

# OCP



International Partnership

# Open Core Protocol Specification

Release 2.2.1 Errata  
OCP-IP Confidential





# Open Core Protocol Specification

## Errata for Release 2.2.1

Document Revision - 0.5

© 2000-2010 OCP-IP Association, All Rights Reserved.

Open Core Protocol Specification 2.2.1 Errata  
Document Revision - 0.5

This document, including all software described in it, is furnished under the terms of the Open Core Protocol Specification License Agreement (the "License") and may only be used or copied in accordance with the terms of the License. The information in this document is a work in progress, jointly developed by the members of OCP-IP Association ("OCP-IP") and is furnished for informational use only.

The technology disclosed herein may be protected by one or more patents, copyrights, trademarks and/or trade secrets owned by or licensed to OCP-IP. OCP-IP reserves all rights with respect to such technology and related materials. Any use of the protected technology and related material beyond the terms of the License without the prior written consent of OCP-IP is prohibited.

This document contains material that is confidential to OCP-IP and its members and licensors. The user should assume that all materials contained and/or referenced in this document are confidential and proprietary unless otherwise indicated or apparent from the nature of such materials (for example, references to publicly available forms or documents). Disclosure or use of this document or any material contained herein, other than as expressly permitted, is prohibited without the prior written consent of OCP-IP or such other party that may grant permission to use its proprietary material.

The trademarks, logos, and service marks displayed in this document are the registered and unregistered trademarks of OCP-IP, its members and its licensors. The following trademarks of Sonics, Inc. have been licensed to OCP-IP: FastForward, CoreCreator, SiliconBackplane, SiliconBackplane Agent, InitiatorAgent Module, TargetAgent Module, ServiceAgent Module, SOCCreator, and Open Core Protocol.

The copyright and trademarks owned by OCP-IP, whether registered or unregistered, may not be used in connection with any product or service that is not owned, approved or distributed by OCP-IP, and may not be used in any manner that is likely to cause customer confusion or that disparages OCP-IP. Nothing contained in this document should be construed as granting by implication, estoppel, or otherwise, any license or right to use any copyright without the express written consent of OCP-IP, its licensors or a third party owner of any such trademark.

Printed in the United States of America.

EXCEPT AS OTHERWISE EXPRESSLY PROVIDED, THE OPEN CORE PROTOCOL (OCP) SPECIFICATION IS PROVIDED BY OCP-IP TO MEMBERS "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

OCP-IP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND OR NATURE WHATSOEVER (INCLUDING, WITHOUT LIMITATION, ANY DAMAGES ARISING FROM LOSS OF USE OR LOST BUSINESS, REVENUE, PROFITS, DATA OR GOODWILL) ARISING IN CONNECTION WITH ANY INFRINGEMENT CLAIMS BY THIRD PARTIES OR THE SPECIFICATION, WHETHER IN AN ACTION IN CONTRACT, TORT, STRICT LIABILITY, NEGLIGENCE, OR ANY OTHER THEORY, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.



# Errata

---

Type	Spec. Page #	Title	Errata Revision	Errata Page #
Specification Corrections	10	Tag Response Re-ordering Restrictions	0.5	5
	33	Module Configuration	0.5	6
	44-45	Transfer Effects	0.5	8
	47	Posting Semantics	0.5	9
	47	Transaction Completion, Transaction Commitment	0.5	10
	52-53	MReqRowLast, MDataRowLast, SRespRowLast	0.5	11
	53-54	Tag Ordering Restrictions	0.5	12
	56	Optional Command Support	0.5	13
	59	Request and Data Together	0.5	14
	59	Write Responses	0.5	15
	59	OCP Interface Interoperability	0.5	16
Specification Clarifications	53	Tag IDs	0.5	17
	48	Burst Definition	0.5	18
	51-52	Single Request, Multiple Data Bursts	0.5	19
Clarifications To Developers Guidelines	110	Non-Posted Write Timing	0.5	20
	115-116	Incorrect Signal Names in Response Accept Sequence	0.5	21
	170-171	Developer Guidelines for Write Semantics	0.5	22
	171-173	Developer Guidelines for Lazy Synchronization	0.5	25
	188-189	Simple Slave Feature Set	0.5	26
	191-192	High-Speed Slave Feature Set	0.5	27
	192-193	OCP Profile Optional Features	0.5	30
	196	OCP Profile Sequential Undefined Length Data Flow Profile Implementation Notes	0.5	31
198	OCP Register Access Profile Implementation Notes	0.5	32	

Type	Spec. Page #	Title	Errata Revision	Errata Page #
Clarifications To Developers Guidelines	199–200	OCP Block Data Flow Profile	0.5	33
	203–205	OCP X-Bus Packet Write Profile	0.5	34
	220	Developer Guidelines for Tags	0.5	35
Corrections to Compliance Rules	248	Modified Compliance Rule 1.2.29	0.5	36
	248	Deprecated Compliance Rule 1.2.30	0.5	37
	263	Deleted Compliance Rule 1.5.2	0.5	38
	273	Modified Compliance Rule 2.1.18	0.5	39
	284	Modified Compliance Rule 2.3.11	0.5	40
Clarifications to Compliance Checks	286	Deleted Compliance Rule 2.4.7	0.5	41
	118–119	Incrementing Imprecise Read Requires Deassertion of MBurstPrecise	0.5	42
	146	Incorrect Signal Names in Response Phase Guidelines	0.5	43
	162–163	Tag Interleaving for Packing Bursts	0.5	44
	300	New Table Note for transTypes	0.5	45
	309	STagInOrder Definition	0.5	46

## Introduction

### Purpose and Scope

This document lists errata for the *OCP Protocol Specification, Release 2.2.1* (OCP2) relative to the *OCP Protocol Specification, Release 3.0* (OCP3). It is intended for users of the OCP2 specification.

### Errata Format

Each errata is presented in two sections:

- *Location* indicates where the original material was located in the OCP2 specification.
- *Modified Text* presents the new material. The original material is shown in strikethrough; the new material is shown underlined.

The original document formatting has been preserved, as far as possible. However, the reader should be aware that the table and figure numbers shown in the errata are not guaranteed to match those in the *OCP Protocol Specification, Release 2.2.1*, particularly where new material has been introduced. Instead, the table numbers should be used as a guide—references to tables and figures are consistent within the body of each specific errata.

## Typographic Conventions

The following typographic conventions are used in this document:

<del>strikethrough</del>	Deleted text, i.e., the original text from OCP2 that has been superseded by material from OCP3.
<u>underlined</u>	New material from OCP3 (or OCP3 errata) which replaces material originally in OCP2 (or OCP3, as applicable).
monospaced	Code examples, or an OCP parameter name, e.g., datahandshake.

## Definitions

The following terms are used throughout this document:

OCP2	<i>Open Core Protocol Specification, Release 2.2.1</i>
OCP3	<i>Open Core Protocol Specification, Release 3.0</i>

## Errata Contributors

Our thanks to the following people who contributed errata:

Contributor	Errata Title
Abhishek Kothari, Cadence Design Systems, Inc.	"Tag Ordering Restrictions," on page 12
Neale Foulds, Duolog Technologies Ltd	"STagInOrder Definition," on page 46
John Ivie, OCP-IP	Parts of "Deprecated Compliance Rule 1.2.30," on page 37, "Tag Ordering Restrictions," on page 12, "Incrementing Imprecise Read Requires Deassertion of MBurstPrecise," on page 42, and others
Anita Vandanapu & Luc Ton, Sonics, Inc.	"Modified Compliance Rule 1.2.29," on page 36
Drew Wingard, Sonics, Inc.	"Deprecated Compliance Rule 1.2.30," on page 37
Harutyun Aslanyan & D. N. (Jay) Jayasimha, Sonics, Inc.	"Tag IDs," on page 17
James Aldis, Texas Instruments Incorporated	"Burst Definition," on page 18

## Revision History

---

<b>Revision</b>	<b>Notes</b>
0.1-0.4	OCP-IP internal errata revisions.
0.5	Simplified errata structure; removed TOC and added cross-reference table. First publication to OCP-IP members.

---

# Specification Corrections

## Tag Response Re-ordering Restrictions

### Location in OCP2 Specification

“Tags,” p10, 3rd paragraph.

### Modified Text

Out-of-order request and response delivery can also be enabled using multiple threads. The major differences between threads and tags are that threads can have independent flow control for each thread and have no ordering rules for transactions on different threads. Tags, on the other hand, exist within a single thread and are restricted to shared flow control. Tagged transactions ~~cannot be re-ordered with respect~~ to overlapping addresses have to be committed in order but their responses may be reordered if the transactions have different tag IDs (see Section 4.7.1 on page 57). Implementing independent flow control requires independent buffering for each thread, leading to more complex implementations. Tags enable lower overhead implementations for out-of-order return of responses at the expense of some concurrency.

# Module Configuration

## Location in OCP2 Specification

“Signal Directions,” p33.

## Modified Text

### Signal Directions

Figure 4 on page 43 summarizes all OCP signals. The direction of some signals (for example, MCmd) depends on whether the module acts as a master or slave, while the direction of other signals (for example, Control) depends on whether the module acts as a system or a core. The combination of these two choices provides four possible module configurations as shown in Table 14.

Table 14 Module Configuration Based on Signal Directions

	<del>Acts as Core or has No System Signals</del>	Acts as System
Acts as OCP Master	Master	System master
Acts as OCP Slave	Slave	System slave

~~For example, if a module acts as OCP master and also as system, it is designated a system master. There is also a monitor type that observes all OCP signals. The permitted connectivity between interface types is shown in Table 15.~~

For example, if a module acts as OCP master and also as system, it is designated a system master. In addition to the notion of modules, it is useful to introduce an “interface” type. All modules have interfaces. Also, there is a “monitor” interface type which observes all OCP signals. The permitted connectivity between interface types is shown in Table 15.

Table 15 Interface Types

Type	Connects To	Cannot Connect To
Master	System slave, Slave, Monitor	Master, System master
Slave	System master, Master, Monitor	Slave, System slave

<b>Type</b>	<b>Connects To</b>	<b>Cannot Connect To</b>
System master	Slave, <del>System slave</del> <u>Monitor</u>	Master, System Master, <u>System slave</u>
System slave	Master, <del>System master</del> <u>Monitor</u>	Slave, System slave, <u>System master</u>
Monitor	<del>Any except monitor</del> <u>Master, System master, Slave, System slave</u>	Monitor

The Clk, ~~and~~ EnableClk, and ConnectCap signals are special in that they are supplied by a third (external) entity that is neither of the two modules connected through the OCP interface.

# Transfer Effects

## Location in OCP2 Specification

Section inserted into “Sideband and Test Signals,” pp44–45, between “Reset” and “Interrupt, Error, and Core Flags.” The material in Section 4.4.3 is also provided in “Transfer Effects,” on page 8.

## Modified Text

A successful transfer is one that completes without error. For write-type requests without responses, there is no in-band error indication. For all other requests, a non-ERR response (that is, a DVA or FAIL response) indicates a successful transfer. The FAIL response is legal only for WriteConditional commands<sup>1</sup>. This section defines the effect that a successful transfer has on a slave. The request acts on the addressed location, where the term address refers to the combination of MAddr, MAddrSpace, and ~~MByteEnable~~MByteEn (or MDataByteEn, if applicable). Two addresses are said to match if they are identical in all components. Two addresses are said to conflict, if the mutual exclusion (lock or monitor) logic is built to alias the two addresses into the same mutual exclusion unit. The transfer effects of each command are:

...

### ReadEx

Returns the latest value of the addressed location on the SData field. Sets a lock for the initiating thread on that location. The next request on the thread that issued a ReadEx must be a Write or WriteNonPost to the matching address. Requests from other threads to a conflicting address that is locked are ~~blocked from proceeding~~ not committed until the lock is released. If the ReadEx request returns an ERR response, it is slave specific whether the lock is actually set or not. Refer to Section 4.4.3 on page 51 for details.

---

<sup>1</sup> For all commands except those following a posted write model, a DVA response also indicates that the transfer is committed.

# Posting Semantics

## Location in OCP2 Specification

“Posting Semantics,” page 47.

## Modified Text

### Posting Semantics

Table 22 below summarizes the posting semantics for write-type commands. WRNP and WRC are always non-posted; a DVA response indicates that the write was committed and an ERR response indicates that the write was not committed (an error occurred along the write path). WR and BCST commands may follow a posted or non-posted model. If the OCP interface is configured to not send a completion response (`writeresp_enable` is set to 0), the write is posted upon command acceptance and is considered to be posted early. When `writeresp_enable` is set to 1, the system designer decides where along the write path the posting point is. The completion response (either DVA or ERR) is then generated from the posting point. The non-posted model has the same semantics as WRNP.

Table 22 *Write Posting Semantics*

Write Command	writeresp_enable	
	0	1
WR, BCST	Posted early	Posted or Non-posted
WRNP, WRC	Non-posted	Non-posted

~~The OCP includes a basic Write command. Typically, a system designer decides what write completion model to assign to a core's write commands. If the OCP is configured to not respond to write-type commands (`writeresp_enable` set to 0), a posted write completion model is assumed. It is also possible to achieve a non-posted model by not accepting the command until the write completes, but this de-pipelines the OCP interface and is not recommended. If the OCP is configured to respond to write-type commands (`writeresp_enable` set to 1), either a posted or a non-posted write completion model can be used. For cores that need to determine on a per-transfer basis whether a write is to be treated as posted or non-posted, the OCP provides the WriteNonPost command.~~

# Transaction Completion, Transaction Commitment

## Location in OCP2 Specification

New material immediately following “Posting Semantics” on page 47.

## Modified Text

### Transaction Completion, Transaction Commitment

It is useful to distinguish between “commitment” of a transaction and the “completion” of a transaction. A transaction is “committed” when the transaction finishes or completes at the final target. In cases where the completion response is sent by the slave or target after commitment, the completion response is a guarantee of transaction commitment. With a posted write model, however, the posted write completion response may be received at the master before the write commitment.

Thus, the OCP completion response implies commitment for all transactions except writes with a posted write model (e.g., WR or BCST with early posting). For posted writes, there is no relationship between commitment and completion.

# MReqRowLast, MDataRowLast, SRespRowLast

## Location in OCP2 Specification

“MReqRowLast, MDataRowLast, SRespRowLast,” pp52–53.

## Modified Text

For all datahandshake phases in a non-BLCK burst except the last one, MDataRowLast is 0. MDataRowLast is 0 for every datahandshake phase in a BLCK burst sequence that is not an integer multiple of MBurstLength. MDataRowLast is 1 for:

- The last datahandshake phase in a burst including the only datahandshake phase of a single word request.
- Every datahandshake phase in a BLCK burst sequence that is an integer multiple of MBurstLength.

For all response phases in a non-BLCK burst except the last one, SRespRowLast is 0. *(Remaining text from specification omitted.)*

# Tag Ordering Restrictions

## Location in OCP2 Specification

“Ordering Restrictions,” pp53–54.

## Modified Text

The sequence of requests by the master determines the initial ordering of tagged transactions. For tagged write transactions with datahandshake enabled, the datahandshake phase must observe the same order as the request phase. The master cannot interleave requests or datahandshake phases ~~with~~ from different tags belonging to the same thread within a transaction.

Tag values can be re-used for multiple outstanding transactions. Slaves are responsible for committing write data and sending responses for multiple transactions that have the same tag, in order.

~~Responses with different tags can be returned in any order for all commands that have responses. Responses with the same tag must remain in order with respect to one another.~~ Responses that are part of the same transaction must stay together, up to the tag\_interleave\_size (see “Burst Interleaving with Tags” on page 58). Beyond the tag\_interleave\_size, responses with different tags can be interleaved. This allows for blocks of responses corresponding to tag\_interleave\_size from one burst to be interleaved with blocks of responses from other bursts.

~~Responses to requests that target overlapping addresses (as determined by MAddrSpace, MAddr, and MByteEn) are never re-ordered with respect to one another. Similarly, responses to requests that are issued with MTagInOrder asserted are also never reordered with respect to one another.~~ Responses with different tags can be returned in any order for all commands that have responses. Responses with the same tag must remain in order with respect to one another. Responses to requests that are issued with MTagInOrder asserted are also never reordered with respect to one another. The value returned on STagInOrder with the slave’s response must match the value provided on MTagInOrder with the master’s request.

~~When tags are in use and writes are not posted, write response order indicates the order that writes are committed by the slave. When writes are posted, the response to a tagged read can only be guaranteed to be ordered with respect to writes to overlapping addresses. This is different than non-tagged versions of OCP interfaces, where read responses are sometimes used to ensure completion of all earlier posted writes.~~

Commitment of transactions with overlapping addresses (as determined by MAddrSpace, MAddr, MByteEn [or MDataByteEn, if applicable]) on different (or the same) tags within a thread is always in order. Note, however, that the completion responses for such transactions with different tag ids may be reordered.

# Optional Command Support

## Location in OCP2 Specification

“Optional Commands,” p56.

## Modified Text

A master with one of these options set to 0 must not generate the corresponding command.

- A slave with one of these options set to 0 cannot service the corresponding command.
- At least one of the command enables must be set to 1.
- ~~writenonpost\_enable can be set to 1 only if writeresp\_enable is 1.~~
- ~~rdlwre\_enable can be set to 1 only if writeresp\_enable is set to 1.~~
- If any read-type command is enabled, or if WRNP is enabled, or if writeresp\_enable is set to 1, resp must be set to 1.
- If readex\_enable is set to 1, write\_enable or writenonpost\_enable must be set to 1.

# Request and Data Together

## Location in OCP2 Specification

“Request and Data Together,” p59.

## Modified Text

### Request and Data Together

While datahandshake is required for OCP interfaces that are capable of communicating single request / multiple data bursts, a fully separated datahandshake may be overkill for some cores. The parameter reqdata\_together is used to specify that the request and datahandshake phases of the first transfer in a single request, multiple data (SRMD) write-type burst begin and end together.

A master with reqdata\_together set to 1 must present the request and first write data word in the same cycle and can expect that the slave will accept them together. If sthreadbusy\_exact and sdatathreadbusy\_exact are both set to 1 and sthreadbusy\_pipelined and sdatathreadbusy\_pipelined are both set to 0, then a request and first write data can be presented only when both SThreadBusy and SDataThreadBusy for the corresponding thread are 0 on that cycle. If sthreadbusy\_exact and sdatathreadbusy\_exact are both set to 1 and sthreadbusy\_pipelined and sdatathreadbusy\_pipelined are both set to 1, then a request and first write data can be presented only on cycle i when both SThreadBusy and SDataThreadBusy for the corresponding thread are 0 during the prior cycle, i.e., cycle (i-1).

~~A master with reqdata\_together set to 1 must present the request and first write data word in the same cycle and can expect that the slave will accept them together. If sthreadbusy\_exact and sdatathreadbusy\_exact are both set to 1, this implies that a request and first write data can be presented only when both SThreadBusy and SDataThreadBusy for the corresponding thread are 0. A slave with reqdata\_together set to 1 must accept the request and first write data word in the same cycle and can expect that they will be presented together.~~

# Write Responses

## Location in OCP2 Specification

“Write Responses,” p59.

## Modified Text

~~To configure the OCP to include response phases for write-type requests, use the `writeresp_enable` parameter.~~

- ~~• If responses are not enabled on writes (`writeresp_enable` set to 0), then all write commands complete on command acceptance, and the `WriteNonPost`, `WriteConditional`, and `ReadLinked` commands are not allowed.~~
- ~~• If responses are enabled (`writeresp_enable` set to 1), writes may follow either a posted or non-posted model depending on when the response is returned. For this case, `resp` must be set to 1.~~
- Writes which follow a non-posted model, i.e., WRNP and WRC, always have a write response. For this case, `resp` must be set to 1.
- For writes which follow a posted model, i.e., WR and BCST: if responses are not enabled on writes (`writeresp_enable` set to 0), then they complete on command acceptance.

# OCP Interface Interoperability

## Location in OCP2 Specification

“OCP Interface Interoperability,” p59.

## Modified Text

3. At the phase level the two interfaces are interoperable if:

- Configuration of the datahandshake parameter is identical between master and slave.
- Configuration of the writersp\_enable parameter is identical between master and slave. Otherwise, the master only issues the write commands WriteNonPost and WriteConditional.
- Configuration of the reqdata\_together parameter is identical between master and slave.

4. At the signal level, two interfaces are interoperable if:

- data\_width is identical for master and slave, or if one or both data\_width configurations are not a power-of-two, if that data\_width rounded up to the next power-of-two is identical for master and slave.
- The master and slave both have mreset or sreset set to 1.
- If the master has mreset set to 1, the slave has mreset set to 1.
- If the slave has sreset set to 1, the master has sreset set to 1.
- The value of connection is identical for master and slave, or if ConnectCap is tied off to logic 0 on the side with connection set to 1.
- Both master and slave have tags set to >1 or if only one core's tags parameter is set to 1, the other core behaves as though MTagInOrder were asserted for every request.

# Specification Clarifications

## Tag IDs

### Location in OCP2 Specification

In “Tags”, on p53.

### Modified Text

Tags allow out-of-order return of responses and out-of-order commit of write data.

A master drives a tag on MTagID during the request phase. The value of the tag is determined by the master and may or may not convey meaning beyond ordering to the slave. For write transactions with data handshake enabled, the master repeats the same tag on MDataTagID during the datahandshake phase. For read transactions and writes with responses the slave returns the tag of the corresponding request on STagID while supplying the response. The same tag must be used for an entire transaction.

Note that a RdEx command and the associated WR or WRNP commands need not have the same tag IDs since they are separate transactions. Similarly, an RDL command and the associated WRC command need not have the same tag IDs since they are separate transactions.

# Burst Definition

## Location in OCP2 Specification

Last paragraph of “Burst Definition” on page 48.

## Modified Text

~~A single (non-burst) request on an OCP interface with burst support is encoded as a request with any legal burst address sequence and a burst length of 1.~~

~~The ReadEx, ReadLinked, and WriteConditional commands can not be used as part of a burst. The unlocking Write or WriteNonPost command associated with a ReadEx command also can not be used as part of a burst.~~

A single word request on an OCP interface with burst support is encoded either a) as any non-BLCK burst address burst sequence with burstlength of 1 or b) as a BLCK burst request with burstlength of 1 and blockheight of 1.

The ReadEx, ReadLinked, and WriteConditional commands can only be used as part of a single-word request. The unlocking Write or WriteNonPost command associated with a ReadEx command can only be used as part of a single word request.

# Single Request, Multiple Data Bursts

## Location in OCP2 Specification

“Single request, Multiple Data Bursts (Packets),” pp51–52.

## Modified Text

Note that the deleted text in the footnote applies to both the OCP2 and OCP3 specification.

When MBurstSingleReq is set to 1, write type transfers have MBurstLength \* height datahandshake phases ~~and a single response phase (if writeresp\_enable=1)~~ per request<sup>1</sup>; while read-type transfers have MBurstLength \* height response phases per request as shown in Table 21 on page 42. The height is MBlockHeight for BLCK address sequences, and 1 for all others.

---

<sup>1</sup> Additionally, there is a single response phase for WRNP and WRC write types while the WR and BCST types have this phase only if writeresp\_enable is set to 1. ~~Note that WRC write type is not allowed in a burst.~~

# Clarifications To Developers Guidelines

## Non-Posted Write Timing

### Location in OCP2 Specification

“Non-Posted Writes,” p110.

### Modified Text

Figure 25 repeats the previous example for a non-posted write transaction. In this case the response must be returned to the master once the write operation ~~occurs~~commits. There is no difference in the command acceptance, but the response may be significantly delayed. If this scheme is used for all posting-sensitive transactions, the result is decreased data throughput but higher system reliability.

# Incorrect Signal Names in Response Accept Sequence

## Location in OCP2 Specification

Step E of the sequence described in “Response Accept,” pp115–116.

## Modified Text

Signal MRespAccept incorrectly written as RespAccept. See correction in Step E, below.

### Sequence

- A. The master starts a read request by driving RD on MCmd and a valid address on MAddr. The slave asserts SCmdAccept immediately, and it drives DVA on SResp and the read data on SData as soon as it sees the read request. The master is not ready to receive the response for the request it just issued, so it deasserts MRespAccept.
- B. Since SCmdAccept is asserted, the request phase ends. The master continues to deassert MRespAccept, however. The slave holds SResp and SData steady.
- C. The master starts a second read request and is ready for the response from its first request, so it asserts MRespAccept. This corresponds to a response accept latency of 2.
- D. Since SCmdAccept is asserted, the request phase ends. The master captures the data for the first read from the slave. Since MRespAccept is asserted, the response phase ends. The slave is not ready to respond to the second read, so it drives NULL on SResp.
- E. The slave responds to the second read by driving DVA on SResp and the read data on SData. The master is not ready for the response, however, so it deasserts MRespAccept.
- F. The master asserts MRespAccept, for a response accept latency of 1.
- G. The master captures the data for the second read from the slave. Since MRespAccept is asserted, the response phase ends.

# Developer Guidelines for Write Semantics

## Location in OCP2 Specification

“Write Semantics,” pp170–171.

## Modified Text

### Write Semantics

As detailed in “Posting Semantics,” on page 47, OCP ~~semantics specify whether a write is~~ writes support posted and non-posted ~~or not~~ models<sup>1</sup>. A non-posted write model is preferred whenever the originator of the request must be aware of the completion of its write command, i.e., command commitment. An example is clearing an interrupt in a peripheral module using a write command. In that case the processor must be sure that the interrupt line has been effectively released before it can acknowledge the interrupt service in the chip-level interrupt controller.

The concept of the posting semantics diverges from the concept of responses on writes in the following ways:

- A write with a response could have posted semantics in a system (so that a response is returned immediately) or it could have non-posted semantics (so that a response is returned only after the write is completed at the final target, i.e., the command is committed).
- A write without a response normally has posted semantics and carries forward the *OCP 1.0 Specification* for backward compatibility.
- A write without a response can be assigned non-posted semantics by not accepting the command until the write has completed, but this is not recommended since it de-pipelines the OCP interface. Since posting makes sense at a system level, adopting a delayed-SCmdAccept scheme can only be efficient locally, with no guarantee of the non-posting semantics at the system level.

~~The writeresp\_enable parameter controls whether writes have responses. writenonpost\_enable controls whether the interface supports the WriteNonPost command, giving the initiator core the ability to launch two different types of writes. Table 28 summarizes the choices.~~

---

<sup>1</sup> Errata apply: see Section on page 9 of this document.

Table 28 ~~Write Parameters~~

<b>writenonpost_enable</b>		
<b>writeresp_enable</b>	<b>0</b>	<b>1</b>
0	Simple posted write model, corresponds to OCP 1.0 Spec	Illegal
1	Initiator core has one flavor of writes (WR), system integrator decides where to post the write requests	Initiator core has two flavors of writes (WR and WRNP), system integrator decides where to post the two different write requests

The `writeresp_enable` parameter controls whether the write-type commands WR and BCST have responses. The write-type commands WRNP and WRC, which are non-posted, always have responses. Table 28 summarizes the behavior with respect to the `writeresp_enable` parameter.

Table 28 Write Command Response Behavior

<b>writeresp_enable</b>	
<b>0</b>	<b>1</b>
WR, BCST (without response) WRNP, WRC (with response)	WR, BCST, WRNP, WRC (with response)

Note that in Table 28, WR and WRNP are the general-purpose write commands; WRC is always associated with an RLC command.

Use of the Broadcast command must be limited to a specific category of designs (some interconnect designs may benefit from simultaneous update through distributed registers). It is not expected that standard cores will support the Broadcast command.

By separating whether writes have responses (`writeresp_enable`) from whether the core has control over where the responses are generated (`writenonpost_enable`), the OCP specification provides the following features:

- The simple, posted model remains intact. The simplest cores only implement WR, and need not worry about write responses.
- Cores that can generate or use write responses should enable write responses, providing support for in-band error reporting on write commands. The read and write state machines are duplicated from the standpoint of flow control, producing a simpler design. Such cores would normally only implement the WR command. In this case, the system integrator is in control of where in the write path the write response is generated, allowing a choice of the level of posting based upon performance and coherence trade-offs.

- Cores that can distinguish between performance and coherence (really only CPUs and bridges) can enable WRNP to implement dynamic choice between WR and WRNP. The additional signaling gives the system integrator the dynamic information needed to choose the posting point as the CPU requests. The only practical difference between WR and WRNP at the protocol level is the expected latency between request and response. This permits some embedded CPUs to achieve high performance—particularly as interconnects become pipelined and posting buffers are needed.

~~When the `writeresp_enable` parameter is enabled, responses are required for any write command issued on the OCP, including WR, WRNP, but also BCST and WRC.~~

~~Use of the Broadcast command must be limited to specific category of designs (some interconnect designs may benefit from simultaneous update through distributed registers). It is not expected that standard cores support the command.~~

# Developer Guidelines for Lazy Synchronization

## Location in OCP2 Specification

“Lazy Synchronization,” pp171–173.

## Modified Text

Because the Write that clears the lock must immediately follow the ReadEx (on the same thread), only a limited number of operations can be performed by a processor between RDEX and WR. ~~Since competing~~ Competing requests to the locked location are ~~also blocked from proceeding~~ not committed until the lock is released. It is highly recommended that this requirement be enforced at the final target with non-blocking flow control for multithreaded applications. Otherwise, for example, if the logic is implemented on the master’s side in an interconnect, you could lock part of the interconnect could be locked for the duration of the RDEX-WR or RDEX-WRNP pair. Such This mechanism of using a ~~mechanism~~ RDEX-write pair, often referred to as *locked synchronization*, is efficient for handling exclusive accesses to a shared resource, but can result in a significant performance loss when used extensively.

# Simple Slave Feature Set

## Location in OCP2 Specification

“Feature Set,” pp188–189.

## Modified Text

### Feature Set

The simple slave profile supports only single accesses, so no burst-related signals are used. The accepted commands are IDLE, RD, WR, and WRNP. ~~Writes have the~~ Posted writes only return a response ~~indicated by the~~ if ~~writeresp\_enable~~ ~~parameter as shown in Table 69~~ is enabled; ~~non-posted writes always return a response.~~ For non-posted writes the response must issue from the receiving slave. ~~In the~~ When responses are required for posted ~~easy~~ writes (~~writeresp\_enable~~ is enabled), any component between the master and the slave ~~(e.g., an interconnect)~~ can provide the response. This process represents a trade-off between the potential speed improvements of posted writes and the more reliable write completion and error tracking of the non-posted writes. The allowed responses for SResp are NULL, DVA, and ERR.

If present, the read and write data signals SData and MData must have the same width. To simplify the interface, the force\_aligned parameter is set to 1, limiting byte enable patterns on the MByteEn signal to power-of-two in size and aligned to that size. A byte enable pattern of all 0s is legal. This means that the byte enable patterns generated by simple slave profile masters are force-aligned. In addition, slaves using this profile can assume that the incoming byte enable patterns are force-aligned. The size of the MByteEn is 2 bits when data\_width is 16, 4 bits for a data\_width of 32, and 8 bits for a data\_width of 64. The ~~allowable~~ ~~allowed~~ MByteEn values for ~~data width a~~ ~~data\_width of 32~~ are ~~indicated by the shaded rows in Table 70~~ indicated by the shaded rows in Table 68.

...

Slaves must be able to support the entire feature set defined in this profile. Masters do not need to be able to issue all the commands since only one WR, RD or WRNP is required. If masters only issue read commands (~~write\_enable~~ and ~~writenonpost\_enable~~ parameters set to 0), they can omit the MData signal and responses to writes (~~writeresp\_enable~~ has to be 0 in the master parameter list). If a master issues only write commands, the SData signal can be omitted. Using these options does not compromise interoperability with Simple Slave Profile slaves.

# High-Speed Slave Feature Set

## Location in OCP2 Specification

“Feature Set,” pp191–192.

## Modified Text

### High Speed Profile

The high-speed profile is intended for subsystem components that require high throughput. ~~Processors~~, for example, processors, co-processors, DMA engines, and memory ~~controllers are typical examples~~ controllers. The signals for this interface are listed in Table 37. This profile adds the burst-related signals MBurstSeq, MBurstLength, MReqLast, and SRespLast to improve throughput and the MRespAccept signal to enable response-phase flow control. All signals with an M prefix are driven by the master while all signals with an S prefix are driven by the slave, with the possible exception of MReset\_n and SReset\_n depending on the system.

Table 37 High Speed Profile Signals and Parameters

Signal*	Enabling Parameter	Width Parameter	Usage
Clk	Required	Fixed	Clock input (page 14)
<i>(Table rows deleted for brevity)</i>			
<b>Additional parameters</b>			
	endian little, big, both, neutral		Modules are required to state their endianness. Little endian is recommended (page 47)
	force_aligned		Byte enables are power-of-two in size and aligned to that size (page 56)
	tags=1 threads=1		No tags or threads are used but the parameter cannot be 0 (pages 53 and 54)
	writeresp_enable		<del>Writes (posted and non-posted)</del> Posted writes expect a response. Can be 0 for a master <del>that only issues read operations</del> that only issues read operations or non-posted write.

\* See “Optional Features” on page 192 for additional signals

For this profile the burstprecise parameter is zero, so the MBurstPrecise signal is not used in the interface and all bursts are precise.

## Feature Set

While the ~~high-speed-profile~~ High-Speed Profile shares many features with the ~~simple-slave-profile~~ Simple Slave Profile, the force aligned requirement is eliminated since the attached modules can afford more complex interface logic. Two additional features are multiple-request multiple-data (MRMD) type bursts and the response accept signal (MRespAccept). MRespAccept ~~aids-grants~~ capability to the master to block the response flow from the ~~slaves by-indicating when their~~ slave if it cannot process new responses ~~anymore~~. ~~have-been-received-by-the-master~~.

Bursts offer higher transfer efficiencies when compared to single transfers by introducing atomicity for multiple associated requests. This atomicity ensures that requests in the same transaction, i.e. with spatial locality, are issued back-to-back. This behavior is crucial when accessing an SDRAM memory to attain high throughputs. Since the length of the burst is transmitted in the beginning (with MBurstLength) and all bursts are precise (MBurstLength value remains constant over the whole burst), further optimizations are possible in the scheduling and arbitration processes. The address sequence of the burst is provided by MBurstSeq. Allowed sequences are:

### Incrementing (INCR)

The address is incremented with the OCP word size for each transfer

### Wrapping (WRAP)

Like incrementing burst but it does not have to start from the first address and the address wraps at the address boundary defined by MBurstLength\*OCP word size

### Streaming (STRM)

The address remains constant over the whole burst

MBurstSeq and MBurstLength provide the information needed to generate and receive MRMD bursts. Additional information can help simplify interconnect and slave design. OCP offers framing signals that can be used by the master to identify the last request (MReqLast) and by the slave to identify the last response (SRespLast) in the burst. These additional framing signals are also part of this profile.

Slaves must support the entire feature set defined in this profile. Masters need not be able to issue all the commands since only one WR, RD or WRNP is required. If masters only issue read commands (`write_enable` and `writenonpost_enable` parameters set to 0), they can omit the MData signal and responses to writes (`writeresp_enable` has to be 0 in the master parameter list). If a master issues only write commands, SData can be omitted. Masters must support at least one of the burst addressing modes.

## Interoperability Issues

The force aligned requirement of the simple slave profile and the MRespAccept signal and burst features of the high-speed profile present some interoperability problems between the interfaces. A bridge or some other component linking simple slave and high-speed profile interfaces needs to break transactions with misaligned byte enables (coming from the high-speed profile to the simple slave profile) into transactions with legal byte enable patterns. A similar process needs to be followed for burst accesses coming

from a high-speed profile master to a simple slave profile slave. The MRMD nature of high-speed profile bursts means that the bridge can ignore the burst related signals on the request side, but needs to generate an SRespLast. ~~In addition, if a master uses the high speed profile, the bridge may need to buffer any simple slave profile slave responses. If the master uses the simple slave profile and the slave the high speed profile, the bridge can tie the MRespAccept signal as asserted.~~ In addition, if the Master is High-Speed Profile, the bridge may need to limit the number of outstanding requests on the interface or to buffer Simple Slave Profile slave responses in case the master de-asserts MRespAccept. If Master is Simple Slave Profile, and the slave is High-Speed Profile, the bridge may tie MRespAccept as asserted.

# OCP Profile Optional Features

## Location in OCP2 Specification

“Optional Features” on pp192–193.

## Modified Text

Tags and threads are identified using ID-signals. MTagID (request tag) and STagID (response tag) are present in the interface if the tags parameter is greater than 1. In addition, the advanced high-speed profile needs MDataTagID if tags are enabled because of datahandshaking. Similar signals exist for threads (MThreadID, SThreadID, and MDataThreadID). They are enabled if threads parameter is greater than 1 and the last one if datahandshake is used (datahandshake = 1).

MTagInOrder, STagInOrder, MConnID, MThreadBusy, SDataThreadBusy, and SThreadBusy are also optional signals in the high-speed ~~profile~~ and advanced high-speed profiles. For systems that use OCP threads and allow interleaving within request phases and data handshake phases, the use of threadbusy signals is recommend.

# OCP Profile Sequential Undefined Length Data Flow Profile Implementation Notes

## Location in OCP2 Specification

Sequential Undefined Length Data Flow Profile, “Implementation Notes” on page 196.

## Modified Text

- To implement a producer/consumer synchronization scheme (typically the case when data written by the IP core to shared memory is read by another core in the system), the IP core should issue a synchronization request (for instance through an OCP flag interface) only after receiving ~~notification~~ a response that the last write transaction ~~is complete~~ has committed. To accomplish this step, perform all write transactions up to the last one as posted writes. Make the final write transaction a non-posted write. This will lead to reception of a response once the non-posted write transaction has committed, i.e., completed at the final destination.

# OCP Register Access Profile Implementation Notes

## Location in OCP2 Specification

Register Access Profile, Table 40, “Register Access Parameter Settings,” page 198.

## Modified Text

In the final row of the table, the comment for parameter `writeresp_enable` should read “Precise write responses needed for posted writes”.

# OCP Block Data Flow Profile

## Location in OCP2 Specification

Block Data Flow Profile, pp199–200.

## Modified Text

In the final row of Table 41, “Block Data Flow Parameter Settings,” the comment for parameter `writeresp_enable` should read “Needed for posted writes”.

## Implementation Notes

When implementing this profile, consider the following suggestions:

- Start read transactions as early as possible to minimize read latency behind ongoing transactions.
- To implement a synchronization scheme (typically the case when data written by the IP core to shared memory is read by another core in the system), the IP core should issue a synchronization request (for instance through an OCP flag interface) only after receiving ~~notification~~ a response that the last write transaction ~~is complete~~ has committed. To accomplish this step, perform all write transactions up to the last one as posted writes. Make the final write transaction a non-posted write. This will lead to reception of a response once the non-posted write transaction has ~~completed~~ committed, i.e., completed at the final destination.

# OCP X-Bus Packet Write Profile

## Location in OCP2 Specification

X-Bus Packet Write Profile, pp203–205.

## Modified Text

In the final row of Table 43, “Block Data Flow Parameter Settings,” the comment for parameter `writeresp_enable` should read “Needed for posted writes”.

# Developer Guidelines for Tags

## Location in OCP2 Specification

“Tags,” p220.

## Modified Text

Tags are labels that associate requests with responses in order to enable out-of-order return of responses. In the face of different latencies for different requests (for instance, DRAM controllers, or multiple heterogeneous targets), allowing the out-of-order delivery of responses can enable higher performance. Responses are returned in the order they are produced rather than having to buffer and re-order them to satisfy strict response order requirements. Tagged transactions to overlapping addresses have to be committed in order but their responses may be reordered if the transactions have different tag IDs (see Section 4.7.1 on page 57). As is the case for threads, to make use of tags, the master will normally need a buffer.

# Corrections to Compliance Rules

## Modified Compliance Rule 1.2.29

### Location in OCP2 Specification

Compliance rule 1.2.29, p248.

### Modified Text

#### Rule 1.2.29 response reorder STagID tag interleave size

When `tags > 1` and `tag_interleave_size > 0` the slave must ensure that responses associated with packing burst sequences stay together up to the `tag_interleave_size`. When `tags > 1` and `tag_interleave_size == 0` no interleaving of responses between any packing burst sequences with different tags is allowed.

For packing bursts, when `tags > 1` and `tag_interleave_size > 0` the number of bursts that can stay together depend on the block alignment. A suggested implementation of this rule can be found in “Tags,” on page 162.

<b>Protocol hierarchy</b>	Response
<b>Signal group</b>	Dataflow - tag extensions
<b>Critical signals</b>	STagID
<b>Assertion type</b>	Reorder
<b>References</b>	OCP2: “Ordering Restrictions” on page 53 “Burst Interleaving with Tags” on page 58 OCP3: Section 4.7.1 on page 57 Section 4.9.1.7 on page 62

## **Deprecated Compliance Rule 1.2.30**

### **Location in OCP2 Specification**

Compliance rule 1.2.30, p248.

### **Modified Text**

Compliance rule 1.2.30 is deprecated.

# Deleted Compliance Rule 1.5.2

## Location in OCP2 Specification

Compliance rule 1.5.2, p263.

## Modified Text

### ~~Rule 1.5.2 rdex\_lock\_release\_no\_WR/WRNP~~

~~If a ReadEx is issued on an address on a particular thread, no other request with the same address can be issued on any other thread until the ReadEx is unlocked.~~

~~The command following the ReadEx on the same thread must be a write command (WR or WRNP). This command unlocks the ReadEx.~~

<del>Protocol hierarchy</del>	<del>ReadEx</del>
<del>Signal group</del>	<del>Dataflow—basic signals</del>
<del>Critical signals</del>	<del>MCmd</del>
<del>Assertion type</del>	<del>Ordering</del>
<del>References</del>	<del>Section 4.4 on page 49 Section 4.6 on page 52</del>

# Modified Compliance Rule 2.1.18

## Location in OCP2 Specification

Configuration check 2.1.18, p273.

## Modified Text

### Rule 2.1.18

**request\_cfg\_burstseq\_<type>\_enable\_burstseq\_enable\_enable\_burstseq**

burstseq\_<type>\_enable can only be enabled if ~~at least one burst sequence~~ burstseq is enabled. ~~burstseq\_<type>\_enable\_burstseq~~ must be enabled if ~~2~~ two or more burst sequences (specified by burstseq\_<type>\_enable) are enabled.

<b>Protocol hierarchy</b>	Request
<b>Critical parameters</b>	burstseq, burstseq_<type>_enable
<b>Reference</b>	“Open Burst Sequences” on page 55

# Modified Compliance Rule 2.3.11

## Location in OCP2 Specification

Configuration check 2.3.11, page 284.

## Modified Text

~~Rule 2.3.11 response\_cfg\_<cmd\_enable>\_enable\_writeresp\_enable  
writenonpost\_enable and rdlwre\_enable are only enabled if  
writeresp\_enable is enabled.~~

<b>Protocol hierarchy</b>	Response
<b>Critical parameters</b>	writenonpost_enable, writeresp_enable, rdlwre_enable
<b>Reference</b>	"Optional Commands" on page 55

# Deleted Compliance Rule 2.4.7

## Location in OCP2 Specification

Configuration check 2.4.7, p286.

## Modified Text

### ~~Rule 2.4.7 sideband\_cfg\_statusbusy\_enable\_status~~

~~statusbusy can only be enabled if status is enabled.~~

<del>Protocol hierarchy</del>	Sideband
<del>Critical parameters</del>	status, statusbusy
<del>Reference</del>	Footnotes on page 32

# Clarifications to Compliance Checks

## Incrementing Imprecise Read Requires Deassertion of MBurstPrecise

### Location in OCP2 Specification

Step A of the sequence described in “Incrementing Imprecise Read,” pp118–119.

### Modified Text

Signal MBurstPrecise should be de-asserted at the start of the read request. See correction to Step A, below.

### Sequence

- A. The master starts a read request by driving RD on MCmd, a valid address on MAddr, three on MBurstLength, INCR on MBurstSeq, and deasserts MBurstPrecise. The burst length is the best guess of the master at this point. MBurstSeq and MBurstPrecise are kept constant during the burst. MReqLast must be deasserted until the last request in the burst. The slave is ready to accept any request, so it asserts SCmdAccept.

# Incorrect Signal Names in Response Phase Guidelines

## Location in OCP2 Specification

Step E of the sequence described in “Response Phase,” p146. (Text shown below is from the OCP3 specification.)

## Modified Text

Signal SResp originally incorrectly written as MResp. See correction, below.

### 12.1.2.2 Response Phase

The response phase begins when the slave drives SResp to a value other than NULL. When SResp != NULL, SResp is referred to as asserted. All of the other response phase outputs of the slave must become valid during the same OCP clock cycle as SResp asserted, and be held steady until the response phase ends. The response phase ends when MRespAccept is sampled asserted (true) by the rising edge of the OCP clock; if MRespAccept is not configured into a particular OCP, MRespAccept is assumed to be always asserted (that is, the response phase always ends in the same cycle it begins). If present, the master can assert MRespAccept in the same cycle that ~~M~~SResp is asserted, or it may stay negated for several OCP clock cycles. The latter choice allows the master to force the slave to hold its response phase outputs so the master can finish the transfer without latching the data signals.

# Tag Interleaving for Packing Bursts

## Location in OCP2 Specification

Immediately before the last paragraph of “Tags,” pages 162–163 of Chapter 9, “Developers Guidelines.”

## Modified Text

When MTagInOrder is asserted any MTagID and MDataTagID values are “don’t care”. Similarly, the STagID value is “don’t care” when STagInOrder is asserted. Nonetheless, it is suggested that the slave return whatever tag value the master provided.

The tag\_interleave\_size parameter specifies that responses of the same packing burst transaction must stay together up to this specified value, i.e., the parameter value is an upper bound. Note, however, that the number of responses that must stay together will be less than the tag\_interleave\_size when the packing burst begins on an unaligned data block boundary. The last set of responses that must stay together for this packing burst can also be less than the tag\_interleave\_size. The following pseudo-code makes this notion precise.

```
// Given two MAddr values MAddr_previous and MAddr_current,
// use the following pseudo-code to determine if MAddr_current
// belongs to the same power-of-two, aligned data block as
// MAddr_previous.
//
// 'alignMask' identifies the least significant bits of the
// power-of-two, aligned data block. E.g.,

| <u>block size (bits)</u> | <u>alignMask</u> |
|--------------------------|------------------|
| <u>32</u>                | <u>0x3</u>       |
| <u>64</u>                | <u>0x7</u>       |

alignMask = [ tag_interleave_size * ( data_wdth / 8 ) ] - 1

if
    (MAddr_current & ~alignMask)
        == (MAddr_previous & ~alignMask)
then
    MAddr_current belongs to the same power-of-two,
    aligned data block as MAddr_previous.
else
    MAddr_current belongs to a different power-of-two,
    aligned data block than MAddr_previous.
```

Multi-threaded OCP interfaces can also have tags. Each thread’s tags are independent of the other threads’ tags and apply only to the ordering of transfers within that thread. There are no ordering restrictions for transfers on different threads. The number of tags supported by all threads must be uniform, but a master need not make use of all tags on all threads.

# New Table Note for transTypes

## Location in OCP2 Specification

Notes to Table 53, on page 300.

## Modified Text

4. RDEX, RDL, and WRC commands only apply to SINGLE transfers.
5. RD, RDEX, and RDL are not controlled by the datahandshake and `writeresp_enable` parameters.
6. WRNP and WRC are not controlled by the `writeresp_enable` parameter.
7. The possible transTypes are:
  - SINGLE transfers RD, WR, BCST, WRNP, RDEX, RDL, and WRC
  - MRMD bursts RD, WR, BCST, and WRNP
  - SRMD bursts RD, WR, BCST, and WRNP

# STagInOrder Definition

## Location in OCP2 Specification

Table 58 on page 309.

## Modified Text

A new row for STagInOrder is added to Table 58, as shown below:

Table 58 OCP Trace File, Line Field Decoding

Field	Parameter Condition	Field Width in Bits	Format
<i>... Table rows eliminated for clarity</i>			
STagID	tags > 1 and resp is 1	tagid_width <sup>2</sup>	hexadecimal
<u>STagInOrder</u>	<u>tagorder is 1</u>	<u>Always 1</u>	<u>hexadecimal</u>
SData	sdata is 1	data_width	hexadecimal
<i>... Table rows eliminated for clarity</i>			



**OCP-IP Administration**

3855 SW 153rd Drive  
Beaverton, OR 97006  
Ph: +1 (503) 619-0560  
Fax: +1 (503) 644-6708  
admin@ocpip.org  
www.ocpip.org

