RSGP)

<31:0> (R/W) The RSGP contains the physical word address of the receive DMA scatter/gather table pointer. This pointer works in the same manner as the transmit DMA scatter/gather table pointer. Upon reaching a page boundary, the receive DMA pointer is loaded with the next pointer value from the receive DMA scatter/gather table. The receive DMA scatter/gather table pointer is then incremented.

3.5 DMA Pointer Mask (DPM)

<31:0> (R/W) These bits contain a mask for both the transmit and receive pointers in intelligent DMA mode. The register is for intelligent controllers that cannot supply a full 32-bit physical word address. When an external transmit or receive address is supplied to the option interface, that address is combined with the corresponding current pointer in the following way:

NewPointer = (OldPointer AND (DPM)) OR (ExternalAddress AND (NOT DPM))
Thus "1" bits in the DPM select bits from the current pointer (they should be the high order bits) and "0" bits in the DPM select bits from the external address (they should be the low order bits) to create a new pointer address. If the option supplies full 32-bit physical memory addresses, this register must be set to zero.

3.6 Option Control Register 1 (OCR1)

<31> (R/W) TDMAEN
1: The TcIA may read words from memory into the transmit FIFO.
0: Reads of memory into the transmit FIFO are disabled.
The option-side signals are unaffected.

<30> (R/W) RDMAEN
1: The TcIA may write receive FIFO words into memory.
0: Words in the receive FIFO are not written to memory.
The option-side signals are unaffected.

<29> (R/W) TFEN
1: Enables the transmit DMA FIFO
0: Clears the transmit FIFO and deasserts op.~trValid.

<28> (R/W) RFEN
1: Enables the receive DMA FIFO
0: Clears the receive FIFO and deasserts op.~rcvValid.

<27:26> (R/W) TUPACK
00:Unpacking starts at byte0, half-word0
01:Unpacking starts at byte1 (Do not use with half-word option data)
10:Unpacking starts at byte2, half-word1
11:Unpacking starts at byte3 (Do not use with half-word option data)
These bits are loaded into the option-data unpacking counter by each op.clk whenever TFEN is 0.

<25:24> (R/W) RPACK
00:Packing starts at byte0, half-word0
01:Packing starts at byte1 (Do not use with half-word option data)
10:Packing starts at byte2, half-word1
11:Packing starts at byte3 (Do not use with half-word option data)
These bits are loaded into the option-data packing counter by each op.clk whenever RFEN is 0.

<23:17> (R/W) BURSTINIT
The value of BURSTINIT must be in the range of 1 through 127 (0x01 through 0x7F); a value of zero should not be used[1]. The transmit DMA engine will move BURSTINIT option-words into the transmit FIFO as the first DMA transaction (of

[1] A value of zero will cause the initial DMA read transaction burst to continue to the end of the page defined by the PAGESIZE field in OCR2.

the whole DMA operation) as long as the TWC is non-zero (after the TWC becomes zero, remaining words in the burst are discarded). If BURSTINIT is less than 15, the TcIA checks to make sure that there are at least BURSTINIT free words in the transmit FIFO before starting a DMA transaction. If BURSTINIT is 15 or greater, the option must empty the FIFO as fast as it is filled.

<16:10> (R/W) BURSTSIZE
The value of BURSTSIZE must be in the range of 1 through 127 (0x01 through 0x7F); a value of zero should not be used[1]. The transmit DMA engine will move BURSTSIZE option-words into the transmit FIFO with each DMA transaction (after the first) as long as the TWC is non-zero (after the TWC becomes zero, remaining words in the burst are discarded). If BURSTSIZE is less than 15, the TcIA checks to make sure that there are at least BURSTSIZE free words in the transmit FIFO before starting a DMA transaction. If BURSTSIZE is 15 or greater, the option must empty the FIFO as fast as it is filled.

<9:8> (R/W) SIZE
Sets the option-data packing/unpacking size and op.ad line utilization for option data.

00: Word	Use op.ad<31:0>
01: Half-word	Use op.ad<15:0>
10: Byte	Use op.ad<7:0>
11: Reserved	

<7:0> (R/W) GPO
The value of these General Purpose Output bits are reflected directly at the output signals op.gpo<7:0>. They assert and deassert in the option time domain.

3.7 Option Control Register 2 (OCR2)

<31:28> (R) RFFULLWORDS
Number of words in receive FIFO based on the head pointer and a tc.clk synchronized version of the tail pointer.

<27:24> (R) TFFREEWORDS
Number of valid free words in transmit FIFO based on the tail pointer and a tc.clk synchronized version of the head pointer.

<23> (R) RESERVED

[1] A value of zero will cause each DMA read transaction burst to continue to the end of the page defined by the PAGESIZE field in OCR2.

<22> (R/W) BIGENDIAN

Selects the packing and unpacking order for option-data bytes or half-words.

1: Packing/unpacking order is most- to least-significant byte (big-endian)

byte 0	byte 1	byte 2	byte 3
half-word 0		half-word 1	

0: Packing/unpacking order is least- to most-significant byte (little-endian)

byte 3	byte 2	byte 1	byte 0
half-word 1		half-word 0	

<21:16> (R/W) RCVFLSHCNT

The value of RCVFLSHCNT, times 16, is loaded into the receive flush counter by each tc.clk when a receive DMA transaction is taking place or when receive DMA is not enabled; RCVFLSHCNT values of 0 through 63 will become count values of 0 through 1008. The receive flush counter is decremented by tc.clk under any other condition. When the counter reaches zero, a receive DMA transaction will start if any words are in the DMA receive FIFO, even if there are less than RCVBURST words; this will limit the amount of time that data words remain in the FIFO and will assure that data at the end of a receive will not remain in the FIFO indefinitely. Note that a value of zero will override the RCVBURST field and will begin a receive burst as soon as one word is in the FIFO.

<15:12> (R/W) SLOTSIZE

These bits must be programmed with the TURBOchannel slot size for the system used if the option can support a slot of greater than 4Mbytes. This allows correct slot-size-dependent address decoding[1]. The TcIA powers-up with the value set to 0000, selecting a slot size of 4 MBytes.

0000: 4 MByte	0010: 16 MByte	0100: 64 MByte	0110: 256 MByte
0001: 8 MByte	0011: 32 MByte	0101: 128 MByte	0111: 512 MByte

1000 to 1111 are reserved for future expansion.

<11:10> (R/W) RCVBURST

These bits set the minimum DMA burst size from the receive FIFO. A DMA write transaction will not be initiated by the receive DMA engine until there are at least this many valid data words at the head of the receive FIFO.

00: At least 1	01: At least 2	10: At least 4	11: At least 8

[1] The *TURBOchannel Hardware Specification* states that for systems with less than 512 Mbyte slot sizes, the high order address bits are undefined and must be treated as *don't care* values.

<9:7> (R/W) RCVTHRESHOLD
000: DO NOT USE. Will corrupt data.
001: Deasserts op.~rcvValid when FIFO free words < 1
010: Deasserts op.~rcvValid when FIFO free words < 2
..
111: Deasserts op.~rcvValid when FIFO free words < 7

<6> (R/W) TRINTELL
0: Simple transmit DMA mode.
1: Intelligent controller transmit DMA mode.

<5:2> (R/W) PAGESIZE
These bits set the scatter/gather block size in bytes. When the TDP or RDP crosses the boundary of a physical memory block of the specified size, it is updated from the corresponding scatter/gather table. This field will usually be set to the memory system page size to support virtual memory transfers. Note that when set to 0000 (none), the TDP and RDP will never be updated from the scatter/gather tables and will point to contiguous physical memory locations.

0000: none	0100: 64	1000: 1K	1100: 16K
0001: 8	0101: 128	1001: 2K	1101: 32K
0010: 16	0110: 256	1010: 4K	1110: 64K
0011: 32	0111: 512	1011: 8K	1111: 128K

<1> (R/W) RCVINTELL
0: Simple receive DMA mode.
1: Intelligent controller receive DMA mode.

<0> (R/W) PEN
1: Enable the parity error check logic.
0: Disable the parity error check logic PER

3.8 Option Interrupt Mask Register (OIMR)

<31:0> (R/W) Each bit in this register is used to enable the corresponding bit in OIR<31:0> to request an interrupt. For example when OIMR<0> is set to a 1, OIR<0> interrupt is enabled. This register, as with all registers, is reset to 0's via tc.~reset, disabling all 32 interrupts.

3.9 Receive DMA Word Counter (RWC)

<31:24> (R) These bits are reserved. Writes have no effect and zeros are returned on reads. These bits should loaded with zeros when writing the RWC.

<23:0> (R/W) These bits contain the word count for the current receive DMA operation. It is decremented with each word written to memory until it reaches zero, at which time the RWCEXPIRED bit in the OIR is set and the receive DMA operation stops. This register MUST be non-zero for Simple Receive DMA to occur.

3.10 Transmit DMA Word Counter (TWC)

<31:24> (R) These bits are reserved. Writes have no effect and zeros are returned on reads. These bits should loaded with zeros when writing the TWC.

<23:0> (R/W) These bits contain the word count for the current transmit DMA operation. It is decremented as each word is read from memory until it reaches zero, at which time the TWCEXPIRED bit in the OIR is set and the transmit DMA operation stops. Any words that are part of a DMA read transaction after the TWC reaches zero are discarded; they do not reach the transmit FIFO. This register MUST be non-zero for both Simple and Intelligent Transmit DMA to occur.

3.11 Option Interrupt Register (OIR)

Bits in this register are ANDed with the OIMR and then ORed together to form a single TURBOchannel interrupt request (tc.~int). A bit labeled R/W0TC (Read/Write 0 To Clear) is set on the transition of the specified condition and may be cleared by writing it with a zero bit; bits labeled R (Read-only) are the result of the specified condition and and can not be cleared in the OIR.

NOTE

Error bits MUST be cleared before a corresponding transmit or receive DMA operation will be initiated by the TcIA.

<31> (R/W0TC) TDMAERR
1: Error during a transmit DMA (TURBOchannel DMA read).
0: No error during transmit DMA.

<30> (R/W0TC) RDMAERR
1: Error during a receive DMA (TURBOchannel DMA write).
0: No error during receive DMA.

<29> (R/W0TC) TSGPLD
1: An entry from the scatter/gather table has been loaded into the transmit pointer upon reaching a page boundary.
0: The TSGP has not yet been used.

<28> (R/W0TC) TSGERR
1: The transmit pointer has reached a page boundary and the TSGP is 0x0.
0: The transmit pointer has not reached a page boundary while the TSGP is 0x0.

<27> (R/W0TC) RSGPLD
1: An entry from the scatter/gather table has been loaded into the receive pointer upon reaching a page boundary.
0: The RSGP has not yet been used.

<26> (R/W0TC) RSGERR
1: The receive pointer has reached a page boundary and the RSGP is 0x0.
0: The receive pointer has not reached a page boundary while the RSGP is 0x0.

<25> (R/W0TC) TWCEXPIRED
1: The transmit word counter has decremented to zero.
0: The transmit word counter has not decremented to zero.

<24> (R/W0TC) RWCEXPIRED
1: The receive word counter has decremented to zero.
0: The receive word counter has not decremented to zero.

<23> (R/W0TC after TWCEXPIRED is clear) TWCEXPTFE
1: Both TWCEXPIRED and TFE are set.
0: Either TWCEXPIRED or TFE is not set.

<22> (R/W0TC) PERR
1: A parity error was detected on the TURBOchannel.
0: No parity error has been detected on the TURBOchannel.

<21> (R/W0TC) IOPERR
1: A parity error was detected during a programmed I/O transaction.
0: If PERR is set, then a parity error was detected during a DMA transaction.

<20> (R/W0TC) WRPERR
1: A parity error was detected during an I/O write or a DMA write transaction.
0: If PERR is set, then a parity error was detected during an I/O read or DMA read.

<19> (R/W0TC) ADDRPERR
1: A parity error was detected during the address cycle of an I/O or DMA trans.
0: If PERR is set, then a parity error was detected during an I/O or DMA data cycle.

<18:13> (R) RESERVED

<12> (R) FLSHCNTEXP
1: The receive DMA flush counter has expired.
0: The receive DMA flush counter has not expired.

<11> (R) TFE
1: The transmit FIFO is empty.
0: The transmit FIFO is not empty.

<10> (R) TFNE
1: The transmit FIFO is not empty.
0: The transmit FIFO is empty.

<9> (R) RFE
1: The receive FIFO is empty.
0: The receive FIFO is not empty.

<8> (R) RFNE
1: The receive FIFO is not empty.
0: The receive FIFO is empty.

<7:6> (R) TUPACK2

These bits are a synchronized version of the transmit unpacking counter. They are reliably read only when the transmit FIFO is not being read.
00: Unpacking byte0, half-word0
01: Unpacking byte1
10: Unpacking byte2, half-word1
11: Unpacking byte3

<5:4> (R) RPACK2

These bits are a synchronized version of the option-data packing counter. They are reliably read only when the receive FIFO is not being written.
00: Packing byte0, half-word0
01: Packing byte1
10: Packing byte2, half-word1
11: Packing byte3

<3:0> (R) GPI

These General Purpose Input bits reflect the state of external signals op.~gpi<3:0>. If the op.~gpi signal is a 0 then the corresponding OIR bit will be a 1. When using any GPI bit as an interrupt source (with the corresponding OIMR bit set), the inputs must adhere to the same rule as the TURBOchannel ~int signal; that is, once asserted, the option may not deassert the signal until software dismisses the interrupt condition.

3.12 FIFO Access Register (FAR)

3.12.1 (R) Receive FIFO

Reads return the value of the next word to be written to memory from the receive FIFO. If the last non-word option data did not fill the byte packer (a receive DMA operation would have ended on a non-word boundary) and the FIFO is empty (RFE=1), then reads to this address will return the value of the incomplete word in the byte-packer. Reads to this register will not affect the state of the receive FIFO, including the head and tail pointers. This means that only the data word at the head of the FIFO or the byte-packer word will be visible through this register.

3.12.2 (W) Transmit FIFO

Writes to this register space will place data into the transmit FIFO. This is useful for writing unaligned data at the end of a transmit DMA operation. Placing data into the transmit FIFO in this way does affect the state the of the transmit FIFO by incrementing the tail pointer.

Blank Page

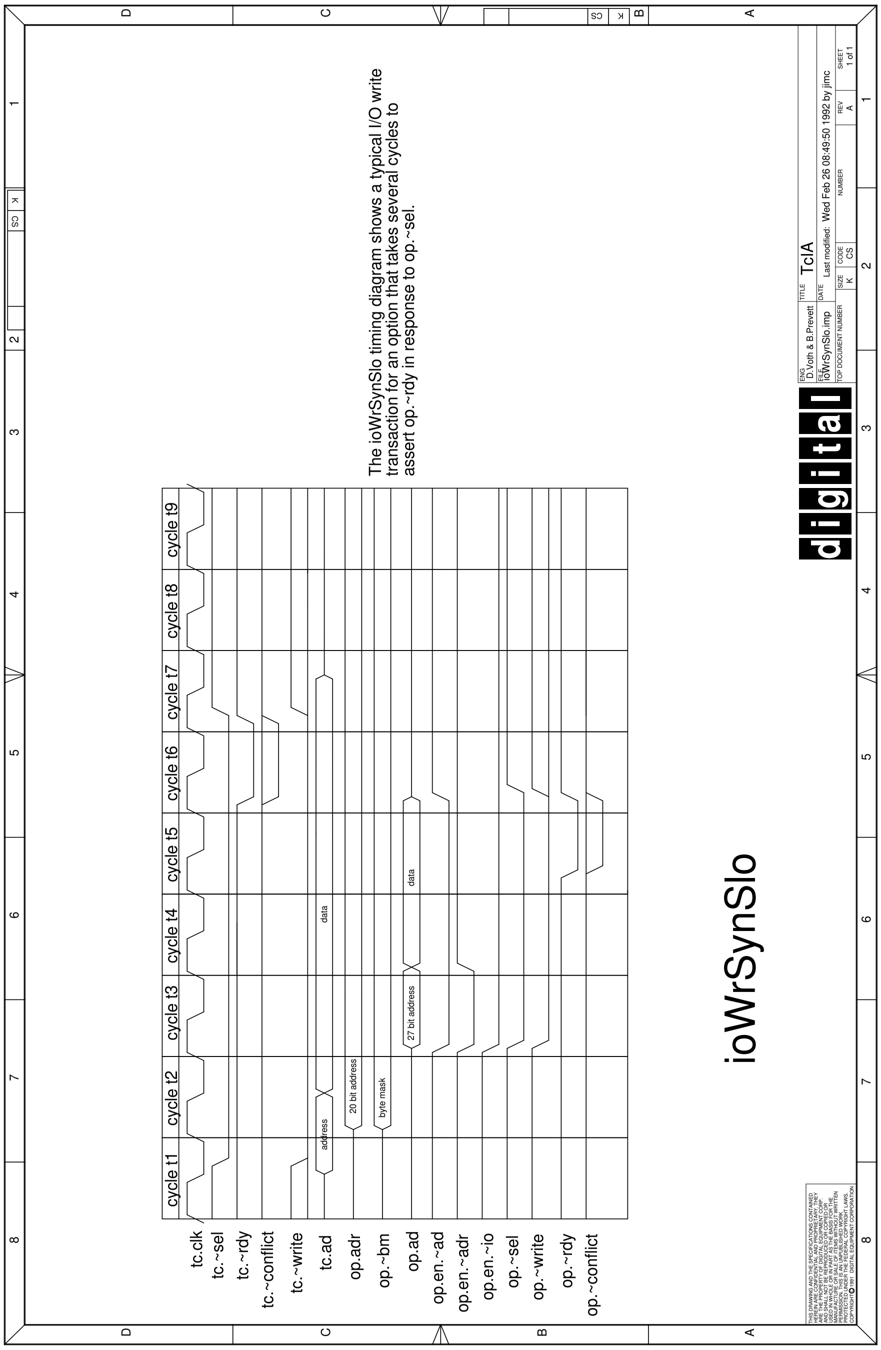
4 Programmed I/O

Many TURBOchannel devices are simple slaves. The TcIA provides an I/O subsystem that is simple and independent of the DMA subsystem. The option has control of the ~conflict signal and may issue op.~conflict with op.~rdy. The asynchronous mode requires a synchronous handshake between op.~rdy and op.~sel. This handshake is performed in the same manner as for the TURBOchannel.

The following sections step through I/O write and read operations, first with the option side driven by tc.clk and then with the option side driven by an independent op.clk. Each of these four combinations is described by a timing diagram and a cycle-by-cycle description of the diagram. When the option side signal op.synchronous is asserted, the option interface runs from the TURBOchannel clock; that is, the option interface is synchronous with the TURBOchannel interface. When op.synchronous is deasserted, the option interface runs off of an independent clock; that is, the option interface is not synchronous with the TURBOchannel interface.

4.1 I/O Write with op.synchronous = 1

Figure 2 I/O Write Timing Diagram with op.synchronous = 1



Cycle-by-cycle description of the **I/O Write Timing Diagram with op.synchronous = 1** (Figure 2):

Cycle t1:

- system asserts tc.~sel and tc.~write to initiate an I/O write transaction to the option.
- system drives I/O address onto tc.ad.

Cycle t2:

- TcIA loads address and byte mask into address/byte mask register on rising edge of tc.clk.
- TcIA drives 20 bit address onto op.adr and byte mask onto op.~bm.
- system drives I/O write data on tc.ad.

Cycle t3:

- TcIA asserts op.~sel and op.~write to signify an I/O write transaction to the option interface.
- TcIA drives full 27 bit address onto op.ad[28:2] if option asserts op.en.~adr, op.en.~io, and op.en.~ad; the full tc.ad data is driven onto op.ad if op.en.~adr is deasserted and both op.en.~ad and op.en.~io are asserted.
- TcIA loads I/O write data.

Cycle t4:

- TcIA drives data onto op.ad if option deasserts op.en.~adr, and asserts op.en.~io and op.en.~ad.

Cycle t5:

- option asserts op.~rdy in response to op.~sel. op.~conflict may be asserted with op.~rdy in this same cycle if the option is committed to a DMA transaction.

Cycle t6:

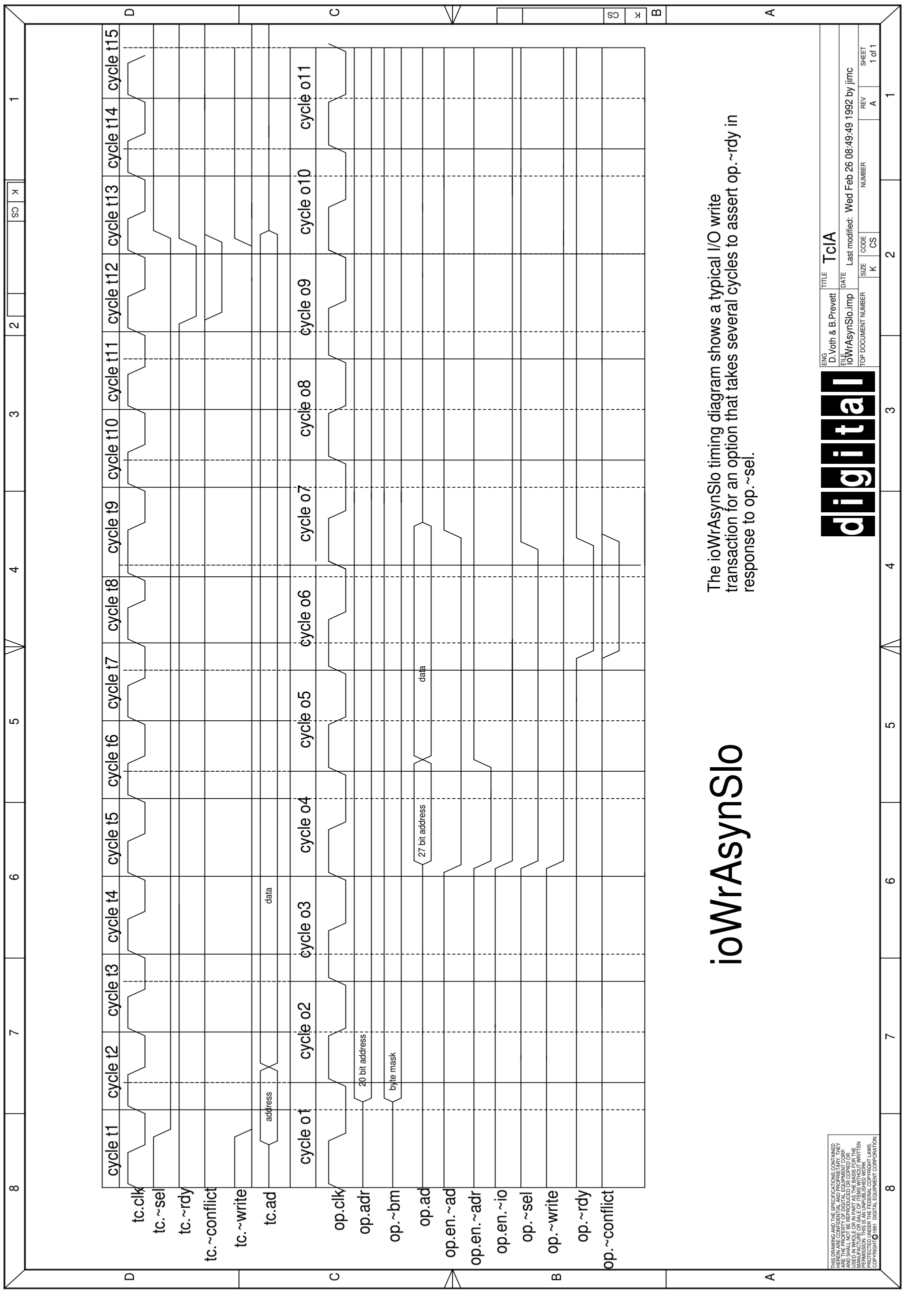
- option latches I/O write data on op.ad; option then deasserts op.en.~ad to tristate op.ad lines.
- TcIA asserts tc.~rdy; tc.~conflict is asserted with tc.~rdy if conflict had been asserted.
- option deasserts op.~rdy, and op.~conflict if applicable.
- TcIA deasserts op.~sel and may deassert op.~write.

Cycle t7:

- TcIA deasserts tc.~rdy, and tc.~conflict if applicable
- system deasserts tc.~sel in response to assertion of tc.~rdy; system may deassert tc.~write.
- system tristates tc.ad

4.2 I/O Write with op.synchronous = 0

Figure 3   I/O Write Timing Diagram with op.synchronous = 0



Cycle-by-cycle description of the **I/O Write Timing Diagram with op.synchronous = 0** (Figure 3):

Cycle t1:

- system asserts tc.~sel and tc.~write to initiate an I/O write transaction to the option.
- system drives I/O address onto tc.ad.

Cycle t2:

- TcIA loads address and byte mask into address/byte mask register on rising edge of tc.clk.
- TcIA loads tc.~sel and tc.~write on rising edge of tc.clk.
- TcIA drives 20 bit address onto op.adr and byte mask onto op.~bm.
- system drives I/O write data on tc.ad

Cycle t3:

- TcIA loads I/O write data.
- TcIA drives full 27 bit address onto op.ad[28:2] if option asserts op.en.~adr, op.en.~io, and op.en.~ad; the full tc.ad data is driven onto op.ad if op.en.~adr is deasserted and both op.en.~ad and op.en.~io are asserted.

Cycle o4:

- TcIA asserts op.~sel and op.~write to signify an I/O write transaction to the option interface.

- TcIA drives full 27 bit address onto op.ad if option asserts op.en.~adr, op.en.~io, and op.en.~ad; the full tc.ad data (shifted address) is driven onto op.ad if op.en.~adr is deasserted.

Cycle o5:

- TcIA drives data onto op.ad if option deasserts op.en.~adr, and asserts op.en.~io and op.en.~ad.

Cycle o6:

- option asserts op.~rdy in response to op.~sel. op.~conflict may be asserted with op.~rdy in this same cycle if the option is committed to a DMA transaction.

Cycle o7:

- option latches I/O write data on op.ad; option then deasserts op.en.~ad to tristate op.ad lines.
- option deasserts op.~rdy, and op.~conflict if applicable.
- TcIA deasserts op.~sel and may deassert op.~write

Cycle t12:

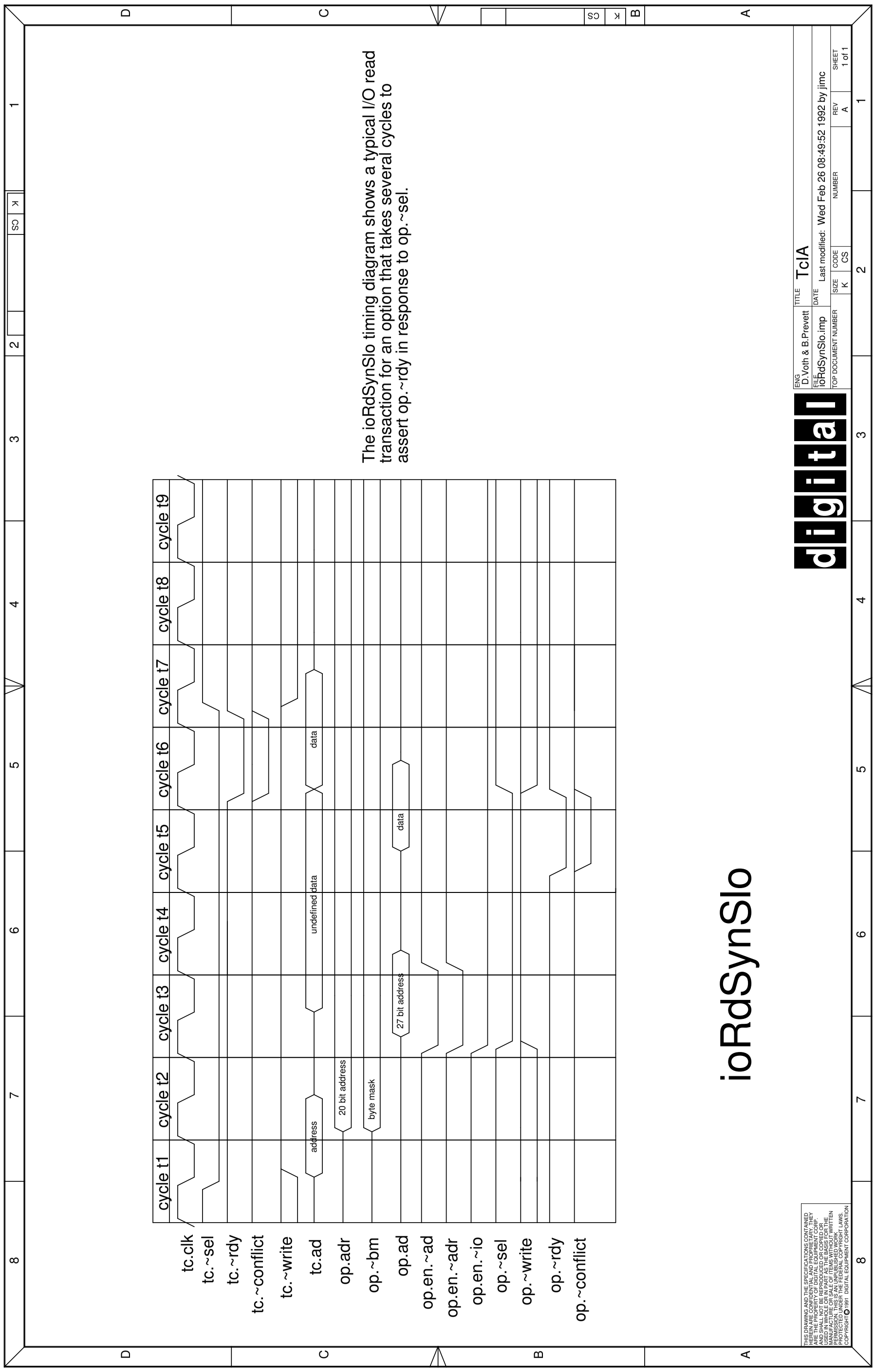
- TcIA asserts tc.~rdy; tc.~conflict is asserted with tc.~rdy if conflict had been asserted.

Cycle t13:

- system deasserts tc.~sel in response to assertion of tc.~rdy; system may deassert tc.~write.
- TcIA deasserts tc.~rdy, and tc.~conflict if applicable
- system tristates tc.ad

4.3 I/O Read with op.synchronous = 1

Figure 4 I/O Read Timing Diagram with op.synchronous = 1



Cycle-by-cycle description of the **I/O Read Timing Diagram with op.synchronous = 1** (Figure 4):

Cycle t1:

- system asserts tc.~sel and deasserts tc.~write to initiate an I/O read transaction to the option.
- system drives I/O address onto tc.ad.

Cycle t2:

- TcIA loads address and byte mask into address/byte mask register on rising edge of tc.clk.
- TcIA drives 20 bit address onto op.adr and byte mask onto op.~bm.
- system tristates tc.ad.

Cycle t3:

- TcIA asserts op.~sel and and deasserts op.~write to signify an I/O read to the option interface.
- TcIA drives full 27 bit address onto op.ad[28:2] if option asserts op.en.~adr, op.en.~io, and op.en.~ad; the full tc.ad data is driven onto op.ad if op.en.~adr is deasserted and both op.en.~ad and op.en.~io are asserted.
- TcIA now drives tc.ad with undefined data during option latency period.

Cycle t4:

- option deasserts op.en.~ad and op.en.~adr; op.ad is now tristated to accept the driven option data. op.ad should not float for more than one tc.clk cycle; option must either drive op.ad or use op.en.~ad to allow the TcIA to drive op.ad during this latency period.

Cycle t5:

- option asserts op.~rdy in response to op.~sel. op.~conflict may be asserted with op.~rdy in this same cycle if the option is committed to a DMA transaction.
- option drives op.ad with valid read data

Cycle t6:

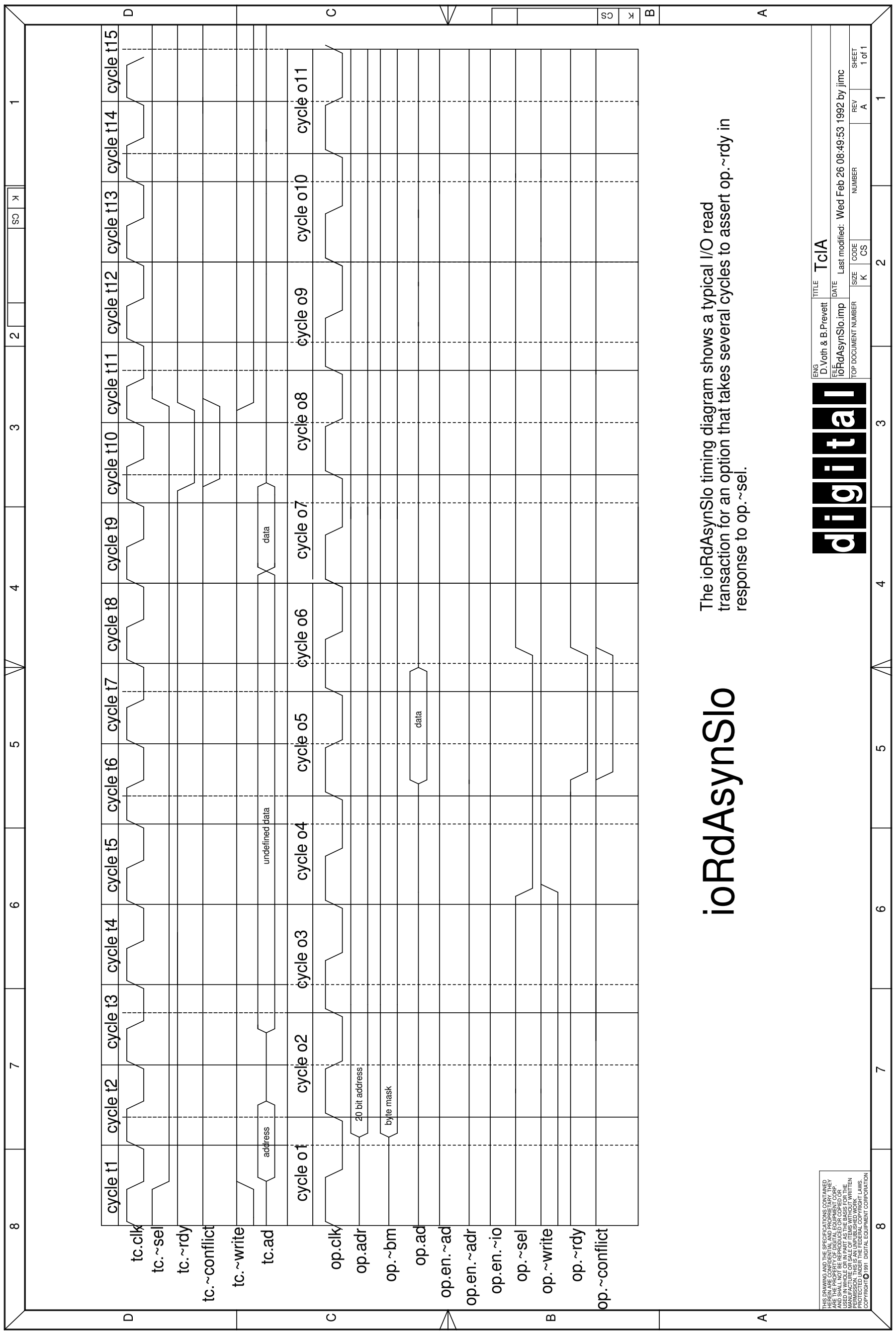
- TcIA drives tc.ad with option data.
- TcIA asserts tc.~rdy; tc.~conflict is asserted with tc.~rdy if op.~conflict had been asserted.
- option deasserts op.~rdy, and op.~conflict if applicable.
- TcIA deasserts op.~sel.
- data on op.ad becomes invalid; op.ad must not be left floating. Option may assert op.en.~ad to allow TcIA to drive op.ad or the option may drive op.ad directly.

Cycle t7:

- TcIA deasserts tc.~rdy, and tc.~conflict if applicable
- system deasserts tc.~sel in response to assertion of tc.~rdy.
- TcIA tristates tc.ad

4.4 I/O Read with op.synchronous = 0

Figure 5 I/O Read Timing Diagram with op.synchronous = 0



Cycle-by-cycle description of the **I/O Read Timing Diagram with op.synchronous = 0** (Figure 5):

Cycle t1:

- system asserts tc.~sel and deasserts tc.~write to initiate an I/O read transaction to the option
- system drives I/O address onto tc.ad

Cycle t2:

- TcIA loads address and byte mask into address/byte mask register on rising edge of tc.clk.
- TcIA loads tc.~sel on rising edge of tc.clk.
- TcIA drives 20 bit address onto op.adr and byte mask onto op.~bm
- system tristates tc.ad

Cycle t3:

- TcIA now drives tc.ad with undefined data during option latency period.

Cycle o4:

- TcIA asserts op.~sel and deasserts op.~write to signify an I/O read transaction to the option interface.
- op.en.~ad and op.en.~adr are deasserted; op.ad is now tristated to accept the driven option data. op.ad should not float for more than one op.clk cycle; option must either op.ad or use op.en.~ad to allow the TcIA to drive op.ad during this latency period.

Cycle o5:

- option asserts op.~rdy in response to op.~sel. op.~conflict may be asserted with op.~rdy in this same cycle if the option is committed to a DMA transaction.
- option drives op.ad with valid read data

Cycle o6:

- data on op.ad becomes invalid; op.ad must not be left floating. Option may assert op.en.~ad to allow TcIA to drive op.ad or the option may drives op.ad directly.

- option deasserts op.~rdy, and op.~conflict if applicable.

- TcIA deasserts op.~sel.

Cycle t10:

- TcIA drives tc.ad with option data.

- TcIA asserts tc.~rdy; tc.~conflict is asserted with tc.~rdy if conflict had been asserted.

Cycle t11:

- TcIA deasserts tc.~rdy, and tc.~conflict if applicable
- system deasserts tc.~sel in response to assertion of tc.~rdy.
- TcIA tristates tc.ad

Blank Page

5 DMA

The TcIA DMA engine transfers transmit data from or receive data to virtually contiguous buffer space in main memory. There are two ways to set up the TcIA to perform DMA operations: simple DMA mode and intelligent controller DMA mode.

Simple DMA devices do not generate their own addresses. They move data to or from locations in memory via the pointers stored in the TcIA. Typically, a simple DMA operation is started and completed by the processor. Interrupts can be used to notify the processor that the DMA operation has completed or that errors have occurred. For multipage DMA operations, the TcIA can utilize a scatter/gather table that is set up in system memory by the controlling software. The table is a list of pointers to the first word of subsequent DMA pages.

Intelligent controllers generate their own addresses and pass them into the receive FIFO. DMA receive operations are initiated by writing the DMA receive address and DMA write data words into the receive FIFO; data words are then written to system memory at the loaded address. DMA transmit operations are preceded by writing a transmit address into the receive FIFO and are initiated by writing a transmit burst size and word count into the receive FIFO; TWC words of data are moved from system memory into the transmit FIFO in as many DMA read transactions of size BURSTSIZE as required.

5.1 Simple Transmit DMA

The TcIA begins a transmit DMA operation by moving BURSTINIT words from memory into the transmit FIFO. After this initial transaction, if BURSTSIZE is less than 15, then BURSTSIZE words will be moved into the transmit FIFO when at least BURSTSIZE words are free in the FIFO. If BURSTSIZE is 15 or greater, then the TcIA will NOT check the number of words in the FIFO before requesting the next DMA read transaction; in this case, the option MUST remove data from the FIFO at least as fast as it is loaded into the FIFO. BURSTSIZE should be chosen to be an integer power of 2 (2^N); larger values of BURSTSIZE give higher TURBOchannel bandwidth. BURSTINIT should be chosen to align all following DMA transactions to "burst boundaries", this is, where the N-1 low order bits of the TDP are zero.

The option removes option-data from the FIFO. The TcIA continues to fill the FIFO each time a DMA read transaction will fit. When all of the words specified by the TWC have been moved, the TcIA stops loading data into the FIFO, even if in the middle of a DMA read transaction (the rest of the words in the burst are ignored).

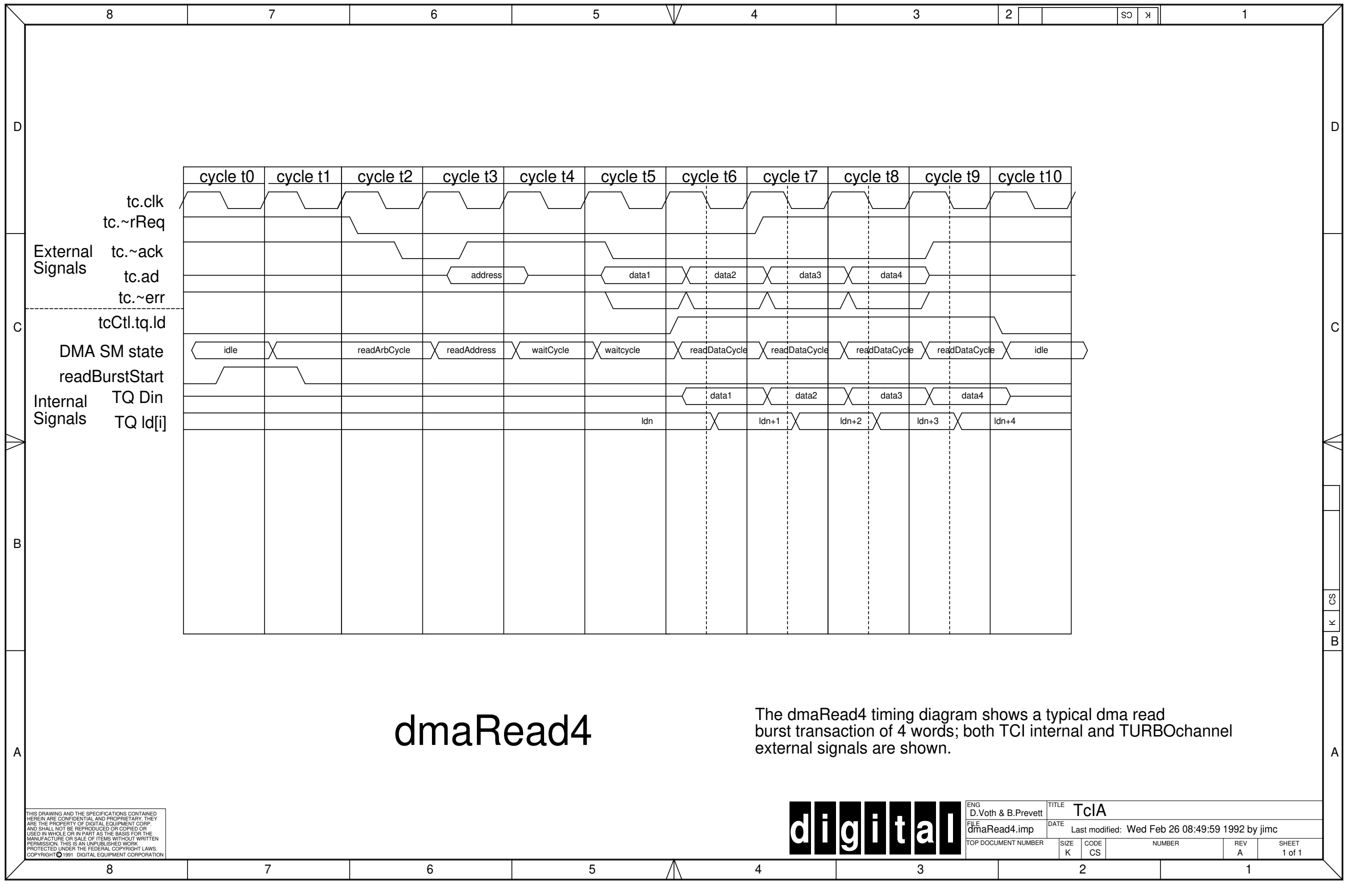
The transmit FIFO is initialized by setting TFEN = 0. The alignment of the first option-word to be unpacked is set by writing TUPACK. TFEN is then set to a 1. The BURSTINIT field is set to BURSTSIZE - (TDP modulo BURSTSIZE). The option-data SIZE field in OCR1 may be byte, half-word or word. Words are unpacked into option-data which are available at the option interface. The transmit word counter is then loaded with the number of words to be transferred. When the counter reaches zero, TWCEXPIRED is set and transmit DMA is disabled. The transmit word counter value may be from 1 word to 16MWords and must end on a word boundary. For DMA operations that cross page or 2Kbyte boundaries, the BURSTSIZE should be set to a power of 2 to help ensure that the page or 2Kbyte boundary will not be crossed by a DMA transaction. If transmit DMA ends on a non word-aligned boundary, the unaligned bytes must be written to the transmit FIFO along with a TUPACK value. The interrupts are cleared and any interrupts that need service are enabled via the Mask Register.

When a DMA operation reaches a page boundary, as defined by the PAGESIZE, the TcIA will read (via a DMA read transaction) the next pointer from the transmit scatter/gather table into the TDP. This pointer should be the starting word address of the next physical page for the operation. After loading the pointer, the TcIA increments the TSGP. The PAGESIZE must be greater than the BURSTSIZE for the scatter/gather mechanism to function correctly.

The transmit FIFO is filled with data as long as there is a non-zero TWC. The DMA is then enabled and transmission begins. Note that DMA from memory will not progress unless TD-MAEN = 1, TFEN =1, TDMAERR = 0, TPLD = 0, and TWCEXPIRED = 0.

The **DMA Read Timing Diagram** (Figure 6) shows the TcIA reading four words from system memory and placing them into the transmit FIFO. The **Transmit DMA Timing Diagram** (Figure 7) shows a simple DMA option reading these four words from the transmit FIFO. When op.en.~ad is asserted TcIA drives op.ad. Asserting op.~trDack reads the next entry from the FIFO in next cycle. When the FIFO is empty, op.~trValid is deasserted.

Figure 6 DMA Read Timing Diagram



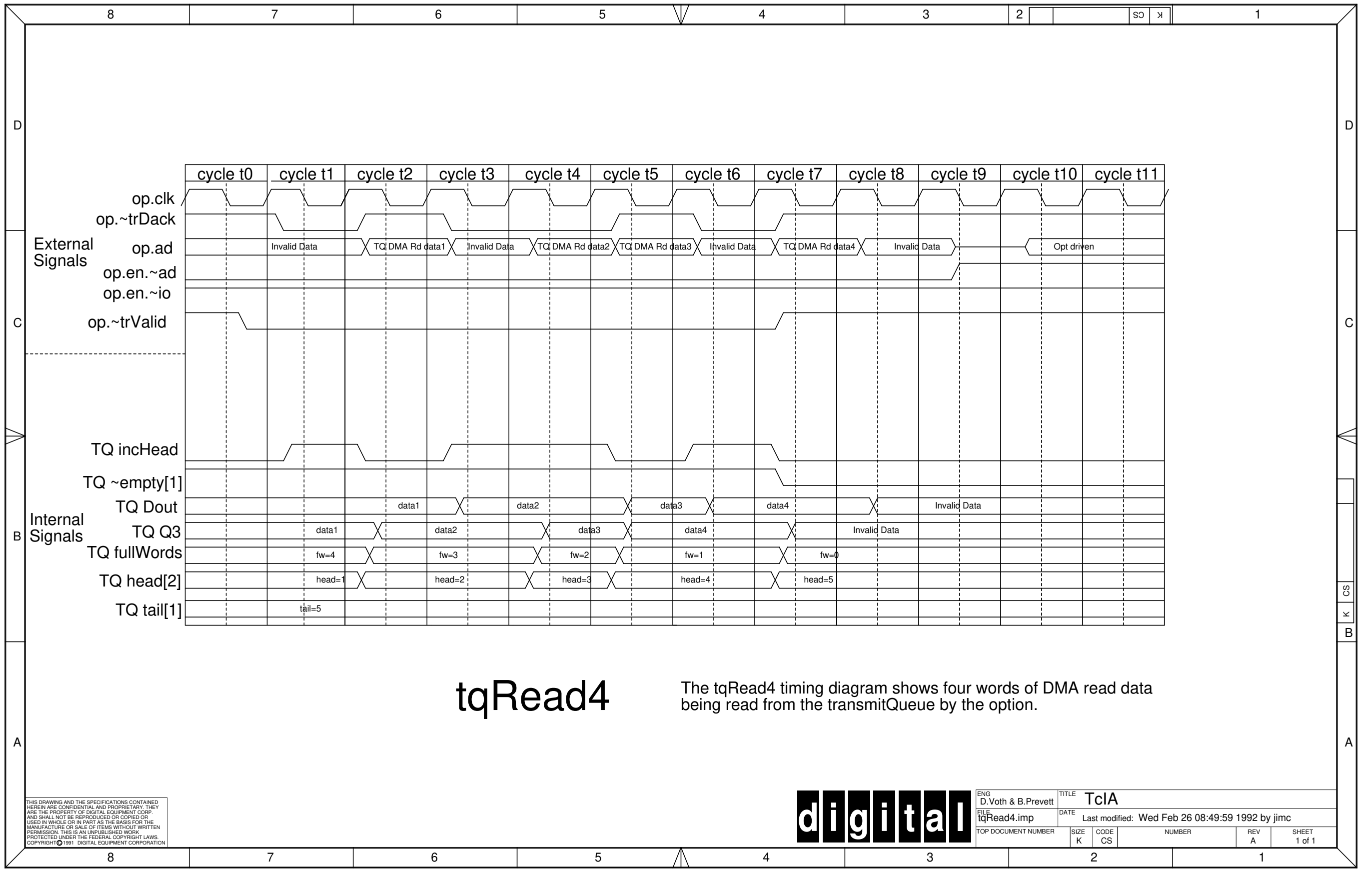


Figure 7 Transmit DMA Timing Diagram

5.2 Simple Receive DMA

The TcIA packs option-data into receive FIFO words and writes those words into memory. DMA write transactions will continue as long as there are RCVBURST words of valid data in the receive FIFO. The DMA engine breaks the write transactions at (word address modulo 64) = 0 to prevent page crossing or transfer length errors. The option-data SIZE field in OCR1 may be byte, half-word or word.

The receive FIFO is initialized by setting RFEN = 0. The alignment of the first option-word to be packed is set by writing RPACK. RFEN is then set to a 1.

The receive DMA pointer may be loaded with any physical word address. The TcIA is only capable of writing words. If the last option-data items are not word aligned, they will remain in the packer at the end of the operation and not be written to memory. These can be read by programmed I/O and moved into memory at the end of the operation. The receive FIFO has a counter (loaded from RCVFLUSHCNT) that determines when to flush the FIFO by writing the data words to system memory. This is useful at the end of DMA operations when the number of data words left in the receive FIFO is less than the RCVBURST size.

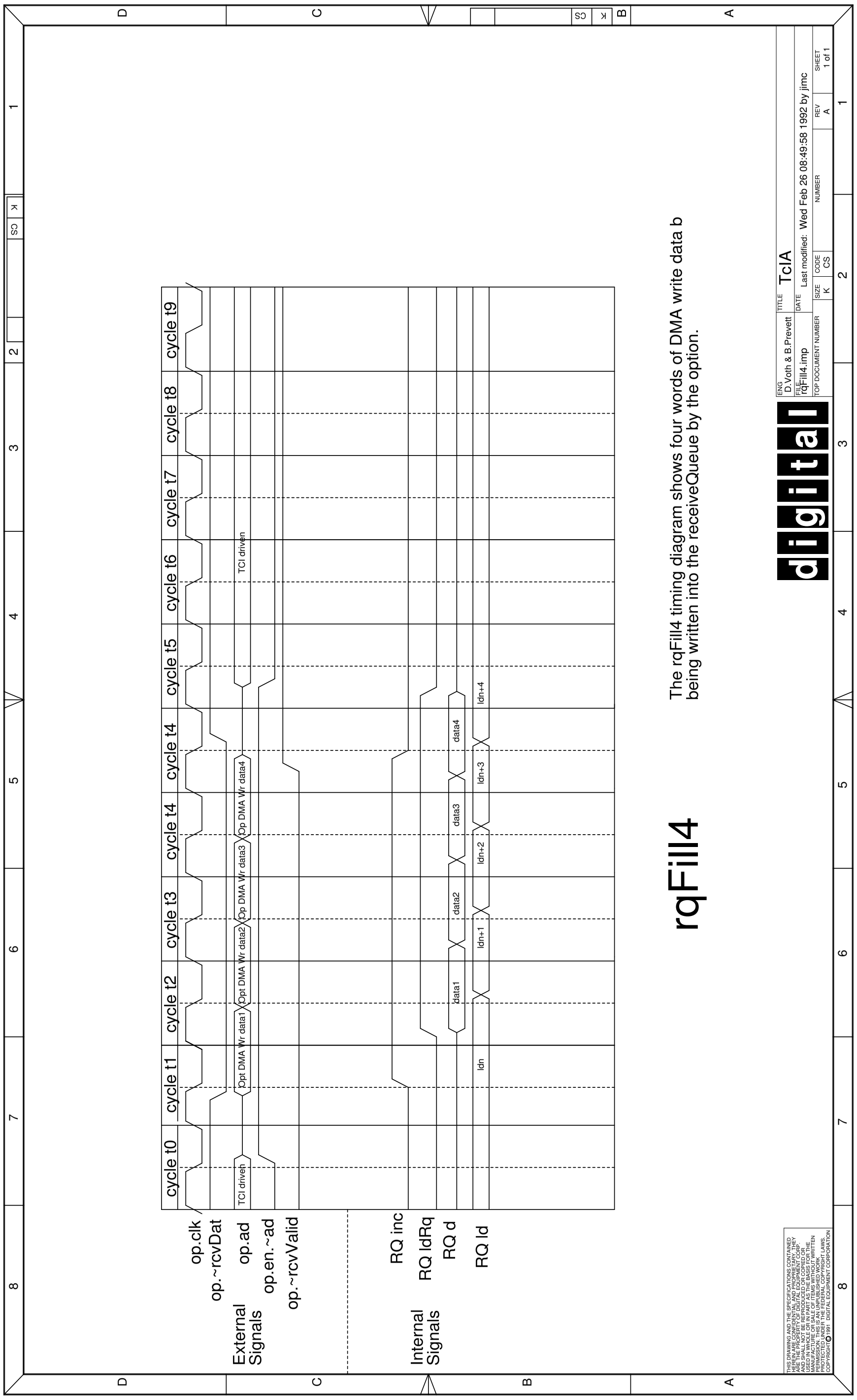
The receive word counter is then loaded with the number of complete words to be transferred. Simple receive DMA operations require the use of the RWC, while intelligent mode DMA does not use the RWC. The DMA operation to memory will stop when the counter reaches zero; RWCEXPIRED is then set. As in transmit DMA, the addresses of subsequent pages will be loaded from the scatter/gather table on page crossings. The PAGESIZE must be greater than the BURSTSIZE and greater than 64 words for the scatter/gather mechanism to work correctly.

Receive DMA to memory will not progress unless RDMAEN = 1, RFEN = 1, RDMAERR = 0 and RWCEXPIRED = 0.

The **Receive DMA Timing Diagram** (Figure 8) shows a simple DMA device loading four words of DMA write data into the receive FIFO. The assertion of op.en.~ad tristates the op.ad drivers so that each option-word can be placed onto op.ad without contention. The signal op.~rcvDat places the option-word into receive FIFO. The deassertion of op.~rcvValid indicates that there are less than RCVTHRESHOLD free option-words remaining in the FIFO.

After synchronization the receive DMA state-machine will read the option-words out of the FIFO and issue a DMA write transaction to system memory. The **DMA Write Timing Diagram** (Figure 9) shows a four word DMA write into system memory.

Figure 8 Receive DMA Timing Diagram



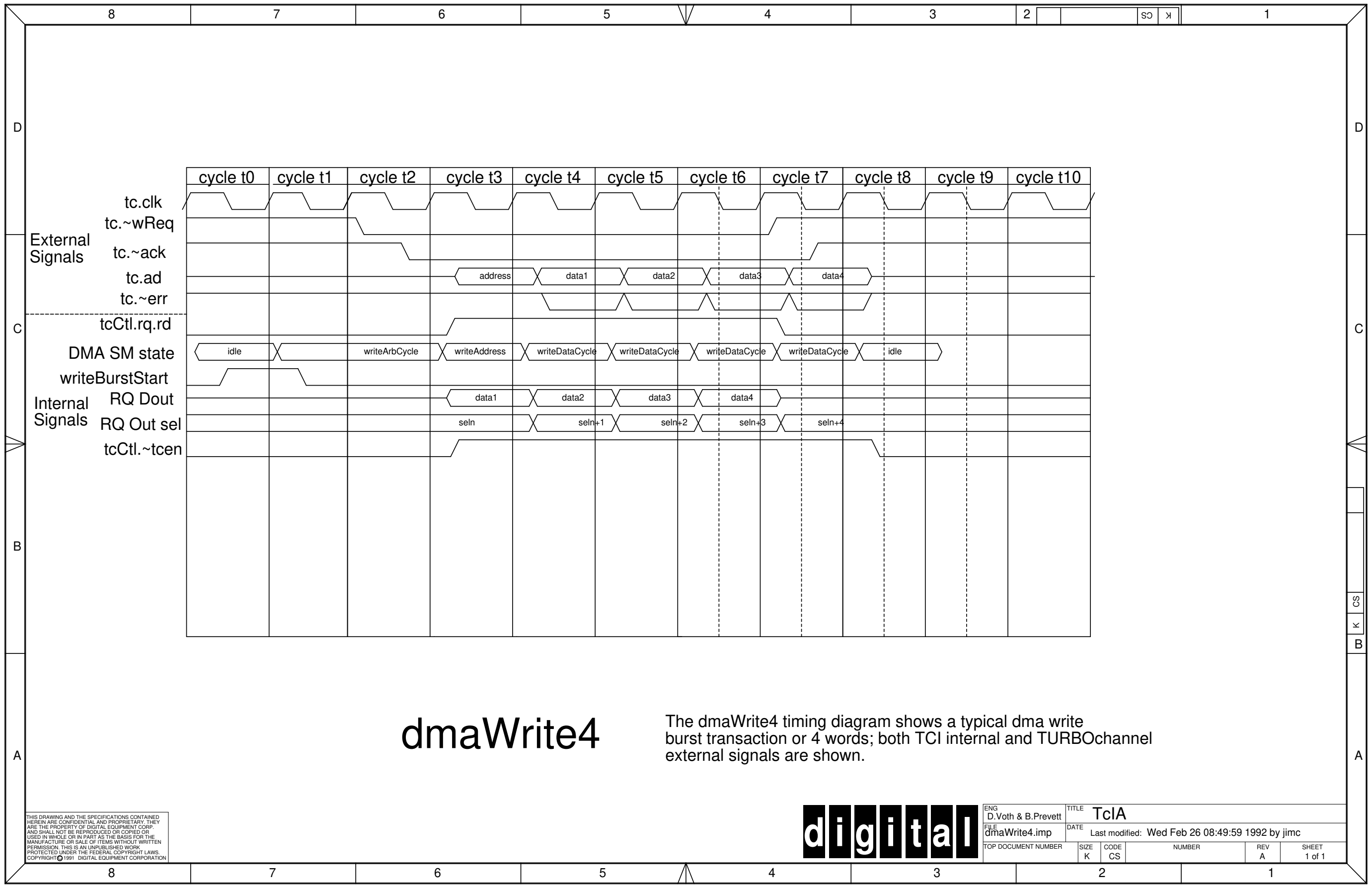


Figure 9 DMA Write Timing Diagram

5.3 DMA Interrupt Handling

When an interrupt occurs, the controlling software examines the OIR and decides on a course of action. It is usually a good idea to first turn off the DMA via TDMAEN or RDMAEN before processing the interrupt so that the pointers will be stable during interrupt processing.

After the interrupt is handled, if the DMA operation is to be restarted the controlling software should first clear the interrupt(s) and then enable the DMA operation (TDMAEN, RDMAEN).

5.3.1 Errors

If an error occurs during a DMA operation, an interrupt is set and the DMA operation causing the error is halted. The DMA pointer will contain the address associated with the error.

5.4 Simple DMA Examples

5.4.1 Simple Transmit DMA Examples

Consider a TURBOchannel HDTV frame buffer that uses the TcIA and the simple DMA transfer mechanism. For this example, 30Kbyte or 7.5KWords of image data must be transferred to the frame buffer. The data is located in virtually contiguous system memory address space beginning at virtual address vAddr(0). The memory system page size for this system is 4Kbyte. Since this operation will cross page boundaries, the scatter/gather function will be used.

To set up the DMA operation, the controlling software must first compute the physical address for the start address, pAddr(0), and then compute the physical page addresses of the subsequent pages, pAddr(n). pAddr(0) is located 3Kbyte into a page. The required physical page computations are shown below:

Start Address	vAddr(0)	==>	pAddr(0)
Page Boundary #1	vAddr(0) + 1Kbyte	==>	pAddr(1)
Page Boundary #2	vAddr(0) + 1Kbyte + 1*4Kbyte	==>	pAddr(2)
Page Boundary #3	vAddr(0) + 1Kbyte + 2*4Kbyte	==>	pAddr(3)
Page Boundary #4	vAddr(0) + 1Kbyte + 3*4Kbyte	==>	pAddr(4)
Page Boundary #5	vAddr(0) + 1Kbyte + 4*4Kbyte	==>	pAddr(5)
Page Boundary #6	vAddr(0) + 1Kbyte + 5*4Kbyte	==>	pAddr(6)
Page Boundary #7	vAddr(0) + 1Kbyte + 6*4Kbyte	==>	pAddr(7)
Page Boundary #8	vAddr(0) + 1Kbyte + 7*4Kbyte	==>	pAddr(8)

The controlling software then loads pAddr(1) through pAddr(8) into a scatter/gather table that is allocated in system memory. The table created is an eight entry table of physically contiguous words pAddr(1) through pAddr(8) located at physical address pSGAddr.

The controlling software is now ready to initiate the required DMA operation. Through programmed I/O to the TcIA register file, the controlling software:

* sets the Transmit DMA Pointer (TDP) = pAddr(0)
* sets the Transmit DMA Scatter/Gather Pointer (TSGP) = pSGAddr

* sets the Transmit DMA Word Counter (TWC) = 30Kbyte/4 = 7.5KWords
* configures the Option Interrupt Mask Register (OIMR) to interrupt when TWC reaches 0 or when an error occurs.
* configures Option Control Register 2 (OCR2) by setting PAGESIZE = 4Kbyte
* configures Option Control Register 1 (OCR1) by :
    * setting the TFEN and TDMAEN bits
    * setting SIZE=word, BURSTSIZE=8
    * setting BURSTINIT = BURSTSIZE - (TDP modulo BURSTSIZE)

The TcIA is now programmed for the DMA operation. Once there are BURSTSIZE free words in the transmit FIFO, the TcIA will initiate a BURSTINIT word DMA read transaction from address pAddr(0). Subsequent DMA read transactions will be of size BURSTSIZE. As each word is received by the TcIA, it is loaded into the transmit FIFO, the TDP is incremented and the TWC is decremented. As the transmit FIFO fills up, the option side signal op.~trValid asserts, signifying valid data in the transmit FIFO. The option then reads the data from the transmit FIFO by asserting the op.~trDack signal. The TcIA will issue DMA read transactions as long as there are BURSTSIZE free words in the transmit FIFO.

When the TDP reaches a page boundary, the TcIA must fetch the next page address pAddr(1) from the scatter/gather table. The TcIA issues a one-word DMA read to the address pSGAddr, loads the data, pAddr(1), into the TDP, and increments the TSGP to point to the next entry in the scatter/gather table. DMA now continues from address pAddr(1). In this way, the pages beginning at pAddr(2) through pAddr(7) are transferred to the transmit FIFO and then to the option interface.

When pAddr(8) is loaded into the TDP, the DMA operation continues until the word count reaches zero. If TWC modulo BURSTSIZE is not equal to zero, at this end condition, a BURSTSIZE unit of data is still DMA read from system memory; however only the data words that correspond to the non-zero word count will be written into the receive FIFO.

The transmit DMA operation is now complete. The signal tc.~int will assert, signifying that the TWC has decremented to zero and that the DMA operation has been completed. The controlling software clears the OCR1 TDMAEN bit to disable DMA transactions and then reads the Option Interrupt Register (OIR) to determine that the DMA operation has indeed completed successfully.

To explore data alignment issues, consider a second simple transmit DMA example: a UART that DMA reads characters from system memory and transmits them out a serial line. This UART is byte oriented, so the OCR1 SIZE field is set to BYTE. For this example operation, there are 97 character bytes to be transmitted; the byte ordering scheme is little-endian. The TcIA transmit DMA engine allows for non-word alignment for the data bytes of the first word. If the data block to be transferred does not end on a word boundary, then the remainder bytes must be programmed I/O written into the transmit FIFO along with the required TUPACK value.

The character data begins at system memory location pAddr(0) + 1, where pAddr(0) is a word aligned address. The character bytes are located in the system memory buffer as follows:

byte(1) ==> pAddr(0) + 1
byte(2) ==> pAddr(0) + 2

byte(3) ==> pAddr(0) + 3
byte(4) ==> pAddr(0) + 1*4 + 0
byte(5) ==> pAddr(0) + 1*4 + 1
byte(6) ==> pAddr(0) + 1*4 + 2
.
.
.
byte(94) ==> pAddr(0) + 23*4 + 2
byte(95) ==> pAddr(0) + 23*4 + 3
byte(96) ==> pAddr(0) + 24*4 + 0
byte(97) ==> pAddr(0) + 24*4 + 1

The controlling software initiates the required transmit DMA operation by:

* setting the TDP = pAddr(0)
* setting the TWC = (first 95 data bytes + 1 undefined byte at pAddr(0) + 0) / 4 = 24
* configuring the OIMR to interrupt when the TWC reaches 0
* configuring Option Control Register 2 (OCR2) by setting BIGENDIAN to 0
* configuring Option Control Register 1 (OCR1) first by:
    * setting TFEN = 0 to clear the transmit FIFO
    * setting TUPACK = 01.

and then by:
    * setting the TDMAEN bit
    * setting SIZE = byte, BURSTSIZE = 4
    * setting BURSTINIT = BURSTSIZE - (TDP modulo BURSTSIZE).

The TcIA is now programmed for the DMA operation. Once there are BURSTSIZE free words in the transmit FIFO, the TcIA will initiate a four word DMA read transaction from pAddr(0). The first word loaded into the transmit FIFO will be configured as follows:

byte(3) | byte(2) | byte(1) | undefined byte from pAddr(0) + 0

The next 23 data words are then written into the transmit FIFO as space allows. When op.~trValid first asserts, the first bytes will be read out of the FIFO with the op.~trValid signal. The first bytes to be read out will be: byte(1), byte(2), byte(3), byte(4), etc. The undefined data will not be read out because the unpacking counter was loaded to ignore it.

After the first 95 characters have been transferred into the transmit FIFO, the TWC will reach 0 and tc.~int will assert. Due to the two data bytes left over, the controlling software must now resolve the end condition. The controlling software checks the TFE bit to determine when the transmit FIFO is empty. Due to the interrupt latency, the transmit FIFO will most likely be empty (TFE=0) by the time the TFE bit is checked. It then loads the byte-unpacker with TU-PACK = 10 and sets TFEN = 0. The controlling software then programmed I/O writes the last two bytes into the transmit FIFO. These last two bytes will be read out of the transmit FIFO by the option.

If necessary, the controlling software can confirm that all characters have indeed been read out of the transmit FIFO to UART by polling the OIR TFE (transmit FIFO empty) bit.

An alternative approach for non-word option devices is to align the bytes or half-words to word boundaries by setting the option-data size to word and then copying the byte or half-word data to the low byte or low half-word of each addressed word. In this way three or two bytes per system memory word will be wasted, but there will be no byte misalignment cases to handle.

5.4.2 Simple Receive DMA Examples

For this example, consider a TURBOchannel disk controller that uses the TcIA in a simple DMA transfer mechanism. The controller must transfer 6Kbyte of data from the option on-board buffer to system memory beginning at location vAddr(0). This operation will cross one page boundary; therefore, the scatter/gather function will be used.

As in the simple transmit DMA example, the controlling software computes the beginning physical address pAddr(0) and the page address pAddr(1). pAddr(1) is loaded into a scatter/gather table that begins at physical address pSGAddr.

The controlling software is now ready to initiate the DMA operation; through programmed I/O to the TcIA register file, the controlling software:

* sets the Receive DMA Pointer (RDP) = pAddr(0)
* sets the Receive DMA Scatter/Gather Pointer (RGSP) = pSGAddr
* sets the Receive DMA Word Counter (RWC) = 6Kbyte/4 = 1.5KW
* configures the Option Interrupt Mask Register (OIMR) to interrupt when the RWC reaches 0 or when an error occurs.
* configures Option Control Register 2 (OCR2) by:
    * setting RCVTHRESHOLD = 001
    * setting RCVBURST = 10 (4 valid words)
    * setting RCVFLSHCNT = 00 (16 TURBOchannel cycles)
* configures Option Control Register 1 (OCR1) by:
    * setting the RFEN and RDMAEN bits
    * setting SIZE = word
    * setting the GPO[0] bit.

The TcIA is now programmed for the DMA operation. The signal op.gpo[0] asserts, signaling to the disk controller to begin the receive DMA operation. The disk controller writes data words into the receive FIFO by depositing data onto op.ad and asserting the op.~rcvDat signal. If the FIFO fills up past the RCVTHRESHOLD limit, the op.~rcvValid signal deasserts. In this example, op.~rcvValid asserts when there are less than two free words in the receive FIFO. Option data loading into the receive FIFO delays until the op.~rcvValid reasserts, signaling free words in the receive FIFO.

Unlike transmit DMA, receive DMA does not use the BURSTSIZE parameter. Instead, the TcIA asserts tc.~wReq and begins a DMA write transaction as soon as there are RCVBURST words of valid data in the receive FIFO. When the receive FIFO is emptied, the DMA write transactions halt; a new DMA write transaction begins when RCVBURST words of new valid data are written into the receive FIFO by the option. With this scheme, the resultant DMA write transactions can be from RCVBURST words to 64 words long; the burst length depends upon the rate at which the option can load the receive FIFO and the rate at which the TURBOchannel can empty the receive FIFO. The DMA write transactions are broken at (word address modulo 64) = 0 to prevent page crossings, 2K boundary crossings, and transfer length errors.

When the RDP reaches the page boundary, the TcIA fetches the next page address pAddr(1) from the scatter/gather table. The RDP is loaded with pAddr(1) and receive DMA continues until the RWC reaches 0 or is less than RCVBURST. If the RWC has decremented to a non-zero value that is less than the RCVBURST size, data in the receive FIFO is flushed through the flush counter until RWC = 0. The signal tc.~int will assert, signaling the completion of the DMA operation.

To examine data alignment issues for receive DMA, again consider the UART example. The UART has now received 95 byte characters from the serial line and must write them into a system memory buffer. The receive DMA engine allows for non-word alignment for the data-bytes of the first word, and the data block can end on a non-word boundary.

For this example receive DMA, the RWC will not be used to control the amount of data transferred, but it must be set to a large enough value that it will not go to zero during the transfer or the transfer will stop. The controlling software has loaded the RDP with a system memory buffer location, pAddr(0), or the RDP may be left over from a previous receive DMA operation and would point to the next available word in the system memory buffer. To enable the operation, the controlling software:

* sets the OIMR to interrupt when op.~gpi[2] is asserted
* sets the RWC to a "large enough" value that it will not go to zero
* configures Option Control Register 1 (OCR1) first by:
    * resetting RFEN = 0 to clear the receive FIFO and packer
    * setting the RPACK = 11

  and then by:
    * setting the RDMAEN bit
    * setting SIZE = byte.

The TcIA is now programmed to transfer the UART characters into system memory. The UART loads the character bytes into the receive FIFO. The first word of the first DMA write transaction contains valid data, byte(1), in the third byte of the word, since RPACK was set to three. The next 92 characters are transferred to system memory via DMA write transactions. The last two characters, byte(94) and byte(95) remain in the byte packing register of the receive FIFO. The UART asserts op.~gpi[2] to signal to the controlling software that a new set of serial line characters has been written into the system memory buffer. In response to the interrupt, the controlling software examines the OIR and determines that the receive FIFO is empty (RFE = 1) but that characters are left over in the byte packer (RPACK2 = 10). The controlling software then uses an I/O read transaction to read the last two bytes and store them into system memory.

5.5 Intelligent Controller DMA

Unlike simple DMA options, intelligent controllers generate their own addresses and execute reads from and writes to memory. The TcIA supports intelligent controller DMA through the DPM register (which allows DMA addresses of less than 32 bits to be extended to a full 32-bit physical address) and the data packing and unpacking functions, but the controller is still responsible for its own management of physical addresses and DMA burst lengths (in other words, it must ensure that DMA read transactions do not break the burst rules).

For transmit DMA, the intelligent controller first loads a DMA transmit address into the receive FIFO by asserting op.~trAdr without op.~rcvAdr while placing the address onto op.ad. If TWC equals zero when the transmit address reaches the head of the FIFO, the address will be merged into the TDP under the control of the DMP; if the TWC is not zero, the address will wait at the head of the receive FIFO until TWC becomes zero. The intelligent controller then loads the burst size/word count word into the receive FIFO by asserting op.~rcvAdr and op.~trAdr while placing the burst size onto op.ad[31:24] and the word count onto op.ad[23:0]. If TWC equals zero when the burst size/word count word reaches the head of the FIFO, the word will be written into the BURSTSIZE (and BURSTINIT) fields and TWC; if the TWC is not zero, the word will wait at the head of the receive FIFO until TWC becomes zero. Once the TWC becomes non-zero, the transmit DMA operation begins and will continue until the TWC again becomes 0. This design has one positive and one negative result: on the positive side, the intelligent option can load multiple pairs of pointer and burst size/word count words into the FIFO and they will be removed as-needed; on the negative side, when these transmit control words are waiting in the FIFO, receive data will be stuck in the FIFO. The burst size is not restricted to a power of two; however, the address used for the DMA read must not generate transactions that cross a 2Kbyte page boundary or are greater than the system-specific burst limit. For this reason it is often simplest to use a power of two burst size and make (TDP modulo burst size) = 0. When in intelligent controller mode, BURSTINIT is automatically set to BURSTSIZE for transmit DMA. The **Intelligent Controller DMA Read Timing Diagram** (Figure 10) shows the loading of a transmit pointer into the receive FIFO. The loading of the burst size/word count word is similar, except for the values of the op.~rcvAdr and op.~trAdr lines.

For receive DMA, the intelligent controller loads a DMA receive address into the receive FIFO. Once this address reaches the head of the receive FIFO, it is merged into the RDP under the control of the DMP. Subsequent option data words will be written to system memory at the address pointed to by the RDP. The RDP is incremented as each word is written into system memory. The TcIA will initiate a DMA write transactions to system memory as soon as the number of words in the receive FIFO is greater than or equal to RCVBURST. A programmable counter is used that flushes the receive FIFO when the counter reaches zero if the head of the receive FIFO is a data word. The **Intelligent Controller DMA Write Timing Diagram** (Figure 11) shows the loading of a receive pointer and four words of DMA data into the receive FIFO.

Intelligent controllers may have addresses of arbitrary widths; for example a particular Ethernet controller generates a 16 bit address that may be extended to 24 bits. To set a specific address width, the pointer mask register should be programmed by loading the DPM register with the correct mask value. This mask indicates which address range the option will supply and which will be fixed in the DMA pointer registers. The address word loaded into the receive FIFO must be loaded on one option clock cycle; there is no packing for DMA addresses. Intelligent controller data, however, may be loaded into the receive FIFO a byte, half-word, or a word at a time. Since the intelligent controller does not have access to the data alignment mechanisms in the CSRs, the data must begin and end on word boundaries.

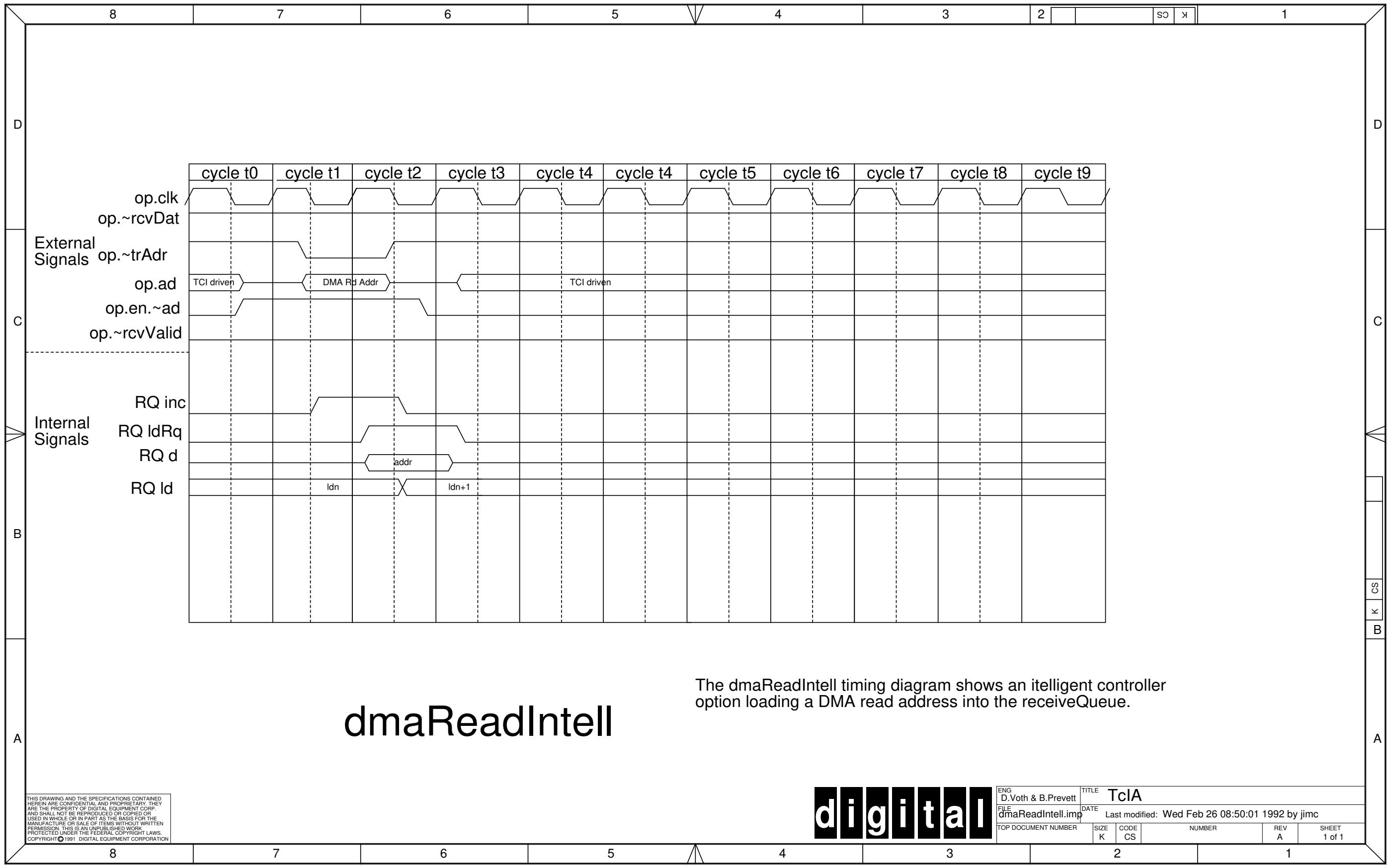
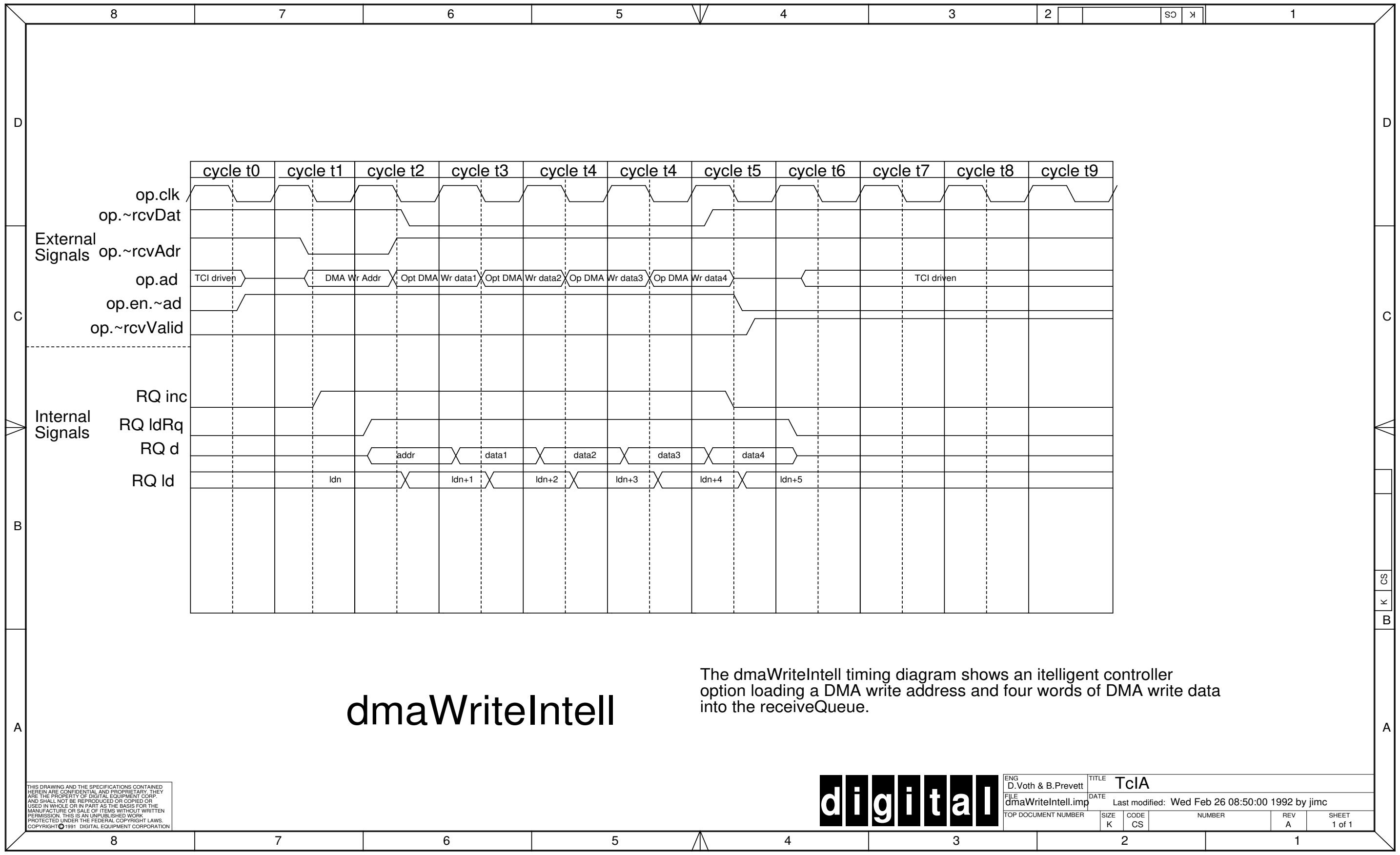


Figure 10 Intelligent Controller DMA Read Timing Diagram

Figure 11 Intelligent Controller DMA Write Timing Diagram



5.6 Intelligent Controller DMA Examples

5.6.1 Intelligent Controller Transmit DMA Example

For this example, consider a Token Ring option that uses an intelligent controller. In the intelligent controller DMA model, the option controller generates its own DMA address. This controller generates 16-bit addresses; therefore, the DMA Pointer Mask (DPM) must be used to supply the upper order address bits of the system memory buffer.

There is 2Kbyte of packet data in a system memory buffer that must be transferred to the token ring controller and onto the network. The controlling software:

* writes the upper-order address bits into the TDP
* sets an address mask into the DPM that selects the upper 16 address bits from the TDP and the lower 16 address bits as supplied by the option
* configures the OIMR to interrupt when op.~gpi[1] is asserted
* configures Option Control Register 1 (OCR1) by:
    * setting the TFEN and TDMAEN bits
    * setting the RFEN bit (to allow control words to be written to the receive FIFO)
    * setting SIZE = word
    * setting the GPO[1] bit.

The TcIA option side signal op.gpo[1] asserts, signaling to the token ring option that there is a group of packets to be transmitted. The option controller, in response to the assertion of op.gpo[1], begins the transmit DMA.

The option controller first writes the transmit address into the receive FIFO by placing the 16 bit address onto op.ad and asserting the op.~trAdr signal for one op.clk cycle. When the transmit address reaches the head of the receive FIFO, this address is combined with the TDP as defined by the DPM and written back into the TDP if the TWC is zero; if the TWC is not zero, the address will wait at the head of the receive FIFO until the TWC becomes zero. The option controller then writes the burst size/word count word into the receive FIFO. When the burst size/word count word reaches the head of the receive FIFO, the BURSTSIZE and BURSTINIT fields of OCR1 and the TWC are updated if the TWC is zero; if the TWC is not zero, the burst size/word count word will wait at the head of the receive FIFO until the TWC becomes zero. When the TWC becomes non-zero, the DMA engine issues DMA read transactions of BURSTSIZE length, beginning at the specified address. As the data words are received by the TcIA, they are written into the transmit FIFO. The presence of valid data words in the transmit FIFO asserts the op.~trValid signal. The option reads out the data words by asserting the op.~trDack signal. When the option has received the full 2Kbyte of packet data, it asserts the op.~gpi[1] signal. The signal tc.~int asserts, signaling to the controlling software that the 2Kbyte transfer has completed.

Intelligent controllers may use the byte unpacker of the transmit FIFO for byte or half-word data for reading data out of the transmit FIFO, but the transmit address supplied to the receive FIFO does not go through the byte packer. The transmit address must be supplied on one op.clk cycle.

5.6.2 Intelligent Controller Receive DMA Example

The Token Ring Controller will again be used for this example. The option has received 2Kbyte of packet data from the token ring network; this data must be transferred to a system memory buffer via receive DMA. The controlling software must configure the TcIA for intelligent controller receive DMA by:

* writing the upper-order address bits of the system memory buffer into the RDP
* setting an address mask into the DPM that selects the upper 16 address bits from the RDP and the lower 16 address bits as supplied by the option
* configuring Option Control Register 2 (OCR2) by:
    * setting the RCVTHRESHOLD = 001
    * setting the RCVBURST = 10 (4 valid words)
    * setting the RCVFLSHCNT = 00 (16 TURBOchannel cycles)
* configuring the OIMR to interrupt when op.~gpi[1] is asserted
* configuring Option Control Register 1 (OCR1) by setting the RFEN and RDMAEN bits.

The option controller writes the receive address into the receive FIFO by placing the 16 bit address onto op.ad and asserting the op.~rcvAdr signal for one op.clk cycle. The option then writes the DMA write data into the receive FIFO by placing the data onto op.ad and asserting the op.~rcvDat signal.

When this address reaches the head of the receive FIFO, it is combined with the RDP as defined by the DPM and written back into the RDP. The DMA write data is written into system memory from the receive FIFO. As the receive FIFO is emptied by the DMA write transactions, the option controller writes more data into the receive FIFO. As in the simple receive DMA example, remaining data words are flushed out of the receive FIFO with the flush counter mechanism. When the option has written all of the 2Kbyte packet data into the receive FIFO, it asserts the op.~gpi[1] signal, triggering a TURBOchannel interrupt to signal the completion of the transfer into the receive FIFO. The controlling software will then poll the OIR RFE bit to determine when the last data words have been written out of the receive FIFO and into the system memory buffer.

NOTE

> In intelligent controller mode, the receive FIFO can contain receive data, receive DMA addresses, transmit DMA addresses and transmit burst size/word count words. Transmit DMA addresses and transmit burst size/word count words will NOT be removed from the FIFO unless TWC = 0. Either of these items in the FIFO will prevent receive DMA addresses or data words from reaching the head of the FIFO.

Intelligent controllers may use the byte packer of the receive FIFO for byte or half-word data, but the byte packer is not used for the receive address. The receive address must be supplied in one op.clk cycle.

Blank Page

6 Programming Guide

Following are detailed TcIA programming sequences for simple (non-intelligent) transmit and receive DMA transfers. Each of the sections present issues that need to be addressed when designing software to control the TcIA. These issues are addressed through example data buffers and assume that the option uses byte-wide data. The examples are defined with the transfers beginning and ending on arbitrary byte boundaries, that is, the data are not word-aligned. The byte-aligned data transfers were chosen to show the most complicated software required to support the TcIA; the software for word-width transfers, which must be word-aligned, will not have to deal with the start-up and end conditions required by byte-wide, byte-aligned transfers.

In addition, each of the software sequences provide variants for handling the end-of-transfer conditions to show potential hardware/software trade-offs.

6.1 Software Set Up for Simple Transmit DMA with Byte-Aligned Data

The discussion begins with an overview of TcIA features supporting transmit DMA operations, provides details of features that support the sample data buffer, covers design issues that must be addressed by the control software for transmit DMA and concludes with a walk-through of the steps required to perform a DMA transfer from the sample buffer.

6.1.1 Review of TcIA simple transmit DMA features

The TcIA provides the interface between option circuitry and the word-wide, word-aligned DMA on the TURBOchannel The TcIA simple transmit DMA was designed to support the transfer of blocks of data from system memory to an option. Some of the general features of the TcIA supporting simple transmit DMA operations are:

- Option data size can be a 32-bit word, a 16-bit half-word or an 8-bit byte.
- Unpacking of 32-bit words into half-words or bytes for transfer to the option.
- With software assistance, option data retrieval can begin and end on data-size boundaries (i.e. bytes on byte-boundaries). Without software assistance, data can begin and end only on word boundaries.
- 15 word receive FIFO with individually programmable DMA read burst sizes for the first and all subsequent DMA read transactions.
- Transmit scatter/gather table for DMA operations from physically discontiguous memory pages (the size of the pages is program-selectable).
- 24-bit counter of words to transfer.

6.1.2 TcIA support for the sample data buffer transmit DMA

1. **Non word-aligned buffer start and end -** The TcIA handles arbitrary-byte alignment (for byte-wide data) at the beginning and end of a transmit DMA buffer as follows:

   a. Beginning of the buffer - The byte-unpack feature of the TcIA can be set up to automatically handle the data from a buffer that begins on an arbitrary byte within a word.

   b. End of buffer - The controlling software and/or surrounding circuitry must help the TcIA pass the last 1 to 3 bytes to the option.

2. **Burst size -** The TURBOchannel is most efficient when used with large DMA read transactions. But there are three requirements for DMA read burst lengths:

   a. Bursts must not cross 2Kbyte boundaries - this is a memory system requirement.

   b. Bursts must not cross "page" boundaries - this will defeat the scatter/gather feature of the TcIA.

   c. Bursts may not be greater than the system-specific limit - this is another memory system requirement.

   These requirements can be resolved fairly easily by making DMA read transactions end on address boundaries such that (address of next word) modulo (burst size) = 0. But aligning the transactions in the middle of the transfer still leaves the end conditions for buffers that don't start and end on burst boundaries. Features of the TcIA solve problems with both of the end conditions:

   The size of the first DMA read transaction can be specified separately from the burst size used for the remainder of the transfer; this allows a small initial transaction to align all of the following transactions to the appropriate boundary. When the first transaction begins on a burst boundary, the initial burst size is set the same as the burst size for the rest of the transfer.

   The TcIA discards all words from the last DMA read transaction after the word count goes to zero; this takes care of the end condition.

6.1.3 Design Issues for transmit DMA control software

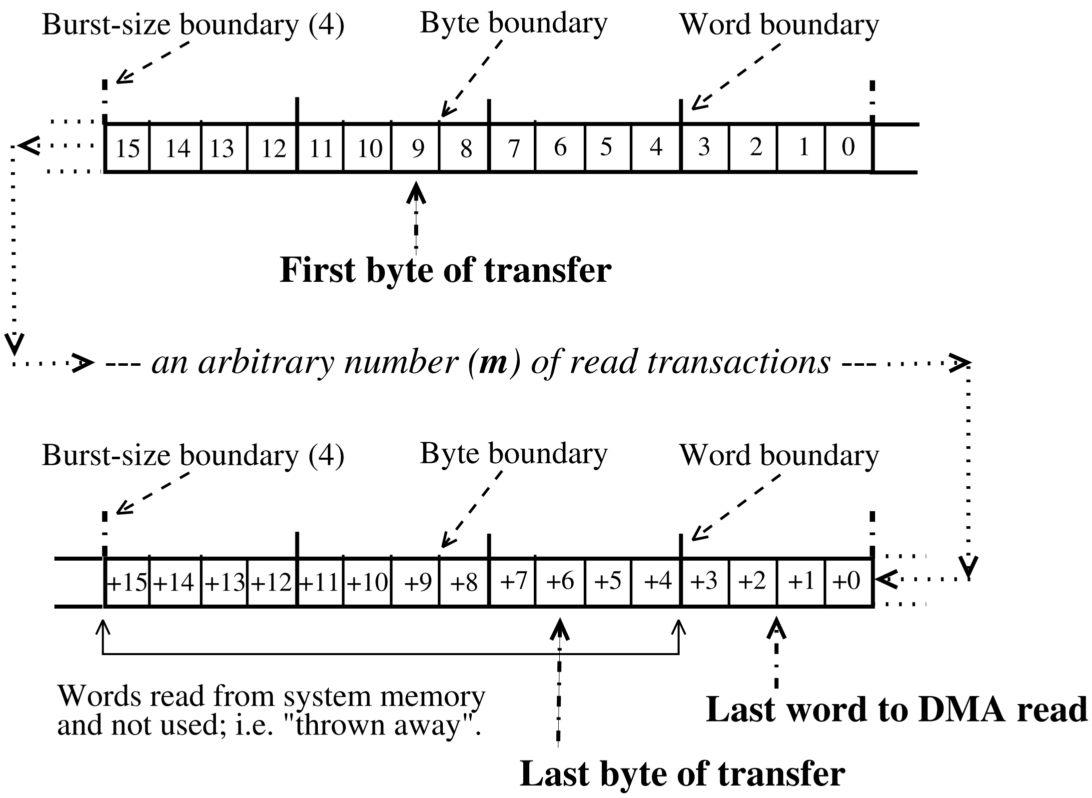
The design issues for transmit DMA are simple, even with byte-wide data. Since the TcIA half-word/byte-unpacker allows the control software to set the position of the first transmitted half-word/byte within the first word of the transfer and the amount of data to transfer is almost always known in advance, the only issue is how to handle the transfer of the final byte(s) or half-word when only part of the final word of a buffer is to be sent to the option.

6.1.4 Software setup for a simple transmit DMA with byte-aligned data

The software that controls the TcIA transmit DMA will generally be split into two parts, (1) the code to do TcIA initialization that only needs to be done once and (2) the code that does each data transfer. The code that does each data transfer will generally also have two sections, (i) the part that sets up the TcIA to start the DMA operation and (ii) the part that does any end-of-transfer cleanup. The set-up part handles partial-word data at the beginning of the transfer and the cleanup part handles the partial-word data at the end of the transfer.

For the rest of this discussion, the sample byte data shown in the Figure 12, **Diagram of Sample Buffer for Simple Byte Transmit DMA**, will be used as an example. The example assumes that a MIPS processor is being used, that the option uses byte-wide data and that the transfer begins and ends on the arbitrary bytes shown (not word-aligned); this shows the most complicated software required by the TcIA.

Figure 12 Diagram of Sample Buffer for Simple Byte Transmit DMA



6.1.4.1 Initialization code (one time) for transmit DMA

Although all registers of the TcIA are reset to zero when power is applied to the chip, it is good programming practice to make no assumptions about the exact state of the chip. The initialization code should do the following steps to guarantee the state of the TcIA for the rest of the code.

1. Load OCR1 with zeros in all fields except for the option-data packing/unpacking size field (SIZE) and the general purpose output field (GPO). (Setting the TFEN and RFEN bits to zero will clear the transmit and transmit FIFOs and associated pointers.) Set the SIZE field to 2 to indicate byte-sized transfers. The GPO field may or may not need to be set to some device-dependent value.

2. Load OCR2 with zeros in all fields except for-

   a. Set scatter/gather page size (PAGESIZE). For this example, set it to 10 (0xa) to match the 4Kbyte virtual memory page size used by MIPS processors. It could be set to some smaller value to match special hardware requirements. If it is set larger than the virtual memory page size of the processor, the TcIA will not be able to assist with gathering data from virtually-contiguous pages of memory that are scattered within physical memory. Since this would defeat a very important feature of the TcIA, it is NOT recommended. Note that the page size field controls the hardware for BOTH transmit and receive DMA operations.

   b. Set TURBOchannel slot size (SLOTSIZE). Change if both the system and the option support a TURBOchannel slot size of greater than 4Mbytes. This field should be set by the option's ROM code during power-up to correspond to the smaller of the option's maximum supported slot size and the value in the "slot size" field returned by the "gettcinfo" callback[1].

   c. Clear option data packing/unpacking order (BIGENDIAN). This field will be zero for "little-endian" for Digital systems.

3. Clear all transmit error bits in the OIR.

4. Clear all bits in the OIMR except parity error (PERR). This can be done by writing 0's to all bits except PERR in OIMR. PERR should be set by the option's ROM code on power-up to correspond to the value in the "parity" field returned by the "gettcinfo" callback[1].

NOTE

> Although the initialization sequences for the transmit and receive DMA examples are shown separately, they will normally both be part of a single initialization routine for the TcIA.

6.1.4.2 Transmit DMA start-up code (in approximate order)

This code sets up the transmit DMA operation for the part of the transfer that ends with the last full word. For this example, the transfer will stop at the word labeled "Last word to DMA read"

[1] See the *TURBOchannel Firmware Specification* for additional information.

for **Scenarios A, B or C below** and at the word labeled "Last byte of transfer" for **Scenario D below** (see Figure 12 on page 51).

1. Do virtual to physical address translation and transmit scatter/gather table creation.

   a. Convert the virtual byte address of the beginning of the transfer to an unpack counter value (for the TUPACK field of OCR1) and the address of the first word of the transfer ("8" in the diagram) by saving the low-order 2 bits of the virtual byte address as the unpack counter value and then clearing these 2 bits.

   b. Convert the virtual address of the first word of the transfer to its physical word address and save the value for loading into the TDP register.

   c. Compute the number of virtual memory pages within the transmit buffer.

   d. Allocate a contiguous section of unmapped (kernel) memory for the transmit scatter/gather table (4 bytes for each entry, word-aligned, 1 entry per virtual memory page). Note that this memory could also be allocated statically to support the maximum usable transfer size for the option.

   e. Convert the virtual address of the first word of the transmit scatter/gather table to its physical word address and save the value for loading into the TSGP register.

   f. For each of the remaining virtual memory pages of the transfer--

      (1) Calculate the virtual address of the beginning of the next page of the transfer (for a MIPS processor with 4Kbyte pages, perform a logical AND of the virtual address with 0xFFFFF000 and then ADD 0x1000; the result will be the next page address).

      (2) Convert the virtual address of the first word of the virtual memory page to its physical word address and save the value in the transmit scatter/gather table.

2. Set OCR1 fields. Fields not explicitly specified below, for example, the option-data packing/unpacking size (SIZE), MUST NOT be changed.

   a. Set initial burst count (BURSTINIT)[1] to 2 for an initial 2-word burst to align with the next burst boundary.

   b. Set burst size (BURSTSIZE)[1] to 4 for 4-word bursts for the rest of the transfer.

[1] (a) The burst size chosen for the transfer should be a power of 2. (b) The initial burst count value should be chosen so that each transaction begins on an address which is aligned with the burst size; this can be achieved by setting the initial burst size to the result of [burst size] - [the starting word (not byte) address] modulo [burst size]. Other choices are possible, but this set of choices will ensure that the transfer does not cross virtual memory system or physical memory system page boundaries.

c. Clear the transmit FIFO enable (TFEN) bit to flush the transmit FIFO and enable setting the transmit unpack counter (TUPACK).

d. Set transmit unpack counter (TUPACK) to 1 to start at byte 1 in the word.

3. Load the TDP with the physical word address where the transmit begins.

4. Load the TSGP with the physical word address of the first entry of the transmit scatter/gather table (to point to the unmapped kernel memory list of "page" start addresses for each "page" crossed during the transfer [see the page size field of OCR1]).

5. Load the TWC {depends on end-of-transfer handling. For **Scenarios A, B or C below:** 4 * ***m*** + 3, for the first 2 words, ***m*** transactions and the last word. For **Scenario D below:** 4 * ***m*** + 4, for the first 2 words, ***m*** transactions, and last 2 words [this requires the option to have its own byte counter, which must be set to ignore the extra byte(s) in the last FIFO word]}.

6. Clear all transmit error bits in the OIR.

7. Set OIMR bits to enable interrupts on transmit DMA error (TDMAERR), transmit word count expired and transmit FIFO empty (TWCEXPTFE) and possibly other bits.

8. Do final setting of OCR1 bits[1].

a. Set transmit DMA enable (TDMAEN) bit.

b. Set option-side transmit DMA enable (TFEN) bit.

6.1.4.3 Transmit DMA end-of-transfer cleanup code

The end-of-transfer code, for our example data buffer, will have to assist in the transfer of the last 3 bytes of the data buffer. Since the required code will depend on the actual implementation of the option-side circuitry surrounding the TcIA, several possible code scenarios are presented below. All of the scenarios assume that the transfer to the option will complete (the transmit word count will go to zero) without an error, the transmit FIFO will empty, the transmit word count expired and transmit FIFO empty bit (TWCEXPTFE) will be set in the OIR and an interrupt will occur.

1. Clear the TWCEXPTFE bit in the OIMR by writing a zero to it (other bits may or may not need to be cleared; write 1's to bits that should NOT be cleared). This is the only way to disable continuing interrupts from this bit.

[1]At least 3 option clock (op.clk) cycles must have elapsed between the time that the transmit unpack field (TUPACK) is written (the TFEN bit in OCR1 must be clear) and the time that the TFEN bit is set in OCR1. If op.synchronous is asserted (the option clock is the same as the TURBOchannel clock), then this can not be a problem. If op.synchronous is deasserted (the option clock is not the same as the TURBOchannel clock), then failing to meet this condition will place the transmit unpacking into an undefined state.

2. Clear any needed bits in the OIR by writing a zero to them (other bits may or may not need to be cleared; write 1's to bits that should NOT be cleared). Note that the TW-CEXPTFE bit can't be cleared in the OIR; it is set statically by the condition of transmit word count expired and transmit FIFO empty.

3. Continue with one of the following scenarios (each implementation of the option-side circuitry external to the TcIA that handles the final bytes of the transfer is described at the beginning of the scenario).

   a. **Scenario A:** Uses the TcIA's unpacking capability and no other hardware to transfer just the last 3 bytes to the option.

      (1) When the transmit FIFO is empty, clear the TFEN bit in OCR1 and load the transmit unpack field (TUPACK) in OCR1 with a transmit unpack value to make just the desired bytes available to the option (3 bytes means set unpack count to 1 to start at byte 1).

      (2) Delay as required to ensure propagation of the TUPACK setting into the option time domain (see footnote on final setting of OCR1 bits during DMA setup).

      (3) Re-enable the transmit FIFO by setting the TFEN bit in OCR1.

      (4) Shift the last word left by the number of unwanted bytes in that word (1 byte).

      (5) PIO write the shifted last word to the transmit FIFO through the FAR.

      (6) Inform the option that the end of data has arrived. This could be performed by setting one of the GPO bits. If the transfer of the last bytes will take a "long" time, one of the GPI bits could be enabled (through the OIMR) to cause an interrupt when the option has finished the transfer.

   b. **Scenario B:** Uses some extra hardware to transfer just the last 3 bytes to the option.

      (1) When the transmit FIFO is empty, load two GPO bits in OCR1 with the count of the remaining bytes (3). The option must have circuitry external to the TcIA that will know how to interpret these bits.

      (2) PIO write the last word to the transmit FIFO. The external circuitry in the option will use just the specified bytes from the last word (3).

      (3) Inform the option that the end of data has arrived. This could be performed by setting another of the GPO bits. If the transfer of the last bytes will take a "long" time, one of the GPI bits could be enabled (through the OIMR) to cause an interrupt when the option has finished the transfer.

c. **Scenario C:** Uses a separate path (e.g. a register mapped into the option address space at a slot offset below 0x300) to transfer just the last 3 bytes to the option.

(1) When the transmit FIFO is empty, clear the TFEN bit in OCR1.

(2) PIO write the remaining 3 bytes of the last word through the separate path to the option.

d. **Scenario D:** Uses a separate byte counter in the option (e.g. a register mapped into the option address space at a slot offset below 0x300). It must be loaded when the word counter in the TcIA is loaded and the word count loaded into the TcIA will include both the first and last (partially used) words. The option circuitry will use only the specified number of bytes. The TcIA may be enabled to interrupt on transmit word count expired or on one of the GPI bits.

NOTE: This scenario changes the diagram of the sample byte data buffer slightly. The last word to DMA read will be the word containing the last byte of the transfer, not the preceding word.

(1) When the transmit FIFO is empty, clear the TFEN bit to flush any remaining byte(s) from the FIFO (1 byte will be left).

6.2 Software Set Up for Simple Receive DMA with Byte-Aligned Data

The discussion begins with an overview of TcIA features supporting receive DMA operations, provides details of features that support the sample data buffer, covers design issues that must be addressed by the control software for receive DMA and concludes with a walk-through of the steps required to perform a DMA transfer into the sample buffer.

6.2.1 Review of TcIA simple receive DMA features

The TcIA provides the interface between option circuitry and the word-wide, word-aligned DMA on the TURBOchannel. The TcIA simple receive DMA was designed to support the transfer of blocks of data from an option to system memory. Some of the general features of the TcIA supporting simple receive DMA operations are:

- Option data size can be a 32-bit word, a 16-bit half-word or an 8-bit byte.
- Packing of half-words or bytes into 32-bit words for transfer to the TURBOchannel.
- With software assistance, option data storage can begin and end on data-size boundaries (i.e. bytes on byte-boundaries). Without software assistance, data can begin and end only on word boundaries.
- 15 word receive FIFO with programmable receive burst size (minimum amount of data before beginning a transaction) and auto-flush after a programmable number of

TURBOchannel cycles.

- Receive scatter/gather table for DMA operations into physically discontiguous memory pages (the size of the pages is program-selectable).

- 24-bit counter of words to transfer.

6.2.2 TcIA support for the sample data buffer receive DMA

1. **Non word-aligned buffer start and end -** The TcIA handles arbitrary-byte alignment (for byte-wide data) at the beginning and end of a receive DMA buffer as follows:

   a. Beginning of the buffer - The byte-pack feature of the TcIA can be set up to position data bytes for a buffer that begins on an arbitrary byte. However, there is no way to pre-load the byte-packer, so the controlling software must either save and restore bytes within the first word of the buffer that precedes the desired first byte position or force the transfer to start on a word boundary.

   b. End of buffer - The controlling software must help the TcIA pass the last 1 to 3 bytes into system memory.

2. **Burst size -** The TURBOchannel is most efficient when used with large DMA write transactions. But there are three requirements for DMA write burst lengths:

   a. Bursts must not cross 2Kbyte boundaries - this is a memory system requirement.

   b. Bursts must not cross "page" boundaries - this will defeat the scatter/gather feature of the TcIA.

   c. Bursts may not be greater than the system-specific limit - this is another memory system requirement.

   In addition to these requirements, two other DMA write transaction parameters need to be controlled for efficient DMA writes:

   d. DMA write transactions should not occur until enough data is in the receive FIFO for an efficient burst.

   e. The receive FIFO needs to be automatically emptied at the end of a data transfer or when the data has been waiting in the FIFO "long enough".

   These requirements are all resolved by features of the TcIA:

   The minimum DMA write burst size can be specified to the TcIA. This allows the controlling software to optimize (trade-off) latency vs. transaction efficiency.

   The TcIA breaks DMA write bursts at 64-word boundaries. This guarantees that

DMA write transactions will not cross 2Kbyte or "page" boundaries or exceed the system transfer length.

The TcIA will flush all words remaining in the receive FIFO after a program-selectable number of TURBOchannel clock cycles. This guarantees that stale data does not remain in the FIFO during or at the end of the DMA operation.

6.2.3 Design Issues for receive DMA control software

The design of receive DMA control software for the TcIA must account for the TcIA receive DMA architecture and the characteristics of the device supplying the data. The TcIA half-word/byte-packer allows the control software to position the first received half-word/byte within the first word of the transfer. When only part of the first word of a buffer is to be modified, software must save and restore the other bytes; otherwise it must ensure that a transfer begins on a word boundary. This is because the TcIA does not have either a read-modify-write cycle or a way to pre-load the packer to assist with modification of a part of the first word of the transfer.

An additional issue can arise with receive DMA - the potential difficulty of predicting the amount of data that will be received. Devices can generally be split into two broad categories: block-oriented (disks) and character/record-oriented (tapes and serial devices). Block-oriented devices typically have known block sizes and the size of a receive DMA transfer will be known when the transfer begins. When data is received from these devices, the receive word count will be set to reflect the size of the buffer allocated (which is also the amount of data expected) and word count expired will be used to indicate transfer completed. For character-oriented devices, this may not be the case: the length of a tape record may not be known in advance and the length of the next input line from a user or the next network message will probably not be known in advance.

For devices with unpredictable amounts of input data, the receive word count will generally be used to specify the size of the buffer allocated, i.e. the maximum number of words that can be received to system memory, and both word count expired and some attention condition (end of line/record, special attention character, timer) will be used to indicate that software attention is required. When the attention interrupt occurs, one of several cases are possible: (1) the word count expired at the same time that the attention condition was detected and both the FIFO and the packer are empty; (2) the attention condition occurred before the word count expired; or (3) the word count expired before the attention condition occurred. In the last two cases, there may also be data in the packer which needs to be transferred to system memory. The software response to these three cases is:

1. When the word count expires at the same time that the attention condition occurs and both the FIFO and packer are empty, the transfer is finished and no further action is required. This is the simplest case

2. When the attention condition occurs before the word count expires, there may still be data in the FIFO and packer, but an appropriate setting of the flush counter will cause the TcIA

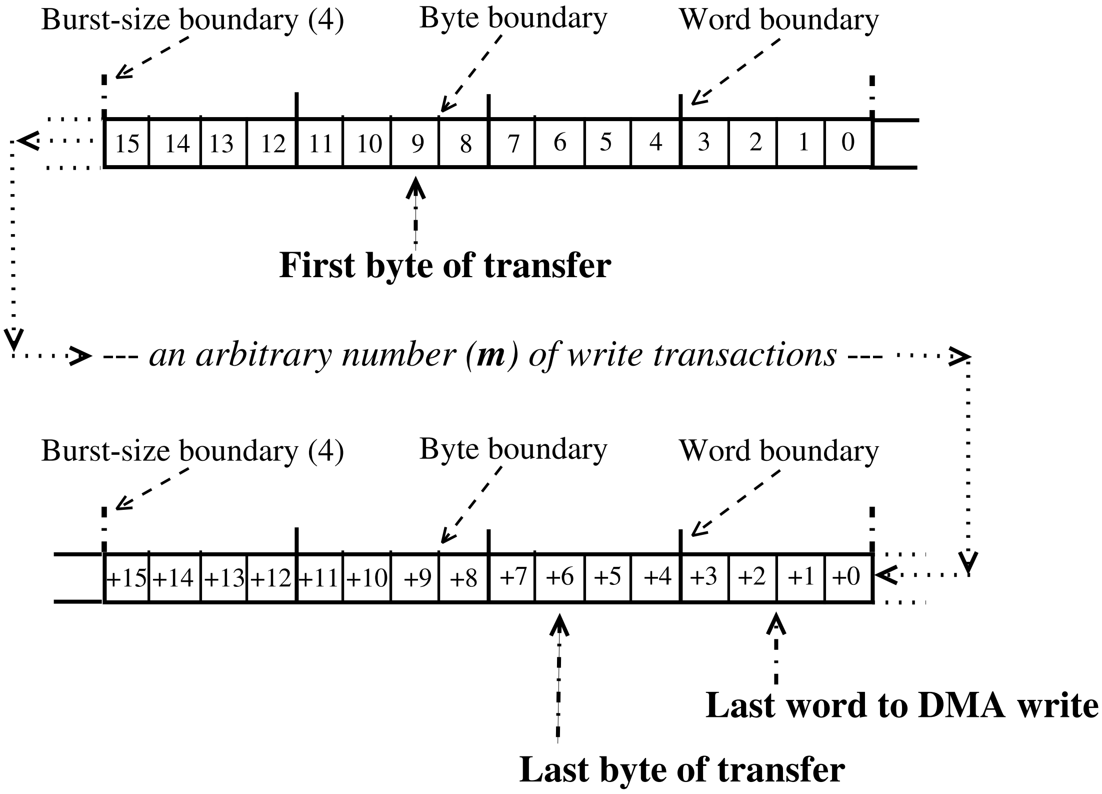
to automatically flush the FIFO while the interrupt response is occurring. After the FIFO has flushed, reading the FIFO Access Register and the pack count field will allow the software to determine the number of bytes in the packer and store them into system memory.

NOTE

> This sequence will not work unless the option-side circuitry that determines the attention condition ALSO stops the option from loading any additional data into the receive FIFO. This is because the FIFO must be empty before the packer can be read and the packer and pack counter must be stable for the software to determine the number of bytes in the packer.

3. When the word count expires before the attention condition occurs, it is highly likely that data may also be left in both the FIFO and the packer. In this case, the controlling software might re-load the pointer and word count registers for a new buffer and re-enable DMA or it might define this as an error condition requiring a retry and flush the FIFO.

Figure 13 Diagram of Sample Buffer for Simple Byte Receive DMA



6.2.4 Software setup for a simple receive DMA with byte-aligned data

When the TcIA is part of a TURBOchannel option, the software that controls the TcIA will generally be split into two parts, (1) the code to do TcIA initialization that only needs to be done once and (2) the code that does each data transfer. The code that does each data transfer will generally also have two sections, (i) the part that sets up the TcIA to start the DMA operation and (ii) the part that does any end-of-transfer cleanup. The set-up part handles partial-word data at the beginning of the transfer, including saving byte(s) within the first word that are not part of the transfer, and the cleanup part handles the partial-word data at the end of the transfer and restores any saved byte(s) from the first word.

For the rest of this discussion, the sample byte data buffer shown in the Figure 13, **Diagram of Sample Buffer for Simple Byte Receive DMA**, will be used as an example. The example assumes that a MIPS processor is being used, that the option provides byte-wide data and that the transfer begins and ends on the arbitrary bytes shown above (not word-aligned); this shows the most complicated software required by the TcIA.

6.2.4.1 Initialization code for receive DMA (one time)

Although all registers of the TcIA are reset to zero when power is applied to the TcIA, it is good programming practice to make no assumptions about the exact state of the chip. The initialization code should do the following steps to guarantee the state of the TcIA for the rest of the code.

1. Load OCR1 with zeros in all fields except for the option-data packing/unpacking size field (SIZE) and the general purpose output field (GPO). (Setting the TFEN and RFEN bits to zero will clear the transmit and receive FIFOs and associated pointers.) Set the SIZE field to 2 to indicate byte-sized transfers. The GPO field may or may not need to be set to some device-dependent value.

2. Load OCR2 with zeros in all fields except for-

    a. Set scatter/gather page size (PAGESIZE). For this example, set it to 10 (0xa) to match the 4Kbyte virtual memory page size used by MIPS processors. It could be set to some smaller value to match special hardware requirements. If it is set larger than the virtual memory page size of the processor, the TcIA will not be able to assist with gathering data from virtually-contiguous pages of memory that are scattered within physical memory. Since this would defeat a very important feature of the TcIA, it is NOT recommended. Note that the page size field controls the hardware for BOTH transmit and receive DMA operations.

    b. Set receive flush counter (RCVFLUSH) and receive FIFO threshold signal (RCVTHRESHOLD). The values to use for these fields are option-dependent. The value to use for RCVFLUSH is a trade-off between DMA write transaction size efficiency and data latency at the end of the transfer. The value to use for RCVTHRESHOLD depends on data rates and pipe-line architecture in the option-side hardware. NOTE: These fields could also be set up in the receive DMA start-up code and the receive burst size (RCVBURST) could also be set here (4 for minimum 4-word bursts).

c. Set TURBOchannel slot size (SLOTSIZE). Change if both the system and the option support a TURBOchannel slot size of greater than 4Mbytes. This field should be set by the option's ROM code during power-up to correspond to the smaller of the option's maximum supported slot size and the value in the "slot size" field returned by the "gettcinfo" callback[1].

d. Clear option data packing/unpacking order (BIGENDIAN). This field will be zero for "little-endian".

1. Clear all bits in the OIMR except parity error (PERR). This can be done by writing 0's to all bits except PERR in OIMR. PERR should be set by the option's ROM code on power-up to correspond to the value in the "parity" field returned by the "gettcinfo" callback[2].

NOTE

Although the initialization sequences for the transmit and receive DMA examples are shown separately, they will normally both be part of a single initialization routine for the TcIA.

6.2.4.2 Receive DMA start-up code (in approximate order)

This code sets up the receive DMA operation for the part of the transfer that ends with the last full word. For this example, the transfer will stop at the word labeled "Last word to DMA write" for **Scenario A below** and at the word labeled "Last byte of transfer" for **Scenario B below** (see Figure 13 on page 59).

1. Do virtual to physical address translation and receive scatter/gather table creation.

   a. Convert the virtual byte address of the beginning of the transfer to a pack counter value (for the RPACK field of OCR1) and the address of the first word of the transfer ("8" in the diagram) by saving the low-order 2 bits of the virtual byte address as the pack counter value and then clearing these 2 bits.

   b. Convert the virtual address of the first word of the transfer to its physical word address and save the value for loading into the RDP.

   c. Compute the number of virtual memory pages within the receive buffer.

   d. Allocate a contiguous section of unmapped (kernel) memory for the receive scatter/gather table (4 bytes for each entry, word-aligned, 1 entry per virtual memory page). Note that this memory could also be allocated statically to support the

[1] See the *TURBOchannel Firmware Specification* for additional information.

[2] See the *TURBOchannel Firmware Specification* for additional information.

maximum usable transfer size for the option.

e. Convert the virtual address of the first word of the receive scatter/gather table to its physical word address and save the value for loading into the RSGP.

f. For each of the remaining virtual memory pages of the transfer--

(1) Calculate the virtual address of the beginning of the next page of the transfer (for a MIPS processor with 4Kbyte pages, perform a logical AND of the virtual address with 0xFFFFF000 and then ADD 0x1000; the result will be the next page address).

(2) Convert the virtual address of the first word of the virtual memory page to its physical word address and save the value in the receive scatter/gather table.

2. Set OCR1 fields. Fields not explicitly specified below, for example, the option-data packing/unpacking size (SIZE), MUST NOT be changed.

a. Clear the receive FIFO enable (RFEN) bit to flush the receive FIFO and enable setting the receive pack counter (RPACK).

b. Set the receive pack counter (RPACK) to 1 to start at byte 1 in the word.

3. Set OCR2 fields. Fields not explicitly specified below, for example, the big/little-endian control bit (BIGENDIAN), MUST NOT be changed.

a. Set the receive burst size (RCVBURST) to 4 for minimum 4-word bursts.

NOTE: This field could also be set up in the TcIA initialization code and the receive flush counter (RCVFLUSH) and receive FIFO disable threshold (RCVTHRESHOLD) could also be set here.

4. Load the RDP with the physical word address where the receive will begin.

5. Load the RSGP with the physical word address of the first entry of the receive scatter/gather table (to point to the unmapped kernel memory list of "page" start addresses for each "page" crossed during the transfer [see the page size field of OCR1]).

6. Load the RWC {depends on end-of-transfer handling. For **Scenario A below:** 4 * ***m*** + 3, for the first 2 words, ***m*** transactions and the last word. For **Scenario B below:** 4 * ***m*** + 4, for the first 2 words, ***m*** transactions, and last 2 words [this requires the option to have its own byte counter, which must be set to supply the extra byte(s) needed to fill in the last FIFO word].}

7. Clear receive error bits in the OIR.

8. Save the contents of byte 0 within the first word of the buffer. This byte is not part of the receive buffer and must be restored after the transfer has taken place.

9. Set OIMR bits (receive DMA error (RDMAERR), receive word count expired (RWCEXPIRED) and possibly other bits).

10. Do final setting of OCR1 bits[1].

    a. Set receive DMA enable (RDMAEN) bit.

    b. Set option-side receive DMA enable (RFEN) bit.

6.2.4.3 Receive DMA end-of-transfer cleanup code

The end-of-transfer code, for our example data buffer, will have to assist in the transfer of the last 3 bytes from the receive FIFO to the data buffer in system memory. Since the required code will depend on the actual implementation of the option-side circuitry surrounding the TcIA, two possible code scenarios are presented below. These scenarios assume that the transfer from the option is a known, fixed size, that it will complete (the receive word count will go to zero) without an error, that the receive word count expired bit (RWCEXPIRED) will be set in the OIR and that an interrupt will occur.

1. Clear the RWCEXPIRED bit in the OIR by writing a zero to it (other bits may or may not need to be cleared; write 1's to bits that should NOT be cleared).

2. Restore the saved contents of byte 0 of the first word of the buffer.

3. Test that the OIR receive FIFO empty bit (RFE) indicates that the FIFO is empty (required before clearing the RFEN bit). If the FIFO is not empty, an error has occurred, so take some action.

4. Continue with one of the following scenarios {each implementation of the option-side circuitry external to the TcIA that handles the final bytes of the transfer is described at the beginning of the scenario}.

    a. **Scenario A:** Uses the TcIA's packing capability with no extra hardware to transfer just the last 3 bytes to the system.

        (1) Check the RPACK2 field in the OIR; it should show 3 bytes packed.

        (2) Read the data from the receive packer through the FAR.

        (3) Store the last 3 bytes into the system buffer.

    b. **Scenario B:** Uses a separate byte counter in the option (e.g. a register mapped into

[1] At least 3 option clock (op.clk) cycles must have elapsed between the time that the receive pack field (RPACK) is written (the RFEN bit in the OCR1 must be clear) and the time that the RFEN bit is set in the OCR1. If op.synchronous is asserted (the option clock is the same as the TURBOchannel clock), then this can not be a problem. If op.synchronous is deasserted (the option clock is not the same as the TURBOchannel clock), then failing to meet this condition will place the receive packing into an undefined state.

the option address space at a slot offset below 0x300). It must be loaded when the word counter in the TcIA is loaded and the word count loaded into the TcIA will include both the first and last (partially used) words. The option circuitry will supply enough extra bytes at the end of the transfer to fill the byte packer so the last bytes will be transferred to system memory. The TcIA may be enabled to interrupt on receive word count expired or on one of the GPI bits.

NOTE: This scenario changes the diagram of the sample byte data buffer slightly. The last word to DMA write will be the word containing the last byte of the transfer, not the preceding word.

(1) Clear the RFEN bit to disable the FIFO.

6.2.5 An alternative for a byte-wide receive to a non-word boundary

The preceding sections described a simple method for handling a receive DMA transfer of byte-wide data that begins on an arbitrary byte boundary. Although the method used is very simple and allows for similar receive and transmit DMA setup code, the contents of byte 0 of the first word of the buffer shown in Figure 13, **Diagram of Sample Buffer for Simple Byte Receive DMA** (page 59), cannot be defined from the time that the receive DMA is enabled until the end-of-transfer cleanup code restores the contents of that byte. Where it is important that the contents of this byte (for the example transfer) not be disturbed during the receive transfer, the receive scatter/gather mechanism can be used to help solve the problem. The receive DMA start-up code would have the following steps added:

- Allocate a page of system memory; it must end (and thus begin) on a page boundary.
- Use the address of the last word of this page as the first word of the receive transfer; the transfer should begin into the same byte specified by the start of the receive DMA buffer.
- Allocate one extra word for the receive DMA scatter/gather table. Load this word with the the physical word address of the first full word of the receive buffer.
- Load the rest of the words of the receive DMA scatter/gather table as described in the previous sections.
- Do all of the other steps (register loads, etc) given in the previous example.

When the receive DMA transfer begins, the TcIA will transfer the first (partial) word to the last word of the allocated page in memory, detect a page crossing, re-load the receive DMA pointer from the first word of the scatter/gather table and continue the transfer.

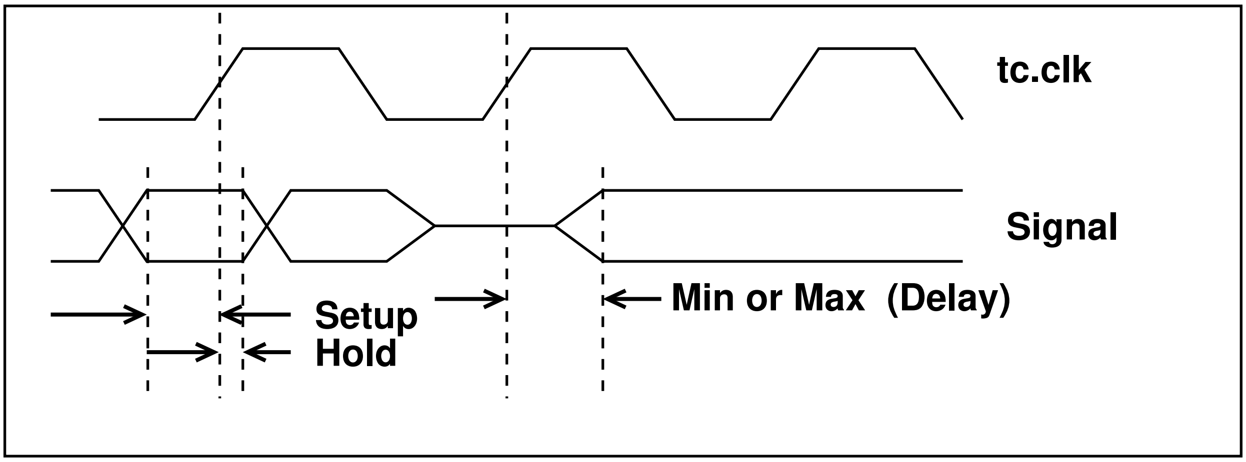
The receive DMA end-of-transfer cleanup code would be changed, also. Instead of restoring the saved byte (or bytes, for a different buffer), it would move the received data byte (or bytes) from the word at the end of the allocated page to the first word of the real buffer.

7 Electrical

The table below lists the timing parameters that are referenced to the rising edge of tc.clk. Timing is measured between signal Vih,Vil to mid-point of clock. All timing values are in units of nanoseconds.

Signal		Source	Min	Max	Setup	Hold
tc.clk	cycle time	system	40	80		
tc.clk	pulse width	system	15			
tc.ad		system			5	2
tc.~sel, tc.~write		system			13	2
tc.~ack, tc.~err		system			13	2
tc.~err, tc.~reset		system			13	2
tc.ad		TcIA	3	34		
tc.ad to 3-state		TcIA	3	22		
tc.~rdy, tc.~conflict		TcIA	1	12		
tc.~int		TcIA	1	12		
tc.~wReq, tc.~rReq		TcIA	1	7		
op.adr		TcIA	1	23		
op.~bm		TcIA	1	23		

Figure 14 tc.clk Signal Timing Diagram

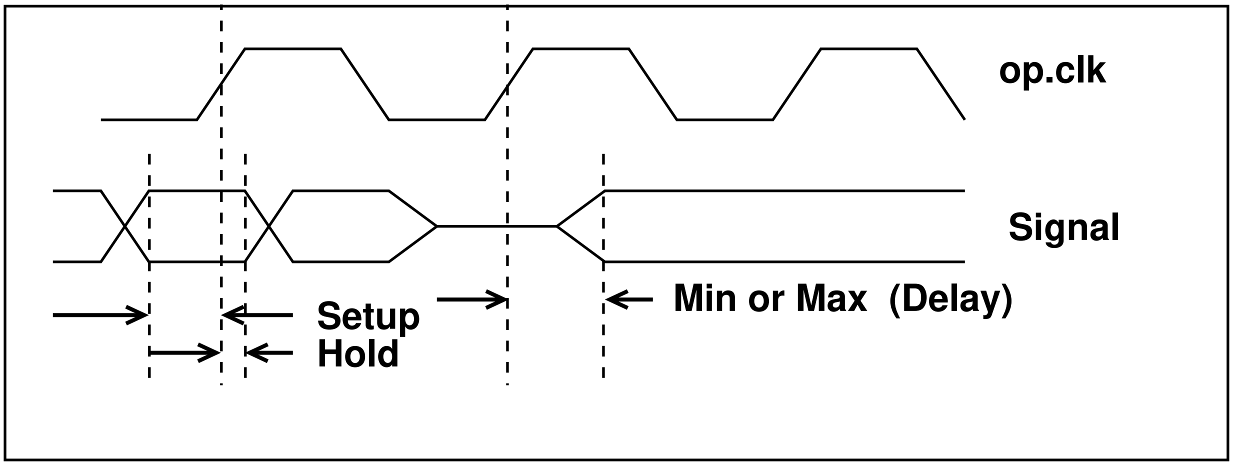


7.0.1 Option Signal Timing

The table below lists the timing parameters that are referenced to the rising edge of op.clk, except for op.~bm and op.~adr which are asserted with tc.clk and op.~sel and op.~write, which are referenced to tc.clk in synchronous mode. Timing is measured between signal Vih,Vil to mid-point of clock (except for the enable lines). All timing values are in units of nanoseconds.

Signal	Source	Min	Max	Setup	Hold
op.clk cycle time	option	TBD	1000		
op.clk pulse width	option	15	950		
op.ad	option			10	0
op.~rcvAdr, op.~rcvDat	option			10	0
op.~rdy	option			10	0
op.~write	TcIA	1	23		
op.~trDack,op.~rcvAdr	option			10	0
op.~conflict	option			10	0
op.ad	TcIA	1	23		
op.gpo, op.~sel	TcIA	1	23		
op.~rcvValid,	TcIA	1	23		
op.~trValid	TcIA	1	23		
op.en.~.ad to op.ad	TcIA	1	23		
op.en.~io to op.ad	TcIA	1	23		
op.en.~adr to op.ad	TcIA	1	23		
op.~gpi	option	Synchronized, no Relationship			

Figure 15 op.clk Signal Timing Diagram



The table below gives the value of the test load and the input capacitance. If option loads are greater than those given in this specification then the associated Max delay value must be derated by 0.15ns/pf.

7.0.2 Loading

Signal(s)	Test Load	Input Capacitance(Max)
tc.ad	180pF	10pF
Other outputs or I/O	65pF	10pF
Any Input		10pF

7.1 DC Electrical Characteristic

Parameter	Value
Voh	2.4V
Vol	0.5V
Vih	2.0V
Vil	0.8V
Idd	TBD ma

7.1.1 Absolute Maximum Ratings

Parameter	Value	Duration
Vin (Max)	VDD+1	Indefinite
Vin (Min)	VSS-1	Indefinite
Vin (Min)	VSS-2.5	20ns
Vdd (Max)	+7V	Indefinite
Storage Temp. (Min)	-65C	Indefinite
Storage Temp. (Max)	+150C	Indefinite
Operating Temp. (Min)	0C	Indefinite
Operating Temp. (Max)	TBD	Indefinite

7.1.2 Power Dissipation

The power dissipation of the TcIA is a function of many variables, including the clock rates and the average sustained bandwidth, but is under 3 Watts. The PQFP (Plastic Quad Flat Package) provides adequate heat dissipation, without a heat sink, even at the maximum clock frequencies.

Table 1 Package Pin Assignments

pin	dir	signal	pin	dir	signal	pin	dir	signal	pin	dir	signal
1	i	Vdd	41	i	Vss	81	i	Vss	121	i	Vss
2	b	op.ad[25]	42	i	op.clk	82	i	tc.clk	122	b	tc.ad[24]
3	b	op.ad[24]	43	i	op.synchronous	83	i	Vdd	123	b	tc.ad[25]
4	b	op.ad[23]	44	o	op.~bm[3]	84	o	tc.~rReq	124	b	tc.ad[26]
5	b	op.ad[22]	45	o	op.~bm[2]	85	i	Vss	125	b	tc.ad[27]
6	b	op.ad[21]	46	o	op.~bm[1]	86	o	tc.~wReq	126	b	tc.ad[28]
7	b	op.ad[20]	47	o	op.~bm[0]	87	i	tc.~err	127	b	tc.ad[29]
8	b	op.ad[19]	48	o	op.adr[19]	88	o	tc.~conflict	128	b	tc.ad[30]
9	b	op.ad[18]	49	o	op.adr[18]	89	b	tc.ad[p]	129	b	tc.ad[31]
10	i	Vss	50	i	Vss	90	i	Vdd	130	i	Vss
11	i	Vdd	51	i	Vdd	91	i	Vss	131	i	Vdd
12	b	op.ad[17]	52	o	op.adr[17]	92	b	tc.ad[0]	132	o	op.gpo[0]
13	b	op.ad[16]	53	o	op.adr[16]	93	b	tc.ad[1]	133	o	op.gpo[1]
14	b	op.ad[15]	54	o	op.adr[15]	94	b	tc.ad[2]	134	o	op.gpo[2]
15	b	op.ad[14]	55	o	op.adr[14]	95	b	tc.ad[3]	135	o	op.gpo[3]
16	b	op.ad[13]	56	o	op.adr[13]	96	b	tc.ad[4]	136	o	op.gpo[4]
17	b	op.ad[12]	57	o	op.adr[12]	97	b	tc.ad[5]	137	o	op.gpo[5]
18	b	op.ad[11]	58	o	op.adr[11]	98	b	tc.ad[6]	138	o	op.gpo[6]
19	b	op.ad[10]	59	o	op.adr[10]	99	b	tc.ad[7]	139	o	op.gpo[7]
20	i	Vdd	60	i	Vss	100	i	Vdd	140	i	Vss
21	i	Vss	61	i	Vdd	101	i	Vss	141	i	op.~gpi[0]
22	b	op.ad[9]	62	o	op.adr[9]	102	b	tc.ad[8]	142	i	op.~gpi[1]
23	b	op.ad[8]	63	o	op.adr[8]	103	b	tc.ad[9]	143	i	op.~gpi[2]
24	b	op.ad[7]	64	o	op.adr[7]	104	b	tc.ad[10]	144	i	op.~gpi[3]
25	b	op.ad[6]	65	o	op.adr[6]	105	b	tc.ad[11]	145	i	op.~rcvAdr
26	b	op.ad[5]	66	o	op.adr[5]	106	b	tc.ad[12]	146	i	op.~trAdr
27	b	op.ad[4]	67	o	op.adr[4]	107	b	tc.ad[13]	147	i	op.~rcvDat
28	b	op.ad[3]	68	o	op.adr[3]	108	b	tc.ad[14]	148	o	op.~rcvValid
29	b	op.ad[2]	69	o	op.adr[2]	109	b	tc.ad[15]	149	i	Vdd
30	i	Vss	70	i	Vss	110	i	Vss	150	i	Vss
31	b	op.ad[1]	71	o	Vdd	111	i	Vdd	151	i	op.~trDack
32	b	op.ad[0]	72	o	op.adr[1]	112	b	tc.ad[16]	152	o	op.~trValid
33	i	op.en.~ad	73	o	op.adr[0]	113	b	tc.ad[17]	153	i	op.~test
34	i	op.en.~io	74	o	tc.~int	114	b	tc.ad[18]	154	b	op.ad[31]
35	i	op.en.~adr	75	i	tc.~reset	115	b	tc.ad[19]	155	b	op.ad[30]
36	i	op.~rdy	76	i	tc.~sel	116	b	tc.ad[20]	156	b	op.ad[29]
37	o	op.~sel	77	o	tc.~rdy	117	b	tc.ad[21]	157	b	op.ad[28]
38	o	op.~write	78	i	tc.~write	118	b	tc.ad[22]	158	b	op.ad[27]
39	i	op.~conflict	79	i	tc.~ack	119	b	tc.ad[23]	159	b	op.ad[26]
40	i	Vdd	80	i	Vss	120	i	Vdd	160	i	Vss

Figure 16   TcIA Package Pin Diagram

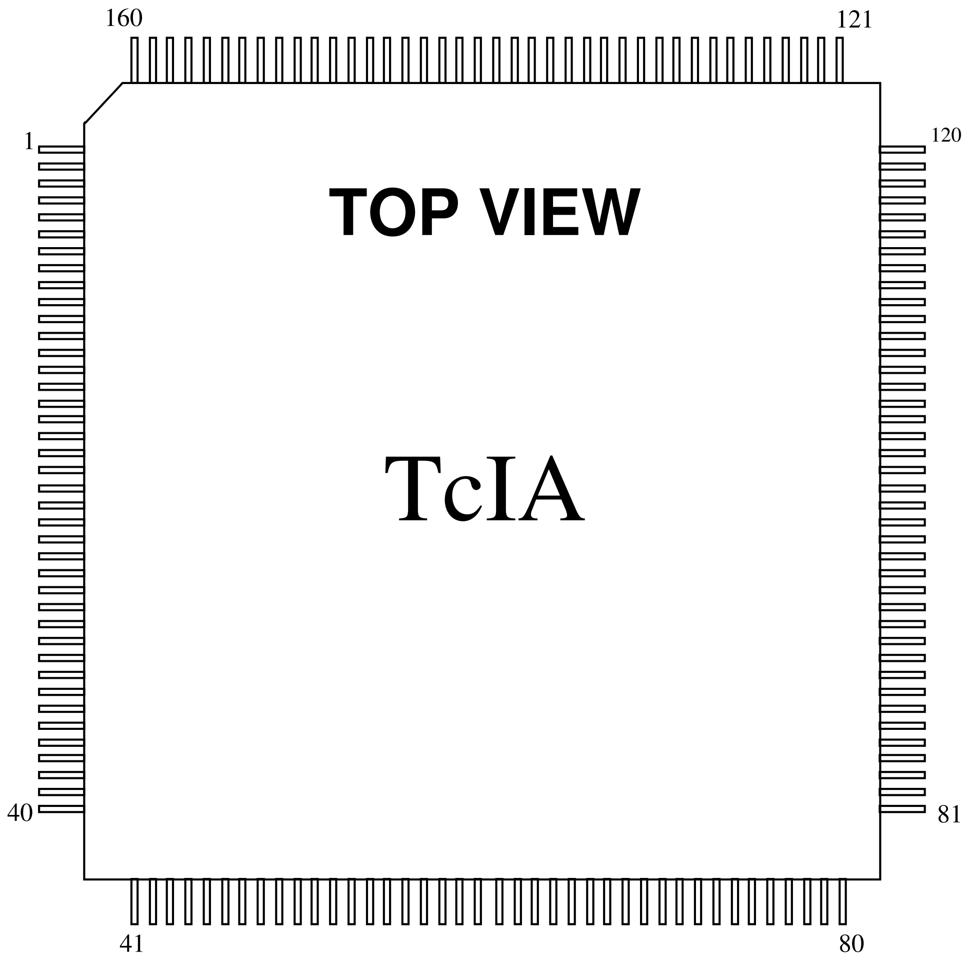


Table 2   Package Characteristics

Total package pins	160
Package type	PQFP
Body size	28mm x 28mm
Package height	4.07mm max
Pin pitch	0.65mm