# Arm® CoreLink™ GFC-100 Generic Flash Controller

**Revision: r0p0**

## Technical Reference Manual

arm


101059_0000_02_en

# Arm® CoreLink™ GFC-100 Generic Flash Controller

## Technical Reference Manual



### 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 |
| 0000-02 | 05 October 2018 | Non-Confidential | Third release for r0p0 |

### Non-Confidential Proprietary Notice



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® CoreLink™ GFC-100 Generic Flash Controller Technical Reference Manual

# Preface

This preface introduces the *Arm® CoreLink™ GFC-100 Generic Flash Controller Technical Reference Manual*.

It contains the following:

# About this book

This book is for the GFC-100 Generic Flash Controller.

## Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

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

p*n* 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 GFC-100 Generic Flash Controller.

## Using this book

This book is organized into the following chapters:

***Chapter 1 Introduction***
Read this for an introduction to the GFC-100 Generic Flash Controller and its features.

***Chapter 2 Functional Description***
Read this for a description of the GFC-100 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.

`mono`space
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.

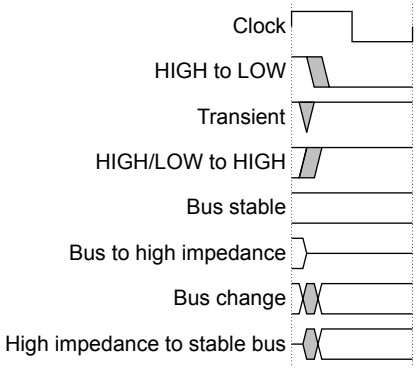


**Figure 1 Key to timing diagram conventions**

### 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

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

**Arm publications**

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

See *Infocenter*, for access to Arm documentation.

The following confidential books are only available to licensees or require registration with Arm:

- *Arm® CoreLink™ GFC-100 Generic Flash Controller Configuration and Integration Manual* (101060).
- *Arm® AMBA® Generic Flash Bus Protocol Specification* (IHI 0083).

**Other publications**

- JEDEC, *Standard Manufacturer's Identification Code*, JEP106.

# 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 CoreLink GFC-100 Generic Flash Controller Technical Reference Manual*.
- The number 101059_0000_02_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 GFC-100 Generic Flash Controller and its features.

It contains the following sections:

## 1.1 About GFC-100

The GFC-100 Generic Flash Controller comprises the generic part of a Flash controller in a *System-on-Chip* (SoC). GFC-100 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.

GFC-100 provides the functions that relate only to services for the system side of the Flash controller. GFC-100 cannot communicate directly with the eFlash macro. Therefore, GFC-100 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 GFC-100.

The following figure shows how GFC-100 is used in a Flash controller implementation.

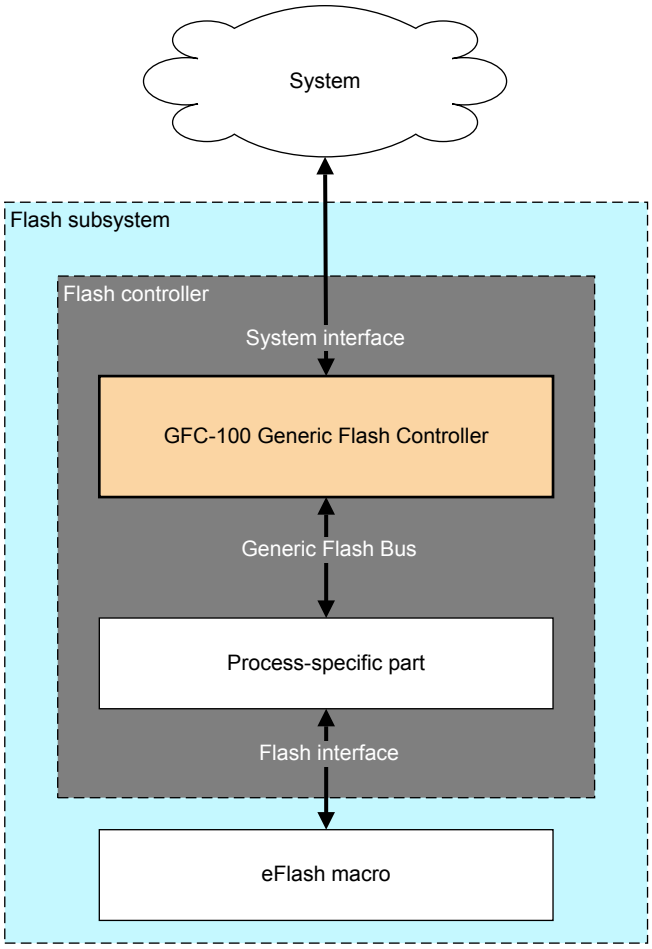


**Figure 1-1 GFC-100 in a Flash controller implementation**

## 1.2 Compliance

GFC-100 interfaces are compliant with Arm specifications and protocols.

GFC-100 is compliant with:

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

## 1.3 Features

GFC-100 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 GFC-100 and the eFlash 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 controller interface:

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

*Generic Flash Bus* (GFB):

- Enables GFC-100 accesses to embedded Flash.
- Simple command-based protocol.
- Synchronous with the AHB clock.
- Simplifies communication between GFC-100 and the attached process-specific part.

## 1.4 Test features

GFC-100 provides components that comprise a Flash model, that simulates the behavior of the process-specific part and the attached embedded Flash.

The GFC-100 deliverables include an *Out-of-Box* (OoB) 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 GFC-100 documentation is as follows:

**Technical Reference Manual**

The TRM describes the functionality and the effects of functional options on the behavior of GFC-100. 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 GFC-100, contact:

- The implementer to determine:
  - The build configuration of the implementation.
  - What integration, if any, was performed before implementing GFC-100.
- 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 RTL with the build configuration options.
- How to integrate GFC-100 into an SoC.
- How to implement GFC-100 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 GFC-100 Generic Flash Controller functions.

It contains the following sections:

## 2.1 Internal structure

GFC-100 comprises several submodules.

The following figure shows the internal high-level structure of an example system that integrates GFC-100 with a process-specific part and embedded Flash.

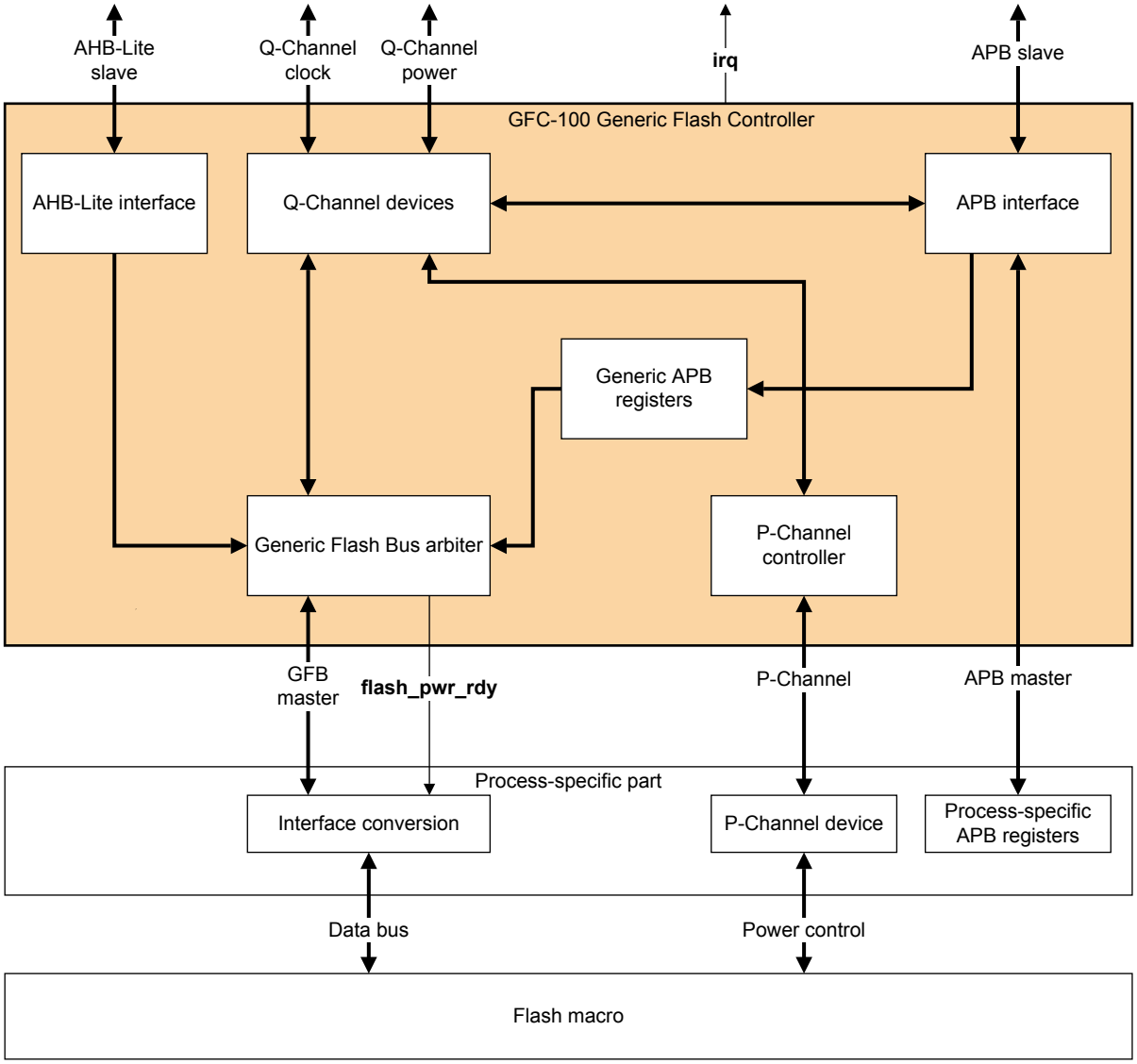


**Figure 2-1 GFC-100 internal structure**

## 2.2 Interfaces

GFC-100 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

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 ignored and generate an error response. Write accesses of any size are ignored 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 *Arm® AMBA® Generic Flash Bus Protocol Specification* 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, GFC-100 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 3-32 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. GFC-100 supports all burst types that are specified in the *Arm® AMBA® 3 AHB-Lite Protocol Specification v1.0*. Burst transfers ensure that read accesses from the memory have priority above any other commands. Burst transfers 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 GFC-100 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 the 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 the AHB master initiates.
- An access is set to a data width that is not 128 bits.
- A GFB transfer generates an error response.

***Related reference***

*A.1 AHB-Lite slave interface signals* on page Appx-A-55

### 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 GFC-100 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 2 × 4KB address spaces. One 4KB region is for internal registers, and the other 4KB region is for 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 3-33 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, because the process-specific part might support byte accesses.

#### Delayed response

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

***Related reference***

*A.2 APB slave interface signals* on page Appx-A-57

### 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 access external registers at byte level. Each strobe signal corresponds to a byte within the 32-bit write data.

#### Delayed response

GFC-100 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 an access is implementation-dependent.

#### Error response

Error responses that the GFC-100 APB master interface receives are forwarded to the GFC-100 APB slave interface. The conditions that determine when the attached APB slave responds with errors is implementation-dependent.

***Related reference***

*A.3 APB master interface signals* on page Appx-A-58

### 2.2.4 Generic Flash Bus

The Generic Flash Bus enables GFC-100 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 GFC-100 is the master.

#### Command bus

GFC-100 supports all commands that are specified in the *Arm® AMBA® Generic Flash Bus Protocol Specification*.

#### 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, that 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

GFC-100 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 `READ` commands. However, `WRITE` and `ERASE` commands can take many more cycles to execute.

#### Error response

Failed transactions can generate a 2-cycle error response from the process-specific part.

#### Aborting commands

The **fabort** signal can abort commands that the GFB master initiates. The signal is driven from the APB registers. The method that the process-specific part uses to support the abort function depends on its implementation.

**Note**

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

***Related reference***

*A.4 Generic Flash Bus signals* on page Appx-A-59

### 2.2.5 Q-Channel interface for clock

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

GFC-100 can accept or deny a request from the clock controller to turn off all operations that are using the clock.

#### Activity indication

GFC-100 uses the **qactive_clk** signal to indicate it has an activity that is using the clock. When GFC-100 asserts **qactive_clk**, the clock controller keeps the clock enabled.

#### Accepting a request

When GFC-100 has no activity, it accepts the quiescence request for the clock, and enters the Q_STOPPED state.

#### Denying a request

When the clock controller requests the clock to be disabled, if the activity indication is HIGH, GFC-100 denies the request.

***Related reference***

*A.5 Q-Channel interface for clock signals* on page Appx-A-61

### 2.2.6 Q-Channel interface for power

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

GFC-100 can accept or deny a request from the *Power Policy Unit* (PPU) to turn off all operations that are using power.

#### Activity indication

GFC-100 uses the **qactive_pwr** signal to indicate that it has an activity that is using power. When GFC-100 asserts **qactive_pwr**, the PPU keeps the power enabled.

#### Accepting a request

When GFC-100 or the embedded Flash is inactive, GFC-100 accepts the quiescence request for power, and enters the Q_STOPPED state. GFC-100 must then drive the process-specific part to a powerdown state before accepting the Q-Channel request.

#### Denying a request

When the PPU requests power to be disabled, if the activity indication is HIGH, GFC-100 denies the request.

***Related reference***

*A.6 Q-Channel interface for power signals* on page Appx-A-62

### 2.2.7 P-Channel controller interface

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

GFC-100 can request changes to the power state when it accepts a powerdown request from the system-level Power Policy Unit. When GFC-100 leaves reset and is not in quiescent state, it enables power to embedded Flash automatically.

GFC-100 supports two power states, all powerup, and all powerdown.

#### State encoding

GFC-100 uses a single bit to encode the two Flash macro power states of either all powerdown (0), or all powerup (1).

#### Activity encoding

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

#### Responding to a request

GFC-100 initiates all activity, and does not expect the process-specific part to assert a request denial. However, GFC-100 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.

***Related reference***

*A.7 P-Channel controller interface signals* on page Appx-A-63

### 2.2.8 System interface

The GFC-100 system interface comprises several system I/O ports that are separate from its interfaces.

#### Interrupt request

GFC-100 generates an interrupt request that indicates to the system master that an important event in GFC-100 has occurred. The interrupt is active-HIGH. It is cleared by accessing registers in GFC-100, and acknowledging the reason for the interrupt, see *3.4.4 IRQ_STATUS_CLR* on page 3-42.

#### Flash Power Ready

When the P-Channel master sets the embedded Flash power to ON, GFC-100 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 requires 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.

***Related reference***

*A.8 System interface signals* on page Appx-A-64

## 2.3 Clocking

GFC-100 has a single clock domain, that is used for clocking all internal registers.

### Synchronous interfaces

The GFC-100 AHB-Lite slave, APB, and GFB interfaces are expected to run on the same clock.

### Asynchronous interfaces

The GFC-100 Q-Channel and P-Channel interfaces might be clocked asynchronously. Therefore, these interfaces have internal synchronizers.

## 2.4 Resets

GFC-100 has a single reset input. GFC-100 expects that this reset is synchronized externally.

The reset input signal must be synchronous with the clock. GFC-100 expects the reset to be asserted asynchronously and deasserted synchronously.

## 2.5 Interrupt sources

The internal generic APB registers in GFC-100 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-64.

## 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

The following information applies to the GFC-100 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 3-39.
- Access type is described as follows:

  **RW** Read and write.

  **RO** Read only.

  **WO** Write only.

## 3.2 Memory maps

GFC-100 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

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. The **hsize[2:0]** signal indicates access size. See *A.1 AHB-Lite slave interface signals* on page Appx-A-55.

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.

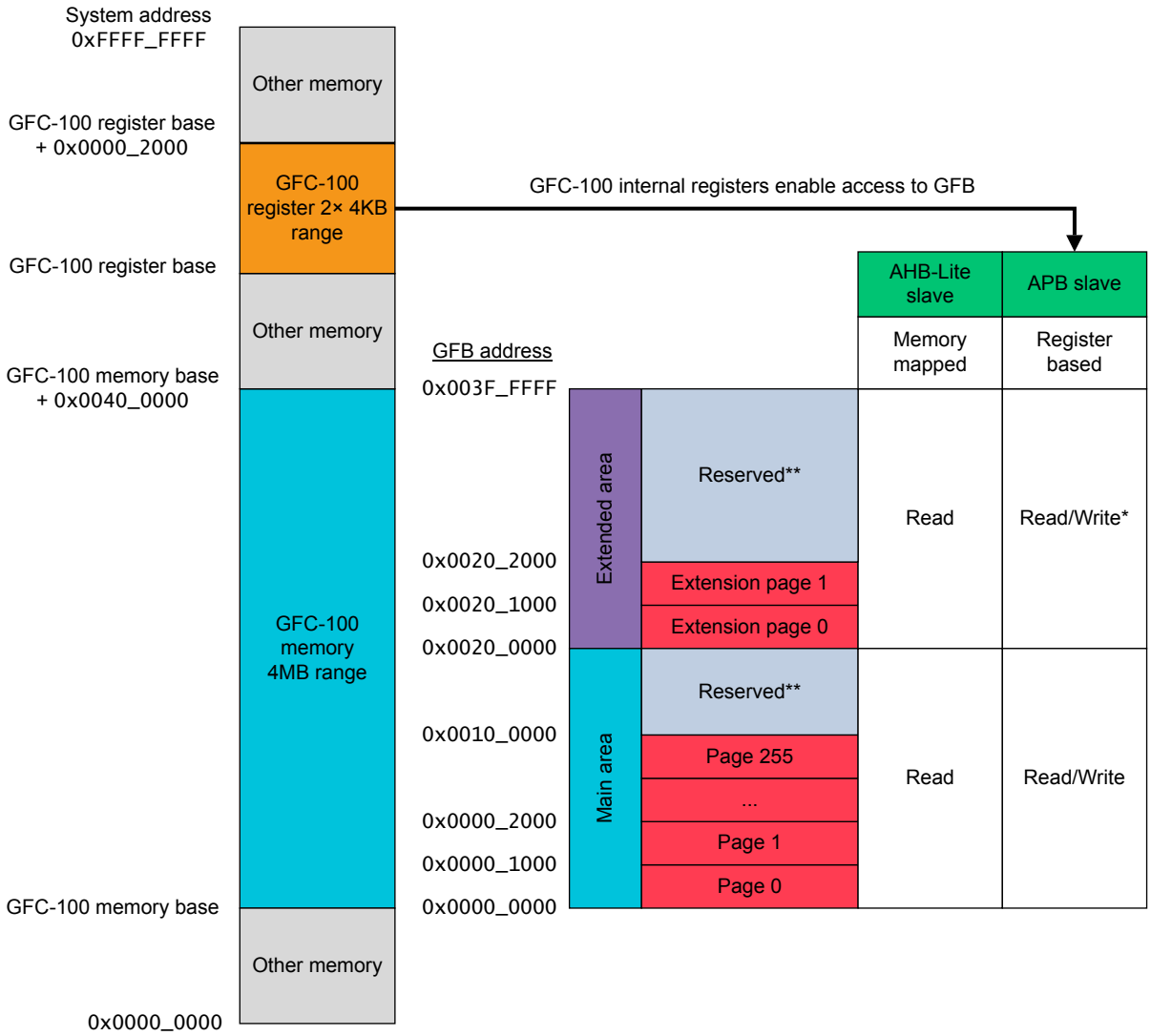


**Figure 3-1 AHB-Lite slave interface memory map**

In the figure:

- * Indicates that access rights depend on the properties of extension pages that the process-specific part handles. GFC-100 supports both read and write, or erase.
- ** Indicates that 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. GFC-100 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 × 4KB of register space.

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

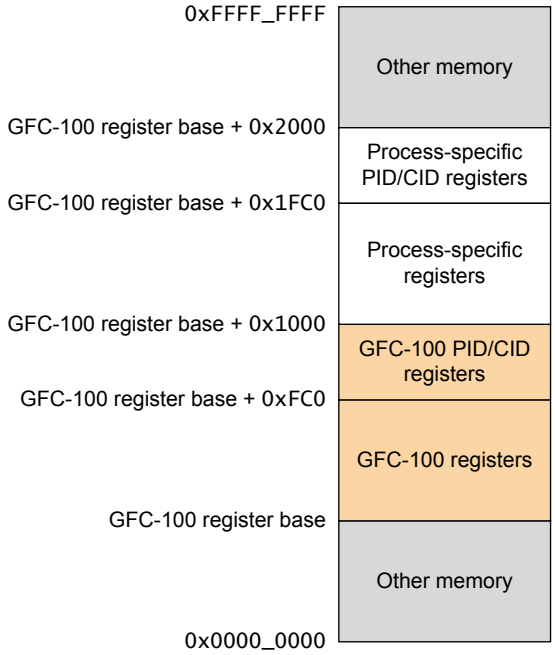


**Figure 3-2 APB slave interface memory map example**

The memory map example shows that GFC-100 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.

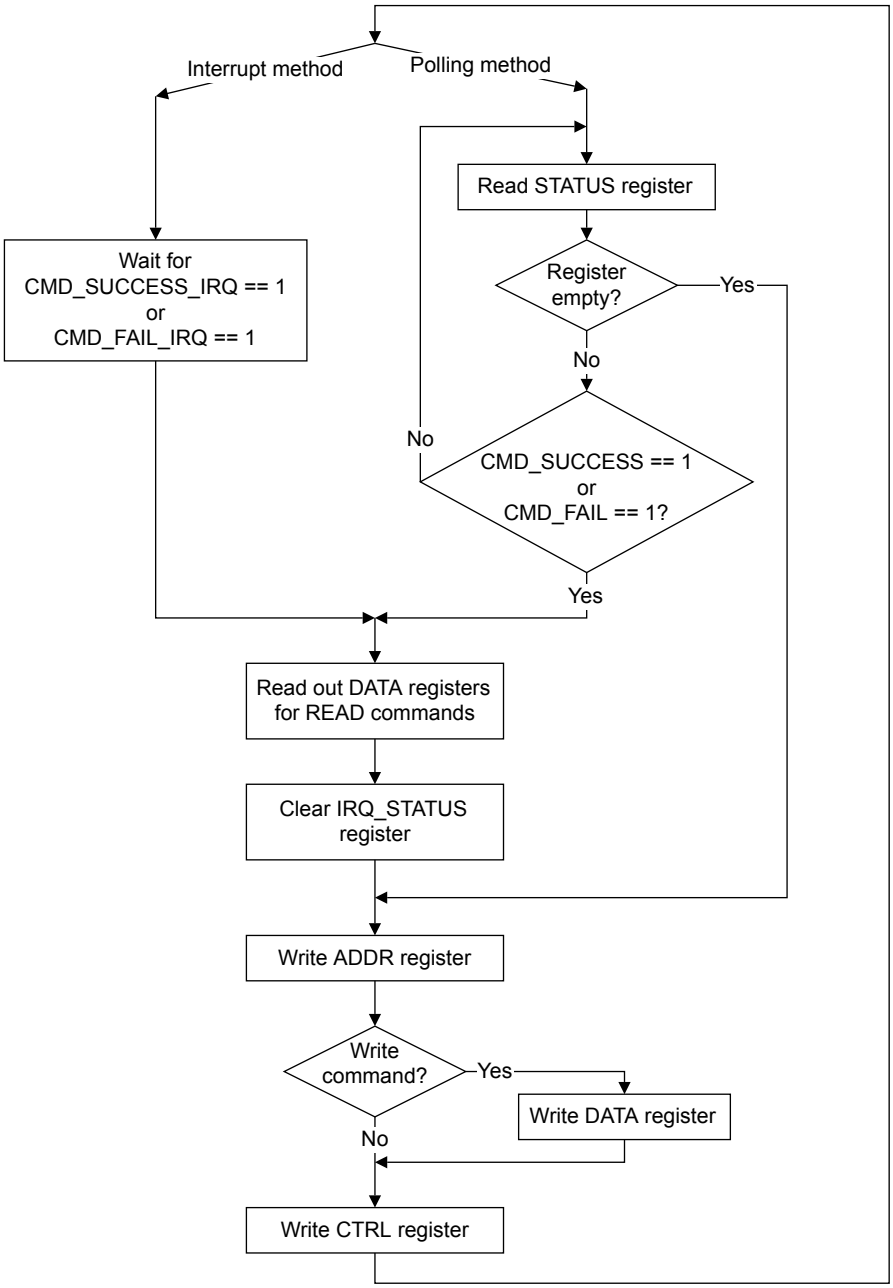


**Figure 3-3 APB slave access to Flash flowchart**

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 GFC-100 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.

***Related reference***

*3.4.5 IRQ_MASKED_STATUS* on page 3-43

*3.4.7 STATUS* on page 3-45

### 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 them.

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.

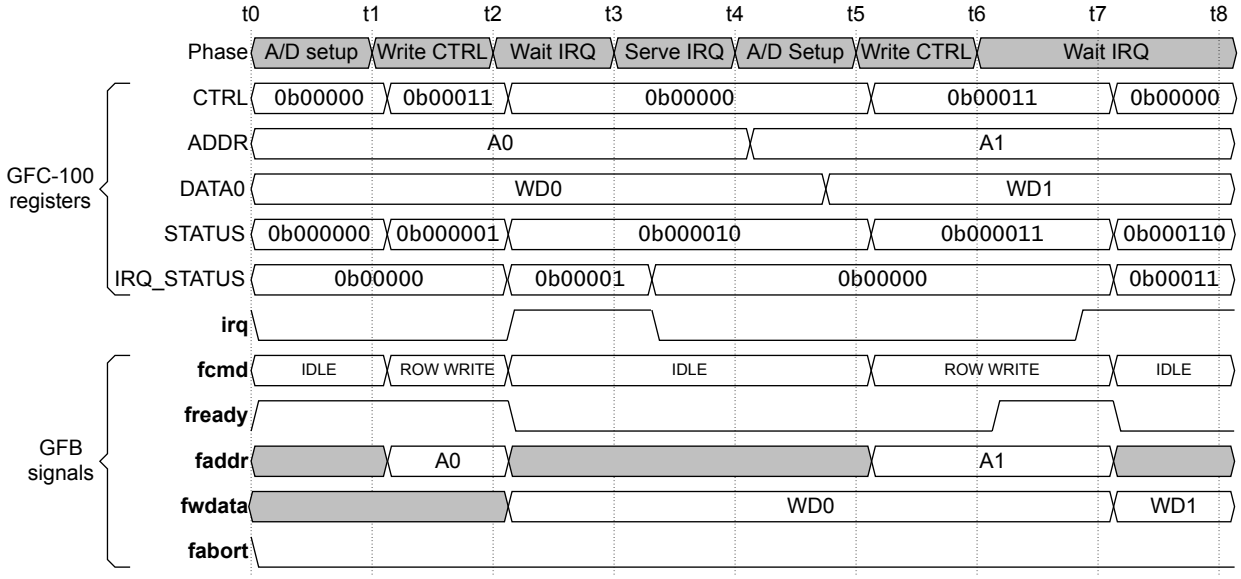


**Figure 3-4 ROW WRITE timing diagram**

#### ROW WRITE command sequence

The `ROW WRITE` command sequence is divided into different phases that are the actions that software requires 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. Software 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 t5 must finish before t7, 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 service 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 required, you must execute AHB reads instead.

The following figure shows the ROW WRITE preloading flowchart.

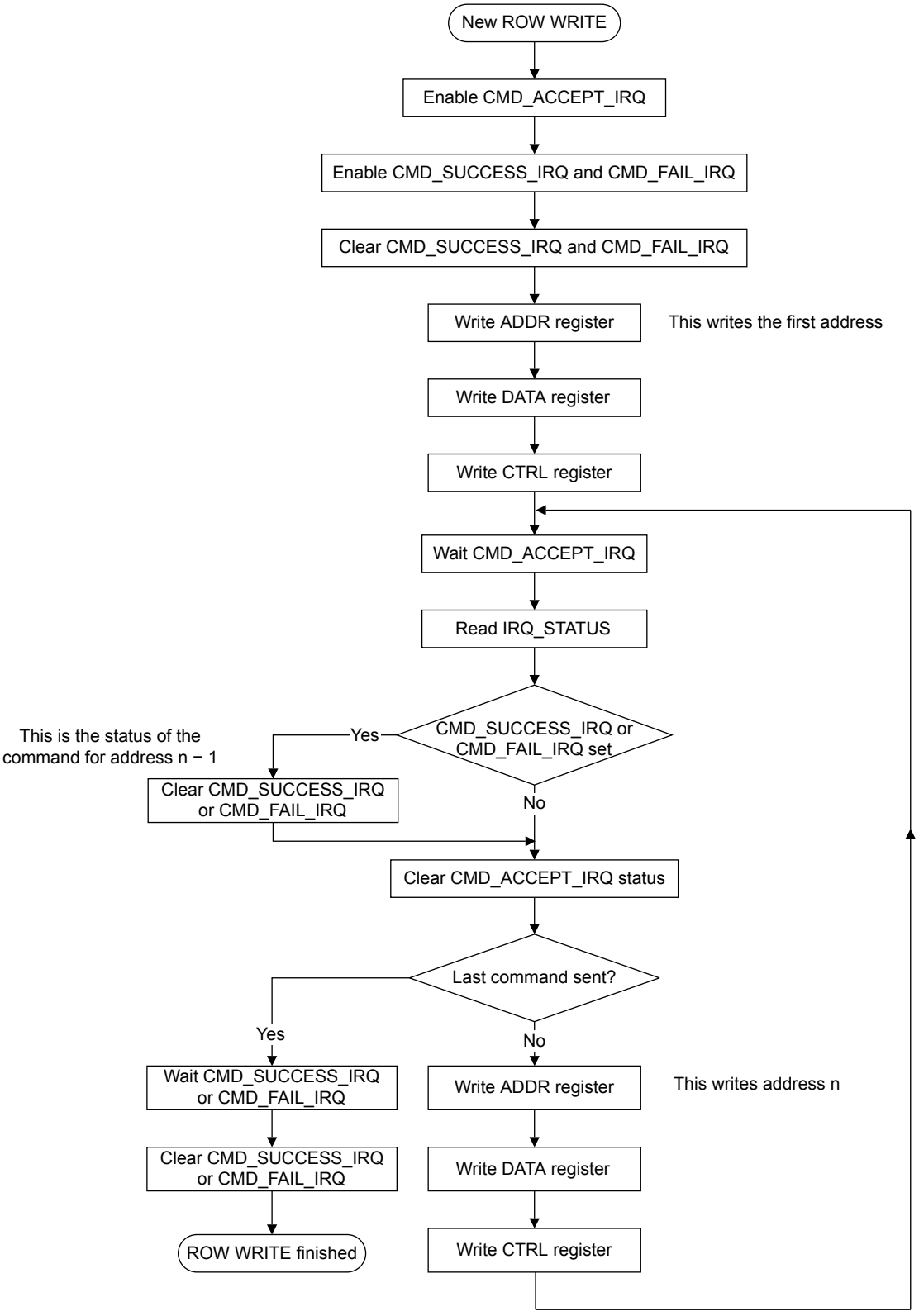


**Figure 3-5 ROW WRITE preloading flowchart**

## 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 3-40 |
| 0x004 | IRQ_ENABLE_CLR | RW | 0x0 | 32 | *3.4.2 IRQ_ENABLE_CLR* on page 3-41 |
| 0x008 | IRQ_STATUS_SET | RW | 0x0 | 32 | *3.4.3 IRQ_STATUS_SET* on page 3-41 |
| 0x00C | IRQ_STATUS_CLR | RW | 0x0 | 32 | *3.4.4 IRQ_STATUS_CLR* on page 3-42 |
| 0x010 | IRQ_MASKED_STATUS | RO | 0x0 | 32 | *3.4.5 IRQ_MASKED_STATUS* on page 3-43 |
| 0x014 | CTRL | RW | 0x0 | 32 | *3.4.6 CTRL* on page 3-44 |
| 0x018 | STATUS | RO | 0x0 | 32 | *3.4.7 STATUS* on page 3-45 |
| 0x01C | ADDR | RW | 0x0 | 32 | *3.4.8 ADDR* on page 3-46 |
| 0x020 | DATA0 | RW | 0x0 | 32 | *3.4.9 DATA0* on page 3-47 |
| 0x024 | DATA1 | RO | 0x0 | 32 | *3.4.10 DATA1* on page 3-47 |
| 0x028 | DATA2 | RO | 0x0 | 32 | *3.4.11 DATA2* on page 3-48 |
| 0x02C | DATA3 | RO | 0x0 | 32 | *3.4.12 DATA3* on page 3-48 |
| 0xFD0 | PIDR4 | RO | 0x4 | 32 | *3.4.13 PIDR4* on page 3-49 |
| 0xFE0 | PIDR0 | RO | 0x32 | 32 | *3.4.14 PIDR0* on page 3-49 |
| 0xFE4 | PIDR1 | RO | 0xB8 | 32 | *3.4.15 PIDR1* on page 3-50 |
| 0xFE8 | PIDR2 | RO | 0xB | 32 | *3.4.16 PIDR2* on page 3-50 |
| 0xFEC | PIDR3 | RO | 0x0 | 32 | *3.4.17 PIDR3* on page 3-51 |
| 0xFF0 | CIDR0 | RO | 0xD | 32 | *3.4.18 CIDR0* on page 3-51 |
| 0xFF4 | CIDR1 | RO | 0xF0 | 32 | *3.4.19 CIDR1* on page 3-52 |
| 0xFF8 | CIDR2 | RO | 0x5 | 32 | *3.4.20 CIDR2* on page 3-52 |
| 0xFFC | CIDR3 | RO | 0xB1 | 32 | *3.4.21 CIDR3* on page 3-53 |

## 3.4 Register descriptions

This section describes the GFC-100 registers.

*3.3 Register summary* on page 3-39 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.

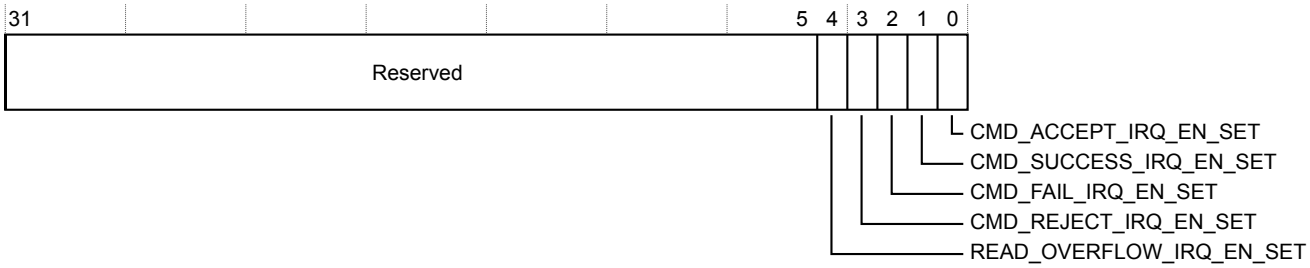


**Figure 3-6 IRQ_ENABLE_SET register bit assignments**

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.

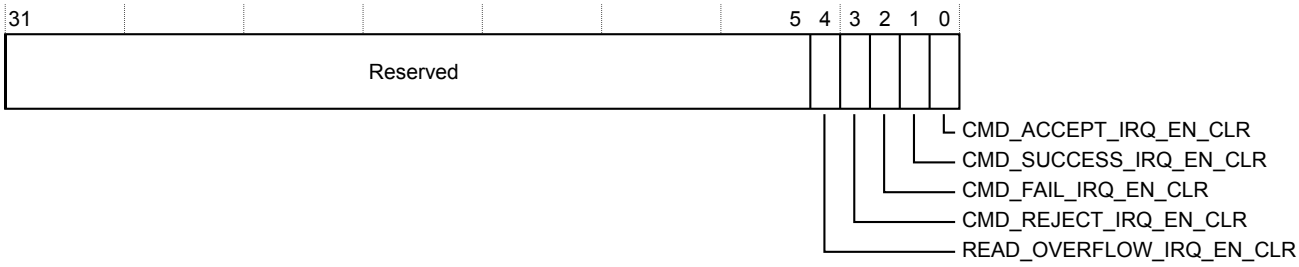


**Figure 3-7 IRQ_ENABLE_CLR register bit assignments**

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.

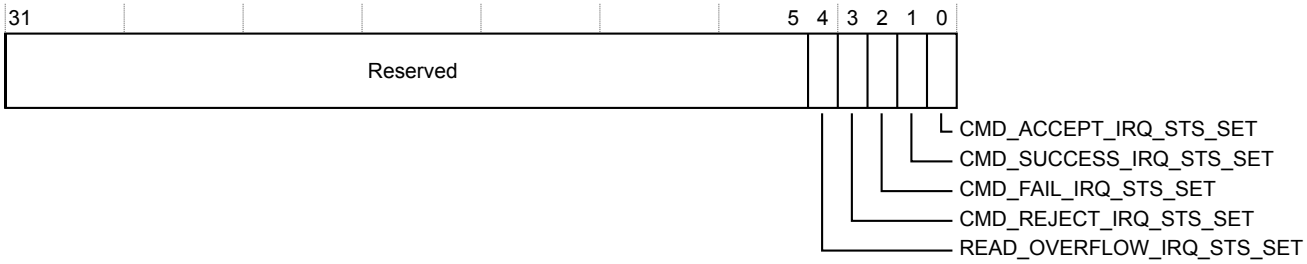


**Figure 3-8 IRQ_STATUS_SET register bit assignments**

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.

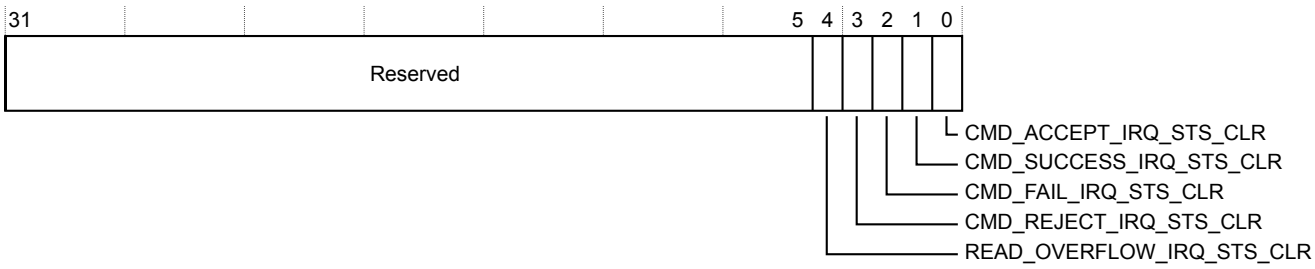


**Figure 3-9 IRQ_STATUS_CLR register bit assignments**

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.

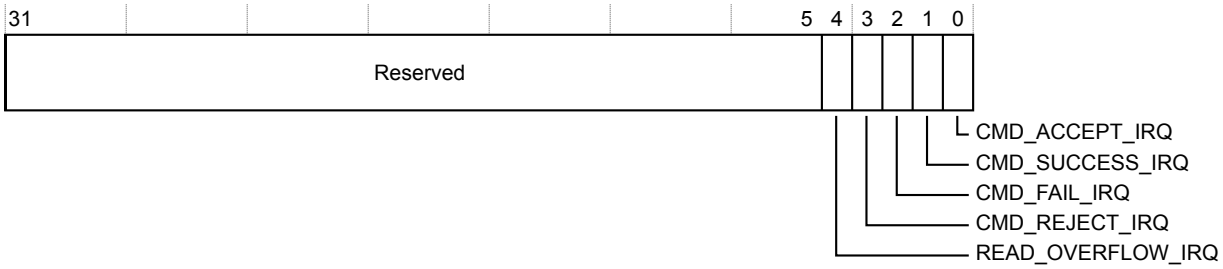


**Figure 3-10 IRQ_MASKED_STATUS register bit assignments**

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.

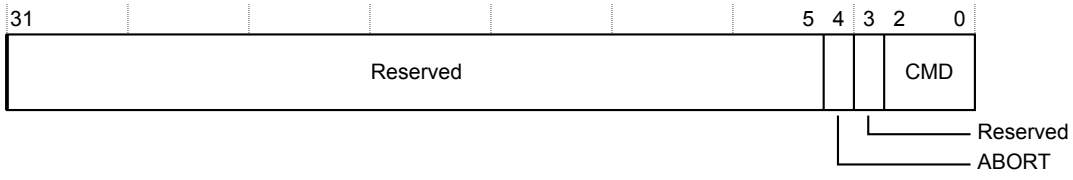


**Figure 3-11 CTRL register bit assignments**

The following list shows the register bit assignments.

**[4] ABORT**

Abort currently ongoing command that is 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 that is stored in the ADDR register and with the data that are 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:

`0b001`: `READ`

`0b010`: `WRITE`

`0b011`: `ROW WRITE`

`0b100`: `ERASE`

`0b111`: `MASS ERASE`

`0b000`: Reserved

`0b101`: Reserved

`0b110`: 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**
When this bit reads 1, it means that the other interface keeps the arbitration in locked state and no APB commands are serviced until the locked state is removed. Software can use this bit to detect why the APB request is being held up.

**[4] CMD_FINISH**
When this bit reads 1, it 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**
When this bit reads 1, it 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**
When this bit reads 1, it 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 embedded Flash arbitrates and accepts the command. 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** The value that is written to the ADDR field must be either:
- 32-bit aligned for write accesses.
- 128-bit aligned for read accesses.

**Configurations** There is only one configuration.

**Attributes**
**Offset** `0x01C`
**Type** Read/write
**Reset** `0x0`
**Width** 32

The following figure shows the bit assignments.

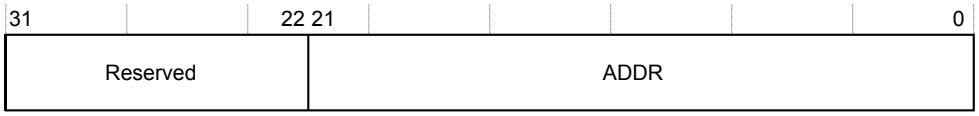


**Figure 3-13 ADDR register bit assignments**

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 ADDR[21] bit selects between memory ranges:
- 0: Main area.
- 1: Extended area.

When selecting the extended area for `MASS ERASE` command, 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.

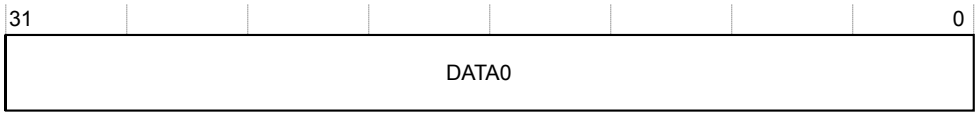


**Figure 3-14 DATA0 register bit assignments**

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.

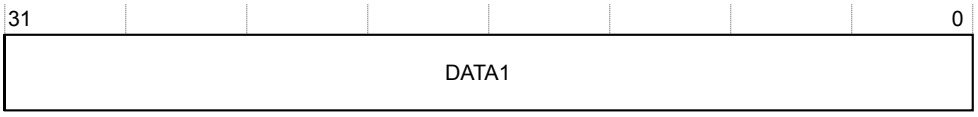


**Figure 3-15 DATA1 register bit assignments**

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.

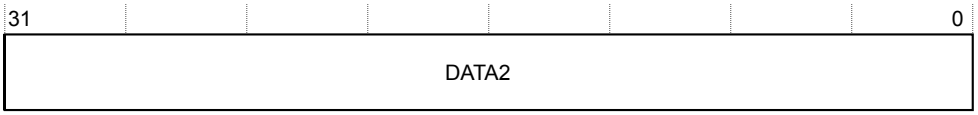


**Figure 3-16 DATA2 register bit assignments**

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.

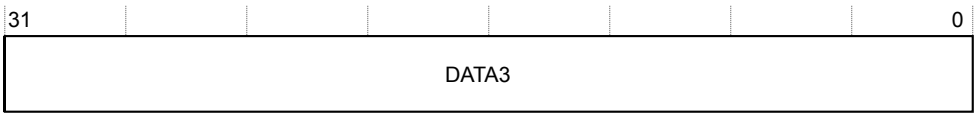


**Figure 3-17 DATA3 register bit assignments**

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.

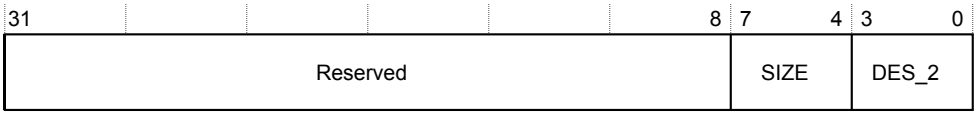


**Figure 3-18 PIDR4 register bit assignments**

The following list shows the register bit assignments.

**[7:4] SIZE** Indicates that the GFC-100 registers occupy a single 4KB page.

**[3:0] DES_2** Indicates how many Continuation Codes (`0x7F`) an Arm device requires. For identifying an Arm device or product, the *Standard Manufacturer's Identification Code* specifies a requirement of four Continuation Codes.

### 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.

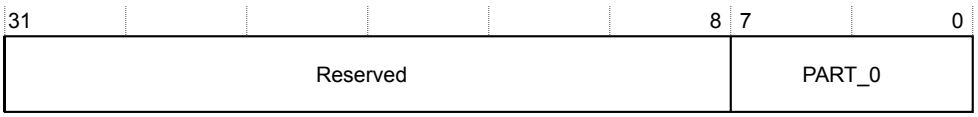


**Figure 3-19 PIDR0 register bit assignments**

The following list shows the register bit assignments.

**[7:0] PART_0** Part number, bits[7:0], for the GFC-100. See also PIDR1.PART_1. The GFC-100 part number is 832.

### 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.

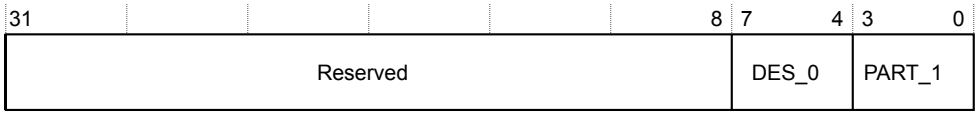


**Figure 3-20 PIDR1 register bit assignments**

The following list shows the register bit assignments.

**[7:4] DES_0** JEDEC JEP106 ID code [3:0]. See also PIDR2.DES_1 and the *Standard Manufacturer's Identification Code*.

**[3:0] PART_1** Part number, bits[11:8], for the GFC-100. See also PIDR0.PART_0. The GFC-100 part number is 832.

### 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** `0x0B`
- **Width** 32

The following figure shows the bit assignments.


**Figure 3-21 PIDR2 register bit assignments**

The following list shows the register bit assignments.

**[7:4] REVISION** Revision identifier for the GFC-100:
- `0x0` = r0p0.

**[3] JEDEC** Indicates the use of a JEDEC-assigned ID value.

**[2:0] DES_1** JEDEC JEP106 ID code [6:4]. See also PIDR1.DES_0[3:0] and the *Standard Manufacturer's Identification Code*.

### 3.4.17 PIDR3

Peripheral identification register 3. Returns byte[3] of the peripheral ID. The PIDR3 register is part of the set of peripheral identification registers.

The PIDR3 register characteristics are:

**Usage constraints** There are no usage constraints.

**Configurations** There is only one configuration.

**Attributes**
- **Offset** `0xFEC`
- **Type** Read-only
- **Reset** `0x00`
- **Width** 32

The following figure shows the bit assignments.

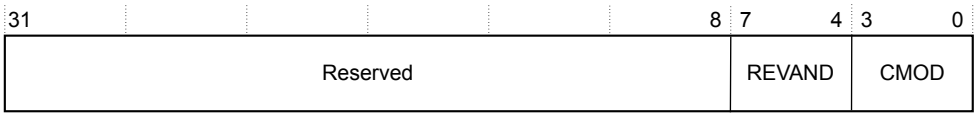


**Figure 3-22 PIDR3 register bit assignments**

The following list shows the register bit assignments.

**[7:4] REVAND** A nonzero value indicates that Arm has approved the application of a post-manufacture metal layer fix to the GFC-100 silicon.

**[3:0] CMOD** Customer modified.

### 3.4.18 CIDR0

Component identification register 0. Returns byte[0] of the component ID. The CIDR0 register is part of the set of component identification registers.

The CIDR0 register characteristics are:

**Usage constraints** There are no usage constraints.

**Configurations** There is only one configuration.

**Attributes**
- **Offset** `0xFF0`
- **Type** Read-only
- **Reset** `0x0D`
- **Width** 32

The following figure shows the bit assignments.

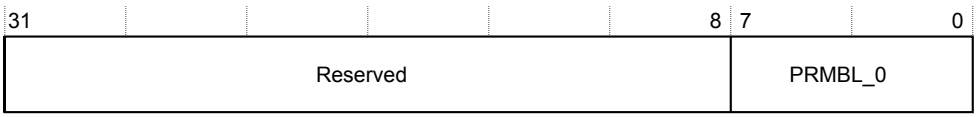


**Figure 3-23 CIDR0 register bit assignments**

The following list shows the register bit assignments.

**[7:0] PRMBL_0** Preamble 0.

### 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.

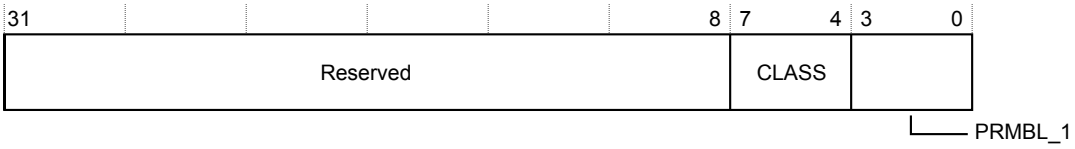


**Figure 3-24 CIDR1 register bit assignments**

The following list shows the register bit assignments.

**[7:4] CLASS** Component class.

**[3:0] PRMBL_1** Preamble 1.

### 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** `0x05`
- **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 2.

### 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.

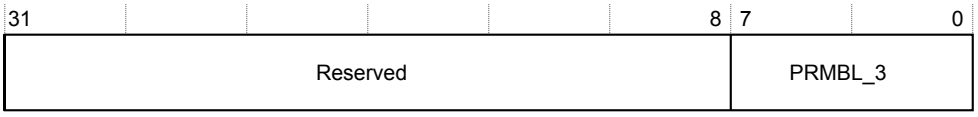


**Figure 3-26 CIDR3 register bit assignments**

The following list shows the register bit assignments.

**[7:0] PRMBL_3** Preamble 3.

# 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

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.<br>Indicates that the current transfer is intended for the Flash controller. |
| **haddr[21:0]** | Input | AHB master | Address bus.<br>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:<br>0 = Main area.<br>1 = Extended area. |
| **htrans[1:0]** | Input | AHB master | Indicates the current transfer type:<br>`0b00` = IDLE.<br>`0b01` = BUSY.<br>`0b10` = NONSEQUENTIAL.<br>`0b11` = SEQUENTIAL. |
| **hwrite** | Input | AHB master | Read-only.<br>Only reads are allowed. Can be tied LOW externally.<br>Writes generate an error. |
| **hsize[2:0]** | Input | AHB master | Indicates the size of access:<br>`0b000` = Byte.<br>`0b001` = Halfword.<br>`0b010` = Word.<br>`0b011` = Doubleword.<br>`0b100` = 4-word line.<br>`0b101` = 8-word line.<br>`0b110` = 512-bit width.<br>`0b111` = 1024-bit width.<br>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:<br>`0b000` = Single.<br>`0b001` = Incr.<br>`0b010` = Wrap4.<br>`0b011` = Incr4.<br>`0b100` = Wrap8.<br>`0b101` = Incr8.<br>`0b110` = Wrap16.<br>`0b111` = Incr16. |
| **hmastlock** | Input | AHB master | Indicates that the transaction on the bus must be atomic.<br>Used for blocking arbitration over the GFB. |
| **hready** | Input | Multiplexer or AHB master | When HIGH, indicates that a bus transfer has completed.<br>To extend a transfer, drive this signal LOW. |
| **hreadyout** | Output | Multiplexer or AHB master | When HIGH, indicates that a bus transfer has completed.<br>To extend a transfer, drive this signal LOW.<br>Isolation and reset value = `0b1`. |
| **hresp** | Output | Multiplexer or AHB master | Transfer response status:<br>`0b0` = OKAY.<br>`0b1` = ERROR.<br>Isolation and reset value = `0b0`. |
| **hrdata[127:0]** | Output | Multiplexer or AHB master | Read data.<br>Isolation and reset value = `0x0`. |

**Note**

AHB-Lite signals that are not shown in the table are not used in GFC-100.

***Related reference***

*2.2.1 AHB-Lite slave interface* on page 2-20

## 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[a] | Description |
|---|---|---|---|
| **psel_s** | Input | APB bridge | Slave select signal.<br>Indicates that the current transfer is intended for the Flash controller. |
| **penable_s** | Input | APB bridge | Strobe to time all accesses.<br>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:<br>0 = Internal registers.<br>1 = External registers. |
| **pstrb_s[3:0]** | Input | APB bridge | Write strobe port. Each bit refers to a byte in the **pwdata_s** signal:<br>[3] **pwdata_s[31:24]**.<br>[2] **pwdata_s[23:16]**.<br>[1] **pwdata_s[15:8]**.<br>[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.<br>Isolation and reset value = `0x00`. |
| **pready_s** | Output | APB bridge | Driven LOW when extra wait states are required to complete access to the external registers.<br>Isolation and reset value = `0b0`. |
| **pslverr_s** | Output | APB bridge | Driven HIGH when an error response is received from an access to the external registers.<br>Isolation and reset value = `0b0`. |

**Note**

APB signals that are not shown in the table are not used in GFC-100.

***Related reference***

*2.2.2 APB slave interface* on page 2-21

[a] The APB bridge is not supplied with GFC-100.

## 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:<br>[3] **pwdata_m[31:24]**.<br>[2] **pwdata_m[23:16]**.<br>[1] **pwdata_m[15:8]**.<br>[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 GFC-100.

***Related reference***

*2.2.3 APB master interface* on page 2-22

## 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 | Address bus.<br>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.<br>**faddr[21]** selects between main and extended regions:<br>0 = Main area.<br>1 = Extended area.<br>**faddr[3:2]** selects the location of the 32-bit write data within the 128-bit interface:<br>`0b00` = [31:0].<br>`0b01` = [63:32].<br>`0b10` = [95:64].<br>`0b11` = [127:96].<br>**faddr[1:0]** is not used because the minimum data width is 32 bits. |
| **fcmd[2:0]** | Output | Process-specific part | Command bus:<br>`0b000` = `IDLE`.<br>`0b001` = `READ`.<br>`0b010` = `WRITE`.<br>`0b011` = `ROW WRITE`.<br>`0b100` = `ERASE`.<br>`0b101` = Reserved.<br>`0b110` = Reserved.<br>`0b111` = `MASS ERASE`. |
| **fabort** | Output | Process-specific part | Abort indication.<br>When HIGH, the master requests to abort the command that is running. |
| **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.<br>Driven LOW if the process-specific part requires wait states to complete the access.<br>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.<br>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 GFC-100.
- The GFC-100 and the GFB slave logic inside the process-specific part are expected to be in the same power domain. Therefore, the GFB does not require isolation values.

***Related reference***

*2.2.4 Generic Flash Bus* on page 2-22

## 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.<br>Synchronized with double-flop synchronizer. |
| **qacceptn_clk** | Output | Clock controller | Accept quiescence request. Active-LOW.<br>Isolation and reset value = `0b0`. |
| **qdeny_clk** | Output | Clock controller | Deny quiescence request. Active-HIGH.<br>Isolation and reset value = `0b0`. |
| **qactive_clk** | Output | Clock controller | Activity indication. Active-HIGH. |

### Driving signals

The system clock controller might be in a different clock or power domain to GFC-100. Therefore, the Q-Channel interface for clock is considered to be fully asynchronous. This means that a double-flop synchronizer captures the input signal. Registers drive all output signals.

***Related reference***

*2.2.5 Q-Channel interface for clock* on page 2-23

## 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* (PPU). | Quiescence request. Active-LOW.<br>Synchronized with double-flop synchronizer. |
| **qacceptn_pwr** | Output | PPU | Accept quiescence request. Active-LOW.<br>Isolation and reset value = `0b0`. |
| **qdeny_pwr** | Output | PPU | Deny quiescence request. Active-HIGH.<br>Isolation and reset value = `0b0`. |
| **qactive_pwr** | Output | PPU | Activity indication. Active-HIGH. |

### Driving signals

The system PPU might be in a different clock or power domain to GFC-100. Therefore, the Q-Channel interface for power is considered to be fully asynchronous. This means that a double-flop synchronizer captures the input signal. Registers drive all output signals.

***Related reference***

*2.2.6 Q-Channel interface for power* on page 2-23

## A.7 P-Channel controller interface signals

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

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

**Table A-7 P-Channel controller 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:<br>0 = All powerdown (default).<br>1 = All powerup. |
| **paccept** | Input | Process-specific part | Accept indication. Active-HIGH.<br>Synchronized with double-flop synchronizer. |
| **pdeny** | Input | Process-specific part | Deny indication. Active-HIGH.<br>Synchronized with double-flop synchronizer. |
| **pactive** | Input | Process-specific part | Activity indication for each power state. Active-HIGH.<br>Synchronized with double-flop synchronizer. |

### Driving signals

GFC-100 drives all output signals from registers, and double-flop synchronizers capture all input signals. This enables the P-Channel device and GFC-100 to be fully synchronous. It also enables the P-Channel device to be placed in a different power domain, if necessary.

***Related reference***

*2.2.7 P-Channel controller interface* on page 2-24

## 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 GFC-100 interfaces. |
| **resetn** | Input | System controller | Active-LOW reset.<br>Assert asynchronously, release 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.<br>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. |

***Related reference***

*2.2.8 System interface* on page 2-24

# 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-66.

## 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 content. | *1.1 About GFC-100* on page 1-12. | All revisions. |
| Moved start and end points of **flash_pwr_rdy signal**. | *Figure 2-1 GFC-100 internal structure* on page 2-19. | |
| Clarified description of data width. | *2.2.1 AHB-Lite slave interface* on page 2-20. | |
| Updated the *Error response* description. | *2.2.2 APB slave interface* on page 2-21. | |
| Updated the *Address width* description. | *2.2.4 Generic Flash Bus* on page 2-22. | |
| Updated content. | *2.3 Clocking* on page 2-26. | |
| Updated content. | *2.4 Resets* on page 2-27. | |
| Updated descriptions for the Command Success and Command Fail interrupts. | *2.5 Interrupt sources* on page 2-28. | |
| Updated the *Arbitration scheme* description. | *2.6 Generic Flash Bus arbiter* on page 2-29. | |
| Added notes to the CMD_FAIL_IRQ_STS_SET and CMD_SUCCESS_IRQ_STS_SET bit descriptions. | *3.4.3 IRQ_STATUS_SET* on page 3-41. | |
| Updated the **faddr[21:0]** description. | *Table A-4 Generic Flash Bus signals* on page Appx-A-59. | |

**Table B-3 Differences between issue 0000-01 and issue 0000-02**

| Change | Location | Affects |
|---|---|---|
| Changed the product name from 'Vultan' to 'GFC-100'. | Throughout the document. | All revisions. |
| Added some usage constraints. | *3.4.8 ADDR* on page 3-46. | All revisions. |
| Updated the register bit descriptions. | • *3.4.14 PIDR0* on page 3-49.<br>• *3.4.15 PIDR1* on page 3-50.<br>• *3.4.16 PIDR2* on page 3-50.<br>• *3.4.17 PIDR3* on page 3-51.<br>• *3.4.13 PIDR4* on page 3-49. | All revisions. |
| Deleted an erroneous table footnote. | • *Table A-2 APB slave interface signals* on page Appx-A-57.<br>• *Table A-5 Q-Channel interface for clock signals* on page Appx-A-61.<br>• *Table A-6 Q-Channel interface for power signals* on page Appx-A-62. | All revisions. |