

Arm® Vultan Generic Flash Controller

Revision: r0p0

Technical Reference Manual



Arm® Vultan Generic Flash Controller

Technical Reference Manual

Copyright © 2017, 2018 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document History

| Issue | Date | Confidentiality | Change |
|---------|-----------------|------------------|-------------------------|
| 0000-00 | 10 October 2017 | Confidential | First release for r0p0 |
| 0000-01 | 18 January 2018 | Non-Confidential | Second release for r0p0 |

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017, 2018 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

<http://www.arm.com>

Contents

Arm® Vultan Generic Flash Controller Technical Reference Manual

Preface

| | |
|------------------------------|----|
| <i>About this book</i> | 7 |
| <i>Feedback</i> | 10 |

Chapter 1

Introduction

| | | |
|-----|------------------------------------|------|
| 1.1 | <i>About Vultan</i> | 1-12 |
| 1.2 | <i>Compliance</i> | 1-14 |
| 1.3 | <i>Features</i> | 1-15 |
| 1.4 | <i>Test features</i> | 1-16 |
| 1.5 | <i>Product documentation</i> | 1-17 |
| 1.6 | <i>Product revisions</i> | 1-18 |

Chapter 2

Functional description

| | | |
|-----|--|------|
| 2.1 | <i>Internal structure</i> | 2-20 |
| 2.2 | <i>Interfaces</i> | 2-22 |
| 2.3 | <i>Clocking</i> | 2-28 |
| 2.4 | <i>Resets</i> | 2-29 |
| 2.5 | <i>Interrupt sources</i> | 2-30 |
| 2.6 | <i>Generic Flash Bus Arbiter</i> | 2-31 |

Chapter 3

Programmers model

| | | |
|-----|---|------|
| 3.1 | <i>About this programmers model</i> | 3-33 |
|-----|---|------|

| | | |
|-----|------------------------------------|------|
| 3.2 | <i>Memory maps</i> | 3-34 |
| 3.3 | <i>Register summary</i> | 3-41 |
| 3.4 | <i>Register descriptions</i> | 3-42 |

Appendix A

Signal descriptions

| | | |
|-----|--|-----------|
| A.1 | <i>AHB-Lite slave interface signals</i> | Appx-A-57 |
| A.2 | <i>APB slave interface signals</i> | Appx-A-59 |
| A.3 | <i>APB master interface signals</i> | Appx-A-60 |
| A.4 | <i>Generic Flash Bus signals</i> | Appx-A-61 |
| A.5 | <i>Q-Channel interface for clock signals</i> | Appx-A-63 |
| A.6 | <i>Q-Channel interface for power signals</i> | Appx-A-64 |
| A.7 | <i>P-Channel master interface signals</i> | Appx-A-65 |
| A.8 | <i>System interface signals</i> | Appx-A-66 |

Appendix B

Revisions

| | | |
|-----|------------------------|-----------|
| B.1 | <i>Revisions</i> | Appx-B-68 |
|-----|------------------------|-----------|

Preface

This preface introduces the *Arm® Vultan Generic Flash Controller Technical Reference Manual*.

It contains the following:

- *About this book* on page 7.
- *Feedback* on page 10.

About this book

This book is for the Vultan Generic Flash Controller.

Product revision status

The *rm**pn* identifier indicates the revision status of the product described in this book, for example, *r1p2*, where:

rm Identifies the major revision of the product, for example, *r1*.

pn Identifies the minor revision or modification status of the product, for example, *p2*.

Intended audience

This book is for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Vultan Generic Flash Controller.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

Read this for an introduction to the Vultan Generic Flash Controller and its features.

Chapter 2 Functional description

Read this for a description of the Vultan Generic Flash Controller functions.

Chapter 3 Programmers model

Read this for a description of the memory map and registers, and for information about programming the device.

Appendix A Signal descriptions

Read this for a description of the input and output signals.

Appendix B Revisions

Read this for a description of changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

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

monospace

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

monospace

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

monospace *italic*

Denotes arguments to monospace text where the argument is to be replaced by a specific value.

monospace **bold**

Denotes language keywords when used outside example code.

<and>

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must 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.

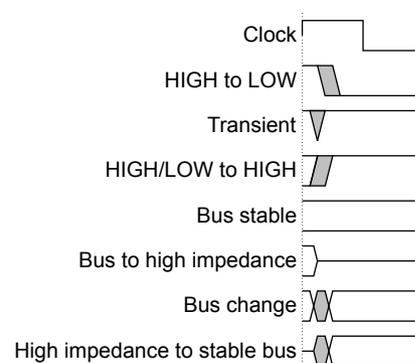


Figure 1 Key to timing diagram conventions

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.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

Information published by Arm and by third parties.

See *Infocenter* <http://infocenter.arm.com>, for access to Arm documentation.

Arm publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Arm® AMBA® APB Protocol Specification* (IHI 0024).
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* (IHI 0068).
- *AMBA® 3 AHB-Lite Protocol Specification v1.0* (IHI 0033).

The following confidential books are only available to licensees:

- *Arm® Vultan Generic Flash Controller Configuration and Integration Manual* (101060).
- *Arm® Vultan Generic Flash Controller Release Note* (PJDOC-1779577084-4407).
- *Arm® Generic Flash Bus Description* (see the *Arm® Vultan Generic Flash Controller Release Note*).

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

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

- The title *Arm Vultan Generic Flash Controller Technical Reference Manual*.
- The number 101059_0000_01_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

————— **Note** —————

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1

Introduction

Read this for an introduction to the Vultan Generic Flash Controller and its features.

It contains the following sections:

- *1.1 About Vultan* on page 1-12.
- *1.2 Compliance* on page 1-14.
- *1.3 Features* on page 1-15.
- *1.4 Test features* on page 1-16.
- *1.5 Product documentation* on page 1-17.
- *1.6 Product revisions* on page 1-18.

1.1 About Vultan

The Vultan Generic Flash Controller comprises the generic part of a Flash controller in a *System-on-Chip* (SoC). Vultan enables an embedded Flash macro to be integrated easily into any system.

An eFlash macro enables a Flash controller to access eFlash memory. The eFlash macros produced by different foundries and processes can have different interfaces, timings, signal names, protocols and features that are determined by the foundry processes that produced the eFlash memory.

Vultan provides the functions that relate only to services for the system side of the Flash controller. Vultan cannot communicate directly with the eFlash macro. Therefore, Vultan must be integrated with a process-specific part that connects to, and communicates with, the eFlash macro.

The process-specific part of the Flash controller is part of the Flash subsystem in your SoC. It communicates directly with the eFlash macro through a Flash interface.

Communication between the system and eFlash memory is through a *Generic Flash Bus* (GFB) supplied with Vultan.

The following figure shows how Vultan is used in a Flash controller implementation.

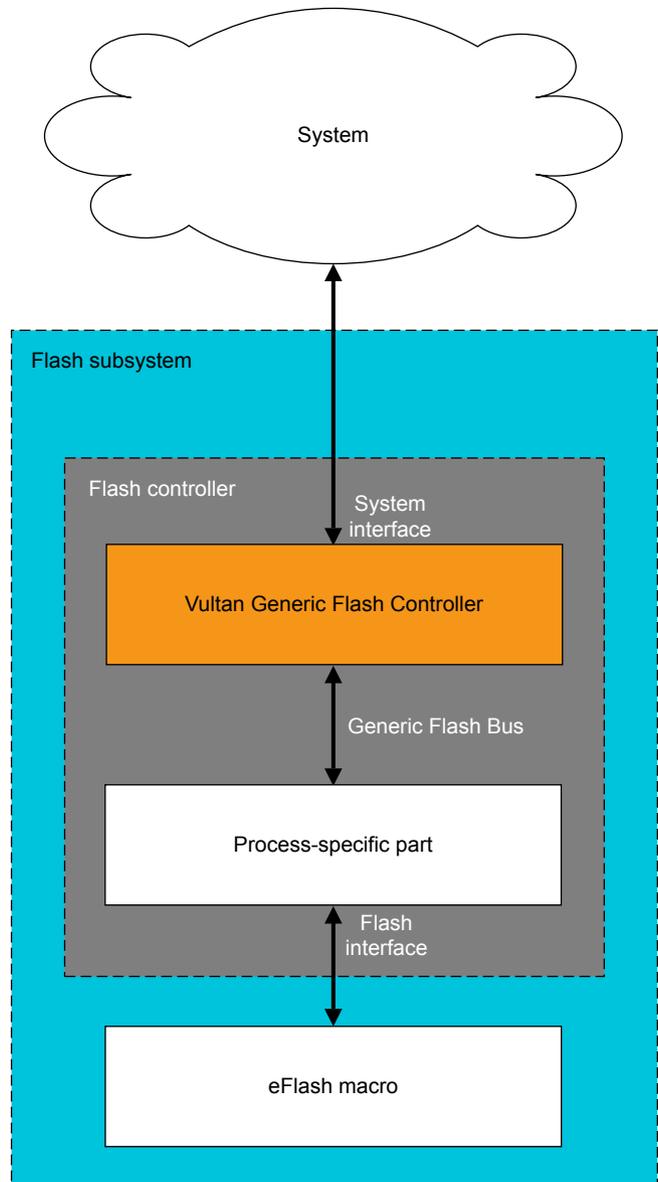


Figure 1-1 Vultan in a Flash controller implementation

1.2 Compliance

Vultan interfaces are compliant with Arm specifications and protocols.

Vultan is compliant with:

- AMBA 3 AHB-Lite Protocol Specification. See *AMBA® 3 AHB-Lite Protocol Specification v1.0*.
- AMBA 4 APB Protocol Specification. See *Arm® AMBA® APB Protocol Specification*.
- AMBA 4 Low Power Interface Specification. See *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.
- *Arm® Generic Flash Bus Description*.

1.3 Features

Vultan provides several interfaces and test features.

Advanced High-performance Bus (AHB-Lite) interface:

- Read access to the main and extended areas of embedded Flash.
- Burst support.
- Low latency.

Advanced Peripheral Bus (APB) slave interface:

- Write and Erase access to the main and extended areas of embedded Flash.
- Debug Read access to the main and extended areas of embedded Flash.
- Control port for Vultan and the Flash macro.
- Interrupt capability for long running commands.
- Access to internal and external registers.

APB register master interface:

- Control port for attached process-specific registers.

Q-Channel interface:

- Control port for system power.
- Control port for the system clock.

P-Channel master interface:

- Control port for power to the attached process-specific part.

Generic Flash Bus (GFB):

- Enables Vultan accesses to embedded Flash.
- Simple command-based protocol.
- Synchronous with the AHB clock.
- Simplifies communication between Vultan and the attached process-specific part.

1.4 Test features

Vultan provides components that comprise a Flash model, which simulates the behavior of the process-specific part and the attached embedded Flash.

The Vultan deliverables include an Out-of-Box execution testbench. You can use the execution testbench to check that the delivered RTL is complete and that it can perform transactions towards the Flash model as expected.

1.5 Product documentation

Documentation that is provided with this product includes a *Technical Reference Manual (TRM)* and a *Configuration and Integration Manual (CIM)*, together with architecture and protocol information.

For relevant protocol and architectural information that relates to this product, see [Additional reading on page 8](#).

The Vultan documentation is as follows:

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of Vultan. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that the TRM describes are not relevant. If you are programming Vultan, contact:

- The implementer to determine:
 - The build configuration of the implementation.
 - What integration, if any, was performed before implementing Vultan.
- The integrator to determine the signal configuration of the device that you use.

The TRM complements architecture and protocol specifications and relevant external standards. It does not duplicate information from these sources.

Configuration and Integration Manual

The CIM describes:

- The available build configuration options.
- How to configure the *Register Transfer Level (RTL)* with the build configuration options.
- How to integrate Vultan into a SoC.
- How to implement Vultan into your design.
- The processes to validate the configured design.

The Arm product deliverables include reference scripts and information about using them to implement your design.

The CIM is a confidential book that is only available to licensees.

1.6 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2

Functional description

Read this for a description of the Vultan Generic Flash Controller functions.

It contains the following sections:

- [2.1 Internal structure](#) on page 2-20.
- [2.2 Interfaces](#) on page 2-22.
- [2.3 Clocking](#) on page 2-28.
- [2.4 Resets](#) on page 2-29.
- [2.5 Interrupt sources](#) on page 2-30.
- [2.6 Generic Flash Bus Arbiter](#) on page 2-31.

2.1 Internal structure

Vultan comprises several submodules.

Figure 2-1 Vultan internal structure on page 2-21 shows the internal high-level structure of an example system that integrates Vultan, with a process-specific part and embedded Flash.

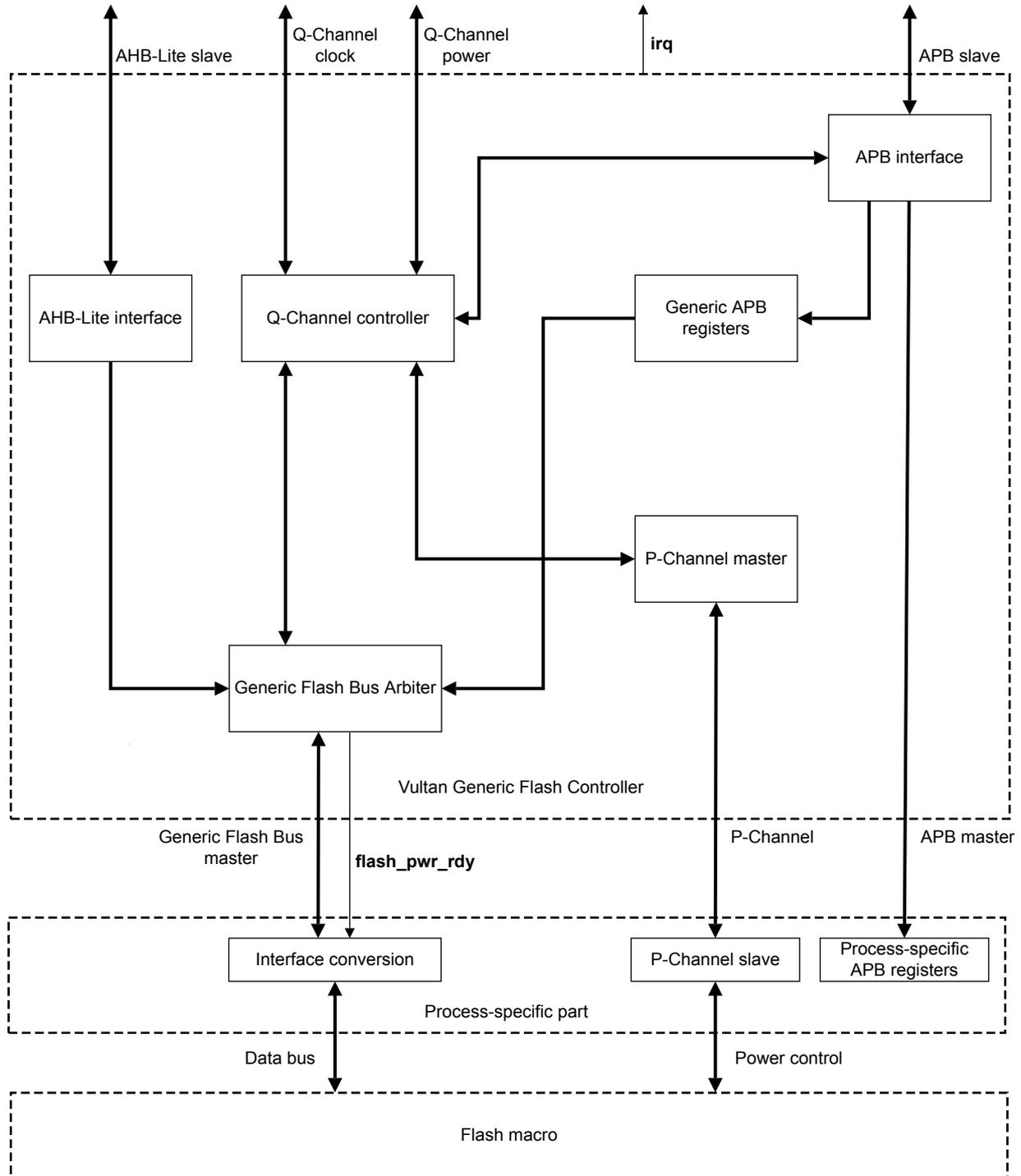


Figure 2-1 Vultan internal structure

2.2 Interfaces

Vultan has several interfaces that enable it to communicate with the system and the process-specific part.

This section contains the following subsections:

- [2.2.1 AHB-Lite slave interface on page 2-22.](#)
- [2.2.2 APB slave interface on page 2-23.](#)
- [2.2.3 APB master interface on page 2-24.](#)
- [2.2.4 Generic Flash Bus on page 2-24.](#)
- [2.2.5 Q-Channel interface for clock on page 2-25.](#)
- [2.2.6 Q-Channel interface for power on page 2-25.](#)
- [2.2.7 P-Channel master interface on page 2-26.](#)
- [2.2.8 System interface on page 2-26.](#)

2.2.1 AHB-Lite slave interface

The AHB-Lite slave interface is a read-only port through which the system can make read-only accesses to the embedded Flash. System accesses through the AHB-Lite slave interface are then transferred over the *Generic Flash Bus* (GFB).

Data width

The data width is fixed at 128-bits. Read accesses that are less than 128-bits are refused and generate an error response. Write accesses of any size are refused and generate an error response.

The address of the access does not have to be 128-bit aligned, because it is forwarded to the GFB directly without any restriction. The GFB description defines that, for wider data widths, the lower address bits are ignored. Therefore, it is implicit that the address is aligned to 128 bits irrespective of what address is sent over the AHB-Lite slave interface. Although the AHB specification defines that the address must be aligned for all bursts, and single bursts, Vultan does not generate an error if an access is not aligned.

Address width

The address width is fixed to allow for access to a 4MB memory area. Depending on the attached process-specific part, the lower 2MB memory region can be a main area, and the upper 2MB region can be an extended area.

The address bus is 22-bits wide. The *Most Significant Bit* (MSB) can be used to select between the main and extended areas, depending on the attached process-specific part.

See [3.2.1 AHB-Lite slave interface memory map on page 3-34](#) for a description of the AHB-Lite slave interface memory map.

Burst transfers

Burst transfers are supported to allow large blocks of data to be read from memory. Vultan supports all burst types that are specified in the AMBA 3 AHB-Lite specification. Burst transfers ensure that Read accesses from the memory have priority above any other commands, and are executed at the same speed as commands are executed in the process-specific part.

Locked transfers

The system AHB master can use locked transfers to control the Vultan arbitration scheme. Access is granted to the AHB-Lite slave port only, and all APB accesses are blocked until the entire locked transfer finishes. This ensures that accesses through the AHB-Lite slave port have deterministic response times.

Delayed response

The response time through the AHB-Lite slave interface is delayed by asserting the **hready** signal LOW. The following conditions assert **hready**:

- A delayed GFB transfer response, because the embedded Flash is slow to respond.
- The GFB Arbiter block selects a different requestor to access the memory.
- A Q-Channel interface is in Q_STOPPED state.
- The GFB has a transfer in progress.

Note

A Flash Write or Erase operation can require millions of clock cycles that can significantly block the AHB interconnect.

Error response

The following conditions can generate an error response:

- A Write access that is initiated by the AHB master.
- An access is set to a data width that is not 128 bits.
- A GFB transfer generates an error response.

AHB-Lite slave interface signals

The signals for the AHB-Lite slave interface are described in [A.1 AHB-Lite slave interface signals](#) on page Appx-A-57.

2.2.2 APB slave interface

The APB slave interface enables Write and Erase access to the main and extended areas of embedded Flash. It also acts as a control port for Vultan and the Flash macro.

The APB slave interface programs the internal generic APB registers to enable Write, Erase, and debug Read access to the embedded Flash over the GFB interface.

The APB slave interface allows direct access to an external register interface to configure Flash interface access times. The external register interface might be located in the process-specific part.

Address width

The address width is fixed at 13 bits to allow for a 2 x 4KB address space for internal registers, and for external registers in the process-specific part. The MSB, **paddr_s[12]** selects either internal or external accesses.

See [3.2.2 APB slave interface memory map](#) on page 3-36 for a description of the APB slave interface memory map.

Strobe signals

The strobe signals are checked for writes to ensure that all bits are set to 1 to indicate 32-bit word accesses, otherwise the write is ignored and has no effect on the registers. The strobe signals are forwarded to the downstream APB master in case the process-specific part supports byte access.

Delayed response

Accesses to the internal register bank are served without delay. The design of the process-specific register bank determines how much delay can be expected for accesses that target its interface.

Error response

APB accesses to the internal register bank always give an OK response. For reserved addresses, the data returned is 0. The response behavior to external register accesses depends on the implementation of the attached process-specific register bank. Therefore, any errors received on the APB master interface are forwarded through the APB slave interface to the access initiator.

APB slave interface signals

The signals for the APB slave interface are described in [A.2 APB slave interface signals](#) on page Appx-A-59.

2.2.3 APB master interface

The APB master interface is a control port for any registers that are connected externally that control the behavior of the process-specific part.

Address width

The address width is 12 bits, and allows access to a 4KB address space.

Strobe signals

Strobe signals can be used to access external registers at byte level. Each strobe signal corresponds to a byte within the 32-bit write data.

Delayed response

Vultan drives transfers through the APB master interface that are based on the incoming accesses from the APB slave interface. Any standard APB slave devices can be attached to the APB master interface. Therefore, how the slave delays accesses is IMPLEMENTATION-DEFINED.

Error response

Error responses received by the Vultan APB master interface are forwarded to the Vultan APB slave interface. The conditions that determine when the attached APB slave responds with errors is IMPLEMENTATION-DEFINED.

APB master interface signals

The signals for the APB master interface are described in [A.3 APB master interface signals](#) on page Appx-A-60.

2.2.4 Generic Flash Bus

The Generic Flash Bus enables Vultan to access embedded Flash.

Commands are received from the system through the AHB-Lite and APB slave interfaces, and converted internally to transfers over the GFB. A process-specific part that is connected to the GFB is expected to serve as a slave while Vultan is the master.

Command bus

Vultan supports all commands that are specified in the *Generic Flash Bus Description*.

Address width

The GFB allows access to a 4MB memory area. The address MSB allows access to an extended region that might be present in the Flash macro. Depending on how the process-specific part handles addresses, the MSB separates the 4MB memory into two halves so that a 2MB main memory and an extended region can be supported. The extended region is assumed to be smaller than 2MB, and therefore, **faddr[21]** is sufficient to access the entire extended region.

Data width

The GFB supports a fixed Read data width of 128 bits, which matches the data width of the AHB-Lite slave interface. Write data is sent over a 32-bit **fwdata** bus. A full width data write of 128 bits is sent over the GFB as four separate writes. Bits **faddr[3:2]** select the location of the 32-bit Write data within the 128-bit data word in little endian order.

Delayed response

Vultan is the GFB master and drives commands over the GFB. A GFB slave is allowed to delay its response to commands. The process-specific part can take several cycles to respond to Reads. However, Write and Erase commands can take many more cycles to execute.

Error response

Failed transactions can generate a two-cycle error response from the process-specific part.

Aborting commands

Commands initiated by the GFB master can be aborted by the **fabort** signal. The signal is driven from the APB registers. The method used by the process-specific part to support the abort function depends on its implementation.

Note

- Commands from the AHB-Lite slave interface cannot be aborted.
 - For more detailed information about GFB transactions, see the *Arm® Generic Flash Bus Description*.
-

Generic Flash Bus signals

The signals for the Generic Flash Bus are described in [A.4 Generic Flash Bus signals](#) on page Appx-A-61.

2.2.5 Q-Channel interface for clock

The Q-Channel interface for clock is a control port for managing the system clock.

Vultan can accept or deny a request from the clock manager to turn off all operations that are using the clock.

Activity indication

Vultan uses the **qactive_clk** signal to indicate it has an activity that is using the clock. When Vultan asserts **qactive_clk**, the clock manager knows that the clock must stay enabled.

Accepting a request

When Vultan has no activity, it accepts the quiescence request for the clock, and enters the Q_STOPPED state.

Denying a request

When the clock manager requests the clock to be disabled, if the activity indication is HIGH, Vultan denies the request.

Q-Channel interface for clock signals

The signals for the Q-Channel interface for clock are described in [A.5 Q-Channel interface for clock signals](#) on page Appx-A-63.

2.2.6 Q-Channel interface for power

The Q-Channel interface for power is a control port for managing the system power.

Vultan can accept or deny a request from the Power Policy Unit (PPU) to turn off all operations that are using power.

Activity indication

Vultan uses the **qactive_pwr** signal to indicate that it has an activity that is using power. When Vultan asserts **qactive_pwr**, the Power Policy Unit knows that the power must stay enabled.

Accepting a request

When Vultan or the embedded Flash is inactive, Vultan accepts the quiescence request for power, and enters the Q_STOPPED state. Vultan must then drive the process-specific part to a powerdown state before accepting the Q-Channel request.

Denying a request

When the Power Policy Unit requests power to be disabled, if the activity indication is HIGH, Vultan denies the request.

Q-Channel interface for power signals

The signals for the Q-Channel interface for power are described in [A.6 Q-Channel interface for power signals](#) on page Appx-A-64.

2.2.7 P-Channel master interface

The P-Channel master interface is a control port for managing the power of the attached process-specific part.

Vultan can request changes to the power state when it accepts a powerdown request from the system-level Power Policy Unit. When Vultan leaves reset and is not in quiescent state, it enables power to embedded Flash automatically.

Vultan supports two power states, all powerup, and all powerdown.

State encoding

Vultan uses a single bit to encode the two Flash macro power states of either all powerdown (0), or all powerup (1).

Activity encoding

The **active** signal indicates the required power state of a particular activity. The **active** signal is expected to be driven by the process-specific part to force power to remain active, otherwise **active** is 0.

Responding to a request

Vultan initiates all activity, and does not expect the process-specific part to assert a request denial. However, Vultan can support request denial if it is used by the process-specific part. The time taken by the process-specific part to accept a request depends on its implementation.

P-Channel master interface signals

The signals for the P-Channel master interface are described in [A.7 P-Channel master interface signals](#) on page Appx-A-65.

2.2.8 System interface

The Vultan system interface comprises several system I/O ports that are separate from its interfaces.

Interrupt request

Vultan generates an interrupt request that indicates to the system master that an important event in Vultan has occurred. An interrupt is active-HIGH. It is cleared by accessing registers in Vultan, and acknowledging the reason for the interrupt, see [3.4.4 IRQ_STATUS_CLR](#) on page 3-44.

Flash Power Ready

When the P-Channel master sets the embedded Flash power to ON, Vultan asserts the **flash_pwr_rdy** signal. This signal is sent to the process-specific part GFB slave so that it can initiate any start-up sequence that needs the embedded Flash to be fully functional.

All the other interfaces operate without any restrictions. The transfers from the AHB-Lite slave interface are forwarded to the GFB, but transfers are blocked when the command ready indication signal **fready** is pulled LOW until the initialization finishes.

System interface signals

The signals for the System interface are described in [A.8 System interface signals](#) on page Appx-A-66.

2.3 Clocking

Vultan has a single clock domain, which is used for clocking all internal registers.

Synchronous interfaces

The Vultan AHB-Lite slave, APB, and GFB interfaces are expected to run on the same clock.

Asynchronous interfaces

The Vultan Q-Channel and P-Channel interfaces might be clocked asynchronously. Therefore, these interfaces have internal synchronizers.

2.4 Resets

Vultan has a single reset input. Vultan expects that this reset is synchronized externally.

The reset input signal must be synchronous with the clock. Vultan expects the reset to be asserted asynchronously and deasserted synchronously.

2.5 Interrupt sources

The internal generic APB registers in Vultan generate an interrupt request that is triggered by several sources.

Command Accept (CMD_ACCEPT_IRQ)

CMD_ACCEPT_IRQ can speed up APB master accesses to the embedded Flash. CMD_ACCEPT_IRQ is set when the CTRL register accepts a Write command, asserts the **fready** signal on the GFB interface, and Flash executes the command. In this state, a new command can be written to the CTRL register where it enters a pending state, enabling back-to-back transfers to be executed. This behavior enables the use of the ROW WRITE command on the GFB.

When back-to-back transfers are used and if CMD_ACCEPT_IRQ and CMD_SUCCESS_IRQ or CMD_FAIL_IRQ are both set, both interrupt sources trigger together.

Command Success (CMD_SUCCESS_IRQ)

The CMD_SUCCESS_IRQ indicates the successful finish of an executed command. It makes the driver aware of the result immediately the command finishes, and prevents new transfers from the APB side being written to the CTRL register.

Command Fail (CMD_FAIL_IRQ)

The CMD_FAIL_IRQ indicates the unsuccessful finish of an executed command. It makes the driver aware of the result immediately the command finishes, and prevents new transfers from the APB side being written to the CTRL register.

Command Reject (CMD_REJECT_IRQ)

The CMD_REJECT_IRQ indicates a programming fault when software tries to modify the content of the CTRL, ADDR, and DATA0 registers either while a command is pending in the CTRL register, or when one or more interrupts are pending. The Write is ignored but CMD_REJECT_IRQ indicates the error for debug purposes.

Read Overflow (READ_OVERFLOW_IRQ)

The READ_OVERFLOW_IRQ indicates that the result of a Read command cannot be updated immediately in the IRQ_STATUS register, and the Read enters the finished state. Hardware assumes this state to be an overflow situation because the new data is lost, and the hardware cannot be sure that all previous results were read by software.

Interrupt request signal

The interrupt request signal is described in [A.8 System interface signals](#) on page Appx-A-66.

2.6 Generic Flash Bus Arbiter

The GFB Arbiter selects requests from either the AHB-Lite slave interface, or the APB slave interface, and forwards them to the Generic Flash Bus. A simple grant-request mechanism selects the appropriate port.

Arbitration scheme

The GFB Arbiter uses a round-robin arbitration scheme between the AHB-Lite slave and APB slave ports. When the AHB-Lite slave port sends a burst or locked transfer, the AHB-Lite slave port has priority over the APB slave port until the transfer finishes.

The incoming APB and AHB requests cannot be always guaranteed to follow the arbitration scheme, for example, due to the occurrence of LPI requests, or the GFB interface is busy when incoming requests arrive. In these cases, outgoing transfers might occur in a different order to the incoming requests.

Chapter 3

Programmers model

Read this for a description of the memory map and registers, and for information about programming the device.

It contains the following sections:

- [3.1 About this programmers model](#) on page 3-33.
- [3.2 Memory maps](#) on page 3-34.
- [3.3 Register summary](#) on page 3-41.
- [3.4 Register descriptions](#) on page 3-42.

3.1 About this programmers model

The following information applies to the Vultan registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- Unless otherwise stated in the accompanying text:
 - Do not modify undefined register bits.
 - Ignore undefined register bits on reads.
 - All register bits are reset to the reset value specified in the [3.3 Register summary on page 3-41](#).
- Access type is described as follows:

RW Read and write.

RO Read only.

WO Write only.

3.2 Memory maps

Vultan has several memory maps that are attached to different interfaces.

The memory map locations are:

- AHB-Lite slave interface, 4MB (2MB main memory area, 2MB extended memory area).
- GFB, 4MB (2MB main memory area, 2MB extended memory area).
- APB slave interface, 8KB.
- APB master interface, 4KB.
- Internal registers, 4KB.

This section contains the following subsections:

- [3.2.1 AHB-Lite slave interface memory map on page 3-34.](#)
- [3.2.2 APB slave interface memory map on page 3-36.](#)
- [3.2.3 Accessing Flash from the APB slave port on page 3-36.](#)
- [3.2.4 Preloading transfers on page 3-38.](#)

3.2.1 AHB-Lite slave interface memory map

The AHB-Lite slave interface can access 4MB of memory space where the contents of the 4MB, or smaller, embedded Flash can be read. The embedded Flash can have an extra extended memory range that can be reachable on the upper 2MB of the given 4MB address range.

The size of access is fixed at `0b100`. Therefore, all wrapping boundaries must be calculated based on the wrap size and the access size having a total value of 4 for 128-bit accesses. Access size is indicated by signal `hsize[2:0]`, see [A.1 AHB-Lite slave interface signals on page Appx-A-57.](#)

The following figure is an example that uses 1MB of large Flash with 4KB page size and two Extension pages. The example shows how the embedded Flash space can be reached through different routes.

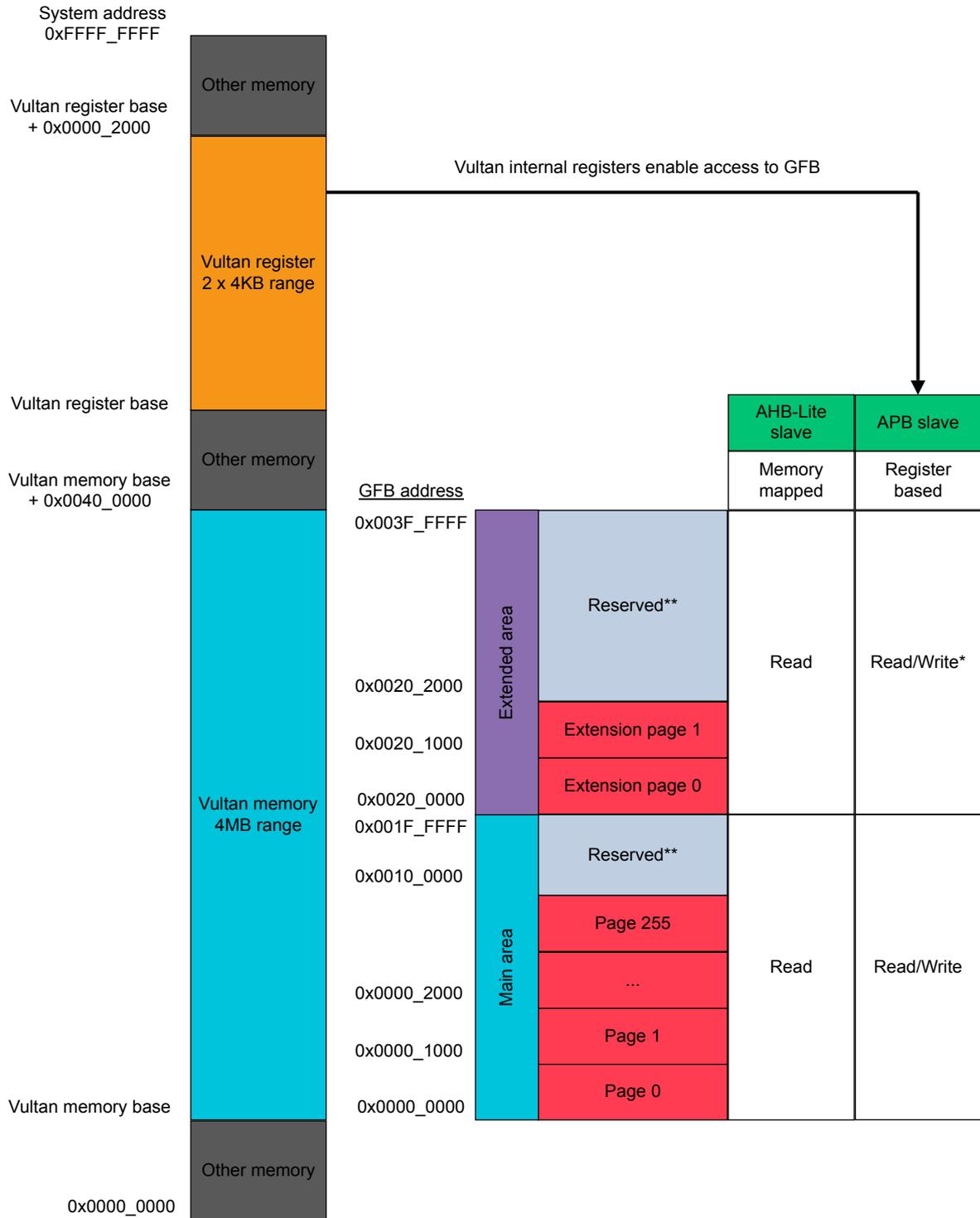


Figure 3-1 AHB-Lite slave interface memory map

Note

* Access rights depend on the properties of extension pages that are handled by the process-specific part. Vultan supports both Read and Write, or Erase.

** Aliasing might occur if the process-specific part maps the same addresses multiple times within the 2MB address space of the main area, or the 2MB address space of the extended area.

The example memory map has 256 pages in the main memory area. Vultan maps 4MB of address space from the system memory to the GFB address range. The first 1MB is mapped to the embedded Flash main area directly, the second 1MB is reserved. The extended memory area has 8KB of space that is mapped to the extended memory area from the third 1MB. The last 1MB is reserved.

3.2.2 APB slave interface memory map

The APB slave interface enables the system-side to access 2 x 4KB of register space.

The following figure is an example memory map for the APB slave interface.

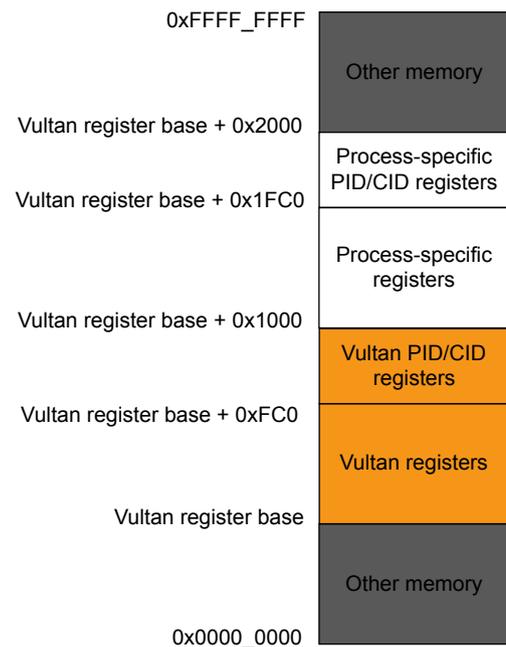


Figure 3-2 APB slave interface memory map example

The memory map example shows that Vultan and the process-specific part can be considered to be two separate peripherals in the system.

3.2.3 Accessing Flash from the APB slave port

Software must follow a particular flow to access the embedded Flash from the APB slave interface.

The following figure shows how to access the embedded Flash from the APB slave interface.

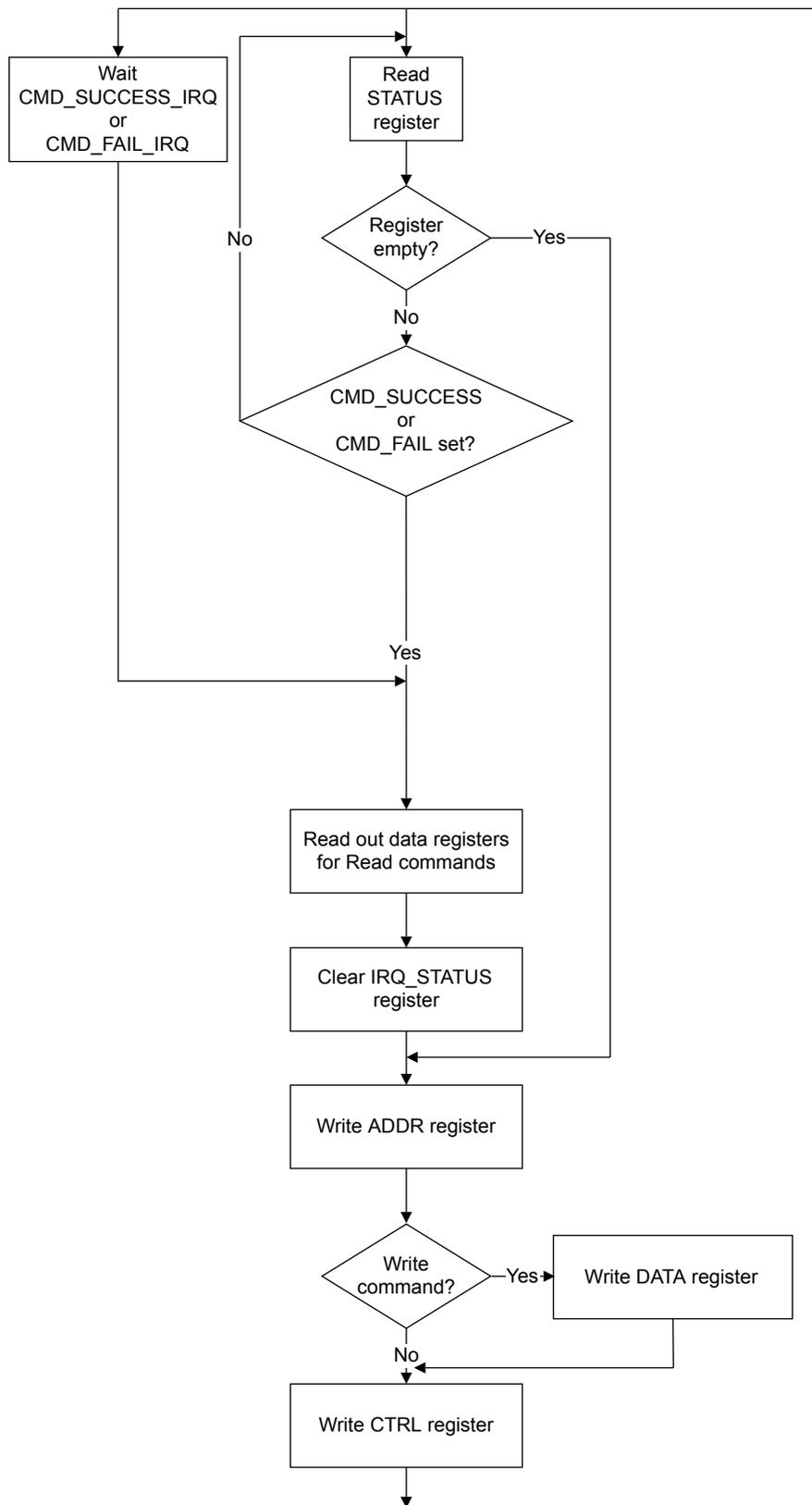


Figure 3-3 APB slave access to Flash flowchart

The APB slave interface can use either of two modes to trigger a new command.

Method 1 allows software to poll the STATUS register for any active bits, where an active bit indicates that Vultan is processing a command.

Method 2 enables CMD_SUCCESS_IRQ and CMD_FAIL_IRQ, and waits for either interrupt to be triggered. This event occurs when a command enters the SUCCESS or FAIL state. To initiate a new transfer, software must write to the ADDR, DATA0, and CTRL registers. The CTRL register must be written last because it triggers access to the embedded Flash. The command then enters the queue until it is executed.

3.2.4 Preloading transfers

To improve the efficiency of ROW WRITE commands that utilize the Command queue, transfers can be preloaded to the CTRL register by enabling either CMD_ACCEPT_IRQ, CMD_SUCCESS_IRQ, or CMD_FAIL_IRQ, or any combination of the three.

When the APB master wants to utilize a ROW WRITE command with improved efficiency, it must send the command for Flash transfers back-to-back, and prevent Flash returning from the high-voltage programming state.

The following figure shows the ROW WRITE timing diagram.

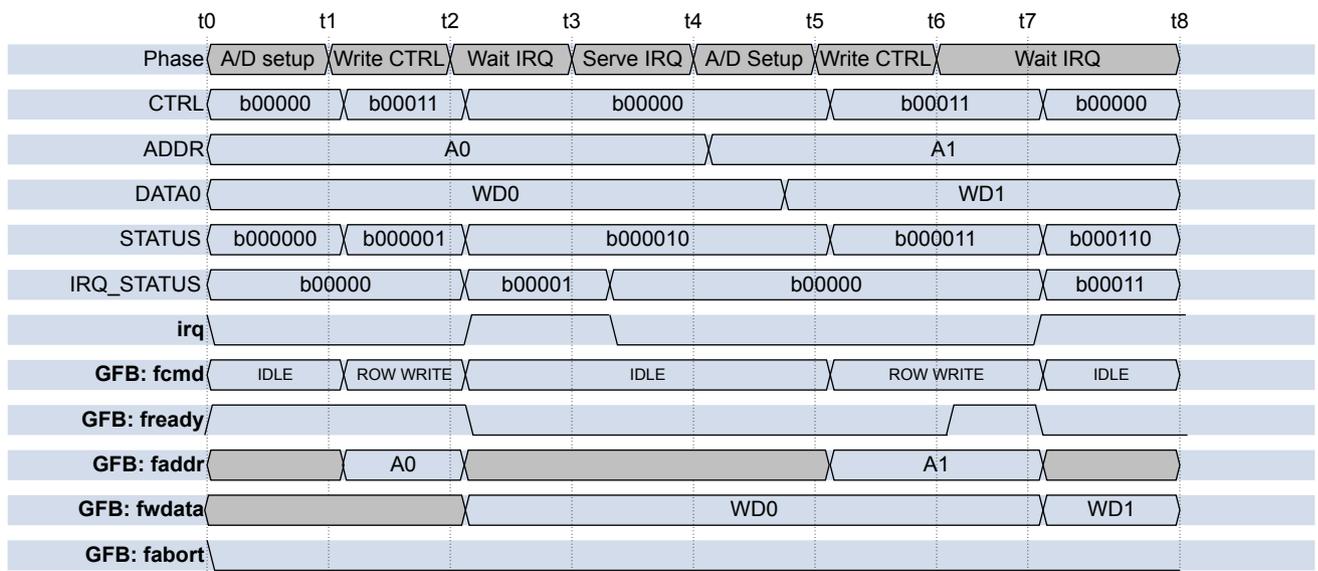


Figure 3-4 ROW WRITE timing diagram

ROW WRITE command sequence

The ROW WRITE command sequence is divided into different phases that are the actions required by software to support the preloading mechanism.

The A/D setup phase comprises write updates to the contents of the ADDR and DATA0 registers for the upcoming command.

In the Write CTRL phase, when ready, software writes the CTRL register and initiates the transfer. The timing diagram shows that the GFB interface is ready to accept the command at t2 immediately, and the command then passes to the GFB.

After t2, hardware clears the CTRL register automatically, and asserts a CMD_ACCEPT_IRQ.

Software is in the Wait IRQ phase until it receives the interrupt, and then enters the Serve IRQ phase where it must acknowledge the interrupt by clearing the IRQ_STATUS register. When the IRQ_STATUS register clears, the address and data for the next transfer can be set up before writing the next ROW WRITE command to the CTRL register.

To ensure that the preload mechanism works correctly, the Write CTRL phase in t_5 must finish before t_7 , when the GFB interface is ready with the next currently executed command, and able to accept new commands. Otherwise, an IDLE command is inserted in the ROW WRITE sequence, and the benefits of the ROW WRITE command are not used. Write commands for the embedded Flash are expected to require thousands of clock cycles, software can use this period to serve the CMD_ACCEPT_IRQ and update all necessary registers.

Read considerations

The CMD_ACCEPT_IRQ is intended for ROW WRITE commands. However, all other commands can use this interrupt because the hardware implementation cannot restrict its use.

In situations when the software driver uses CMD_ACCEPT_IRQ for READ commands, the Read data of the current READ command might be lost if the resulting interrupts are not processed in time. If this occurs, a READ_OVERFLOW_IRQ indicates this error. Reads through the APB slave interface and the internal generic APB registers are intended for debug purposes only, and therefore the CMD_SUCCESS_IRQ and CMD_FAIL_IRQ can be used for Reads. When higher efficiency Reads are needed, you must execute AHB Reads instead.

The following figure shows the ROW WRITE preloading flowchart.

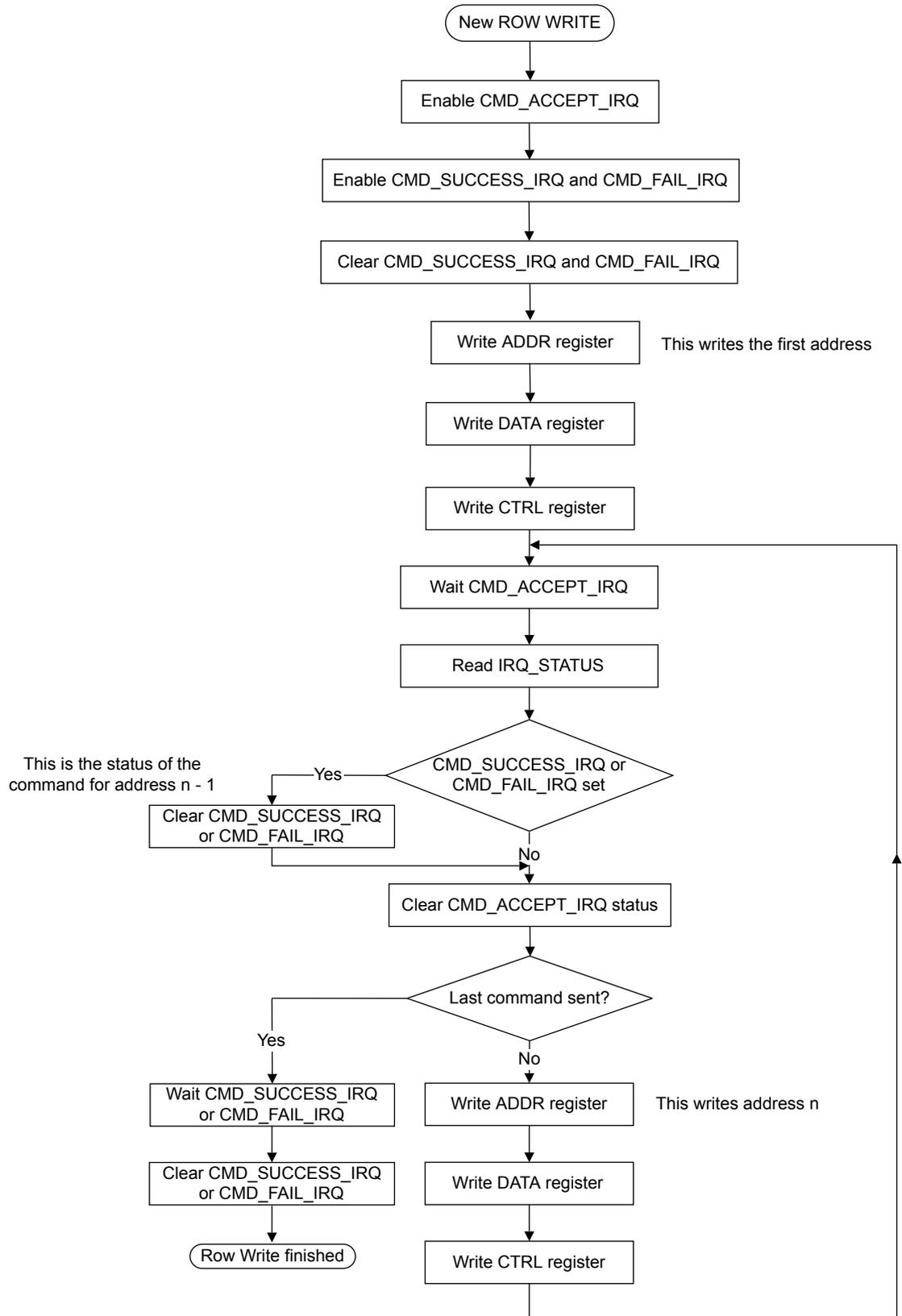


Figure 3-5 ROW WRITE preloading flowchart

3.3 Register summary

The following table shows the registers in offset order from the base memory address.

Table 3-1 Register summary

| Offset | Name | Type | Reset | Width | Description |
|--------|-------------------|------|-------|-------|--|
| 0x000 | IRQ_ENABLE_SET | RW | 0x0 | 32 | 3.4.1 IRQ_ENABLE_SET on page 3-42 |
| 0x004 | IRQ_ENABLE_CLR | RW | 0x0 | 32 | 3.4.2 IRQ_ENABLE_CLR on page 3-43 |
| 0x008 | IRQ_STATUS_SET | RW | 0x0 | 32 | 3.4.3 IRQ_STATUS_SET on page 3-43 |
| 0x00C | IRQ_STATUS_CLR | RW | 0x0 | 32 | 3.4.4 IRQ_STATUS_CLR on page 3-44 |
| 0x010 | IRQ_MASKED_STATUS | RO | 0x0 | 32 | 3.4.5 IRQ_MASKED_STATUS on page 3-45 |
| 0x014 | CTRL | RW | 0x0 | 32 | 3.4.6 CTRL on page 3-46 |
| 0x018 | STATUS | RO | 0x0 | 32 | 3.4.7 STATUS on page 3-47 |
| 0x01C | ADDR | RW | 0x0 | 32 | 3.4.8 ADDR on page 3-48 |
| 0x020 | DATA0 | RW | 0x0 | 32 | 3.4.9 DATA0 on page 3-49 |
| 0x024 | DATA1 | RO | 0x0 | 32 | 3.4.10 DATA1 on page 3-49 |
| 0x028 | DATA2 | RO | 0x0 | 32 | 3.4.11 DATA2 on page 3-50 |
| 0x02C | DATA3 | RO | 0x0 | 32 | 3.4.12 DATA3 on page 3-50 |
| 0xFD0 | PIDR4 | RO | 0x4 | 32 | 3.4.13 PIDR4 on page 3-51 |
| 0xFE0 | PIDR0 | RO | 0x32 | 32 | 3.4.14 PIDR0 on page 3-51 |
| 0xFE4 | PIDR1 | RO | 0xB8 | 32 | 3.4.15 PIDR1 on page 3-52 |
| 0xFE8 | PIDR2 | RO | 0xB | 32 | 3.4.16 PIDR2 on page 3-52 |
| 0xFEC | PIDR3 | RO | 0x0 | 32 | 3.4.17 PIDR3 on page 3-53 |
| 0xFF0 | CIDR0 | RO | 0xD | 32 | 3.4.18 CIDR0 on page 3-53 |
| 0xFF4 | CIDR1 | RO | 0xF0 | 32 | 3.4.19 CIDR1 on page 3-54 |
| 0xFF8 | CIDR2 | RO | 0x5 | 32 | 3.4.20 CIDR2 on page 3-55 |
| 0xFFC | CIDR3 | RO | 0xB1 | 32 | 3.4.21 CIDR3 on page 3-55 |

3.4 Register descriptions

This section describes the Vultan registers.

[3.3 Register summary on page 3-41](#) provides cross references to individual registers.

3.4.1 IRQ_ENABLE_SET

Interrupt enable set register for enabling generation of interrupts for different sources.

The IRQ_ENABLE_SET register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|------------|
| Offset | 0x000 |
| Type | Read-write |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

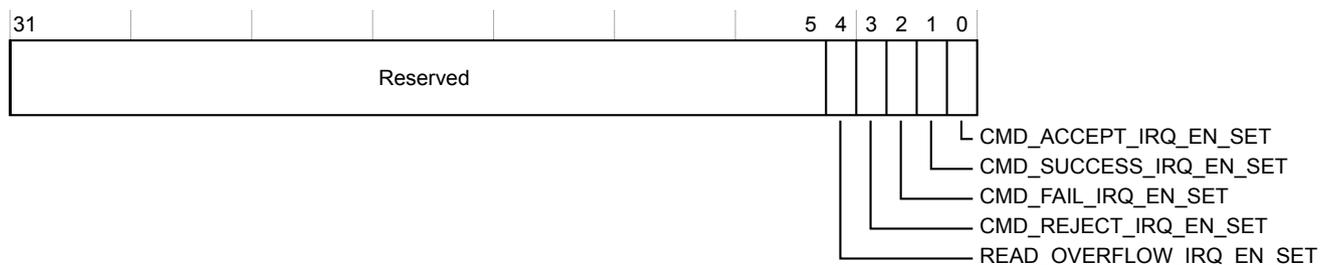


Figure 3-6 IRQ_ENABLE_SET register bit assignments

The following list shows the register bit assignments.

[4] READ_OVERFLOW_IRQ_EN_SET

Enables READ_OVERFLOW_IRQ bit to generate interrupts on the interrupt line. Write to 1 sets the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[3] CMD_REJECT_IRQ_EN_SET

Enables CMD_REJECT_IRQ bit to generate interrupts on the interrupt line. Write to 1 sets the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[2] CMD_FAIL_IRQ_EN_SET

Enables CMD_FAIL_IRQ bit to generate interrupts on the interrupt line. Write to 1 sets the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[1] CMD_SUCCESS_IRQ_EN_SET

Enables CMD_SUCCESS_IRQ bit to generate interrupts on the interrupt line. Write to 1 sets the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[0] CMD_ACCEPT_IRQ_EN_SET

Enables CMD_ACCEPT_IRQ bit to generate interrupts on the interrupt line. Write to 1 sets the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

3.4.2 IRQ_ENABLE_CLR

Interrupt enable clear register for disabling generation of interrupts for different sources.

The IRQ_ENABLE_CLR register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|------------|
| Offset | 0x004 |
| Type | Read-write |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

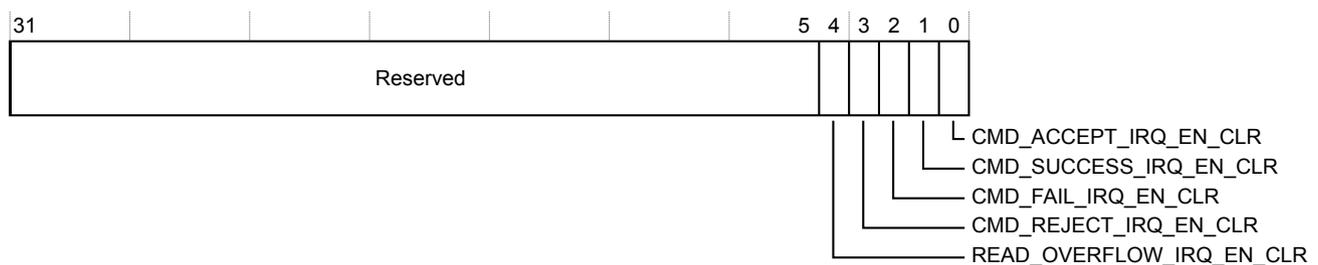


Figure 3-7 IRQ_ENABLE_CLR register bit assignments

The following list shows the register bit assignments.

[4] READ_OVERFLOW_IRQ_EN_CLR

Disables READ_OVERFLOW_IRQ bit. Write to 1 clears the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[3] CMD_REJECT_IRQ_EN_CLR

Disables CMD_REJECT_IRQ bit. Write to 1 clears the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[2] CMD_FAIL_IRQ_EN_CLR

Disables CMD_FAIL_IRQ bit. Write to 1 clears the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[1] CMD_SUCCESS_IRQ_EN_CLR

Disables CMD_SUCCESS_IRQ bit. Write to 1 clears the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

[0] CMD_ACCEPT_IRQ_EN_CLR

Disables CMD_ACCEPT_IRQ bit. Write to 1 clears the interrupt enable. Write to 0 has no effect. Reading this bit shows the current status of the enable bit.

3.4.3 IRQ_STATUS_SET

Interrupt status set register for activating different interrupt sources.

The IRQ_STATUS_SET register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-------|
| Offset | 0x008 |
|---------------|-------|

Type Read-write
Reset 0x0
Width 32

The following figure shows the bit assignments.

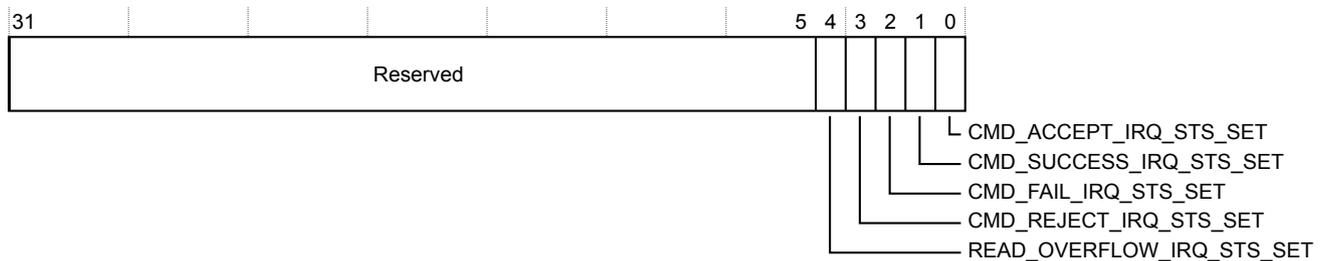


Figure 3-8 IRQ_STATUS_SET register bit assignments

The following list shows the register bit assignments.

[4] READ_OVERFLOW_IRQ_STS_SET

Sets the status of the READ_OVERFLOW_IRQ bit. Write to 1 sets the interrupt. Write to 0 has no effect. Can be used to force the set status of this interrupt bit for debug purposes. Reading this bit shows the current status of the interrupt bit regardless of the interrupt enable setting.

[3] CMD_REJECT_IRQ_STS_SET

Sets the status of the CMD_REJECT_IRQ bit. Write to 1 sets the interrupt. Write to 0 has no effect. Can be used to force the set status of this interrupt bit for debug purposes. Reading this bit shows the current status of the interrupt bit regardless of the interrupt enable setting.

[2] CMD_FAIL_IRQ_STS_SET

Sets the status of the CMD_FAIL_IRQ bit. Write to 1 sets the interrupt. Write to 0 has no effect. Can be used to force the set status of this interrupt bit for debug purposes. Reading this bit shows the current status of the interrupt bit regardless of the interrupt enable setting.

Note

Write to 1 also sets STATUS.CMD_FAIL.

[1] CMD_SUCCESS_IRQ_STS_SET

Sets the status of the CMD_SUCCESS_IRQ bit. Write to 1 sets the interrupt. Write to 0 has no effect. Can be used to force the set status of this interrupt bit for debug purposes. Reading this bit shows the current status of the interrupt bit regardless of the interrupt enable setting.

Note

Write to 1 also sets STATUS.CMD_SUCCESS.

[0] CMD_ACCEPT_IRQ_STS_SET

Sets the status of the CMD_ACCEPT_IRQ bit. Write to 1 sets the interrupt. Write to 0 has no effect. Can be used to force the set status of this interrupt bit for debug purposes. Reading this bit shows the current status of the interrupt bit regardless of the interrupt enable setting.

3.4.4 IRQ_STATUS_CLR

Interrupt status clear register for acknowledging different interrupt sources.

The IRQ_STATUS_CLR register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0x00C
Type Read-write
Reset 0x0
Width 32

The following figure shows the bit assignments.



Figure 3-9 IRQ_STATUS_CLR register bit assignments

The following list shows the register bit assignments.

[4] READ_OVERFLOW_IRQ_STS_CLR

Clears the status of the READ_OVERFLOW_IRQ bit. Write to 1 clears the interrupt. Write to 0 has no effect. Reading this bit shows the current status of the interrupt bit.

[3] CMD_REJECT_IRQ_STS_CLR

Clears the status of the CMD_REJECT_IRQ bit. Write to 1 clears the interrupt. Write to 0 has no effect. Reading this bit shows the current status of the interrupt bit.

[2] CMD_FAIL_IRQ_STS_CLR

Clears the status of the CMD_FAIL_IRQ bit. Write to 1 clears the interrupt. Write to 0 has no effect. Reading this bit shows the current status of the interrupt bit.

[1] CMD_SUCCESS_IRQ_STS_CLR

Clears the status of the CMD_SUCCESS_IRQ bit. Write to 1 clears the interrupt. Write to 0 has no effect. Reading this bit shows the current status of the interrupt bit.

[0] CMD_ACCEPT_IRQ_STS_CLR

Clears the status of the CMD_ACCEPT_IRQ bit. Write to 1 clears the interrupt. Write to 0 has no effect. Reading this bit shows the current status of the interrupt bit.

3.4.5 IRQ_MASKED_STATUS

Interrupt status register that shows if each interrupt is pending and is the cause of the interrupt line being asserted.

The IRQ_MASKED_STATUS register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0x010
Type Read-only
Reset 0x0
Width 32

The following figure shows the bit assignments.

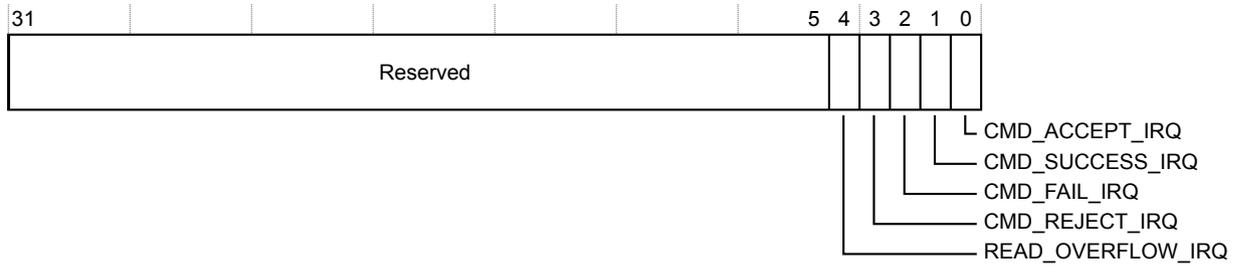


Figure 3-10 IRQ_MASKED_STATUS register bit assignments

The following list shows the register bit assignments.

[4] READ_OVERFLOW_IRQ

When this bit reads 1, it means that READ_OVERFLOW_IRQ is the cause of the interrupt line being asserted.

[3] CMD_REJECT_IRQ

When this bit reads 1, it means that CMD_REJECT_IRQ is the cause of the interrupt line being asserted.

[2] CMD_FAIL_IRQ

When this bit reads 1, it means that CMD_FAIL_IRQ is the cause of the interrupt line being asserted.

[1] CMD_SUCCESS_IRQ

When this bit reads 1, it means that CMD_SUCCESS_IRQ is the cause of the interrupt line being asserted.

[0] CMD_ACCEPT_IRQ

When this bit reads 1, it means that CMD_ACCEPT_IRQ is the cause of the interrupt line being asserted.

3.4.6 CTRL

Control register for initiating Flash accesses.

The CTRL register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|------------|
| Offset | 0x014 |
| Type | Read-write |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

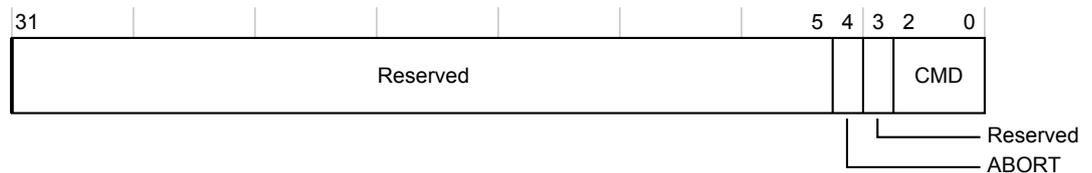


Figure 3-11 CTRL register bit assignments

The following list shows the register bit assignments.

[4] ABORT

Abort currently ongoing command initiated towards the embedded Flash from the APB interface. When ABORT is written, the CMD field is ignored. ABORT is effective only when STATUS.CMD_ACCEPT is set. Reading back ABORT field shows the status of the GFB interface **fabort** signal.

[2:0] CMD

Initiate a command to the embedded Flash for the address stored in the ADDR register and with the data stored in DATA0 register. For WRITE and ROW WRITE commands. Software must ensure that the state of the addressed word is cleared. The commands are:

- 001: READ
- 010: WRITE
- 011: ROW WRITE
- 100: ERASE
- 111: MASS ERASE
- 000: Reserved
- 101: Reserved
- 110: Reserved

Writing reserved commands has no effect. Reading back the CMD value shows the currently pending CMD. When the CMD is accepted, the CMD field is cleared to 0b000.

3.4.7 STATUS

Status register for showing the state of the embedded Flash accesses.

The STATUS register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0x018 |
| Type | Read-only |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.



Figure 3-12 STATUS register bit assignments

The following list shows the register bit assignments.

[5] ARBITRATION_LOCKED

Setting this bit means that the other interface keeps the arbitration in locked state and no APB commands are served until the locked state is removed. This can be used by the software to detect why the APB request is being held up.

[4] CMD_FINISH

Setting this bit means that the previously accepted command is finished but the result cannot be updated in the STATUS register because the interrupt bits are still set. When the result interrupt bits are cleared, the status can be updated and this bit is automatically cleared.

[3] CMD_FAIL

Setting this bit means that the previously accepted command has finished with a failure. Either this bit or the CMD_SUCCESS bit is set for the finished command. Updated when CMD_SUCCESS_IRQ or CMD_FAIL_IRQ is set. Valid until the CMD_SUCCESS_IRQ or CMD_FAIL_IRQ bit is cleared.

[2] CMD_SUCCESS

Setting this bit means that the previously accepted command has finished successfully. Either this bit or the CMD_FAIL bit is set for the finished command. Updated when CMD_SUCCESS_IRQ or CMD_FAIL_IRQ is set. Valid until the CMD_SUCCESS_IRQ or CMD_FAIL_IRQ bit is cleared.

[1] CMD_ACCEPT

Selecting CMD_PENDING to be executed towards the embedded Flash sets this bit to 1 and means that the command is being forwarded to the embedded Flash. Set to 1 when the CMD_PENDING bit is cleared. Cleared when the command has finished and the CMD_SUCCESS_IRQ or CMD_FAIL_IRQ status bits are cleared for the previous transfer. If the status results of the previous command are not cleared, the command might still be finished for the embedded Flash, but the results cannot update until the previous results are cleared. If an already pending command is accepted when the current command has finished, this bit stays asserted.

[0] CMD_PENDING

When the CTRL register is written, the command goes into the arbitration queue and waits to be arbitrated towards the embedded Flash. This bit is set when the command is initiated but still pending in the queue, and is cleared when the command is arbitrated and accepted by the embedded Flash. This bit is also set when ABORT is written to the CTRL register, and is cleared when the aborted command has finished.

3.4.8 ADDR

Address register for the embedded Flash access.

The ADDR register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|------------|
| Offset | 0x01C |
| Type | Read-write |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.



Figure 3-13 ADDR register bit assignments

The following list shows the register bit assignments.

[21:0] ADDR

The 22-bit wide byte address for the current Flash access, allows 2MB address range for the main area and 2MB address range for the extended area.

The MSB Addr[21] selects between memory ranges:

0: Main area

1: Extended area

When selecting the extended area for mass erase commands, both main and extended areas are cleared. Otherwise, only the main area is cleared. The contents of the extended area remain unchanged after the mass erase finishes.

3.4.9 DATA0

Data register for the embedded Flash access containing read and write data bits[31:0].

The DATA0 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|------------|
| Offset | 0x020 |
| Type | Read-write |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

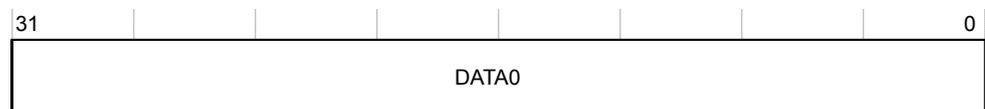


Figure 3-14 DATA0 register bit assignments

The following list shows the register bit assignments.

[31:0] DATA0

Data register for the embedded Flash access has read and write data bits[31:0].

3.4.10 DATA1

Data register for the embedded Flash access containing read data bits[63:32].

The DATA1 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0x024 |
| Type | Read-only |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

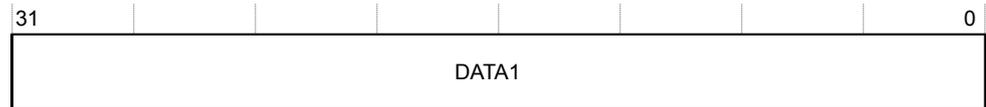


Figure 3-15 DATA1 register bit assignments

The following list shows the register bit assignments.

[31:0] DATA1

Data register for the embedded Flash access has read data bits[63:32].

3.4.11 DATA2

Data register for the embedded Flash access containing read data bits[95:64].

The DATA2 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0x028 |
| Type | Read-only |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

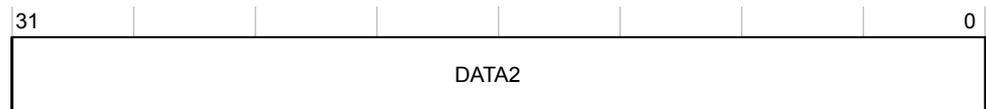


Figure 3-16 DATA2 register bit assignments

The following list shows the register bit assignments.

[31:0] DATA2

Data register for the embedded Flash access has read data bits[95:64].

3.4.12 DATA3

Data register for the embedded Flash access containing read data bits[127:96].

The DATA3 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0x02C |
| Type | Read-only |
| Reset | 0x0 |
| Width | 32 |

The following figure shows the bit assignments.

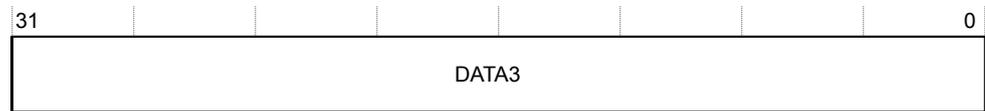


Figure 3-17 DATA3 register bit assignments

The following list shows the register bit assignments.

[31:0] DATA3

Data register for the embedded Flash access has read data bits[127:96].

3.4.13 PIDR4

Peripheral identification register byte[4]. Returns byte[4] of the peripheral ID. The PIDR4 register is part of the set of peripheral identification registers.

The PIDR4 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0xFD0 |
| Type | Read-only |
| Reset | 0x4 |
| Width | 32 |

The following figure shows the bit assignments.



Figure 3-18 PIDR4 register bit assignments

The following list shows the register bit assignments.

[7:4] SIZE

4KB Count.

[3:0] DES_2

JEP106 Continuation Code.

3.4.14 PIDR0

Peripheral identification register byte[0]. Returns byte[0] of the peripheral ID. The PIDR0 register is part of the set of peripheral identification registers.

The PIDR0 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0xFE0 |
| Type | Read-only |
| Reset | 0x32 |
| Width | 32 |

The following figure shows the bit assignments.



Figure 3-19 PIDR0 register bit assignments

The following list shows the register bit assignments.

[7:0] PART_0
Part Number [7:0].

3.4.15 PIDR1

Peripheral identification register byte[1]. Returns byte[1] of the peripheral ID. The PIDR1 register is part of the set of peripheral identification registers.

The PIDR1 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0xFE4
Type Read-only
Reset 0xB8
Width 32

The following figure shows the bit assignments.

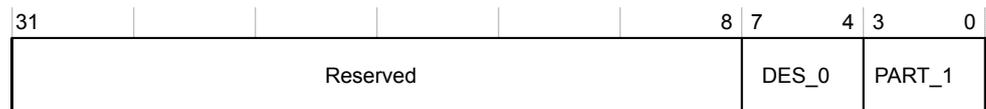


Figure 3-20 PIDR1 register bit assignments

The following list shows the register bit assignments.

[7:4] DES_0
JEP106 Identity Code [3:0].
[3:0] PART_1
Part Number [11:8].

3.4.16 PIDR2

Peripheral identification register 2. Returns byte[2] of the peripheral ID. The PIDR2 register is part of the set of peripheral identification registers.

The PIDR2 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0xFE8
Type Read-only
Reset 0xB

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0xFF0
Type Read-only
Reset 0xD
Width 32

The following figure shows the bit assignments.

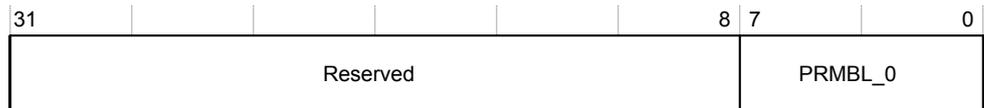


Figure 3-23 CIDR0 register bit assignments

The following list shows the register bit assignments.

[7:0] PRMBL_0

Preamble.

3.4.19 CIDR1

Component identification register 1. Returns byte[1] of the component ID. The CIDR1 register is part of the set of component identification registers.

The CIDR1 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

Offset 0xFF4
Type Read-only
Reset 0xF0
Width 32

The following figure shows the bit assignments.

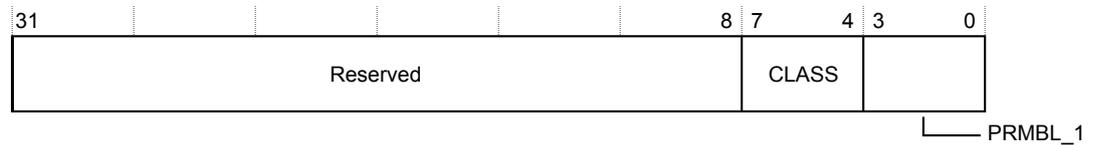


Figure 3-24 CIDR1 register bit assignments

The following list shows the register bit assignments.

[7:4] CLASS

Component class.

[3:0] PRMBL_1

Preamble.

3.4.20 CIDR2

Component identification register 2. Returns byte[2] of the component ID. The CIDR2 register is part of the set of component identification registers.

The CIDR2 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0xFF8 |
| Type | Read-only |
| Reset | 0x5 |
| Width | 32 |

The following figure shows the bit assignments.

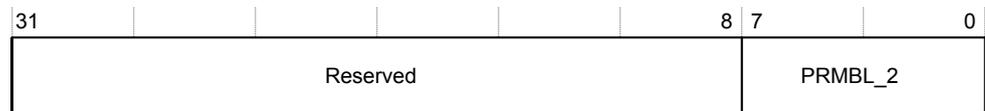


Figure 3-25 CIDR2 register bit assignments

The following list shows the register bit assignments.

[7:0] PRMBL_2

Preamble.

3.4.21 CIDR3

Component identification register 3. Returns byte[3] of the component ID. The CIDR3 register is part of the set of component identification registers.

The CIDR3 register characteristics are:

Usage constraints

There are no usage constraints.

Configurations

There is only one configuration.

Attributes

| | |
|---------------|-----------|
| Offset | 0xFFC |
| Type | Read-only |
| Reset | 0xB1 |
| Width | 32 |

The following figure shows the bit assignments.



Figure 3-26 CIDR3 register bit assignments

The following list shows the register bit assignments.

[7:0] PRMBL_3

Preamble.

Appendix A

Signal descriptions

Read this for a description of the input and output signals.

It contains the following sections:

- *A.1 AHB-Lite slave interface signals* on page Appx-A-57.
- *A.2 APB slave interface signals* on page Appx-A-59.
- *A.3 APB master interface signals* on page Appx-A-60.
- *A.4 Generic Flash Bus signals* on page Appx-A-61.
- *A.5 Q-Channel interface for clock signals* on page Appx-A-63.
- *A.6 Q-Channel interface for power signals* on page Appx-A-64.
- *A.7 P-Channel master interface signals* on page Appx-A-65.
- *A.8 System interface signals* on page Appx-A-66.

A.1 AHB-Lite slave interface signals

The AHB-Lite slave interface processes several input and output AHB-Lite signals.

The following table shows the signals that are used by the AHB-Lite slave interface.

Table A-1 AHB-Lite slave interface signals

| Signal name | Direction | Source or Destination | Description |
|--------------------|-----------|-----------------------|---|
| hsel | Input | Decoder | Slave select signal. Indicates that the current transfer is intended for the Flash controller. |
| haddr[21:0] | Input | AHB Master | Address bus. 22 bits wide. Allows a 4MB address space to be split into a 2MB main and a 2MB extended memory area. MSB haddr[21] specifies the selection between the two areas: 0 = Main area. 1 = Extended area. |
| htrans[1:0] | Input | AHB Master | Indicates the current transfer type: 0b00 = IDLE. 0b01 = BUSY. 0b10 = NONSEQUENTIAL. 0b11 = SEQUENTIAL. |
| hwrite | Input | AHB Master | Read only. Only reads are allowed. Can be tied to 0b0 externally. Writes generate an error. |
| hsize[2:0] | Input | AHB Master | Indicates the size of access: 0b000 = Byte. 0b001 = Halfword. 0b010 = Word. 0b011 = Doubleword. 0b100 = 4-word line. 0b101 = 8-word line. 0b110 = 512-bit width. 0b111 = 1024-bit width. Only 0b100 is allowed. Other access sizes generate an error. |

Table A-1 AHB-Lite slave interface signals (continued)

| Signal name | Direction | Source or Destination | Description |
|----------------------|-----------|---------------------------|---|
| hburst[2:0] | Input | AHB Master | Indicates if the transfer is part of the burst: 0b000 = Single. 0b001 = Incr. 0b010 = Wrap4. 0b011 = Incr4. 0b100 = Wrap8. 0b101 = Incr8. 0b110 = Wrap16. 0b111 = Incr16. |
| hmastlock | Input | AHB Master | Indicates that the transaction on the bus must be atomic. Used for blocking arbitration over the GFB. |
| hready | Input | Multiplexer or AHB Master | When HIGH, indicates that a bus transfer has completed. To extend a transfer, drive this signal LOW. |
| hreadyout | Output | Multiplexer or AHB Master | When HIGH, indicates that a bus transfer has completed. To extend a transfer, drive this signal LOW. Isolation and reset value = 0b1 ^a . |
| hresp | Output | Multiplexer or AHB Master | Transfer response status: 0b0 = OKAY. 0b1 = ERROR. Isolation and reset value = 0b0 ^a . |
| hrdata[127:0] | Output | Multiplexer or AHB Master | Read data. Isolation and reset value = 0x0 ^a . |

Note

AHB-Lite signals that are not shown in the table are not used in Vultan.

^a This signal must be set to this value at reset or when Vultan powers down.

A.2 APB slave interface signals

The APB slave interface processes several input and output APB signals.

The following table shows the signals that are used by the APB slave interface.

Table A-2 APB slave interface signals

| Signal name | Direction | Source or Destination ^b | Description |
|-----------------------|-----------|------------------------------------|---|
| psel_s | Input | APB Bridge | Slave select signal. Indicates that the current transfer is intended for the Flash controller. |
| penable_s | Input | APB Bridge | Strobe to time all accesses. Indicates the start of the second cycle of an APB transfer. |
| paddr_s[12:0] | Input | APB Bridge | Address bus. paddr_s[12] selects either the internal or an external register bank: 0 = Internal registers. 1 = External registers. |
| pstrb_s[3:0] | Input | APB Bridge | Write strobe port. Each bit refers to a byte in the pwdata_s signal: [3] pwdata_s[31:24] . [2] pwdata_s[23:16] . [1] pwdata_s[15:8] . [0] pwdata_s[7:0] . |
| pwrite_s | Input | APB Bridge | APB transfer direction. Write only. |
| pwdata_s[31:0] | Input | APB Bridge | 32-bit Write data bus. |
| prdata_s[31:0] | Output | APB Bridge | 32-bit Read data bus. Isolation and reset value = 0x00 ^c . |
| pready_s | Output | APB Bridge | Driven LOW when extra wait states are required to complete access to the external registers. Isolation and reset value = 0b0 ^c . |
| pslverr_s | Output | APB Bridge | Driven HIGH when an error response is received from an access to the external registers. Isolation and reset value = 0b0 ^c . |

————— **Note** —————

APB signals that are not shown in the table are not used in Vultan.

^b The APB Bridge is not supplied with Vultan.

^c This signal must be set to this value at reset or when Vultan powers down.

A.3 APB master interface signals

The APB master interface processes several input and output APB signals.

The following table shows the signals that are used by the APB master interface.

Table A-3 APB master interface signals

| Signal name | Direction | Source or Destination | Description |
|-----------------------|-----------|-----------------------|---|
| psel_m | Output | Process-specific part | Slave select signal. Indicates that the current transfer is intended for the Flash controller. |
| penable_m | Output | Process-specific part | Strobe to time all accesses. Indicates the start of the second cycle of an APB transfer. |
| paddr_m[11:0] | Output | Process-specific part | Address bus. |
| pstrb_m[3:0] | Output | Process-specific part | Write strobe port. Each bit refers to a byte in the pwdata_m signal: [3] pwdata_m[31:24] . [2] pwdata_m[23:16] . [1] pwdata_m[15:8] . [0] pwdata_m[7:0] . |
| pwrite_m | Output | Process-specific part | APB transfer direction. Write only. |
| pwdata_m[31:0] | Output | Process-specific part | 32-bit Write data bus. |
| prdata_m[31:0] | Input | Process-specific part | 32-bit Read data bus. |
| pready_m | Input | Process-specific part | Driven LOW when extra wait states are required to complete access to the external registers. |
| pslverr_m | Input | Process-specific part | Driven HIGH when an error response is received from an access to the external registers. |

————— **Note** —————

APB signals that are not shown in the table are not used in Vultan.

A.4 Generic Flash Bus signals

The GFB interface processes several input and output signals.

The following table shows the GFB signals.

Table A-4 Generic Flash Bus signals

| Signal name | Direction | Source or Destination | Description |
|----------------------|-----------|-----------------------|--|
| faddr[21:0] | Output | Process-specific part | <p>Address bus.</p> <p>Address width is fixed at 22 bits to allow accesses to a 4MB embedded Flash that can be divided into a 2MB main memory and extended memory.</p> <p>faddr[21] selects between main and extended regions:</p> <p>0 = Main area.</p> <p>1 = Extended area.</p> <p>faddr[3:2] selects the location of the 32-bit write data within the 128-bit interface:</p> <p>00 = [31:0].</p> <p>01 = [63:32].</p> <p>10 = [95:64].</p> <p>11 = [127:96].</p> <p>faddr[1:0] is not used because the minimum data width is 32 bits.</p> |
| fcmd[2:0] | Output | Process-specific part | <p>Command bus:</p> <p>000 = IDLE.</p> <p>001 = READ.</p> <p>010 = WRITE.</p> <p>011 = ROW WRITE.</p> <p>100 = ERASE.</p> <p>101 = Reserved.</p> <p>110 = Reserved.</p> <p>111 = MASS ERASE.</p> |
| fabort | Output | Process-specific part | <p>Abort indication.</p> <p>When HIGH, the master requests to abort the command that is running.</p> |
| fwdata[31:0] | Output | Process-specific part | 32-bit Write data bus. |
| frdata[127:0] | Input | Process-specific part | 128-bit Read data bus. |

Table A-4 Generic Flash Bus signals (continued)

| Signal name | Direction | Source or Destination | Description |
|---------------|-----------|-----------------------|--|
| fready | Input | Process-specific part | Command ready indication. Driven LOW if the process-specific part requires wait states to complete the access. Driven HIGH when the process-specific part is ready with the previous access and is able to accept a new command. |
| fresp | Input | Process-specific part | Flash error indication for the previously accepted command. Driven HIGH for two cycles when an error is indicated for the command that is running. |

————— **Note** —————

GFB signals that are not shown in the table are not used in Vultan.

Vultan and the GFB slave logic inside the process-specific part are expected to be in the same power domain. Therefore, the GFB does not need isolation values.

A.5 Q-Channel interface for clock signals

The Q-Channel interface for clock processes all Q-Channel clock control signals.

The following table shows the signals that are used by the Q-Channel interface for clock.

Table A-5 Q-Channel interface for clock signals

| Signal name | Direction | Source or Destination | Description |
|---------------------|-----------|-----------------------|--|
| qreqn_clk | Input | Clock controller | Quiescence request. Active-LOW. Synchronized with double-flop synchronizer. |
| qacceptn_clk | Output | Clock controller | Accept quiescence request. Active-LOW. Isolation and reset value = 0b0 ^d . |
| qdeny_clk | Output | Clock controller | Deny quiescence request. Active-HIGH. Isolation and reset value = 0b0 ^d . |
| qactive_clk | Output | Clock controller | Activity indication. Active-HIGH. Isolation and reset value = 0b0 ^d . |

Driving signals

The system clock controller might be in a different clock or power domain to Vultan. Therefore, the Q-Channel interface for clock is considered to be fully asynchronous. This means that all input pins are captured by double-flop synchronizers, and all output pins are driven from registers.

^d This signal must be set to this value at reset or when Vultan powers down.

A.6 Q-Channel interface for power signals

The Q-Channel interface for power processes all Q-Channel power control signals.

The following table shows the signals that are used by the Q-Channel interface for power.

Table A-6 Q-Channel interface for power signals

| Signal name | Direction | Source or Destination | Description |
|---------------------|-----------|-----------------------|--|
| qreqn_pwr | Input | Power Policy Unit | Quiescence request. Active-LOW. Synchronized with double-flop synchronizer. |
| qacceptn_pwr | Output | Power Policy Unit | Accept quiescence request. Active-LOW. Isolation and reset value = 0b0 ^e . |
| qdeny_pwr | Output | Power Policy Unit | Deny quiescence request. Active-HIGH. Isolation and reset value = 0b0 ^e . |
| qactive_pwr | Output | Power Policy Unit | Activity indication. Active-HIGH. Isolation and reset value = 0b0 ^e . |

Driving signals

The system Power Policy Unit might be in a different clock or power domain to Vultan. Therefore, the Q-Channel interface for power is considered to be fully asynchronous. This means that all input pins are captured by double-flop synchronizers, and all output pins are driven from registers.

^e This signal must be set to this value at reset or when Vultan powers down.

A.7 P-Channel master interface signals

The P-Channel master interface processes all P-Channel signals.

The following table shows the signals that are used by the P-Channel master interface.

Table A-7 P-Channel master interface signals

| Signal name | Direction | Source or Destination | Description |
|----------------|-----------|-----------------------|---|
| preq | Output | Process-specific part | Power state change request. Active HIGH. |
| pstate | Output | Process-specific part | Requested power state value: 0 = All powerdown (default). 1 = All powerup. |
| paccept | Input | Process-specific part | Accept indication. Active HIGH. Synchronized with double-flop synchronizer. |
| pdeny | Input | Process-specific part | Deny indication. Active HIGH. Synchronized with double-flop synchronizer. |
| pactive | Input | Process-specific part | Activity indication for each power state. Active HIGH. Synchronized with double-flop synchronizer. |

Driving signals

Vultan drives all output signals from registers, and double-flop synchronizers capture all input signals. This enables the P-Channel slave and Vultan to be fully synchronous, and also enables the P-Channel slave to be placed in a different power domain, if required.

A.8 System interface signals

The system interface processes several input and output signals.

The following table shows the system interface signals.

Table A-8 System interface signals

| Signal name | Direction | Source or Destination | Description |
|----------------------|-----------|-----------------------|--|
| clk | Input | Clock generator | Core clock for all Vultan interfaces. |
| resetsn | Input | System controller | Reset input pin. Active-LOW. Asserts asynchronously, releases synchronously with clk . |
| irq | Output | Interrupt controller | Interrupt request. Active-HIGH. |
| flash_pwr_rdy | Output | Process-specific part | Indicates that power to the Flash macro is stable. Active-HIGH. The process-specific part monitors this signal for a rising edge after reset. Transactions towards the Flash macro are not initiated before the rising edge occurs. Additional transitions are ignored. |

Appendix B

Revisions

Read this for a description of changes between released issues of this book.

It contains the following section:

- [B.1 Revisions on page Appx-B-68.](#)

B.1 Revisions

This appendix describes changes between released issues of this book.

Table B-1 Issue 0000-00

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

Table B-2 Differences between issue 0000-00 and issue 0000-01

| Change | Location | Affects |
|---|---|---------|
| Updated text | 1.1 About Vultan on page 1-12 | r0p0 |
| Moved start and end points of flash_pwr_rdy signal | Figure 2-1 Vultan internal structure on page 2-21 | r0p0 |
| Clarified description of data width | 2.2.1 AHB-Lite slave interface on page 2-22 | r0p0 |
| Updated description of subsection <i>Error response</i> | 2.2.2 APB slave interface on page 2-23 | r0p0 |
| Updated description of subsection <i>Address width</i> | 2.2.4 Generic Flash Bus on page 2-24 | r0p0 |
| Updated section | 2.3 Clocking on page 2-28 | r0p0 |
| Updated section | 2.4 Resets on page 2-29 | r0p0 |
| Updated descriptions for Command Success and Command Fail interrupts | 2.5 Interrupt sources on page 2-30 | r0p0 |
| Updated description in subsection <i>Arbitration scheme</i> | 2.6 Generic Flash Bus Arbiter on page 2-31 | r0p0 |
| Added notes to descriptions for bits CMD_FAIL_IRQ_STS_SET and CMD_SUCCESS_IRQ_STS_SET | 3.4.3 IRQ_STATUS_SET on page 3-43 | r0p0 |
| Added text to Description column for signal faddr[21:0] | Table A-4 Generic Flash Bus signals on page Appx-A-61 | r0p0 |