# AMBA® APB

## Protocol Specification


ARM IHI 0024D (ID041221)

# AMBA APB
## Protocol Specification



### Release Information

The following changes have been made to this book.

**Change history**

| Date | Issue | Confidentiality | Change |
|---|---|---|---|
| 25 September 2003 | A | Non-Confidential | First release, version 1.0 |
| 17 August 2004 | B | Non-Confidential | Second release, version 1.0 |
| 13 April 2010 | C | Non-Confidential | First release, version 2.0 |
| 09 April 2021 | D | Non-Confidential | New features for APB5 protocol |

### Proprietary Notice



Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
LES-PRE-21451 version 2.2

**AMBA SPECIFICATION LICENCE**

THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED ("ARM") FOR THE USE OF ARM'S INTELLECTUAL PROPERTY (INCLUDING, WITHOUT LIMITATION, ANY COPYRIGHT) IN THE RELEVANT AMBA SPECIFICATION ACCOMPANYING THIS LICENCE. ARM LICENSES THE RELEVANT AMBA SPECIFICATION TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING "I AGREE" OR OTHERWISE USING OR COPYING THE RELEVANT AMBA SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE.

"LICENSEE" means You and your Subsidiaries.

"Subsidiary" means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, Arm hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to:

   (i) use and copy the relevant AMBA Specification for the purpose of developing and having developed products that comply with the relevant AMBA Specification;

   (ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by Arm in Clause 1(i) of such third party's AMBA Specification Licence; and

   (iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).

2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions:

   (i) where a product created under Clause 1(i) is an integrated circuit which includes a CPU then either: (a) such CPU shall only be manufactured under licence from Arm; or (b) such CPU is neither substantially compliant with nor marketed as being compliant with the Arm instruction sets licensed by Arm from time to time;

   (ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant AMBA Specification; and

   (iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.

3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any Arm technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any Arm technology except the relevant AMBA Specification.

4. THE RELEVANT AMBA SPECIFICATION IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT ANY USE OR IMPLEMENTATION OF SUCH ARM TECHNOLOGY WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

5. NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS AGREEMENT, TO THE FULLEST EXTENT PETMITTED BY LAW, THE MAXIMUM LIABILITY OF ARM IN AGGREGATE FOR ALL CLAIMS MADE AGAINST ARM, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS AGREEMENT (INCLUDING WITHOUT LIMITATION (I) LICENSEE'S USE OF THE ARM TECHNOLOGY; AND (II) THE IMPLEMENTATION OF THE ARM TECHNOLOGY IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS AGREEMENT) SHALL NOT EXCEED THE FEES PAID (IF ANY) BY LICENSEE TO ARM UNDER THIS AGREEMENT. THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

6. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the Arm tradename, or AMBA trademark in connection with the relevant AMBA Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of Arm in respect of the relevant AMBA Specification.

7. This Licence shall remain in force until terminated by you or by Arm. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination

of this Licence by You or by Arm LICENSEE shall stop using the relevant AMBA Specification and destroy all copies of the relevant AMBA Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.

8. The validity, construction and performance of this Agreement shall be governed by English Law.

### Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

### Product Status

The information in this document is final, that is for a developed product.

### Web Address

`http://www.arm.com`

# Contents
# AMBA APB Protocol Specification

# Preface

This preface introduces the *AMBA APB Protocol Specification*. It contains the following sections:

# About this specification

This specification is for the *Advanced Microcontroller Bus Architecture* (AMBA) *Advanced Peripheral Bus* (APB) Protocol Specification.

## Intended audience

This specification is written for hardware and software engineers who want to become familiar with the AMBA APB protocol.

## Using this book

This specification is organized into the following chapters:

**Chapter 1 *Introduction***
Read this for an overview of the APB protocol.

**Chapter 2 *Signal Descriptions***
Read this for a description of the APB signals and their characteristics.

**Chapter 3 *Transfers***
Read this for information about the typical types of APB transfer, error responses, and protection support.

**Chapter 4 *Operating States***
Read this for descriptions of APB operating states and their transpositions.

**Chapter 5 *Interface parity protection***
Read this for information about parity protection and check signals.

**Appendix A *Signal validity***
Read this for a summary of rules when signals are valid.

**Appendix B *Signal list***
Read this for signal matrix of all APB protocol signals.

**Appendix C *Revisions***
Read this for a description of the technical changes between released issues of this book.

## Conventions

Conventions that this specification can use are described in:

- *Typographical*
- *Timing diagrams* on page ix
- *Signals* on page ix

### Typographical

The typographical conventions are:

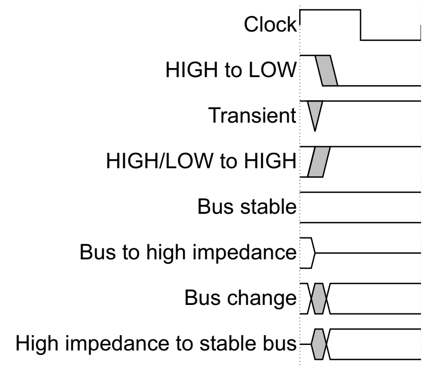
***italic*** Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

**bold** Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

### Timing diagrams

The components used in timing diagrams are explained in the figure *Key to timing diagram conventions*. Variations have clear labels, when they occur. Do not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.




**Key to timing diagram conventions**

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

### Signals

The signal conventions are:

**Signal level** The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals
- LOW for active-LOW signals.

**Lower-case n** At the start or end of a signal name denotes an active-LOW signal.

## Additional reading

This section lists publications by Arm and by third parties.

See Arm Developer https://developer.arm.com/documentation for access to Arm documentation.

### Arm publications

This specification contains information that is specific to this product. See the following documents for other relevant information:
- *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022)

# APB Revisions

## APB Specification Rev E

The *APB Specification Rev E*, released in 1998, is now obsolete.

## AMBA 2 APB Specification (Issue A)

The AMBA 2 APB Specification is detailed in *AMBA Specification Rev 2* (ARM IHI 0011A).

This version of the specification is referred to as APB2.

## AMBA 3 APB Specification (Issue B)

The *AMBA 3 APB Protocol Specification v1.0* defines the following additional functionality:

- Wait states. See Chapter 3 *Transfers*.
- Error reporting. See *Error response* on page 3-25.

The following interface signals support this functionality:

**PREADY** A ready signal to indicate completion of an APB transfer.

**PSLVERR** An error signal to indicate the failure of a transfer.

This version of the specification is referred to as APB3.

## AMBA APB Specification (Issue C)

The *AMBA APB Protocol Specification v2.0* defines the following additional functionality:

- Transaction protection. See *Protection unit support* on page 3-27.
- Sparse data transfer. See *Write strobes* on page 3-22.

The following interface signals support this functionality:

**PPROT** A protection signal to support both non-secure and secure transactions on APB.

**PSTRB** A write strobe signal to enable sparse data transfer on the write data bus.

This version of the specification is referred to as APB4.

## AMBA APB Specification (Issue D)

The *AMBA APB Protocol Specification Issue D* defines the following additional functionality:

- Wakeup signaling. See *Wake-up signaling* on page 3-28.
- User signaling. See *User signaling* on page 3-29.
- Parity protection and check signals. See Chapter 5 *Interface parity protection*.

This version of the specification is referred to as APB5.

## Feedback

Arm welcomes feedback on this product and its documentation.

### Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title, AMBA APB Protocol Specification
- The number, ARM IHI 0024D
- The page numbers to which your comments apply
- A concise explanation of your comments

Arm also welcomes general suggestions for additions and improvements.

## Progressive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have terms that can be offensive.

Arm strives to lead the industry and create change.

Previous issues of this document included terms that can be offensive. We have replaced these terms. If you find offensive terms in this document, please contact terms@arm.com.

# Chapter 1
# **Introduction**

This chapter provides an overview of the APB protocol. It contains the following sections:

- 

## 1.1 About the APB protocol

The APB protocol is a low-cost interface, optimized for minimal power consumption and reduced interface complexity. The APB interface is not pipelined and is a simple, synchronous protocol. Every transfer takes at least two cycles to complete.

The APB interface is designed for accessing the programmable control registers of peripheral devices. APB peripherals are typically connected to the main memory system using an APB bridge. For example, a bridge from AXI to APB could be used to connect a number of APB peripherals to an AXI memory system.

APB transfers are initiated by an APB bridge. APB bridges can also be referred to as a Requester. A peripheral interface responds to requests. APB peripherals can also be referred to as a Completer. This specification will use Requester and Completer.

# Chapter 2
# Signal Descriptions

This chapter describes the AMBA APB signals. It contains the following section:

## 2.1 AMBA APB signals

This section describes the APB interface signals.

Some signals on the APB interface have a fixed width and some can take a variety of widths. When the width is not fixed, it is described using a property. If the property value is zero, this means the signal is not present on the interface.

Table 2-1 provides a description of the APB protocol interface signals.

**Table 2-1 APB signal descriptions**

| Signal | Source | Width | Description |
|---|---|---|---|
| **PCLK** | Clock | 1 | Clock.<br>**PCLK** is a clock signal. All APB signals are timed against the rising edge of **PCLK**. |
| **PRESETn** | System bus reset | 1 | Reset.<br>**PRESETn** is the reset signal and is active-LOW. **PRESETn** is normally connected directly to the system bus reset signal. |
| **PADDR** | Requester | ADDR_WIDTH | Address.<br>**PADDR** is the APB address bus. **PADDR** can be up to 32 bits wide. |
| **PPROT** | Requester | 3 | Protection type.<br>**PPROT** indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.<br>See *Protection unit support* on page 3-27. |
| **PSELx** | Requester | 1 | Select.<br>The Requester generates a **PSELx** signal for each Completer. **PSELx** indicates that the Completer is selected and that a data transfer is required. |
| **PENABLE** | Requester | 1 | Enable.<br>**PENABLE** indicates the second and subsequent cycles of an APB transfer. |
| **PWRITE** | Requester | 1 | Direction.<br>**PWRITE** indicates an APB write access when HIGH and an APB read access when LOW. |
| **PWDATA** | Requester | DATA_WIDTH | Write data.<br>The **PWDATA** write data bus is driven by the APB bridge Requester during write cycles when **PWRITE** is HIGH.<br>**PWDATA** can be 8, 16, or 32 bits wide. |
| **PSTRB** | Requester | DATA_WIDTH/8 | Write strobe.<br>**PSTRB** indicates which byte lanes to update during a write transfer. There is one write strobe for each 8 bits of the write data bus. **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**. **PSTRB** must not be active during a read transfer.<br>See *Write strobes* on page 3-22. |

**Table 2-1 APB signal descriptions (continued)**

| Signal | Source | Width | Description |
|---|---|---|---|
| **PREADY** | Completer | 1 | Ready.<br>**PREADY** is used to extend an APB transfer by the Completer. |
| **PRDATA** | Completer | DATA_WIDTH | Read data.<br>The **PRDATA** read data bus is driven by the selected Completer during read cycles when **PWRITE** is LOW.<br>**PRDATA** can be 8, 16, or 32 bits wide. |
| **PSLVERR** | Completer | 1 | Transfer error.<br>**PSLVERR** is an optional signal that can be asserted HIGH by the Completer to indicate an error condition on an APB transfer.<br>See *Error response* on page 3-25. |
| **PWAKEUP** | Requester | 1 | Wake-up.<br>**PWAKEUP** indicates any activity associated with an APB interface.<br>See *Wake-up signaling* on page 3-28. |
| **PAUSER** | Requester | USER_REQ_WIDTH | User request attribute.<br>**PAUSER** is recommended to have a maximum width of 128 bits.<br>See *User signaling* on page 3-29. |
| **PWUSER** | Requester | USER_DATA_WIDTH | User write data attribute.<br>**PWUSER** is recommended to have a maximum width of DATA_WIDTH/2.<br>See *User signaling* on page 3-29. |
| **PRUSER** | Completer | USER_DATA_WIDTH | User read data attribute.<br>**PRUSER** is recommended to have a maximum width of DATA_WIDTH/2.<br>See *User signaling* on page 3-29. |
| **PBUSER** | Completer | USER_RESP_WIDTH | User response attribute.<br>**PBUSER** is recommended to have a maximum width of 16 bits.<br>See *User signaling* on page 3-29. |

### 2.1.1 Address bus

An APB interface has a single address bus, **PADDR**, for read and write transfers. **PADDR** indicates a byte address.

**PADDR** is permitted to be unaligned with respect to the data width, but the result is UNPREDICTABLE. For example, a Completer might use the unaligned address, aligned address, or signal an error response.

### 2.1.2 Data buses

The APB protocol has two independent data buses, one for read data and one for write data. The buses can be 8, 16, or 32 bits wide. The read and write data buses must have the same width.

Data transfers cannot occur concurrently because the read data and write data buses do not have their own individual handshake signals.

# Chapter 3
# **Transfers**

This chapter describes typical AMBA APB transfers, the error response, and protection unit support. It contains the following sections:

## 3.1 Write transfers

This section describes the following types of write transfer:

- With no wait states
- With wait states

All signals shown in this section are sampled at the rising edge of **PCLK**.

### 3.1.1 With no wait states

Figure 3-1 shows a basic write transfer with no wait states.

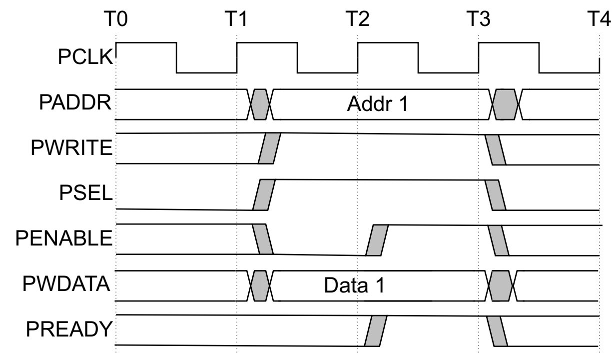



**Figure 3-1 Write transfer with no wait states**

The Setup phase of the write transfer occurs at T1 in Figure 3-1. The select signal, **PSEL**, is asserted, which means that **PADDR**, **PWRITE** and **PWDATA** must be valid.

The Access phase of the write transfer is shown at T2 in Figure 3-1 where **PENABLE** is asserted. **PREADY** is asserted by the Completer at the rising edge of **PCLK** to indicate that the write data will be accepted at T3. **PADDR**, **PWDATA**, and any other control signals, must be stable until the transfer completes.

At the end of the transfer, **PENABLE** is deasserted. **PSEL** is also deasserted, unless there is another transfer to the same peripheral.

### 3.1.2 With wait states

Figure 3-2 shows how the Completer can use **PREADY** to extend the transfer.

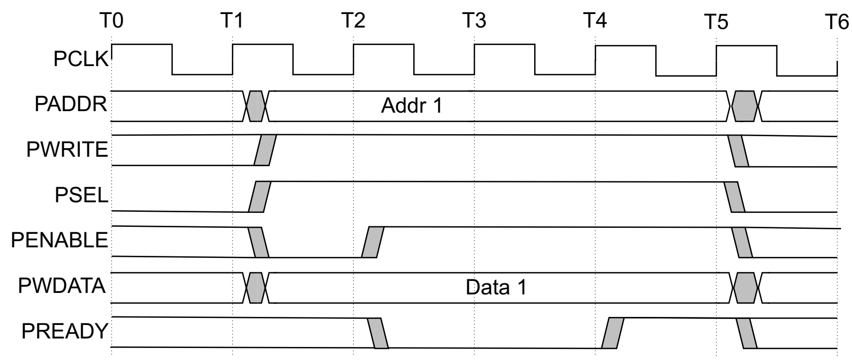



**Figure 3-2 Write transfer with wait states**

During an Access phase, when **PENABLE** is HIGH, the Completer extends the transfer by driving **PREADY** LOW. The following signals remain unchanged while **PREADY** remains LOW:

- Address signal, **PADDR**
- Direction signal, **PWRITE**
- Select signal, **PSELx**
- Enable signal, **PENABLE**
- Write data signal, **PWDATA**
- Write strobe signal, **PSTRB**
- Protection type signal, **PPROT**
- User request attribute, **PAUSER**
- User write data attribute, **PWUSER**

**PREADY** can take any value when **PENABLE** is LOW. This ensures that peripherals that have a fixed two cycle access can tie **PREADY** HIGH.

## 3.2 Write strobes

**PSTRB** enables sparse data transfer on the write data bus. Each **PSTRB** corresponds to 1 byte of the write data bus. When asserted HIGH, **PSTRB** indicates that the corresponding byte lane of the write data bus contains valid information.

There is one write strobe for each 8 bits of the write data bus, so **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**.

Figure 3-3 shows this relationship on a 32-bit data bus.

| 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|
| PSTRB[3] | PSTRB[2] | PSTRB[1] | PSTRB[0] |

**Figure 3-3 Byte lane mapping**

For read transfers, the Requester must drive all bits of **PSTRB** LOW.

### 3.2.1 PSTRB presence and compatibility

**PSTRB** is an optional signal. An APB peripheral might support a limited set of access types, which must be documented for the programmer. This means that all combinations of **PSTRB** presence might be compatible, if this document states that sparse writes are not supported.

The compatibility of **PSTRB** when connecting Requesters and Completers is described inTable 3-1.

**Table 3-1 PSTRB presence and compatibility**

| PSTRB | Completer: signal not present | Completer: signal present |
|---|---|---|
| **Requester: signal not present** | Compatible.<br>Sparse writes are not supported. | Compatible.<br>All write data byte lanes are valid for a write. Tie the **PSTRB** inputs to the **PWRITE** output from the Requester. |
| **Requester: signal present** | Compatible.<br>Sparse writes are not supported. | Compatible. |

## 3.3 Read transfers

Two types of read transfer are described in this section:

- With no wait states
- With wait states

All signals shown in this section are sampled at the rising edge of **PCLK**.

### 3.3.1 With no wait states

Figure 3-4 shows a read transfer.

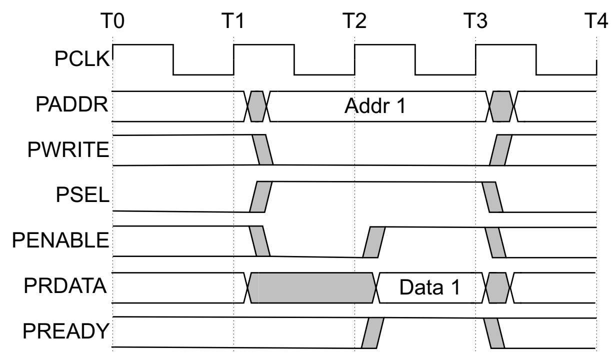



**Figure 3-4 Read transfer with no wait states**

The timing of the address, **PADDR**, write, **PWRITE**, select, **PSEL**, and enable, **PENABLE**, signals are the same as described in *Write transfers* on page 3-20. The Completer must provide the data before the end of the read transfer.

### 3.3.2 With wait states

Figure 3-5 shows how the **PREADY** signal can extend the transfer.

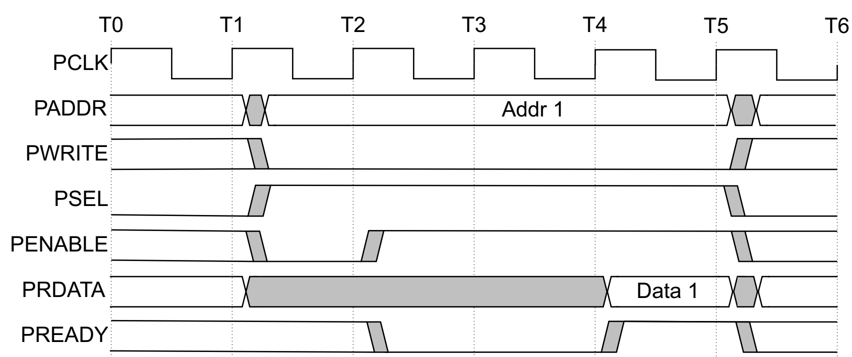



**Figure 3-5 Read transfer with wait states**

The transfer is extended if **PREADY** is driven LOW during an Access phase. The following signals remain unchanged while **PREADY** remains LOW:

- Address signal, **PADDR**
- Direction signal, **PWRITE**
- Select signal, **PSEL**
- Enable signal, **PENABLE**
- Protection signal, **PPROT**
- User signal, **PAUSER**

Figure 3-5 on page 3-23 shows that two cycles are added using **PREADY**. However, any number of additional cycles can be added, from zero upwards.

## 3.4 Error response

**PSLVERR** can be used to indicate an error condition on an APB transfer. Error conditions can occur on both read and write transactions.

**PSLVERR** is only considered valid during the last cycle of an APB transfer, when **PSEL**, **PENABLE**, and **PREADY** are all HIGH.

It is recommended, but not required, that **PSLVERR** is driven LOW when **PSEL**, **PENABLE**, or **PREADY** are LOW.

Transactions that receive an error might or might not have changed the state of the peripheral. This is peripheral-specific and either state is acceptable.

When a write transaction receives an error, this does not mean that the register within the peripheral has not been updated.

Read transactions that receive an error can return invalid data. There is no requirement for the peripheral to drive the data bus to all 0s for a read error. A Requester which receives an error response to a read transfer might still use the data. A Completer cannot rely on the error response to prevent the reading of a value on **PRDATA**.

Completers are not required to support **PSLVERR**. Where a Completer does not include **PSLVERR**, the appropriate input to the Requester is tied LOW.

### 3.4.1 Write transfer

Figure 3-6 shows an example of a failing write transfer that completes with an error.

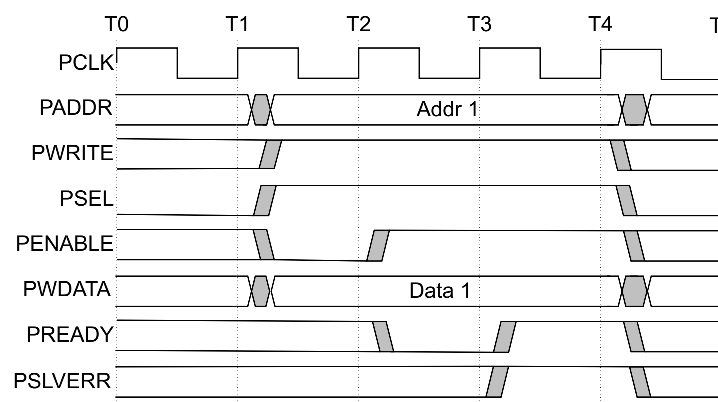



**Figure 3-6 Example failing write transfer**

### 3.4.2 Read transfer

A read transfer can also complete with an error response, indicating that there is no valid read data available.

Figure 3-7 shows a read transfer completing with an error response.

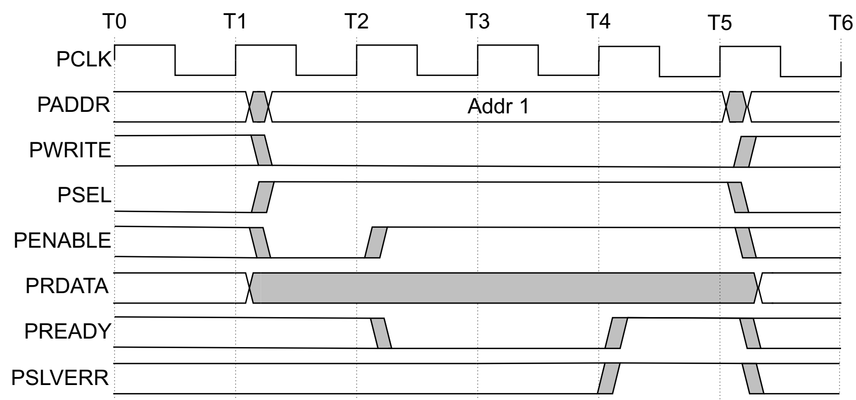



**Figure 3-7 Example failing read transfer**

### 3.4.3 Mapping of PSLVERR

When bridging:

**From AXI to APB** An APB error on **PSLVERR** is mapped back to **RRESP** for reads and **BRESP** for writes.

**From AHB to APB** An APB error on **PSLVERR** is mapped back to **HRESP** for reads and writes.

## 3.5 Protection unit support

To support complex system designs, it is often necessary for both the interconnect and other devices in the system to provide protection against illegal transactions. For the APB interface, this protection is provided by the **PPROT[2:0]** signals.

Table 3-2 shows the three levels of access protection with the protection level encoding.

**Table 3-2 Access protection**

| PPROT | Protection | Description | Comments |
|---|---|---|---|
| **PPROT[0]** | Normal or Privileged | **PPROT[0]** is used by Requesters to indicate processing mode. A privileged processing mode typically has a greater level of access within a system. | • LOW indicates normal access.<br>• HIGH indicates privileged access. |
| **PPROT[1]** | Secure or Non-secure | **PPROT[1]** is used in systems where a greater degree of differentiation between processing modes is required. | • LOW indicates secure access.<br>• HIGH indicates non-secure access. |
| **PPROT[2]** | Data or Instruction | **PPROT[2]** gives an indication if the transaction is a data or instruction access.<br>The transaction indication is provided as a hint and might not be accurate in all cases. | • LOW indicates data access.<br>• HIGH indicates instruction access. |

**Note**

The primary use of **PPROT** is as an identifier for Secure or Non-secure transactions. It is acceptable to use different interpretations of the **PPROT[0]** and **PPROT[2]** identifiers.

### 3.5.1 PPROT presence and compatibility

**PPROT** is an optional signal on Requester and Completer interfaces.

Table 3-3 describes the **PPROT** compatibility when connecting a Completer to a Requester.

**Table 3-3 PPROT presence and compatibility**

| PPROT | Completer: signal not present | Completer: signal present |
|---|---|---|
| **Requester: signal not present** | Compatible. | Not compatible.<br>If fixed protection attributes are functionally correct, then the interfaces are Compatible. See Table 3-2 on page 3-27. |
| **Requester: signal present** | Compatible.<br>The Completer has no access protection so **PPROT** can be ignored. | Compatible. |

# 3.6 Wake-up signaling

This section describes wake-up signaling used in an APB interface.

## 3.6.1 Introduction

The wake-up signal, **PWAKEUP**, is used to indicate any activity associated with any APB interface. **PWAKEUP** provides a glitch-free signal that can be routed to a clock controller, or similar component, to enable power and clocks to connected components.

The Wakeup_Signal property is used to indicate whether a component supports wake-up signaling:

**True** Wake-up signal is present.

**False** Wake-up signal is not present. If the Wakeup_Signal property is not declared, it is considered False.

Wake-up signaling can only be added to APB5 protocol interfaces.

## 3.6.2 PWAKEUP signaling

Table 3-4 describes the **PWAKEUP** signal.

**Table 3-4 PWAKEUP signal description**

| Signal | Width | Source | Description |
|---|---|---|---|
| **PWAKEUP** | 1 | Requester | **PWAKEUP** indicates any activity associated with a Requester interface. |

The rules and recommendations for **PWAKEUP** are:

- **PWAKEUP** is synchronous to **PCLK** and must be suitable for sampling asynchronously in a different clock domain. This requires **PWAKEUP** to be glitch-free. This can be achieved, for example, by being generated directly from a register, or from a glitch-free OR tree.
- **PWAKEUP** is allowed to be asserted before, during, or after the assertion of **PSELx**.
- A Completer is permitted to wait for **PWAKEUP** to be asserted, before asserting **PREADY**. The interface could deadlock if **PWAKEUP** is present but never asserted.
- **PWAKEUP** must remain asserted until **PREADY** is asserted if **PWAKEUP** and **PSELx** are HIGH in the same cycle.
- It is recommended that **PWAKEUP** be asserted at least one cycle before the assertion of **PSELx** to prevent the acceptance of a new transaction being delayed.
- It is recommended **PWAKEUP** be deasserted when no further transfers are required.
- It is permitted, but not recommended, asserting **PWAKEUP** then deasserting it without a transfer occurring.
- It is recommended that the Requester and Completer sides of a connection are clock gated together.

  If a Completer interface clock is gated independently of the Requester clock and **PWAKEUP** is used to enable the Completer clock, there is a possibility that the Setup phase of a transfer is missed by the Completer.

## 3.7 User signaling

This section describes user signaling in an APB interface.

### 3.7.1 Introduction

The users of APB protocols can encounter an application that requires the addition of signaling that is not specified in the APB protocol. User signaling defines a standard method of adding this signaling to a transaction, without defining the signal usage.

Generally, it is recommended that User signals are not used. The APB protocol interface does not define the function of these signals, causing interoperability problems if two components use the same User signals in an incompatible way.

User signaling can only be added to APB5 protocol interfaces.

### 3.7.2 Signaling

All signals are optional. If the associated width property is zero, the signal is not present.

Table 3-5 on page 3-29 provides a description of User signals.

**Table 3-5 User signal descriptions**

| Signal | Width | Source | Description |
|---|---|---|---|
| **PAUSER** | USER_REQ_WIDTH | Requester | User-defined request attribute.<br>• **PAUSER** must be valid when **PSELx** is asserted.<br>• **PAUSER** must have the same value in the Setup and Access phase of a transfer.<br>• **PAUSER** must have the same value in every cycle during the Access phase of a transfer. |
| **PWUSER** | USER_DATA_WIDTH | Requester | User-defined write data attribute.<br>• **PWUSER** must be valid when **PSEL** and **PWRITE** are asserted.<br>• **PWUSER** must have the same value in the Setup and Access phases of a transfer.<br>• **PWUSER** must have the same value in every cycle during the Access phase of a transfer. |
| **PRUSER** | USER_DATA_WIDTH | Completer | User-defined read data attribute.<br>• **PRUSER** must be valid when **PSEL**, **PENABLE**, and **PREADY** are asserted, and **PWRITE** is deasserted. |
| **PBUSER** | USER_RESP_WIDTH | Completer | User-defined response attribute.<br>• **PBUSER** must be valid when **PSEL**, **PENABLE**, and **PREADY** are asserted. |

### 3.7.3 User signal recommendations

Where User signals are implemented, this specification does not require support for all User signals. The width of the User-defined signals is IMPLEMENTATION DEFINED and can be different for request, data, and responses.

It is recommended including provision for all User signals on a domain crossing bridge or interconnect. However, there is no requirement to include them on Completers.

It is recommended USER_DATA_WIDTH is an integer multiple of the width of the data buses in bytes to assist with data width and protocol conversion.

Chapter 4
# Operating States

This chapter describes the AMBA APB operating states. It contains the following section:

## 4.1 Operating states

Figure 4-1 shows the operating states of the APB interface.

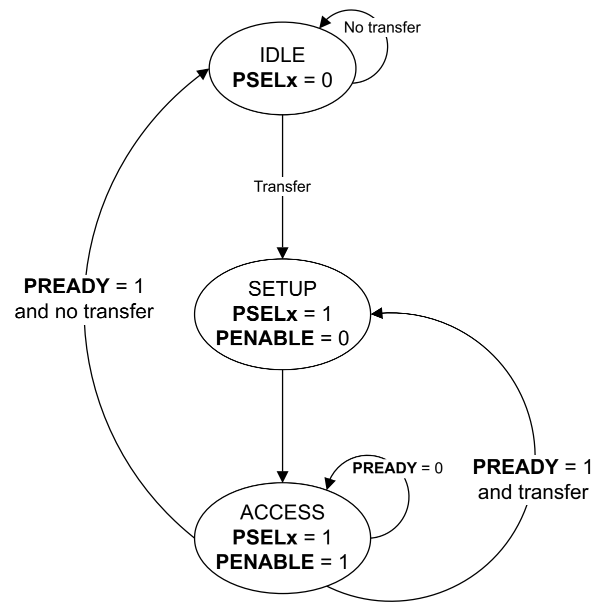



**Figure 4-1 State diagram**

The state machine operates through the following states:

**IDLE** This is the default state of the APB interface.

**SETUP** When a transfer is required, the interface moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The interface only remains in the SETUP state for one clock cycle and always moves to the ACCESS state on the next rising edge of the clock.

**ACCESS** The enable signal, **PENABLE**, is asserted in the ACCESS state. The following signals must not change in the transition between SETUP and ACCESS and between cycles in the ACCESS state:

- **PADDR**
- **PPROT**
- **PWRITE**
- **PWDATA**, only for write transactions
- **PSTRB**
- **PAUSER**
- **PWUSER**

Exit from the ACCESS state is controlled by the **PREADY** signal from the Completer:

- If **PREADY** is held LOW by the Completer, then the interface remains in the ACCESS state.
- If **PREADY** is driven HIGH by the Completer then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required. Alternatively, the bus moves directly to the SETUP state if another transfer follows.

# Chapter 5
# Interface parity protection

This chapter describes a parity scheme for detecting single-bit errors on the interface between components. It contains the following sections:

## 5.1 Protection using parity

For safety-critical applications, it is necessary to detect and correct transient and functional errors on individual wires within an SoC.

An error in a system component can propagate and cause numerous errors across connected components. *Error Detection and Correction* (EDC) is required to operate end-to-end, covering all logic and wires from source to destination.

One way to implement end-to-end protection is to employ customized EDC schemes in components and implement a simple error detection scheme between components. Between these components, there is no logic and single-bit errors do not propagate to multi-bit errors. This section describes a parity scheme for detecting single-bit errors on the interface between components. Multi-bit errors can be detected if they occur in different parity signal groups.

Figure 5-1 shows the locations where parity can be used.

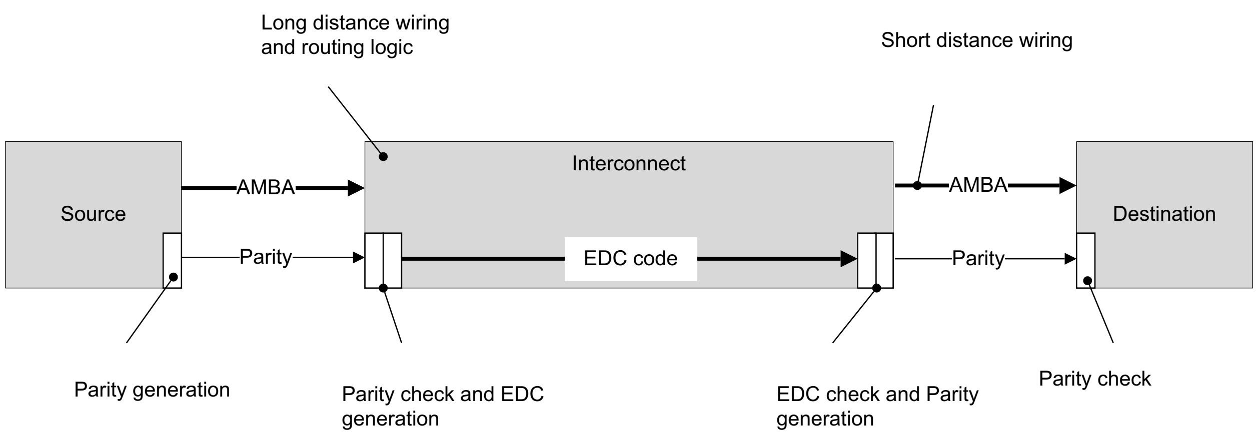



**Figure 5-1 Example use of parity protection**

## 5.2 Configuration of interface protection

The EDC scheme of the interface is defined by the Check_Type property. The following Check_Type values are defined:

**False**

There are no checking signals on the interface.

**Odd_Parity_Byte_All**

Odd parity checking is included for all signals. Each bit of the parity signal covers up to 8 bits.

If Check_Type is not declared, it is considered to be False.

Check signaling can be added to APB5 interfaces only.

## 5.3 Parity check

The following attributes are common to the check signals added for byte parity interface protection:

- Odd parity is used. Odd parity means that there is always an odd number of bits asserted across the interface signal and check signal. Check signals are associated with each interface signal.
- Each parity check bit covers no more than 8 bits of payload. This limitation assumes that there is a maximum of three logic levels available in the timing allowance for generating each parity bit.
- Parity signals that cover critical control signals are defined with a single parity bit. The single odd parity bit is the inversion of the original critical control signal. Critical control signals are likely to have a smaller timing allowance available.
- For a check signal that is wider than 1 bit:
  — Check bit `[n]` corresponds to `[(8n+7):8n]` in the payload.
  — If the payload is not an integer number of bytes, the most significant bit of the check signal covers fewer than 8-bits in the most significant portion of the payload.
- Check signals must be driven correctly in every cycle that the Check Enable term is True. See Table 5-1 on page 5-38.
- Parity signals must be driven appropriately to all the bits in the associated payload, whether or not the bits are actively used in the transfer. For example, all bits of **PWDATACHK** must be driven correctly, even if some bytes are not valid data bytes.
- If some of the signals covered by a check signal are not present on an interface, then the missing signals are assumed to be LOW.
- If none of the signals covered by a check signal are present on an interface, then the check signal is omitted from the interface.

## 5.4 Error detection behavior

This specification is not prescriptive regarding component or system behavior when a parity error is detected. Depending on the system and affected signals, a flipped bit can have a wide range of effects. It might be harmless, cause performance issues, cause data corruption, cause security violations, or deadlock.

When an error is detected, the Completer can:

- Terminate or propagate the transfer.
- Correct the parity check signal or propagate the error.
- Update its memory or leave untouched.
- Signal an error response through other means, for example with an interrupt.

## 5.5 Parity check signals

Check signals are synchronous to **PCLK** and must be driven correctly every cycle in which the Check Enable term is True.

Table 5-1 shows the parity check signals.

**Table 5-1 Check signal descriptions**

| Check signal | Signals covered | Width | Granularity | Check Enable |
|---|---|---|---|---|
| **PADDRCHK** | **PADDR** | `ceil(ADDR_WIDTH/8)`[a] | 1-8 | **PSEL** |
| **PCTRLCHK** | **PPROT, PWRITE** | 1 | 4 | **PSEL** |
| **PSELxCHK**[b] | **PSELx** | 1 | 1 | **PRESETn** |
| **PENABLECHK** | **PENABLE** | 1 | 1 | **PSEL** |
| **PWDATACHK** | **PWDATA** | DATA_WIDTH/8 | 8 | **PSEL** & **PWRITE** |
| **PSTRBCHK** | **PSTRB** | 1 | 1-4 | **PSEL** & **PWRITE** |
| **PREADYCHK** | **PREADY** | 1 | 1 | **PSEL** & **PENABLE** |
| **PRDATACHK** | **PRDATA** | DATA_WIDTH/8 | 8 | **PSEL** & **PENABLE** & **PREADY** & **!PWRITE** |
| **PSLVERRCHK** | **PSLVERR** | 1 | 1 | **PSEL** & **PENABLE** & **PREADY** |
| **PWAKEUPCHK** | **PWAKEUP** | 1 | 1 | **PRESETn** |
| **PAUSERCHK** | **PAUSER** | `ceil(USER_REQ_WIDTH/8)` | 1-8 | **PSEL** |
| **PWUSERCHK** | **PWUSER** | `ceil(USER_DATA_WIDTH/8)` | 1-8 | **PSEL** & **PWRITE** |
| **PRUSERCHK** | **PRUSER** | `ceil(USER_DATA_WIDTH/8)` | 1-8 | **PSEL** & **PENABLE** & **PREADY** & **!PWRITE** |
| **PBUSERCHK** | **PBUSER** | `ceil(USER_RESP_WIDTH/8)` | 1-8 | **PSEL** & **PENABLE** & **PREADY** |

a. The function `ceil()` returns the lowest integer value that is equal to or greater than the input to the function.

b. There is a separate check signal per **PSEL** signal as only one **PSEL** bit is routed to each Completer.

# Appendix A
# Signal validity

This appendix summarizes the rules describing when signals must be valid.

## A.1 Validity rules

The following signals must always be valid:

- **PSEL**
- **PWAKEUP**

The following signals must be valid when **PSEL** is asserted:

- **PADDR**
- **PPROT**
- **PENABLE**
- **PWRITE**
- **PAUSER**
- **PSTRB**
- **PWDATA**, active write data lanes only
- **PWUSER**, write only

The following signal must be valid when **PSEL** and **PENABLE** are asserted:

- **PREADY**

The following signals must be valid when **PSEL**, **PENABLE**, and **PREADY** are asserted:

- **PRDATA**, read only
- **PSLVERR**
- **PRUSER**, read only
- **PBUSER**

It is recommended that signals which are not required to be valid are driven to zero.

# Appendix B
# **Signal list**

This appendix provides a summary of all signals on the APB interface.

## B.1 APB signals

Table B-1 describes the list of the APB signals. Table B-2 describes the list of APB check signals. Optional signals have a default that should be used for any un-driven inputs.

The following codes used in Table B-1 and Table B-2 are:

| | |
|---|---|
| **Y** | Mandatory |
| **N** | Must not be present |
| **O** | Optional for inputs and outputs |
| **OO** | Optional for output ports, mandatory for inputs |
| **C** | Conditional, must be present if the property is True |
| **OC** | Optional conditional, optional but can only be present if the property is True |

**Table B-1 APB signals**

| Signal | Width | Default | Property | APB5 | APB4 | APB3 | APB2 |
|---|---|---|---|---|---|---|---|
| **PCLK** | 1 | - | - | Y | Y | Y | Y |
| **PRESETn** | 1 | - | - | Y | Y | Y | Y |
| **PADDR** | ADDR_WIDTH | - | - | Y | Y | Y | Y |
| **PPROT** | 3 | 0b000 | - | O | O | N | N |
| **PSELx** | 1 | - | - | Y | Y | Y | Y |
| **PENABLE** | 1 | - | - | Y | Y | Y | Y |
| **PWRITE** | 1 | - | - | Y | Y | Y | Y |
| **PWDATA** | DATA_WIDTH | - | - | Y | Y | Y | Y |
| **PSTRB** | DATA_WIDTH/8 | - | - | O | O | N | N |
| **PREADY** | 1 | 0b1 | - | OO | OO | OO | N |
| **PRDATA** | DATA_WIDTH | - | - | Y | Y | Y | Y |
| **PSLVERR** | 1 | 0b0 | - | OO | OO | OO | N |
| **PWAKEUP** | 1 | - | Wakeup_Signal | C | N | N | N |
| **PAUSER** | USER_REQ_WIDTH | - | USER_REQ_WIDTH | OC | N | N | N |
| **PWUSER** | USER_DATA_WIDTH | - | USER_DATA_WIDTH | OC | N | N | N |
| **PRUSER** | USER_DATA_WIDTH | - | USER_DATA_WIDTH | OC | N | N | N |
| **PBUSER** | USER_RESP_WIDTH | - | USER_RESP_WIDTH | OC | N | N | N |

**Table B-2 Check signal**

| Signal | Width | Property | APB5 | APB4 | APB3 | APB2 |
|---|---|---|---|---|---|---|
| **PADDRCHK** | ADDR_WIDTH/8 | Check_Type | C | N | N | N |
| **PCTRLCHK** | 1 | Check_Type | C | N | N | N |
| **PSELxCHK** | 1 | Check_Type | C | N | N | N |
| **PENABLECHK** | 1 | Check_Type | C | N | N | N |
| **PWDATACHK** | DATA_WIDTH/8 | Check_Type | C | N | N | N |

**Table B-2 Check signal (continued)**

| Signal | Width | Property | APB5 | APB4 | APB3 | APB2 |
|---|---|---|---|---|---|---|
| **PSTRBCHK** | 1 | Check_Type | C | N | N | N |
| **PREADYCHK** | 1 | Check_Type | C | N | N | N |
| **PRDATACHK** | DATA_WIDTH/8 | Check_Type | C | N | N | N |
| **PSLVERRCHK** | 1 | Check_Type | OC | N | N | N |
| **PWAKEUPCHK** | 1 | Check_Type & Wakeup_Signal | C | N | N | N |
| **PAUSERCHK** | `ceil(USER_REQ_WIDTH/8)` | Check_Type & USER_REQ_WIDTH | OC | N | N | N |
| **PWUSERCHK** | `ceil(USER_DATA_WIDTH/8)` | Check_Type & USER_DATA_WIDTH | OC | N | N | N |
| **PRUSERCHK** | `ceil(USER_DATA_WIDTH/8)` | Check_Type & USER_DATA_WIDTH | OC | N | N | N |
| **PBUSERCHK** | `ceil(USER_RESP_WIDTH/8)` | Check_Type & USER_RESP_WIDTH | OC | N | N | N |

Table B-3 shows the interface properties.

If an entry is not Y, the property must be False or undeclared for that interface type.

**Table B-3 Interface properties**

| Property | Issue introduced | APB5 | APB4 | APB3 | APB2 |
|---|---|---|---|---|---|
| Check_Type | D | Y | - | - | - |
| USER_DATA_WIDTH | D | Y | - | - | - |
| USER_REQ_WIDTH | D | Y | - | - | - |
| USER_RESP_WIDTH | D | Y | - | - | - |
| Wakeup_Signal | D | Y | - | - | - |

# Appendix C
# **Revisions**

This appendix describes the technical changes between released issues of this book.

**Table C-1 Issue A**

| Change | Location |
|---|---|
| First release | - |

**Table C-2 Differences between issue A and issue B**

| Change | Location |
|---|---|
| APB signal **PREADY** added | • Table 2-1 on page 2-16<br>• *Write transfers* on page 3-20<br>• *Read transfers* on page 3-23<br>• *Error response* on page 3-25<br>• Chapter 4 *Operating States* |
| APB signal **PSLVERR** added | • Table 2-1 on page 2-16<br>• *Error response* on page 3-25 |

**Table C-3 Differences between issue B and issue C**

| Change | Location |
|---|---|
| Section added listing the changes made to this specification at each revision of the document. | Appendix C *Revisions* |
| APB signal **PPROT** added | • Table 2-1 on page 2-16<br>• *Protection unit support* on page 3-27 |
| APB signal **PSTRB** added | • Table 2-1 on page 2-16<br>• *Write strobes* on page 3-22 |

**Table C-4 Changes from Issue C to Issue D**

| Change | Location |
|---|---|
| Added signal width properties | *AMBA APB signals* on page 2-16 |
| Added wake-up signaling | *Wake-up signaling* on page 3-28 |
| Added user signaling | *User signaling* on page 3-29 |
| Added interface parity | Chapter 5 *Interface parity protection* |
| Included regularized terminology using Completer and Requester | Throughout the specification |
| Added signal validity rules | Appendix A *Signal validity* |
| Added signal matrix | Appendix B *Signal list* |